**SQL Basics**

SQL (Structured Query Language) is a database sublanguage for querying and modifying relational databases. It was developed by IBM Research in the mid 70's and standardized by ANSI in 1986.

**Terminology**

* Database = a collection of tables
* Tables = a collection of rows and columns, called 'fields'.
* Field = named columns in a table that have a specific type
* Relational database = databases with tables that contain related fields

SQL can be divided into two parts: The Data Manipulation Language (DML) and the Data Definition Language (DDL). The query and update commands form the DML part of SQL:

* **SELECT** - extracts data from a database
* **UPDATE** - updates data in a database
* **DELETE** - deletes data from a database
* **INSERT INTO** - inserts new data into a database

The DDL part of SQL permits database tables to be created or deleted. It also define indexes (keys), specify links between tables, and impose constraints between tables. The most important DDL statements in SQL are:

* **CREATE DATABASE** - creates a new database
* **ALTER DATABASE** - modifies a database
* **CREATE TABLE** - creates a new table
* **ALTER TABLE** - modifies a table
* **DROP TABLE** - deletes a table
* **CREATE INDEX** - creates an index (search key)
* **DROP INDEX** - deletes an index

The exact syntax of some of these commands can be found in the table below.

|  |  |
| --- | --- |
| **Command** | **Syntax** |
| CREATE | CREATE TABLE table\_name (col\_name col\_type [not null]  [,col\_name col\_type [not null]])CREATE [UNIQUE] INDEX index\_name ON table\_name  (field\_name [,field\_name]) |
| DROP | DROP TABLE table\_nameDROP INDEX index\_name FROM table\_name |
| INSERT | INSERT INTO table\_name [(column [,column]) VALUES (value [,value])  |
| SELECT | SELECT [table.]column [,[table.]column]  FROM table [,table]  [WHERE [table.]column OPERATOR VALUE  [AND | OR [table.]column OPERATOR VALUE]]  [ORDER BY [table.]column [DESC] [,[table.]column [DESC]]  |
| DELETE | DELETE FROM table\_name  WHERE column OPERATOR value [AND | OR column OPERATOR value ] |
| UPDATE | UPDATE table\_name SET column=value [, column=value]  WHERE column OPERATOR value [AND | OR column OPERATOR value ] |

OPERATOR can be <, >, =, <=, >=, <>, or additional ones such as LIKE

VALUE can be a literal value or a column name

TYPE can be one of the following:

|  |  |
| --- | --- |
| **Type** | **Description** |
| char(len) | String of characters (or other 8 bit data) |
| varchar(len) | String of characters of given length or longer. Has performance penalty |
| text(len) | Variable length string of characters (or other 8 bit data) The defined length is used to indicate the expected average length of the data. Any data longer than the specified length will be split between the data table and external overflow buffers. Note: text fields are slower to access than char fields and cannot be used in an index nor in LIKE tests. Some |
| int | Signed integer values |
| real | Decimal or Scientific Notation real values |
| date | Date values in the format of ‘DD-Mon-YYYY’ such as ‘1-Jan-1997’ |
| time | Time values stored in 24 hour notation in the format of ‘HH:MM:SS’ |

**MySQL**

MySQL is a particular version of a relational SQL-compatible database known for its efficiency. It is widely used and free for non-commercial applications. See <http://www.mysql.com/> for details, in particular for downloading the free community edition. Other widely used SQL databases are Oracle, Microsoft SQL-Server, and Postgres.

MySQL comes in three parts: a server program, that has already been setup on Sciris, a text-based client program, and a number of utilities. To start a MySQL session, login to Sciris as usual and invoke the MySQL client by typing:

mysql –utestuser –p testuser

which will start the client, connect as user 'testuser' to the server, prompt you for the password, and use the database named 'testuser'. The databases and permissions have been already setup for you.

**Tasks**

* Create a table named people with fields named 'first', 'last', and 'email'. Note that this table can only store one email per person. Add a few persons and search.

You would use, for example, the following SQL commands (in the mysql client program) to create a new table, insert a few records, and perform some searches.

create table people (first varchar(10), last varchar(20), email varchar(30));

insert into people values ('Bert', 'Wachsmuth', 'wachsmut@shu.edu');

insert into people values ('Jane', 'Doe', 'doejane@nowhere.com');

insert into people values ('John', 'Doe', 'doejohn@nowhere.com');

select \* from people;

select last, first from people where first='Bert';

select email from people where first like 'ber%';

* Drop the people table again. Create two tables, one named people with fields named 'id', 'first', and 'last', and one named 'emails' with fields 'id' and 'email'. This collection can store arbitrary emails per person. Add a person with two or more emails and search.

The above table can only hold one email per person, unless you duplicate a name. Most people have more than one email, though, so we need to create two tables linked via an id field to allow one person to have an unlimited amount of emails. The first table, 'people', will contain a first and last name, together with a unique id number. The second table, 'emails', will contain email addresses together with an id number that links an email to one of the persons. Here is an appropriate MySQL session to accomplish this:

drop table people;

create table people

 (id int unique not null, first varchar(10), last varchar(20));

create table emails

 (id int not null, email varchar(40));

insert into people values (1, 'Bert', 'Wachsmuth');

insert into people values (2, 'Jane', 'Doe');

insert into people values (3, 'John', 'Doe');

insert into emails values (1, 'wachsmut@shu.edu');

insert into emails values (1, 'wachsmut@gmail.com');

insert into emails values (1, 'webmaster@mathcs.org');

insert into emails values (2, 'doejane@nowhere.com');

select \* from people, emails;

select \* from people, emails where people.id = emails.id;

select people.first, people.last,emails.email from people,emails

 where people.id=emails.id and people.first='Bert';

* Update a few names and emails in the various tables. Retrieve them using the select statement. Delete an address completely.

update emails set email='wachsmut@googlemail.com'

 where email like '%gmail%';

select people.first, people.last,emails.email from people,emails

 where people.id=emails.id and people.first='Bert';

delete from people where id = 3;

* Write a shell script to extract all names and email addresses from the database and display them nicely formatted. You can feed commands to the mysql client through a pipe, you can remove the first line via sed, and you can format the fields of the output using awk.

Note that this script is longer than the command line window allows, i.e. it does not fit on one line. To make it look reasonable, you can use a single "backslash" character at the end of a line to signal that the command is continuing in the next line. So, with that in mind our script would look as follows (note the backslashes at the end of each line except for the last one, which is the end of our script) – of course you need to replace the PASSWORD with your true password.

echo "select people.first,people.last,emails.email \

 from people, emails \

 where people.id = emails.id" \

 | mysql -utestuser -pPASSWORD testuser \

 | sed '1 d' \

 | awk '{print "Name = " $1 " " $2 ", Email: " $3}'

* Write a CGI script to extract all names and email addresses and format them as HTML. Make sure each name is actually a reference to the email address, so that when that name is selected, and a 'go' button is clicked, the email address(es) associated with that person will be returned.

We clearly need to create a file in our /home/testuser/httpd/cgi-bin directory and mark it as executable (do not forget). Let's name our file **names**, so we would say "**pico names**" (and after we create the file don't forget to do "**chmod u+x names**"). Our script needs to create the mandatory portions of an HTML file, extract the names and id numbers from the database, displays names inside a form, where each name is preceded by a radio button with value being the id of that name. When the 'submit' button is pressed, the id is passed on to a second script that will do the email search and format the answer. That second script is your homework assignment. Here is the complete script, including some variables so that you can easily adjust the script to your database. And don't forget to enter the proper PASSWORD!

**#!/bin/sh**

# Author: Bert Wachsmuth

# Date: Feb 2011

#

# This script extracts names from an SQL database and presents them to the

# user preceeded by radio buttons. If the user selected a name and clicks

# the submit button, the id of the selected name is passed to a second

# script for the email lookup.

# mysql access command, including the right username, password, and datatbase

**MYSQL\_CLIENT="mysql -utestuser -pPASSWORD testuser"**

# SQL query to extract the names and ids for the list of people

**SQL\_QUERY="select id, first, last from people order by last"**

# Writing the standard HTML content type

**echo "Content-type: text/html"**

**echo**

# Now the standard HTML header

**echo "<html>"**

**echo "<head><title>Address List</title></head>"**

**echo "<body>"**

**echo "<h1>My Address List</h1>"**

# The form definition (note the name of the script to search for emails)

**echo "<form method='get' action='/cgi-bin/name-email'>"**

**echo "<blockquote>"**

# The names inside a blockquote. Each name is formatted using awk to be

# preceeded by a radio button with the proper ID as value. Note that the

# formatting command looks messy because the quotes inside the 'print'

# statement must be escaped, i.e. to print a quote we need to use the

**echo $SQL\_QUERY \**

 **| $MYSQL\_CLIENT \**

 **| sed '1 d' \**

 **| awk '{print "<input type=\"radio\" \**

 **name=\"id\" \**

 **value=\"" $1 "\">" $2 " " $3 "<br />"}'**

# we are done with the main script and need to close our form

**echo "</blockquote>"**

**echo "<input type='submit' value='Search'>"**

**echo "</form>"**

# Finally the standard HTML footer

**echo "</body>"**

**echo "</html>"**

**Homework**

Your homework is to create the missing "names-email" script. It needs to:

* Extract the id value from the submitted data. You need to look up how we did that via sed and a regular expression in one of our earlier scripts.
* Create a search for the emails and name for that ID in the appropriate tables, which are related by the id field
* Format the results of the search properly. Note that it would be best to embed each email inside an appropriate "mailto" anchor. In other words, if an extracted email was wachsmut@shu.edu, you should try to generate the line: <a href="mailto:wachsmut@shu.edu">wachsmut@shu.edu</a>
* Put everything in between the proper header and footer code for html documents

To see how it all is supposed to work, check

<http://sciris.shu.edu:7770/cgi-bin/names>