71
DBMS_DDL

This package provides access to some SQL data definition language (DDL) statements from stored procedures. It also provides special administration operations that are not available as Data Definition Language statements (DDLs).

This chapter contains the following topics:

DBMS_DDL Deprecated Subprograms

Oracle recommends that you do not use deprecated subprograms in new applications. Support for deprecated features is for backward compatibility only

The following subprograms are deprecated with release Oracle Database 10*g*:

DBMS_DDL Security Model

This package runs with the privileges of the calling user, rather than the package owner `SYS`.

DBMS_DDL Operational Notes

The `ALTER_COMPILE` procedure commits the current transaction, performs the operation, and then commits again.

Summary of DBMS_DDL Subprograms

This table lists the `DDBMS_DDL` subprograms in alphabetical order and briefly describes them

Table 71-1 ***DBMS_DDL Package Subprograms***

Subprogram	Description
ALTER_COMPILE Procedure	Compiles the PL/SQL object
ALTER_TABLE_NOT_REFERENCEABLE Procedure	Reorganizes object tables
ALTER_TABLE_REFERENCEABLE Procedure	Reorganizes object tables

Table 71-1 (Cont.) *DBMS_DDL Package Subprograms*

Subprogram	Description
CREATE_WRAPPED Procedures	Takes as input a single `CREATE OR REPLACE` statement that specifies creation of a PL/SQL package specification, package body, function, procedure, type specification or type body, generates a `CREATE OR REPLACE` statement with the PL/SQL source text obfuscated and executes the generated statement
IS_TRIGGER_FIRE_ONCE Function	Returns `TRUE` if the specified DML or DDL trigger is set to fire once. Otherwise, returns `FALSE`
SET_TRIGGER_FIRING_PROPERTY Procedures	Sets the specified DML or DDL trigger's firing property
WRAP Functions	Takes as input a `CREATE OR REPLACE` statement that specifies creation of a PL/SQL package specification, package body, function, procedure, type specification or type body and returns a `CREATE OR REPLACE` statement where the text of the PL/SQL unit has been obfuscated

ALTER_COMPILE Procedure

This procedure is equivalent to the SQL statement: `ALTER PROCEDURE|FUNCTION|PACKAGE [<schema>.] <name> COMPILE [BODY]`

> **Note:**
>
> This procedure is deprecated in Oracle Database 10*g* Release 2 (10.2) While the procedure remains available in the package for reasons of backward compatibility, Oracle recommends using the DDL equivalent in a dynamic SQL statement.

Syntax

```
DBMS_DDL.ALTER_COMPILE (
   type             VARCHAR2,
   schema           VARCHAR2,
   name             VARCHAR2
   reuse_settings   BOOLEAN := FALSE);
```

Parameters

Table 71-2 ALTER_COMPILE Procedure Parameters

Parameter	Description
`type`	Must be either `PROCEDURE`, `FUNCTION`, `PACKAGE`, `PACKAGE BODY` or `TRIGGER`
`schema`	Schema name If `NULL`, then use current schema (case-sensitive)
`name`	Name of the object (case-sensitive)
`reuse_settings`	Indicates whether the session settings in the objects should be reused, or whether the current session settings should be adopted instead

Exceptions

Table 71-3 ALTER_COMPILE Procedure Exceptions

Exception	Description
ORA-20000:	Insufficient privileges or object does not exist
ORA-20001:	Remote object, cannot compile
ORA-20002:	Bad value for object type: should be either PACKAGE, PACKAGE BODY, PROCEDURE, FUNCTION, or TRIGGER

ALTER_TABLE_NOT_REFERENCEABLE Procedure

This procedure alters the given object table table_schema.table_name so it becomes not the default referenceable table for the schema affected_schema.

This is equivalent to SQL:

```
ALTER TABLE [<table_schema>.]<table_name> NOT REFERENCEABLE FOR <affected_schema>
```

which is currently not supported or available as a DDL statement.

Syntax

```
DBMS_DDL.ALTER_TABLE_NOT_REFERENCEABLE (
   table_name        IN           VARCHAR2,
   table_schema      IN  DEFAULT  NULL,
   affected_schema   IN  DEFAULT  NULL);
```

Parameters

Table 71-4 ALTER_TABLE_NOT_REFERENCEABLE Procedure Parameters

Parameter	Description
table_name	Name of the table to be altered. Cannot be a synonym. Must not be NULL. Case sensitive.
table_schema	Name of the schema owning the table to be altered. If NULL then the current schema is used. Case sensitive.
affected_schema	Name of the schema affected by this alteration. If NULL then the current schema is used. Case sensitive.

Usage Notes

This procedure simply reverts for the affected schema to the default table referenceable for PUBLIC; that is., it simply undoes the previous ALTER_TABLE_REFERENCEABLE call for this specific schema. The affected schema must a particular schema (cannot be PUBLIC).

The user that executes this procedure must own the table (that is, the schema is the same as the user), and the affected schema must be the same as the user.

If the user executing this procedure has ALTER ANY TABLE and SELECT ANY TABLE and DROP ANY TABLE privileges, the user doesn't have to own the table and the affected schema can be any valid schema.

ALTER_TABLE_REFERENCEABLE Procedure

This procedure alters the given object table `table_schema.table_name` so it becomes the referenceable table for the given schema `affected_schema`.

This is equivalent to SQL:

```
ALTER TABLE [<table_schema>.]<table_name>  REFERENCEABLE FOR <affected_schema>
```

which is currently not supported or available as a DDL statement.

Syntax

```
DBMS_DDL.ALTER_TABLE_REFERENCEABLE
   table_name       IN  VARCHAR2,
   table_schema     IN  DEFAULT  NULL,
   affected_schema  IN  DEFAULT  NULL);
```

Parameters

Table 71-5 ALTER_TABLE_REFERENCEABLE Procedure Parameters

Parameter	Description
`table_name`	Name of the table to be altered. Cannot be a synonym. Must not be `NULL`. Case sensitive.
`table_schema`	Name of the schema owning the table to be altered. If `NULL` then the current schema is used. Case sensitive.
`affected_schema`	Name of the schema affected by this alteration. If NULL then the current schema is used. Case sensitive.

Usage Notes

When you create an object table, it automatically becomes referenceable, unless you use the `OID AS` clause when creating the table. The `OID AS` clause makes it possible for you to create an object table and to assign to the new table the same EOID as another object table of the same type. After you create a new table using the `OID AS` clause, you end up with two object table with the same `EOID`; the new table is not referenceable, the original one is. All references that used to point to the objects in the original table still reference the same objects in the same original table.

If you execute this procedure on the new table, it makes the new table the referenceable table replacing the original one; thus, those references now point to the objects in the new table instead of the original table.

CREATE_WRAPPED Procedures

The procedure takes as input a single `CREATE OR REPLACE` statement that specifies creation of a PL/SQL package specification, package body, function, procedure, type specification or type body. It then generates a `CREATE OR REPLACE` statement with the PL/SQL source text

obfuscated and executes the generated statement. In effect, this procedure bundles together the operations of wrapping the text and creating the PL/SQL unit.

> **See Also:**
>
> WRAP Functions

This procedure has 3 overloads. Each of the three functions provides better performance than using a combination of individual WRAP Functions and `DBMS_SQL.PARSE` (or `EXECUTE IMMEDIATE`) calls. The different functionality of each form of syntax is presented with the definition.

Syntax

Is a shortcut for `EXECUTE IMMEDIATE SYS.DBMS_DDL.WRAP`(*ddl*):

```
DBMS_DDL.CREATE_WRAPPED (
   ddl      VARCHAR2);
```

Is a shortcut for `DBMS_SQL.PARSE`(`cursor`, `SYS.DBMS_DDL.WRAP` (`input`, `lb`, `ub`)):

```
DBMS_DDL.CREATE_WRAPPED(
   ddl      DBMS_SQL.VARCHAR2A,
   lb       PLS_INTEGER,
   ub       PLS_INTEGER);
```

Is a shortcut for `DBMS_SQL.PARSE`(`cursor`, `SYS.DBMS_DDL.WRAP` (`input`, `lb`, `ub`)):

```
DBMS_DDL.CREATE_WRAPPED(
   ddl      DBMS_SQL.VARCHAR2S,
   lb       PLS_INTEGER,
   ub       PLS_INTEGER);
```

Parameters

Table 71-6 CREATE_WRAPPED Procedure Parameters

Parameter	Description
`ddl`	A `CREATE OR REPLACE` statement that specifies creation of a PL/SQL package specification, package body, function, procedure, type specification or type body
`lb`	Lower bound for indices in the string table that specify the `CREATE OR REPLACE` statement
`ub`	Upper bound for indices in the string table that specify the `CREATE OR REPLACE` statement.

Usage Notes

- The `CREATE OR REPLACE` statement is executed with the privileges of the user invoking `DBMS_DDL.CREATE_WRAPPED`.
- Any PL/SQL code that attempts to call these interfaces should use the fully qualified package name `SYS.DBMS_DDL` to avoid the possibility that the name `DBMS_DDL` is captured by a locally-defined unit or by redefining the `DBMS_DDL` public synonym.

- Each invocation of any accepts only a single PL/SQL unit. By contrast, the PL/SQL `wrap` utility accepts a entire SQL*Plus file and obfuscates the PL/SQL units within the file leaving all other text as-is. These interfaces are intended to be used in conjunction with or as a replacement for PL/SQL's dynamic SQL interfaces (`EXECUTE IMMEDIATE` and `DBMS_SQL.PARSE`). Since these dynamic SQL interfaces only accept a single unit at a time (and do not understand the SQL*Plus "`/`" termination character), both the CREATE_WRAPPED Procedures and the WRAP Functions require input to be a single unit.

Exceptions

`ORA-24230`: If the input is not a `CREATE OR REPLACE` statement specifying a PL/SQL unit, exception `DBMS_DDL.MALFORMED_WRAP_INPUT` is raised.

Examples

```
DECLARE
    ddl VARCHAR2(32767);
BEGIN
    ddl := GENERATE_PACKAGE(...);
    SYS.DBMS_DDL.CREATE_WRAPPED(ddl); -- Instead of EXECUTE IMMEDIATE ddl
END;
```

IS_TRIGGER_FIRE_ONCE Function

This function returns `TRUE` if the specified DML or DDL trigger is set to fire once. Otherwise, it returns `FALSE`.

A fire once trigger fires in a user session but does not fire in the following cases:

- For changes made by a Streams apply process
- For changes made by executing one or more Streams apply errors using the `EXECUTE_ERROR` or `EXECUTE_ALL_ERRORS` procedure in the `DBMS_APPLY_ADM` package
- For changes made by a Logical Standby apply process

> **Note:**
>
> Only DML and DDL triggers can be fire once. All other types of triggers always fire.

> **See Also:**
>
> "SET_TRIGGER_FIRING_PROPERTY Procedures"

Syntax

```
DBMS_DDL.IS_TRIGGER_FIRE_ONCE
   trig_owner         IN  VARCHAR2,
   trig_name          IN  VARCHAR2)
 RETURN BOOLEAN;
```

Parameters

Table 71-7 IS_TRIGGER_FIRE_ONCE Function Parameters

Parameter	Description
trig_owner	Schema of trigger
trig_name	Name of trigger

SET_TRIGGER_FIRING_PROPERTY Procedures

This procedure sets the specified DML or DDL trigger's firing property whether or not the property is set for the trigger.

Use this procedure to control a DML or DDL trigger's firing property for changes:

- Applied by a Streams apply process
- Made by executing one or more Streams apply errors using the EXECUTE_ERROR or EXECUTE_ALL_ERRORS procedure in the DBMS_APPLY_ADM package.
- Applied by a Logical Standby apply process

Syntax

```
DBMS_DDL.SET_TRIGGER_FIRING_PROPERTY (
   trig_owner         IN  VARCHAR2,
   trig_name          IN  VARCHAR2,
   fire_once          IN  BOOLEAN);

DBMS_DDL.SET_TRIGGER_FIRING_PROPERTY (
   trig_owner         IN  VARCHAR2,
   trig_name          IN  VARCHAR2,
   property           IN INTEGER,
   setting            IN BOOLEAN);
```

Parameters

Table 71-8 SET_TRIGGER_FIRING_PROPERTY Procedure Parameters

Parameter	Description
trig_owner	Schema of the trigger to set
trig_name	Name of the trigger to set
fire_once	• If TRUE, the trigger is set to fire once. By default, the fire_once parameter is set to TRUE for DML and DDL triggers. • If FALSE, the trigger is set to always fire unless apply_server_only property is set to TRUE, which overrides fire_once property setting.
property	• DBMS_DDL.fire_once to set the fire_once property of the trigger • DBMS_DDL.apply_server_only to indicate whether trigger fires only in the context of SQL apply processes maintaining a logical standby database or Streams apply processes
setting	Value of property being set

Usage Notes

DML triggers created on a table have their fire-once property set to `TRUE`. In this case, the triggers only fire when the table is modified by an user process, and they are automatically disabled inside Oracle processes maintaining either a logical standby database (SQL Apply) or Oracle processes doing replication (Streams Apply) processes, and thus do not fire when a SQL Apply or a Streams Apply process modifies the table. There are two ways for a user to fire a trigger as a result of SQL Apply or a Streams Apply process making a change to a maintained table: (a) setting the fire-once property of a trigger to `FALSE`, which allows it fire both in the context of a user process or a SQL or Streams Apply process, or (b) by setting the apply-server-only property to `TRUE` and thus making the trigger fire only in the context of a SQL Apply or a Streams Apply process and not in the context of a user process.

- `FIRE_ONCE=TRUE`, `APPLY_SERVER_ONLY=FALSE`

  This is the default property setting for a DML trigger. The trigger only fires when user process modifies the base table.

- `FIRE_ONCE=TRUE` or `FALSE`, `APPLY_SERVER_ONLY=TRUE`

  The trigger only fires when SQL Apply or Streams Apply process modifies the base table. The trigger does not fire when a user process modifies the base table.Thus the apply-server-only property overrides the fire-once property of a trigger.

> **Note:**
>
> - If you dequeue an error transaction from the error queue and execute it without using the `DBMS_APPLY_ADM` package, then relevant changes resulting from this execution cause a trigger to fire, regardless of the trigger firing property.
> - Only DML and DDL triggers can be fire once. All other types of triggers always fire.

WRAP Functions

This function takes as input a single `CREATE OR REPLACE` statement that specifies creation of a PL/SQL package specification, package body, function, procedure, type specification or type body and returns a `CREATE OR REPLACE` statement where the text of the PL/SQL unit has been obfuscated.

The function has 3 overloads to allow for the different ways in which DDL statements can be generated dynamically and presented to `DBMS_SQL` or `EXECUTE IMMEDIATE`. The different functionality of each form of syntax is presented with the definition.

> **See Also:**
>
> CREATE_WRAPPED Procedures

Syntax

Provides basic functionality:

```
DBMS_DDL.WRAP(
   ddl      VARCHAR2)
  RETURN VARCHAR2;
```

Provides the same functionality as the first form, but allows for larger inputs. This function is intended to be used with the PARSE Procedures in the DBMS_SQL package and its argument list follows the convention of DBMS_SQL.PARSE:

```
DBMS_DDL.WRAP(
   ddl      DBMS_SQL.VARCHAR2S,
   lb       PLS_INTEGER,
   ub       PLS_INTEGER)
  RETURN DBMS_SQL.VARCHAR2S;
```

Provides the same functionality as the second form and is provided for compatibility with multiple forms of the PARSE Procedures in the DBMS_SQL package:

```
DBMS_DDL.WRAP(
   ddl      DBMS_SQL.VARCHAR2A,
   lb       PLS_INTEGER,
   ub       PLS_INTEGER)
  RETURN DBMS_SQL.VARCHAR2A;
```

Parameters

Table 71-9 WRAP Function Parameters

Parameter	Description
ddl	A CREATE OR REPLACE statement that specifies creation of a PL/SQL package specification, package body, function, procedure, type specification or type body
lb	Lower bound for indices in the string table that specify the CREATE OR REPLACE statement
ub	Upper bound for indices in the string table that specify the CREATE OR REPLACE statement.

Return Values

A CREATE OR REPLACE statement with the text obfuscated. In the case of the second and third form, the return value is a table of strings that need to be concatenated in order to construct the CREATE OR REPLACE string containing obfuscated source text.

Usage Notes

- Any PL/SQL code that attempts to call these interfaces should use the fully qualified package name SYS.DBMS_DDL to avoid the possibility that the name DBMS_DDL is captured by a locally-defined unit or by redefining the DBMS_DDL public synonym.
- Each invocation of any accepts only a single PL/SQL unit. By contrast, the PL/SQL wrap utility accepts a full SQL file and obfuscates the PL/SQL units within the file leaving all other text as-is. These interfaces are intended to be used in conjunction with or as a replacement for PL/SQL's dynamic SQL interfaces (EXECUTE IMMEDIATE and DBMS_SQL.PARSE). Since these dynamic SQL interfaces only accept a single unit at a time (and do not understand the SQL*Plus "/" termination character), both the CREATE_WRAPPED Procedures and the WRAP Functions require input to be a single unit.

Exceptions

ORA-24230: If the input is not a CREATE OR REPLACE statement specifying a PL/SQL unit, exception DBMS_DDL.MALFORMED_WRAP_INPUT is raised.

Examples

```
DECLARE
   ddl VARCHAR2(32767);
BEGIN
   ddl := GENERATE_PACKAGE(...);
EXECUTE IMMEDIATE SYS.DBMS_DDL.WRAP(ddl); -- Instead of EXECUTE IMMEDIATE ddl
END;
```