31
DBMS_ASSERT

The DBMS_ASSERT package provides an interface to validate properties of the input value.

> **See Also:**
>
> *Oracle Database PL/SQL Language Reference* for more information about "Avoiding SQL Injection in PL/SQL"

This chapter contains the following topics:

- Operational Notes
- Summary of DBMS_ASSERT Subprograms

DBMS_ASSERT Operational Notes

If the condition which determines the property asserted in a function is not met then a value error is raised. Otherwise the input value is returned through the return value. Most functions return the value unchanged, however, several functions modify the value.

Summary of DBMS_ASSERT Subprograms

This section describes the subprograms of the DBMS_ASSERT package.

Table 31-1 DBMS_ASSERT Package Subprograms

Subprogram	Description
ENQUOTE_LITERAL Function	Enquotes a string literal
ENQUOTE_NAME Function	Ensures that a string is enclosed by quotation marks, then checks that the result is a valid SQL identifier.
NOOP Functions	Returns the value without any checking
QUALIFIED_SQL_NAME Function	Verifies that the input string is a qualified SQL name
SCHEMA_NAME Function	Verifies that the input string is an existing schema name
SIMPLE_SQL_NAME Function	Verifies that the input string is a simple SQL name
SQL_OBJECT_NAME Function	Verifies that the input parameter string is a qualified SQL identifier of an existing SQL object

ENQUOTE_LITERAL Function

This function adds leading and trailing single quotes to a string literal.

Syntax

```
DBMS_ASSERT.ENQUOTE_LITERAL (
   str            VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 31-2 *ENQUOTE_LITERAL Function Parameters*

Parameter	Description
`str`	String to enquote

Usage Notes

- Verify that all single quotes except leading and trailing characters are paired with adjacent single quotes.
- No additional quotes are added if the name was already in quotes.

ENQUOTE_NAME Function

This function encloses the provided string in double quotes (quotation marks). No additional quotes are added if the string was already in quotes (quotation marks). The quoted string is then checked to see if it is a valid (quoted) simple SQL name.

For more information on Database object names and qualifiers, see Oracle Database SQL Language Reference.

Syntax

```
DBMS_ASSERT.ENQUOTE_NAME (
   str            VARCHAR2,
   capitalize     BOOLEAN DEFAULT TRUE)
RETURN VARCHAR2;
```

Parameters

Table 31-3 *ENQUOTE_NAME Function Parameters*

Parameter	Description
`str`	String to enquote
`capitalize`	If `TRUE` or defaulted, alphabetic characters of `str` which was not in quotes are translated to upper case

Usage Notes

- No additional quotes are added if the name was already in quotes.
- Verify that all other double quotes in the string are adjacent pairs of double quotes.

Examples

```
-- This procedure creates a single column table in the createOneColumnTable's schema.
create or replace procedure createOneColumnTable(proposedTableNamevarchar2) is
BEGIN
  IF
    (proposedTableName is NULL)
  THEN
    raise value_error;
  END IF;
  -- The use of ENQUOTE_NAME ensures that the table will be created in the
  -- definer's schema and not in some other schema even if the definer has
  -- privileges to create tables in other schemas.
  EXECUTE IMMEDIATE 'create table ' ||
  DBMS_ASSERT.ENQUOTE_NAME(proposedTableName) || ' (c1 number)';
  EXCEPTION
    WHEN
      others
    THEN
      dbms_output.put_line('Table creation failed due to: ' || SQLERRM);
END;
/

-- Examples of ENQUOTE_NAME showing input/output relationships
BEGIN
  -- 'eMp' becomes '"EMP"' since it is unquoted
  dbms_output.put_line(DBMS_ASSERT.ENQUOTE_NAME('eMp'));
END;
/

BEGIN
  -- For quoted strings, the case is preserved
  dbms_output.put_line(DBMS_ASSERT.ENQUOTE_NAME('"EmP"'));
END;
/

-- Invalid identifier example
BEGIN
  dbms_output.put_line(DBMS_ASSERT.ENQUOTE_NAME('SCOTT."EMP"'));
END;
/

-- CHR(0) examples
-- The following examples illustrates that CHR(0), the NULL character, cannot appear
-- in the string; such a string poses a SQL injection risk.
BEGIN
  dbms_output.put_line(DBMS_ASSERT.ENQUOTE_NAME('BAD' || CHR(0) || 'IDENTIFIER'));
END;
/

BEGIN
   dbms_output.put_line(DBMS_ASSERT.ENQUOTE_NAME('"SCOTT' || CHR(0) || '.EMP"'));
END;
/

-- Oracle allows a period (.) to be a part of a quoted string
BEGIN
  dbms_output.put_line(DBMS_ASSERT.ENQUOTE_NAME('"SCOTT.EMP"'));
END;
/
```

```
-- The single quotation mark ('), as opposed to a double quotation mark, can appear in
the string
-- Note: In Oracle, a single quotation mark is specified in a literal using two single
-- quotes. The first quotation mark escapes the second quotation mark in the same way
that
-- backslash (\) in POSIX is an escape character.
BEGIN
   dbms_output.put_line(DBMS_ASSERT.ENQUOTE_NAME('"O''LEARY"'));
END;
/
```

NOOP Functions

This function returns the value without any checking.

Syntax

```
DBMS_ASSERT.NOOP (
   str      VARCHAR2 CHARACTER SET ANY_CS)
 RETURN     VARCHAR2 CHARACTER SET str%CHARSET;

DBMS_ASSERT.NOOP (
   str      CLOB CHARACTER SET ANY_CS)
 RETURN     CLOB CHARACTER SET str%CHARSET;
```

Parameters

Table 31-4 *NOOP Function Parameters*

Parameter	Description
str	Input value

QUALIFIED_SQL_NAME Function

This function verifies that the input string is a qualified SQL name.

Syntax

```
DBMS_ASSERT.QUALIFIED_SQL_NAME (
   str      VARCHAR2 CHARACTER SET ANY_CS)
 RETURN     VARCHAR2 CHARACTER SET str%CHARSET;
```

Parameters

Table 31-5 *QUALIFIED_SQL_NAME Function Parameters*

Parameter	Description
str	Input value

Exceptions

ORA44004: string is not a qualified SQL name

Usage Notes

A qualified SQL name <qualified name> can be expressed by the following grammar:

```
<local qualified name> ::= <simple name> {'.' <simple name>}
<database link name> ::= <local qualified name> ['@' <connection string>]
<connection string> ::= <simple name>
<qualified name> ::= <local qualified name> ['@' <database link name>]
```

SCHEMA_NAME Function

This function verifies that the input string is an existing schema name.

Syntax

```
DBMS_ASSERT.SCHEMA_NAME (
   str      VARCHAR2 CHARACTER SET ANY_CS)
 RETURN     VARCHAR2 CHARACTER SET str%CHARSET;
```

Parameters

Table 31-6 *SCHEMA_NAME Function Parameters*

Parameter	Description
str	Input value

Exceptions

ORA44001: Invalid schema name

Usage Notes

By definition, a schema name need not be just a simple SQL name. For example, "FIRST LAST" is a valid schema name. As a consequence, care must be taken to quote the output of schema name before concatenating it with SQL text.

SIMPLE_SQL_NAME Function

This function verifies that the input string is a simple SQL name.

Syntax

```
DBMS_ASSERT.SIMPLE_SQL_NAME (
   str      VARCHAR2 CHARACTER SET ANY_CS)
 RETURN     VARCHAR2 CHARACTER SET str%CHARSET;
```

Parameters

Table 31-7 *SIMPLE_SQL_NAME Function Parameters*

Parameter	Description
str	Input value

Exceptions

ORA44003: string is not a simple SQL name

Usage Notes

- The input value must be meet the following conditions:
  - The name must begin with an alphabetic character. It may contain alphanumeric characters as well as the characters _, $, and # in the second and subsequent character positions.
  - Quoted SQL names are also allowed.
  - Quoted names must be enclosed in double quotes.
  - Quoted names allow any characters between the quotes.
  - Quotes inside the name are represented by two quote characters in a row, for example, "a name with "" inside" is a valid quoted name.
  - The input parameter may have any number of leading and/or trailing white space characters.
- The length of the name is not checked.

SQL_OBJECT_NAME Function

This function verifies that the input parameter string is a qualified SQL identifier of an existing SQL object.

Syntax

```
DBMS_ASSERT.SQL_OBJECT_NAME (
   str      VARCHAR2 CHARACTER SET ANY_CS)
 RETURN     VARCHAR2 CHARACTER SET str%CHARSET;
```

Parameters

Table 31-8 *SQL_OBJECT_NAME Function Parameters*

Parameter	Description
`str`	Input value

Exceptions

`ORA44002`: Invalid object name

Usage Notes

The use of synonyms requires that the base object exists.