Top Tip: SQL*Plus (again) – saving your environment

By Tim Onions TOdC Limited

The summer floods seem to have washed away contributions for this issue's Top Tips. This is a shame as the quality of previous tips has been extremely high and the range of topics covered a very welcome breath of fresh air – I'm sure my personal DBA and developer ramblings only hit home to a relatively small percentage of Oracle Scene users. I know you are out there and I know you are full of great ideas suitable for presenting here. I sit here alone by the phone (ok, alone by the computer) waiting for your call (i.e. email)!

So as it is just little ole me to do the tipping again I'm back on the subject of SQL*Plus once more. I find it really is the development environment of choice for me (being as old as I am). I have the fashionable GUIs installed but find them slow in comparison to my firm favourite when used in conjunction with a "developer's" editor (heck, even Notepad is good enough for most jobs). Throw in a few scripts I now consider old friends and most jobs can be tackled effortlessly.

By judicious application of setting up your environment via the LOGIN.SQL script you can tailor the basics to function in a way that works for you – my particular favourite is changing the prompt to show the username and database instance currently connected to. However, these settings may need changing to suit a particular query or report being developed/run – things line PAGESIZE, LINESIZE, FEEDBACK and my particular bug-bear DEFINE. This latter one used to get me time and time again (not true – it STILL does get me far too often!). DEFINE is the setting which tells SQL*Plus the special character to use for substitution variables and is the ampersand character, &, by default. However, sometimes the code and or data you are working on contains the ampersand and it is too much trouble to use escape codes and such like. One place where you are sure to stumble across this is compiling PL/SQL packages – everyone at some time on some project has used the ampersand in a well intentioned comment (go on admit it). When this is the case then your daydreams are rudely interrupted with SQL*Plus asking you to enter a value for some obscure term. So after the first two or three of these nightmares you learn to turn DEFINE off or use some other character as an alternative (# for instance – SET DEF #). Your script/package compilation now works fine but do you remember to reset DEFINE (SET DEF ON) back? Not me! Like as not even if you write a script and tweak the environment some time or other (more often than not I'd wager) you will forget to put the settings back at the end of said script to how they were, we're talking more than just the setting of DEFINE here.

Hey - but hang on a moment, you do not actually know what the original settings were, you can take a stab and may get lucky (not all settings are going to be as easy to reverse as DEFINE). Even so you may change your set-up at some later date and the script, when run, now changes the settings back to how they were (or you thought they were) when the script was first developed.

Well, you don't have actually have to worry about this as SQL*Plus can be made to store the settings in a file of your choosing and then the file can be "run" to re-instate the settings back to their original values (i.e. how they were just before you or your script had them changed).

The command to save the SQL*Plus settings is simplicity itself (in the example below we stored into a file called mysqlplussettings.tmp in the working directory but any legal file name can be used and a directory path can be added too if desired):

```
SQL> store set mysqlplussettings.tmp replace
```

and to restore the settings when needed just run the file the settings were previously stored to (and also for housekeeping reasons delete the temporary file too – I've givenUNIX flavour here but on Windows just replace *rm* with *del*) via:

```
SQL> @mysqlplussettings.tmp
SQL> !rm mysqlplussettings.tmp
```

Everything that used a SET command is saved and subsequently restored.

Unfortunately, COLUMN commands are not saved only SET commands (open the file up and you will see a line for each possible SET command). You can sort of store COLUMN settings by spooling the results from a COLUMN command. However, the output although containing all the information needed to recreate COLUMNS is split over multiple lines per statement and hence is not directly executable. All the file needs is a few line continuations added (that is a dash, -, added at the end of each line that is continued) but that is a manual task and a bit of a bind. Anybody out there have a tip to simply and easily help me preserve my COLUMN commands?

About the Author

Tim Onions is an independent database consultant with over 15 years' experience with Oracle databases. Tim specialises in the application and database design of high performance systems, as well as tuning and optimisation techniques and can be reached at Tim.Onions@TOdC.co.uk