|  |
| --- |
| **Write fast and efficient PLSQL - DBMS\_PROFILER** |
| Posted by [Zahid](http://www.oracleflash.com/about.html) on October 11, 2011. |

Before DBMS\_PROFILER was introduced tuning long PL/SQL programs was a great pain. Tuning PL/SQL was done by SQL TRACE and TKPROF. Code of thousands of lines was monitored via DBMS\_UTILITY.GET\_TIME to measure time consumed by individual statements.

With the presence of DBMS\_PROFILER this has become quite easy to monitor the performance of the PL/SQL programs and identifying the culprit statements consuming too much time while the run of the program.

In this article we have two procedures. One of them uses literals in the queries and the other uses bind variables. We noticed that the program using literals taking too much time. We will then use DBMS\_PROFILER to pin point the statement consuming most of the time.

$ cd $ORACLE\_HOME/rdbms/admin

$ sqlplus / as sysdba

/\*

Run profload.sql which will install the DBMS\_PROFILER package.

\*/

**SQL> @profload.sql**

Package created.

Grant succeeded.

Synonym created.

Library created.

Package body created.

Testing for correct installation

SYS.DBMS\_PROFILER successfully loaded.

PL/SQL procedure successfully completed.

/\*

Now create the user for a test of DBMS\_PROFILER.

\*/

SQL> create user plsql\_prof\_test identified by test;

User created.

SQL> grant connect , resource to plsql\_prof\_test;

Grant succeeded.

SQL> conn plsql\_prof\_test/test

Connected.

/\*

Once connected run proftab.sql which will create special tables where

profiler puts its results.

\*/

**SQL> @proftab.sql**

drop table plsql\_profiler\_data cascade constraints

\*

ERROR at line 1:

ORA-00942: table or view does not exist

drop table plsql\_profiler\_units cascade constraints

\*

ERROR at line 1:

ORA-00942: table or view does not exist

drop table plsql\_profiler\_runs cascade constraints

\*

ERROR at line 1:

ORA-00942: table or view does not exist

drop sequence plsql\_profiler\_runnumber

\*

ERROR at line 1:

ORA-02289: sequence does not exist

Table created.

Comment created.

Table created.

Comment created.

Table created.

Comment created.

Sequence created.

**DBMS\_PROFILER**

setup is complete now. Create table that we will use in our PL/SQL code that we will investigate through DBMS\_PROFILER package.

SQL> create table t1 (col1 varchar2(30), col2 varchar2(30));

Table created.

Create two different procedures one using literals in the queries and other using bind variables.

SQL> create or replace procedure literals

2 is

3 vNumber number;

4 begin

5 for i in 1..100000 loop

6 vNumber := dbms\_random.random;

7 execute immediate

8 'insert into t1 values ('||vNumber||','||vNumber||')';

9 end loop;

10 end;

11 /

Procedure created.

SQL> create or replace procedure binds

2 is

3 vNumber number;

4 begin

5 for i in 1..100000 loop

6 vNumber := dbms\_random.random;

7 insert into t1 values (vNumber,vNumber);

8 end loop;

9 end;

10 /

Procedure created.

**dbms\_profiler.start\_profiler**

starts the profiler before the execution of the PL/SQL program.

**dbms\_profiler.stop\_profiler**

stops the profiler once the program finishes executing.  
First run the procedure with the queries that contain literals.

SQL> execute dbms\_profiler.start\_profiler('literals');

PL/SQL procedure successfully completed.

SQL> exec literals;

PL/SQL procedure successfully completed.

SQL> execute dbms\_profiler.stop\_profiler;

PL/SQL procedure successfully completed.

The literals run is finished now run the program which contains queries with bind variables.

SQL> execute dbms\_profiler.start\_profiler('binds');

PL/SQL procedure successfully completed.

SQL> exec binds;

PL/SQL procedure successfully completed.

SQL> execute dbms\_profiler.stop\_profiler;

PL/SQL procedure successfully completed.

As binds run is complete now we can see the result of these runs captured by DBMS\_PROFILER.

SQL> set lines 10000

SQL> column run\_owner format a30

SQL> column run\_comment format a10

SQL> select runid,

2 run\_owner,

3 run\_date,

4 run\_total\_time/1000000000 run\_total\_time,

5 run\_comment

6 from plsql\_profiler\_runs;

RUNID RUN\_OWNER RUN\_DATE RUN\_TOTAL\_TIME RUN\_COMMEN

---------- ------------------- --------- -------------- ----------

1 PLSQL\_PROF\_TEST 21-MAY-10 91.0 literals

2 PLSQL\_PROF\_TEST 21-MAY-10 17.1 binds

/\*

plsql\_profiler\_runs has summary of all dbms\_profiler runs.

\*/

SQL> column text format a55

SQL> column total\_time format 99.9

SQL> column min\_time format 99.9

SQL> column max\_time format 99.9

SQL> select s.text ,

2 p.total\_occur ,

3 p.total\_time/1000000000 total\_time,

4 p.min\_time/1000000000 min\_time,

5 p.max\_time/1000000000 max\_time

6 from plsql\_profiler\_data p, user\_source s, plsql\_profiler\_runs r

7 where p.line# = s.line

8 and p.runid = r.runid

9 and r.run\_comment = 'literals'

10\* and s.name ='LITERALS'

SQL> /

TEXT TOTAL\_OCCUR TOTAL\_TIME MIN\_TIME MAX\_TIME

------------------------------ ----------- ---------- -------- --------

procedure literals 1 .0 .0 .0

procedure literals 3 .0 .0 .0

procedure literals 0 .0 .0 .0

for i in 1..100000 loop 100001 .1 .0 .0

vNumber := dbms\_random.random; 100000 .3 .0 .0

execute immediate 100000 71.2 .0 .5

'insert into t1 values ('||vN

umber||','||vNumber||')';

end; 1 .0 .0 .0

procedure literals 2 .0 .0 .0

8 rows selected.

SQL> select s.text ,

2 p.total\_occur ,

3 p.total\_time/1000000000 total\_time,

4 p.min\_time/1000000000 min\_time,

5 p.max\_time/1000000000 max\_time

6 from plsql\_profiler\_data p, user\_source s, plsql\_profiler\_runs r

7 where p.line# = s.line

8 and p.runid = r.runid

9 and r.run\_comment = 'binds'

10\* and s.name ='BINDS'

SQL> /

TEXT TOTAL\_OCCUR TOTAL\_TIME MIN\_TIME MAX\_TIME

------------------------------ ----------- ---------- -------- --------

procedure binds 1 .0 .0 .0

procedure binds 3 .0 .0 .0

procedure binds 0 .0 .0 .0

for i in 1..100000 loop 100001 .0 .0 .0

vNumber := dbms\_random.random; 100000 .2 .0 .0

insert into t1 values (vNumber 100000 6.9 .0 .4

,vNumber);

end; 1 .0 .0 .0

procedure binds 2 .0 .0 .0

8 rows selected.

SQL>

As it can clearly be seen that all the time was being consumed by the INSERT statement that was using literals. Rest of the lines in the code are doing just fine.

The test that we ran is also an evidence of using bind variables is better then using literals in most cases. The statement that was using literals consumed 71 seconds and the statement using bind variables consumed only 7 seconds. And this difference will be even more when multiple sessions are running this program because they will then cause contention on shared pool latches as they all will be parsing same statement again and again.