Introduction "Structured Query Language" (SQL)

John Porter

University of Virginia

Why use SQL?

- Provides the tools needed to manage relational databases including:
  - Creating Tables
  - Adding Data
  - Queries / Searches
- It's a STANDARD! – multiple vendors produce products that support SQL queries

SEEK KNB NCEAS OBFS LTER NSF VCR LTER

Standards – A Caveat

- Just because there are standards for SQL implementations does not mean that all databases will have all the capabilities in the SQL standard.
- Most relational databases implement some non-standard extensions or lack some features of the full standard

Examples of Variation

- MiniSQL - implements only a critical subset of SQL commands
- MySQL – fairly compatible - no sub-selects (nested selects)
- Postgres – not fully standardized, object extensions
- **“The wonderful thing about standards is that there are so many of them to choose from” - anonymous**

Critical SQL Commands

- Table Management
  - Create Table
  - Drop Table
- Editing
  - Insert
  - Update
  - Delete
- Query
  - Select

There are many other commands, but these six will allow you to do almost anything you need to do

Create Table

```
CREATE TABLE mytable (
name CHAR(40) NOT NULL,
age  INT )
```

- Creates a table named “mytable” with two fields
  - A required character field called “name”
  - An optional numeric (integer) field called “age”

Insert

```
INSERT INTO mytable (name,age)
VALUES ('George',20)
```

- Inserts a data row into the table
  - "name" is set to "George"
  - "age" is set to 20

Select

```
SELECT name,age FROM mytable WHERE
age = 20
```

- Searches the table for rows where "age" is 20 and returns the associated name and age. This query resulted in:

```
+--------+------+
| name   | age  |
+--------+------+
| George |   20 |
+--------+------+
1 row in set (0.02 sec)
```

Update

```
UPDATE mytable SET age=21
  WHERE name LIKE 'George'
```

- Searches the table for rows where "name" is "George" and sets age to 21. Note: if we had more than one row with name "George" all would be set to age=21.

```
+--------+------+
| name   | age  |
+--------+------+
| George |   21 |
+--------+------+
1 row in set (0.02 sec)
```

Delete (a row from a table)

```
DELETE FROM mytable WHERE name
  LIKE 'George'AND age = 21
```

- Searches the table for rows where "name" is "George" and age is 21 and deletes them

Drop Table (delete a table)

```
DROP TABLE mytable
```

- Completely eliminates table "mytable." All data in the table is lost.

Putting the Relations in Relational Database

- SELECT statements are not restricted to single tables. For example:

```
SELECT DISTINCT
mytable.age, yourtable.address
FROM mytable, yourtable
WHERE mytable.name LIKE
  yourtable.name
```

Multi-table selects create a “join”

Relational SELECT

```
SELECT DISTINCT
mytable.age, yourtable.address
FROM mytable, yourtable
WHERE mytable.name LIKE yourtable.name
```

- Accesses two different tables: “mytable” and “yourtable”
- Returns “age” from mytable, and “address” from yourtable where the “name” field in the two tables match.
- DISTINCT means that if the same age and address shows up in multiple rows, only the first instance will be displayed.

Why SQL?

- Despite its power to manipulate data, SQL makes a poor user interface
  - Few ecologists will want to take the time to learn SQL
  - Effective use also requires knowledge of the underlying fields and tables
- For this reason, most SQL is imbedded into programs where it is hidden from the users

Example Program

- This example program uses PHP to talk to a MYSQL relational database
- The details of each step will differ between databases and languages, but will share many similarities
- Here we insert information from a web form into a database and retrieve an observation number for later use.

A Brief Orientation to PHP

- PHP is a language that can be embedded in web pages
- Variables start with $
  - E.g., **`$myVariable`**
- Arguments to Functions are in parentheses
  - E.g., **`sin($theta)`** returns the sine of the value stored in the variable **$theta**
- Statements end with a semicolon **;**
  - Lines don't matter

PHP Orientation

- `$_REQUEST[myFieldName]`
  returns the value of the web form field called: myFieldName

Sample program

```
$hello1="Hello";
$hello2= $hello1 . " World";
print($hello2);
```

Period (.) is a string concatenation operator in PHP

Steps in a PHP Program

- Make a Connection to the database server

```
$link =
  mysql_connect("data.vcrlter.virginia.edu
  “, “myID") or die("Could not connect");
```

- Select the Database on that server to use

```
mysql_select_db("www") or die("Could not
  select database");
```

Steps in a Program

- Prepare a query (here an INSERT statement) for execution:

```
$query = "insert into waiver
  (date_req,station,name,isVert,healthtype)
values('" . date("Y-m-d") . "',
 '$_REQUEST[station]',
 '$_REQUEST[fullname]',
 'Y',
 $_REQUEST[health])";
```

Note: **Date** is a function that returns the current date in the format specified

Steps in a Program

- Run the query we stored earlier:

```
$result = mysql_query($query) or
  die("Unable to log waiver
  creation, Query failed");
```

- The variable $result contains the query results in a complex form that includes all the rows and field values

Steps in a Program

- Prepare and run a query to get a copy of the “waiver_num” value. Here we use the MYSQL “max” function to return the highest value of waiver_num

```
$query = "select max(waiver_num) as
waiver_num from waiver";

$result = mysql_query($query) or
die("Unable to get waiver number, Query
failed");
```

Steps in a Program

- Convert the $result of the query into variables that PHP can use

```
extract(mysql_fetch_assoc($result));
```

- This creates the variable $waiver_num for use in PHP programs

Steps in Program

- Close our link to the MYSQL server

```
mysql_close($link);
```

```
/* Connecting, selecting database */
   $link = mysql_connect("data.vcrlter.virginia.edu", “myID")
       or die("Could not connect");
   mysql_select_db("www") or die("Could not select
database");

   /* Performing SQL query */
$query = "insert into waiver
(date_req,station,name,isVert,healthtype)
values('" . date("Y-m-d") . ','$_REQUEST[station]',
‘$_REQUEST[fullname]','Y', $_REQUEST[health])";

   $result = mysql_query($query) or die("Unable to log waiver
creation, Query failed");

$query = "select max(waiver_num) as waiver_num from
waiver";
$result = mysql_query($query) or die("Unable to get waiver
number, Query failed");
extract(mysql_fetch_assoc($result));
   mysql_close($link);
```

Open connection

Select database

Set up insert

Run Insert Query

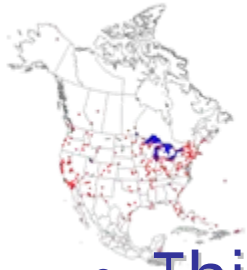
Set up query

Run Query

Store result as PHP variable

SQL Exercise

- Now it's time for you to try out your SQL skills using the web pages:
- http://www.sqlcourse.com/
  - Do all
- http://www.sqlcourse2.com
  - Do part 10 (table joins)



Acknowledgements

- This material is based upon work supported by:
- The National Science Foundation under Grant Numbers: 0080381, 0129792 ,9980154, 0225676 and 0072909.
- Collaborators: University of New Mexico (Long Term Ecological Research Network Office), NCEAS (UC Santa Barbara), San Diego Supercomputer Center, University of Kansas (Center for Biodiversity Research), University of Virginia, University of California Berkeley (Hastings Biological Station), University of Wisconsin
- The Andrew W. Mellon Foundation