oracle dbms jobs 调用操作系统脚本

参考 ： http://www.dba-oracle.com/t_linux_oracle_dbms_scheduler_create_job.htm

Creating a Job That Calls an Executable

The job type executable allows you to create jobs which execute a command or script at the operating system level. The syntax is similar to creating other jobs, but the job type is set to executable and the job action should include the full path to the command or script to be executed.

```
begin
dbms_scheduler.create_job (
   job_name           =>  'migrate_files',
   job_type           =>  'executable',
   job_action         =>  '/home/oracle/bin/migrate_files.sh',
   start_date         =>  '01-mar-2010 07:00:00 am',
   repeat_interval    =>  'freq=daily',
   enabled            =>  true);
end;
/
```

When the date comes up to execute this job, Oracle executes the migrate_files.sh script as the user who the database is running under (typically oracle.) That user must already have execute privileges on the script or command to be run in order for the job to succeed.

Changing a Job

You can change anything about a scheduled job, except its name, using the dbms_scheduler.set_attribute procedure. The job name is given, followed by the attribute that you wish to change, and finally, the new value for that attribute.

```
begin
dbms_scheduler.set_attribute (
name               =>  'run_load_sales',
attribute          =>  'repeat_interval',
value              =>  'freq=daily; byhour=3');
end;
/
```

A job can be changed while it is running, but the changes will not take effect until the next run of the job.

Running a Job Manually

If you want to run a job immediately, call the dbms_scheduler.run_job procedure.

```
begin
dbms_scheduler.run_job (job_name => 'run_load_sales');
end;
/
```

This causes the named job to be run immediately.

Stopping Running Jobs

Running jobs can be stopped using the dbms_scheduler.stop_jobprocedure.

```
begin
dbms_scheduler.stop_job (job_name => 'run_load_sales');
end;
/
```

This only stops the running job and does not affect future running of this job.

Disabling and Enabling Jobs

The dbms_schedulerpackage includes the procedures disable and enable to disable and enable jobs. When a job is disabled, it will not be run.

```
begin
dbms_scheduler.disable (job_name => 'run_load_sales');
end;
/
```

If the job is running when the disable procedure is called, you get an error. You can stop the running job as shown in the last example, or you can add force => true to the disable statement.

To re-enable a job which has been disabled, use the enable procedure.

```
begin
dbms_scheduler.enable (job_name => 'run_load_sales');
end;
/
```

The job will now be run based on its original schedule. Multiple jobs can be disabled or enabled at the same time by separating their names with a comma.

Dropping Jobs

To permanently drop a job, call the procedure dbms_scheduler.drop_job. As with disabling and enabling jobs, multiple jobs can be specified by separating them with commas.

```
begin
dbms_scheduler.drop_job ('run_load_sales');
end;
/
```

If a job is running when you try to drop it, you get an error. You can stop the job, then drop it or set the force parameter to true. Setting force to true causes the running job to be stopped, and then the job is dropped from the scheduler.