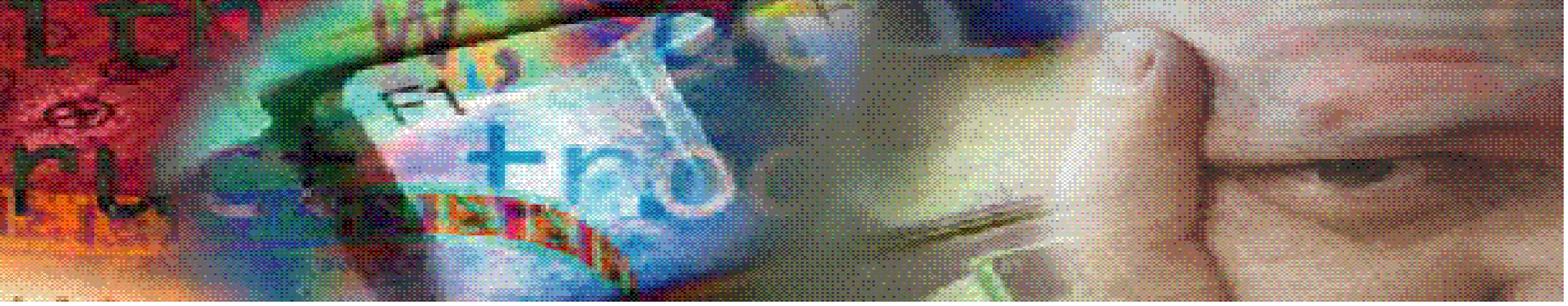


Using the Oracle PL/SQL Profiler

By Dan Hotka

Oracle has provided the ability to see how much time each step of a PL/SQL routine takes since Oracle8i. The environment is easy to setup and the information is easy to retrieve.

Introduction

Why Profile? When tuning SQL, it is easy because there is just the single SQL statement. With PL/SQL, there are SQL statements, SQL imbedded in implicit and explicit cursors, called routines (functions/procedures), and the PL/SQL code itself. When a PL/SQL routine is taking 5 minutes to run, exactly **what code** is taking how **much time**. This is the information that the PL/SQL profiler provides. Without this information, the person trying to tune the PL/SQL is only shooting in the dark, perhaps pulling out and tuning the SQL within the code, but otherwise has no idea what a PL/SQL routine is doing when it comes to time spent on each line of code.

The profiler is easy to setup and easy to use. Tools like Quest Software's TOAD provides a nice GUI interface to this profiler.

Installation

The profiler has two scripts that setup the environment. Both are found in the <ORACLE_HOME>/rdbms/admin folder.

The PROFLOAD.sql script needs to be run as SYS (connect <connect string> AS SYSDBA). This script will create the DBMS_PROFILER package and create synonyms and permissions for usage.

The PROFTAB.sql script is recommended to be executed for each user desiring to run the DBMS_PROFILER package. This script sets up the three main tables: PLSQL_PROFILER_RUNS, PLSQL_PROFILER_UNITS, and the PLSQL_PROFILER_DATA. This script can be setup so that all the users share these tables but this topic is beyond the scope of this paper.

Understanding the Profiler Process

Profiling is initiated with the START_PROFILER program. Start this process then execute the PL/SQL routine to be profiled. When execution is done, run the

STOP_PROFILER program. This will stop the profiling process and write the collected information to the profiler tables.

There are two more routines that control the collection of profiler data: PAUSE_PROFILER and RESUME_PROFILER. These routines might be useful if only certain statistics are of interest from a rather large PL/SQL program, or perhaps called subroutines are not desired to be profiled. These routines are typically imbedded in the PL/SQL code.

The FLUSH_DATA routine can also be called to periodically write the collected information to the profiler tables. This might be useful if the STOP_PROFILER routine is taking an excessive amount of time when profiling larger PL/SQL routines. This routine is typically embedded in the PL/SQL code.

When the START_PROFILER routine is started, the Oracle RDBMS collects a variety of information about the PL/SQL routine while it is being executed. The STOP_PROFILER then stops this collection process and writes the collected information to the three profiler tables.

These tables are then examined using SQL to view the results of the profiler collection.

Using the PL/SQL Profiler

There will be additional profiler information collected if the object being profiled has been compiled using the debug option.

This example will use this simple LOOPING_EXAMPLE code:

```
CREATE OR REPLACE PROCEDURE looping_example
IS
   loop_counter   NUMBER := 0;
BEGIN
   FOR rec IN (SELECT *
                 FROM emp)
   LOOP
      loop_counter := loop_counter + 1;
      DBMS_OUTPUT.put_line (
         'Record ' || loop_counter || ' is Employee ' || rec.ename
      );
   END LOOP;

   DBMS_OUTPUT.put_line ('Procedure Looping Example is done');
END;
```

To capture PL/SQL profile information, execute the following statements. The comment submitted with the START command will be populated into the RUN_COMMENT in the PLSQL_PROFILER_RUNS table, see below.

```
SQL> execute DBMS_PROFILER.START_PROFILER(‘User0 Looping_Example’);
SQL>
```

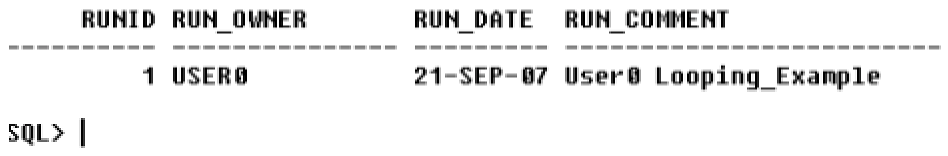
SQL> execute LOOPING_EXAMPLE:
SQL>
SQL> execute DBMS_PROFILER.STOP_PROFILER;

This example code and the PROFILER_RPT.sql SQL*Plus script (runs all 3 SQL statements in an interactive script) are available from www.DanHotka.com (downloads). You can also me for the PROFILER_RPT.sql script.

The profiler populates three tables with related information. PLSQL_PROFILER_RUNS has information about each time the profiler is started, including the comment entered when the profiler session was initiated. The PLSQL_PROFILE_UNITS contains information about the PL/SQL code executed during the run. Each procedure, function, and package will have its own line in this table. The PLSQL_PROFILE_DATA contains the executed lines of code, code execution time, and more. The following SQL is useful in extracting the profiler information.

First, find the profiler run of interest. The RUN_COMMENT column has the

```
select runid, run_owner, run_date, run_comment
from plsql_profiler_runs;
```

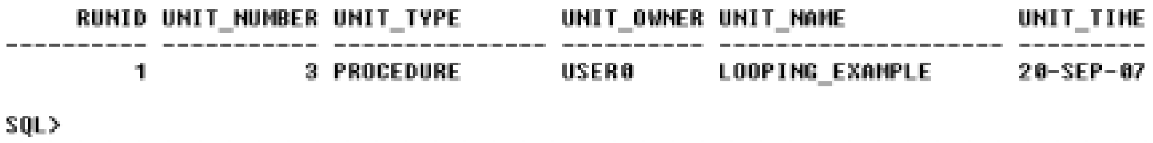


```
     RUNID RUN_OWNER       RUN_DATE  RUN_COMMENT
---------- --------------- --------- ------------------------
         1 USER0           21-SEP-07 User0 Looping_Example

SQL> |
```

In this SQL, enter the RUNID from the prior SQL statement. Oracle will place several lines of '<anonymous>' in the UNIT_OWNER column. This information is the overhead that Oracle incurred executing the code, not the code itself. Since I am not interested in this clutter, I coded the SQL to just show me the profiler information of interest to me.

```
select runid, unit_number, unit_type, unit_owner, unit_name,
unit_timestamp
from plsql_profiler_units
where unit_owner <> '<anonymous>'
and runid = &rpt_runid;
```

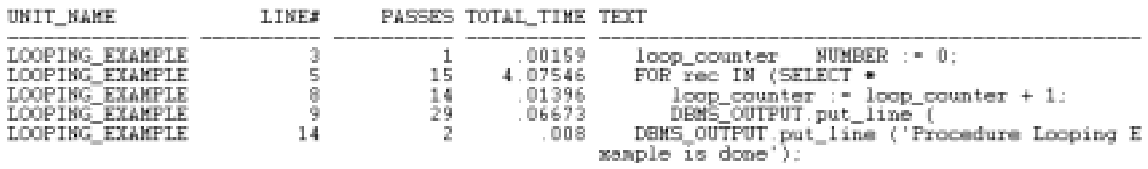


```
     RUNID UNIT_NUMBER UNIT_TYPE       UNIT_OWNER UNIT_NAME            UNIT_TIME
---------- ----------- --------------- ---------- -------------------- ---------
         1           3 PROCEDURE       USER0      LOOPING_EXAMPLE      20-SEP-07

SQL>
```

```
select pu.unit_name, pd.line#, pd.total_occur passes,
round(pd.total_time / 1000000000,5) total_time, us.text text
from plsql_profiler_data pd, plsql_profiler_units pu, user_source us
where pd.runid = &rpt_runid
```

```
and pd.unit_number = &rpt_unitid
and pd.runid = pu.runid
and pd.unit_number = pu.unit_number
and us.name = pu.unit_name
and us.line = pd.line#
and us.type in ('PACKAGE BODY','PROCEDURE','FUNCTION');
```



```
UNIT_NAME             LINE#     PASSES TOTAL_TIME TEXT
--------------- ---------- ---------- ---------- --------------------------------------------
LOOPING_EXAMPLE          3          1     .00159    loop_counter   NUMBER := 0;
LOOPING_EXAMPLE          5         15    4.07546    FOR rec IN (SELECT *
LOOPING_EXAMPLE          8         14     .01396       loop_counter := loop_counter + 1;
LOOPING_EXAMPLE          9         29     .06673       DBMS_OUTPUT.put_line (
LOOPING_EXAMPLE         14          2       .008    DBMS_OUTPUT.put_line ('Procedure Looping E
                                                 xample is done');
```

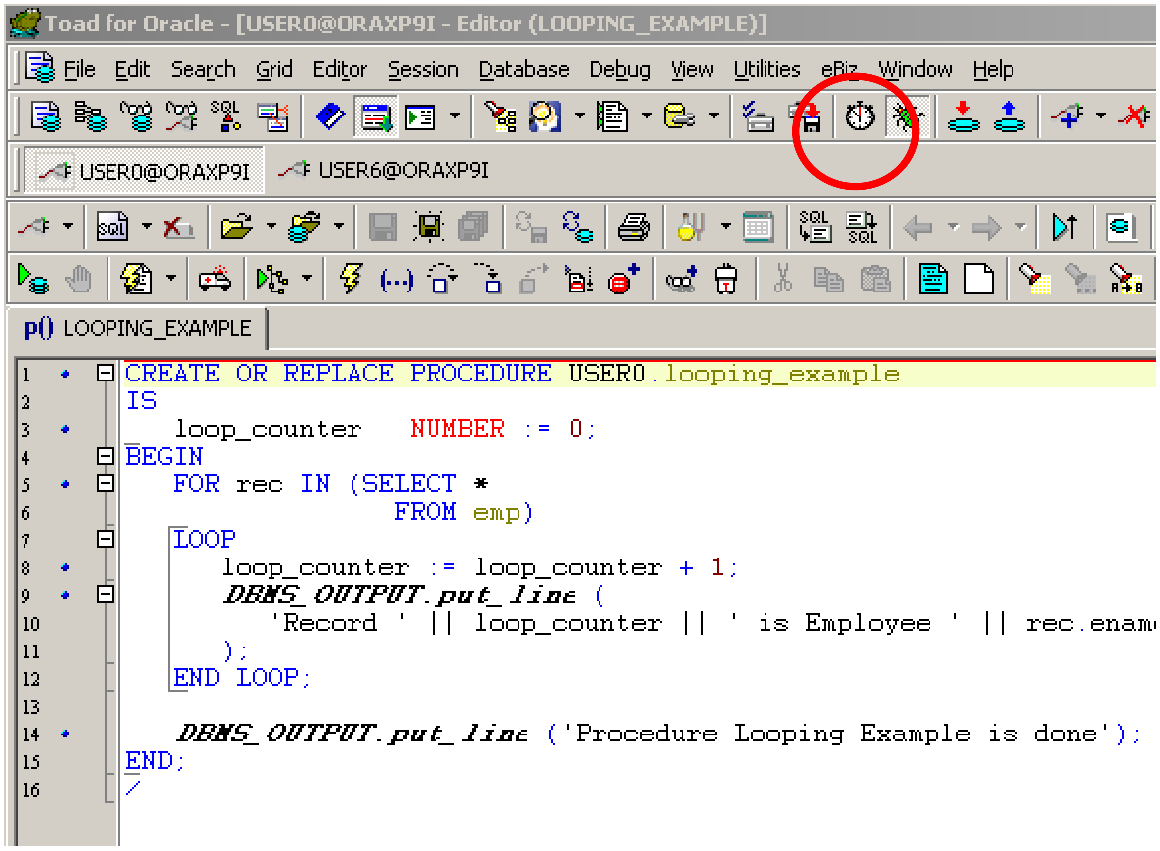
This code cleans up the profiler tables.

```
delete from plsql_profiler_data;
delete from plsql_profiler_units;
delete from plsql_profiler_runs;
```

TOAD Users

Quest Software has implemented the PL/SQL Profiler into the TOAD tool. This option has been available for quite a while. It too is easy to use and the whole process is easily handled from the TOAD environment.

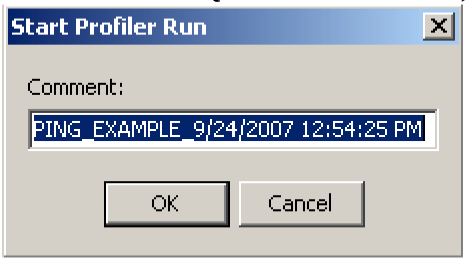
Starting and stopping the profiler is easily accomplished by clicking on the Toggle PL/SQL Profiler button ⏱ . If this button is grayed out, then the TOAD Server Side objects need to be executed (Database → Administrate → TOAD Server Side Objects Wizard).

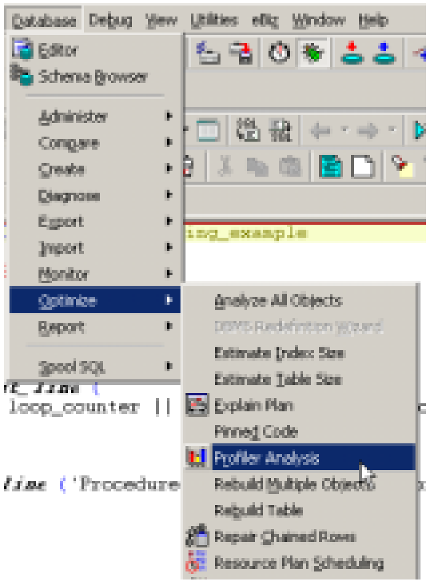

Toad for Oracle - [USER0@ORAXP9I - Editor (LOOPING_EXAMPLE)]
File Edit Search Grid Editor Session Database Debug View Utilities eBiz Window Help
USER0@ORAXP9I
USER6@ORAXP9I
p() LOOPING_EXAMPLE

```
CREATE OR REPLACE PROCEDURE USER0.looping_example
IS
    loop_counter   NUMBER := 0;
BEGIN
    FOR rec IN (SELECT *
                  FROM emp)
    LOOP
        loop_counter := loop_counter + 1;
        DBMS_OUTPUT.put_line (
          'Record ' || loop_counter || ' is Employee ' || rec.enam
        );
    END LOOP;

    DBMS_OUTPUT.put_line ('Procedure Looping Example is done');
END;
/
```

When the PL/SQL code is executed, a dialog box will popup for the start comment.

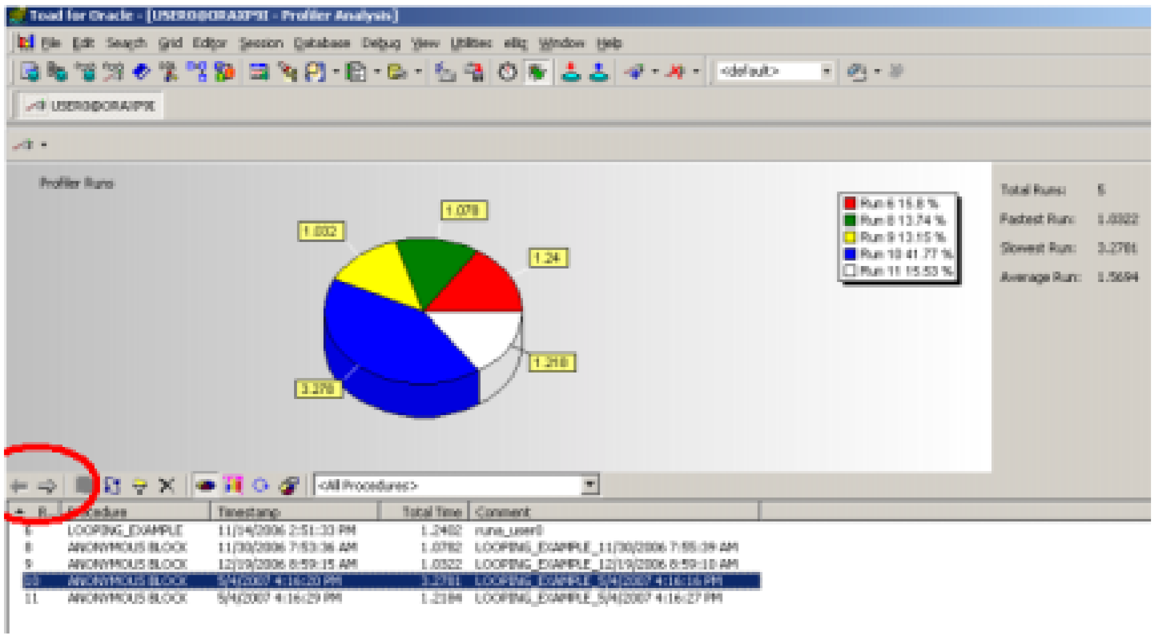




Click the Toggle PL/SQL Profiler button again to stop profiling.

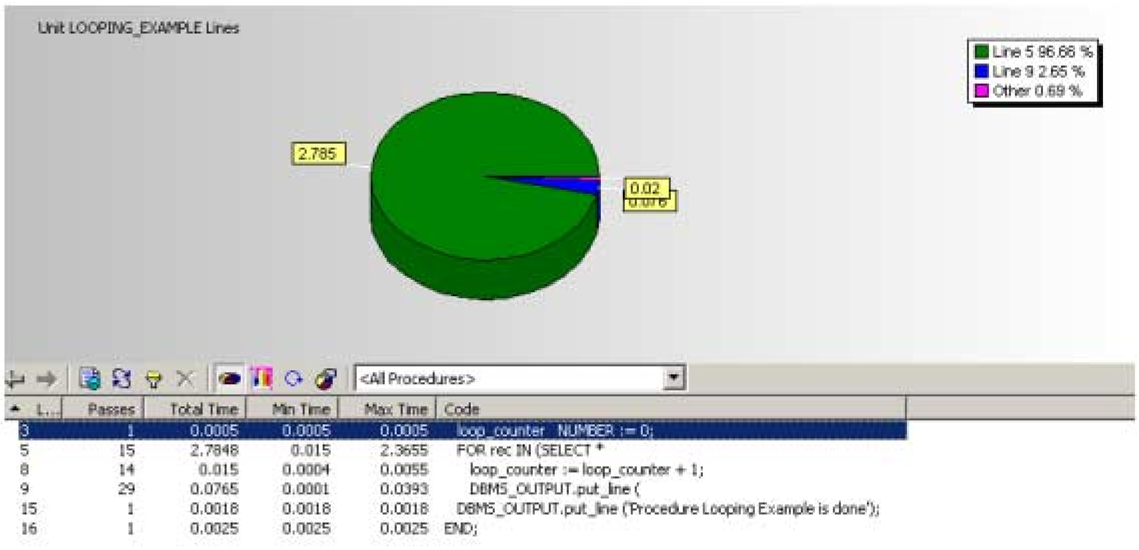
The same profile tables are populated but TOAD also formats this same outout using a nice interactive wizard. Use Database → Optimize → Profiler Analysis menu item to access the Profiler Analysis.

The output is easily visible. Select your executed code from the list and click on the arrow button in the circle.



Total Runs:	5
Fastest Run:	1.0322
Slowest Run:	3.2781
Average Run:	1.5694

R..	Procedure	Timestamp	Total Time	Comment
6	LOOPING_EXAMPLE	11/14/2006 2:51:33 PM	1.2402	runs_user0
8	ANONYMOUS BLOCK	11/30/2006 7:53:36 AM	1.0782	LOOPING_EXAMPLE_11/30/2006 7:55:39 AM
9	ANONYMOUS BLOCK	12/19/2006 8:59:15 AM	1.0322	LOOPING_EXAMPLE_12/19/2006 8:59:10 AM
10	ANONYMOUS BLOCK	5/4/2007 4:16:20 PM	3.2781	LOOPING_EXAMPLE_5/4/2007 4:16:16 PM
11	ANONYMOUS BLOCK	5/4/2007 4:16:29 PM	1.2184	LOOPING_EXAMPLE_5/4/2007 4:16:27 PM



L...	Passes	Total Time	Min Time	Max Time	Code
3	1	0.0005	0.0005	0.0005	loop_counter NUMBER := 0;
5	15	2.7848	0.015	2.3655	FOR rec IN (SELECT *
8	14	0.015	0.0004	0.0055	loop_counter := loop_counter + 1;
9	29	0.0765	0.0001	0.0393	DBMS_OUTPUT.put_line (
15	1	0.0018	0.0018	0.0018	DBMS_OUTPUT.put_line ('Procedure Looping Example is done');
16	1	0.0025	0.0025	0.0025	END;

Summary

The PL/SQL Profiler is an essential tool when tuning PL/SQL and the SQL coded into these same routines. Without something like this profiler process, it is impossible to tell where the time is spend when tuning PL/SQL code.

Dan Hotka

Dan Hotka is a Training Specialist who has over 31 years in the computer industry and over 26 years of experience with Oracle products. He is an internationally recognized Oracle expert with Oracle experience dating back to the Oracle V4.0 days. Dan's latest book is the TOAD Handbook by Pearson. He is also the author of SQL Developer Handbook by Oracle Press, SQL Developer Handbook by Oracle Press, Oracle9i Development By Example, and Oracle8i from Scratch by Que and has co-authored 7 other popular books including the Database Oracle10g Linux Administration by Oracle Press. He is frequently published in Oracle trade journals, and regularly speaks at Oracle conferences and user groups around the world. Visit his website at www.DanHotka.com. Dan can be reached at dhotka@earthlink.net .

Dan Hotka - Author/Instructor/Expert
www.DanHotka.com
DHotka@Earthlink.net
515 279-3361