Creating Databases for Biological Information: An Introduction

The essence of bioinformatics is dealing with large quantities of information. Whether it be sequencing data, microarray data files, mass spectroscopy fingerprints, the catalog of strains arising from an insertional mutagenesis project, or even large numbers of PDF files, there inevitably comes a time when the information can simply no longer be managed with files and directories. This is where databases come into play.

A database manages information. It allows you to organize data, ensure completeness and integrity, transform it from one form to another, and search through the data efficiently to find the desired information.

Although strictly speaking, the term “database” applies to any collection of information, and can therefore be applied to a stack of index cards or a box of papyrus scrolls, it has come to mean a collection of data that is managed by a computerized database management system, or DBMS.

How do you know when you have reached the point of needing a real DBMS? Some of the physical signs are easily recognized:

1. The information you need is scattered among hundreds of files. You spend much of your time searching for the file you need using the operating system’s Find command or a command-line utility such as grep.
2. You find yourself creating complex, multilevel naming schemes in order to keep track of files and directories.
3. You’ve stored everything into an Excel spreadsheet, but the number of rows and columns have become so huge that the spreadsheet takes minutes to load.
4. You’ve started to lose data, sometimes by inadvertently overwriting or deleting a file, and sometimes just by losing track of it.

Even if none of the outward signs are manifest, you’ll know when the time for a DBMS has come when you start experiencing the symptoms of “information overload,” the anxiety associated with feeling overwhelmed with the size and complexity of your data.

DBMSs provide effective medicine against information overload. Despite their reputation for complexity, setting up a DBMS can be relatively painless and very educational, in part because the task will force you to look at your data in a novel way. Once installed, a DBMS will empower you to explore your data in ways that were previously impractical, and to undertake larger projects in the future.

DBMS CHARACTERISTICS

Once you have decided that you need a database, the first task becomes choosing a suitable database management system. DBMSs come in a bewildering variety of sizes and shapes, and are, like most software products, subject to the tides of fashion consciousness among bioinformaticists and computer scientists. We will walk through the major types of database system so that you get a feel for the range of offerings, and then offer some guidelines for choosing the one that is right for your needs.

Contributed by Lincoln Stein

Current Protocols in Bioinformatics (2003) 9.1.1-9.1.9

Flat File Databases

We begin with “flat file” databases, which consist of a set of one or more files containing information, and one or more programs that people use to look up, add, and delete information. A typical flat file database might be a list of researchers’ names and addresses. Each line of the file contains information about a single person. Different items of data, such as the researcher’s first name, last name, affiliation, and address are separated from the others by a comma or tab character. To apply database terms to this, each line of the file is a “record” and the individual data items are called “fields.”

If you’ve used Microsoft Excel or another spreadsheet program to store lists of information, you’ve essentially created a flat file database. Flat file databases are simple to set up and understand, but are limited in their ability to represent the relationships among pieces of information. Also, since the information is stored in one long list, the time it takes to find a particular piece of data increases proportional to the length of the file.

Indexed File Databases

Indexed file databases are similar to flat file databases, except that the programs that manipulate the data files maintain indexes of one or more of the fields. The presence of an index on a field allows searches to occur much more quickly—i.e., a big file can be searched in milliseconds rather than seconds. A great many proprietary DBMSs are indexed file databases at heart, including such industry stalwarts as Microsoft Access and the Filemaker series.

In addition to these commercial systems, machines running the Unix and Linux operating systems come with a family of libraries for creating indexed file databases known as the DBM series. Because of the easy availability of these libraries, many bioinformaticists have built custom databases on top of them.

Relational Databases

Relational databases (RDBMSs) are the mainstream of serious DBMSs, and include such industry heavyweights as Oracle and DB2. Relational databases are distinguished by the following features:

1. The data is broken down into a series of “tables,” each with a set of records and fields. The structure of the tables and their relation to one another are described formally by something called the “schema.”
2. A standard query language called “SQL” (*UNIT 9.2*) is used to insert information into the database, to update it, and to look information up in it.
3. The DBMS allows you to define “constraints” on the data in order to maintain internal consistency.
4. The system provides a guarantee of no corruption of the data if multiple users update the database concurrently, even if the program that a user is using to update the data crashes unexpectedly.

Most relational DBMSs also provide the following features:

1. Network access to the database over the local area network and/or internet.
2. Support for user authentication and access controls, which limit who can access the database and what they can do once they gain access.
3. Support for backup and recovery of the database.

4. Support for a variety of programming languages. Most relational databases support the Java programming language, and many provide additional support for C, C++, Perl, and Python.

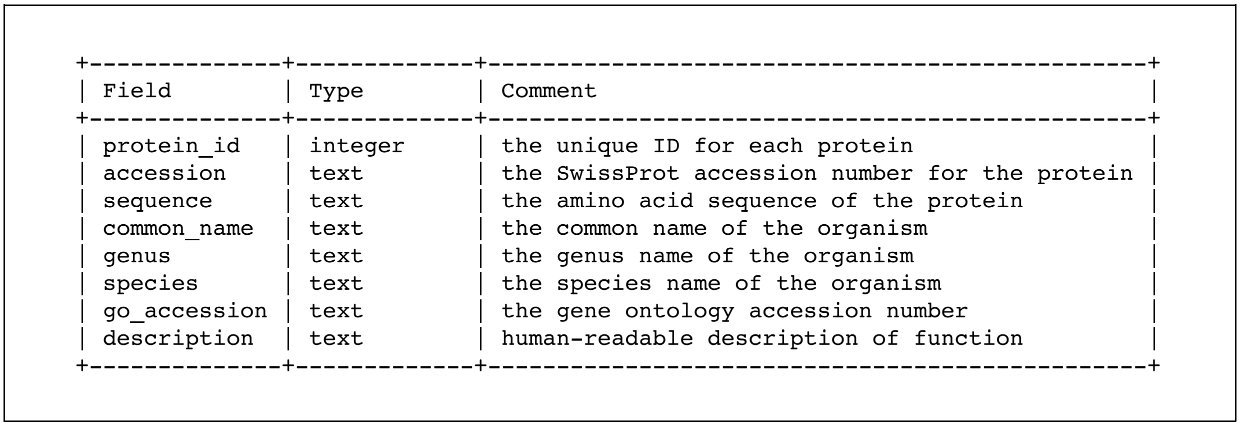
A sample schema

The way that data is broken down into tables is the essence of relational databases. For example, a database of protein sequences from different taxa might have three tables: a "sequence" table, a "taxon" table, and a protein "function" table (Fig. 9.1.1). The sequence table has fields for the name of the protein, the common name of the species it came from, and the protein sequence itself. The taxon table provides information about each species, including its formal kingdom, phylum, class, family, genus, and species. The function table contains fields that describe the function of each protein using gene ontology.

Because information about protein sequences, protein functions, and species are separated into tables, each data item is present in the database only once, making it much easier to maintain. Fields that appear in multiple tables can be used to "relate" them together. For example, the `taxon_id` can be used to relate the sequence table to the taxon table, while the `function_id` can be used to relate the sequence table to the function table. In contrast, a flat file or indexed database (Fig. 9.1.2) would force the taxon and function information to be repeated multiple times.

```
TABLE: protein_sequence
+-------------+-------------+-------------------------------------------------+
| Field       | Type        | Comment                                         |
+-------------+-------------+-------------------------------------------------+
| protein_id  | integer     | the unique ID for each protein                  |
| taxon_id    | integer     | the unique ID for each taxon                    |
| function_id | integer     | the unique ID for each protein function         |
| accession   | text        | the SwissProt accession number for the protein |
| sequence    | text        | the amino acid sequence of the protein          |
+-------------+-------------+-------------------------------------------------+

TABLE: taxon
+-------------+-------------+-----------------------------------+
| Field       | Type        | Comment                           |
+-------------+-------------+-----------------------------------+
| taxon_id    | integer     | the unique ID for the taxon       |
| common_name | text        | the common name of the organism   |
| genus       | text        | the genus name of the organism    |
| species     | text        | the species name of the organism |
+-------------+-------------+-----------------------------------+

TABLE: protein_function
+--------------+-------------+----------------------------------------+
| Field        | Type        | Comment                                |
+--------------+-------------+----------------------------------------+
| function_id  | integer     | the unique ID for the function         |
| go_accession | text        | the gene ontology accession number     |
| description  | text        | human-readable description of function |
+--------------+-------------+----------------------------------------+
```

Figure 9.1.1 A relational schema for protein sequences separates information in distinct tables to minimize redundancy.



```
+--------------+-------------+---------------------------------------------+
| Field        | Type        | Comment                                     |
+--------------+-------------+---------------------------------------------+
| protein_id   | integer     | the unique ID for each protein              |
| accession    | text        | the SwissProt accession number for the protein |
| sequence     | text        | the amino acid sequence of the protein      |
| common_name  | text        | the common name of the organism             |
| genus        | text        | the genus name of the organism              |
| species      | text        | the species name of the organism            |
| go_accession | text        | the gene ontology accession number          |
| description  | text        | human-readable description of function      |
+--------------+-------------+---------------------------------------------+
```

Figure 9.1.2 A flat-file representation of the same data will cause two proteins that share the same function of taxon to duplicate the information in "common_name," "genus," "species," "go-accession," and "description."

Constraints

Constraints are a set of consistency rules and tests that can be used to prevent inappropriate values from being entered into the database, and to maintain internal consistency. In the protein sequence database of Figure 9.1.1, constraints can be used to ensure that every protein has a sequence associated with it, and to prevent the deletion of a species from the taxon table if there are still entries that referred to it from the sequence table.

Regulating access

The ability of most relational DBMSs to be accessed from the network creates the problem of managing writes by multiple users. If two users try to update the same record simultaneously, there is a risk that one user's modifications will overwrite the other's. For this reason, relational DBMSs can lock a record so that a user can't alter it while another one is making changes. There are also techniques for "rolling back" the database to a consistent state if a user starts to make changes but later changes his mind (or the program he's using to make the changes crashes). Programming language support allows programmers to write software that accesses the database in order to add or retrieve information. In addition to allowing many database management tasks to be automated, this is the key to providing a Web-based interface to the database.

The advantage of SQL

A signal advantage of relational DBMSs is that the SQL query language is standardized (*UNIT 9.2*). Once you learn SQL, you can talk to any relational DBMS. You can also move data from one relational DBMS to another with relative ease, although there are multiple small differences among the various products that make the process not as automatic as it should be.

Relational database products

Relational database products include the heavy-duty commercial products Oracle, Sybase, Microsoft SQL Server, IBM's DB2, and Informix. There are also two popular open source (freeware) relational database products: PostgreSQL (*http://www.postgresql.org*) and MySQL (*http://www.mysql.org*).

The biggest downside of relational DBMSs is that they are complex pieces of software that have to be installed, managed, and maintained. A cadre of specially trained individuals known as Database Administrators (DBAs) are available—at a price—to manage relational databases.

The ACeDB Data Management System

The ACeDB DBMS (*http://www.acedb.org*) was designed specifically to manage biological data. It was developed to manage the *C. elegans* genome sequencing project (hence its name A *C. elegans* database), and has since been adopted for use in a number of other biological databases, particularly for various plant species.

Like relational databases, ACeDB uses a formal schema language to describe the structure of the data, and a query language to ask questions about the data. ACeDB also supports remote access via the network, a user authentication and access system, and access via the C, Java, and Perl programming languages.

Despite the similarities, ACeDB is not a relational database system. The data is structured differently (it looks like a multi-level word processor outline rather than a spreadsheet table), the schema language is different, and the ACeDB query language is only superficially similar to SQL. ACeDB offers four advantages:

1. Built-in support for nucleotide and protein sequences. Some DBMSs have traditionally had difficulty dealing with large amounts of text information, but ACeDB supports arbitrarily large stretches of DNA and protein sequences. However, this distinction has become of less importance as more relational database have begun to provide support for large text objects.
2. A rich set of schemas for representing biological data, such as genetic and physical maps, genomic sequence annotation data, phenotypic information, and bibliographic citations.
3. A graphical user interface with built-in displays for genetic and physical maps, annotated DNA sequences, phylogenetic trees, and other common biological data types (but not microarray data).
4. ACeDB is free software, and will run equally well on Windows and Unix systems.

The main disadvantage of ACeDB is its shrinking user community, which makes it difficult to get answers to problems and to exchange ideas. Although ACeDB is actively maintained and supported by its developers, it has become increasingly difficult to find programmers who have experience working with ACeDB. Figure 9.1.3 shows the schema for protein databases in ACeDB format.

```
?Protein      Taxon        UNIQUE ?Taxon   XREF   Protein
             Function      ?Function XREF Protein
             Accession     Text
             Sequence      ?DNA

?Taxon       Common_name   Text
             Genus         Text
             Species       Text
             Protein       ?Protein XREF Taxon

?Function    GO_Accession  Text
             Description   Text
             Protein       ?Protein XREF Function
```

Figure 9.1.3 The protein database as an ACeDB schema.

Other Types of DBMS

There are a variety of other DBMS architectures that you may hear about, including Object-Relational DBMSs, pure Object DBMSs, On-Line Transaction Processing (OLTP) and On-Line Analytic Processing (OLAP) databases. Object-relational DBMSs are essentially relational databases in which the restrictions on table contents have been relaxed to allow table cells to hold more complicated things than numbers or text. For example, a cell can hold a list of numbers, or even another table. Newer versions of Oracle and PostgreSQL both have object-relational features.

Pure object databases, once touted as the wave of the future, have now been relegated to the status of niche products. These databases allow programmers to create programs that store "persistent" data. That is, the program can create a large complex data structure and exit. The next time the program is run, the data structure is "magically" restored to its original state.

OLTP describes a class of DBMSs that are specialized for very high volume activity, such as airline booking systems. Similar databases are found in biology in the form of laboratory information management system (LIMS) databases, which manage highly automated procedures such as laboratory robotics.

OLAP databases are more informally known as "data warehouses." An OLAP database is a data repository that periodically collects the information from many other databases. The information is then available for leisurely data mining and analysis. An OLAP is often paired with an OLTP, because the first is good for queries but not good for high volume transactions, whereas the latter has the opposite characteristics.

CHOOSING A DBMS

Practical considerations dominate the choice of a suitable DBMS. Obtaining and installing the software itself is just the first of a long series of steps required to get a useful running system. More important in the long run are issues of maintenance and support. What support is available for the DBMS? How easy is it to find programmers and administrators who are familiar with the DBMS? Will the DBMS grow with you? And finally, what is the likelihood that you can move your data to a different DBMS should you ever need to?

Flat and Indexed File Database Management Systems

Flat and indexed file database management systems offer easy installation, a pretty graphical user interface, and an intuitive data structure; however, they are limited in their ability to handle the interrelatedness of biological data, have little in the way of internal consistency checks, and are particularly pernicious with respect to "lock in." Indexed file database systems tend to be operating system specific (e.g., Microsoft Access is only available for the Windows operating system), and although it is possible to move the data itself to another DBMS, other aspects of the database, such as custom data entry forms, cannot be easily moved to other software products.

The better commercial offerings, including Filemaker Pro and Microsoft Access, allow their databases to be used as front ends to relational DBMSs, in effect combining the nice graphical user interface with the expressiveness of a relational DBMS.

ACeDB

The lock-in concerns that apply to flat and indexed DBMSs apply doubly to ACeDB. ACeDB offers the ability to load a text file containing genetic mapping information or

BLAST hits and immediately obtain an interactive display of the data. The data can then be put on the network or displayed by a Web server. Unfortunately, once data is in ACeDB format it cannot easily be moved to any other DBMS. This, coupled with the observation that the number of bioinformaticists familiar with ACeDB is steadily shrinking, should give you pause before considering ACeDB as the basis for a new database project; however, it is an excellent product for data exploration and for projects that are expected to have a short half-life.

RDBMSs

Relational DBMSs are both well-supported and widely used in bioinformatics. Because of this, an RDBMS should probably be the first solution that you consider. The main choice is between a freeware open source solution such as MySQL or PostgreSQL, and a commercial RDBMS, such as Oracle, MS SQL Server, or DB2.

Open source products (freeware)

The MySQL database offers the core RDBMS features, including SQL, multiuser access, and transactions. It runs on Unix systems, on Windows 2000, on Linux, and on Macintosh OS X (but not OS version 9). It has a deserved reputation for being very fast, and has become the RDBMS of choice for Web site operators because of its ability to handle large loads; however, MySQL does not provide the full gamut of integrity checking that other DBMSs offer, and its implementation of SQL is incomplete. Nevertheless, its gentle learning curve and great price (free) has made MySQL the most popular RDBMS in academic bioinformatics. Many biological schemas are available for MySQL, and many full software systems use MySQL as a back end. For example, both the EnsEMBL and UCSC genome browsers are built on top of MySQL. For this reason, the *Current Protocols in Bioinformatics* units that follow this introduction use MySQL as their example RDBMS (*UNIT 9.2*).

PostgreSQL is a full-featured Object-Relational DBMS that is making inroads against MySQL in the bioinformatics community. It offers a virtually complete implementation of SQL, and an extensive repertoire of constraints and other integrity checking features. It runs on Windows 2000, Linux, and many variants of Unix, but is not documented to work with Macintosh OS X. PostgreSQL's performance is not as good as MySQL's, but this should not be an issue for the vast majority of users.

Commercial products

The commercial RDBMSs offer a large number of features not available in the open source DBMSs, including such things as fancy graphical administrative interfaces and failover databases that will take over when the master database becomes inaccessible. There is also the customer support agreement, which guarantees phone and/or on-site assistance.

All this comes at a price of course. Commercial DBMSs are typically licensed on a per-seat basis, where each seat allows a single developer access to the database. Base licenses, which typically allow for five seats, are available for a modest sum, but prices rise steeply for larger numbers of seats. In particular, if you intend to use the database as the back-end for a Web site, you may be required to purchase a number of seats equal to the number of people who will simultaneously access the Web site. This can be difficult to estimate and quite costly indeed.

Commercial RDBMSs also require more in the way of care and feeding than their open source cousins. The Oracle database, in particular, comes configured out of the box in such a way that its performance is extremely poor. It requires extensive configuration ("tuning") in order to achieve its full potential. Fortunately, there is no dearth of books that describe how to achieve this; just consult the computer book shelf at your local bookstore.

Of the commercial databases in use in bioinformatics, Oracle is currently the market leader. This reflects its preeminent position in the business world at large and the consequent abundance of Oracle programmers and DBAs. This large reservoir of talent makes Oracle a good choice for bioinformatics development; however, any of the commercial RDBMSs will handle bioinformatics tasks, and the standardization of SQL is such that it is not too painful to move a database from one to another should you change your mind in the future. (This assumes that you stick to the basic features of the DBMS; as soon as you use the special features that are specific to a particular vendor's DBMS you are lost to lock-in.)

In summary, a relational database system is probably the best place to start when looking for a solution to information overload. If you already have an RDBMS handy, for example an institutional license for a commercial system, then by all means use it. Otherwise I recommend starting with either of the open source engines, MySQL or PostgreSQL. When and if you run up against the limitations of the open source product, you can always move your data to a commercial DBMS later.

Using DBMSs

The protocols contained within this chapter describe how to use databases to solve common problems, such as storing a significant amount of sequence information. Although some of the protocols are based on MySQL and others ACeDB, many broad aspects of interacting with the database are shared in common. The following are important steps in working with a DBMS.

Install the database software

Modern DBMSs consist of two components: a database server and a database client. The server is the heavyweight component of the system; it manages the disk files on which the database is physically stored, and handles backup and recovery operations. Database server software is typically installed in such a way that the server application is started automatically when the computer boots up.

The database client is a lightweight application that interacts with the user. It accepts requests from the desktop or command line, and forwards the requests to the server. The server's response is then formatted and displayed for the user. The client can run on the same machine as the server, or can connect to the server remotely via the network. It is also commonplace for there to be several different types of clients. For example, MySQL has a text-only client that comes with the MySQL package, but the server can also be accessed from the graphical application Microsoft Access, which serves as an alternative client on Microsoft Windows platforms.

The steps for installing software under Unix are given in *APPENDIX 1C*.

Create user accounts

Because a DBMS can be accessed remotely via the network, access to the database must be restricted, either by defining a set of computers that are allowed to connect to the database, or by creating a set of users who are allowed to connect to the database by providing an authorized username and password. Typically one user account has special database administrator (DBA) permissions, which allows its owner to add and delete other users, create databases, and perform other administrative tasks. Other user accounts have restricted privileges. At the discretion of the DBA, some users may be able to read but not alter a database, while others may have less restricted privileges.

The process of creating user accounts is described for MySQL in *UNIT 9.2*, and in a forthcoming unit for ACeDB.

Create named databases

A single DBMS server can manage many individual databases. The ability to maintain separate databases allows you to keep your database of laboratory protocols separate from your database of PDF files without risk of "cross talk."

On many systems, creating a new database involves no more than choosing an easy-to-remember name and issuing a single command while logged in with DBA privileges. On some systems, you can also specify options such as the expected size of the database and other characteristics.

The process of creating a database under MySQL is described in *UNIT 9.2*.

Design the database schema

Before you can enter data into the database, you have to design the schema. The schema should be designed to represent the type of data you wish to store, and the relationships between the various data objects. The schema should also be designed with a view to the types of searches you will perform on the data once loaded.

Schema design is as much an art as a science. We will endeavor to provide guidelines for this art in many of the subsequent units in this chapter, which will present basic schemas for storing common biological data types. After experimenting with the basic schemas in cookbook style, we urge you to modify them to meet your specific needs. Modern DBMSs all provide you with the ability to modify an existing schema without invalidating the existing contents of the database.

The mechanics of writing and loading a schema into a newly-created database are covered in *UNIT 9.2*.

Load the data

Once a database has been created and initialized with a schema, you can load data into it. DBMS clients provide you with two general methods for loading data. You can load data interactively by typing the data a line at a time, or you can perform a "bulk load," in which the data is loaded rapidly from one or more text files. A similar system can be used to update existing information.

Loading, updating, and deleting the data in a relational databases is described in *UNIT 9.2*.

Query the database

After loading the data you can query the database in an unlimited number of ways in order to retrieve data and to discover relationships within it. Queries are expressed in a query language, either the standard SQL query language for relational databases, or a DMBS-specific query language such as ACeDB Query Language (AQL).

UNIT 9.2 introduces the SQL query language, while future modules will describe AQL.

Contributed by Lincoln Stein
Cold Spring Harbor Laboratory
Cold Spring Harbor, New York

Structured Query Language (SQL) Fundamentals

The Structured Query Language (SQL) is the universally accepted mechanism for accessing and manipulating data stored in a relational database management system (RDMS). SQL is a text-based language that allows the user to fully describe the hierarchical structure of a relational database in a query, making it possible to concoct arbitrarily complex and powerful queries in a straightforward manner. This unit will use the MySQL database to show how to use the SQL language to create and alter tables (see Basic Protocol 1 and Support Protocol 1), populate them with data (see Basic Protocols 2, 3, and 4, and Alternate Protocol 1), and then extract the data in a sophisticated manner (see Basic Protocol 5).

The examples in this unit will utilize the database schema shown in Figure 9.2.1. The example database contains information for the tracking of PCR primers. The database is composed of four tables: the oligonucleotides, sequence, protocol, and buffer tables. For more information about relational databases and schemas, see *UNIT 9.1*.

The general syntax of a SQL statement is a mixture of keywords, identifiers, and literals. Keywords are specific SQL commands like `CREATE` or `SELECT` (although MySQL and most other RDMSs do not care, the commands in the examples provided in this unit are written in all capital letters to distinguish them from identifiers and literals). An identifier is the name of a table, column, or other database-specific name. For example, in the primer database, `Oligo` is the identifier of a table, and `Sequence` is the identifier for a column within that table. Finally, literals are exact values to be inserted or matched. Identifiers are often tested against literals using mathematical operators like <, =, or >. For example, a clause like `protocol = 2` would compare the value in the `protocol` column of every row to see if it was equal to `2`.

Literals conform to the various datatypes available in the relational database. Table 9.2.1 lists the datatypes available in MySQL. Each column must be a specific datatype, and only that specific datatype can be stored there without an error.

CREATING A DATABASE

BASIC PROTOCOL 1

The first step in using a relational database is to create the database and tables. This step sets aside a space within the database and defines the tables and columns.

Necessary Resources

Hardware

A computer capable of running MySQL, such as one with a Windows, OS/2, or Unix-based operating system

Software

A working installation of MySQL, version 3.22.11 or higher. One must also have DBA permissions (see Support Protocol 2). MySQL is available for free under the GNU Public License. It may be downloaded from *http://www.mysql.com*.

1. In a shell window, start the MySQL client without specifying a database. In this step, and all others, the input is given in boldface, and the computer response is shown in lightface. Also, note that the interactions with the author's computer are shown. Individual computer prompts might look somewhat different. And of course, substitute the correct user name in place of the author's:

Contributed by D. Curtis Jamison

Current Protocols in Bioinformatics (2003) 9.2.1-9.2.29

Table 9.2.1 MySQL Data Types

Data type[a]	MySQL datatype[b]	Description
Binary	longblob	Binary Large Object used to store data that is not character-based. The long-, medium-, and tiny- prefix refers to the amount of storage to be set aside for the object.
	mediumblob	
	tinyblob	
Character	char	Array-based character storage up to 255 characters in length. char and nchar are fixed lengths, nvarchar and varchar are variable lengths (the storage grows and shrinks as needed). nchar and nvar char store the extended Unicode character strings rather than ASCII.
	nchar	
	nvarchar	
	varchar	
Date and time	datetime	Time-based data type. Datetime stores any date and time from 1000 to 9999 A.D.
	timestamp	Timestamp stores from 1970 to 2037 A.D.
	year	Year stores the two (1970-2069) or four (1901-2155) digit years
Decimal	decimal	Exact numeric values. These two terms are synonymous.
	numeric	
Double precision	double	Double-precision exact numeric values. These two terms are synonymous.
	real	
Floating point	float	Stores floating point numbers with a precision of 8 or less.
Integer	bigint	Integer numbers. The standard int is between −2,147,483,548 and 2,147,483,547. Big ints are between -9×10^{18} and 9×10^{18}.
	int	
	mediumint	Mediumints are between −8,338,608 and 8,388,607.
	smallint	Smallints are between −32,758 and 32,757.
	tinyint	Tinyints are between −128 to 127.
Text	longtext	Textual data like memo fields or long descriptions. A tinytext is the same size as a varchar, while a longtext holds text data of up to 4,294,967,295 characters.
	mediumtext	
	tinytext	
Non-standard types	enum	A char datatype where each of the entries must correspond to a list of possible values. An enum can handle up to 65,535 values.
	set	A set can handle up to 64 values.

[a] SQL99 ANSI standard type definitions.

[b] The datatype names for MySQL are listed and are grouped by the SQL99 ANSI standard type definitions.

```
transposon:cjamison% mysql -u cjamison -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or /g.
Your MySQL connection id is 12 to server version: 3.23.46
Type 'help;' or '/h' for help. Type '/c' to clear the
buffer.
mysql>
```

The MySQL program provides a text interface to the MySQL server. The `-u` *command tells the MySQL program program to login to the database using the user name specified. The* `-p` *command tells the program to prompt for a password.*

The MySQL program is the general way of interaction with the database. SQL as well as MySQL commands are typed in following the `mysql>` *prompt. All commands end with a semicolon or* `/g` *but the commands can be stretched across multiple lines. The MySQL program indicates that a new line is part of the previous command by switching from the* `mysql>` *prompt to the* `->` *prompt. Previous lines can be recalled using the up- and down-arrow keys.*

Forgetting to put the semicolon at the end of the statement is the most common error of novice and intermediate MySQL users. But if you enter a command and hit the Return key prematurely, simply put the semicolon on the line that you are presently on and pretend that a multiline command was intended.

2. Use the `CREATE DATABASE [database_name]` command to create a database:

```
mysql> CREATE DATABASE primers;
Query OK, 1 row affected (0.00 sec)
```

The `CREATE DATABASE` *command sets aside a database directory named with the identifier supplied as the* `[database_name]`. *In this example,* `primers` *is the database name.*

3. Switch to the new database with the `USE [database_name]` command:

```
mysql> USE primers;
Database changed
```

4. Create the `oligo` table using the `CREATE TABLE [table_name] [column_list]` command. The `[table_name]` is an identifier for the table, and the `[column_list]` is a comma-separated list of column names followed by the datatype and any options. The list is enclosed in parentheses. This example creates the `oligo` table shown in Figure 9.2.1.

```
mysql> CREATE TABLE oligo
-> (ID INT NOT NULL UNIQUE,
-> direction ENUM(“Forward”, “Reverse”),
-> sequence VARCHAR(50),
-> derived_from VARCHAR(12),
-> protocol INT DEFAULT 1);
Query OK, 0 rows affected (0.35 sec)
```

Note that this is a multiline command, and MySQL does not process the command until the semicolon is reached. When the user hits the Return key, the MySQL program provides the `->` *prompt to continue the command.*

The column list specifies how the table is to look, defining what data can be put in. The general form is `[column_name] [datatype options]`. *The column name has to be a unique identifier for the table: duplicate column names are not permitted within a table (although other tables can have columns with the same name). The column options are listed in Table 9.2.2. Some of the more common ones are used in the* `oligo` *table creation above. For example, the* `ID` *field is going to be our key, a unique identifier for*

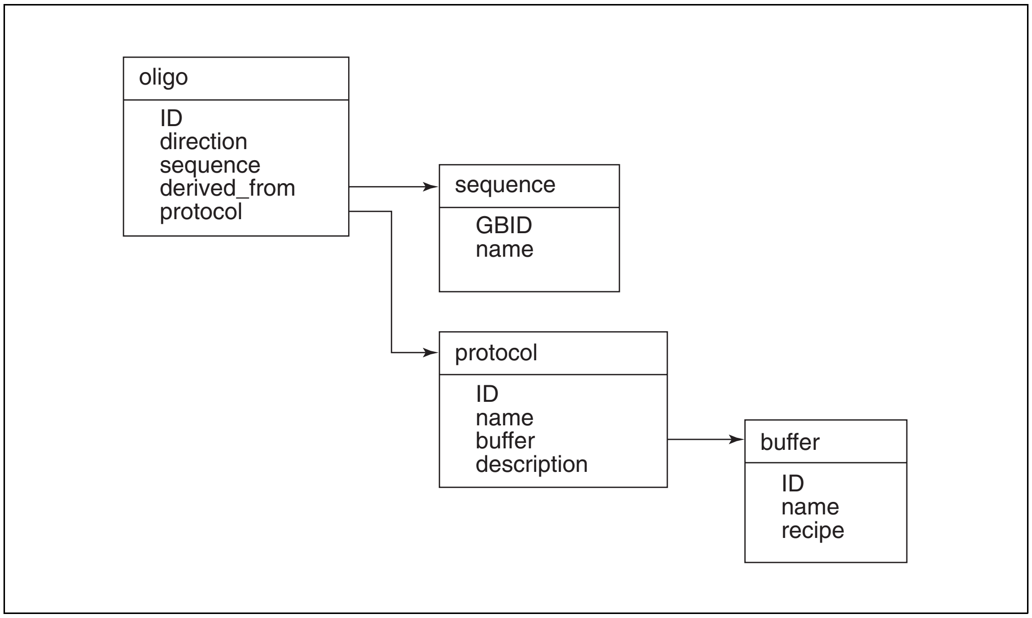


Figure 9.2.1 Example schema for protocol examples. Each box represents a table to be used in the `primers` database. The table name is at top, and the fields are shown within. Arrows from fields to other tables shows the relational schema.

Table 9.2.2 Column Creation Options: Allowed Keywords for Defining Column Specifications in the `CREATE TABLE` Command

Keyword	Arguments	Description

`NULL`
`NOT NULL`
		Allows or disallows `NULL` values. The default is to allow `NULL`s

`DEFAULT`	default value	Defines a value to input if not otherwise specified
`AUTO_INCREMENT`		Automatically set entry to next sequential number
`PRIMARY KEY`	reference (optional)	Specifies how each row is uniquely identified
`CHECK`	expression	Checks input to be valid as defined by the expression
`INDEX`	index column name list	Assigns the column to be an indexed column in all indices listed
`UNIQUE`		Prevents values from being duplicated in the column, creating a key
`CONSTRAINT`	constraint name	Constrains input to a previously defined rule

every primer. Therefore, the column options are `NOT NULL` *because every row must have an* `ID`*, and* `UNIQUE` *because the* `ID` *for every row must be different. This makes the* `ID` *a field that always differentiates rows.*

Another useful option is shown in the `protocol` *column. Here the* `DEFAULT` *value for entries into this column is going to be* `1`*. Thus, when entering data into the* `oligo` *table (see Basic Protocol 2), if a* `protocol` *is unspecified, it is automatically set to* `1`*. For the most part, the column options can be applied to any data type. For some data types, an argument in parentheses is required. For example, the number in parentheses following the* `Varchar` *type tells how many characters the field will hold.*

5. Create the rest of the tables:

```
mysql> CREATE TABLE sequence
-> (GBID VARCHAR(12) NOT NULL UNIQUE,
-> name VARCHAR(100));
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE protocol
-> (ID INT NOT NULL UNIQUE,
-> name VARCHAR(100),
-> buffer INT NOT NULL,
-> description LONGTEXT);
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE buffer
-> (ID INT NOT NULL UNIQUE,
-> name VARCHAR(100),
-> recipe LONGTEXT);
Query OK, 0 rows affected (0.00 sec)
```

6. Check your database with the SHOW command:

```
mysql> SHOW tables;
Tables_in_primers
buffer
oligo
protocol
sequence
4 rows in set (0.00 sec)
```

These steps have created a database structure within MySQL that can be used to store primer data. Support Protocol 1 shows how to make changes in the database. Basic Protocol 2 will show how to populate the database with actual data.

CHANGING A SCHEMA

SUPPORT PROTOCOL 1

Often, even the best designed schema is inadequate. As the use of a database evolves, it may be necessary to revise and extend the tables in order to accommodate changes in data collection or analysis. SQL has several commands that allow the database administrator to alter the schema.

For example, upon reviewing the schema for the primers database, a couple of oversights become apparent. First, it appears that the schema lacks a field to show where the primer is in the sequence. Second, if the maximum primer length will be 35 nucleotides, setting aside 50 characters is somewhat wasteful.

Necessary Resources

Hardware

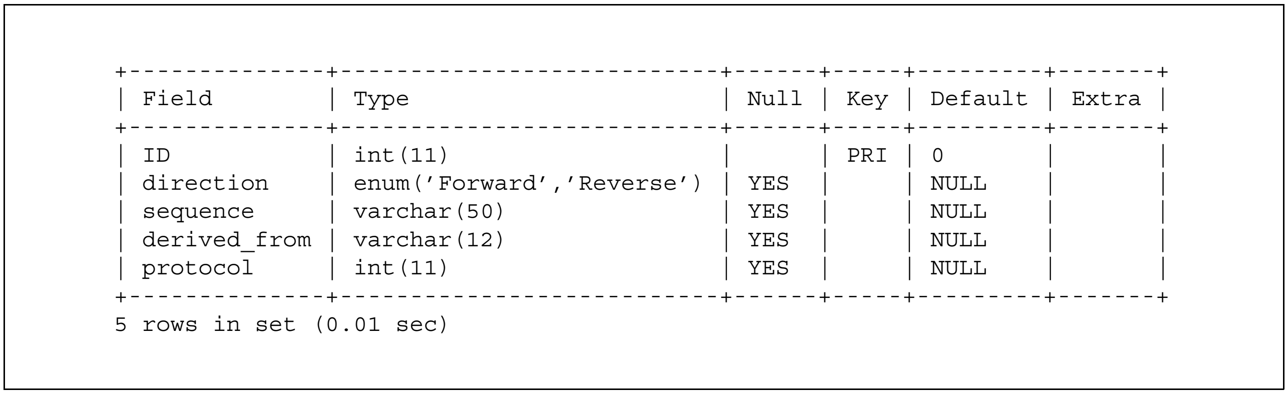
A computer capable of running MySQL, such as one with a Windows, OS/2, or Unix-based operating system

Software

A working installation of MySQL, version 3.22.11 or higher. One must also have DBA permissions. MySQL is available for free under the GNU Public License. It may be downloaded from *http://www.mysql.com.*

Files

The primers database created in Basic Protocol 1. Schema shown in Figure 9.2.1



```
+--------------+--------------------------+------+-----+---------+-------+
| Field        | Type                     | Null | Key | Default | Extra |
+--------------+--------------------------+------+-----+---------+-------+
| ID           | int(11)                  |      | PRI | 0       |       |
| direction    | enum('Forward','Reverse') | YES |     | NULL    |       |
| sequence     | varchar(50)              | YES  |     | NULL    |       |
| derived_from | varchar(12)              | YES  |     | NULL    |       |
| protocol     | int(11)                  | YES  |     | NULL    |       |
+--------------+--------------------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Figure 9.2.2 Output obtained upon submitting the command in Support Protocol 1, step 1. The annotation in parentheses following the type indicates the valid range or size of the variable.

1. After connecting to the database (see Basic Protocol 1), examine the tables. Use the DESCRIBE command to list the columns and attributes found in a table. This example will use the oligo table. In this step, and all others, the input is given in boldface, and the computer response is shown in lightface. Also, note that the interactions with the author's computer are shown. Individual computer prompts might look somewhat different.

   ```
   mysql> DESCRIBE oligo;
   ```

   The output that will be displayed upon submitting this command is shown in Figure 9.2.2.

   The DESCRIBE command returns a table listing the column names (the Field column), the datatype stored in the column (the Type column), whether NULLs are allowed (the Null column), what type of key the column is (the Key column), the default value (the Default column), and any additional information (the Extra column).

2. Add a column for the position with the ALTER command. The syntax for the command is ALTER TABLE [table_name alter_command]. The [alter_command] keywords range from adding a column to renaming the table. The full set of [alter_command] keywords is given in Table 9.2.3. This example uses the ADD COLUMN keyword, which allows the user to define a new column using the same syntax as the CREATE table command (see Basic Protocol 1). To add a column called position, which will contain integer values:

   ```
   mysql> ALTER TABLE oligo ADD COLUMN position INT;
   Query OK, 0 rows affected (0.06 sec)
   Records: 0 Duplicates: 0 Warnings: 0
   ```

   Note that the ALTER command reports back how many records were affected by the schema change. The ALTER command actually makes a temporary copy of the table, alters the copy, deletes the original, and renames the copy to the original name. The number of records affected should be equal to the number of rows in the table. In this case, the number is zero, since the table does not contain any data at this point.

   Look at the oligo table using the DESCRIBE command from step 1. The output now shows six columns, with position being the last one.

3. Now, modify the sequence column to reflect the shorter primer length. Again, use the ALTER TABLE command, this time using the MODIFY COLUMN keywords. The change in the column is specified by using the CREATE column syntax.

Table 9.2.3 Allowed Options for the `ALTER TABLE` Command

Alter keyword	Arguments	Description
`ADD COLUMN`	column name, specification	Adds a new column to the table using the specification as defined in `CREATE TABLE` command (see Basic Protocol 1)
`ADD INDEX`	index name, column name	Adds a table index based upon a specific column
`ADD PRIMARY KEY`	column name	Makes the named column a primary key
`ADD UNIQUE`	column name	Sets the column attribute to `UNIQUE`
`ALTER COLUMN`	column name, `SET` or `DROP DEFAULT`, literal	Changes the default value attribute of a column, with the `SET DEFAULT` command, a literal value must be supplied
`CHANGE COLUMN`	column name, specification	Changes the column specification using the syntax in the `CREATE TABLE` command (see Basic Protocol 1)
`DROP COLUMN`	column name	Deletes a column
`DROP PRIMARY KEY`		Deletes the primary key (but not the column)
`DROP INDEX`	index name	Deletes the index
`MODIFY COLUMN`	column name, datatype, attributes	Changes the data type and attributes
`RENAME AS`	table name	Renames the entire table

```
mysql> ALTER TABLE oligo MODIFY COLUMN sequence
VARCHAR(35);
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Again using the `DESCRIBE` *command verifies that the* `sequence` *is now of type* `VARCHAR(35)` *rather than the original* `VARCHAR(50)`.

ADDING USERS AND PERMISSIONS

SUPPORT PROTOCOL 2

When newly installed, the MySQL database defines a single-user named root, and an anonymous user with no name. Neither account has a password. Basically, this means the MySQL database has no security and anyone can connect to the database. For security, each user of the MySQL RDMS should have their own password-protected account.

Each user has a set of privileges defined for each database, which determines their usage rights. The usage rights determine what the user can and cannot do to the data within the database. Table 9.2.4 shows the privileges available to users. The privilege levels fall into three broad categories: data access, data manipulation, and database manipulation. Roughly, the first category represents data consumers, the second represents data generators, and the third represents database administrators. Depending on why the database was set up, the majority of users will fall into either the consumer or the generator category. Administration should be restricted to one or two people.

For the example primer database, suppose there are two users in addition to the administrator. One, Maureen Johnson, might be in charge of generating the primers and entering the data where she would need to have enter and edit privileges. The other, Brian Smith, is in charge of running the PCR, thus he needs to be able to look up data about the primers, but should not be allowed to edit the data. These two users need to be added to the database.

Table 9.2.4 User Privilege Types

Privilege[a]	Category[b]	Note
ALL	Admin	Has access to all functions
SELECT	Access	Can only make queries
DELETE	Data	
INSERT	Data	
UPDATE	Data	
ALTER	Admin	
CREATE	Admin	
DROP	Admin	
INDEX	Admin	
FILE	Admin	
RELOAD	Admin	
SHUTDOWN	Admin	
PROCESS	Admin	
USAGE	None	Currently stands for no privileges

[a] The ALL privilege encompasses every other privilege, all others have to be allocated specifically.

[b] Privileges categorized as "Access" are those for data access; the "Data" category privileges are for data manipulation; and the "Admin" category privileges are for database manipulation.

Necessary Resources

Hardware

Computer capable of running MySQL

Software

Newly installed copy of MySQL, version 3.22.11 or higher

Files

The `primers` database created in Basic Protocol 1

Restricting access

1. Launch the MySQL shell program as root, using the MySQL database:

```
transposon:cjamison%mysql -u root mysql
```

2. Set the root password:

```
mysql> UPDATE user SET Password=PASSWORD('new_password')
WHERE user='root';
mysql> FLUSH PRIVILEGES;
```

The `root` *login to the database is now password protected. Whenever you log in next, you will have to specify the* `-p` *option to get a* `password` *prompt and supply whatever password you decided to use (something other than* `new_password`*).*

Adding users

3. Determine the permission level allowed for each user.

Since Maureen Johnson is adding and editing data, give her `SELECT`, `INSERT`, `UPDATE`, *and* `DELETE` *privileges. Brian Smith will get only* `SELECT` *privileges.*

4. Determine user names and passwords for each user.

   The format of the user name is rather like that of an e-mail address: user@machine. *The* user *portion should be the user's Unix login name, and the* machine *portion should be the name of the machine the user is going to be accessing the database from. Following this convention allows users to dispense with the* -u *option. For example, for the author to access his SQL database from his account on his development server, he would create a user name that looked like* cjamison@haydn.gmu.edu.

   Since the users in this example are going to be using the database from the same machine it is installed on, use the localhost *alias for the machine:* mjohnson@localhost *and* bsmith@localhost. *This will allow Maureen to connect to the database by typing* transposon:mjohnson% **mysql -p** *and entering her password.*

   Note the use of the –p *argument to get a* password *prompt. Without the* -p, *MySQL refuses to connect to the database the error message* ERROR 1045: Access denied for user: maureen@localhost (Using password: NO) *to remind you that you need to use a password.*

   Some database administrators take the easy way out and assign a password based upon the user's name, relying upon the user to change their password into something more secure. However, users are generally lazy and an easily remembered (and guessed) password is often not changed. It is best to assign a very cryptic password from the start.

5. If needed, start MySQL as root, using the primer database.

```
transposon:cjamison% mysql -u root -p primers
Enter password:
```

6. Use the GRANT command to both create users and confer privileges.

   The GRANT *command is multifunctional in that it not only will change the privileges of an existing user, but will create a new user with specified privileges if the user does not exist. Typos in the user name can have unintended affects, so it is important to be very careful when doing this and any other system administration task.*

   The GRANT *command syntax is* GRANT [permission_list] ON [table_list] TO [user_name] IDENTIFIED BY [password], *where the* [permission_list] *is a comma-separated list of permissions from Figure 9.2.5 and* [table_list] *is a list of tables within the current database to apply those permissions. An asterisk is the wild-card symbol denoting all tables in the database. The* [password] *is written as regular unencrypted text delimited with quotation marks.*

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON * TO
-> mjohnson@localhost IDENTIFIED BY 'H22ASK8';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT SELECT ON * TO bsmith@localhost
-> IDENTIFIED BY 'D9KLL32';
Query OK, 0 rows affected (0.00 sec)
mysql> FLUSH PRIVILEGES;
```

   The FLUSH PRIVILEGES *command propagates the changes made in the database. Otherwise the new privileges (and logins) would not work until the MySQL server is restarted.*

7. Check the permissions for each person (see Fig. 9.2.3).

8. Restrict privileges with the REVOKE command. To deny Maureen the ability to delete information from the database, remove that permission from her account by following Figure 9.2.4.

```
mysql> show grants for mjohnson@localhost;
+-----------------------------------------------------------------------------------+
| Grants for mjohnson@localhost|
+-----------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'mjohnson'@'localhost' IDENTIFIED BY PASSWORD '49ceb761562ce1c1'|
| GRANT SELECT, INSERT, UPDATE, DELETE ON primers.* TO 'mjohnson'@'localhost'|
+-----------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
mysql> show grants for bsmith@localhost;
+--------------------------------------------------------------------------------------+
| Grants for bsmith@localhost|
+--------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'bsmith'@'localhost' IDENTIFIED BY PASSWORD '29b91ad7683e3d19'|
| GRANT SELECT ON primers.* TO 'bsmith'@'localhost'|
+--------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

Figure 9.2.3 Checking permissions for a specific user. The first line shows that the users are allowed to connect to the database server, and are required to login. The following line(s) show their privileges on specific databases.

```
mysql> REVOKE DELETE ON * FROM mjohnson@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for mjohnson@localhost;
+---------------------------------------------------------------------------------------+
| Grants for mjohnson@localhost|
+---------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'mjohnson'@'localhost' IDENTIFIED BY PASSWORD '49ceb761562ce1c1' |
| GRANT SELECT, INSERT, UPDATE ON primers.* TO 'mjohnson'@'localhost'
+---------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

Figure 9.2.4 The `grants` table reflects the revocation of the `DELETE` privilege.

BASIC PROTOCOL 2

ADDING DATA TO A TABLE

After the tables are finished, the next big job is getting data into the database. Data items can be added either singly in an interactive mode, or en mass in a batch data load (see Alternate Protocol 1). Both methods are row-based methods; that is, data are placed into the database one table row at a time. Thus, it is important to have the data somewhat organized prior to sitting down for data entry; otherwise much time can be lost.

This protocol inserts four hypothetical primers into the `oligo` table. The data associated with each primer is arranged in a logical manner, as one might find in a spreadsheet or a laboratory notebook. A good database schema serves to aid data input, as the tables reflects the manner in which data are either collected or arranged.

Necessary Resources

Hardware

A computer capable of running MySQL, such as one with a Windows, OS/2, or Unix-based operating system.

Software

A working installation of MySQL, version 3.22.11 or higher. One must also have DBA permissions. MySQL is available for free under the GNU Public License. It may be downloaded from *http://www.mysql.com*.

Files

The `primers` database created in Basic Protocol 1, as altered in Support Protocol 1

1. Connect to the database as described in Basic Protocol 1, step 1. Switch to the primer database with the `USE` command as described in Basic Protocol 1, step 3.

2. Insert data rows. In this step, and all others, the input you should type is given in boldface, and the computer response is shown in lightface. Also, note that the interactions with the author's computer are shown. Individual computer prompts might look somewhat different:

```
mysql> INSERT INTO oligo VALUES (1, 'Forward',
'ATCGGTATGATCAT', 'G19982', 1, 3111);
Query OK, 1 row affected (0.32 sec)
```

   The `INSERT INTO` *command takes a table name (*`oligo`*) and a list of comma-separated* `VALUES`*, which are enclosed in parentheses. Numeric values are entered directly, while textual data (*`ENUM` *and* `VARCHAR`*) are enclosed by quotation marks. Note that the order of the values is important.* `INSERT` *places the values into the table in the order in which the columns appear in the table.*

3. The above procedure is sufficient if the table never changes, but to avoid confusion specify a column list that explicitly tells the database what order the data are in:

```
mysql> INSERT INTO oligo (ID, direction, sequence,
derived_from, protocol, position) VALUES (2, 'Reverse',
'AGACATTGATACGA', 'G19982', 1, 3433);
Query OK, 1 row affected (0.00 sec)
```

   Even though the order is the same, this form of the command assures that the data are inserted into the correct columns, even if the layout of the table is altered with additional columns.

4. Another issue arises when values are not available for all columns. If the column list is unspecified, explicitly set the missing column values to `NULL`:

```
mysql> INSERT INTO oligo VALUES (3, 'Forward',
'CTTAGTCGATCCAG', NULL, NULL, NULL);
Query OK, 1 row affected (0.08 sec)
```

   The `NULL` *value is written like a keyword (e.g., no quotation marks). Alternatively, if the user specifies a column list such as (*`ID`*,* `direction`*,* `sequence`*) it is possible to skip the* `NULL` *specification.*

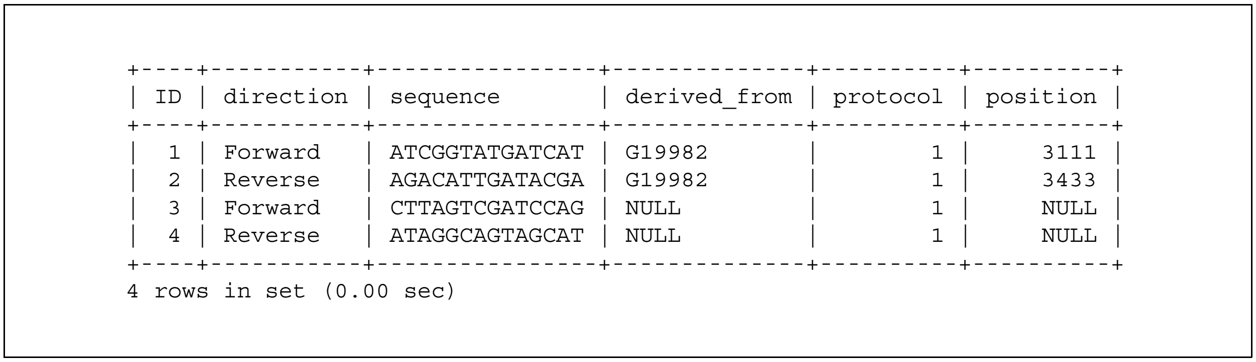
5. Another way to deal with the `INSERT` command is to use the `SET` keyword and then a comma-separated list of columns and values tied together with an "=" sign. Again, numeric values are written as literals and text data are enclosed with quotation marks:

```
mysql> INSERT INTO oligo SET ID=4, direction='Reverse',
sequence='ATAGGCAGTAGCAT';
Query OK, 1 row affected (0.00 sec)
```

   Using the `SET` *version of* `INSERT` *has the advantage that it is clearly understandable when read.*

6. While the `SET` keyword makes it easier and more understandable to put values into the database, it is still important to understand the table structure to prevent trying to put the wrong type of data into the wrong column. For example, the `ID` column of the `oligo` table is defined to be `UNIQUE` and `NOT NULL`. Attempting to insert a value that is already present or to put in a `NULL`, will result in an error:

```
mysql> INSERT INTO oligo SET ID = 1, direction =
'Reverse', sequence='ATTATTTATT';
ERROR 1062: Duplicate entry '1' for key 1
```



```
+----+-----------+----------------+-------------+----------+----------+
| ID | direction | sequence       | derived_from | protocol | position |
+----+-----------+----------------+-------------+----------+----------+
|  1 | Forward   | ATCGGTATGATCAT | G19982      |        1 |     3111 |
|  2 | Reverse   | AGACATTGATACGA | G19982      |        1 |     3433 |
|  3 | Forward   | CTTAGTCGATCCAG | NULL        |        1 |     NULL |
|  4 | Reverse   | ATAGGCAGTAGCAT | NULL        |        1 |     NULL |
+----+-----------+----------------+-------------+----------+----------+
4 rows in set (0.00 sec)
```

Figure 9.2.5 Output obtained upon submitting the command in Basic Protocol 2, step 7.

```
mysql> INSERT INTO oligo VALUES (NULL, 'Forward',
'ATTGTAAGTAA', NULL, NULL, NULL);
ERROR 1048: Column 'ID' cannot be null
```

The error messages returned from the MySQL program are not fatal, and it is possible to go back and edit the INSERT *statements. However, if using the batch loading procedure in Alternate Protocol 1, the errors will interrupt the entire process.*

7. Verify the data entry:

```
mysql> SELECT * FROM oligo;
```

The output that will be displayed upon submitting this command is shown in Figure 9.2.5.

The SELECT *command returns data from the table. Basic Protocol 5 will deal with this statement in depth. For now, just note that this version of the command returns all entries from every row. Also note that for primers* 3 *and* 4*, the* protocol *has been set to* 1*, even though data was not explicitly entered for it, since we specified a default value of* 1 *for the column when we created the table (see Basic Protocol 1). Otherwise, the default value for columns is* NULL*, as can be seen in the* derived_from *and* position *columns.*

ALTERNATE PROTOCOL 1

A BATCH METHOD TO LOAD DATA INTO A TABLE

Although the INSERT method is good for data entered row by row, it can be tedious if one has a lot of data to enter. For example, the information entered into the primers database might have been previously stored in a spreadsheet, with several hundred primers that would be nearly impossible to input individually by hand. Fortunately, there is an easy way to input many rows at once.

The LOAD DATA command inserts rows into the database from a text file. The text file should be a delimited text file, with the data values specified in column order, with one row per line. Most spreadsheet programs will output some form of delimited text file, the most common being tab-delimited, meaning that each value is separated by a tab character. The tab character is the default delimiter, but can be altered so any type of delimited file can be used.

The order of the values in the text file must be the same as the order of the columns in the table into which one is loading the data. The LOAD DATA command works like the INSERT INTO command without column specifications (see Basic Protocol 2, step 2).

Necessary Resources

Hardware

A computer capable of running MySQL, such as one with a Windows, OS/2, or Unix-based operating system

Software

A working installation of MySQL, version 3.22.11 or higher. One must also have DBA permissions. MySQL is available for free under the GNU Public License. It may be downloaded from *http://www.mysql.com.*

Files

The `primers` database created in Basic Protocol 1, as altered in Support Protocol 1

Tab-delimited trial data file, `oligo.txt`. This file is available at the Current Protocols Website: *http://www3.interscience.wiley.com/c_p/cpbi_sample datafiles.htm.*

1. If needed, start the MySQL program using the `primers` database. Providing the name of the database as the last argument causes the MySQL program to do an automatic `USE`:

```
transposon:cjamison% mysql -u cjamison -p primers
Enter password:
```

Table 9.2.5 Keywords That Allow Change in Defaults for Delimiters, Line Terminators, and Other Aspects Controlling Reading the Data From the File

`LOW_PRIORITY | CONCURRENT`	Specifies the priority of the data load. `LOW_PRIORITY` makes the data load wait until no other clients are reading from the table. `CONCURRENT` allows other clients to fetch from the table while the load is in progress. The default behavior is to block all other clients.
`REPLACE | IGNORE`	Specifies how duplicate records are handled. `REPLACE` allows new rows to replace old rows with the same unique key value. The default behavior is `IGNORE`, which skips the new row that is a duplicate.
`FIELDS` *(must be followed by at least one of the following options)*	
`TERMINATED BY [string]`	Changes the field separator from the tab character to that specified by `[string]`. Another common field separator is the comma character.
`ENCLOSED BY [string]`	Changes the field enclosure from nothing to that specified by `[string]`.
`ESCAPED BY [string]`	Changes the escape string from \\ to that specified by `[string]`. The loader ignores any field prefaced by the escape `[string]`.
`LINES TERMINATED BY [string]`	Changes the line end character from `\n` to that specified by `[string]`. This option is dangerous to change randomly.
`IGNORE` [number] `LINES`	Tells the loader to skip over a certain number of lines. Useful when reading from files that have header lines.

2. Load the `oligo` table. In this step, and all others, the input is given in boldface, and the computer response is shown in lightface. Also, note that the interactions with the author's computer are shown. Individual computer prompts might look somewhat different:

```
mysql> LOAD DATA LOCAL INFILE "oligo.txt" INTO TABLE
oligo;
Query OK, 8 rows affected (0.01 sec)
Records: 8 Deleted: 0 Skipped: 0 Warnings: 0
```

The LOAD DATA *command has several options. The* LOCAL *keyword tells the computer to look for the input file on the local computer rather than the database server (which is important if connecting to the database from a remote client). The* INFILE *keyword is required to specify the string following as the name of the file. The* INTO TABLE *command specifies the table to insert the data into.*

Other keywords change the defaults for delimiters, line terminators, and other aspects controlling reading the data from the file and are shown in Table 9.2.5. Using the default of IGNORE*, any rows with duplicate primary keys would be skipped (and would show in the skipped: statistic).*

3. Check the table using the SELECT command again:

```
mysql> SELECT * FROM oligo;
```

The table should have eight rows.

BASIC PROTOCOL 3

REMOVING DATA FROM A TABLE

Often it will become necessary to remove data from the database. The DELETE FROM command is the reciprocal action of INSERT INTO (see Basic Protocol 2). The command also requires a WHERE clause that controls what rows are deleted by matching values within the columns of the table.

Necessary Resources

Hardware

A computer capable of running MySQL, such as one with a Windows, OS/2, or Unix-based operating system

Software

A working installation of MySQL, version 3.22.11 or higher. One must also have DBA permissions. MySQL is available for free under the GNU Public License. It may be downloaded from *http://www.mysql.com*.

Files

Primer database created in Basic Protocol 1, as altered in Support Protocol 1, loaded with the data from `oligo.txt` as shown in Alternate Protocol 1

1. If needed, start the MySQL program using the `primers` database:

```
transposon:cjamison%  mysql -u cjamison -p primers
Enter password:
```

2. Delete some records. In this step, and all others, the input is given in boldface, and the computer response is shown in lightface. Also, note that the interactions with the author's computer are shown. Individual computer prompts might look somewhat different:

```
mysql DELETE FROM oligo WHERE derived_from = 'M10992';
Query OK, 2 rows affected (0.00 sec)
```

The `WHERE` *clause controls what records are selected for deletion, so care must be taken when writing the clause to make it as specific as possible. More information on formulation of* `WHERE` *clauses is found in Basic Protocol 3. The* `DELETE FROM` *command returns the number of rows in the table that matched the* `WHERE` *clause criteria and were deleted.*

3. Check the `oligo` table again with the `SELECT` command:

```
mysql> SELECT * FROM oligo
```

Note that primer records `3` *and* `4` *were the rows removed. Both primers had the value* `M10992` *in the* `derived_from` *column.*

CHANGING DATA IN A TABLE

BASIC PROTOCOL 4

The most common data maintenance task is probably updating data to reflect new conditions. For example, suppose it is necessary to `SET protocol = 3` for all primers derived from sequence `G172889`. It is possible to find all the rows that have a `[derived_from]` value of `G172889`, delete those rows, and then insert the corrected row back into the database. Obviously this will be a tedious job if there are a lot of rows.

The `UPDATE` command is used to edit specific rows. `UPDATE` uses a `WHERE` clause to find specific rows, and a `SET` clause to alter the value of the columns.

Necessary Resources

Hardware

A computer capable of running MySQL, such as one with a Windows, OS/2, or Unix-based operating system.

Software

A working installation of MySQL, version 3.22.11 or higher. One must also have DBA permissions. MySQL is available for free under the GNU Public License. It may be downloaded from *http://www.mysql.com*

Files

The `primers` database created in Basic Protocol 1, as altered in Support Protocol 1.

1. If needed, start the MySQL program using the primer database:

```
transposon:cjamison% mysql -u cjamison -p primers
Enter password:
```

2. Update the database. In this step, and all others, the input is given in boldface, and the computer response is shown in lightface. Also, note that the interactions with the author's computer are shown. Individual computer prompts might look somewhat different:

```
mysql UPDATE oligo SET protocol = 3 WHERE derived_from =
'G172889';
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2 Changed: 2 Warnings: 0
```

As with the `DELETE` command, the `WHERE` clause contains the criteria for finding the records to change. The `UPDATE` command returns information telling us that the `WHERE` clause matched two rows, and that two rows were changed.

3. Check the changes using the SELECT command:

```
mysql> SELECT * FROM oligo;
```

Note that primers 5 and 6 are now changed to protocol 3.

BASIC PROTOCOL 5

RETRIEVING DATA

The advantage of using SQL is the simplicity and flexibility of the language for retrieving data. The SELECT statement is used to retrieve columns from tables based upon arbitrarily complex selection criteria.

The basic structure of the command is SELECT [column_list] FROM [table]. Previous examples in this unit have used an asterisk as a “wildcard” for the [column_list] to specify all columns. However, providing a comma-separated list of column names in the column_list would restrict the results to only those columns (e.g., a column list of ID, direction, position would list only those three columns). Additionally, it is possible to add a WHERE clause to the SELECT statement and restrict the return values to only those rows that match the value.

Necessary Resources

Hardware

A computer capable of running MySQL, such as one with a Windows, OS/2, or Unix-based operating system

Software

A working installation of MySQL, version 3.22.11 or higher. One must also have DBA permissions. MySQL is available for free under the GNU Public License. It may be downloaded from *http://www.mysql.com.*

Files

The primers database created in Basic Protocol 1, as altered in Support Protocol 1

Four data files containing the larger search set: cpboligo.txt, cpbprotocol.txt, cpbbuffer.txt, and cpbsequence.txt. These files are available from the Current Protocols Web site: *http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm.*

1. If needed, start the MySQL program using the primer database:

```
transposon:cjamison% mysql -u cjamison -p primers
Enter password:
```

2. Load expanded tables. In this step, and all others, the input is given in boldface, and the computer response is shown in lightface. Also, note that the interactions with the author’s computer are shown. Individual computer prompts might look somewhat different.

   In order to make the sample queries more realistic, the database has to be a realistic size. Thus, first load real data, derived from DB-STS (see Alternate Protocol 1):

```
mysql LOAD DATA LOCAL INFILE “cpboligo.txt” INTO TABLE
oligo;
Query OK, 124 rows affected (0.01 sec)
Records: 124 Deleted: 0 Skipped: 0 Warnings: 2
```

```
mysql LOAD DATA LOCAL INFILE “cpbprotocol.txt” INTO TABLE
protocol;
```

```
Query OK, 9 rows affected (0.01 sec)
Records: 9 Deleted: 0 Skipped: 0 Warnings: 0
```

```
mysql LOAD DATA LOCAL INFILE “cpbbuffer.txt” INTO TABLE
buffer;
Query OK, 6 rows affected (0.01 sec)
Records: 6 Deleted: 0 Skipped: 0 Warnings: 0
```

```
mysql LOAD DATA LOCAL INFILE “cpbsequence.txt” INTO
TABLE sequence;
Query OK, 43 rows affected (0.00 sec)
Records: 62 Deleted: 0 Skipped: 19 Warnings: 1
```

Using the basic `SELECT` *statement*

3. With the expanded data loaded, begin retrieving data from the tables. Using the basic `SELECT` command structure, put an asterisk into the command for the column list to obtain all the columns and examine the entire table at once:

   ```
   mysql> SELECT * FROM buffer;
   ```

   The output that will be displayed upon submitting this command is shown in Figure 9.2.6.

   The output of the `SELECT` *command (Fig. 9.2.6) is presented in tabular form. The names of the columns being viewed are at the top, separated from the rows of data by lines of dashes. After the end of the output, there is a line stating the number of rows returned by the query.*

   Note that the size of the response can be quite large. A line of text is returned for every row, and if the line is longer than the screen is wide, it will wrap down to the next line, as seen in the first example. Thus, it is best to only request the columns of interest.

4. Alternatively, one can look at a subset of the columns:

   ```
   mysql> SELECT ID, name FROM protocol;
   ```

   The output that will be displayed upon submitting this command is shown in Figure 9.2.7.

Using the `WHERE` *clause*

5. The `WHERE` clause adds much power to the query. Instead of looking at all the data, this clause restricts the results to rows that match a criteria. For example, to list all the `oligo`s that use protocol 2:

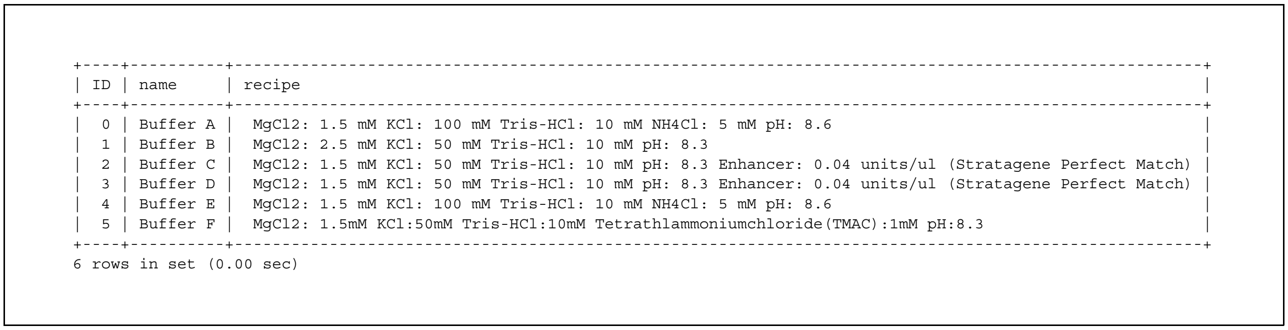
   ```
   mysql> SELECT ID, sequence, protocol FROM oligo WHERE
   protocol = 2;
   ```

   The output that will be displayed upon submitting this command is shown in Figure 9.2.8.

   Note that the column used in the `WHERE` *clause does not have to be included in the column list for the results. The clause is simply a filter to screen the rows. The clause is evaluated to a Boolean value, and only the* `TRUE` *results are included in the returned data.*

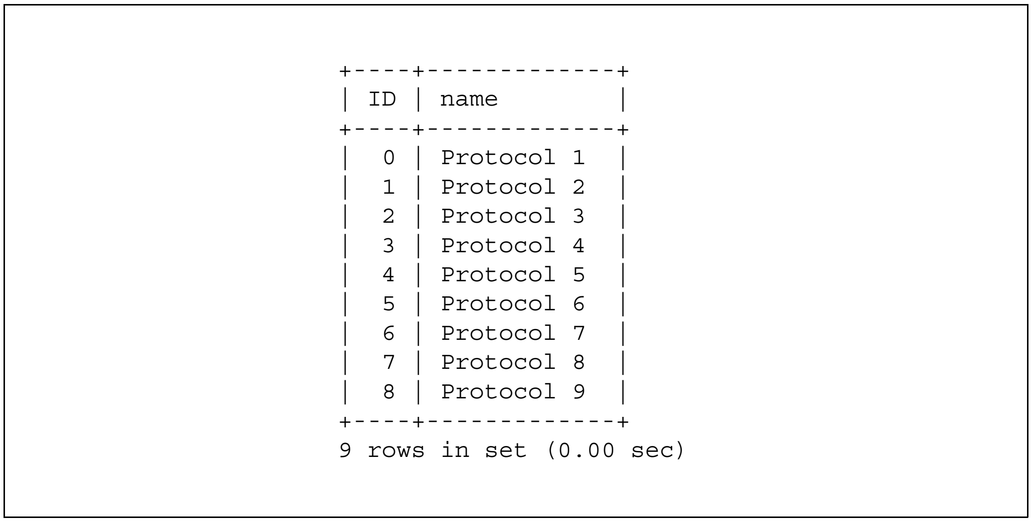
6. In the example in step 5, we used an equality test to filter the protocols. Table 9.2.6 lists the available comparison operators and the type of data they work on. Given the number of operators, it is easy to see that a `WHERE` clause can become arbitrarily complex, making it possible to write practically any conceivable search condition. For example, one might want to list only the forward primers that use protocol 2:

   ```
   mysql> SELECT ID, sequence, protocol FROM oligo WHERE
   protocol = 2 AND direction = ’Forward’;
   ```



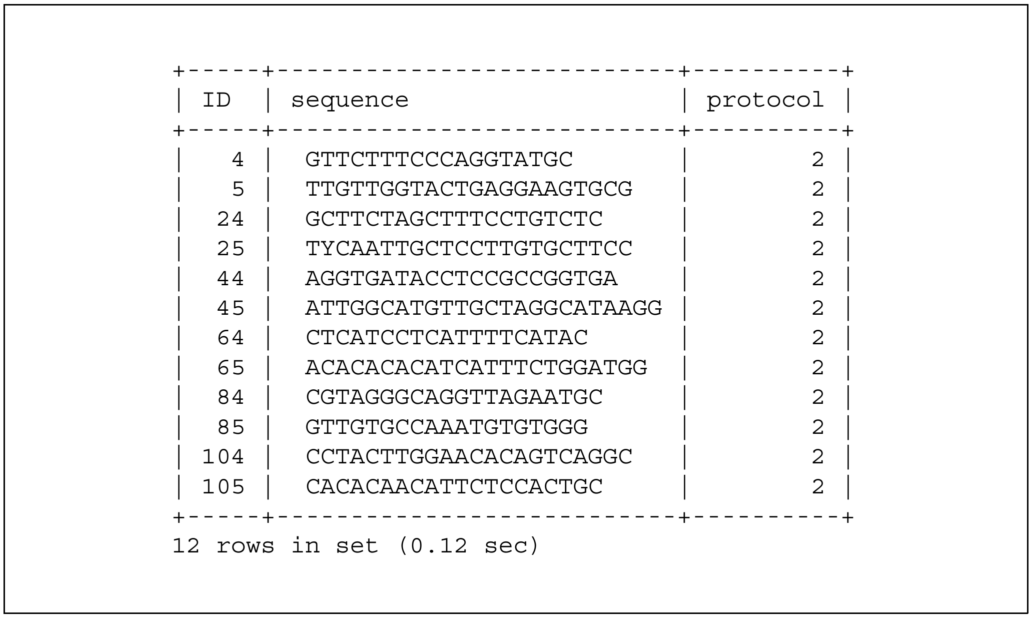
```
+----+----------+------------------------------------------------------------------------------------------------------+
| ID | name     | recipe                                                                                               |
+----+----------+------------------------------------------------------------------------------------------------------+
|  0 | Buffer A |  MgCl2: 1.5 mM KCl: 100 mM Tris-HCl: 10 mM NH4Cl: 5 mM pH: 8.6                                       |
|  1 | Buffer B |  MgCl2: 2.5 mM KCl: 50 mM Tris-HCl: 10 mM pH: 8.3                                                    |
|  2 | Buffer C |  MgCl2: 1.5 mM KCl: 50 mM Tris-HCl: 10 mM pH: 8.3 Enhancer: 0.04 units/ul (Stratagene Perfect Match) |
|  3 | Buffer D |  MgCl2: 1.5 mM KCl: 50 mM Tris-HCl: 10 mM pH: 8.3 Enhancer: 0.04 units/ul (Stratagene Perfect Match) |
|  4 | Buffer E |  MgCl2: 1.5 mM KCl: 100 mM Tris-HCl: 10 mM NH4Cl: 5 mM pH: 8.6                                       |
|  5 | Buffer F |  MgCl2: 1.5mM KCl:50mM Tris-HCl:10mM Tetrathlammoniumchloride(TMAC):1mM pH:8.3                       |
+----+----------+------------------------------------------------------------------------------------------------------+
6 rows in set (0.00 sec)
```

Figure 9.2.6 Output obtained upon submitting the command in Basic Protocol 5, step 3.



```
+----+-------------+
| ID | name        |
+----+-------------+
|  0 | Protocol 1  |
|  1 | Protocol 2  |
|  2 | Protocol 3  |
|  3 | Protocol 4  |
|  4 | Protocol 5  |
|  5 | Protocol 6  |
|  6 | Protocol 7  |
|  7 | Protocol 8  |
|  8 | Protocol 9  |
+----+-------------+
9 rows in set (0.00 sec)
```

Figure 9.2.7 Output obtained upon submitting the command with the WHERE clause (see Basic Protocol 5, step 4).



```
+-----+---------------------------+----------+
| ID  | sequence                  | protocol |
+-----+---------------------------+----------+
|   4 |  GTTCTTTCCCAGGTATGC       |        2 |
|   5 |  TTGTTGGTACTGAGGAAGTGCG   |        2 |
|  24 |  GCTTCTAGCTTTCCTGTCTC     |        2 |
|  25 |  TYCAATTGCTCCTTGTGCTTCC   |        2 |
|  44 |  AGGTGATACCTCCGCCGGTGA    |        2 |
|  45 |  ATTGGCATGTTGCTAGGCATAAGG |        2 |
|  64 |  CTCATCCTCATTTTCATAC      |        2 |
|  65 |  ACACACACATCATTTCTGGATGG  |        2 |
|  84 |  CGTAGGGCAGGTTAGAATGC     |        2 |
|  85 |  GTTGTGCCAAATGTGTGGG      |        2 |
| 104 |  CCTACTTGGAACACAGTCAGGC   |        2 |
| 105 |  CACACAACATTCTCCACTGC     |        2 |
+-----+---------------------------+----------+
12 rows in set (0.12 sec)
```

Figure 9.2.8 Output obtained upon submitting the command in Basic Protocol 5, step 5.

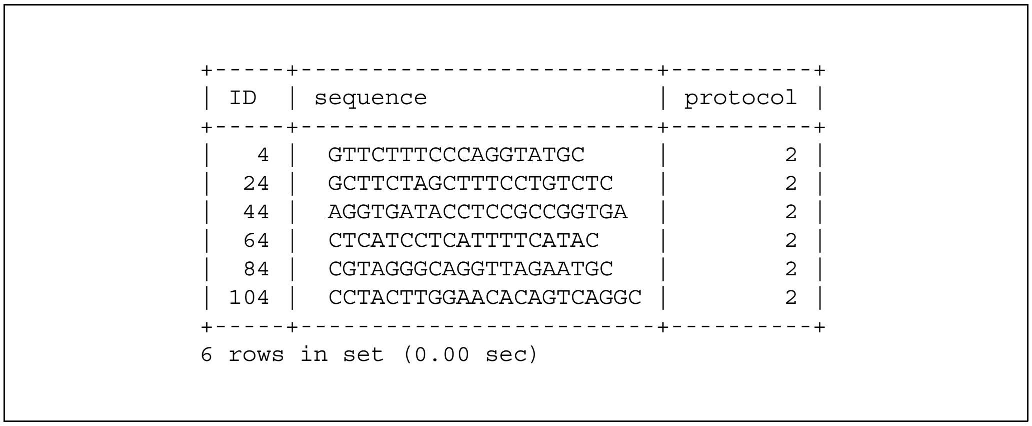
Table 9.2.6 MySQL Comparison Operators[a]

Class	Operator	Description
Grouping	( )	Groups a complex expression
Unary	+	Positive numeric value
	-	Negative numeric value
	~	Bitwise complement of number
Mathematical	*	Multiplication
	/	Division
Arithmetic	+	Addition
	-	Subtraction
Comparison	=	Equal to
	>	Greater than
	<	Less than
	>=	Greater or equal to
	<=	Less than or equal to
	<>	Not equal to
	!=	Not equal to
	!>	Not greater than
	!<	Not less than
Bitwise	|	Bitwise OR
	&	Bitwise AND
	<<	Shift left
	>>	Shift right
	~	Invert bits
Logical	NOT	Inverts meaning
	AND	Logical AND
	BETWEEN	Value within specified range
	IN	Value in list
	LIKE	Pattern matching
	OR	Logical OR
Assignment	=	Place value on right into variable

[a] Operator groups are listed by order of precedence. Operator precedence within a group varies, either following the order listed or having left-to-right precedence.

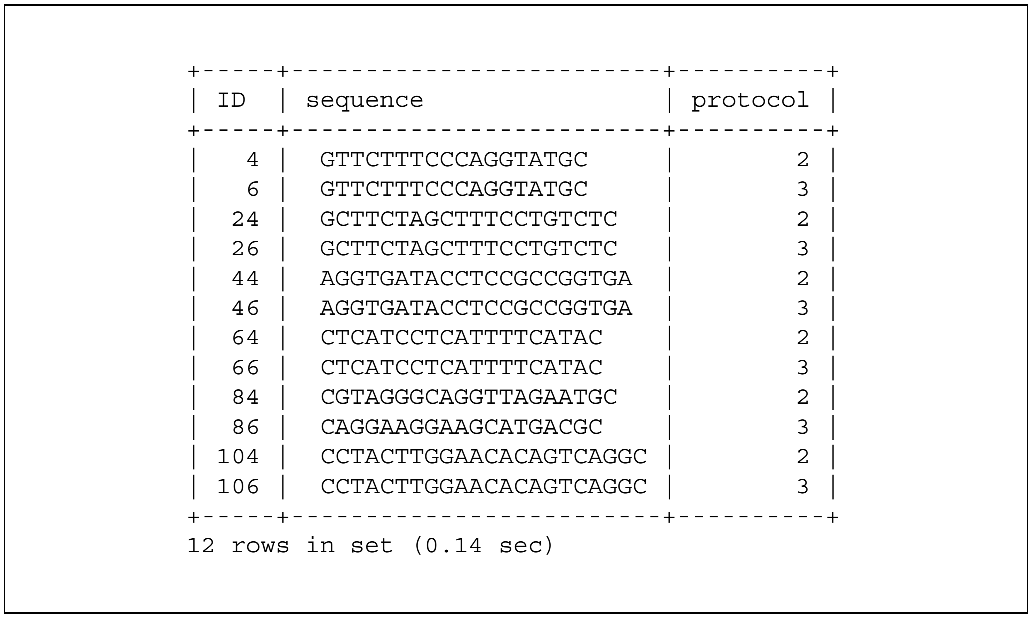
The output that will be displayed upon submitting this command is shown in Figure 9.2.9. This query is identical to the one in step 5, except it has an added constraint with the AND operator. Not surprisingly, now half as many primers are returned as before (since the experiments employ primer pairs).

7. As selection clauses get more complex, they need to have some rules of precedence to remove ambiguities. The rules of precedence are similar to those learned in algebra to understand that the equation 5 + 3 × 4 should be equal to 17 rather than 32. The operators in Table 9.2.6 are grouped into order of operation, from highest to lowest. As in algebra, it is possible to use parentheses to group operands and alter the order of execution. For example, to return the forward primers for either protocol 2 or protocol 3, write:



```
+-----+------------------------+----------+
| ID  | sequence               | protocol |
+-----+------------------------+----------+
|   4 |  GTTCTTTCCCAGGTATGC    |        2 |
|  24 |  GCTTCTAGCTTTCCTGTCTC  |        2 |
|  44 |  AGGTGATACCTCCGCCGGTGA |        2 |
|  64 |  CTCATCCTCATTTTCATAC   |        2 |
|  84 |  CGTAGGGCAGGTTAGAATGC  |        2 |
| 104 |  CCTACTTGGAACACAGTCAGGC |       2 |
+-----+------------------------+----------+
6 rows in set (0.00 sec)
```

Figure 9.2.9 Output obtained upon submitting the command in Basic Protocol 5, step 6.



```
+-----+------------------------+----------+
| ID  | sequence               | protocol |
+-----+------------------------+----------+
|   4 | GTTCTTTCCCAGGTATGC     |        2 |
|   6 | GTTCTTTCCCAGGTATGC     |        3 |
|  24 | GCTTCTAGCTTTCCTGTCTC   |        2 |
|  26 | GCTTCTAGCTTTCCTGTCTC   |        3 |
|  44 | AGGTGATACCTCCGCCGGTGA  |        2 |
|  46 | AGGTGATACCTCCGCCGGTGA  |        3 |
|  64 | CTCATCCTCATTTTCATAC    |        2 |
|  66 | CTCATCCTCATTTTCATAC    |        3 |
|  84 | CGTAGGGCAGGTTAGAATGC   |        2 |
|  86 | CAGGAAGGAAGCATGACGC    |        3 |
| 104 | CCTACTTGGAACACAGTCAGGC |        2 |
| 106 | CCTACTTGGAACACAGTCAGGC |        3 |
+-----+------------------------+----------+
12 rows in set (0.14 sec)
```

Figure 9.2.10 Output obtained upon submitting the command in Basic Protocol 5, step 7.

```
mysql> SELECT ID, sequence, protocol FROM oligo WHERE
(protocol = 2 OR protocol = 3) AND direction = 'Forward';
```

The output that would be displayed upon submitting this command is shown in Figure 9.2.10.

Filtering text data

8. Filtering text data is a little trickier, but is possible using the LIKE operator. LIKE searches the entry in a column for a pattern. The pattern is created using characters from the pattern and wildcard symbols that stand for any character(s).

The most common wildcard is the percent sign (%*), which stands in for any character or set of characters (including no characters). Thus, the pattern* %jam% *would match* Curt Jamison *(an author),* James Doohan *(an actor), and* strawberry jam *(a sugary fruit preserve). Note that the* LIKE *operator is case-insensitive, i.e.,* J *is the same as* j *when matching.*

Another useful wildcard operator is the pair of square brackets, [*and*]*, which denote a set or range of characters. For example, the pattern* 'jam[ie]son' *would match both* Jamison *(a biologist) or* Jameson *(a fine Scotch whiskey). The square brackets wildcard* *is more restrictive than the underscore wildcard, which stands for any possible character:* `'jam_son'` *would find both the* `i` *and the* `e` *variants, as well as any other possible permutations of single characters like* `Jamoson`, `Jamyson`, *and* `Jam8son`.

One can use the `LIKE` operator to filter rows based upon text fields. For example, one might want to know which of our buffers use Stratagene products:

```
mysql> SELECT name, recipe FROM buffer WHERE recipe LIKE
'%stratagene%';
```

Examination of the results shows that both records returned have a Stratagene product in the recipe.

Joining tables

9. The single most powerful aspect of SQL is the ability to draw information from multiple tables in a process called joining. The tables to be joined must have a column in common that links the two tables. For example, the `protocol` table has a column called `buffer` that contains the `ID` value of an entry in the `buffer` table. This is a common column that allows joining the two tables and creating a report that tells which buffers are used by which `protocol`:

```
mysql> SELECT protocol.name, buffer.name FROM protocol,
buffer WHERE protocol.buffer = buffer.ID;
```

The output displayed upon submitting this command is shown in Figure 9.2.11. The first thing to note is that the query uses the fully specified column names. A fully qualified table name is simply the table name and the column name concatenated together with a period (e.g., `protocol.name` *and* `buffer.name`*). The second thing to note is that both tables are specified in the* `FROM` *clause. Finally, the* `WHERE` *clause relates the* `protocol.buffer` *column to the* `buffer.ID` *column. The result is a list of* `protocol` *names and their associated buffer names.*

The database has several table-linking columns. In Figure 9.2.1, arrows are drawn from the name of a linking column to the corresponding linking column in another table. In addition to the linking columns between the `protocol` *and the* `buffer` *tables, there are linking columns between* `oligo` *and* `protocol`*, as well as between* `oligo` *and* `sequence` *tables.*

10. Often, using fully qualified names can become tedious, especially when the table names are long. To simplify the query, assign an alias to a table or column name using the `AS` statement. For example, abbreviate the table names in the previous query:

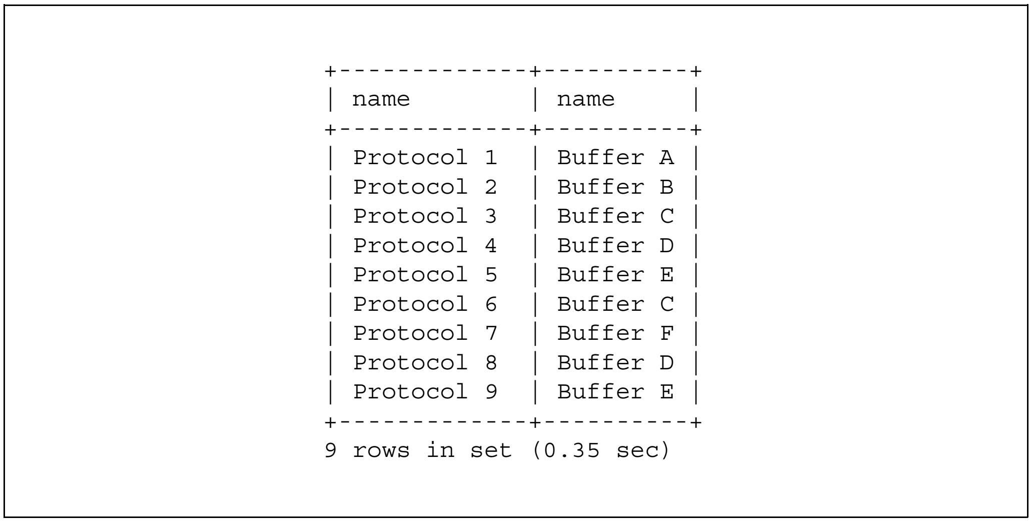
```
mysql> SELECT p.name, b.name FROM protocol AS p, buffer
AS b WHERE p.buffer = b.ID;
```

The `AS` *statement aliases* `p` *to* `protocol` *and* `b` *to* `buffer`*. The results returned by this query are identical to the query in step 9.*

11. Join across multiple tables. For example, to generate a list of sequences and the assay conditions associated with their STS primers:

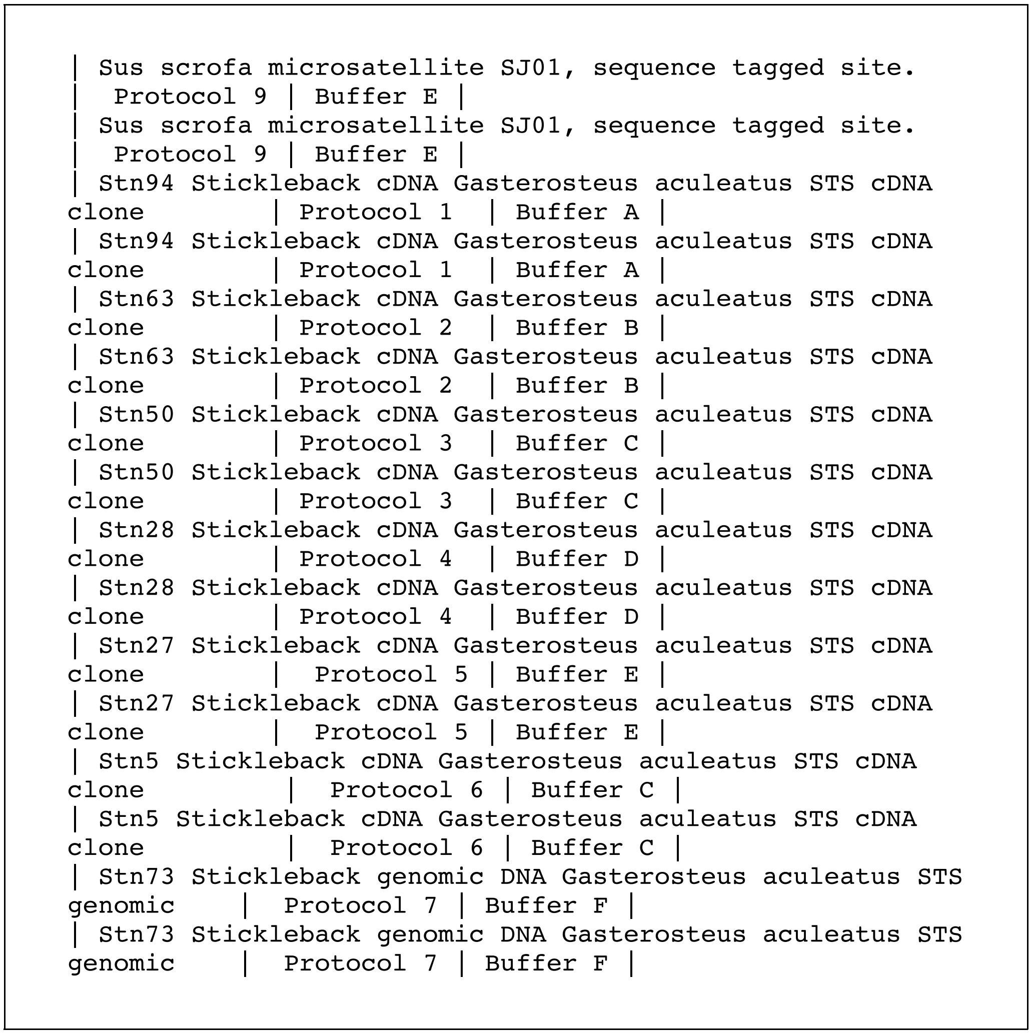
```
mysql> SELECT s.name, p.name, b.name FROM sequence AS s,
protocol AS p, buffer AS b, oligo AS o WHERE
o.derived_from = s.GBID AND o.protocol = p.ID AND
p.buffer = b.ID;
```

The result of this query lists the sequence names, `protocols`, and buffers for all 112 oligos. Note that even though the query does not output a column from the `oligo` table, the `oligo` table still needs to be included in the `FROM` clause because it is used in the `WHERE` clause.



```
+-------------+----------+
| name        | name     |
+-------------+----------+
| Protocol 1  | Buffer A |
| Protocol 2  | Buffer B |
| Protocol 3  | Buffer C |
| Protocol 4  | Buffer D |
| Protocol 5  | Buffer E |
| Protocol 6  | Buffer C |
| Protocol 7  | Buffer F |
| Protocol 8  | Buffer D |
| Protocol 9  | Buffer E |
+-------------+----------+
9 rows in set (0.35 sec)
```

Figure 9.2.11 Output obtained upon submitting the command in Basic Protocol 5, step 9.



```
| Sus scrofa microsatellite SJ01, sequence tagged site.
|  Protocol 9 | Buffer E |
| Sus scrofa microsatellite SJ01, sequence tagged site.
|  Protocol 9 | Buffer E |
| Stn94 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone        | Protocol 1  | Buffer A |
| Stn94 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone        | Protocol 1  | Buffer A |
| Stn63 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone        | Protocol 2  | Buffer B |
| Stn63 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone        | Protocol 2  | Buffer B |
| Stn50 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone        | Protocol 3  | Buffer C |
| Stn50 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone        | Protocol 3  | Buffer C |
| Stn28 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone        | Protocol 4  | Buffer D |
| Stn28 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone        | Protocol 4  | Buffer D |
| Stn27 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone        |  Protocol 5 | Buffer E |
| Stn27 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone        |  Protocol 5 | Buffer E |
| Stn5 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone         |  Protocol 6 | Buffer C |
| Stn5 Stickleback cDNA Gasterosteus aculeatus STS cDNA
clone         |  Protocol 6 | Buffer C |
| Stn73 Stickleback genomic DNA Gasterosteus aculeatus STS
genomic    |  Protocol 7 | Buffer F |
| Stn73 Stickleback genomic DNA Gasterosteus aculeatus STS
genomic    |  Protocol 7 | Buffer F |
```

Figure 9.2.12 Output obtained upon submitting the command in Basic Protocol 5, step 12.

Manipulating the output

12. Review the output from Figure 9.2.12, note that there are multiple entries for each sequence. In fact, there are two, because the forward and reverse primers are entered in the `oligo` table. The net effect is that there are two of each row in the answer. If one adds the `DISTINCT` keyword to the `SELECT` command, duplicate rows *within* the answer are filtered out. Thus, the modified query:

```
mysql> SELECT DISTINCT s.name, p.name, b.name FROM
sequence AS s, protocol AS p, buffer AS b, oligo AS o
WHERE o.derived_from = s.GBID AND o.protocol = p.ID AND
p.buffer = b.ID;
```

    returns half the number of rows (56), since it filters out the duplicates.

    Use the up arrow to recall a previous command, and the left and right arrows to move to the appropriate place to insert or delete changes to the query.

13. The order in which the rows are returned is based upon the order in which the data are placed in the table. Since this order is typically meaningless, SQL allows sorting and grouping the data in different ways using the `ORDER BY` clause. The order can be either ascending (`ASC`) or descending (`DESC`), and there can be multiple sort orders within the query. For example, edit the previous query to return only sequences using buffer E, sorted by `protocol` and then sequence name:

```
mysql> SELECT DISTINCT s.name, p.name, b.name FROM
sequence AS s, protocol AS p, buffer AS b, oligo AS o
WHERE o.derived_from = s.GBID AND o.protocol = p.ID AND
p.buffer = b.ID AND b.name = 'Buffer E' ORDER BY p.name
ASC, s.name ASC;
```

    Examination of the resulting list of twelve sequences should show the first six sequences using protocol 5 and sorted from `Stn27` *to* `sWSS1280`*, and the second group of six using protocol 9 and sorted from* `Bos Taurus` *to* `sWSS 1139`.

14. It is often useful to retrieve data in aggregate. An aggregate function combines rows into a summary statistic (like averages or counts). For example, to know how many sequences used buffer E, count the instances in the column list:

```
mysql> SELECT p.name, b.name, COUNT(s.name) AS 'Seq #'
FROM sequence AS s, protocol AS p, buffer AS b, oligo AS
o WHERE o.derived_from = s.GBID AND o.protocol = p.ID
AND p.buffer = b.ID AND b.name = 'Buffer E' GROUP BY
p.name, b.name;
```

    The output displayed upon submitting this command is shown in Figure 9.2.13. The table is arranged by the non-aggregated columns in the `GROUP BY` *clause.*

15. Note that by aggregating the sequences with the `COUNT`, the user loses the ability to `SELECT DISTINCT` based upon `[sequence.name]` (since it is no longer in the result table), and again there are twice as many sequences as necessary (compared to the table in step 13). In this case, go back and add a restriction to the `WHERE` clause to look at only the `Forward` primers:

```
mysql> SELECT p.name, b.name, COUNT(s.name) AS 'Seq #'
FROM sequence AS s, protocol AS p, buffer AS b, oligo AS
o WHERE o.derived_from = s.GBID AND o.protocol = p.ID
AND p.buffer = b.ID AND b.name = 'Buffer E' AND
o.direction = 'Forward' GROUP BY p.name, b.name;
```

    The output displayed upon submitting this command is shown in Figure 9.2.14.

```
+-------------+---------+-------+
| name        | name    | Seq # |
+-------------+---------+-------+
|  Protocol 5 | Buffer E |    12 |
|  Protocol 9 | Buffer E |    12 |
+-------------+---------+-------+
2 rows in set (0.16 sec)
```

Figure 9.2.13 Output obtained upon submitting the command in Basic Protocol 5, step 14.

```
+-------------+---------+-------+
| name        | name    | Seq # |
+-------------+---------+-------+
|  Protocol 5 | Buffer E |     6 |
|  Protocol 9 | Buffer E |     6 |
+-------------+---------+-------+
2 rows in set (0.01 sec)
```

Figure 9.2.14 Output obtained upon submitting the command in Basic Protocol 5, step 15.

```
mysql> SELECT COUNT(*) FROM sequence AS s, protocol AS p, buffer AS b,
oligo AS o WHERE o.derived_from = s.GBID AND o.protocol = p.ID AND
p.buffer = b.ID;
+----------+
| COUNT(*) |
+----------+
|      112 |
+----------+
1 row in set (0.00 sec)
```

Figure 9.2.15 Query and output obtained from Basic Protocol 5, step 16.

16. Aggregates can be useful in other ways as well. By using the COUNT function in place of column names, one can find the number of items in any particular table:

```
mysql> SELECT COUNT(*) FROM oligo;
```

reports that there are 124 items in the oligo table. It is also possible to estimate how many lines will be returned by a particular query. For example, count how many lines will be returned from the query used in Step 11 (see Fig. 9.2.15).

ALTERNATE PROTOCOL 2

BATCH PROCESSING COMMAND SCRIPTS

SQL queries can be arbitrarily complex, and it often seems the more useful a query is, the more complex it gets. Typing a complex query into the MySQL command line can be a frustrating experience, especially if you are running a query on a regular basis. Fortunately, MySQL provides a mechanism for reading SQL queries and commands from a file and executing them in a batch mode.

Necessary Resources

Hardware

Computer capable of running MySQL

Software

Working installation of MySQL version 3.22.11 or higher.

Files

The `primers` database created in Basic Protocol 1, as altered in Support Protocol 1, and loaded with the larger search data set as described in Basic Protocol 5, step 2

`Script.txt` file, which contains a set of four SQL commands. This file is available from the Current Protocols Web site: *http://www3.interscience.wiley.com/c_p/cpbi_sampledatafiles.htm*.

1. Use your favorite text editor (*APPENDIX 1C*) and examine the `script.txt` file. This file contains four SQL commands:

```
USE primers;
SELECT count(*) FROM oligo;
SELECT ID, name FROM protocol;
SELECT ID, sequence, protocol FROM oligo WHERE protocol
= 2;
```

The batch processing utility in MySQL will execute these four lines in order, as if they had been typed in at the command line. The first line makes sure one uses the proper database. The next three lines are increasingly complex SQL queries that were run by hand in Basic Protocol 5.

Running a script from inside the MySQL program

2. Login to the MySQL database server:

```
transposon:cjamison% mysql -u cjamison -p
Enter password:
```

Note it is not necessary to specify a particular database, since the first line of the script does that.

3. Use the `SOURCE` command to run the script as in Figure 9.2.16.

Each command generates a normal output. The `USE` *command generated a database-changed message, and each query produced an output table. The actual commands are not echoed.*

4. Quit the MySQL program:

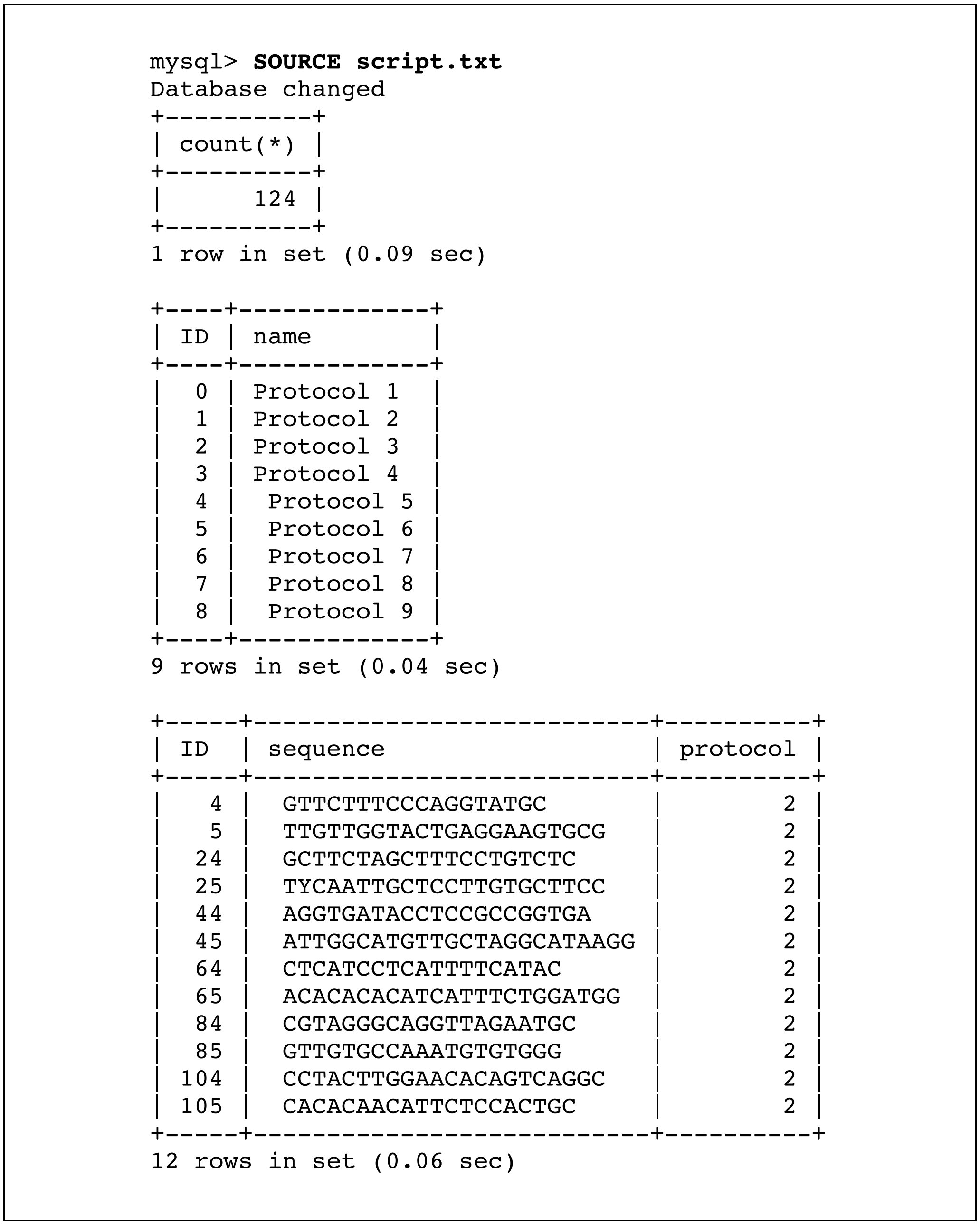
```
mysql> exit
Bye
```

Running a script from the Unix command line

5. A MySQL script can be executed directly from the command line using the input redirect as in Figure 9.2.17.

The MySQL program reads the file and outputs the answers directly to the screen. Note that the ASCII boxes around the tables and the query statistics are now gone. All extraneous text has been removed, and the results are in a tab-delimited format suitable for copying and pasting into other analysis programs.

6. Use the output redirect to put the results of the query into a file:



```
mysql> SOURCE script.txt
Database changed
+----------+
| count(*) |
+----------+
|      124 |
+----------+
1 row in set (0.09 sec)

+----+-------------+
| ID | name        |
+----+-------------+
|  0 | Protocol 1  |
|  1 | Protocol 2  |
|  2 | Protocol 3  |
|  3 | Protocol 4  |
|  4 |  Protocol 5 |
|  5 |  Protocol 6 |
|  6 |  Protocol 7 |
|  7 |  Protocol 8 |
|  8 |  Protocol 9 |
+----+-------------+
9 rows in set (0.04 sec)

+-----+---------------------------+----------+
| ID  | sequence                  | protocol |
+-----+---------------------------+----------+
|   4 |  GTTCTTTCCCAGGTATGC       |        2 |
|   5 |  TTGTTGGTACTGAGGAAGTGCG   |        2 |
|  24 |  GCTTCTAGCTTTCCTGTCTC     |        2 |
|  25 |  TYCAATTGCTCCTTGTGCTTCC   |        2 |
|  44 |  AGGTGATACCTCCGCCGGTGA    |        2 |
|  45 |  ATTGGCATGTTGCTAGGCATAAGG |        2 |
|  64 |  CTCATCCTCATTTTCATAC      |        2 |
|  65 |  ACACACACATCATTTCTGGATGG  |        2 |
|  84 |  CGTAGGGCAGGTTAGAATGC     |        2 |
|  85 |  GTTGTGCCAAATGTGTGGG      |        2 |
| 104 |  CCTACTTGGAACACAGTCAGGC   |        2 |
| 105 |  CACACAACATTCTCCACTGC     |        2 |
+-----+---------------------------+----------+
12 rows in set (0.06 sec)
```

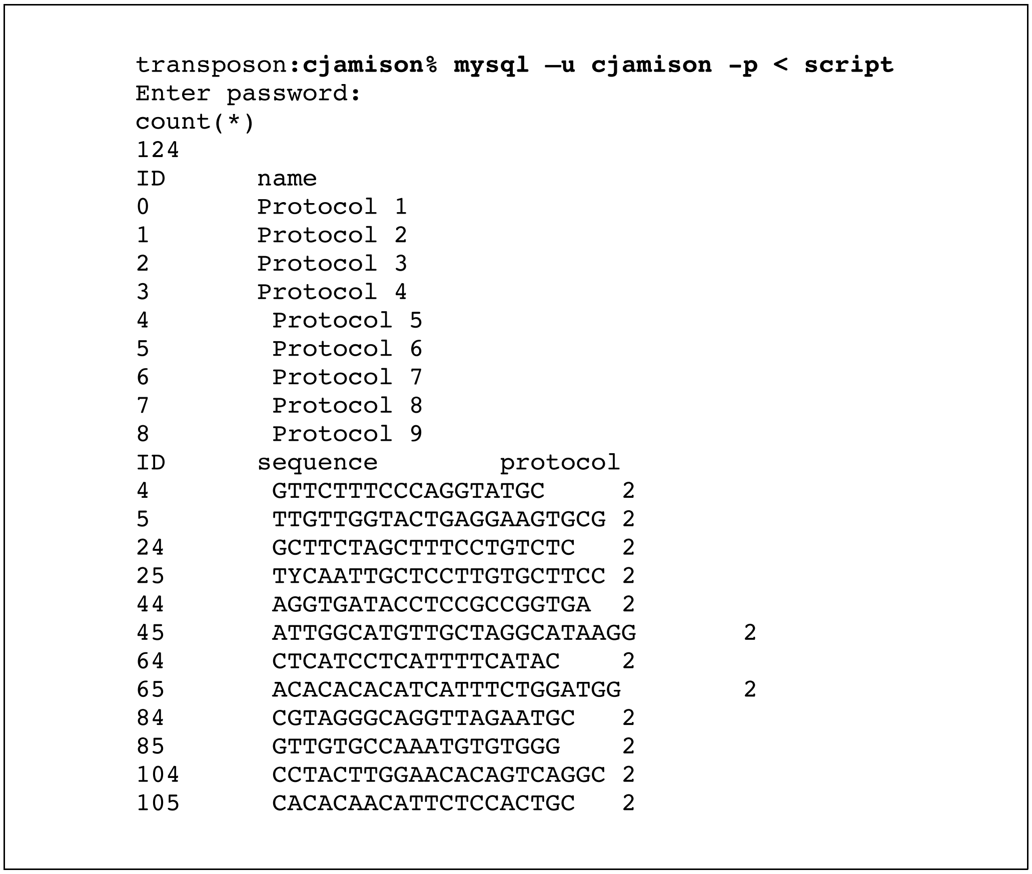
Figure 9.2.16 SOURCE command output from running script in Alternate Protocol 2, step 3.

```
transposon:cjamison% mysql -p < script.txt > output.txt
Enter password:
```

The output script is now in the directory:

```
transposon:cjamison% ls -la
total 176
drwxr-xr-x 12 cjamison 364 May 13 09:55 .
drwxrwxrwx 12 cjamison 364 May 11 15:55 ..
-rw-r—r— 1 cjamison 498 May 13 09:55 output.txt ...
```

and contains exactly the same output as went to the screen. The difference is that the output is bundled into a nice neat file for importing into other programs.



```
transposon:cjamison% mysql —u cjamison -p < script
Enter password:
count(*)
124
ID      name
0       Protocol 1
1       Protocol 2
2       Protocol 3
3       Protocol 4
4        Protocol 5
5        Protocol 6
6        Protocol 7
7        Protocol 8
8        Protocol 9
ID      sequence        protocol
4        GTTCTTTCCCAGGTATGC      2
5        TTGTTGGTACTGAGGAAGTGCG  2
24       GCTTCTAGCTTTCCTGTCTC    2
25       TYCAATTGCTCCTTGTGCTTCC  2
44       AGGTGATACCTCCGCCGGTGA   2
45       ATTGGCATGTTGCTAGGCATAAGG        2
64       CTCATCCTCATTTTCATAC     2
65       ACACACACATCATTTCTGGATGG         2
84       CGTAGGGCAGGTTAGAATGC    2
85       GTTGTGCCAAATGTGTGGG     2
104      CCTACTTGGAACACAGTCAGGC  2
105      CACACAACATTCTCCACTGC    2
```

Figure 9.2.17 Input redirect output from Alternate Protocol 2, step 5.

7. Sometimes it is useful to actually have the SQL commands echoed into the output. To do this, use the `-vvv` switch, which makes the MySQL program act as if the user had logged in and issued the `SOURCE` command, depicted in Figure 9.2.18.

   The SQL query commands are now echoed, and the ASCII line art surrounding the result tables is back. Note also the `Bye` *on the last line of the output, which resulted from the implicit exit command that terminated every batch session.*

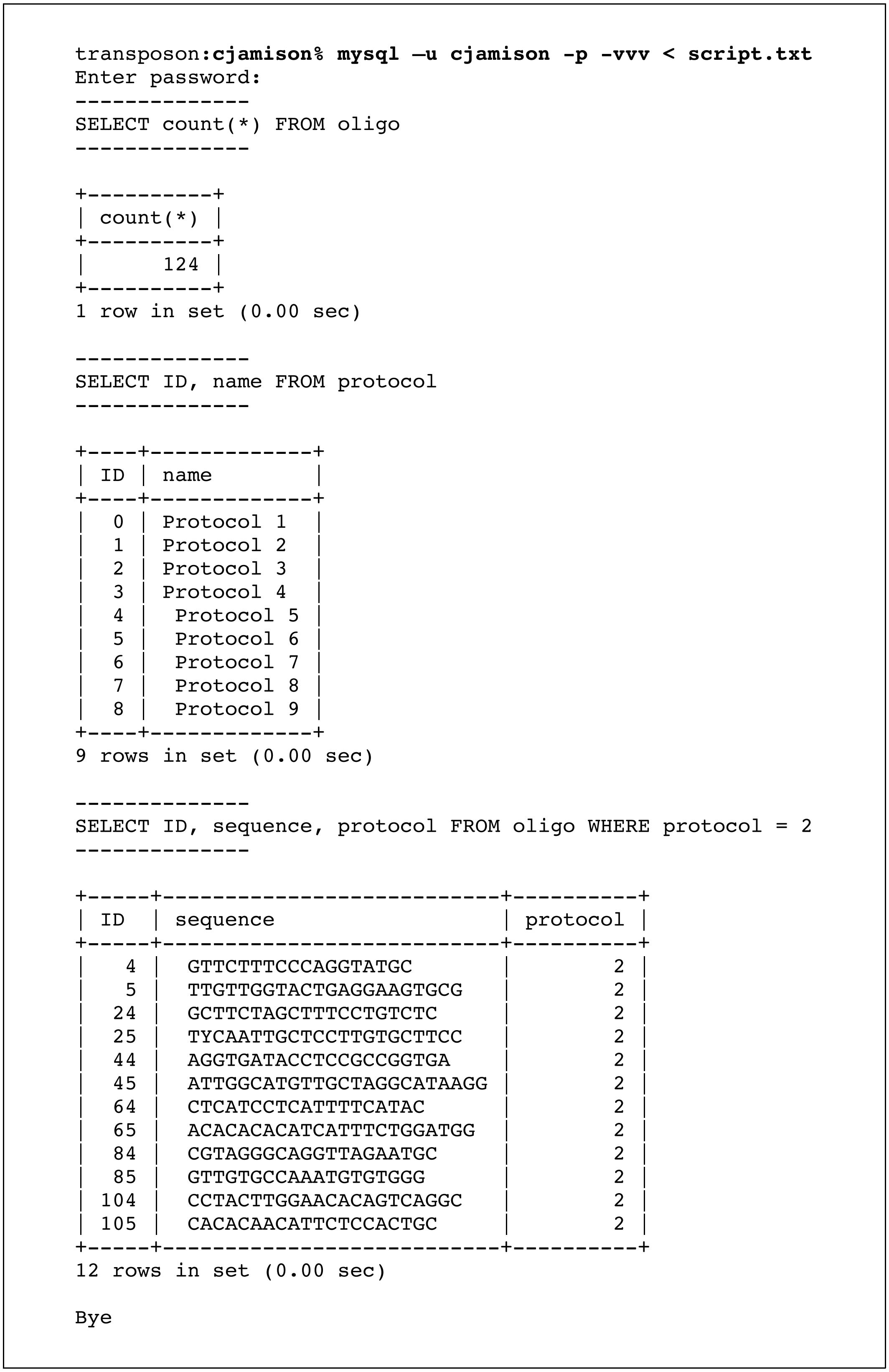
   The most common use of the batch session utility is to import database setup scripts. For example, all the SQL commands used to create the `primers` *database in Basic Protocol 1 and to load the data in Basic Protocol 5 could have been placed into a single text file to be used as the input to the batch mode. Since most biological databases are orders of magnitude more complex than the simple one used as example here, the batch mode comes in quite handy.*

COMMENTARY

Background Information

MySQL (favored pronunciation “my-es-queue-ell”) is a powerful relational database system. It is available for free under the GNU Public License, and runs on Windows, OS/2, Linux, Sun Solaris, and a wide variety of other Unix-based operating systems (e.g., the code examples in this unit were tested using MySQL installed on an Apple Macintosh G4 laptop running MacOS X). While MySQL does not have all the powerful features one might find in a commercial RDMS, databases created using MySQL are nearly as powerful and certainly as fast as any other RDMS.

While MySQL was used as the platform for this chapter, most of the example SQL is valid for any relational database system that implements the ANSI SQL99 standard (with the exception of the `enum` type, which is specific to MySQL). This includes databases such as Oracle, Sybase, Microsoft SQL Server, and Postgres. Each of these systems implements the



```
transposon:cjamison% mysql —u cjamison -p -vvv < script.txt
Enter password:
--------------
SELECT count(*) FROM oligo
--------------

+----------+
| count(*) |
+----------+
|      124 |
+----------+
1 row in set (0.00 sec)

--------------
SELECT ID, name FROM protocol
--------------

+----+-------------+
| ID | name        |
+----+-------------+
|  0 | Protocol 1  |
|  1 | Protocol 2  |
|  2 | Protocol 3  |
|  3 | Protocol 4  |
|  4 |  Protocol 5 |
|  5 |  Protocol 6 |
|  6 |  Protocol 7 |
|  7 |  Protocol 8 |
|  8 |  Protocol 9 |
+----+-------------+
9 rows in set (0.00 sec)

--------------
SELECT ID, sequence, protocol FROM oligo WHERE protocol = 2
--------------

+-----+--------------------------+----------+
| ID  | sequence                 | protocol |
+-----+--------------------------+----------+
|   4 |  GTTCTTTCCCAGGTATGC      |        2 |
|   5 |  TTGTTGGTACTGAGGAAGTGCG  |        2 |
|  24 |  GCTTCTAGCTTTCCTGTCTC    |        2 |
|  25 |  TYCAATTGCTCCTTGTGCTTCC  |        2 |
|  44 |  AGGTGATACCTCCGCCGGTGA   |        2 |
|  45 |  ATTGGCATGTTGCTAGGCATAAGG |        2 |
|  64 |  CTCATCCTCATTTTCATAC     |        2 |
|  65 |  ACACACACATCATTTCTGGATGG |        2 |
|  84 |  CGTAGGGCAGGTTAGAATGC    |        2 |
|  85 |  GTTGTGCCAAATGTGTGGG     |        2 |
| 104 |  CCTACTTGGAACACAGTCAGGC  |        2 |
| 105 |  CACACAACATTCTCCACTGC    |        2 |
+-----+--------------------------+----------+
12 rows in set (0.00 sec)

Bye
```

Figure 9.2.18 SQL commands echoed output using `-vvv` switch in Alternate Protocol 2, step 7.

standard to a varying degree, and each has a specific dialect and extension; however, the core language remains the same.

This unit has touched upon only a minimal subset of the SQL, just enough to get started creating databases and moving data in and out. Using these principles, the reader should be able to interact successfully with any SQL-compliant database and to generate relatively sophisticated queries. However, there are many more commands and nuances than are given here, especially in the realms of joins, views, and virtual tables.

An important SQL concept is that of a `NULL` value. Simply put, a `NULL` value is a value that is not there. Any datatype can have a `NULL` value. It is important to note that a `NULL` value is very different from a zero or a blank. A `NULL` cannot be used in a comparison or a calculation, since two `NULL` values are not equal to one another. However, the `NULL` value can be used in a Boolean expression test clause like 'protocol `IS NULL`' or 'sequence `IS NOT NULL`'. In cases where it is desirable to have an entry for every row, `NULL` values can be specifically disallowed for a column.

Critical Parameters and Troubleshooting

While SQL is not a programming language per se, interacting with the language often feels like programming. Specifically, the SQL interpreter will issue cryptic complaints about errors in the SQL query, and queries must often be fine-tuned to return the desired result. However, most command-line SQL interpreters only offer the most rudimentary interactive editing capabilities (e.g., MySQL only allows for cycling through the command list with the arrow keys). Many people who interact with SQL on a constant basis keep a text editor like emacs or vi open in which to compose their queries, then copy/paste the line into the SQL interpreter.

Key References

DuBois, P. 1999. MySQL. New Riders. Indianapolis, Ind.

A comprehensive guide, with many examples and tutorials. An excellent reference for beginners.

Gulutzan, P. and Pelzer, T. 1999. SQL-99 Complete, Really. CMP Books. Gilroy, Calif.

A complete description of the SQL99 standards.

Kline, K. and Kline, D. 2001. SQL in a Nutshell. O'Reilly and Associates. Sebastopol, Calif.

A compact yet comprehensive guide to SQL statements. Includes several different SQL dialects like MySQL and Oracle.

Internet Resources

http://www.mysql.com

The main MySQL site.

http://www.useractive.com

A hands-on tutorial.

Contributed by D. Curtis Jamison
George Mason University
Manassas, Virginia

Modeling Biology Using Relational Databases

Experimental data as well as information extracted from the various sequence and other bioinformatics databases can all be stored in a relational database allowing for easy querying and data exploration (*UNIT 9.1*). The information is stored in sets of tables; the layout of the tables, termed the schema, can be designed using one or more standard methods. Certainly, one could enter all of the data into a single table, but this would simply perpetuate the problems encountered using a standard spreadsheet: high data redundancy, missing data, and difficulty in arbitrarily mixing and grouping the data. The power of a relational database comes from nonredundantly sequestering pieces of information and then enabling queries that recombine them in many different ways.

There are several different methodologies that can be used for designing a database schema; no one is the best for all occasions. This unit demonstrates two different techniques for designing relational tables and discusses when each should be used. These two techniques are (1) traditional Entity-Relationship (E-R) modeling (as described in the Basic Protocol) and (2) a hybrid method that combines aspects of data warehousing and E-R modeling (described in the Alternate Protocol). The method of choice depends on (1) how well the information and all its inherent relationships are understood, (2) what types of questions will be asked, (3) how many different types of data will be included, and (4) how much data exists.

The naming scheme used in the following protocols and figures does not correspond to any particular database management system (DBMS) and is used for its readability. Refer to the DBMS's documentation for the characters allowed in table and column names. In particular, note that the MySQL DBMS used in other units does not allow spaces in table or column names. To create valid MySQL schemas from these examples, replace spaces with the underscore character _.

USING ENTITY-RELATIONSHIP MODELING TO DESIGN A DATABASE

BASIC PROTOCOL

This protocol describes the use of Entity-Relationship modeling to design a database and covers designing the table schema (outlined in the flowchart in Fig. 9.3.1), creating tables, and finally, querying the database to ask interesting biological questions. A typical problem in genomics is used as an example: storing results from gene expression studies. Figure 9.3.2 shows a very small portion of typical data obtained from gene chip studies looking for changes in gene expression important in diabetes. A number of different stimuli were used on samples from different tissues in different organisms and the fold expression differences were recorded. Even in this small subset, the data duplication is readily apparent as shown by the number of times the name, description, and ID of glucose-6-phosphatase appear. In addition, it is hard to obtain an answer to the question, "Which genes are up-regulated in liver in the presence of glucose and down-regulated in the presence of insulin?" In this example, each gene is described by a unique identifier, unique even across all of the organisms of interest. The relationships between the different general types of information (referred to as "entities") are well known since they are the experimental parameters and results: the genes are the probes on the chip, the organisms, tissue, and stimulus comprise the sample, and the expression levels are the experimental results. As discussed in the Commentary section below, the schema design method highlighted in this protocol, Entity-Relationship (E-R) modeling, works well when the relationships between entities are well known.

Contributed by Robert M. Peitzsch

Current Protocols in Bioinformatics (2003) 9.3.1-9.3.28

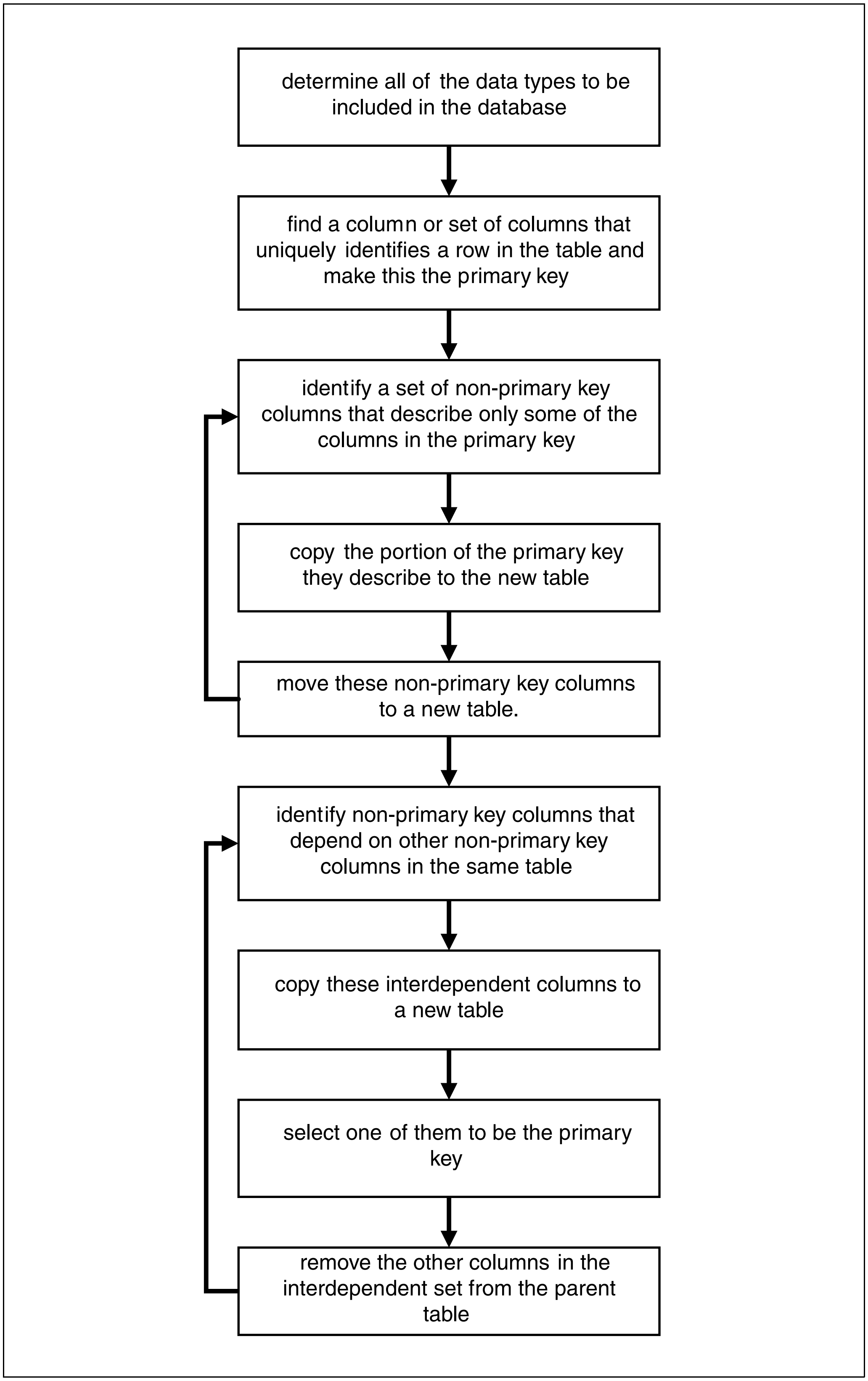
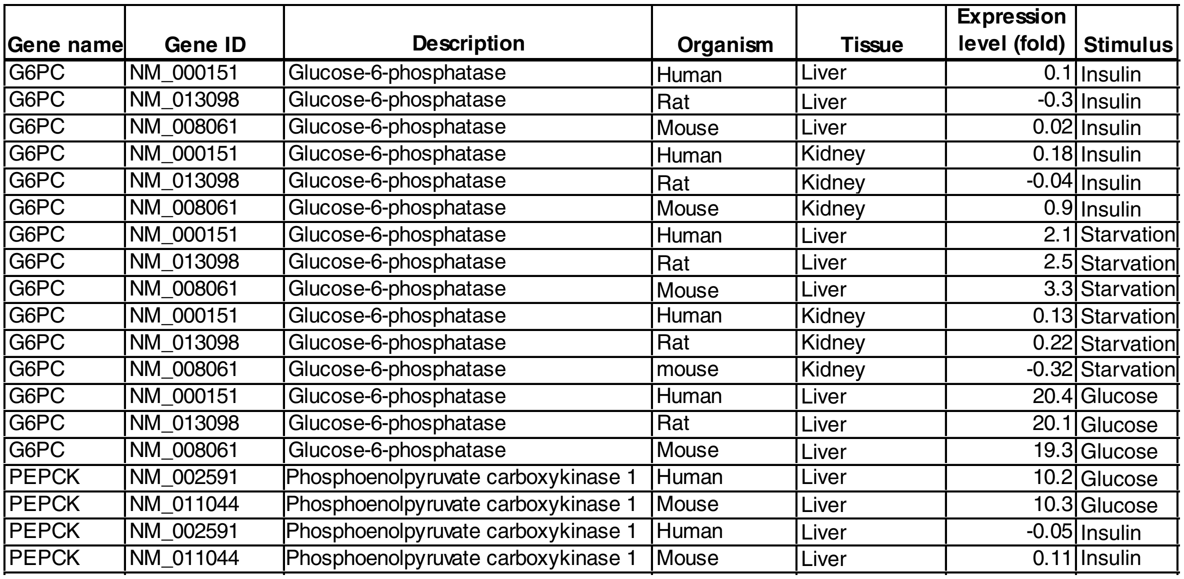

determine all of the data types to be included in the database
find a column or set of columns that uniquely identifies a row in the table and make this the primary key
identify a set of non-primary key columns that describe only some of the columns in the primary key
copy the portion of the primary key they describe to the new table
move these non-primary key columns to a new table.
identify non-primary key columns that depend on other non-primary key columns in the same table
copy these interdependent columns to a new table
select one of them to be the primary key
remove the other columns in the interdependent set from the parent table

Figure 9.3.1 Flowchart outlining steps for designing table schema in Entity-Relationship modeling, as described in the Basic Protocol.



Gene name	Gene ID	Description	Organism	Tissue	Expression level (fold)	Stimulus
G6PC	NM_000151	Glucose-6-phosphatase	Human	Liver	0.1	Insulin
G6PC	NM_013098	Glucose-6-phosphatase	Rat	Liver	-0.3	Insulin
G6PC	NM_008061	Glucose-6-phosphatase	Mouse	Liver	0.02	Insulin
G6PC	NM_000151	Glucose-6-phosphatase	Human	Kidney	0.18	Insulin
G6PC	NM_013098	Glucose-6-phosphatase	Rat	Kidney	-0.04	Insulin
G6PC	NM_008061	Glucose-6-phosphatase	Mouse	Kidney	0.9	Insulin
G6PC	NM_000151	Glucose-6-phosphatase	Human	Liver	2.1	Starvation
G6PC	NM_013098	Glucose-6-phosphatase	Rat	Liver	2.5	Starvation
G6PC	NM_008061	Glucose-6-phosphatase	Mouse	Liver	3.3	Starvation
G6PC	NM_000151	Glucose-6-phosphatase	Human	Kidney	0.13	Starvation
G6PC	NM_013098	Glucose-6-phosphatase	Rat	Kidney	0.22	Starvation
G6PC	NM_008061	Glucose-6-phosphatase	mouse	Kidney	-0.32	Starvation
G6PC	NM_000151	Glucose-6-phosphatase	Human	Liver	20.4	Glucose
G6PC	NM_013098	Glucose-6-phosphatase	Rat	Liver	20.1	Glucose
G6PC	NM_008061	Glucose-6-phosphatase	Mouse	Liver	19.3	Glucose
PEPCK	NM_002591	Phosphoenolpyruvate carboxykinase 1	Human	Liver	10.2	Glucose
PEPCK	NM_011044	Phosphoenolpyruvate carboxykinase 1	Mouse	Liver	10.3	Glucose
PEPCK	NM_002591	Phosphoenolpyruvate carboxykinase 1	Human	Liver	-0.05	Insulin
PEPCK	NM_011044	Phosphoenolpyruvate carboxykinase 1	Mouse	Liver	0.11	Insulin

Figure 9.3.2 A small portion of data typically obtained from gene expression studies. The full dataset contains the name, an identifier (accession number), and a description for every probe on the chip; the organism and tissue from which the sample was obtained; how the sample was treated; and the fold change in the gene expression versus the normal tissue. The identifier uniquely specifies a gene, even across organisms.

Necessary Resources

Hardware

Personal computers with the Microsoft Windows operating system

Software

There are several software packages for designing formal schemas, some examples are: Microsoft Access, Designer from Oracle, PowerDesigner from Sybase, and Visio also from Microsoft. Pencil and paper also work well, and are recommended for those who are just starting out.

Files

None

Designing the table schema

1. Place all of the different types of information in a single table as in Figure 9.3.3.

   The first step is to list all of the different types of information in a single table. This exercise is often done mentally or by simply listing the types of data on a piece of paper. Examples of types of data are `gene name`, `gene description`, `expression level`, *and* `tissue`. *In contrast,* `G6PC` *is a value for the data type* `gene name`.

   Each type of information should be as specific as possible. For instance, instead of having a column `expression` *to hold the expression level and the name of the tissue, there are two separate columns:* `expression level` *and* `tissue`.

2. Identify a column or a set of columns that will uniquely identify an entry in the table as in Figure 9.3.4. This is the primary key for this table.

   In this example, the values in the columns `gene id`, `tissue`, *and* `stimulus`, *together, uniquely identify a row in the table and are chosen to be the primary key. For example, the combination of values (*`NM_013098`, `Liver`, `Insulin`*) identifies the specific entry in*

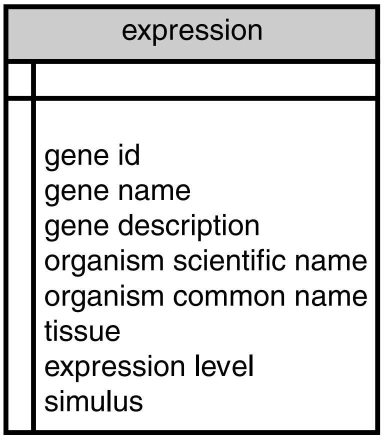


Figure 9.3.3 A single table is created containing all of the different types of information obtained from the gene expression study. Often, this is done as a mental exercise. In this and subsequent figures showing tables, the table name is shown in the gray area at the top of the box representing a table. In the area below the gray area are the names of the columns used in the table's primary key, these will have a PK preceding them. In some of the figures, this will be blank depending on the step in the protocol. The names of all other columns in the table will be found in the bottom portion. Columns involved in foreign keys are indicated by a FK# where the # is a number that differentiates between the different foreign keys. Note that the naming scheme used in these figures does not correspond to any database product. Please refer to individual database product documentation for the proper naming of tables and columns. It is standard practice to use the singular form of a word when naming tables and columns.

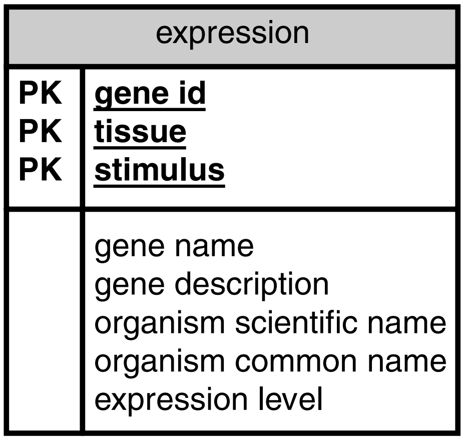


Figure 9.3.4 Together, the columns `gene id`, `tissue`, and `stimulus` uniquely identify a row in the table. Accordingly, they are designated the primary key as indicated by the PK to the left of the column names.

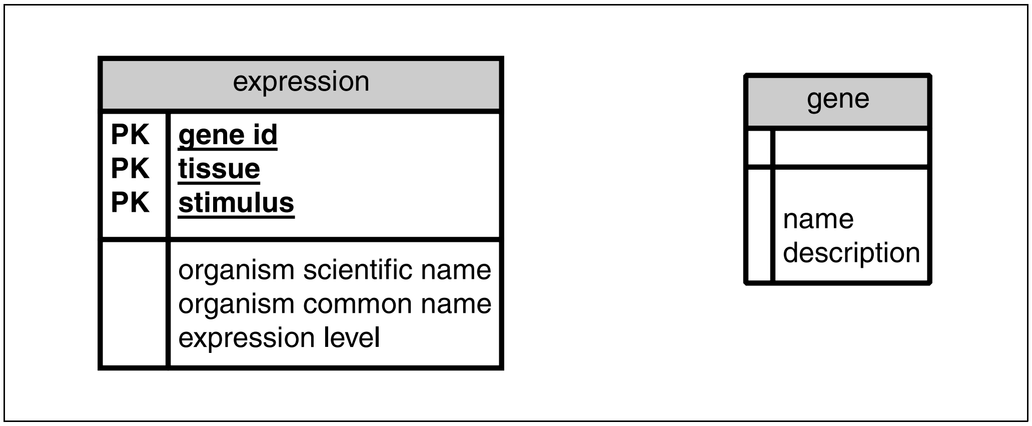


Figure 9.3.5 A related set of columns that are attributes of only part of the compound (multiple column) primary key have been removed from the original table and placed in a new table `gene`. The prefix `gene` has been dropped from the column names for brevity.

> *the table that contains that gene's expression level in that tissue with that stimulus. Since this example problem defines the* `gene id` *as being unique across all organisms, neither form of the organism name needs to be included in the primary key.*
>
> *It is possible that there is more than one column or sets of columns that can uniquely identify a row; pick one set to be the primary key.* `Expression level` *cannot be the used as a key since the values in that column are experimental results and do not aid in uniquely identifying a row.*

3. Identify a set of non-primary key columns that describe only a subset of the columns in the primary key.

   `Gene name` *and* `gene description` *describe a gene and hence describe only* `gene id` *and not the entire primary key. Neither form of organism name directly describes any specific part of the primary key and hence will remain in the* `expression` *table.*

4. Move these non-primary key columns into a separate table. Remove the non-primary key columns from the original table. Name the new table as specifically as possible. This is shown in Figure 9.3.5.

   `Gene name` *and* `gene description` *have been moved into a separate table called* `gene` *and have been removed from the original table. Since these columns are now in the* `gene` *table, the prefix* `gene` *is dropped from both column names for brevity.*

5. Copy the corresponding columns in the primary key into this table as well. The columns copied from the original table's primary key comprise the primary key for this new table as shown in Figure 9.3.6.

   `Gene id` *has been copied into* `gene` *and has been made the primary key of that table. This column still resides in the table* `expression` *as well.*

6. Set up a foreign key from the original table to the new table.

   The foreign key links a column or set of columns in a table (the child) to a column or set of columns in a unique index, a primary key is a type of unique index, in another table (the parent). This linkage ensures data consistency between the two tables. With a foreign key in place, the DBMS will only allow values in the `gene id` *column of the* `expression` *table that are also in the* `gene id` *column of the* `gene` *table. The relationship between tables is shown, where appropriate, in each figure using an arrow: the arrow points from the child to the parent. In the example shown in Figure 9.3.7, the table* `gene` *is the parent table for the column* `gene id` *and the table* `expression` *is the child table.*

7. Repeat steps 2 to 6 until all non-primary key columns in the table describe all columns in the primary key. This should be done for every table.

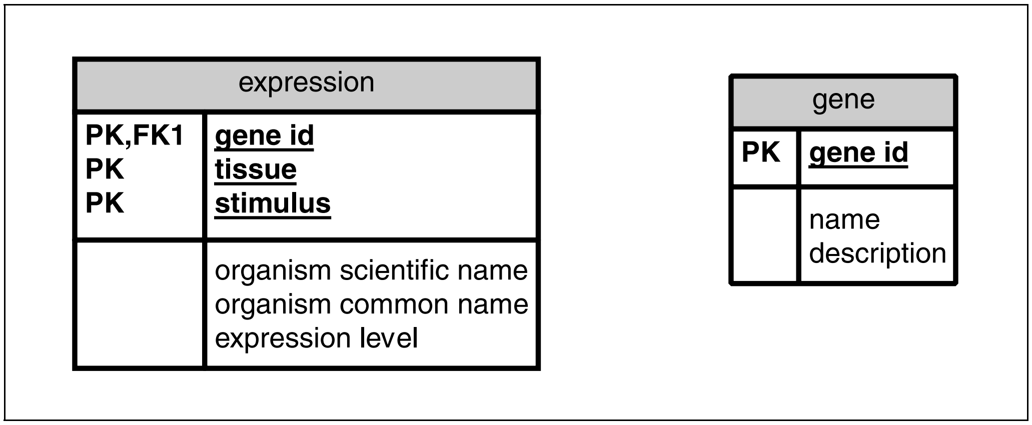


Figure 9.3.6 The portion of the original table's primary key, `gene id`, described by the columns in the new table becomes the primary key of the new table.

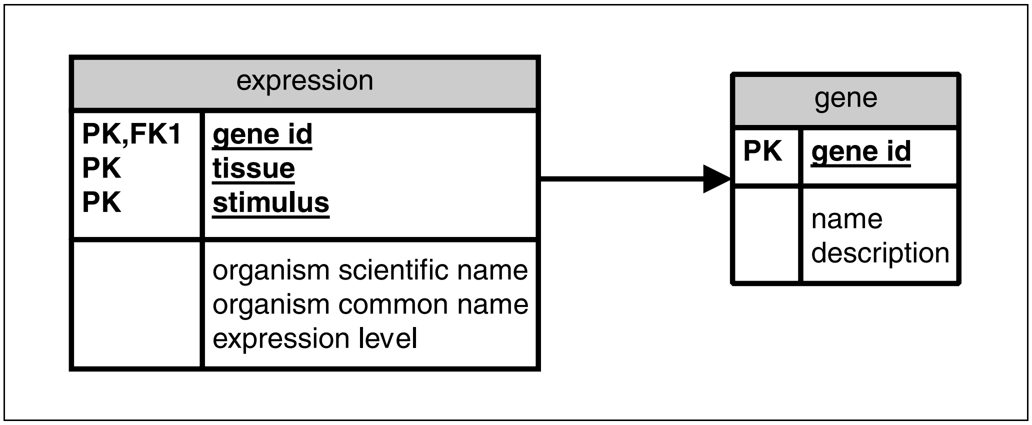


Figure 9.3.7 A foreign key is set up between the `gene id` column in the `expression` table (the child table) and the `gene id` column in the `gene` table (the parent table). A foreign key is used to ensure data integrity between two tables; the only values allowed in the `gene id` column in the `expression` table are those that are found in the `gene id` column in the `gene` table. The arrow between the two tables indicates the parent-child relationship between the two tables; the arrow points from the child to the parent. The FK1 in the `expression` table indicates the child column in the child table.

In this example, there are no non-primary key columns that describe the `tissue` *column, so no more needs to be done.*

8. For each table, identify non-primary key columns that are interdependent on each other.

   `Organism common name` *depends on* `organism scientific name` *in the table* `expression`*; they are simply two ways of describing an organism (e.g., human versus Homo sapiens).*

9. Create a new table for each set of interdependent columns and identify a primary key for the new table as shown in Figure 9.3.8.

   A new table `organism` *is created with the columns* `organism scientific name` *and* `organism common name`*;* `organism scientific name` *is arbitrarily selected to be the primary key. Since these columns are now in the* `organism` *table, the prefix* `organism` *is dropped from both column names for brevity.*

10. Replace each set of interdependent columns in the parent table with the primary key from the child table. To ensure data integrity, declare the column to be a foreign key in the parent table. This is shown in Figure 9.3.9.

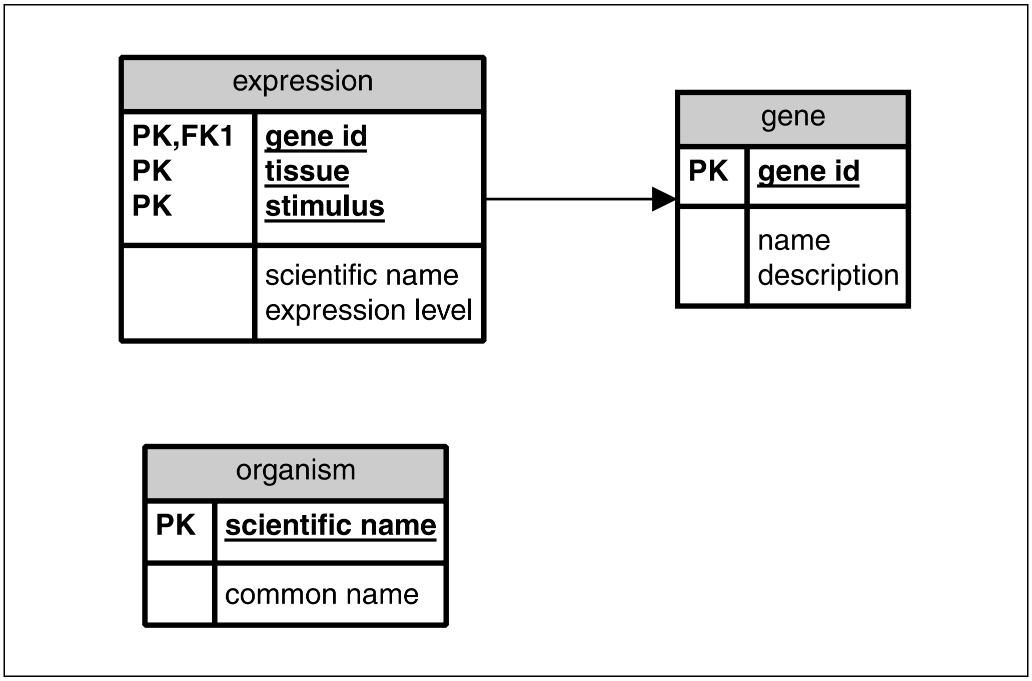


Figure 9.3.8 The columns `organism scientific name` and `organism common name` in the table `expression` are interdependent: they describe each other. `Organism common name` is moved to a new table `organism`. Arbitrarily, `organism scientific name` is selected to be the primary key in the new table `organism` and remains in the table `expression`. The `organism` prefix is removed from the column names for brevity.

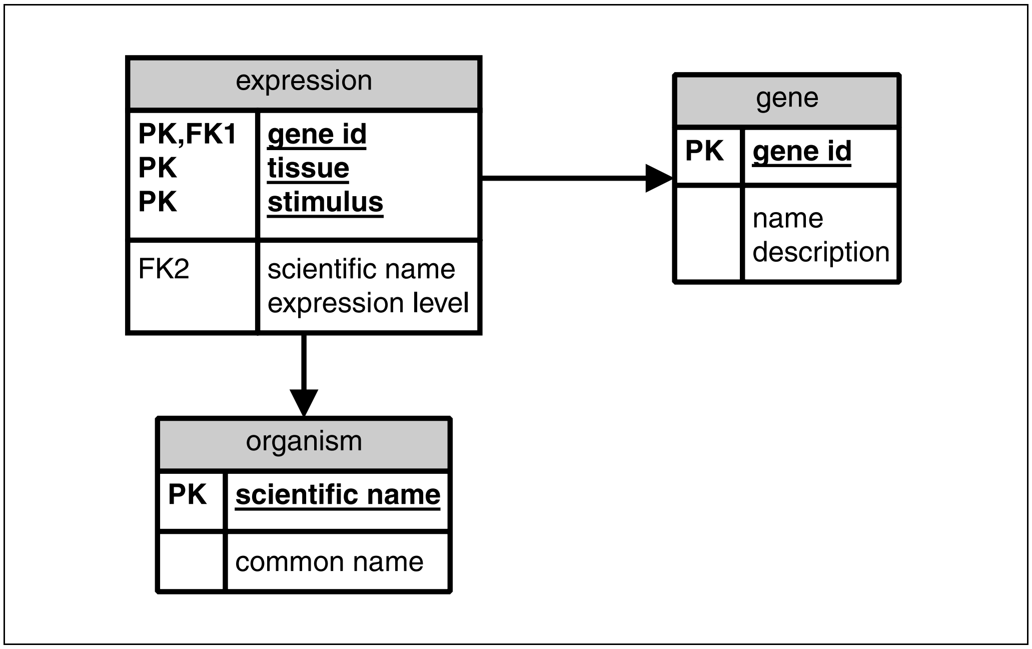


Figure 9.3.9 A foreign key is set up from `scientific name` in the `expression` table to `scientific name` in the `organism` table as indicated by the arrow between the two tables and the FK2 in the `expression` table.

```
create table gene
            (gene_id varchar(50) not null,
             name    varchar(10) not null,
             defn    text        not null,
             primary key (gene_id));

create table organism
            (scientific_name varchar(100) not null,
             common_name     varchar(100) not null,
             primary key (scientific_name));

create table expression
            (gene_id           varchar(50)   not null,
             tissue            varchar(100)  not null,
             scientific_name   varchar(100)  not null,
             exp_level         real(10,5)    not null,
             stimulus          varchar(100)  not null,
             primary key (gene_id, tissue, stimulus),
             foreign key expression_f1 (gene_id)
                     references gene (gene_id),
             foreign key expression_f2 (scientific_name)
                     references organism (scientific_name));
```

Figure 9.3.10 The SQL statements to create the tables in MySQL for the schema shown in Figure 9.3.9. To implement this schema in a different DBMS, the column types (e.g., `text`) will need to be adapted appropriately. Please refer to individual DBMS documentation for supported data types.

`Organism common name` *is removed from the* `expression` *table, and* `scientific name` *becomes a foreign key in the* `expression` *table.*

11. Repeat steps 8 through 10 for each table.

The tables are now in third normal form (see Commentary); data redundancy has been eliminated within each table and the final form of the database is shown in Figure 9.3.9.

Creating the tables

12. Create tables using SQL statements as shown in Figure 9.3.10. (An introduction to SQL can be found in *UNIT 9.2*).

For simplicity, several column names have been abbreviated (e.g., the column `expression level` *is* `exp_level` *in the table;* `defn` *for* `description` *since* `desc` *is a reserved word in SQL). Note that the order of the create statements depends on the relationships between the tables: a parent table must be created prior to a child table. In this example, the table* `expression` *has a foreign key to the table* `gene` *and to the table* `organism` *and so must be created after them. The same order must be followed when data is loaded into the tables: data must be entered into a parent table before it can be used in a child table. While foreign keys are a part of standard SQL, the exemplar DBMS, MySQL, used in other units does not yet implement them. Currently, MySQL will ignore any mention of foreign keys in the* `create` *statements and will not check for data consistency between tables. Support for foreign keys will be added with the upcoming release of version 4.1.*

Querying the database

13. Create queries to address questions of interest.

Once the database has been created and the data loaded, answers may be obtained to many interesting questions. Queries are set up by determining what information is wanted and which tables contain it, then finding a way to relate that information to each other through a process similar to the children's game of connecting the dots. In the small example schema, the relationships are obvious. In a database with many tables, the information

```
select g.name,   g.gene_id,   g.defn, o.common_name,
       e.tissue, e.exp_level, e.stimulus
  from gene g,
       organism o,
       expression e
 where g.gene_id         = e.gene_id
   and e.scientific_name = o.scientific_name
 order by g.name, e.stimulus, e.tissue, o.common_name;
```

Figure 9.3.11 The SQL statement to reproduce the spreadsheet shown in Figure 9.3.2 based on the schema shown in Figure 9.3.9.

```
select g.name, g.gene_id, g.defn, e.tissue, e.exp_level
  from gene g,
       organism o,
       expression e
 where g.gene_id         = e.gene_id
   and e.exp_level      >= 2
   and e.scientific_name = o.scientific_name
   and o.common_name     = 'human'
 order by g.name, e.stimulus, e.tissue, o.common_name;
```

Figure 9.3.12 The SQL query to answer the question, "What genes have an expression level greater or equal to two-fold in humans?"

may be related only by bringing together other information in other tables and all these tables are included in the query. The number of tables in a query should be kept to a minimum, too many tables in a query of a schema designed using the E-R method can make the query run excruciatingly slow. Try to limit the number of tables to around six or so depending on the number of columns and rows in each table. Additional indices may have to be added to selected tables to improve the performance of a query, though care must be taken not to adversely affect other queries. If it is impossible to relate the information with just a few tables, then either the query will have to be broken down into multiple smaller queries with the interim results stored in a temporary table, or the schema will have to be denormalized. Denormalization is the process used to introduce redundancy back into a normalized database: a column from one table is duplicated in another table used in the query and a foreign key is set up between the new column and the original one. This process reduces the number of tables from frequently used queries that would otherwise execute too slowly.

The following examples show how to ask different types of questions of the schema shown in Figure 9.3.9.

a. Figure 9.3.11 shows the SQL to recreate the original spreadsheet in Figure 9.3.2.

A column in one table must be distinguished from a column with the same name in a second table (e.g., `gene_id` *in the table* `gene` *and* `gene_id` *in the table* `expression`*). One method is the prefix each column name with* `tablename` *(e.g.,* `gene.gene_id` *and* `expression.gene_id`*). However, this can lead to an inordinate amount of typing and can even make the query difficult to read. SQL allows for table abbreviations to be used; the* `g`*,* `o`*, and* `e` *in Figure 9.3.11 are the abbreviations for the tables* `gene`*,* `organism`*, and* `expression`*, respectfully, and are defined in the* `from` *clause in the query. Use of sensibly selected one- to three-letter abbreviations reduces amount of typing while still making it obvious which table is being referenced. The* `order by` *clause determines how the results will be sorted, in this example, column names are used.*

```
+-------+-----------+-----------------------------------+--------+-----------+
| name  | gene_id   | defn                              | tissue | exp_level |
+-------+-----------+-----------------------------------+--------+-----------+
| g6pc  | nm_000151 | Glucose-6-phosphatase             | liver  |  20.00000 |
| g6pc  | nm_000151 | Glucose-6-phosphatase             | liver  |   2.00000 |
| pepck | nm_002591 | phosphoenolpyruvate carboxykinase 1 | liver |  10.20000 |
+-------+-----------+-----------------------------------+--------+-----------+
```

Figure 9.3.13 Results to the SQL query shown in Figure 9.3.13 obtained from a database implemented in MySQL.

```
select g.defn, e1.tissue,
      (e1.exp_level - e2.exp_level), e1.stimulus, e2.stimulus
  from gene g,
       organism o,
       expression e1,
       expression e2
 where g.gene_id          = e1.gene_id
   and e1.scientific_name = o.scientific_name
   and e1.gene_id         = e2.gene_id
   and e1.tissue          = e2.tissue
   and e1.scientific_name = e2.scientific_name
   and abs(e1.exp_level - e2.exp_level) >= 2
   and o.common_name      = 'human'
 order by 3, 1, 4, 5;
```

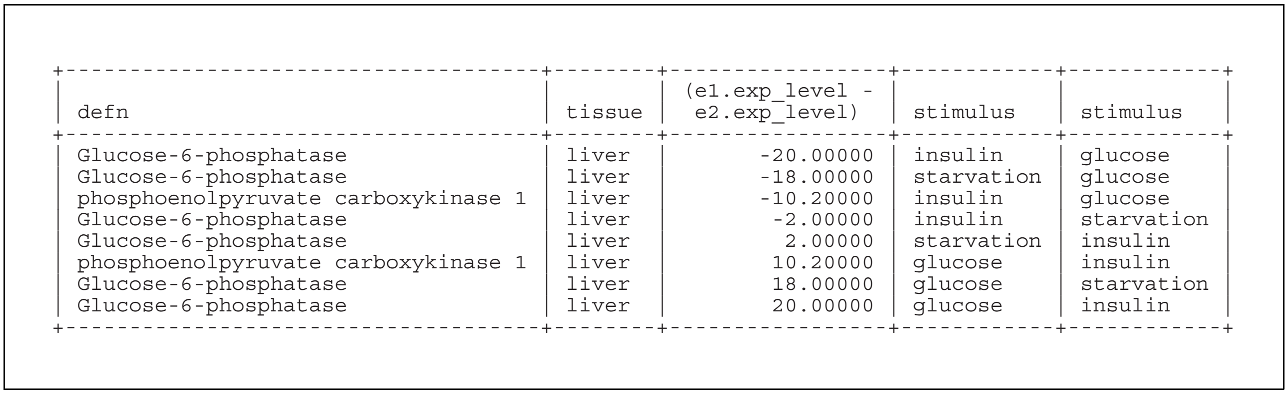
Figure 9.3.14 The SQL query to answer the question, "What genes are differentially expressed in human tissues and what are the stimuli that caused the differential expression?" The `order by` clause uses position numbers since the `select` clause has an expression in it. The position numbers correspond to the columns named in the `select` clause: `g.defn` is in position 1 and `e2.stimulus` is in position 5.

b. Figure 9.3.12 shows the SQL for the question, "What genes have an expression level greater or equal to two-fold in humans?" The results are shown in Figure 9.3.13.

This query requires that expression data from one stimulus be compared to expression data from another stimulus. This is achieved by having two copies of the `expression` *table, denoted by the abbreviations* `e1` *and* `e2` *thus showing another use for abbreviations, in the query. Since the* `select` *clause of the query has an expression in it, position numbers are used in the* `order by` *clause; positions 5, 1, 6, and 7 correspond to the expression* `(e1.exp_level - e2.exp_level)` *and the columns* `g.name`, `e1.stimulus`, *and* `e2.stimulus` *in the* `select` *clause, respectfully.*

c. Figure 9.3.14 shows the SQL for the question, "What genes are differentially expressed in human tissues and what are the stimuli that cause the differential expression?" Figure 9.3.15 shows the results.

More complex queries of this sort can be created using subselects and group functions like `intersect` *and* `minus` *(scheduled to be included in release 4.1 of MySQL and already available in other DBMS). The group function* `intersect` *returns entries that are common between two* `selects` *and* `minus` *returns entries from one* `select` *that are not found in a second* `select`. *A subselect is a way of using a* `select` *statement inside of a second* `select` *statement. An example of the group function* `minus` *is found in the protocol for the hybrid schema (see Alternate Protocol).*



```
+-------------------------------------+--------+----------------+------------+------------+
|                                     |        | (e1.exp_level - |            |            |
| defn                                | tissue |  e2.exp_level) | stimulus   | stimulus   |
+-------------------------------------+--------+----------------+------------+------------+
| Glucose-6-phosphatase               | liver  |      -20.00000 | insulin    | glucose    |
| Glucose-6-phosphatase               | liver  |      -18.00000 | starvation | glucose    |
| phosphoenolpyruvate carboxykinase 1 | liver  |      -10.20000 | insulin    | glucose    |
| Glucose-6-phosphatase               | liver  |       -2.00000 | insulin    | starvation |
| Glucose-6-phosphatase               | liver  |        2.00000 | starvation | insulin    |
| phosphoenolpyruvate carboxykinase 1 | liver  |       10.20000 | glucose    | insulin    |
| Glucose-6-phosphatase               | liver  |       18.00000 | glucose    | starvation |
| Glucose-6-phosphatase               | liver  |       20.00000 | glucose    | insulin    |
+-------------------------------------+--------+----------------+------------+------------+
```

Figure 9.3.15 Results to the SQL query shown in Figure 9.3.14.

USING THE HYBRID METHOD TO DESIGN A DATABASE

ALTERNATE PROTOCOL

This protocol describes the use of hybrid method modeling to design a database and covers designing the table schema (outlined in the flowchart in Fig. 9.3.16), creating tables, and finally, querying the database to ask interesting biological questions. Another typical problem in genomics is used as an example: storing sequence annotation. Figure 9.3.17 shows a small subset of information frequently obtained from public data sources that describe sequences: a sequence identifier (or accession number), the gene name, a description, where the sequence was obtained, and the tissue and organism from which that sequence was putatively obtained. The data redundancy in this subset is readily apparent. In addition, it is difficult to spot the gene that is found in human and not in mouse, much less to find more intricate differences or similarities between the genes. Even with this rather simple example, the relationships between the three main pieces of information (sequence, tissue, and organism) are not always well known or well documented. The schema design method highlighted in the following protocol works well when the relationships are uncertain since any changes to the schema are highly localized and thus have a minimal impact on the rest of the database.

Necessary Resources

Hardware

Personal computers with the Microsoft Windows operating system

Software

There are several software packages for designing databases, some examples are: Microsoft Access, Designer from Oracle, PowerDesigner from Sybase, and Visio also from Microsoft. Pencil and paper also work well, and are recommended for those who are just starting out.

Files

None

Designing the table schema

1. Place all information in a single table as in Figure 9.3.18.

As in the E-R example (see Basic Protocol), the first step is to put all of the information in a single table, at least conceptually, and then eliminate data redundancy. Note that the abbreviation `seq` *is being used as the table name and not the word* `sequence`*. In many database products,* `sequence` *is a reserved word for a function that automatically generates a sequence of numbers.*

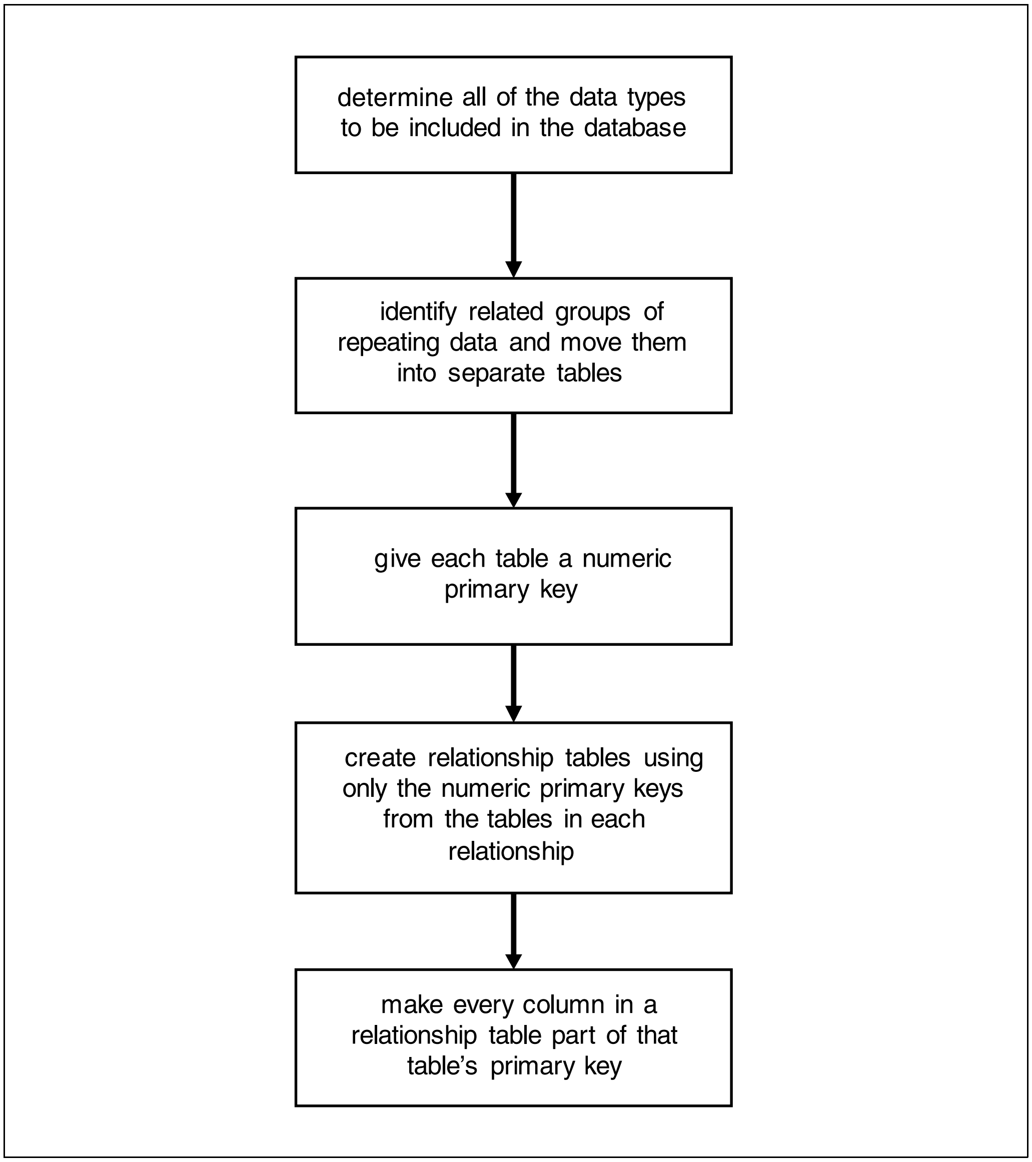
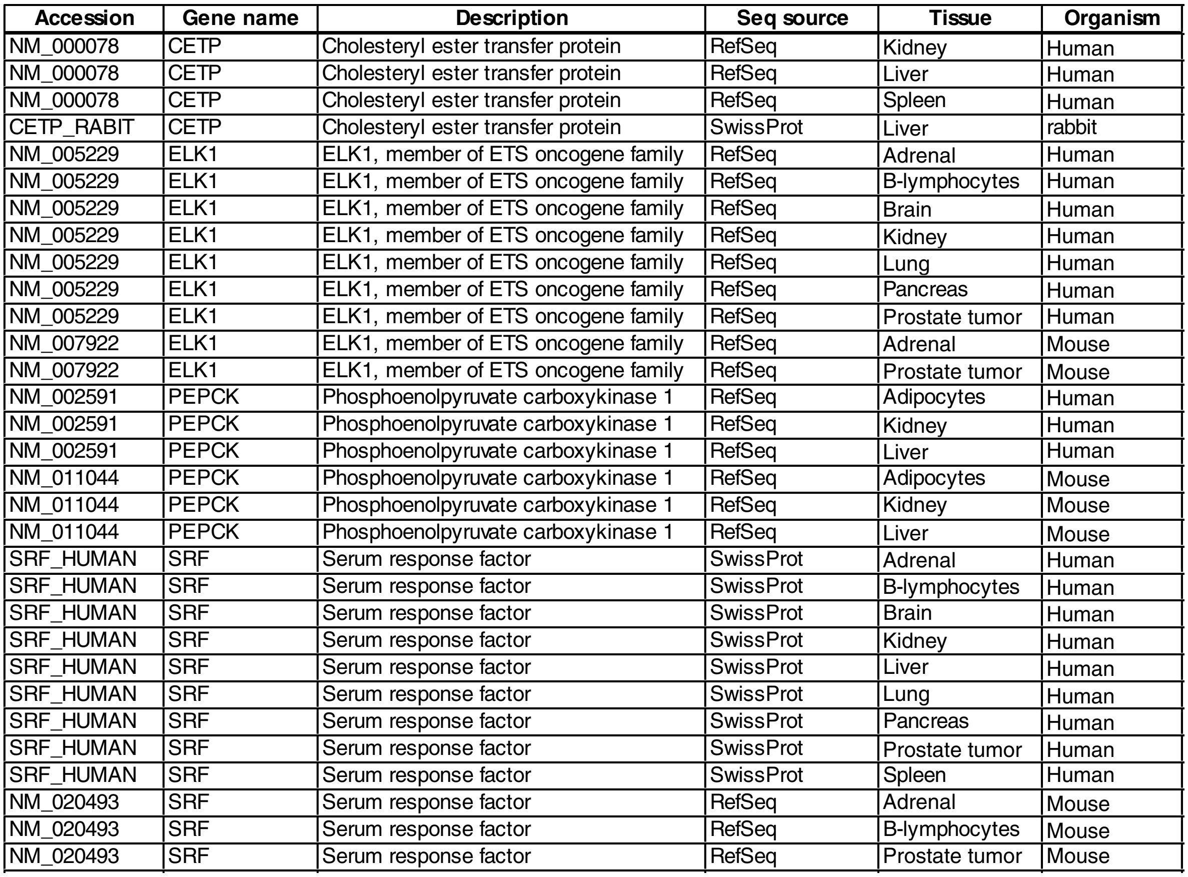

determine all of the data types to be included in the database
identify related groups of repeating data and move them into separate tables
give each table a numeric primary key
create relationship tables using only the numeric primary keys from the tables in each relationship
make every column in a relationship table part of that table’s primary key

Figure 9.3.16 Flowchart outlining steps for designing table schema for hybrid method modeling, as described in Alternate Protocol.

2. Identify related repeating groups of data and move them into separate tables as in Figure 9.3.19.

This is the most important step in the protocol and requires some practice. There are two guides for identifying the base types of information (often termed “entities”), look for controlled vocabularies and biology. Data using controlled vocabularies, e.g., an ontology, should be moved into separate tables. Many entities fall right out of biology, e.g., organism. Tables should be named as specifically and succinctly as possible, using the singular form of the word(s). Figure 9.3.20 shows the general form of an entity table along with two examples.

There are three sources of redundancy in the single table in Figure 9.3.20: `seq source` *(a sequence can be found multiple times in the same data source),* `tissue`*, and* `organism`*; with many entries in the table, the same values will be seen repeatedly for these fields. Each set of related sources of redundancy is moved to a separate table. Many will be obvious since they reflect biological entities. For example,* `scientific name` *and* `common name` *are sources of redundancies but are related as they both describe an organism. Note that the* `organism` *prefix has been dropped from* `scientific name` *and* `common name` *for brevity.*



Accession	Gene name	Description	Seq source	Tissue	Organism
NM_000078	CETP	Cholesteryl ester transfer protein	RefSeq	Kidney	Human
NM_000078	CETP	Cholesteryl ester transfer protein	RefSeq	Liver	Human
NM_000078	CETP	Cholesteryl ester transfer protein	RefSeq	Spleen	Human
CETP_RABIT	CETP	Cholesteryl ester transfer protein	SwissProt	Liver	rabbit
NM_005229	ELK1	ELK1, member of ETS oncogene family	RefSeq	Adrenal	Human
NM_005229	ELK1	ELK1, member of ETS oncogene family	RefSeq	B-lymphocytes	Human
NM_005229	ELK1	ELK1, member of ETS oncogene family	RefSeq	Brain	Human
NM_005229	ELK1	ELK1, member of ETS oncogene family	RefSeq	Kidney	Human
NM_005229	ELK1	ELK1, member of ETS oncogene family	RefSeq	Lung	Human
NM_005229	ELK1	ELK1, member of ETS oncogene family	RefSeq	Pancreas	Human
NM_005229	ELK1	ELK1, member of ETS oncogene family	RefSeq	Prostate tumor	Human
NM_007922	ELK1	ELK1, member of ETS oncogene family	RefSeq	Adrenal	Mouse
NM_007922	ELK1	ELK1, member of ETS oncogene family	RefSeq	Prostate tumor	Mouse
NM_002591	PEPCK	Phosphoenolpyruvate carboxykinase 1	RefSeq	Adipocytes	Human
NM_002591	PEPCK	Phosphoenolpyruvate carboxykinase 1	RefSeq	Kidney	Human
NM_002591	PEPCK	Phosphoenolpyruvate carboxykinase 1	RefSeq	Liver	Human
NM_011044	PEPCK	Phosphoenolpyruvate carboxykinase 1	RefSeq	Adipocytes	Mouse
NM_011044	PEPCK	Phosphoenolpyruvate carboxykinase 1	RefSeq	Kidney	Mouse
NM_011044	PEPCK	Phosphoenolpyruvate carboxykinase 1	RefSeq	Liver	Mouse
SRF_HUMAN	SRF	Serum response factor	SwissProt	Adrenal	Human
SRF_HUMAN	SRF	Serum response factor	SwissProt	B-lymphocytes	Human
SRF_HUMAN	SRF	Serum response factor	SwissProt	Brain	Human
SRF_HUMAN	SRF	Serum response factor	SwissProt	Kidney	Human
SRF_HUMAN	SRF	Serum response factor	SwissProt	Liver	Human
SRF_HUMAN	SRF	Serum response factor	SwissProt	Lung	Human
SRF_HUMAN	SRF	Serum response factor	SwissProt	Pancreas	Human
SRF_HUMAN	SRF	Serum response factor	SwissProt	Prostate tumor	Human
SRF_HUMAN	SRF	Serum response factor	SwissProt	Spleen	Human
NM_020493	SRF	Serum response factor	RefSeq	Adrenal	Mouse
NM_020493	SRF	Serum response factor	RefSeq	B-lymphocytes	Mouse
NM_020493	SRF	Serum response factor	RefSeq	Prostate tumor	Mouse

Figure 9.3.17 A small portion of typical sequence annotation obtained from public data sources. The full dataset contains the gene name, a sequence identifier (accession number), a description of the gene, the name of the repository where the identifier can be used to retrieve the sequence, and the tissue and organism from which the sequence was putatively obtained.

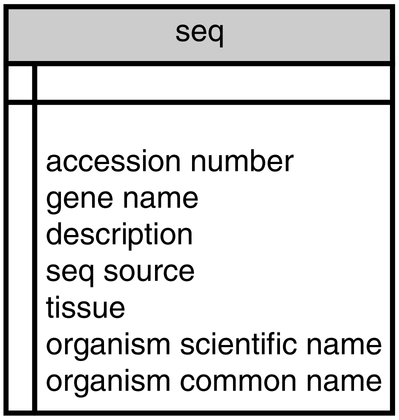


Figure 9.3.18 A single table is created containing all of different types of information obtained from the gene expression study. Often, this is done as a mental exercise. The abbreviation `seq` is used for the table name since `sequence` is a reserved word in many implementations of SQL.

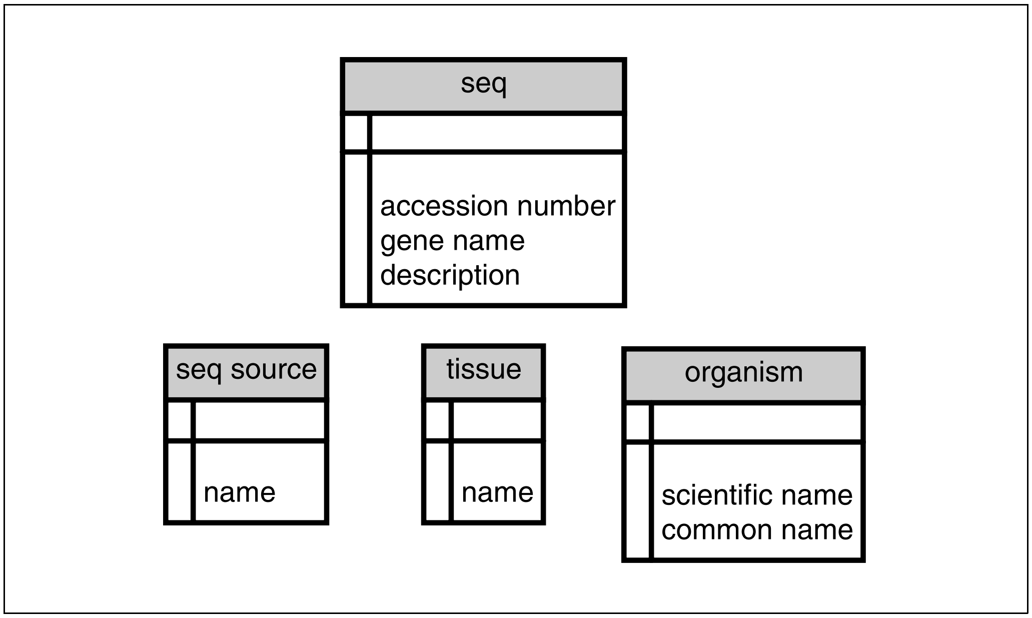


Figure 9.3.19 Related sets of repeating data (entities) have been identified and moved into their own entity tables.

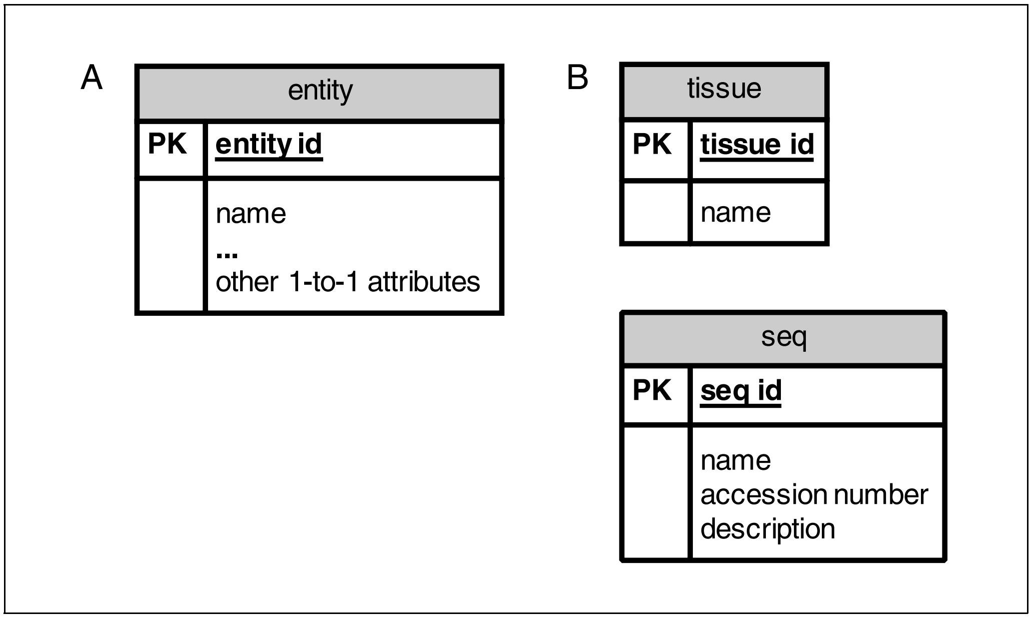


Figure 9.3.20 (**A**) The general form of an entity table used in the hybrid method. It consists of a numeric primary key and 1 to N attributes. (**B**) Two examples of entity tables. Redundancy in the table is either accepted or reduced through the liberal use of unique indices other than the primary key.

3. Give each table a numeric primary key as shown in Figure 9.3.21.

   The numeric primary key identifies a row in the table and is used to relate the tables to each other.

4. For each table, identify the non-primary key column or combination of non-primary key columns (properties) that define a unique entry in the table or that should not be found more than once in the table. Create a unique index using them, this will minimize data redundancy. Figure 9.3.22 shows the tables with the indices, the "U1" in a table shows the column with the index. (If there were a second unique index on a table, it would be indicated with "U2.")

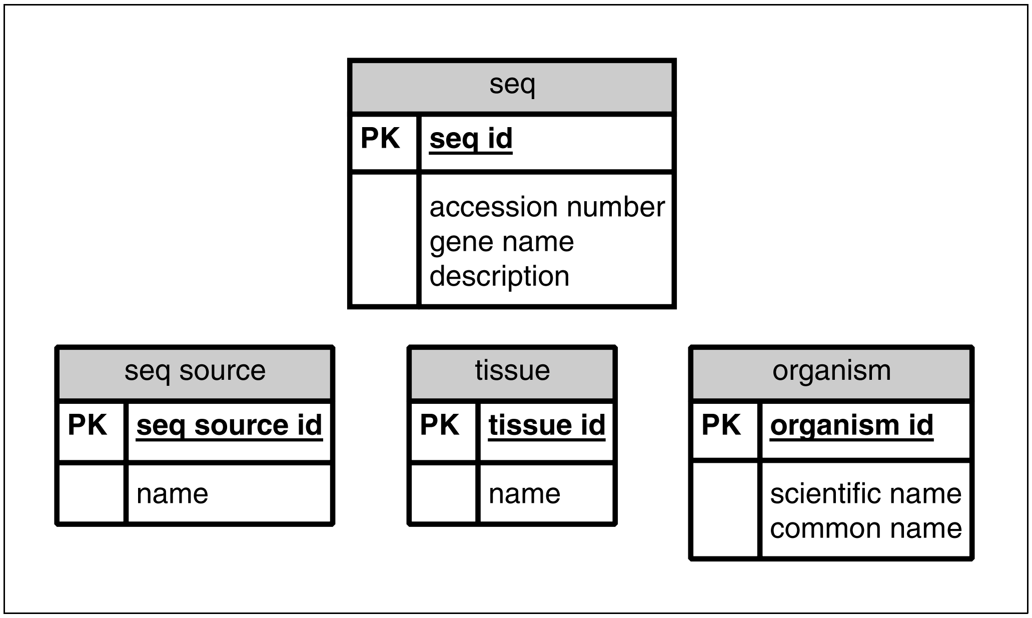


Figure 9.3.21 Each table is given a numeric primary key. This numeric identifier uniquely identifies a row in the table and is used to relate information in the table to information in a different table.

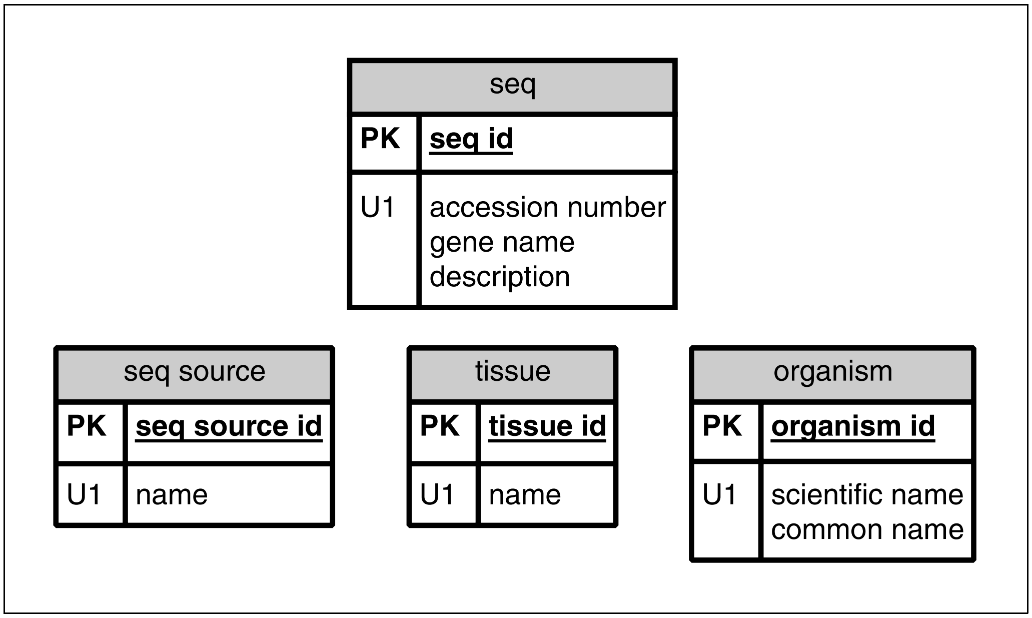


Figure 9.3.22 Data redundancy within a table is eliminated by identifying a column or a set of columns that uniquely identifies a row and then using them in a unique index, not in the primary key. A table will often have more than one unique index.

Liberal use of unique indices eliminates the data redundancy problem and it is not unusual for a table to have more than one unique index on it. For example, a unique index on `name` *in the table* `tissue` *will prevent a tissue name from being entered multiple times. The table* `organism` *is an interesting case. In Figure 9.3.22, it is shown with only* `scientific name` *in the unique index. In many cases, the unique index would consist of the pair of columns* `scientific name` *and* `common name`*. This would prevent a particular combination of* `scientific name` *and* `common name` *from being entered more than once. However,* `scientific name` *could be entered more than once, each time with a different* `common name`*. To fully prevent any duplication, the table would need three unique indices: one on* `scientific name`*, one on* `common name`*, and*

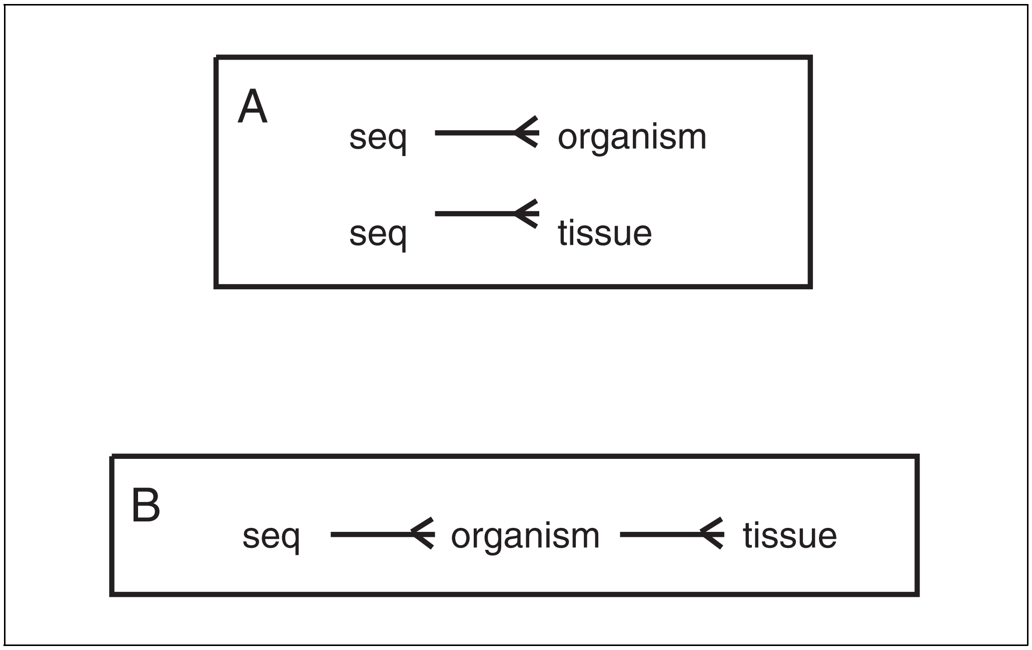


Figure 9.3.23 The figure shows the two different ways that a sequence can be linked to an organism and tissue. (**A**) A sequence can be linked independently an organism and a tissue. This can lead to combinations of organism and tissue that might not have been found experimentally. (**B**) A sequence is linked to an organism and a tissue.

one on the pair (`scientific name`*,* `common name`*). However, by putting the unique index on just* `scientific name`*, the same* `common name` *can now be entered multiple times. This can be an advantage if combining the information from different species is desired. For example, combining the information for* `Rattus rattus` *and* `Rattus norvegicus` *simply by querying on the common name* `rat` *would be possible. Since this table will most likely be small, putting an index on* `common name` *is not necessary.*

5. Identify related entities.

Some relationships are obvious, the relationship between `seq` *and* `seq source` *for instance. Other relationships, however, are not what they initially appear. Case in point,* `tissue` *and* `organism` *might be unrelated or they might be related. If they are unrelated, then there can be an* `organism` *entry for every* `tissue` *entry. If they are related, then only certain combinations of* `organism` *and* `tissue` *are allowed. See Figure 9.3.23.*

6. Create a relationship table for each set of related entities.

A relationship table will only contain columns corresponding to the primary keys of each entity in the relationship. The primary key of a relationship table will be comprised of every column in a relationship table. Unlike in a data warehouse fact table, no other data will appear in relationship table. Figure 9.3.24 shows a general form for naming a relationship table along with an example. Each column will also be a foreign key back to its associated entity table. A foreign key links a column or set of columns in a table (the child) to a column or set of columns in a unique index, a primary key is a type of unique index, in another table (the parent). This linkage ensures data consistency between the two tables. With a foreign key in place, the DBMS will only allow values in the `seq id` *column of the* `seq2seq source` *table that are also in the* `seq id` *column of the* `seq` *table. The relationship between tables is shown where appropriate in each figure using an arrow: the arrow points from the child to the parent.*

Figure 9.3.25 shows the schema assuming that there is no relationship between the tables `organism` *and* `tissue`*.*

If the `organism` *and* `tissue` *tables are related, then a relationship table must be set up to reflect this. Figure 9.3.26 shows one version of the schema. Here,* `organism id`*,*

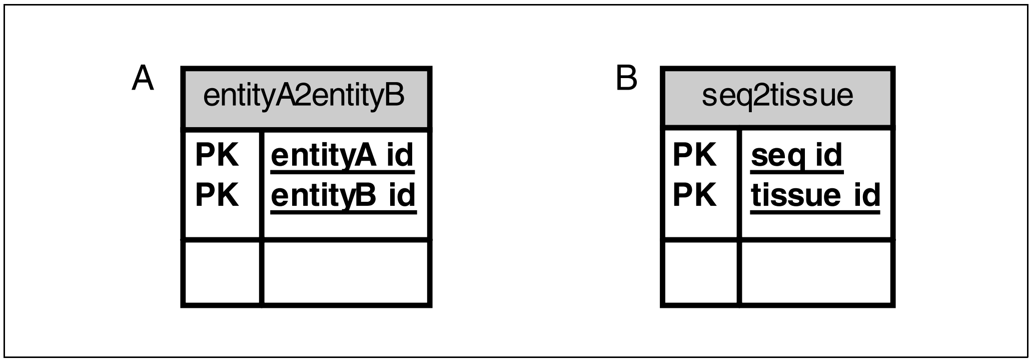


Figure 9.3.24 (**A**) The general form of a relationship table used in the hybrid method is shown. It consists of numeric columns corresponding to the primary keys of the entity tables in the relationship. All columns in a relationship table comprise its primary key and are linked back to their respective parent tables by foreign keys. (**B**) An example of relationship tables is shown.

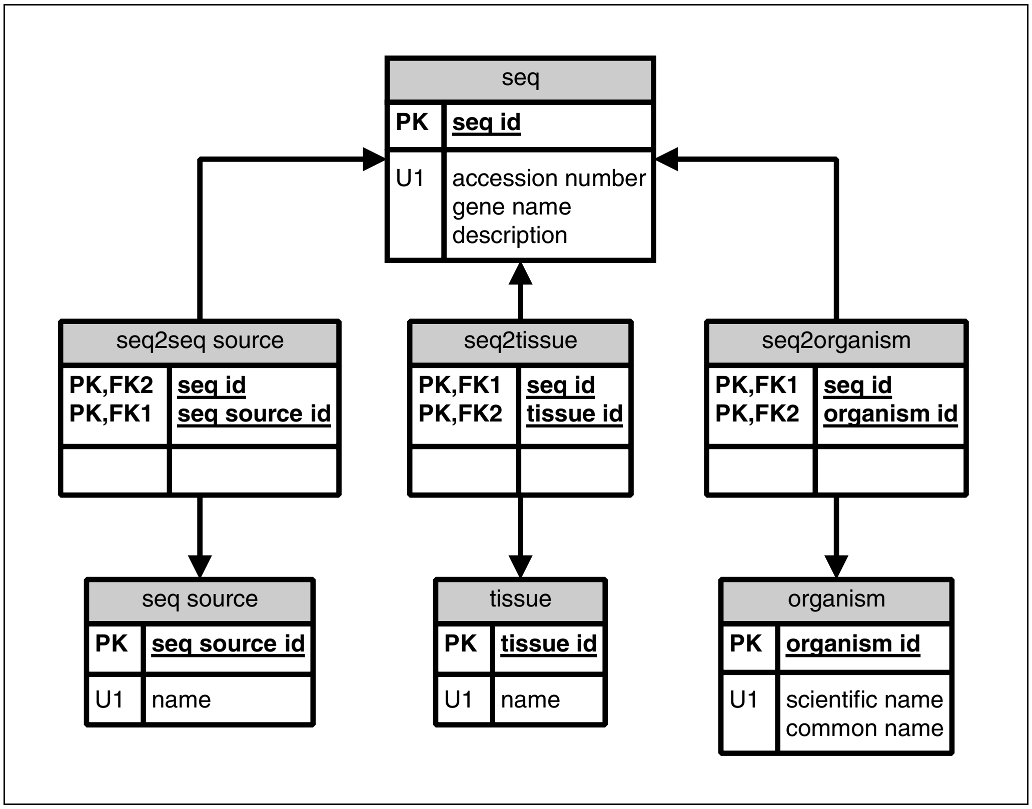


Figure 9.3.25 Relationship tables are created by linking related entities. In this schema, it is assumed that there is no direct relationship between `organism` and `tissue`. The primary key in each relationship table is comprised of all of the columns in the table. Foreign keys are set up between each relationship table and the entity tables involved in the relationship.

`tissue id`, *and* `seq id` *are combined in a single relationship table and all three columns comprise the primary key. Figure 9.3.27 shows a second version. Here, an intermediary relationship table is set up combining* `organism id` *and* `tissue id` *to create the new entity* `library` *with its own unique primary key* `library id` *(the name was chosen under the assumption that the sequence data derives from a cloning step).* `Organism id` *and* `tissue id` *together comprise a standard unique index and not the primary key. This new entity itself can now be used in other relationships. Either form of the schema, Figure 9.3.26 or 9.3.27, is equally valid when* `organism` *and* `tissue` *are related.*

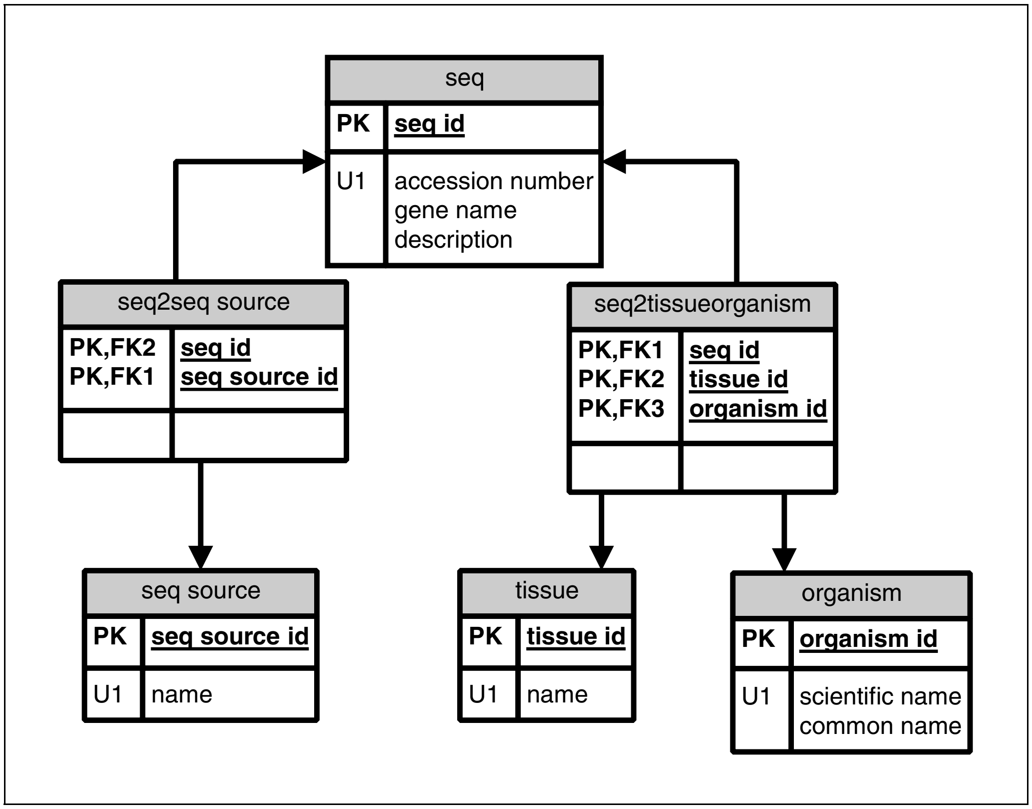


Figure 9.3.26 Relationship tables are created by linking related entities. In this schema, it is assumed that the `organism` and `tissue` tables are directly related. This is shown in the relationship table `seq2organismtissue` which contains three columns, one for each entity in the relationship. A comparison of this schema to the one in Figure 9.3.25 shows its flexibility: the base tables `seq`, `seq source`, `tissue` and `organism` and the relationship table `seq2seq source` remain untouched.

A comparison of these three schemas shows the flexibility of schemas developed using the hybrid method. All of the base tables and the relationship table `seq2seq source` *have remained untouched.*

Creating the tables

7. Examine Figure 9.3.28, which shows the SQL statements to create the base tables `seq`, `seq_source`, `organism`, and `tissue` used in both of the final forms of the schema where `organism` and `tissue` are related (Figs. 9.3.26 and 9.3.27). Figure 9.3.29 shows the statements to create the relationship tables `seq2seq_source` and `seq2tissueorganism` used in the schema shown in Figure 9.3.20. Figure 9.3.30 shows the statements to create the relationship tables `seq2seq_source`, `library`, and `seq2library` used in the schema shown in Figure 9.3.27. Altering the SQL statements to create the tables in the schema where `organism` and `tissue` are unrelated (Fig. 9.3.25) is straightforward.

Note that the order of the `create` *statements depends on the relationships between the tables: a parent table must be created prior to a child table. In this example, the table* `seq2seq_source` *has foreign keys to both the* `seq` *and the* `seq_source` *tables and so must be created after them. (As mentioned previously, MySQL is scheduled to implement foreign keys in version 4.1 and currently ignores any mention of foreign keys in the* `create` *statements and does not do any checking for referential integrity.) The same order must be followed when data is loaded into the tables: data must be entered into a parent table before it can be used in a child table. Typically, data is loaded by first checking to see if it already exists in an entity table. If it does, then the* `ID` *for that row in the table is obtained. If not, then the data is entered into a new row and the* `ID` *for the new row is obtained. This is then*

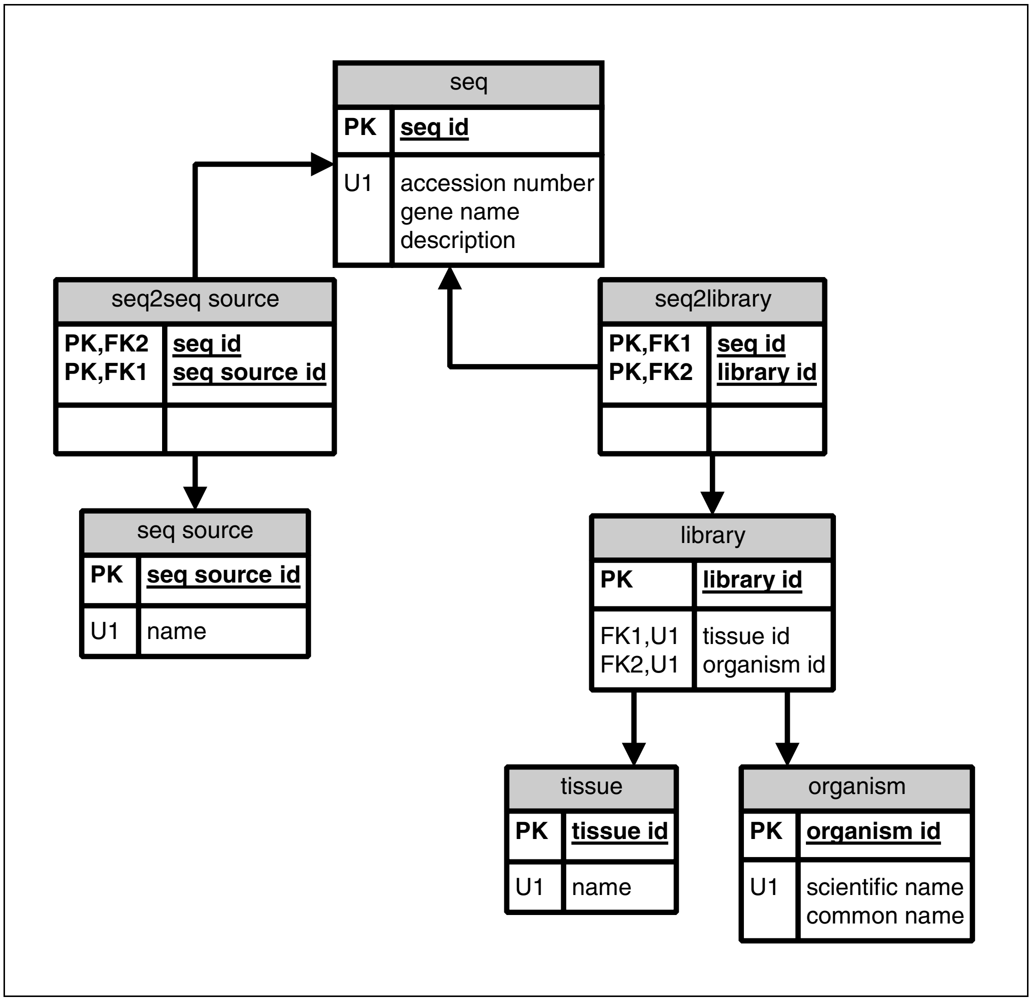


Figure 9.3.27 This figure shows a second way of setting up the sequence-organism-tissue relationship. This form sets up a relationship between `organism` and `tissue` called `library` with its own primary key. This new entity is then related to sequence in `seq2library` and can be used in other relationships.

done for all other tables whose identifiers are found in a relationship table. The relationship table is then checked to see if that combination of IDs already exists. If that combination is not found, then they are entered into the table. This process of checking each entry increases the amount of time required to load the information into the database (see Commentary).

Querying the database

8. Create queries to address questions of interest.

Once the database has been created and the data loaded, answers may be obtained to many interesting questions. Queries are set up by determining what information is wanted and which tables contain it, then finding a way to relate that information to each other through a process similar to the children's game of connecting the dots. In this small example schema, the relationships are obvious. In a database with many tables, the information may be related only by bringing together other information in other tables. The very nature of a hybrid schema means that many tables will be involved in even a simple query. Because of the structure of the tables (data-rich but short entity tables and data-poor but long relationship tables built from the primary keys of entity tables), this generally does not have a significant impact on the time required for a query to execute (but not always). It is not unusual to have ten or twelve tables included in a query; some queries can become quite lengthy and complex and still run in an acceptable amount of time with an acceptable load on the DBMS. This increase in query complexity along with the increase time for loading

```
create table seq
            (seq_id  bigint       not null,
             accno   varchar(25)  not null,
             name    varchar(100) not null,
             defn    text         not null,
             primary key (seq_id),
             unique index seq_i1 (accno));

create table seq_source
            (seq_source_id tinyint     not null,
             name          varchar(50) not null,
             primary key (seq_source_id),
             unique index seq_source_i1 (name));

create table tissue
            (tissue_id tinyint      not null,
             name      varchar(100)  not null,
             primary key (tissue_id),
             unique index tissue_i1 (name));

create table organism
            (organism_id     tinyint      not null,
             scientific_name varchar(100) not null,
             common_name     varchar(100) not null,
             primary key (organism_id),
             unique index organism_i1 (scientific_name));
```

Figure 9.3.28 The SQL statements to create the base tables `seq`, `seq_source`, `tissue`, and `organism` in MySQL for the schemas shown in Figures 9.3.25, 9.3.26, and 9.3.27. To implement this schema in a different DBMS, the column types (e.g., `bigint`) will need to be adapted appropriately. Please refer to specific DBMS documentation for supported data types.

```
create table seq2seq_source
           (seq_id        bigint  not null,
            seq_source_id  tinyint not null,
            primary key (seq_id, seq_source_id),
            foreign key seq2seq_source_f1 (seq_id)
                    references seq (seq_id),
            foreign key seq2seq_source_f1 (seq_source_id)
                    references seq_source (seq_source_id));

create table seq2tissueorganism
            (seq_id      bigint  not null,
             tissue_id   tinyint not null,
             organism_id tinyint not null,
             primary key (seq_id, tissue_id, organism_id),
             foreign key seq2tissueorganism_f1 (seq_id)
                     references seq (seq_id),
             foreign key seq2tissueorganism_f2 (tissue_id)
                     references tissue (tissue_id),
             foreign key seq2tissueorganism_f3 (organism_id)
                     references organism (organism_id));
```

Figure 9.3.29 The SQL statements to create the relationship tables in the schema where `organism` and `tissue` are related as shown in Figure 9.3.26.

```
create table seq2seq_source
           (seq_id         bigint  not null,
            seq_source_id  tinyint not null,
            primary key (seq_id, seq_source_id),
            foreign key seq2seq_source_f1 (seq_id)
                    references seq (seq_id),
            foreign key seq2seq_source_f1 (seq_source_id)
                    references seq_source (seq_source_id));

create table library
            (library_id  int     not null,
             tissue_id   tinyint not null,
             organism_id tinyint not null,
             primary key (library_id),
             unique index library_i1 (tissue_id, organism_id),
             foreign key library_f1 (tissue_id)
                     references tissue (tissue_id),
             foreign key library_f2 (organism_id)
                     references organism (organism_id));

create table seq2library
            (seq_id      bigint not null,
             library_id  int    not null,
             primary key (seq_id, library_id),
             foreign key seq2library_f1 (seq_id)
                     references seq (seq_id),
             foreign key seq2library_f2 (library_id)
                     references library (library_id));
```

Figure 9.3.30 The SQL statements to create the relationship tables in the schema where `organism` and `tissue` are related as shown in Figure 9.3.27.

data are the trade-offs for having a flexible schema that readily handles arbitrary data-mining queries. If a query is found to run too slowly, additional indices (often non-unique) may have to be added to selected base tables or the database can be denormalized. Denormalization is the process used to introduce redundancy back into a normalized database. In a hybrid schema, denormalization is typically done by either creating new relationship tables or adding a column to an existing relationship table. The entity tables are only rarely touched during denormalization.

The following examples show how to ask different types of questions of the schema shown in Figure 9.3.27. The same queries can be executed on the schemas shown in Figures 9.3.25 and 9.3.26 with some slight modifications.

a. Figure 9.3.31 shows the SQL to recreate the original spreadsheet in shown Figure 9.3.17.

Even with this simple query, seven tables are required. As discussed previously (see Basic Protocol), a column in one table must be distinguished from a column with the same name in a second table (e.g., `seq_id` *in the table* `seq` *and* `seq_id` *in the table* `seq2seq_source`*). One method is the prefix each column name with* `tablename.` *(e.g.,* `seq.seq_id` *and* `seq2seq_source.seq_id`*). However, this can lead to an inordinate amount of typing and can even make the query difficult to read. SQL allows for table abbreviations to be used; for example the* `s`*,* `s2s`*, and* `ss` *in Figure 9.3.31 are abbreviations for the tables* `seq`*,* `seq2seq_source`*, and* `seq_source`*, respectfully, and are defined in the* `from` *clause in the query. Use of sensibly selected one- to three-letter abbreviations reduces the amount of typing while still making it obvious which table is being referenced.*

b. Figure 9.3.32 shows the SQL to return only those genes found in humans, Figure 9.3.33 shows the results.

```
select s.accno, s.name, s.defn, ss.name,
        t.name, o.common_name
  from seq s,
       seq2seq_source s2s,
       seq_source ss,
       seq2library s2l,
       library l,
       tissue t,
       organism o
 where s.seq_id         = s2s.seq_id
   and ss.seq_source_id = s2s.seq_source_id
   and s2l.seq_id       = s.seq_id
   and s2l.library_id   = l.library_id
   and l.tissue_id      = t.tissue_id
   and l.organism_id    = o.organism_id
 order by s.name, t.name, o.common_name;
```

Figure 9.3.31 The SQL statement to reproduce the spreadsheet shown in Figure 9.3.17.

```
select distinct s.accno, s.name, s.defn, t.name
  from seq s,
       seq2library s2l,
       library l,
       organism o,
       tissue t
 where s2l.seq_id        = s.seq_id
   and s2l.library_id    = l.library_id
   and l.organism_id     = o.organism_id
   and o.common_name     = 'human'
   and t.tissue_id       = l.tissue_id
 order by s.name, t.name;
```

Figure 9.3.32 The SQL query to return only those genes found in humans based on the schema shown in Figure 9.3.27.

Requiring `o.common_name = 'human'` *restricts the query to just the human genes, though a gene with the same name could certainly exist in another organism.*

c. Figure 9.3.34 shows the SQL for the question, "What information exists in the database for the sequence with the RefSeq identifier NM_005229?" The results are shown in Figure 9.3.35.

Requiring `s.seq_id = 'nm_005229'` *restricts the query to only the information on that accession number. Since the* `seq id` *has been defined to be specific to an organism, only information for that organism will be retrieved. Note that the DBMS search engine considers* `'nm_005229'` *and* `'NM_005229'` *to be different entries. When data is loaded into the database, accession numbers and other types of information should be consistently set to either upper or lower case if possible.*

d. Figure 9.3.36 shows the SQL for the question, "What information exists in the database for the sequence with the name pepck?" The results are shown in Figure 9.3.37.

Unlike the previous query where an organism-specific `seq id` *was used, this uses a name that can be shared by multiple organisms.*

e. Figure 9.3.38 shows the SQL for the question, "What genes are found in human but not in mouse?" using the group function `minus`. Figure 9.3.39 shows the results for this query.

```
+-----------+-------+-------------------------------------+----------------+
| accno     | name  | defn                                | name           |
+-----------+-------+-------------------------------------+----------------+
| nm_000078 | cetp  | cholesteryl ester transfer protein  | kidney         |
| nm_000078 | cetp  | cholesteryl ester transfer protein  | liver          |
| nm_000078 | cetp  | cholesteryl ester transfer protein  | spleen         |
| nm_005229 | elk1  | elk1, member of ETS oncogene family | adrenal        |
| nm_005229 | elk1  | elk1, member of ETS oncogene family | b-lymphocytes  |
| nm_005229 | elk1  | elk1, member of ETS oncogene family | brain          |
| nm_005229 | elk1  | elk1, member of ETS oncogene family | kidney         |
| nm_005229 | elk1  | elk1, member of ETS oncogene family | lung           |
| nm_005229 | elk1  | elk1, member of ETS oncogene family | pancreas       |
| nm_005229 | elk1  | elk1, member of ETS oncogene family | prostate tumor |
| nm_002591 | pepck | phosphoenolpyruvate carboxykinase 1 | adipocytes     |
| nm_002591 | pepck | phosphoenolpyruvate carboxykinase 1 | kidney         |
| nm_002591 | pepck | phosphoenolpyruvate carboxykinase 1 | liver          |
| srf_human | srf   | serum response factor               | adrenal        |
| srf_human | srf   | serum response factor               | b-lymphocytes  |
| srf_human | srf   | serum response factor               | brain          |
| srf_human | srf   | serum response factor               | kidney         |
| srf_human | srf   | serum response factor               | liver          |
| srf_human | srf   | serum response factor               | lung           |
| srf_human | srf   | serum response factor               | pancreas       |
| srf_human | srf   | serum response factor               | prostate tumor |
| srf_human | srf   | serum response factor               | spleen         |
+-----------+-------+-------------------------------------+----------------+
```

Figure 9.3.33 Results to the SQL query shown in Figure 9.3.32 obtained from a database implemented in MySQL. `Name` is shown as a column header since it is found in two tables and both are listed in the `select` clause of the query.

```
select s.accno, s.name, s.defn, t.name,
o.common_name
  from seq s,
       seq2library s2l,
       library l,
       tissue t,
       organism o
 where s.accno          = 'nm_005229'
   and s2l.seq_id       = s.seq_id
   and s2l.library_id   = l.library_id
   and l.tissue_id      = t.tissue_id
   and l.organism_id    = o.organism_id
 order by s.name, t.name, o.common_name;
```

Figure 9.3.34 The SQL query to show all of the annotation in the database for the sequence with the identifier (accession number) `nm_005229`. Note that a DBMS query engine will consider `nm_005229` and `NM_005229` to be different entries. Care must be taken prior to loading the data to ensure that all identifiers are either all uppercase or all lowercase.

```
+-----------+------+------------------------------------+---------------+-------------+
| accno     | name | defn                               | name          | common_name |
+-----------+------+------------------------------------+---------------+-------------+
| nm_005229 | elk1 | elk1, member of ETS oncogene family | adrenal       | human       |
| nm_005229 | elk1 | elk1, member of ETS oncogene family | b-lymphocytes | human       |
| nm_005229 | elk1 | elk1, member of ETS oncogene family | brain         | human       |
| nm_005229 | elk1 | elk1, member of ETS oncogene family | kidney        | human       |
| nm_005229 | elk1 | elk1, member of ETS oncogene family | lung          | human       |
| nm_005229 | elk1 | elk1, member of ETS oncogene family | pancreas      | human       |
| nm_005229 | elk1 | elk1, member of ETS oncogene family | prostate tumor | human       |
+-----------+------+------------------------------------+---------------+-------------+
```

Figure 9.3.35 Results to the SQL query shown in Figure 9.3.34.

```
select s.accno, s.name, s.defn, t.name,
o.common_name
  from seq s,
       seq2library s2l,
       library l,
       tissue t,
       organism o
 where s.name           = 'pepck'
   and s2l.seq_id       = s.seq_id
   and s2l.library_id   = l.library_id
   and l.tissue_id      = t.tissue_id
   and l.organism_id    = o.organism_id
 order by s.name, t.name, o.common_name;
```

Figure 9.3.36 The SQL query to retrieve information on the gene with the name `pepck` regardless of the organism. As mentioned previously, the DBMS query engine will consider `PEPCK` or any variation with different upper and lowercase letters to be different from the all lowercase `pepck`.

```
+-----------+-------+------------------------------------+------------+-------------+
| accno     | name  | defn                               | name       | common_name |
+-----------+-------+------------------------------------+------------+-------------+
| nm_002591 | pepck | phosphoenolpyruvate carboxykinase 1 | adipocytes | human       |
| nm_011044 | pepck | phosphoenolpyruvate carboxykinase 1 | adipocytes | mouse       |
| nm_002591 | pepck | phosphoenolpyruvate carboxykinase 1 | kidney     | human       |
| nm_011044 | pepck | phosphoenolpyruvate carboxykinase 1 | kidney     | mouse       |
| nm_002591 | pepck | phosphoenolpyruvate carboxykinase 1 | liver      | human       |
| nm_011044 | pepck | phosphoenolpyruvate carboxykinase 1 | liver      | mouse       |
+-----------+-------+------------------------------------+------------+-------------+
```

Figure 9.3.37 Results to the SQL query shown in Figure 9.3.35.

More complex queries can be created using subselects and group functions like `intersect` *and* `minus`*. The* `minus` *function removes entries from the top* `select` *that are found in the bottom* `select`*. The* `intersect` *function returns only those entries that are found in both* `select` *statements. These functions, available in many other DBMS, are scheduled to be included in release 4.1 of MySQL. With MySQL, the query will have to be broken down into subqueries (e.g., "What genes are found in human?" "What genes are found in mice?"). The results of each subquery must then be processed and compared programmatically using a language like Perl.*

```
select distinct s.name, s.defn
  from seq s,
       seq2library s2l,
       library l,
       organism o
 where s2l.seq_id       = s.seq_id
   and s2l.library_id   = l.library_id
   and l.organism_id    = o.organism_id
   and o.common_name    = 'human'
minus
select distinct s.name, s.defn
  from seq s,
       seq2library s2l,
       library l,
       organism o
 where s2l.seq_id       = s.seq_id
   and s2l.library_id   = l.library_id
   and l.organism_id    = o.organism_id
   and o.common_name    = 'mouse'
 order by 1, 2;
```

Figure 9.3.38 The SQL query to find all genes in humans that are not found in mice. This query uses the group function `minus` which returns all entries in the first query that are not found in the second query. This function along with `intersect` and subselects are scheduled to be implemented in version 4.1 of MySQL.

```
NAME    DEFN
------  ----------------------------------
cetp    Cholesteryl ester transfer protein
```

Figure 9.3.39 Results to the SQL query shown in Figure 9.3.38 obtained from a database implemented in Oracle.

COMMENTARY

Background Information

As mentioned in the introduction, there are many methodologies that can be used to design a database schema and no one is the best for every type of database. A large database can contain areas, called "domains," which have been designed using different methodologies; a database to hold gene annotation is a perfect example of this. The tables to hold the evidence for the different pieces of annotation are usually designed using E-R modeling while the hybrid method is used to design the tables to hold the annotation itself and for relating the annotation to the evidence. This hypothetical gene annotation database could also be part of a larger data warehouse that brings together a wide range of biological knowledge.

Entity-Relationship modeling

Entity-Relationship (E-R) modeling is the standard method taught in database classes (Codd, 1990; Date, 1995; Yarger et al., 1999). An entity is any fundamental topic that is to be stored in the database. In the genomics world, an entity might be `tissue`, `organism`, or `gene`. An entity can have properties that describe it; for example, `genus` and `species` are properties of the entity `organism`. A relationship describes how the entities go together and is frequently considered to be a property of an entity. There are four types of relationships as shown in Figure 9.3.40.

There are two levels to E-R modeling: conceptual and physical. The conceptual level treats the entities as abstract objects while the physical level treats them as components of physical tables that will be implemented using

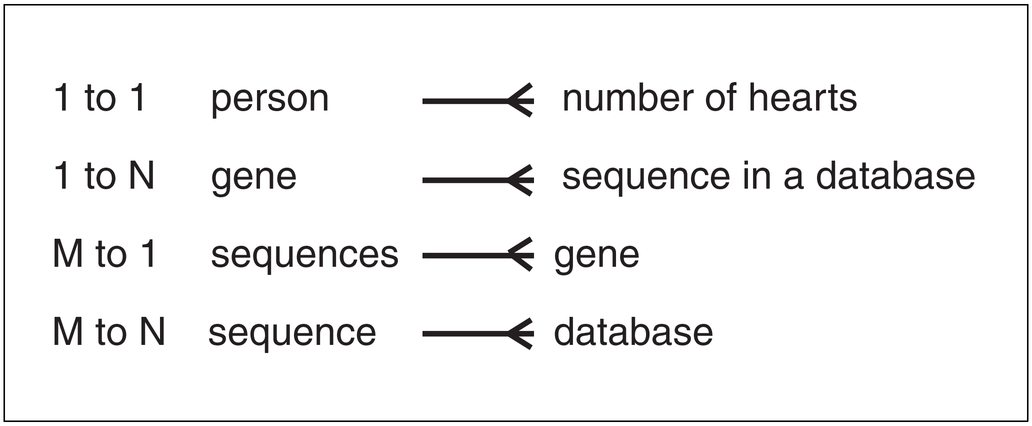

1 to 1 person number of hearts
1 to N gene sequence in a database
M to 1 sequences gene
M to N sequence database

Figure 9.3.40 The four types of relationships: 1-to-1, 1-to-N, M-to-1, and M-to-N. The crows-feet indicate the item with the multiplicity. There are many more symbols used, these are the most common.

SQL `create table` statements. Most professional modelers tend to bounce back and forth between them: thinking of the tables while working on the conceptual entities and vice-versa.

E-R modeling works well if the information is well understood, if the relationships between the different types of information are well understood, and if only a few types of questions will be asked. This method should also be considered if data needs to be loaded at a high rate, e.g., data captured off of a piece of experimental equipment as the experiment is running.

An example of well-understood information and relationships is sequence similarity results from using a BLAST search (Altschul et al., 1990; *UNITS 3.3 & 3.4*). There are three categories of information associated with BLAST results. The first category is the data needed to set up the search (e.g., the query sequence, the database searched, the BLAST program, and date of the search). The second category is the summary of the result (i.e., the summary section of a BLAST report). The third category is the detailed result (i.e., the high-scoring pairs, HSPs). Since a computer program takes in information and generates the results, the relationships between the different categories will never change unless the program itself is changed. A database schema to hold BLAST results could have just three tables, one for each category.

E-R modeling is designed to reduce data redundancy, if not eliminate it completely. There are two basic types of data redundancy: (1) redundancy within a table and (2) redundancy in the results of a query that combines multiple tables. Rules have been developed to eliminate data redundancy; these rules are called normalization. The first three rules cover most of the database redundancy problems a designer will encounter. These three rules are: (1) 1st Normal Form (1NF), where there can be no duplicated rows in the table; each cell in a table is single valued, there can be no repeating groups or arrays; and a column can contain only one type of information. (2) 2nd Normal Form (2NF), all the characteristics of 1NF in addition to all non-primary key columns must describe every column in the primary key and there can be no partial dependencies. (3) 3rd Normal Form (3NF), all the characteristics of 2NF in addition to the information in each non-primary key column must be independent of the information in any other non-primary key column in the same table.

To ensure that data used in more than one table is used consistently; columns in pairs of tables can be set up in parent-child relationships called a foreign key. Foreign keys maintain data integrity between tables, the only values that can be entered into a foreign keyed column(s) in the child table are those that are in the parent. Thus, data must be loaded into the parent before it can be loaded into the child. The reverse is true when deleting data. A foreign key can be a single or multiple columns. The number, type, and order of the child columns involved must be the same as found in a unique index in the parent table. The index does not have to be the primary key, just a unique index.

There are two major drawbacks to databases designed using E-R modeling. First is that the tables tend to be tightly related; adding new entities can potentially necessitate a complete redesign of the schema. Second, queries involving many large tables often run slowly, even if only one column is used in some of the tables. The speed of a search is especially important for data mining queries.

The hybrid method

Data warehousing

To address the data mining problem, another method was developed called data warehousing, along with its smaller cousin the data-mart (Inmon et al., 1995; Kimball, 1996; Dodge and Gorman, 1998). A data warehouse consists of a central table called the fact table that is supported by additional tables, the dimension tables, in a form that looks somewhat like a star. Accordingly, queries of data warehouses have been called "star queries."

A dimension table contains information that describes a narrowly defined entity; all of the attributes of an entity are placed in that entity's dimension table. Unlike E-R modeling, normalization is not used and redundancy is dealt with, even accepted, within the dimension table itself through a combination of logic in the program that loads the data and liberal use of unique indices. Since data duplication is acceptable in a dimension table, each row is identified with a numeric primary key. A dimension table can be thought of as "short and wide" reflecting the large number of attributes and few entries that are often found.

In contrast, the fact table brings together the primary keys of each dimension table these columns make up the primary key of the fact table, each column of which is also a foreign key back to the appropriate dimension table. Other data that may be used in a query may also be included in a fact table. A fact table can be thought of as "long and thin" since it contains lots of rows and few columns. Arbitrary data-mining queries involving many tables run very fast since (1) the meat of the information is contained in the dimension tables, (2) the relationships between the tables are numeric and indexed, and (3) there is only one intermediate table between any pair of dimension tables.

The fact table is designed based on the anticipated questions that may be asked. As a result, from a scientist's point of view, information may be brought together in the fact table that has no biological relationship. In addition, many entities encountered in biology are for the most part not very well understood and the relationships even less so. Thus, a database schema is needed to sequester each area of uncertainly into its own table and minimizes the interrelationships between the tables. Such a database should allow new tables to be easily added, old tables dropped, and the data in table completely revamped with only a minimal impact on the rest of the database and any computer programs that query the database.

To create such a database, a method was developed that brings some aspects of E-R modeling into data warehousing; this is the hybrid method. In the hybrid method, both the dimension and fact tables are narrowly defined and there are frequently many fact tables that relate information between only two or three dimension tables. The hybrid method, therefore, gives the advantage of a rapidly changeable database.

The hybrid method

As in E-R modeling, the first step in designing a database using the hybrid model is to identify the entities to be stored. There are two types of entities: abstract entities and concrete entities. Abstract entities are those terms that biologists use on a daily basis that are hard to define concretely in computer terms; `gene` is one of the most familiar abstract entities encountered. Abstract entities are defined only within the scope of the information stored in the database that describes it and are often the focus of the database or a domain of the database. Concrete entities, on the other hand, can be easily defined at some level and are frequently described using controlled language. The Gene Ontologies (The Gene Ontology Consortium, 2000; *UNIT 7.2*) are examples of controlled language; each ontology (biological process, cellular component, and molecular function) is a separate entity. Each entity has its own table; each table is set up with a numeric primary key and columns that contain non-entity properties of that entity. The more narrowly defined an entity is the better; an entity table has the general form shown in Figure 9.3.20.

After defining the entities, the second step is to identify which entities are related; a relationship table has the general form shown in Figure 9.3.24. Concrete entities are often properties of abstract entities.

A simple example contrasting entity and non-entity properties is foot size versus shoe size. A person's foot can be any size; this is an example of uncontrolled language. Shoes, however, come in predefined sizes; this is an example of controlled language. Within the definition of a hybrid database, foot size is a non-entity property and would be contained within a table for the entity `foot`. `Shoe size`, however, would have its own table containing all possible shoe sizes and would be related to `foot` through a relationship table `foot to shoe size`. This relationship table would contain only two columns, foot id and shoe size id, both part of the primary key for the table, which would be foreign keys back to the tables foot and shoe size, respectively.

Just as in data warehousing, data redundancy issues are handled within each entity table independently using programming logic in the input program and the liberal use of unique indices as appropriate. Unlike data warehousing, the relationship tables rarely hold any information beyond the numeric row identifiers of the entity tables in the relationship and are biologically descriptive. Thus, relationship tables are more like the E-R tables that handle many-to-one and many-to-many relationships. Hybrid databases readily handle queries containing many tables; ten tables in a query is not unusual.

Additional types of information can easily be added to a hybrid database by creating new entity and relationship tables, or adding to existing tables, to link them into the database. Tables holding computational analysis results designed using E-R modeling can be easily linked in with a relationship table. In addition, whole domains each with a different focus, e.g., sequence annotation, pathway information, expression data, etc., can each be designed using the hybrid model and then linked to each other again using relationship tables.

Like a data warehouse, these databases take in data slowly mainly due to the data checking that must be done by the loading program. In general, this fits the genomics world well since data is usually added on a daily or weekly basis rather than a second-by-second basis.

Critical Parameters and Troubleshooting

Externally versus internally controlled data

Data controlled by a person or process external to the database can change at any time. These changes can cause many headaches if the data is used to link tables together. Regardless whether E-R modeling or the hybrid method is used, it is a good idea to use a numeric representation of externally controlled data within the database itself. The hybrid method does this automatically, but it must be done explicitly E-R modeling. For example, imagine a database that uses a user ID comprised of a person's last name plus first initial. In this hypothetical database, this user ID is a primary key in a user table and is a property in tables containing information about experiments run, results, etc., with the appropriate foreign keys set up to link the child tables back to the user tables. If a person changes their last name, the user ID will have to be changed in many tables. Many database products do not have a "cascade on update" feature, this feature propagates a change to a field in a parent table throughout the child tables. Thus, either the foreign key constraints must be dropped and then the changes made to each table individually, or a new record must be made in the parent table, new records created in each child table, the old records deleted from the child tables, and then the old record deleted from the parent table. As one can imagine, this can get very complicated for a large database. In addition, the person may still be making additions to the database while the change to the user id is being made! Using an internal ID that is completely under the database's control can eliminate this problem; the person's name would only have to be changed in one place.

It may require some practice to develop a well-designed database, but even a poorly designed database can be a great aid in answering interesting and complicated questions. Of the two modeling methods, the hybrid model is the easiest and most general method to use and database schemas resulting from it are the easiest to expand. In either case, if redundant data is found to exist in a query, then further analysis is needed in the design of the database.

Literature Cited

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. 1990. Basic local alignment search tool. *J. Mol. Biol.* 215:403-410.

Codd, E.F. 1990. The Relational Model for Database Management; Version 2. Addison-Wesley Publishing, New York.

Date, C.J. 1995. An Introduction to Database Systems, 6th ed. Addison-Wesley Publishing, New York.

Dodge, G. and Gorman, T. 1998. Oracle8 Data Warehousing. John Wiley & Sons, New York.

The Gene Ontology Consortium. 2000. Gene ontology: Tool for the unification of biology. *Nature Gen.* 25:25-29

Inmon, W.H., Imhoff, C., and Battas, G. 1995. Building the Operational Data Store. John Wiley & Sons, New York.

Kimball, R. 1996. The Data Warehouse Toolkit. John Wiley & Sons, New York.

Yarger, R.J., Reese, G., and King, T. 1999. MySQL & mSQL. O'Reilly & Associates, CA.

Robert M. Peitzsch
Pfizer Global Research and Development
Groton, Connecticut

Using Relational Databases for Improved Sequence Similarity Searching and Large-Scale Genomic Analyses

As protein and DNA sequence databases have grown, characterizing evolutionarily related sequences (homologs) through sequence similarity has paradoxically become a more challenging endeavor. In the early 1990s, a similarity search might identify a dozen homologs only once in three searches; many searches would reveal only one or two homologs, if any. With today's comprehensive sequence libraries, most similarity searches will identify several dozen homologous sequences, and many searches will yield hundreds of homologs from dozens of species. As scientifically interesting as these results may be, they are often impractical to organize and analyze manually. Moreover, modern genome-scale studies do 1,000 to 10,000 searches in a single analysis, producing millions of lines of comparison results. Fortunately, relational databases (*UNITS 9.1 & 9.2*) can manage large sets of search results, greatly simplifying genome-scale analyses—for example identifying the most conserved sequences shared by two organisms, or the proteins that are found in plants but not animals. Relational databases are designed to integrate diverse types of information: e.g., sequence, taxonomy, similarity to other proteins, and gene location. Relational databases can also make searches more efficient by focusing on subsets of the protein databases—proteins found in similar organisms or with similar functions. Thus, relational databases are not only essential for the management and analysis of large-scale sequence analyses, but can also be used to improve the statistical significance of similarity searches by focusing the search on subsets of sequence libraries most likely to contain homologs, based, e.g., on taxonomy, structure, or function.

The protocols in this unit use relational databases to improve the efficiency of sequence similarity searching and to demonstrate various large-scale genomic analyses of homology-related data. Basic Protocol 1 illustrates the installation and use of a simple protein sequence database, `seqdb_demo`, which will be used as a basis for all the other protocols. Basic Protocol 2 then demonstrates basic use of the `seqdb_demo` database to generate a novel sequence library subset. Basic Protocol 3 shows how to extend and use `seqdb_demo` for the storage of sequence similarity search results. Basic Protocols 4 to 6 make use of various kinds of stored search results to address three different aspects of comparative genomic analysis. All of the SQL statements used in these protocols are available in the `seqdb_demo` package, described in Basic Protocol 1. While many of the SQL statements are briefly explained in each protocol, the concepts in Basic Protocols 2 to 4 will easier to understand if the reader is familiar with basic SQL (*UNIT 9.2*).

BASIC PROTOCOL 1

INSTALLING AND POPULATING THE `seqdb_demo` RELATIONAL DATABASE

In this protocol, a very simple protein sequence database, `seqdb_demo` (Fig. 9.4.1) will be installed and then populated with data obtained from a "flat-file" sequence library. The database includes: (1) a table for the raw sequence data; (2) a table to hold information about the sequence, including its description and various public database accession numbers; and (3) tables to store taxonomic information about the organism from which the sequence was obtained, and how those organisms are themselves related to each other.

Sequence and annotation information are loaded from a sequence library "flat file" into an empty `seqdb_demo` database using the Perl program `load_seqdemo.pl`, found in

Contributed by Aaron J. Mackey and William R. Pearson

Current Protocols in Bioinformatics (2004) 9.4.1-9.4.25

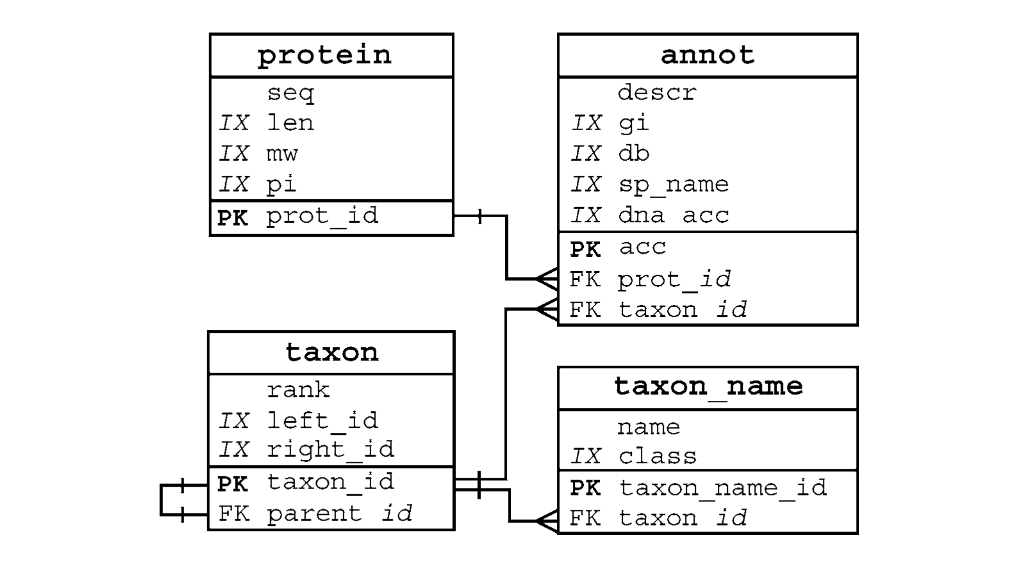


Figure 9.4.1 A schema for protein sequence data. Each of the boxes represents one of the tables in the `seqdb_demo` database. Sequences are stored in the `protein` table, their descriptions and accession information are stored in the `annot` table, and taxonomic information is stored in the `taxon` and `taxon_name` tables. The links between the tables are shown with dashed lines. The symbols at the ends of the line indicate the type of relationship; e.g., the `protein:annot` relationship is a *one-to-many* relationship; each protein sequence can have *many* descriptions or annotations but an annotation refers to only *one* protein sequence. The abbreviations to the left of the table entry names indicate whether the entry is a primary key (PK) or foreign key (FK, a foreign key in one table is a primary key in another, and allows the information in the two tables to be “joined”), or if the entry is indexed (IX) for rapid lookup.

the `seqdb_demo.tar.gz` package. Although the comprehensive `nr` protein sequence library from the NCBI will be used, any FASTA-formatted database (*APPENDIX 1B*) can be used, provided that descriptions follow the NCBI nonredundant DefLine format, e.g.:

```
>gi|15241446|ref|NP_196966.1| (NM_121466) putative protein
[Arabidopsis thaliana]ˆAgi|11281152|pir||T48635
hypothetical protein T15N1.110 — Arabidopsis
thalianaˆAgi|7573311|emb|CAB87629.1| (AL163792) putative
protein [Arabidopsis thaliana]
```

See *ftp://ftp.ncbi.nih.gov/blast/db/blastdb.txt* for further description of this specialized FASTA sequence format.

The protocol steps below demonstrate how to extract subsets of sequences from specific taxonomic groupings.

Necessary Resources

Hardware

Computer with at least 2 Gb of disk space available for the raw data flat-files and the MySQL sequence database files

Software

Windows- or Unix-based operating system (including Linux or Mac OS X)

Working version of MySQL, with functional database permissions. MySQL can be downloaded from *http://www.mysql.com* and installed as described in *UNIT 9.2*. All interactions with MySQL databases in these protocols will be via the `mysql` command-line utility.

A terminal application connected to a Unix environment in which one can execute Unix-like commands. For Windows-based operating systems, this entails installing the Cygwin Unix emulation (*http://www.cygwin.com*).

The Perl scripting language interpreter (any version since 5.005_03) and the `DBI`, and `DBD::mysql` modules. With Unix-like systems, the `DBI` and `DBD::mysql` modules can be installed from the CPAN Perl software repository with the following commands (typed input indicated in bold):

```
% perl -MCPAN -e shell
cpan> install DBI
cpan> install DBD::mysql
```

In some cases, it may be necessary to type **`force install DBD::mysql`** (at the `cpan` prompt) if errors are encountered (generally, these errors can safely be ignored). Under Windows-based operating systems, the `ppm` package management utility should be used instead to install both the `DBI` and `DBD::mysql` packages.

Files

The `seqdb_demo` package of SQL and Perl scripts for creating and maintaining a relational database of protein sequences, downloaded from *ftp://ftp.virginia.edu/pub/fasta/CPB/seqdb_demo.tar.gz*. This package includes all of the utilities to create, load, and maintain the simple protein sequence database described in these protocols.

A FASTA-format (*APPENDIX 1B*) “flat-file” protein sequence library, such as SwissProt or `nr`. These sequence libraries can be downloaded from *ftp://ftp.ncbi.nih.gov/blast/db/FASTA/swissprot.gz*, or *nr.gz*. The `nr` library is more comprehensive, but the SwissProt library is a smaller, more manageable dataset. In these protocols, the `nr` sequence library will be exclusively used.

Creating the seqdb_demo database

1. In a Unix terminal window, traverse into the directory in which the `seqdb_demo.tar.gz` package file was downloaded and execute the commands listed below (type the text in **`bold`** to issue the command; the computer response is in `lightface` and comments about the commands are written in *italics*).

```
% tar -xzvf seqdb_demo.tar.gz
```

Uncompresses and unpacks `seqdb_demo`.

```
% cd seqdb_demo
```

Changes directory into `seqdb_demo`.

```
% mysql < mysql/seqdb_demo.sql
```

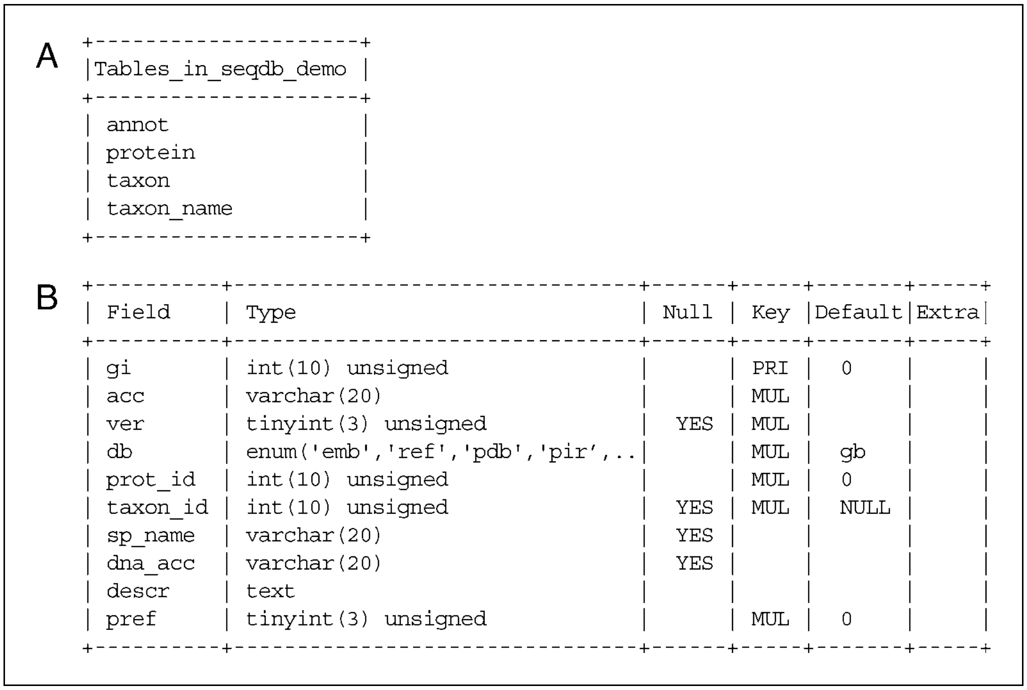
Creates the database and its tables.

Before executing the third command, one may wish to edit the top few lines of `mysql/seqdb_demo.sql` to change the user name, and password from the defaults (`seqdb_user and seqdb_pass`, respectively).

2. To confirm that the database has been created correctly (and to become familiar with the database’s schema), type the following:

```
% mysql -u seqdb_user -pseqdb_pass seqdb_demo
mysql> SHOW TABLES;
```

Provides a listing of the tables found in this database (Fig. 9.4.2A).



A

```
+---------------------+
|Tables_in_seqdb_demo |
+---------------------+
| annot               |
| protein             |
| taxon               |
| taxon_name          |
+---------------------+
```

B

```
+----------+-------------------------------+------+-----+-------+-----+
| Field    | Type                          | Null | Key |Default|Extra|
+----------+-------------------------------+------+-----+-------+-----+
| gi       | int(10) unsigned              |      | PRI |  0    |     |
| acc      | varchar(20)                   |      | MUL |       |     |
| ver      | tinyint(3) unsigned           | YES  | MUL |       |     |
| db       | enum('emb','ref','pdb','pir',..|     | MUL |  gb   |     |
| prot_id  | int(10) unsigned              |      | MUL |  0    |     |
| taxon_id | int(10) unsigned              | YES  | MUL |  NULL |     |
| sp_name  | varchar(20)                   | YES  |     |       |     |
| dna_acc  | varchar(20)                   | YES  |     |       |     |
| descr    | text                          |      |     |       |     |
| pref     | tinyint(3) unsigned           |      | MUL |  0    |     |
+----------+-------------------------------+------+-----+-------+-----+
```

Figure 9.4.2 (**A**) List of tables in the database created in Basic Protocol 1, step 1, retrieved via the **SHOW TABLES** command. (**B**) Description of columns in the database, retrieved via the **DESCRIBE annot** command.

```
mysql> DESCRIBE annot;
```

Gets a description of the columns in the annot *table (Fig. 9.4.2B).*

These commands confirm that one has successfully created the seqdb_demo *database with four tables, as described in Fig. 9.4.1. Briefly, the* protein *table will store raw protein sequences and the* annot *table (short for "annotation") will contain the description of the protein and any links to external public databases (SwissProt, Genpept, PIR, TrEMBL, etc.), while the other two tables (*taxon *and* taxon_name*) will provide taxonomic species information.*

Populating the seqdb_demo *database*

3. To load the sequences from the nr FASTA-format sequence library, type the following:

```
% gunzip /seqdata/nr.gz
```

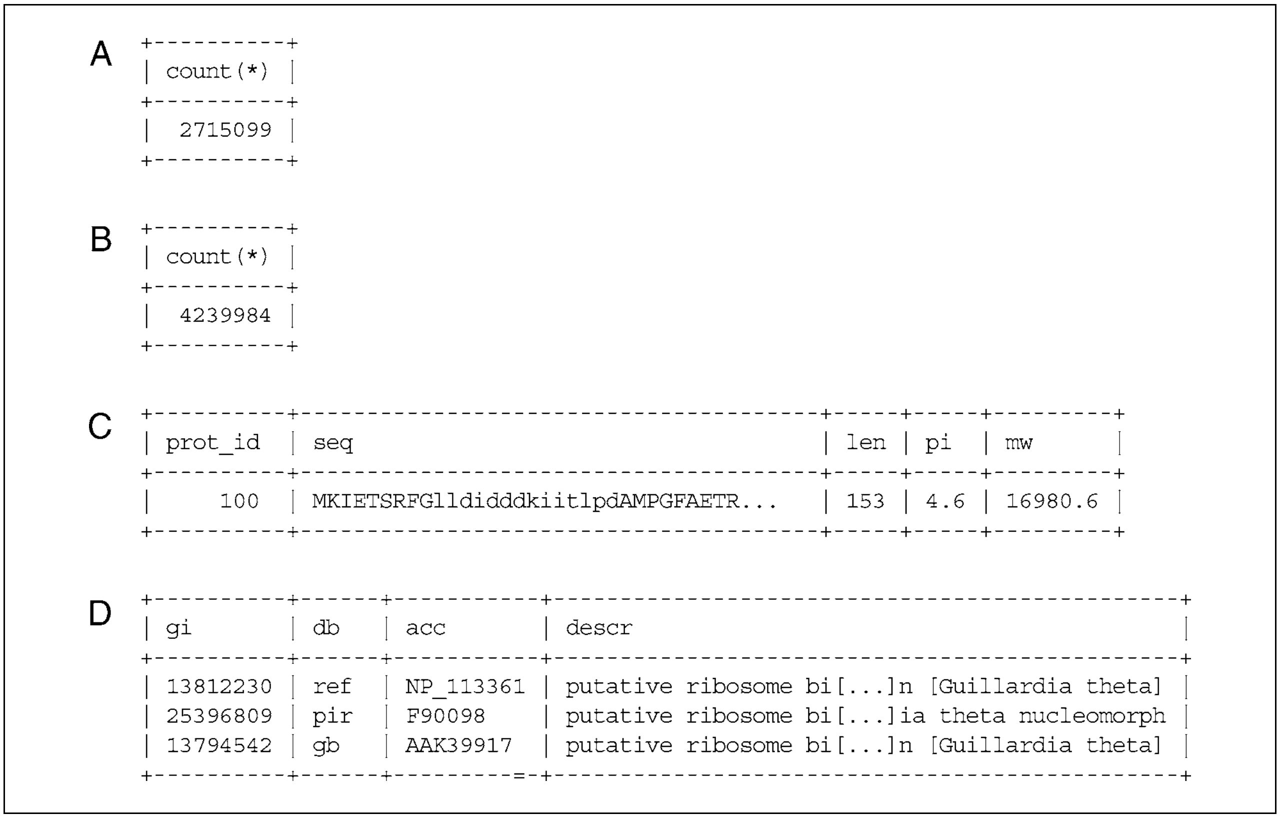
Uncompresses the file.

```
% load_seqdb.pl /seqdata/nr
```

Loads the data into the database.

In these commands, /seqdata should be changed to the directory of the compressed nr.gz file previously downloaded from *ftp://ftp.ncbi.nih.gov/blast/db/FASTA/nr.gz* (see Necessary Resources, above). The load_seqdb.pl script reads every sequence entry from the specified sequence library, storing the sequence data in the protein table and the header information in the annot table. For a large protein database like nr (which in March of 2004 contained nearly 2 million entries), this initial loading may take 6 to 12 hr.

4. To confirm that the database has successfully loaded the protein sequences and their annotations, type (from a MySQL prompt):



A
```
+----------+
| count(*) |
+----------+
|  2715099 |
+----------+
```

B
```
+----------+
| count(*) |
+----------+
|  4239984 |
+----------+
```

C
```
+----------+------------------------------------+-----+-----+---------+
| prot_id  | seq                                | len | pi  | mw      |
+----------+------------------------------------+-----+-----+---------+
|     100  | MKIETSRFGlldidddkiitlpdAMPGFAETR...  | 153 | 4.6 | 16980.6 |
+----------+------------------------------------+-----+-----+---------+
```

D
```
+----------+------+-----------+--------------------------------------------+
| gi       | db   | acc       | descr                                      |
+----------+------+-----------+--------------------------------------------+
| 13812230 | ref  | NP_113361 | putative ribosome bi[...]n [Guillardia theta] |
| 25396809 | pir  | F90098    | putative ribosome bi[...]ia theta nucleomorph |
| 13794542 | gb   | AAK39917  | putative ribosome bi[...]n [Guillardia theta] |
+----------+------+---------=-+--------------------------------------------+
```

Figure 9.4.3 (**A**) Number of protein sequences loaded into database from the `nr` sequence library, retrieved via the **`SELECT COUNT(*) FROM protein`** command. (**B**) Number of different descriptions loaded into the database from `nr`, retrieved via the **`SELECT COUNT (*) FROM annot`** command. (**C**) Information on a single protein, retrieved via the **`SELECT * FROM protein WHERE prot_id = 100`** command. (**D**) All annotations of a protein, retrieved via the **`SELECT gi, db, acc, descr`** command.

```
mysql> SELECT COUNT(*) FROM protein;
```

Reports the number of protein sequences (Fig. 9.4.3A).

```
mysql> SELECT COUNT (*) FROM annot;
```

Reports the number of different descriptions (Fig. 9.4.3B).

```
mysql> SELECT * FROM protein WHERE prot_id = 100;
```

Get a single protein (Fig. 9.4.3C).

```
mysql> SELECT gi, db, acc, descr
   +> FROM annot WHERE prot_id = 100;
```

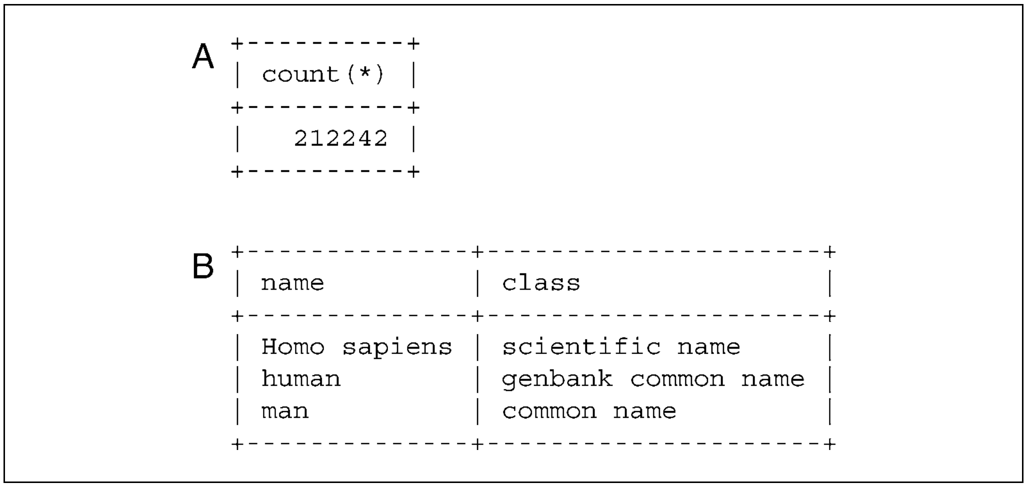
Get all annotations of a protein (Fig. 9.4.3D).

Because the `nr` *database is constantly growing, results may not exactly match those above.*

5. To add species taxonomic information to all of the protein sequence entries in the database, it is necessary to download information from the NCBI Taxonomy database. The `updatetax.pl` script automatically downloads this information and uses it to load the taxonomy-related tables in the `seqdb_demo` database. Type the following:

```
% mkdir /seqdata/taxdata
```

Makes a new directory for NCBI Taxonomy download.



A
```
+----------+
| count(*) |
+----------+
|   212242 |
+----------+
```

B
```
+--------------+---------------------+
| name         | class               |
+--------------+---------------------+
| Homo sapiens | scientific name     |
| human        | genbank common name |
| man          | common name         |
+--------------+---------------------+
```

Figure 9.4.4 (**A**) Total number of taxa loaded from the NCBI Taxonomy database, retrieved via the **SELECT COUNT(*) FROM taxon** command. (**B**) NCBI's ID for human, retrieved via the **FROM taxon_name WHERE taxon_id = 9606** command.

```
% updatetax.pl /seqdata/taxdata
```

Downloads and imports the NCBI Taxonomy database.

6. To confirm that the NCBI Taxonomy database was successfully loaded into the database, type the following commands:

```
mysql> SELECT COUNT(*) FROM taxon
```

Gets total number of taxa (Fig. 9.4.4A).

```
mysql> SELECT name, class

   +> FROM taxon_name WHERE taxon_id = 9606
```

Gets NCBI's ID for human (Fig. 9.4.4B).

Again, one may expect to see slightly different values, as the NCBI Taxonomy database continues to grow.

BASIC PROTOCOL 2

EXTRACTING SEQUENCES FROM seqdb_demo FOR SIMILARITY SEARCHING TO IMPROVE HOMOLOG SEARCHING

The inference of sequence homology is based on the identification of statistically significant sequence similarity. If an alignment between two sequences is statistically significant, one can reliably infer that the sequences are homologous. However, if the score is *not* significant, one cannot be certain the sequences are not homologous; in fact, many truly homologous proteins (where homology is inferred by significant structural similarity) do not share significant sequence similarity. The significance of an alignment is measured by the expectation value E, which describes the number of alignments of similar or better similarity that could be expected to occur by chance alone. The E value is calculated as $E = P \times D$, where P is the probability of seeing an alignment this good between any given pair of sequences and D is the total number of pairwise comparisons performed during the search. Therefore, one of the easiest ways to improve the sensitivity of a similarity search is to search a subset of sequence libraries, reducing D and improving the significance of all E values (nonhomologous alignments will continue to have E values $\approx$1.0 or greater). This strategy is particularly effective now that many complete prokaryotic and eukaryotic genomes and proteomes are available. For example, searching only against the proteins predicted from a complete genome instead of the entire nr sequence library, can improve the statistical significance of homologous alignments by 100 to 1000-fold, greatly enhancing the efficiency of the search for homologs in the given organism.

In addition, by searching against specific taxonomic subsets of a sequence library, one can tailor various scoring parameters to the evolutionary distance being considered. For example, modern mammals shared a common ancestor only about 100 million years ago, and so most mammalian orthologs share modestly high protein sequence identity (70% to 85%, on average). The BLOSUM50 scoring matrix (the default for FASTA), or BLOSUM62 scoring matrix (the default for BLAST), is "tuned" to be able to identify distant homologs that share less than 30% identity over long regions, but in return may not be able to reliably identify shorter homologies that have high identity. Conversely, the PAM40 matrix is targeted to sequences that share approximately 70% identity, and thus should be more effective at identifying and accurately aligning mammalian orthologs, particularly those that are too short to identify using the default matrices. Gap penalties can be similarly adjusted to be more or less forgiving, based on the approximate evolutionary distance between library and query sequences.

There are many other motivations for wanting to search against smaller subsets of available sequences. The most general strategy for searching against a taxonomic (or other) subset of a larger sequence database is to use the fully populated `seqdb_demo` database to generate customized, FASTA-formatted sequence libraries. This protocol will demonstrate how to generate both species-specific and clade-specific sequence database flat files from the `seqdb_demo` relational database.

Necessary Resources

Hardware

Computer with at least 2 Gb of disk space available

Software

Windows- or Unix-based operating system (including Linux or Mac OS X)

Working version of MySQL, with functional database permissions. MySQL can be downloaded from *http://www.mysql.com* and installed as described in *UNIT 9.2*. All interactions with MySQL databases in these protocols will be via the `mysql` command-line utility.

A terminal application connected to a Unix environment in which one can execute Unix-like commands. For Windows-based operating systems, this entails installing the Cygwin Unix emulation (*http://www.cygwin.com*).

The Perl scripting language interpreter (any version since 5.005_03) and the `DBI`, and `DBD::mysql` modules

Files

Generated as in Basic Protocol 1

1. Complete Basic Protocol 1.

To generate a species-specific sequence library

2a. To generate a library of human sequences (or sequences from any other species for which the preferred scientific name is known), create a text file (e.g., `human.sql`, found in the `seqdb_demo` distribution) with SQL code (see *UNIT 9.2*) that generates the desired sequences. In this case the SQL code would be that shown in Figure 9.4.5.

3a. Once this file has been created and saved, use it as input to the `mysql` client with the following command:

```
SELECT  CONCAT(“>gi|”, gi, “|”, db, “|”, acc, “| ”, descr, “\n”, seq)
FROM    protein
        INNER JOIN annot USING (prot_id)
        INNER JOIN taxon_name USING (taxon_id)
 WHERE  annot.pref = 1
   AND  taxon_name.class = “scientific name”
   AND  taxon_name.name = “Homo sapiens”;
```

Figure 9.4.5 SQL code used to generate a library of human sequences (note the space following the fourth “|” symbol).

```
>gi|8924190|ref|NP_061004| hypothetical protein PRO2714
MQILRFSVYRSQPREHLFFTFLLKIHWGGIKFTPKKFAWAKSLQVPSENKILMIFFFFWL...
>gi|18585169|ref|XP_102310| hypothetical protein XP_102310
MGWGSNqqvvlcfllcqlYSSRLFFYggkklrvvkgvgDLQERQSGSCPEGGQGITNCDK...
```

Figure 9.4.6 FASTA-formatted human sequences, printed to `human.lib`.

```
% mysql -rN seqdb_demo < human.sql > human.lib
```

The –`r` flag tells mysql that the output should be left “raw,” so that the embedded newline characters, `\n`, will be correctly interpreted; the `-N` flag prevents mysql from printing any column names. Together, this command selects all human sequences (and their preferred annotations) from the `seqdb_demo` database and prints them to `human.lib`, already converted into FASTA format, e.g., Figure 9.4.6.

4a. The SQL command script in Figure 9.4.5 generates valid FASTA-formatted files, but the sequence is all on one line. This can be problematic for sequence analysis tools that read sequences line-by-line into small buffers. To reformat the database so that sequences are on multiple lines with a maximum length of 60, the `reformat.pl` Perl script is included in the `seqdb_demo` distribution.

```
% reformat.pl human.lib
```

To generate taxonomic subsets

The `updatetax.pl` script described in Basic Protocol 1 calculates additional information (the `left_id` and `right_id` values) that can be used to select entire taxonomic subgroupings of species, e.g., all mammals or all vertebrate species. These two `left_id`/`right_id` numbers have the useful property that any descendents of a taxonomic node will have `left_id`’s (and `right_id`’s) that are between the `left_id` and `right_id` range of all their parent node; this is referred to as a “nested-set” representation of the hierarchy, and can be used to select entire hierarchical subsets without recursion (Celko, 1999).

2b. Generate a library of mammalian sequences from `seqdb_demo`; to do so, create a file (e.g., `mammalia.sql`, found in the `seqdb_demo` distribution) with the SQL code shown in Figure 9.4.7.

3b. As in Step 3a, use this `SQL` script to generate the sequence library with the following command:

```
% mysql -rN seqdb_demo < mammalia.sql > mammalia.lib
```

4b. Reformat the library as in step 4a with the following command.

```
% reformat.pl mammalia.lib
```

```
SELECT  CONCAT(“>gi|”, gi, “|”, db, “|”, acc, “| ”, descr, “\n”,
seq)
FROM    protein
        INNER JOIN annot USING (prot_id)
        INNER JOIN taxon AS all_mammalia USING (taxon_id)
        INNER JOIN taxon AS mammalia
            ON (all_mammalia.left_id BETWEEN mammalia.left_id
                                     AND mammalia.right_id)
        INNER JOIN taxon_name USING (taxon_id)
 WHERE annot.pref = 1
   AND taxon_name.name = ‘Mammalia’
   AND taxon_name.class = ‘scientific name’;
```

Figure 9.4.7 SQL code used to generate a library of mammalian sequences from `seqdb_demo`.

To generate a BLAST-searchable taxonomic subset

The BLAST algorithms (*UNITS 3.3 & 3.4*) require sequence libraries to be specially formatted and indexed to accelerate searches. The NCBI-BLAST and WU-BLAST versions use the `formatdb` and `xdformat` utilities, respectively, to perform this reformatting. However, the NCBI-BLAST versions provide a mechanism to specify a subset of a sequence library (by GI numbers) without the generation of custom sequence libraries and reformatting.

5. Using the `formatdb` utility, reformat the `nr` database for use with NCBI-BLAST programs:

```
% formatdb -p T -i /seqdata/nr
```

6. Alter the `SELECT` line from the SQL script (in step 3a or step 3b) to select only `gi` numbers:

```
SELECT gi
FROM [...]
WHERE [...]
```

7. Execute the revised SQL:

```
% mysql -rN seqdb_demo < mammalia_gi.sql > mammalia.gi
```

8. Use this `GI` list file (specified with `-l`) for any BLAST search against the `nr` sequence library:

```
% blastall -p blastp -i query.fa -l mammalia.gi -d
/seqdata/nr
```

See UNITS 3.3 & 3.4 for further discussion of many of the commands and arguments used in the steps above.

STORING SIMILARITY SEARCH RESULTS IN seqdb_demo

BASIC PROTOCOL 3

Most sequence-similarity search programs produce human-readable, textual output. While this text has important information embedded within it—sequence descriptions, scores, alignment boundaries, etc.—it is not practical for an investigator to look at all the results when hundreds of homologies are detected, or when thousands of independent searches are run. To manage and make efficient use of large sets of search results, the data must be organized and indexed for easy querying and retrieval. Furthermore, the ratio of actual similarity and alignment data to white space and formatting text in the output is often fairly low, making the files easy to read, but much larger than necessary. Finally, keeping the search results in separate results files makes it more difficult to integrate search results with other information. This protocol addresses many of these problems by storing results from sequence similarity searches in the `seqdb_demo` relational database.

Every similarity-searching program—e.g., BLAST (*UNITS 3.3 & 3.4*), FASTA (*UNIT 3.9*), SSEARCH (*UNIT 3.10*), or HMMER—produces somewhat different similarity and alignment results. Some programs produce alignments with both gaps and frameshifts, while other programs may provide many separate alignment blocks (e.g., BLAST HSP's). To create a generic table structure able to store results from most similarity-search programs, the focus of this protocol will be on the common types of data produced by these programs; any data specific only to one algorithm will be ignored, and the BioPerl software will be used to extract these common data. In general, the programs perform many pairwise comparisons between one (or more) query sequence(s) and many entries in a sequence library, reporting only the most similar (or most significant) sequence comparisons. Each pairwise comparison produces an alignment with an associated raw score and statistical score (usually expressed as *bits*), as well as an overall estimate of the alignment's statistical significance (*E* value). Additionally, some alignment information, including the boundaries in the query and library sequences, the number and position of gaps, etc., is usually available. Finally, summary information such as percent identity and lengths of the two sequences may be provided.

Necessary Resources

Hardware

Computer with at least 2 Gb of disk space available

Software

Windows- or Unix-based operating system (including Linux or Mac OS X)

Working version of MySQL, with functional database permissions. MySQL can be downloaded from *http://www.mysql.com* and installed as described in *UNIT 9.2*. All interactions with MySQL databases in these protocols will be via the `mysql` command-line utility.

A terminal application connected to a Unix environment in which one can execute Unix-like commands. For Windows-based operating systems, this entails installing the Cygwin Unix emulation (*http://www.cygwin.com*).

The Perl scripting language interpreter (any version since 5.005_03) and the `DBI`, and `DBD::mysql` modules.

The BioPerl toolkit (*http://www.bioperl.org*; available via CPAN, see Basic Protocol 1) should be installed.

Files

The `seqdb_demo` package of SQL and Perl scripts for creating and maintaining a relational database of protein sequences, downloaded from *ftp://ftp.virginia.edu/pub/fasta/CPB/seqdb_demo.tar.gz*. This package includes all of the utilities to create, load, and maintain the simple protein sequence database described in these protocols.

Similarity search results from FASTA (*UNIT 3.8*), BLAST (*UNITS 3.3 & 3.4*), SSEARCH (*UNIT 3.10*), or HMMER

A sample set of similarity results is available from *ftp://ftp.virginia.edu/pub/fasta/CPB/ec_human.results.gz* to produce the file `ec_human.results`.

1. Complete Basic Protocol 1.

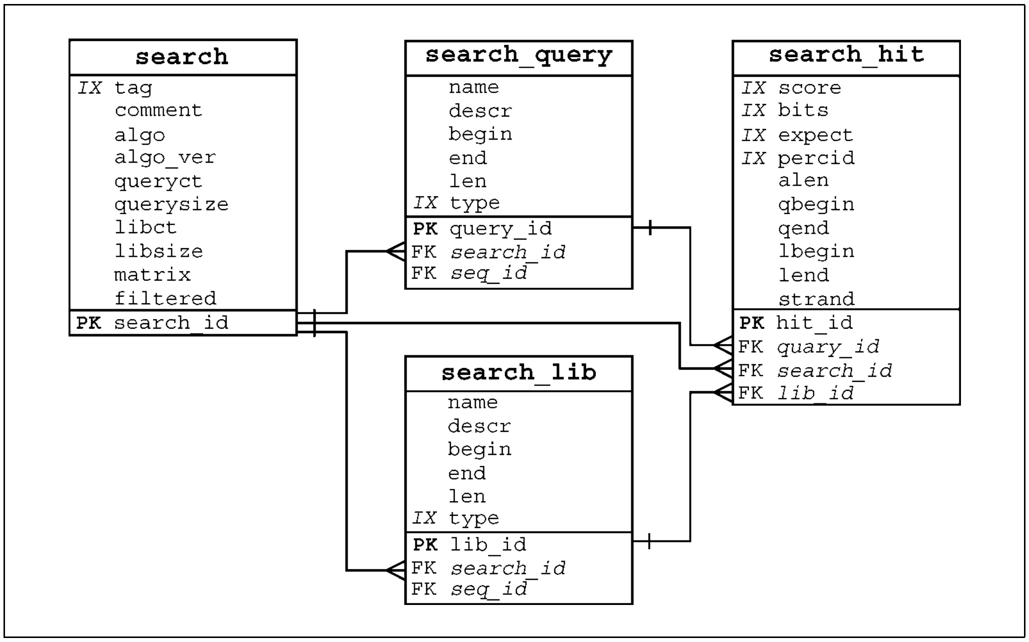


Figure 9.4.8 A schema for similarity search results. Each of the boxes represents one of the tables used to collect alignment data in the `seqdb_demo` database. The `search` table records the parameters of the search; `search_query` and `search_lib` record information about the query and library sequences used for the search, and the `search_hit` table records the scores and boundaries of alignments between query and library sequences. The links between tables, primary keys (PK), and foreign keys (FK) are indicated as in Figure 9.4.1.

Extending **seqdb_demo** *to include similarity search results*

2. An SQL script, `search.sql`, is included in the `seqdb_demo` distribution to add the tables related to sequence similarity search results:

```
% mysql seqdb_demo < mysql/search.sql
```

3. As in Basic Protocol 1, step 2, again execute `SHOW TABLES` and `DESCRIBE <table>` statements for each of the `search`, `search_query`, `search_lib`, and `search_hit` tables to confirm their existence in the database, and to become familiar with them (also see Fig. 9.4.8). Briefly, for any one set of similarity results, a single row will be stored in the `search` table, summarizing the search (algorithm used, parameters, etc.). Each query used for the search will be stored in the `search_query` table, while any library sequence reported in the search will be stored once in the `search_lib` table. Information about the alignments between any query and library sequences is stored in the `search_hit` table.

Importing similarity search results

4. Run the `loadsearch.pl` script, provided with the `seqdb.demo` distribution to parse and load the sequence similarity search data (e.g., `ec_human.results`) into the database:

```
% loadsearch.pl --format fasta --tag ecoli_vs_human \
--comment ‘E. coli vs human proteome’ < fasta.results
```

Similarity search results are imported into the database by parsing the raw text output and entering the sequence names, scores, and boundaries into the various search-related tables. The BioPerl toolkit provides functions for parsing BLAST, FASTA, and HMMER text results, among others, which are easily combined with Perl DBI database modules to store search results. The provided `loadsearch.pl` *script from* *the* `seqdb_demo` *distribution makes use of the BioPerl-based result parsers, so it theoretically should be able to accommodate any result formats that BioPerl can parse. Furthermore,* `loadsearch.pl` *assumes that all query and library sequences either (a) have the NCBI-like “DefLine” header ID found in the* `nr` *and similar flat files (e.g.,* `gi|123456|gb|CAA1128383.1`*), or (b) have a customized ID of the form* `table.field|key` *(e.g.,* `contig.contig_id|9876` *or* `annot.acc|X12983`*) that references a sequence obtainable via the provided table and key field. The key will be used in the* `seq_id` *field of the* `search_query` *and* `search_lib` *tables, and either* `GI` *or* `annot.acc`*, etc., will be used as the* `type`*.*

Additionally, the `FASTA`*-specific* `@C:1001` *syntax for defining the coordinate offset of the sequence (which, for this parser to work, must follow the ID) may also be included. An example entry might look like:*

```
mysql> SELECT COUNT(*)
    +> FROM search_query
    +> INNER JOIN search USING (search_id)
    +> WHERE search.tag = “ecoli_vs_human”;
+----------+
| COUNT(*) |
+----------+
|     4289 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT query_id, lib_id, score, expect, percid
    +> FROM search_hit
    +> INNER JOIN search USING (search_id)
    +> WHERE search.tag = “ecoli_vs_human”
    +> LIMIT 5;
+----------+--------+-------+--------+--------+
| query_id | lib_id | score | expect | percid |
+----------+--------+-------+--------+--------+
|        1 |      1 |    23 |    9.4 |      1 |
|        2 |      2 |    32 |   0.81 |  0.253 |
|        2 |      3 |    33 |    2.3 |  0.299 |
|        2 |      4 |    30 |    2.4 |  0.291 |
|        2 |      5 |    32 |    2.4 |  0.252 |
+----------+--------+-------+--------+--------+
5 rows in set (0.00 sec)

mysql> SELECT COUNT(*)
    +> FROM search_hit
    +> INNER JOIN search USING (search_id)
    +> WHERE search.tag = “ecoli_vs_human”
    +> AND search_hit.expect <= 1e-3;
+----------+
| COUNT(*) |
+----------+
|     8588 |
+----------+
1 row in set (1 min 1.18 sec)
```

Figure 9.4.9 SQL statements to confirm successful importing of results. Bold text represents input; lightface text represents output.

```
>contig.contig_id|9876 @C:1001 Fragment of assembled
contig
ACTAGCTACGACTACGATCAGCGACTACGAGCGCGCATCGAC ...
```

Finally, `loadsearch.pl` *also assumes that if the report contains multiple results from multiple queries, then the same library database and parameters were used in all searches (i.e., the* `search` *table data remains constant, and the entire result set is considered as one search execution, with multiple independent queries). The script expects to receive the report via STDIN, and to obtain the name "tag" and any descriptive commentary via command-line arguments.*

Confirm successful result importing

5. Execute a few basic SQL statements to check that the data has been successfully imported into the database (Fig. 9.4.9).

The result shown in Figure 9.4.9 further exemplifies the need to store similarity results in a relational database: manually examining and evaluating over 8500 statistically significant alignments is simply not feasible.

ANALYZING SIMILARITY SEARCH RESULTS: IDENTIFYING ANCIENT PROTEINS

BASIC PROTOCOL 4

Once the data from sequence similarity searches are stored in a relational database, it becomes possible to build "genome-scale" summaries that incorporate data about thousands of sequences almost as easily as reporting results from one or two searches. Once one has saved all the results of a large-scale sequence comparison (e.g., all *E. coli* protein sequences used as queries in searches against a database of human protein sequences), comprehensive summaries of the similarities between the proteins in two genomes can be generated with a few SQL statements. To illustrate, the authors of this unit searched all 4,289 *E. coli* K12 predicted proteins against approximately 40,000 human sequences from the `nr` database that are also found in the curated human IPI database, and saved the results in a `seqdb_demo` database as `ecoli-vs-human`. It is then possible to identify ancient genes—genes shared by human and *E. coli*, presumed to be present in the last common ancestor of bacteria and man.

Necessary Resources

Hardware

Computer with at least 2 Gb of disk space available

Software

Windows- or Unix-based operating system (including Linux or Mac OS X)

Working version of MySQL, with functional database permissions. MySQL can be downloaded from *http://www.mysql.com* and installed as described in *UNIT 9.2*. All interactions with MySQL databases in these protocols will be via the `mysql` command-line utility.

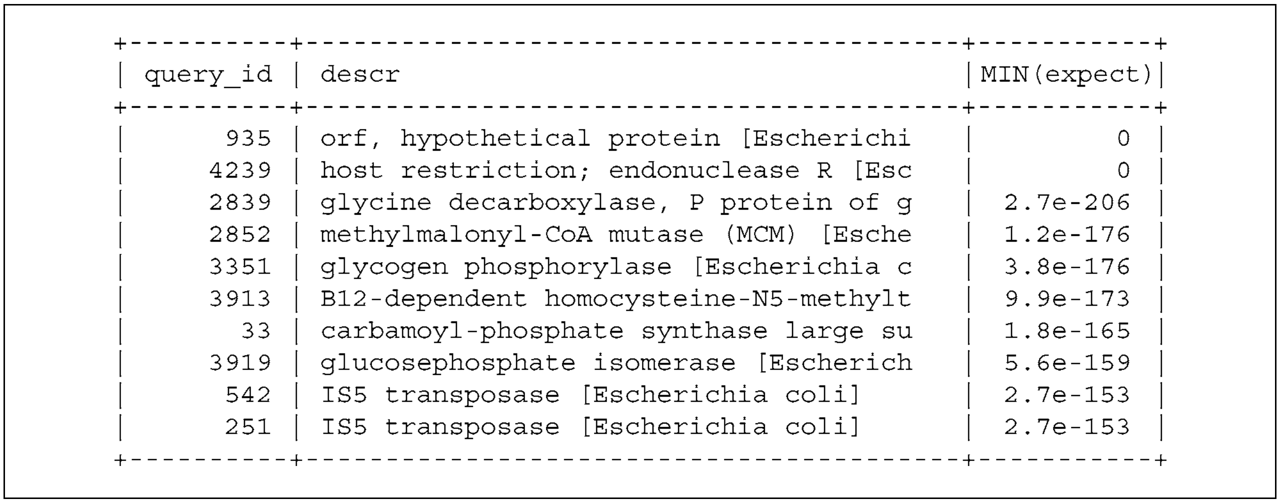
A terminal application connected to a Unix environment in which one can execute Unix-like commands. For Windows-based operating systems, this entails installing the Cygwin Unix emulation (*http://www.cygwin.com*).

The Perl scripting language interpreter (any version since 5.005_03) and the `DBI`, and `DBD::mysql` modules.

Files

Generated as in Basic Protocols 1 and 3

1. Complete Basic Protocols 1 and 3.



```
+----------+------------------------------------------+-----------+
| query_id | descr                                    |MIN(expect)|
+----------+------------------------------------------+-----------+
|      935 | orf, hypothetical protein [Escherichi    |         0 |
|     4239 | host restriction; endonuclease R [Esc    |         0 |
|     2839 | glycine decarboxylase, P protein of g    |  2.7e-206 |
|     2852 | methylmalonyl-CoA mutase (MCM) [Esche    |  1.2e-176 |
|     3351 | glycogen phosphorylase [Escherichia c    |  3.8e-176 |
|     3913 | B12-dependent homocysteine-N5-methylt    |  9.9e-173 |
|       33 | carbamoyl-phosphate synthase large su    |  1.8e-165 |
|     3919 | glucosephosphate isomerase [Escherich    |  5.6e-159 |
|      542 | IS5 transposase [Escherichia coli]       |  2.7e-153 |
|      251 | IS5 transposase [Escherichia coli]       |  2.7e-153 |
+----------+------------------------------------------+-----------+
```

Figure 9.4.10 List of highest-scoring *E. coli* homologs to human sequences, obtained via the commands shown in step 3 of Basic Protocol 4.

2. Once the search results are loaded (using `loadsearch.pl`, as described in Basic Protocol 3), a simple summary of the number of *E. coli* sequences that share significant similarity to human sequences can be produced:

```
mysql > SELECT COUNT(DISTINCT search_hit.query_id) AS
shared
-> FROM search_hit
-> INNER JOIN search USING (search_id)
-> WHERE search.tag = ‘‘ecoli-vs-human’’
-> AND expect < 1e-6;
```

This query returns a count of 926 *E. coli* sequences.

One could also ask the opposite question, how many human proteins have a significant match with E. coli, simply by changing the `DISTINCT query_id` *clause to* `DISTINCT lib_id`.

3. In addition to knowing the numbers of matches that obtain an *E* value less than `1e—6`, one might also like to identify the highest-scoring homologs. It is relatively easy to identify the *E. coli* sequences involved in the ten most significant (i.e., lowest *E* value) alignments between *E. coli* and human sequences:

```
mysql> SELECT search_hit.query_id, search_query.descr,
MIN(expect)
    -> FROM search
    -> INNER JOIN search_hit USING (search_id)
    -> INNER JOIN search_query USING (query_id)
    -> WHERE search.tag = ‘‘ecoli-vs-human’’
    -> GROUP BY query_id
    -> ORDER BY expect
    -> LIMIT 10;
```

To get the listing (Fig. 9.4.10) of *E. coli* sequences (rather than just the count), the `COUNT (DISTINCT search_hit.query_id)` clause from step 2 was replaced with a `GROUP BY query_id`; both statements ensure that *E. coli* proteins that match several human proteins will be counted only once.

```
SELECT   search_hit.query_id,  search_hit.expect,
         search_query.descr AS query_descr,
         search_lib.descr AS lib_descr
  FROM   search
         INNER JOIN search_hit USING (search_id)
         INNER JOIN search_query USING (query_id)
         INNER JOIN search_lib
             ON (search_hit.lib_id = search_lib.lib_id)
  WHERE expect IN (
         SELECT MIN(besthit.expect)
         FROM    search_hit AS besthit
           WHERE   besthit.query_id = hit.query_id
    )
   AND search.tag = “ecoli-vs-human”
ORDER BY expect ASC
LIMIT  10;
```

Figure 9.4.11 SQL statement to identify human sequences involved in alignments from from step 3 of Basic Protocol 4, for a database system that allows subselects (see step 4a of Basic Protocol 4).

4a. *For database systems that allow "subselects":* It is more difficult to identify the human sequences involved in each of these alignments because the `GROUP BY` clause used in step 3 means that all the rows from `search_hit` that share the same `query_id` have been collapsed; if one were also to request `search_hit.lib_id`, from which of the collapsed rows will the `lib_id` come? One might guess that the selected `lib_id` would be from the same row where the value of `expect` is equal to `MIN(expect)`, but, with SQL, *there is nothing that guarantees this to be true*. In a database system that allows "subselects" (SQL clauses that are themselves complete `SELECT` statements), one could instead do something like what is illustrated in Figure 9.4.11. Note that in this solution, multiple rows may be obtained for a given query, if the best hits happen to share the same expectation value (e.g., an `expect` of `0`).

4b. *For database systems that do not allow "subselects":* Versions of MySQL prior to 4.1 lacked "subselect" capability; getting the related hit information without subselects is a bit more complicated, but demonstrates a useful approach. A temporary intermediate table is first created to store the `hit_id` and `query_id` values for the rows of interest (i.e., the `hit_id` corresponding to the row or rows having `MIN(expect)` for each `query_id`). Because the aggregate functions `MIN` and `MAX` only operate on the first numeric value found in an entry, the trick to getting valid `hit_id`'s is to embed each `hit_id` in a string that also contains the numeric log-transformed *E* value, separated by white space. One can then extract the `hit_id` that corresponds to `MIN(expect)` [or `MAX(-LOG(expect)`, as the case may be] from the aggregate function's result (see `ancient.sql`, found in the `seqdb_demo` distribution), using the statement shown in Figure 9.4.12. The intermediate `besthits` table (Fig. 9.4.13) can now be used to retrieve only the rows of interest. For instance, the script shown in Figure 9.4.14 produces a list of the ten best matches between *E. coli* and human proteins, excluding any obvious transposase insertion sequences.

These SQL queries show that there are many very highly conserved proteins shared by both E. coli and humans; because these genes have shared ancestry, they must have been present in the last common ancestor of bacteria and humans.

```
DROP TABLE IF EXISTS besthits;
CREATE  TEMPORARY  TABLE  besthits
SELECT  query_id,
        SUBSTRING(
          MAX(CONCAT(RPAD(LPAD(
            FLOOR(IF(expect = 0,
                    10000,       -- special case for E value of 0
                    -LOG(expect) -- turn E value into whole number
                   ) * 10000
                 ), 9, " "), 30, " "), hit_id)),31
        ) AS hit_id
FROM    search
        INNER JOIN search_hit USING (search_id)
WHERE   search.tag  =  “ecoli-vs-human”
GROUP  BY  query_id;
```

Figure 9.4.12 SQL statement to identify human sequences involved in alignments from step 3 of Basic Protocol 4, for versions of MySQL that do not allow subselects (see step 4b of Basic Protocol 4).

```
+----------+------+------+----------------------------------+------------------+
| E()      | %ID  | alen | query_descr                      | lib_descr        |
+----------+------+------+----------------------------------+------------------+
| 6.3e-207 | 53.3 |  953 | glycine decarboxylase, P protein o| Glycine dehydroge|
| 1.3e-177 | 59.3 |  708 | methylmalonyl-CoA mutase (MCM)    | Methylmalonyl-CoA|
| 4.8e-177 |   50 |  812 | glycogen phosphorylase            | Glycogen phosphor|
| 5.2e-173 | 54.2 | 1252 | B12-dependent                     | 5-methyltetrahydr|
|   8e-166 | 41.2 | 1046 | carbamoyl-phosphate synthase large| Carbamoyl-phospha|
| 1.4e-160 | 64.8 |  549 | glucosephosphate isomerase        | Glucose-6-phospha|
|   7e-144 | 52.9 |  867 | aconitate hydrase 1               | Iron-responsive e|
| 5.2e-141 | 60.5 |  626 | chaperone Hsp70; DNA biosynthesis;| Stress-70 protein|
| 1.6e-135 | 71.7 |  466 | membrane-bound ATP synthase, F1 se| ATP synthase beta|
| 2.7e-121 | 55.4 |  554 | succinate dehydrogenase, flavoprot| Succinate dehydro|
+----------+------+------+----------------------------------+------------------+
```

Figure 9.4.13 Intermediate `besthits` table produced by SQL from Figure 9.4.14.

```
SELECT   expect  AS 'E()',
         percid*100 AS '%ID', alen,
         search_query.descr AS query_descr,
         search_lib.descr AS lib_descr
FROM   search
       INNER  JOIN  search_query  USING  (search_id)
       INNER  JOIN  besthits  USING  (query_id)
       INNER  JOIN  search_hit  USING  (hit_id)
       INNER  JOIN  search_lib  USING  (lib_id)
WHERE  search_query.descr NOT LIKE  "%IS5 transposase%"
AND  search_query.descr  NOT LIKE "%IS2  hypo%"
AND  search.tag = “ecoli-vs-human”
ORDER  BY  expect ASC
LIMIT  10;
```

Figure 9.4.14 Script used to produce a list of the ten best matches between *E. coli* and human proteins from the intermediate `besthits` table shown in Figure 9.4.13.

ANALYZING SIMILARITY SEARCH RESULTS: TAXONOMIC SUBSETS

BASIC PROTOCOL 5

One can generalize the genome-genome comparison from Basic Protocol 4 to determine a taxonomic distribution (i.e., the presence or absence in a given species or taxonomic clade) for any gene of interest. In this protocol, sequence similarity searches will be used against a database such as that described in Basic Protocol 1, where species information is available for each sequence. For any library sequence identified, it is possible to use the `seq_id` field from the `search_lib` table to look up `taxon_id` values from the `annot` table. The goal is to generate a summary table of gene counts that reflect various taxonomic subsets, i.e., the number of genes that have homology with proteins in *Bacteria*, *Archaea*, and *Eukaryota*, or only with proteins found in *Bacteria* (but not *Archaea* or *Eukaryota*), or only with proteins found in *Archaea*, or with proteins found in both *Bacteria* and *Archaea* but not *Eukaryota*, etc. Although the relational database concepts required to generate the summary table are a bit more complex than in the examples given elsewhere in this unit, which involve "joining" only a handful tables, the SQL shown in this protocol demonstrates how relational databases can provide summaries of datasets where the data must satisfy many conditions.

The data for this example come from a sequence similarity search of all 4289 *E. coli* K12 proteins against the entire NCBI `nr` database. The goal is to generate the necessary data to create a summary table, shown in Table 9.4.1. Note that this protocol is not intended to obtain knowledge about matches that occurred to other *E. coli* proteins already in the database, only to homologs in bacterial species other than *E. coli*. Thus, the last line in the table demonstrates that 355 *E. coli* proteins have no known homologs in any other species.

Necessary Resources

Hardware

Computer with at least 2 Gb of disk space available

Software

Windows- or Unix-based operating system (including Linux or Mac OS X)

Working version of MySQL, with functional database permissions. MySQL can be downloaded from *http://www.mysql.com* and installed as described in UNIT 9.2. All interactions with MySQL databases in these protocols will be via the `mysql` command-line utility.

Table 9.4.1 Taxonomic Distribution of *E. coli* Homologs

Eukaryota	Archaea	Bacteria	Totals
+	+	+	893
+	−	+	**661**
−	+	+	394
+	+	−	0
−	−	+	1986
−	+	−	0
+	−	−	0
−	−	−	355
1560	**1289**	3934	4289

```
CREATE TEMPORARY TABLE temp_results (
   query_id  INT UNSIGNED NOT NULL DEFAULT 0,
   taxon_id INT UNSIGNED NOT NULL DEFAULT 0,
   PRIMARY KEY (query_id, taxon_id)
) TYPE = HEAP;

INSERT INTO temp_results ( query_id, taxon_id )
SELECT search_query.seq_id, kingdom.taxon_id
FROM   search
        INNER JOIN search_query USING (search_id)
        INNER JOIN search_hit USING (query_id)
        INNER JOIN search_lib USING (lib_id)
        INNER JOIN annot ON (search_lib.seq_id = annot.gi)
        INNER JOIN taxon USING (taxon_id)

        INNER JOIN taxon AS kingdom
             ON (taxon.left_id BETWEEN kingdom.left_id
                                   AND kingdom.right_id)
        INNER JOIN taxon_name
             ON (kingdom.taxon_id = taxon_name.taxon_id)

        INNER JOIN taxon AS ecoli
             ON (taxon.left_id NOT BETWEEN ecoli.left_id
                                       AND ecoli.right_id)
        INNER JOIN taxon_name AS ecoli_name
             ON (ecoli.taxon_id = ecoli_name.taxon_id)

WHERE   taxon_name.name IN ('Bacteria', 'Eukaryota', 'Archaea')
   AND  taxon_name.class = 'scientific name'
   AND  ecoli_name.name = 'Escherichia coli'
   AND  ecoli_name.class = 'scientific name'
   AND  search.tag = 'eco-vs-nr'
;

Query OK, 7125 rows affected (2 min 18.64 sec)
Records: 565464  Duplicates: 558339  Warnings: 0
```

Figure 9.4.15 SQL statement used to create a temporary intermediate results table to store the `taxon_id` of all species in which a homolog to each query was found (see step 2 of Basic Protocol 5). Bold text represents input; lightface text represents output.

A terminal application connected to a Unix environment in which one can execute Unix-like commands. For Windows-based operating systems, this entails installing the Cygwin Unix emulation (*http://www.cygwin.com*).

The Perl scripting language interpreter (any version since 5.005_03) and the `DBI`, and `DBD::mysql` modules

Files

Generated as in Basic Protocols 1 and 3

1. Complete Basic Protocols 1 and 3.

2. Create a temporary intermediate results table to store the `taxon_id` of all species in which a homolog to each query was found, using the SQL statement shown in Figure 9.4.15 (see `taxcat.sql`, found in the `seqdb_demo` distribution). For efficiency, specify that the table should exist only in memory (remove the `TYPE=HEAP` clause if the results do not fit into available memory). Having built this `temp_result` table, it can now be used for every combination of desired taxonomic subsets.

3. To generate the counts for genes found in *Bacteria* and *Eukaryota*, but not *Archaea*, generate a second temporary table, `excludes`, which contains the `query_id`'s of homologs in the *undesired* taxonomic subsets, using the SQL statement shown in Figure 9.4.16.

```
DROP TABLE IF EXISTS excludes;
CREATE TEMPORARY TABLE excludes TYPE = HEAP
SELECT DISTINCT(query_id) AS query_id
FROM   temp_results
       INNER JOIN taxon USING (taxon_id)
       INNER JOIN taxon_name
           ON (taxon.taxon_id = taxon_name.taxon_id)
WHERE  taxon_name.name NOT IN ('Bacteria', 'Eukaryota')
   AND taxon_name.class = 'scientific name';

Query OK, 1289 rows affected (0.42 sec)
Records: 1289  Duplicates: 0  Warnings: 0
```

Figure 9.4.16 SQL statement used to generate the temporary `excludes` table (see step 3 of Basic Protocol 5). Bold text represents input; lightface text represents output.

```
SELECT COUNT(temp_results.query_id) AS total, annot.descr
FROM   temp_results
       INNER JOIN taxon USING (taxon_id)
       INNER JOIN taxon_name
           ON (taxon.taxon_id = taxon_name.taxon_id
       INNER JOIN annot
           ON (temp_results.query_id = annot.gi)
       LEFT JOIN exclude
           ON (temp_results.query_id = exclude.query_id)
WHERE  exclude.query_id IS NULL
   AND taxon_name.name IN ('Bacteria', 'Eukaryota')
   AND taxon_name.class = 'scientific name')
GROUP BY temp_results.query_id
HAVING total = 2;
```

Figure 9.4.17 SQL statement used to select the count of rows in `temp_results` where the `query_id` appears, given the desired taxonomic subsets.

The `WHERE` *constraint in this query is equivalent to* `taxon_name.name` = `` `Archaea' ``*; therefore the number of records inserted (*`1289`*) is the total number of E. coli proteins that have homologs in Archaea (regardless of what other homologies there may be). These are the source of the column totals found at the bottom of the summary table.*

4. For each `query_id` not in `excludes`, select the *count* of rows in `temp_results` where the `query_id` appears, given the desired taxonomic subsets, using the SQL statement shown in Figure 9.4.17. If that count equals the number of taxonomic subsets, then that `query_id` satisfies the condition (note the `HAVING` clause that enforces this behavior).

   The number of rows that this query returns (`661`*; Fig. 9.4.18) is the number of genes that have hits against proteins in both Bacteria and Eukaryota species, but have no significant hits against proteins from Archaea species (the +/−/+ row in Table 9.4.1). Also, by joining the results back to the* `annot` *table, it is possible to see which genes have this taxonomic distribution.*

5. Repeat steps 3 and 4 for each taxonomic combination of interest (changing only the names of the taxa to include, and the `HAVING` clause to reflect the number of taxa) to generate the summary table. Note that the last combination (—/—/—) denotes *E. coli* proteins that did not align against any other protein sequence; the value for that row (`355`) is the difference between the total number of *E. coli* proteins used in the search (`4289`) and the sum of all the other totals.

```
+-------+----------------------------------------------------------------------------+
| total | descr                                                                      |
+-------+----------------------------------------------------------------------------+
|     2 | (3R)-hydroxymyristol acyl carrier protein dehydratase                      |
|     2 | (p)ppGpp synthetase I (GTP pyrophosphokinase) reg of RNA synth; stringent  |
|     2 | (p)ppGpp synthetase II; G-3',5'-bis pyrophosphate 3'-pyrophosphohydrolase  |
|     2 | 1-acyl-sn-glycerol-3-phosphate acyltransferase                             |
|     2 | 10-formyltetrahydrofolate:L-methionyl-tRNA(fMet) N-formyltransferase       |

                                 ...

|     2 | uroporphyrinogen III synthase                                              |
|     2 | vitamin B12 transport                                                      |
|     2 | [2FE-2S] ferredoxin, electron carrier protein                              |
+-------+----------------------------------------------------------------------------+
661 rows in set (5.74 sec)
```

Figure 9.4.18 Table returned by the query in Figure 9.4.17. The number of rows that this query returns (`661`) is the number of genes that have hits against proteins in both *Bacteria* and *Eukaryota* species, but have no significant hits against proteins from Archaea species.

BASIC PROTOCOL 6

ANALYZING SIMILARITY SEARCH RESULTS: INTERGENIC REGIONS

While ab initio gene prediction is difficult in eukaryotes (and can be difficult for prokaryotes with sequencing errors), many genes are easily identifiable by homology to known protein sequences. However, comparing complete genomic DNA sequences against the entire `nr` protein database is time consuming. Gene finding by homology can be much more efficient if one only searches against protein sequences from closely related organisms. Having identified the "low-hanging fruit," remaining stretches of intergenic sequence can be searched against a larger database. This approach is both more sensitive and faster, because a smaller database is used in the initial search, and fewer comparisons are made overall. Here, a two-step search strategy will be described, which could also be extended over multiple iterations using subsequent nested taxonomic subsets.

First, a taxonomic subset of proteins are selected that share homology with most of the genes in the target organism. For example, to identify genes in *E. coli*, one might search against the approximately 45,000 proteins from the parental family *Enterobacteriaceae*. The choice depends on the evolutionary distance to organisms with comprehensive protein data: for the puffer fish (*Fugu rubripes*), the parent order *Tetraodontiformes* includes only about 700 protein sequences; the parent class *Actinopterygii* (ray-finned fishes) includes approximately 16,000 protein sequences, while the parent superclass *Gnathostomata* (jawed vertebrates) 330,000 proteins; however, species from across the superclass have diverged over 500 million years of evolution, and these may be difficult to identify. Next, the genomic DNA would be compared to the chosen taxonomic subset of protein sequences (using a DNA-translating search algorithm—e.g., BLASTX (*UNIT 3.3*) or FASTX (*UNIT 3.9*)—and the search results would be stored in `seqdb_demo`. Then, the next step in this process would be to identify the unmatched regions of "intergenic" DNA sequence—i.e., subregions of `search_query` entries that did not produce a significant alignment, and use only these regions to search a more complete protein set. This protocol demonstrates how to produce intergenic regions from prior search results, using *S. typhimurium* (STM) sequences searched against *E. coli* (ECO) proteins.

While the process of searching a new sequence library with unmatched DNA sequences is easy to conceptualize, identifying those sequences requires several steps. Importantly, the approach illustrated here assumes a bacterial or archaeal genome without introns—i.e., any sequence-similarity hit can be considered a gene and any unmatched DNA as intergenic (and not intronic). However, the same technique could be used in eukaryotes, but only after exon-based alignments have been assembled into complete gene models and the ranges of those gene models saved as search hits in the database.

Necessary Resources

Hardware

Computer with at least 2 Gb of disk space available

Software

Windows- or Unix-based operating system (including Linux or Mac OS X)
Working version of MySQL, with functional database permissions. MySQL can be downloaded from *http://www.mysql.com* and installed as described in *UNIT 9.2*. All interactions with MySQL databases in these protocols will be via the `mysql` command-line utility.
A terminal application connected to a Unix environment in which one can execute Unix-like commands. For Windows-based operating systems, this entails installing the Cygwin Unix emulation (*http://www.cygwin.com*).
The Perl scripting language interpreter (any version since 5.005_03) and the `DBI`, and `DBD::mysql` modules.

Files

Generated as in Basic Protocols 1 and 3

1. Complete Basic Protocols 1 and 3.

   A sample set of similarity results is available from ftp://ftp.virginia.edu/pub/fasta/CPB/stm_eco.results.gz. This file must be uncompressed with the command `gunzip stm_eco.results.gz` *to produce the file* `stm_eco.results`*, which can then be loaded into the database with the* `loadsearch.pl` *command.*

2. Build a temporary table that contains the ranges of the successful hits using the SQL statement shown in Figure 9.4.19 (see `ranges.sql`, found in the `seqdb_demo` distribution). Note that it is not possible to declare this table as `TEMPORARY` because it is later going to be joined against itself).

3. For each set of hits *A* that have the same beginning on the same DNA sequence, pair them with all hits *B* on the same DNA sequence that begin after any of the *A* hits end. Take the max of the endings of *A* as the beginning of an intergenic range; from all the *B*'s, choose the smallest begin as the end of the intergenic range. Use the SQL statement shown in Figure 9.4.20.

```
CREATE  TABLE  hitranges  TYPE=HEAP
SELECT  search_query.seq_id AS query_id,
        IF(strand = 'f', qbeg, qend) AS begin,
        IF(strand = 'f', qend, qbeg) AS end
FROM    search
        INNER JOIN search_hit USING (search_id)
        INNER JOIN search_query USING (query_id)
WHERE   search.tag = 'stm-vs-eco'
  AND   search_hit.expect <= 1e-10;
```

Figure 9.4.19 SQL statement used to build a temporary table that contains the ranges of the successful hits, used in step 2 of Basic Protocol 6.

```
CREATE  TEMPORARY TABLE igranges TYPE=HEAP
SELECT  A.query_id AS query_id,
        MAX(A.end + 1) AS begin,
        MIN(B.begin - 1) AS end
FROM    hitranges AS A
        INNER JOIN hitranges AS B
          ON (A.query_id = B.query_id AND B.begin > A.end)
GROUP BY A.query_id, A.begin;
```

Figure 9.4.20 SQL statement used in step 3 of Basic Protocol 6, which contains an initial set of intergenic ranges for each query_id.

```
INSERT INTO igranges (query_id, begin, end)
SELECT query_id AS query_id,
       1 AS begin,
       MIN(begin - 1) AS end
FROM   hitranges
GROUP BY query_id
HAVING end >= begin;

INSERT INTO igranges (query_id, begin, end)
SELECT hitranges.query_id AS query_id,
       MAX(hitranges.end + 1) AS begin,
       search_query.len AS end
FROM   hitranges
       INNER JOIN search_query
           ON (hitranges.query_id = search_query.seq_id)
GROUP BY hitranges.query_id
HAVING end >= begin;
```

Figure 9.4.21 Two SQL statements used for adding the missed classes of beginning and ending "intergenic" DNA sequence to the igranges table (see step 4 of Basic Protocol 6).

4. The SELECT statement used in step 2 (Fig. 9.4.20) missed two important classes of "intergenic" DNA sequence: the range from the beginning of the DNA sequence to the first hit, and the range from the last hit to the end of the DNA sequence. The two SQL statements in Figure 9.4.21 add those ranges to the igranges table.

5. Finally, it is desirable to add any DNA sequence queries that did not match against anything (and thus have no rows in the hitranges table), using the SQL statement in Figure 9.4.22. This must be done in two steps because it is not possible to simultaneously SELECT from a table into which one is also INSERT-ing.

6. What remains is to clean the igranges table of a few sources of artifactually overlapping ranges. The first is caused when a collection of hits look like the diagram shown in Figure 9.4.23A, leading to two igrange's as shown in Figure 9.4.23B. Only the lowermost igrange, marked by the arrow, is desired. The unwanted longer range is removed by grouping the igrange's on end and selecting MAX(begin) as the new boundary:

```
CREATE TEMPORARY TABLE clean_igranges TYPE=HEAP
SELECT query_id, MAX(begin) AS begin, end
FROM igranges
GROUP BY query_id, end;
```

```
CREATE  TEMPORARY TABLE missing TYPE=HEAP
SELECT  search_query.seq_id AS query_id,
        1 AS begin,
        search_query.len AS end
FROM    search_query
        LEFT JOIN hitranges
            ON (search_query.seq_id = hitranges.query_id)
WHERE   hitranges.query_id IS NULL;

INSERT  INTO igranges (query_id, begin, end)
SELECT  query_id, begin, end
FROM    missing;
```

Figure 9.4.22 SQL statement to add any DNA sequence queries that did not match against anything and that have no rows in the `hitranges` table (see step 5 of Basic Protocol 6).

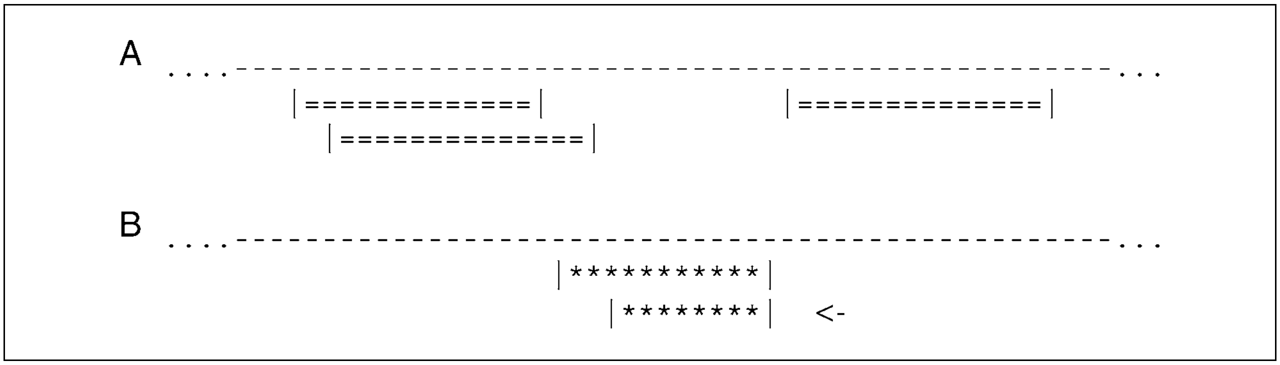


Figure 9.4.23 Schematic illustration of one possible source of artifactually overlapping ranges; the collection of hits in (**A**) lead to two `igrange`'s as shown in (**B**). Only the lowermost `igrange`, marked by the caret, is desired. See step 6 of Basic Protocol 6.

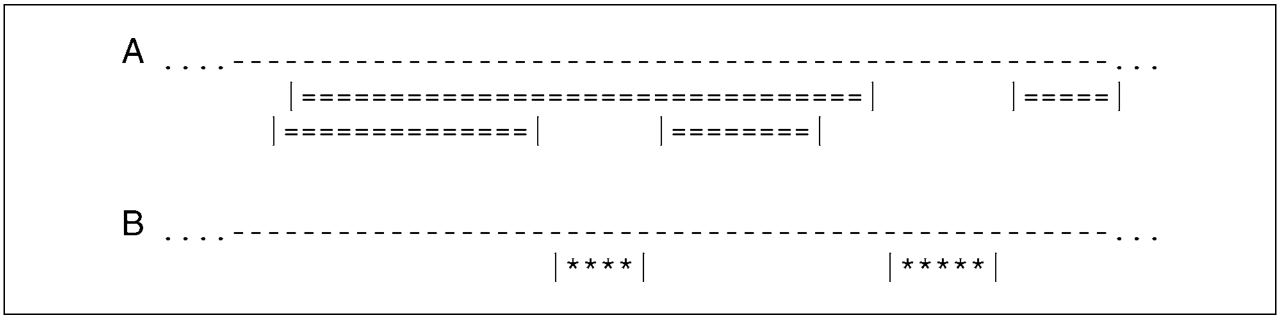


Figure 9.4.24 Schematic illustration of a second possible source of artifactually overlapping ranges: (**A**) the begin and end of two small hits are spanned by a third, larger hit, leading to the ranges shown in (**B**).

7. The second set of artifactual overlap ranges stems from hits where the begin and end of two small hits are spanned by a third, larger hit as shown in Figure 9.4.24A, leading to the ranges shown in Figure 9.4.24B. The unwanted ranges are eliminated by checking to see if any of the ranges overlap within the original set of hits using the SQL statement in Figure 9.4.25; any that do are not selected into the final set of intergenic ranges.

   The `final_igranges` *table now contains the intergenic regions. These regions could be used as the basis for queries in a subsequent search of a larger taxonomic subset of protein sequences; the above process can then be repeated for each new subset of intergenic regions.*

```
CREATE TEMPORARY TABLE overlaps TYPE=HEAP
SELECT DISTINCT clean_igranges.query_id,
                clean_igranges.begin, clean_igranges.end
FROM   clean_igranges
       INNER JOIN hitranges USING (query_id)
WHERE  clean_igranges.end <= hitranges.end
  AND  clean_igranges.begin >= hitranges.begin;

CREATE TEMPORARY TABLE final_igranges (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  query_id INT UNSIGNED NOT NULL DEFAULT 0,
  begin INT UNSIGNED NOT NULL DEFAULT 0,
  end INT UNSIGNED NOT NULL DEFAULT 0
);

INSERT INTO final_igranges (query_id, begin, end)
SELECT clean_igranges.query_id, clean_igranges.begin,
clean_igranges.end
FROM   clean_igranges
       LEFT JOIN overlaps USING (query_id, begin, end)
WHERE  overlaps.query_id IS NULL;
```

Figure 9.4.25 SQL statement for eliminating unwanted ranges from the final set of intergenic ranges.

COMMENTARY

Background Information

Relational databases provide a powerful and flexible foundation for large-scale sequence comparison, and make it much easier to implement the "management controls" necessary to keep track of sequences, alignment positions, and scores. The seqdb_demo database and the accompanying Basic Protocols in this unit are meant to serve as examples of the many ways that relational databases can simplify genome-scale analyses in an investigator's research.

These protocols use relational databases and SQL to provide comprehensive summaries of large-scale sequence comparisons. To provide relatively compact examples, the authors have focused on evolutionary analyses, e.g., the number of homologs that are shared between different taxonomic classes. The power of relational approaches greatly increases as additional data are added to the database. In addition to sequence and taxonomic data, relational databases can store information about protein families and domains (e.g., PFAM) or protein functional assignments (the Gene Ontology or GO classification). Relational databases are particularly powerful when they are used to associate different kinds of data; for example, one might ask how often homologous proteins (proteins that share statistically significant similarity) are distant in the GO hierarchy and thus are likely to have different functions. As biological databases become more diverse, including not only sequence data but also genome locations, biological function, interaction results, and biological pathways, SQL databases provide powerful tools for exploring relationships between very different sets of biological data on a genome scale.

Literature Cited

Celko, J. 1999. Joe Celko's SQL for Smarties. Morgan Kaufmann, San Francisco.

Internet Resources

ftp://ftp.ncbi.nih.gov/pub/blast/db/FASTA/nr.gz

Comprehensive nr *database (flat file protein sequence database).*

ftp://ftp.ncbi.nih.gov/pub/blast/db/FASTA/swissprot.gz

SwissProt protein database (flat file protein sequence database).

ftp://ftp.pir.georgetown.edu.pir_databases/
psd/mysql/

The Protein Identification Resource (PIR) at Georgetown University, which distributes the PIR protein database in relational format for the MySQL database program.

Contributed by Aaron J. Mackey and
William R. Pearson
University of Virginia
Charlottesville, Virginia

Using Apollo to Browse and Edit Genome Annotations

UNIT 9.5

An annotation is any feature that can be tied to sequence, such as an exon, promoter, or transposable element. The Apollo tool (Lewis et al., 2002) allows researchers to explore genomic annotations at many levels of detail, and to perform expert annotation curation, all in a graphical environment. It is currently used by FlyBase (Drysdale et al., 2005) and Ensembl (Hubbard et al., 2005) to display genomic sequence annotations for many different species. The Apollo Java application can be downloaded from the Web and run locally on any Windows, Mac OS X, or Unix-type system (including Linux). Version 1.6.0, described in this protocol, was released in November, 2005, but Apollo is actively under development and new versions are released every few months.

The simplest use of Apollo is to browse gene annotations and supporting evidence for the annotations of a single species, like *Drosophila melanogaster*, that are stored in GAME Extensible Markup Language (XML) format (Basic Protocol). However, one can access data from a Chado database via Chado XML files (Alternate Protocol 1) or directly via Java Database Connectivity (JDBC; Alternate Protocol 2), from an Ensembl database (Alternate Protocol 3), from Gene Finding Format (GFF) files (Alternate Protocol 4), or from GenBank or EMBL formatted files (Alternate Protocol 5). Support Protocol 1 describes moving beyond browsing to create, delete, split, and merge annotations of different types, modify intron-exon structures, or add and edit text comments. Support Protocol 2 describes the configuration files necessary to view data from each source. This unit does not discuss using the Apollo Synteny Browser and Editor to compare annotations and evidence between multiple species, but detailed information is available at the Apollo User Guide Web site at *http://www.fruitfly.org/annot/apollo/userguide.html*.

BROWSING GENOME ANNOTATIONS IN GAME XML FILES

BASIC PROTOCOL

Apollo can display a region of genomic sequence, obtained by querying a live database (by a gene symbol or identifier, map location, genomic sequence coordinates, or GenBank accession for the genomic sequence segment), or by simply loading a local file. This protocol illustrates how to download and install the program, query for a gene, and browse computational evidence and other information associated with the annotated gene model and sequence, using the *Drosophila melanogaster* annotations in GAME XML format as an example.

Necessary Resources

Hardware

Unix (Linux, Solaris, or other variety) workstation, PC with Microsoft Windows, or Macintosh with OS X v. 10.2.3 or later
At least 164 Mb of RAM
Internet connection if querying database interactively
Three-button mouse (see Troubleshooting for information on simulating a three-button mouse)

Software

Apollo program version 1.6.0 or higher
Java JDK 1.4 (included with the Apollo distribution for Unix and Windows)

Contributed by Sima Misra and Nomi Harris

Current Protocols in Bioinformatics (2005) 9.5.1-9.5.28

Files

Configuration files (included with installation)

Data in GAME XML format (sample file: `example.xml` is included with installation; more data can be obtained from within Apollo transparently over the Internet). If edited, files must be saved in plain text format with line breaks, i.e., as a simple ASCII file.

Download and install Apollo

1. Download Apollo from the Apollo home page at *http://www.fruitfly.org/annot/apollo/* and install on the local computer by clicking Install and following the instructions on the screen. When Apollo is installed, a Java Virtual Machine (JVM) is bundled with it, except under Mac OS X. If the browser being used does not support Java, download the installer and then launch it by double-clicking on the installer icon or (for Unix/Linux) by typing `sh./apollo.bin.`

   On Unix or Linux, be aware that, when asked where to make links to Apollo, one cannot choose the Apollo installation directory, or the link will overwrite the Apollo executable.

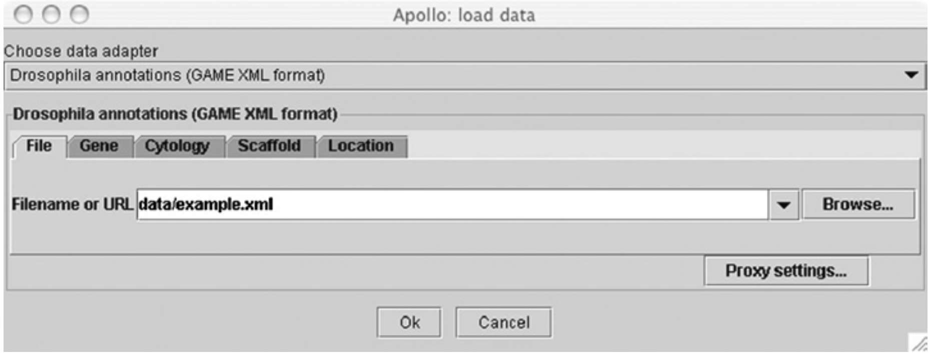
   On Mac OS X version 10.2.3 or higher, install JDK1.4, if not already installed, using Software Updates. Sometimes, the Apollo installer stops responding due to a known problem with the installer software. If this happens, close the installer and restart it.

Browse genome annotations graphically

2. *Start an Apollo session.* On a Windows or Macintosh machine, click on the Apollo icon. In Unix or Linux, type `Apollodir/Apollo`, where `Apollodir` is the directory in which Apollo is installed. The Apollo splash screen will stay up for a few seconds while the Apollo software loads. A window opens offering various ways to find the region of interest, as shown in Figure 9.5.1.

   The first time Apollo is opened on a Mac, a warning is displayed about JDK1.4. If JDK1.4 is already installed, ignore the warning and restart Apollo.

3. *Query for and load a region of interest.* From the pull-down menu under "Choose data adapter," select *Drosophila* Annotations (GAME XML Format), click on a tab (e.g., File), and enter text in the appropriate box(es). For this example, load the default `example.xml` file that was downloaded into the `Apollo/data/` directory during installation. Click the OK button at the bottom of the window.



Apollo: load data

Choose data adapter
Drosophila annotations (GAME XML format)

Drosophila annotations (GAME XML format)

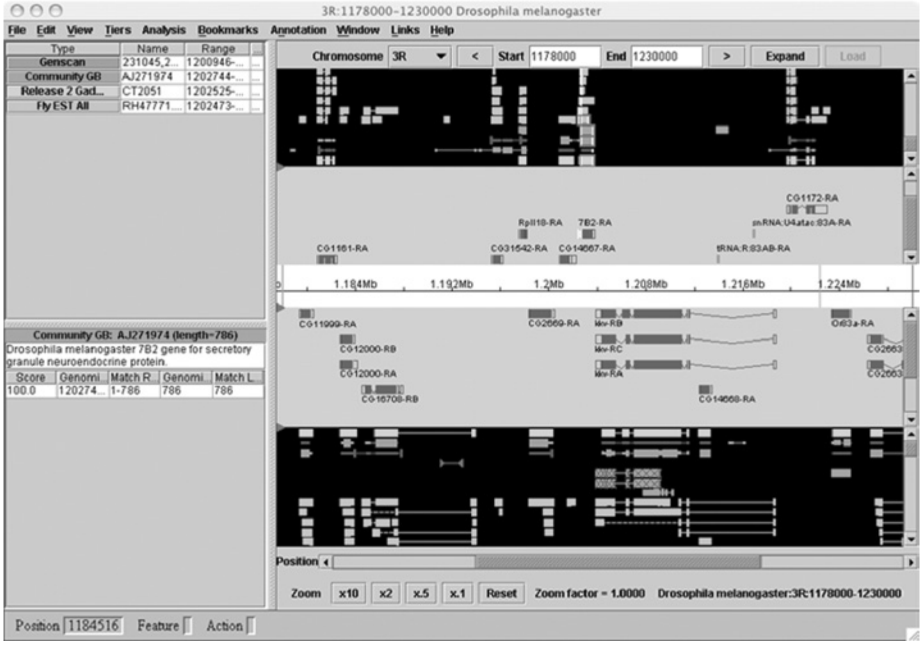
Filename or URL data/example.xml

Figure 9.5.1 Loading an XML data file into Apollo. *Drosophila* annotations may also be loaded over the Internet by Gene name, Cytology, etc. by clicking on the tabs.

Apollo provides several ways to search for the region of interested: Gene (symbol or identifier); Scaffold (GenBank accession, a 250- to 350-kb section of the Drosophila genomic sequence); Location (chromosome arm, start, and end position, e.g., arm:3R, start:100000, end:300000); or File (e.g., `data/example.xml`*, or use Browse to select a file, then click the OK button). To download all the Drosophila annotation data to the computer to access it locally via the File tab, see http://flybase.net/annot/.*

4. *Browse the computational evidence and annotations in the main display.* The main window, shown in Figure 9.5.2, displays a portion of *Drosophila* chromosome arm 3R. Each feature is displayed as a colored box; different feature types have different colors, with feature sets (groups of features like the exons in a transcript) connected by lines. All forward strand features are shown above the “axis,” which is labeled in base pairs in the middle of this panel, and reverse strand features are below the axis. Green and red vertical bars on the axis indicate the limits of the currently displayed sequence range. Raw “computational evidence” or “results” (e.g., BLAST, Genscan) produced by computational algorithms are displayed in the black zones above and below the axis. “Annotations,” features that can be tied to sequence, such as transcripts, promoters, or transposable elements, can be synthesized from the results in the black zones, e.g., by a human curator. Manually annotated blue and green gene transcript models appear in light blue zones above and below the axis. Dragging the small red arrows at the left edge of the display controls the amount of space dedicated to results versus annotations.

5. *Zoom and scroll to a region of interest*. To zoom in for a closer view, use the “×2” and “×10” buttons at the bottom of the window (Fig. 9.5.2). Zooming in enough will show the individual bases near the axis and residues for many features, and all possible start codons (in green) and stop codons (in red) will appear (Fig. 9.5.3). To zoom out, use the “×.1” and “×.5” buttons. Use the Reset button to go back to the original zoom level. To scroll horizontally, use the scroll bar just above the Zoom buttons. The main panel also has vertical scroll bars. If there are many different



Type	Name	Range
Genscan	231045,2...	1200946-...
Community GB	AJ271974	1202744-...
Release 2 Gad...	CT2051	1202525-...
Fly EST All	RH47771....	1202473-...

Community GB: AJ271974 (length=786)

Drosophila melanogaster 7B2 gene for secretory granule neuroendocrine protein.

Score	Genomi...	Match R...	Genomi...	Match L...
100.0	120274...	1-786	786	786

Zoom factor = 1.0000 Drosophila melanogaster:3R:1178000-1230000

Position 1184516

Figure 9.5.2 The main display of Apollo, with detail panel at the left, result panels in black, annotations panels in light blue, navigation panel at top right, and scroll and Zoom buttons at the bottom. For the color version of this figure go to *http://www.currentprotocols.com*.

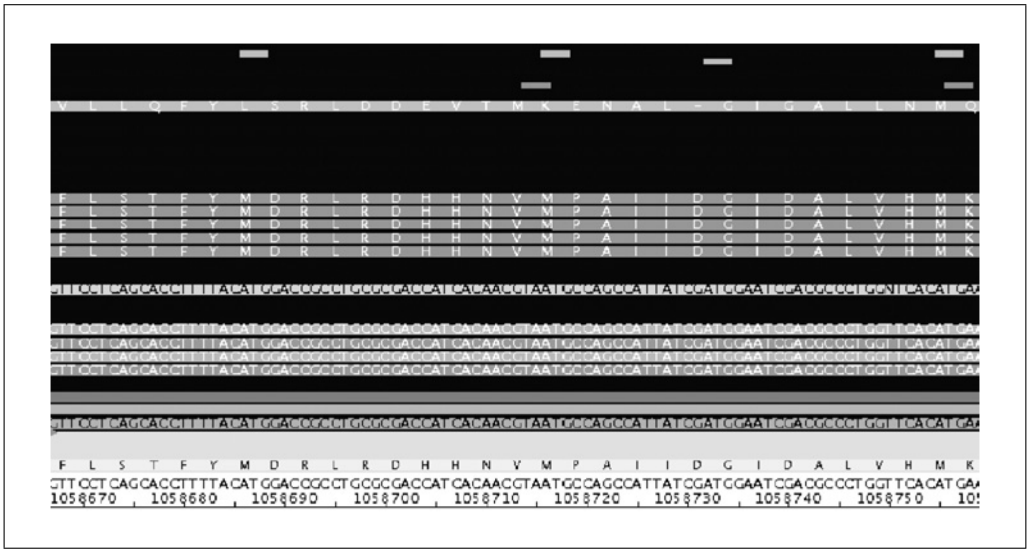

V L L Q F Y L S R L D D E V T M K E N A L - G I G A L L N M Q
F L S T F Y M D R L R D H H N V M P A I I D G I D A L V H M K
CTT CCT CAGCACCTTTTACATGGACCGCCTGCGCGACCATCACAACGTAATGCCAGCCATTATCGATGGAATCGACGCCCTGGTTCACATGAA
1058670 , 1058680 , 1058690 , 1058700 , 1058710 , 1058720 , 1058730 , 1058740 , 1058750 , 10

Figure 9.5.3 Zoomed-in main display of Apollo, showing green start and stop codons at the top, sequence of result features, and genomic reference sequence and metric at the bottom. For the color version of this figure go to *http://www.currentprotocols.com*.

results or annotations stacked up, one may need to scroll vertically with the scroll bars or mouse wheel to see all of them. To zoom in and center on a particular feature, click on the feature to select it, then choose the Zoom to Selected function from the View menu (keyboard shortcut, `Ctrl-z`).

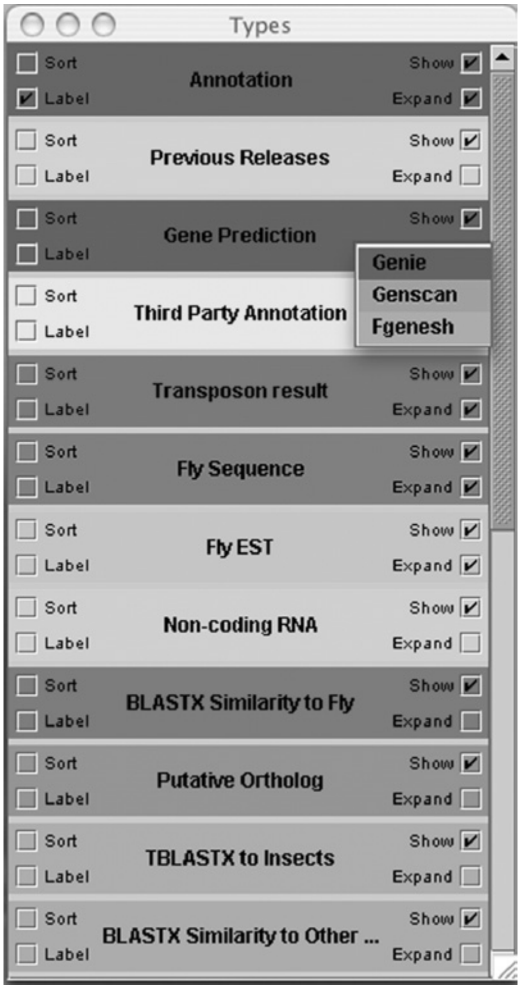
Clicking the middle mouse button anywhere in the main panel will center the panel on that point. If any of the Zoom buttons is then pressed, the display will stay centered around the selected position as it is zoomed in or out.

6. *Choose strand to inspect.* To facilitate examination of genes on the reverse strand, the entire display (including the sequence) can be reverse complemented by selecting Reverse Complement from the View menu. When the display is reverse complemented, the axis turns red. To toggle strands on and off, use the “Show forward strand” and “Show reverse strand” check boxes in the View menu.

   Selecting “Flip strands” from the View menu reverse complements if both strands are currently displayed. If only one strand is shown, “Flip strands” will show the other strand, reverse complemented.

7. *Select features.* Clicking on a feature with the left mouse button selects it; selected features appear boxed in red (Fig. 9.5.2). To select more than one feature, hold down the middle mouse button while dragging the mouse around the features to be selected (“rubber-banding”). Holding down the Shift key while left clicking (“Shift-clicking”) can be used to toggle features in or out of the selection. Add features to the currently selected set by Shift-clicking with the left mouse button (to add single features) or dragging while holding down the Shift key and the middle mouse button (to add multiple features). To deselect one or more of the features selected while leaving the others selected, Shift-click (or drag while holding down the Shift key and the middle mouse button) the feature(s) to be deselected. Click on an intron or double-click an exon to select the whole transcript or feature set. To select all transcripts of an annotated gene, double-click an intron. Double-clicking an intron belonging to a computational result feature selects all results of that particular type.

8. *Determine how features are displayed with the Types Panel.* Computational evidence features are organized into “tiers,” i.e., horizontal rows of data that can be labeled and controlled as a unit. In addition to the tier, each feature is associated with a



☐ Sort ☑ Label — Annotation — Show ☑ Expand ☑

☐ Sort ☐ Label — Previous Releases — Show ☑ Expand ☐

☐ Sort ☐ Label — Gene Prediction — Show ☑

☐ Sort ☐ Label — Third Party Annotation

☐ Sort ☐ Label — Transposon result — Show ☑ Expand ☑

☐ Sort ☐ Label — Fly Sequence — Show ☑ Expand ☑

☐ Sort ☐ Label — Fly EST — Show ☑ Expand ☑

☐ Sort ☐ Label — Non-coding RNA — Show ☑ Expand ☐

☐ Sort ☐ Label — BLASTX Similarity to Fly — Show ☑ Expand ☐

☐ Sort ☐ Label — Putative Ortholog — Show ☑ Expand ☐

☐ Sort ☐ Label — TBLASTX to Insects — Show ☑ Expand ☐

☐ Sort ☐ Label — BLASTX Similarity to Other ... — Show ☑ Expand ☐

Figure 9.5.4 Types panel, indicating color and presentation of each tier of data. Performing a right mouse click over a tier, in this example over Gene Prediction, brings up a pop-up menu with the individual data types in the tier (Genscan, Genie, and FgenesH analyses). For the color version of this figure go to *http://www.currentprotocols.com*.

particular data "type" (e.g., the Gene Prediction tier includes the Genie type and the Genscan type, corresponding to the results from different algorithms). Each data type can be individually assigned a color, indicated in the Types Panel (Fig. 9.5.4), and a shape or "glyph," set in the tiers configuration file, described in Support Protocol 2.

a. To bring up the Types panel, select Show Types Panel in the Tiers menu. The Types panel shows the names of the different types of computational results and the colors that were assigned to them, organized by tier. To see a pop-up menu of types that belong to each tier, click the right mouse button while the mouse cursor is over that tier panel in the Types panel (Fig. 9.5.4). Selecting one of the types from the popup menu brings up a color editor to make it possible to change the color of that result type. Middle clicking over a tier panel makes it possible to adjust the score threshold for showing results of that type.

b. The panel for each tier has check boxes that can be used to change the view in the main panel, e.g., to conserve space. The Show check box determines whether or not to display a tier. The Expand check box determines whether all features of a given type are collapsed into the same row, possibly on top of each other (e.g., the light blue tier in Fig. 9.5.2) or expanded on different rows so that no two features overlap (e.g., the light green tier in Fig. 9.5.2). The Sort box determines whether the features are sorted, so that the highest-scoring features in a tier are closest to the center of the display. The Label check box determines whether the features in that tier are labeled with their names.

An alternative way to expand/collapse or show/hide a result tier is to select a feature, bring up the right-mouse-button pop-up menu (RMPM), and select the Expand/Collapse tier or Show/Hide tier options to force all features of that type to be expanded or collapsed, or shown or hidden. To expand/collapse or show/hide all tiers, use the options under the Tiers menu.

9. *Identify matching, new, and suspicious evidence*. Apollo conveys a great deal of information graphically in the main display; see Figure 9.5.5 for examples of several useful visual flags. To identify features with matching endpoints, all features that have the same 5′ or 3′ boundary as the currently selected feature are highlighted with white lines at the edges of the feature boxes. If a type in the results panel has a manually set date defined in the tiers configuration file (described in Support Protocol 2), then results of that type representing matches to sequences newer than that date are shown with a white box around them. Sequencing gaps, e.g., when 5′ and 3′ Expressed Sequence Tags (ESTs) from the same cDNA clone are matched but internal sequence is missing, are indicated by dashed lines. If a splice site is incorrect or unconventional (not a GT donor or AG acceptor), an orange triangle appears in the annotation at the site of the splice site. If cDNA or EST evidence has been tagged with a comment indicating that it is incomplete or suspect, the feature appears cross-hatched in bright pink. Finally, if a protein-coding annotation is missing a start or stop codon, it will appear with a green or red arrowhead, respectively.

Read more information associated with genome annotations

10. *Read more about a feature in the detail panel*. Summary information about selected feature(s) will appear in the Detail panel at the left or bottom of the main window, shown at the left in Figure 9.5.2. The far left side of the Detail panel shows a list of the types of features selected and their names, sequence ranges, and scores. The right side of the panel displays the coordinates of the feature selected in the left panel, and usually other information, depending on the feature type.

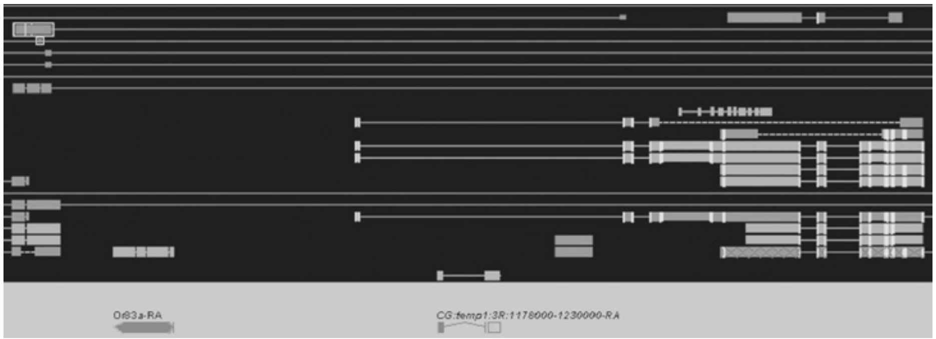


Figure 9.5.5 Visual cues in the Apollo main display indicate new, matching, or problematic data. For example, boxed features indicate new results, white edge lines indicate matching features, cross-hatched features have been flagged, dotted lines indicate sequencing gaps, and arrowheads indicate missing start or stop codons or nonconsensus splice sites. For the color version of this figure go to *http://www.currentprotocols.com*.

By default, the selected features are sorted by start position (Range) in the detail panel on the lower left. One can change the sort by clicking (for forward sort) or Shift-clicking (for reverse sort) on a column header. For example, clicking on Name sorts the selected features in alphabetical order by name, and Shift-clicking on Score sorts them in descending order by score. The details displayed and the default sort order for any feature type are configurable. The panel can be oriented horizontally or vertically by selecting Make Evidence Panel Horizontal/Vertical from the View menu.

11. *Read more about a feature via the Web.* If any feature is selected and the right mouse button is clicked, the right mouse pop-up menu (RMPM) will appear (Fig. 9.5.6). Select “Get info about this feature via the Web” to get more information from the parent database of that feature (e.g., SwissProt, GenBank) in the browser.

    If the browser window is hidden at this point, it may be necessary to bring it back up to see the report. If a Web browser is not running, Apollo will try to launch the default browser; if this does not work, start the Web browser manually and try again.

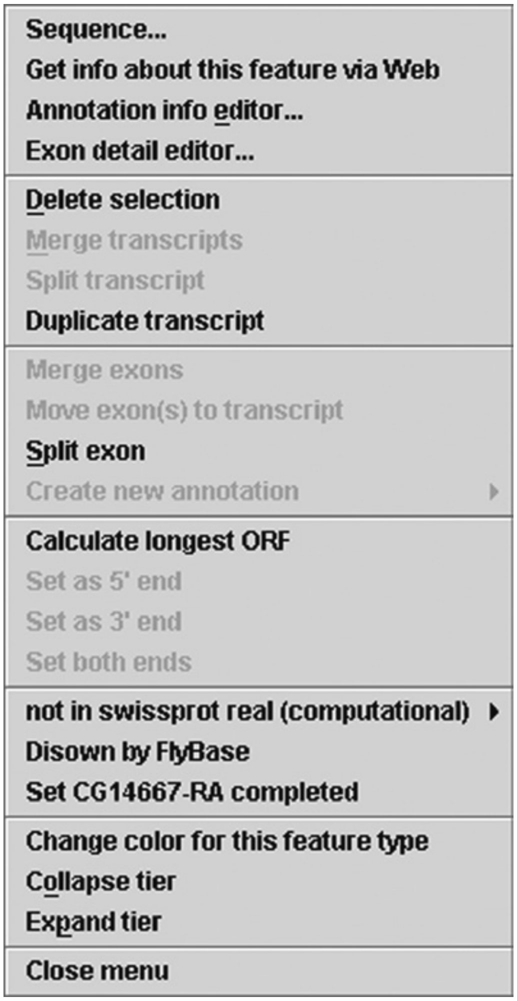
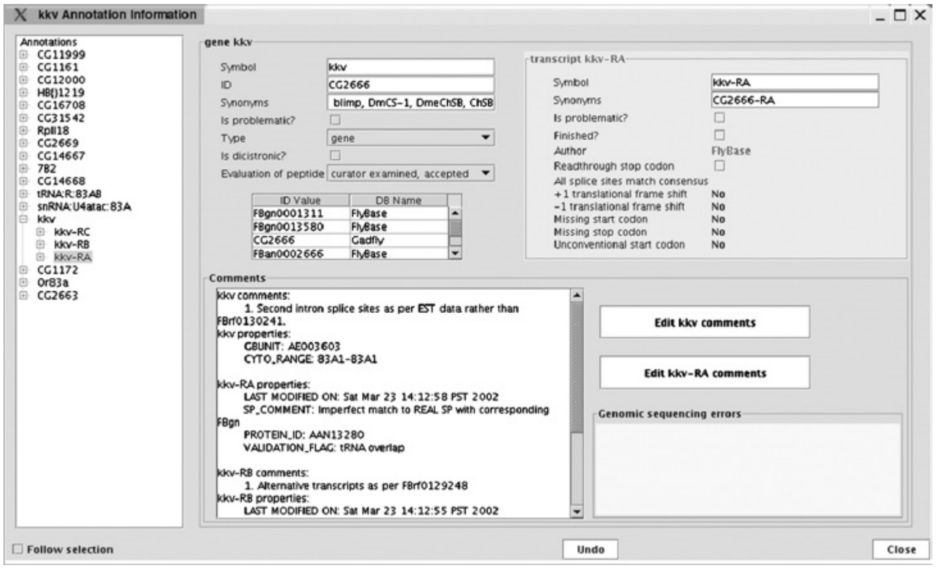


Figure 9.5.6 The right-mouse pop-up menu (RMPM) brings up various browsing and editing functions, depending on which feature is selected. In this example, an annotated gene was selected.



gene kkv

Symbol: kkv
ID: CG2666
Synonyms: blimp, DmCS-1, DmeChSB, ChSB
Is problematic? ☐
Type: gene
Is dicistronic? ☐
Evaluation of peptide: curator examined, accepted

ID Value	DB Name
FBgn0001311	FlyBase
FBgn0013580	FlyBase
CG2666	Gadfly
FBan0002666	FlyBase

transcript kkv-RA

Symbol: kkv-RA
Synonyms: CG2666-RA
Is problematic? ☐
Finished? ☐
Author: FlyBase
Readthrough stop codon ☐
All splice sites match consensus
+1 translational frame shift: No
−1 translational frame shift: No
Missing start codon: No
Missing stop codon: No
Unconventional start codon: No

Comments

```
kkv comments:
      1. Second intron splice sites as per EST data rather than
FBrf0130241.
kkv properties:
      GBUNIT: AE003603
      CYTO_RANGE: 83A1-83A1

kkv-RA properties:
      LAST MODIFIED ON: Sat Mar 23 14:12:58 PST 2002
      SP_COMMENT: Imperfect match to REAL SP with corresponding
FBgn
      PROTEIN_ID: AAN13280
      VALIDATION_FLAG: tRNA overlap

kkv-RB comments:
      1. Alternative transcripts as per FBrf0129248
kkv-RB properties:
      LAST MODIFIED ON: Sat Mar 23 14:12:55 PST 2002
```

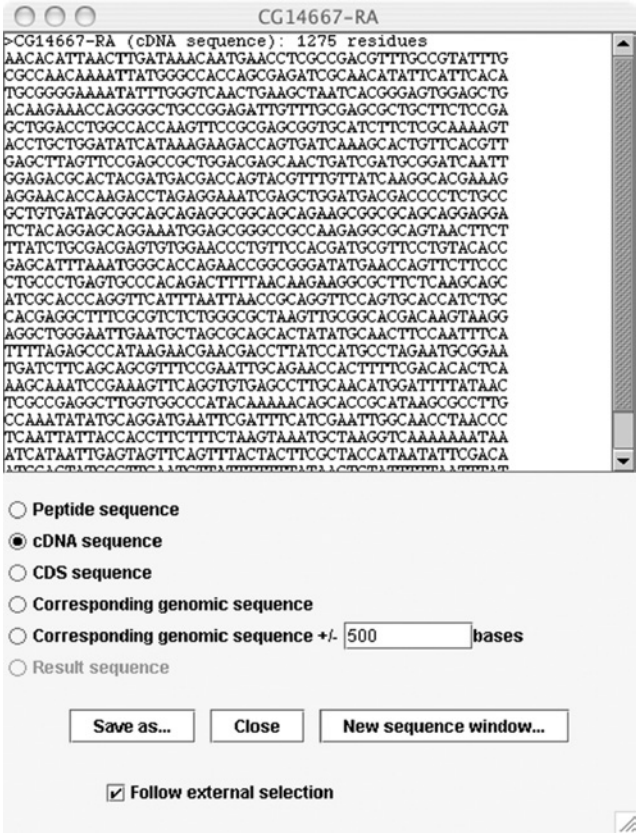
Figure 9.5.7 The Annotation Info Editor in Apollo allows browsing and editing of text information, including comments, associated with annotations.

12. *Read more text information about annotations*. The RMPM item Annotation Info Editor brings up a window that has text information about the annotation, shown in Figure 9.5.7. Comments and properties associated with the annotation are shown in the scrollable Comments panel, followed by comments and properties associated with individual transcripts belonging to the annotation. Above the Comments panel is a table listing database identifier(s) for this annotation. If an identifier is selected, more information about that database entry (if available) will appear in the Web browser. For more see Support Protocol 1 and the Apollo User Guide (also see step 20 of this protocol).

    The Follow Selection check box on the lower left side of the window is used to follow selections from other parts of Apollo. If it is checked, when another annotation is selected in the main Apollo window, the Annotation Info Editor will load that annotation; if an annotation name is selected in the Annotation Info window, the main window will scroll to show that annotation. Alternatively, multiple Annotation Info Editors can be opened at once.

13. *Display and save the sequence of any feature*. To display the sequence of one or more features in FASTA format, select the feature(s) and choose the Sequence option on the RMPM to open a Sequence Window (Fig. 9.5.8). The Sequence Window display options for annotations include the translated peptide sequence, cDNA sequence (exons only), CDS (from start to stop of translation), and corresponding genomic sequence (plus and minus the desired amount of genomic sequence on either side of the feature). The default display is amino acid sequence for gene annotations and result sequence for results that have sequence. The sequence can be copied and pasted into another application; alternatively the currently displayed sequence(s) can be saved as a multiple-FASTA file by clicking the Save As button.

    By default, the sequence window does not follow selections made in the main panel. If the Follow External Selection check box is checked, then, as new features in the main panel are selected, their sequences show up in the Sequence window. Launch new sequence windows from an existing one by clicking the New Sequence Window button. For annotations or features that extend over an edge of the region being displayed (indicated by green and red vertical lines across the axis), the Sequence menu item is disabled.



```
>CG14667-RA (cDNA sequence): 1275 residues
AACACATTAACTTGATAAACAATGAACCTCGCCGACGTTTGCCGTATTTG
CGCCAACAAAATTATGGGCCACCAGCGAGATCGCAACATATTCATTCACA
TGCGGGGAAAATATTTGGGTCAACTGAAGCTAATCACGGGAGTGGAGCTG
ACAAGAAACCAGGGGCTGCCGGAGATTGTTTGCGAGCGCTGCTTCTCCGA
GCTGGACCTGGCCACCAAGTTCCGCGAGCGGTGCATCTTCTCGCAAAAGT
ACCTGCTGGATATCATAAAGAAGACCAGTGATCAAAGCACTGTTCACGTT
GAGCTTAGTTCCGAGCCGCTGGACGAGCAACTGATCGATGCGGATCAATT
GGAGACGCACTACGATGACGACCAGTACGTTTGTTATCAAGGCACGAAAG
AGGAACACCAAGACCTAGAGGAAATCGAGCTGGATGACGACCCCTCTGCC
GCTGTGATAGCGGCAGCAGAGGCGGCAGCAGAAGCGGCGCAGCAGGAGGA
TCTACAGGAGCAGGAAATGGAGCGGGCCGCCAAGAGGCGCAGTAACTTCT
TTATCTGCGACGAGTGTGGAACCCTGTTCCACGATGCGTTCCTGTACACC
GAGCATTTAAATGGGCACCAGAACCGGCGGGATATGAACCAGTTCTTCCC
CTGCCCTGAGTGCCCACAGACTTTTAACAAGAAGGCGCTTCTCAAGCAGC
ATCGCACCCAGGTTCATTTAATTAACCGCAGGTTCCAGTGCACCATCTGC
CACGAGGCTTTCGCGTCTCTGGGCGCTAAGTTGCGGCACGACAAGTAAGG
AGGCTGGGAATTGAATGCTAGCGCAGCACTATATGCAACTTCCAATTTCA
TTTTAGAGCCCATAAGAACGAACGACCTTATCCATGCCTAGAATGCGGAA
TGATCTTCAGCAGCGTTTCCGAATTGCAGAACCACTTTTCGACACACTCA
AAGCAAATCCGAAAGTTCAGGTGTGAGCCTTGCAACATGGATTTTATAAC
TCGCCGAGGCTTGGTGGCCCATACAAAAACAGCACCGCATAAGCGCCTTG
CCAAATATATGCAGGATGAATTCGATTTCATCGAATTGGCAACCTAACCC
TCAATTATTACCACCTTCTTTCTAAGTAAATGCTAAGGTCAAAAAAATAA
ATCATAATTGAGTAGTTCAGTTTACTACTTCGCTACCATAATATTCGACA
```

○ Peptide sequence
● cDNA sequence
○ CDS sequence
○ Corresponding genomic sequence
○ Corresponding genomic sequence +/- 500 bases
○ Result sequence

☑ Follow external selection

Figure 9.5.8 The Apollo Sequence window displays the sequence of selected features in FASTA format; a variety of options are available for protein-coding gene annotations.

14. *Display and save the sequence of any genomic region.* The Save Sequence option under the File menu saves the entire sequence of the current region as a FASTA-formatted file. To save any part of the genomic sequence, e.g., for further analyses such as BLAST or primer prediction, use the middle mouse button to rubber-band the region of interest immediately above (for the sequence of the forward strand) or below (for the reverse strand) the axis. If a very large region is selected, a prompt will appear instructing the user to enter a filename to save the sequence as a FASTA file; otherwise, the selected region will appear in a new Sequence window.

Navigate from within Apollo

15. *Navigate between regions in the genome.* Use the navigation bar at the top of the main display (Fig. 9.5.2) to move between regions. To go to a particular region, select a chromosome arm from the pull-down menu, enter start and end positions, then click the Load button to fetch the new region over the Internet. To request the region immediately upstream or downstream, use the < and > buttons, then click the Load button. To extend the current region by 50%, click the Expand button, and then click the Load button. The Load button is disabled until there is a change in the requested region.

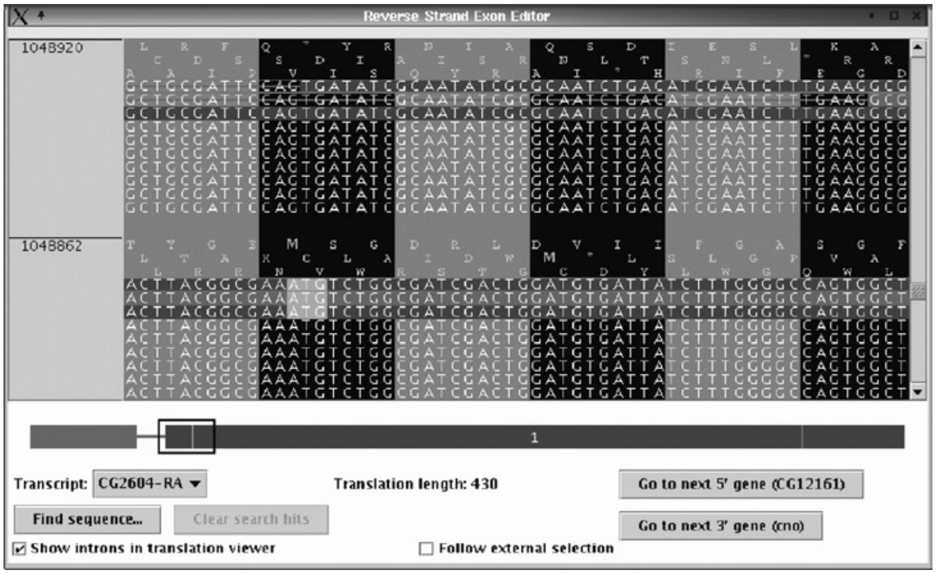
    This option only works for datasets that query transparently over the Internet, such as Drosophila.

16. *Navigate with the Annotation Menu and Annotation Tree*. The Annotation Menu lists all of the annotations in the current region alphabetically and color-coded by annotation type. When an annotation name is selected, Apollo will zoom to that annotation. The Annotation Tree lists annotations in the left-to-right order in which they appear in the displayed region, in the Annotation Info Editor described in step 12 (Fig. 9.5.7), or as a separate window, opened by selecting Show Annotation Tree under the View menu. After making an initial selection in the annotation tree, use the keyboard arrow keys to navigate up and down the tree. If the + icon to the left of an annotation name is clicked, that item will expand to show all the transcripts for that gene. Clicking the + to the left of a transcript will show the coordinates of the exons for that transcript.

    On a Mac, one will see horizontal and vertical arrowheads instead of + and −. Click on the base of the arrowhead, not the tip, to open or close a node.

17. *Search by position, name, or sequence*. To search within the current region, select Find from the Edit menu. Entering a base-pair coordinate in the Position field and pressing “Go to” will center the main panel display on that base pair. Entering an accession number, gene name, or other main feature identifier in the Name field and pressing Find will search through all of the features and select all that have that name; the main panel will center on the first. Search is case-insensitive, and adds an implicit wildcard (∗) to the end of the search string. Entering a nucleic acid sequence in the Sequence field on the Search window will search for all occurrences in the current genomic sequence on the selected strand and display them in a table. Clicking on an entry will zoom to and highlight the matching subsequence next to the axis.

    If the “Use Regular Expressions?” box is checked, one can search for a sequence or a feature name that matches a pattern of some kind. For example, by querying for `ATCG. {0,20} GGAC`*, all sequences flanked by ATCG and GGAC with up to 20 bases in between would be identified.*



Transcript: CG2604-RA ▾

Translation length: 430

☑ Show introns in translation viewer

□ Follow external selection

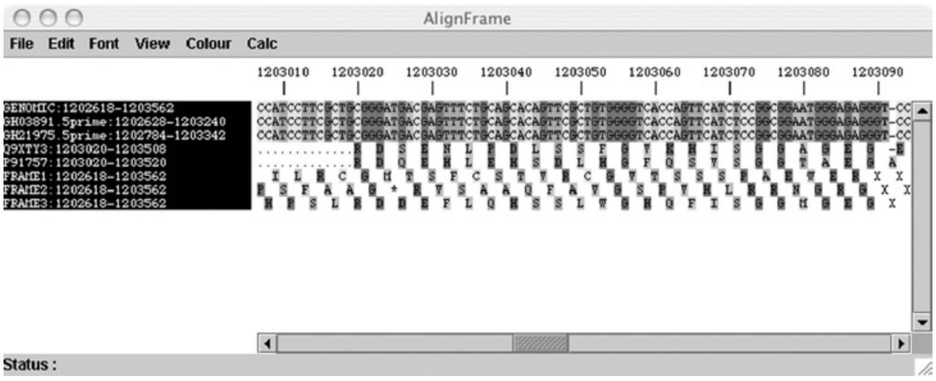
Figure 9.5.9 The Exon Detail Editor (EDE) allows browsing and editing of annotations while viewing the genomic sequence and three-frame translation. Click the diagram at the bottom of the EDE to navigate within the annotated model. For the color version of this figure go to *http://www.currentprotocols.com*.

View features in greater detail

18. *View annotated transcripts with reading frames.* The Exon Detail Editor (EDE) window (Fig. 9.5.9) is invoked from the RMPM after selecting an annotation. A separate line of reference nucleotide sequence is displayed for each transcript that appears in the annotation zone in the region being viewed. The three-frame translation of the reference sequence is also displayed, with all start codons highlighted in green and all stop codons highlighted in red. The exons for each transcript are denoted on the sequence in blue with successive exons in alternating light and dark blue shades. The selected transcript appears outlined in red, and its name is indicated on the lower left side of the panel—the pull-down menu can be used to select a different transcript. The graphic at the bottom of the window shows the exon structure of the selected transcript (with or without introns) and the translation start (in green) and stop (in red). The numbers on the exons indicate the translation reading frame, with respect to the genomic sequence displayed in Apollo: `1` refers to the top reading frame in the viewer, `2` to the middle, and `3` to the bottom. The black outline indicates the region of sequence currently visible in the exon detail view; this region is also indicated in the main display with a colored rectangle matching the color of the stripe below the title bar at the top of the EDE. Click on the graphic at the bottom to navigate within the transcript. Alternatively, drag the colored rectangle in the main display near the axis, or use the scroll bar at the right of the EDE. Buttons at the bottom right of the EDE move to the next 5′ or 3′ gene. Click the Find Sequence button to search for amino acid or nucleotide sequences; click on any result in the table to center on the identified sequence, outlined in yellow.

    For more details, consult the EDE section in the Apollo User Guide (see step 20, below) and Support Protocol 1.

19. *View alignments of computational results with annotations.* Apollo does not calculate sequence alignments, but data loaded into Apollo often contain alignments (from BLAST, sim4, etc.). Zooming into the main display shows aligned sequences, but without any gaps. To inspect alignments with appropriate gaps in the reference and result sequences, invoke the Jalview multiple alignment viewer by selecting one or more result features on one strand, then use the RMPM to select either Align Selected Features or Align Same Type Features in Region. A new window will open, showing the selected protein or DNA sequences aligned to that strand's reference sequence and its three-frame virtual translation (Fig. 9.5.10). To see detailed pairwise comparisons between sequences, click on the labels (at the left of the Jalview display)



```
                              1203010   1203020   1203030   1203040   1203050   1203060   1203070   1203080   1203090
GENOMIC:1202618-1203562       CCATCCTTCGCTGCGGGATGACGAGTTTCTGCAGCACAGTTCGCTGTGGGGTCACCAGTTCATCTCCGGCGGAATGGGAGAGGGT-CC
GH03891.5prime:1202628-1203240 CCATCCTTCGCTGCGGGATGACGAGTTTCTGCAGCACAGTTCGCTGTGGGGTCACCAGTTCATCTCCGGCGGAATGGGAGAGGGT-CC
GH21975.5prime:1202784-1203342 CCATCCTTCGCTGCGGGATGACGAGTTTCTGCAGCACAGTTCGCTGTGGGGTCACCAGTTCATCTCCGGCGGAATGGGAGAGGGT-CC
Q9XTY3:1203020-1203508        ............R  D  S  E  N  L  P  D  L  S  S  F  G  Y  K  H  I  S  G  G  A  G  E  G  -E
P91757:1203020-1203520        ............R  D  Q  E  H  L  E  H  S  D  L  H  G  F  Q  S  V  S  G  G  T  A  E  G  A
FRAME1:1202618-1203562         I  L  R  C  G  M  T  S  F  C  S  T  V  R  C  G  V  T  S  S  S  P  A  E  W  E  R  X  X
FRAME2:1202618-1203562        P  S  F  A  A  G  *  R  V  S  A  A  Q  F  A  V  G  S  P  Y  H  L  R  R  N  G  R  G  X  X
FRAME3:1202618-1203562         H  P  S  L  R  D  D  E  F  L  Q  H  S  S  L  W  G  H  Q  F  I  S  G  G  M  G  E  G  X
```

Figure 9.5.10 The Jalview alignment viewer can be called from within Apollo to view multiple alignments of nucleotide and peptide sequences.

of the sequences for which pairwise comparisons are to be viewed, then under the Calc menu, choose Pairwise Alignments.

Many more options are available in Jalview; see the Jalview documentation at http://www.jalview.org/documentation.html for more details (but note that not all of the options described there are available within Apollo).

Introns in annotations are displayed in Jalview as gaps (dashes) flanked by 10 base pairs of the intronic genomic sequence next to the splice site acceptor and/or splice site donor.

20. *Access more detailed instructions*. The Apollo User Guide can be viewed in the Web browser by selecting the Help menu inside Apollo or by visiting *http://www.fruitfly.org/annot/apollo/userguide.html*. This document includes detailed instructions and keyboard shortcuts for the steps described above, information on how to browse annotations from multiple species simultaneously, analyze sequence for restriction sites, and analyze GC content, as well as discussion on other options.

SUPPORT PROTOCOL 1

EDITING GENOME ANNOTATIONS

Apollo, unlike many genome browsers, can also be used as an editor to create, delete, split, and merge annotations of different types, modify intron-exon structures, and add and edit text comments. The modified annotations can then be saved in a flat file or saved directly to the database.

Necessary Resources

Hardware

Unix (Linux, Solaris, or other variety) workstation, PC with Microsoft Windows, or Macintosh with OS X 10.2.3 or later
At least 164 Mb of RAM
Internet connection if querying database interactively
Three-button mouse (see Troubleshooting for information on simulating a three-button mouse)

Software

Apollo program version 1.6.0 or higher
Java JDK 1.4 (included with the Apollo distribution for Unix and Windows)

Files

Configuration files (included with installation); data in GAME XML format (sample file: `example.xml` is included with installation; more data can be obtained from within Apollo transparently over the Internet)

1. *Save annotations*. Immediately after creating or editing annotations, select Save As from the File menu to save the changes. One can choose to Save annotations, evidence (computational evidence), or both annotations and evidence in the selected file by checking the appropriate boxes and clicking OK. To protect against losing changes inadvertently, Apollo also automatically saves data in the `apollo.backup` file in the user's personal `.apollo` directory (see Support Protocol 2) every 20 min.

Modify exon-intron structure of a gene model

2. *Create annotations*. To create a new gene model or transcript, select results on which to base the gene annotation (see step 7 of the Basic Protocol) and drag-and-drop the results into the blue annotation zone. The Action box in the lower left corner of the main display (Fig. 9.5.2) will indicate the editing action, and a ghost image will

appear in the blue zone while dragging. If a previously annotated gene transcript open reading frame (ORF) overlaps the ORF of the new transcript, the new gene model will be added to the existing gene as a new transcript. To add a new transcript as a separate gene, select the result, but instead of dragging, use the right mouse button pop-up menu (RMPM) option Create New Annotation and select the type of the new annotation (protein-coding gene, transposable element, etc.) from the RMPM. To create a new annotation with no supporting evidence, place the mouse in the blue annotation zone at the 5′ edge of the exon (the coordinate will be indicated in the Position box in the lower left of the main display panel) and select Create New Annotation from the RMPM. Choose the annotation type and enter the length of the exon when prompted.

3. *Modify annotations at the exon level.* To duplicate a transcript, e.g., to create an alternatively spliced model, select it and choose Duplicate Transcript from the RMPM (Fig. 9.5.6). To add one or more exons to an existing annotated transcript, hold down the Shift key while dragging result features on top of the transcript. To delete annotated exons or transcripts, select the unwanted feature(s), then select Delete Selection from the RMPM. To merge two exons, hold down the Shift key (Shift-click) to select the exons, and then choose Merge Exons from the RMPM; the intron between the exons will disappear. To transfer exons from one transcript to another, select all exons of the recipient transcript, Shift-click while selecting the exons to be transferred from the donor transcript, and select Move Exon(s) to Transcript from the RMPM. To transfer all exons from one transcript to another, repeat this procedure, but select Merge Transcripts from the RMPM. To split exons from one transcript into two separate transcripts, Shift-click the exons on either side of the intron at the desired site of the split and select Split Transcript from the RMPM.

4. *Modify exon edges in the main display.* To set the 5′ or 3′ edge of an annotated exon to match the 5′ or 3′ edge of a result feature, first select the exon to be modified in the blue annotation zone, then Shift-click the result feature and select Set as 5′ End or Set as 3′ End from the RMPM. To set both 5′ and 3′ boundaries to match a result feature's boundaries, repeat this procedure but select Set Both Ends from the RMPM. To split one exon into two exons, position the mouse where the split is desired and select Split Exon from the RMPM. A 1-bp break will be created that can be modified into a proper intron (see step 5).

5. *Modify exons in the Exon Detail Editor.* Invoke the Exon Detail Editor (EDE; described in step 18 of the Basic Protocol) by selecting the annotation to be edited and choosing Exon Detail Editor from the RMPM. Clicking and dragging on the edges of exons in the EDE will make it possible to modify their boundaries, regardless of whether the new boundary is a proper splice site or not. The ORF will be recalculated and changes will be reflected in the main display as well as in the glyph at the bottom of the EDE. Changes can be monitored in the Sequence Window, described in step 13 of the Basic Protocol. Selecting a base and bringing up the RMPM in the EDE indicates the position of the base within the genomic reference sequence, annotated transcript, and annotated exon, and allows fine-grained changes to exons. For example, trim an exon by selecting the desired edge and choosing Set as 5′ End or Set as 3′ End from the RMPM. To make an intron within an exon, select a base within the exon and choose the Make Intron option from the RMPM. Apollo creates a 1-bp break that can be made into an intron by dragging the edges of the adjacent exons to appropriate splice donor and acceptor sites. To create an exon within an intron, select a base within the intron and choose the Create Exon option from the RMPM. Apollo creates a 1-bp "exon" that can be modified by clicking and dragging the left and right edges. To delete an exon from within the EDE, select the exon

and then choose Delete Exon from the RMPM. To remove an intron and merge two exons, select Merge with 5′ Exon or Merge with 3′ Exon from the RMPM.

Modify information associated with annotations or results

6. *Set the start of translation.* When transcripts are modified, Apollo calculates the longest ORF and sets the start of translation accordingly, then calculates the stop of translation based on the modified ORF (see the Apollo User Guide for details). Thus, if one wishes to set the start or stop of translation manually, it must be done after other changes have been made to the structure of the gene model.

   a. To set the translation start or stop for a transcript in the main display, zoom in to see the green start and red stop codons at the top and bottom of the result panel (Fig. 9.5.3), select the start or stop codon of interest, and then drag it down to the annotated transcript.

   b. To manually reset the start of translation of a protein-coding transcript annotation to reflect the longest ORF, select the annotation and choose Calculate Longest ORF from the RMPM. Within the EDE, one can manually select any base to be the start of translation by clicking on the base and choosing Set Start of Translation from the RMPM. If the translation start is an ATG, the site is marked in green; otherwise, it is marked in purple, and the unconventional start codon will be listed in the Annotation Info Editor (see step 7). Unconventional start codons are automatically translated as methionines. Missing start or stop codons are identified in the main display by green or red arrowheads (Fig. 9.5.5).

7. *Edit text associated with an annotation.* Open the Annotation Info Editor, described in step 12 of the Basic Protocol (Fig. 9.5.7), by selecting an annotation and choosing Annotation Info Editor from the RMPM.

   a. Select the appropriate transcript with the annotation tree at the left of the editor.

   b. Type in the text boxes to change the annotation or transcript symbols or synonyms. Special settings allow users to modify the annotation identifier (ID), but this must be done with caution, and the ID must match the format specified in the tiers configuration file (described in Support Protocol 2).

   c. Use the check boxes to mark an annotation as dicistronic, an annotation or transcript as problematic, or a transcript as finished, or to approve a nonconsensus splice site (non-GT donor or AG acceptor).

   d. Use the pull-down menu to change the Type of entity for the annotation, e.g., gene (protein-coding gene, the default), tRNA, transposable element, etc.

   e. Click the appropriate Edit Comments box to add comments to annotations or individual transcripts. A new Comment window will open that lists existing comments for the annotation or transcript. Click the author/date pair of an existing comment to edit or delete it, or click Add to create a new comment, either by typing in free text or by selecting a comment (specified in the style configuration file, described in Support Protocol 2) from the pull-down menu. Check For Internal Viewing Only to mark the comment as internal. Click Close in the Comment window when finished editing Comments. If the Undo button is clicked in the main Annotation Info Editor, the most recent change made will be discarded. Clicking Undo repeatedly will discard each change going backwards in time.

   f. Click Close in the main Annotation Info Editor to save the changes. After the Annotation Info Editor is closed, it will be impossible to undo the changes made (although, of course, the user can decide whether to save the changes to a file or just exit Apollo and leave the file unchanged).

8. *Indicate translation exceptions.*

   a. To indicate that a stop codon in a transcript is read through, check the box in the Annotation Info Editor next to Read Through Stop Codon with Selenocysteine, and the ORF will be extended to the next stop codon. The original stop codon shows up in pink in the main display and in the EDE.

   b. Set a +1 translational frame shift in the EDE by selecting the nucleotide to be skipped and choosing Set +1 Translational Frame Shift Here from the RMPM. The base will be highlighted in orange in both the main display and EDE, and Apollo will skip over this base and move forward one frame to continue the translation of the sequence. A similar option allows one to set a −1 frame shift to re-read a base pair.

   c. Frame shifts are shown in the Annotation Info Editor for that transcript. To reset the ORF, select the base in the EDE and choose Remove [+1/−1] Translation Frame Shift Here from the RMPM.

9. *Indicate genomic sequencing errors.* A limited number of single-base-pair edits on the genomic sequence can be performed to allow the correct translation of a gene model. To delete a single base pair, select it in the EDE, then choose Adjust For Sequencing Error Here, then select Deletion. The base pair will remain in the main display and EDE, highlighted in orange, as well as in the saved genomic sequence, but any features overlapping this base pair (annotated transcripts, transcript translations) will not include it. To make an insertion, repeat this procedure but select Insertion. A new pop-up will allow the user to choose which nucleotide to add at that position. The new base pair will not appear in the display or in the saved genomic sequence, but the position just downstream of the inserted base pair will be highlighted in orange in both the EDE and main display, and any features overlapping this base pair will include it. To make a 1-bp substitution, repeat the procedure but select Substitution and select the appropriate nucleotide from the popup. Any of these changes will also be detailed in the bottom right panel of the Annotation Info Editor. To remove a change to the genomic sequence, select the highlighted base in the Exon Detail Editor, and select Remove [Deletion/Insertion/Substitution] from the RMPM.

10. *Move or tag results and annotations in the main display.* If a result feature (e.g., an EST) has been assigned to the wrong strand, choose the Move to Other Strand option from the RMPM. As mentioned in step 9 of the Basic Protocol, computational evidence can be tagged with a comment, e.g., indicating it is incomplete or suspect. To tag evidence, select the feature, then from the RMPM select Change Tag from None to the appropriate comment from a new pop-up menu (specified by the style configuration file, described in Support Protocol 2); the feature will then appear cross-hatched in bright pink to indicate that it has a comment associated with it, and the comment will be saved when the work is saved. Annotations can also be tagged in the main display using the RMPM. The Peptide Status options specified in the style configuration menu can be altered, the owner of the annotation changed, or the transcript marked as finished. All of these settings are also indicated in the Annotation Info Editor. See Basic Protocol, step 20 for more details.

    Note that only result types with ResultTag entries in the style configuration file can be tagged.

Add annotations or results

11. As described in step 1, one can save separate annotations and evidence GAME XML files. These files can also be loaded sequentially. For example, when browsing an annotation file, it is possible to layer an evidence file for the corresponding region

by selecting Layer More Results or Annotations from the File menu. One can also layer computational evidence files, like BLAST files, by selecting Computational Analysis Results from the drop-down menu under Choose Data Adapter.

SUPPORT PROTOCOL 2

CONFIGURATION FILES

Apollo relies on a number of configuration files to set the parameters to handle data from different sources. The format and contents of the configuration files are described in detail in the Apollo user manual (included with the Apollo distribution and also available at *http://www.fruitfly.org/annot/apollo/userguide.html*). The configuration files can be confusing, but it is not usually necessary to read, understand, or modify these files. This section describes the organization of the configuration files in case it is necessary to change them—for example, to add a new Chado database to the list of known databases, or to add a new data type to the tiers file. See Troubleshooting section for information about how to address problems that may arise from improper modification of the configuration files.

By default, the configuration files reside in the `conf` subdirectory inside the Apollo main directory. There are three levels of configuration files: global configuration file, style file, and tiers file. The steps below describe how the user can, if necessary, modify Apollo's configuration files. If Apollo is running while any of the configuration files are edited, one will generally have to exit and restart Apollo in order to see the changes made.

Necessary Resources

Hardware

Unix (Linux, Solaris, or other variety) workstation, PC with Microsoft Windows, or Macintosh with OS X 10.2.3 or later
At least 164 Mb of RAM

Software

Apollo program version 1.6.0 or higher
Java JDK 1.4 (included with the Apollo distribution for Unix and Windows)
Any text-editing program

Files

Configuration files (included with installation). If edited, files must be saved in plain text format with line breaks, i.e., as a simple ASCII file. Take care not to add any line breaks in the middle of lines (some text editors tend to do this).

1. *Save personal preferences in the* `.apollo` *directory*. The `.apollo` subdirectory in the user's personal home directory is created the first time that Apollo is launched. Configuration files in the `.apollo` directory override or modify the settings in the default configuration files in the `Apollo/conf` directory. Apollo first reads the global configuration file `apollo.cfg` and style files in its `conf` directory, and then parses the user's personal configuration files in the user's `.apollo` directory (if any).

2. *Modify global settings*. The `apollo.cfg` file sets some of the parameters that apply to every data source (e.g., GAME XML, Chado, etc.) and also tells Apollo which data source–specific configuration files (style files) correspond with which data readers (data adapters). Use a text editor to change parameter settings (for example, to change the detail panel FrameOrientation to vertical). Each data adapter has a `DataAdapterInstall` line in `apollo.cfg`, which tells Apollo where to find the Java class and the style file for that data adapter, e.g.:

```
DataAdapterInstall "apollo.dataadapter.gamexml.
GAMEAdapter" "fly.style"
```

If a new data adapter, or a new style file for an existing data adapter is added to Apollo, `apollo.cfg` will need to be modified.

If changes are made in `apollo.cfg`*, one will need to quit and restart Apollo to see the changes reflected.*

3. *Modify the settings for a particular data source.* Each data source has at least one style file associated with it (e.g., `fly.style` and `game.style` for *Drosophila* annotations, which are in GAME XML format; `ensembl.style` for Ensembl data). Besides setting various display preferences, the style file indicates where to find the tiers file for the particular data source. Use the Preferences editor inside Apollo, by selecting Preferences (Style) from the Edit menu, to change a parameter in the style file (for example, to change EnableEditing to True) and then click Save. The file will be saved by default in the `.apollo` directory in the personal home directory. Click the Cancel or Restore Original buttons to restore the style file to its original state. Besides setting various display preferences, the style file tells Apollo where to find the tiers file for that particular data source. For example, `fly.style` has the line:

   ```
   Types "fly.tiers"
   ```

   If the tiers file has been saved it under a different name, the style file must be hand edited to point to the new tiers file. It is preferable to save the new tiers file as `fly.tiers` in the `.apollo` directory.

   It is possible to import supplementary style files into the main file by using ImportStyle:

   ```
   ImportStyle "extra.style"
   ```

   For example, `fly.style` imports `game.style` and then overrides some of `game.style`'s parameters.

   Do not to try to import a file into itself—this results in a Java error due to stack overflow.

   It is necessary to keep only the settings that one wishes to change in the personal style file in the `.apollo` directory—for example, if one is happy with all of the settings in game.style but wants the result background color to be white instead of black, the `.apollo/game.style` file could contain only the lines

   ```
   FeatureBackgroundColor "white"

   EdgematchColor "black"
   ```

   The rest of the settings will keep the values that they were assigned by the default `game.style` in the `Apollo/conf` directory.

   Sometimes it is necessary to restart Apollo to see changes take effect.

4. *Modify the tiers files.* The tiers file describes the expected data types and how they should be grouped into tiers and displayed. As described in the Basic Protocol step 8, tiers are collections of (generally related) data types that are displayed in the same row. For example, gene predictions made by Genscan and Genie might be grouped in the same tier but displayed in different colors (Fig. 9.5.4), and the colors assigned to each type can be changed from the Types panel. To save this sort of change, select the Save Type Preferences option under the File menu. To add new data types,

change information displayed in the detail panel, etc., use a text editor very carefully to make changes in the tiers file. To add a new data type to the tiers file, first decide which tier (i.e., row) the type is to appear in (or create a new tier). Next, add a new `[Type]` record for the data type. The most critical field in the `[Type]` record is datatype or resulttype, which identifies how this type of data is described in the input. For example:

```
datatype : blastx_masked:aa_SPTR.yeast
```

identifies masked BLASTX hits to a database called `aa_SPTR.yeast`. The corresponding computational results in the GAME XML file might look like this:

```
<result_set id=":830665">

<name>P39515-AE003603.Sept-aa_SPTR.yeast-
blastx_masked</name>

<result_span>

<type>blastx_masked:aa_SPTR.yeast</type>
```

These datatypes are constructed from the program and database identified in the computational_analysis record in the GAME XML input:

```
<computational_analysis>

  <program>blastx_masked</program>

  <database>aa_SPTR.yeast</database>
```

It may be necessary to restart Apollo after editing the tiers file. Apollo preferentially uses the tiers files that it finds in the `.apollo` *directory. If there is no tiers file in* `.apollo` *for a given style, it uses the default tiers file in the* `Apollo/conf` *directory.*

ALTERNATE PROTOCOL 1

BROWSING GENOME ANNOTATIONS IN CHADO XML FILES

The Chado XML adapter allows Apollo to read files in Chado XML format (*UNIT 9.6*), e.g., if one wishes to set up a private or customized genome database. Tools such as XORT (*UNIT 9.6*) can be used to export Chado XML from a Chado database or import Chado XML back into the database. Apollo's Chado XML adapter was developed to work well with FlyBase's Chado XML format; if one's own Chado XML is significantly different, one may need to modify the Chado XML adapter.

Note that the Chado XML adapter, like the GAME XML adapter, can be used to read in or write out separate files for the results and the annotations (see steps 1 and 11 in Support Protocol 1).

Necessary Resources

Hardware

Unix (Linux, Solaris, or other variety) workstation, PC with Microsoft Windows, or Macintosh with OS X 10.2.3 or later
At least 164 Mb of RAM
Internet connection
Three-button mouse (see Troubleshooting for information on simulating a three-button mouse)

Software

Apollo program version 1.6.0 or higher
Java JDK 1.4 (included with the Apollo distribution for Unix and Windows)

Files

Configuration files (included with installation); data in Chado XML format (sample file: `CG16983.chado.xml` are included with installation). Files must be in plain-text format with line breaks (simple ASCII files).

1. Download Apollo and start an Apollo session as described in the Basic Protocol.
2. *Load a region of interest.* From the pull-down menu under Choose Data Adapter (Fig. 9.5.1), select "Chado XML file" and enter a file name in the box. For this example, load the `CG16983.chado.xml` sample file that downloaded into the `Apollo/data/` directory during installation. Then click the OK button at the bottom of the window.
3. Proceed to step 4 of the Basic Protocol.

BROWSING GENOME ANNOTATIONS FROM A CHADO DATABASE

ALTERNATE PROTOCOL 2

The Chado adapter allows Apollo to read data directly from a Chado database (*UNIT 9.6*) via Java Database Connectivity (JDBC). This direct connection allows one to modify a customized genome database without having to dump or load flat files that can become stale and asynchronous with the database, although Chado writeback via JDBC is still a work in progress. Using Chado databases will require modification of some configuration files.

Necessary Resources

Hardware

Unix (Linux, Solaris, or other variety) workstation
PC with Microsoft Windows, or Macintosh with OS X 10.2.3 or later
At least 164 Mb of RAM
Internet connection
Three-button mouse (see Troubleshooting for information on simulating a three-button mouse)

Software

Apollo program version 1.6.0 or higher
Java JDK 1.4 (included with the Apollo distribution for Unix and Windows)
Any text-editing program

Files

Configuration files (included with installation). Edited files must be saved in plain text format with line breaks, i.e, as simple ASCII files.

1. Download Apollo as described in the Basic Protocol.
2. *Modify chadodb element in the configuration file.* Before starting Apollo it will be necessary to edit text in the configuration file `conf/chado-adapter.xml` in order to fill in the "chadodb" element. An example is shown in Figure 9.5.11. Add a

```
<!-- Indiana public FlyBase server r4.2 -->

     <chadodb>

        <name>FlyBase Chado Indiana public server r4.2</name>

        <adapter>apollo.dataadapter.chado.jdbc.PostgresChadoAdapter</adapter>

        <url>jdbc:postgresql://flybase.org:5432/dmel_r4.2</url>

        <dbName>dmel_r4.2</dbName>

        <dbInstance>flybaseInstance</dbInstance>

        <style>fly.style</style>

        <default-command-line-db>true</default-command-line-db>

     </chadodb>
```

Figure 9.5.11 Part of the Chado configuration file `conf/chado-adapter.xml` containing the "chadodb" element. For definitions, see Table 9.5.1.

Table 9.5.1 Definitions of Terms in the Configuration File `conf/chado-adapter.xml` (see Fig. 9.5.11)

Term	Definition
`name`	The label that will appear in the drop-down list of databases
`url`	URL for the database to be used
`adapter`	Identifies the Apollo class to use for the database. If the database uses Postgres, use `apollo.dataadapter.chado.jdbc.PostgresChadoAdapter`; if it uses Sybase, use `apollo.dataadapter.chado.jdbc.SybaseChadoAdapter`.
`dbName`	The database name to use on the server
`dbUser`	The database user/login
`dbInstance`	Identifies the type of Chado database (see Alternate Protocol 2, step 3)
`style`	Style configuration file for this database (see Support Protocol 2)
`default-command-line-db`	Database used when Apollo is run from the command line when set to "true"

similar entry to `chado-adapter.xml` describing the particular Chado database to be used, referring to the definitions in Table 9.5.1.

If using a different database vendor, try Postgres and Sybase to see if one of them works. If these changes do not work, see Troubleshooting for instructions on how to contact the Apollo mailing list.

3. *Modify chadoInstance element in the configuration file.* Edit text in the file `conf/chado-adapter.xml` to fill in the "chadoInstance" element. Whereas chadodb captures how to connect to a Chado database, chadoInstance captures the differences in Chado instantiations. For example, if there were several servers for the same Chado database, they would have different chadodb elements, but the same chadoInstance. The chadoInstance element captures two things: a Java class to use, and a list of annotation types in the particular Chado database. An example is shown

```
<chadoInstance id="flybaseInstance">
      <inheritsInstance>referenceInstance</inheritsInstance>
      <featureCV>sequence</featureCV>
      <relationshipCV>relationship</relationshipCV>
      <propertyTypeCV>feature_property</propertyTypeCV>
      <partOfCvTerm>part_of</partOfCvTerm>
      <transProtRelationTerm>derives_from</transProtRelationTerm>
      <sequenceTypes>
            <type>
                  <name>chromosome_arm</name>
                  <useStartAndEnd>true</useStartAndEnd>
                  <queryForValueList>false</queryForValueList>
                  <values>4,2L,2R,3L,3R,X,2h,3h,4h,Xh,Yh,U</values>
            </type>
            <type>gene</type>
            <type>golden_path_region</type>
      </sequenceTypes>
      <writebackXmlTemplateFile>transactionWritebackTemplate_crm.
            xml</writebackXmlTemplateFile>
      <retrieveAnnotations>true</retrieveAnnotations>
      <genePredictionPrograms>
            <program>genscan_masked</program>
            <program>genie_masked</program>
      </genePredictionPrograms>
      <searchHitPrograms>
            <program>sim4</program>
            <program>%blastx%</program>
      </searchHitPrograms>
      <searchHitsHaveFeatLocs>true</searchHitsHaveFeatLocs>
      <oneLevelResultPrograms>
            <program>primer3</program>
      </oneLevelResultPrograms>
      <clsName>apollo.dataadapter.chado.jdbc.FlybaseChadoInstance
            </clsName>
      <oneLevelAnnotTypes>
            <type>promoter</type>
            <type>insertion site</type>
            <type>transposable_element</type>
            <type>remark</type>
            <type>repeat_region</type>
      </oneLevelAnnotTypes>
      <threeLevelAnnotTypes>
            <type>gene</type>
            <type>pseudogene</type>
            <type>tRNA</type>
            <type>rRNA</type>
            <type>miRNA</type>
      </threeLevelAnnotTypes>
</chadoInstance>
```

Figure 9.5.12 Part of the Chado configuration file `conf/chado-adapter.xml` containing the “chadoInstance” element.

in Figure 9.5.12. Add a similar entry to `chado-adapter.xml` describing the particular Chado database to be used.

> *`clsName` is the class name to be used for the particular Chado instance. There are currently three Chado instance classes:* `apollo.dataadapter.chado.jdbc.TigrSybilChadoInstance`*;* `apollo.dataadapter.chado.jdbc.FlybaseChadoInstance`*; and* `apollo.dataadapter.chado.jdbc.RiceInstance`*. These Java instances capture differences in schema and ontologies between these three Chado instances.*
>
> `oneLevelAnnotTypes` *and* `threeLevelAnnotTypes` *list annotation types consisting of one and three levels. One-level annotation types are currently used by Apollo for annotations such as transposons or transposon insertions. Three-level annotation types are used for hierarchical annotations like genes/transcripts/exons. Eventually, the annotation containment hierarchy will be determined automatically using Sequence Ontology terms.*
>
> *Because Chado database instantiations may vary, both in schema and ontologies, it is possible that these changes to configurations will not be enough. RiceInstance is the most up-to-date of the three clsNames, so start by trying that with the database; if it does not work, see the Troubleshooting section for instructions on how to contact the Apollo mailing list.*

4. *Choose Chado database to load data over the Internet*. Select the database to be used from the pull-down menu next to Chado Database.

5. *Select a region to display*. The list for Type of Region includes the "sequenceTypes" listed in `chado-adapter.xml` that correspond to the top-level types in the Chado database. For example, one might query the FlyBase Chado database by gene or by golden_path_region, a 250 to 350 kb region of genomic sequence. Enter the identifier for the Type next to Region ID and click OK. Be patient; large regions may take time to retrieve.

6. Proceed to step 4 of the Basic Protocol.

ALTERNATE PROTOCOL 3

BROWSING GENOME ANNOTATIONS FROM AN ENSEMBL DATABASE

The Ensembl project has created public MySQL databases of annotated genomes for many species, including human, mouse, mosquito, rat, zebrafish, and fugu. Each species is stored in a separate, publicly accessible database. Apollo's Ensembl (EnsJ) adapter can read these databases, as well as any other MySQL database that uses Ensembl Schema 20 or above. This protocol describes how to read data interactively over the Internet from these databases.

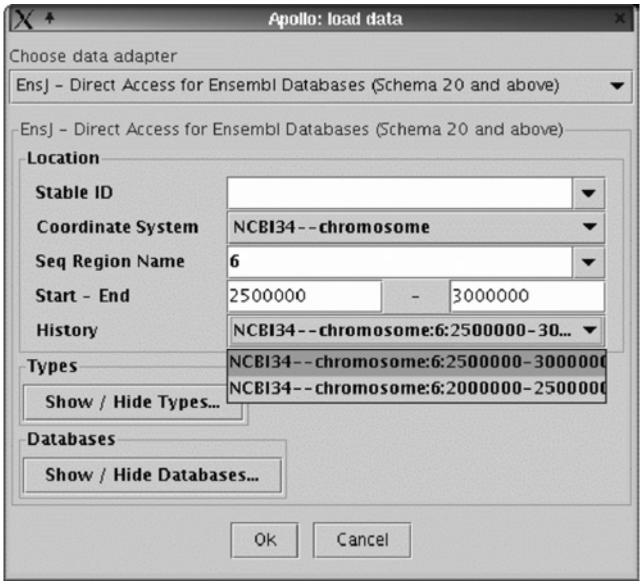
Necessary Resources

Hardware

Unix (Linux, Solaris, or other variety) workstation, PC with Microsoft Windows, or Macintosh with OS X 10.2.3 or later
At least 164 Mb of RAM
Internet connection
Three-button mouse (see Troubleshooting for information on simulating a three-button mouse)

Software

Apollo program version 1.6.0 or higher
Java JDK 1.4 (included with the Apollo distribution for Unix and Windows)



Choose data adapter
EnsJ - Direct Access for Ensembl Databases (Schema 20 and above)

EnsJ - Direct Access for Ensembl Databases (Schema 20 and above)

Stable ID
Coordinate System NCBI34--chromosome
Seq Region Name 6
Start - End 2500000 - 3000000
History NCBI34--chromosome:6:2500000-30...
NCBI34--chromosome:6:2500000-3000000
NCBI34--chromosome:6:2000000-2500000

Figure 9.5.13 Loading data into Apollo from an Ensembl database via the Internet.

Files

Configuration files (included with installation)

1. Download Apollo and start an Apollo session as described in the Basic Protocol.
2. Read data from Ensembl over the Internet. From the pull-down menu, under Choose Dataadapter, select “EnsJ - Direct Access for Ensembl Databases (Schema 20 and above),” as shown in Fig. 9.5.13). Expand the Databases panel by clicking the Show/Hide Databases button. For Ensembl data, the Host should be set to `ensembldb.sanger.ac.uk`, the Port to 3306, and the User name to anonymous (no password is necessary). Data in other Ensembl-type databases will have other Host and Port settings.

   When the Ensembl adapter is requested, it takes a moment to appear. A pop-up error message saying that one did not select a database may be displayed. Click the OK button to continue. Sometimes Apollo will stop responding; see Troubleshooting for more information.

3. *Select a region to display*. At the bottom of the Databases panel, next to Ensembl Database Name, select an available Ensembl database from the pull-down menu. Under the Location Panel, a specific gene can be chosen for viewing, either by Stable ID (e.g., ENSG00000187981) or by specifying a Coordinate System and Chromosome Sequence Region (e.g., one could choose “chromosome—NCBI35” for the coordinate system, “6” for the seq region name, and “500000-100000” for start-end). The History pull-down should contain a list of preset regions to try.
4. *Choose which features to display*. Ensembl features are categorized into various Types, as described in step 8 of the Basic Protocol. To determine which Types to display in the region loaded, click the Show/Hide Tracks button. This will bring up an expanded Types panel. The numbers beside each type—e.g., Genes (23578)—represent the number of features of that type in the whole Ensembl database. Select at least one type of feature to view (Genes is a good starting point). Click the OK button at the bottom of the panel, and Apollo will bring up the selected annotations in the chosen region.

Be aware that the Ensembl data adapter uses lazy loading, which means it delays loading information—such as the actual sequence residues—until it is requested. This makes the initial load faster, but when the user selects a feature and attempts to zoom in, there will be a pause while the sequence residues are fetched from the server.

5. Proceed to step 4 of the Basic Protocol.

ALTERNATE PROTOCOL 4

BROWSING GENOME ANNOTATIONS FROM ENSEMBL GFF FILES

Gene Finding Format (GFF) is a simple tab-delimited text format for storing genomic annotations. A version of GFF called Ensembl GFF is used at the Sanger Institute for storing the annotations in flat files. Note that there is more than one type of GFF, and Apollo only supports the Ensembl type (as described in *http://www.sanger.ac.uk/Software/formats/GFF*).

The GFF adapter can be a useful starting point for groups that are trying to import data into Apollo, since many analysis tools produce output that is in GFF or GFF-like format.

Necessary Resources

Hardware

Unix (Linux, Solaris, or other variety) workstation, PC with Microsoft Windows, or Macintosh with OS X 10.2.3 or later
At least 164 Mb of RAM
Internet connection if querying database interactively
Three-button mouse (see Troubleshooting for information on simulating a three-button mouse)

Software

Apollo program version 1.5.2 or higher
Java JDK 1.4 (included with the Apollo distribution for Unix and Windows)

Files

Configuration files (included with installation)
Data in Ensembl GFF format (sample file: `chr.200000-400000.gff` is included with installation). If edited, files must be saved in plain text format with line breaks, i.e., as simple ASCII files.

1. Download Apollo and start an Apollo session as described in the Basic Protocol.

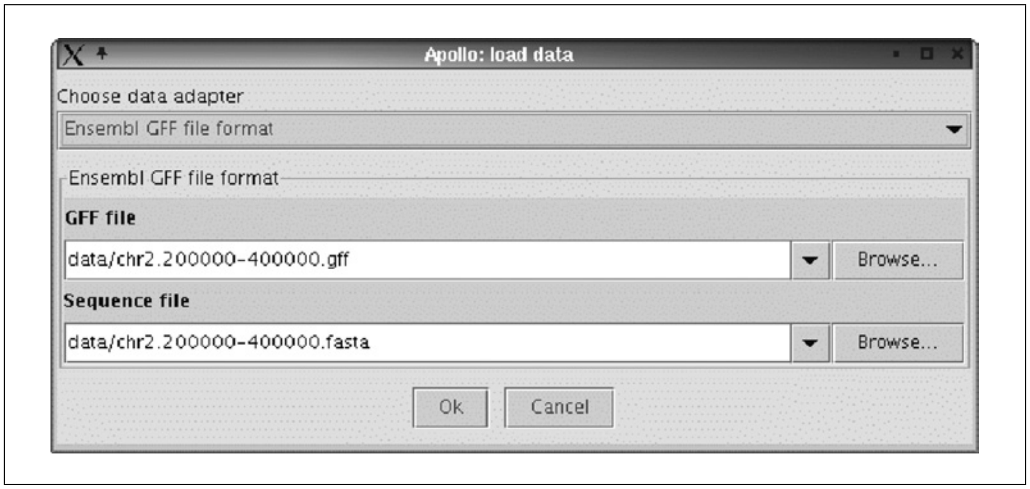
2. *Read a GFF file*. From the pull-down menu, under Choose data adapter, choose the Ensembl GFF option (Fig. 9.5.14). Type the GFF file name in the text box or press the Browse button to bring up a file chooser.

3. In order to see the genomic sequence of the region, one may optionally enter a FASTA-format sequence file name for the region that corresponds to the GFF file.

4. Proceed to step 4 of the Basic Protocol.

5. *Optional*. To save data in GFF format, choose the Save As menu item from the File menu and then select Ensembl GFF file format from the Choose Data Adapter pull-down menu. It is possible at this point to supply names for the GFF file and (optionally) the FASTA-format sequence file.

   Note that Ensembl GFF format is not rich enough to support curated annotation, so even if editing is turned on (which, by default, is not enabled in the Ensembl GFF data adapter), it will not be possible to create and save curated annotations on GFF-format data. Loading data from another data source and saving as GFF is also not recommended, as one will lose much of the information.



Choose data adapter
Ensembl GFF file format

Ensembl GFF file format
GFF file
data/chr2.200000-400000.gff
Sequence file
data/chr2.200000-400000.fasta

Figure 9.5.14 Loading data into Apollo from a file in GFF format.

ALTERNATE PROTOCOL 5

BROWSING GENOME ANNOTATIONS FROM GenBank OR EMBL FILES

Apollo can read GenBank and EMBL format from files or directly over the Internet (given an accession number). The GenBank and EMBL adapters are still under development, and are not as robust as the GAME XML adapter. In particular, reading as GenBank or EMBL and then saving in GenBank format does not yet perfectly preserve everything from the original source.

Necessary Resources

Hardware

Unix (Linux, Solaris, or other variety) workstation, PC with Microsoft Windows, or Macintosh with OS X 10.2.3 or later
At least 164 Mb of RAM
Internet connection if querying database interactively
Three-button mouse (see Troubleshooting for information on simulating a three-button mouse)

Software

Apollo program version 1.6.0 or higher
Java JDK 1.4 (included with the Apollo distribution for Unix and Windows)

Files

Configuration files (included with installation)
Data in GenBank or EMBL format. If edited, files must be saved in plain text format with line breaks (i.e., as simple ASCII files).

1. Download Apollo and start an Apollo session as described in the Basic Protocol.

2. *Load from a file or via the Internet*. To load from a file, choose the GenBank/EMBL option from the “Choose data adapter” menu (Fig. 9.5.1), then select the File tab. Type the file name in the text box or press the Browse button to bring up a file chooser. Note that Apollo can read only one GenBank or EMBL record at a time, so if the file contains a series of records, only the first one will be read. To load via the Internet, if the GenBank or EMBL accession number of the sequence of interest is known, select the Accession tab instead of the File tab. Choose the appropriate Database (GenBank or EMBL) from the pull-down list, enter the accession number in the text box, and press the OK button. The GenBank or EMBL data will be downloaded from

the Internet and displayed in Apollo. If the accession number requested is not found, a pop-up message saying Read Failed will appear.

3. Proceed to step 4 of the Basic Protocol.

4. *Optional*. It is possible to save data in GenBank format regardless of whether it was loaded from GenBank or from another data source. Select Save As from the File menu and choose GenBank/EMBL Format from the pull-down menu. One can choose tabular format, supplying a directory name in which to save the tables (this is generally used for submitting annotations to GenBank), or the more familiar GenBank human-readable format, which generates a single file.

COMMENTARY

Background Information

In order to provide biological insight, DNA sequences need annotation. The process of annotation starts with computational analyses that look for sequences that seem to correspond to interesting biological features. There are a variety of tools available for this task—including those that look for similarity to known gene transcripts or other types of features, such as BLAST (*UNITS 3.3 & 3.4*), sim4 (Florea et al., 1998), and BLAT (*UNIT 1.4*)—and those that construct abstract models to identify possible regions of interest, such as Genscan (Burge and Karlin, 1997), FgenesH (Salamov and Solovyev, 2000), promoter predictors, etc. While automated analyses point to regions of interest on the sequence, sequence analysis tools still fall short of the expert knowledge that biologists can bring to the task of annotation. Thus, Web browsers such as the UCSC Genome Browser (*UNIT 1.4*) and NCBI Map Viewer (*UNIT 1.5*) are helpful for browsing computational results, but do not offer the possibility of easily synthesizing computational data into a coherent model of the gene. Apollo provides interactive tools to allow biologists to curate the preliminary gene model and other annotations generated by automatic analysis tools, both to ensure the correctness of the annotations themselves and to deepen the current understanding of biology by connecting these annotations to the biology of the organism. Apollo was written in Java to make it easily portable to multiple sites with different data-management environments. Apollo is the annotation editor for the Generic Model Organism Database (GMOD) project (Stein et al., 2002) and is the editing tool of choice for an increasing number of genome centers, many of which have made changes to better suit their individual needs. These changes range from adding new data types to the tiers configuration file to writing new adapters to enable Apollo to read annotation data from proprietary databases.

The first major public release of Apollo (version 1.2.3) was made available in December, 2002. Since then, new updates of the Apollo software have been released every few months at the Apollo Web site. Apollo has been downloaded over 5000 times. The most recent public release at the time of publication is Version 1.6.0 (November, 2005), described in this document.

Troubleshooting

Apollo mailing list

To submit questions about Apollo, send a message to the Apollo mailing list. It is first necessary to visit *http://mail.fruitfly.org/mailman/listinfo/apollo* and join the mailing list before sending a question.

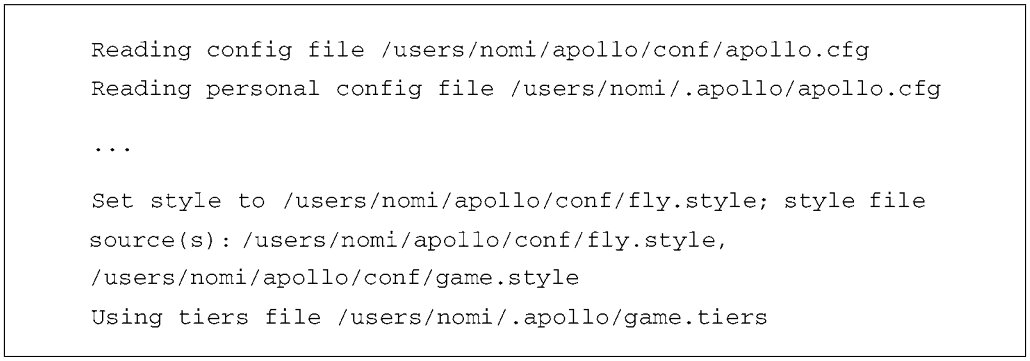
Apollo and memory use

Apollo makes it possible to load and examine a large amount of annotation data at once. However, if the local computer does not have very much memory, one may run out of memory when trying to load a large region. Apollo should be able to load relatively small regions (e.g., for *Drosophila*, 100 kb) even on a low-memory computer, but it will run very slowly. When the machine is about to run out of memory, a warning message should pop up, in which case all work should be immediately saved, and the user should exit and restart Apollo.

Memory problems may also be encountered by keeping the same Apollo window open and repeatedly loading new regions. Again, the easiest solution is to quit and restart.

Simulating a three-button mouse

The hardware requirements for Apollo include a three-button mouse; however, it is possible to use Apollo with a one-button mouse. If using a Mac with a single-button mouse, a right mouse click can be simulated by holding



```
Reading config file /users/nomi/apollo/conf/apollo.cfg
Reading personal config file /users/nomi/.apollo/apollo.cfg

...

Set style to /users/nomi/apollo/conf/fly.style; style file
source(s): /users/nomi/apollo/conf/fly.style,
/users/nomi/apollo/conf/game.style
Using tiers file /users/nomi/.apollo/game.tiers
```

Figure 9.5.15 Messages printed to Java console window indicating where Apollo finds its configuration files. See Support Protocol 2 for a description of the files.

down the Ctrl or Alt key while clicking the mouse; a middle mouse click can be simulated by holding down the Apple key while clicking the mouse. On some laptop computers running Windows, the middle mouse button pops up a scrollbar instead of bringing up Apollo's middle mouse popup menu. To simulate a middle mouse click, use the Alt key with the left mouse button.

Configuration files

For instructions on adding a new data type to the tiers file, see step 4 of Support Protocol 2. Most users should not need to modify the configuration files. If they have been changed, be sure that no line breaks have been inserted in any lines—this is a common source of problems.

When trying to track down problems relating to the configuration files, it is helpful to figure out which configuration files are being used. The console window reports which configuration files Apollo is using. On Mac OS X, it is possible to start a Java console by running the Console application in the `Applications/Utilities` folder. On Windows, go to the Control Panel, select Internet Options, select Advanced, scroll down to Microsoft VM, and check "Java console enabled." Windows will have to be restarted in order to see the Java console. The console should come up when starting Apollo. On Unix, the text output will go to the shell window from which Apollo was invoked. The messages printed to the console window (Fig. 9.5.15) will say where Apollo finds its configuration files. This can be useful in troubleshooting—many users do not realize that they actually have personal configuration files in their `.apollo` directories that are causing problems. If having Apollo problems that might conceivably be related to configuration files, the best course of action is to find and remove the `.apollo` folder, uninstall Apollo, and install it again.

Ensembl (EnsJ) adapter

The Ensembl data adapter can be tricky to use because the Ensembl databases are in constant flux. Be sure to select a database before trying to access a region. This can be done by clicking the Show/Hide Databases button to show the database selection panel.

For each data adapter, Apollo saves a history of where data has been loaded (and where it has been saved). This history file is saved in the `.apollo` directory as `apollo.history`. Normally, this file does not need to be changed, but with the Ensembl adapter, the database names stored in the history become out-of-date, so the best recourse if having trouble using the Ensembl adapter is to remove `.apollo/apollo.history` and restart Apollo.

To ask for help with trouble accessing the Ensembl databases, use the Apollo mailing list (see above).

GAME XML adapter

The GAME Adapter can read GAME XML data directly from a file or by gene name, location, etc. The latter options access the FlyBase database over the Internet. Occasionally, the FlyBase database is down or too busy to answer, in which case a message may appear saying that the region requested could not be found. If that happens, try the query again later.

The *Drosophila* annotation data are very dense, so it is best to load regions of no more than 200 kb. Attempts to load bigger regions may cause the local computer to run out of memory (see above).

Acknowledgements

This work was supported by NIH grant HG00739 to FlyBase. The authors of this unit would like to thank M. Gibson for his assistance with the Chado JDBC adapter section, and M. Gibson, S. Lewis, G. Wu, J. Crabtree, V. Iyer, J. Day-Richter, M.E. Clamp, and S.M.J. Searle for their work developing the Apollo annotation tool. The authors also wish to thank G.M. Rubin.

Literature Cited

Burge, C. and Karlin, S. 1997. Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.* 268:78-94.

Drysdale, R.A., Crosby, M.A., Gelbart, W., Campbell, K., Emmert, D., Matthews, B., Russo, S., Schroeder, A., Smutniak, F., Zhang, P., Zhou, P., Zytkovicz, M., Ashburner, M., de Grey, A., Foulger, R., Millburn, G., Sutherland, D., Yamada, C., Kaufman, T., Matthews, K., DeAngelo, A., Cook, R.K., Gilbert, D., Goodman, J., Grumbling, G., Sheth, H., Strelets, V., Rubin, G., Gibson, M., Harris, N., Lewis, S., Misra, S., and Shu, S.Q. 2005. FlyBase: Genes and gene models. *Nucl. Acids Res.* 33:D390-D395.

Florea, L., Hartzell, G., Zhang, Z., Rubin, G.M., and Miller, W. 1998. A computer program for aligning a cDNA sequence with a genomic DNA sequence. *Genome Res.* 8:967-974.

Hubbard, T., Andrews, D., Caccamo, M., Cameron, G., Chen, Y., Clamp, M., Clarke, L., Coates, G., Cox, T., Cunningham, F., Curwen, V., Cutts, T., Down, T., Durbin, R., Fernandez-Suarez, X.M., Gilbert, J., Hammond, M., Herrero, J., Hotz, H., Howe, K., Iyer, V., Jekosch, K., Kahari, A., Kasprzyk, A., Keefe, D., Keenan, S., Kokocinsci, F., London, D., Longden, I., McVicker, G., Melsopp, C., Meidl, P., Potter, S., Proctor, G., Rae, M., Rios, D., Schuster, M., Searle, S., Severin, J., Slater, G., Smedley, D., Smith, J., Spooner, W., Stabenau, A., Stalker, J., Storey, R., Trevanion, S., Ureta-Vidal, A., Vogel, J., White, S., Woodwark, C., and Birney, E. 2005. Ensembl 2005. *Nucl. Acids Res.* 33:D447-D453.

Lewis, S.E., Searle, S.M., Harris, N., Gibson, M., Iyer, V., Richter, J., Wiel, C., Bayraktaroglu, L., Birney, E., Crosby, M.A., Kaminker, J.S., Matthews, B.B., Prochnik, S.E., Smithy, C.D., Tupy, J.L., Rubin, G.M., Misra, S., Mungall, C.J., and Clamp, M.E. 2002. Apollo: A sequence annotation editor. *Genome Biol.* 3: RESEARCH0082.

Salamov, A.A. and Solovyev, V.V. 2000. Ab initio gene finding in *Drosophila* genomic DNA. *Genome Res.* 10:516-522.

Stein, L.D., Mungall, C., Shu, S., Caudy, M., Mangone, M., Day, A., Nickerson, E., Stajich, J.E., Harris, T.W., Arva, A., and Lewis, S. 2002. The generic genome browser: A building block for a model organism system database. *Genome Res.* 12:1599-1610.

Key References

Lewis et al., 2002. See above.

This article gives background on Apollo development and features.

Internet Resources

http://www.fruitfly.org/annot/apollo/

Download Apollo.

http://www.fruitfly.org/annot/apollo/userguide.html

Get more detailed and current information about Apollo features and usage.

http://mail.fruitfly.org/mailman/listinfo/apollo

Join the Apollo mailing list to ask questions or be notified of new releases.

http://www.jalview.org/documentation.html

Get detailed information about using Jalview, the alignment viewer in Apollo.

Contributed by Sima Misra and
Nomi Harris
University of California
Berkeley, California

Using Chado to Store Genome Annotation Data

Chado was originally developed to integrate the information resources in two independent *Drosophila* databases. Since then it has evolved into a powerful ontology-driven genome database schema in response to feedback from end users and the bioinformatics community. It is an integral component of the NIH/USDA ARS-funded Generic Model Organism Database (GMOD) project, and now supplies the database infrastructure for numerous software packages both within and outside the GMOD project. This Chado-compatible packages include the Gbrowse Web-based genome annotation browser (Stein et al., 2002) and Apollo (Lewis et al., 2002; also see *UNIT 9.5*), a genome annotation viewer and editor.

These protocols describe how to use the Chado relational database schema to store genome annotation data, in both the Unix/Linux/Mac OS X (Basic Protocol 1) and Windows environments (Support Protocol). This includes installing Chado and XORT in the Unix/Linux environment (Basic Protocol 1), obtaining the Chado schema data definition language script (DDL) and initializing the database (Basic Protocol 2), importing genome annotation data into the database (Basic Protocols 2 and 3), running useful queries across the data (Basic Protocol 4), and, finally, exporting the data into different standard data formats (Basic Protocol 5). An additional protocol will guide the reader through the acquisition and loading of genome data in GenBank flat-file format into Chado (Basic Protocol 3). A schematic representation of the organizational relationship between the protocols and the data flow for Chado is shown in Figure 9.6.1.

These protocols are intended for biologists and computer scientists who have experience working with whole-genome data as well as a basic working knowledge of Perl and RDBMS principles. It is assumed that the user has access to a computer running a Unix-based or PC/Windows operating system with the PostgreSQL RDBMS installed, and has permissions to create databases. All of the examples presented here come from FlyBase, which is a public database of genetic and molecular data for *Drosophila*.

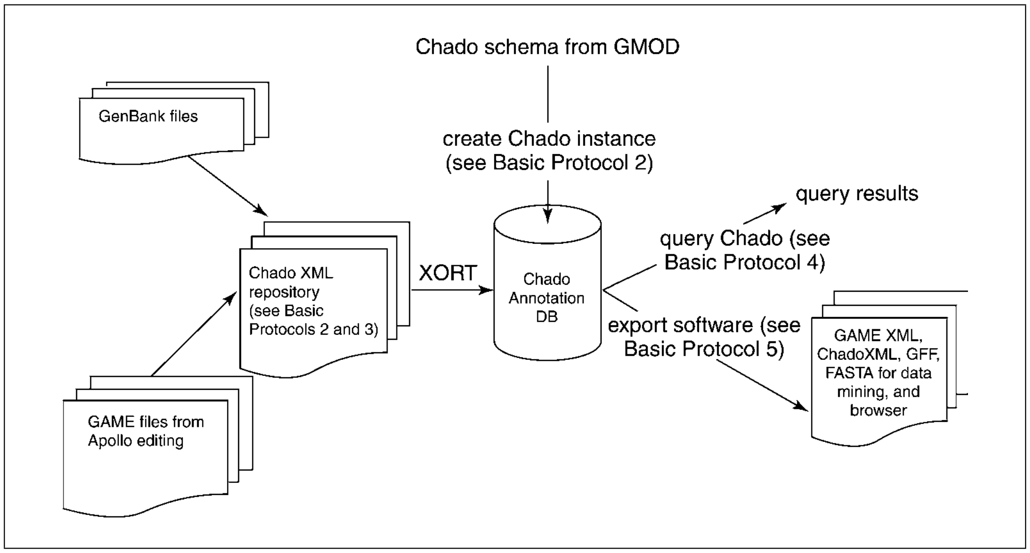

Chado schema from GMOD
GenBank files
create Chado instance (see Basic Protocol 2)
query results
query Chado (see Basic Protocol 4)
Chado XML repository (see Basic Protocols 2 and 3)
XORT
Chado Annotation DB
export software (see Basic Protocol 5)
GAME XML, ChadoXML, GFF, FASTA for data mining, and browser
GAME files from Apollo editing

Figure 9.6.1 A schematic representation of the protocols and the organizational relationship between the protocols and data flow for Chado.

Contributed by Pinglei Zhou, David Emmert, and Peili Zhang

Current Protocols in Bioinformatics (2005) 9.6.1-9.6.28

BASIC PROTOCOL 1

INSTALLING CHADO AND XORT IN THE UNIX/LINUX ENVIRONMENT

This protocol describes the installation of Chado and XORT in the Unix/Linux environment and creation a new instance of a Chado database. Support Protocol describes the corresponding procedures for a Windows computer.

Necessary Resources

Hardware

Any recent computer system running Macintosh OS X, Solaris, Linux, or other Unix variant

Software

Standard software:

PostgreSQL and BioPerl: Instructions for the installation of PostgreSQL and BioPerl under a Unix/Linux environment are beyond the scope of this unit; more information on these installation procedures can be found at *http://www.postgresql.org/docs/7.4/static/installation.html* (for PostgreSQL) and *http://www.bioperl.org* and *http://bioperl.org/Core/Latest/INSTALL* (for BioPerl)

The following software packages should be preinstalled:
Perl v. 5.0 or higher (*http://www.perl.org*)
PostgreSQL v. 7.3.2 or higher (*http://www.postgresql.org*)
Java 1.4.1 or higher (*http://java.sun.com*)

Nonstandard software:

These packages will need to be installed:
XML Parser (*http://search.cpan.org/~kmacleod/libxml-perl-0.08/*)
XML::DOM (*http://search.cpan.org/~tjmather/XML-DOM-1.43/*)
DBI (*http://search.cpan.org/~timb/DBI/*)
DBD-Pg (*http://search.cpan.org/dist/DBD-Pg/*)
XML-XORT (*http://www.gmod.org*)
BioPerl (*http://www.bioperl.org/Core/Latest/index.shtml*)

For the purposes of this demonstration, it is assumed that the PostgreSQL database has been installed and that a user account has been created and has been granted "create database" privileges. It is also assumed that the XORT/Gamebridge package will be saved into the `/home/work` directory.

1. Download XORT. The XORT package can either be downloaded via SourceForge CVS or via FTP.

   a. To download via CVS, use the following command:

   ```
   $ cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/
     cvsroot/gmod \co XML-XORT
   ```

   b. To download via FTP, use the following command:

   ```
   http://prdownloads.sourceforge.net/gmod/
   ```

2. Install XORT using the following commands:

```
$ cd /home/work/XML-XORT
$ perl Makefile.PL
```

3. `Makefile.PL` will then take the user through the configuration of XORT by asking a series of questions. In most cases, one can use the default configuration (presented in square brackets below).

```
What is the database name? chado
What is the database username? [zhou]
What is the password for ‘zhou’? zhoupg
What is the database host? [localhost]
What is your database port? [5432]
Where will the tmp directory go?
  [/home/work/XML-XORT/tmp]
Where will the conf directory go?
  [/home/work/XML-XORT/conf]
Where is the DDL file?
  [/home/work/XML-XORT/examples/chado.ddl]
Where do you want to install XORT if other than
  default, press ENTER if default: [/tmp/xort]
$ make
$ make install
```

Besides installing the XORT executables, the steps above are required to collect database information that will be written to a very important configuration file called `chado.properties` *(default location:* `/home/work/XML-XORT/conf/chado.properties`*). The* `chado.properties` *file contains all the necessary information for XORT to connect to the Chado database that one is working with. If at some point it is necessary to connect to a different instance of Chado on the PostgreSQL server, one can either edit the* `chado.properties` *file to indicate the new Chado database name or create a new file, with the suffix* `.properties` *(e.g.,* `my_new_chado.properties`*). In order to be used by XORT, any* `.properties` *file needs to be located in the configuration directory (*`/home/work/XML-XORT/conf/`*). When executing XORT (see below), the* `-d` *parameter specifies which* `.properties` *file to use (e.g.,* `-d my_new_chado`*). For more documentation, see* `/home/work/XML-XORT/doc/readme_xort`*.*

BUILDING A CHADO ANNOTATION DATABASE

BASIC PROTOCOL 2

This protocol describes how to download the Chado DDL and use it to create an empty Chado database instance on a PostgreSQL server. It then describes how to convert a GAME XML file (the output format from Apollo) into ChadoXML, the XML format for all data going directly into and coming directly out of a Chado database when using XORT. Finally, the protocol describes how to use the XORT validator and loader utilities to validate ChadoXML and load it into the Chado database.

Necessary Resources

Hardware

Any recent computer system running Macintosh OS X, Solaris, Linux, or other Unix variant

Software

Standard software:

PostgreSQL and BioPerl: Instructions for the installation of PostgreSQL and BioPerl under a Unix/Linux environment are beyond the scope of this unit; more information on these installation procedures can be found at *http://www.postgresql.org/docs/7.4/static/installation.html* (for PostgreSQL) and *http://www.bioperl.org* and *http://bioperl.org/Core/Latest/INSTALL* (for BioPerl)

The following software packages should be preinstalled:
Perl v. 5.0 or higher (*http://www.perl.org*)
PostgreSQL v. 7.3.2 or higher (*http://www.postgresql.org*)
Java 1.4.1 or higher (*http://java.sun.com*)
CVS (*http://www.cvs.org*)

Nonstandard software:

These packages will need to be installed:
XML Parser (*http://search.cpan.org/~kmacleod/libxml-perl-0.08/*)
XML::DOM (*http://search.cpan.org/~tjmather/XML-DOM-1.43/*)
DBI (*http://search.cpan.org/~timb/DBI/*)
DBD-Pg (*http://search.cpan.org/dist/DBD-Pg/*)
XML-XORT (*http://www.gmod.org*)
BioPerl (*http://www.bioperl.org/Core/Latest/index.shtml*)

For the purposes of this demonstration, it is assumed that the PostgreSQL database has been installed and that a user account has been created and has been granted "create database" privileges. It is also assumed that the XORT/Gamebridge package will be saved into the `/home/work` directory.

Files

The sample data file `GAME.xml` used in this protocol is included as part of the XORT download

Create the Chado instance

1. Get the Chado schema DDL. The most recent Chado schema DDL is included with the XORT package distribution in directory `XML-XORT/examples/chado.ddl`. It may also be downloaded from the GMOD Sourceforge CVS using the following command:

```
$ cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/
  cvsroot/gmod \co schema/chado
```

2. In the psql client, create the Chado instance on the PostgreSQL database server using the Chado schema DDL:

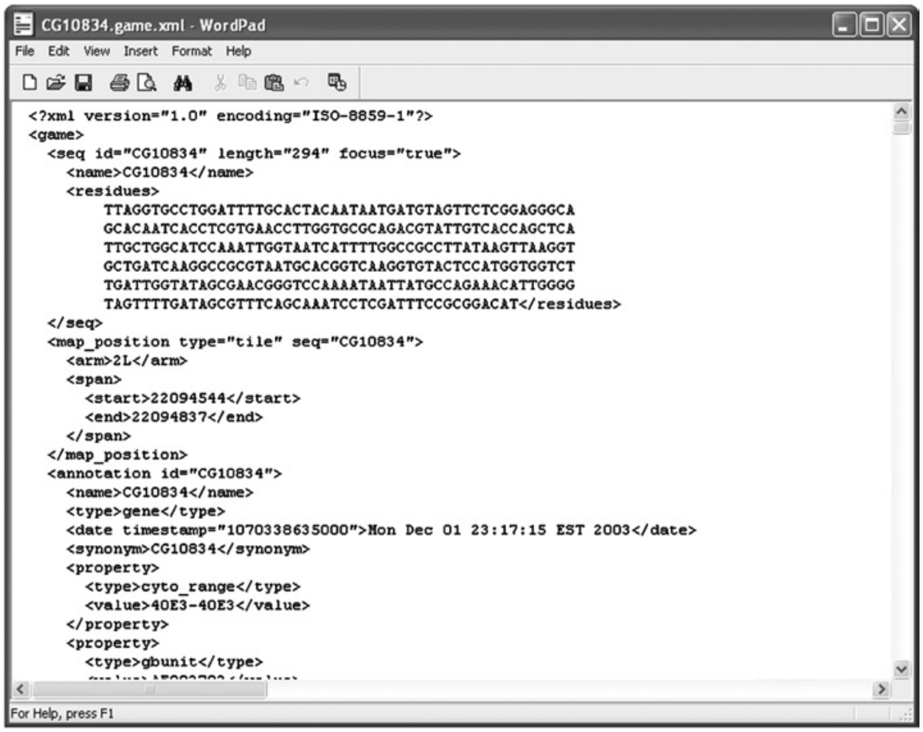
```
psql> create database chado;
psql> \c chado;
psql> \i /home/work/XML-XORT/examples/chado.ddl
psql> \q
```

Load a GAME-XML File

3. Download GAME files from FlyBase. GAME XML files can be either created using Apollo (*UNIT 9.5*) or can be downloaded from the FlyBase Web site (which currently hosts annotation for two *Drosophila* species GAME XML) using the following command:

```
ftp://flybase.net/genomes/Drosophila_melanogaster/
  current/xml-game/
```

An example of GAME XML file format is shown in Figure 9.6.2.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<game>
  <seq id="CG10834" length="294" focus="true">
    <name>CG10834</name>
    <residues>
        TTAGGTGCCTGGATTTTGCACTACAATAATGATGTAGTTCTCGGAGGGCA
        GCACAATCACCTCGTGAACCTTGGTGCGCAGACGTATTGTCACCAGCTCA
        TTGCTGGCATCCAAATTGGTAATCATTTTGGCCGCCTTATAAGTTAAGGT
        GCTGATCAAGGCCGCGTAATGCACGGTCAAGGTGTACTCCATGGTGGTCT
        TGATTGGTATAGCGAACGGGTCCAAAATAATTATGCCAGAAACATTGGGG
        TAGTTTTGATAGCGTTTCAGCAAATCCTCGATTTCCGCGGACAT</residues>
  </seq>
  <map_position type="tile" seq="CG10834">
    <arm>2L</arm>
    <span>
      <start>22094544</start>
      <end>22094837</end>
    </span>
  </map_position>
  <annotation id="CG10834">
    <name>CG10834</name>
    <type>gene</type>
    <date timestamp="1070338635000">Mon Dec 01 23:17:15 EST 2003</date>
    <synonym>CG10834</synonym>
    <property>
      <type>cyto_range</type>
      <value>40E3-40E3</value>
    </property>
    <property>
      <type>gbunit</type>
```

Figure 9.6.2 GAME XML format, which is one of the input formats for the annotation editor Apollo.

Convert GAME XML file into ChadoXML

The following steps use the sample file `GAME.xml`.

4. Convert the GAME XML file into ChadoXML using the GTC converter in XORT. Software for converting between GAME XML and ChadoXML formats is included as part of the XML-XORT package. The following command will convert the sample GAME XML (`GAME.xml`) into ChadoXML (`chado.xml`):

```
$ java -cp /home/work/XML-XORT/conf/
  GAMEChadoConv.jar GTC GAME.xml chado.xml -a
```

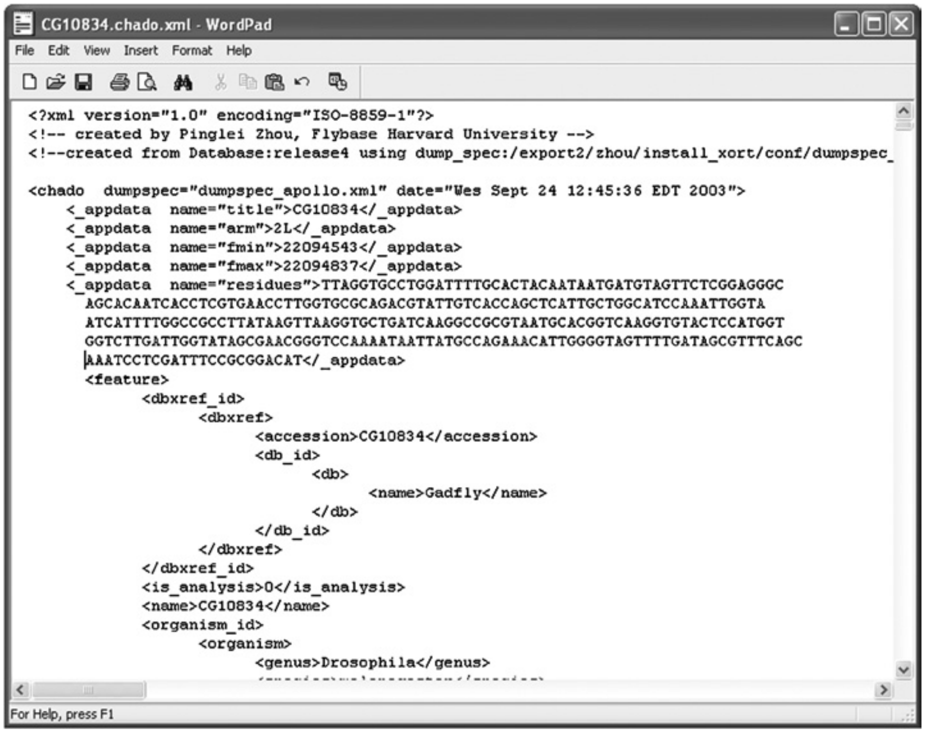
Examples of GAME XML and ChadoXML file formats are shown in Figures 9.6.2 and 9.6.3, respectively.

Possible switches for GTC are:

`-a`*: Convert all, both annotation and computational data.*
`-g`*: Convert annotation data only.*
`-c`*: Convert computational data only.*

Validate the ChadoXML

5. Validate the ChadoXML using the validator from the XORT package. The XORT validator serves several purposes. It may be used in stand-alone mode, which will verify the syntax of the ChadoXML file or it may be used in connection with the database, which will allow content testing to spot some potential problems before any update transactions are actually executed. The command syntax is:


CG10834.chado.xml - WordPad
File Edit View Insert Format Help
For Help, press F1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- created by Pinglei Zhou, Flybase Harvard University -->
<!--created from Database:release4 using dump_spec:/export2/zhou/install_xort/conf/dumpspec_

<chado  dumpspec="dumpspec_apollo.xml" date="Wes Sept 24 12:45:36 EDT 2003">
    <_appdata  name="title">CG10834</_appdata>
    <_appdata  name="arm">2L</_appdata>
    <_appdata  name="fmin">22094543</_appdata>
    <_appdata  name="fmax">22094837</_appdata>
    <_appdata  name="residues">TTAGGTGCCTGGATTTTGCACTACAATAATGATGTAGTTCTCGGAGGGC
     AGCACAATCACCTCGTGAACCTTGGTGCGCAGACGTATTGTCACCAGCTCATTGCTGGCATCCAAATTGGTA
     ATCATTTTGGCCGCCTTATAAGTTAAGGTGCTGATCAAGGCCGCGTAATGCACGGTCAAGGTGTACTCCATGGT
     GGTCTTGATTGGTATAGCGAACGGGTCCAAAATAATTATGCCAGAAACATTGGGGTAGTTTTGATAGCGTTTCAGC
     AAATCCTCGATTTCCGCGGACAT</_appdata>
     <feature>
           <dbxref_id>
                 <dbxref>
                       <accession>CG10834</accession>
                       <db_id>
                             <db>
                                   <name>Gadfly</name>
                             </db>
                       </db_id>
                 </dbxref>
           </dbxref_id>
           <is_analysis>0</is_analysis>
           <name>CG10834</name>
           <organism_id>
                 <organism>
                       <genus>Drosophila</genus>
```

Figure 9.6.3 Structure for ChadoXML, which serves as intermediate format between Chado database and other file formats.

```
$ perl /home/work/XML-XORT/bin/
  xort_validator.pl -d chado -f chado.xml -v 1
```

Options and parameters include:

-h*: help message.*
-d*: database alias.*
-f*: ChadoXML file to be validated.*
-b debug*:* 0 *(default) without debug message,* 1 *with debug message.*
-v 0 *or* -v 1*:* 0 *(default) without or* 1 *with database connection during validation process.*

In the example above, -v 1 *is specified, causing XORT to connect to the database for content testing, which helps to identify data problems in the input ChadoXML before the loader actually attempting to load it into the database. Possible data problems originate from: (1) inconsistency between different version of the database schema, which may happen if the database schema has changed but converter configuration has not been followed, or (2) inconsistency in data implementation from different data sources, which may happen when ChadoXML generated by different projects and from different sources are loaded into the same Chado instance. The types of problems* -v 1 *can detect include: (1) use of an object reference in ChadoXML before it has been defined, (2) attempting to delete an object which does not exist in the database, or (3) attempting to update an object without specifying enough constraints, which could lead to an erroneous update of multiple records.*

6. Load the file into the database using the loader from the XORT package. Assuming that the ChadoXML file validated successfully, the XORT loader can be used to update the database with the validated ChadoXML file. The command syntax is as follows:

```
$ perl /home/work/XML-XORT/bin/xort_loader.pl -d Chado
  -f chado.xml
```

The above command loads the ChadoXML into the database. At the end of loading, `xort_loader.pl` *indicates whether the loading was successful or not. If the loading was not successful, the loader attempts to indicate the nature of the problem it encountered.*

Options and parameters include:

`-h`*: help message.*
`-d`*: database alias.*
`-f`*: ChadoXML file to be loaded.*
`-b debug`*:* `0` *(default) without debug message,* `1` *with debug message.*

LOADING A GenBank FILE

BASIC PROTOCOL 3

Essentially all genome sequence and annotation data are submitted to GenBank by individual genome project teams and research groups. It is a common practice to acquire data by downloading relevant records from GenBank. A GenBank record can be fed into the BioPerl module Bio::SeqIO to build a Bio::Seq object. Peili Zhang at FlyBase Harvard has contributed a BioPerl module which converts all the sequence and annotation data contained in a Bio::Seq object into ChadoXML. The resulting ChadoXML can then be easily loaded into Chado using the XORT package. This protocol provides users an option to populate the Chado database with data from GenBank, a rich data source.

Necessary Resources

Hardware

Any recent computer system running Macintosh OS X, Solaris, Linux, or other Unix variant

Software

Standard software:
PostgreSQL and BioPerl: Instructions for the installation of PostgreSQL and BioPerl under a Unix/Linux environment are beyond the scope of this unit; more information on these installation procedures can be found at *http://www.postgresql.org/docs/7.4/static/installation.html* (for PostgreSQL) and *http://www.bioperl.org* and *http://bioperl.org/Core/Latest/INSTALL* (for BioPerl)
The following software packages should be preinstalled:
Perl v. 5.0 or higher (*http://www.perl.org*)
PostgreSQL v. 7.3.2 or higher (*http://www.postgresql.org*)
Java 1.4.1 or higher (*http://java.sun.com*)
Nonstandard software:
These packages will need to be installed:
XML Parser (*http://search.cpan.org/~kmacleod/libxml-perl-0.08/*)
XML::DOM (*http://search.cpan.org/~tjmather/XML-DOM-1.43/*)
DBI (*http://search.cpan.org/~timb/DBI/*)
DBD-Pg (*http://search.cpan.org/dist/DBD-Pg/*)
XML-XORT (*http://www.gmod.org*)
BioPerl (*http://www.bioperl.org/Core/Latest/index.shtml*)
For the purposes of this demonstration, it is assumed that the PostgreSQL database has been installed and that a user account has been created and has been granted "create database" privileges. It is also assumed that the XORT/Gamebridge package will be saved into the `/home/work` directory.

Download GenBank flat files

There are two ways to download the GenBank records:

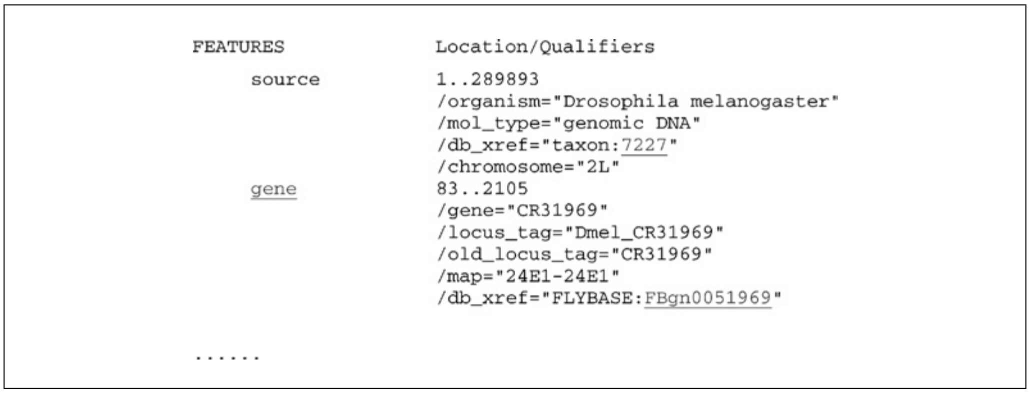
1a. Go to the NCBI Web site (*http://www.ncbi.nlm.nih.gov*), choose Nucleotide from the Search drop-down menu, then enter the accession number or a valid identifier to search for the sequence. In the Search Results page, click on the relevant record to view its content. On the top of the record details page, choose the Text option from the "Send to" drop-down menu to save the page as a text file.

1b. Write a simple perl script making using of the BioPerl package (see sample code below) to download from GenBank over the Internet:

```
# Perl script to download AE003576 and save into file
  AE003576.g in local directory
use Bio::DB::GenBank;
use Bio::SeqIO;
my $gb = new Bio::DB::GenBank;
my $seq = $gb->get_Seq_by_acc(‘AE003576’);
my $o = Bio::SeqIO->new(-file=>‘>AE003576.gb’,
  -format=>‘genbank’);
$o->write_seq($seq);
```

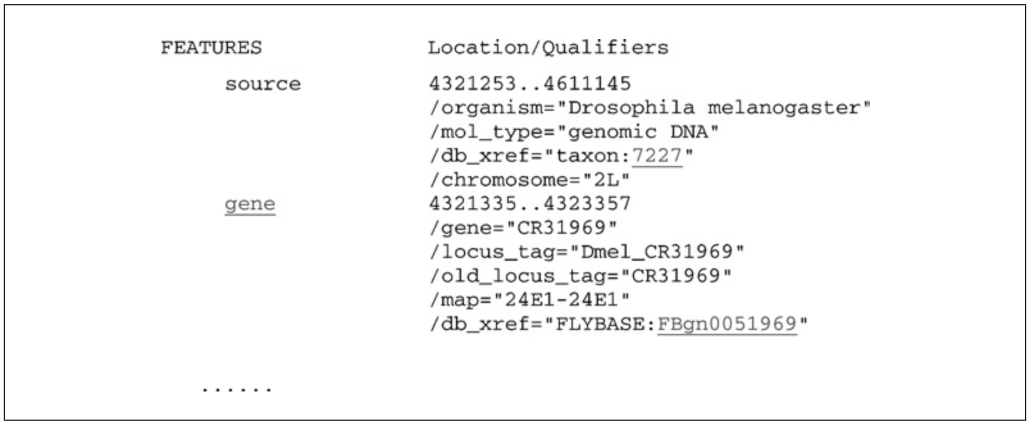
Convert GenBank flat files into ChadoXML and load into Chado

2. The GenBank data file may need some preprocessing on the feature coordinates before being converted into ChadoXML and loaded into the Chado database. In general, GenBank records have record-based coordinates for all features in the record. However, in a Chado database, it is far preferable to have a consistent coordinate system that is biologically meaningful and that allows better data management and data analysis. Thus, the feature locations are often represented in Chado by the features' genomic coordinates on chromosomes. The record-based coordinates in the GenBank record must be transformed into coordinates in the system adopted in Chado (e.g. Chromosomes) before loading. For example, the GenBank record for accession AE003576 is to be loaded into Chado. The FEATURES section of the record resembles Figure 9.6.4. AE004576 is one section of the *Drosophila melanogaster* chromosome 2L, spanning bases 2L:4321253..4611145. Because the Chado database that the record will be loaded into localizes all genomic features on the chromosomes, the GenBank record has to be modified to reflect the chromosomal coordinates, as illustrated in Figure 9.6.5. The chromosome name information (2L) will be passed into the BioPerl module that converts the GenBank record into ChadoXML, as described in step 3, below.



```
FEATURES             Location/Qualifiers
     source          1..289893
                     /organism="Drosophila melanogaster"
                     /mol_type="genomic DNA"
                     /db_xref="taxon:7227"
                     /chromosome="2L"
     gene            83..2105
                     /gene="CR31969"
                     /locus_tag="Dmel_CR31969"
                     /old_locus_tag="CR31969"
                     /map="24E1-24E1"
                     /db_xref="FLYBASE:FBgn0051969"

......
```

Figure 9.6.4 FEATURES section of GenBank record to be loaded into Chado.



```
        FEATURES             Location/Qualifiers
             source          4321253..4611145
                             /organism="Drosophila melanogaster"
                             /mol_type="genomic DNA"
                             /db_xref="taxon:7227"
                             /chromosome="2L"
             gene            4321335..4323357
                             /gene="CR31969"
                             /locus_tag="Dmel_CR31969"
                             /old_locus_tag="CR31969"
                             /map="24E1-24E1"
                             /db_xref="FLYBASE:FBgn0051969"


          ......
```

Figure 9.6.5 FEATURES section of GenBank record to be loaded into Chado, modified to reflect chromosomal coordinates.

3. After the necessary coordinate translation and/or other data preprocessing are completed, the GenBank flat data file can be converted into ChadoXML through a simple Perl script, making use of the aforementioned BioPerl module. An example of the code to use would be:

```
#Perl script to convert a GenBank formatted flat file
  to Chadoxml using BioPerl
use Bio::SeqIO;
#the GenBank records need to be pre-processed to
  convert the record-based coordinates
#to coordinates on the chosen coordinate system (e.g.,
  chromosome arm)
my $seqin = Bio::SeqIO->new(-format=>‘genbank’,
  -file=>‘AE003576_processed.gb’);
my $seq = $seqin->next_seq();
#AE003576 is a scaffold on Drosophila melanogaster
  chromosome arm 2L
my $arm = ‘2L’;
my $seqout = Bio::SeqIO->new(-format=>‘Chadoxml’,
  -file=>‘>AE003576.Chadoxml’);
#if the optional input parameters ‘src_feature’ and
  ‘src_feat_type’ are absent,
#the biological entity corresponding to the $seq object
  and its type are used as the
#src_feature and src_feat_type, respectively.
$seqout->write_seq(-seq=>$seq, -src_feature=>$arm,
  -src_feat_type=> ‘chromosome_arm’);
```

4. Load the ChadoXML into Chado as described in Basic Protocol 2.

QUERYING A CHADO ANNOTATION DATABASE USING SQL

BASIC PROTOCOL 4

This protocol presents some basic SQL queries for exploring annotation data in Chado. It is expected that once these queries, and the relational structures implicit in them, have been thoroughly understood, a user will be able to navigate through the tables storing annotation data in Chado and improvise queries as the need arises.

```
select g.name as gn_name, g_c.fmin as gn_fmin, g_c.fmax as gn_fmax,
     g_c.strand as gn_strand, ch.name as gn_arm
from feature g, featureloc g_c, feature ch
where g.feature_id = g_c.feature_id   and g_c.srcfeature_id = ch.feature_id
     and g.name = 'oaf'
;
```

Figure 9.6.6 Example query to retrieve location information for the gene *oaf*.

Necessary Resources

Hardware

Any recent computer system running Macintosh OS X, Solaris, Linux, or other Unix variant

Software

Standard software:

PostgreSQL and BioPerl: Instructions for the installation of PostgreSQL and BioPerl under a Unix/Linux environment are beyond the scope of this unit; more information on these installation procedures can be found at *http://www.postgresql.org/docs/7.4/static/installation.html* (for PostgreSQL) and *http://www.bioperl.org* and *http://bioperl.org/Core/Latest/INSTALL* (for BioPerl)

The following software packages should be preinstalled:
Perl v. 5.0 or higher (*http://www.perl.org*)
PostgreSQL v. 7.3.2 or higher (*http://www.postgresql.org*)
Java 1.4.1 or higher (*http://java.sun.com*)

Nonstandard software:

These packages will need to be installed:
XML Parser (*http://search.cpan.org/~kmacleod/libxml-perl-0.08/*)
XML::DOM (*http://search.cpan.org/~tjmather/XML-DOM-1.43/*)
DBI (*http://search.cpan.org/~timb/DBI/*)
DBD-Pg (*http://search.cpan.org/dist/DBD-Pg/*)
XML-XORT (*http://www.gmod.org*)
BioPerl (*http://www.bioperl.org/Core/Latest/index.shtml*)

For the purposes of this demonstration, it is assumed that the PostgreSQL database has been installed and that a user account has been created and has been granted "create database" privileges. It is also assumed that the XORT/Gamebridge package will be saved into the `/home/work` directory.

Using SQL to retrieve details of a single gene model

1. Design an SQL query (see Fig. 9.6.6 for example) to retrieve the location information for a given gene (e.g., *oaf*). This query joins three tables:

   a. `feature` (`g`, for the gene, *oaf*).

   b. `feature` (`ch`, for the chromosome arm).

   c. `featureloc` (`g_c`, localizing *oaf* on the chromosome arm).

2. The result of the query described in Figure 9.6.6 is shown in Figure 9.6.7.

```
--   gn_name | gn_fmin | gn_fmax | gn_strand | gn_arm
--   oaf     | 2492954 | 2498846 |         1 | 2L
-- (1 row)
```

Figure 9.6.7 Results returned for the query depicted in Figure 9.6.6.

```
select g.name as gn_name, tcv.name as feature_type,
     tx.name as tx_name, txc.fmin as tx_fmin, txc.fmax as tx_fmax,
     txc.strand as tx_strand, ch.name as tx_arm
from feature g, feature tx, feature_relationship g_tx, featureloc txc,
     feature ch, cvterm tcv
where      g.feature_id = g_tx.object_id
     and g_tx.subject_id = tx.feature_id
     and tx.feature_id = txc.feature_id
     and txc.srcfeature_id = ch.feature_id
     and tx.type_id = tcv.cvterm_id
     and tcv.name = 'mRNA'
     and g.name = 'oaf'
;
```

Figure 9.6.8 Example query to get transcripts and their locations for the gene *oaf*.

3. For a given gene (eg, *oaf*), design a query to get transcripts and their locations (see Fig. 9.6.8 for example). This query joins six tables:

   a. `feature` (`g` for the gene *oaf*).
   b. `feature` (`tx` for the transcripts linked to the gene).
   c. `feature_relationship` (`g_tx`, links gene and transcripts).
   d. `featureloc` (`txc,` localizing transcripts on the chromosome arm).
   e. `feature` (`ch` for the chromosome arm).
   f. `cvterm` (`tcv,` for specifying the type of record linked to the gene).

4. The result of the query described in Figure 9.6.8 is shown in Figure 9.6.9.

5. For a given transcript (e.g., “oaf-RB”), design a query to get exons and their locations (see Fig. 9.6.10 for example). This query joins six tables:

   a. `feature` (`tx`, for the transcript “oaf-RB”).
   b. `feature` (`ex`, for the exons linked to the transcript).
   c. `feature_relationship` (`tx_ex`, links transcript and exons).
   d. `featureloc` (`exc`, localizing exons on the chromosome arm).
   e. `feature` (`ch`, for the chromosome arm).
   f. `cvterm` (`ecv`, for specifying the type of record linked to transcript).

   The same query can be used to get the protein product and location, simply by switching `protein` *for* `exon` *in the text of Figure 9.6.10.*

```
--   gn_name  |  feature_type  |  tx_name  |  tx_fmin  |  tx_fmax  | tx_strand | tx_arm
--  oaf       | mRNA           | oaf-RD    | 2492954   | 2498846   |         1 | 2L
--  oaf       | mRNA           | oaf-RB    | 2492954   | 2498234   |         1 | 2L
--  oaf       | mRNA           | oaf-RA    | 2492954   | 2498846   |         1 | 2L
--  oaf       | mRNA           | oaf-RC    | 2492954   | 2497357   |         1 | 2L
-- (4 rows)
```

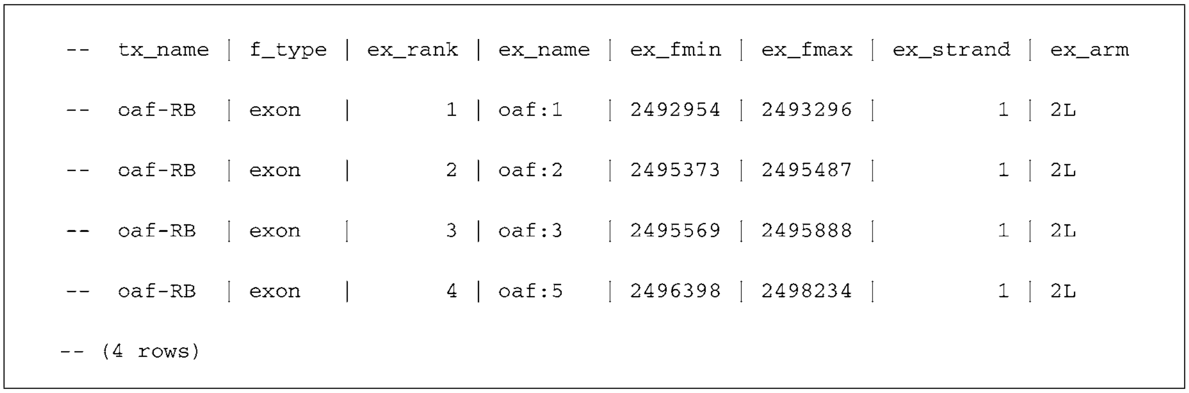
Figure 9.6.9 Results returned for the query depicted in Figure 9.6.8.

```
select tx.name as tx_name, ecv.name as f_type, tx_ex.rank as ex_rank,
      ex.name as ex_name, exc.fmin as ex_fmin, exc.fmax as ex_fmax,
      exc.strand as ex_strand, ch2.name as ex_arm
from feature tx, feature ex, feature_relationship tx_ex, featureloc exc,
      feature ch2, cvterm ecv
where      tx.feature_id = tx_ex.object_id
      and tx_ex.subject_id = ex.feature_id
      and ex.feature_id = exc.feature_id
      and exc.srcfeature_id = ch2.feature_id
      and ex.type_id = ecv.cvterm_id
      and ecv.name = 'exon'
      and tx.name = 'oaf-RB'
order by ex_rank
;
```

Figure 9.6.10 Example query to get exons and their locations for a given transcript, “oaf-RB.”.

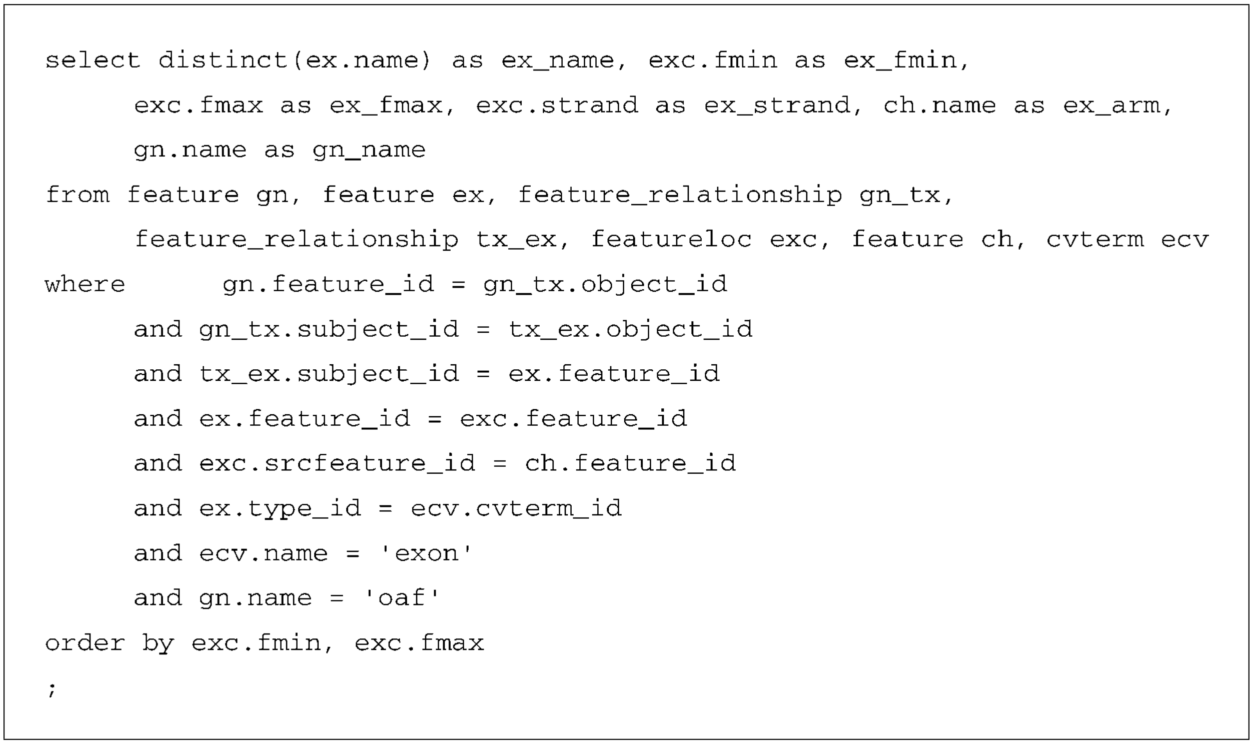
6. The result of the query described in Figure 9.6.10 is shown in Figure 9.6.11.
7. For a given gene (eg, *oaf*), design a query to list exons and their locations (see Fig. 9.6.12 for example). This query joins seven tables:
   a. `feature` (gn, for the gene, *oaf*).
   b. `feature` (ex, for the exons linked to the gene via the transcripts).
   c. `feature_relationship` (`gn_tx`, linking the gene and its transcripts).
   d. `feature_relationship` (`tx_ex`, linking the transcripts and their exons).
   e. `featureloc` (`exc`, localizing the exons on the chromosome arm).
   f. `feature` (`ch`, for the chromosome arm).
   g. `cvterm` (`ecv`, for specifying the type of record linked to transcripts).

   Note that the exons for a given gene model may be used in multiple transcripts, and, therefore, in order to return each exon only once (though they are related to the gene in Chado via the transcripts), it is necessary to formulate the query as “select distinct” (see Fig. 9.6.12).
8. The result of the query described in Figure 9.6.12 is shown in Figure 9.6.13.



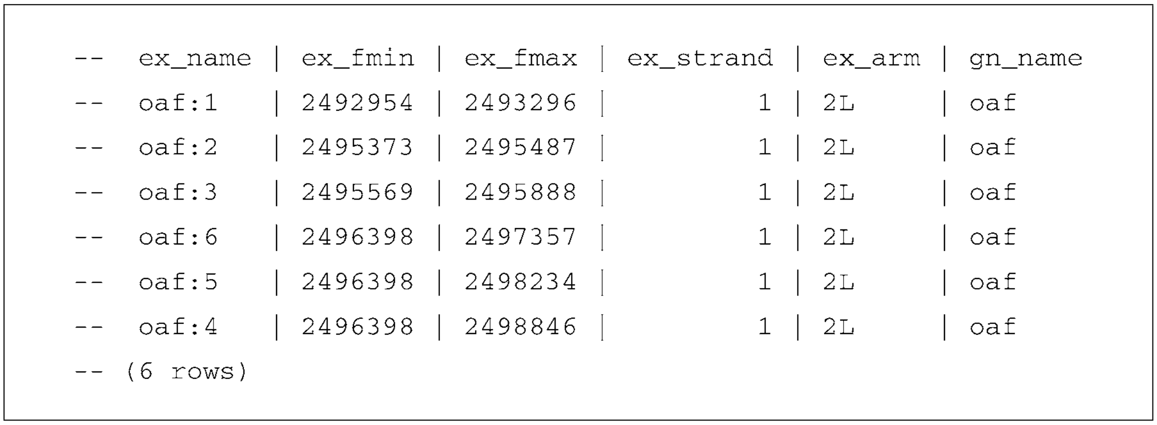
```
--  tx_name | f_type | ex_rank | ex_name | ex_fmin | ex_fmax | ex_strand | ex_arm

--  oaf-RB  | exon   |       1 | oaf:1   | 2492954 | 2493296 |         1 | 2L

--  oaf-RB  | exon   |       2 | oaf:2   | 2495373 | 2495487 |         1 | 2L

--  oaf-RB  | exon   |       3 | oaf:3   | 2495569 | 2495888 |         1 | 2L

--  oaf-RB  | exon   |       4 | oaf:5   | 2496398 | 2498234 |         1 | 2L

-- (4 rows)
```

Figure 9.6.11 Results returned for the query depicted in Figure 9.6.10.



```
select distinct(ex.name) as ex_name, exc.fmin as ex_fmin,
     exc.fmax as ex_fmax, exc.strand as ex_strand, ch.name as ex_arm,
     gn.name as gn_name
from feature gn, feature ex, feature_relationship gn_tx,
     feature_relationship tx_ex, featureloc exc, feature ch, cvterm ecv
where      gn.feature_id = gn_tx.object_id
     and gn_tx.subject_id = tx_ex.object_id
     and tx_ex.subject_id = ex.feature_id
     and ex.feature_id = exc.feature_id
     and exc.srcfeature_id = ch.feature_id
     and ex.type_id = ecv.cvterm_id
     and ecv.name = 'exon'
     and gn.name = 'oaf'
order by exc.fmin, exc.fmax
;
```

Figure 9.6.12 Example query to get exons and their locations for the gene *oaf*.



```
--  ex_name | ex_fmin | ex_fmax | ex_strand | ex_arm | gn_name
--  oaf:1   | 2492954 | 2493296 |         1 | 2L     | oaf
--  oaf:2   | 2495373 | 2495487 |         1 | 2L     | oaf
--  oaf:3   | 2495569 | 2495888 |         1 | 2L     | oaf
--  oaf:6   | 2496398 | 2497357 |         1 | 2L     | oaf
--  oaf:5   | 2496398 | 2498234 |         1 | 2L     | oaf
--  oaf:4   | 2496398 | 2498846 |         1 | 2L     | oaf
-- (6 rows)
```

Figure 9.6.13 Results returned for the query depicted in Figure 9.6.12.

```
select distinct(a.program), a.sourcename
from feature hsp, feature ch, featureloc hsp_ch, featureloc hsp_al,
      analysisfeature af, analysis a
where hsp.is_analysis = 't'
      and hsp.feature_id = af.feature_id
      and af.analysis_id = a.analysis_id
      and hsp.feature_id = hsp_al.feature_id
      and hsp_al.feature_id = hsp_ch.feature_id
      and hsp_ch.srcfeature_id = ch.feature_id
      and ch.name = '2L'
      and hsp_ch.fmax < 50000
      and hsp_ch.fmin > 0
;
```

Figure 9.6.14 Example query to list types of analysis available and sets of data used in the analysis for a given genomic region (arm 2L, bases 1 to 49,999).

SQL to retrieve supporting evidence tiers for specific gene model

9. For a given (genomic) region (e.g., arm 2L, bases 1 to 49,999), design a query to list types of analysis available and sets of data used in the analysis (see Fig. 9.6.14 for example). This query uses six tables:

    a. `feature` (`hsp,` for HSP hits).

    b. `feature` (`ch`, for the chromosome arm; e.g., `2L`).

    c. `featureloc` (`hsp_ch`, linking HSPs to the chromosome arm).

    d. `featureloc` (`hsp_al,` linking HSPs to the aligned object).

    e. `analysisfeature` (`af,` linking HSP to analysis details).

    f. `analysis` (`a`, for analysis details).

10. The result of the query described in Figure 9.6.14 is shown in Figure 9.6.15.

11. For a given (genomic) region (eg, arm 2L, bases 1 to 49,999), design a query to list aligned objects (see Figure 9.6.16 for example). This query uses four tables:

    a. `feature` (`al`, for the aligned objects).

    b. `feature` (`ch`, for the chromosome arm).

    c. `featureloc` (`hsp_ch`, linking HSPs to the chromosome arm).

    d. `featureloc` (`hsp_al`, linking HSPs to the aligned object).

    Note that, since the aligned object is related to the chromosome via the HSP(s) that constitute the alignment, in order to avoid reporting a given aligned object multiple times (if there are multiple HSPs in its alignment to the arm), it is necessary to use the `distinct` *constraint.*

12. The result of the query described in Figure 9.6.16 is shown in Figure 9.6.17.

```
--            program             |         sourcename
--  blastx_masked                 | aa_SPTR.dmel
--  blastx_masked                 | aa_SPTR.insect
--  blastx_masked                 | aa_SPTR.othinv
--  blastx_masked                 | aa_SPTR.othvert
--  blastx_masked                 | aa_SPTR.plant
--  blastx_masked                 | aa_SPTR.primate
--  blastx_masked                 | aa_SPTR.rodent
--  blastx_masked                 | aa_SPTR.worm
--  blastx_masked                 | aa_SPTR.yeast
--  dmel_r3_to_dmel_r4_migration  | dmel_r3_affy_oligos
--  genscan_masked                | dummy
--  promoter                      | dummy
--  repeatmasker                  | dummy
--  sim4                          | na_DGC.in_process.dros
--  sim4                          | na_dbEST.diff.dmel
--  sim4                          | na_dbEST.same.dmel
--  sim4                          | na_gadfly.dros.RELEASE2
--  sim4                          | na_gb.dmel
--  sim4                          | na_transcript.dmel.RELEASE31
--  sim4                          | na_transcript.dmel.RELEASE32
--  tblastx_masked                | na_dbEST.insect
--  tblastxwrap_masked            | na_baylorf1_scfchunk.dpse
--  tblastxwrap_masked            | na_scf_chunk_agambiae.fa
-- (23 rows)
```

Figure 9.6.15 Results returned for the query depicted in Figure 9.6.14.

```
select distinct(al.name)
from feature al, feature ch, featureloc hsp_ch, featureloc hsp_al
where al.is_analysis = 't'
      and al.feature_id = hsp_al.srcfeature_id
      and hsp_al.feature_id = hsp_ch.feature_id
      and hsp_ch.srcfeature_id = ch.feature_id
      and ch.name = '2L'
      and hsp_ch.fmin > 0
      and hsp_ch.fmax < 50000
;
```

Figure 9.6.16 Example query to list aligned objects for a given genomic region (arm 2L, bases 1 to 49,999).

```
-- name
-- AT04953.5prime
-- AY051654
-- BU038928
-- CG11023-RA.31
-- CG11023-RA.32
-- ...
-- (178 rows)
```

Figure 9.6.17 Results returned for the query depicted in Figure 9.6.16.

```
    select al.name as source_object, hsp.uniquename as hsp_name, hsp_al.fmin as
    hsp_source_fmin, hsp_al.fmax as hsp_source_fmax, ch.name as target_object,
    hsp_ch.fmin as hsp_target_fmin, hsp_ch.fmax as hsp_target_fmax, hsp_ch.strand as
    hsp_target_strand
    from feature al, featureloc hsp_al, feature hsp, featureloc hsp_ch, feature ch
    where hsp_al.srcfeature_id = al.feature_id
          and hsp_al.feature_id = hsp.feature_id
          and hsp_al.rank = 1
          and hsp.feature_id = hsp_ch.feature_id
          and hsp_ch.rank = 0
          and hsp_ch.srcfeature_id = ch.feature_id
          and al.uniquename = 'AY129461'
    order by hsp_source_fmin, hsp_source_fmax
;
```

Figure 9.6.18 Example query to retrieve the alignment details for the alignment of a given sequence against the chromosome arm (e.g., GenBank record “AY129461”).

13. Query to retrieve the alignment details for the alignment of a given sequence against the chromosome arm (e.g., GenBank record “AY129461”; see Fig. 9.6.18). This query uses five tables:

    a. `feature` (`al`, for the aligned objects).
    b. `feature` (`hsp`, for the HSPs).
    c. `featureloc` (`hsp_ch`, localizing HSPs to the chromosome arm).
    d. `featureloc` (`hsp_al`, localizing HSPs to the aligned object).
    e. `feature` (`ch`, for the chromosome arm).

14. The result of the query described in Figure 9.6.18 is shown in Figure 9.6.19.

```
--  source_object | hsp_name | hsp_source_fmin | hsp_source_fmax | target_object | hsp_target_fmin
| hsp_target_fmax | hsp_target_strand
--  AY129461      | :5031336 |               0 |              54 | 2L            |           59189
|           59242 |                -1
--  AY129461      | :5031337 |              54 |             156 | 2L            |           58080
|           58182 |                -1
--  AY129461      | :5031338 |             156 |             713 | 2L            |           39300
|           39857 |                -1
--  AY129461      | :5031339 |             713 |             910 | 2L            |           38534
|           38731 |                -1
--  AY129461      | :5031340 |             910 |            1403 | 2L            |           34719
|           35212 |                -1
--  AY129461      | :5031341 |            1403 |            1450 | 2L            |           34557
|           34604 |                -1
--  AY129461      | :5031342 |            1450 |            1894 | 2L            |           33844
|           34288 |                -1
--  AY129461      | :5031343 |            1894 |            1981 | 2L            |           28981
|           29068 |                -1
--  AY129461      | :5031344 |            1981 |            2175 | 2L            |           28732
|           28926 |                -1
--  AY129461      | :5031345 |            2175 |            2401 | 2L            |           28014
|           28240 |                -1
--  AY129461      | :5031346 |            2401 |            2839 | 2L            |           27052
|           27490 |                -1
--  AY129461      | :5031347 |            2839 |            3038 | 2L            |           26765
|           26964 |                -1
--  AY129461      | :5031348 |            3038 |            4327 | 2L            |           25399
|           26688 |                -1
--  AY129461      | :5031349 |            4327 |            4341 | 2L            |           13464
|           13479 |                -1
-- (14 rows)
```

Figure 9.6.19 Results returned for the query depicted in Figure 9.6.18.

GENERATING STANDARD REPORTS FROM A CHADO ANNOTATION DATABASE

BASIC PROTOCOL 5

This protocol explains first how to dump data in ChadoXML format from a Chado annotation database using XORT, and then how to convert ChadoXML-formatted data files into the commonly used sequence annotation data formats GAME XML, GFF3, and FASTA,

Necessary Resources

Hardware

Any recent computer system running Macintosh OS X, Solaris, Linux, or other Unix variant

Software

Standard software:

PostgreSQL and BioPerl: Instructions for the installation of PostgreSQL and BioPerl under a Unix/Linux environment are beyond the scope of this unit; more information on these installation procedures can be found at *http://www.postgresql.org/docs/7.4/static/installation.html* (for PostgreSQL) and *http://www.bioperl.org* and *http://bioperl.org/Core/Latest/INSTALL* (for BioPerl)

The following software packages should be preinstalled:

Perl v. 5.0 or higher (*http://www.perl.org*)

PostgreSQL v. 7.3.2 or higher (*http://www.postgresql.org*)

Java 1.4.1 or higher (*http://java.sun.com*)

Nonstandard software:
These packages will need to be installed:
XML Parser (*http://search.cpan.org/~kmacleod/libxml-perl-0.08/*)
XML::DOM (*http://search.cpan.org/~tjmather/XML-DOM-1.43/*)
DBI (*http://search.cpan.org/~timb/DBI/*)
DBD-Pg (*http://search.cpan.org/dist/DBD-Pg/*)
GMODTools (*http://www.gmod.org*)
XML-XORT (*http://www.gmod.org*)
BioPerl (*http://www.bioperl.org/Core/Latest/index.shtml*)

For the purposes of this demonstration, it is assumed that the PostgreSQL database has been installed and that a user account has been created and has been granted "create database" privileges. It is also assumed that the XORT/Gamebridge package will be saved into `/home/work directory`.

Files

The sample data files `dumpspec_gene_model.xml` and `chado_dump.xml` used in this protocol are included as part of the XORT download

1. Export data in ChadoXML format using the following command:

```
$ perl /home/work/XML-XORT/conf/xort_dump.pl -d Chado
  -g /home/work/XML-XORT/conf/
  dumpspec_gene_model.xml -f output_file
```

Users can configure the XORT dumper to generate specific data sets into different ChadoXML formats using a project-specific "dump spec." The sample file (`dumpspec_gene_model.xml`*) is a typical dump spec to export Central Dogma structure into three-level XML file which, when converted to GAME XML, can be read by Apollo (see sample file:* `chado_dump.xml`*).*

2. Convert ChadoXML to GAME XML. Software for converting between GAME XML and ChadoXML formats is included as part of the XML-XORT package. The following command will convert the sample GAME XML (`GAME.xml`) into ChadoXML (`chado.xml`):

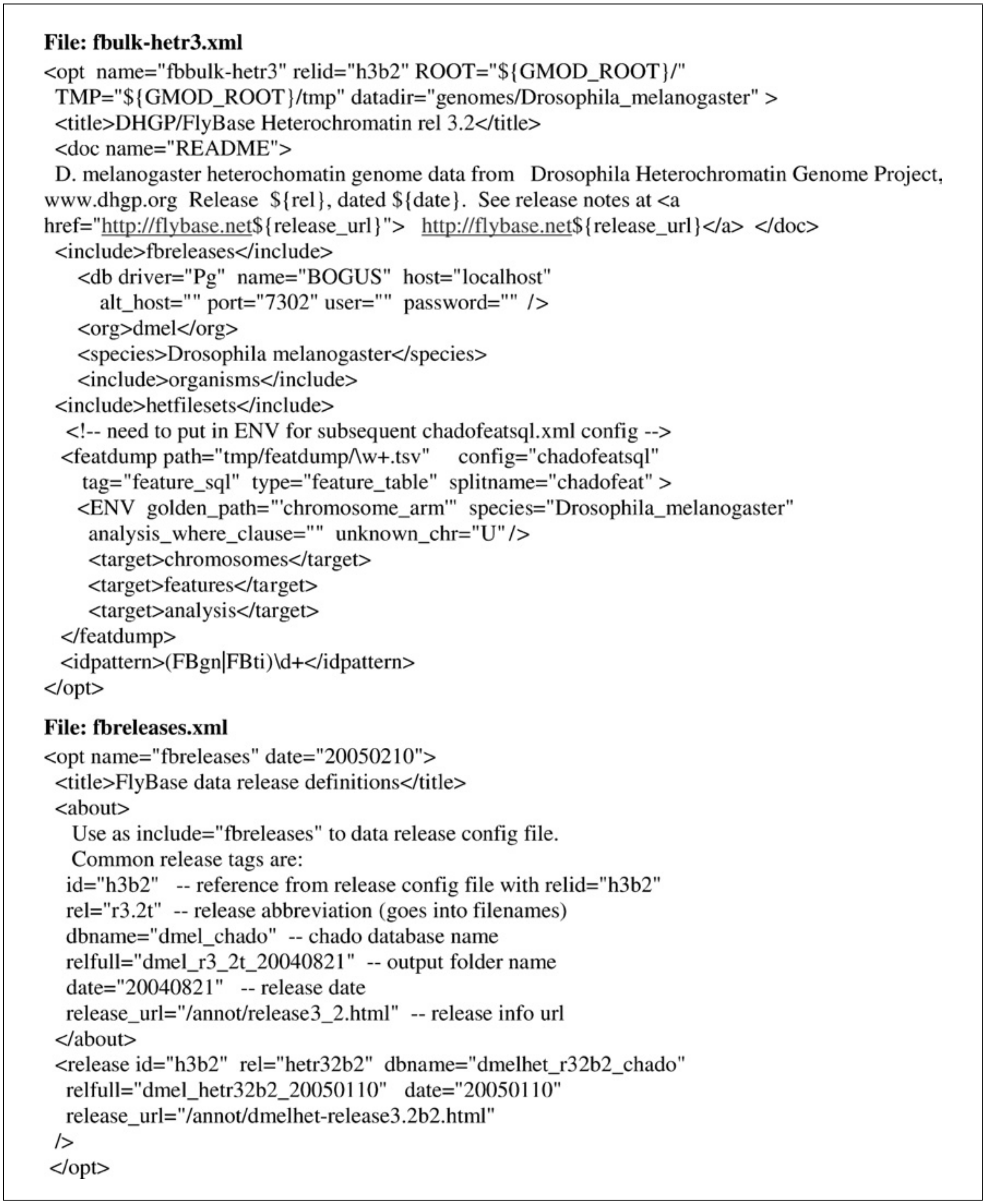
```
$ java -cp /home/work/XML-XORT/conf/
  GAMEChadoConv.jar CTG chado.xml GAME.xml -a
```

Possible switches for GTC are:

`-a`*: Convert all, both annotation and computational data.*
`-g`*: Convert annotation data only.*
`-c`*: Convert computational data only.*

3. Produce GenBank flat-file format report. For the moment, the preferred way to produce GenBank flat-file format records from GAME XML is to load the GAME XML file into Apollo, and then save again in GenBank format. See *UNIT 9.5* for details on how to do this.

4. Produce bulk FASTA or GFF3 dumps. Assume that the GMODTools package has been downloaded from the GMOD Web site (see Necessary Resources) and saved in the `/home/work` directory. The following three configuration files need to be modified to reflect the database connection: `conf/gmod.conf`, `conf/bulkfiles/fbreleases.xml`, and the main configuration file `conf/bulkfiles/fbbulk-hetr3.xml`. Sample modified configuration files are shown in Figure 9.6.20.



```
File: fbulk-hetr3.xml
<opt  name="fbbulk-hetr3" relid="h3b2" ROOT="${GMOD_ROOT}/"
  TMP="${GMOD_ROOT}/tmp" datadir="genomes/Drosophila_melanogaster" >
  <title>DHGP/FlyBase Heterochromatin rel 3.2</title>
  <doc name="README">
  D. melanogaster heterochomatin genome data from   Drosophila Heterochromatin Genome Project,
www.dhgp.org  Release  ${rel}, dated ${date}.  See release notes at <a
href="http://flybase.net${release_url}">  http://flybase.net${release_url}</a>  </doc>
  <include>fbreleases</include>
      <db driver="Pg"  name="BOGUS"  host="localhost"
         alt_host="" port="7302" user=""  password="" />
      <org>dmel</org>
      <species>Drosophila melanogaster</species>
      <include>organisms</include>
  <include>hetfilesets</include>
    <!-- need to put in ENV for subsequent chadofeatsql.xml config -->
   <featdump path="tmp/featdump/\w+.tsv"     config="chadofeatsql"
      tag="feature_sql"  type="feature_table"  splitname="chadofeat" >
     <ENV  golden_path="'chromosome_arm'"  species="Drosophila_melanogaster"
       analysis_where_clause=""  unknown_chr="U" />
       <target>chromosomes</target>
       <target>features</target>
       <target>analysis</target>
   </featdump>
   <idpattern>(FBgn|FBti)\d+</idpattern>
</opt>

File: fbreleases.xml
<opt name="fbreleases" date="20050210">
  <title>FlyBase data release definitions</title>
  <about>
     Use as include="fbreleases" to data release config file.
     Common release tags are:
    id="h3b2"   -- reference from release config file with relid="h3b2"
    rel="r3.2t"  -- release abbreviation (goes into filenames)
    dbname="dmel_chado"  -- chado database name
    relfull="dmel_r3_2t_20040821"  -- output folder name
    date="20040821"   -- release date
    release_url="/annot/release3_2.html"  -- release info url
  </about>
  <release id="h3b2"  rel="hetr32b2"  dbname="dmelhet_r32b2_chado"
    relfull="dmel_hetr32b2_20050110"   date="20050110"
    release_url="/annot/dmelhet-release3.2b2.html"
  />
 </opt>
```

Figure 9.6.20 Examples of `conf/bulkfiles/fbreleases.xml` and `conf/bulkfiles/fbbulk-hetr3.xml` files modified to reflect the database.

The file name is not important; any file name with proper information can be used.

Don Gilbert at FlyBase Indiana developed the necessary software to generate FASTA/GFF3 report files.

5. After editing the configuration files to reflect the project specific information, it is possible to generate report files using a command similar to the one in Figure 9.6.21. This prepares the Chado database for reporting, assuming configuration file `fbbulk-hetr3.xml` has the correct data information:

```
:
     % cd /tmp/GMODTools/bin
     %       perl       -IGMODTools/lib       /bulkfiles.pl       -
conf../conf/bulkfiles/fbbulk-hetr3.xml -
dnadump -featdump
```

Figure 9.6.21 Example of commands used for generating report files.

```
% perl -IGMODTools/lib bulkfiles.pl -conf
  ../conf/bulkfiles/fbbulk-hetr3.xml -make
```

makes all FASTA and GFF3 files for all chromosomes.

```
% perl -IGMODTools/lib bulkfiles.pl -conf
  ../conf/bulkfiles/fbbulk-hetr3.xml -chr X -format
  fasta -make
```

makes subset of FASTA files for chromosome X, (subset of GFF3 for chromosome X if replacing FASTA with GFF).

SUPPORT PROTOCOL

INSTALLING SOFTWARE FOR A UNIX-LIKE ENVIRONMENT ON A PC

For many reasons, much open-source bioinformatics software runs on Linux, Mac OS X, Solaris, or one of the many other Unix variants. Nevertheless, many biologists favor the Windows operating system, partially due to the user-friendly GUI software that can be used under this system. Cygwin is a Linux-like environment for Windows that attempts to solve the conflict between Unix and Windows operating systems. With Cygwin, one can run most programs with Linux-like command. This Support Protocol guides the user through the steps needed to create, populate, and use the Chado database in a Windows environment running Cygwin.

Necessary Resources

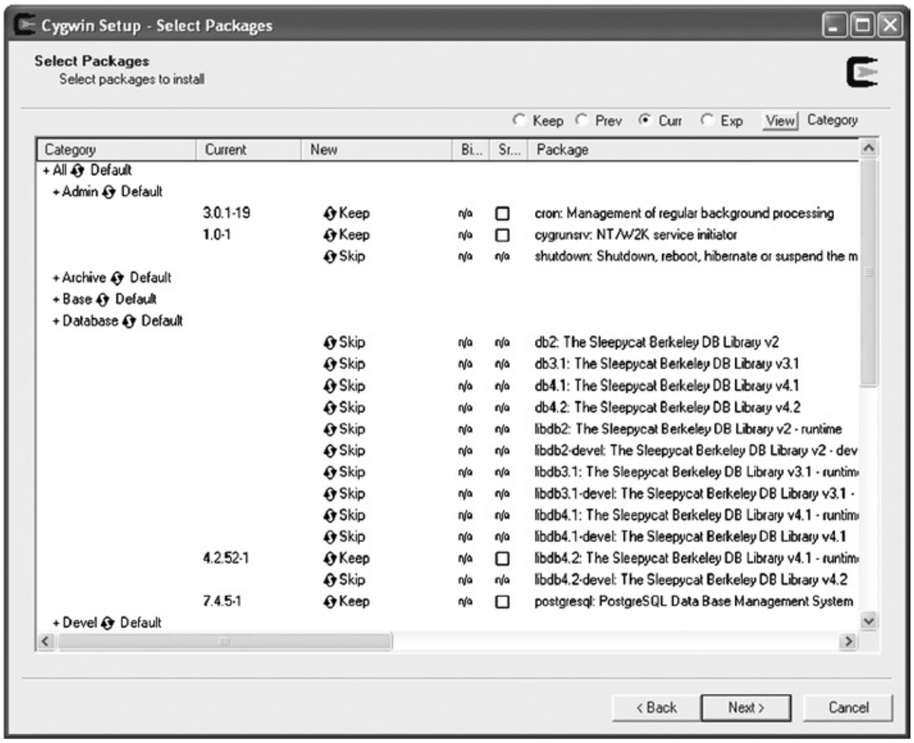
Hardware

PC with ≥256 Mb RAM, ≥1 GHz processor, and ≥10 Gb hard disk

Software

Cygwin (*http://www.cygwin.com*)
Perl v. 5.0 and higher (*http://www.perl.org*)
PostgreSQL v. 7.3.2 and higher (comes with Cygwin installation)
Java 1.4.1 and higher (*http://java.sun.com*)
XML::DOM (*http://search.cpan.org/~tjmather/XML-DOM-1.43/*)
XML::Parser (*http://sea rch.cpan.org/~kmacleod/libxml-perl-0.08/*)
DBI (*http://search.cpan.org/~timb/DBI/*)
DBD-Pg (*http://search.cpan.org/dist/DBD-Pg/*)
XML-XORT (*http://www.gmod.org*)

1. Download and install the Windows version of Java 1.4.2. From *http://java.sun.com/j2se/1.4.2/download.html*, click "Download Windows J2SE SDK." When the download is completed, click on the downloaded executable file to start the installation. After installation, add the Java directory to CLASSPATH. See Troubleshooting section ("How to set environment variable in Windows environment") for more details.



Cygwin Setup - Select Packages
Select Packages
Select packages to install
○ Keep ○ Prev ● Curr ○ Exp View Category
< Back
Next >
Cancel

Category	Current	New	Bi...	Sr...	Package
+ All Default					
+ Admin Default					
	3.0.1-19	Keep	n/a	□	cron: Management of regular background processing
	1.0-1	Keep	n/a	□	cygrunsrv: NT/W2K service initiator
		Skip	n/a	n/a	shutdown: Shutdown, reboot, hibernate or suspend the m
+ Archive Default					
+ Base Default					
+ Database Default					
		Skip	n/a	n/a	db2: The Sleepycat Berkeley DB Library v2
		Skip	n/a	n/a	db3.1: The Sleepycat Berkeley DB Library v3.1
		Skip	n/a	n/a	db4.1: The Sleepycat Berkeley DB Library v4.1
		Skip	n/a	n/a	db4.2: The Sleepycat Berkeley DB Library v4.2
		Skip	n/a	n/a	libdb2: The Sleepycat Berkeley DB Library v2 - runtime
		Skip	n/a	n/a	libdb2-devel: The Sleepycat Berkeley DB Library v2 - dev
		Skip	n/a	n/a	libdb3.1: The Sleepycat Berkeley DB Library v3.1 - runtim
		Skip	n/a	n/a	libdb3.1-devel: The Sleepycat Berkeley DB Library v3.1 -
		Skip	n/a	n/a	libdb4.1: The Sleepycat Berkeley DB Library v4.1 - runtim
		Skip	n/a	n/a	libdb4.1-devel: The Sleepycat Berkeley DB Library v4.1
	4.2.52-1	Keep	n/a	□	libdb4.2: The Sleepycat Berkeley DB Library v4.1 - runtim
		Skip	n/a	n/a	libdb4.2-devel: The Sleepycat Berkeley DB Library v4.2
	7.4.5-1	Keep	n/a	□	postgresql: PostgreSQL Data Base Management System
+ Devel Default					

Figure 9.6.22 Screen shot of the Cygwin setup window.

To test if `CLASSPATH` *is set properly, open a command-line window, type* `java -h`*, and check if the command parameters are displayed. If not, check to see if the* `CLASSPATH` *variable is set properly. Remember the every time the variable value is modified, a new command-line window must be opened to refresh the setting.*

2. Download and install the latest version of Cygwin at *http://www.cygwin.com/* by clicking on one of the several Install Cygwin Now links on the home page. This will start an interactive download. After several steps, a screen will appear that makes it possible to select which packages to download (Fig. 9.6.22). The groups that must be installed in order to install the rest of the package are tabulated in Figure 9.6.23. Note that some of these will be installed by default, and some are interdependent, so that when one is chosen, another may automatically be selected.

   For demonstration purpose, it is assumed that the Root Install Directory is set as `d:\cygwin`*..*

3. Download and install Perl under Cygwin. As stated in the last step, if `interpreter->perl` and `interpreter->perl-libwin32` are chosen, the setup process automatically installs the core Perl modules.

4. Download and install additional Perl modules. The modules listed in Table 9.6.1 are not installed with core Perl, and need to be downloaded and installed from CPAN. Remember to download and install in the correct order.

5. Here `bioperl-1.5.0.tar.gz` is used as an example to detail the installation. From *http://www.bioperl.org*, click Download, then download the core BioPerl. Assuming that `bioperl-1.5.0.tar.gz` has been saved in `d:/cygwin/tmp`,

```
admin->cron                                            (BIN)
admin->cygrunsrv                                       (BIN)
base->ash                                              (BIN)
base->bash                                             (BIN)
base->cygwin                                           (BIN)
base->diff                                             (BIN)
base->diffutils                                        (BIN)
base->fileutils                                        (BIN)
base->findutils                                        (BIN)
base->gawk                                             (BIN)
base->gdbm                                             (BIN)
base->grep                                             (BIN)
base->gzip                                             (BIN)
base->libncurses5                                      (BIN)
base->libncurses6                                      (BIN)
base->libreadline4                                     (BIN)
base->libreadline5                                     (BIN)
base->login                                            (BIN)
base->ncurses                                          (BIN)
base->readline                                         (BIN)
base->sed                                              (BIN)
base->sh-utils                                         (BIN)
base->tar                                              (BIN)
base->termcap                                          (BIN)
base->terminfo                                         (BIN)
base->textutils                                        (BIN)
base->which                                            (BIN)
base->zlib                                             (BIN)

database->postgresql


Devel->cvs

Devel->cmake

Devel->cygipc

Devel->expat

Devel->gcc

Devel->gcc-core

Devel->make


Devel->cvs

Gnome->libxml2


Interpreters->expat

Interperters->libxml2

Interpreters->perl

Interpreters->perl-libwin32


Misc->setup


Shell->ash

Shell->bash

System->rebase
```

Figure 9.6.23 Groups that must be installed in order to install Cygwin.

Table 9.6.1 Perl Modules Needed for Running Chado and XoRT Under Windows

Module	URL	Current version file
HTML::Tagset	*http://search.cpan.org/~sburke/HTML-Tagset-3.04/*	`HTML-Tagset-3.04.tar.gz`
HTML::TokeParser	*http://search.cpan.org/dist/HTML-Parser*	`HTML-parser-3.45.tar.gz`
URI	*http://search.cpan.org/dist/URI/*	`URI-1.35.tar.gz`
LWP::UserAgent	*http://search.cpan.org/dist/libwww-perl/*	`libwww-perl-5.803.tar.gz`
XML::RegExp	*http://search.cpan.org/~tjmather/XML-RegExp-0.03/*	`XML-EegExp-0.03.tar.gz`
XML::DOM	*http://search.cpan.org/~tjmather/XML-DOM-1.43/*	`XML-DOM-1.43.tar.gz`
XML::Parser	*http://search.cpan.org/~kmacleod/libxml-perl-0.08/*	`libxml-perl-0.08.tar.gz`
XML::Parser::PerlSAX	*http://search.cpan.org/~kmacleod/libxml-perl-0.08/*	`libxml-perl-0.008.tar.gz`
XML::Writer	*http://search.cpan.org/~josephw/XML-Writer-0.520/*	`XML-Writer-0.530.tar.gz`
DBI	*http://search.cpan.org/~timb/DBI/*	`DBI-1.47 DBI-1.47.tar.gz`
DBD::Pg	*http://search.cpan.org/dist/DBD-Pg/*	`DBD-Pg-1.32.tar.gz`
XML::Simple	*http://search.cpan.org/~grantm/XML-Simple-2.12/*	`XML-Simple-2.14.tar.gz`
BioPerl	*http://www.bioperl.org/Core/Latest/index.shtml*	`bioperl-1.5.0.tar.gz`

launch the Cygwin shell window (not the usual command-line window) and install BioPerl using the following series of commands:

```
$ cd /tmp
$ gunzip bioperl-1.5.0.tar.gz
$ tar xvf bioperl-1.5.0.tar
$ cd bioperl-1.5.0
$ perl Makefile.PL
$ make
$ make test
$ make install
```

6. Download, install and configure PostgreSQL under Cygwin. The easiest way is to download and install PostgreSQL via the Cygwin setup program. Launch the setup program, choose PostgreSQL from the "database" menu, and press Next PostgreSQL will be downloaded and installed under Cygwin. Subsequently, configure PostgreSQL as follows:

   a. Set the environment variable CYGWIN to "server." Refer to troubleshooting section ("How to set environment variable in Window environment") for details.

   b. Install and configure cygserver using the command:
   ```
   $ cygserver-config
   ```

   c. Start cygserver:
   ```
   $ /usr/sbin/cygserver &
   ```

   d. Initialize PostgreSQL:
   ```
   $ initdb -D /var/postgresql/data
   ```

   e. Start the PostgreSQL postmaster:
   ```
   $ postmaster -D /var/postgresql/data -i &
   ```

   f. Connect to PostgreSQL:
   ```
   $ psql -d template1 -U username -W
   ```

   Here, username/password are the same as the Window login username/password.

g. Note that, the next time one connects to the database, only the following steps are necessary:

```
$ postmaster -D /var/postgresql/data -i &
$ psql -d template1 -U username -W
```

h. To shut down the database properly before logging out use:

```
$ pg_ctl stop -D /var/postgresql/data -m fast
```

7. Download and install XORT as described in Basic Protocol 1.

8. Create the Chado instance as described in Basic Protocol 2.

COMMENTARY

Background Information

Controlled vocabularies and relationships between biological objects in Chado

According to Chado schema documentation, most objects with biological meanings are defined as "features" in Chado, which are localized (if they have been localized) to a specific location on a genomic contig, linked via the featureloc table. Feature relations (in the feature_relationship table) are used to establish relationships between features other than localizations. For example, the relationship between a gene and a transcript is represented by a feature relationship of type "partof," while an allele is linked to a gene by a feature relationship of the type "AlleleOf." The core representation of genes, transcripts, and proteins, and the relationships between them, is referred to as the Central Dogma. In Chado, it is implemented as a three-level hierarchical structure: genes, transcripts, proteins, and exons are stored as features and the transcript features are stored as "partof" the gene feature through feature_relationship, whereas the exon and protein features are saved as "partof" and "producedby" the transcript features, respectively (Fig. 9.6.24).

Alignment data and other evidence in Chado

The evidence to support genome annotation includes gene prediction generated by programs such as GenScan (Burge and Karlin, 1997) and Genie (Reese et al., 2000), and other biological data such as ESTs aligned using the program Sim4 (Florea et al., 1998) and protein homologies revealed by BLASTX (Altschul et al., 1990; Mungall et al., 2002). As with the Central Dogma, they are stored in Chado as features with featureloc and feature_relationship information (Fig. 9.6.25).

Understanding the GAME XML format

GAME (genome annotation markup extensive) XML, designed by the Berkeley Drosophila Genome Project, is the major input file format for Apollo. The basic structure includes the following elements:

1. <game>: The root element, this represents the curation of one or more sequences of DNA, RNA, or amino acids. Most commonly, the <game> element represents the curation of a single sequence.

2. <seq>: Represents a sequence of DNA, RNA, or amino acids. There is generally one <seq> in the document representing the primary sequence being curated, and other <seq>'s that support the curation of the primary sequence. The primary <seq> is directly under the <game> element, and is identified by having its "focus" attribute set to "true." Each <seq> has one or more <db_xref>'s. <db_xref> indicates where the <seq> can be found using a particular unique identifier. For Drosophila curation, BDGP uses the primary <seq> to represent an accession, and other <seq>'s to represent cDNAs, protein coding sequences, and homologous sequences that are referenced by computational analyses such as tblastx.

3. <annotation> This represents a set of related sequence features and a collection of genetic information describing them. The term "sequence feature" means a segment of DNA. An annotation will generally contain a number of <feature_set>'s, each of which represents a set of related sequence features that have a specific location. A <feature_set> can contain nested <feature_set>'s (although in practice this has not yet occurred), as well as one or more <feature_span>'s, each of which represents an individual sequence feature. A <feature_span> can contain <evidence>, which specifies a result id and result type. An <annotation> can have one or more <db_xref>'s.

For Drosophila curation, the types of annotations are: gene, pseudogene, transposon, tRNA, rRNA, snRNA, snoRNA, "misc. noncoding RNA," and "miscellaneous curator's

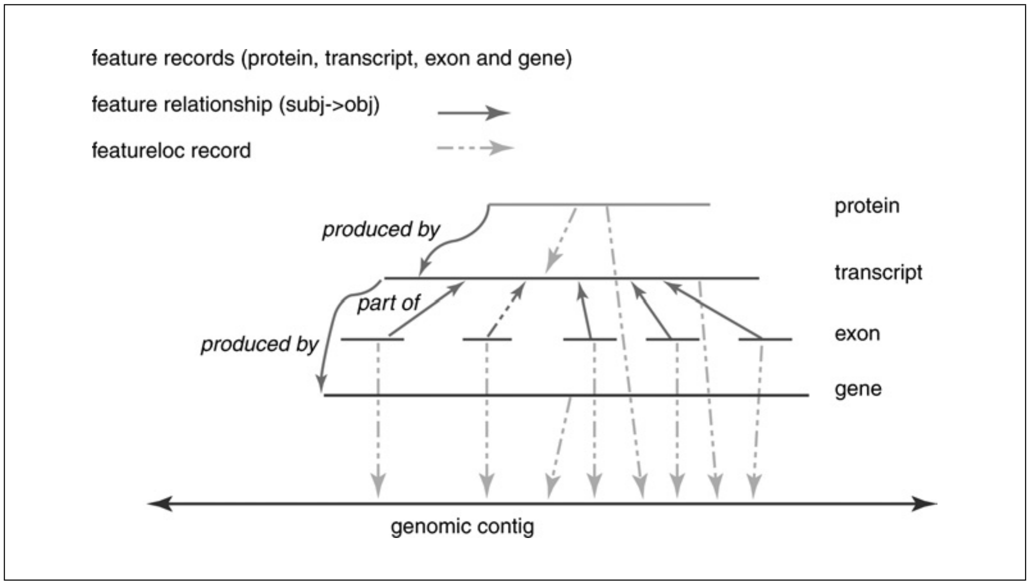


Figure 9.6.24 The Central Dogma model for a protein-coding gene with one known spliced transcript. The dashed lines denote the featureloc records of features aligned to the genomic contig, while the solid lines denote the feature_relationship records between two features (subject and object). For the color version of this figure go to *http://www.currentprotocols.com*.

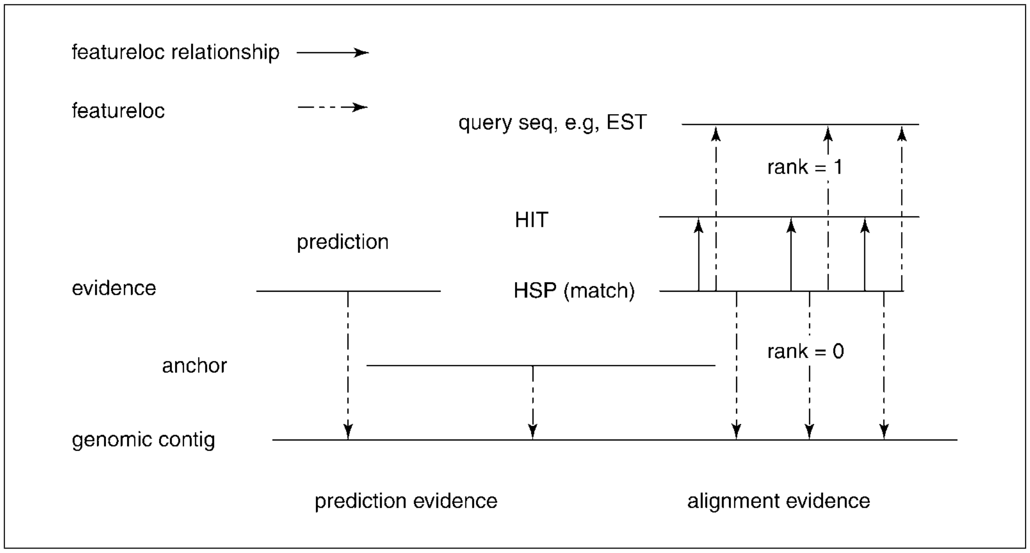


Figure 9.6.25 Data implementation of prediction and alignment evidence in Chado to support genome annotation. The dashed line denotes the featureloc of features aligned to genomic contig, while solid line denotes the feature relationship between two features.

observation." For an <annotation> of type "gene", one <feature_set> element represents each transcript, and for each transcript, one <feature_span> element represents each exon.

4. <computational_analysis> Contains evidence from computational analysis programs such sim4 and blastx. <result_set>'s and <result_span>'s represent a tree structure of results, with <result_set>'s representing branch nodes (e.g., gene matches), and <result_span>'s representing leaf nodes (e.g., exon matches). The elements <feature_set>, <feature_span>, <result_set>, and <result_span> run parallel to one another. They allow multiple levels of nesting and have physical location(s) on sequences. The key differences are that "features" have results as evidence and "results" have some form of an associated score for the assay. <seq_relationship>'s provides the locations on the underlying <seq>'s.

Critical Parameters and Troubleshooting

File system affecting file recognition in Cygwin environment

If possible, install Cygwin on a drive or partition that's NTFS-formatted instead of FAT32-formatted. When installing Cygwin on a FAT32 partition, it is not possible to set permissions and ownership correctly, which may be problematic in certain situations. If trying to use some application or resource "outside" of Cygwin and a problem is encountered, remember that Cygwin's path syntax may not be the correct one. Cygwin understands `/home/jacky` or `/cygdrive/e/cygwin/home/jacky` (when referring. e.g., to the `E:` drive), but the external resource may want `E:/cygwin/home/jacky`. Depending on these issues, the *rc files, which are normally named `.tcshrc` and `.wishrc`, and contain startup instructions for an application program, may end up with paths written in these different syntaxes.

File permission under Cygwin

Cygwin PostgreSQL may fail to start or not function properly if certain files and directories have incorrect permissions. The following usually solves such problems:

```
$ chmod a+rwx /tmp
$ chmod a+rx /usr/bin
  /usr/bin/*
$ chmod a+rw /var/log # could
  adversely affect other
  daemons
```

Make test for DBD::Pg under Cygwin

While installing DBD::Pg, the make tests are designed to connect to a live database. The following environment variables must be set for the tests to run:

```
DBI_DSN=dbi:Pg:dbname=
  <database>
DBI_USER=<username>
DBI_PASS=<password>
```

Under Cygwin, set those variables as follows: (assuming that one is logged in as `system administrator: zhou/zhoupgsql`):

```
$ DBI_DSN=dbi:Pg:dbname=
  template1
$ export DBI_DSN
$ DBI_USER=zhou
$ export DBI_USER
$ DBI_PASS=zhoupgsql
$ export DBI_PASS
```

Memory issue when converting big GAME XML files into ChadoXML

The converter uses DOM structure to build the tree structure into memory. It requires a large amount of memory. If the converter displays an "out of memory" error, it is possible to explicitly allocate more memory (e.g., 500 Mb if possible) to the Java process with the following command:

```
java -Xms500M ... ... ./conf/
  GAMEChadoConv.jar GTC ... ..
```

If the machine has limited memory, one way to get around this problem is to copy `GAMEChadoConv.jar` and input the GAME file to another machine with Java installed that has more memory, because this utility is independent of other modules.

How to enable PostgreSQL for TCP/IP

In order to use DBI to access PostgreSQL (which is required by XML-XORT), it is necessary to have one's database enabled for TCP/IP connections. Either start the database up with the `-i` switch:

```
$ postmaster -D /var/
  postgresql/data -i &
```

or enable TCP/IP settings in the database. Edit the `/var/postgresql/data/postgresql.conf` file and set:

```
tcpip_socket = true
```

How to set environment variables in the Windows environment

As an example, to add `d:\jdk1.4\bin` to `CLASSPATH` according to the instructions for the respective operating system as described under the headers below.

For Windows NT 4

Activate the Control Panel from the Start menu (under Settings), then double-click on the System icon. Choose the Environment tab. Select the `CLASSPATH` variable (in the System Variables section) by clicking on it. In the Value edit box, add `d:\jdk1.4\bin` to the front of the variable definition, but be sure not to overwrite what is already there. Note the semicolon, which separates path segments. Click the Set button, then OK.

For Windows 2000

Activate the Control Panel from the Start menu (under Settings), then double-click on the System icon. Select the Advanced tab and click on Environment Variables. Select the `CLASSPATH` variable (in the

```
D:\cygwin\bin\cygssl-0.9.7.dll     to     same     address     as
parent(0x1270000) != 0x1280000

15  [main]  perl  3132  sync_with_child:  child  1460(0x660)  died
before initialization with status code 0x1    1019 [main] perl
3132 sync_with_child: *** child state child loading dlls

LOG:  could not receive data from client: Connection reset by
peer

LOG:  unexpected EOF on client connection
```

Figure 9.6.26 The "rebase" error message from Cygwin.

System Variables section), then click Edit. Add `d:\jdk1.4\bin` to the front of the variable definition, but be sure not to overwrite what is already there. Note the semicolon, which separates path segments. Click OK on each successive screen to return to the Control Panel, then close the Control Panel window.

For Windows XP Home or Professional

Activate the Control Panel from the Start menu (under Settings). From the vertical menu on the left, select Switch to Classic View, then click on the System icon. Select the Advanced tab and click on Environment Variables. Select the `CLASSPATH` variable (in the System Variables section), then click Edit. Add `d:\jdk1.4\bin` to the front of the variable definition, but be sure not to overwrite what is already there. Note the semicolon, which separates path segments. Click OK on each successive screen to return to the Control Panel, then close the Control Panel window.

For Windows 98

Activate the Run dialog box by selecting Run from the Start menu, then enter notepad `c:\autoexec.bat`. Click OK. When the editor window appears, add the following line to the end of the file:

```
set CLASSPATH=
  d:\jdk1.4\bin;%CLASSPATH%
```

Save the file, quit the editor, then restart the computer.

Note that, to add a forward-slash directory to an existing value, one should separate the new value from the old one by a colon (`:`) instead of a semicolon (`;`). For example to add `/home/XML-XORT/lib` to an existing PERL5LIB that already has the value `/tmp/GMOD`, the value should read `/tmp/GMOD:/home/XML-XORT/lib`.

How fix a "rebase" error from Cygwin

If the error message shown in Figure 9.6.26 is encountered when running a Perl code under Cygwin, then try issuing the command `$ rebaseall -v`.

Literature Cited

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. 1990. Basic local alignment search tool. *J. Mol. Biol.* 5:215:403-10.

Burge, C. and Karlin, S. 1997. Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.* 268:78-94.

Florea, L., Hartzell, G., Zhang, Z., Rubin, G.M., and Miller, W. 1998. A computer program for aligning a cDNA sequence with a genomic DNA sequence. *Genome Res.* 8:967-974.

Lewis, S.E., Searle, S.M.J., Harris, N., Gibson, M., Iyer, V., Richter, J., Wiel, C., Bayraktarogly, L., Birney, E., Crosby, M.A., Kaminker, J.S., Matthews, B.B., Prochnik, S.E., Smith, C.D., Tupy, J.L., Rubin, G.M., Misra, S., Mungall, C.J., and Clamp, M.E. 2002. Apollo: A sequence annotation editor. *Genome Biol.* 3(12).

Mungall, C.J., Misra, S., Berman, B.P., Carlson, J., Frise, E., Harris, N., Marshall, B., Shu, S., Kaminker, J.S., Prochnik, S.E., Smith, C.D., Smith, E., Tupy, J.L., Wiel, C., Rubin, G.M., and Lewis, S.E. 2002. An integrated computational pipeline and database to support whole-genome sequence annotation. *Genome Biol.* 3(12).

Reese, M.G., Kulp, D., Tammana, H., and Haussler, D. 2000. Genie: Gene finding in *Drosophila melanogaster*. *Genome Res.* 10:529-538.

Stein, L.D., Mungall, C., Shu, S., Caudy, M., Mangone, M., Day, A., Nickerson, E., Stajich, J.E., Harris, T.W., Arva, A., and Lewis, S. 2002. The generic genome browser: A building block for a model organism system database. *Genome Res.* 12:1599-1610.

Internet Resources

http://www.gmod.org

Web site of GMOD.

http://www.flybase.org

Web site of FlyBase.

http://www.fruitfly.org/annot/gamexml.dtd.txt

Location of GAME XML DTD.

Contributed by Pinglei Zhou,
David Emmert, and Peili Zhang
Harvard University
Cambridge, Massachusetts

PubSearch and PubFetch: A Simple Management System for Semiautomated Retrieval and Annotation of Biological Information from the Literature

Database curators and biology researchers must keep track of the literature concerning their genes of interest. Such investigators have an interest in obtaining and using more sophisticated tools for this purpose than spreadsheets and laboratory notebooks. PubSearch and PubFetch comprise a literature curation system that integrates and stores literature and gene information into a single relational database. The PubSearch system provides curators with a central Web application interface to support querying and editing publication articles, genes, and keywords such as the Gene Ontology (GO) terms. It also facilitates annotating genes with keywords and article references, and allows controlled access to protected PDF documents. The PubFetch system supports PubSearch by providing a general interface to search and retrieve publications from online literature sources. An overview of the PubSearch workflow is shown in Figure 9.7.1.

In this unit, a set of protocols is provided for populating and using PubSearch and PubFetch. Basic Protocol 1 describes, in a step-by-step fashion, how to populate articles, genes, keywords, and annotations in standard format into the database. The Alternate Protocol is a procedure for populating articles using a Web interface, GO terms from the GO database, and annotations in a tab-delimited format. Basic Protocol 2 describes how to index the articles for full-text searching. Basic Protocol 3 shows how to use PubSearch to search for genes, articles, keywords, and annotations using a Web browser. Basic Protocol 4 describes ways to update and add data one item at a time using the Web browser. Basic Protocol 5 describes how to annotate genes using GO and other controlled

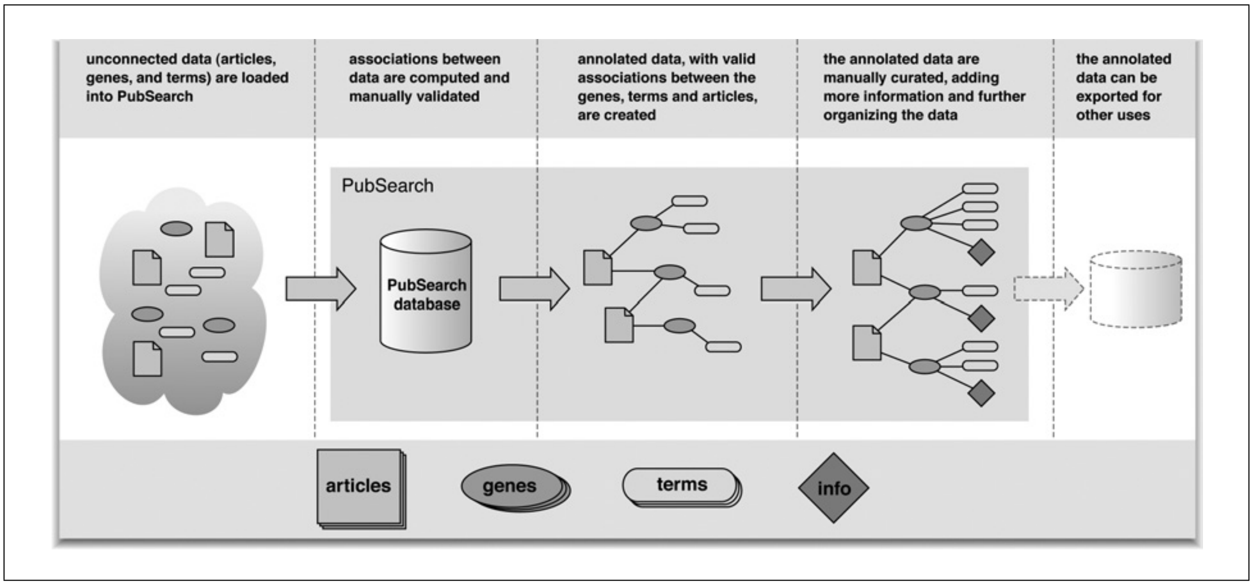

unconnected data (articles, genes, and terms) are loaded into PubSearch
associations between data are computed and manually validated
annolated data, with valid associations between the genes, terms and articles, are created
the annolated data are manually curated, adding more information and further organizing the data
the annolated data can be exported for other uses
PubSearch
PubSearch database
articles
genes
terms
info

Figure 9.7.1 An overview of the PubSearch workflow illustrating how published articles, genes, and key biological terms are brought together and integrated within PubSearch. Manual review and annotation of these data creates an annotated database of literature, genes, and related information that can be used within PubSearch alone or exported to other applications.

Contributed by Danny Yoo, Iris Xu, Vijay Narayanasamy, Tanya Z. Berardini, Simon Twigger, and Seung Yon Rhee

Current Protocols in Bioinformatics (2006) 9.7.1-9.7.27

Table 9.7.1 Guide to Conventions Used for Naming Directories

Name	Explanation
`${TMPDIR}`	A temporary scratch directory
`${WebAPP}`	The Web application directory where the servlet engine looks for installed Web applications. Apache Tomcat uses a variation of `jakarta-tomcat/Webapps`.
`${PUBHOME}`	The root directory where PubSearch will be installed. This will be `WebAPP/PubSearch` for typical installations.
`${DOMAINNAME}`	A placeholder for the domain name of the PubSearch hosting machine
`${PASSWORD}`	A placeholder for the database password used to access the "pubdb" database in MySQL
`${TOMCATBIN}`	Stands for the binary directory for Apache Tomcat. Typically, this is `apache-tomcat-[version]/bin`.

vocabulary terms. Basic Protocol 6 shows how to generate and load GO annotations from proteins that have been annotated with InterPro (*http://www.ebi.ac.uk/interpro/*) domains. The unit also provides support protocols for installing PubSearch and PubFetch. Support Protocol 1 describes how to install PubSearch; Support Protocol 2 describes how to install and run PubFetch as a stand-alone software application. Finally, the Commentary provides background information and related resources and includes information on troubleshooting potential problems and future directions of software development. More information about this project, including schema and software documentation, can be found online at *http://www.pubsearch.org*.

Conventions for naming directories that are referred to in this unit are given in Table 9.7.1. Unix conventions (also see *APPENDIX 1C*) for navigating through directories will be assumed. Unix commands will be prefixed with > to indicate the shell prompt. When commands are to be sent to the MySQL console, these commands will be prefixed with the prompt `mysql>`.

BASIC PROTOCOL 1

POPULATING PubSearch

In this protocol, an instance of the PubSearch curation system will be populated. A description of how to load articles, genes, and keywords in XML format, and annotations in GO annotation format, is provided. In addition, the protocol describes how to generate indices of the articles using the gene and keyword names. Upon completion of this protocol, a functional installation of the system will have genes, articles, keywords, and gene annotations that can be edited and queried from the Web interface.

Necessary Resources

Hardware

PubSearch has been tested on the following systems:
 Intel Xeon, 866 MHz, 2 CPUs (512 Mb RAM)
 Apple PowerBook, 1 GHz (1 Gb RAM)
Dedicated hard drive space required for indexing full text

Software

PubSearch has been tested on the following operating systems:
 Red Hat Enterprise Linux 3
 Mac OS 10.3
 PubSearch has not yet been tested on the Windows platform.

Installation of the following list of programs is a prerequisite for installing and running PubSearch:

Java JDK 1.4 or higher (*http://java.sun.com/j2se/1.4/*)
Any Java Servlet platform, such as Apache's Tomcat (*http://jakarta.apache.org/tomcat/*)
MySQL 4 (*http://mysql.com*)
In order to have transactional support, MySQL should be configured to support the INNODB table type. INNODB is described online at *http://dev.mysql.com/doc/mysql/en/innodb-overview.html*
Perl 5.8 (*http://www.cpan.org/src/README.html*)
Python 2.3 (*http://www.python.org/2.3/*)
Perl and Python are used as scripting languages to administer many of the subsystems, including cron jobs and other maintenance

The following are software requirements for performing this protocol:

GO-DB-PERL and GO-PERL Perl bindings for loading GO terms. These bindings are used to process data files that have been released by the Gene Ontology. GO-PERL and GO-DB-PERL are available as part of the standard set of development tools from the Gene Ontology's SourceForge repository at *http://sourceforge.net/projects/geneontology*.

The "go-dev" download package linked from the SourceForge page contains both Perl modules, and instructions on installing them are included in the package. PubSearch has been tested against the `go-dev-20040609-amigo2.0` release, and that version is strongly recommended.

XPDF tools from *http://www.foolabs.com/xpdf/download.html*. The current version at the time of writing is `xpdf-3.00pl3-linux.tar.gz`. XPDF is a separate set of tools to parse PDF files. XPDF includes the "pdftotext" utility, which is used to extract full text from a PDF file for searching and indexing. The XPDF source and binary distributions include instructions on how to install the XPDF toolset.

Files

A sample dataset is provided on the PubSearch Web site for demonstration purposes. The archived demonstration database can be downloaded from *http://pubsearch.org/releases/pubsearch-database-newest.sql.bz2* Note that the file is compressed (using bzip2) to conserve space.

1. Install PubSearch and PubFetch as described in Support Protocols 1 and 2.

2. Download the sample dataset from the PubSearch Web site at *http://pubsearch.org/releases/pubsearch-database-newest.sql.bz2* using the following command:

```
>wget http://pubsearch.org/releases/pubsearch-database-
newest.sql.bz2
```

3. Load the dataset into the database using the command:

```
>bzcat pubsearch-database-newest.sql.bz2 | mysql pubdb
```

Bulk load articles from XML

As an alternative to bulk loading using PubFetch, articles can be added in bulk through the command-line interface. The bulk input format is a subset of the document type definition (DTD) file `pubmed_020114.dtd` (*http://www.ncbi.nlm.nih.gov/entrez/query/DTD/pubmed_060101.dtd*) used by PubMed as part of the PubMedArticleSet (*http://eutils.ncbi.nlm.nih.gov/entrez/query/DTD/index.html*). This subset includes the following attributes: PMID, MedlinePgn, Volume, Issue, MedlineID, ArticleTitle, AbstractText, AbstractNumber, ISSN, PubSourceId, PubSourceName, and PublicationType. Note that the DTD from PubMed may change in the future. The authors will update PubSearch to work with the latest version of the DTD. Users are advised to check for the latest version of the DTD when installing PubSearch.

4. Construct a data file. This file must conform to the DTD definition described above. A sample set of articles in the PubSearch Article XML format is included in `${PUBHOME}/data/test/sample_pubmed_articles.dtd`. NCBI PubMed queries can also produce appropriate XML output for the bulk loader.

   The XML parser that is bundled with PubSearch is nonvalidating, because the selected PubMed subset that is used in the bulk article loader is itself not compliant with the PubMed DTD. In future releases of this software, the bulk article loader will define its own custom DTD format and use a validating parser for greater safety.

5. Run article importing script on the data file. Assuming that the input file is called ARTICLE.XML, go into the PubSearch home directory:

   ```
   >cd ${PUBHOME}
   ```

   Execute the article bulk loading command:

   ```
   >bin/bulk_load_articles.pl -pubsources ARTICLE.XML
   ```

   The `-pubsources` flag tells the loader to add new Publication Sources such as Journals, as necessary. If the flag is not given, then articles that refer to a nonexistent pub source will be ignored. After the command is executed, the set of articles will be entered into the PubSearch database, along with any necessary publication journals as PubSource entries.

Bulk load Term Ontologies with XML

PubSearch provides two ways of loading the Gene Ontology terms into its internal databases: loading straight from an XML data file, or from a MySQL dump of the GO database. Both methods are documented below. GO provides two types of XML dump for the ontology. PubSearch provides a loader for the GO-RDF format defined at *http://www.godatabase.org/dev/database/archive/latest/go_yyyymm-rdf.dtd.gz*. To get the latest DTD, replace "*yyyymm*" with the year and month, e.g., "*....go_200601-rdf.dtd.gz*."

6. Download the GO RDF XML file at *http://www.godatabase.org/dev/database/archive/latest/go_200503-termdb.obo-xml.gz*, using the following command:

   ```
   >wget
   http://www.godatabase.org/dev/database/archive/
   latest/go_yyyymm-termdb.obo-xml.gz
   ```

   For example, "*...go_200601-termdb.obo-xml.gz*."

7. Decompress the file:

   ```
   >gunzip go_yyyymm-termdb.obo-xml.gz
   ```

   When the year and month are substituted, this will generate a file named `go_200503-termdb.obo-xml` *in the same directory.*

8. Run the bulk term loader over the decompressed file:

```
>$ {PUBHOME}/bin/bulk_load_terms.pl go_yyyymm-
termdb.obo-xml
```

This should load all the terms and the term-to-term ontology structure into the PubSearch database.

9. Restart PubSearch by restarting the servlet container. The system keeps a cached view of the ontology graph that is updated every hour. A restart forces the system to refresh its view of the ontology:

```
>cd ${TOMCATBIN}

>bin/shutdown.sh

>bin/startup.sh
```

Bulk load genes from xml

The PubSearch software provides a simple data format for bulk loading gene objects into the system. This format is defined in:

```
${PUBHOME}/data/dtds/bulk_gene.dtd
```

10. Prepare a GeneXML file in the format described in the DTD. An example file in this format can be found in:

```
${PUBHOME}/data/test/one_gene.xml
```

Once a file has been prepared in this format, it can be run through the provided `bulk_gene_loader.pl script`.

11. Load the data file into PubSearch. For example, to load the `one_gene.xml` file:

```
>${PUBHOME}/bin/bulk_load_genes.pl

>${PUBHOME}/data/test/one_gene.xml
```

Bulk load GO gene annotations

There are currently two types of file formats for loading annotations: user-submission annotation file format, which is used by TAIR (*UNIT 1.11*), and GO annotation file format, which is used by all databases contributing GO annotations to the GO Web site. More information about the user-submission file format is found online at *http://arabidopsis.org/info/functional_annotation.submission.jsp*. More information about the GO annotation file format can be found online at *http://www.geneontology.org/GO.annotation.shtml#file*.

12. Download a GO gene association file from the GO Web site (*http://www.geneontology.org/GO.current.annotations.shtml*). For example, to retrieve *Arabidopsis thaliana* annotation data from TAIR, download the file from *ftp://ftp.geneontology.org/pub/go/gene-associations/gene_ association.tair.gz*.

13. Save file in the `${PUBHOME}/maint/tigrannotation/data` directory and unzip that file by executing the following commands:

```
>cd ${PUBHOME}/maint/tigrannotation/data

>wget ftp://ftp.geneontology.org/pub/go/gene-
associations/gene_association.tigr_Athaliana.gz
```

```
>gunzip gene_association.tigr_Athaliana.gz
```

This generates a file called `gene_association.tigr_Athaliana` *in the same directory.*

14. Run `generateAnnotationFromTigrFile.pl` on this new file to load GO annotation file by executing the following commands from the `${PUBHOME}` directory:

```
>cd maint/tigrannotation

>perl generateAnnotationFromTigrFile.pl -D
database_name
```

Generating hits

Hits are associations between terms and articles that can be generated by exact-term matching. The program `generate_hits.pl` will do a bulk search for database terms within the titles and abstracts of all articles.

15. Execute `generate_hits.pl`:

```
>${PUBHOME}/bin/generate_hits.pl
```

This step may take several minutes, depending on how many terms exist in the database.

SUPPORT PROTOCOL 1

INSTALLING PubSearch

The PubSearch Web application is one of the main components of the literature curation system. This protocol describes how to install the software on a clean machine, configure its connection to a relational database, and add initial users to the system.

Necessary Resources

Hardware

PubSearch has been tested on the following systems:

Intel Xeon, 866 MHz, 2 CPUs (512 Mb RAM)
Apple PowerBook, 1 GHz (1 Gb RAM)

Dedicated hard drive space required for indexing full text

Software

PubSearch has been tested on the following operating systems:

Red Hat Enterprise Linux 3
Mac OS 10.3

PubSearch has not yet been tested on the Windows platform.

Installation of the following list of programs is a prerequisite for installing and running PubSearch:

Java JDK 1.4 or higher (*http://java.sun.com/j2se/1.4/*)

Any Java Servlet platform, such as Apache's Tomcat (*http://jakarta.apache.org/tomcat/*)

MySQL 4 (*http://mysql.com*)

In order to have transactional support, MySQL should be configured to support the INNODB table type. INNODB is described online at *http://dev.mysql.com/doc/mysql/en/innodb-overview.html*

Perl 5.8 (*http://www.cpan.org/src/README.html*)

Python 2.3 (*http://www.python.org/2.3/*)

Perl and Python are used as scripting languages to administer many of the subsystems, including cron jobs and other maintenance

The following are software requirements for performing this protocol:

GO-DB-PERL and GO-PERL Perl bindings for loading GO terms. These bindings are used to process data files that have been released by the Gene Ontology. GO-PERL and GO-DB-PERL are available as part of the standard set of development tools from the Gene Ontology's SourceForge repository at *http://sourceforge.net/projects/geneontology*.

The "go-dev" download package linked from the SourceForge page contains both Perl modules, and instructions on installing them are included in the package. PubSearch has been tested against the `go-dev-20040609-amigo2.0` release, and that version is strongly recommended.

XPDF tools from *http://www.foolabs.com/xpdf/download.html*. The current version at the time of writing is `xpdf-3.00pl3-linux.tar.gz`. XPDF is a separate set of tools to parse PDF files. XPDF includes the "pdftotext" utility, which is used to extract full text from a PDF file for searching and indexing. The XPDF source and binary distributions include instructions on how to install the XPDF toolset.

1. Download the binary distribution of PubSearch from *http://pubsearch.org/releases/pubsearch-newest.tar.gz* into `${TMPDIR}`. Most versions of UNIX have a command called `wget` that can be used to retrieve the contents of Web URLs from a shell prompt:

```
>cd ${TMPDIR}
```

```
>wget http://pubsearch.org/releases/pubsearch-newest.
tar.gz
```

   See Table 9.7.1 for explanations of the directory names used here.

2. Untar this file from within the `${WebAPP}` directory:

```
>cd ${WebAPP}
```

```
>tar xzvf ${TMPDIR}/PubSearch-newest.tar.gz
```

   A new subdirectory called `PubSearch` will be produced underneath the `${WebAPP}` directory.

3. Change the current working directory to `${WebAPP}/pubsearch`:

```
>cd pubsearch
```

4. Initialize the PubSearch Database. Use MySQL's administrative tool `mysqladmin` to create a new database called `pubdb`:

```
>mysqladmin createdb pubdb
```

   An empty database called `pubdb` will be created.

5. Load schema structure into the pubdb database. This structure is defined in the files `schema.mysql` and `schema-support.mysql`:

```
>mysql pubdb < data/schema.mysql
```

```
>mysql pubdb < data/schema-support.mysql
```

6. Create a separate MySQL user account for PubSearch. It is strongly recommended that a separate MySQL database user be used to connect to the database, with a separate password. This can be done through the MySQL console:

```
>mysql pubdb
```

```
mysql> grant all on pubdb.* to pubuser@${DOMAINNAME}
identified by "${PASSWORD}";;
```

```
mysql> exit
```

7. Configure PubSearch's global preference file. The PubSearch Web application maintains its configuration settings in the following file:

```
${PUBHOME}/Web-INF/classes/pub/config/program.
properties
```

This file must be edited so that the system knows what resources it can use (e.g., database settings, index directories, PDF repositories):

```
>emacs ${PUBHOME}/Web-INF/classes/pub/config/program.
properties
```

The most relevant of the properties are:

```
pub.database_username
```

```
pub.database_password
```

```
pub.database_connection_string
```

```
pub.aux_data_dir
```

which should be adjusted to appropriate values. The first three property values define the values necessary to connect to the MySQL database. The last value, `pub.aux_data_dir`*, defines an auxiliary data directory that is used to store indices for the full-text search engine as well as temporary scratch space. Full-text indices typically take up 30% of the full text.*

8. Test the Database Connection. To verify that the `program.properties` file has been successfully modified, execute the following command from within the ${PUBHOME} directory:

```
>bin/test_database_connection.pl
```

If the PubSearch system can successfully connect to the `pubdb` *database, then the message* `Database connection looks good` *should be displayed. Otherwise, correct the* `program.properties` *file and repeat this step until PubSearch can connect to the database.*

9. Notify the Servlet Engine of the PubSearch Web application. Apache Tomcat rescans the Web application directory WebAPP on startup. If Tomcat is already running, shut it down, and then start it again.

```
>${TOMCATBIN}/shutdown.sh
```

```
>${TOMCATBIN}/startup.sh
```

Otherwise, start Tomcat:

```
>${TOMCATBIN}/startup.sh
```

The PubSearch Web application should be running at this point. Under default settings, the page will show up under the URL *http://${DOMAINNAME}:8080/pubsearch/*.

10. View PubSearch on a Web browser to see that the application is active.

Adding curators to the system

Only curator users are allowed to make changes to PubSearch. Curators can be added through a command-line interface. Also, if a PDF repository has been constructed, curators have access to those protected links.

11. Execute the `add_curator.pl` program. Change directory to `${PUBHOME}` and run the `command bin/add_curator.pl`.

```
>cd ${PUBHOME}

>bin/add_curator.pl
```

Prompts from the program will ask for username and initial password.

Adding regular users to the system

Regular users are allowed to inspect and query the PubSearch system, but are not allowed to make changes or to view PDF files. Regular users can be added through a command line interface, similarly to curators. The command `add_user.pl` is used.

12. Execute the `add_user.pl` program. Change directory to `${PUBHOME}` and run the `command bin/add_user.pl`:

```
>cd ${PUBHOME}

>bin/add_user.pl
```

Prompts from the program will ask for username and initial password.

Setting up administrative cron jobs for periodic maintenance

On Unix systems, a cron job can be initialized to perform regular tasks on the PubSearch system. Such tasks might include running the full text indexing, generating hits, and exporting bulk output out of PubSearch.

An example cronable script is included in `${PUBHOME}/maint/cron/pub_daily.sh`.

13. To schedule `pub_daily.sh` on a regular basis, execute:

```
>crontab -e
```

This will execute the crontab editor. Add the following entry into the crontab:

```
0 0 * * * ${PUBHOME}/maint/cron/pub_daily.sh
```

which will schedule the execution of `pub_daily.sh` every midnight.

14. Create a cronjob entry for each maintenance script desired.

INSTALLING PubFetch FOR USE OUTSIDE OF PubSearch

SUPPORT PROTOCOL 2

PubFetch is available as Java Archive (JAR) library that can be used by any other Java application. PubFetch currently contains adaptors to allow the standardized retrieval of literature references from PubMed at NCBI (National Center for Biotechnology Information) via the eUtils interface and from Agricola (National Agricultural Library) via their Machine Readable Cataloging (MARC)–based system. PubFetch is part of the standard

PubSearch release, so no further action is required to use PubFetch with PubSearch. The protocol below is employed to obtain and use PubFetch as an independent software library.

Necessary Resources

Hardware

Any computer that runs Java and has an Internet connection

Software

Java SDK available at *http:www.java.sun.com*
Xerces XML Parser (*http://xml.apache.org/*): `xercesImpl.jar` can be found in the `lib` folder of PubFetch releases
MARC4J (*http://marc4j.tigris.org/*): provides an easy to use Application Programming Interface (API) for working with MARC records in Java (`marc4j.jar` can be found in the `lib` folder of PubFetch releases
Log4j (*http://logging.apache.org/log4j/*): a logging package for Java (`log4j-1.2.8.jar`) can be found in the `lib` folder of PubFetch releases). The logging behavior can be controlled by editing the configuration file (see below)
Apache Ant: a common build utility for Java

Files

Log4J configuration file (`log_configuration.properties` can be found in the `data` folder of the release)
Entrez Journal List file containing journals in PubMed and the molecular biology databases (`J_Entrez.txt.gz` can be found in the `data` folder of the release)

1. Download the latest PubFetch binary or source files from GMOD project (Generic Model Organism Database) SourceForge site (*http://sourceforge.net/project/showfiles.php?group_id=27707*).

2. Unarchive the files using appropriate software for the operating system. For example use WinZip on the Windows operating system. On Unix and Macintosh OS X operating systems use the command:

```
>tar -zxvf file-name
```

3. To use PubFetch as an API for fetching records from Agricola and/or PubMed in Java applications, it is first necessary to install the `pubfetch.jar` final in `CLASSPATH.PubFetch.jar` is provided as part of the binary release or can be built from scratch from the source code release by running `ant` with the `jar` target:

```
>cd ${PUBFETCH}

>ant jar
```

4. Add other essential `jar` files to `CLASSPATH` such as XML Parser (`xercesImpl.jar`), MARC4J (`marc4j.jar`) and Log4J (`log4j-1.2.8.jar`). These `jar` files can be found in the `lib` folder of the release. They can also be downloaded from the appropriate software application's Web site.

5. Once installed, PubFetch provides the following features to Java clients:

Common format: A common output format is implemented so that downstream applications can easily use the retrieved literature. PubFetch retrieves articles in

MEDLINE Display Format, which is also one of the standard formats used by the GMOD (Generic Model Organism Database) project. PubFetch converts MARC Record Format to MEDLINE Display Format, in the case of Agricola, by replacing MARC tags with corresponding MEDLINE tag. For example MARC tag `245` is MEDLINE tag `TI`, which corresponds to the Title of the article.

Full text URL: PubFetch can return the URL for the full text of each document if the full text link is available in PubMed LinkOut, or PubMed Central, or if a CrossRef/DOI (Digital Object Identifier) is provided. This can be used for the subsequent download of the full text PDF for full text indexing or printing.

Duplicate filtering: When searching multiple databases, the potential exists for records to be present in both databases, resulting in a duplicate record. PubFetch provides a duplicate filtering algorithm based upon common attributes such as the Title, ISSN number, and starting page, which can be used to identify and then remove duplicate records. Cross-references to the duplicated record are maintained, so links can be created to both sources.

Explicit examples for using PubFetch as a stand-alone tool are provided in the `README` *files distributed with the release. These illustrate how to search and retrieve documents from a literature repository and also acquire URLs for full-text articles, where available.*

For more in depth explanations of the PubFetch API, `javadoc` *files are available for the source code. The can be found in the* `htdocs/javadoc` *folder in the binary release, or run* `ant-doc` *in the source release to produce the* `javadoc` *from scratch.*

OTHER WAYS TO POPULATE PubSearch

ALTERNATE PROTOCOL

As an alternative to Basic Protocol 1, this protocol explores other ways of loading data into the PubSearch database. Procedures are described for loading articles from Agricola and Medline using a Web interface, for loading an entire ontology directly from the Gene Ontology database, and for processing a bulk set of annotations from a tab-delimited input file.

Necessary Resources

See Basic Protocol 1 and Support Protocol 1

Bulk loading articles using the Web browser

Users can query online publication databases and load the results into the PubSearch database. This functionality uses the PubFetch software in the background. The user must be logged in to perform this task.

1. From the PubSearch Web interface (see Basic Protocol 1), go to Add Articles in Bulk on the Add toolbar.

2. Select the type of data source: Agricola or PubMed.

3. Enter published date range in the format `YYYY/MM/DD` (e.g., `2005/01/08`) of the articles to be retrieved.

4. Enter the keyword to be used in the input box following Search For; this will limit the search to those articles which have this keyword in their titles or abstracts.

5. Click the FetchToPub button.

   After the user clicks the FetchToPub button, the underlying PubFetch application will fetch the articles the user wants from the user-specified data source and add the articles into PubSearch after filtering out the duplicates. If the publication source of the article (e.g., journal) does not exist in the database, the application adds the journal automatically.

6. Confirm/Check the inserted articles. After the software fetches and inserts the articles, a summary page will be displayed with three sections, as follows:

   a. Number of new articles that were added to the database. For each added article, the page has a link to the article detail page, where the user can check the detail of that article and add/modify article information.

   b. Number of new journals that were added. This will also link to the journal detail page, where the user can modify journal information. Journal entries are occasionally duplicated with some existing journals in database due to slight differences in naming conventions in the source data. If this is the case, the user can merge the two entries by "obsoleting" (i.e., by marking the entry obsolete in the database, one is effectively "deleting" it without removing it from the database) the new journal and replacing it with the other journal. This step will also associate any articles that were linked to the old journal with the new one.

   c. Number of articles that were skipped because they were duplicates of existing entries. For the duplicated articles, the article entries in the database and entries from the data source are displayed side by side so the user can compare them and modify/add information to the article entries in database.

Bulk loading term ontologies from a Gene Ontology database

7. Load a Gene Ontology database.

   The Gene Ontology defines an SQL schema for storing terms and the associated relationships. These can be found on the GO Web site at http://geneontology.org. PubSearch contains a set of loading scripts to read the native Gene Ontology databases and to import the terms and ontologies into a local database.

8. Download a suitable ontology dump file. For example, the GO consortium publishes a dump of its term database once a month. Use the following command:

```
>wget
http://archive.godatabase.org/latest/go_yyyymm-termdb-
tables.tar.gz
```

   The string "yyyymm" is to be replaced with the latest year and month in the file name, e.g., for January, 2006, the file name would be go_200601-termdb-tables.tar.gz.

9. Restore the MySQL dump into the local database. For the purposes of this guide, assume the database is named gene_ontology.

```
>mysqladmin create gene_ontology

>tar xzvf go_yyyymm-termdb-tables.tar.gz

>cd go_yyyymm-termdb-tables

>cat *.sql | mysql gene_ontology

>mysqlimport -L gene_ontology *.txt
```

10. Run Gene Ontology loaders. Once the Gene Ontology MySQL database is created, the following steps will add terms and term-to-term ontology relationships into the PubSearch system.

```
>cd ${PUBHOME}/maint/gene_ontology

>perl add_goterm_to_pubterm.pl

>python load_term2term.py
```

```
>cd ${PUBHOME}

>perl bin/import_go_term_synonyms.pl gene_ontology
```

Loading user-submitted annotations

The bulk annotation loader provides a way to load gene-to-term associations with article references. The loader takes a tab-delimited file as input, and the format that the file reads is documented in `${PUBHOME}/maint/bulk_annotation_loader/format.txt`. An example file is included in `${PUBHOME}/maint/bulk_annotation_loader/sample/example.txt`.

11. To load the annotations, execute the `bulk_load_annotations.pl` script on the data file:

```
>cd ${PUBHOME}/maint/bulk_annotation_loader

>perl bulk_load_annotations.pl sample/example.txt
```

SETTING UP A PDF REPOSITORY FOR FULL-TEXT INDEXING

BASIC PROTOCOL 2

Articles in the PubSearch database often have Adobe Portable Document Format (PDF) files associated with them. These documents can be processed by PubSearch's full-text index system, thereby enabling a powerful full-text search engine. However, access to these PDFs may need to be restricted due to licensing issues. PubSearch provides a rudimentary scheme for restricting PDF access to curators only. The protocol below describes how to set aside a protected PDF repository for PubSearch and schedule regular maintenance of a full-text index for PDF documents.

Necessary Resources

See Basic Protocol 1 and Support Protocol 1

Setting up A PDF repository for full-text indexing

1. Set aside a directory for documents. This directory can be placed in any file system with sufficient storage. Once a directory has been made, PubSearch must be configured to use that directory, using the `program.properties` configuration file described in the PubSearch installation section.
2. Edit the `program.properties` file in the `${PUBHOME}/Web-INF/classes/pub/config` directory. Change the property `pub.pdf_document_base` to the PDF repository directory.
3. Once a PDF repository directory is configured, PDFs can be copied into the repository. The filename of each PDF must be named to match the article id within the PubSearch system.

   In a future revision of the system, a Web interface for adding PDFs will be implemented.
4. Once a PDF has been copied into the PDF repository directory, a FullText URL will be available from the article detail page. Only logged-in curators with the "can_access_pdfs" capability will be allowed to access the URL. At the current time, SQL update statements issued from the MySQL client are required to change this property.

Generating full-text indices

5. Generate text files for article PDFs by executing the following command from the `${PUBHOME}` directory:

```
$cd maint/perl

$perl extract_full_text.pl
```

PubSearch sets up a cron job for this task (see Support Protocol 1).

6. Set up a PDF repository for storing article full text, and specify this resource in `program.properties` file in the `${PUBHOME}` directory. The following illustrates the settings from PubSearch:

```
## Where are the PDFs located on the system?

pub.pdf_document_base = /opt2/pub_documents
```

7. Set aside a directory for storing the indices that the Lucene search engine will generate. The following lines are the configuration from PubSearch's `program.properties` file:

```
## Directory where Pub can store auxiliary data
(indices)

pub.aux_data_dir = /opt2/PubSearch/var
```

8. Add the following lines to the `program.properties` file (see Basic Protocol 1). The settings below indicate to the system which specific content types are used as document collections. Classes are separated from one another by a comma. PubSearch comes with three standard document type index classes listed below:

```
pub.lucene_document_iterators = pub.db.search.
LuceneTermIterator,
pub.db.search.LuceneGeneIterator,
pub.db.search.LuceneArticleIterator
```

9. To generate the indices for full text, execute the following commands under the `${PUBHOME}` directory:

```
>cd ${PUBHOME}

>bin/index_full_text.pl
```

BASIC PROTOCOL 3

USING PubSearch TO SEARCH DATA

Once data have been warehoused into PubSearch, the data can be queried from the Web interface. Procedures are described below for logging into the system as a privileged curator and performing simple and complex queries. The intended users for this protocol are biologists, database curators, computational biologists, bioinformaticians, or bench scientists who need to manage a large amounts of literature and gene data. This protocol uses the following convention for the URL where PubSearch can be accessed: `http://${DOMAINNAME}:8080/pubsearch`.

Necessary Resources

See Basic Protocol 1 and Support Protocol 1

Navigation and home page

The page header displayed on top of every page (Fig. 9.7.2) illustrates all of the functionalities available in PubSearch. In addition to logging in, searching, browsing, and adding data, users can access curation and usage guides, as well as submit bugs. The home page displays a short description of the software, recent changes, and database statistics.

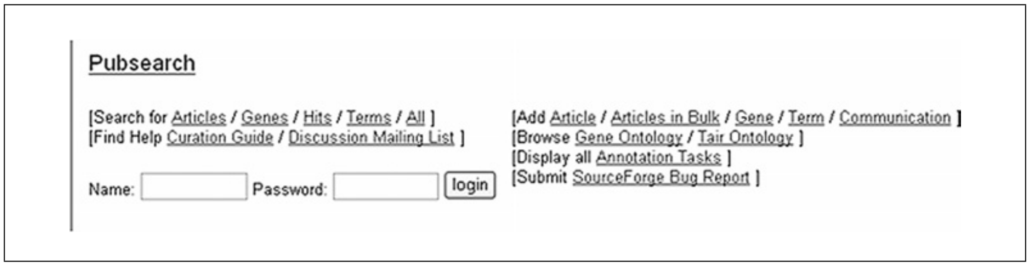

Pubsearch
[Search for Articles / Genes / Hits / Terms / All]
[Find Help Curation Guide / Discussion Mailing List]
Name:
Password:
login
[Add Article / Articles in Bulk / Gene / Term / Communication]
[Browse Gene Ontology / Tair Ontology]
[Display all Annotation Tasks]
[Submit SourceForge Bug Report]

Figure 9.7.2 A screenshot of the Navigation toolbar of PubSearch Web user interface. It lists the different types of user functions, links to usage guide, and text boxes for logging in.

Logging into the database

Users can log in from any page in PubSearch. Logging in is required for updating, inserting, and viewing full-text articles. Searching and viewing results do not require logging in.

1. In the page header, type the user name in the Name input box, and enter the password in the Password input box.

Searching the database

It is not necessary to be logged in to search the database. There are two types of search interfaces available, a simple search of all datatypes, and advanced searches for each of the major datatypes.

Simple searching

2. From the page header, click the All hyperlink in the "Search for" section or go directly to the simple search interface from the URL (*http://${DOMAINNAME}:8080/pubsearch/Search?middle_page=ALL*).

3. Type the search string (both words and phrases are accepted) in the text input box next to Query:, then click the Submit button. For example, try typing "`water channel`" (including the quotes, which searches for the phrase).

   This search uses Lucene's full-text search algorithm. For a complete list of available query string options and formats, refer to http://lucene.apache.org/java/docs/queryparsersyntax.html.

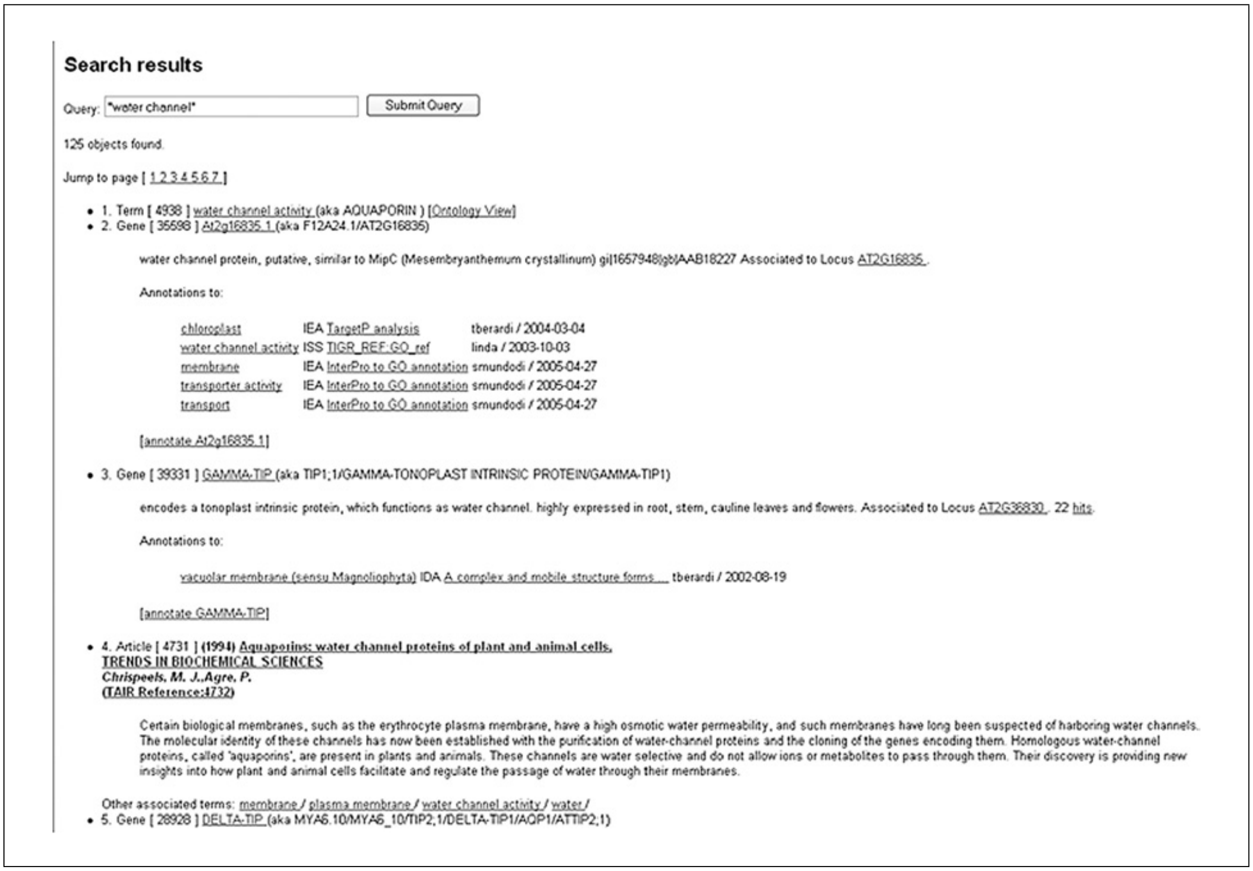
4. Results are ordered in terms of how they score in terms of "density" in the Lucene search engine (frequency of term occurrence per document size; Fig. 9.7.3). In the example query, from TAIR's PubSearch instance, 125 results are retrieved, which include genes, controlled vocabulary terms, and articles. Clicking on the name/title of the gene, term, or article leads to the detail page of the data object.

   The simple search function searches all of the data fields. It is experimental and is not supported.

Advanced searching

There are four types of data—articles, genes, terms, and hits—that can be searched by using more parameters. These data types are listed in the page header. The user interface and usage of the search and result pages for all of these data types are similar. Therefore, only the hit search is described here. "Hit search" is used to find papers that are associated with a gene (or other types of terms) of interest. Both articles and terms can be restricted for finding matches between papers and terms.

5. Go to the Hit search page by clicking on the Hit hyperlink in the Search section of the page header (Fig. 9.7.4).



Search results

Query: "water channel" Submit Query

125 objects found.

Jump to page [1 2 3 4 5 6 7]

- 1. Term [4938] water channel activity (aka AQUAPORIN) [Ontology View]
- 2. Gene [35598] At2g16835.1 (aka F12A24.1/AT2G16835)

  water channel protein, putative, similar to MipC (Mesembryanthemum crystallinum) gi|1657948|gb|AAB18227 Associated to Locus AT2G16835 .

  Annotations to:

  chloroplast IEA TargetP analysis tberardi / 2004-03-04
  water channel activity ISS TIGR_REF:GO_ref linda / 2003-10-03
  membrane IEA InterPro to GO annotation smundodi / 2005-04-27
  transporter activity IEA InterPro to GO annotation smundodi / 2005-04-27
  transport IEA InterPro to GO annotation smundodi / 2005-04-27

  [annotate At2g16835.1]

- 3. Gene [39331] GAMMA-TIP (aka TIP1;1/GAMMA-TONOPLAST INTRINSIC PROTEIN/GAMMA-TIP1)

  encodes a tonoplast intrinsic protein, which functions as water channel. highly expressed in root, stem, cauline leaves and flowers. Associated to Locus AT2G36830 . 22 hits.

  Annotations to:

  vacuolar membrane (sensu Magnoliophyta) IDA A complex and mobile structure forms ... tberardi / 2002-08-19

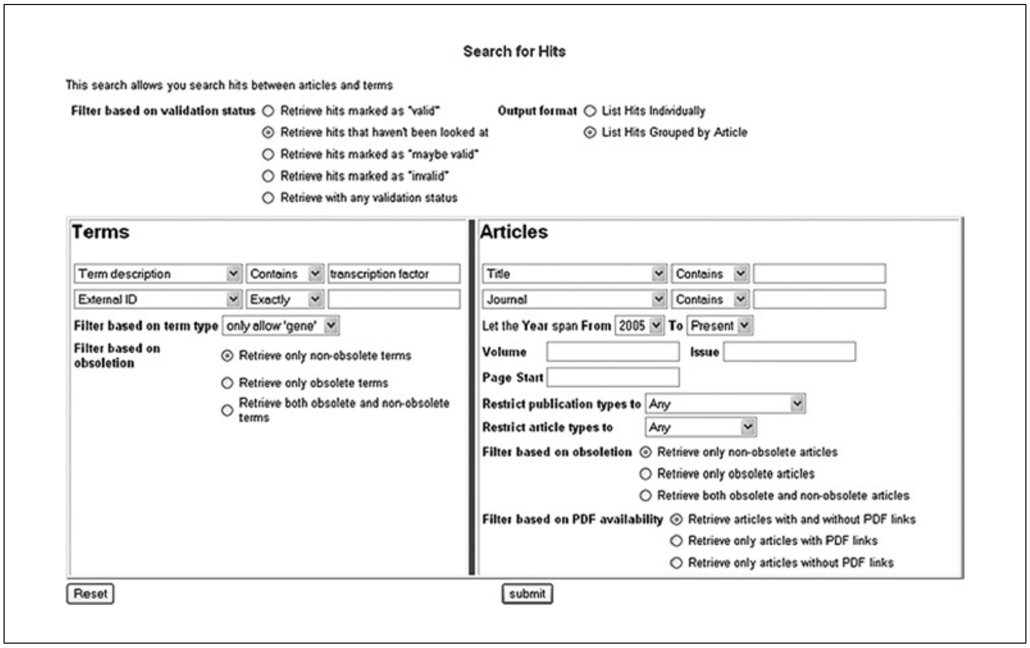
  [annotate GAMMA-TIP]

- 4. Article [4731] **(1994) Aquaporins: water channel proteins of plant and animal cells.**
  TRENDS IN BIOCHEMICAL SCIENCES
  Chrispeels, M. J.,Agre, P.
  (TAIR Reference:4732)

  Certain biological membranes, such as the erythrocyte plasma membrane, have a high osmotic water permeability, and such membranes have long been suspected of harboring water channels. The molecular identity of these channels has now been established with the purification of water-channel proteins and the cloning of the genes encoding them. Homologous water-channel proteins, called 'aquaporins', are present in plants and animals. These channels are water selective and do not allow ions or metabolites to pass through them. Their discovery is providing new insights into how plant and animal cells facilitate and regulate the passage of water through their membranes.

  Other associated terms: membrane / plasma membrane / water channel activity / water /
- 5. Gene [28928] DELTA-TIP (aka MYA6.10/MYA6_10/TIP2;1/DELTA-TIP1/AQP1/ATTIP2;1)

Figure 9.7.3 A screenshot of the Search All function's result page showing the first page of results from a search with "water channel" (including quotes) as query string. Results are displayed in the order of "density" of the match, which is a measure of the frequency of the matching string over the length of the entry. Underlined text (shown also in blue on screen) indicates a hyperlink to more information.



Search for Hits

This search allows you search hits between articles and terms

Filter based on validation status
○ Retrieve hits marked as "valid"
● Retrieve hits that haven't been looked at
○ Retrieve hits marked as "maybe valid"
○ Retrieve hits marked as "invalid"
○ Retrieve with any validation status

Output format
○ List Hits Individually
● List Hits Grouped by Article

Terms

Term description | Contains | transcription factor
External ID | Exactly |
Filter based on term type only allow 'gene'
Filter based on obsoletion
● Retrieve only non-obsolete terms
○ Retrieve only obsolete terms
○ Retrieve both obsolete and non-obsolete terms

Articles

Title | Contains |
Journal | Contains |
Let the Year span From 2005 **To** Present
Volume **Issue**
Page Start
Restrict publication types to Any
Restrict article types to Any
Filter based on obsoletion
● Retrieve only non-obsolete articles
○ Retrieve only obsolete articles
○ Retrieve both obsolete and non-obsolete articles
Filter based on PDF availability
● Retrieve articles with and without PDF links
○ Retrieve only articles with PDF links
○ Retrieve only articles without PDF links

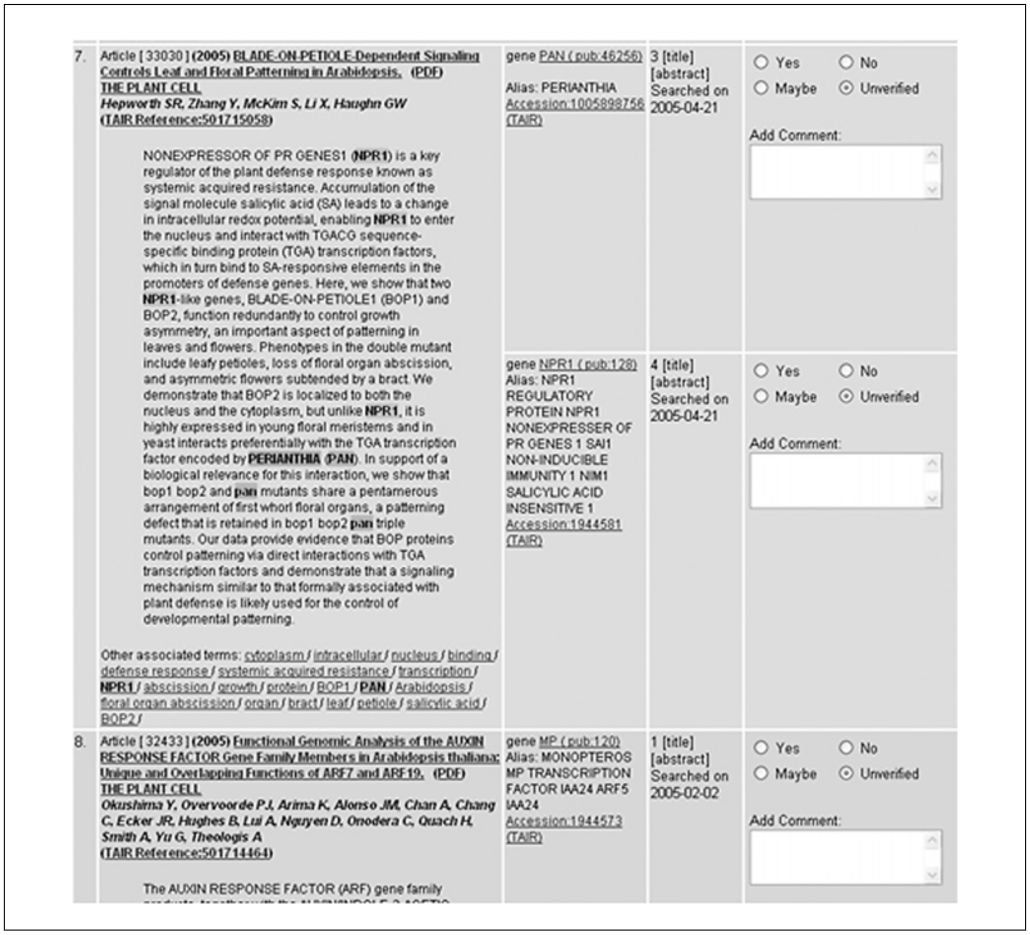
Figure 9.7.4 A screenshot of the Search Hits function. Users can restrict the search by terms (lower left box), articles (lower right box), and validation status of the automated hits between terms and articles (upper left). Options for displaying the results are listed in the upper right corner.

6. For the “Filter based on validation status” parameter, click “Retrieve hits that haven’t been looked at” radio button to retrieve hits that have not been validated manually.

   There are three types of parameters to restrict the search: validation status, terms, and articles. All hits are generated automatically by the software, which can be validated by users using the Web browser. Valid hits refer to those that have been validated by a user.

7. To restrict the search by terms, the Term section on the left side of the search page can be used. Users can limit the search by term name, description, ID(s), type, and obsoletion status. As an example, type `transcription factor` (without quotes) in the first text input box, change the drop-down menu to “Term description,” and choose the “Contains” option. Leave the “Filter by term type” drop-down menu as the default. This will limit the search to all genes whose description contains the phrase `transcription factor`.

8. To restrict the search by articles, the Article section on the right side of the search page can be used. Users can limit the search by year of publication, title, authors, abstract, journal name, ID(s), publication type (e.g., journal article or book chapter), article type (e.g., research article or review), obsoletion status, and local full-text availability. As an example, limit the year of the publication by selecting “2005” from the “Let the Year span From” parameter. This will limit the search to all articles published in 2005 or later.



7. Article [33030] **(2005)** BLADE-ON-PETIOLE-Dependent Signaling Controls Leaf and Floral Patterning in Arabidopsis. (PDF)
THE PLANT CELL
Hepworth SR, Zhang Y, McKim S, Li X, Haughn GW
(TAIR Reference:501715058)

NONEXPRESSOR OF PR GENES1 (**NPR1**) is a key regulator of the plant defense response known as systemic acquired resistance. Accumulation of the signal molecule salicylic acid (SA) leads to a change in intracellular redox potential, enabling **NPR1** to enter the nucleus and interact with TGACG sequence-specific binding protein (TGA) transcription factors, which in turn bind to SA-responsive elements in the promoters of defense genes. Here, we show that two **NPR1**-like genes, BLADE-ON-PETIOLE1 (BOP1) and BOP2, function redundantly to control growth asymmetry, an important aspect of patterning in leaves and flowers. Phenotypes in the double mutant include leafy petioles, loss of floral organ abscission, and asymmetric flowers subtended by a bract. We demonstrate that BOP2 is localized to both the nucleus and the cytoplasm, but unlike **NPR1**, it is highly expressed in young floral meristems and in yeast interacts preferentially with the TGA transcription factor encoded by **PERIANTHIA** (**PAN**). In support of a biological relevance for this interaction, we show that bop1 bop2 and **pan** mutants share a pentamerous arrangement of first whorl floral organs, a patterning defect that is retained in bop1 bop2 **pan** triple mutants. Our data provide evidence that BOP proteins control patterning via direct interactions with TGA transcription factors and demonstrate that a signaling mechanism similar to that formally associated with plant defense is likely used for the control of developmental patterning.

Other associated terms: cytoplasm / intracellular / nucleus / binding / defense response / systemic acquired resistance / transcription / **NPR1** / abscission / growth / protein / BOP1 / **PAN** / Arabidopsis / floral organ abscission / organ / bract / leaf / petiole / salicylic acid / BOP2 /

gene PAN (pub:46256)

Alias: PERIANTHIA
Accession:1005898756
(TAIR)

3 [title]
[abstract]
Searched on
2005-04-21

○ Yes ○ No
○ Maybe ◉ Unverified

Add Comment:

gene NPR1 (pub:128)
Alias: NPR1
REGULATORY
PROTEIN NPR1
NONEXPRESSER OF
PR GENES 1 SAI1
NON-INDUCIBLE
IMMUNITY 1 NIM1
SALICYLIC ACID
INSENSITIVE 1
Accession:1944581
(TAIR)

4 [title]
[abstract]
Searched on
2005-04-21

○ Yes ○ No
○ Maybe ◉ Unverified

Add Comment:

8. Article [32433] **(2005)** Functional Genomic Analysis of the AUXIN RESPONSE FACTOR Gene Family Members in Arabidopsis thaliana: Unique and Overlapping Functions of ARF7 and ARF19. (PDF)
THE PLANT CELL
Okushima Y, Overvoorde PJ, Arima K, Alonso JM, Chan A, Chang C, Ecker JR, Hughes B, Lui A, Nguyen D, Onodera C, Quach H, Smith A, Yu G, Theologis A
(TAIR Reference:501714464)

The AUXIN RESPONSE FACTOR (ARF) gene family

gene MP (pub:120)
Alias: MONOPTEROS
MP TRANSCRIPTION
FACTOR IAA24 ARF5
IAA24
Accession:1944573
(TAIR)

1 [title]
[abstract]
Searched on
2005-02-02

○ Yes ○ No
○ Maybe ◉ Unverified

Add Comment:

Figure 9.7.5 A screenshot of Search Hits result page. Results are grouped by each article. The first column shows article information, the second column shows matching genes, the third column indicates information about the matching, and the fourth column displays the options for validating the matches between the papers and genes. Underlined text (shown also in blue on screen) indicates hyperlinks to more information.

9. There are two types of output formats: List Hits Individually and List Hits Grouped by Article. The first option lists individual hits ordered by article ID. Choose the second option, and hit the Submit button.

10. The results page (Fig. 9.7.5) shows how many matches (in this case, matching articles) are retrieved. Each row of result has four columns that contain article information, matched term information, details about the match, and a column that allows logged in users to validate the match. Hyperlinks lead to the detail pages of the articles, hits, and other PubSearch objects.

BASIC PROTOCOL 4

USING PubSearch TO ADD AND UPDATE DATA

PubSearch also provides basic interfaces for curating data in its database. All searchable data types can be edited using the Web browser by logged in users. The mechanism for editing is similar for all data types. This protocol describes the updating of existing article data as well as the addition of new articles from the Web interface.

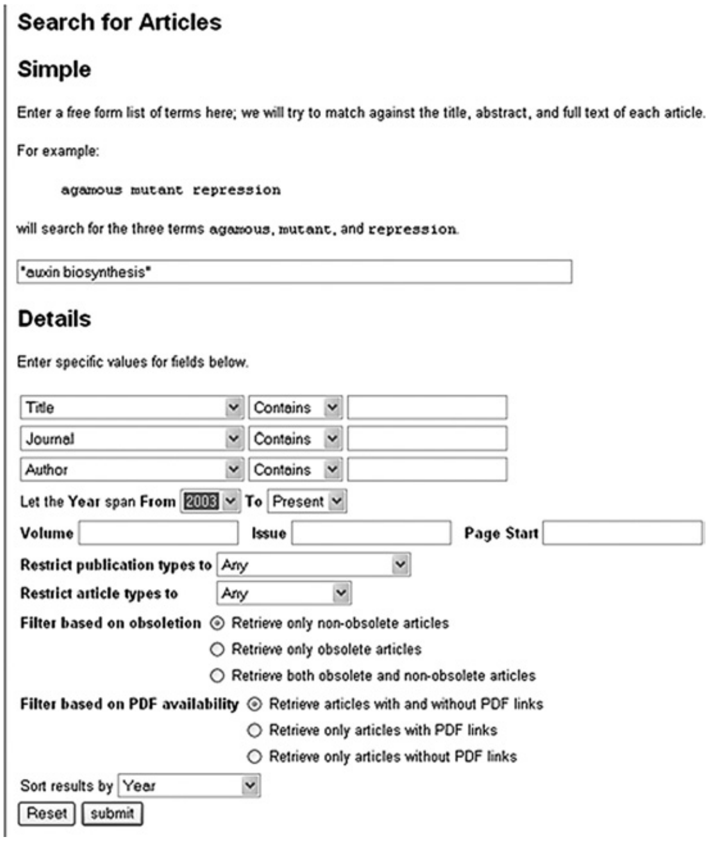
Necessary Resources

See Basic Protocol 1 and Support Protocol 1

Updating article information

1. Go to the “Article search” page (Fig. 9.7.6) by clicking on the Article hyperlink in the “Search for” page header.

2. Search for articles of interest by using the parameters provided on the search page. For example, type `"auxin biosynthesis"` (including quotes) in the text input



Search for Articles

Simple

Enter a free form list of terms here; we will try to match against the title, abstract, and full text of each article.

For example:

```
agamous mutant repression
```

will search for the three terms `agamous`, `mutant`, and `repression`.

"auxin biosynthesis"

Details

Enter specific values for fields below.

Title	Contains	
Journal	Contains	
Author	Contains	

Let the Year span From 2003 **To** Present

Volume **Issue** **Page Start**

Restrict publication types to Any

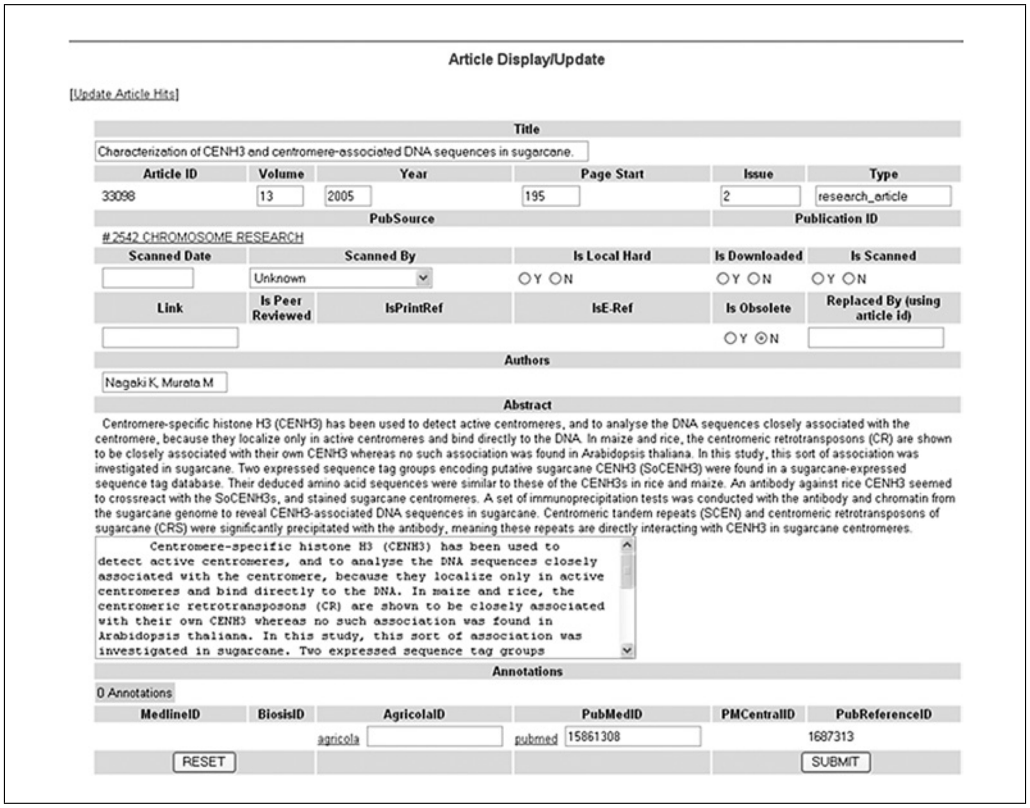
Restrict article types to Any

Filter based on obsoletion ● Retrieve only non-obsolete articles
○ Retrieve only obsolete articles
○ Retrieve both obsolete and non-obsolete articles

Filter based on PDF availability ● Retrieve articles with and without PDF links
○ Retrieve only articles with PDF links
○ Retrieve only articles without PDF links

Sort results by Year

Figure 9.7.6 A screenshot of the Article Search form.



Article Display/Update

[Update Article Hits]

Title

Characterization of CENH3 and centromere-associated DNA sequences in sugarcane.

Article ID	Volume	Year	Page Start	Issue	Type
33098	13	2005	195	2	research_article

PubSource	Publication ID
# 2542 CHROMOSOME RESEARCH	

Scanned Date	Scanned By	Is Local Hard	Is Downloaded	Is Scanned
	Unknown	○ Y ○ N	○ Y ○ N	○ Y ○ N

Link	Is Peer Reviewed	IsPrintRef	IsE-Ref	Is Obsolete	Replaced By (using article id)
				○ Y ● N	

Authors

Nagaki K, Murata M

Abstract

Centromere-specific histone H3 (CENH3) has been used to detect active centromeres, and to analyse the DNA sequences closely associated with the centromere, because they localize only in active centromeres and bind directly to the DNA. In maize and rice, the centromeric retrotransposons (CR) are shown to be closely associated with their own CENH3 whereas no such association was found in Arabidopsis thaliana. In this study, this sort of association was investigated in sugarcane. Two expressed sequence tag groups encoding putative sugarcane CENH3 (SoCENH3) were found in a sugarcane-expressed sequence tag database. Their deduced amino acid sequences were similar to these of the CENH3s in rice and maize. An antibody against rice CENH3 seemed to crossreact with the SoCENH3s, and stained sugarcane centromeres. A set of immunoprecipitation tests was conducted with the antibody and chromatin from the sugarcane genome to reveal CENH3-associated DNA sequences in sugarcane. Centromeric tandem repeats (SCEN) and centromeric retrotransposons of sugarcane (CRS) were significantly precipitated with the antibody, meaning these repeats are directly interacting with CENH3 in sugarcane centromeres.

```
      Centromere-specific histone H3 (CENH3) has been used to
detect active centromeres, and to analyse the DNA sequences closely
associated with the centromere, because they localize only in active
centromeres and bind directly to the DNA. In maize and rice, the
centromeric retrotransposons (CR) are shown to be closely associated
with their own CENH3 whereas no such association was found in
Arabidopsis thaliana. In this study, this sort of association was
investigated in sugarcane. Two expressed sequence tag groups
```

Annotations

0 Annotations

MedlineID	BiosisID	AgricolaID	PubMedID	PMCentralID	PubReferenceID
		agricola	pubmed 15861308		1687313

Figure 9.7.7 A screenshot of the Article Detail page. Logged users can update the fields on this page.

box under the Simple section, and change the "Let the Year span From:" drop-down menu to "2003."

The text input box in the Simple section of this page searches the full text of all of the articles in a manner similar to that of Google. If phrases are not enclosed within quotes, individual words will be searched separately.

3. All articles that contain the phrase `"auxin biosynthesis"` in the text, and that were published in 2003 or later, will be retrieved.

4. Results are displayed grouped by articles. For each article, the year, title, journal, authors, and abstract are displayed. In addition, links to the PDF version of the full text, associated terms and genes, and article detail page are provided. To go to the article detail page to edit the information, click on the title.

5. If logged in, a number of fields will be seen that can be updated in the form of text boxes, radio buttons, and drop-down menus (Fig. 9.7.7). Multiple fields can be updated at once. Update the fields as necessary, then click the Submit button.

6. If the publication source information (e.g., periodicals) needs to be updated, click on the PubSource name.

7. Modify the updatable fields as necessary, then click the Submit button.

Adding data individually

In addition to the bulk import of data described in Basic Protocol 1 and the Alternate Protocol, logged-in users can insert new data entries using the Web forms. Currently data types that can be added into the database via these Web forms include articles, publication sources, genes, sequences, alleles, germplasms, terms, hits, and controlled vocabulary annotations. Web forms for adding new data can be found by clicking on the data object names in the "Add:" toolbar on top of each page (e.g., articles, genes, terms; Fig. 9.7.2). In addition, data that are associated with genes or articles such as alleles, sequences, germplasms, or publication sources can be added from the gene or article update pages. The principle for adding new data objects is the same for all objects. In this chapter, adding articles individually is used as an example.

Adding article information

8. Log in to the database as described in Basic Protocol 3.

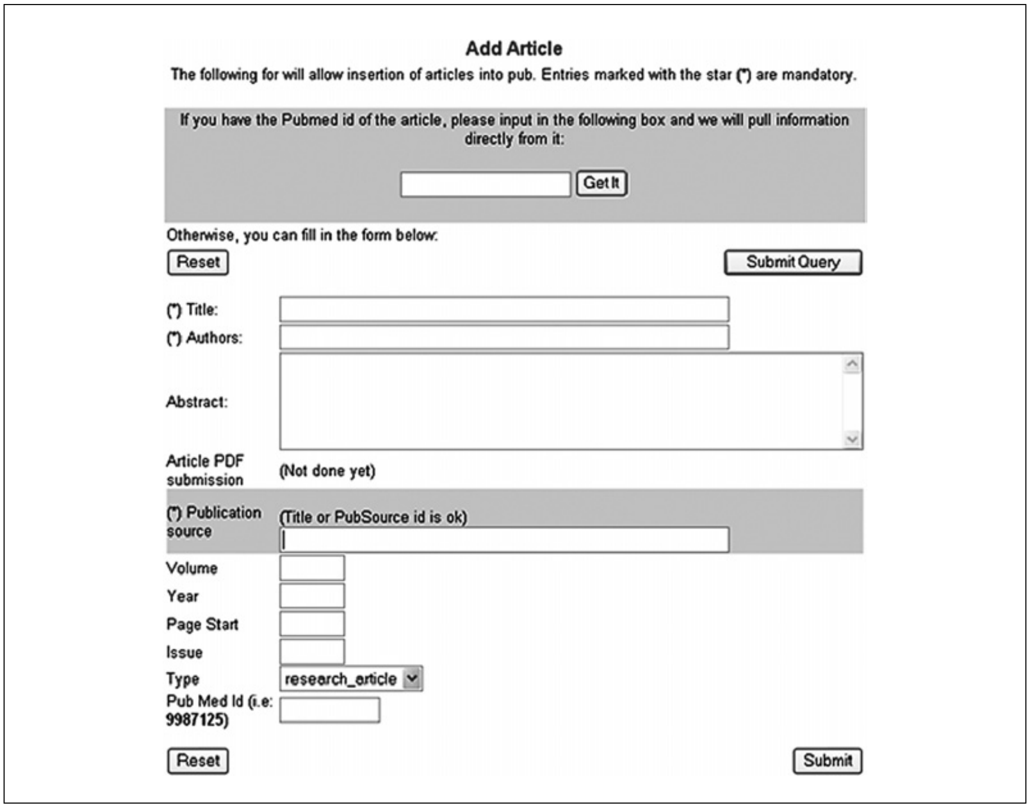
9. Click on Article in the "Add:" toolbar to get to the Add Article page (Fig. 9.7.8).

10. There are two ways of adding an article into the database. If the PUBMED ID is known, enter the ID in the text input box. For example, enter `15861308` and click the "Get it" button.

11. If the article does not exist in the database, the article information will be automatically entered into the database and an article update page will be displayed. Check the data and update the information if necessary.

    If the article already exists in the database, the user will be given an error page that includes a link to the existing article entry. Click on the link to check that the correct article is in the database.

12. If the article to be entered into the database does not have PubMed ID, the fields can be filled in manually on the Add Article page (Fig. 9.7.8). Fields marked with an asterisk (*) are required.



Add Article

The following for will allow insertion of articles into pub. Entries marked with the star (*) are mandatory.

If you have the Pubmed id of the article, please input in the following box and we will pull information directly from it:

Otherwise, you can fill in the form below:

(*) Title:

(*) Authors:

Abstract:

Article PDF submission (Not done yet)

(*) Publication source (Title or PubSource id is ok)

Volume

Year

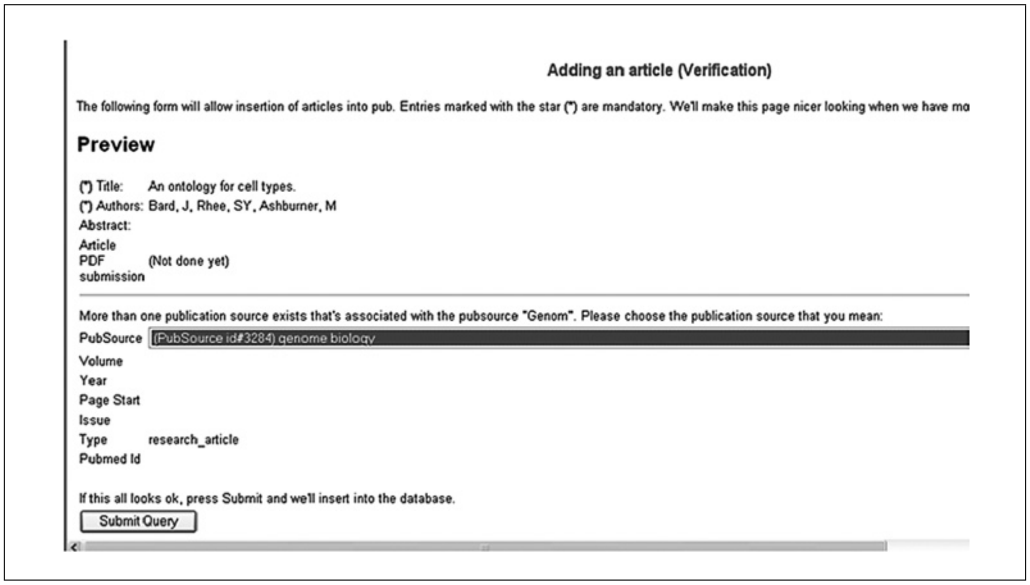
Page Start

Issue

Type research_article

Pub Med Id (i.e: 9987125)

Figure 9.7.8 A screenshot of the Add an Article form. This form allows users to insert an individual article. Entering the PubMed ID will retrieve all the article information from PubMed automatically using the PubFetch software, check for duplicates with articles in PubSearch database, insert the article if it does not yet exist in the database, and allow users to update the retrieved fields if necessary. If the PubMed ID is not known, users can enter the fields of the article.



Adding an article (Verification)

The following form will allow insertion of articles into pub. Entries marked with the star (*) are mandatory. We'll make this page nicer looking when we have mo

Preview

(*) Title: An ontology for cell types.
(*) Authors: Bard, J, Rhee, SY, Ashburner, M
Abstract:
Article PDF submission (Not done yet)

More than one publication source exists that's associated with the pubsource "Genom". Please choose the publication source that you mean:

PubSource (PubSource id#3284) genome biology
Volume
Year
Page Start
Issue
Type research_article
Pubmed Id

If this all looks ok, press Submit and we'll insert into the database.

Figure 9.7.9 A screenshot of the Add an Article function's preview page. If the fields of the new article have been entered manually, the preview page allows users to choose the correct publication source using a drop-down menu, or to add a new publication source.

13. If the input string for the publication source (e.g., Journal or Book Series) matches existing publication sources, one will be redirected to a page with a drop-down list of publication sources from which to choose the source (Fig. 9.7.9). Choose the correct journal and, if all other fields are correct, enter Submit. To change any fields, use the Back button of the browser to go back to the previous page to update the fields as necessary.

14. If the input string for the publication source does not exist in the database, the user will be notified so as to be able to go back and update the search parameter or go to a page to add it as a new publication source.

15. Click on the hyperlink to add a new publication source. Add the necessary fields and click the Submit button. This will return the article update page with the article data that has just been inserted into the database. If any of the fields need to be updated, update the necessary information and click the Submit button.

USING PubSearch TO MAKE GENE ONTOLOGY ANNOTATIONS

BASIC PROTOCOL 5

This section describes how to use PubSearch to make associations between genes and Gene Ontology terms using the Web browser. The intended users for this section are database curators. This protocol uses the demo version URL. For one's own version of PubSearch, the base URL will be different. By default, it takes the form: `http://${DOMAINNAME}:8080/PubSearch`.

Users can login from any page in PubSearch (see Basic Protocol 3). Logging in is required for making or updating GO annotations.

Necessary Resources

Hardware

Computer with Internet access

Software

Up-to-date browser such as Netscape 6.X, Internet Explorer 5.X, Safari 1.X

Select the gene to be annotated

1. Go to Gene search page either by clicking on the Genes hyperlink in the "Search for" page header or by going to the URL *http://${DOMAINNAME}:8080/pubsearch/Search?middle_page=genes_exp*.

2. Enter the name of the gene to be annotated; for example, `HST`. Click Submit.

3. From the search results page, either click on the "annotate HST" link at the bottom of the gene entry or click on the gene name and then click the Add Annotations link on the Gene Detail page.

Select a GO term

4. Type in the term to be used in the input box below Term Name. For example, type in `kinase`.

5. Click the Term Search button. This brings up the Term Search page with `kinase` filled in for a "contains" search.

6. Restrict "term type" to the aspect of interested. For example, if doing a function annotation, select "only allow func" from the "Filter based on term type" drop-down menu. Click Submit.

7. From the search results page, select the term that looks the most appropriate for the particular annotation. One can click either on the "term name" or on the Ontology View link.

   Clicking on "term name" opens a "term detail page" with the definition of the term and the term's parentage. Clicking on the Ontology View link opens a term browser and allows one to traverse up and down the structured hierarchy of the GO. In either case, it is possible to click on the button with the GO id to be used for the annotation. Doing so will enter both the term name and the term id into the annotation window in the appropriate slots.

Select a relationship type

8. The relationship type clarifies the gene-to-term relationship. Select the appropriate one for the annotation. For example, when using a GO biological process term, a commonly used relationship type is "involved in."

Select an evidence code

9. From the Type drop-down menu, select the appropriate three-letter evidence code for the annotation.

Select an evidence description

10. Depending on the evidence code selected, a number of evidence descriptions will be displayed in the Description drop-down menu. Select the one that is most appropriate for the annotation.

Select a reference

11. Click on either Article, Communication, Analysis Reference, or Book, depending on what type of reference is appropriate. If Article, Analysis Reference, or Communication are chosen, click the Select button beside the appropriate reference to use it in the annotation. If Book is chosen, select the title of the book from the Book menu, highlight the chapter to use within the Book Chapter menu, and click the Select button.

Enter the completed annotation into the database

12. Click on the Update button. The annotation will appear in the list of completed associations with the user's name and the current date in the "Annotated by" and Date fields.

Updating existing annotations

13. It is possible to update the term name, relationship type, evidence code, or evidence description by making changes and then clicking the Update button. However, if changing from one reference type to another (i.e., Communication to Article), it is necessary to obsolete the old annotation by clicking on the Obsolete "Y" radio button and then creating the new annotation as described above.

Propagating annotations

14. The user may find it desirable to propagate an annotation that has been made for one gene to other genes discussed in the same paper. If the annotations will be identical, except for the gene being annotated, one can use the annotation propagation function that is built into PubSearch.

15. Start on the Gene Detail page of the gene that has the annotation(s) to be propagated.

16. Select the annotations to propagate from the Annotations band by ticking the boxes beside the annotations in question, then click the Propagate button at the top of the Annotations band.

17. Enter the gene name for propagation. If there are several genes with the same base name, for example, ABC1-10, select "contains," type in the base name `ABC`, and click on Search Gene. If there are several genes that do not share a base name, it is possible to upload a file with all of the gene names from the computer desktop using the Browse button.

18. From the search results page, select the genes to which the annotation(s) are to be propagated. One may get multiple matches to the query, so make sure that the correct genes are selected.

19. Click the Propagate button.

20. A list of annotations that will be propagated will come up. Confirm that this is the course of action to take. Click Propagate Annotations.

GENERATING AND LOADING InterProToGo ANNOTATIONS

BASIC PROTOCOL 6

A common source of associations between genes and GO terms is via InterPro protein mappings. This protocol describes how to generate GO annotations from InterPro mappings to proteins and load them to PubSearch database on the fly using a perl script.

Necessary Resources

Hardware

See Support Protocol 1

Software

PubSearch (see Support Protocol 1 and Support Protocol 2)

Files

To generate GO annotations from InterPro mappings to proteins, the authors of this unit use the Interpro2gene mapping file and Interpro2Go mapping file. The Interpro2gene mapping file is a two-column file in which the first column is the gene name and second column contains the InterPro Ids. This file is generated using `InterProScan.pl`, which can be downloaded from *http://www.ebi.ac.uk/interpro*.

The sample file for *Arabidopsis*, `INTERPRO.Arab_R5.txt`, is located in the `maint/interpro2goAnnotation under ${PUBHOME}` directory.

The Interpro2Go file contains a mapping between InterPro domains and corresponding GO terms. It is manually generated and maintained by InterPro curators and is available from the GO Website *http://www.geneontology.org/external2go/interpro2go*.

The Perl script used in this protocol will automatically retrieve this mapping file.

1. To generate and load GO annotations from InterPro annotations, run `addAnnotationFromInterproGo.pl`. This script retrieves the latest Interpro2Go mapping file from the GO Website, generates GO annotations from the Interpro2Go mapping file and the Interpro2Gene mapping file, and loads the converted GO annotations into a PubSearch database. To run the script, execute the following commands from the `${PUBHOME} directory`:

```
>cd maint/interpro2goAnnotation

>perl addAnnotationFromInterproGo.pl -D database_name
-U user_id
```

COMMENTARY

Background Information

The systematic review and curation of scientific literature to extract relevant information is a task faced by every researcher. Broad-based, systematic curation of literature has been transformed by the World Wide Web. Tasks that were impossible when journals and articles were only available in print are now routine. Online literature repositories like PubMed, online journals, and the move toward providing the full text of research articles online have all made the literature more accessible than ever before. However, with this increase in accessibility has come an increase in complexity that is familiar to any user of the World Wide Web. How do you effectively find the literature you are looking for out of the ever-increasing quantities of literature that are not relevant to the task at hand?

Model Organism databases such as TAIR and RGD rely on the primary research literature as one of the main sources of information about an organism's genes and proteins and their functional role in that organism. While a variety of other information is stored in these databases, genes remain the primary focus. PubSearch was developed with the goal of facilitating the curation of literature pertaining to the genes of *Arabidopsis thaliana*. Figure 9.7.10 shows how PubSearch is used at TAIR for facilitating literature curation. In step 1, articles from literature databases such as PubMed, gene data from TAIR, and ontologies from GO and PO (Plant Ontology, *http://www.plantontology.org*) are imported. Next, the articles are indexed with gene data and ontologies. After the automatic associations are made in the database, curators can access the data using a Web browser and perform a number of functions such as editing data, validating the associations between data objects and articles, making controlled vocabulary annotations, and adding missing information (step 3). Finally, curated data are exported to TAIR and other databases such as the GO database (step 4).

The literature identification strategy of PubSearch is to first collect a larger corpus of broadly relevant articles and then narrow it down using specific terms relevant to the task at hand. In the case of PubSearch at TAIR, all articles mentioning *Arabidopsis* published since the previous analysis are identified. These are then searched for "terms" (known keywords of interest such as *Arabidopsis* gene symbols, ontology terms, etc.) to identify papers that have a higher chance of being useful for the curation of data relevant to a gene. The process of screening the articles against the list of relevant terms generates "hits" between an article and a term. The hits are then validated by human review of the full abstract to determine if the article does indeed pertain to the expected gene and has potentially useful information about its function. The combination of automated term matching and the manual validation of the subsequent hits is a relatively quick way to screen large numbers of articles to identify those that should be read in full by a human curator.

PubSearch provides a variety of search interfaces to allow curators to retrieve genes,

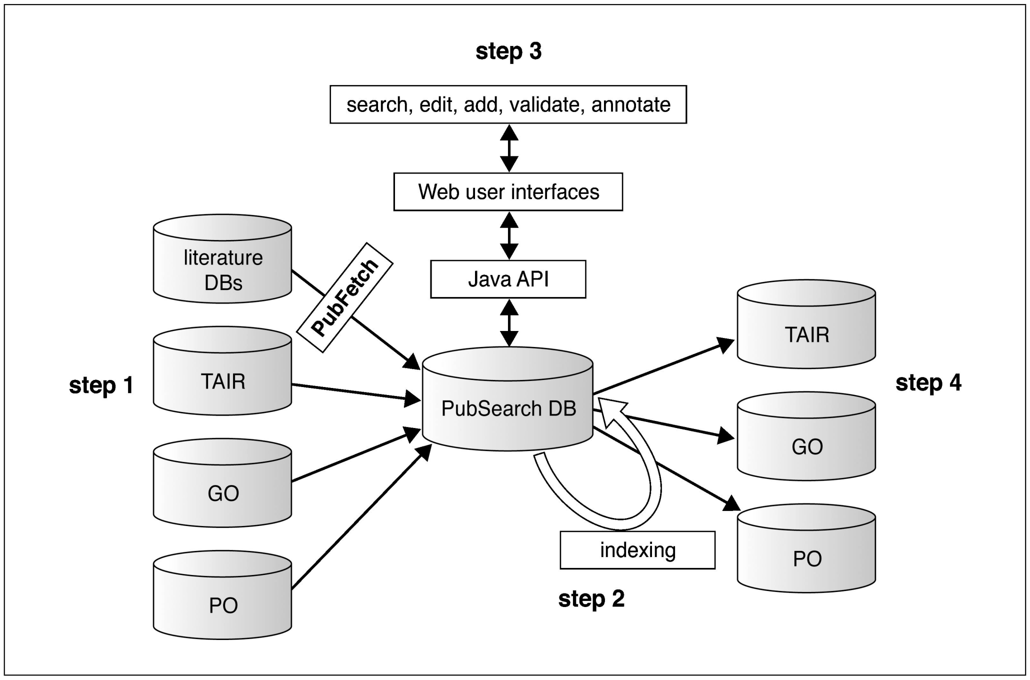


Figure 9.7.10 The PubSearch database is the central component of the PubSearch system. The following operations are performed during PubSearch use. In step 1, the PubSearch database is loaded in batch mode using input from other databases—e.g., articles from literature databases such as PubMed and Agricola using PubFetch software, biological data like gene, allele, and germplasm information from TAIR (an example model organism database), and ontologies from Ontology databases such as Gene Ontology and Plant Ontology. In step 2, the PubSearch database indexes the information by populating the Hit table using the Lucene engine. In step 3, through the Java API and a set of Web user interfaces, curators search, browse, edit, and add data, relying on the indexed data in the database. Finally, in step 4, the edited biological, literature, and annotation data are exported to the TAIR production database and other databases such as Gene Ontology and Plant Ontology.

keywords, articles, and matches between articles and terms using a variety of search parameters. Upon retrieval of relevant information, curators can annotate gene function, cellular location, expression patterns, genomic location, and other attributes by reading the matched articles. The Web user interface for editing annotations is designed to reduce free-text data entry, in order to increase the efficiency of annotation and reduce data-entry errors. The appropriate controlled vocabulary terms can be selected using an integrated ontology browser—a modified version of the AmiGO browser developed by the GO Consortium (The Gene Ontology Consortium 2001)—which allows interactive traversing of structured vocabularies and point-and-click selection of terms. The annotation interface facilitates data entry using pull-down menus or clickable lists that are generated on the fly with the appropriate data for the annotation task at hand. For example, the GO evidence codes (*http://www.geneontology.org/doc/GO.evidence.html*) and evidence descriptions, a controlled vocabulary of experiment types developed at the *Arabidopsis* Information Resource (Rhee et al., 2003), can be selected from pull-down menus.

PubSearch can be used as a stand-alone literature-management tool for biologists. In this case, all that is required in addition to periodic literature downloads using PubFetch is to update the gene and term lists to keep pace with modifications to the various ontologies and identification of new genes. Data import/export tools are provided to upload new vocabularies and updates to the Genes and to export the curated information for use in downstream applications or databases. By default, PubSearch is set up as an internal-use-only Web application and is thus password-protected; a login is required when starting a session. This authentication scheme would also work well for a group of investigators working on the annotation of a gene family or a microarray result set. The login also allows tracking of operations that a user has performed during a session, which can be used to verify consistency of annotation between users.

Alternative approaches

The PubSearch/PubFetch system is unique in that it provides a stand-alone, integrated literature-management and data-curation environment. Literature-curation software developed by other curation groups is typically tightly coupled to the computing environment of the particular curation group, making it difficult if not impossible for others to reuse the software components. As an alternative to a server-based approach, many researchers are familiar with desktop bibliography software such as EndNote and Reference Manager. These provide extensive literature retrieval and searching capabilities; however, they provide no capacity for curating information from this literature. In this scenario, data are often recorded in other desktop applications such as Microsoft Excel, which certainly has advantages for small-scale endeavors; however, this quickly becomes inconvenient for larger-scale annotation efforts. Another open-source text-mining project, Textpresso (Müller et al., 2004), is being developed as a component of the Generic Model Organism Database project. Texpresso uses customized biological concept ontologies to search the full text of an article, providing a sophisticated semantic search algorithm that goes beyond the existing PubSearch term-matching approach. Texpresso currently provides no data-curation functionality, so one might envisage integrating PubSearch with the Textpresso search engine to enable more precise categorization of the literature for subsequent curation.

Future directions

PubSearch is an open-source project, and, as such, all contributions by interested developers are most welcome. Below are listed a number of potential areas for extensions of the software that might provide ideas about how PubSearch could be used in the future.

PubSearch as a framework for the implementation of additional classification algorithms

PubSearch uses a robust but simple term-matching technique to identify relevant articles. This approach will miss relevant articles that do not contain these keywords. In addition to tools such as Texpresso, described above, other machine-learning algorithms exist that could be implemented within the PubSearch environment to identify articles in a more sophisticated fashion. The manual validation of articles in PubSearch divides them into those that have been shown to be relevant for further curation and those that are not relevant for further curation. One could use these datasets to train a Support Vector Machine to recapitulate this classification, with the expectation that it might work better at identifying articles that were lacking the exact keywords but nonetheless had overall content that indicated relevance. Even if these algorithms were not implemented inside PubSearch, the annotated literature corpus that is created through the use of PubSearch would be of great use to natural language processing researchers.

Expansion of PubSearch to allow curation of additional data beyond genes

PubSearch is currently gene-centric, allowing the curation of gene-related information from the literature. PubSearch could be expanded to allow the curation of data for other objects of interest, such as quantitative trait loci, subspecies of an organism (e.g., inbred rat strains, plant ecotypes), or other objects of interest. This would enable PubSearch to become a broader literature curation platform, allowing researchers to integrate a variety of data types with links to the literature, terms and vocabularies, and other data objects.

Further support for desktop applications such as EndNote

PubSearch provides the means to download articles from online literature databases and link them to genes and other biological terms. It would be convenient if relevant articles could be exported in a format compatible with EndNote or Reference Manager, so that articles stored in PubSearch could be easily used as citations in a manuscript.

Troubleshooting

Failure in database connection

If there is a failure in database connection to MySQL, there are two major possibilities: first that MySQL's network support has been turned off, and second that MySQL's permissions are too restrictive. In the first case, the MySQL configuration file `/etc/my.cnf` may contain the directive "skip networking." If this is the case, comment this directive out and restart MySQL. In the second case, the MySQL administrator must grant privileges to allow PubSearch to communicate with the database. The administrator may need to execute step 6 of Support Protocol 1 and inspect the `program.properties` file, to make sure that the granting SQL statement uses the same hostname as the configuration file.

Failure in unarchiving the distribution

If the `tar` utility fails with an error message about directory checksum errors, it is likely that the native `tar` utility on the system does not support long filenames. In this case, it is recommended that GNU `tar` be used to unpack the PubSearch distribution. GNU `tar` can be found online at *http://gnu.org/*.

Failure in logging in

If a user cannot log in, then it is possible that the user has not yet been added to the User table. To show a list of users on the system, execute:

```
>mysql pubdb
mysql> select * from pub_user
```

to verify that the user does exist in the PubSearch database. If not, then the user can be added by using the `bin/add_curator.pl` or `bin/add_user.pl` commands.

If the user does exist in the pub_user table, then it is likely that the password has been entered incorrectly and may need to be updated. There is no command-line utility to update a user's password, but the following SQL command will refresh the database:

```
mysql> update pub_user set
password=PASSWORD(“[pass-
word here]”) where name=
“[username]”
```

Acknowledgement

The development of PubSearch is supported in part by NHGRI grant number R01HG02728 (SYR, ST), NSF grant number DBI-9978564 (SYR), and NIH grant number HL64541(ST). The authors of this unit wish to thank Barbara Buchanan at NAL for compiling the *Arabidopsis* papers from Agricola and BIOSIS. The authors also thank Julie Tacklind for designing and maintaining the Web site, and are grateful to Suparna Mundodi, Leonore Reiser, Eva Huala, Margarita Garcia-Hernandez, Hartmut Foerster, Katica Ilic, Chris Tissier, Rachael Huntley, Nick Moseyko, and Peifen Zhang for their valuable input in improving the usability of the software. The authors are also grateful to the former members, Bengt Anell, Behzad Mahini, Victor Ruotti, and Lukas Mueller, who were involved in the project during its initial stages, and also thank Doug Becker, Dan MacLean, Chris Wilks, Jon Slenk, Susan Bloomberg for their careful reading of the manuscript.

Literature Cited

The Gene Ontology Consortium. 2001. Creating the gene ontology resource: Design and implementation. *Genome Res.* 11:1425-1433.

Müller, H., Kenny, E.E., and Sternberg, P.W. 2004. Textpresso: An ontology-based information retrieval and extraction system for biological literature. *PLoS Biol* 2:e309.

Rhee, S.Y., Beavis, W., Berardini, T.Z., Chen, G., Dixon, D., Doyle, A., Garcia-Hernandez, M., Huala, E., Lander, G., Montoya, M., Miller, N., Mueller, L.A., Mundodi, S., Reiser, L., Tacklind, J., Weems, D.C., Wu, Y., Xu, I., Yoo, D., Yoon, J., and Zhang, P. 2003. The Arabidopsis Information Resource (TAIR): A model organism database providing a centralized, curated gateway to Arabidopsis biology, research materials and community. *Nucl. Acids Res.* 31:224-228.

Internet Resources

http://sourceforge.net/projects/geneontology

Gene Ontology's SourceForge repository.

http://pubsearch.org

PubSearch homepage.

http://tesuque.stanford.edu:9999/pubdemo

PubSearch demo version.

http://lists.sourceforge.net/lists/listinfo/gmod-pubsearch-dv

PubSearch support mailing list.

http://www.gmod.org

Generic Model Organism Database project home page.

Contributed by Danny Yoo, Iris Xu, Tanya
Z. Berardini, and Seung Yon Rhee
Carnegie Institution
Stanford, California

Vijay Narayanasamy and Simon Twigger
Medical College of Wisconsin
Milwaukee, Wisconsin

Installing and Configuring CMap

Genomic data often comes in the form of maps of ordered markers. Maps run the gamut from genetic maps consisting of ordered genetic markers through physical maps of ordered clones and sequence-based maps of DNA base pairs. A species will often have multiple maps of different types and sometimes several of a single type: for example, the human genome has the subtly different Celera and public sector sequence assemblies. When performing positional cloning experiments or analyzing differences among closely related species, it is often critical to be able to align multiple maps together. The CMap application was designed to make this possible. It is a generic map alignment and visualization tool that works on all types of maps, whether they are genetic, physical, or sequence-based.

Since CMap is generic, it has many uses. It can be used to view synteny between species, map QTL data, and compare marker patterns. It can compare fingerprint-mapped clone (physical) contigs and sequence contigs for use as a quality control for the assemblies of each. CMap can be used to compare different versions of sequence assemblies to show how the assembly has changed. For example, Figure 9.8.1 shows a physical map compared to a genetic map in CMap.

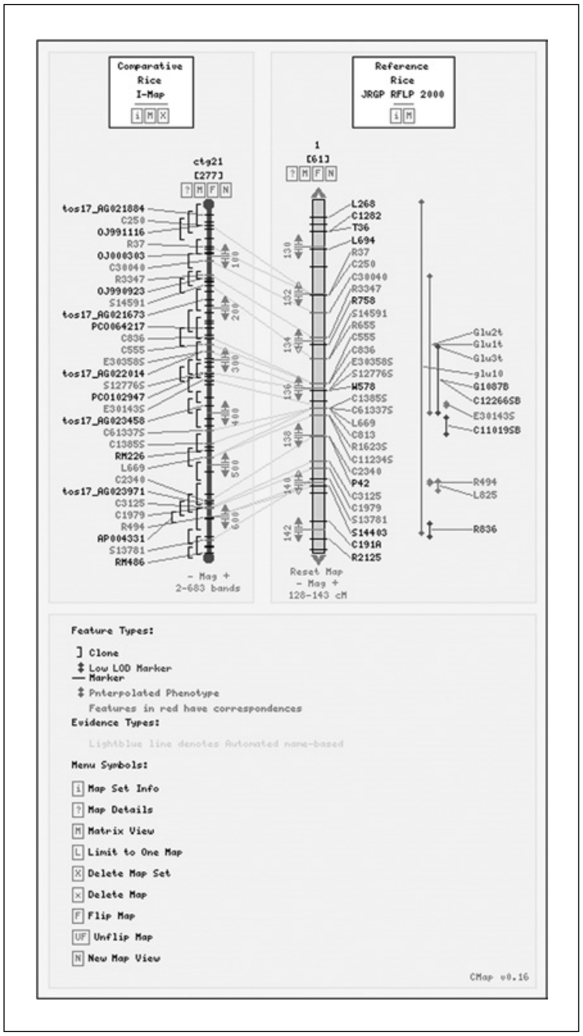


Figure 9.8.1 CMap image of comparison between a physical map and a genetic map. The lines between the maps are correspondences that point to the feature location on each map. The features in red are involved in the displayed correspondences. For the color version of this figure go to *http://www.currentprotocols.com*.

Contributed by Ben Faga

Current Protocols in Bioinformatics (2007) 9.8.1-9.8.30

CMap has a variety of concepts: maps, map sets, features, correspondences, and correspondence evidence. A map is any linearly ordered set of features, e.g., genomic assemblies, fingerprint contigs, QTL maps, and physical maps. Maps from one study are grouped into map sets. For instance, sequence contigs from a specific assembly version would go into one map set while fingerprint contigs would go into a separate map set. Features are annotations on the map, e.g., sequence reads, markers, or any other locatable annotations. Correspondences connect related features to establish links between maps. Each correspondence has one or more pieces of evidence that provide information about why the correspondence was created, e.g., matching names or BLAST similarity (see *UNIT 3.3*).

The overall goal of these protocols is to install and run CMap. Basic Protocol 1 gives an overview of the CMap user interface. Then there are protocols for creating a CMap database (Basic Protocol 2), configuring CMap (Basic Protocol 3), creating and importing data (Basic Protocol 4), and configuring CMap for speed and clarity (Basic Protocol 5). The Support Protocol directs installing CMap in a Linux environment.

BASIC PROTOCOL 1

GETTING STARTED WITH CMap

This protocol describes becoming familiar with the CMap interface. The Gramene Web site (*http://www.gramene.org*), a resource for comparative grass genomics, is used as an example. Since this is an active data repository, the actual data may change as well as the version of CMap used. The examples use the Gramene CMap present at the time of writing.

Necessary Resources

Hardware

Computer with Internet access

Software

JavaScript (*http://java.sun.com*) enabled up-to-date Internet browser, e.g., Internet Explorer (*http://www.microsoft.com/ie*); Netscape (*http://browser.netscape.com*); Firefox (*http://www.mozilla.org/firefox*); or Safari (*http://www.apple.com/safari*)

1. Go to the CMap start page for Gramene by either direct link (*http://www.gramene.org/cmap*) or by clicking on the MAPS link from the Gramene home page (*http://www.gramene.org*).

   Genomic resources other than Gramene also use CMap (see Commentary, Background Information).

2. At the starting page for CMap, choose one of the Maps links to go to the CMap viewer menu (*http://www.gramene.org/db/cmap/viewer*).

   There are a number of other ways to access data within CMap from the start page. These are described in the tutorial provided with the CMap distribution. There is also a Gramene-specific tutorial linked from this page on the Gramene site.

   The viewer menu is truncated until a species and map set are chosen.

3. Select Rice (*Oryza sativa*) from the Ref. Species drop-down box. Then click Change Species to load only Rice map sets into the Ref. Set drop-down box.

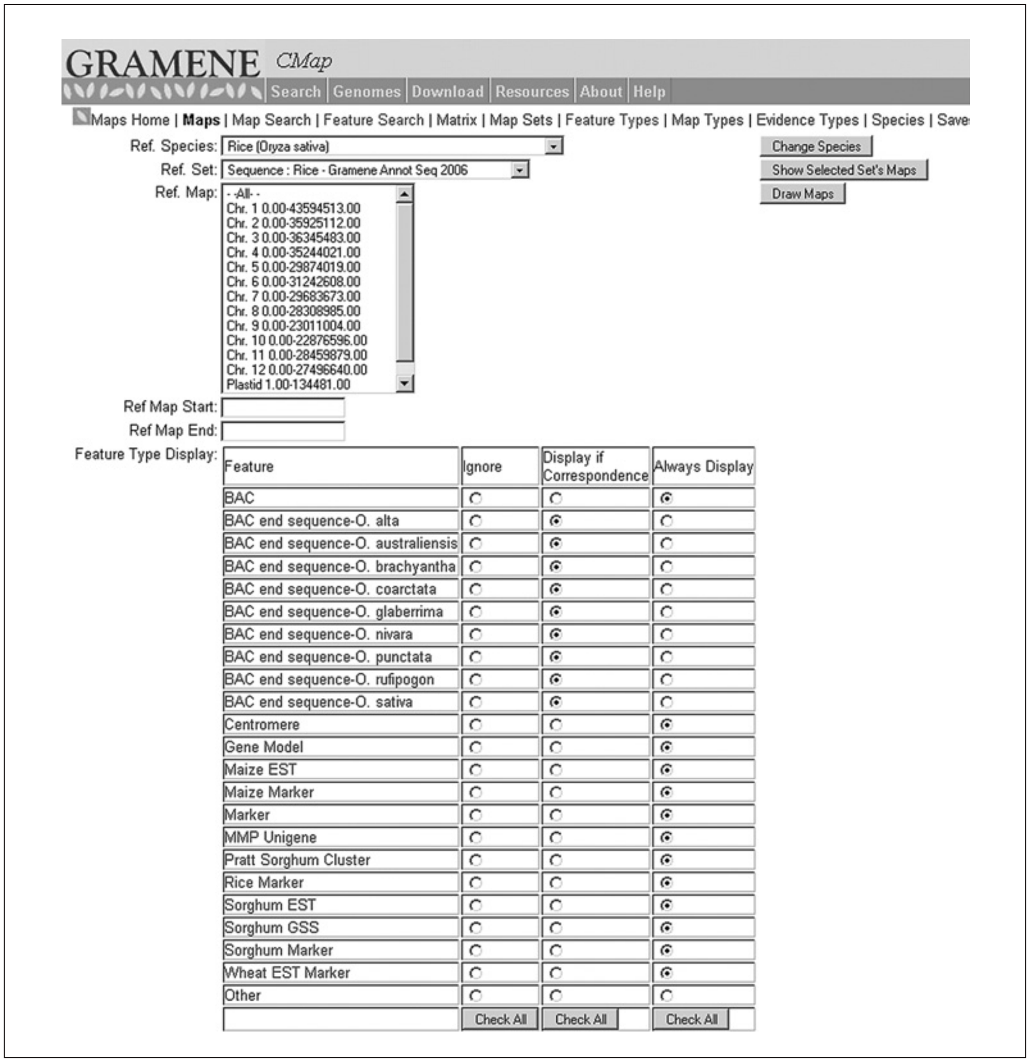
4. Select Sequence: Rice-Gramene Annot Seq 2006 from the Ref. Set drop-down box. Then click the Show Selected Set's Maps button. Figure 9.8.2 shows the view after these selections have been made.

Become familiar with the menu options that will have appeared (See Fig. 9.8.2). There will be a Ref. Map drop-down box with the maps in that set are listed with their start and stop values. Multiple maps can be selected in this box.

Be aware that when many maps are loaded the server must work harder to present the data.

There are Ref Map Start and Ref Map Stop boxes, which control the start and stop of the selected map when there is only one reference map selected.

Lastly, there is the Feature Type Display table. These settings dictate how features of each type are handled. Setting a feature type to Display if Correspondence will cause features of that type to only display if they are being used in a correspondence. This can improve performance greatly, but for this tutorial leave the defaults.


GRAMENE CMap
Search | Genomes | Download | Resources | About | Help
Maps Home | Maps | Map Search | Feature Search | Matrix | Map Sets | Feature Types | Map Types | Evidence Types | Species | Save
Change Species
Show Selected Set's Maps
Draw Maps

Ref. Species: Rice (Oryza sativa)

Ref. Set: Sequence : Rice - Gramene Annot Seq 2006

Ref. Map:
- -All- -
Chr. 1 0.00-43594513.00
Chr. 2 0.00-35925112.00
Chr. 3 0.00-36345483.00
Chr. 4 0.00-35244021.00
Chr. 5 0.00-29874019.00
Chr. 6 0.00-31242608.00
Chr. 7 0.00-29683673.00
Chr. 8 0.00-28308985.00
Chr. 9 0.00-23011004.00
Chr. 10 0.00-22876596.00
Chr. 11 0.00-28459879.00
Chr. 12 0.00-27496640.00
Plastid 1.00-134481.00

Ref Map Start:

Ref Map End:

Feature Type Display:

Feature	Ignore	Display if Correspondence	Always Display
BAC	○	○	●
BAC end sequence-O. alta	○	●	○
BAC end sequence-O. australiensis	○	●	○
BAC end sequence-O. brachyantha	○	●	○
BAC end sequence-O. coarctata	○	●	○
BAC end sequence-O. glaberrima	○	●	○
BAC end sequence-O. nivara	○	●	○
BAC end sequence-O. punctata	○	●	○
BAC end sequence-O. rufipogon	○	●	○
BAC end sequence-O. sativa	○	●	○
Centromere	○	○	●
Gene Model	○	○	●
Maize EST	○	○	●
Maize Marker	○	○	●
Marker	○	○	●
MMP Unigene	○	○	●
Pratt Sorghum Cluster	○	○	●
Rice Marker	○	○	●
Sorghum EST	○	○	●
Sorghum GSS	○	○	●
Sorghum Marker	○	○	●
Wheat EST Marker	○	○	●
Other	○	○	○
	Check All	Check All	Check All

Figure 9.8.2 The maps selection menu. Initially, only the Ref. Species and Ref. Set drop-down boxes are displayed. The remaining options appear when a reference set is selected. These options allow for a more refined view.

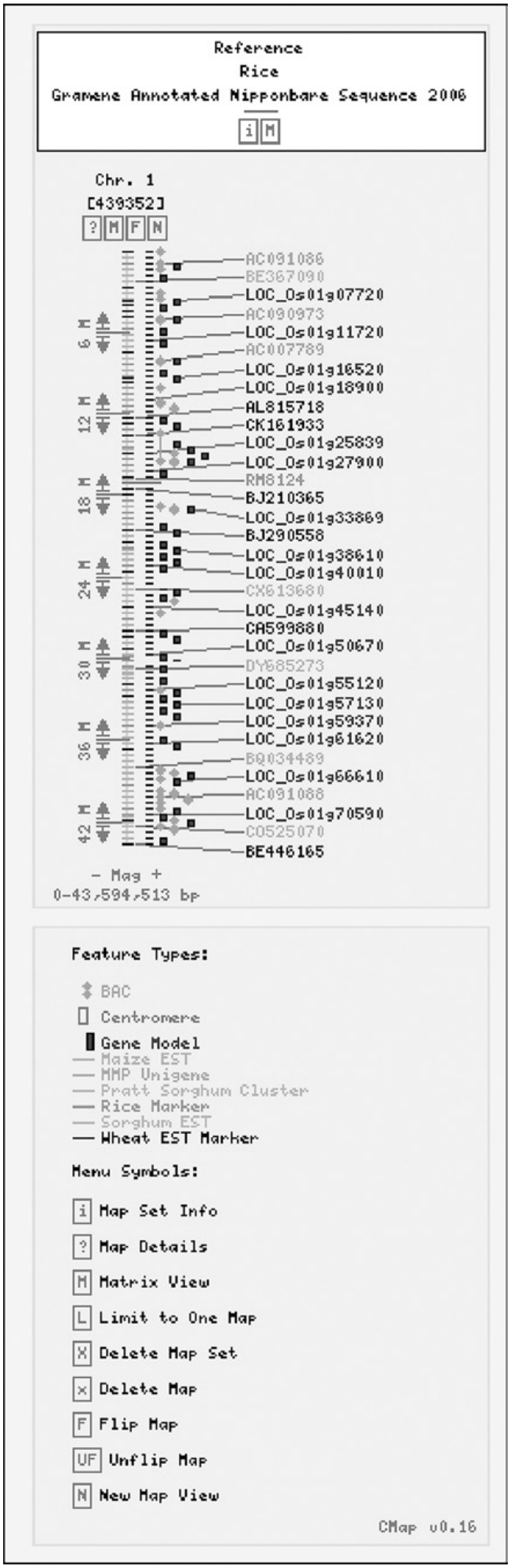
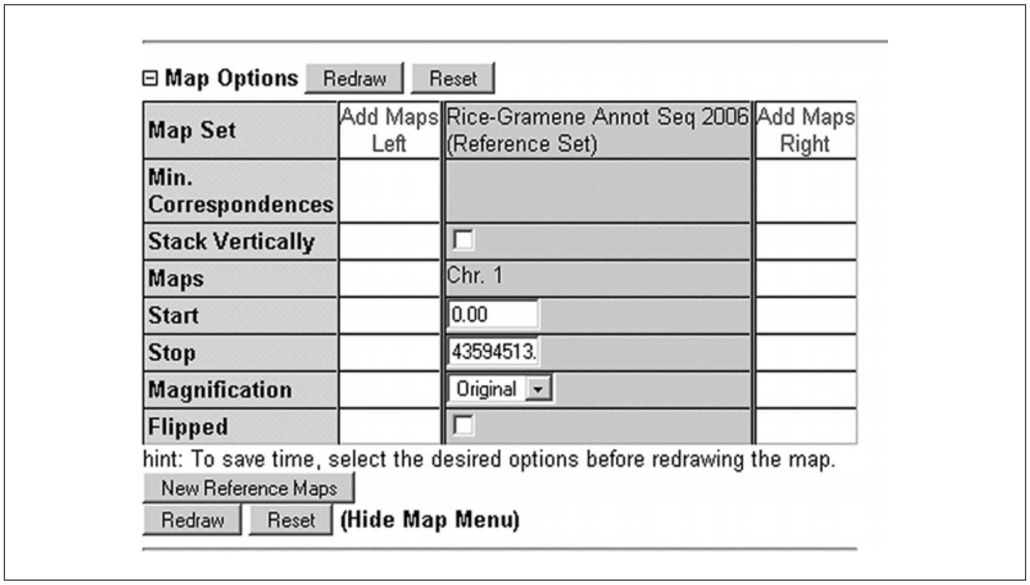


Figure 9.8.3 Legend at right.



⊟ **Map Options** Redraw Reset

Map Set	Add Maps Left	Rice-Gramene Annot Seq 2006 (Reference Set)	Add Maps Right
Min. Correspondences			
Stack Vertically		□	
Maps		Chr. 1	
Start		0.00	
Stop		43594513.	
Magnification		Original ▾	
Flipped		□	

hint: To save time, select the desired options before redrawing the map.

New Reference Maps

Redraw Reset **(Hide Map Menu)**

Figure 9.8.4 The Map Options menu with one reference map. Clicking the Add Maps Left and Add Maps Right links in the corners will bring up the comparative maps menu which allows for selection of the next corresponding maps.

5. Select Chr. 1 0.00-43594513.00 from the Ref. Map select box. Click on the Draw Maps button and wait for the image to return. Figure 9.8.3 shows the image that returns.

   Depending on server load and the state of the cache this may take a few seconds.

   Chromosome 1 is displayed as a vertical line with features attached. The features are drawn in different ways and colors depending on how the feature type is configured. For instance, a BAC is represented by a light blue, dumbbell shape whereas gene model is represented by a brown rectangle.

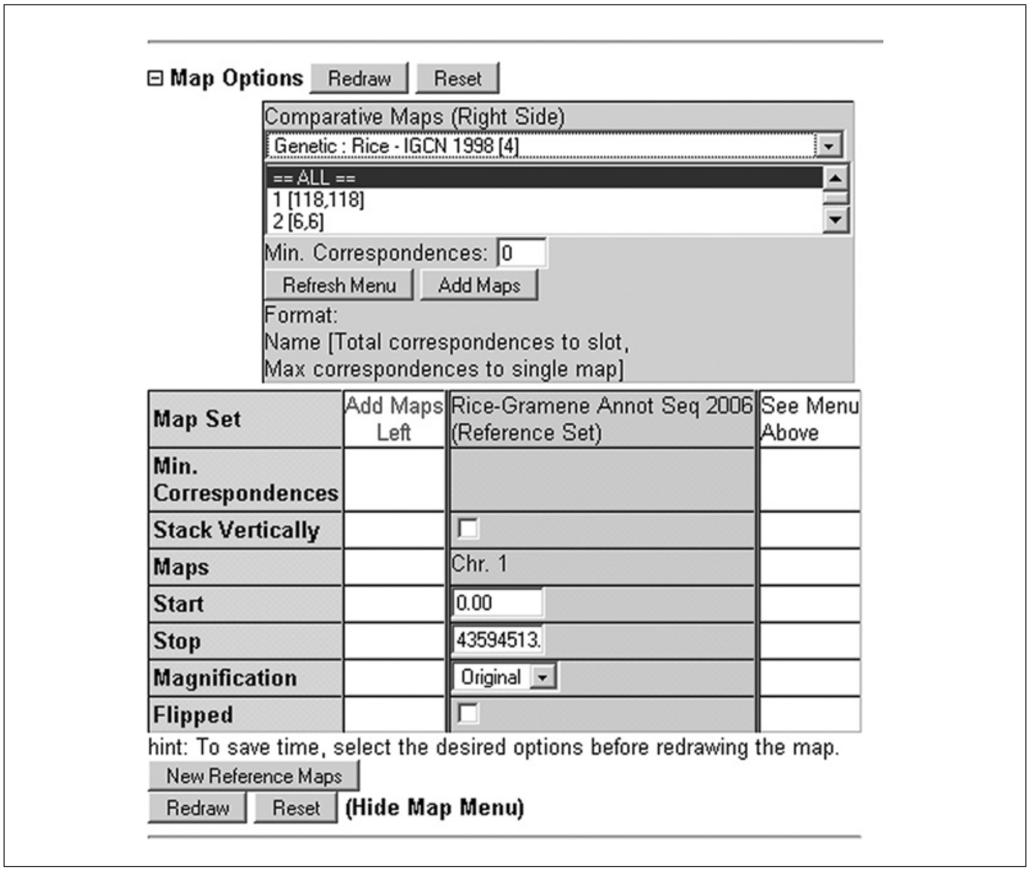
   There are also several navigation buttons on the image. The map-unit tick marks have arrows on each side. Clicking on one of the arrows will truncate the map at that point, zooming in on the side of the map that the arrow was pointing at. Above each map is a small menu which contains options such as flipping a map vertically or deleting it. The legend details what each option does. The maps and features are also links that default to the details page but can be configured to point elsewhere.

6. To add comparative maps to the image, click Add Maps Right in the Maps Menu (Fig. 9.8.4). This will bring up a small Comparative Maps menu (Fig. 9.8.5).

7. Select Genetic: Rice-IGCN 1998 [4] in the top drop-down box and 1 [118, 118] in the second. Click Add Maps. Figure 9.8.6 will appear showing the correspondences between the two maps.

   The crossing correspondence lines indicate that these maps are oriented opposite in respect to each other.

Figure 9.8.3 *(at left)* Chromosome 1 from a rice sequence assembly. The features are drawn, according to the feature type configuration, as variously colored glyphs to differentiate themselves from each other. Various navigation buttons are present, including truncation arrows and menu icons. The map and features are also links which can be configured. For the color version of this figure go to *http://www.currentprotocols.com*.



⊟ **Map Options** Redraw Reset

Comparative Maps (Right Side)

Genetic : Rice - IGCN 1998 [4]

== ALL ==
1 [118,118]
2 [6,6]

Min. Correspondences: 0

Refresh Menu Add Maps

Format:
Name [Total correspondences to slot,
Max correspondences to single map]

Map Set	Add Maps Left	Rice-Gramene Annot Seq 2006 (Reference Set)	See Menu Above
Min. Correspondences			
Stack Vertically		□	
Maps		Chr. 1	
Start		0.00	
Stop		43594513.	
Magnification		Original	
Flipped		□	

hint: To save time, select the desired options before redrawing the map.

New Reference Maps

Redraw Reset **(Hide Map Menu)**

Figure 9.8.5 The Comparative Maps menu. This menu is where the next corresponding maps are chosen. A minimum number of correspondences can be entered to limit the maps to those with a high number of correspondences.

8. Orient the maps with respect to one another by clicking the F button above the map, causing one chromosome to flip (Fig. 9.8.7).

   Figure 9.8.8 shows the Map Options menu as it is after this step. (The views shown in Fig. 9.8.7 and Fig. 9.8.8 are both produced by step 8.). This menu is represented as a table of all the maps in the image.

   The Min. Correspondences option will limit maps in a map set by removing maps that have fewer correspondences than the minimum number. This is useful for winnowing away maps that may not be as interesting.

   When multiple maps are used as reference maps, selecting the Stack Vertically option will cause the reference maps to be stacked on top of each other rather than side by side.

   The remaining options in this table are covered in the tutorial included with CMap.

9. Open up the Feature Options menu by clicking on Feature Options (Fig. 9.8.9). This menu contains the same Feature Type Display table that was described in the step 4 annotation.

   The Highlight Features box works by typing a feature name into the box.

   The Collapse Overlapping Features option dictates whether features that would be drawn exactly the same are to be collapsed into one glyph.

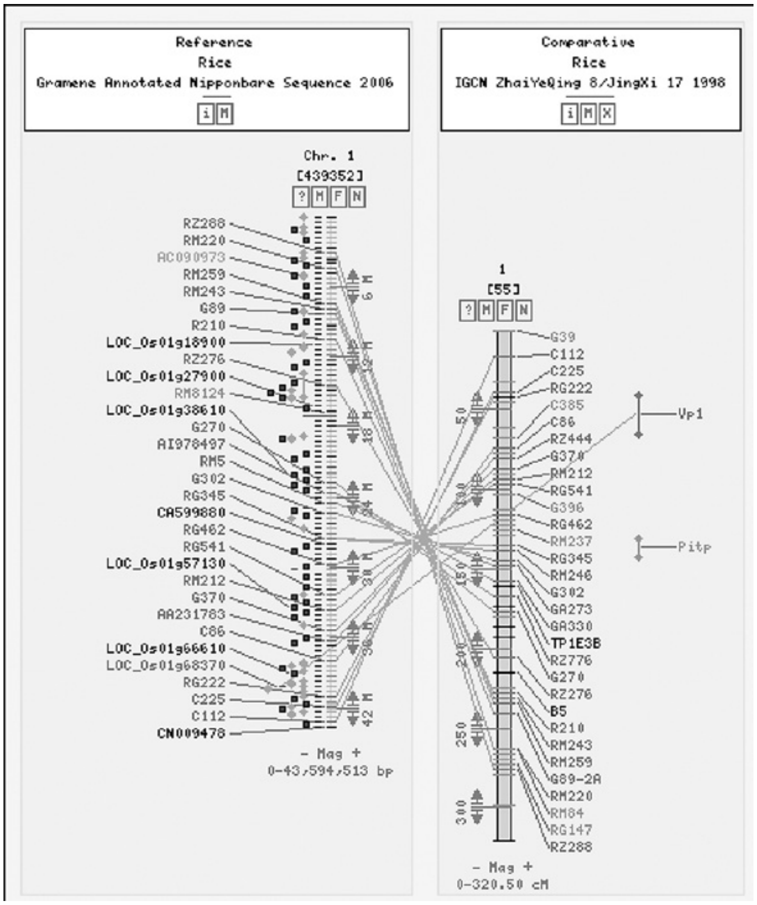


Figure 9.8.6 Chromosome 1 of a sequence assembly compared to chromosome 1 of a genetic map set. The gray lines represent correspondences between features on these maps. The crossing correspondences indicate that these maps are not oriented correctly. For the color version of this figure go to *http://www.currentprotocols.com*.

10. Open the Correspondence Options menu. This menu has a table for selecting how each correspondence evidence is handled—ignored, used, or compared to a score.

    The most used option in this menu is Aggregate Correspondences. When set to one line or two lines and when CMap is displaying more than one map from a set, CMap will aggregate the correspondences between maps as one line or two lines (representing the span of features on each map). For more information about correspondence aggregation, see Basic Protocol 5.

    The other menus, Display Options and Advanced Options, hold less used options. For more information about these, see the tutorial included with the CMap distribution.

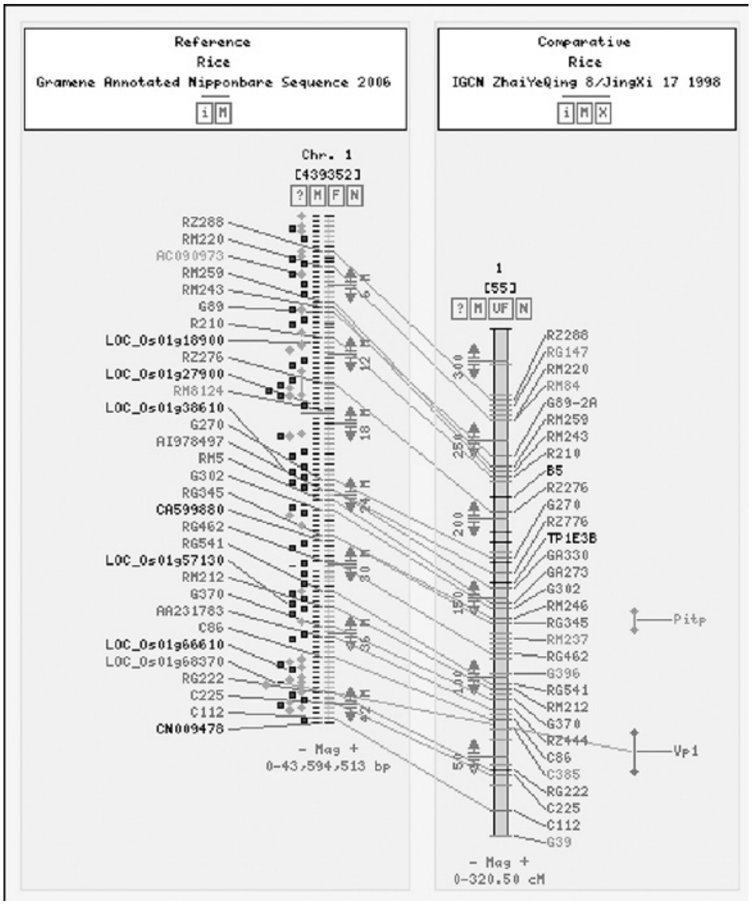
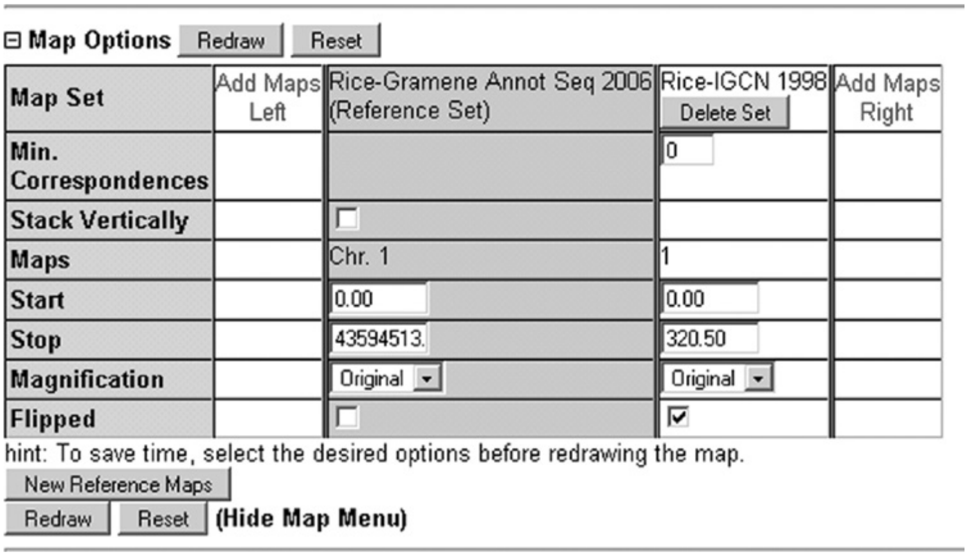


Figure 9.8.7 Maps from Figure 9.8.6 with the genetic map flipped. Note the F (flip) icon above the genetic map is now UF (un-flip). The correspondences are less tangled, and the image is more clear. For the color version of this figure go to *http://www.currentprotocols.com*.



⊟ **Map Options** Redraw Reset

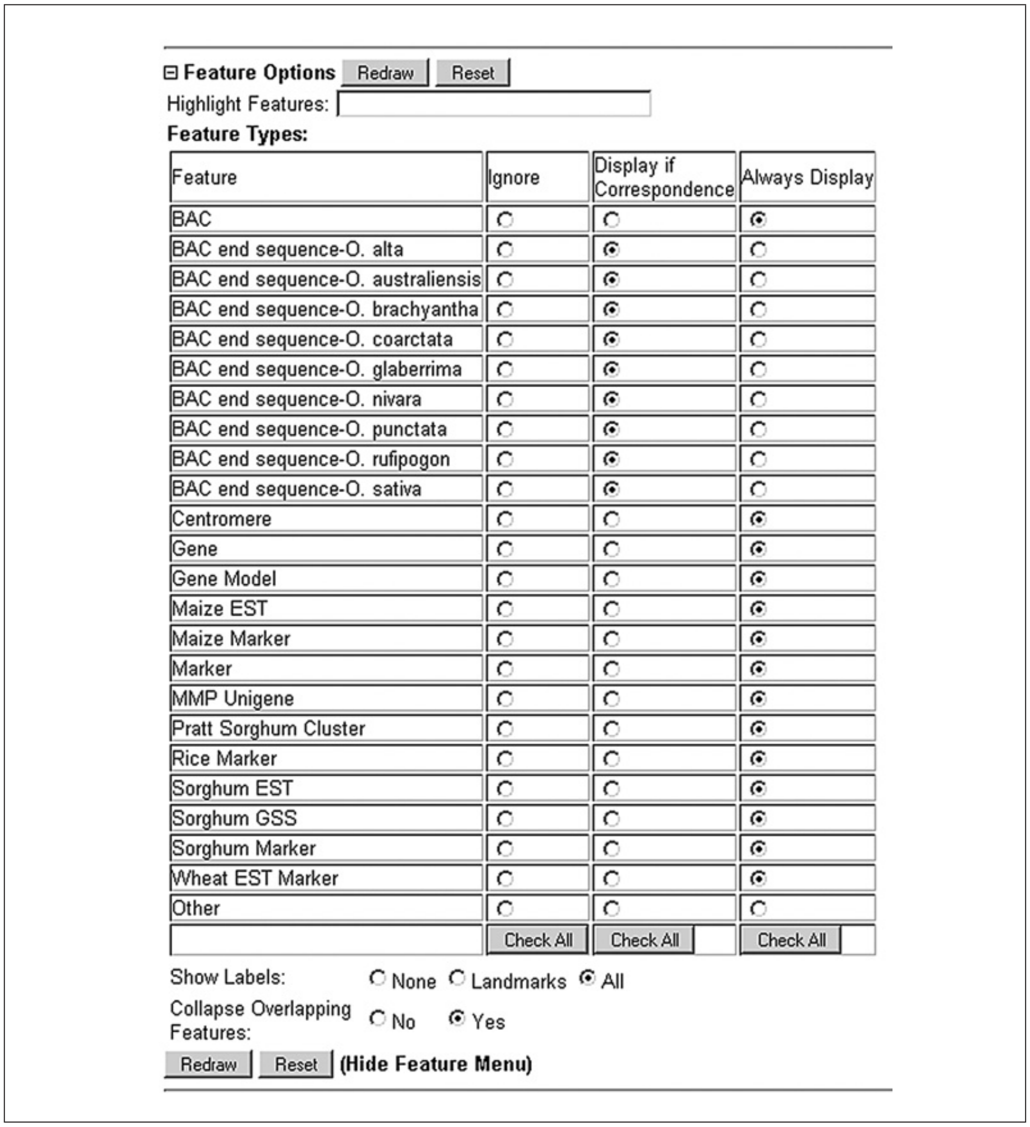
Map Set	Add Maps Left	Rice-Gramene Annot Seq 2006 (Reference Set)	Rice-IGCN 1998 Delete Set	Add Maps Right
Min. Correspondences			0	
Stack Vertically		□		
Maps		Chr. 1	1	
Start		0.00	0.00	
Stop		43594513.	320.50	
Magnification		Original ▾	Original ▾	
Flipped		□	☑	

hint: To save time, select the desired options before redrawing the map.

New Reference Maps

Redraw Reset **(Hide Map Menu)**

Figure 9.8.8 The maps selection menu after adding a correspondence map. The reference map options are in blue while the other map options are in white. As more corresponding maps are added, this table will reflect that growth.



⊟ **Feature Options** Redraw Reset

Highlight Features:

Feature Types:

Feature	Ignore	Display if Correspondence	Always Display
BAC	○	○	●
BAC end sequence-O. alta	○	●	○
BAC end sequence-O. australiensis	○	●	○
BAC end sequence-O. brachyantha	○	●	○
BAC end sequence-O. coarctata	○	●	○
BAC end sequence-O. glaberrima	○	●	○
BAC end sequence-O. nivara	○	●	○
BAC end sequence-O. punctata	○	●	○
BAC end sequence-O. rufipogon	○	●	○
BAC end sequence-O. sativa	○	●	○
Centromere	○	○	●
Gene	○	○	●
Gene Model	○	○	●
Maize EST	○	○	●
Maize Marker	○	○	●
Marker	○	○	●
MMP Unigene	○	○	●
Pratt Sorghum Cluster	○	○	●
Rice Marker	○	○	●
Sorghum EST	○	○	●
Sorghum GSS	○	○	●
Sorghum Marker	○	○	●
Wheat EST Marker	○	○	●
Other	○	○	○
	Check All	Check All	Check All

Show Labels: ○ None ○ Landmarks ● All

Collapse Overlapping Features: ○ No ● Yes

Redraw Reset **(Hide Feature Menu)**

Figure 9.8.9 The Feature Options menu. This menu allows for individual feature types to be treated differently. Some can be always displayed, while others can be only displayed if they are being used in a correspondence and still others can be ignored completely. Highlight Features will highlight the labels of the features with names written in the box. The options Show Labels and Collapse Overlapping Features affect how the image is drawn.

CREATING THE CMap DATABASE

BASIC PROTOCOL 2

This protocol describes setting up the CMap database. CMap stores data in a specialized schema which needs to be set up before running any CMap scripts. Since CMap strives to be database independent, it should not matter which relational database management system (RDBMS) is used. The following examples will be demonstrated with MySQL 5.0 commands but other RDBMSs will work.

Necessary Resources

Hardware

Computer with Internet access

Software

An installation of CMap (Support Protocol)

Files

Files provided with the CMap distribution

Create CMap database

1. Create a database for use by CMap by typing the following command.

```
$ mysql -uroot -p -e ‘create database CMAP’
```

The database name does not matter. For this example, it will simply be called "CMAP".

2. Install the CMap schema using the schema file in the `sql/` directory that corresponds with the RDBMS being used (e.g., MySQL uses `sql/cmap.create.mysql`).

```
$ mysql -uroot -p CMAP < sql/cmap.create.mysql
```

There are schema files provided for five different RDBMSs: MySQL, PostgreSQL, Oracle, Sybase and SQLite.

The schema files create the database tables but do not populate them with data.

3. Create a user account and give it permission to read and modify the CMap database.

```
$ mysql -uroot -p CMAP -e 'grant select, insert, up-
date, delete on CMAP.* to mysql@localhost identified by
""'
```

4. Flush the privilege table to allow the new user permissions to take effect.

```
$ mysqladmin -uroot -p flush-privileges
```

NOTE: *It is reasonable to create two database users, one for the Web page that can only read the database and one for administration that can read and modify the data. Each will need its own individual configuration file (described in the next section).*

BASIC PROTOCOL 3

CONFIGURING CMap

CMap comes with example configuration files. This protocol describes how to modify these files to get CMap running and provides descriptions of some of the useful basic configuration options. Please note that if the configuration files are changed after CMap is in use, the query cache will need to be purged using the `cmap_admin.pl` tool (see Basic Protocol 5, step 8).

Necessary Resources

Hardware

Computer with Internet access

Software

JavaScript (*http://java.sun.com*) enabled up-to-date Internet browser, e.g., Internet Explorer (*http://www.microsoft.com/ie*); Netscape (*http://browser.netscape.com*); Firefox (*http://www.mozilla.org/firefox*); or Safari (*http://www.apple.com/safari*)
An installation of CMap (Support Protocol)
Text editor (e.g., vi)

Files

Files provided with the CMap distribution

Configure an individual data source configuration file

1. Move to the `cmap.conf/` directory in the `web servers conf/` directory and copy `example.conf` to a new file.

```
$ cd /usr/local/apache/conf/cmap.conf/
$ sudo cp example.conf cmap_live.conf
```

   The new file can have any name with the `.conf` *extension. For this example,* `cmap_live.conf` *will be used.*

2. Open `cmap_live.conf` with a text editor such as "vi" (this might require super user privileges).

```
$ sudo vi cmap_live.conf
```

3. Modify the `<database>` option as indicated for the unmodified commands.

```
<database>
name EXAMPLE_CMAP
```

   Change the `name` *field to a unique name that will identify this data source, for this example,* `CMap_Live`. *It must be unique across the whole server even if there are multiple installations of CMap.*

```
datasource dbi:mysql:EXAMPLE_CMAP
```

   Change the `datasource` *to* `dbi:mysql:CMAP`. *The* `datasource` *contains the DSN value for connecting to the database. The value for MySQL is of the form* `dbi:mysql:database_name`. *The database from the previous section was named* `CMAP`, *so the* `datasource` *will be* `dbi:mysql:CMAP`. *For the format of other database systems, view the documentation for the* `DBD:: modules` *(e.g.,* `perldoc DBD::mysql` *or* `perldoc DBD::Pg`*).*

```
user mysql
password
```

   Modify the `user` *and* `password` *fields to reflect the user information created for the database.*

```
</database>
```

4. The modified version of the `<database>` option becomes the following.

```
<database>
name CMap_Live
datasource dbi:mysql:CMAP
user mysql
password </database>
```

5. Go farther down in the file and change the `is_enabled` value from `0` to `1`.

   When `is_enabled` *is set to* `0`, *the configuration file is ignored. The* `example.conf` *file is not enabled by default in order to keep it from unintentionally disrupting CMap during an upgrade.*

6. Configure map, feature, and evidence types for each type of object that will be inserted into CMap. Replace configurable `*` values with the words `map`, `feature`, or `evidence` and words in all capital letters with the appropriate wording in the following format.

```
<*_type ACCESSION>
*_type_acc ACCESSION
*_type NAME_OF_TYPE
</*_type>
```

Note that due to technical reasons, the accession value must be declared both in the initial tag and as its own field.

Types for maps, features, and correspondence evidence must be defined in the configuration file before objects of that type can be entered into the CMap database. For CMap to function correctly, it is very important for the CMap objects to have their types defined in the configuration file. The types hold mostly drawing information. The example configuration file has several predefined types. These are not required types and can be modified or copied.

To get started, each type requires an accession and a name. The accession is a unique identifier for that feature, map, or evidence type. It is used internally and never seen. It must be unique and contain only numbers and letters. For more information, see Critical Parameters and Troubleshooting. Additionally, map_types *need a* map_units *value which defines the unit name such as* bp, cM *or* bands. *All other options have reasonable defaults and can be left out without breaking CMap. For details about the other options, please see the administration document included with the CMap distribution.*

7. The following is a sample map_type definition from the example.conf file.

```
<map_type genetic>
map_type_acc genetic
map_type Genetic
map_units cM
is_relational_map
width 1
shape box
color
</map_type>
```

In this example, the map_type *accession is* genetic *and is defined in the opening tag and again as the* map_type_acc *option.*

The map_type *of* Genetic *is the value displayed on the Web pages when referring to this* map_type.

The units that maps of this type have are centimorgans (cM).

The field is_relational_map *is left blank and will be filled in with the default. When set to* 1*, the* is_relational_map *tells CMap that maps of this type cannot be used as a starting reference map but can only be used as correspondence maps.*

The remainder of the options deal with the drawing of maps. The genetic maps defined by this map_type *will have a width of 1 pixel and be shaped like a box. The field* color *is left blank and will be filled in with the default color (gray).*

This configuration is reflected by the genetic map on the right side of Figure 9.8.1 of the introduction. These and other options are covered in more detail by the administration document included with the CMap distribution.

One last important point is that the feature and evidence types are continually used by CMap. If a feature_type color *is changed, that change will be reflected immediately after the query cache is purged (see Basic Protocol 5, step 8). By contrast, the* map_type *is mostly used during map set creation. The values such as color and width are stored in the database. This allows the protocol user to change the options for individual map sets of the same type. Thus changing the color of a* map_type *will have no effect on the color of map sets that were created before the change.*

8. Validate the data source configuration after modification by running bin/cmap_validate_config.pl on it. Call the script with the configuration file as follows.

```
$ bin/cmap_validate_config.pl/usr/local/apache/conf/cmap.
conf/cmap_live.conf
```

This will check the syntax, check for required options, check for deprecated options, and give a list of any optional options that have not been specified.

If the data source configuration is valid, the script will print out a list of missing optional entries and a statement of validity as follows.

```
The config file, /usr/local/apache/conf/cmap.conf/cmap_
live.conf is valid
```

If the file is invalid, the script will declare the configuration file invalid and explain where the problem is.

```
The config file,/usr/local/apache/conf/cmap.conf/cmap_
live.conf is INVALID
```

Configure global configuration file

9. Set the default data source (the `default_db` option) to the name of the data source that should be used when no data source is given, to read in the above example, `default_db CMap_Live`.

   NOTE: *The required configuration options in the* `global.conf` *file will have been prepopulated based on the installation settings.*

   For additional useful configuration settings see Basic Protocol 5.

CREATING AND IMPORTING DATA

BASIC PROTOCOL 4

This protocol describes creating and importing data for a CMap database. There are multiple ways to affect a CMap database. This protocol will focus on the most useful CMap administration tool, the `cmap_admin.pl` script.

The `cmap_admin.pl` script will be installed in the execution path. It can be controlled either by command line or through a menu system within the script. Each step in this protocol will describe the steps through the menu system and then provide the command line options to do the same thing. To get a complete listing of the command line options, execute `perldoc cmap_admin.pl`.

Necessary Resources

Hardware

Computer with Internet access

Software

JavaScript (*http://java.sun.com*) enabled up-to-date Internet browser, e.g., Internet Explorer (*http://www.microsoft.com/ie*); Netscape (*http://browser.netscape.com*); Firefox (*http://www.mozilla.org/firefox*); or Safari (*http://www.apple.com/safari*)

An installation of CMap (Support Protocol)

Files

CMap sample import files: `tabtest1` and `tabtest2` (included with distribution)

1. Run the `cmap_admin.pl` script.

   ```
   $ cmap_admin.pl
   ```

   This following menu will appear.

```
Current data source: CMap_Live
-=-=-=-=-=-=-=-=-=-=
--= Main Menu =--
-=-=-=-=-=-=-=-=-=-=
[1] Change current data source
[2] Create new species
[3] Create new map set
[4] Import data
[5] Export data
[6] Delete data
[7] Make name-based correspondences
[8] Delete duplicate correspondences
[9] Reload correspondence matrix
[10] Purge the cache to view new data
[11] Import links
[12] Quit
What would you like to do? (one choice only):
```

Select an option by typing the number in the last line and hitting enter.

2a. *To change the data source:* Select `[1] Change current data source`. Then select the correct data source from the list.

The `Current data source` *is initially set to the default data source.*

2b. *To set the data source from the command line:* Use the `-d` flag.

```
cmap_admin.pl -d CMap_Live.
```

Assuming no other flags are set, the data source flag will still enter the menu system but using the selected data source.

Create a new species

3a. *To create a new species from the main menu:* Select `[2] Create new species.` and respond as indicated to the following prompts:

```
Full Species Name (long) Enter a long form of the
species name (e.g., ``Mus musculus'').
```

```
Common Name
```

Enter the common form of the species name (e.g., `` ``Mouse'' `` *).*

```
Accession ID
```

Enter an optional, unique accession id (e.g., `` ``m_musculus'' `` *). If not defined, the accession id will be assigned a numeric value.*

```
OK to create species ``Mus musculus'' in data source
`CMap_Live'?.
```

To confirm the species creation, say `yes`, and the species will be created.

3b. *To confirm species creation from the command line:* Type the following.

```
$ cmap_admin.pl -d CMap_Live --action create_species --
species_full_name ``Mus musculus'' --species_common_name
``Mouse'' --species_acc ``m_musculus''
```

Create a map set

4a. *To create a map set from the main menu:* Select `[3] Create new map set` and respond as indicated to the following prompts:

```
What species?
```

Choose from a list of previously created species.

```
What type of map?
```

Choose from a list of `map_types` *in the database (e.g.,* `Sequence`*).*

```
Map Study Name (long)
```

Enter the long form of the map set name (e.g., ``` ``Pretend Mouse Sequence Assembly 2006'' ```*).*

```
Short Name
```

Enter the short form of the map set name (e.g., ``` ``Mouse Seq Assm 2006'' ```*).*

```
Accession ID
```

Enter an optional, unique accession id (e.g., ``` ``PMSA2006'' ```*). If not defined, the accession id will be assigned a numeric value.*

```
What color should this map set be?
```

Enter the color to draw maps from this set (e.g., `black`*). A list of available colors is in the* `Bio/GMOD/CMap/Constants.pm module`*.*

```
What shape should this map set be?
```

Enter the shape to draw maps from this set. Choose from `box`, `dumbbell` *or* `I-beam`*.*

```
What width should this map set be?
```

Enter the number of pixels wide the maps should be. A good starting value is 4.

A map set is any collection of maps that are related, have the same `map_type` *and are from the same species. For instance the contigs of a specific version of sequence assembly would be in the same map set.*

```
OK to create set 'Pretend Mouse Sequence Assembly 2006'
in data source 'CMap_Live'?
```

Indicate `yes`*, and the map set will be created.*

4b. *To confirm the map set creation from the command line:* Type the following.

```
$ cmap_admin.pl -d CMap_Live --action create_map_set
--map_set_name ``Pretend Mouse Sequence Assembly
2006'' --species_acc m_musculus --map_type_acc Seq --
map_set_short_name ``Mouse Seq Assm 2006'' --map_set_acc
PMSA2006 --map_shape box --map_color black --map_width 4
```

Create a CMap import file

5. Create the following required fields with a tab delimited list of the following field headers: `map_name`, `feature_name`, `feature_start`, `feature_stop`, and `feature_type_acc`.

Each line of tab delimited data represents information about a single feature corresponding to the field headers. While the creation of these import files is left to the user, there are a few scripts included with the distribution to help; `cmap_parseagp.pl` *for GenBank assembly files in AGP format (http://www.ncbi.nlm.nih.gov/Genbank/WGS.agpformat.html),* `cmap_parsefpc.pl` *for files exported from the fingerprint contig program, FPC (http://www.agcol.arizona.edu/software/fpc) and* `cmap_parseWashUAceFiles.pl` *which parses Ace files output from the sequence assembly program, Phrap (UNIT 11.4; http://www.phrap.org). Sample import files are also included in the distribution (*`data/tabtest1` *and* `data/tabtest2`*).*

6. *Optional:* Create other useful fields including the following.

`map_acc/feature_acc`

Accession IDs for the maps and features. If not defined, the accession id will be assigned a numeric value.

`map_start/map_stop`

If these are not provide, the map start and stop are set by the span of the features.

`feature_direction`

The feature's direction in relation to the map. `−1` *or* `1`.

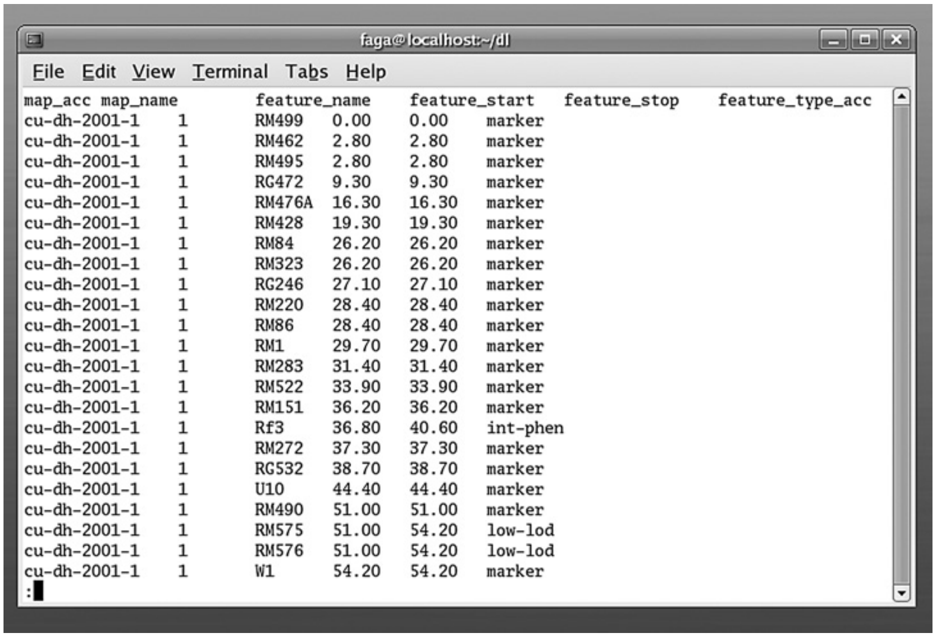
`feature_aliases`

Alternate feature names separated by a comma.

`feature_attributes`

A list of feature attributes separated by a semicolon. The attributes are in `name:value` *format.*

Figure 9.8.10 shows an example tab delimited import file with a subset of the possible columns.



```
map_acc map_name        feature_name    feature_start   feature_stop    feature_type_acc
cu-dh-2001-1    1       RM499   0.00    0.00    marker
cu-dh-2001-1    1       RM462   2.80    2.80    marker
cu-dh-2001-1    1       RM495   2.80    2.80    marker
cu-dh-2001-1    1       RG472   9.30    9.30    marker
cu-dh-2001-1    1       RM476A  16.30   16.30   marker
cu-dh-2001-1    1       RM428   19.30   19.30   marker
cu-dh-2001-1    1       RM84    26.20   26.20   marker
cu-dh-2001-1    1       RM323   26.20   26.20   marker
cu-dh-2001-1    1       RG246   27.10   27.10   marker
cu-dh-2001-1    1       RM220   28.40   28.40   marker
cu-dh-2001-1    1       RM86    28.40   28.40   marker
cu-dh-2001-1    1       RM1     29.70   29.70   marker
cu-dh-2001-1    1       RM283   31.40   31.40   marker
cu-dh-2001-1    1       RM522   33.90   33.90   marker
cu-dh-2001-1    1       RM151   36.20   36.20   marker
cu-dh-2001-1    1       Rf3     36.80   40.60   int-phen
cu-dh-2001-1    1       RM272   37.30   37.30   marker
cu-dh-2001-1    1       RG532   38.70   38.70   marker
cu-dh-2001-1    1       U10     44.40   44.40   marker
cu-dh-2001-1    1       RM490   51.00   51.00   marker
cu-dh-2001-1    1       RM575   51.00   54.20   low-lod
cu-dh-2001-1    1       RM576   51.00   54.20   low-lod
cu-dh-2001-1    1       W1      54.20   54.20   marker
:
```

Figure 9.8.10 Tab delimited files used to import data into CMap. The first line in the file is the column headers. For an exhaustive list of possible fields, execute `perldoc Bio::GMOD::CMap::Admin::Import`.

7. Validate the import file using the `validate_import_file.pl script` by entering the following.

```
$ validate_import_file.pl -d CMap_Live -f import_file.cmap
```

The program will report any problems with the file.

8a. *To import the tab delimited file from step 5 into CMap*: Use `cmap_admin.pl` to select from the main menu `cmap_admin` the option `[4] Import data`. Then select `[1] Import tab-delimited data for existing map set` and respond to the following prompts.

```
Please specify the files?
```

Enter the location of the import file(s).

```
How would you like to select map sets?
```

Choose whether to use a menu system to pick the map set to import data into or to simply supply it. If the map set accession id is known, simply choose Supply Map Set Accession ID and enter the accession. Otherwise, a menu system will guide the map set selection process.

```
Remove data in map set not in import file?
```

In most cases, the answer should be no but if only data in the import files should be in the database, answer `yes`. *Using this option slows down the import process so it is recommended to just delete the data from the database first and then do a fresh install.*

```
Check for duplicate data (slow)?
```

When doing incremental imports, CMap can check for features and maps already in the database and update where needed. This can slow the process. If no duplicate data is going to be imported, answer no and all of the data in the import file will be inserted.

```
OK to import?
```

Indicate `yes` *and the data will be imported.*

Importing a tab delimited file will create the maps and features that are described.

8b. *To confirm data importation from the command line:* type the following.

```
$ cmap_admin.pl -d CMap_Live --action import_tab_data
--map_set_acc PMSA2006 data/tabtest1 data/tabtest2
```

9a. *To create correspondences between features:* Select from the main menu `[7] Make name-based correspondences` and respond to the following prompts.

```
First you will select the starting map sets
```

A map set accession id can be supplied or the menu system can be used to choose map sets.

```
Use the starting map sets as the target sets?
```

Choose if the starting group should also be the target group. If not, choose the target map sets.

It is important to note that when creating name based correspondences, there are two groups of map sets, the starting sets and the target sets. Each of the starting sets will have correspondences made with each of the target sets. This prevents the map sets in the starting group from creating correspondences between each other. A map set can be in both the starting and target groups if correspondences between the maps in that set are desired.

```
Select any feature types to skip in check
```

Choose feature types that should not be looked at when making correspondences.

```
Check for duplicate data?
```

It is recommended to not check for duplicates during creation because it can slow the process. Simply, create the correspondences and the run the delete duplicate correspondences portion of `cmap_admin.pl` *(see the following step for instructions).*

```
Select the match type that you desire
```

Choose how the feature names will be compared. The default choices are to compare the whole name (`[1] exact match only`*) or use a Perl regular expression to strip off the extensions of read pairs (* `[2] read pairs '(\S+)\.\w\d$'` *). Other options can easily be created by someone comfortable with editing the* `cmap_admin.pl` *script.*

```
Number of 'from' maps to consider at once
```

The default is to take the maps one at a time. If the starting map sets have a lot of maps with few features on each, maps can be grouped together and considered at the same time. This is more efficient but it can drive memory usage up.

The easiest way to create correspondences is based on the feature names. Under this system, if two features on two separate maps share the same name, a correspondence will be created between them.

```
OK to make correspondences?
```

Say `yes` *and the correspondences will be created.*

9b. *To confirm the correspondence creation from the command line:* Enter the following command.

```
$ cmap_admin.pl -d CMap_Live --action
make_name_correspondences --evidence_type_acc ANB
--from_map_set_accs PMSA2006 --to_map_set_accs
PMSA2005,PMSA2004 --name_regex read_pair
```

10a. *Alternatively, to import a file using the menu:* Create and import tab delimited file of feature correspondences. The import file fields are `feature_name1`, `feature_acc1`, `feature_name2`, `feature_acc2` and `evidence`.

The feature accession IDs are optional but recommended as a way to link CMap data to other data sources (see the Background Information for a more in depth discussion). The evidence consists of an evidence type accession and an optional score in this format, `evidence_type_acc[:score]` *(e.g.,* `ANB` *or* `blast:1e-10`*).*

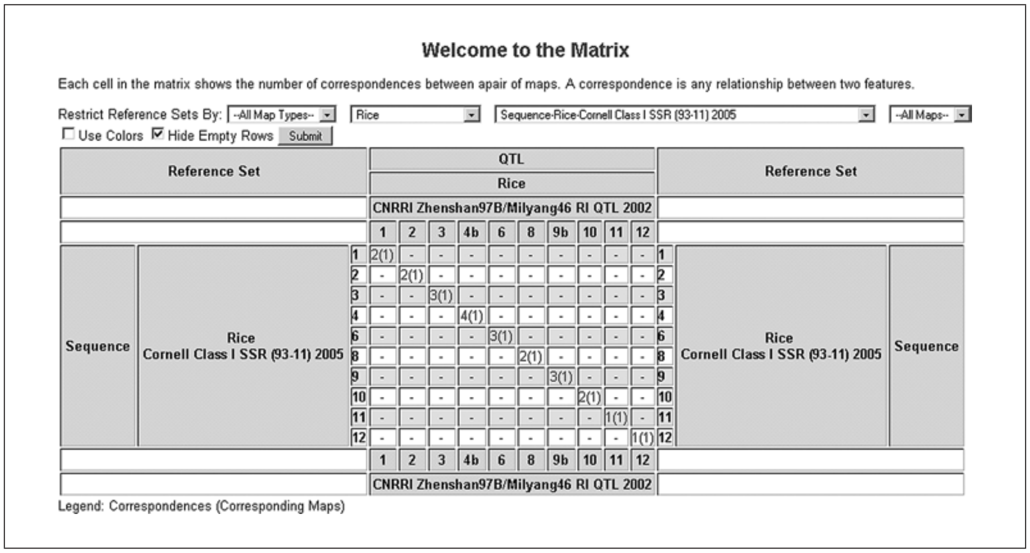
10b. *To import a file from the command line:* Type the following.

```
$ cmap_admin.pl -d CMap_Live --action
import_correspondences --map_set_accs
PMSA2006,PMSA2005,PMSA2004 import_file
```

11a. *To delete duplicate correspondences from the main menu:* Select option `[8] Delete duplicate correspondences`. The script will immediately go to work clearing duplicate correspondences. It will report the number of duplicate correspondences that were removed.

11b. *To delete duplicate correspondences from the command line:* Type the following.

```
$ cmap_admin.pl -d CMap_Live --action
delete_duplicate_correspondences
```



Welcome to the Matrix

Each cell in the matrix shows the number of correspondences between apair of maps. A correspondence is any relationship between two features.

Restrict Reference Sets By: --All Map Types-- ▾ Rice ▾ Sequence-Rice-Cornell Class I SSR (93-11) 2005 ▾ --All Maps-- ▾

☐ Use Colors ☑ Hide Empty Rows Submit

Reference Set			QTL											Reference Set	
			Rice												
			CNRRI Zhenshan97B/Milyang46 RI QTL 2002												
			1	2	3	4b	6	8	9b	10	11	12			
Sequence	Rice Cornell Class I SSR (93-11) 2005	1	2(1)	-	-	-	-	-	-	-	-	-	1	Rice Cornell Class I SSR (93-11) 2005	Sequence
		2	-	2(1)	-	-	-	-	-	-	-	-	2		
		3	-	-	3(1)	-	-	-	-	-	-	-	3		
		4	-	-	-	4(1)	-	-	-	-	-	-	4		
		6	-	-	-	-	3(1)	-	-	-	-	-	6		
		8	-	-	-	-	-	2(1)	-	-	-	-	8		
		9	-	-	-	-	-	-	3(1)	-	-	-	9		
		10	-	-	-	-	-	-	-	2(1)	-	-	10		
		11	-	-	-	-	-	-	-	-	1(1)	-	11		
		12	-	-	-	-	-	-	-	-	-	1(1)	12		
			1	2	3	4b	6	8	9b	10	11	12			
			CNRRI Zhenshan97B/Milyang46 RI QTL 2002												

Legend: Correspondences (Corresponding Maps)

Figure 9.8.11 The matrix entry point into the CMap viewer. This view shows the correspondences between maps in a rice sequence map set and maps in a rice QTL map set. The number of correspondences between each map is given as the first number, and the number of maps involved is given in parentheses. The second is useful when viewing the matrix at the map set level (not shown).

If name based correspondences were created twice or there is reason to believe duplicate correspondences have been inserted into the database, it is a good idea to delete any duplicates.

12a. *To reload the correspondence matrix from the menu:* Select option [9] Reload correspondence matrix. A prompt will display asking to confirm. Say `yes` and the correspondences matrix will be reloaded.

12b. *To reload the correspondence matrix from the command line:* Type the following.

```
$ cmap_admin.pl -d CMap_Live --action
reload_correspondence_matrix
```

The matrix is a table that displays the number of corresponding maps between map sets or the number of correspondences between maps (see Fig. 9.8.11). This can be used as an alternate entry point into the CMap viewer. For speed purposes, the matrix is precalculated and stored in a database table. After new correspondences are created, the correspondence matrix needs to be regenerated in order for the CMap matrix view to display correctly.

ADVANCED CONFIGURATION AND DATA MANIPULATION OPTIONS

BASIC PROTOCOL 5

This protocol describes configuration options for enhancing the usability of CMap. There are also configuration and data manipulation options for increasing the speed and clarity of CMap when the database grows exceptionally large. CMap has a large number of configuration options. To get the full list, see the administration document included with the CMap distribution. Each step in this protocol is independent of the others; feel free to pick and choose among the options.

NOTE: If the configuration files are changed after CMap is in use, the query cache will need to be purged using the `cmap_admin.pl` tool (see step 8).

Necessary Resources

Hardware

Computer with Internet access

Software

JavaScript (*http://java.sun.com*) enabled up-to-date Internet browser, e.g., Internet Explorer (*http://www.microsoft.com/ie*); Netscape (*http://browser.netscape.com*); Firefox (*http://www.mozilla.org/firefox*); or Safari (*http://www.apple.com/safari*)

An installation of CMap (Support Protocol)

Files

Files provided with the CMap distribution

Define useful configuration options

1. To configure the map scaling to a default where maps with the same base unit can be drawn in scale with each other:

   a. Set the `scale_maps` option to `1` in the configuration file.

   b. Declare the map units (as defined in the `map_type` options) scalable by placing the unit name inside `<scalable>` tags with a value of `1`. For an example, see below.

   Maps with units set to `0` *or not declared will not be drawn to scale.*

```
<scalable>
bp 1
bands 1
cM 1
</scalable>
```

   Maps with the same base unit can be drawn in scale with each other. If this option is not set, maps will be drawn all the same size.

   In the previous example, maps with units `bp`, `bands` *and* `cM` *will all be drawn to scale but a map with the map unit "*`minutes`*" will not because "*`minutes`*" is not defined as scalable.*

   It is also possible for maps with different map units to have a conversion factor applied and scale accordingly. For information on this option, see the CMap administration document.

2. To allow the `cmap_admin.pl` script to create correspondences between features of different types, add an `add_name_correspondence` line to the individual configuration file. List the feature type accessions (`feature_type_acc`) of the feature types on the same line to allow name based correspondences between features of those types.

   For example, `add_name_correspondence clone read` *allows clones and sequence reads to have correspondences between them.*

   By default, CMap will only create correspondences between features of the same type.

3. To validate the data source configuration after modification:

   a. Run `bin/cmap_validate_config.pl`. Call the script with the configuration file.

```
$ bin/cmap_validate_config.pl/usr/local/apache/conf/cmap.
conf/cmap_live.conf
```

This will check for syntax, required options, and deprecated options and give a list of any discretionary options that have not been specified.

b. *If the file configuration file is valid:* The script will print out a list of `Missing optional entr[ies]` and declare the file valid:

```
The config file, /usr/local/apache/conf/cmap.conf/
cmap_live.conf is VALID.
```

c. *If the configuration file is invalid:* the output will declare the file invalid and explain where the problem is:

```
The config file, /usr/local/apache/conf/cmap.conf/cmap_
live.conf is INVALID.
```

Configure CMap for speed and clarity

This set of configuration options is to be inserted into the individual data source configuration files to increase speed and clarity. All of the options are simply defaults and can be overridden by the user if they choose.

4. Set the `collapse_features` option to `1` (`collapse_features 1`). This causes features that would be drawn exactly the same way to be collapsed into one glyph. This improves the clarity of the image by preventing redundant feature glyphs on the map. It also reduces the time taken to draw features.

   Examine the differences between Figures 9.8.12 and 9.8.13 to see the effects of collapsing features.

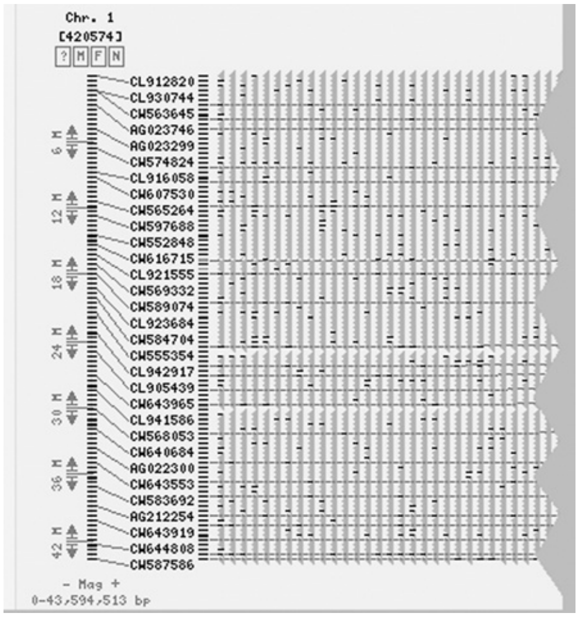


Figure 9.8.12 A view of a rice sequence map without using the `collapse_features` option. Thousands of redundant features are drawn. The actual image continues for eleven more screen widths which makes this view unusable and quite slow to load.

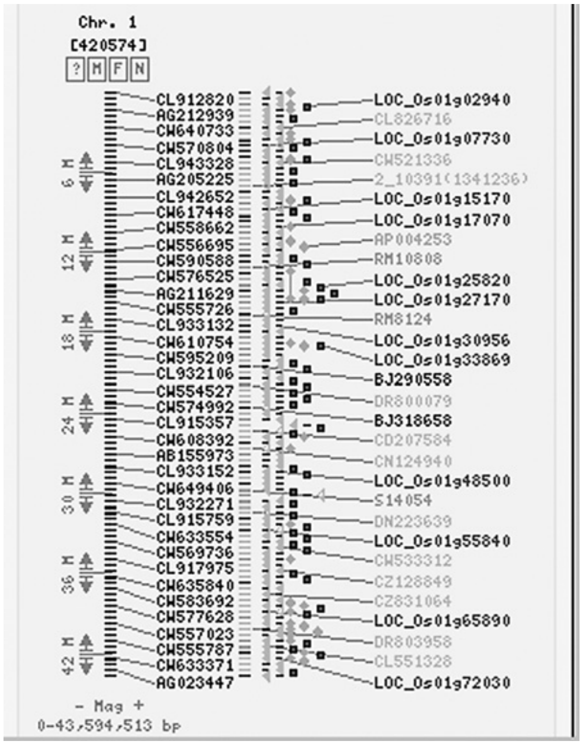


Figure 9.8.13 The map is using the `collapse_features` option resulting in a cleaner and quicker view of the map. For the color version of this figure go to *http://www.currentprotocols.com*.

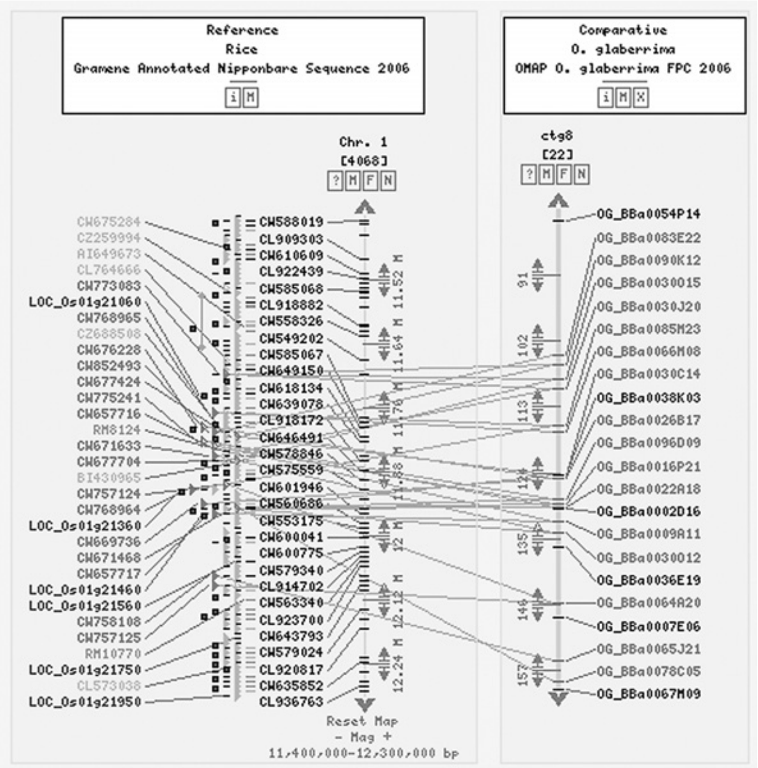


Figure 9.8.14 All of the feature types in this view are set to always display. For the color version of this figure go to *http://www.currentprotocols.com*.

5. Set the `feature_default_display` option to `corr_only` (`feature_default_display corr_only`) Only features that have correspondences between maps in the view will be displayed. This greatly reduces the time to process features when displaying a dense map.

   Examine the differences between Figures 9.8.14 and 9.8.15 to see the effects of displaying only features with correspondences.

6. Set `aggregate_correspondences` to `1` or `2` (`aggregate_correspondences 1`). This causes all of the correspondences between maps to be condensed into either one line or two lines. The single line setting joins maps from the average location of the features. The double line shows the span of the features on each map.

   This increases visibility when there are a lot individual correspondences crossing and cluttering up the screen. Figure 9.8.16 shows how aggregating correspondences can create a clean and usable image when there are many corresponding maps.

7. Set `omit_area_boxes` to `1` (`omit_area_boxes 1`) to reduce rendering time caused by a large number of features.

   Every feature has HTML associated with it to allow the user to click the mouse on it and have an action performed (this action can be configured but defaults to the sending the user to the feature details page). As the number of features increases, the HTML associated with them can grow to be larger than the image itself causing long, data transmission times and can break browsers like Internet Explorer with the amount of data

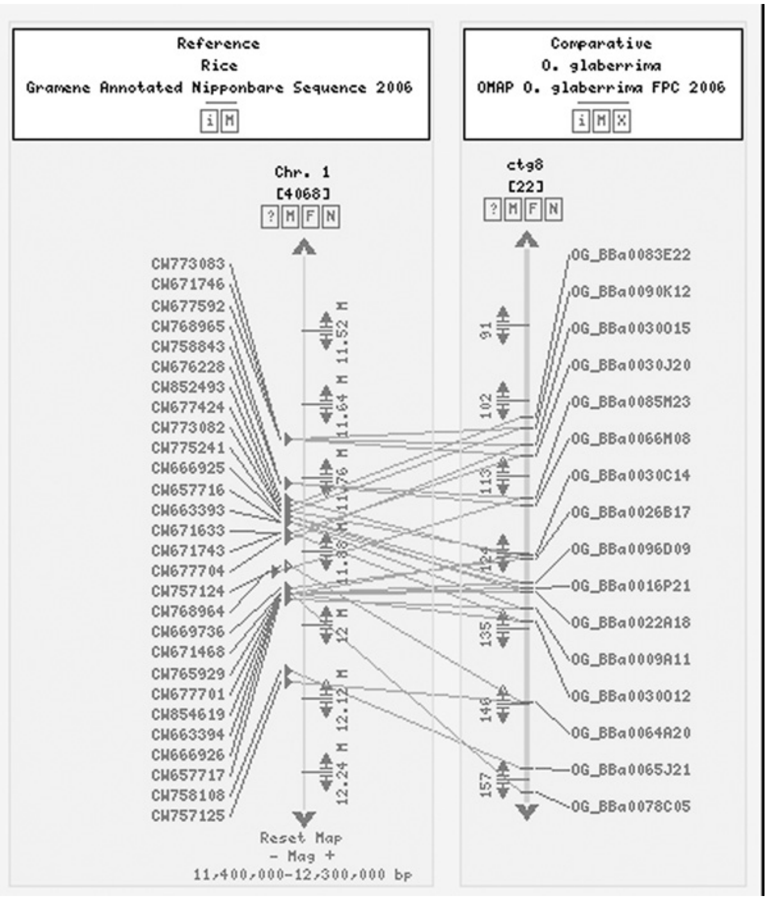


Figure 9.8.15 The feature types in this view are set to only display features with correspondences resulting in a cleaner look. Each feature type default can be set individually.

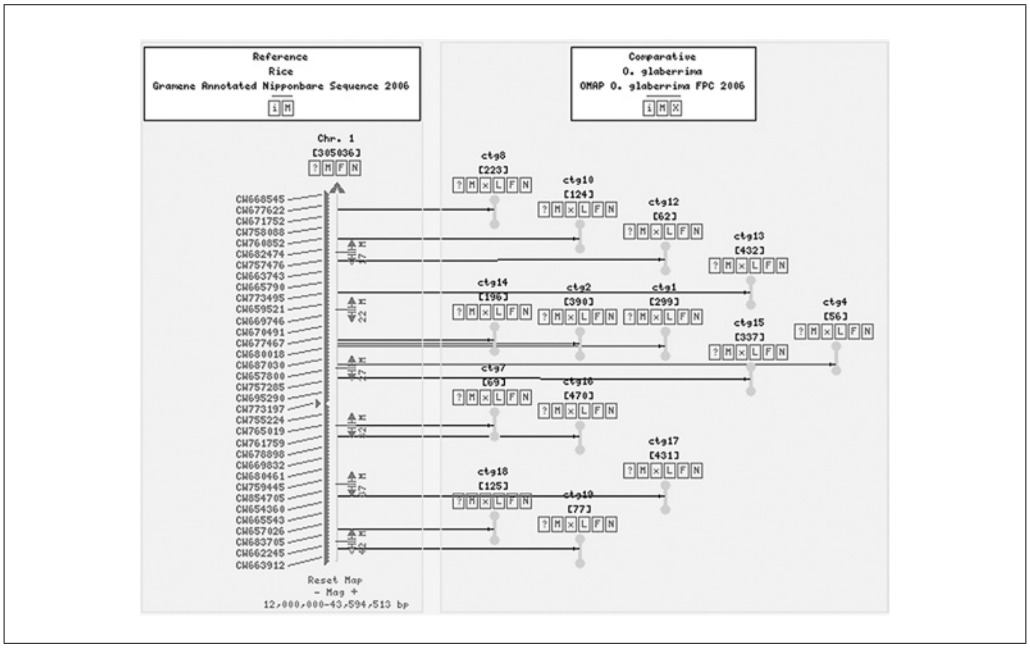


Figure 9.8.16 Correspondences can be aggregated into a single line (as shown) or two lines to decrease the number of lines on the screen.

to render. The omit_area_boxes *option removes the clickable area above the features which removes some functionality. Features will no longer be clickable, but this can help the user get data in a more timely fashion.*

Setting omit_area_boxes *to* 2 *will remove all clickable elements from the image which removes a large set of the navigation controls, so that is not recommended.*

Purge the query cache

8a. *To purge the query cache using menu options:* After editing, run cmap_admin.pl with the -d flag specifying the data source name, cmap_admin.pl -d CMap_Live. Choose option 10, [10] Purge the cache to view new data. Select 1, [1] Cache Level 1 Purge All to purge the whole cache.

8b. *To purge the query cache from the command line:* Type the following command.

```
cmap_admin.pl -d CMap_Live --action purge_query_cache.
```

Every time the configuration file is altered, the query cache should be purged to remove any old configuration data.

Manage data for speed and clarity

9. Use cmap_create_stacked_maps.pl as follows to condense a large set of comparison (relational) maps into contigs, based on a reference set, to ease viewing in CMap:

   a. Create a map set for the maps that will be created (see Basic Protocol 4, step 4).

   b. Supply the accession ID of the relational map set, the accession ID of the reference map set, the accession id of the new map set and the feature type accession ID for the features that represent the original relational maps on the new maps (which will need to be defined in the individual configuration file).

   c. Run the cmap_create_stacked_maps.pl script.

```
$ cmap_create_stacked_maps.pl -d CMap_Live --
map_set_to_stack FPC_SET_ACC --reference_map_set
```

```
SEQ_SET_ACC --new_map_set NEW_SET_ACC --
stack_feature_type_acc fpc_contig
```

Use this script on map sets that have a large number of maps and are mainly used in relation to another reference set to order and orient maps into large contigs based on correspondences to a reference map set.

An example of when this would be useful is a set of fingerprint contigs (FPCs) that has correspondences to a sequence assembly. Each chromosome of the sequence assembly could have correspondences to hundreds of FPCs. Displaying this severely taxes the server. The `cmap_create_stacked_maps.pl` *script takes those FPCs and finds the chromosome to which it has the most correspondences. The FPCs for each chromosome are ordered and oriented based on correspondences. New maps are then created by combining the FPCs into one giant contig for each chromosome. The features from the FPCs are carried over to the new maps and features representing the original FPCs are added. Instead of hundreds of maps, CMap can display a fewer, increasing readability by removing the clutter.*

This is a nondestructive process. The original map sets will be untouched. To keep the original relational map set from being used, it must be removed. However, it will then become difficult to modify the stacked maps because the position information on the maps will have changed from the original import.

10. Remove useless data and condense dense features into a depth display.

A proof of concept is provided in the form of the `cmap_manageParsedAceFile.pl` *script. This script takes a CMap import file of a sequence assembly with individual reads as features (created by the* `cmap_parseWashUAceFiles.pl` *script which is also included) and greatly reduces the number of features that CMap is required to display while maintaining usability.*

It removes read pairs that are within 100,000 bases on the same map because they are uninteresting. However before removing the reads, a read depth is tallied for a specified window size. In this way the read density is kept. Read pairs that are more than 100,000 bases apart are condensed into one `far_apart` *glyph to make them stand out. Reads that don't have a pair on the same map are left as anchors for correspondences. This reduces the number of reads on a map while leaving the singleton reads to be used to make correspondences with their pair.*

Configure the `read_depth` glyph as in the following example.

```
<feature_type read_depth>
feature_type_acc read_depth
feature_type Read Depth
default_rank 1
color red
shape read_depth
drawing_lane 1
drawing_priority 1
</feature_type>
```

INSTALLING CMap

SUPPORT PROTOCOL

This protocol describes how to download and install CMap on a Linux server.

Necessary Resources

Hardware

A fast desktop or server machine running a recent (<2 years old) version of Linux or Unix

Software

Relational database management system: MySQL (*http://www.mysql.com*), PostgreSQL (*http://www.postgresql.org*) or Oracle (*http://www.oracle.com*)

Perl version 5.8 or higher (*http://www.perl.org*)

Web server (e.g., Apache HTTP Server; *http://httpd.apache.org*).

gd Graphics Library version 2.0.28 or higher (*http://www.boutell.com/gd*)

Required Perl Modules: downloaded from CPAN (*http://search.cpan.org/*) or installed using the CPAN Perl module from the command line (`perl -MCPAN -e ''install module::name''`); installed separately; or as a group by installing `Bundle::CMap` from CPAN (bundle excludes the GD module because it requires a C library)

```
Algorithm::Numerical::Sample
Apache::Htpasswd
Bit::Vector
Cache::Cache
CGI::Session
Class::Base
Clone
Config::General
Data::Page
Data::Pageset
Data::Stag
Date::Format
Filesys::Df
GD
GD::SVG
IO::Tee
Module::Build
Params::Validate
Regexp::Common
Template
Template::Plugin::Comma
Text::RecordParser
Time::ParseDate
Time::Piece
version
XML::Parser::PerlSAX
XML::Simple
```

Optional Perl Modules:

Apache (*http://search.cpan.org/~gozer/mod_perl/*): needed if mod_perl is running on the Web server

BioPerl (*http://www.bioperl.org*): required for BLAST parsing

1. Download the CMap release from Generic Model Organism Database project (GMOD) project at SourceForge (*https://sourceforge.net/project/showfiles.php?group_id=27707*). The file will be named `cmap-##.tar.gz` where `##` is the version number.

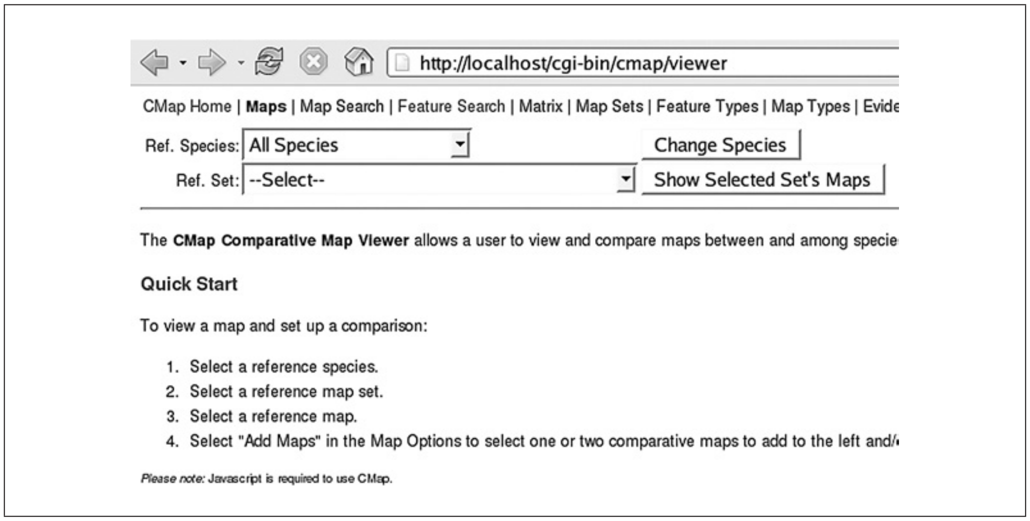
2. Untar the CMap tarball and enter the newly created `cmap/` directory.

```
$ tar -zxf cmap-##.tar.gz
$ cd cmap-##/
```

3. Create the Build file.

```
$ perl Build.pl
```

To change the directories into which CMap components are installed, use the `Build.pl` *options. For a list of these options, execute* `perl Build.pl --help`.


http://localhost/cgi-bin/cmap/viewer
CMap Home | Maps | Map Search | Feature Search | Matrix | Map Sets | Feature Types | Map Types | Evide
Change Species
Show Selected Set's Maps

Ref. Species: All Species

Ref. Set: --Select--

The **CMap Comparative Map Viewer** allows a user to view and compare maps between and among specie

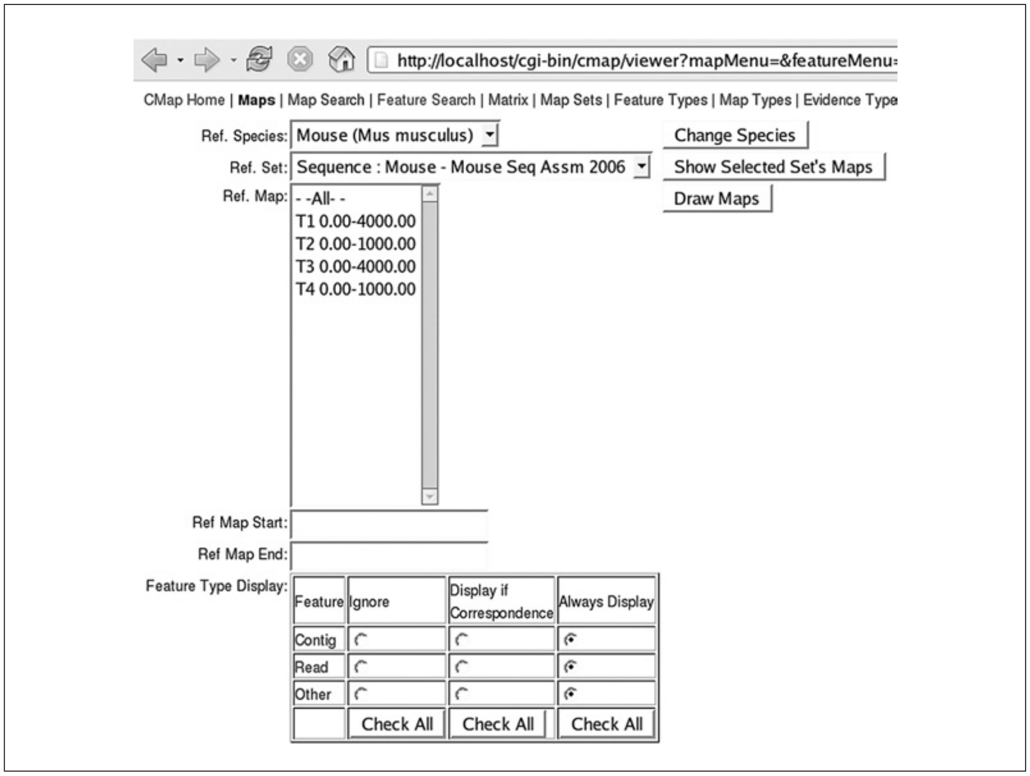
Quick Start

To view a map and set up a comparison:

1. Select a reference species.
2. Select a reference map set.
3. Select a reference map.
4. Select "Add Maps" in the Map Options to select one or two comparative maps to add to the left and/

Please note: Javascript is required to use CMap.

Figure 9.8.17 The initial CMap menu allows the user to select a species and a map set.


http://localhost/cgi-bin/cmap/viewer?mapMenu=&featureMenu=
CMap Home | Maps | Map Search | Feature Search | Matrix | Map Sets | Feature Types | Map Types | Evidence Type
Change Species
Show Selected Set's Maps
Draw Maps

Ref. Species: Mouse (Mus musculus)

Ref. Set: Sequence : Mouse - Mouse Seq Assm 2006

Ref. Map:
- -All- -
T1 0.00-4000.00
T2 0.00-1000.00
T3 0.00-4000.00
T4 0.00-1000.00

Ref Map Start:

Ref Map End:

Feature Type Display:

Feature	Ignore	Display if Correspondence	Always Display
Contig	○	○	●
Read	○	○	●
Other	○	○	●
	Check All	Check All	Check All

Figure 9.8.18 After selecting a map set, the user can select from a list of maps in the selected set. The user can also preselect the map start and stop (if selecting a single map) and the feature display options.

4. Build and install CMap.

```
$ ./Build
$ ./Build install
```

If there is a previous install on the machine, prompts will appear asking to overwrite various files. It is good practice to respond "no" to any locally modified files. The new versions will still exist in this `cmap-##/` *directory for later review.*

```
'/usr/local/apache/htdocs/cmap/index.html' exists.
Overwrite? [n]
```

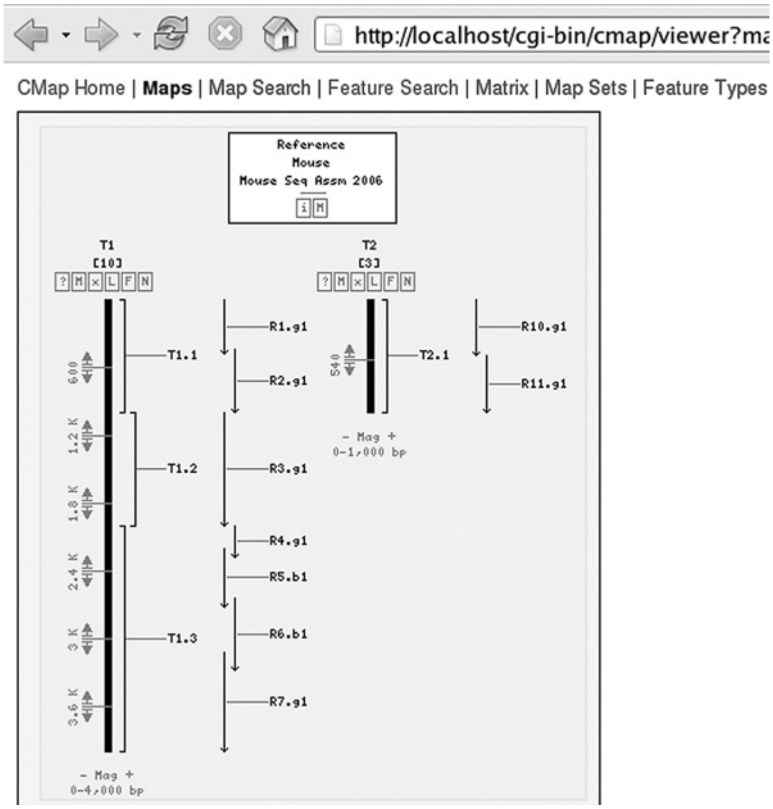


Figure 9.8.19 An initial view of two test maps imported from the `tabtest1` file.

```
'/usr/local/apache/conf/cmap.conf/global.conf' exists.
Overwrite? [n]

'/usr/local/apache/conf/cmap.conf/example.conf' exists.
Overwrite? [n]

'/usr/local/apache/cgi-bin/cmap' exists. Overwrite? [n]
```

CMap should be fully functional after completing the above steps.

5. Verify correct installation of CMap by performing the following operations:

   a. Point a Web browser to *http://localhost/cmap* (or substitute "localhost" with the correct domain) and the default introduction page (CMap Installation Summary) will be displayed.

      This page can be customized for each install.

   b. Click on the Maps link at the top of the page and the initial CMap menu will be displayed (see Fig. 9.8.17).

   c. Select a map set and click Show Selected Set's Maps to display the maps in the set (see Fig. 9.8.18).

   d. Select one or more of the maps from the list and click Draw Maps to display an image of the map(s) (see Fig. 9.8.19).

   e. Click on the various buttons on the image.

   If no errors occur, CMap has been installed correctly and can be used.

COMMENTARY

Background Information

CMap was written for the Gramene project (*http://www.gramene.org*) to display comparisons between various cereal genome data. It has since been moved under the umbrella of the Generic Model Organism Database project (GMOD, *http://www.gmod.org*) and is in use for both plant (e.g., The Legume Information System, *http://www.comparative-legumes.org*) and animal (e.g., The Honey Bee Genome Database, *http://racerx00.tamu.edu/bee_resources.html*) data.

One of the guiding ideas for CMap is for it to be generic. Rather than being bound to a specific data type (i.e., sequence, genetic, or physical data) the CMap database can handle a variety of map data. This is what allows CMap to display comparisons between a wide array of data.

As of this writing, the current version is CMap 0.16. CMap is a more mature application than the version number suggests. There have been 16 releases up to this point.

CMap only uses standard SQL. This means that CMap can run on any relational database. It is tested on MySQL and PostgreSQL but it should work on Oracle, SQLite or any other modern database.

Configuration files

CMap allows for multiple databases to be used, allowing the user to select among them. In fact, the same database can have multiple configurations to allow for different views and different permissions. CMap requires one configuration file for each configuration. These individual configuration files are referred to as data sources.

Options that relate to the computer system such as location of the temp directory are stored in one global configuration file named `global.conf`. The options in `global.conf` are shared across all data sources. The `global.conf` file also dictates which data source will be used as the default.

The `global.conf` and data source configuration files are installed in the `CONF` directory, which unless specified elsewhere during installation, is located in `/usr/local/apache/conf/cmap.conf/`. Data source files are identified by the `.conf` extension.

Apache-style configuration files

The configuration files are written in Apache-style syntax. Options are written in `Name Value` format and groups of options are clustered between tags in angle brackets (XML-like). For more information on the format, execute `perldoc Config::General` from the command line.

Caching system

CMap employs a caching system to speed up long database queries. The full page of the first view of a map is also cached. It is important to purge the cache whenever a change is made to the configuration files or the database is manipulated (without using `cmap_admin.pl`). For instructions on how to purge the cache, see Basic Protocol 5, step 8.

Session system

Due to the number of user options that CMap offers and the complex data required for some vital functionalities, CMap requires server-side sessions. Without sessions, the URLs needed to convey the all of the display information were longer than browsers like Internet Explorer could handle. These sessions store all of the information about the view. CMap navigation links rely on the sessions to create the next view. The disadvantage to the sessions is that they eventually expire and get removed. This renders old bookmarks useless.

To address this issue CMap has a Save Link button below every image that will store the session information on the server indefinitely and provide a permanent link to access the view. The permanent link can then be bookmarked or emailed reliably. Even better, after visiting the permanent link once, the whole page gets cached which makes viewing that page quicker for the people being sent the link.

HTML templates

Instead of generating HTML in Perl, CMap uses the templating system of Template Toolkit (*http://www.template-toolkit.org*). In the templates directory, which can be set at install time, are templates that can be modified for each installation. This allows protocol users to easily change the Web pages without having to open up the code.

Critical Parameters and Troubleshooting

Accession IDs

Most of the objects in CMap have an accession ID. These are unique identifiers that can be set during data creation. It is a recommended practice to use meaningful accession

IDs for features, maps, map sets and species. This can make tying different data sources together easier. An example would be to use GenBank accession IDs for sequence maps or use an in-house identifier that links to separate databases.

Accession IDs are alphanumeric (a-zA-Z0-9) and can be up to 30 characters long. Always use at least one letter when creating an accession ID. A strictly numeric accession ID can cause conflicts because when not given, CMap will automatically use the main table ID as the accession ID which is numeric. A record with a duplicate accession will not be allowed into the database.

Missing configuration files

The configuration files (one `global.conf` and one or more data source files) must be in the `cmap.conf/` directory and the data source configuration files must be set to enabled. If the `global.conf` file is missing, a `No global.conf found` error will be displayed. If there are no data source configuration files present with the `is_enabled` option set, a `No database conf files found` error will appear.

Permissions

In order for CMap to run, the Web server user needs to have the correct privileges to CMap files and the directory path to those files. These permissions should have been set correctly during the install. The following will describe what privileges are needed.

If it doesn't have read and execute permissions to the `cmap.conf/`"directory or doesn't have read access for the actual configuration files, CMap will display a variety of errors complaining that the configuration files couldn't be found or accessed.

The Web server must have read and execute permissions for the template directory and read permissions for the templates themselves or an error related to a `.tmpl` file will be displayed.

The server must have read, write and execute permissions to the cache directory. This directory is assigned at install but defaults to `/usr/local/apache/htdocs/cmap/tmp/` or there will be `File does not exist` errors regarding the image file.

Sessions

The `Invalid session_id:` error usually means the session has expired. To avoid seeing this, use the `Save Link` button underneath the image. If the error is occurring soon after starting the session (usually the second image view), that could mean that the sessions directory is not being written to due to fullness or permissions. The sessions directory must have read, write and execute permissions for the server.

Cache problems

If changes are made to the data or the configuration files and those changes are not being used, or are being used sporadically, this is a sign that the cache needs to be cleared. See Basic Protocol 5, step 8 for a description of how to clear a cache.

Unique data source naming

Another cause of sporadically changing data is configuration files with the same `<database>` name. CMap will check for this if the duplicates are in the same installation. However, if there are two installations on the same machine, data sources with the same name can share the same cache space. Data from one installation can seep into the display of the other. The easy solution is to rename one of the data sources.

E-mail list

If other assistance is needed, send a message detailing the issue to the CMap list, *gmod-cmap@lists.sourceforge.net*.

Internet Resources

https://sourceforge.net/project/showfiles.php?group_id=27707

The location of the CMap package.

http://www.gramene.org

The Gramene project Web site for which CMap was initially created.

http://www.gmod.org

The Generic Module Organism Database project Web site.

http://www.gmod.org/cmap

The CMap home page.

Contributed by Ben Faga
Cold Spring Harbor Laboratories
Cold Spring Harbor, New York

Using the Generic Genome Browser (GBrowse)

UNIT 9.9

Maureen J. Donlin[1]

[1]Department of Biochemistry and Molecular Biology and Department of Molecular Microbiology and Immunology, Saint Louis University School of Medicine, St. Louis, Missouri

ABSTRACT

A genome browser is software that allows users to visualize DNA, protein, or other sequence features within the context of a reference sequence, such as a chromosome or contig. The Generic Genome Browser (GBrowse) is an open-source browser developed as part of the Generic Model Organism Database project (Stein et al., 2002). GBrowse can be configured to display genomic sequence features for any organism and is the browser used for the model organisms *Drosophila melanogaster* (Grumbling and Strelets, 2006) and *Caenorhabditis elegans* (Schwarz et al., 2006), among others. The software package can be downloaded from the Web and run on a Windows, Mac OS X, or Unix-type system. Version 1.64, as described in the original protocol, was released in November 2005, but the software is under active development and new versions are released about every six months. This update includes instructions on updating existing data sources with new files from NCBI. *Curr. Protoc. Bioinform.* 28:9.9.1-9.9.25. © 2009 by John Wiley & Sons, Inc.

Keywords: computational biology • genome • GBrowse • genome browser

INTRODUCTION

The GBrowse genome browser was designed to be highly configurable and portable. It can be run locally on a laptop computer with modest memory and CPU, or be installed on a high-end server to provide a browsable genome to the online research community. GBrowse can be used to display genome annotations on small genomes, such as *S. cerevisiae*, as easily as displaying large genomes such as human—the main constraint being how much disk space is available.

A central feature of GBrowse is its use of adaptors to connect to various types of databases. This unit has two main protocols. Basic Protocol 1 shows how to set up a browser on a small genome using the GBrowse flat file adaptor. This adaptor is suitable for genomes up to 20,000 features (genes or other annotations). Basic Protocol 2 shows how to use the relational database adaptor, which is suitable for very large genomes. Two support protocols provide additional information on using GBrowse. Support Protocol 1 describes how to install the software, and Support Protocol 2 shows how to load and view sequence annotation records from GenBank. A troubleshooting guide describes how to isolate and remedy problems.

CONFIGURATION AND USE OF GBrowse 1.X

BASIC PROTOCOL 1

This protocol will take the user through the main features of GBrowse and point out some of the most common problems encountered when configuring the software for the first time. This protocol assumes that the user has successfully set up GBrowse as described in Support Protocol 1. In this protocol, the user will use a file-based GBrowse database with simulated Volvox genome annotation data. In Basic Protocol 2, the user will set up

Current Protocols in Bioinformatics 9.9.1-9.9.25, December 2009
Published online December 2009 in Wiley Interscience (www.interscience.wiley.com).
DOI: 10.1002/0471250953.bi0909s28

GBrowse using a MySQL database. This protocol is a short introduction to the many things GBrowse can do. There are numerous other features to be explored using the online tutorial. The user can find additional information in the `CONFIGURE_HOWTO` document, and in the documentation for Bio::DB::GFF and Bio::Graphics in the BioPerl distribution. Should the user have questions or problems with the installation and setup, contact the GMOD-GBrowse mailing list for help. This protocol was adapted, with permission, from the online tutorial written by Lincoln Stein that is installed with GBrowse.

Necessary Resources

Hardware

Unix (Linux, Solaris, or other variety) workstation or Macintosh with OS X 10.2.3 or higher

A minimum of 500 Mb RAM

Internet connection

Software

All necessary software should be installed if Support Protocol 1 has been completed

Files

The data files used during the tutorial are found in the install directory under the `/docs/tutorial` directory path. The `data_files` directory contains the DNA and features files to load into the local database. The `conf_files` directory contains the GBrowse configuration files for the user to work with and modify.

Setting the file permissions

The user will use a file-based database, which allows GBrowse to run directly off text files. To prepare this database, find the GBrowse databases directory, which was created on the Apache Web server directory during installation. When working with a RedHat Linux system, it should be located at `/var/www/html/gbrowse/databases`. Similarly, check that the `gbrowse.conf` configuration directory can be located. It should be located at: `/etc/httpd/conf/gbrowse.conf` and contain the configuration file `yeast_chr1.conf`. If one is unsure of where these directories are located, the main page of the GBrowse installation under `Directory Paths` should be checked. It will list the directory paths for the GBrowse documentation and in-memory databases, as well as the path for the configuration files.

1. Change the permissions of the database and configuration directories so that the user can write to them without root privileges using the commands:

```
$ su
Password: *********
# chown my_user_name /var/www/html/gbrowse/databases
# chown my_user_name /etc/httpd/conf/gbrowse.conf
# exit
```

   Replace `my_user_name` with the user's login name. The `$` and `#` sign in these examples are the command-line prompts for the unprivileged and root users respectively. Unix systems are variable, but the prompt usually ends with `$`, `%`, or `#`.

   Examine the contents of the databases directory. There should be a single subdirectory named `yeast_chr1. The yeast_chr1` subdirectory is where the example yeast chromosome 1 dataset is stored. For each file-based genome displayed in GBrowse, a separate directory needs to be created.

2. Create an empty `volvox` subdirectory, and make it world writable using the commands:

```
$ cd /var/www/html/gbrowse/databases
$ mkdir volvox
$ chmod go+rwx volvox
```

Next, put the first of several data files into the `volvox` database directory. Within the `data_files` subdirectory of this tutorial, the user should find the file `volvox1.gff`.

3. Copy `volvox1.gff` into the `volvox` database directory using the commands:

```
$ cp volvox1.gff /var/www/html/gbrowse/databases/volvox/
```

Next, one will need a GBrowse configuration file to tell GBrowse how to render this data set.

4. Change to the subdirectory `conf_files`, where one should find a sample configuration file named `volvox.conf`.

5. Copy `volvox.conf` into the GBrowse configuration directory:

```
$ cp volvox.conf /etc/httpd/conf/gbrowse.conf/.
```

6. To view the dataset, open a Web browser and type in the address:

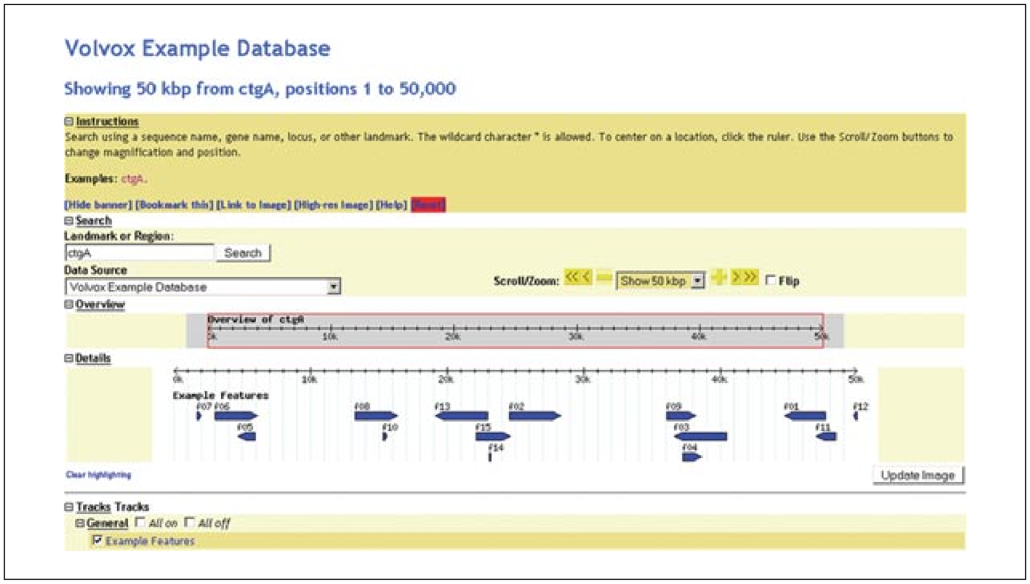
```
http://localhost/cgi-bin/gbrowse/volvox
```

If viewing the pages from a computer other than the one on which GBrowse was installed, replace `localhost` *with the name of the Web server.*

7. Type in `ctgA` in the search box. The page as shown in Figure 9.9.1 should now be seen.

Figure 9.9.1: Example features from the `volvox1.gff` *file as displayed with the* `volvox.conf` *configuration file.*

If a blank page or an `Internal server error`*, message appears, there are a few things that can be checked. First, open the file* `volvox.conf` *with a text editor and confirm that the path to the* `volvox` *database directory in this section is correct:*



Volvox Example Database

Showing 50 kbp from ctgA, positions 1 to 50,000

⊟ Instructions

Search using a sequence name, gene name, locus, or other landmark. The wildcard character * is allowed. To center on a location, click the ruler. Use the Scroll/Zoom buttons to change magnification and position.

Examples: ctgA.

Landmark or Region: ctgA

Data Source: Volvox Example Database

Scroll/Zoom: Show 50 kbp □ Flip

⊟ Tracks Tracks

⊟ General □ All on □ All off

☑ Example Features

Figure 9.9.1 Example features from the `volvox1.gff` file as displayed with the `volvox.conf` configuration file.

```
db_args = -adaptor memory
-gff '/var/www/html/gbrowse/databases/volvox'
```

If there is a space in the argument to the `-gff` *option, then put single quotes around the path as shown in the example above. Next, check that the* `volvox1.gff` *file does exist in the* `volvox` *database directory and that it is readable by all users on the system being used. Check that the* `volvox.conf` *configuration file is in the same directory as* `yeast_chr1.conf`*, and that it is readable by all users on the system being used. If none of the above suggestions fix the problem, check the Apache server error log for error messages as described in the Troubleshooting section at the end of this unit. If all else fails, submit the errors to the GMOD-GBrowse mailing list for help.*

Working with the data file

The data file for GBrowse is based on a format called the General Feature Format (GFF). It is organized into nine columns, separated by tabs. Table 9.9.1 lists the nine columns of the GFF file in order, from left to right.

The group field format is `class name`, where `class` describes the class of the feature and `name` describes its name. Separate the class and name by a space, **not** a tab. The feature `class` is just a prefix that distinguishes one name from another, but it can be confusing because it is very similar to the feature `type`. Features with the same name can be distinguished by giving them distinctive prefixes. For example, use the feature `class` to distinguish "Transcript M1.2" from "Gene M1.2." Later versions of GFF, such as GFF3, will likely do away with the class entirely. The author suggests reusing the type field here. In these examples, an initial capital letter in the class field distinguishes the class field from the type field.

Open the `volvox1.gff` file in a text editor and examine the data file in more detail. One should see a series of genome "features," of which the first few lines are shown in Figure 9.9.2.

Each of the features seen in Figure 9.9.2 also has a source of `example` and a type of `my_feature`, and occupies a short range (roughly 1.5 k) on the reference contig named `ctgA`. In addition to the features themselves, there is an entry for the contig itself (type `contig`). This entry tells GBrowse the length of ctgA. Each line of the file corresponds to a feature on the genome and has the nine columns defined by the GFF specification as described in Table 9.9.1. The last column has features separated by spaces, not tabs.

Defining the feature tracks

8. Using a text editor, open the `volvox.conf` file from its location in the `gbrowse.conf` directory.

   If a mistake is made, just copy a fresh version of `volvox.conf` *from the* `/docs/tutorial/conf_files` *directory.*

   Scan down the file until finding the part that starts with the line:

```
### TRACK CONFIGURATION ###:
[ExampleFeatures]
feature = my_feature
glyph = generic
stranded = 1
bgcolor = blue
height = 10
key = Example Features
```

Table 9.9.1 The GFF3 File Specifications[a]

Column	Description
Seqid	This is the id of the landmark, which establishes the coordinate system for the annotation or current feature. This is usually the name of a chromosome, clone, or contig. In this protocol, the seqid is “ctgA.” A single GFF file can refer to multiple reference sequences.
Source	The column lists the source of the annotation or describes how the feature was derived. In the protocol, the source is “example” for lack of a better description. Many people use the source as a way of distinguishing between similar features that were derived by different methods; for example, gene annotations derived from different prediction software. This column can be left blank by replacing the source with a single dot (“.”).
Type	This column describes the feature type. The user can choose anything they like to describe the feature type, but common names are “gene,” “repeat,” “exon,” and “CDS.” For lack of a better name, the features in this protocol are of type “my_feature.”
Start	This column lists the position that the feature starts at, relative to the reference sequence. The first base of the reference sequence is position 1.
End	This column lists the end of the feature, again relative to the reference sequence. The end is always greater than or equal to start.
Score	For features that have a numeric score, such as sequence similarities, this field holds the score. Score units are arbitrary, but it is common to use the expectation value for similarity features and *p*-values for ab initio gene prediction features. This can be left blank by replacing the column with a dot (“.”).
Strand	For features that are strand-specific, this field is the strand on which the annotation resides. It is “+” for the forward strand, “–” for the reverse strand, or “.” for annotations that are not stranded. If the user is unsure of whether a feature is stranded, use a “?” here.
Phase	For CDS features that encode proteins, this field describes the part of the codon on which the first base falls. The field is a number from 0 to 2, where 0 means that the first base of the feature corresponds to the start of the codon, 1 means that the second base of the feature corresponds to the start of the codon, and 2 means that the third base of the feature corresponds to the start of the codon. Phase information is represented by a glyph called “cds” to show how the reading frame changes across splice sites. For all other feature types, use a dot here.
Attributes	A list of feature attributes in the format tag=value. Multiple tag=value pairs are separated by semicolons. URL escaping rules are used for tags or values containing the following characters: “, = ;”. Spaces are allowed in this field, but tabs must be replaced with the %09 URL escape. This field is not required.

[a] GFF3 format can be used to represent features with multiple tiers, such as the relationship between genes, transcripts, and spliced isoforms. The full GFF3 specification can be found at *http://www.sequenceontology.org/gff3.shtml*.

```
ctgA   example   contig       1      50000   .   .   .   Contig ctgA
ctgA   example   my_feature   1659   1984    .   +   .   My_feature f07
ctgA   example   my_feature   3014   6130    .   +   .   My_feature f06
ctgA   example   my_feature   4715   5968    .   -   .   My_feature f05
ctgA   example   my_feature   13280  16394   .   -   .   My_feature f08
...
```

Figure 9.9.2 The first few lines of the genome “features” from the `volvox1.gff` file.

This "stanza" describes one of the tracks displayed by GBrowse. The track has an internal name of `ExampleFeatures`*, which can be used on the HTML page to turn on the track. The internal name is enclosed by square brackets. Following the track name is a series of options that configure the track. The* `feature` *option indicates what feature type(s) is displayed inside the track. Currently, it displays the* `my_feature` *feature type. The* `glyph` *option specifies the shape of the rendered feature. The default is* `generic`*, which is a simple filled box, but there are dozens of glyphs to choose from. The* `stranded` *option tells the generic glyph to display the strandedness of the feature—this creates a little arrow at the end of the box. The options* `bgcolor` *and* `height` *control the background color and height of the glyph respectively, and* `key` *assigns the track a human-readable label. Experiment with changing the track definition by changing the color of the glyph using a text editor.*

9. Set the option `bgcolor = orange`.

10. Save the file and reload the page in the browser.

    The features should be seen rendered in orange rather than blue as in the initial page.

11. Set `height = 5`, and `key = Skinny features`.

12. Set `stranded = 0` (which means "false").

13. Save the file and reload the page in the browser.

    By changing just a few options, a very distinctive track can be created. Now try changing the glyph. One of the standard glyphs was designed to show PCR primer pairs and it is called `primers`.

14. Change `glyph = generic` to `glyph = primers`.

15. Save the file and reload the page in the browser.

    Depending on other changes that the user might have made earlier, the result will look similar to the image in Figure 9.9.3.

    Far more glyph options are available than can be described in this protocol. Refer to the `CONFIGURE_HOWTO` *file, available as a link from the initial GBrowse page or as a pod document in the* `/docs/pod/` *directory of the GBrowse distribution for a list of the most popular glyphs and the options available for them. There is also a PDF file located in the* `/docs/` *directory of the GBrowse distribution that lists all of the available colors for GBrowse.*

Searching for named features

An important function of GBrowse is the ability to search for named features. If the user looks through the `volvox1.gff` data file, they will see that all the example features are named in column nine, and that their class is `My_feature`. GBrowse has a very flexible search feature; try a few searches to see how they work.

16. Type in the name of the reference sequence, `ctgA`, and it will display the entire contig.

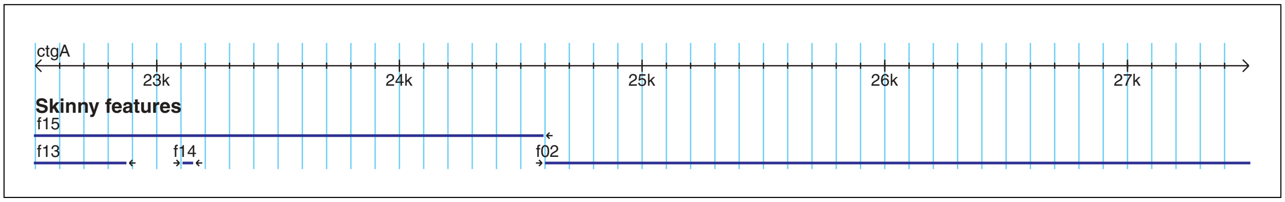


Figure 9.9.3 Display of the feature using the primers glyph.

17. Type in a range in the format `ctgA:start..stop`, such as `ctgA:5000..8000` to see a portion of the contig.

    In addition, GBrowse can search for features by name. By default, the name of the object must be preceded by its class in the format `Class:name`*.*

18. To search for My_feature f07, type `My_feature:f07` into the search box.

    Although this search works, users will not always know the class name of a given feature to put into the search box. The user can declare one or more classes `automatic` *and specify the order in which GBrowse will search for them to eliminate the need for including the class name in the search field. Do this with the* `volvox` *database as follows.*

19. Open the `volvox.conf` configuration file in a text editor.

20. Find the option named `automatic classes`, and set it to:

    ```
    automatic classes = My_feature.
    ```

21. Save the file and reload the page in the browser. This tells GBrowse to search for the `My_feature` class for a match whenever a user types in an unqualified search term. Now type `f07` directly into the search field and GBrowse will find the feature and display it. Several `automatic classes` can be listed on this line by separating them with spaces.

22. Set the `automatic classes = My_feature Gene Transcript Contig Chromosome`.

23. Save the file and reload the page in the browser.

24. To become familiar with other search options, try the following searches:

    ```
    f1*
    f07:-5000..5000
    *3
    ```

    The `*` *is a wild-card option for searches.*

Adding descriptions to a feature

By default, GBrowse displays the name of the feature above its glyph provided that there is sufficient space to do this. Optionally, the user can attach descriptive text to the feature. This text will be displayed below the feature, and can also be searched. The user can add descriptions, notes, and other comments into the ninth column of the GFF load file. The example file `volvox2.gff` shows how this is done. An excerpt from the top of the file follows in Figure 9.9.4.

This GFF file (see Fig. 9.9.4) defines several features of type `motif`. The ninth column, in addition to giving each of the motifs names (e.g., `Motif m11`), adds a `Note` attribute to each feature. Attributes are `name=value` pairs, where the attribute name is a single word, and the value is a piece of text. If the value text contains white space (spaces or

```
ctgA   example   motif   11911   15561   .   +   .   Motif m11 ; Note "kinase"
ctgA   example   motif   13801   14007   .   -   .   Motif m05 ; Note "helix loop helix"
ctgA   example   motif   14731   17239   .   -   .   Motif m14 ; Note "kinase"
ctgA   example   motif   15396   16159   .   +   .   Motif m03 ; Note "zinc finger"
```

Figure 9.9.4 An excerpt from the example file `volvox2.gff` is shown here.

tabs), the text must be enclosed by double quotes as shown above for `helix loop helix`. It does not hurt to enclose the text in quotes even if it does not contain white space, as shown in the example. Attribute pairs are separated from the Class/name pair by a semicolon, as shown above. The online tutorial and `CONFIGURE_HOWTO` file list many ways to take advantage of different types of attributes. For now, it is only important to know that an attribute named `Note` is automatically displayed and made searchable.

25. Add `volvox2.gff` to the `volvox` database by copying the file into `/var/www/html/gbrowse/databases/volvox` so that the directory contains both the `volvox1.gff` and `volvox2.gff` files.

26. Open `volvox.conf` in a text editor and add the following stanza to the configuration file:

```
[Motifs]
feature = motif
glyph = span
height = 5
description = 1
key = Example motifs
```

This defines a new track whose internal name is Motifs. The corresponding feature type is `motif` *and it uses the* `span` *glyph, a graphic that displays a horizontal line capped by vertical endpoints. The* `height` *is set to five pixels, and the human-readable key is set to* `Example motifs`*. A new option,* `description`*, is a flag that tells GBrowse to display the* `Note` *attribute, if any. Any nonzero value means true.*

27. Save the file and reload the page in the browser.

28. Turn on the “Example motifs” checkbox below the main image and click the Update Image button.

The result is shown in Figure 9.9.5.

29. In the search text box, type the word `kinase`.

The resulting page should be a list of all the motifs whose `Note` *attribute contains the word “kinase.”*

Showing multi-segmented features

Many features are discontinuous; for example, spliced transcripts, and gapped sequence similarity alignments such as the alignment of cDNAs to the genome. Discontinuous features in GBrowse can be displayed using specific formatting options in the GFF file.

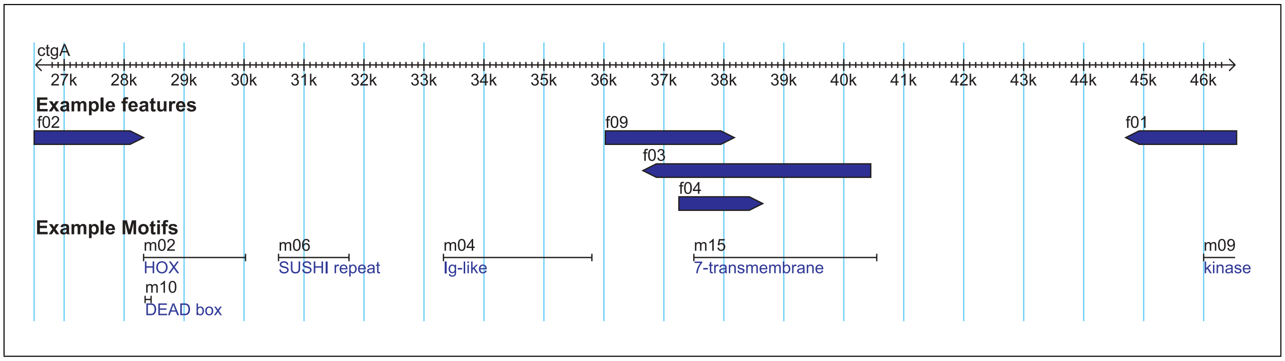


Figure 9.9.5 The GBrowse details view displaying the `Note` attribute from the GFF file.

```
ctgA   example   match 6885   8999   .   -   .   Match seg03
ctgA   example   HSP   6885   7241   .   -   .   Match seg03
ctgA   example   HSP   7410   7737   .   -   .   Match seg03
ctgA   example   HSP   8055   8080   .   -   .   Match seg03
ctgA   example   HSP   8306   8999   .   -   .   Match seg03
…
```

Figure 9.9.6 The data file `volvox3.gff` contains a simulated dataset of a series of gapped nucleotide alignments. An excerpt from the file is shown here.

The data file `volvox3.gff` contains a simulated dataset of a series of gapped nucleotide alignments. An excerpt from the file is shown in Figure 9.9.6.

Several lines in a GFF file can share the same feature name in column nine and thus represent a single segmented feature. In the example seen in Figure 9.9.6, all five lines define a single feature. The first line, with the type `match`, has start and end coordinates that correspond to the full length of the alignment. The next four lines, of the type `HSP`, have start and end coordinates indicating one section of the match. In this example, `Match seq03` starts at position `6885` and ends at `8999`. It has four subsegments: `6885` to `7241`; `7410` to `7737`; `8055` to `8080` and `8306` to `8999`. The types `match` and `HSP` define for GBrowse the relationship between the full-length feature and its subparts. A series of `aggregators`, code modules that are loaded when GBrowse starts, mediate the specific type names and the display parameters for the gapped features.

30. Copy the `volvox3.gff` file into the `volvox` database directory.

31. Open `volvox.conf` in a text editor and add the following track definition:

```
[Alignments]
feature = match
glyph = segments
key = Example alignments
```

This declares a new track named `Alignments`*, which displays features of type* `match` *using a glyph named* `segments`*. The segments glyph is specialized for displaying objects that have multiple similar subparts.*

32. Save the modified configuration file and reload the page in the browser. Check the Example Alignments checkbox and click the "Update Image" button to show the new feature.

The user should see something like Figure 9.9.7. However, instead of showing multi-segmented features, the track called "Example alignments" shows a single solid box that spans the entire length of the feature, which is not quite the desired outcome.

To make multipart features display correctly, the user must activate or define an appropriate aggregator. This is very easy for the similarity/match relationship, because there is already a predefined aggregator named `match`.

33. Open the `volvox.conf` configuration file, and find the option line near the top of the file that reads `aggregators =`.

34. Change this to read: `aggregators = match`.

35. Save the file and reload the page in the browser.

This tells GBrowse to turn on the `match` *aggregator. The user should now see an image similar to Figure 9.9.8, with the multi-segmented glyphs for the Example alignments.*

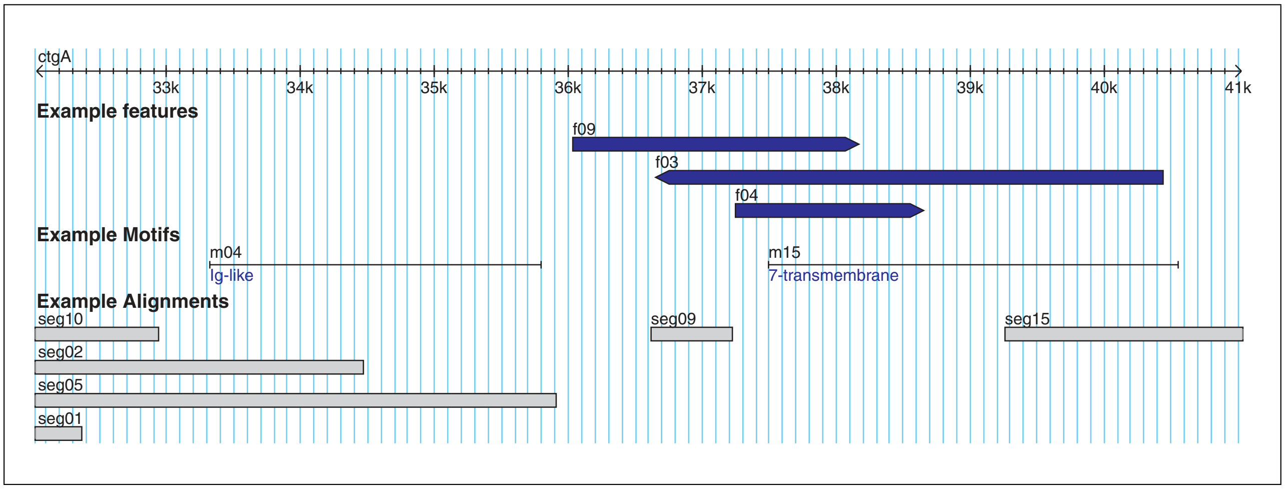


Figure 9.9.7 Multi-segmented features are not properly displayed unless the appropriate aggregator is activated in the configuration file.

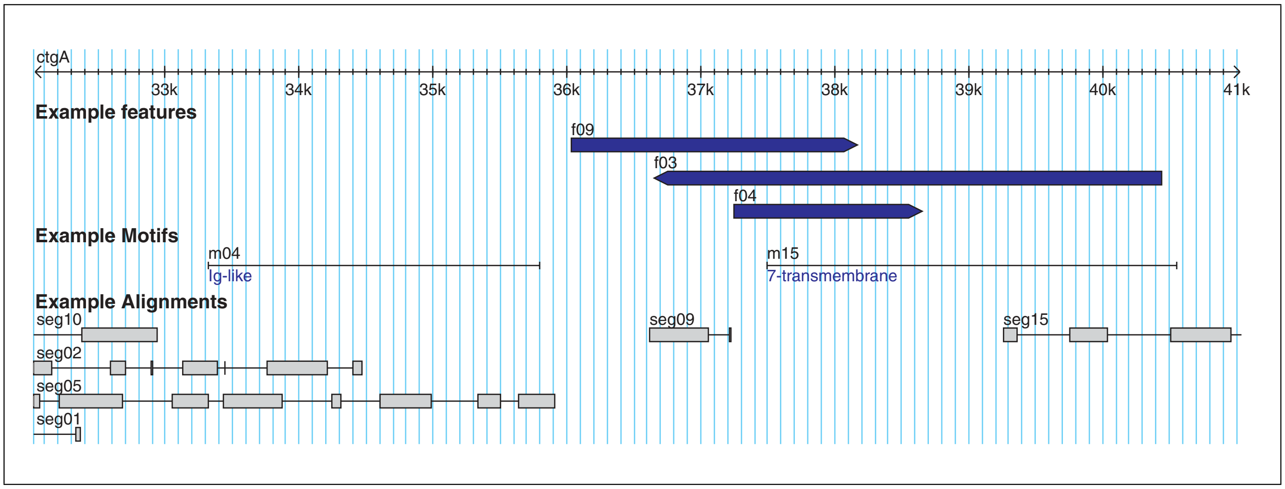


Figure 9.9.8 Turning on the `match` aggregator in the configuration file displays the match feature with its subparts correctly.

Using aggregators

There are several predefined aggregators, each of which expects particular combinations of feature type names. Table 9.9.2 summarizes the most useful ones.

To use any of these aggregators, follow this recipe: (1) Give features and their subparts the specific type names expected by the aggregators. (2) Add the aggregator to the list of aggregators in the config file, e.g., `aggregators = match processed_transcript clone`. (3) In the appropriate track definition, use the aggregator's name as the argument for `feature`. For example: `feature = processed_transcript`.

The user can also define custom aggregators, though that is beyond the scope of this protocol. Refer to the `CONFIGURE_HOWTO` file for more details on configuring custom aggregators.

Showing protein coding genes

GBrowse can display protein-coding genes in various shapes and styles. Use the `processed_transcript` aggregator and its companion glyph, also called `processed_transcript`, to set this up. The user can see how this is done by

Table 9.9.2 Summary of the Predefined Aggregators for Displaying Segmented Features

Aggregator name	Main type	Subtype(s)	Purpose
Alignment	(none)	Similarity	Used for nucleotide and protein alignments where the full extent of the match is unknown.
Coding	mRNA	CDS	Used in concert with the “cds” glyph to display the reading frame used by the coding portion of each exon.
Clone	(none)	Clone_left_end Clone_right_end	Used for cases in which clone ends have been mapped to the genome, but one of the ends may be missing.
Match	Match	Similarity, HSP	Used for nucleotide and protein alignments.
processed_transcript	mRNA	CDS, UTR, 5′-UTR, 3′-UTR, transcription_start_site, polyA_site	This is used to display the canonical spliced gene.
Transcript	Transcript	Exon TSS PolyA	This is used for a spliced transcript with exon features.

```
ctgA    example gene    1050    9000    .    +    .    Gene EDEN ; Note "protein kinase"

ctgA    example mRNA    1050    9000    .    +    .    mRNA EDEN.1 ; Gene EDEN
ctgA    example 5'-UTR  1050    1200    .    +    .    mRNA EDEN.1
ctgA    example CDS     1201    1500    .    +    0    mRNA EDEN.1
ctgA    example CDS     3000    3902    .    +    0    mRNA EDEN.1
ctgA    example CDS     5000    5500    .    +    0    mRNA EDEN.1
ctgA    example CDS     7000    7608    .    +    0    mRNA EDEN.1
ctgA    example 3'-UTR  7609    9000    .    +    .    mRNA EDEN.1

ctgA    example mRNA    1050    9000    .    +    .    mRNA EDEN.2 ; Gene EDEN
ctgA    example 5'-UTR  1050    1200    .    +    .    mRNA EDEN.2
ctgA    example CDS     1201    1500    .    +    0    mRNA EDEN.2
ctgA    example CDS     5000    5500    .    +    0    mRNA EDEN.2
ctgA    example CDS     7000    7608    .    +    0    mRNA EDEN.2
ctgA    example 3'-UTR  7609    9000    .    +    .    mRNA EDEN.2

ctgA    example mRNA    1300    9000    .    +    .    mRNA EDEN.3 ; Gene EDEN
ctgA    example 5'-UTR  1300    1500    .    +    .    mRNA EDEN.3
ctgA    example 5'-UTR  3000    3300    .    +    .    mRNA EDEN.3
ctgA    example CDS     3301    3902    .    +    0    mRNA EDEN.3
ctgA    example CDS     5000    5500    .    +    1    mRNA EDEN.3
ctgA    example CDS     7000    7600    .    +    1    mRNA EDEN.3
ctgA    example 3'-UTR  7601    9000    .    +    .    mRNA EDEN.3
```

Figure 9.9.9 The contents of the file `volvox4.gff`.

examining the file `volvox4.gff`, which defines a gene named EDEN, and its three spliced forms named `EDEN.1`, `EDEN.2`, and `EDEN.3`. The contents of the file can be seen in Figure 9.9.9.

The first line of the file seen in Figure 9.9.9 defines the gene as a whole, starting at position 1050 of ctgA and extending to position 9000. Following this, there are three sets of lines that define the structure of the spliced forms `EDEN.1`, `EDEN.2`, and `EDEN.3`.

By convention, the whole transcript is represented as type "mRNA." It has subparts named `5'-UTR`, `CDS`, and `3'-UTR`, where the UTR features are the 5′ and 3′ untranslated regions, respectively, and CDS is the coding region. Note how the CDS is split by splicing among multiple discontinuous locations on the reference sequence. The UTRs can be split in this way too. Each mRNA and its subparts are grouped together under a common name in the ninth column (`mRNA EDEN.1`, `mRNA EDEN.2`, etc.). In addition, each mRNA has a Gene *attribute* that ties it to the EDEN gene itself (`Gene EDEN`). Although this is not required for the display, doing this will identify the various alternative transcripts as belonging to the same gene should the user wish to use the GBrowse database for data mining. It will also show the user to which gene the transcript belongs to when he or she clicks on it for details.

If the user prefers not to distinguish between 5′ and 3′ UTRs, simply use "UTR" as the type. If the user does not know where the UTRs are, just leave them blank. If the user would rather think in terms of exons and introns, then use the `transcript` aggregator and its corresponding `transcript` glyph.

To set up the protein coding genes for GBrowse:

36. Copy the `volvox4.gff` to the database directory.

37. Open the `volvox.conf` file and change the aggregators line to read as follows:

```
aggregators = match
processed_transcript
```

The indent is important as it allows the command to continue across multiple lines in the configuration file.

38. Add the following new stanza to the bottom of the file:

```
[Transcripts]
feature = processed_transcript
glyph = processed_transcript
bgcolor = peachpuff
description = 1
key = Protein-coding genes
```

The updated aggregators option loads the `processed_transcript` *aggregator, which knows how to put CDS and UTR features together to form a spliced transcript. The new* `Transcripts` *track associates aggregated* `processed_transcript` *features with the like-named* `glyph`*, sets its background color to* `peachpuff`*, turns on the description lines, and sets the human readable track name to* `Protein-coding genes`.

The aggregators option demonstrates that GBrowse config file options can continue across multiple lines provided that each additional line is indented.

39. Save the config file, reload the page, and turn on the `Protein-coding genes` track.

40. If an image under the `Protein-coding genes` track is not seen, change the text in the "Landmark or Region" textbox to: `ctgA:1..10000` and click the "Search" button.

An image underneath the `Protein-coding genes` *track similar to that shown in Figure 9.9.10 should be seen.*

The image in Figure 9.9.10 can be improved by displaying the gene descriptions. The gene description (the Note in the EDEN GFF line) is not displayed because the description is attached to the gene and not to the individual mRNAs. To fix this, tell GBrowse to display features of type `gene`*, as well as those of type* `processed_transcript`.

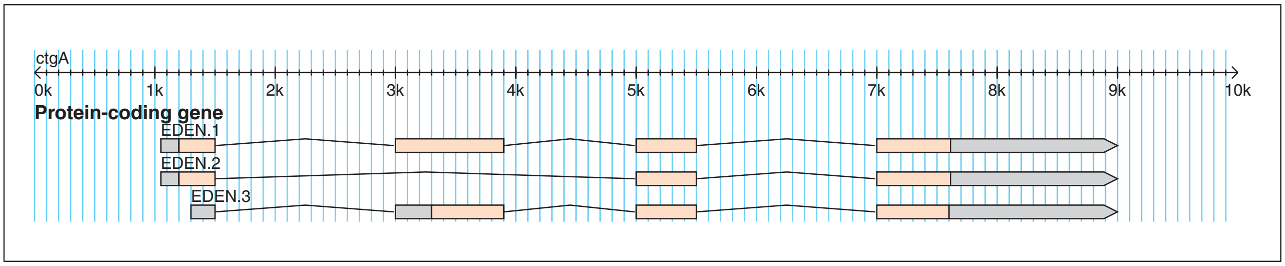


Figure 9.9.10 An image of the canonical `processed_transcript` glyph.

41. Modify `volvox.conf` so the last stanza looks like this:

```
[Transcripts]
feature          = processed_transcript gene
glyph            = processed_transcript
bgcolor          = peachpuff
description      = 1
key              = Protein-coding genes
```

The only change is that there are now ***two*** *types listed for the feature option,* `processed_transcript` *and* `gene`*. This tells GBrowse to place both feature types in the same track.*

42. Save the file and reload the page in the browser.

Now the display should render a glyph of the gene above the spliced transcripts and include the description of the gene, "protein kinase."

The `processed_transcript` *glyph can be customized using the options listed in Table 9.9.3.*

As an example of how the `processed_transcript` *glyph can be customized, use the options listed in Table 9.9.3 to make the track look like the UCSC Genome Browser, located at: http://genome.ucsc.edu.*

43. Open the volvox.conf file and change the `Transcripts` stanza to the following options.

```
[Transcripts]
Feature          = processed_transcript gene
Glyph            = processed_transcript
Height           = 8
bgcolor          = black
utr_color        = black
thin_utr         = 1
decorate_introns = 1
description      = 1
key              = Protein-coding genes
```

44. Save the file and reload the page in the browser. The image should look like Figure 9.9.11.

Showing the reading frame

Continuing with the example from the last section, it is known that the third exon of EDEN.1 is shared with EDEN.3. How does one tell if the reading frame is preserved? One can use the `coding` aggregator together with the `cds` glyph to create a display that will display the reading frame for each CDS.

Table 9.9.3 Configuration Options for the `process_transcript` Glyph

Option name	Possible values	Description
Thin_utr	0 (false), 1 (true)	If true, makes UTRs half the height of the exon.
utr_color	A color name (“gray” by default)	Changes the UTR color.
decorate_introns	0 (false), 1 (true)	If true, puts little arrowheads on the introns to indicate direction of transcription.

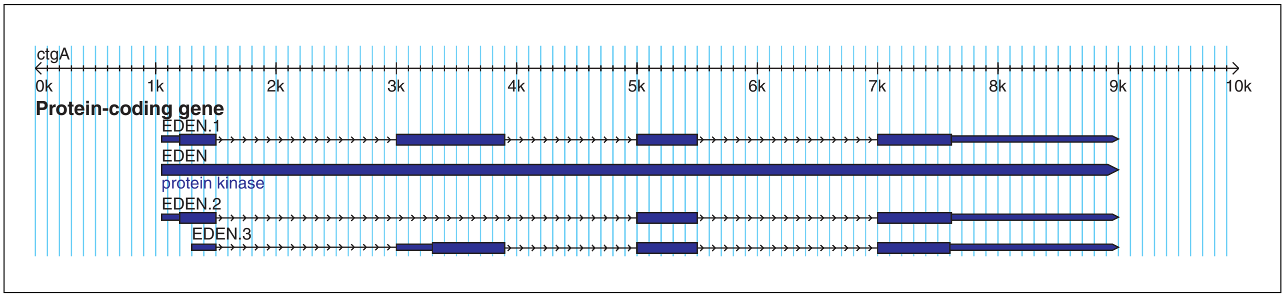


Figure 9.9.11 Image of a UCSC Genome Browser look-alike `processed_transcript` glyph.

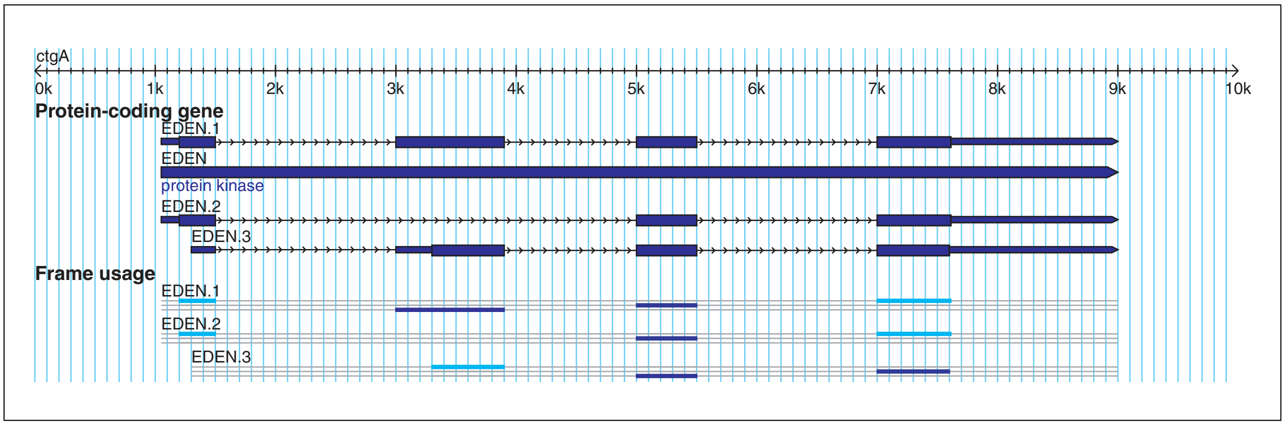


Figure 9.9.12 The `cds` glyph shows the reading frame using a musical staff notation.

45. Open the `volvox.conf` file and add the predefined `coding` aggregator to the list of aggregators:

```
aggregators = match
processed_transcript
coding
```

The `coding` *aggregator is similar to* `processed_transcript`*, except that it only pays attention to the CDS parts of the transcript. It was designed to work with the* `cds` *glyph to display the reading frame.*

46. Add the following short stanza to the bottom of the configuration file:

```
[CDS]
feature = coding
glyph = cds
key = Frame usage
```

47. Save the file. Reload the page and turn on the CDS track.

A “musical staff” representation of the frame usage as shown in Figure 9.9.12 should now be seen. From this display, it can be seen that the alternative splicing does change the reading frame of the third exon in EDEN.3 compared to EDEN.1 and EDEN.2.

SETTING UP A MySQL-BASED GBrowse

BASIC PROTOCOL 2

GBrowse can either use file-based data or connect to a relational database. For datasets with fewer than 20,000 features, a file-based data system works quite well. For larger datasets and for ease of data management, one will want to use a relational database management system. GBrowse supports MySQL, PostgreSQL, Oracle, and BioSQL database systems. For this protocol, one will set up a MySQL database using real, though out of date, data from the yeast genome, *S. cerevisiae*. MySQL is a good choice as it has wide acceptance in the bioinformatics community, is open-source, and is capable of handling large amounts of data with excellent performance.

Necessary Resources

Hardware

Unix (Linux, Solaris or other variety) workstation or Macintosh with OS X 10.2.3 or higher
A minimum of 500 Mb RAM
Internet connection

Software

Upon completion of Support Protocol 1 all necessary software should be installed

Files

In the Generic-Genome-Browser-1.X/sample_data/ directory: `yeast_data.gff`
In the Generic-Genome-Browser-1.64/contrib/conf/ directory: `01.yeast.conf`
Download the yeast genome sequence data from GMOD section of Sourceforge at: *http://prdownloads.sourceforge.net/gmod/yeast.fasta.gz*

Create database and set permissions

1. Create an empty MySQL database called `yeast` using the following commands:

```
$ mysql -u root -p
Enter password: *******
```

Depending on how MySQL was installed, a password may or may not be needed for the MySQL root user.

```
mysql> create database yeast;
Query OK, 1 row affected (0.01 sec)
```

2. Set the permissions for the yeast database so that data can be loaded into the database and Web users can access the database through GBrowse.

```
mysql> grant all privileges on yeast.* to
user@localhost with password pass;
```

This user can load the database with data.

```
Query OK, 0 rows affected (0.01 sec)
mysql> grant select on yeast.* to apache@localhost;
Query OK, 0 rows affected (0.00 sec)
```

This user can only select or view data from the database.

```
Mysql> quit
Bye
```

When granting "all" privileges to the user who will be able to load data into the database, replace `user` *with your own login or the username of the person using the loading programs. Likewise, when granting select privileges to the Web server program, use the* *login name for the Web server. The Web user on RedHat systems is* `apache`*; on other systems, it may be* `nobody`*,* `httpd`*,* `www-data`*, or something else. Consult your system documentation and change* `apache` *to the default Web user name used on the system being used.*

Load the data into database

3. Uncompress the `yeast.fasta.gz` file and move `yeast.fasta` into the `/sample_data/` directory.

4. Rename `01.yeast.conf` to `yeast.conf` and copy `yeast.conf` into the `/sample_data/` directory so that all three files needed for this protocol are in the same location.

5. Load the yeast data into the database using the following command:

```
bp_load_gff.pl -c -d yeast yeast.fasta yeast_data.gff
```

6. If you receive an error message stating that you do not have sufficient privileges to upload the data, try the following command:

```
bp_load_gff.pl -u user -p password -c -d yeast
  yeast.fasta yeast_data.gff
```

Substitute `user` and `password` with the login and password that you granted “all” privileges to in the previous section.

This will likely return a series of seven warning messages starting with `Unknown table`, followed by:

```
yeast_data.gff: loading...
yeast_data.gff: 13298 records loaded
Loading fasta file yeast.fasta
yeast.fasta: 17 records loaded
```

The script, `bp_load_gff.pl`*, is a BioPerl script that is installed in the* `/usr/bin` *or* `/usr/local/bin` *directory when BioPerl-1.5.1.is installed. The switch* `-c` *tells the script to initialize the database. The switch* `-d yeast` *tells the script to use the* `yeast` *database. The* `yeast.fasta` *contains the sequence data for the 16 chromosomes plus the mitochondria. The* `yeast_data.gff` *file contains the annotations or features for the yeast genome. See the Troubleshooting section if the database loading did not go as described above.*

Move and modify the configuration file

7. Copy the configuration file into the gbrowse.conf:

```
$ cd /etc/httpd/conf/gbrowse.conf/
$ cp /path_to_sample_data/yeast.conf
```

8. Open `yeast.conf` in a text editor and note the following lines:

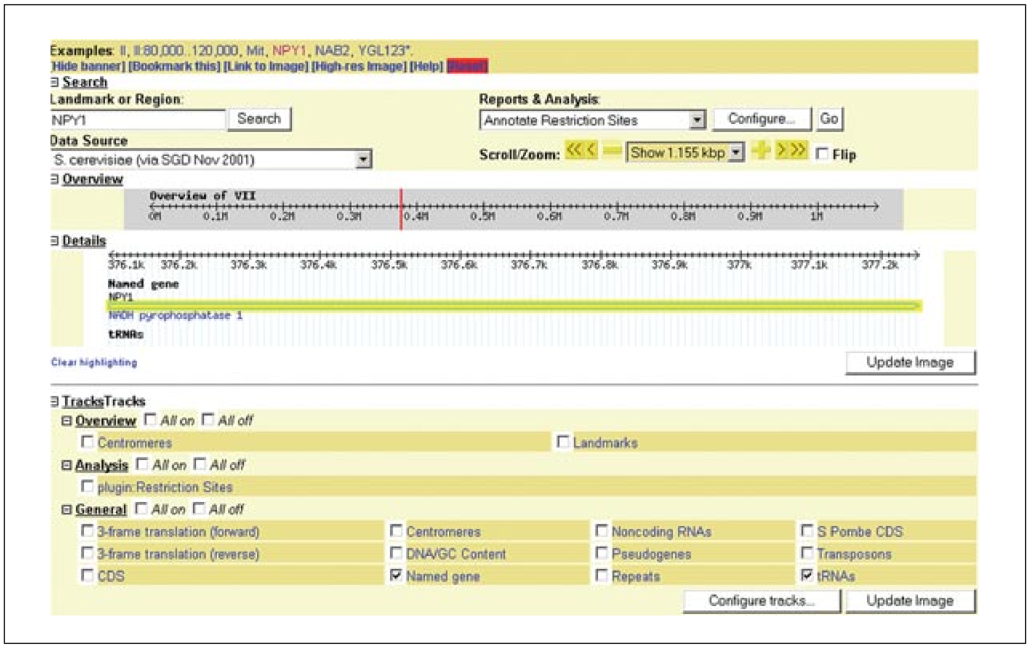
```
db_args = -adaptor dbi::mysql
-dsn dbi:mysql:databases=yeast;host=localhost
```

These two lines tell GBrowse how to connect to the database (using the `dbi::mysql` *adapter) and to connect to the database named* `yeast`*. Be sure to use two colons in the adaptor argument and a series of single colons for the* `-dsn` *argument.*

9. Change the following line to match the name of the Web user on the system being used:

```
user = apache
pass =
```

Leave `pass =` *to nothing, otherwise Web users will be unable to connect to the data.*


Examples: II, II:80,000..120,000, Mit, NPY1, NAB2, YGL123*.
[Hide banner] [Bookmark this] [Link to Image] [High-res Image] [Help] [Reset]

⊟ Search

Landmark or Region: NPY1 [Search]

Data Source: S. cerevisiae (via SGD Nov 2001)

Reports & Analysis: Annotate Restriction Sites [Configure...] [Go]

Scroll/Zoom: Show 1.155 kbp □ Flip

⊟ Overview

⊟ Details

Clear highlighting [Update Image]

⊟ TracksTracks

⊟ Overview □ All on □ All off
□ Centromeres □ Landmarks

⊟ Analysis □ All on □ All off
□ plugin:Restriction Sites

⊟ General □ All on □ All off
□ 3-frame translation (forward) □ Centromeres □ Noncoding RNAs □ S Pombe CDS
□ 3-frame translation (reverse) □ DNA/GC Content □ Pseudogenes □ Transposons
□ CDS ☑ Named gene □ Repeats ☑ tRNAs

[Configure tracks...] [Update Image]

Figure 9.9.13 GBrowse detailed view with default viewing options for yeast genome.

10. Save the file and open the newly created page by pointing the browser to: *http://localhost/cgi-bin/gbrowse/yeast/*.

11. Click on the example link "NPY1" and an image similar to that shown in Figure 9.9.13 should appear.

    Initially just the named genes and tRNA tracks are turned on. Turn on other tracks and zoom out to see the types of annotation data that can be displayed with GBrowse. An up-to-date version of the S. cerevisiae genome can be accessed at http://db.yeastgenome.org/cgi-bin/gbrowse/yeast/.

    If an error is seen when opening the GBrowse window, confirm select privileges have been given to the correct Web user on the system being used. Confirm that the `-dsn` *specifies the correct database.*

12. Try clicking on the example links chrII, chrII:80,000..120,000, or YGL123*.

    Any one of these three should give an error similar to `The landmark named chrII is not recognized. See the help pages for suggestions.` *This error is caused by two different issues. One is inconsistent nomenclature between the* `01.yeast.conf` *file installed with GBrowse and the* `yeast_data.gff` *file. The second is that YGL123* represents an ORF, which is not listed as one of the automatic classes to display. Both problems can be fixed by making the following edits to the* `yeast.conf` *file.*

13. Open the `yeast.conf` file and note the line beginning with `#examples to show in the introduction`. Below that is a line that starts `example = chrII`.

14. Open the `yeast_data.gff` file and note that the name of the reference chromosome sequences are listed as `I, II`, etc.; not `chrI, chrII`.

15. In the `yeast.conf` file, change `example = chrII` to `example = II`.

16. Change `chrII:80,000..120,000` to `II:80,000..120,000`.

17. Change `automatic classes = Symbol Gene Clone` to `automatic classes = Symbol Gene Clone ORF`.

18. Save the configuration file and reload the GBrowse Web page. All of the example links should now work.

SUPPORT PROTOCOL 1

INSTALLING GBrowse IN THE UNIX/LINUX ENVIRONMENT

This protocol describes the installation of GBrowse in a Unix/Linux environment. It is assumed that the user has a working knowledge of Unix and has root privileges to install software.

Necessary Resources

Hardware

Unix (Linux, Solaris or other variety) workstation or Macintosh with OS X 10.2.3 or higher
A minimum of 500 Mb RAM
Internet connection

Software

Standard software:

Perl 5.8.6 or higher. Perl will generally be installed on most machines with a Unix-variant operating system, but is available from *http://ww.perl.org* if not already installed.

MySQL, PostgreSQL, or other relational database. This protocol will use MySQL. If MySQL is not already installed, instructions for obtaining and installing MySQL can be found at *http://dev.mysql.com/doc/refman/5.0/en/installing.html*. A user account with "create databases" privileges should be available.

Apache Web server: Available at *http://httpd.apache.org/download.cgi*.

Nonstandard software (in all cases, use the latest version available):

BioPerl 1.6 or higher (*http://www.bioperl.org*)
GD 2.07 or higher (*http://search.cpan.org/~lds/GD-2.32/*)
CGI (*http://search.cpan.org/~lds/CGI.pm-3.45/*)
CGI::Session (*http://search.cpan.org/~markstos/CGI-Session-4.42/lib/CGI/*)
DBI (*http://search.cpan.org/~timb/DBI-1.6/*)
DBD::mysql (*http://search.cpan.org/~capttofu/DBD-mysql-4.012/*)
Digest::MD5 (*http://search.cpan.org/~gaas/Digest-MD5-2.39/*)
Text::Shellwords (*http://search.cpan.org/~lds/Text-Shellwords-1.08/*)
If these links do not work, search the Perl module by name at *http://search.cpan.org*.

Files

The `INSTALL` and `README` files for the protocol are located in the `Generic-Genome-Browser-1.X/` directory after unpacking the Generic Genome Browser tar file.

Download and install GBrowse

1. Download the required Perl modules and install using the method of choice.

   GD may not install correctly on some versions of RedHat or SuSE Linux if Perl was pre-installed or installed as an RPM. It may be necessary to install Perl from the source code in order for GD to install correctly. See the `README.unix` *file found in the GD install directory for more information about this issue.*

2. Download GBrowse from either the home page for the Generic Model Organism Database project (*http://www.gmod.org*) or directly from SourceForge (*http://sourceforge.net/projects/gmod/files/*. Use the latest version of GBrowse available. The download will be a `.tar.gz` file, which must be uncompressed and unpacked before installation.

   As of this writing, the most current version was 1.70. Read the `INSTALL` *and* `README` *files for any updated information.*

3. Install GBrowse from source by running the following commands:

   ```
   $ perl Makefile.PL
   $ make
   $ make test (recommended, but optional)
   $ make install UNINST=1
   ```

   The `$` *represents a command line prompt. The prompt may be represented as a* `#`, `$`, *or* `%`, *depending on the system.*

   This will install the software in the default location under `/usr/local/apache`. *The GBrowse installation includes a number of Perl modules. The* `UNINST=1` *checks for older versions of any of these Perl modules and removes them before installing the newer versions to prevent conflicts.*

   The default locations for the files installed are:

CGI script:	`/usr/local/apache/cgi-bin/gbrowse`
Static HTML files:	`/usr/local/apache/htdocs/gbrowse`
Config files:	`/usr/local/apache/conf/gbrowse.conf`
The module:	*standard site-specific Perl library location*

   The default locations may not work on the system being used, but the location of the installation can be changed by passing to the `Makefile.PL` *one or more* `NAME=VALUE` *pairs. For example, on a RedHat Linux system, the CGI and HTML files are located in the* `/var/www/cgi-bin` *and* `/var/www/html` *directories, respectively. The conf directory is* `/etc/httpd/conf`. *For a RedHat installation, specify the following configuration:*

   ```
   $ perl Makefile.PL HTDOCS=/var/www/html \
   $ CONF=/etc/httpd/conf \
   $ CGIBIN=/var/www/cgi-bin
   ```

   The backslash is there only to split the command across multiple lines.

   The following arguments are recognized by the `Makefile` *script:*

`CONF:`	*Configuration file directory*
`HTDOCS:`	*HTML static files directory*
`CGIBIN:`	*CGI script directory*
`APACHE:`	*Base directory for Apache conf, htdocs, and cgi-bin directories*
`LIB:`	*Perl site-specific modules directory*
`BIN:`	*Perl executable script directory*
`NONROOT:`	*Set to 1 to install GBrowse in a way that does not require root access*

   The configuration files are always placed in a subdirectory named `gbrowse.conf`. *This cannot be changed. Similarly, HTML static files are placed in a directory named* `gbrowse`. *More information about the installation process is available in the* `INSTALL` *document located in the installation directory.*

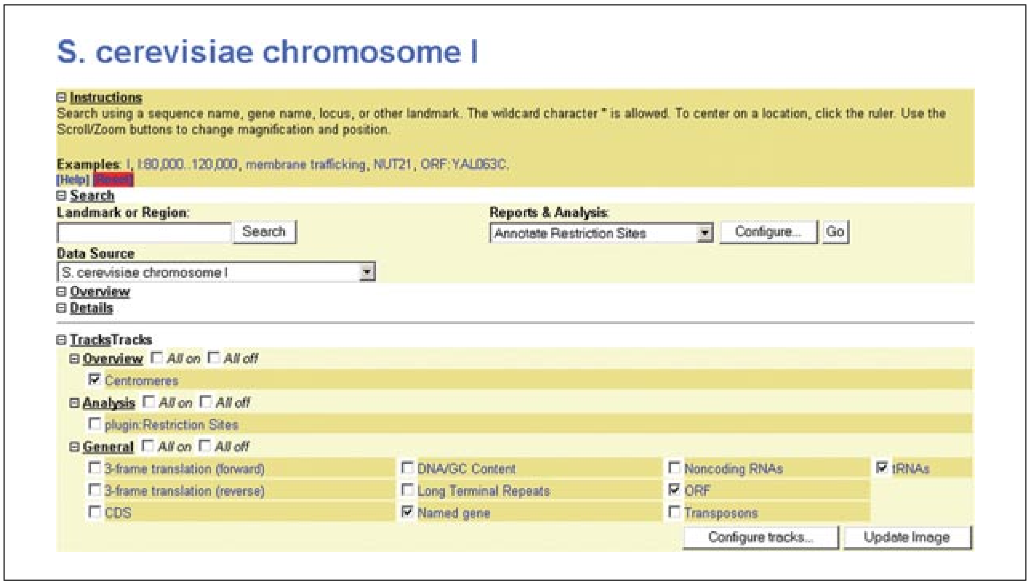
4. Test the installation.

Check that the installation went correctly by opening the following page in the browser of preference:

```
http://localhost/cgi-bin/gbrowse
```

If accessing the page from a different computer, replace `localhost` with the name of the Web server on which Gbrowse is installed. If the installation worked, a page should open titled “Generic Genome Browser.” On that page is a list of directory paths where the various components of GBrowse were installed. This page can be used a reference to locate the different components. There is a link to an example database based on “yeast chromosome 1.” Click on that link and the GBrowse page should open, as shown in Figure 9.9.14.

Click on the NUT21 link and it should open a page as shown in Figure 9.9.15.



S. cerevisiae chromosome I

⊟ Instructions
Search using a sequence name, gene name, locus, or other landmark. The wildcard character * is allowed. To center on a location, click the ruler. Use the Scroll/Zoom buttons to change magnification and position.

Examples: I, I:80,000..120,000, membrane trafficking, NUT21, ORF:YAL063C.
[Help]

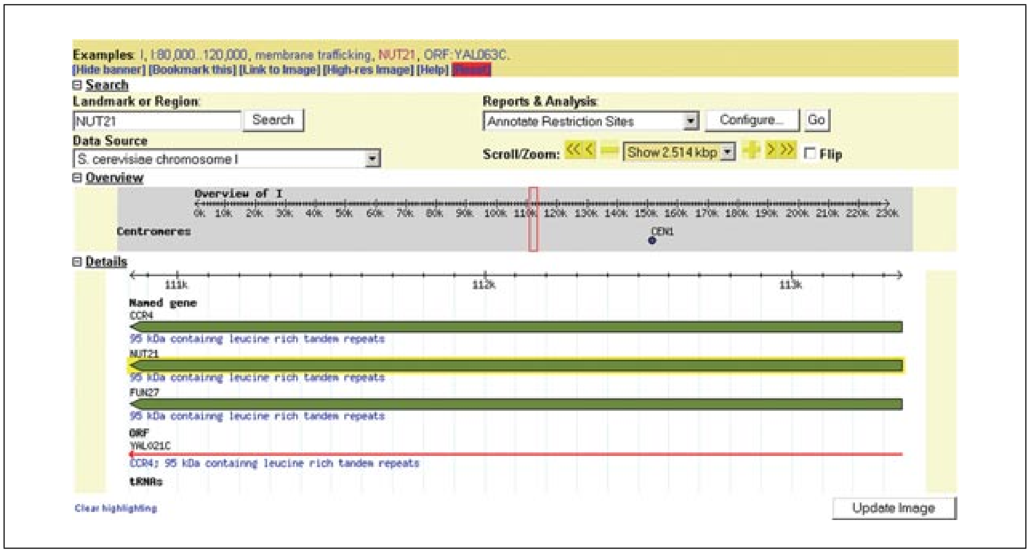
⊟ Search
Landmark or Region: [ ] Search
Reports & Analysis: Annotate Restriction Sites ▾ Configure... Go
Data Source: S. cerevisiae chromosome I ▾

⊟ Overview
⊟ Details

⊟ TracksTracks
⊟ Overview □ All on □ All off
☑ Centromeres
⊟ Analysis □ All on □ All off
□ plugin:Restriction Sites
⊟ General □ All on □ All off
□ 3-frame translation (forward) □ DNA/GC Content □ Noncoding RNAs ☑ tRNAs
□ 3-frame translation (reverse) □ Long Terminal Repeats ☑ ORF
□ CDS ☑ Named gene □ Transposons

Configure tracks... Update Image

Figure 9.9.14 Initial page view for the `yeast_chr1` sample database installed with GBrowse.



Examples: I, I:80,000..120,000, membrane trafficking, NUT21, ORF:YAL063C.
[Hide banner] [Bookmark this] [Link to Image] [High-res Image] [Help]

⊟ Search
Landmark or Region: NUT21 Search
Reports & Analysis: Annotate Restriction Sites ▾ Configure... Go
Data Source: S. cerevisiae chromosome I ▾
Scroll/Zoom: << < − Show 2.514 kbp ▾ + > >> □ Flip

⊟ Overview
Overview of I
0k 10k 20k 30k 40k 50k 60k 70k 80k 90k 100k 110k 120k 130k 140k 150k 160k 170k 180k 190k 200k 210k 220k 230k
Centromeres
CEN1

⊟ Details
111k
112k
113k
Named gene
CCR4
95 kDa containng leucine rich tandem repeats
NUT21
95 kDa containng leucine rich tandem repeats
FUN27
95 kDa containng leucine rich tandem repeats
ORF
YAL021C
CCR4; 95 kDa containng leucine rich tandem repeats
tRNAs

Clear highlighting Update Image

Figure 9.9.15 Detailed view for the Nut21 link from `yeast_chr1` database.

If the two pages seen are those shown in Figures 9.9.14 and 9.9.15, Gbrowse has been successfully installed.

If at this point an error message is received, record the error message. See the Troubleshooting section at the end of this unit for suggestions on how to determine why a particular GBrowse installation is not behaving correctly and how to seek help from the GBrowse community.

SETTING UP A DATABASE FROM NCBI GENBANK FILES

SUPPORT PROTOCOL 2

Almost all genome sequence data submitted to Genbank by individual research groups or genome project teams are available with the associated annotation from the genomes division of the NCBI nucleotide database. While the NCBI annotation team has provided GFF3 formatted files for all genomes, the default Genbank `conf_file` provided in the `Gbrowse1.X/contrib/` directory assumes that you will create the GFF file from a Genbank file. There are BioPerl scripts installed as part of the BioPerl installation that allow users to create convert NCBI Genbank files into GFF files suitable for uploading into a GBrowse database (v. 1.69 and higher). These can be used to set up a GBrowse display of almost any sequenced genome or part of a genome. This protocol will take the user through the process of setting up a MySQL database from Genbank genome files. The Genbank files can be obtained from *ftp://ftp.ncbi.nih.gov/genomes/*. The sequences are generally split into chromosomes or large contigs, depending on the type of data available for a particular organism. For this example, the genome from *Deinococcus radiodurans*, an extremely radiation-resistant bacteria, whose genome was published in 1999 will be used. The genome consists of two chromosomes and two plasmids, represented on four separate Genbank files.

Necessary Resources

Hardware

Unix (Linux, Solaris or other variety) workstation or Macintosh with OS X 10.2.3 or higher.
A minimum of 500 Mb RAM
Internet connection

Software

All necessary software should be installed if Support Protocol 1 has been completed

Files

From *ftp://ftp.ncbi.nih.gov/genomes/Bacteria/Deinococcus_radiodurans/* download the following 4 files:
`NC_000958.gbk`
`NC_000959.gbk`
`NC_001263.gbk`
`NC_001264.gbk`
In the `Generic-Genome-Browser-1.x/contrib/conf/` directory:
`08.genbank.conf`
A sample, edited version of the `dradians.conf` is available for download from *http://www.currentprotocols.com/protocol/bi0909*

1. Download the Genbank file and convert to GFF3 format1.

2. To simplify loading the database, you can concatenate the four Genbank data files into one Genbank file:

```
$cat *.gbk > dradians.gbk
```

The Genbank file contains the accession numbers for the Gene database records associated with the annotated genes. As there is a lot of data that is put into this record, it would be convenient to create a link to that record from the GBrowse page. This can be accomplished through the db_xref links, but to simplify adding external links out of GBrowse, you can also edit the Genbank file so that references to specific NCBI databases are loaded into the attributes table with a unique feature name.

3. Open the Genbank file in a text editor. In the FEATURES section, there is a gene category, which lists a locus_tag for each gene in the *D. radiodurans* genome. There is also a db_xref="GeneID:#######". This is the id number of the Gene database record. Using Find/Replace in your text editor or GREP, replace the db_xref tag with the following:

   ```
   Find: db_xref="GeneID:
   Replace with: geneid="
   ```

   For each gene in the Genbank file, there should now be 2 identifiers:

   /locus_tag="DR_X####"

   /gene_id="#######"

   These will be loaded into the attribute table as named features.

4. Convert the Genbank formatted file into a GFF file:

   ```
   $ bp_genbank2gff3.pl dradians.gbk
   ```

   You will see a series of statements `#working on region:NC_000958..` *as it works through each of the 2 chromosomes and 2 plasmids.*

   The final output will be GFF3; save the file to `./path/dradians.gbk.gff`.

5. Open the `dradians.gbk.gff` file in a text editor. The first few lines should read:

   ```
   ##gff-version 3
   #sequence-region NC_000958.1 177466
   #conversion-by bp_genbank2gff3.pl
   #organism Deinococcus radiodurans R1 ect
   ```

6. The data in the 9th or attributes column of the GFF file are loaded into the attributes table. Here you can see the identifiers of the various features. As an example, the first gene on the plasmid MP1 is:

   ```
   ID=DR=B0001; geneid=1799842; locus_tag=DR_B0001
   ```

Create MySQL database and load the GFF file

7. Create a MySQL database called `dradians`.

8. Grant the appropriate privileges as described in Basic Protocol 2 for setting up a MySQL database.

9. Use the `bp_bulk_load_gff.pl` script to load the gff file into the newly created database.

   ```
   $bp_bulk_load_gff.pl -u user -p password -d dradians
     dradians.gbk.gff
   ```

   The output on the screen should state that this operation would delete all existing data in the database `dradians` and ask if you want to continue. Type `y`. If all works properly, 12921 features should have successfully loaded.

Modify the configuration script

It is necessary to move the generic configuration file into the appropriate directory and modify the connection parameters and user-specific settings so it will work with the *D. radiodurans* database that was created.

10. Change the name of `08.genbank.conf` to `dradians.conf`.

11. Move `dradians.conf` to your `gbrowse.conf` directory. On the author's Red-Hat system it is: `/etc/httpd/conf/gbrowse.conf/`.

12. Open `dradians.conf` in a text editor and change the `-dsn` line to reflect the name of the database:

```
-dsn    dbi:mysql:database=dradians;host=localhost13
```

Change `user = nobody` to `user = apache`. Save the file.

13. Bring up the page in the browser by opening it at *http://localhost/cgi-bin/gbrowse/dradians/*.

    The page still looks generic, as the `dradians.conf` *file has not been changed to reflect the name of the organism being displayed or the example searches available.*

14. Change `description = Genbanks Tests` to `description = D. radiodurans genome`.

15. Change `examples = NC_001320 AP003256` to read `examples = DR_B0015 DR_C0024 NC_000958`.

16. Save the file and reload the page in the browser.

17. Try the example searches.

    It may be that the DR_B0015, DR_C0024, and NC_000958 links work, but clicking on NC_001263 gives an error: `Detailed view is limited to 1 Mbp. Click in the overview to select a region 100 kbp wide.` *Click in the salmon-colored overview box and the display adjusts to show 100 kbp; many annotated features will be seen. There is much more that can be modified, but this protocol gives the user a sense of how to utilize the large amount of genomic data and annotation available from NCBI.*

18. You can add an external link to the NCBI Gene by incorporating a call-back subroutine in the GENE track configuration. Replace `description = 1` with the following code:

```
description = sub {
my $feature = shift;
return $feature->attributes('gene_id')
}
link = sub {
my $feature = shift;
my ($geneid) = $feature->attributes('gene_id');
return
http://www.ncbi.nlm.nih.gov/sites/entrez?db=
  gene&cmd=search&term=$geneid;
}
```

This should create a link below the track for Annotated Genes with the GeneID number shown. If you click on this number, it should take you to the corresponding record in the Gene database. If this link does not appear to be active, try clicking in the Reset link in the instructions section at the top of the page.

COMMENTARY

Background Information

Over 200 eukaryotic and over 900 prokaryotic genomes have been fully sequenced and published as of August 2009. The sequence data and its associated annotation are uploaded to one of three public sequence repositories, all of which have sophisticated search interfaces for querying and accessing the data. A subset of the sequence data are also represented in a genome browser at the public sequence repositories. Genome browsers give users the ability to navigate genomic sequence information and visualize various features in a series of tracks within the context of a reference sequence. The data available in a genome browser vary, but any data that can be aligned to a reference sequence can be displayed. This allows users to quickly answer questions that depend on the sequence location, such as the proximity of genes to each other or the placement of microarray probes in alternatively spliced exons. There are several sites that maintain genome browsers for a number of organisms (Furey, 2006), but not all sequenced genomes are represented at these sites or by model organism databases such as Flybase (Grumbling and Strelets, 2006) or Wormbase (Schwarz et al., 2006).

Ideally, all genomes would have an associated database with a genome viewer, but the cost and effort of developing and maintaining a model organism database is prohibitive for most research groups. However, in 2000, four model organism databases agreed to pool resources to design and make available, free of charge, the software components necessary to support a model organism database. The project, called the Generic Model Organism Database (GMOD), released the first two components in 2002, the Apollo genome annotation editor (Lewis et al., 2002) and the Generic Genome Browser (Gbrowse; Stein et al., 2002). GBrowse is implemented as a series of CGI scripts, which can display in a browser an arbitrary set of features on a nucleotide or protein sequence. It can accommodate genome-scale sequences up to several megabases in length. GBrowse was designed from the outset to be portable and extensible. It is written entirely in Perl, a language widely used in bioinformatics, and runs on LINUX/UNIX, Mac OSX, and Windows PC platforms. It can be easily integrated with other datasets, either at the database level or through the use of URL links. It is not dependent on any particular data model or annotation pipeline and thus is suitable for any research group that needs to maintain a set of sequence annotations.

GBrowse2.0 is a complete rewrite of the original GBrowse that adds several major new features including:

1. User interface improvements.
2. Configuration changes to allow consolidation of most configuration options into one file, allowing the source specific configuration files to be more concise.
3. Support for multiple databases within a given data source.
4. Support for slave renderers so the tasks can be split across multiple processors and machines.

The final version of GBrowse2 will be released within a few months. It can be downloaded and installed using the CVS system. Fully functional, but possibly buggy, beta releases are available on CPAN (*http://www.cpan.org*) under the module name Generic-Genome-Browser.

Critical Parameters and Troubleshooting

The most useful resource for sorting out issues with GBrowse is the GMOD-GBrowse mailing list. One can subscribe to this mailing list at *https://lists.sourceforge.net/lists/listinfo/gmod-gbrowse*. This is a relatively low volume list, but the developers responsible for GBrowse and many other users regularly monitor it and provide feedback to help troubleshoot problems encountered when setting up or configuring GBrowse. You can also search archives of this list at *http://www.nabble.com/gmod-gbrowse-f3500.html*.

When an error message is received, always record the text of the message. Check the Web server error log file for other error messages, which can be critical to understanding the problem. The error log file, `error_log`, is located in the server log directory. Two common locations for the log files are: `/usr/local/apache/logs` or `/etc/httpd/logs`. Open the `error_log` file using a text editor and at the end of the file will be the most recent entries. Include the text of the errors in the email sent to contact the GBrowse mailing list for help.

The two most common errors encountered when loading databases using the `load_gff` scripts are not granting the user enough privileges to load data into the database, or, having granted privileges to user with a password, trying to run the `load_gff` script without a password.

Acknowledgments

The author would like to thank Lincoln Stein for allowing her to adapt the GBrowse online tutorial for use in this protocol. The author is also indebted to the members of the GMOD community who monitor the GMOD-GBROWSE mailing list and have answered numerous questions.

Literature Cited

Furey, T.S. 2006. Comparison of human (and other) genome browsers. *Hum. Genomics* 2:266-270.

Grumbling, G. and Strelets, V. 2006. FlyBase: Anatomical data, images and queries. *Nucleic Acids Res.* 34:D484-D488.

Lewis, S.E., Searle, S.M., Harris, N., Gibson, M., Lyer, V., Richter, J., Wiel, C., Bayraktaroglir, L., Birney, E., Crosby, M.A., Kaminker, J.S., Matthews, B.B., Prochnik, S.E., Smithy, C.D., Tupy, J.L., Rubin, G.M., Misra, S., Mungall, C.J., and Clamp, M.E. 2002. Apollo: A sequence annotation editor. *Genome Biol.* 3:0082.1-0082.14.

Schwarz, E.M., Antoshechkin, I., Bastiani, C., Bieri, T., Blasiar, D., Canaran, P., Chan, J., Chen, N., Chen, W.J., Davis, P., Fiedler, T.J., Girard, L., Harris, T.W., Kenny, E.E., Kishore, R., Lawson, D., Lee, R., Muller, H.M., Nakamura, C., Ozersky, P., Petcherski, A., Rogers, A., Spooner, W., Tuli, M.A., Van Auken, K., Wang, D., Durbin, R., Spieth, J., Stein, L.D., and Sternberg, P.W. 2006. WormBase: Better software, richer content. *Nucleic Acids Res.* 34:D475-D478.

Stein, L.D., Mungall, C., Shu, S., Caudy, M., Mangone, M., Day, A., Nickerson, E., Stajich, J.E., Harris, T.W., Arva, A., and Lewis, S. 2002. The generic genome browser: A building block for a model organism system database. *Genome Res.* 12:1599-1610.

Key Reference

Stein et al., 2002. See above.

This article gives extensive background and descriptive details on how and why the Generic Genome Browse was developed.

Internet Resources

http://gmod.org/wiki/Main_Page

Generic Model Organism Database (GMOD): Tutorials, overviews, and links for downloading GBrowse.

https://lists.sourceforge.net/lists/listinfo/gmod-gbrowse

GMOD-GBrowse mailing list: Join the mailing list to ask questions about installation and use of GBrowse and to be notified of new releases.

http://gmod.org/wiki/GBrowse_2.0_HOWTO

GBrowse 2.0 HOWTO: This site provides instructions on installing and configuring GBrowse 2. There are links to HOWTOs on migrating from GBrowse 1.X to 2.

http://www.nabble.com/gmod-gbrowse-f3500.html

Nabble: Search engine for various internet forums, including GMOD-Gbrowse

http://www.sanger.ac.uk/Software/formats/GFF/GFF_Spec.shtml

General Feature Format (GFF) specification: Get detailed information about the GFF and download scripts for converting various computational analyses to GFF format.

Supplemental File

Supplemental files can be downloaded from *http://www.currentprotocols.com/protocol/bi0909* by clicking the Supplemental Files tab and selecting the needed file.

`dradians.conf`

Configuration file for use with Support Protocol 2. This file is an edited version of 08.genbank.conf file that is installed with the gbrowse package.

Installing a Local Copy of the Reactome Web Site and Database

Imre Vastrik[1]

[1] EMBL-European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge, U.K.

ABSTRACT

The Reactome project builds, maintains, and publishes a database of biological pathways. The information in the database is gathered from the experts in the field, peer reviewed, and edited by Reactome editorial staff and then published to the Reactome Web site, *http://www.reactome.org* (see UNIT 8.7). Reactome software is open source and builds on top of other open-source or freely available software. Reactome data and code can be freely downloaded in its entirety and the Web site installed locally. This allows for more flexible interrogation of the data and also makes it possible to add one's own information to the database. *Curr. Protoc. Bioinform.* 21:9.10.1-9.10.9. © 2008 by John Wiley & Sons, Inc.

Keywords: pathways database • open source software

INTRODUCTION

The Reactome project, described in UNIT 8.7, builds, maintains, and publishes a database of biological pathways. The Reactome database contains a curated collection of well documented molecular reactions assembled into pathways ranging from intermediary metabolism through signal transduction to complex cellular events such as the cell cycle. The information in the database is gathered from the experts in the field, then peer reviewed and edited by Reactome editorial staff. It is then published to the Reactome Web site.

The Reactome Web site provides facilities to search and browse the database contents as well as to export the data in various formats. Reactions and pathways can be exported in BioPAX and SBML formats, and in automatically created diagrams in SVG format. The narrative in Reactome can be exported in RTF and PDF formats.

While the Reactome Web site provides free access to the data, the data can also be downloaded in their entirety and the Reactome software installed locally. The amount of data and code to be downloaded from the Reactome Web site in compressed format is around 100 Mb. This allows for more flexible interrogation of the data, and also makes it possible to add one's own information to the database. This unit describes setting up your own copy of Reactome database together with the Web site software for accessing and viewing the data.

The Basic Protocol describes the installation of a local copy of Reactome Web site and database, while Support Protocol 1 covers the installation of other software required to run the system.

INSTALLATION OF THE REACTOME WEB SITE AND DATA

BASIC PROTOCOL

This protocol describes how to set up a local copy of Reactome Web site and database.

Current Protocols in Bioinformatics 9.10.1-9.10.9, March 2008
Published online March 2008 in Wiley Interscience (www.interscience.wiley.com).
DOI: 10.1002/0471250953.bi0910s21

Necessary Resources

Hardware

Computer with Unix, Linux, or Mac OS X operating system, an Internet connection, and at least 800 MB free disk space. Processing power requirements depend on the planned use of the installation. The Reactome Web server has been run on computers with a single Pentium III or PowerPC G3 processor, but the response is slow. For personal or light group use, one "high-end" CPU (Xeon, Opteron, G5) is sufficient if the computer is dedicated to this task. A dual-processor or duo-core CPU will significantly improve response time. For reference, at the time of this writing, the Reactome public Web site was hosted on a machine with four 2.80 GHz Intel Xeon CPUs. You need to have root (administrative) access to the installation machine for the installation with these instructions to succeed.

Software

All the software apart from that for the BioMart functionality of the Web site is described in Support Protocols 1 to 3. Installation of BioMart software as well as creation of the appropriately formatted database is described at *http://wiki.reactome.org/index.php/Release#105_Upgrade_BioMart_and_Restart_the_Server*.

The wget utility is also needed; this may be part of your operating system distribution. If not, its source code and installation instructions can be obtained from *http://www.gnu.org/software/wget/*.

Download Reactome and configure code

1. Download the Reactome Web site and Perl code, unpack it, and rename it to `/usr/local/gkb`.

```
$ wget http://www.reactome.org/download/current/
  GKB.tar.gz
$ tar xvzf GKB.tar.gz
$ mv GKB /usr/local/gkb
```

You will probably need to log in as "root" (or use the `sudo` *command) in order to obtain sufficient privileges to create the* `/usr/local/gkb` *directory.*

2. Open `/usr/local/gkb/modules/GKB/Config.pm` with a text editor. Find the line:

```
$JAVA_PATH = '/usr/bin/java';
```

and change `/usr/bin/java` to reflect the correct path to the Java executable on your system (you can find out by issuing the command `which java` in the command-line shell.) Similarly, find and edit the line that sets `$WWW_USER` to reflect the user account under which the Web server runs.

On Mac OS X this is `www` *while on Unix/Linux it tends to be* `nobody`. *Consult your system documentation if you are unsure.*

This file also specifies the MySQL user name, password, port, and name of the database that the Web server is using to connect to the database server. Make sure that values of `$GK_DB_NAME, $GK_DB_USER` *and* `$GK_DB_PASS` *are as specified in the MySQL installation step in Support Protocol 1.*

Download and install Reactome data

3. Download the Reactome data as two MySQL database dumps. The first of those is the main database while the second is derived from the main database and supports the Web site's "skypainter" utility.

```
$ wget http://www.reactome.org/download/current/
  sql.gz
$ wget http://www.reactome.org/download/current/
  sql_dn.gz
```

4. Start the MySQL server if it is not running already. Note that the location of the `mysql directory` may be different if MySQL was preinstalled on your operating system. See your system documentation for help.

```
$ cd /usr/local/mysql
$ bin/mysqld_safe &
cd -
```

5. Connect to the database server with MySQL command line client and create empty databases. The name of the main database has to be the same as the value of `$GK_DB_NAME` in `Config`.pm. The name of the skypainter database should be the name of the main database with "`_dn`" appended. Replace `reactome_user` and `reactome_pass` with the username and password you chose when you created the MySQL database in Support Protocol 1.

```
$ mysql -ureactome_user -preactome_pass
mysql> CREATE DATABASE reactome;
mysql> CREATE DATABASE reactome_dn;
```

6. Fill the databases with downloaded data.

```
$ gunzip -c sql.gz |\
mysql -ureactome_user -preactome_pass reactome
$ gunzip -c sql_dn.gz |\
mysql -ureactome_user -preactome_pass reactome_dn
```

7. Create the reaction map images and other files that will be used to create the front page of the Web site. This procedure creates a directory with the same name as the main database in `/usr/local/gkb/website/html/img-tmp`. If you are updating your database rather than doing everything from the scratch, remove this directory first, as otherwise the new files will not be created.

```
$ /usr/local/gkb/scripts/release/
  create_frontpage_files.pl
```

8. Make sure that the Web server can write into and read from the directory used to store the temporary images.

```
$ chmod 777 /usr/local/gkb/website/html/img-tmp
```

This latter makes `/usr/local/gkb/website/html/img-tmp` *directory writable by anybody able to log into the server. If this seems too lax, assign the ownership of this directory to the user account running the Web server and make it writable by this user only.*

Configure and start the Reactome Web server

9. Choose the Web server configuration file for starting the Web server. The Reactome distribution comes with one configuration file (`httpd.conf.static`) for servers with compiled-in modules and another (`httpd.conf.dso`) for servers that use the Dynamic Shared Object (DSO) mechanism.

Both of those files are located in the `/usr/local/gkb/website/conf` *directory. The default Apache server installation compiles all the modules statically. However, the server that comes with Mac OS X, for example, uses the DSO mechanism.*

10. To determine which mechanism your Web server uses, invoke the `httpd` executable with `-l` command line parameter. If the output contains all of the following lines, use the `httpd.conf.static` file; otherwise use `httpd.conf.dso`.

```
mod_log_config.c
mod_mime.c
mod_include.c
mod_autoindex.c
mod_dir.c
mod_cgi.c
mod_alias.c
mod_access.c
mod_so.c
```

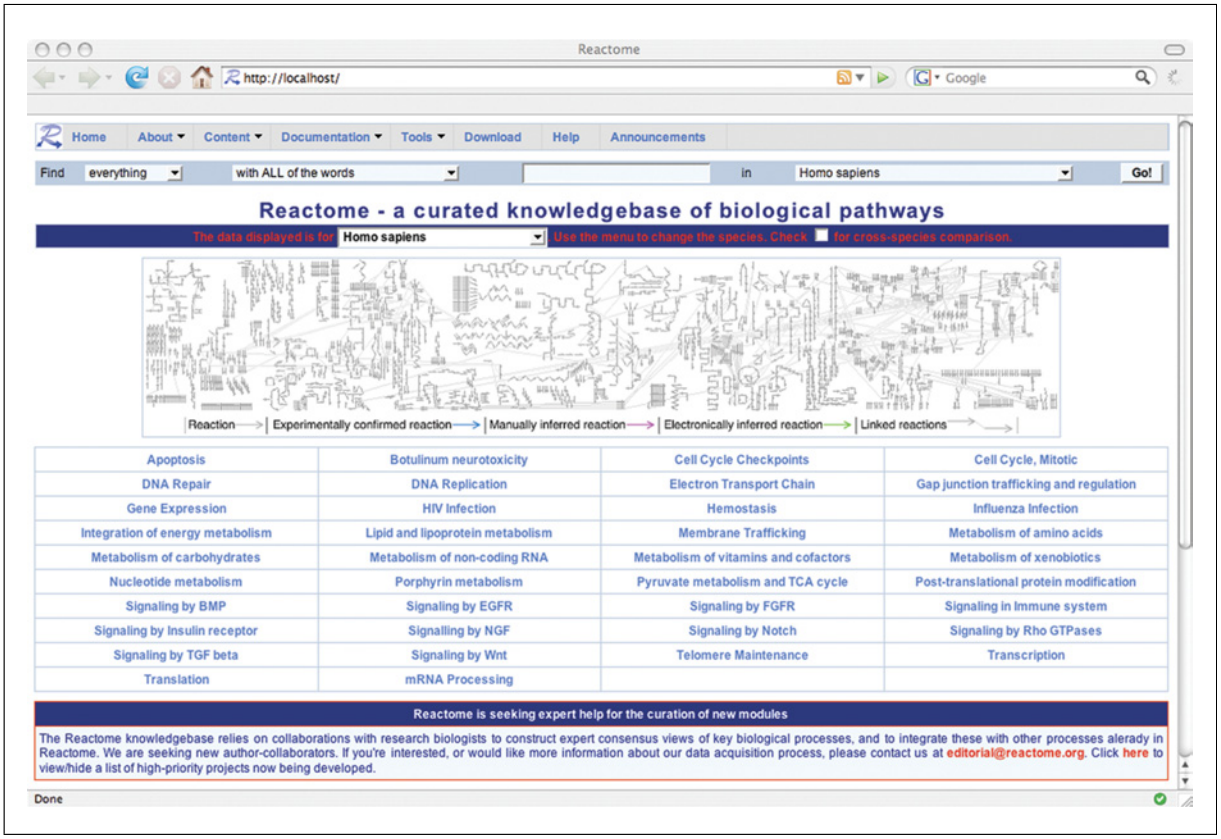
11. Start the Web server with the appropriate configuration file. The following example assumes that the server has compiled-in modules.

```
$ httpd -f /usr/local/gkb/website/conf/
  httpd.conf.static
```

Note that if Apache was preinstalled on your system, the `httpd` *binary may be named something different, such as* `apache` *or* `apache2`.

If you are not logged in as the root user, execute the command via `sudo`. *If the* `httpd` *executable is not in the root users path, you will have to specify the full path to the* `httpd` *executable. For example, if* `httpd` *is located in* `/usr/local/apache/bin` *directory, issue the command:*

```
$ sudo /usr/local/apache/bin/httpd \
-f /usr/local/gkb/website/conf/httpd.conf.static
```



Reactome - a curated knowledgebase of biological pathways

The data displayed is for Homo sapiens . Use the menu to change the species. Check □ for cross-species comparison.

Apoptosis	Botulinum neurotoxicity	Cell Cycle Checkpoints	Cell Cycle, Mitotic
DNA Repair	DNA Replication	Electron Transport Chain	Gap junction trafficking and regulation
Gene Expression	HIV Infection	Hemostasis	Influenza Infection
Integration of energy metabolism	Lipid and lipoprotein metabolism	Membrane Trafficking	Metabolism of amino acids
Metabolism of carbohydrates	Metabolism of non-coding RNA	Metabolism of vitamins and cofactors	Metabolism of xenobiotics
Nucleotide metabolism	Porphyrin metabolism	Pyruvate metabolism and TCA cycle	Post-translational protein modification
Signaling by BMP	Signaling by EGFR	Signaling by FGFR	Signaling in Immune system
Signaling by Insulin receptor	Signalling by NGF	Signaling by Notch	Signaling by Rho GTPases
Signaling by TGF beta	Signaling by Wnt	Telomere Maintenance	Transcription
Translation	mRNA Processing		

Reactome is seeking expert help for the curation of new modules

The Reactome knowledgebase relies on collaborations with research biologists to construct expert consensus views of key biological processes, and to integrate these with other processes alerady in Reactome. We are seeking new author-collaborators. If you're interested, or would like more information about our data acquisition process, please contact us at editorial@reactome.org. Click here to view/hide a list of high-priority projects now being developed.

Figure 9.10.1 Screenshot of the Web site front page after successful installation of a local copy of the Reactome database and Web site.

12. If you have a browser on the same computer where you installed Reactome, point your browser at *http://localhost*, otherwise replace `localhost` with the name or IP address of the computer hosting your Reactome Web site. You should see the Reactome front page (Fig. 9.10.1).

How to shut down Reactome installation

13. To stop the Web server, issue:

```
$ kill 'cat /usr/local/gkb/website/logs/httpd.pid'
```

Again, you may have to use `sudo` to be able to successfully complete this.

14. To stop the MySQL server:

```
$ mysqladmin -u root -p shutdown
```

INSTALLATION OF OTHER APPLICATIONS AND LIBRARIES

This module describes how to install third-party software required for installation and running of Reactome database and Web site.

Necessary Resources

Hardware

A computer with Unix or Linux operating system, Internet connection, and at least 200 Mb free disk space. You need to have root access on the installation machine for the installation with these instructions to succeed.

Software

Utilities such as gcc, gunzip, and tar. Often these come as part of the distribution of the operating system. However, on Max OS X, these have to be installed from the accompanying Developer Tools disk.

SUPPORT PROTOCOL 1

Installing Apache, Perl, and Java

Installing Apache Web server

If you need to install the Apache Web server, you can download the source code from *http://httpd.apache.org/download.cgi*. Although the Reactome project itself uses version 1.3 of the Apache Web server, later versions (2.2.4 is the most recent one at the time of this writing) will also suffice. For installation instructions, please follow the appropriate Documentation link on the Web page.

Installing Perl Version 5.8.0 or newer

Reactome can also be run with Perl 5.6, but this requires installation of a few extra modules that have become part of the standard Perl distribution since version 5.8. You can check for the presence of Perl and find the version of Perl installed on your system by issuing following command in the terminal:

```
$ perl -v
```

The Reactome Web site is written in Perl. The Reactome Web applications assume that the Perl executable resides at `/usr/local/bin/perl`. If Perl is installed on your system at a different location, create a symbolic link `/usr/local/bin/perl` pointing to the real location of your Perl executable. For example, on Mac OS X, the

Perl interpreter that comes with the system is located as `/usr/bin/perl`. To create the symbolic link, you need to open a terminal and type:

```
$ ln -s /usr/bin/perl /usr/local/bin/perl
```

Please note that if you do not have `/usr/local/` and `/usr/local/bin/` directories, you will have to create them first.

If you do not have Perl installed, or need to upgrade, you can get the source code from *http://www.cpan.org/src/README.html*. Follow the installation instructions on the Web page.

Installing Java version 1.5 (also known as Java 5.0)

Java is required for export of pathways in BioPAX and SVG formats only. You can find out which version of Java your system has by issuing following command in the terminal window:

```
$ java -version
```

Most recent version of Java can be obtained from *http://java.com/en/download/*.

SUPPORT PROTOCOL 2

Installing MySQL

MySQL is a popular open-source relational database system. The easiest way to install MySQL is to use the precompiled binaries from *http://dev.mysql.com*. Although internally Reactome uses version 4.0, versions 4.1 and 5.0 (the stable release at the time of this writing) also work.

Install MySQL

1. Download the appropriate standard binaries from http:*//dev.mysql.com/downloads/mysql*.
2. Unpack the downloaded file with:

   ```
   $ tar xvzf MYSQL-VERSION-OS.tar.gz
   ```

3. Follow the installation instructions in the `mysql/INSTALL-BINARY` file.

   Please note that your operating system may come with user and group "mysql" already defined. If this is the case, you can skip the two first steps in the Mysql installation instructions.

4. Add `/usr/local/mysql/bin` to your path. For Bash shell users, add this line to ~`/.bashrc`:

   ```
   PATH = ``$PATH:/usr/local/mysql/bin''
   ```

   For csh and tcsh users, add this line to ~`/.cshrc`:

   ```
   setenv PATH ``${PATH}:/usr/local/mysql/bin''
   ```

5. Once the database is running, connect to it as root:

   ```
   $ mysql -u root -p mysql
   ```

   Please note that MySQL will ask you for the root password that you set up during the installation process. Just press the return key if you did not set up the MySQL root password.

Create a MySQL user account to be used by Reactome Web server

6. Add a MySQL user account that the Reactome Web server will be using to connect to the database and retrieve data. In this example the user name is `reactome_user` and the password is `reactome_pass`. These will be used later when configuring Reactome Web code to access the database.

   ```
   mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
   ON reactome.*
   TO 'reactome_user'@'localhost'
   IDENTIFIED BY 'reactome_pass';
   ```

7. Extend the user's permissions to any database with name containing 'reactome':

   ```
   mysql> UPDATE db SET Db = '%reactome%'
   WHERE User = 'reactome_user';
   ```

Installing Perl Modules and Supporting Software

SUPPORT PROTOCOL 3

Install Perl modules and the software they require

The Reactome Web site needs a few Perl modules to be installed in order for it to run. These modules can all be downloaded from *http://www.cpan.org*, and are all installed in much the same way: download the compressed archive, unpack in a working directory, and install the module:

```
$ tar xvzf module.tar.gz
$ cd module
$ /usr/local/bin/perl Makefile.PL
$ make
$ make test
$ make install
```

Some of these modules require installation of further Perl modules. In addition, some of the modules depend on C libraries or executables that have to be installed before the Perl module in question can be installed successfully. Installation of these also follows a similar pattern:

```
$ tar xvzf package.tar.gz
$ cd package
$./configure
$ make
$ make install
```

The last step in both of the instructions above requires you to have the root privileges or execute the command as the root user. The latter is achieved by:

```
$ sudo make install
```

The Perl modules can also be installed by using the CPAN Perl module:

```
$ /usr/local/bin/perl --MCPAN ``install module::name''
```

The CPAN module automatically takes care of installation of required Perl modules. However, non-Perl libraries and executables still need to be installed manually.

A list of required Perl modules along with their purposes and dependencies is provided in Table 9.10.1.

Table 9.10.1 A Brief Description of Perl Modules

Module	Description
DBI	A common database interface for Perl. Defines a set of methods, variables, and conventions that provide a consistent database interface, independent of the actual database being used. The Reactome Perl API uses DBI to interact with Reactome database.
DBD::mysql	The MySQL driver for DBI, which mediates communication between DBI and the MySQL API
BioPerl	Perl modules for biology. Please note, though, that only the Bio::Root modules of the BioPerl core package are required. These are used for throwing exceptions and handling file input/output.
GD	Modules for programmatic drawing and manipulation of images. Used for drawing reaction diagrams and "reaction map." The module is an interface to gd graphics library in C (libgd), which is thus also required. libgd is available from *http://www.libgd.org* and requires the libpng library for creation and manipulation of images in Portable Network Graphics (PNG) format. libpng further requires zlib compression. linpng and zlib are available from *http://www.libpng.org* and *http://www.zlib.net*, respectively.
XML::Simple	An Easy API to maintain XML. Used to read an XML configuration file which determines the types of lists downloadable from dynamically created content pages and also contains the instructions for creating those lists. The module requires XML::Parser Perl module which itself requires expat XML parser library in C. expat is available from *http://sourceforge.net/projects/expat*.
PDF::API2	Module for creation and modification of Portable Document Format (PDF) files. Used for exporting description of Reactome pathways and reactions in PDF. Requires Compress::Zlib, Compress::Raw::Zlib, IO::Compress::Base, and IO::Compress::Zlib Perl modules.
GraphViz	Perl interface to the identically named toolkit for layout and image generation of directed and undirected graphs. Used for automatic layout of entity-level pathway diagrams. GraphViz toolkit is available from *http://www.graphviz.org*.
Archive::Tar	Module for manipulations of tar archives. Used for exporting Reactome data as Protégé projects. Requires IO::Zlib Perl module.
WWW::Search	Requires Date::Manip, IO::Capture, User, and Test::File.
Search:: Tools:: HiLiterm	Requires Data::Dump, File::Slurp, and Search::QueryParser

COMMENTARY

Background Information

The concept of a pathway database is not a novel one and there are numerous sources offering information under various access terms ranging from free-for-all to paying-subscriber-only. However, the feature that distinguishes the Reactome project from many of its peers is that, in addition to freely accessible data, it also offers the possibility to download and replicate the whole database and Web site. While the Reactome project attempts to provide easy access to various bits of information in various formats, having a local copy of the database and API code gives the ultimate freedom and flexibility to extract whatever is necessary.

While the Reactome project's own curation efforts concentrate mainly on human biology, the setup can be used to annotate biochemical processes of any cellular organism. Indeed, the Reactome project also produces orthology-based computational predictions of pathways in numerous other organisms. These can be used as a starting point for manual curation of pathways in other species. The Reactome Curator Tool, available from the Reactome download page

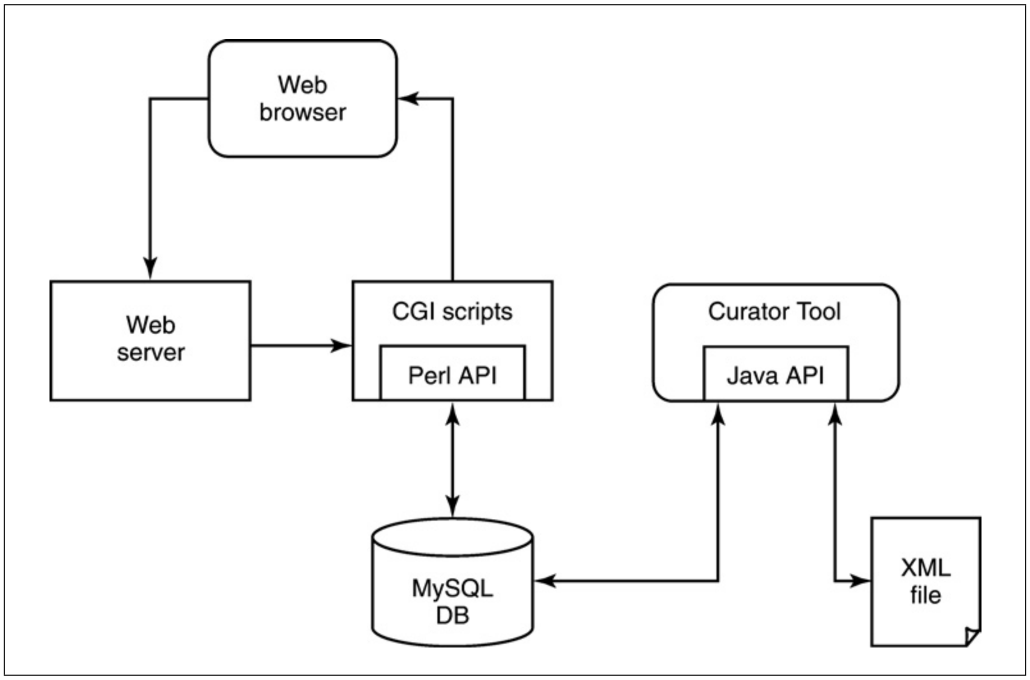


Figure 9.10.2 Architectural diagram of the Reactome software. The information is entered into the database with the help of Reactome Curator Tool. To enable off-line work, the latter can also store information in an XML file in the local file system. End users usually access Reactome database content with a Web browser via the Web server, where the request is handled by the CGI scripts that interrogate the MySQL database via Perl API.

at *http://www.reactome.org/download/*, is a stand-alone Java application that allows users to edit existing database entries and to enter new information. The same Web page also offers access to the Reactome Author Tool, which provides a more graphical way to enter and edit the information and hides many of the intricacies of the Reactome data model. However, in order to write the information assembled in the Author Tool back to the database, one has to use the Curator Tool.

The Reactome project also makes available Perl and Java APIs for accessing the data in the database. The Perl API comes as part of the Web site and code download, while the Java API is available as part of the Curator Tool installation. Although both of them are extensively used internally by the Reactome project, their documentation is limited; therefore, they should be approached only by individuals who are comfortable with writing software.

Both the software developed as part of the Reactome project and the external software used by Reactome installation are open source and freely available. The data are stored in a MySQL database, the Web server is Apache, and the Web site dynamic pages are written in Perl. The Reactome Web site and database can be installed on any computer with the Unix, Linux, or Mac OS X operating system. An architectural diagram of the software is shown in Figure 9.10.2.

Critical Parameters and Troubleshooting

The instructions presented in this unit assume that the user has root privileges on the computer where the local copy of Reactome is being installed. These privileges are required for installation of software at system-wide locations, as well as for starting up the Web server.

For the local installation of Reactome to work, both the Web and database servers have to be running. Perl has to be located at (or be symbolically linked from) `/usr/local/bin/perl`. The name of the database served by MySQL has to be as specified in the `/usr/local/gkb/modules/GKB/Config.pm`.

The most useful resource for resolving issues with Reactome installation is the *help@reactome.org* mailing list. When an error message is received, always record the text of the message. Check the Web server error log file (`/usr/local/gkb/website/logs/error.log`) for other error messages, which can be critical to understanding the problem, and include these in the E-mail sent to the *help@reactome.org* mailing list.

Browsing Multidimensional Molecular Networks with the Generic Network Browser (N-Browse)

Huey-Ling Kao[1] and Kristin C. Gunsalus[1]

[1]New York University, New York, New York

ABSTRACT

N-Browse is a graphical network browser for the visualization and navigation of heterogeneous molecular interaction data. N-Browse runs as a Java applet in a Web browser, providing highly dynamic and interactive on-demand access to network data available from a remote server. The N-Browse interface is easy to use and accommodates multiple types of functional linkages with associated information, allowing the exploration of many layers of functional information simultaneously. Although created for applications in biology, N-Browse uses a generic database schema that can be adapted to network representations in any knowledge domain. The N-Browse client-server package is freely available for distribution, providing a convenient way for data producers and providers to distribute and offer interactive visualization of network-based data. *Curr. Protoc. Bioinform.* 23:9.11.1-9.11.21. © 2008 by John Wiley & Sons, Inc.

Keywords: network • molecular • interaction • graph • browser • Web-based • client-server system • JAVA • functional genomics • GUI • visualization • database • MySQL

INTRODUCTION

New views of biological networks are emerging from the combination of large-scale experimental and computational approaches directed at understanding gene/protein function and functional relationships on many different levels. To help make sense of the wealth of data being generated, effective tools for visualizing and exploring these data are necessary. A natural paradigm for visualizing molecular interaction data is a network graph. However, extracting useful information about a local gene neighborhood from the entire network—which often can be very large and highly interconnected, thus colloquially termed a "giant hairball" (or "ridiculogram")—can be challenging. The goal of N-Browse is to provide a freely available software package that allows the biology community to share and explore functional interaction networks in an efficient, interactive, and user-friendly way.

Inspired by interactive graphical interfaces for coordinate-based genome annotations such as the Generic Genome Browser (GBrowse; UNIT 9.9), we have developed a similarly intuitive, easy to use, interactive tool for navigating gene network neighborhoods based on different kinds of functional links. This "Generic Network Browser," N-Browse, is available at *http://www.gnetbrowse.org*. N-Browse operates within a Web browser as a Java applet and uses a client-server system composed of a server-side MySQL database and a client-side graphical user interface (GUI). The N-Browse Web-based client allows users to quickly access and explore a variety of publicly available interaction data. In addition, the freely distributed N-Browse client-server package allows producers and providers of network-based data to employ N-Browse as a visual interface and distribution mechanism for serving their own combination of data from one or more species of interest.

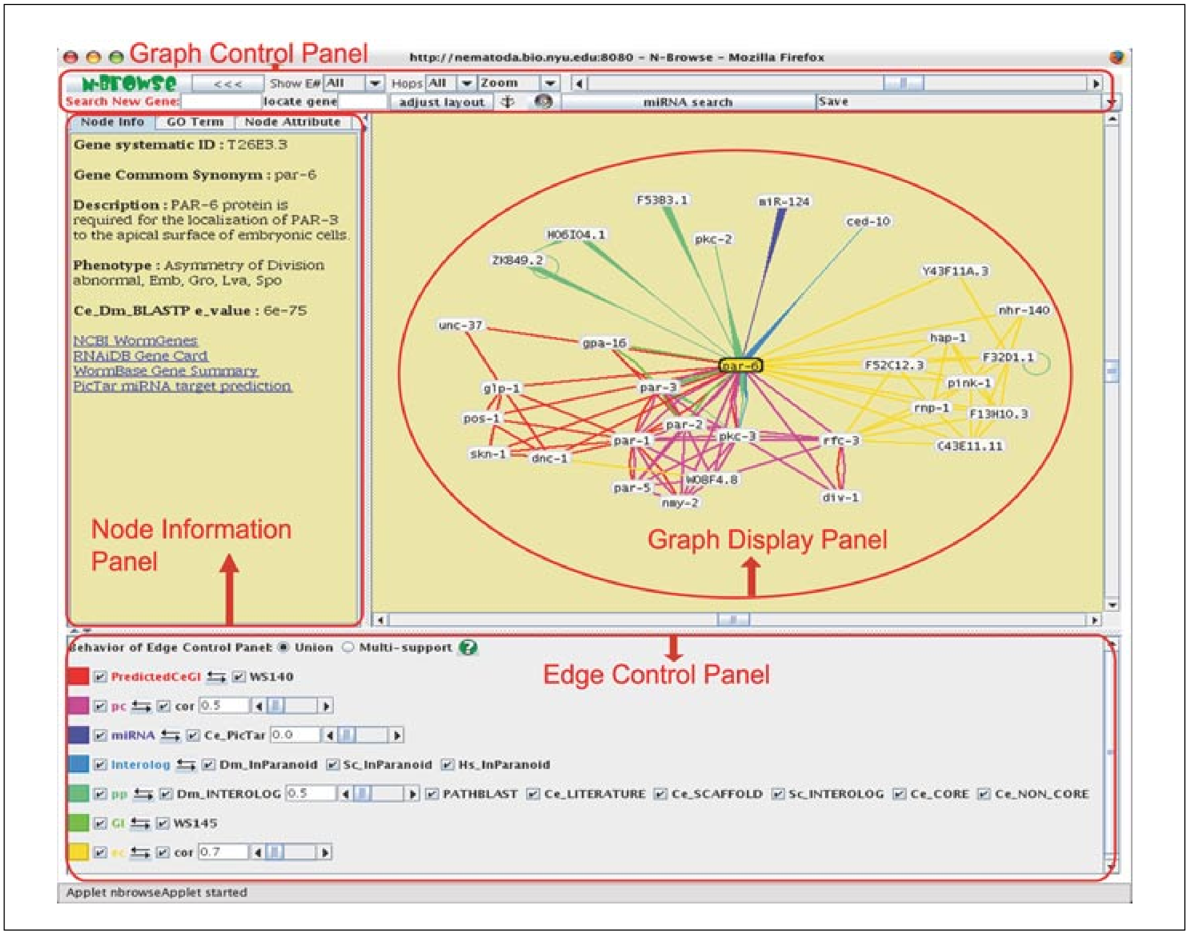
Current Protocols in Bioinformatics 9.11.1-9.11.21, September 2008
Published online September 2008 in Wiley Interscience (www.interscience.wiley.com).
DOI: 10.1002/0471250953.bi0911s23

N-Browse seeks to provide both a user-friendly client-side interface and a straightforward procedure for server-side installation and configuration. This unit has two basic protocols that describe usage and features of the client-side GUI. Basic Protocol 1 shows how to access useful information from any N-Browse Web site using the main N-Browse site at *http://gnetbrowse.org* as an illustrative example. Basic Protocol 2 shows how to use advanced functions to select and configure different combinations of data for network browsing. In addition, Basic Protocols 3 and 4 describe how to set up an independent N-Browse server site from the N-Browse client-server distribution package. A fully functional N-Browse Web site will require both installing and configuring the Web server host software (Basic Protocol 3), as well as setting up and populating an N-Browse database (Basic Protocol 4). A troubleshooting section describes how to detect and fix potential problems that might be encountered during installation.

NAVIGATING THE N-Browse GUI

A quick tour describes the main features of the N-Browse GUI, which consists of four panels (Fig. 9.11.1). The online tutorial at *http://gnetbrowse.org/* includes a more detailed description, as well as demonstration videos illustrating different aspects of the GUI; an overview of each panel's functions is described here.

The **Graph display** panel is the central component of the GUI and provides a network representation of available interaction data. It offers a number of interactive features for manipulating the graph (described in more depth in sections below) and communicates with other panels in the N-Browse GUI.



Gene systematic ID : T26E3.3

Gene Commom Synonym : par-6

Description : PAR-6 protein is required for the localization of PAR-3 to the apical surface of embryonic cells.

Phenotype : Asymmetry of Division abnormal, Emb, Gro, Lva, Spo

Ce_Dm_BLASTP e_value : 6e-75

Behavior of Edge Control Panel: ● Union ○ Multi-support

🟥 ☑ PredictedCeGI ⇆ ☑ WS140

🟪 ☑ pc ⇆ ☑ cor 0.5

🟦 ☑ miRNA ⇆ ☑ Ce_PicTar 0.0

🟦 ☑ Interolog ⇆ ☑ Dm_InParanoid ☑ Sc_InParanoid ☑ Hs_InParanoid

🟩 ☑ pp ⇆ ☑ Dm_INTEROLOG 0.5 ☑ PATHBLAST ☑ Ce_LITERATURE ☑ Ce_SCAFFOLD ☑ Sc_INTEROLOG ☑ Ce_CORE ☑ Ce_NON_CORE

🟩 ☑ GI ⇆ ☑ WS145

🟨 ☑ ec ⇆ ☑ cor 0.7

Figure 9.11.1 The N-Browse GUI. For color version of this figure see *http://www.currentprotocols.com*.

The **Edge control** panel provides a menu of the different functional edge types available. Essential features of this menu include the following:

1. The menu is constructed automatically from stored data types. Mousing over each item in the menu will display a brief description of it.
2. Different edge data types are distinguished by different colors, which can be changed by clicking on the color swatches for each edge type.
3. For each edge data type, different datasets (e.g., from multiple independent large-scale studies) can be defined and appear as individual sub-items in the menu.
4. The display of each data type or dataset can be manipulated independently using a toggle switch, by checking (to show) or unchecking (to hide) the adjacent box.
5. Navigation within the Graph display panel is restricted to the datasets listed in this menu. Thus, at each expansion step, the number and types of edges (and nodes) drawn into the graph are limited to those datasets available in the menu. Basic Protocol 2 explains how to configure which data sets are included in this panel, which can be done by either (i) preselecting specific subsets of interest (using the Advanced tab) and/or (ii) uploading your own data. Showing/hiding edges for display and preselecting datasets for navigation produce different graphs; this is because hidden edges are still included in expansion steps, so neighbors of hidden nodes will be drawn into the graph in the former but not the latter case. See the online tutorial for more detail on this topic.

The **Node information** panel contains three main sections, each of which can be opened by choosing the corresponding tab, as described below. (Additional tabs may be included in future releases of the N-Browse client-server core package or in association with plug-ins.)

1. **Node Info**: Provides a brief description and some useful links about any gene in the graph. This information appears dynamically when mousing over a node with the cursor.
2. **Node Attribute**: Allows different types of node attributes to be highlighted on the graph. A menu of available node attributes is dynamically generated when this tab is opened. Both categorical (e.g. phenotypes) and ordinal data (e.g. expression levels) can be highlighted.
3. **GO Term**: Displays the Gene Ontology DAG (directed acyclic graph) and highlights terms for selected gene(s) in the current graph. At present, this tab is only available on the main N-Browse Web site (*http://gnetbrowse.org*), as the current implementation is experimental.

The **Graph control** panel provides options to manipulate the Graph display panel, including:

1. Locate a specific node in the current network.
2. Back up one navigation step in the current network (retract last expansion step).
3. Zoom or rotate the network.
4. Display a new network by entering a new query.
5. Auto-launch Cytoscape using Java Webstart.
6. Save the current network in a variety of formats (text or image).

BASIC PROTOCOL 1

BASICS OF THE N-Browse GUI

The N-Browse user interface provides a simple, easily accessible way to interactively browse many different kinds of functional linkages at once. The GUI was designed with several features in mind: (1) dynamic graphical interface for network browsing and expansion, (2) dynamic edge and node attribute detection, (3) easily accessible information on nodes and edges, with links to useful external resources, and (4) highly configurable selection of edge data sets and score cutoffs.

This protocol will take the user through the main features of the N-Browse client-side user interface. Below we present the general idea of each function and some examples of how to view and explore network data, navigate local neighborhoods, and visualize properties of interest. The N-Browse online tutorial also includes demonstration videos illustrating each of these functions.

Necessary Resources

Hardware

Any computer with internet access

Software

Java-compatible internet browser
Java Runtime Environment (JRE) 1.4 or above

Files

N-Browse homepage: *http://gnetbrowse.org*
N-Browse tutorial: *http://gnetbrowse.org/N-Browse_tutorial.html*
N-Browse system requirements: *http://gnetbrowse.org/info.html*

Browsing the network neighborhood around a single query gene

1. Start a Java-compatible Web browser and open the N-Browse homepage at *http://gnetbrowse.org* (Fig. 9.11.2).

   Make sure Javascript and Java are enabled in your browser preferences, since N-Browse requires both (Javascript is needed to render the homepage and Java for the GUI).

   Follow steps 2 to 4 below to generate a network display centered around a single gene/protein query.

   Basic Protocol 2 provides instructions on selecting specific datasets for display and integrating user-defined data uploads in network navigation.

2. Type in the name of a gene.

   Sample queries for different species are provided for reference.

3. Select the species from the drop-down menu.

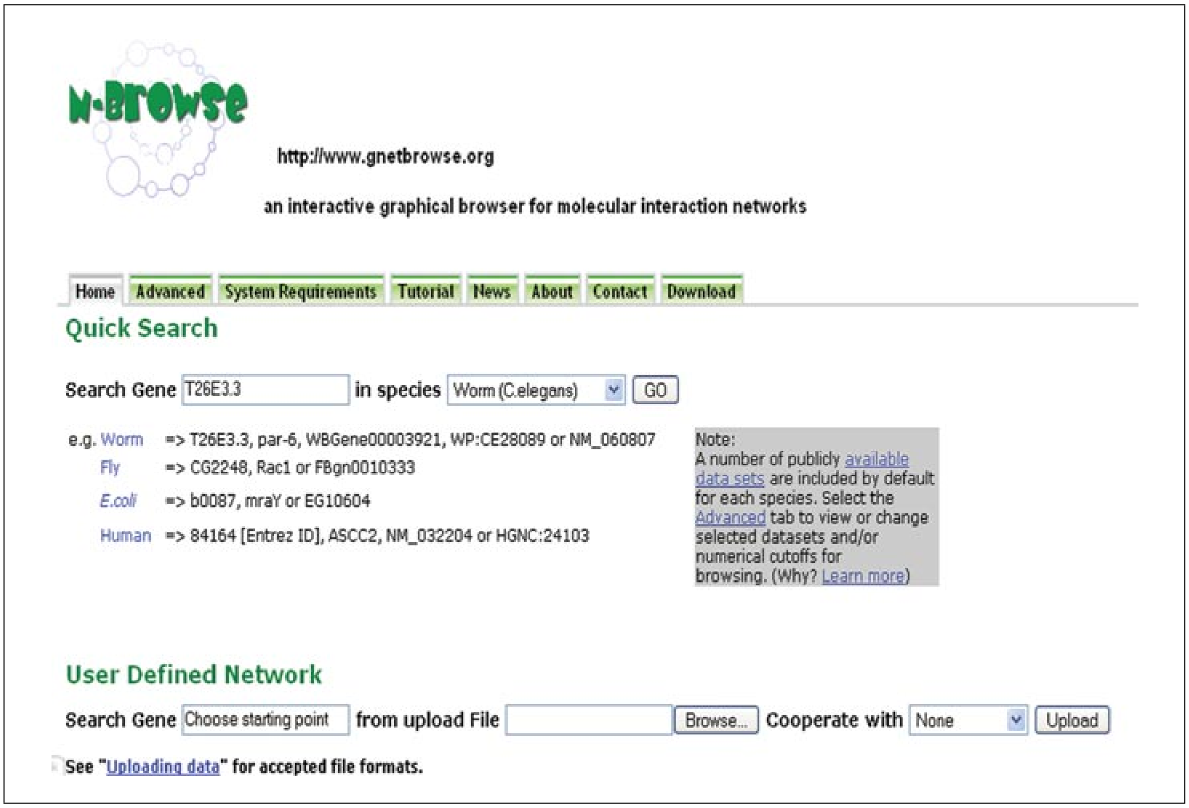
4. Click the GO button.

   This will open a new window containing the N-Browse GUI described above. Since the Java applet requires access to your computer's hard drive, you will also need to click Trust when prompted about the Java applet's certificate.

   In Windows operating systems running Explorer, the GUI window will occupy the full screen (pressing Ctrl+Esc will exit full-screen mode).

   If you do not see the new window pop up, check that your browser preferences are set to allow pop-up windows for the N-Browse Web site. If a new window is launched but you do not see the N-Browse GUI (Fig. 9.11.2), double-check your browser preferences to make sure Java is enabled.

   All subsequent operations described below refer to the main N-Browse GUI window.



http://www.gnetbrowse.org

an interactive graphical browser for molecular interaction networks

Quick Search

Search Gene T26E3.3 in species Worm (C.elegans)

e.g. Worm => T26E3.3, par-6, WBGene00003921, WP:CE28089 or NM_060807
Fly => CG2248, Rac1 or FBgn0010333
E.coli => b0087, mraY or EG10604
Human => 84164 [Entrez ID], ASCC2, NM_032204 or HGNC:24103

Note:
A number of publicly available data sets are included by default for each species. Select the Advanced tab to view or change selected datasets and/or numerical cutoffs for browsing. (Why? Learn more)

User Defined Network

Search Gene Choose starting point from upload File Cooperate with None

See "Uploading data" for accepted file formats.

Figure 9.11.2 The N-Browse homepage.

5. *Inspect a node:* Mousing over a node causes relevant information about that node (gene) to be displayed in the Node information panel, including links to external databases (Fig. 9.11.1). Right-clicking (or Control-clicking with a one-button mouse) on a node causes a drop-down menu to appear with several control options to manipulate the selected node(s). To select multiple nodes, press and hold the Shift button on the keyboard while clicking nodes in the current network.

6. *Inspect an edge:* Mousing over an edge will display the edge type, edge dataset, and its numerical value, if any. Right-clicking (or Control-clicking with a one-button mouse) on an edge will open a drop-down menu with options to hide the edge or to show any external links associated with that edge (e.g., in other databases), which will appear in a new browser window. In order to see information resulting from mousing over, the size of the network can be adjusted by selecting Zoom in the Graph control panel and moving the adjacent slider bar as described in step 16.

   If you do not see a new window pop up when following external links, please make sure that your pop-up blockers are disabled for the N-Browse Web site.

7. *Select a new query:* In the Graph control panel, look for Search New Gene. Type a gene ID or name in the box and press Enter/Return to build a new network graph around this gene as the starting point.

   Ambiguous or unidentifiable IDs will trigger a warning prompt with a new window offering a list of available choices. Select one of these and click the Submit button to generate a new query.

Expanding the network around a node in the graph

8. *Expand the graph:* Double-click any node to retrieve any additional functional links to that gene/protein that are not currently displayed (Fig. 9.11.3). With the default view setting ("matrix" view), this will display all edges between any newly retrieved

first-degree neighbors of this node and all other nodes in the current graph. Browse the local network neighborhood by sequentially expanding around successive nodes as illustrated in Figure 9.11.3.

Expanding the network in the default matrix view setting may generate a tremendous number of new edges. To avoid this, users can switch to a "spoke" view, which displays only links between the selected node and its direct first-degree neighbors. See Basic Protocol 2 for information on invoking the alternate "spoke" display view.

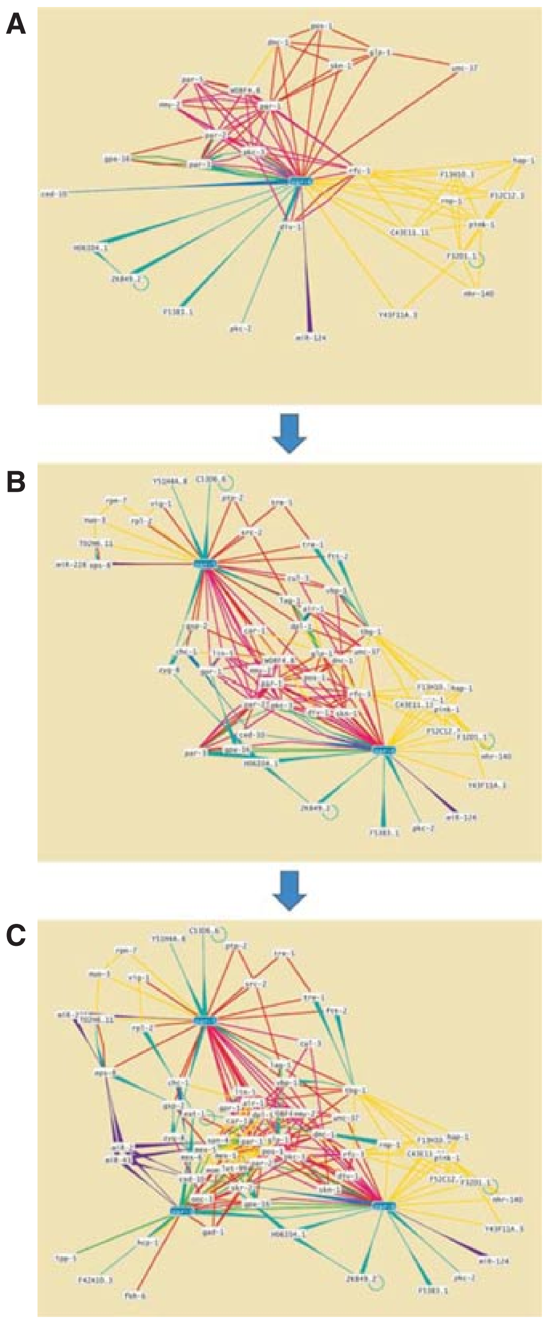


Figure 9.11.3 Network navigation: sequential expansion around selected network nodes. In this example, the user has entered an initial query `par-6` (**A**) and subsequently expands the network around `par-5` (**B**) and `par-3` (**C**). For demonstration video see the N-Browse online tutorial. For color version of this figure see *http://www.currentprotocols.com*.

9. *Step backward:* Recent steps in the network expansion sequence can be sequentially reversed by clicking the "<<<" button in the Graph control panel, causing all nodes and edges gathered in each step to disappear from view.

10. *Select "miRNA search:"* This will cause all microRNAs predicted to target genes in the current graph to be displayed.

    miRNA-target relationships at gnetbrowse.org are currently based on predictions by the PicTar algorithm, available at http://pictar.org.

    This function is hidden in the N-Browse distribution package, as it depends on the presence of a particular data type in the corresponding species database. It can be restored by uncommenting out this part of the code and recompiling.

Customizing the display

11. Select specific data types, datasets, and/or numerical thresholds for display.

    By default, all available data types and data sets (listed in the Edge control menu) are included for display in the network graph. The Edge control panel provides the ability to selectively hide or show edges, either by toggling check boxes for different data types and datasets or by changing numerical cutoffs (where present).

    See Basic Protocol 2 for more information on preselecting basis datasets for network navigation.

12. *Display Gene Ontology terms:* Switching to the GO Term tab in the Node information panel displays a hierarchical term list from the Gene Ontology DAG (directed acyclic graph). There are two ways to highlight genes annotated with particular GO terms on the current graph (see *UNIT 7.2* for information on GO):

    a. Clicking on a specific GO term in the term list displayed in this panel will highlight any nodes in the graph that are annotated with that term.

    b. Selecting one or more nodes in the graph and choosing Show GO Term (*) from the drop-down menu that appears in response to a right-click (or control-click with a one-button mouse) on the selected node(s) will also activate the GO Term panel. In this case, all GO terms associated with the selected set of genes will then be highlighted in the DAG term list. Right-clicking selected nodes in the graph will also open a second window that displays a graphical hierarchy of the GO terms associated with those genes.

    This function is not included in the N-Browse distribution package as the current implementation is experimental, but future releases will contain a corresponding feature.

13. *Highlight node attributes on the network graph:* Switching to the Node Attribute tab in the Node information panel allows the visualization of various properties associated with genes/proteins in the current graph. When the Node Attribute tab is opened, a list of available attributes will be displayed. Select a checkbox to load the corresponding attribute class for highlighting on the graph. Node attributes may be highlighted with either a graded scale of color intensity (for ordinal data) or a solid color (for categorical data). A description of this control menu is also available when clicking the "?" icon on the top-right corner of the Node Attribute tab.

    Node attributes can be any type of data associated with nodes (genes/proteins), such as phenotypes, expression levels, protein domains, etc.

14. *Anchor or weigh nodes in the graph:* All nodes in the graph can be frozen in place or unfrozen by clicking the Anchor icon in the Graph control panel. Alternatively, selected node(s) can be pasted to the background using the right-click drop-down menu that becomes available when mousing over nodes in the graph and selecting Anchor node or Weigh anchor, with the former anchoring the node and the latter removing the anchor, respectively.

15. *Adjust the current graph layout:* Clicking the “adjust layout” button in the Graph control panel resets the rendering of edges in the graph, resulting in the straightening or bowing of different edges.

   Normally, multiple edges between node pairs are splayed to avoid superimposition, and single edges appear straight. Because the rendering method is not automatically adjusted after each operation, hiding or unhiding selected edge types can sometimes leave, for example, single edges that appear bowed. Adjusting the layout may take a long time for a large graph with numerous edges, so be patient.

16. *Zoom, rotate, or move the graph:* The size and orientation of the network view can be adjusted by selecting Zoom or Rotate in the Graph control panel and moving the adjacent slider bar. The entire graph can be moved by positioning the cursor anywhere in the background field and dragging it while holding down the mouse button.

Saving and exporting network information

17. *Save network data:* Information about the current graph can be saved in a variety of formats:

   a. A list of nodes (as a tab-delimited text file).

   b. A list of interactions (as a tab-delimited text file).

   c. A Cytoscape `.sif` network file (see *UNIT 8.13*).

   d. A screenshot image (Save PNG image).

   e. An Encapsulated PostScript (EPS) file (Save EPS image).

   The EPS format provides a vector representation of the image, producing very high quality views that are especially useful for publications such as posters.

18. *Auto-launch Cytoscape:* Further analysis of the currently displayed network can be performed in the stand-alone graph layout application Cytoscape (also see *UNIT 8.13* and *http://cytoscape.org*). Clicking on the Cytoscape icon in the Graph control panel will cause all of the data in the current graph to be packaged for export to and displayed using Cytoscape, which will be launched automatically on the user’s computer using Java Web Start.

   The Cytoscape-compatible files created by N-Browse do not automatically specify the geometry of the graph layout, so a new layout will need to be generated from within Cytoscape after import.

BASIC PROTOCOL 2

WORKING WITH DATASETS AND USER-DEFINED UPLOADS

By default, network navigation in N-Browse includes all data in an N-Browse database and operates using a “matrix,” or complete, view of all defined edges. N-Browse provides the ability to configure the range of data used for network navigation, either by: (1) preselecting specific subsets of data and thresholds from an N-Browse database, or (2) uploading user-defined data for integrated visualization with publicly available data.

In addition, N-Browse allows users to configure the display method for visualizing links between selected nodes and their first-degree neighbors by toggling between the default “matrix” view and an alternative “spoke” view. This protocol describes each of these features. The N-Browse online tutorial also includes demonstration videos illustrating each of these functions.

Necessary Resources

Hardware

Any computer with internet access

Software

Java-compatible internet browser
Java Runtime Environment (JRE) 1.4 or above

Files

N-Browse homepage: *http://gnetbrowse.org*
N-Browse Advanced page: *http://gnetbrowse.org* (select "Advanced" tab)
N-Browse tutorial for user-defined data upload: *http://gnetbrowse.org/upload_tutorial.html*

1. Start a Java-compatible Web browser (such as Firefox, Safari, or Internet Explorer) and open the N-Browse homepage at *http://gnetbrowse.org*.

2. Select the Advanced tab (Fig. 9.11.5).

3. Type in the name of a gene.

4. Select the species from the drop-down menu.

Selecting "matrix" versus "spoke" views

Two display options are available for visualizing functional links between selected nodes of interest and their first-degree neighbors (Fig. 9.11.4): (i) ***Matrix view:*** This is the default for N-Browse and displays all links between all nodes in the graph (Fig. 9.11.4, left-hand panel). The matrix view guarantees that if any known edge exists between any two nodes in the current graph, it will appear in the network diagram (unless any of these have been manually hidden by the user). (ii) ***Spoke view:*** In contrast, the "spoke" view (Fig. 9.11.4, right-hand panel) shows only links directly attached to query nodes (either the initial query or a node selected for expansion). Since any edges between two neighbors of a query node will not be shown, many existing edges are typically not displayed in this view. The spoke view reduces visual "clutter," but makes a trade-off with information content (since many potentially interesting functional links will not be revealed). In some cases the spoke view may be preferred, particularly where relationships between neighbors are not clear (e.g., co-immunoprecipitation can recover many proteins pulled down by a single query protein, but no information is available on whether any of these directly interact with each other).

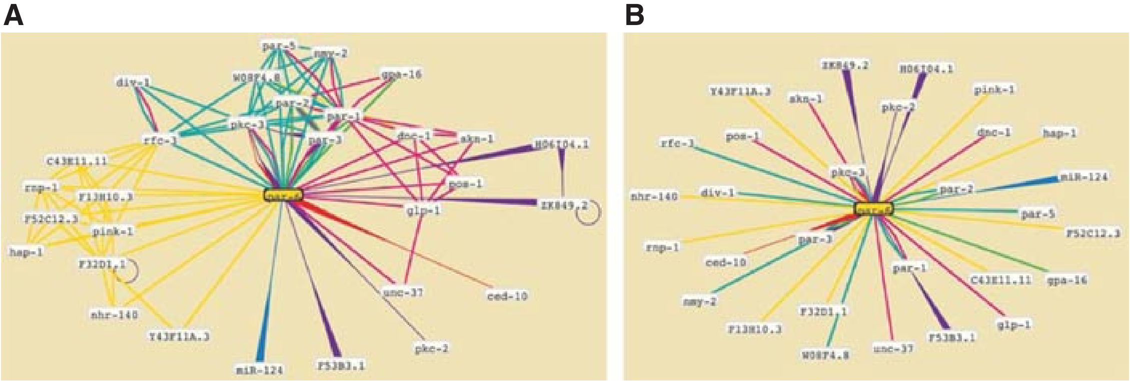


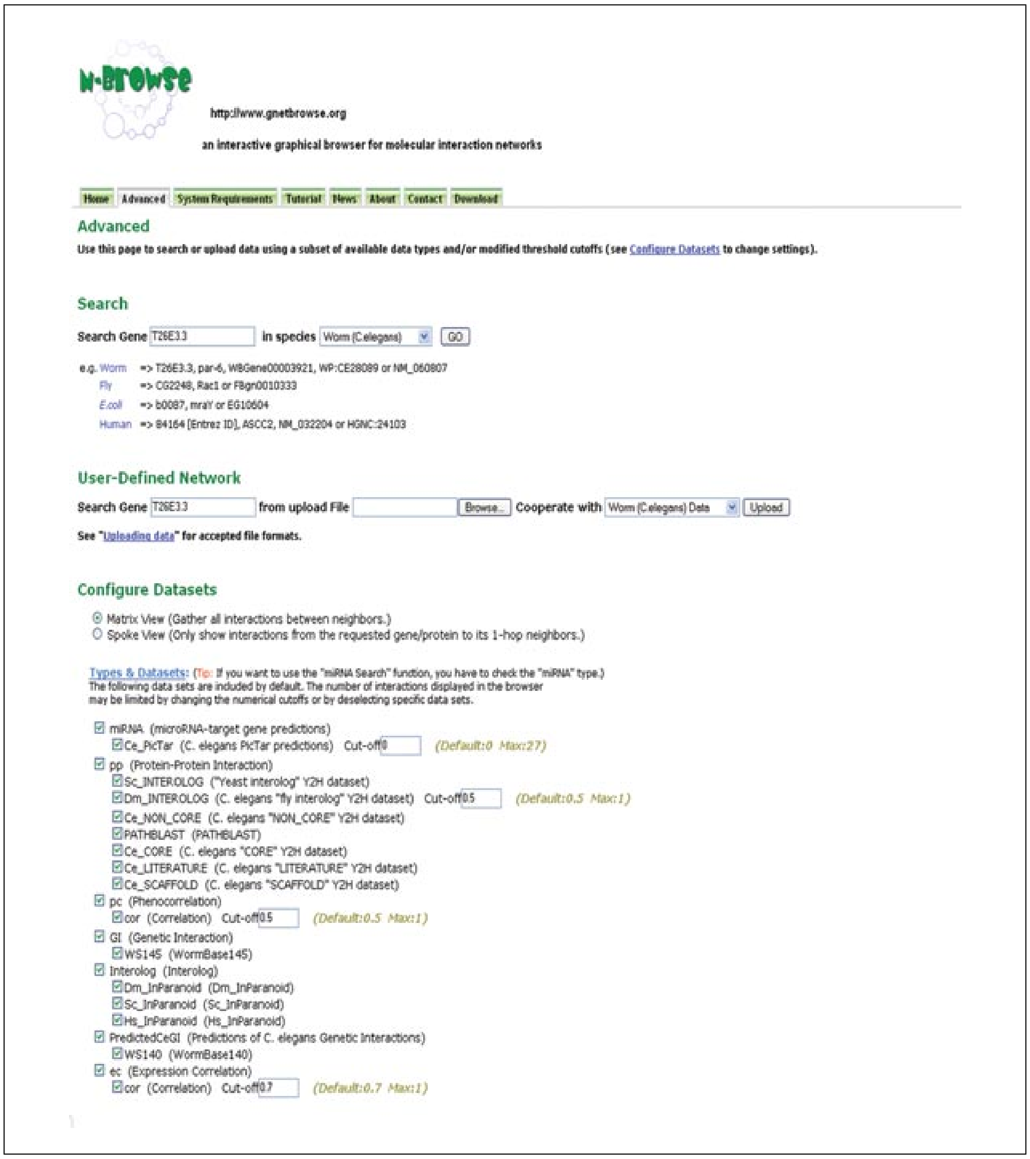
Figure 9.11.4 Matrix (**A**) and Spoke (**B**) views for the query `par-6` in *C. elegans* at *http://gnetbrowse.org*. For color version of this figure see *http://www.currentprotocols.com*.

5. In the Configure Datasets section on the Advanced page, select either Matrix View (Gather all interactions between neighbors) *or* Spoke View (Only show interactions from the requested gene/protein to its 1-hop neighbors).

Selecting specific datasets and thresholds for network navigation

If some data types available in the database are not of interest to the user for some reason, it is possible to exclude them from the base data sets used for network navigation. For example, a user may be interested only in physical or genetic interactions, or may consider some datasets unreliable, or may wish to impose a more stringent cutoff than the default threshold for a correlation coefficient or other score.

6. *Configure datasets for network navigation:* The Configure Datasets section on the Advanced page will dynamically list the available datasets in the N-Browse database for the selected species (Fig. 9.11.5). Selecting a different species from the menu in the Search section will automatically refresh the contents of the list. Configure datasets as follows:



N-Browse

http://www.gnetbrowse.org

an interactive graphical browser for molecular interaction networks

Advanced

Use this page to search or upload data using a subset of available data types and/or modified threshold cutoffs (see Configure Datasets to change settings).

Search

Search Gene T26E3.3 in species Worm (C.elegans) GO

e.g. Worm => T26E3.3, par-6, WBGene00003921, WP:CE28089 or NM_060807
Fly => CG2248, Rac1 or FBgn0010333
E.coli => b0087, mraY or EG10604
Human => 84164 [Entrez ID], ASCC2, NM_032204 or HGNC:24103

User-Defined Network

Search Gene T26E3.3 from upload File Browse... Cooperate with Worm (C.elegans) Data Upload

See "Uploading data" for accepted file formats.

Configure Datasets

● Matrix View (Gather all interactions between neighbors.)
○ Spoke View (Only show interactions from the requested gene/protein to its 1-hop neighbors.)

Types & Datasets: (Tip: If you want to use the "miRNA Search" function, you have to check the "miRNA" type.)
The following data sets are included by default. The number of interactions displayed in the browser may be limited by changing the numerical cutoffs or by deselecting specific data sets.

☑ miRNA (microRNA-target gene predictions)
  ☑ Ce_PicTar (C. elegans PicTar predictions) Cut-off 0 (Default:0 Max:27)
☑ pp (Protein-Protein Interaction)
  ☑ Sc_INTEROLOG ("Yeast interolog" Y2H dataset)
  ☑ Dm_INTEROLOG (C. elegans "fly interolog" Y2H dataset) Cut-off 0.5 (Default:0.5 Max:1)
  ☑ Ce_NON_CORE (C. elegans "NON_CORE" Y2H dataset)
  ☑ PATHBLAST (PATHBLAST)
  ☑ Ce_CORE (C. elegans "CORE" Y2H dataset)
  ☑ Ce_LITERATURE (C. elegans "LITERATURE" Y2H dataset)
  ☑ Ce_SCAFFOLD (C. elegans "SCAFFOLD" Y2H dataset)
☑ pc (Phenocorrelation)
  ☑ cor (Correlation) Cut-off 0.5 (Default:0.5 Max:1)
☑ GI (Genetic Interaction)
  ☑ WS145 (WormBase145)
☑ Interolog (Interolog)
  ☑ Dm_InParanoid (Dm_InParanoid)
  ☑ Sc_InParanoid (Sc_InParanoid)
  ☑ Hs_InParanoid (Hs_InParanoid)
☑ PredictedCeGI (Predictions of C. elegans Genetic Interactions)
  ☑ WS140 (WormBase140)
☑ ec (Expression Correlation)
  ☑ cor (Correlation) Cut-off 0.7 (Default:0.7 Max:1)

Figure 9.11.5 The N-Browse Advanced Web page. For color version of this figure see *http://www.currentprotocols.com.*

a. To restrict the use of specific data types or datasets, deselect them by unchecking the corresponding checkbox.

b. To change the threshold cutoff used for data with numerical ranges, enter a new number in the text box for a specific dataset (the ranges present in the database are shown for reference).

c. For more information about each data type and dataset available in the database, click on the Types and Datasets link.

7. Press **Go** in the **Search** section.

   If no prior query has been entered, a new N-Browse GUI window will open containing only the selected data types, datasets, or data subsets. If a previous query was entered in the Advanced page, the open N-Browse GUI will be refreshed and the edge menu in the Edge control panel will now contain only the selected sets. Eliminated datasets will no longer be considered during network navigation.

   Different network information is retrieved from the database when navigating using the full database contents versus selected subsets of the data. Using preselected subsets limits the edges and nodes gathered during network expansion steps to those subsets matching the preselection criteria. Thus, data that are eliminated from network navigation by preselection will never be displayed in the network graph and are not considered for expansion steps. Typically, this results in gathering fewer new edges and nodes at each step. Users should be aware that preconfiguring datasets is very different from toggling datasets for display in the Edge control panel menu, in which case all data listed in the edge menu (including hidden data) are still gathered in expansion steps. This is necessary so that data hidden from view can still be retrieved for display at any point in the session.

Uploading data for integrated viewing

Users are often interested in visualizing data that are not available in an N-Browse database, either from their own laboratory's work or other data sources. N-Browse allows users to upload their own data for integrated viewing with the publicly available data in an online N-Browse database. Currently, the file upload function accepts a simple tab-delimited file format; descriptions and a sample file for *C. elegans* are provided in the N-Browse online tutorial at *http://gnetbrowse.org/upload_tutorial.html*.

8. Open either the N-Browse homepage or Advanced page on the N-Browse Web site.

   Both provide the ability to integrate with available data in the N-Browse database. If data are uploaded using the Advanced page, you can simultaneously configure the network view (matrix or spoke) and the datasets in the N-Browse database to be included for network navigation.

9. *Specify a file containing network data:* In the section User-Defined Network, click Choose File and select a file located on your computer that contains network data in one of the accepted file formats described on the N-Browse Web site.

   User-defined data are not stored by the application, but are temporarily cached during an active session. Thus these data are no longer accessible once the current session is closed or expires.

10. *Upload the data file:* Select the species network with which you wish your data to be integrated, and press Upload. If a previous query was entered in the Advanced page and the corresponding N-Browse GUI is still open, the GUI will be refreshed and the edge menu in the Edge control panel will now contain one or more new menu items listing the data you provided. If your file explicitly specified one or more data types or datasets, each of these will be listed by name with the prefix `UD_` (for user-defined), for example `UD_Y2H`; otherwise, your data will appear under the moniker `UD_unknown`.

BASIC PROTOCOL 3

INSTALLING AND CONFIGURING THE N-BROWSE CLIENT-SERVER PACKAGE

As described in the Introduction, a fully functional N-Browse site will require completing Basic Protocols 3 and 4. This protocol describes how to install the N-Browse client-server package in the Unix/Linux environment. It is assumed that the user has proper knowledge and privileges to install software in the Unix/Linux environment.

After installation, the N-Browse Web pages should appear through an HTTP connection. To test the N-Browse GUI, you will need to populate an N-Browse database with either the test data provided with the distribution or your own data, as described in Basic Protocol 4.

Necessary Resources

Hardware

Any Unix (Linux, Solaris or other) workstation or Macintosh OS X
A minimum of 500 Mb RAM
Internet connection

Software

Most of the required software can be installed using a package manager for your OS platform or downloaded directly from the providers at the Universal Resource Locators (URLs) listed below

Standard software

N-Browse client-server package [the package can be downloaded either from the main N-Browse Web site (*http://gnetbrowse.org*) or from SourceForge (*http://sourceforge.net/projects/nbrowse/*)]
Tomcat4 or higher (Tomcat5 is preferred; the package is available at *http://tomcat.apache.org/*)
Java Software Development Kit (Java SDK) 1.4 or higher (the package is available at *http://java.sun.com*)
Perl v. 5.0 or higher (Perl 5.8 or higher is preferred; Perl will generally be installed on most machines with a Unix-like operating system, but it is also available at *http://www.perl.org*)
MySQL Server 4.1 or higher. MySQL Server 5 is preferred (MySQL is available at *http://www.mysql.com/*)
JDBC Driver for MySQL (Connector/J) (the library is available at *http://dev.mysql.com/downloads/connector/j/5.0.html*)

Nonstandard software

These packages will be needed if you want to set up Cytoscape auto-launch using Java Web Start:
Apache Web server (available at *http://httpd.apache.org/*)
Perl CGI package (available at *http://search.cpan.org/dist/CGI.pm/*)
Perl Driver for MySQL (DBD::mysql) (the library is available at *http://search.cpan.org/dist/DBD-mysql/*)

Files

The `install.pl` and `README` files are located in the `nbrowse_server_client/` directory after unpacking the N-Browse tarball

Table 9.11.1 Required Parameters in the `install_conf` File[a]

Parameter	Description
`TOMCAT_SERVER [nematoda.bio.nyu.edu]`	Domain name of the Tomcat Web server. This will serve as the base URL for HTTP connections.
`TOMCAT_PORT [8180]`	Port number for Tomcat HTTP connections. The default port for Tomcat5 is 8180.
`TOMCAT_WEBAPPS_PATH [/var/lib/tomcat5/webapps/]`	Physical (directory) location of the Tomcat Web application in the file system. The default location for Tomcat5 is `/var/lib/tomcat5/webapps/.`
`TOMCAT_APP_FOLDER [NBrowse]`	Directory name to be used for the N-Browse Web application. If you plan to run multiple N-Browse servers on the same machine, this name can be customized to distinguish different instances.
`MYSQL_SERVER [localhost]`	MySQL server location for database connections. If the server resides on the same machine as the N-Browse server package, you can use "localhost." If the database resides on a different machine, a domain name is required to make remote database connections.
`MYSQL_PORT [3306]`	MySQL server port number. You can leave it empty if you connect to "localhost" (MySQL default setting).
`MYSQL_DATABASE_NAME [nbrowse]`	Name of the MySQL database containing the N-Browse database schema (see Basic Protocol 4).
`MYSQL_USERNAME [handler]`	MySQL username that the N-Browse package will use as a handler for database connections.
`MYSQL_PASSWORD []`	MySQL password for the above MySQL user. It can be empty if the MySQL user has no password.
`INSTALL_CYTOSCAPE_AUTOLAUNCH [Y/N]`	Choose Y (yes) or N (no) to set up the Cytoscape auto-launch function. If Y, the next 3 parameters must be specified (`JNLP_CODEBASE, JAVA_LOCATION`, and `CYJNLP_LOCATION`).
`JNLP_CODEBASE [nematoda.bio.nyu.edu/cgi-bin/nbTest/]`	Web address (URL) for a directory on the N-Browse server machine in which Perl CGI scripts have permission to run (specified in Tomcat or Apache config files). This is essential to create the files for Cytoscape auto-launch.
`JAVA_LOCATION[/usr/bin/]`	Physical (directory) location of the JAVA binary.
`CYJNLP_LOCATION [/usr/lib/cgi-bin/nbTest/]`	Physical (directory) location of the Perl CGI scripts required for Cytoscape auto-launch.
`WEBSITEHOSTBY [NYU Center for Genomics & Systems Biology]`	Text for customizing the N-Browse Web site at your institution, to be displayed in an iFrame container at the top of the N-Browse Web pages at your site. If desired, you can further customize the Web site (with graphics etc.) by directly editing the HTML code in the `containerInfo.html` file included in the distribution.

[a] These parameters must be customized for your server machine prior to N-Browse installation. Example values for each parameter are provided in square brackets.

Download and install N-Browse

1. *Download and install the required software:* Download N-Browse from either of the locations listed above to the prospective N-Browse server machine.

   The file is in `.tar.gz` format and will need to be uncompressed and unpacked before installation.

2. Configure the `install_conf` file in the `nbrowse_server_client/` directory. Edit the file so that the required parameters suit your machine's configuration.

   This configuration file MUST be modified before installing the N-Browse package. Table 9.11.1 provides an explanation of each parameter with example values.

3. *Install the N-Browse package from source:* Run the `install.pl` script located in the nbrowse_server_client/ directory:

   ```
   $ perl install.pl
   ```

   The "`$`" symbol represents a command line prompt. The prompt may be represented by other symbols on different systems.

   This will install the required software components in the locations and using the parameter setting specified in the `install_conf` *file.*

4. *Test the installation:* Check if the installation went correctly by opening the following pages in your favorite browser (replacing the text string `TOMCAT_APP_FOLDER` in the URL below with the parameter value you specified in the `install_conf` file):

   ```
   http://localhost:8180/TOMCAT_APP_FOLDER/NBrowse.html
   ```

   If the installation was successful, you should see your customized N-Browse homepage at this URL (similar to Fig. 9.11.2). If you do not see the NBrowse.html page, it is likely that your Tomcat is not configured properly. In this case, see the Troubleshooting section below.

BASIC PROTOCOL 4

INSTALLING AND POPULATING THE N-Browse GENERIC DATABASE

The design considerations for the N-Browse database schema included the need to accommodate a diversity of data types without prior knowledge of their content or structure, and the ability of the system to automatically discover the types of data and ranges of values present across the entire database. To facilitate populating an N-Browse database, the N-Browse distribution package provides a set of Perl scripts that will automatically populate the generic N-Browse database schema using user-supplied data in simple tab-delimited (`.cvs`) files. This protocol provides a step-by-step guide to setting up and populating an N-Browse database by running these scripts. With default parameters, these scripts will load a set of sample data included in the distribution package. The resulting sample database can be used as a test version for the installation process. For convenience, this protocol also includes an optional shortcut for generating the sample database directly from a MySQL data dump to facilitate testing other aspects of the installation.

Necessary Resources

Hardware

Any Unix (Linux, Solaris or other) workstation or Macintosh OS X
A minimum of 500 Mb RAM
Internet connection

Software

All necessary software should be installed if Basic Protocol 3 has been completed

Files

After unpacking the N-Browse tarball, the Perl dataloader scripts and `README` file are located in the `nbrowse_dataloader/` directory

Dataloader Perl scripts:

```
dataloader_node_syn.pl
dataloader_edge_meta.pl
dataloader_url.pl
dataloader_node_attr.pl
dataloader_truncate_tbs.pl
```

Data file format specification:

```
dataloader_csv_format.txt
```

README file containing short descriptions of the Perl dataloader scripts:

```
README.txt
```

Create an N-Browse generic database

1. Create an empty MySQL database called `nbrowse` and make it accessible to the N-Browse MySQL database handler. This can be accomplished using the following commands:

```
$ mysql -u root -p
Enter password: ********
```

The “`$`” symbol represents a command line prompt. The prompt may be represented as other symbols on different systems. The MySQL user does not need to be “`root`,” but must have the privilege to create databases.

```
mysql> create database nbrowse;
Query OK, 1 row affected (0.01 sec)
```

Here we use “`nbrowse`” as a database name for demonstration purposes. This name can be anything but should be the same as the one indicated in the `install_conf` file described in Basic Protocol 3.

```
mysql> grant all privileges on nbrowse.*
to handler@localhost;
Query OK, 0 rows affected (0.01 sec)
```

When granting privileges to the user, replace `handler` with the username of the nbrowse database handler and `nbrowse` with the name of your database. These should be the same as specified in the `install_conf` file described in Basic Protocol 3.

```
mysql> quit
Bye
```

2. Load the N-Browse database schema into the newly created database. Start MySQL as the N-Browse database handler:

```
$ mysql -u handler -p
Enter password: ********
```

Switch to the `nbrowse` database and load the schema file `nbrowse_schema.sql` located in the `nbrowse_dataloader/` directory:

```
mysql> use nbrowse;
mysql> source /home/bob/nbrowse_install_package/
nbrowse_dataloader/nbrowse_schema.sql;
```

This example uses the Unix username "bob" as the N-Browse package administrator. When loading the schema into the database, replace the above path with the absolute path of the nbrowse_schema.sql file on your system.

```
mysql> quit
Bye
```

Load a sample N-Browse database from a MySQL data dump

This section is optional. It will allow you to immediately test the N-Browse GUI for your server setup using a prebuilt database. You can generate the same database by skipping this section and following the instructions in the next section.

NOTE: If you chose to carry out steps 3 and 4, make sure to either truncate all the tables in the nbrowse database (e.g., using the provided script dataloader_truncate_tbs.pl), or drop the database completely and repeat steps 1 and 2 prior to proceeding with the next section.

3. Load the sample N-Browse database dump into the newly created database. Start MySQL as the N-Browse database handler:

   ```
   $ mysql -u handler -p
   Enter password: ********
   ```

   Switch to the nbrowse database and load the SQL dump for the sample database (nbrowse_sample_data.sql located in the nbrowse_dataloader/ directory):

   ```
   mysql> use nbrowse;
   mysql> SOURCE /home/bob/nbrowse_install_package/
     nbrowse_dataloader/ nbrowse_sample_data.sql;
   ```

   As in step 2, replace the Unix path above with the absolute path of the nbrowse_schema.sql file on your system.

4. *Test the GUI for your N-Browse installation:* Go to the N-Browse URL on your system, enter a query (e.g., "par-6"), and click GO. The N-Browse GUI should appear with a small sample network.

Populate an N-Browse database from flat files

This section provides instructions for populating an N-Browse database using a set of Perl scripts provided with the distribution package for the user's convenience. However there are many other ways to populate the database, and users should feel free to use whatever method works best for them. Other methods will typically involve generating the appropriate SQL commands with customized scripts. To load the database with sample data included in the N-Browse distribution, follow the steps outlined in this section using default configuration parameters. Upon completion of these steps (assuming you have previously completed Basic Protocol 3) you should be able to navigate network data on your site using the N-Browse GUI.

In addition to the table definition file, a diagram of the N-Browse database schema is included in the N-Browse distribution package. To help users learn to understand the schema, examples of SQL queries to retrieve different kinds of data from a populated N-Browse database are also provided. The N-Browse database schema uses autoincrement IDs as primary keys in many of the data tables. Users who prefer to load data using scripts that automatically generate and cross-reference these IDs should remove the autoincrement flags from the corresponding table definitions.

Table 9.11.2 Parameters in the `dataloader_conf` File[a]

Parameter	Description
`TAXON_ID [6239]`	NCBI Taxonomy ID. N-Browse uses this ID to distinguish network data from different species within the same database and to retrieve species names from NCBI for display. A current list of species IDs and names can be found at *ftp://ftp.ncbi.nih.gov/pub/taxonomy/* in the `names.dmp` file contained in the `taxdump` archive (distributed in various formats: `.zip,.tar.Z,.tar.gz`).
`TABLE_EDGEDEF [./example_data/table_edgedef.csv]`	Edge type definition file.
`TABLE_GNBI [./example_data/table_gnbi.csv]`	Binary interaction data file.
`TABLE_NODES [./example_data/table_nodes.csv]`	File containing node primary names and descriptions.
`TABLE_SYN [./example_data/table_syn.csv]`	Node synonyms for search and display functions. A priority score specifies the preferred names for display.
`TABLE_URL [./example_data/table_url.csv]`	Optional data file specifying the construction of call strings for external URLs that can be attached to nodes or edges (e.g. links to other database resources).
`TABLE_NODE_ATTR [./example_data/table_node_attr.csv]`	Optional data file of node attributes (e.g. BlastP E-values, phenotypes, domains, expression levels, etc.)
`MYSQL_SERVER [localhost]`	MySQL server location.
`MYSQL_PORT [3306]`	MySQL server port number. You can leave it empty if you connect to "localhost" (MySQL default setting).
`MYSQL_DATABASE_NAME [nbrowse]`	MySQL N-Browse database name.
`MYSQL_USERNAME [handler]`	MySQL username with database write privilege.
`MYSQL_PASSWORD []`	MySQL password for the above MySQL user. It can be empty if the MySQL user has no password.

[a] These parameters must be customized for your server machine and data filenames. Using the default parameter values shown in square brackets to run the Perl dataloader scripts will populate an empty N-Browse database with the sample data provided with the N-Browse distribution package.

5. *Prepare tab-delimited* `.csv` *files containing the data you wish to load to the database:* The required fields for each `.csv` file are described in the `dataloader_csv_format.txt` file in `the nbrowse_dataloader/` directory. The directory `example_data/` contains the sample data files.

   The `.csv` *format can be generated programmatically with a script, manually using a text editor, or automatically by Microsoft Excel or OpenOffice Calc. To export data in this format, place the data in different columns and then save as a "tab-delimited" file. (Data in each column can include spaces, but should not include the "tab" character.)*

6. *Configure the* `dataloader_conf` *file in the* `nbrowse_dataloader/` *directory:* The Perl dataloader scripts consult this configuration file for the names of the various data files to be loaded to the database. Specify the names of the data files you have prepared as values for the corresponding parameters in the configuration file (or, to load the sample database, use the default parameters).

   Table 9.11.2 presents parameters in the `dataloader_conf` *file.*

7. *Populate tables in the nbrowse database:* Each Perl dataloader script populates the database with a different type of information, as described below. Only the first two are essential for network display: (i) information about the identity (and descriptions) of nodes and (ii) information about edges (the types of functional links and any associated numerical values). Node data should be loaded first.

   a. Load information about nodes (genes and/or proteins) and their synonyms:

   ```
   $ cd nbrowse_dataloader
   $ perl dataloader_node_syn.pl
   ```

   This script populates the tables `node`, `synonym`, *and* `attribute` *in the nbrowse database. The* `.csv` *files* `TABLE_NODES` *and* `TABLE_SYN` *are required to run this script.*

   b. Load information about edges:

   ```
   $ perl dataloader_edge_meta.pl
   ```

   The script populates the tables (`gnb_interactions`, `edge_group`, `edge_attribute`, `attribute`, *and* `metadata`*) in the nbrowse generic database. The* `.csv` *files* `TABLE_EDGEDEF` *and* `TABLE_GNBI` *are required to run this script. Currently, this script only populates binary interactions in the generic schema (vs. interactions with multiple partners, such as obtained from co-IP data).*

   c. Load information about external links:

   ```
   $ perl dataloader_url.pl
   ```

   This is optional. The script populates the tables (`external_url`, `url_attribute`, `attribute`, *and* `metadata`*) in the nbrowse database. The* `.csv` *file* `TABLE_URL` *is required to run this script. This is useful for you to show your client-users more detailed information on nodes or edges.*

   d. Load information about node attributes:

   ```
   $ perl dataloader_node_attr.pl
   ```

   This is optional. The script populates the tables (`node_attribute`, `attribute`, *and* `metadata`*) in the nbrowse generic database. The* `.csv` *file* `TABLE_NODE_ATTR` *is required to run this script. This is for auto-constructing menu options for the Node Attribute menu (see Basic Protocol 1: Highlight node attributes on the network graph.).*

   If you completed steps 4-7 above using the default parameters and thus have loaded the sample database, make sure to truncate all the tables in the nbrowse database or drop the database completely and repeat steps 1 and 2 above prior to loading the actual data you wish to use.

COMMENTARY

Background Information

The central idea behind N-Browse is to develop an easily accessible, simple yet powerful tool that enables biomedical researchers to quickly extract data and generate hypotheses from the results of large-scale analyses in diverse organisms. Inspired by GBrowse (*UNIT 9.9*), an open-source software package that provides a Web-based GUI for coordinate-based genome annotations supported by a light-weight database, N-Browse aims to provide an analogous intuitive portal for network exploration and an easily configurable client-server package for distribution. The data content available from an N-Browse server, in terms of both functional linkage types and species-specific data, will vary at different providers' sites, but any data that can be described as nodes and can be displayed.

Several applications now provide similar network visualization tools, including Cytoscape (*UNIT 8.13*), Osprey, VisANT (*UNIT 8.8*), and STRING (*http://string.embl.de/*). Each was designed with differing goals and implemented independently. Different tools share certain features with the vision of N-Browse, such as navigating functional relationships based on data available from a remote server (e.g., Osprey applet version, VisANT, and STRING) or providing an open-source package for distribution (Cytoscape). N-Browse occupies a unique niche as a simple yet powerful on-demand navigation tool that allows researchers access to heterogeneous data through a Web browser in a highly interactive way and in a rich contextual environment. N-Browse can be easily integrated with other Web resources via URL links and its functionality is extensible through the integration of new data types and software plug-ins. Among the tools mentioned above, N-Browse is unique in offering an open-source client-server system supported by a generic database schema that is freely available for distribution. The N-Browse client-server package is suitable as a data distribution and visualization mechanism for any research group that wishes to serve network-related data to the public. N-Browse is affiliated with the Generic Model Organism (GMOD) project, which provides open-source software components for distribution of genomic and functional genomic data for any organism. A description of N-Browse and links to other N-Browse resources can be found on the GMOD Wiki site at *http://www.gmod.org/wiki/index.php/nbrowse*.

Critical Parameters and Troubleshooting

Below are addressed some common issues and questions encountered during use or installation of the N-Browse package:

Expansion steps are taking a long time to appear after double-clicking on a selected node. Is there a way to improve the querying process?

Everything is calculated on-the-fly when a client-end user queries a node. If the subnetwork of each node you have in your database is very large, you may want to activate a function that caches the edge number around each node. This will potentially save querying time if the hairball around the querying node is humongous. To implement this option (after deploying the package and populating the database), perform the following steps.

1. Change directory to the servelets location (`TOMCAT_WEBAPPS_PATH/TOMCAT_APP_FOLDER/WEB-INF/classes`), replacing `TOMCAT_WEBAPPS_PATH` and `TOMCAT_APP_FOLDER` with the parameters specified in your `install.conf` file (see Basic Protocol 3 and Table 9.11.1).

2. Run the following command (you must have permission to run `sudo`):

```
$ sudo java -classpath . databaseProcess.UpdateEdgeNum
```

Note that this script might take a while to run, depending on how many nodes and edges are contained in the database.

No data shows in the network browser GUI. What happened?

This is most likely a problem with the Tomcat security policy. To check this, perform the following steps.

1. Examine whether your Tomcat opens the SocketPermission.

2. Using your Web browser, try linking to the following URL (substituting the uppercase text with appropriate values specified in your `install_conf` file): *http://TOMCAT_SERVER:TOMCAT_PORT/TOMCAT_APP_FOLDER/database.jsp*.

If you see the text “1 2 3” appear in your browser window, your Tomcat server is communicating well with your MySQL server. If you see “1 2” and tons of exception messages, you may need to change your Tomcat policy to allow the connection to establish. Each version of Tomcat might behave differently. One possible fix you can try is to change the tomcat5 security policy as follows: (a) Find the file `policy.d/04webapps.policy` on your machine. (b) Copy the following lines and paste into the `04webapps.policy` file:

```
//allow MySQL connect
permission java.net.SocketPermission ''localhost'',
  ''connect,resolve'';
//allow getting species information from NCBI
permission java.net.SocketPermission ''www.ncbi.nlm.
  nih.gov:80'', ''connect,resolve'';
```

(c) Run the following command to restart the Tomcat Web server (you must have `sudo` permission):

```
$ sudo /etc/init.d/tomcat5 restart
```

Why is my user upload function not working?

Again, this may be a problem with the tomcat5 policy. Here are some suggestions you may want to try:

Modify the tomcat5 policy:

1. Find the file `policy.d/04webapps.policy` on your machine.
2. Copy the following lines and paste into the `04webapps.policy` file:

```
permission java.io.FilePermission
  ''/var/lib/tomcat5/temp/-'', ''read,write,delete'';
permission java.io.FilePermission ''/tmp/-'',
  ''read,write,delete'';
permission java.io.FilePermission ''./temp/-'',
  ''read,write,delete'';
permission java.io.FilePermission ''./uploads/-'',
  ''read,write,delete'';
permission java.util.PropertyPermission
  ''java.io.tmpdir'', ''read'';
```

3. Run the following command to restart Tomcat Web server (you must have `sudo`):

```
$ sudo /etc/init.d/tomcat5 restart
```

4. Re-compile the servelets for the data upload function: Change directory to the Tomcat Web application directory (`NBrowse` or whatever name you gave it in the `install_conf` file), and run the following commands:

```
$ cd WEB-INF/classes
$ javac -classpath . com/raditha/megaupload/*.java
$ sudo /etc/init.d/tomcat5 restart
```

5. Test the user upload function again. You can use the example file for user uploads `user_upload_example.txt` located in the `nbrowse_server_client/` directory to test the user upload function.

Why is my Cytoscape auto-launch function not working?

Check the permissions of the `cgi-bin/` directory. The default setting for apache2 is to allow execution of all file extensions in this directory as CGI scripts. If you change these permissions, you must at least allow files with `.cgi`, `.pl`, and `.jnlp` extensions to run as executable CGI scripts in `CYJNLP_LOCATION/` (this directory is specified in the `install_conf` file; see Basic Protocol 3 and Table 9.11.2).

ACKNOWLEDGEMENTS

We thank Fabio Piano and Yih-Shien Chiang for invaluable brainstorming sessions, advice, and suggestions during the development process and Leslie Greengard for his encouragement and support. We thank the following beta testers of the N-Browse software for helpful feedback on the installation process: Payan Canaran, Todd Harris, and Igor Antoshechkin (from WormBase) and Nicolas Simonis and Changyu Fan at the Center for Cancer Systems Biology (Dana Farber Cancer Center, Harvard). This work was supported by Department of the Army Award W81XWH-04-1-0307 and NYSTAR Contract #C040066.

KEY REFERENCE

Lall, S., Grun, D., Krek, A., Chen, K., Wang, Y.L., Dewey, C.N., Sood, P., Colombo, T., Bray, N., MacMenamin, P., Kao, H.L., Gunsalus, K.C., Pachter, L., Piano, F., and Rajewsky, N. 2006. A genome-wide map of conserved microRNA targets in *C. elegans*. *Curr. Biol.* 16:460-471.

This is the first article in the literature to describe the use of N-Browse for integrating a new genome-scale dataset with other available molecular interaction data. N-Browse was used to integrate microRNA-target predictions with multiple types of functional links in C. elegans gathered from a variety of sources (these datasets are described on the gnetbrowse.org Web site).

INTERNET RESOURCES

http://gnetbrowse.org

The main N-Browse Web site, currently providing access to heterogeneous functional data in E. coli, C. elegans, D. melanogaster, and H. sapiens (see the Web site for details on available datasets). Provides a link to the downloadable N-Browse client-server distribution package.

http://sourceforge.net/projects/nbrowse

The N-Browse client-server distribution package can be downloaded from here.

http://www.gmod.org/wiki/index.php/nbrowse

Provides a description of the N-Browse project with news and links to other N-Browse resources.

http://www.wormbase.org

The first example of an independent N-Browse client-server installation. WormBase currently uses N-Browse as a graphical interface to server molecular interaction data curated there. Links to the N-Browse GUI at WormBase are available on the Gene Summary pages. Also see UNIT 1.8*.*

http://interactome.dfci.harvard.edu/C_elegans/host.php

An N-Browse portal is provided by the CCSB Interactome Database to visualize C. elegans protein-protein interaction data in the context of other functional genomic data.