111 DBMS_JOB

The `DBMS_JOB` package schedules and manages jobs in the job queue.

> **Note:**
>
> The `DBMS_JOB` package has been superseded by the `DBMS_SCHEDULER` package, and support for `DBMS_JOB` might be removed in future releases of Oracle Database. In particular, if you are administering jobs to manage system load, you are encouraged to disable `DBMS_JOB` by revoking the package execution privilege for users.
>
> For more information, see DBMS_SCHEDULER and "Moving from `DBMS_JOB` to `DBMS_SCHEDULER`" in the *Oracle Database Administrator's Guide*.

This chapter contains the following topics:

- Security Model
- Operational Notes
- Summary of DBMS_JOB Subprograms

DBMS_JOB Security Model

`DBMS_JOB` uses the same security policies as `DBMS_SCHEDULER`. You must have the `CREATE JOB` privilege to use `DBMS_JOB`.

Jobs cannot be altered or deleted other than jobs owned by the user. This is true for all users including those users granted DBA privileges.

You can execute procedures that are owned by the user for which the user is explicitly granted `EXECUTE`. However, procedures for which the user is granted the execute privilege through roles cannot be executed.

Note that, once a job is started and running, there is no easy way to stop the job.

DBMS_JOB Operational Notes

These notes describe stopping a job, and working with Oracle Real Application Clusters.

Stopping a Job

Note that, once a job is started and running, there is no easy way to stop the job.

Working with Oracle Real Application Clusters

`DBMS_JOB` supports multi-instance execution of jobs. By default jobs can be executed on any instance, but only one single instance will execute the job. In addition, you can force instance binding by binding the job to a particular instance. You implement instance binding by specifying an instance number to the instance affinity parameter. Note, however, that in Oracle Database 10g Release 1 (10.1) instance binding is not recommended. Service affinity is preferred. This concept is implemented in the DBMS_SCHEDULER package.

The following procedures can be used to create, alter or run jobs with instance affinity. Note that not specifying affinity means any instance can run the job.

- `DBMS_JOB.SUBMIT`
- `DBMS_JOB.INSTANCE`
- `DBMS_JOB.CHANGE`
- `DBMS_JOB.RUN`

DBMS_JOB.SUBMIT

To submit a job to the job queue, use the following syntax:

```
DBMS_JOB.SUBMIT(
   job       OUT    BINARY_INTEGER,
   what      IN     VARCHAR2,
   next_date IN     DATE DEFAULT SYSDATE,
   interval  IN     VARCHAR2 DEFAULT 'NULL',
   no_parse  IN     BOOLEAN DEFAULT FALSE,
   instance  IN     BINARY_INTEGER DEFAULT ANY_INSTANCE,
   force     IN     BOOLEAN DEFAULT FALSE);
```

Use the parameters `instance` and `force` to control job and instance affinity. The default value of `instance` is 0 (zero) to indicate that any instance can execute the job. To run the job on a certain instance, specify the `instance` value. Oracle displays error `ORA-23319` if the `instance` value is a negative number or `NULL`.

The `force` parameter defaults to `false`. If `force` is `TRUE`, any positive integer is acceptable as the job instance. If `force` is `FALSE`, the specified instance must be running, or Oracle displays error number `ORA-23428`.

DBMS_JOB.INSTANCE

To assign a particular instance to execute a job, use the following syntax:

```
   DBMS_JOB.INSTANCE(  JOB IN BINARY_INTEGER,
     instance               IN BINARY_INTEGER,
     force                  IN BOOLEAN DEFAULT FALSE);
```

The `FORCE` parameter in this example defaults to `FALSE`. If the instance value is 0 (zero), job affinity is altered and any available instance can execute the job despite the value of force. If the `INSTANCE` value is positive and the `FORCE` parameter is `FALSE`, job affinity is altered only if the specified instance is running, or Oracle displays error `ORA-23428`.

If the `force` parameter is `TRUE`, any positive integer is acceptable as the job instance and the job affinity is altered. Oracle displays error ORA-23319 if the `instance` value is negative or `NULL`.

DBMS_JOB.CHANGE

To alter user-definable parameters associated with a job, use the following syntax:

```
   DBMS_JOB.CHANGE(  JOB IN BINARY_INTEGER,
   what                  IN VARCHAR2 DEFAULT NULL,
   next_date             IN DATE DEFAULT NULL,
   interval              IN VARCHAR2 DEFAULT NULL,
   instance               IN BINARY_INTEGER DEFAULT NULL,
   force                  IN BOOLEAN DEFAULT FALSE );
```

Two parameters, `instance` and `force`, appear in this example. The default value of `instance` is `null` indicating that job affinity will not change.

The default value of `force` is `FALSE`. Oracle displays error `ORA-23428` if the specified instance is not running and error `ORA-23319` if the `instance` number is negative.

DBMS_JOB.RUN

The `force` parameter for `DBMS_JOB.RUN` defaults to `FALSE`. If `force` is `TRUE`, instance affinity is irrelevant for running jobs in the foreground process. If force is `FALSE`, the job can run in the foreground only in the specified instance. Oracle displays error `ORA-23428` if force is `FALSE` and the connected instance is the incorrect instance.

```
   DBMS_JOB.RUN(
      job     IN BINARY_INTEGER,
      force   IN BOOLEAN DEFAULT FALSE);
```

Summary of DBMS_JOB Subprograms

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

Table 111-1 DBMS_JOB Package Subprograms

Subprogram	Description
BROKEN Procedure	Disables job execution
CHANGE Procedure	Alters any of the user-definable parameters associated with a job
INSTANCE Procedure	Assigns a job to be run by a instance
INTERVAL Procedure	Alters the interval between executions for a specified job
NEXT_DATE Procedure	Alters the next execution time for a specified job
REMOVE Procedure	Removes specified job from the job queue
RUN Procedure	Forces a specified job to run
SUBMIT Procedure	Submits a new job to the job queue
USER_EXPORT Procedures	Re-creates a given job for export, or re-creates a given job for export with instance affinity
WHAT Procedure	Alters the job description for a specified job

BROKEN Procedure

This procedure sets the broken flag. Broken jobs are never run.

Syntax

```
DBMS_JOB.BROKEN (
   job       IN  BINARY_INTEGER,
   broken    IN  BOOLEAN,
   next_date IN  DATE DEFAULT SYSDATE);
```

Parameters

Table 111-2 BROKEN Procedure Parameters

Parameter	Description
`job`	System-assigned ID of the job being run. To find this ID, query the `JOB` column of the `USER_JOBS` or `DBA_JOBS` view.
`broken`	Sets the job as broken or not broken. `TRUE` sets it as broken; `FALSE` sets it as not broken.
`next_date`	Next date when the job will be run.

Note:

If you set job as broken while it is running, Oracle resets the job's status to normal after the job completes. Therefore, only execute this procedure for jobs that are not running.

Usage Notes

- Your job will not be available for processing by the job queue in the background until it is committed.
- If a job fails 16 times in a row, Oracle automatically sets it as broken and then stops trying to run it.

CHANGE Procedure

This procedure changes any of the fields a user can set in a job.

Syntax

```
DBMS_JOB.CHANGE (
   job       IN  BINARY_INTEGER,
   what      IN  VARCHAR2,
   next_date IN  DATE,
   interval  IN  VARCHAR2,
   instance  IN  BINARY_INTEGER DEFAULT NULL,
   force     IN  BOOLEAN DEFAULT FALSE);
```

Parameters

Table 111-3 CHANGE Procedure Parameters

Parameter	Description
`job`	System-assigned ID of the job being run. To find this ID, query the `JOB` column of the `USER_JOBS` or `DBA_JOBS` view.
`what`	PL/SQL procedure to run.
`next_date`	Next date when the job will be run.
`interval`	Date function; evaluated immediately before the job starts running.

Table 111-3 (Cont.) CHANGE Procedure Parameters

Parameter	Description
`instance`	When a job is submitted, specifies which instance can run the job. This defaults to `NULL`, which indicates that instance affinity is not changed.
`force`	If this is `FALSE`, then the specified instance (to which the instance number change) must be running. Otherwise, the routine raises an exception. If this is `TRUE`, then any positive integer is acceptable as the job instance.

Usage Notes

- Your job will not be available for processing by the job queue in the background until it is committed.
- The parameters `instance` and `force` are added for job queue affinity. Job queue affinity gives users the ability to indicate whether a particular instance or any instance can run a submitted job.
- If the parameters `what`, `next_date`, or `interval` are `NULL`, then leave that value as it is.

Example

```
BEGIN
   DBMS_JOB.CHANGE(14144, null, null, 'sysdate+3');
   COMMIT;
END;
```

INSTANCE Procedure

This procedure changes job instance affinity.

Syntax

```
DBMS_JOB.INSTANCE (
   job          IN BINARY_INTEGER,
   instance     IN BINARY_INTEGER,
   force        IN BOOLEAN DEFAULT FALSE);
```

Parameters

Table 111-4 INSTANCE Procedure Parameters

Parameter	Description
`job`	System-assigned ID of the job being run. To find this ID, query the `JOB` column of the `USER_JOBS` or `DBA_JOBS` view.
`instance`	When a job is submitted, a user can specify which instance can run the job.
`force`	If this is `TRUE`, then any positive integer is acceptable as the job instance. If this is `FALSE` (the default), then the specified instance must be running; otherwise the routine raises an exception.

Usage Notes

Your job will not be available for processing by the job queue in the background until it is committed.

INTERVAL Procedure

This procedure changes how often a job runs.

Syntax

```
DBMS_JOB.INTERVAL (
   job       IN  BINARY_INTEGER,
   interval  IN  VARCHAR2);
```

Parameters

Table 111-5 INTERVAL Procedure Parameters

Parameter	Description
`job`	System-assigned ID of the job being run. To find this ID, query the `JOB` column of the `USER_JOBS` or `DBA_JOBS` view.
`interval`	Date function, evaluated immediately before the job starts running.

Usage Notes

- If the job completes successfully, then this new date is placed in `next_date`. `interval` is evaluated by plugging it into the statement select `interval` into `next_date` from dual;
- The `interval` parameter must evaluate to a time in the future. Legal intervals include:

Interval	Description
`'sysdate + 7'`	Run once a week.
`'next_day(sysdate,''TUESDAY'')'`	Run once every Tuesday.
`'null'`	Run only once.

- If `interval` evaluates to `NULL` and if a job completes successfully, then the job is automatically deleted from the queue.
- Your job will not be available for processing by the job queue in the background until it is committed.

NEXT_DATE Procedure

This procedure changes when an existing job next runs.

Syntax

```
DBMS_JOB.NEXT_DATE (
   job       IN  BINARY_INTEGER,
   next_date IN  DATE);
```

Parameters

Table 111-6 NEXT_DATE Procedure Parameters

Parameter	Description
`job`	System-assigned ID of the job being run. To find this ID, query the `JOB` column of the `USER_JOBS` or `DBA_JOBS` view.
`next_date`	Date of the next refresh: it is when the job will be automatically run, assuming there are background processes attempting to run it.

Usage Notes

Your job will not be available for processing by the job queue in the background until it is committed.

REMOVE Procedure

This procedure removes an existing job from the job queue. This currently does not stop a running job.

Syntax

```
DBMS_JOB.REMOVE (
   job       IN  BINARY_INTEGER );
```

Parameters

Table 111-7 REMOVE Procedure Parameters

Parameter	Description
`job`	System-assigned ID of the job being run. To find this ID, query the `JOB` column of the `USER_JOBS` or `DBA_JOBS` view.

Usage Notes

Your job will not be available for processing by the job queue in the background until it is committed.

Example

```
BEGIN
   DBMS_JOB.REMOVE(14144);
   COMMIT;
END;
```

RUN Procedure

This procedure runs job `JOB` now. It runs it even if it is broken.

Running the job recomputes `next_date`. See data dictionary view `USER_JOBS` or `DBA_JOBS`.

Syntax

```
DBMS_JOB.RUN (
   job       IN  BINARY_INTEGER,
   force     IN  BOOLEAN DEFAULT FALSE);
```

Parameters

Table 111-8 RUN Procedure Parameters

Parameter	Description
`job`	System-assigned ID of the job being run. To find this ID, query the `JOB` column of the `USER_JOBS` or `DBA_JOBS` view.
`force`	If this is `TRUE`, then instance affinity is irrelevant for running jobs in the foreground process. If this is `FALSE`, then the job can be run in the foreground only in the specified instance.

Example

```
EXECUTE DBMS_JOB.RUN(14144);
```

WARNING:

This re-initializes the current session's packages.

Exceptions

An exception is raised if `force` is `FALSE`, and if the connected instance is the wrong one.

SUBMIT Procedure

This procedure submits a new job. It chooses the job from the sequence `sys.jobseq`.

Syntax

```
DBMS_JOB.SUBMIT (
   job       OUT BINARY_INTEGER,
   what      IN  VARCHAR2,
   next_date IN  DATE DEFAULT SYSDATE,
   interval  IN  VARCHAR2 DEFAULT 'null',
   no_parse  IN  BOOLEAN DEFAULT FALSE,
   instance  IN  BINARY_INTEGER DEFAULT any_instance,
   force     IN  BOOLEAN DEFAULT FALSE);
```

Parameters

Table 111-9 SUBMIT Procedure Parameters

Parameter	Description
`job`	System-assigned ID of the job being run. To find this ID, query the `JOB` column of the `USER_JOBS` or `DBA_JOBS` view

Table 111-9 (Cont.) SUBMIT Procedure Parameters

Parameter	Description

`what`	PL/SQL text o the job to be run. This must be a valid PL/SQL statement or block of code. For example, to run a stored procedure `P`, you could pass the string `P`; (with the semi-colon) to this routine. The SQL that you submit in the `what` parameter is wrapped in the following PL/SQL block:
	`DECLARE`
	` job BINARY_INTEGER := :job;`
	` next_date DATE := :mydate;`
	` broken BOOLEAN := FALSE;`
	`BEGIN`
	` WHAT`
	` :mydate := next_date;`
	` IF broken THEN :b := 1; ELSE :b := 0; END IF;`
	`END;`
	Ensure that you include the `;` semi-colon with the statement.

`next_date`	Next date when the job will be run.
`interval`	Date function that calculates the next time to run the job. The default is `NULL`. This must evaluate to a either a future point in time or `NULL`.
`no_parse`	A flag. The default is `FALSE`. If this is set to `FALSE`, then Oracle parses the procedure associated with the job. If this is set to `TRUE`, then Oracle parses the procedure associated with the job the first time that the job is run. For example, if you want to submit a job before you have created the tables associated with the job, then set this to `TRUE`.
`instance`	When a job is submitted, specifies which instance can run the job.
`force`	If this is `TRUE`, then any positive integer is acceptable as the job instance. If this is `FALSE` (the default), then the specified instance must be running; otherwise the routine raises an exception.

Usage Notes

- Your job will not be available for processing by the job queue in the background until it is committed.
- The parameters `instance` and `force` are added for job queue affinity. Job queue affinity gives users the ability to indicate whether a particular instance or any instance can run a submitted job.

Example

This submits a new job to the job queue. The job calls the procedure `DBMS_DDL.ANALYZE_OBJECT` to generate optimizer statistics for the table `DQUON.ACCOUNTS`. The statistics are based on a sample of half the rows of the `ACCOUNTS` table. The job is run every 24 hours:

```
VARIABLE jobno number;
BEGIN
   DBMS_JOB.SUBMIT(:jobno,
      'dbms_ddl.analyze_object(''TABLE'',
      ''DQUON'', ''ACCOUNTS'',
      ''ESTIMATE'', NULL, 50);'
      SYSDATE, 'SYSDATE + 1');
   COMMIT;
END;
/
Statement processed.
print jobno
JOBNO
----------
14144
```

USER_EXPORT Procedures

There are two overloaded procedures. The first produces the text of a call to re-create the given job. The second alters instance affinity (8*i* and after) and preserves the compatibility.

Syntax

```
DBMS_JOB.USER_EXPORT (
   job    IN     BINARY_INTEGER,
   mycall IN OUT VARCHAR2);

DBMS_JOB.USER_EXPORT (
   job      IN     BINARY_INTEGER,
   mycall   IN OUT VARCHAR2,
   myinst   IN OUT VARCHAR2);
```

Parameters

Table 111-10 USER_EXPORT Procedure Parameter

Parameter	Description
`job`	System-assigned ID of the job being run. To find this ID, query the `JOB` column of the `USER_JOBS` or `DBA_JOBS` view.
`mycall`	Text of a call to re-create the given job.
`myinst`	Text of a call to alter instance affinity.

WHAT Procedure

This procedure changes what an existing job does, and replaces its environment.

Syntax

```
DBMS_JOB.WHAT (
   job       IN  BINARY_INTEGER,
   what      IN  VARCHAR2);
```

Parameters

Table 111-11 WHAT Procedure Parameters

Parameter	Description
`job`	System-assigned ID of the job being run. To find this ID, query the `JOB` column of the `USER_JOBS` or `DBA_JOBS` view.
`what`	PL/SQL procedure to run.

Usage Notes

- Your job will not be available for processing by the job queue in the background until it is committed.
- Some legal values of `what` (assuming the routines exist) are:
  - `'myproc(''10-JAN-82'', next_date, broken);'`
  - `'scott.emppackage.give_raise(''JENKINS'', 30000.00);'`
  - `'dbms_job.remove(job);'`