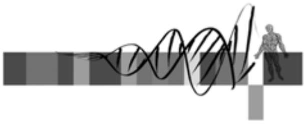


Relational Databases for Biologists: Efficiently Managing and Manipulating Your Data

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

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

```
+-----------------+----------+-------+
| affyId          | exptId   | level |
+-----------------+----------+-------+
| AFFX-MurIL2_at  | hs-cer-1 |    20 |
| AFFX-MurIL10_at | hs-cer-1 |     8 |
| AFFX-MurIL4_at  | hs-cer-1 |    77 |
| AFFX-MurFAS_at  | hs-cer-1 |    30 |
| AFFX-BioB-5_at  | hs-cer-1 |   258 |
+-----------------+----------+-------+
```

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

```
+---------+
| species |
+---------+
| Hs      |
| Mm      |
+---------+
```

WHERE And ORDER BY

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

```
+--------+-----------+-----------+
| linkId | ntRefSeq  | aaRefSeq  |
+--------+-----------+-----------+
|     50 | NM_001098 | NP_001089 |
|     51 | NM_004035 | NP_004026 |
|     52 | NM_004300 | NP_004291 |
|     53 | NM_001610 | NP_001601 |
|     54 | NM_001611 | NP_001602 |
+--------+-----------+-----------+
```

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

```
+--------+-----------+-----------+
| linkId | ntRefSeq  | aaRefSeq  |
+--------+-----------+-----------+
|     70 | NM_005159 | NP_005150 |
|     81 | NM_004924 | NP_004915 |
|     91 | NM_004302 | NP_004293 |
|     86 | NM_004301 | NP_004292 |
|     52 | NM_004300 | NP_004291 |
+--------+-----------+-----------+
```

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;
```

```
+-------------+------+-------+
| affyId      | min  | max   |
+-------------+------+-------+
| 100047_at   |   20 |  7784 |
| 100068_at   |  414 |  5883 |
| 100069_at   |  616 |  6349 |
| 100329_at   |   20 | 21455 |
| 100342_i_at |  786 |  7931 |
+-------------+------+-------+
```

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

```
+----------+-------------+
| gbId     | num_affyIds |
+----------+-------------+
| J04423   |          14 |
| AC002397 |          12 |
| AF109905 |           9 |
| AF100956 |           9 |
| AL031228 |           8 |
+----------+-------------+
```

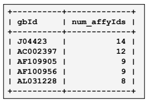
Table Joining

```
> 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;
```

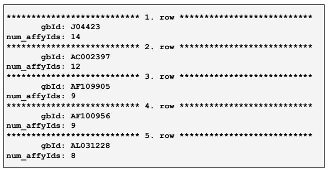
```
+-----------+-------------------------------+
| uId       | GO_description                |
+-----------+-------------------------------+
| Hs.373554 | calcium ion binding           |
| Hs.74561  | protein carrier               |
| Hs.155956 | arylamine N-acetyltransferase |
| Hs.2      | arylamine N-acetyltransferase |
| Hs.234726 | serine protease inhibitor     |
+-----------+-------------------------------+
```

Output Formats

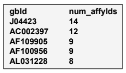
- Query from MySQL prompt
- Ending query with \G
  (in place of ‘;’)
- mysql < q.sql
  – tab-delimited output



```
+----------+-------------+
| gbId     | num_affyIds |
+----------+-------------+
| J04423   |          14 |
| AC002397 |          12 |
| AF109905 |           9 |
| AF100956 |           9 |
| AL031228 |           8 |
+----------+-------------+
```



```
*************************** 1. row ***************************
       gbId: J04423
num_affyIds: 14
*************************** 2. row ***************************
       gbId: AC002397
num_affyIds: 12
*************************** 3. row ***************************
       gbId: AF109905
num_affyIds: 9
*************************** 4. row ***************************
       gbId: AF100956
num_affyIds: 9
*************************** 5. row ***************************
       gbId: AL031228
num_affyIds: 8
```



```
gbId        num_affyIds
J04423      14
AC002397    12
AF109905    9
AF100956    9
AL031228    8
```

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

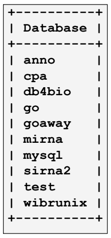
> GRANT ALL PRIVILEGES ON db4bio.* TO superuser@”%” IDENTIFIED BY “password”;

> 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)

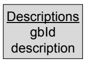
> SHOW DATABASES;
> CREATE DATABASE go;
> SHOW DATABASES;
> USE go;



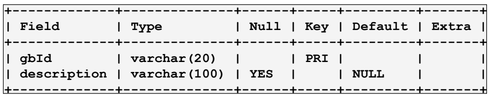
```
+----------+
| Database |
+----------+
| anno     |
| cpa      |
| db4bio   |
| go       |
| goaway   |
| mirna    |
| mysql    |
| sirna2   |
| test     |
| wibrunix |
+----------+
```

CREATE TABLE

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

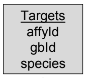


```
> CREATE TABLE Descriptions (
  gbId          VARCHAR(20)   NOT NULL,
  description   VARCHAR(100),
  PRIMARY KEY (gbId)
  );
```

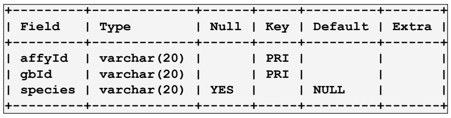


```
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| gbId        | varchar(20)  |      | PRI |         |       |
| description | varchar(100) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
```

CREATE TABLE



```
> CREATE TABLE Targets (
  affyId        VARCHAR(20)   NOT NULL,
  gbId          VARCHAR(20)   NOT NULL,
  species       VARCHAR(20),
  PRIMARY KEY (affyId, gbId)
  );
```



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

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;
```

INSERT INTO

- Finally, add data into tables

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

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

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

DELETE FROM

- Delete data from tables
- Similar syntax as SELECT

```
> DELETE FROM Data
  WHERE exptId=“hs-hrt-1”;

> DELETE FROM Sources                   BE CONSISTENT
  WHERE exptId= “hs-hrt-1”;
```

UPDATE

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

```
> UPDATE Data                                     MODIFY
  SET exptId=“hs-hrt-2”
  WHERE exptId=“hs-hrt-1”;

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

> UPDATE Data                                     INTERNAL
  SET level=level*1.27                            “NORMALIZATION”
  WHERE exptId=“hs-hrt-1”;
```

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’;
```

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

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

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

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

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- uusername- p- Dusername
- hhebrides.wi.mit.edu
```