Using Query Database for Oracle Stored Procedures and Functions

With the Query Database 2.4, Query Database can now be used to access both Oracle stored procedures and functions which have been compiled on Oracle databases.

Basics

The syntax is fairly simple where the query is in the Oracle SQL Plus format. The critical point is that the flag '**ORACLE_SP**' must appear on the query database invocation.

```
query_database, query, nrows, data, ...(whatever other parameters are relevant) ...,
/ORACLE_SP
```

Examples

Stored Procedure example

The following examples show the correct IDL declaration and invocation for stored procedures and functions. Consider a stored procedure with the following definition:

```
CREATE OR REPLACE PROCEDURE INOUTTEST
(
  v_input IN VARCHAR2 DEFAULT NULL ,
  v_valid OUT NUMBER,
  v_inOut IN OUT NUMBER,
  v_length OUT NUMBER
)
```

Note that this stored procedure has IN, OUT, and IN OUT parameters. The invocation consists of the stored procedure name (inOutTest) followed by a left parenthesis, the input (either IN or IN OUT) parameters in order, and a left parenthesis. In our example above the first input is 'v_input' so in our invocation the string "xyz" will passed to Oracle. In the example below the number 27 is the next input parameter, an IN OUT, named 'v_inOut'. These will be passed to the stored procedure

```
query = 'inOutTest(xyz, 27)'
    query_database, query, data, nrows, dbloginfile='.qdbResources',
dbResourceId='ORACLE_TEST_READER', /ORACLE_SP
    print, 'query = ', query, ', nrows = ', nrows, ', data = ', data[0].V_VALID,
data[0].V_INOUT, data[0].V_LENGTH
```

The first parameter to the stored procedure is in IDL (or Java) a string; the second a number. The behavior would be exactly the same if the query were written as below where quotation marks make it even more explicit that the first parameter is a string.

```
query = 'inOutTest("xyz", 27)'
```

Note that the data structure returned in the data variable by Query Database uses the names as defined by the stored procedures. E.g. 'v_valid' is an out parameter in the stored procedure declaration. Hence data[0].v_valid is an element of the returned data structure.

The order for import parameters is critcal. If the order of the parameters were reversed, re the following,

```
query = 'inOutTest(27, "xyz")'
```

it would be an error because although the first element "27" will work as a string, the second element "xyz" will not work as a number.

All input parameters must be explicitly specified. Defaults are not allowed, even if they are declared in the SQL. In the above SQL the first IN parameter, v_input, has a default value of null. However, both of the following will produce an error.

```
query = 'inOutTest(, 27)'
 query = 'inOutTest(27)'
```

Function example

Function call are almost the same with one difference. Consider the following

```
CREATE OR REPLACE FUNCTION CharacterExtractionFunction(v_input IN VARCHAR2,  v_upper
OUT VARCHAR2)
RETURN VARCHAR2 is v_lower VARCHAR2(1);
```

A sample correct invocation would be

```
query = 'CharacterExtractionFunction(xyz)'
  query_database, query, data, nrows, dbloginfile='.qdbResources',
dbResourceId='ORACLE_TEST_READER', /ORACLE_SP
  print, 'query = ', query, ', nrows = ', nrows, ', data = ', data[0].result, ',
v_upper = ', data[0].v_upper
```

Note that the function result is returned in a structure member named 'result'

Queries with either null inputs or null return values

Specifying a null input to a query is very trivial. In our stored procedure invocation above a null rather than a string can be passed as the first input as follows:

```
query = 'inOutTest(null, 0)'
```

The 'Gotcha' for nulls

At present Query Database cannot distinguish between the four character string variable 'null' and a null input. When 'null' appears in an any query the assumption is made that the user wants a null input rather than the four character string 'null'.

How nulls are returned

Because IDL does not have a genuine null like Java or other similar languages, when the query results in a null for a string type, the empty (zero length) strength is returned. For numeric types, the value 0 is returned.

Queries that pass times or return times (Oracle Date or Timestamp variables /

columns)

For the following query where the input is an Oracle Timestamp

```
CREATE OR REPLACE FUNCTION dateToGPSMicrosecond(date_in timestamp)
 RETURN float is microseconds_out float;
```

For the correct invocation the time must be in the format 'YYYY-MM-DD HH:MM:SS.FFF' where ".FFF' is an optional fraction of a second. The following is a correct invocation that will produce the desired result.

```
query = 'dateToGPSMicrosecond(2013-01-07 20:58:46.463)'
```

Correct results are neither guaranteed nor predicted for any other time formats.

Similarly when a time value is returned in a query, it is returned as an IDL string in that same format.

Specifying url, server, etc.

For ALL Oracle queries (not just stored procedures or functions), query database works well when a user, password, and url are specified OR when a dbResourceId is specified in conjunction with a multi-database resource files, re Query Database single and multi-database resource file formats.

However, unlike Sybase, where the name of the server combined with the database (schema) form a valid database url, the database (schema) has no relevance to an Oracle url. Specifying a database in the query database call is actively discouraged, and typically, although not always produces an error. The form that is optimal and correct for Sybase connections is not likely to produce correct results for Oracle databases.