**Using PL/SQL Stored Functions and Procedures**

PL/SQL is Oracle's procedural language extension to SQL. PL/SQL procedures and functions are stored and run in the database. Using PL/SQL lets all database applications reuse logic, no matter how the application accesses the database. Many data-related operations can be performed in PL/SQL faster than extracting the data into a program (for example, Python) and then processing it. Oracle also supports Java stored procedures.

In this tutorial, you will create a PL/SQL stored function and procedure and call them in Python scripts. Perform the following steps:

.

.

.

.

.

|  |
| --- |
| Start SQL\*Plus and create a new table, ptab with the following command:  sqlplus pythonhol/welcome@127.0.0.1/orcl    create table ptab (mydata varchar(20), myid number);  exit  sp01-gif |
| Review the [create\_func.sql](http://www.oracle.com/technology/obe/11gr2_db_prod/appdev/opensrclang/python/files/create_func.sql) script which creates a PL/SQL stored function myfunc() to insert a row into the ptab table, and return double the inserted value:  set echo on    create or replace function    myfunc(d\_p in varchar2, i\_p in number) return number as    begin    insert into ptab (mydata, myid) values (d\_p, i\_p);    return (i\_p \* 2);    end;    /    show errors    Start SQL\*Plus and run the script:  sqlplus pythonhol/welcome@127.0.0.1/orcl  @create\_func  exit    sp02-gif |
| Review the code as follows that is contained in the [plsql\_func.py](http://www.oracle.com/technology/obe/11gr2_db_prod/appdev/opensrclang/python/files/plsql_func.py) file in the $HOME directory.  import cx\_Oracle  con = cx\_Oracle.connect('pythonhol/welcome@127.0.0.1/orcl')  cur = con.cursor()  res = cur.callfunc('myfunc', cx\_Oracle.NUMBER, ('abc', 2))  print res  cur.close()  con.close()  This uses callfunc() to execute the function. The constant cx\_oracle.NUMBER indicates that the return value is numeric. The two PL/SQL function parameters are passed as a tuple and bound to the function parameter arguments.  From a terminal window, run:  **python plsql\_func.py**  plsql-func-gif  The output is result of the PL/SQL function calculation. |
| To call a PL/SQL procedure, use the cur.callproc() method.  Review the [create\_proc.sql](http://www.oracle.com/technology/obe/11gr2_db_prod/appdev/opensrclang/python/files/create_proc.sql) script which creates a PL/SQL procedure myproc() to accept two parameters. The second parameter contains an OUT return value.  set echo on    create or replace procedure    myproc(v1\_p in number, v2\_p out number) as    begin    v2\_p := v1\_p \* 2;    end;    /    show errors    Start SQL\*Plus and run the script::  sqlplus pythonhol/welcome@127.0.0.1/orcl  @create\_proc  exit  sp03-gif |
| Review the code as follows that is contained in the [plsql\_proc.py](http://www.oracle.com/technology/obe/11gr2_db_prod/appdev/opensrclang/python/files/plsql_proc.py) file in the $HOME directory.  import cx\_Oracle  con = cx\_Oracle.connect('pythonhol/welcome@127.0.0.1/orcl')  cur = con.cursor()  myvar = cur.var(cx\_Oracle.NUMBER)  cur.callproc('myproc', (123, myvar))  print myvar.getvalue()  cur.close()  con.close()  This creates a numeric variable myvar to hold the OUT parameter. The number 123 and the return variable name are bound to the procedure call parameters using a tuple.  From a terminal window, run:  **python plsql\_proc.py**  sp04-gif  The getvalue() method displays the returned value. |