Info about Oracle

Scott D. Anderson

Spring 2001

The Oracle at Delphi, famous in Greek mythology, was known for being inscrutable as well as for giving information. I think you'll find that the software from Oracle Corporation shares these attributes. I'm not going to try to explain *all* of it; the following is just some selected facets.

1 Objects, Things, Stuff

The software is enormous and supplies a lot. In fact, it's a little like a whole computer system, with files, directories, users, passwords and things like that. Here, in no particular order, is a subset of the things it provides, some of which are interesting. It may help to see the analogies between these things and aspects of computer systems, which we're already familiar with.

1.1 Tablespaces

A tablespace is analogous to directories: a place to put data, particularly tables. The DBA can create more tablespaces, and usually will for production systems, but here are some built-in ones:

- System: where the system keeps its tables of users, roles, tablespaces and all that. Yes, all the information about the database is stored in the database, so you can inquire about it using the same SQL constructs you use to query your data.
- Rollback: used for ensuring that the database stays consistent; we'll talk more about it later.
- Temp: an area that the system uses for sorting, preventing other areas from becoming fragmented (which degrades performance).
- Tools: for add-on packages from Oracle or third-party vendors.
- Users: for tables created by users. We'll put most of our stuff here, unless we start creating separate tablespaces for each user

1.2 Users

A user is a little like a computer account. It has privileges and owns stuff. In a database, a user owns tables, views and the like.

- SYS is the most powerful user in Oracle, owner of all the internal objects. The default password is "change_on_install." See below for how to change the password. For security from outsiders, I have changed this password, but I will tell you what this new password is in class.
- SYSTEM is the account that the DBA usually uses. Default password is "manager," but we've also changed that.

Others are listed in table `sys.dba_users`.

1.3 Roles

A role is a collection of permissions. For example, a payroll clerk should have different permissions (access to different objects) than a clerk in the purchasing department. These roles would have to be defined. Some built-in roles:

- connect
- dba

All the roles are listed in the table `sys.dba_roles`.

1.4 Tables

Everything in Oracle is kept in a table. Here are some:

- `all_tables`
- `all_indexes`
- `user_tables`
- `sys.dba_objects`
- `sys.dba_roles`
- `sys.dba_role_privs`
- `sys.dba_sys_privs`

2 Administration

One useful thing to know is that Oracle is not case-sensitive, so you can ignore capitalization in all of the following examples. However, it converts input that is not within quote marks into upper case, so if you're looking in a table for an object (such as a table name), the table name is probably in all caps.

2.1 Creating Users

Often the DBA creates users, but we've decided to create a special user that has that priviledge. It's always safer to restrict powers as much as possible, so that you can limit the possible damage caused by a mistake or even malice. (This is one of the problems with Unix security as opposed to Oracle security—Unix security is all or none. The root user can do *anything*, yet often too many people need the root password.) We have created an account called "creator," with the same word as the password.

To create a user for yourself, run `sqlplus` and connect as `creator/creator`. Then execute the following statements, making the appropriate substitutions for the italicized items.

```
Create User PAYROLL
Identified By PAYME
Default   Tablespace USERS
Temporary Tablespace TEMP
Quota 10M on USERS
Profile DEFAULT;

GRANT CONNECT TO PAYROLL;
GRANT CREATE SESSION TO PAYROLL;
GRANT CREATE TABLE TO PAYROLL;
GRANT MANAGE TABLESPACE to PAYROLL;
GRANT CREATE VIEW TO PAYROLL;
GRANT SELECT ANY TABLE TO PAYROLL ;
```

The "identified by" part sets the password. Statements in SQL are terminated by a semi-colon (unless this command terminator character is changed; there's a command to do that). Then we need to grant particular privileges to the user. By default, users have no privileges.

You can change a password with

```
alter user PAYROLL
identified by FRED;
```

The CREATOR user has the power to drop users as well, so if you mess up, you can:

```
drop user PAYROLL;
```

and start over again. Or, you can use the word ALTER instead of CREATE to change properties of a user.

3 Connecting

It's possible to connect to an Oracle instance either via the command line or over the network. We're going to just use the command line.

3.1 Locally

If you're logged into Kefalonia, which is the CS department's machine that has Oracle installed, and which we will therefore call our "database server," you can connect to the database by running the program `sqlplus` or `svrmgrl`. The first is the program we'll use all of the time. The latter is for certain management and administration tasks.

(Note that if you are unable to run these programs because the shell says "command not found" or "permission denied," there is a problem with your account that you should bring to my attention as soon as possible.)

```
% sqlplus
```

3.2 Remotely

There is something called SQL*NET that allows connection to Oracle databases across a local area network (LAN). We name our database instance using a unique SID so that it can be distinguished from all the other instances on the LAN. Thus, if your computer console (Mac, PC, Linux, whatever) has a Oracle client installed on it, it can connect to the instance of the database that we have running on our server (Kefalonia).

4 Data Definition

This is basic SQL. We will try to avoid learning anything Oracle-specific. All of these forms should work by running the `sqlplus` program and connecting to the database.

4.1 Tables

If we have a schema such as:

emps(eid *number*, name *string*, hired *date*)

We can create a table as follows

```
create table emps(eid number,
                  name varchar(20),
                  hired date);
```

You can put the table creation code on one line if you want. In fact, that makes it easier in Emacs, since you can retrieve an old line of input if desired.

A useful command to know is to *describe* a table:

```
sql> describe table;
```

This command will tell you the names and types of all the columns, so you can then pose queries.

5 Data Manipulation

This is basically all the SQL queries we've learned. Try the following, which is based on a "test" account database that Oracle supplies with its installation (by coincidence, named "scott" with password "tiger"):

```
describe scott.emp;
describe scott.dept;
select * from scott.emp;
select * from scott.dept;
select ename,hiredate
  from scott.emp E, scott.dept D
  where E.deptno=D.deptno
  ;
```

As usual, we can put the last query on one line if we want.

We can also try inserting data. Let's insert only in our own table:

```
insert
into emps(number, name, hired)
values (24601, 'Jean Valjean', '14-jul-1886');
```

The "into" line is optional, but if it's there, the second line will follow the order of the first, so you don't have to remember the order of the columns. Otherwise, the insertion is positional. Note the syntax for numbers, strings, and dates.

6 Miscellaneous[1]

- You can save your queries in a `.sql` file and execute them from within `sqlplus` by saying `@filename.sql`. The file should be in Oracle's path, but the current directory seems to work.
- The `pagesize 10000` command sets the length of the "page" of a printout to the given number, and the column headings are only given at the top of a page, so 10000 effectively means they're only printed once. Or, you can set the pagesize to zero to suppress all headings.

  ```
  SQL> set pagesize 10000;
  ```

- Typing "ed" on a line in oracle will open your editor with your last query. When you close the editor, you type "/" to run the modified query. The "/" also re-runs the last command. Of course, it can be slow to start up Emacs, if that's your editor of choice, so you might run `sqlplus` under Emacs (in a shell) and edit queries in a file and just use the `@filename.sql` technique to re-run them. See section 7.
- If you put data in, and it is not showing up, you probably forgot to commit it. Just type `commit;`. This will be more of a problem when we start using Perl. Oracle automatically commits when you exit.
- If you make a change to a table that you want to undo, type `rollback;` which will undo until the last commit. If you've never commited anything, it will remove everything you've inserted, so use `commit` when you are at a place you're happy with. You can use `commit` as often as you like. You can't roll back DDL statements (creating/dropping tables and such).
- The `truncate` *`tablename`* command gets rid of all rows in a table.
- To copy a table, use the following

  ```
  create dest_tablename
  as select source_tablename;
  ```

- If you get an `ORA-###` error that you don't understand, you can type it into the little box at `technet.oracle.com`.

[1]Thanks to Rachel Lomasky for most of these tips

7 Running an Inferior Shell

This section really doesn't have anything to do with SQL, but discusses an Emacs feature that you might find useful. The feature is to have Emacs run a shell for you, such that input you type into Emacs is passed to the shell, and output from the shell is captured by Emacs into a buffer that you can edit, save to a file, or anything else you do with Emacs buffers. Any programs you run from the shell (such as `sqlplus`) share this I/O behavior, so the output from SQL can easily be captured to a buffer that you can manipulate with Emacs. Because the shell is thought of as running "below" Emacs, it's called an "inferior shell" or, more generally, an "inferior process"—not due to any loss in quality!

There are a few other conveniences that shell mode affords. Emacs keeps track of the directory you're in, so "find file" and other such commands can default the correct way. Emacs also keeps track of previous inputs you've submitted to the inferior process and you can call them back (using M-p), optionally edit them, and re-submit them. This is very convenient if you're trying to get the syntax for something right and you need several tries. It's also useful if you're doing a series of commands that are variations on a theme, so it's easy to call the previous one back, make the necessary changes, and re-submit.

You start a shell (or return to an existing one) by the command M-x shell RET. The buffer will be named `*shell*`. You can call back a previous command with M-p. You re-submit it by just pressing RET, as usual. Because you're now giving your input to Emacs, which will execute some things and pass others on to the shell, a few commands change. To kill or interrupt the process, type C-c C-c, instead of just one C-c. When you're done with the inferior shell, you can exit it with the "exit" command (to the shell). Emacs will warn you if you try to exit Emacs when you still have an inferior process. These commands should be all you need for most purposes; they're most of what I use. You can find out more by typing C-h m when you're in shell mode—that's mode-specific help.

One unfortunate side-effect is that Emacs records the input you give, but the shell usually "echoes" it as well, so you'll often see your input twice. I usually just ignore this, but there is a way to turn off echoing in the shell. Consult the man page for `stty`. Also, sometimes the shell ends lines with carriage-return and linefeed, and the carriage return shows up in Emacs as a $\hat{M}$. Again, I just ignore it, especially since it'll be invisible in printouts, and I haven't found a way to turn it off.

8 Exercises

1. Login to Kefalonia
2. Connect to Oracle as DBA using `sqlmrgl`.
3. Create a User for yourself
4. Disconnect
5. Reconnect as that user, using `sqlplus`.
6. Create a table
7. Describe it
8. Insert some data into it
9. Query it
10. Drop it
11. View the "scott" tables (emp, dept)
12. View some of the built-in tables

9 Resources

- There are web pages of documentation at `file:/home/oracle/OraHome1/doc/unixdoc/index.htm`
- There are books on reserve in the science library
- There is technical information at `www.oracle.com`
- Info at Yahoo at `http://dir.yahoo.com/Computers_and_Internet/Programming_Languages/SQL/`
- There is lots of other stuff online, so web searches should be effective