Relational Databases for Biologists: Efficiently Managing and Manipulating Your Data

Session 3
Building and modifying a database with SQL

George Bell, Ph.D.
WIBR Bioinformatics and Research Computing
Session 3 Outline

- SQL query review
- Creating databases
- Creating tables
- Altering table structure
- Inserting data
- Deleting data
- Updating/Modifying data
- Automating repetitive tasks
SELECT

```sql
> SELECT *
FROM Data
LIMIT 5;
```

```sql
> # Comments after ‘#’
# Get non-redundant list
SELECT DISTINCT species
FROM LocusDescr;
```

<table>
<thead>
<tr>
<th>affyId</th>
<th>exptId</th>
<th>level</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFX-MurIL2_at</td>
<td>hs-cer-1</td>
<td>20</td>
</tr>
<tr>
<td>AFFX-MurIL10_at</td>
<td>hs-cer-1</td>
<td>8</td>
</tr>
<tr>
<td>AFFX-MurIL4_at</td>
<td>hs-cer-1</td>
<td>77</td>
</tr>
<tr>
<td>AFFX-MurfFAS_at</td>
<td>hs-cer-1</td>
<td>30</td>
</tr>
<tr>
<td>AFFX-BioB-5_at</td>
<td>hs-cer-1</td>
<td>258</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>species</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hs</td>
</tr>
<tr>
<td>Mm</td>
</tr>
</tbody>
</table>

Relational Databases for Biologists © Whitehead Institute, 2006
WHERE And ORDER BY

> SELECT *  
  FROM RefSeqs  
  WHERE linkId BETWEEN 50 AND 100  
  LIMIT 5;

> SELECT *  
  FROM RefSeqs  
  WHERE linkId BETWEEN 50 AND 100  
  ORDER BY ntRefSeq DESC  
  LIMIT 5;

<table>
<thead>
<tr>
<th>linkId</th>
<th>ntRefSeq</th>
<th>aaRefSeq</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>NM_001098</td>
<td>NP_001089</td>
</tr>
<tr>
<td>51</td>
<td>NM_004035</td>
<td>NP_004026</td>
</tr>
<tr>
<td>52</td>
<td>NM_004300</td>
<td>NP_004291</td>
</tr>
<tr>
<td>53</td>
<td>NM_001610</td>
<td>NP_001601</td>
</tr>
<tr>
<td>54</td>
<td>NM_001611</td>
<td>NP_001602</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>linkId</th>
<th>ntRefSeq</th>
<th>aaRefSeq</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>NM_005159</td>
<td>NP_005150</td>
</tr>
<tr>
<td>81</td>
<td>NM_004924</td>
<td>NP_004915</td>
</tr>
<tr>
<td>91</td>
<td>NM_004302</td>
<td>NP_004293</td>
</tr>
<tr>
<td>86</td>
<td>NM_004301</td>
<td>NP_004292</td>
</tr>
<tr>
<td>52</td>
<td>NM_004300</td>
<td>NP_004291</td>
</tr>
</tbody>
</table>
GROUP BY And HAVING

> SELECT affyId, MIN(level) as min, MAX(level) as max
FROM Data
GROUP BY affyId
HAVING max - min > 5000
LIMIT 5;

<table>
<thead>
<tr>
<th>affyId</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>100047_at</td>
<td>20</td>
<td>7784</td>
</tr>
<tr>
<td>100068_at</td>
<td>414</td>
<td>5883</td>
</tr>
<tr>
<td>100069_at</td>
<td>616</td>
<td>6349</td>
</tr>
<tr>
<td>100329_at</td>
<td>20</td>
<td>21455</td>
</tr>
<tr>
<td>100342_i_at</td>
<td>786</td>
<td>7931</td>
</tr>
</tbody>
</table>

> SELECT gbId, count(affyId) AS num_affyIds
FROM Targets
GROUP BY gbId
HAVING COUNT(gbId) > 4
ORDER BY num_affyIds DESC
LIMIT 5;

<table>
<thead>
<tr>
<th>gbId</th>
<th>num_affyIds</th>
</tr>
</thead>
<tbody>
<tr>
<td>J04423</td>
<td>14</td>
</tr>
<tr>
<td>AC002397</td>
<td>12</td>
</tr>
<tr>
<td>AF109905</td>
<td>9</td>
</tr>
<tr>
<td>AF100956</td>
<td>9</td>
</tr>
<tr>
<td>AL031228</td>
<td>8</td>
</tr>
</tbody>
</table>
Table Joining

$> \text{SELECT DISTINCT Unigenes.uId, GO\_Descr.description AS GO\_description FROM Unigenes, LocusLinks, Ontologies, GO\_Descr WHERE Unigenes.linkId=LocusLinks.linkId AND LocusLinks.linkId=Ontologies.linkId AND Ontologies.goAcc=GO\_Descr.goAcc LIMIT 5;}$

<table>
<thead>
<tr>
<th>uId</th>
<th>GO_description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hs.373554</td>
<td>calcium ion binding</td>
</tr>
<tr>
<td>Hs.74561</td>
<td>protein carrier</td>
</tr>
<tr>
<td>Hs.155956</td>
<td>arylamine N-acetyltransferase</td>
</tr>
<tr>
<td>Hs.2</td>
<td>arylamine N-acetyltransferase</td>
</tr>
<tr>
<td>Hs.234726</td>
<td>serine protease inhibitor</td>
</tr>
</tbody>
</table>
Output Formats

- Query from MySQL prompt
- Ending query with \G (in place of ‘;’)
- mysql < q.sql – tab-delimited output
Access Privileges

- Restrict access and prevent accidental alteration of important information
- Can limit what individual users can see and do on particular databases and specific tables
- Access privileges are stored in the “mysql” database

```sql
> GRANT ALL PRIVILEGES ON db4bio.* TO superuser@”%” IDENTIFIED BY “password”;
> GRANT SELECT,INSERT ON db4bio.Data TO admin@”18.157.*.*” IDENTIFIED BY “pass2”;
```
CREATE DATABASE

- Allows you to create a new database on the database server (if you have permission)

> SHOW DATABASES;
> CREATE DATABASE go;
> SHOW DATABASES;
> USE go;
CREATE TABLE

• Translate an E-R diagram (schema) into a functioning database

```
CREATE TABLE Descriptions (
    gbId VARCHAR(20) NOT NULL,
    description VARCHAR(100),
    PRIMARY KEY (gbId)
);
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>gbId</td>
<td>varchar(20)</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>varchar(100)</td>
<td>YES</td>
<td>NULL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CREATE TABLE Targets

| Field   | Type         | Null | Key | Default | Extra |
|---------+--------------+------|-----|--------+-------|
| affyId  | VARCHAR(20)  |      | PRI |         |       |
| gbId    | VARCHAR(20)  |      | PRI |         |       |
| species | VARCHAR(20)  | YES  |     | NULL    |       |

PRIMARY KEY (affyId, gbId)
ALTER TABLE

• Modify a table’s attributes
  – Attribute names, type, null, key, default
  – Add or drop attributes

> ALTER TABLE Data
  CHANGE level level DOUBLE;
> ALTER TABLE Data
  RENAME level expression;
> ALTER TABLE Data
  ADD PRIMARY KEY (exptId);
> ALTER TABLE Data
  DROP COLUMN affyId;
> ALTER TABLE Data
  ADD date TIMESTAMP;
> DROP TABLE Data;
• Finally, add data into tables

  > INSERT INTO Data (level, exptId, affyId)  
    VALUES (215, "hs-hrt-1", "100008_at");  
    EXPLICIT ORDER

  > INSERT INTO Data  
    VALUES ("100008_at", "hs-hrt-1", 215);  
    IMPLIED ORDER

  > INSERT INTO Data2 (affyId2,level2)  
    SELECT Data.affyId, Data.level  
    FROM Data  
    WHERE Data.level < 250;  
    DATA COPYING
DELETE FROM

• Delete data from tables
• Similar syntax as SELECT

> DELETE FROM Data
    WHERE exptId="hs-hrt-1";

> DELETE FROM Sources
    WHERE exptId= "hs-hrt-1";

BE CONSISTENT
UPDATE

• Modify data already stored in a table
• Again, similar syntax as SELECT

> UPDATE Data
  SET exptId="hs-hrt-2"
  WHERE exptId="hs-hrt-1";

> UPDATE Source
  SET exptId= "ms-hrt-1", source="Mm"
  WHERE exptId="hs-hrt-1";

> UPDATE Data
  SET level=level*1.27
  WHERE exptId="hs-hrt-1";

MODIFY

FIX

INTERNAL
“NORMALIZATION”
LOAD DATA And Export

• Read rows from a text file (in the current directory) into a table and vice versa

> LOAD DATA LOCAL INFILE “data.txt”
  INTO TABLE db4bio.Data
  FIELDS TERMINATED BY ‘\t’
  LINES TERMINATED BY ‘\n’;

> LOAD DATA LOCAL INFILE “data.txt”
  INTO TABLE db4bio.Data;

> SELECT * INTO OUTFILE “data.txt”
  FIELDS TERMINATED BY ‘,’
  FROM Data;

Standard line ends:
Macintosh = ‘\r’
Windows = ‘\r\n’

Assumes tab-delimited file, with lines ending in “\n”

But need access to computer with MySQL
LOAD DATA warnings

mysql> LOAD DATA LOCAL INFILE "Hs_sources_test.txt"
    -> INTO TABLE Sources;
Query OK, 4 rows affected, 3 warnings (0.00 sec)
Records: 4 Deleted: 0 Skipped: 0 Warnings: 3

mysql> SHOW warnings;
+---------+----------------------------------------------------+
| Level   | Code | Message                                           |
|---------+------|---------------------------------------------------|
| Warning | 1265 | Data truncated for column 'exptId' at row 3       |
| Warning | 1265 | Data truncated for column 'exptId' at row 4       |
| Warning | 1262 | Row 4 was truncated; it contained ---             |
+---------+------|---------------------------------------------------|
3 rows in set (0.00 sec)

mysql> LOAD DATA LOCAL INFILE "Hs_sources_test.txt"
    -> INTO TABLE Sources;
Query OK, 0 rows affected, 3 warnings (0.00 sec)
Records: 4 Deleted: 0 Skipped: 4 Warnings: 3
Automating Repetitive Tasks

• Use .SQL files to perform SQL commands automatically

• Automatically create a series of tables

  % mysql -h hebrides.wi.mit.edu -u guest -p -D databasename < create.sql

• Feed a complicated query to the database and receive the results in a text file

  % mysql -h hebrides.wi.mit.edu -u web -p -D db4bio < query1.sql > query1.out
Summary

- Design databases with E-R diagrams
- Data mine using combinations of SELECT/FROM with WHERE, GROUP BY, HAVING, ORDER BY, and aggregates
- Create and implement databases
- Input and output data from databases
- Modify existing data within databases
Advanced topics

• Query optimization (adding indexes)
• Dates and times
  – all expected functionality
• Mathematics functions: logs, trig, etc.
• “String” (text) functions
  – substring, concatenate, replace, case change, etc.
• Nested queries
  – SELECT * FROM Ontologies WHERE linkId IN
    (SELECT linkId FROM LocusLinks
     WHERE gbId LIKE “A82%”);
Where To Go From Here?

- Consult SQL And MySQL Resources
  - http://www.mysql.com
  - Tutorial, Reference Manual
- Graphical interfaces to MySQL
  - DBDesigner (free)
  - MySQL Administrator
  - SQL4XManagerJ (inexpensive)
  - Visio (Microsoft)
  - Visual Case (expensive)
- Ensembl databases with open access
- Sources of data to build your own:
  - UCSC Bioinformatics; Gene Ontology; Entrez Gene
Course Goals

• Conceptualize data in terms of relations (database tables)
• Design relational databases
• Use SQL commands to extract data from (mine) databases
• Use SQL commands to build and modify databases
Exercises

• Create tables
• Input data
• Modify/delete particular data

• Accessing your own database:
  mysql -u username -p -D username
  -h hebrides.wi.mit.edu