Using Databases in the Web of Things Environment

During this class, you will have accounts on the tlab, the Wilkinson lab, and on two of the resources in the Web Of Things (WOT) environment: Murphy and the Parkinson cluster. Murphy has been configured to support EECS 339 with relational databases, while Parkinson has been configured to support distributed databases.

Tlab? Wilkinson Lab? WOT? Murphy? Parkinson?

The Tlab ("Teaching Lab"), Tech F252, consists of workstation machines (tlab-01 and up) with very large monitors that you can use locally or remotely, and a server machine (tlab-login). This is the lab in the Tech building that is adjacent to the bridge to the Ford building.

The Wilkinson lab ("Wilk Lab"), Tech M338, consists of a range of workstation machines with very large monitors that you can use locally or remotely. Wilkinson also has huge amounts of project space and seating space to work collaboratively on a project.

The Web Of Things is a group of specialized servers, mobile phones, tablets, and sensor network nodes that are intended to support education in pervasive computing, sensing, and actuation. When not in use for a course, some of the mobile hardware is available to be checked out by students. The Web Of Things is supported by a Murphy Society grant.

For the class, you will be able to log into **murphy.wot.eecs.northwestern.edu.** Murphy is a Dell R410 with dual Xeon hexacore processors (24 hardware threads total), 128 GB of RAM, and 6 TB of RAID 5 storage. It's running the current release of Red Hat Enterprise Linux. The relational database engines that are installed include Oracle 11g Enterprise (the primary database we will use in the class), MySQL, and Postgres. Apache is installed and configured to serve individual user www directories, including CGI, and using suexec. Perl, Python, and PHP are installed and configured to support access to the Oracle and MySQL databases from CGI scripts.

During the class, you will also be able to log into **parkinson-mgmt.wot.eecs.northwestern.edu**, the head node of a small (11-node) cluster. Each cluster node is a Dell 1850 with 8 GB RAM and 300 GB of local storage, while the head is a Dell 2650 with 8 GB of RAM, and 2 TB of RAID 5 storage. All nodes run Red Hat Enterprise Linux. Parkinson is set up to support distributed "NoSQL" databases, in particular the Cassandra engine. While this is a small cluster, and 3 TB is a small distributed database workspace, the system is sufficient to help you learn about such systems.

Logging in

You can log into murphy and parksinon from anywhere using an ssh client. The username and password are the same as you would use for the Wilkinson Lab or Tlab.

You can also start and maintain a desktop session on murphy using NoMachine NX. Free clients are available for Windows, Mac, and Linux.

We assume in the class that you are using the "bash" shell, or a shell compatible with it. If not, you can contact systems support about changing your login shell, or you can run "bash" from your login shell to invoke bash.

What's in your home directory on Murphy?

Your home directory is the same as on the TLab or Wilkinson Lab machines – it's your shared home directory. If it does not already exist, you will want to create a subdirectory:

```
mkdir ~/www
chmod 711 ~/www
```

The contents of this directory (~/www) and all of its subdirectories will be served by apache as http://murphy-wot.eecs.northwestern.edu/~you. The directory is also CGI-enabled in Apache's configuration files. This means that any file you place in there with a .cgi, .pl, .py, or .php extension that is executable will be run by Apache when it is requested, instead of being simply sent verbatim to the web browser making the request. Apache has been configured with suexec, which means that your scripts will run as you.

If needed, we will also give you access to a directory in /raid, the large local raid array on the machine. If you want to import or export large files, you will need this.

Accessing Oracle on Murphy

You will want to grab a copy of ~pdinda/339/HANDOUT/oraenv.sh. This is a shell a shell script that contains the minimum set of environment variables that need to be set to make Oracle's client tools work. You need to source this file (source ~/oraenv.sh) before you run any Oracle tools. You may want to source this in your ~/.bashrc file so that this happens automatically when you log in.

To access Oracle, you will need an Oracle account. We will create this for you during the class and let you know the account name and password. For now, we refer to these as "youora" and "orapasswd".

Oracle is a giant system with lots of quirks that have been acquired over its long history and its design goals of being largely platform-independent. In class, and in the following, we will ony touch the surface in terms of the tools available to use and manage Oracle.

SQL*Plus is the basic Oracle client, and the one I will typically use in class. It provides a command-line interface to essentially all of the Oracle database functionality. To run SQL*Plus, do the following:

```
source ~you/oraenv.sh
sqlplus youora/orapasswd
```

If everything is OK, you'll see something like this:

```
SQL*Plus: Release 11.2.0.1.0 Production on Wed Sep 12 11:53:59 2012

Copyright (c) 1982, 2009, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit
Production
With the Partitioning, OLAP, Data Mining and Real Application Testing
options

SQL>
```

That last line is the SQL*Plus interactive prompt. You can now type in a SQL*Plus command ("help" to learn what they are), a SQL statement, or a PL/SQL block. For example:

```
SQL> create table students (id number, lastname varchar(32));
```

will create a table in you account.

Note that in addition to the tables in your own account, we have arranged it so that you can read tables in the class account. For example, to find the number of stock price records, you would say

```
SQL> select count(*) from cs339.stocksdaily;
```

You will often want to write a whole bunch of SQL or PL/SQL in a file and then have SQL*Plus evaluate it. Here's how:

```
SQL> @file.sql
```

You can also do the same thing from the command line:

```
sqlplus youora/orapasswd @file.sql
```

XEmacs (run as "xemacs") has a mode that can run SQL*Plus for you, allowing you to edit using Emacs instead of with the primitive line-editing possible in SQL*Plus. To do this, run "M-x sql-oracle RET". "M-x" means "meta-X", which is usually typed as either ALT-X or by hitting the ESC key and then x. "C-x" means "Control-X". You

may find it useful to deal with the Unix shell in the same way "M-x shell RET". You'll also find that Emacs understands both .sql and .pl files.

Another sometimes useful tool is SQL*Loader (sqlldr). This tool is used to efficiently bulk load data into an Oracle database.

You can access Oracle from Perl, Python, and PHP. For examples on how to set these up, look in ~pdinda/339/HANDOUT/test or ~pdinda/www/test

Using MySQL on Murphy

MySQL is relatively straightforward to access on Murphy. For the class, we have established two databases, cs339data and cs339play, as well as a shared class account. The shared class account has read permissions on cs339data and both read and write permissions on cs339play. We will not create individual student accounts for MySQL automatically, but if you need on, please contact us.

To access mysql using the shared class account, run

```
mysql –u cs339 –p
```

After you type the password (which we will give you separately), you will see something like this:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 29
Server version: 5.1.61 Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
```

At the mysql prompt, you can now type sql commands. You'll need to first select a database:

```
Mysql> use cs339play;
```

Then you can manipulate it:

```
mysql> create table students (id int, lastname varchar(32));
```

Similar to Oracle, XEmacs has a mode that can run mysql for you, allowing you to edit using. To do this, run "M-x sql-mysql RET".

Bulk data load in mysql is usually done using mysql, with the "load data local infile" command.

You can access MySQL from Perl, Python, and PHP. For examples on how to set these up, look in ~pdinda/339/HANDOUT/test or ~pdinda/www/test

What's in your home directory on Parkinson?

Your home directory on the Parkinson machines is distinct from the home directory on Murphy, Tlab, or Wilkinson lab machines. However, you will see the same home directory (served from parkinson-mgmt.wot.eecs.northwestern.edu) on all the Parkinson machines. Your home directory is stored in the raid array attached to parkinson-mgmet.

In this class, you will generally just log into parkinson-mgmt in order to run Cassandra. Cassandra handles data storage internally.

Accessing Cassandra on Parkinson-mgmt

To access Cassandra, ssh into parkinson-mgmt and run `cassandra-cli`. This is the basic command-line interface. It will communicate to the Cassandra instance on Parkinson-mgmt, which works in coordination with the instances running on all the other nodes. Actions on keyspaces will spread out over the cluster according to the involved keys.

You will probably spend most of your time using "CQL", which is accessed by running `cqlsh`.

Currently, all users have maximum permissions on Casandara. This means you can read and write each others data. Please use discretion. If this turns out to be an issue, we will add authentication, etc. The Cassandra ports are not visible outside of the Parkinson cluster – you must be logged into a Parkinson machine to access them.