Databases and Database Access

A table of genes

Gene ID	Chromosome	Start	End	Strand	Class
GRMZM2G306328	chr2	175194049	175196453	-1	est
GRMZM2G027393	chr2	175212542	175213269	-1	cdna
GRMZM2G002915	chr2	175243929	175246053	1	est
GRMZM2G419606	chr2	175320426	175321226	-1	cdna
GRMZM2G119906	chr2	175323967	175325504	-1	cdna
GRMZM2G119950	chr2	175325765	175331607	-1	cdna
GRMZM2G125775	chr2	175462240	175463416	-1	cdna
GRMZM2G425965	chr2	175482597	175484512	-1	est
AC195825.3_FG001	chr2	176152209	176155132	-1	fgenesh

- Each row is a *record* of a gene
- Each column is a set of values constrained by a *type*
- A simple query: What is the location of gene ‘GRMZM2G42775’?

A more complex query

Table 1: GO Terms of genes

Gene ID	Go Term
GRMZM2G002903	nucleic acid binding
GRMZM2G002903	intracellular
GRMZM2G002903	transport
GRMZM2G002915	DNA binding
GRMZM2G002915	transcription factor activity
GRMZM2G002915	nucleus
GRMZM2G002915	transcription
GRMZM2G002915	transcription regulator activity
GRMZM2G002948	multicellular organismal development
GRMZM2G002948	cellular process
GRMZM2G002950	nucleotide binding
GRMZM2G002950	protein binding

Table 2: Expression values

Gene ID	Exp1	Exp2	Exp3	Exp4
GRMZM2G003109	127.24	86.973	214.73	109.8
GRMZM2G003138	124.73	119.41	125.77	107.08
GRMZM2G003165	77.78	163.4	69.063	51.56
GRMZM2G003167	231.41	420.47	82.018	88.929
GRMZM2G003179	239.6	399.86	483.38	361.11
GRMZM2G003234	107.14	99.023	125.07	84.288
GRMZM2G003246	151.39	94.289	69.389	54.414
GRMZM2G003252	374.61	966.41	560.12	464.19
GRMZM2G003354	4170.1	3378.6	1876.9	2153.5
GRMZM2G003368	13835	5958.7	77.495	100.6

Table 3: Gene table

Gene ID	Chromosome	Start	End	Strand	Class
GRMZM2G306328	chr2	175194049	175196453	-1	est
GRMZM2G027393	chr2	175212542	175213269	-1	cdna
GRMZM2G002915	chr2	175243929	175246053	1	est
GRMZM2G419606	chr2	175320426	175321226	-1	cdna
GRMZM2G119906	chr2	175323967	175325504	-1	cdna
GRMZM2G119950	chr2	175325765	175331607	-1	cdna
GRMZM2G125775	chr2	175462240	175463416	-1	cdna
GRMZM2G425965	chr2	175482597	175484512	-1	est
AC195825.3_FG001	chr2	176152209	176155132	-1	fgenesh

"What are the classes of highly expressed genes in region 50Mb-55Mb of chromosome 5?

What is a database

- A collection of data
  - Text file with a list of genes
  - GFF text file
  - BAM file
  - Excel spreadsheet
  - Set of tables in MySQL

DBMS: Software for managing databases

- Database Management Systems (DBMS)
  - General term for software for managing data
    - Creating tables
    - Loading data
    - Querying data
  - E.g: **MySQL, SQLite**, Oracle, Microsoft Access, Berkley DB, MongoDB

RDBMS:
Relational Database Management Systems

- Software for managing related data that is stored across multiple tables

Using a DBMS

- Through a user-interface
  - E.g: MySQL workbench, HeidiSQL, SequelPro, SQLite Manager, SQLite Spy
- Programmatically through SQL
  - Structured Query Language
  - E.g: “select gene_id from gene_table where chromosome = ‘chr2’;”
- Programmatically through an API in another programming language
  - **Perl DBI**
  - Java JDBC, C ODBC

What we will do today

- Creating databases, tables in MySQL
- Querying and manipulating data in SQL
- Querying and manipulating data using Perl DBI

MySQL

- A robust RDBMS is very popular for large bioinformatics databases. Great for:
  - Very large, persistent datasets
  - Multi users with different permission levels
  - High volume transactions

- To access the mysql client from the command line, you need 4 pieces of information:
  1. Host (*Defaults to localhost)*
  2. Port (*Defaults to 3306)*
  3. Username
  4. Password

- When MySQL is first installed, a 'root' account for administration is initialized, without a password

Using MySQL shell

- Start the MySQL client from the command line, and this will bring up a MySQL shell, connected to the MySQL server on you local machine

  $ mysql –u root

- For example, to connect to the public Ensembl MySQL:

  $ mysql –h ensembldb.ensembl.org –P 5306 –u anonymous

- Basic commands in the MySQL shell, line of commands must end with a ';'

  mysql> show databases;
  mysql> create database progbio2011; # progbio2011 is the database name
  mysql> use progbio2011; # Use another database
  mysql> help;

- To quit:

  mysql> \q;

- To cancel a command:

  mysql> \c;

Creating a table

Things to consider:

- Table name
- Name of each column
- Data type of each column
- Range of values of data in each column

Basic Datatypes

- Numeric
  - INT : for integers
  - Double : numerical data with decimals
- Strings
  - CHAR : for strings up to 255 in length
  - TEXT : large strings

- Lots more on the MySQL website
  http://dev.mysql.com/doc/refman/5.6/en/data-types.html

- Also see cheat sheet on course webpage.

SQL: Creating a table

gene_id	chr	start	end
GRMZM2G306328	chr2	175194049	175196453
GRMZM2G027393	chr2	175212542	175213269
GRMZM2G002915	chr2	175243929	175246053
GRMZM2G419606	chr2	175320426	175321226

```
CREATE TABLE genes (
    `gene_id` char(25) NOT NULL DEFAULT '',
   `chr` char(5) NOT NULL DEFAULT '',
   `start` int(9) NOT NULL DEFAULT '0',
   `end` int(9) NOT NULL DEFAULT '0',
   PRIMARY KEY (`gene_id`)
);
```

CREATE TABLE tablename (

column_1_name datatype [optional constraint]

column_2_name datatype [optional constraint]

….

);

KEYS and Indexes

- INDEX
  - Synonymous with KEY
  - It is the lookup column, or a set of columns, for a table.
  - There can be more than one KEY in a table

- PRIMARY KEY
  - The primary key for a table represents the column, or set of columns, that is mostly frequently used as an index to the table
  - Columns used as the primary keys must be contain values that are unique to each row
  - There can only be one primary key in a table

SQL: Simple query

 # SELECT column_name [,column_names] from table;

```
SELECT gene_id from genes;
```

 # Use limit when the list is too long

```
SELECT gene_id, chr, start, end from genes
limit 10;
```

 # Wildcard character "" for all columns in table*

```
SELECT * from genes limit 10;
```

SQL: SELECT …. WHERE for filtering results

 # what are the genes on chromosome 5

```
SELECT gene_id FROM genes  WHERE chr='chr5';
```

 # what are the genes that lie within 50Mb – 55 Mb of chromosome 5

```
SELECT gene_id
FROM genes
WHERE chr='chr5'
and end >=  50000000
and start <= 55000000;
```

SQL: SELECT …. WHERE with OR

 # what are the genes that lie within chr5 with evidence 'est' or 'cdna' ?

```
SELECT gene_id , class
FROM genes
WHERE chr='chr5'
and (evidence='cdna' OR evidence= 'est');
```

SQL: Sorting and Distinct

 # What are the last 20 genes on chr 10;

```
SELECT gene_id, chr, start, end FROM genes
WHERE chr='chr10'
ORDER BY end desc  LIMIT 20;
```

What is the unique list of gene evidences in the genes table?

```
SELECT DISTINCT evidence from genes;
```

SQL: SELECT COUNT… GROUP BY

count the number of rows in the table

```
SELECT COUNT(*) FROM genes;
```

count the number of genes in chromosome 5

```
SELECT COUNT(*) From genes where chr='chr5';
```

what if we want to return the number of genes in each chromosome?

```
SELECT chr, COUNT(*) FROM genes GROUP BY chr;
```

- Other SQL functions besides COUNT
  - avg, min, max, concat

SQL: select - join

Gene ID	Go Term
GRMZM2G002903	nucleic acid binding
GRMZM2G002903	intracellular
GRMZM2G002903	transport
GRMZM2G002915	DNA binding
GRMZM2G002915	transcription factor activity
GRMZM2G002915	nucleus
GRMZM2G002915	transcription
GRMZM2G002915	transcription regulator activity
GRMZM2G002948	multicellular organismal development
GRMZM2G002948	cellular process
GRMZM2G002950	nucleotide binding
GRMZM2G002950	protein binding

Gene ID	Exp1	Exp2	Exp3	Exp4
GRMZM2G003109	127.24	86.973	214.73	109.8
GRMZM2G003138	124.73	119.41	125.77	107.08
GRMZM2G003165	77.78	163.4	69.063	51.56
GRMZM2G003167	231.41	420.47	82.018	88.929
GRMZM2G003179	239.6	399.86	483.38	361.11
GRMZM2G003234	107.14	99.023	125.07	84.288
GRMZM2G003246	151.39	94.289	69.389	54.414
GRMZM2G003252	374.61	966.41	560.12	464.19
GRMZM2G003354	4170.1	3378.6	1876.9	2153.5
GRMZM2G003368	13835	5958.7	77.495	100.6

Gene ID	Chromosome	Start	End	Strand	Class
GRMZM2G306328	chr2	175194049	175196453	-1	est
GRMZM2G027393	chr2	175212542	175213269	-1	cdna
GRMZM2G002915	chr2	175243929	175246053	1	est
GRMZM2G419606	chr2	175320426	175321226	-1	cdna
GRMZM2G119906	chr2	175323967	175325504	-1	cdna
GRMZM2G119950	chr2	175325765	175331607	-1	cdna
GRMZM2G125775	chr2	175462240	175463416	-1	cdna
GRMZM2G425965	chr2	175482597	175484512	-1	est
AC195825.3_FG001	chr2	176152209	176155132	-1	fgenesh

SQL: simple join

what are the expression values for all transcription factors in experiment 1?

```
SELECT genes_go.gene_id, go_term, exp1
FROM genes_go, expression
WHERE genes_go.gene_id = expression.gene_id
and go_term = 'transcription regulator
activity' ;
```

Let's do this now

- "What are the classes of highly expressed genes in region 50Mb-55Mb of chromosome 5?

Perl DBI

- DBI is a module that provides access to DBMS in Perl
- It hides the nuts and bolts for connecting to each type of DBMS, leaving a consistent interface for connecting to a database
- The key object in DBI is the database handle ($dbh), which represents a connection to a DBMS.

Three easy steps for database transaction in DBI:

1. Create a database handle

```
$dbh = DBI->connect(…..)
```

2. Execute a SQL statement using the database handle

```
$dbh->do something
```

3. Disconnect the handle

```
$dbh->disconnect
```

Perl DBI Step 1: Constructing the handle

```
# for MySQL
my $dbname = 'prog2011';
my $driver = 'mysql';
my $user = 'root';
my $passwd = '';
my $host = 'localhost';
my $port = 3306;

my $dsn = "DBI:$driver:database=$dbname;host=$host;port=$port";
my $dbh = DBI->connect($dsn,$user,$passwd);


# for SQL lite
my $dbname = 'prog2011';
my $driver='SQLite';
my $dsn = "DBI:$driver:$dbname";
my $dbh = DBI->connect($dsn);
```

Perl DBI Step 2: Executing the SQL

1. Construct the SQL query

```
my $sql = "SELECT count(*) From gene";
```

2. Execute the transaction using the database handle
   - For querying and fetching data:

```
my $results_array_ref = $dbh->selectall_arrayref($sql);
```

DBI Example: creating a table

With $dbh->do()

```
=pseudocode
        $dbh = DBI->connect($dsn)
        $dbh->do(SQL)
        $dbh->disconnect
=end
```

```
#! /usr/bin/perl
use strict;
use warnings;
use DBI;

my $dbname = 'progbio2011';
my $user = 'root';
my $passwd = '';
my $host = 'localhost';
my $port = 3306;

my $dsn = "DBI:mysql:database=$dbname;host=$host;port=$port”;
my $dbh = DBI->connect($dsn,$user,$passwd);

my $sql = ”CREATE table foo (bar char(10)); ";

$dbh->do($sql);

$dbh->disconnect;
exit;
```

DBI : Fetch a list of genes using DBI

With $dbh->selectall_arrayref

```
=pseudocode
        $dbh = DBI->connect($dsn)
        $fetched_results1 = $dbh->fetch SQL query
        do something with results
        $dbh->disconnect
=end
```

```
#! /usr/bin/perl
use strict;
use warnings;
use DBI;

my $dbname = 'progbio2011';
my $user = 'root';
my $passwd = '';
my $host = 'localhost';
my $port = 3306;

my $dsn = "DBI:mysql:database=$dbname;host=$host;port=$port";
my $dbh = DBI->connect($dsn,$user,$passwd);

my $query = "SELECT gene_id, chr, start, end from genes";
my @results = @{$dbh->selectall_arrayref($query)};

foreach my $row_ref ( @results){
    my $str = join "\t",@{$row_ref};
    print $str,"\n";
}

$dbh->disconnect;
exit;
```

DBI : Count using selectrow_arrayref

For SQL queries which will only return a *single* row,
e,g: SELECT COUNT query
We can use $dbh->selectrow_arrayref

```
=pseudocode
        $dbh = DBI->connect($dsn)
        $fetched_row = $dbh->fetch SQL query
        do something with results
        $dbh->disconnect
=end
```

```
#! /usr/bin/perl
use strict;
use warnings;
use DBI;

my $dbname = 'progbio2011';
my $user = 'root';
my $passwd = '';
my $host = 'localhost';
my $port = 3306;

my $dsn = "DBI:mysql:database=$dbname;host=$host;port=$port";
my $dbh = DBI->connect($dsn,$user,$passwd);

my $query = "SELECT COUNT(*) from genes where chr ='chr10'";
my @row = @{$dbh->selectrow_arrayref($query)};

print join ("\t",@row),"\n";;

$dbh->disconnect;
exit;
```

DBI : Querying using placeholders

fetch expression level for a list of genes

```
……
my $dbh = DBI->connect( $dsn, $user, $passwd );

my $query =     "SELECT exp1, exp2, exp3, exp4
          FROM expression
          WHERE gene_id = ?";
my $sth = $dbh->prepare($query);

my $file =shift;
open IN,"<",$file || die ("Can't open file $file $!");
while (my $gene_id = <IN>){
    chomp $gene_id;

    $sth->execute($gene_id);
    my @results = @{ $sth->fetchall_arrayref};

    foreach my $row_ref (@results) {
        my $str = join "\t", @{$row_ref};
        print $gene_id,"\t",$str, "\n";
    }
}
close IN;
$dbh->disconnect;
exit;
```

```
=pseudocode
        $dbh = DBI->connect($dsn)
        $sth = $dbh->prepare(SQL)
        loop:
                $sth->execute
                $sth->fetchrow_array;
        end loop
        $dbh->disconnect
=end
```

DBI : Placeholders vs selectall

fetch expression level for a list of genes

Using placeholders

```
=pseudocode

        @genelist = read from file

        $dbh = DBI->connect($dsn)
        $sql = "SELECT exp1
                FROM expression
                WHERE gene_id = ?"

        $sth = $dbh->prepare($sql)

        loop through genelist:
                $sth->execute($gene)
                $expr = $sth->fetchrow_array
                print $expr
        end loop
        $dbh->disconnect
=end
```

- 1 database transaction for each gene queried
- Slow if you have many genes to query

Using selectall

```
=pseudocode

        @genelist = read from file

        $dbh = DBI->connect($dsn)
        $sql = "SELECT gene, exp1  FROM expression";

        %gene_expr hash
        for each result in $dbh->selectall_arrayref($sql)
                $gene_expr hash{$gene} = $expr
        end

        loop through genelist:
                print $expr if exist in %gene_expr hash
        end loop

        $dbh->disconnect
=end
```

- A single database transaction
- Slow if you're fetching millions of rows and you end up only needing a small fraction

For other DBI functions, see CPAN

With $dbh->selectall_hashref

```
….

my $dsn = "DBI:mysql:database=$dbname;host=$host;port=$port”;
my $dbh = DBI->connect($dsn,$user,$passwd);

my $query = "SELECT gene_id, chr, start, end from genes";

my %results = %{$dbh->selectall_hashref($query,'gene_id')};

foreach my $gene (keys %results){
    print $gene,"\t",
          $results{$gene}->{'chr'},"\t",
          $results{$gene}->{'start'},"\t",
          $results{$gene}->{'end'},"\n";
}

$dbh->disconnect;

exit;
```

Other useful MySQL commands

- For loading a text file into a table from the unix command line:

  $ mysqlimport --local –u root databasename filename.txt

  Filename must have the same name as the table you are trying to load data into

- For dumping data from a table:

  $ mysqldump –u root databasename tablename > table.sql

  – Or for dumping the entire MySQL database:

  $ mysqldump –u root databasename > database.sql

Other useful SQL commands

- Inserting new rows into table

```
INSERT into genes (gene_id, chr, start, end,
evidence) values ('foo','chrX',1000,1500,'cdna');
```

- Updating data in table

```
UPDATE genes SET gene_id = 'bar' WHERE gene_id =
'foo';
```

Alternate RDBMS - SQLite

- Simple RDBMS that is installed by default in most systems
  - sqlite3
- To create a new database in SQLite from the unix command line

  ```
  $ sqlite3 mydatabase
  ```
- The command above also brings you into the sqlite3 client
- sqlite3 client allows

  ```
  sqlite> .help
  sqlite> .tables
  sqlite> .import FILE table
  sqlite> .exit
  ```