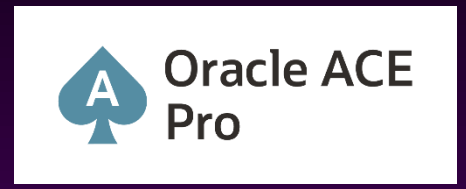
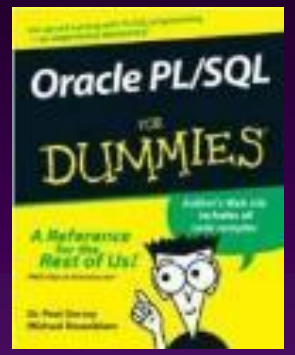


User-Defined Function Calls in SQL: One, Two…Many?

Michael Rosenblum
www.dulcian.com

Who Am I? – "Misha"

- Oracle ACE
- Co-author of 3 books
  - *PL/SQL for Dummies*
  - *Expert PL/SQL Practices*
  - *Oracle PL/SQL Performance Tuning Tips & Techniques*



- Known for:
  - SQL and PL/SQL tuning
  - Complex functionality
    - Code generators
    - Repository-based development

DULCIAN INC.

The Problem

- Question:
  - Do you know how many times your function is being executed within a SQL statement?
- Issue:
  - Few developers know how to properly ask this question.
  - … and even fewer know how to correctly interpret the answer!

Reminder!

- SQL statement execution order:
  - 1. JOIN
  - 2. WHERE
  - 3. GROUP BY
  - 4. SELECT (including analytics functions)
  - 5. HAVING
  - 6. ORDER BY

Approaches (1)

- Home-grown:
  - How: build your own set of APIs
    - Counter:
      - PL/SQL package variable – if you can get it within the same session
      - SYS_CONTEXT – to get data across multiple session
    - Procedure that increments that counter
  - Pro:
    - Easy to setup and manage
  - Contra:
    - Limited applicability and transportability
    - Requires code change

Approaches (2)

- Provided by Oracle
  - How: Oracle Hierarchical Profiler
  - Pro:
    - extremely detailed and deep
  - Contra:
    - Requires coordination with DBA/Security team
    - Generates A LOT of data
    - Starting 12.2 includes parts of SQL statements– may be a security concern

DULCIAN INC.

WARNING!

- Today's topic – COUNTING function calls
  - … NOT how to decrease that count, although you can check
    - Deterministic clause, scalar subqueries
    - Result cache
    - Rewriting SQL and PL/SQL code
  - … also NOT how decrease the cost, although you can check
    - Native compilation
    - PRAGMA UDF clause
    - SQL Transpiler (23ai)

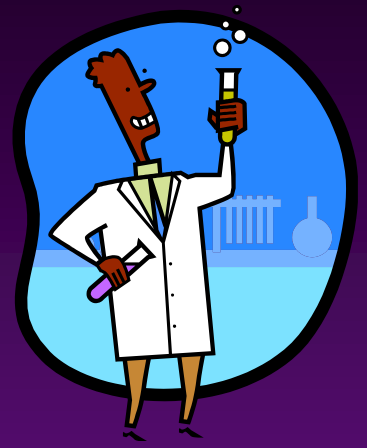
Counting Function Calls

Basic Testing Framework

```
create package counter_pkg is
    v_nr number:=0;
    procedure p_check;
end;

create package body counter_pkg is
  procedure p_check is
  begin
    dbms_output.put_line ('fired:'||counter_pkg.v_nr);
    counter_pkg.v_nr:=0;
  end;
end;

create function f_change_nr (i_nr number) return number is
begin
    counter_pkg.v_nr:=counter_pkg.v_nr+1;
    return i_nr*(-1);
end;
```



SELECT and WHERE

- Output:

```
SQL> SELECT empno, ename, f_change_nr(empno) change_nr
  2  FROM emp
  3  WHERE f_change_nr(empno) IS NOT NULL
  4  AND deptno = 20;
...
5 rows selected.
SQL> exec counter_pkg.p_check;
Fired:10
```

Twice the number of returned rows

- Explanation:
  - CBO orders predicates decreasing the total cost
    - DEPNO=20 is applied first to get 5 rows back
    - CBO needs correct info (statistics, indexes, constraints etc.) to make that decision
  - The same functions in SELECT and WHERE clauses are being fired independently.

SELECT and ORDER BY - 11g (1)

- Output:

```
SQL> SELECT empno, ename, f_change_nr(empno) change_nr_nr
  2  FROM emp WHERE deptno = 20
  3  ORDER BY 3;
5 rows selected.
SQL> exec counter_pkg.p_check;
Fired:5
SQL> SELECT empno, change_nr
  2  FROM (SELECT empno, ename, f_change_nr(empno) change_nr
  3        FROM emp WHERE deptno = 20
  4        ORDER BY 3);
5 rows selected.
SQL> exec counter_pkg.p_check;
Fired:10
```

In-line view causes double-firing

- Problem:
  - Why does the inline view cause a double fire?
  - Fixed in 19.7 (still was reoccurring in early versions of 12c)

SELECT and ORDER BY – 11g (2)

- Research of 10053 trace:

```
Order-by elimination (OBYE)
***************************
OBYE:     OBYE performed.
OBYE:     OBYE bypassed: no order by to eliminate.

Final query after transformations:*** UNPARSED QUERY IS ***
SELECT "EMP"."EMPNO" "EMPNO","EMP"."ENAME" "ENAME",
       "SCOTT"."F_CHANGE_NR"("EMP"."EMPNO") "CHANGE_NR"
FROM "SCOTT"."EMP" "EMP"
WHERE "EMP"."DEPTNO"=20
ORDER BY "SCOTT"."F_CHANGE_NR"("EMP"."EMPNO")
```

- Explanation:
  - ORDER BY elimination is fired before the query transformation.
  - It does not find an ORDER BY clause in the root SELECT and stops.
  - After query transformation, ORDER BY suddenly appears.
  - It is not removed now.

SELECT and ORDER BY – 11g (3)

- /*+ NO_MERGE */ hint solves the problem:

```
SQL> select empno, change_nr
  2  from (select /*+ no_merge */
  3             empno, ename, f_change_nr(empno) change_nr
  4          from emp where deptno = 20 order by 3);
5 rows selected.
SQL> exec counter_pkg.p_check;
fired:5
```

Query rewrite is skipped. ➔ Order-by elimination is applied directly.

- Why bother? Because of views:

```
SQL> create or replace view v_emp as
  2  select empno, ename, f_change_nr(empno) change_nr
  3  from emp where deptno = 20 order by 3;
view created.
SQL> select empno, change_nr from v_emp;
5 rows selected.
SQL> exec counter_pkg.p_check;
fired:10
```

Same effect as in-line view

Multi-Table Example (1)

- Output:

```
SQL> SELECT empno, f_change_nr(empno) change_nr, dname
  2  FROM emp,
  3       dept
  4  WHERE emp.deptno = dept.deptno;
     EMPNO  CHANGE_NR DNAME
      7782       -7782 ACCOUNTING
      ...

14 rows selected.
SQL> exec counter_pkg.p_check;
Fired:14
```

- Explanation
  - JOIN is being applied first. It results in 14 rows.
  - Function in SELECT is fired as many times as the number of rows.

Multi-Table Example (2)

- Output:

```
SQL> SELECT empno, f_change_nr(empno) change_nr, dname
  2  FROM emp,
  3       dept
  4  WHERE emp.deptno(+) = dept.deptno;
     EMPNO  CHANGE_NR DNAME
      7782       -7782 ACCOUNTING
      ...
      7654       -7654 SALES
                       OPERATIONS
15 rows selected.
SQL> exec counter_pkg.p_check;
Fired:15
```

No employees!

- Explanation
  - Outer joint ➔ 15 rows in the output

Multi-Table Example (3)

- Output:

```
SQL> SELECT empno, f_change_nr(dept.deptno) change_nr, dname
  2  FROM emp,
  3       dept
  4  WHERE emp.deptno = dept.deptno;
     EMPNO  CHANGE_NR DNAME
      7782       -7782 ACCOUNTING
       ...
14 rows selected.
SQL> exec counter_pkg.p_check;
Fired:14
```

DEPT has only 4 rows

- Explanation
  - Even if you reference smaller table – number of output rows is the same!
  - Can be fixed by using caching techniques [sorry, different topic!]

Multi-Table Example (4)

- Output:

```
SQL> SELECT empno, f_change_nr(dept.deptno) change_nr, dname
  2  FROM emp,
  3       dept
  4  WHERE emp.deptno = dept.deptno
  5  AND   dept.dname = 'RESEARCH';
     EMPNO  CHANGE_NR DNAME
      7566       -7566 RESEARCH
       ...

5 rows selected.
SQL> exec counter_pkg.p_check;
Fired:5
```

Selected 1 department (but CBO doesn't know about it)!

- Another reminder ☺:
  - Function in SELECT is fired as many times as the number of rows.

Multi-Table Example (5)

- Output:

```
SQL> SELECT empno, f_change_nr(empno) change_nr, dname
  2  FROM emp,
  3       dept
  4  WHERE emp.deptno = dept.deptno
  5  AND   dept.dname = 'RESEARCH'
  6  AND   f_change_nr(empno)<0;
     EMPNO  CHANGE_NR DNAME
      7566       -7566 RESEARCH
       ...

5 rows selected.
SQL> exec counter_pkg.p_check;
Fired:19
```

Extra condition

- Different behavior :
  - Function in SELECT is fired as many times as the number of rows.
  - Function in WHERE is fired unless the row was already eliminated

Multi-Table Example (6)

- Output:

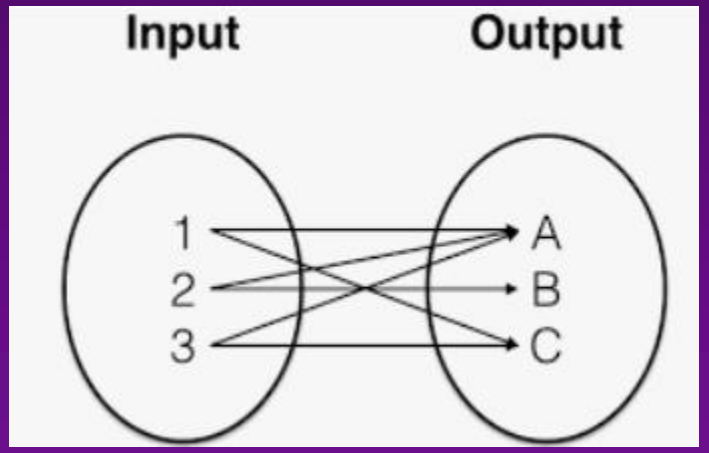
```
SQL> SELECT empno, f_change_nr(empno) change_nr, dname
  2  FROM emp,
  3       dept
  4  WHERE emp.deptno = dept.deptno
  5  AND   dept.deptno = 20
  6  AND   f_change_nr(empno)<0;
     EMPNO  CHANGE_NR DNAME
      7566       -7566 RESEARCH
      ...

5 rows selected.
SQL> exec counter_pkg.p_check;
Fired:10
```

Let's get help from CBO (by using FK column which is indexed)!

- Lesson learned:
  - Different plan ➔ Rows were eliminated by different execution plan

Functions that Return Objects



Test Case (1)

```
CREATE TYPE object_t IS OBJECT
(a_nr NUMBER,
 b_tx VARCHAR2(10),
 c_dt DATE,
 d_tx varchar2(10))

CREATE FUNCTION f_getObject_t RETURN object_t IS
    v_out_t object_t;
BEGIN
    counter_pkg.v_nr:=counter_pkg.v_nr+1;

    v_out_t:=object_t(1,'A',sysdate,'B');

    RETURN v_out_t;
END;
```

SQL Call without Columns

- Output:

```
SQL> SELECT *
  2  FROM
  3       (SELECT f_getObject_t obj FROM dual) t;
...
1 rows selected.
SQL> exec counter_pkg.p_check;
Fired:1
```

- Seems like it works as expected

SQL Call with Columns (1)

- Output:

```
SQL> SELECT t.obj.a_nr, t.obj.b_tx, t.obj.c_dt, t.obj.d_tx
  2  FROM
  3      (SELECT f_getObject_t obj FROM dual) t;
...
1 rows selected.
SQL> exec counter_pkg.p_check;
Fired:4
```

One call per column?

- One call per column?!
  - Behavior validated in 23ai!

SQL Call with Columns (2)

- Output:

```
SQL> SELECT t.obj.a_nr, t.obj.b_tx, t.obj.c_dt
  2  FROM
  3     (SELECT /*+ NO_MERGE */ f_getObject_t obj FROM dual) t;
...
1 rows selected.
SQL> exec counter_pkg.p_check;
Fired:2
```

Always less – but not consistent (and version-dependent)

- Hints/rewrite tricks?
  - WITH clause doesn't help (tried even MATERIALIZE)
  - NO_QUERY_TRANSFORMATION – doesn't help in 11g, same as NO_MERGE in 19c/23ai

SQL Call with Columns (3)

- Output:

```
SQL> SELECT t.obj.a_nr, t.obj.b_tx, t.obj.c_dt, t.obj.d_tx
  2  FROM
  3     (SELECT f_getObject_t obj FROM dual) t
  4  WHERE t.obj.a_nr = 1;
...
1 rows selected.
SQL> exec counter_pkg.p_check;
Fired:5
```

One call per reference?

- One call per reference ???

SQL Call with Columns (4)

- Output:

```
SQL> SELECT t.obj.a_nr, t.obj.b_tx, t.obj.c_dt, t.obj.d_tx
  2  FROM
  3     (SELECT f_getObject_t obj FROM dual) t
  4  WHERE t.obj.a_nr = 1 and t.obj.b_tx = 'A';
...
1 rows selected.
SQL> exec counter_pkg.p_check;
Fired:6
```

- Yes – for hard references

SQL Call with Columns (5)

- Output:

```
SQL> SELECT t.obj.a_nr, t.obj.b_tx, t.obj.c_dt, t.obj.d_tx
  2  FROM
  3     (SELECT f_getObject_t obj FROM dual) t
  4  WHERE t.obj.a_nr = 1
  5  ORDER BY t.obj.b_tx;
...
1 rows selected.
SQL> exec counter_pkg.p_check;
Fired:5
```

Still the same count

- .Well, sometimes – whenever CBO can get away with ☺
  - … i.e. CBO tries to optimize the processing

Function-Based Views

DML Implementation

- UPDATE on view = SELECT + UPDATE
- DELETE on view = SELECT + DELETE
- INSERT on view = Only INSERT

Test Case (1)

```
CREATE TYPE test_tab_ot AS OBJECT (owner_tx VARCHAR2(30),
                                   name_tx VARCHAR2(30),
                                   object_id NUMBER,
                                   type_tx VARCHAR2(30));
CREATE TYPE test_tab_nt IS TABLE OF test_tab_ot;

CREATE FUNCTION f_searchTestTab_tt (i_type_tx VARCHAR2) RETURN test_tab_nt IS
    v_out_tt test_tab_nt;
begin
    SELECT test_tab_ot(owner, object_name, object_id, object_type)
    BULK COLLECT INTO v_out_tt
    FROM test_tab
    WHERE object_type = i_type_tx;
    dbms_output.put_line('Inside f_searchTestTab_tt:'||v_out_tt.count);
    RETURN v_out_tt;
END;
```

Test Case (2)

```
create or replace view v_search_table as
select *
from table(f_searchtesttab_tt('TABLE'));

create or replace trigger v_search_table_iiud
instead of insert or update or delete on v_search_table
begin
    if inserting then
      dbms_output.put_line('Insert');
    elsif updating then
      dbms_output.put_line('Update');
    elsif deleting then
      dbms_output.put_line(`Delete');
    end if;
end;
```

DML Check

```
SQL> INSERT INTO v_search_table (object_id, name_tx)
  2   VALUES (-1,'A');
Insert
1 row created.
```

Function is not fired!

```
SQL> UPDATE v_search_table SET name_tx = 'Test'
  2   WHERE object_id = 5;
Inside f_searchTestTab_tt:1571
Update
1 row updated.
```

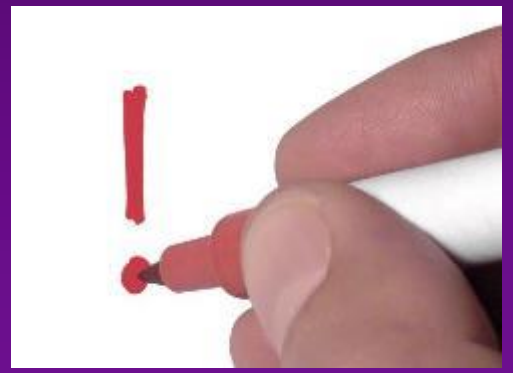
Process all to update one?

```
SQL> DELETE FROM v_search_table WHERE object_id = 5;
Inside f_searchTestTab_tt:1571
Delete
1 row deleted.
```

DULCIAN INC.

Important!

- To fire INSTEAD OF UPDATE/DELETE, Oracle must locate row first.
- Oracle must have the whole resulting set available to filter it
  - … but you can cheat ☺



Parameterized View (1)

```
CREATE PACKAGE global_pkg IS
    v_object_id NUMBER;
END;

CREATE FUNCTION f_searchTestTab_tt (i_type_tx varchar2) RETURN test_tab_nt is
    v_out_tt test_tab_nt;
BEGIN
    IF global_pkg.v_object_id IS NULL THEN
        SELECT test_tab_ot(owner, object_name, object_id, object_type)
        BULK COLLECT INTO v_out_tt
        FROM test_tab
        WHERE object_type = i_type_tx;
    ELSE
        SELECT test_tab_ot(owner, object_name, object_id, object_type)
        BULK COLLECT INTO v_out_tt
        FROM test_tab
        WHERE object_id = global_pkg.v_object_id;
    END IF;
    dbms_output.put_line
       ('Inside f_searchTestTab_tt:'||v_out_tt.count);
    RETURN v_out_tt;
END;
```

Global parameter

Usage (if needed)

Parameterized View (2)

```
SQL> BEGIN global_pkg.v_object_id:=5; END;
  2  /
SQL> UPDATE v_search_table SET name_tx = 'Test'
 2   WHERE object_id = 5;
Inside f_searchTestTab_tt:1
Update
1 row updated.

SQL> DELETE FROM v_search_table WHERE object_id = 5;
Inside f_searchTestTab_tt:1
Delete
1 row deleted.
```

Parameterized View (3)

```
SQL> exec  runstats_pkg.rs_start;
SQL> BEGIN
  2      global_pkg.v_object_id:=NULL;
  3      FOR i IN 1..100 LOOP
  4          UPDATE v_search_table SET name_tx = 'Test'||i
  5          WHERE object_id = 5;
  6      END LOOP;
  7  END;
  8  /
SQL> exec  runstats_pkg.rs_middle;
SQL> BEGIN
  2      global_pkg.v_object_id:=5;
  3      FOR i IN 1..100 LOOP
  4          UPDATE v_search_table SET name_tx = 'Test'||i
  5          WHERE object_id = 5;
  6      END LOOP;
  7  END;
  8  /
SQL> exec  runstats_pkg.rs_stop;
Run1 ran in 181 cpu hsecs
Run2 ran in 28 cpu hsecs
run 1 ran in 646.43% of the time
```

5x improvement

Existing Code Base

DULCIAN INC.

Challenges of Real Systems

- Code instrumentation is THE key success factor.
  - ... Since you cannot optimize what you don't know
- All new projects have to be developed performance-aware
  - ... but it is hard to add a new monitoring framework to an existing system
- Oracle Hierarchical Profiler could be the answer
  - … Since you only need a wrapper on top of the existing code

Typical Situation

- Help-desk client's performance complaints:
  - "Developer checked 10046 trace and couldn't find anything suspicious."
  - "I noticed that the core query contains a user-defined PL/SQL function."
- Action:
  - Wrap suspicious call in HProf start/stop in TEST instance (with the same volume of data)

Suspect

```
SQL> exec dbms_hprof.start_profiling ('IO', 'HProf_Case1.txt');

SQL> declare
  2       v_tx varchar2(32767);
  3  begin
  4       select listagg(owner_tx,',') within group (order by 1)
  5       into v_tx
  6       from (
  7            select distinct scott.f_change_tx(owner) owner_tx
  8            from scott.test_tab
  9            );
 10  end;
 11  /

SQL> exec dbms_hprof.stop_profiling;
```

1. Only 26 owners!
2. Function is doing basic formatting

Profile

Function Elapsed Time (microsecs) Data sorted by Total Subtree Elapsed Time (microsecs)

508391 microsecs (elapsed time) & 100006 function calls

Subtree	Ind%	Function	Ind%	Descendants	Ind%	Calls	Ind%	Function Name	SQL ID	SQL TEXT
508391	100%	14	0.0%	508377	100%	2	0.0%	__plsql_vm		
508377	100%	171	0.0%	508206	100%	2	0.0%	__anonymous_block		
508206	100%	328430	64.6%	179776	35.4%	1	0.0%	__static_sql_exec_line4 (Line 4)	27t27npwd3n0j	SELECT LISTAGG(OWNER_TX,',') WITHIN GROUP (ORDER B
179776	35.4%	66436	13.1%	113340	22.3%	50000	50.0%	__plsql_vm@1		
113340	22.3%	113340	22.3%	0	0.0%	50000	50.0%	SCOTT.F_CHANGE_TX.F_CHANGE_TX (Line 1)		
0	0.0%	0	0.0%	0	0.0%	1	0.0%	SYS.DBMS_HPROF.STOP_PROFILING (Line 453)		

50k calls?!

Here is my time!

<Show files>

Findings

- Problem:
  - Time is wasted on very cheap function which is fired many times
  - … because the original developer "guessed" at the query behavior
  - … function was doing basic formatting, so the output would also be distinct
  - … but forgot to tell the CBO ➔ GIGO!
- Solution:
  - Rewrite query in a way that helps the CBO
  - … and remind all developers:
    - The number of function calls in SQL will surprise you if you don't measure them.

Fix

```
SQL> exec dbms_hprof.start_profiling ('IO', 'HProf_Case1_fix.txt');

SQL> declare
  2       v_tx varchar2(32767);
  3  begin
  4       select listagg(owner_tx,',') within group (order by 1)
  5       into v_tx
  6       from (
  7            select  scott.f_change_tx(owner) owner_tx
  8            from (select distinct owner
  9                  from scott.test_tab)
 10             );
 11  end;
 12  /

SQL> exec dbms_hprof.stop_profiling
```

Filter first!

Updated Profile

Function Elapsed Time (microsecs) Data sorted by Total Subtree Elapsed Time (microsecs)

18230 microsecs (elapsed time) & 58 function calls

Subtree	Ind%	Function	Ind%	Descendants	Ind%	Calls	Ind%	Function Name	SQL ID	SQL TEXT
18230	100%	15	0.1%	18215	100%	2	3.4%	__plsql_vm		
18215	100%	139	0.8%	18076	99.2%	2	3.4%	__anonymous_block		
18076	99.2%	17954	98.5%	122	0.7%	1	1.7%	__static_sql_exec_line4 (Line 4)	b4pduc9z5xybc	SELECT LISTAGG(OWNER_TX,',') WITHIN GROUP (ORDER B
122	0.7%	42	0.2%	80	0.4%	26	44.8%	__plsql_vm@1		
80	0.4%	80	0.4%	0	0.0%	26	44.8%	SCOTT.F_CHANGE_TX.F_CHANGE_TX (Line 1)		
0	0.0%	0	0.0%	0	0.0%	1	1.7%	SYS.DBMS_HPROF.STOP_PROFILING (Line 453)		

28 times faster!

26 calls

Now What?

DULCIAN INC.

First Level

- "Sanity" questionnaire:
  - Should detected functions be called at all?
  - Are correct versions of functions being called (overloads!)?
  - Are functions being called the correct number of times?
    - … correct = matching the business case

Sorry, not today!
That a topic
for a full-day seminar!

DULCIAN INC.

Second Level

- "Optimization" questionnaire
  - Should this functionality be implemented as a function at all?
    - … doing everything in SQL allows avoiding context switches!
      - … or at least the cost can be decreased via PRAGMA UDF/Native compilation/SQL Transpiler
  - Any way to cache results?
    - Deterministic functions/Scalar subqueries
    - Result cache

Sorry, also not today!
That a topic
for TWO-day seminar…

DULCIAN INC.

Summary

- User-defined PL/SQL functions are VERY powerful
  - … so use it CAREFULLY!
- Understanding your data help writing better code
  - … and don't forget to share that knowledge with CBO
- Measure everything
  - … and you will not be surprised!

Contact Information

- Michael Rosenblum – mrosenblum@dulcian.com
- Dulcian, Inc. website - www.dulcian.com
- Blog: wonderingmisha.blogspot.com