Using Oracle in the TLAB

During this class, you will have Linux accounts on both the tlab workstation machines (tlab-01 and up) and on the tlab server machine (tlab-login). The server machine is running the Apache web server and the Oracle database. Your server account includes a directory that is served by the web server and is CGI-enabled. Each of you will also have a database account that will give you a private area to create tables, indexes, sequences, procedures, packages, etc.

Logging in

You can log into the tlab workstations and the server from anywhere on campus using an ssh client that support ssh2. Once you log into a machine with the username and password we'll supply you, you'll land in you home directory, ~you. Your home directory is the same regardless of which machine you log into. Generally, however, you'll find it easiest to log into the server, as that is where Apache and Oracle will be running. Unless you ask otherwise, your login shell is /bin/bash. All shell scripts in this class will assume that shell.

What's in your home directory?

In ~you, you will see ~you/public_html. The contents of this directory will be served by apache as http://tlab-login/~you. The directory is CGI-enabled in Apache's configuration files. This means that any file you place in there with a .cgi or .pl extension, and with the right permissions (chmod 755 file) will be run by Apache when it is requested, instead of being simply sent verbatim to the web browser making the request. When invoked in this way, your scripts will run as user apache, group apache.

~you/oraenv.sh is 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 ~you/oraenv.sh) before you run any Oracle tools. You may want to source this in your ~you/.bashrc file so that this happens automatically when you log in. Note that because a CGI script will run as apache, not you, it is necessary to do something similar to ~you/oraenv.sh within your script. You can find a version of oraenv.sh in ~pdinda/HANDOUT.

Running SQL*Plus

SQL*Plus is the basic Oracle client. 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 you/orapasswd
```

orapasswd is your oracle password, which is different from your login password. "you" is also a different account, although, for convenience, we have given it the same name as your Linux user name. If everything is OK, you'll see something like this:

```
SQL*Plus: Release 10.1.0.2.0 - Production on Tue Sep 7 17:34:20 2004

Copyright (c) 1982, 2004, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining 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.

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 you/orapasswd @file.sql
```

Running SQL*Plus in Emacs

XEmacs (run as "xemacs") and GNU Emacs (run as "emacs") have 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.

Using Oracle from Perl

One way to use Oracle from Perl is to simply shell out SQL*Plus to do the work, such as:

```
use FileHandle;
open(SQL,"| sqlplus you/orapasswd") or die "Can't open sqlplus!";
SQL->autoflush(1);
print SQL "create table students (id number, lastname varchar(32);\n";
print SQL "quit\n";
close(SQL);
```

A much faster and elegant way to do this is to use Perl's DBI, the standard database interface. The advantage is that DBI is database-independent, meaning that you can take your Perl code that runs with Oracle and make it run with MySQL or DB2 or Postgres

very easily. There are some caveats on that “independence”, but it is essentially true. You generally also need to use DBI if you want to store binary data (“blobs”, images for example) in the database . Here is a short snippet showing an example of DBI in use:

```
use DBI;
my $querystring=
   ”create table students (id number, lastname varchar(32))”;
my $dbh = DBI->connect("DBI:Oracle:",”you”,”orapasswd”)
          or die "Can't connect to database";
my $sth = $dbh->prepare($querystring)
          or die "Can't prepare $querystring because”.$dbh->errstr;
$sth->execute()
  or die "Can't execute $querystring because of ".$dbh->errstr;
my @data;
while (@data=$sth->fetchrow_array()) {
    print join(“\t”,@data), “\n”;
}
$sth->finish();
$dbh->disconnect();
```

That code snippet will work for any $querystring, including select statements. It will print the result rows as the return from the executing statement.

There is a lot to DBI. In fact, O’Reilly publishes a whole book on DBI. You can find out more on the web as well.

Take a look at ~pdinda/HANDOUT for some additional examples of using Oracle and Perl.

Using Oracle From Other Languages

Oracle, and most databases, can be used from multiple languages. For each language, there is a standard database interface API, which is usually very much like Perl’s DBI. Java’s interface is called JDBC, while the interface for C/C++ is typically ODBC.

Consulting Your Own Oracle, Ye Delphians

All of Oracle’s software (and there is a lot of it) is available for download from oracle.com. If you like, you can set up a similar configuration to ours on your own machine so that you don’t have to use the lab. Oracle is available for every platform known to man, including Linux and Windows. In essence, if you download it, it is licensed only for development use. If you want to deploy Oracle-based software, you need to pay the piper. IBM’s DB2 is available under similar constraints. Microsoft’s SQL Server is another commercial RDBMS. MySQL and Postgres are two open-source, free alternatives, although they have compromises.

If you have access to VMWare, we can provide you with a pre-configured Linux virtual machine that includes everything you need for this course all installed and running. Ask the TA or instructor for more information if you’re interested.

Please note that your projects will be graded only in the tlab-login environment. Also, understand that setting up an RDBMS can be a painful process, and we cannot provide any help if you decide to do it. Be careful if you run $ORACLE_HOME/sibylline_verses.sql.[1]

[1] Your instructor has a penchant for obscure references.