121
DBMS_LOCK

The `DBMS_LOCK` package provides an interface to Oracle Lock Management services.

You can request a lock of a specific mode, give it a unique name recognizable in another procedure in the same or another instance, change the lock mode, and release it.

This chapter contains the following topics:

See Also:

For more information, and an example of how to use the `DBMS_LOCK` package, *Oracle Database Development Guide*

DBMS_LOCK Overview

The `DBMS_LOCK` package has many beneficial uses.

These uses include the following:

- Providing exclusive access to a device, such as a terminal
- Providing application-level enforcement of read locks
- Detecting when a lock is released and cleanup after the application
- Synchronizing applications and enforcing sequential processing

DBMS_LOCK Security Model

There might be operating system-specific limits on the maximum number of total locks available. This *must* be considered when using locks or making this package available to other users. Consider granting the `EXECUTE` privilege only to specific users or roles.

A better alternative would be to create a cover package limiting the number of locks used and grant `EXECUTE` privilege to specific users. An example of a cover package is documented in the `DBMS_LOCK.SQL` package specification file. The abbreviations for these locks as they appear in Enterprise Manager monitors are in parentheses.

DBMS_LOCK Constants

The `DBMS_LOCK` package includes several constants to use when specifying parameter values.

These constants are shown in the following table.

Table 121-1 DBMS_LOCK Constants

Name	Alternate Name(s)	Type	Value	OEM Abbreviation	Description
`NL_MODE`	`NuL1`	`INTEGER`	`1`	-	-
`SS_MODE`	`Sub Shared`	`INTEGER`	`2`	`ULRS`	This can be used on an aggregate object to indicate that share locks are being acquired on subparts of the object.
`SX_MODE`	• `Sub eXclusive` • `Row Exclusive Mode`	`INTEGER`	`3`	`ULRX`	This can be used on an aggregate object to indicate that exclusive locks are being acquired on sub-parts of the object.
`S_MODE`	• `Shared` • `Row Exclusive Mode` • `Intended Exclusive`	`INTEGER`	`4`	`ULRSX`	-
`SSX_MODE`	• `Shared Sub eXclusive` • `Share Row Exclusive Mode`	`INTEGER`	`5`	-	This indicates that the entire aggregate object has a share lock, but some of the sub-parts may additionally have exclusive locks.
`X_MODE`	`Exclusive`	`INTEGER`	`6`	`ULX`	-

These are the various lock modes (nl -> "NuLl", ss -> "Sub Shared", sx -> "Sub eXclusive", s -> "Shared", ssx -> "Shared Sub eXclusive", x -> "eXclusive").

DBMS_LOCK. Rules and Limits

When another process holds "held", an attempt to get "get" succeeds or fails, based on the held mode and type of get.

The following table describes the results:

Table 121-2 Lock Compatibility

HELD MODE	GET NL	GET SS	GET SX	GET S	GET SSX	GET X
`NL`	`Success`	`Success`	`Success`	`Success`	`Success`	`Success`
`SS`	`Success`	`Success`	`Success`	`Success`	`Success`	`Fail`
`SX`	`Success`	`Success`	`Success`	`Fail`	`Fail`	`Fail`
`S`	`Success`	`Success`	`Fail`	`Success`	`Fail`	`Fail`
`SSX`	`Success`	`Success`	`Fail`	`Fail`	`Fail`	`Fail`
`X`	`Success`	`Fail`	`Fail`	`Fail`	`Fail`	`Fail`

```
maxwait  constant integer := 32767;
```

The constant `maxwait` waits forever.

DBMS_LOCK Operational Notes

User locks never conflict with Oracle locks because they are identified with the prefix "UL". You can view these locks using the Enterprise Manager lock monitor screen or the appropriate fixed views.

User locks are automatically released when a session terminates. The lock identifier is a number in the range of 0 to 1073741823.

Because a reserved user lock is the same as an Oracle lock, it has all the functionality of an Oracle lock, such as deadlock detection. Be certain that any user locks used in distributed transactions are released upon `COMMIT`, or an undetected deadlock may occur.

`DBMS_LOCK` is most efficient with a limit of a few hundred locks for each session. Oracle strongly recommends that you develop a standard convention for using these locks in order to avoid conflicts among procedures trying to use the same locks. For example, include your company name as part of your lock names.

Summary of DBMS_LOCK Subprograms

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

Table 121-3 ***DBMS_LOCK Package Subprograms***

Subprogram	Description
ALLOCATE_UNIQUE Procedure	Allocates a unique lock ID to a named lock
ALLOCATE_UNIQUE_AUTONOMOUS Procedure	Allocates a unique lock ID to a named lock
CONVERT Function	Converts a lock from one mode to another
RELEASE Function	Releases a lock
REQUEST Function	Requests a lock of a specific mode.

ALLOCATE_UNIQUE Procedure

This procedure allocates a unique lock identifier (in the range of 1073741824 to 1999999999) a specified lock name. Lock identifiers are used to enable applications to coordinate their use of locks. This is provided because it may be easier for applications to coordinate their use of locks based on lock names rather than lock numbers.

Syntax

```
DBMS_LOCK.ALLOCATE_UNIQUE (
   lockname         IN  VARCHAR2,
   lockhandle       OUT VARCHAR2,
   expiration_secs  IN  INTEGER   DEFAULT 864000);
```

Parameters

Table 121-4 ALLOCATE_UNIQUE Procedure Parameters

Parameter	Description
`lockname`	Name of the lock for which you want to generate a unique ID. Do not use lock names beginning with `ORA$`; these are reserved for products supplied by Oracle.
`lockhandle`	Returns the handle to the lock ID generated by `ALLOCATE_UNIQUE`. You can use this handle in subsequent calls to `REQUEST`, `CONVERT`, and `RELEASE`. A handle is returned instead of the actual lock ID to reduce the chance that a programming error accidentally creates an incorrect, but valid, lock ID. This provides better isolation between different applications that are using this package. `LOCKHANDLE` can be up to `VARCHAR2` (128). All sessions using a lock handle returned by `ALLOCATE_UNIQUE` with the same lock name are referring to the same lock. Therefore, do not pass lock handles from one session to another.
`expiration_secs`	Number of seconds to wait after the last `ALLOCATE_UNIQUE` has been performed on a specified lock, before permitting that lock to be deleted from the `DBMS_LOCK_ALLOCATED` table. The default waiting period is 10 days. You should not delete locks from this table. Subsequent calls to `ALLOCATE_UNIQUE` may delete expired locks to recover space.

Usage Notes

If you choose to identify locks by name, you can use `ALLOCATE_UNIQUE` to generate a unique lock identification number for these named locks.

The first session to call `ALLOCATE_UNIQUE` with a new lock name causes a unique lock ID to be generated and stored in the `dbms_lock_allocated` table. Subsequent calls (usually by other sessions) return the lock ID previously generated.

A lock name is associated with the returned lock ID for at least `expiration_secs` (defaults to 10 days) past the last call to `ALLOCATE_UNIQUE` with the specified lock name. After this time, the row in the `dbms_lock_allocated` table for this lock name may be deleted in order to recover space. `ALLOCATE_UNIQUE` performs a commit.

WARNING:

Named user locks may be less efficient, because Oracle uses SQL to determine the lock associated with a specified name.

Exceptions

`ORA-20000`, `ORU-10003`: Unable to find or insert lock <`lockname`> into catalog `dbms_lock_allocated`.

ALLOCATE_UNIQUE_AUTONOMOUS Procedure

This procedure allocates a unique lock identifier (in the range of 1073741824 to 1999999999) a specified lock name and is an autonomous version of the ALLOCATE_UNIQUE procedure. This procedure works exactly same as that of ALLOCATE_UNIQUE, except that the procedure will run as an autonomous transaction. Therefore the commits in ALLOCATE_UNIQUE_AUTONOMOUS procedure will not affect the calling procedure. The ALLOCATE_UNIQUE_AUTONOMOUS procedure is implemented in DB 12.1 and later releases.

Syntax

```
DBMS_LOCK.ALLOCATE_UNIQUE_AUTONOMOUS (
   lockname         IN  VARCHAR2,
   lockhandle       OUT VARCHAR2,
   expiration_secs  IN  INTEGER   DEFAULT 864000);
```

Parameters

Table 121-5 ALLOCATE_UNIQUE_AUTONOMOUS Procedure Parameters

Parameter	Description
lockname	Name of the lock for which you want to generate a unique ID. Do not use lock names beginning with ORA$; these are reserved for products supplied by Oracle.
lockhandle	Returns the handle to the lock ID generated by ALLOCATE_UNIQUE_AUTONOMOUS. You can use this handle in subsequent calls to REQUEST, CONVERT, and RELEASE. A handle is returned instead of the actual lock ID to reduce the chance that a programming error accidentally creates an incorrect, but valid, lock ID. This provides better isolation between different applications that are using this package. LOCKHANDLE can be up to VARCHAR2 (128). All sessions using a lock handle returned by ALLOCATE_UNIQUE_AUTONOMOUS with the same lock name are referring to the same lock. Therefore, do not pass lock handles from one session to another.
expiration_secs	Number of seconds to wait after the last ALLOCATE_UNIQUE_AUTONOMOUS has been performed on a specified lock, before permitting that lock to be deleted from the ALLOCATE_UNIQUE_AUTONOMOUS table. The default waiting period is 10 days. You should not delete locks from this table. Subsequent calls to ALLOCATE_UNIQUE_AUTONOMOUS may delete expired locks to recover space.

Usage Notes

If you choose to identify locks by name, you can use ALLOCATE_UNIQUE_AUTONOMOUS to generate a unique lock identification number for these named locks.

The first session to call ALLOCATE_UNIQUE_AUTONOMOUS with a new lock name causes a unique lock ID to be generated and stored in the dbms_lock_allocated table. Subsequent calls (usually by other sessions) return the lock ID previously generated.

A lock name is associated with the returned lock ID for at least `expiration_secs` (defaults to 10 days) past the last call to `ALLOCATE_UNIQUE_AUTONOMOUS` with the specified lock name. After this time, the row in the `dbms_lock_allocated` table for this lock name may be deleted in order to recover space. `ALLOCATE_UNIQUE_AUTONOMOUS` performs a commit.

WARNING:

Named user locks may be less efficient, because Oracle uses SQL to determine the lock associated with a specified name.

Exceptions

`ORA-20000`, `ORU-10003`: Unable to find or insert lock <`lockname`> into catalog `dbms_lock_allocated`.

CONVERT Function

This function converts a lock from one mode to another. `CONVERT` is an overloaded function that accepts either a user-defined lock identifier, or the lock handle returned by the `ALLOCATE_UNIQUE` procedure.

Syntax

```
DBMS_LOCK.CONVERT(
   id          IN INTEGER ||
   lockhandle  IN VARCHAR2,
   lockmode    IN INTEGER,
   timeout     IN NUMBER DEFAULT MAXWAIT)
  RETURN INTEGER;
```

Parameters

Table 121-6 CONVERT Function Parameters

Parameter	Description
`id` or `lockhandle`	User assigned lock identifier, from 0 to 1073741823, or the lock handle, returned by `ALLOCATE_UNIQUE`, of the lock mode you want to change
`lockmode`	New mode that you want to assign to the specified lock. For the available modes and their associated integer identifiers, see Constants.
`timeout`	Number of seconds to continue trying to change the lock mode. If the lock cannot be converted within this time period, then the call returns a value of 1 (timeout).

Return Values

Table 121-7 CONVERT Function Return Values

Return Value	Description
`0`	Success

Table 121-7 (Cont.) CONVERT Function Return Values

Return Value	Description
1	Timeout
2	Deadlock
3	Parameter error
4	Don't own lock specified by `id` or `lockhandle`
5	Illegal lock handle

RELEASE Function

This function explicitly releases a lock previously acquired using the `REQUEST` function.

Locks are automatically released at the end of a session. `RELEASE` is an overloaded function that accepts either a user-defined lock identifier, or the lock handle returned by the `ALLOCATE_UNIQUE` procedure.

Syntax

```
DBMS_LOCK.RELEASE (
   id          IN INTEGER)
  RETURN INTEGER;

DBMS_LOCK.RELEASE (
   lockhandle IN VARCHAR2)
  RETURN INTEGER;
```

Parameters

Table 121-8 RELEASE Function Parameter

Parameter	Description
`id` or `lockhandle`	User assigned lock identifier, from 0 to 1073741823, or the lock handle, returned by `ALLOCATE_UNIQUE`, of the lock mode you want to change

Return Values

Table 121-9 RELEASE Function Return Values

Return Value	Description
0	Success
3	Parameter error
4	Do not own lock specified by `id` or `lockhandle`
5	Illegal lock handle

REQUEST Function

This function requests a lock with a specified mode.

`REQUEST` is an overloaded function that accepts either a user-defined lock identifier, or the lock handle returned by the `ALLOCATE_UNIQUE` procedure.

Syntax

```
DBMS_LOCK.REQUEST(
   id                IN  INTEGER ||
   lockhandle        IN  VARCHAR2,
   lockmode          IN  INTEGER DEFAULT X_MODE,
   timeout           IN  INTEGER DEFAULT MAXWAIT,
   release_on_commit IN  BOOLEAN DEFAULT FALSE)
  RETURN INTEGER;
```

The current default values, such as `X_MODE` and `MAXWAIT`, are defined in the `DBMS_LOCK` package specification.

Parameters

Table 121-10 REQUEST Function Parameters

Parameter	Description
`id` or `lockhandle`	User assigned lock identifier, from 0 to 1073741823, or the lock handle, returned by `ALLOCATE_UNIQUE`, of the lock mode you want to change
`lockmode`	Mode that you are requesting for the lock. For the available modes and their associated integer identifiers, see Constants.
`timeout`	Number of seconds to continue trying to grant the lock. If the lock cannot be granted within this time period, then the call returns a value of 1 (`timeout`).
`release_on_commit`	Set this parameter to `TRUE` to release the lock on commit or roll-back. Otherwise, the lock is held until it is explicitly released or until the end of the session.

Return Values

Table 121-11 REQUEST Function Return Values

Return Value	Description
0	Success
1	Timeout
2	Deadlock
3	Parameter error
4	Already own lock specified by `id` or `lockhandle`
5	Illegal lock handle