Free OEBS SQL
Think Quality! Think Global!

By

G.Piper

www.PIPER-Rx.com

July 2013

I am amazed how often I see the very same piece of code used by many different bloggers, often with very little thought as to the quality of the code or the audience it's going to. We all want to get the most out of free code and avoid the pitfalls, so here's some things to consider for bloggers and users alike.

Think global!

OEBS is a global product and so often I see code written that will not work with multiple language sets (typically the SQL will return one row for each language installed) so remember even if you have only one language installed, write the code as if you had several languages installed by making sure you complete the language restriction language = userenv('Lang') when using tables with multiple language translations.

```
SELECT fcp.concurrent_program_id ID,
       fcp.concurrent_program_name NAME,
       fcpt.user_concurrent_program_name FULL_NAME,
       to_char(fcp.last_update_date, 'DD-Mon-YY HH24:MI') LAST_UPDATE
  FROM applsys.fnd_concurrent_programs fcp,
       applsys.fnd_concurrent_programs_tl fcpt
 WHERE fcp.application_id = fcpt.application_id
   and fcp.concurrent_program_id = fcpt.concurrent_program_id
   and fcpt.language = userenv('Lang')
   and fcp.application_id = 0
 ORDER by fcp.concurrent_program_id desc;
```

Think Quality!

Often the free code you find in blogs is just copied or very little information is given about its use and limitations. Sometimes even the basics are wrong; outer Joins missing and the like. Other times the issue is more about the limitations of the code E.g. The code says it does " X" but in fact does not do it completely or in some cases does more than you would expect.

Example 1:

As part of a set of a set of OEBS diagnostic scripts to trouble shoot concurrent managers a blog recommended you run the following piece of code to make sure your managers are "UP":

```
SELECT max_processes, running_processes
  FROM applsys.fnd_concurrent_queues
 WHERE concurrent_queue_name = 'STANDARD';
```

The diagnostic script then goes on to state that if the value for **max_processes** is equal the value for **running_processes** the manager are "UP". But alas, if you terminate a request the process running the terminated request is killed. The process will be replaced when the internal manager wakes up and restarts a new process which is generally every 20 minutes. So for a period of time, which could be up to 20 minutes, the number of running processes may be lower than the value for MAX_PROCESSES but the managers are still UP…

Also, what if the managers crashed, and did not have time to update the base table before they crashed. This piece of SQL will show them as "UP" when in fact they are DOWN.

Example 2:

I recently came across yet another script to list all pending requests:

```
SELECT FCR.REQUEST ID REQUEST ID
       ,FCPT.USER CONCURRENT PROGRAM NAME REQUEST_NAME
       ,FCR.ACTUAL START DATE START DATE
       ,DECODE(FCR.PHASE_CODE, 'C', 'Completed',
                               'I', 'Inactive',
                               'P', 'Pending',
                               'R', 'Running') PHASE
       ,DECODE(FCR.STATUS CODE, 'A', 'Waiting',   'B', 'Resuming',     'C', 'Normal',
                                'D', 'Cancelled' , 'E', 'Error',       'F', 'Scheduled',
                                'G', 'Warning',    'H', 'On Hold',     'I', 'Normal',
                                'M', 'No Manager', 'Q', 'Standby',     'R', 'Normal',
                                'S', 'Suspended',  'T', 'Terminating', 'U', 'Disabled',
                                'W', 'Paused',     'X', 'Terminated',  'Z', 'Waiting') STATUS
      ,FU.USER NAME REQUESTED BY
FROM  FND_CONCURRENT_PROGRAMS FCP,
      FND_CONCURRENT_PROGRAMS_TL FCPT,
      FND CONCURRENT_REQUESTS FCR,
      FND USER FU
WHERE  FCR.CONCURRENT PROGRAM ID = FCP.CONCURRENT PROGRAM_ID
AND    FCR.PROGRAM_APPLICATION_ID = FCP.APPLICATION_ID
AND    FCR.CONCURRENT_PROGRAM_ID = FCPT.CONCURRENT_PROGRAM_ID
AND    FCR.PROGRAM_APPLICATION_ID = FCPT.APPLICATION_ID
AND    FU.USER ID = FCR.REQUESTED BY
AND    FCPT.LANGUAGE = USERENV('Lang')
AND    FCR.PHASE CODE = 'P'
ORDER BY FCR.ACTUAL_START_DATE DESC;
```

Firstly, if you are listing only Pending requests as was the intent of this piece of code, why add all the phase and status codes not associated with pending requests? And would it not have been more meaningful for pending requests to use *fcr.requested_start_date* rather than *fcr.actual_start_date*.

Whilst the script does list all pending requests as was stated, it will have somewhat of an issue listing such pending statuses as On Hold, Scheduled. In this case the author appeared to assume that because there is a status code in the lookups object for on-hold requests (H) and scheduled requests (F) (Note: in some releases Scheduled requests have a status code of "P") this does not mean they are directly used by the application. The Oracle Sys Admin form that displays the full list of requests derives a number of the status codes prior to displaying the information.

In this example both On-hold and Scheduled requests will have a phase code of P (Pending) and a status code of I (Normal). But in reality, a request is defined as On-hold when the *fnd_concurrent_requests* attribute *hold_flag* is set to "Y" and a scheduled request is one that at a minimum has a future *requested_start_date*.

So it is not just a simple case of linking phase and status codes to the lookups object thus making any SQL that lists all pending request more complex. One "possibly" example is as follows.

```
SELECT fcr.request_id,
       fcpt.user_concurrent_program_name,
       to_char(fcr.requested_start_date, 'DD-Mon-YY HH24:MI')
requested_start_date,
       decode( (decode(fcr.hold_flag, 'Y', 'On Hold', 'X')),
                'X', (decode(sign(fcr.requested_start_date - sysdate),
                          1, 'Scheduled Request',
```



```
                                decode( fcr.status_code,
                                        'A','Waiting',
                                        'Q','Stand-By',
                                        'Pending Normal') )),
                'On Hold') status,
       fcr.resubmit_interval||'
'||initcap(fcr.resubmit_interval_unit_code)||
             decode(fcr.resubmit_interval_type_code, null,
                      decode(fcr.release_class_id,
                               null,
                               'Single Run',
                               'Periodic'),
                  ' ('||initcap(fcr.resubmit_interval_type_code)||')')
resubmit_details,
       decode (fcr.status_code, 'Q', 'ICM', null) ICM
  FROM applsys.fnd_concurrent_requests fcr,
       applsys.fnd_concurrent_programs_tl fcpt
 WHERE fcr.phase_code = 'P' -- Pending Only
   and fcr.program_application_id = fcpt.application_id
   and fcr.concurrent_program_id = fcpt.concurrent_program_id
   and fcpt.language = userenv('LANG')
 ORDER by fcr.request_id DESC;
```

Not perfect but covers most of the bases…

Example Output

REQUEST	USER_CONCURRENT_PROGRAM_NAME	REQUESTED_START	STATUS	RESUBMIT_DETAILS	ICM
2216799	email Center Standalone Workflow Worker	11-Jul-13 12:22	Scheduled	Request10 Minutes (End)	
2216798	OAM Applications Dashboard Collection	11-Jul-13 12:20	Scheduled	Request10 Minutes (End)	ICM
2216731	Workflow Control Queue Cleanup	11-Jul-13 18:31	Scheduled	Request12 Hours (Start)	
2216724	Workflow Background Process	12-Jul-13 06:00	Scheduled	Request1 Days (Start)	
2216700	Workflow Background Process	12-Jul-13 04:00	Scheduled	Request1 Days (Start)	
2216675	Program - Update Today's Average Rate	12-Jul-13 02:00	Scheduled	Request1 Days (Start)	
2216674	Program - Update Limit Utilizations	12-Jul-13 02:00	Scheduled	Request1 Days (Start)	
2215360	Program - Optimizer	06-Aug-13 11:44	Scheduled	Request Periodic	ICM
2160528	Synchronize WF LOCAL tables	19-May-12 21:45	On Hold	Single Run	ICM

Firstly, if a scheduled resubmit request has been placed On-hold the status column will show "On Hold" and the resubmit details will be shown. This piece of code **does not** list Pending Error (Inactive No Manager) but that's a whole other story.

The ICM column indicates that an incompatibility rule exists for this request and it will need to be resolved by the Conflict Resolution Manager prior to being run.

Example 3

The following piece of code from a blog stating that it will cancel **scheduled** concurrent request statistics:

```
update  applsys.fnd_concurrent_requests
 set  phase_code='C',status_code='D'
 WHERE  phase_code  =  'P'
 and  status_code  in  ('Q','I')
 and  concurrent_program_id=38121;
```

As we previously stated, Scheduled requests are defined as those requests with a future dated *requested_start_date*. So this piece of SQL will cancel "Gather Schema Statistics" requests that are either pending standby (Q) or pending normal (I) but will not be restricted to scheduled requests only.

Example 4

This same blog also suggests the code to put **all** concurrent jobs on hold is as follows:

```
Update  applsys.fnd_concurrent_requests
set  hold_flag='Y'
where  phase_code  in  ('R','P','I');
```

Setting the on hold flag for a request that is running will do very little and the phase codes of "I" (Inactive) are derived.

In summary, we always review code based on our own requirements (e.g. listing all pending requests), biases and understanding of OEBS. It is important for Apps DBAs looking for help via blogs to at least be provided with code that is well thought through and limitations identified. Just because it solved your issue does not mean it will easily solve the next person's issue.

So the take home message is…...

Blogs and code that go the extra mile are worth the effort,

and for new Apps DBAs

BEWARE; don't just cut and paste some one else's code, check it first.