Orphan Oracle Processes

By
By David Fitzjarrell, DBA, SPR Inc.

Corporate Headquarters
Embarcadero
Technologies
100 California Street
12th Floor
San Francisco, CA 94111
www.embarcadero.com

EMEA Headquarters
Embarcadero Technologies,
Ltd.
Thames House
17 Marlow Road
Maidenhead
Berkshire
SL6 7AA
United Kingdom
www.embarcadero.co.uk

Oracle keeps track of both database processes and user sessions; aside from the usual database startup processes each subsequent process should be attached, or associated, with a session. Two views report on session and process information for the given Oracle instance, v$process and v$session. Each record in v$session describes a current user session connected to the Oracle instance. Each record in v$process describes a process running against the current Oracle instance. These two views can be connected, or joined, in a query to generate a report of who is connected to the database and what process or processes each user is running.

The DBA will want to know the spid (the operating system process id) and terminal (the tty, or communication line, whence the connection originates) from v$process (to identify which OS session the Oracle session is connected to) and the sid, serial# , program, username and osuser from v$session (to identify the session in Oracle and provide enough information to the DBA in the event that the session should be terminated):

```
Select s.sid, s.serial#, s.username, s.osuser, p.spid, p.terminal, s.program
From v$session s, v$process p
Where s.paddr = p.addr;
```

Sessions with a null username are usually Oracle system processes; the SPID values should be closely grouped, signifying that these processes started when the database instance was started.

The sid and serial# values are necessary for the DBA to kill a session in Oracle:
Alter system kill session ‘sid, serial#’;

where the sid and serial# placeholders are filled with the actual sid and serial# values. This command, executed from the DBA user account, will terminate an Oracle session internally, which will, in turn, terminate the OS session tied to it.

The spid and terminal values, returned from v$process, identify the UNIX session that spawned, or created, the Oracle session identified by the sid and serial# values. A ps listing (under UNIX) can be searched for the spid value returned from the above query; one entry should be returned providing the UNIX process id (PID) and the UNIX parent process id (PPID) along with the process name and the process owner. The PID value in UNIX is the same as the spid value in v$process. An Oracle process is created, or spawned, by a user process on the UNIX system; a parent process id (PPID) of ‘1’ indicates that a daemon, or background system, process created the current Oracle process. This is usually the case with SQL*Net or Net8 connections. If the user is connected to the UNIX server by login then the parent process id will be the same as the process id of the shell, or command interpreter, running for the UNIX user.

This query is fine for runaway processes or "problem" queries, however it will not cover any process that is not linked to a session. To identify processes that no longer are attached to active or inactive sessions the following query should be used:

```
Select *
From v$process
Where addr not in (select paddr from v$session)
And spid is not null;
```

There could possibly be one or more valid processes not connected to a session; these would be multiple database writer processes, parallel query processes, multiple SMON processes to name a

few possibilities. Some of these processes will be identified, but they will all have SPID values very close to one another. Sample output is shown below:

```
Addr    Pid     Spid    Username        Serial# Terminal        Program
B Latchwait Latchspin
-------------------     --------  --------  ------------      ------------      ----------      ----------
0000000008058D50(S000)      9       757     oracle  1       ?       oracle@pwuis2
0000000008059090(D000)      10      759     oracle  1       ?       oracle@pwuis2
00000000080593D0(D001)      11      761     oracle  1       ?       oracle@pwuis2
0000000008059A50            13      763     oracle  1       ?       oracle@pwuis2
0000000008059D90            14      765     oracle  1       ?       oracle@pwuis2
```

These processes are legitimate Oracle processes that will exist for the instance queried due to parallel query and asynchronous writes being enabled. Had there been another process with an SPID value outside the given range this process would be suspect. Identifying "rogue" processes in Oracle is fairly simple, however these processes cannot be killed from within Oracle since they are not connected to a session. How, then, are they disposed of? A database shutdown is the only way to close these processes and clear out the database; a simple shutdown will not suffice, though, as a "garden-variety" shutdown will wait for all processes to complete before continuing. Since killing a rogue process is the goal of the shutdown a shutdown immediate, or, if all else fails, a shutdown abort would be the proper action, the shutdown immediate being the action of choice. [A shutdown immediate will terminate all sessions to the database and begin an orderly shutdown of the instance. In some cases a shutdown immediate will also terminate any "rogue" processes running against the instance; in some cases this will not be the case, and a shutdown abort will be necessary to terminate the process. It is after such a shutdown that an immediate startup is necessary; a shutdown abort CAN create havoc with an instance, possibly corrupting data and putting the instance into a state that is not usable. Only use a shutdown abort when it is absolutely necessary; a shutdown immediate may take anywhere from 15 minutes to an hour to complete. If the shutdown takes longer than an hour it is likely that the shutdown is waiting for the rogue process to complete. At this point it will be necessary to connect to the instance with another server manager session and execute a shutdown abort. From this second session the instance should be re-started.] Once the database shutdown has completed the instance should be re-started to ensure that the instance will, indeed, start, mount and open and to ensure that the database data is intact and free from corruption that could occur when processes are stopped dead in their tracks. Once the instance is again "live" the users may connect and resume their work.

David’s DBA projects have spanned from work for American Airlines and the U.S. Postal Services. He is with SPR Consulting (http://www.sprinc.com), a provider of project management services to organizations with large and complex IT infrastructures, and is currently working on a Y2K implementation project for the city of Tulsa, OK. He can be reached at oratune@aol.com.