# ORACLE

# TECHNOLOGY NETWORK

| ORACLE.COM   | PARTNERS | <u>BUY</u> | DOWNLOAD | <u>SUPPORT</u> | CONTACT US |
|--------------|----------|------------|----------|----------------|------------|
| PRODUCT CENT | •        |            |          |                |            |

| TECHNOLOGY CENTERS |     |  |
|--------------------|-----|--|
| COMMUN             | ΙТΥ |  |
| Search             |     |  |

Welcome howard ( Sign Out | Account )

Page 1 of :

SEARCH SELECT COUNTRY -

# Table of Contents

## . Home

.<u>Q&A</u>

- . Downloads
- <u>Test Your PL/SQL Knowledge</u>
- Submit a Question

## **Top 5 Questions**

Cursor best practices (viewed 6894 times) SQL%ROWCOUNT in cursor FOR loop? Sure, for all the good it will do you... (viewed 5510 times)

Does Oracle have a tool for formatting PL/SQL? (viewed 4723 times)

Can I start a timer inside PL/SQL? (viewed 4251 times)

Exception, Exception, Where Did You Raise? (viewed 3561 times)

#### Other PL/SQL Resources

- Oracle Application Express
- . PL/SQL Technology Center
- New PL/SQL Features in Oracle Database 10g Version 10.1.0
- . Sample Code
- Discussion Forum

## Ways to hide your code

Asked by Chandan Singh Bhamra from Bangalore, Karanataka on Tuesday, December 6, 2005

# Question:

I want to allow everyone to run my procedures and functions, but I don't want them to be able to see the code behind the program header. How can I accomplish that?

#### Answer:

Oracle offers two ways to hide your code:

"Wrap" or obfuscate your code so that it can still be compiled and executed, but cannot be read Hide program implementations within a package body 2.

Wrapping is certainly the best way to ensure that others cannot read your code, even if they connect to the schema that owns the code. However, it does have drawbacks. Let's take a look at both options.

To wrap your code, call the wrap executable. Suppose, for example, that I start with the following procedure definition in the file before\_wrap.sql:

| CREA | ΤE  | OR  | REPLACE   | PROCEI | URE  | my_proc | cedure       |    |
|------|-----|-----|-----------|--------|------|---------|--------------|----|
| IS   |     |     |           |        |      |         |              |    |
| BEGI | Ν   |     |           |        |      |         |              |    |
| D    | BMS | _Ot | JTPUT.put | _line  | ( ': | running | my_procedure | ') |
| END  | my_ | pro | cedure;   |        |      |         |              |    |
| ,    |     |     |           |        |      |         |              |    |

I then call the wrap executable as follows:

wrap iname=before\_wrap.sql oname=after\_wrap.plb

The after\_wrap.plb file (plb is a rough-hewn acronym for "PL/SQL Binary" and is the standard extension used by Oracle for its wrapped code) then contains the following text:

CREATE OR REPLACE PROCEDURE my\_procedure wrapped a000000 h2

abcd 7 66 96

JglYKBnago+nNjoDigwhI233Q8Ywg5nnm7+fMr2ywFyl8F8oqVJTeta+bnFVAHNTjo6OcW7P loZzhITh5oTSqVcZxutyXDWENe1NIZaWgpZMqMRlvA4WJK0qY6AWXEw16+zZPXJE1SH2OaY9 CKRI



While that doesn't look anything like PL/SQL code, you can compile it into the database and execute the program, just as you would if it were not wrapped. For example:

SOL> @after wrap.plb

Procedure created.

SQL> exec my\_procedure running my procedure

For more information about Oracle's obfuscation capabilities, read the documentation.

By the way, in Oracle Database 10g Release 2, you can use the DBMS\_DDL package to obfuscate PL/SQL program units that are created dynamically (at run-time).

Obfuscation is very handy for independent software vendors who need to hide their proprietary code

from the prying eyes of competitors. It is less obvious that one should use it as a general mechanism to hide code from other developers in your organization.

The problem with obfuscating your code is that no one can read the source code in the database, even the owner of the code, even those who need to debug the code to fix problems.

If you would like to compile your program into the database so that the owner of the program can read and work with code, but others (who are allowed to run the program) cannot see the implementation, you should define your programs in packages.

If you implement your functionality in schema-level (a.k.a, stand-alone) procedures and functions, then granting execute authority also allows programmers connected to other schema to see the source code of those program units. This fact is demonstrated below:

SQL> CONNECT scott/tiger Connected. SQL> CREATE OR REPLACE PROCEDURE my\_procedure 2 IS 3 BEGIN 4 DBMS\_OUTPUT.put\_line ( 'my\_procedure' ); 5 END my\_procedure; 6 /

Procedure created.

SQL> GRANT EXECUTE ON my\_procedure TO hr 2 /

Grant succeeded.

```
SQL> CONNECT hr/hr
Connected.
SQL> SELECT
            TYPE, text
 2
       FROM all_source
   WHERE OWNER = 'SCOTT' AND NAME = 'MY_PROCEDURE'
ORDER BY TYPE, line
 3
 4
 5
    /
TYPE
                   TEXT
      _____
                                      ------
PROCEDURE
                  PROCEDURE my procedure
```

| Incoord   | Incomposite my_procedure                            |
|-----------|---|
| PROCEDURE | IS  |
| PROCEDURE | BEGIN   |
| PROCEDURE | <pre>DBMS_OUTPUT.put_line ( 'my_procedure' );</pre> |
| PROCEDURE | END my_procedure;                                   |

Suppose that you instead implement your procedures and functions within a package. You then can grant execute authority on that package to other schemas. Users connected to those schemas will be able to only see the package specifications, and not the bodies. This fact is demonstrated below:

```
SQL> CONNECT scott/tiger
Connected.
SQL> CREATE OR REPLACE PACKAGE my_package
2 IS
3 PROCEDURE hidden_implementation;
4 END my_package;
5 /
```

Package created.

SQL> CREATE OR REPLACE PACKAGE BODY my\_package

```
2
   IS
 3
       PROCEDURE hidden_implementation
  4
       TS
 5
       BEGIN
  6
         DBMS_OUTPUT.put_line ( 'hidden_implementation' );
 7
       END hidden_implementation;
 8
   END my_package;
 9
Package body created.
SQL> GRANT EXECUTE ON my_package TO hr
 2 /
Grant succeeded.
SQL> CONNECT hr/hr
Connected.
SQL> SELECT
           TYPE, text
 2
       FROM all_source
       WHERE owner = 'SCOTT' AND NAME = 'MY_PACKAGE'
 3
   ORDER BY TYPE, line
 4
 5
    1
TYPE
                  TEXT
  _____ ___
                         _____
PACKAGE
                  PACKAGE my_package
```

\_ Printer View

PACKAGE PACKAGE PACKAGE IS
 PROCEDURE hidden\_implementation;
END my\_package;

Notice that none of the lines from the package body are visible through ALL\_SOURCE. Back to the Table of Contents

Copyright © 1999-2005, Oracle Corporation. All Rights Reserved.

About OTN I Contact Us I About Oracle I Legal Notices and Terms of Use I Privacy Statement Powered by Oracle Application Express