75
DBMS_DESCRIBE

You can use the `DBMS_DESCRIBE` package to get information about a PL/SQL object. When you specify an object name, `DBMS_DESCRIBE` returns a set of indexed tables with the results. Full name translation is performed and security checking is also checked on the final object.

This chapter contains the following topics:

- Overview
- Security Model
- Types
- Exceptions
- Examples
- Summary of DBMS_DESCRIBE Subprograms

DBMS_DESCRIBE Overview

This package provides the same functionality as the Oracle Call Interface `OCIDescribeAny` call.

> **See Also:**
>
> *Oracle Call Interface Developer's Guide*

DBMS_DESCRIBE Security Model

This package is available to `PUBLIC` and performs its own security checking based on the schema object being described.

DBMS_DESCRIBE Types

The `DBMS_DESCRIBE` package declares two PL/SQL table types, which are used to hold data returned by `DESCRIBE_PROCEDURE` in its `OUT` parameters.

The types are:

```
TYPE VARCHAR2_TABLE IS TABLE OF VARCHAR2(30)
    INDEX BY BINARY_INTEGER;

TYPE NUMBER_TABLE IS TABLE OF NUMBER
    INDEX BY BINARY_INTEGER;
```

DBMS_DESCRIBE Exceptions

`DBMS_DESCRIBE` can raise application errors in the range -20000 to -20004.

Table 75-1 DBMS_DESCRIBE Errors

Error	Description
ORA-20000	ORU 10035: cannot describe a package ('X') only a procedure within a package.
ORA-20001	ORU-10032: procedure 'X' within package 'Y' does not exist.
ORA-20002	ORU-10033: object 'X' is remote, cannot describe; expanded name 'Y'.
ORA-20003	ORU-10036: object 'X' is invalid and cannot be described.
ORA-20004	Syntax error attempting to parse 'X'.

DBMS_DESCRIBE Examples

One use of the DESCRIBE_PROCEDURE procedure is as an external service interface.

For example, consider a client that provides an OBJECT_NAME of SCOTT.ACCOUNT_UPDATE, where ACCOUNT_UPDATE is an overloaded function with specification:

```
TABLE account (accnt_no NUMBER, person_id NUMBER,
               balance NUMBER(7,2))
TABLE person  (person_id number(4), person_nm varchar2(10))

CREATE OR REPLACE PACKAGE ACCOUNT_PKG is   FUNCTION ACCOUNT_UPDATE (accnt_no    NUMBER,
                           person      person%rowtype,
                           amounts     DBMS_DESCRIBE.NUMBER_TABLE,
                           trans_date  DATE)
                           return       account.balance%type;

   FUNCTION ACCOUNT_UPDATE (accnt_no     NUMBER,
                           person       person%rowtype,
                           amounts      DBMS_DESCRIBE.NUMBER_TABLE,
                           trans_no     NUMBER)
                           return       account.balance%type;
END;
```

This procedure might look similar to the following output:

```
overload position  argument level  datatype length prec scale rad
-------- --------- -------- ------ -------- ------ ---- ----- ---
       1         0                0        2     22    7     2  10
       1         1   ACCNT_NO     0        2      0    0     0   0
       1         2   PERSON       0      250      0    0     0   0
       1         1   PERSON_ID    1        2     22    4     0  10
       1         2   PERSON_NM    1        1     10    0     0   0
       1         3   AMOUNTS      0      251      0    0     0   0
       1         1                1        2     22    0     0   0
       1         4   TRANS_DATE   0       12      0    0     0   0
       2         0                0        2     22    7     2  10
       2         1   ACCNT_NO     0        2     22    0     0   0
       2         2   PERSON       0        2     22    4     0  10
       2         3   AMOUNTS      0      251     22    4     0  10
       2         1                1        2      0    0     0   0
       2         4   TRANS_NO     0        2      0    0     0   0
```

The following PL/SQL procedure has as its parameters all of the PL/SQL datatypes:

```
CREATE OR REPLACE PROCEDURE p1 (
        pvc2    IN     VARCHAR2,
```

```
        pvc     OUT    VARCHAR,
        pstr    IN OUT STRING,
        plong   IN     LONG,
        prowid  IN     ROWID,
        pchara  IN     CHARACTER,
        pchar   IN     CHAR,
        praw    IN     RAW,
        plraw   IN     LONG RAW,
        pbinint IN     BINARY_INTEGER,
        pplsint IN     PLS_INTEGER,
        pbool   IN     BOOLEAN,
        pnat    IN     NATURAL,
        ppos    IN     POSITIVE,
        pposn   IN     POSITIVEN,
        pnatn   IN     NATURALN,
        pnum    IN     NUMBER,
        pintgr  IN     INTEGER,
        pint    IN     INT,
        psmall  IN     SMALLINT,
        pdec    IN     DECIMAL,
        preal   IN     REAL,
        pfloat  IN     FLOAT,
        pnumer  IN     NUMERIC,
        pdp     IN     DOUBLE PRECISION,
        pdate   IN     DATE,
        pmls    IN     MLSLABEL) AS

BEGIN
    NULL;
END;
```

If you describe this procedure using the following:

```
CREATE OR REPLACE PACKAGE describe_it AS

    PROCEDURE desc_proc (name VARCHAR2);

END describe_it;

CREATE OR REPLACE PACKAGE BODY describe_it AS

  PROCEDURE prt_value(val VARCHAR2, isize INTEGER) IS
    n INTEGER;
  BEGIN
    n := isize - LENGTHB(val);
    IF n < 0 THEN
      n := 0;
    END IF;
    DBMS_OUTPUT.PUT(val);
    FOR i in 1..n LOOP
      DBMS_OUTPUT.PUT(' ');
    END LOOP;
  END prt_value;

  PROCEDURE desc_proc (name VARCHAR2) IS

      overload     DBMS_DESCRIBE.NUMBER_TABLE;
      position     DBMS_DESCRIBE.NUMBER_TABLE;
      c_level      DBMS_DESCRIBE.NUMBER_TABLE;
      arg_name     DBMS_DESCRIBE.VARCHAR2_TABLE;
      dty          DBMS_DESCRIBE.NUMBER_TABLE;
      def_val      DBMS_DESCRIBE.NUMBER_TABLE;
```

```
        p_mode       DBMS_DESCRIBE.NUMBER_TABLE;
        length       DBMS_DESCRIBE.NUMBER_TABLE;
        precision    DBMS_DESCRIBE.NUMBER_TABLE;
        scale        DBMS_DESCRIBE.NUMBER_TABLE;
        radix        DBMS_DESCRIBE.NUMBER_TABLE;
        spare        DBMS_DESCRIBE.NUMBER_TABLE;
        idx          INTEGER := 0;

    BEGIN
        DBMS_DESCRIBE.DESCRIBE_PROCEDURE(
                name,
                null,
                null,
                overload,
                position,
                c_level,
                arg_name,
                dty,
                def_val,
                p_mode,
                length,
                precision,
                scale,
                radix,
                spare);

        DBMS_OUTPUT.PUT_LINE('Position    Name        DTY  Mode');
        LOOP
            idx := idx + 1;
            prt_value(TO_CHAR(position(idx)), 12);
            prt_value(arg_name(idx), 12);
            prt_value(TO_CHAR(dty(idx)), 5);
            prt_value(TO_CHAR(p_mode(idx)), 5);
            DBMS_OUTPUT.NEW_LINE;
        END LOOP;
    EXCEPTION
       WHEN NO_DATA_FOUND THEN
          DBMS_OUTPUT.NEW_LINE;
          DBMS_OUTPUT.NEW_LINE;

    END desc_proc;
END describe_it;
```

Then the results list all the numeric codes for the PL/SQL datatypes:

```
Position  Name     Datatype_Code  Mode
1         PVC2       1              0
2         PVC        1              1
3         PSTR       1              2
4         PLONG      8              0
5         PROWID     11             0
6         PCHARA     96             0
7         PCHAR      96             0
8         PRAW       23             0
9         PLRAW      24             0
10        PBININT    3              0
11        PPLSINT    3              0
12        PBOOL      252            0
13        PNAT       3              0
14        PPOS       3              0
15        PPOSN      3              0
16        PNATN      3              0
```

```
17        PNUM      2              0
18        PINTGR    2              0
19        PINT      2              0
20        PSMALL    2              0
21        PDEC      2              0
22        PREAL     2              0
23        PFLOAT    2              0
24        PNUMER    2              0
25        PDP       2              0
26        PDATE     12             0
27        PMLS      106            0
```

Summary of DBMS_DESCRIBE Subprograms

The `DBMS_DESCRIBE` package includes the `DESCRIBE_PROCEDURE` procedure.

Table 75-2 DBMS_DESCRIBE Package Subprograms

Subprogram	Description
DESCRIBE_PROCEDURE Procedure	Provides a brief description of a PL/SQL stored procedure

DESCRIBE_PROCEDURE Procedure

The procedure `DESCRIBE_PROCEDURE` provides a brief description of a PL/SQL stored procedure.

It takes the name of a stored procedure and returns information about each parameter of that procedure.

Syntax

```
DBMS_DESCRIBE.DESCRIBE_PROCEDURE(
   object_name                    IN  VARCHAR2,
   reserved1                      IN  VARCHAR2,
   reserved2                      IN  VARCHAR2,
   overload                       OUT NUMBER_TABLE,
   position                       OUT NUMBER_TABLE,
   level                          OUT NUMBER_TABLE,
   argument_name                  OUT VARCHAR2_TABLE,
   datatype                       OUT NUMBER_TABLE,
   default_value                  OUT NUMBER_TABLE,
   in_out                         OUT NUMBER_TABLE,
   length                         OUT NUMBER_TABLE,
   precision                      OUT NUMBER_TABLE,
   scale                          OUT NUMBER_TABLE,
   radix                          OUT NUMBER_TABLE,
   spare                          OUT NUMBER_TABLE
   include_string_constraints     OUT BOOLEAN DEFAULT FALSE);
```

Parameters

Table 75-3 DBMS_DESCRIBE.DESCRIBE_PROCEDURE Parameters

Parameter	Description
object_name	Name of the procedure being described. The syntax for this parameter follows the rules used for identifiers in SQL. The name can be a synonym. This parameter is required and may not be null. The total length of the name cannot exceed 197 bytes. An incorrectly specified OBJECT_NAME can result in one of the following exceptions: ORA-20000 - A package was specified. You can only specify a stored procedure, stored function, packaged procedure, or packaged function. ORA-20001 - The procedure or function that you specified does not exist within the given package. ORA-20002 - The object that you specified is a remote object. This procedure cannot currently describe remote objects. ORA-20003 - The object that you specified is invalid and cannot be described. ORA-20004 - The object was specified with a syntax error.
reserved1 reserved2	Reserved for future use -- must be set to NULL or the empty string.
overload	A unique number assigned to the procedure's signature. If a procedure is overloaded, then this field holds a different value for each version of the procedure.
position	Position of the argument in the parameter list. Position 0 returns the values for the return type of a function.
level	If the argument is a composite type, such as record, then this parameter returns the level of the datatype. See the *Oracle Call Interface Developer's Guide* for a description of the ODESSP call for an example.
argument_name	Name of the argument associated with the procedure that you are describing.

Table 75-3 (Cont.) DBMS_DESCRIBE.DESCRIBE_PROCEDURE Parameters

Parameter	Description
datatype	Oracle datatype of the argument being described. The datatypes and their numeric type codes are: 0 placeholder for procedures with no arguments 1 VARCHAR, VARCHAR, STRING 2 NUMBER, INTEGER, SMALLINT, REAL, FLOAT, DECIMAL 3 BINARY_INTEGER, PLS_INTEGER, POSITIVE, NATURAL 8 LONG 11 ROWID 12 DATE 23 RAW 24 LONG RAW 58 OPAQUE TYPE 96 CHAR (ANSI FIXED CHAR), CHARACTER 106 MLSLABEL 121 OBJECT 122 NESTED TABLE 123 VARRAY 178 TIME 179 TIME WITH TIME ZONE 180 TIMESTAMP 181 TIMESTAMP WITH TIME ZONE 231 TIMESTAMP WITH LOCAL TIME ZONE 250 PL/SQL RECORD 251 PL/SQL TABLE 252 PL/SQL BOOLEAN
default_value	1 if the argument being described has a default value; otherwise, the value is 0.
in_out	Describes the mode of the parameter: 0 IN 1 OUT 2 IN OUT
length	For %rowtype formal arguments, the length constraint is returned, otherwise 0 is returned.If the include_string_constraints parameter is set to TRUE, the argument's formal length constraint is passed back if it is of the appropriate type. Those are the string types: 1;8;23;24;96
precision	If the argument being described is of datatype 2 (NUMBER), then this parameter is the precision of that number.
scale	If the argument being described is of datatype 2 (NUMBER), then this parameter is the scale of that number.
radix	If the argument being described is of datatype 2 (NUMBER), then this parameter is the radix of that number.
spare	Reserved for future functionality.
include_string_constraints	The default is FALSE. If the parameter is set to TRUE, the arguments' formal type constraints is passed back if it is of the appropriate type.Those are the string types: 1;8;23;24;96

Return Values

All values from DESCRIBE_PROCEDURE are returned in its OUT parameters. The datatypes for these are PL/SQL tables, to accommodate a variable number of parameters.