

Using ORACLE in the CSLab

Dr. Weining Zhang
Department of Computer Science
University of Texas at San Antonio

October 15, 2009

1 Introduction

A version of ORACLE, a popular Object-Relational Database Management System (ORDBMS), is installed on CSLab computers for students who are registered in CS database courses. The DBMS has many important features, such as standard SQL database language, interactive and programming interfaces, integrity and security enforcement, transaction processing, data import/export, and report generation. It can be used to create, query and update databases, to manage storage and users, and to build database application programs. The purpose of this article is to provide information on using ORACLE for database course work.

The rest of this article is organized as follows. Section 2 discusses the ORACLE user accounts. Section 3 discusses some online resources such as ORACLE manuals and sample databases. Section 4 discusses the SQL*PLUS command interface. Section 5 discusses the use of PRO*C/C++ language. Finally, Section 6 discusses the use of JDBC and SQLJ.

2 ORACLE User Account

Each ORACLE user will have a system (UNIX/Linux/Windows) account and an ORACLE account, typically with different user names and passwords. To use ORACLE, the user will first log into the user's system account and then log into the ORACLE account.

The user name and initial password of an ORACLE account is assigned according to following rules. The user name of a student will consist of the first letter of the first name, followed by up to 4 letters of the last name, and then followed by the first 3 digits of the course number. For example, the user name of Joe Smith in CS3743 is `jsmit374`. The initial password is in the form of `utsannnn`, where `nnnn` is the last 4 digits in the student's UTSA ID number. For example, if `@00456789` is the UTSA ID number, the initial password will be `utsa6789`. See Section 4.1 for information on how to log on to an ORACLE database through the SQL*PLUS command interface, and Section 4.3 for information on changing password using SQL statements.

2.1 Environment Variables Used by ORACLE

In order to access ORACLE databases, a user needs to define some environment variables in the user's UNIX/Linux configuration file, such as `.cshrc` or `.bashrc`, depending on the user's default shell.

For `csh` users, the `.cshrc` file should contain following lines.

```
setenv ORACLE_SID "acdb"
setenv ORACLE_BASE /oracle/u01/app/oracle
setenv ORACLE_DATA /oracle/u01/app/oracle/oradata
setenv PATH ${PATH}:$ORACLE_HOME
setenv LD_LIBRARY_PATH $ORACLE_HOME/lib32:$ORACLE_HOME:$LD_LIBRARY_PATH
setenv LD_LIBRARY_PATH_64 $ORACLE_HOME/lib:$LD_LIBRARY_PATH
setenv ORACLE_DOC $ORACLE_BASE/oradocs
```

For `bash` or `ksh` users, the following lines should be included in the corresponding `.bashrc` or `.kshrc` file.

```
export ORACLE_SID=acdb
export ORACLE_BASE=/oracle/u01/app/oracle
export ORACLE_DATA=/oracle/u01/app/oracle/oradata
export PATH=$PATH:$ORACLE_HOME/bin
export LD_LIBRARY_PATH=$ORACLE_HOME/lib32:$ORACLE_HOME:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH_64=$ORACLE_HOME/lib:$LD_LIBRARY_PATH
export ORACLE_DOC=$ORACLE_BASE/oradocs
```

Furthermore, if the user wants to use Java/JDBC or SQLJ to program a database application, the user may need to change environment variables `PATH`, `CLASSPATH`, and `LD_LIBRARY_PATH`. The `PATH` must contain `/usr/local/java/jdk/bin`; the `CLASSPATH` must contain

```
$ORACLE_HOME/jlib
$ORACLE_HOME/jdbc/lib/ojdbc14.jar
$ORACLE_HOME/sqlj/lib/translator.jar
$ORACLE_HOME/sqlj/lib/runtime12.jar
```

and the `LD_LIBRARY_PATH` needs to contain `$ORACLE_HOME/lib`.

2.2 Use of Database Resources

Each user is by default given a 5 MB space quota in the database for course assignments and projects. To effectively use the storage space, the user needs to keep the number of database tables under control. It is important to remove temporary tables, using SQL statement `DROP TABLE`, as soon as they are no longer needed. Also, users should exit ORACLE system appropriately using the `EXIT` command of `SQL*PLUS` to avoid leaving processes running in the background, which can significantly slow down the system.

3 Online Resources

3.1 ORACLE Manuals

The Oracle Technology Network web site¹ provides links to a set of ORACLE manuals. The following manuals are especially helpful for database course assignments and can be accessed through the “View Library” link from the main page of the online manual. Once get to the library page, click the “Books” tab and looking for books about SQL, PL/SQL, SQL*PLUS and Error messages.

- The description and suggested solution of SQL errors with error message starting with “ORA #####” can be found in the “Error Messages” book.
- Concepts, hints, and examples of PL/SQL can be found in the “PL/SQL User’s Guide and Reference” and the “Supplied PL/SQL Packages and Types Reference”.
- The syntax and features of ORACLE SQL language can be found in the “SQL Reference”.
- The use of triggers is discussed in the “Application Developer’s Guide - Fundamentals”.

Since most database textbooks provide examples based on the standard SQL, which may not work on ORACLE, it is important to consult the ORACLE SQL Reference manual for doing course projects and assignments

3.2 A Sample Database

Students of a database course can query a sample company database that contains six relations: employee, department, dept_locations, dependent, works_on, and project. To use SQL statements on these tables, a table name needs to be prefixed by “sample.”. For example one can use the following SQL statement to list all employees .

```
SELECT *  
FROM sample.employee;
```

4 Using SQL*PLUS

The SQL*PLUS command-line interface provides a command interpreter and a single command buffer. A user can enter SQL queries in SQL*PLUS and get answers on the screen of the monitor.

4.1 Start SQL*PLUS

To run SQL*PLUS, enter the following command at the operating system prompt, say \$ on a UNIX/Linux computer.

```
$ sqlplus
```

¹<http://www.oracle.com/pls/db102/homepage>

SQL*PLUS will prompt for user name and password. On a CSLab computer, the user should enter the ORACLE user name with an “@acdb” suffix, that is, *userName@acdb*, and followed by the password. For example, user Scott will enter the user name as *scott@acdb*. The “@acdb” suffix is required for the computer to connect to the ORACLE database server. If the suffix is missing, the user will get an ORACLE unavailable error message.

Once logged into SQL*PLUS, the screen prompt will change to

```
SQL>
```

which indicates that the system is ready to take SQL or SQL*PLUS commands.

4.2 Exit SQL*PLUS

To exit SQL*PLUS, enter the command

```
SQL> exit
```

The system will return to the operating system prompt.

4.3 Changing ORACLE Password

Soon after the first log in, a user should change the initial ORACLE password with the following SQL ALTER statement.

```
SQL> ALTER USER username IDENTIFIED BY newpassword;
```

where *username* is the ORACLE user name (without the “@acdb” suffix) and *newpassword* is the replacing password. The semicolon “;” tells the system to execute the SQL statement.

Alternatively, the user change the password using the following SQL GRANT statement.

```
SQL> GRANT CONNECT TO username IDENTIFIED BY  
      newpassword;
```

4.4 Get Help Inside of SQL*PLUS

While inside of SQL*PLUS, a user can obtain information about command and their usage using the `help` (or simply “?”) command. In addition, the user can query several system tables for useful information. For example, the following SQL statement lists tables and views that have been created by the user:

```
SQL> select * from cat;
```

where *cat* is the short name of a system table. As another example, the following SQL*PLUS command will list the columns of a table named *table1*, :

```
SQL> describe table1
```

Notice that, since `describe` is an SQL*PLUS rather than SQL command, a semicolon is not needed.

Each ORACLE database contains a large number of system tables and views, whose names can be listed by the following SQL statement.

```
SQL> select * from dictionary;
```

4.5 Manipulating Databases

Within SQL*PLUS, a user can create, alter, or drop tables, to insert tuples to a table, delete tuples from a table, update tuples in a table, and to query the database using SQL statements. The SQL*PLUS keeps the most recently entered SQL statement in a command buffer. A user may display, execute, or modify the SQL statement in this buffer. SQL*PLUS provides a set of commands for editing the command in the command buffer. The SQL statement in the command buffer can also be saved into an SQL script file (with the file extension `.sql`). SQL*PLUS also provides an `edit` command, which will the SQL statement in the command buffer into an external editor chosen by the user, such as `vi` or `emacs`, for editing. After the editing is completed, the user can exit the external editor in the normal way and return to SQL*PLUS. The SQL*PLUS command or SQL statement in the command buffer can be executed by entering command `run` at the SQL> prompt.

A user can create an SQL script using any text editor and execute the script within SQL*PLUS. For example, the following SQL*PLUS command will load and execute a file named `test.sql`.

```
SQL> @test
```

This method is useful if the same statements will be executed over and over again, for example, during debugging of an SQL statement or experimenting with a database creation.

4.6 Web Access to SQL*PLUS

Oracle SQL*PLUS can also be accessed over the Web at <http://malibu.cs.utsa.edu:5560/isqlplus>. This web link leads you to a login page where you need to fill in your Oracle user name (without `@cs`), your Oracle password, and “`cs`” as the connect identifier.

5 Using Pro*Languages

In addition to interactive user interfaces such as SQL*PLUS, ORACLE databases can also be accessed through programming interfaces, which allows SQL statements to be embedded in a program written in one of six host languages: Pascal, C/C++, PL/I, Fortran, Ada and Cobol. In ORACLE, embedded SQL results in six new languages, referred to as the Pro*Languages. They are Pro*Pascal, Pro*C/C++, Pro*PLI, Pro*Fortran, Pro*Ada and Pro*Cobol. (The current installation of ORACLE provides only Pro*C/C++.) ORACLE provides a *precompiler* to translate programs of each pro*language to programs in the corresponding host language.

5.1 Pro*C/C++ Program

A Pro*C/C++ program embeds SQL statements in C/C++ statements in a format specified in the *Pro*C/C++ Precompiler Programmer's Guide* (available online). Such a program, in a .pc file, needs to be compiled using two compilers: first the proc precompiler and then the cc or gcc/g++ host language compiler. The precompiler generates a C/C++ program from a Pro*C/C++ program by replacing embedded SQL statements with suitable C/C++ subroutine calls to the SQL run-time library. The host language compiler then compiles the resulting C/C++ program into object code. Finally the object code is linked with the ORACLE SQL library routines to generate the executable code.

A number of sample Pro*C/C++ programs are located in

```
$ORACLE_HOME/precomp/demo/proc.
```

These sample programs can be copied to, compiled, and executed in a working directory in of any user.

The simplest way to compile a sample Pro*C program is to copy it into a working directory and use the make command to build the executable file. (Notice that it may be necessary to modify a Pro*C/C++ program, for example, to use scott@acdb instead of scott as the user name.) For example, the following steps illustrate how to compile the program `example1.pc`.

```
cd workdir
cp $ORACLE_HOME/precomp/demo/proc/sample1.pc .
vi sample1.pc
(edit sample1.pc to change the user name
 from scott to scott@acdb)
make -f $ORACLE_HOME/precomp/demo/proc/demo_proc.mk sample1
```

The make command will create files `sample1.c`, `sample1.o` and an executable file `sample1`. To run the sample program, simply enter `sample1`.

The make command and `demo_proc.mk` file can also be used to compile user's own PRO*C/C++ programs. For example, the following command compiles a user Pro*C program called `foo.pc`:

```
make -f $ORACLE_HOME/precomp/demo/proc/demo_proc.mk EXE=foo \
  OBJS="foo.o" build
```

However, if a program needs some non-standard include-files or non-standard library, the user may have to copy the makefile `demo_proc.mk` into the working directory and change it accordingly.

5.2 Pro*C/C++ with Embedded PL/SQL

In addition to SQL statements, PL/SQL statements can also be embedded in a C/C++ program to create sophisticated applications, as illustrated by programs in `$ORACLE_HOME/plsql/demo`. Of particular interest are `sample5.pc` and `sample6.pc`, which can also be compiled using the `demo_plsql.mk` file (in the same directory) as previously described.

6 Using Java/JDBC and SQLJ

Similar to Pro*C/C++, SQL statements can also be embedded in Java programs, and in Java Server Pages (JSP). However, to enable a Java program to access an ORACLE database, the environment variable CLASSPATH must include jar files in \$ORACLE_HOME/jdbc/lib (in case of using JDBC) and in \$ORACLE_HOME/sqlj/lib (in case of using SQLJ). See Section 2 for appropriate settings of environment variables. While JDBC programs are normal Java programs that use Java jdbc package and are compiled with the ordinary javac command, SQLJ programs have the .sqlj file type and must be compiled using the special sqlj command. For example, an SQLJ program called myprog.sqlj will be compiled with the following command:

```
sqlj myprog.sqlj
```

which will result in two files: myprog.java and myprog.class. Both Java/JDBC and SQLJ programs should use the connection string “jdbc.oracle.thin@oracle:1521:acdb” together with ORACLE user name, without the “@acdb” suffix, and password to make connections to ORACLE databases. Furthermore, SQLJ allows the connection string to be placed in a separate file, typically named as connect.properties.

7 Web Access to Oracle Database

A PHP access to Oracle database can be made through a CS department web server. To set up a connection, the user should place PHP files in the dev_html directory in the user’s home directory (similar to placing HTML files in the public_html directory). To execute these PHP files from a web client, simply use the URL: `http://webdev.cs.utsa.edu/~<username>/<filename>`.

8 Final Words

It is without doubt that a lot of questions are unanswered by this article. However, my goal is to provide enough information to help students to start using ORACLE software. Of course, I am always open to suggestions that improves this article.