Top Tip: Formatting code, copy do you read me?

By Tim Onions TOdC Limited

In compiling this section of OracleScene the intent is always to try and produce reasonably fresh content and not simply regurgitate knowledge gleaned from one or more of the plethora of excellent resources available these days. It is acceptable to write up some work or something that has been thoroughly researched as a result of some spark of half an inkling of an idea, but not to outrageously plagiarise from a newsgroup or the wonderful askTom web site (oooh – a free extra Top Tip for anybody, if there is anybody in the Oracle world still unaware of it, to check out Tom Kyte's site http://askTom.Oracle.com). So I rather awkwardly start this issue's article with something I discovered from one of those e-newsletters certain organizations like to encourage us all to subscribe to. My only defense (m'lud) is that the newsletter in question was not an Oracle specific one (indeed it was for a rival database product) and so a good proportion of the readers here will quite likely not have seen it.

What I want to commend to you is a little Java application called SQLinForm. This does just one task but does it very well – takes SQL code and formats it into "standardized" code. To put this into blunt plain English used in the part of the world I come from – it makes the *bloomin'* stuff readable. For a while now I've tried to use the SQL formatters available within the development environments such as TOAD and SQLDeveloper but have found they rarely produce what I want (and in one case does not produce anything for me!). It is also tiresome to have to load these, quite large applications, should I just want to format a few dozen lines of code.

SQLinForm allows you to paste in the SQL and then format it – you can set a few rules as to what you want the result to look like, things like where the line breaks should go, where to put whitespace, where tab stops are, what to have in uppercase/what to have in lower case and how to align things – but basically it's a point and clickety-click application: paste in the SQL; click the format button; take the results and paste them back into you code. I have found it particularly useful for taking SQL from things like AWR reports or direct from V$SQL where all formatting from the original code has been stripped away and making it easily readable and understandable. It could equally be used during development to ensure a standard in-house style of SQL code writing is adhered to. SQLinForm can be found at http://www.sqlinform.com.

Moving on …. SQL*Plus has had a nugget of a command in it, the COPY command, for many years - yet it is surprising how many developers and DBAs do not know about it. This command uses 2 concurrent SQL*net connections to copy data from a source to a destination database (and hence no DB links are required). It does this by utilising array fetches and array inserts of data - the array size can be tuned and can result in some stunningly quick results. The data is taken from the source into SQL*Plus and then inserted into the destination by SQL*Plus - which implies that the data passes through whatever machine the SQL*Plus session is running on. For this reason you should really

only ever use COPY from within a SQL*Plus session that is running directly on the source or destination database server - if you run it from some other machine (your PC workstation for example) all of the data must be shipped via that machine. In which case the additional network traffic will have a serious adverse affect on the rate at which data is loaded.

There are a number of options that can be used - for instance, like SQL*Loader you can tell COPY whether to create a new destination table; replace or insert or append data to an existing table etc. The command can also deal with the LONG datatype (although sadly not CLOBS or BLOBS).

The most interesting options however are arraysize and copycommit. Arraysize is how many rows of data to fetch at a time - the bigger you make this then in theory the faster the command should run. Copycommit defines how often to commit the data copied - it defines how many array fetches will be committed as a batch. Tinkering with these parameters can be a great help when large volumes of data need to be inserted yet UNDO space is limited.

There is plenty of great stuff on COPY already out there on the web (enter "sql*plus copy command" into your favourite search engine) but most forget to show how to use it in its simplest form - in this form you copy data from a remote source database to the currently connected database (the destination) where the source and destination table structures are identical. This is great for migrating data between databases - even if they are of different versions or releases (remember that all that COPY needs is a connection to a source and a destination database, it makes no demands on the versions of either). Below is how this is done, a simple one line command can copy millions of rows of data in a matter of minutes! Note that COPY is not SQL so does not need a slash or semi-colon to run the command. NB The dash character in SQL*Plus tells it to continue reading the next line as part and parcel of the same command.

```
SQL> set arraysize 1024
SQL> set copycommit 1024
SQL> COPY FROM scott/tiger@remotedb.world -
INSERT bigtable -
USING SELECT /*+ FULL(t) PARALLEL(t,4) */ * FROM bigtable t

Array fetch/bind size is 1024. (arraysize is 1024)
Will commit after every 1024 array binds. (copycommit is 1024)
Maximum long size is 5000. (long is 5000)
   14732163 rows selected from scott@remotedb.world.
   14732163 rows inserted into bigtable.
   14732163 rows committed into bigtable at DEFAULT HOST connection.
```

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

taken to ensure the examples given function properly and are totally unobtrusive and benign (when used properly), neither the author nor TOdC Limited can take any responsibility or liability for what effect they have when you use them.