300 UTL_REF

The `UTL_REF` package provides PL/SQL procedures to support reference-based operations. Unlike SQL, `UTL_REF` procedures enable you to write generic type methods without knowing the object table name.

This chapter contains the following topics:

UTL_REF Overview

`UTL_REF` procedures enable you to write generic type methods without knowing the object table name.

Oracle supports user-defined composite type or object type. Any instance of an object type is called an object. An object type can be used as the type of a column or as the type of a table.

In an object table, each row of the table stores an object. You can uniquely identify an object in an object table with an object identifier.

A reference is a persistent pointer to an object, and each reference can contain an object identifier. The reference can be an attribute of an object type, or it can be stored in a column of a table. Given a reference, an object can be retrieved.

UTL_REF Security Model

The procedural option is needed to use this package. This package must be created under `SYS` (`CONNECT /AS SYSDBA`). Operations provided by this package are performed under the current calling user, not under the package owner `SYS`.

You can use the `UTL_REF` package from stored PL/SQL procedures/packages on the server, as well as from client/side PL/SQL code.

When invoked from PL/SQL procedures/packages on the server, `UTL_REF` verifies that the invoker has the appropriate privileges to access the object pointed to by the `REF`.

> **Note:**
>
> This is in contrast to PL/SQL packages/procedures on the server which operate with definer's privileges, where the package owner must have the appropriate privileges to perform the desired operations.

Thus, if UTL_REF is defined under user SYS, and user A invokes UTL_REF.SELECT to select an object from a reference, then user A (the invoker) requires the privileges to check.

When invoked from client-side PL/SQL code, UTL_REF operates with the privileges of the client session under which the PL/SQL execution is being done.

UTL_REF Types

An object type is a composite datatype defined by the user or supplied as a library type.

You can create the object type employee_type using the following syntax:

```
CREATE TYPE employee_type AS OBJECT (
   name     VARCHAR2(20),
   id       NUMBER,

member function GET_ID
    (name VARCHAR2)
   RETURN MEMBER);
```

The object type employee_type is a user-defined type that contains two attributes, name and id, and a member function, GET_ID().

You can create an object table using the following SQL syntax:

```
CREATE TABLE employee_table OF employee_type;
```

UTL_REF Exceptions

Exceptions can be returned during execution of UTL_REF functions for various reasons.

For example, the following scenarios would result in exceptions:

- The object selected does not exist. This could be because either:
  1. The object has been deleted, or the given reference is dangling (invalid).
  2. The object table was dropped or does not exist.
- The object cannot be modified or locked in a serializable transaction. The object was modified by another transaction after the serializable transaction started.
- You do not have the privilege to select or modify the object. The caller of the UTL_REF subprogram must have the proper privilege on the object that is being selected or modified.

Table 300-1 ***UTL_REF Exceptions***

Exceptions	Description
`errnum == 942`	Insufficient privileges.
`errnum == 1031`	Insufficient privileges.
`errnum == 8177`	Unable to serialize, if in a serializable transaction.
`errnum == 60`	Deadlock detected.
`errnum == 1403`	No data found (if the REF is NULL, and so on.).

The UTL_REF package does not define any named exceptions. You may define exception handling blocks to catch specific exceptions and to handle them appropriately.

Summary of UTL_REF Subprograms

This table lists the `UTL_REF` subprograms and briefly describes them.

Table 300-2 UTL_REF Package Subprograms

Subprogram	Description
DELETE_OBJECT Procedure	Deletes an object given a reference
LOCK_OBJECT Procedure	Locks an object given a reference
SELECT_OBJECT Procedure	Selects an object given a reference
UPDATE_OBJECT Procedure	Updates an object given a reference

DELETE_OBJECT Procedure

This procedure deletes an object given a reference.

The semantic of this subprogram is similar to the following SQL statement:

```
DELETE FROM object_table
WHERE REF(t) = reference;
```

Unlike the preceding SQL statement, this subprogram does not require you to specify the object table name where the object resides.

Syntax

```
UTL_REF.DELETE_OBJECT (
   reference IN REF "<typename>");
```

Parameters

Table 300-3 DELETE_OBJECT Procedure Parameters

Parameter	Description
`reference`	Reference of the object to delete.

Exceptions

May be raised.

Examples

The following example illustrates usage of the `UTL_REF` package to implement this scenario: if an employee of a company changes their address, their manager should be notified.

... declarations of `Address_t` and others...

```
CREATE OR REPLACE TYPE Person_t (
   name    VARCHAR2(64),
   gender  CHAR(1),
   address Address_t,
   MEMBER PROCEDURE setAddress(addr IN Address_t)
);
```

```
CREATE OR REPLACE TYPE BODY Person_t (
   MEMBER PROCEDURE setAddress(addr IN Address_t) IS
   BEGIN
      address := addr;
   END;
);

CREATE OR REPLACE TYPE Employee_t (
```

Under `Person_t`: Simulate implementation of inheritance using a `REF` to `Person_t` and delegation of `setAddress` to it.

```
   thePerson  REF Person_t,
   empno      NUMBER(5),
   deptREF    Department_t,
   mgrREF     Employee_t,
   reminders  StringArray_t,
   MEMBER PROCEDURE setAddress(addr IN Address_t),
   MEMBER procedure addReminder(reminder VARCHAR2);
);

CREATE TYPE BODY Employee_t (
   MEMBER PROCEDURE setAddress(addr IN Address_t) IS
      myMgr Employee_t;
      meAsPerson Person_t;
   BEGIN
```

Update the address by delegating the responsibility to `thePerson`. Lock the Person object from the reference, and also select it:

```
      UTL_REF.LOCK_OBJECT(thePerson, meAsPerson);
      meAsPerson.setAddress(addr);
```

Delegate to `thePerson`:

```
        UTL_REF.UPDATE_OBJECT(thePerson, meAsPerson);
        if mgr is NOT NULL THEN
```

Give the manager a reminder:

```
         UTL_REF.LOCK_OBJECT(mgr);
         UTL_REF.SELECT_OBJECT(mgr, myMgr);
         myMgr.addReminder
         ('Update address in the employee directory for' ||
         thePerson.name || ', new address: ' || addr.asString);
         UTL_REF.UPDATE_OBJECT(mgr, myMgr);
      END IF;
   EXCEPTION
      WHEN OTHERS THEN
      errnum := SQLCODE;
      errmsg := SUBSTR(SQLERRM, 1, 200);
```

LOCK_OBJECT Procedure

This procedure locks an object given a reference. In addition, this procedure lets the program select the locked object.

The semantic of this subprogram is similar to the following SQL statement:

```
SELECT VALUE(t)
  INTO object
  FROM object_table t
  WHERE REF(t) = reference
  FOR UPDATE;
```

Unlike the preceding SQL statement, this subprogram does not require you to specify the object table name where the object resides. It is not necessary to lock an object before updating/deleting it.

Syntax

```
UTL_REF.LOCK_OBJECT (
   reference IN REF "<typename>");

UTL_REF.LOCK_OBJECT (
   reference IN REF "<typename>",
   object    IN OUT "<typename>");
```

Parameters

Table 300-4 LOCK_OBJECT Procedure Parameters

Parameter	Description
`reference`	Reference of the object to lock.
`object`	The PL/SQL variable that stores the locked object. This variable should be of the same object type as the locked object.

Exceptions

May be raised.

SELECT_OBJECT Procedure

This procedure selects an object given its reference. The selected object is retrieved from the database and its value is put into the PL/SQL variable 'object'.

The semantic of this subprogram is similar to the following SQL statement:

```
SELECT VALUE(t)
INTO object
FROM object_table t
WHERE REF(t) = reference;
```

Unlike the preceding SQL statement, this subprogram does not require you to specify the object table name where the object resides.

Syntax

```
UTL_REF.SELECT_OBJECT (
   reference IN REF "<typename>",
   object    IN OUT "<typename>");
```

Parameters

Table 300-5 SELECT_OBJECT Procedure Parameters

Parameter	Description
`reference`	Reference to the object to select or retrieve.

Table 300-5 (Cont.) SELECT_OBJECT Procedure Parameters

Parameter	Description
`object`	The PL/SQL variable that stores the selected object; this variable should be of the same object type as the referenced object.

Exceptions

May be raised.

UPDATE_OBJECT Procedure

This procedure updates an object given a reference. The referenced object is updated with the value contained in the PL/SQL variable 'object'.

The semantic of this subprogram is similar to the following SQL statement:

```
UPDATE object_table t
SET VALUE(t) = object
WHERE REF(t) = reference;
```

Unlike the preceding SQL statement, this subprogram does not require you to specify the object table name where the object resides.

Syntax

```
UTL_REF.UPDATE_OBJECT (
   reference IN REF "<typename>",
   object    IN     "<typename>");
```

Parameters

Table 300-6 UPDATE_OBJECT Procedure Parameters

Parameter	Description
`reference`	Reference of the object to update.
`object`	The PL/SQL variable that contains the new value of the object. This variable should be of the same object type as the object to update.

Exceptions

May be raised.