Database Programming with PL/SQL

15-4
Hiding Your Source Code
Objectives

This lesson covers the following objectives:

• Describe the benefits of obfuscated PL/SQL source code

• **Use the DBMS_DDL.CREATE_WRAPPED** server-supplied procedure

• Describe how to use the Wrapper utility to obfuscate PL/SQL source code
Purpose

• Imagine that you have spent a lot of time and money inventing a completely new and different type of DVD player. You want people to buy and use it - of course - but you don’t want them to look inside to find out how it works.
• If they did, they could steal your invention and make and sell the DVD player themselves.
• Similarly, when you create a clever PL/SQL package, you may want other users to execute it, but you don’t always want them to be able to see the details of the package’s source code.
• Let’s examine how you can hide your source code from other users.
PL/SQL Source Code in the Data Dictionary

• You already know that when you create a PL/SQL program – procedure, function or package – the source code is loaded into the Data Dictionary, and you can see it using the Data Dictionary view `USER_SOURCE`:

```sql
CREATE OR REPLACE PROCEDURE mycleverproc
    (p_param1 IN     NUMBER, p_param2 OUT NUMBER)
IS BEGIN
    ... /* some clever but private code here */
END mycleverproc;

SELECT TEXT FROM USER_SOURCE
    WHERE TYPE = 'PROCEDURE' AND NAME = 'MYCLEVERPROC'
ORDER BY LINE;
```

• If you now grant `EXECUTE` privilege on the procedure to other users, what can they see?
PL/SQL Source Code in the Data Dictionary

• SUSAN can describe your procedure.
• That’s fine; she needs to know its parameters and their data types in order to invoke it successfully.
• Can she also see your source code?

You> GRANT EXECUTE ON mycleverproc TO susan;

Susan> DESCRIBE you.mycleverproc

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Argument</th>
<th>In Out</th>
<th>Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>MYCLEVERPROC</td>
<td>P_PARAM1</td>
<td>IN</td>
<td>NUMBER</td>
</tr>
<tr>
<td></td>
<td>P_PARAM2</td>
<td>OUT</td>
<td>NUMBER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - 2</td>
</tr>
</tbody>
</table>
PL/SQL Source Code in the Data Dictionary

Yes, \textit{Susan} can see your source code.

\begin{verbatim}
Susan> SELECT TEXT FROM ALL_SOURCE
    WHERE OWNER = 'YOU' AND TYPE = 'PROCEDURE'
    AND NAME = 'MYCLEVERPROC'
    ORDER BY LINE;
\end{verbatim}

\begin{table}[h]
\centering
\begin{tabular}{l}
\hline
\textbf{TEXT} \\
\hline
PROCEDURE mycleverproc \\
(p_param1 IN NUMBER, p_param2 OUT NUMBER) \\
IS BEGIN \\
... /* some clever but private code here */ \\
END mycleverproc;
\hline
\end{tabular}
\end{table}
Obfuscating PL/SQL Source Code

- Anyone who has EXECUTE privilege on a procedure or function can see your source code in ALL_SOURCE.

- We can hide the source code by converting it into a set of cryptic codes before we compile the subprogram.

- Hiding the source code is called obfuscation, and converting the source code to cryptic codes is called wrapping the code.

- When we compile the subprogram, only the wrapped code (the cryptic codes) are loaded into the Data Dictionary.
Obfuscating PL/SQL Source Code

• There are two ways to wrap the source code:
  – Using the `DBMS_DDL.CREATE_WRAPPED` Oracle-supplied package procedure
  – Using the PL/SQL wrapper utility program, `WRAP`.

• To use the `WRAP` utility you must be able to log into the database server computer, so we can’t use this utility in Application Express.

• We can use the `DBMS_DDL.CREATE_WRAPPED` package and the end results are exactly the same.
Using the `DBMS_DDL.CREATE_WRAPPED` Procedure

- We must pass the complete code of our subprogram as a single **IN** argument with data type `VARCHAR2`.

- A PL/SQL `VARCHAR2` variable has a maximum size of 32,767 characters, so this is the maximum size of our source code.

- Our source code is wrapped, and the wrapped code is automatically compiled.
Using 
DBMS_DDL.CREATE_WRAPPED:

Example 1

• Here we obfuscate the code.

```
BEGIN

  DBMS_DDL.CREATE_WRAPPED
  ('CREATE OR REPLACE PROCEDURE mycleverproc
   (p_param1 IN     NUMBER, p_param2 OUT NUMBER)
   IS BEGIN
     ... /* some clever but private code here */
   END mycleverproc;');

END;
```

• What can SUSAN see now?

```
GRANT EXECUTE ON mycleverproc TO SUSAN;
```
Using `DBMS_DDL.CREATE_WRAPPED`: Example 1

**SUSAN** can still **DESCRIBE** your procedure to see the parameters and their data types, but the source code has been obfuscated.

```
Susan> SELECT TEXT FROM ALL_SOURCE
    WHERE OWNER = 'YOU' AND TYPE = 'PROCEDURE'
    AND NAME = 'MYCLEVERPROC'
    ORDER BY LINE;
```

```
PROCEDURE mycleverproc wrapped a000000 369 abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd abcd yoAPHESujEupD01kt/bISuHDafqvJCE2xU29PV2I2l9d/LkpwW4KENa6uKMkOCk9trC4aAO9D2 dfZpplAn9jmmh+AcQA==
```
Using `DBMS_DDL.CREATE_WRAPPED`: Example 1

- What happens when you view your own source code?

```sql
You> SELECT TEXT FROM USER_SOURCE
   WHERE TYPE = 'PROCEDURE' AND NAME = 'MYCLEVERPROC'
   ORDER BY LINE;
```

- Even you see only the obfuscated code, because the original source code has not been loaded into the Data Dictionary.

- Make sure you keep a private copy of the source code in case you want to modify it later!
Using `DBMS_DDL.CREATE_WRAPPED`: Example 2

If the source code is long, it may be easier to assign it line by line to a `VARCHAR2` variable and pass the variable as an actual parameter to the `CREATE_WRAPPED` procedure:

```sql
DECLARE
  v_code  VARCHAR2(32767);
BEGIN
  v_code := 'CREATE OR REPLACE FUNCTION myclevererfunc ' || '(p_param1 IN NUMBER) ' || 'RETURN NUMBER IS BEGIN ' || '... /* some even cleverer but private code */ ' || 'END myclevererfunc;';
  DBMS_DDL.CREATE_WRAPPED(v_code);
END;
```
Wrapping Package Code

• You can wrap PL/SQL package body code just like procedures and functions:

```
BEGIN
    DBMS_DDL.CREATE_WRAPPED
    ('CREATE OR REPLACE PACKAGE BODY mycleverpack
     ...
     END mycleverpack;');
END;
```

• You can also try to wrap the package specification.

• You won’t get an error, but the specification will *not* be obfuscated.

• Why not?
Using the PL/SQL Wrapper Utility

• To use the WRAP utility program, you must log into the operating system of your database server computer.

• There are three steps:
  1. Create a text file containing your complete unwrapped source code.
  2. Execute WRAP to create a second text file containing the wrapped code.
  3. Connect to the database and execute the wrapped text file as a script to compile the wrapped code into the Data Dictionary.
Using the PL/SQL Wrapper Utility: Example

Step:

1. Use a text editor to create a file containing your complete source code, starting with `CREATE OR REPLACE ...` and ending with `END ...;` Let’s suppose the text file is called `mysourcecode.sql`.

2. Execute the WRAP utility at the operating system prompt (for example, a DOS prompt on Windows) and pass the name of your text file as an argument:

   ```
   C:> WRAP INAME=mysourcecode.sql
   ```

   – This creates the wrapped code in a second file called `mysourcecode.plb`.
Using the PL/SQL Wrapper Utility: Example

Step:
2. (continued): You can give your .plb file a different name:

   C:> WRAP INAME=mysourcecode.sql ONAME=mywrappedcode.plb

3. Connect to the database and execute mysourcecode.plb as a script.
   – To do this in Application Express, choose SQL Workshop > SQL Scripts > Upload, choose your .plb file, then click Upload.
   – Then, execute the script just like any other script.
   – This compiles the wrapped PL/SQL code into the Data Dictionary.
   – Now, it can be executed just like any other PL/SQL subprogram.
Comparing the Two Methods for Wrapping Code

- Which method is better, `DBMS_DDL.CREATE_WRAPPED` or the Wrapper utility?
- For very large PL/SQL programs where the source code is more than 32,767 bytes, you must use the `WRAP` utility.
- For smaller programs, `DBMS_DDL.CREATE_WRAPPED` is easier because you don’t need to log on to the database server machine, and everything is done in a single step.
Terminology

Key terms used in this lesson included:

• `DBMS_DDL.CREATE_WRAPPED`

• Obfuscation

• Wrapper utility

• Wrapping PL/SQL source code
Summary

In this lesson, you should have learned how to:

• Describe the benefits of obfuscated PL/SQL source code

• **Use the** DBMS_DDL.CREATE_WRAPPED **server-supplied procedure**

• Describe how to use the Wrapper utility to obfuscate PL/SQL source code