Scheduling Jobs in Oracle DB with DBMS_SCHEDULER

Commands and Scripts with Examples

Asfaw Gedamu

12/02/2024

A. Introduction:

This Standard Operating Procedure (SOP) outlines the process of scheduling jobs in Oracle Database using the DBMS_SCHEDLER package. It covers creating schedules, programs, and jobs to automate tasks at specific times or intervals.

B. Purpose:

- Automate repetitive tasks, such as backups, maintenance scripts, data import/export, and report generation.
- Improve operational efficiency and reduce manual intervention.
- Ensure tasks are executed timely and reliably.

C. Prerequisites:

- Access to an Oracle Database with the DBMS_SCHEDULER package enabled.
- Basic understanding of SQL and PL/SQL.
- Privileges to grant execution and scheduling rights.

D. Scheduling Process:

1. **Define a Schedule:**
    - Use DBMS_SCHEDULER.CREATE_SCHEDULE to specify the date/time or recurrence pattern for the job.
    - Options include daily, weekly, monthly, annual, or interval-based schedules.
2. **Create a Program:**
    - Use DBMS_SCHEDULER.CREATE_PROGRAM to define the executable object (PL/SQL block, stored procedure, external script) to be run.
    - Specify program type (executable, stored procedure, external), name, and execution details.
3. **Schedule the Job:**
    - Use DBMS_SCHEDULER.CREATE_JOB to link the program with the schedule and define job details.

- Specify job name, owner, enabled/disabled status, retry attempts, and notification options.

E. Best Practices:

- Test scheduled jobs in a non-production environment before deployment.
- Use descriptive names for schedules, programs, and jobs for clarity.
- Implement logging and error handling within scheduled programs.
- Monitor job execution logs and address failures promptly.
- Secure job definitions and execution with appropriate privileges.

To schedule a job at a particular time in the database, first we need to create a schedule, then a program and finally a job.

1. Create a schedule

A schedule defines the start date, end time and repeat interval details

```
BEGIN
DBMS_SCHEDULER.CREATE_SCHEDULE (
Schedule_name => 'DAILYBILLINGJOB',
Start_date => SYSTIMESTAMP,
Repeat_interval =>'FREQ=DAILY;BYHOUR=11; BYMINUTE=30',
Comments => 'DAILY BILLING JOB'
);
END;
```

2. Create a program:

A program defines the name and type of the procedure, executed .package or script which executed.

```
BEGIN
DBMS_SCHEDULER.CREATE_PROGRAM (
program_name => 'DAILYBILLINGJOB',
program_type => 'STORED_PROCEDURE',
program_action => 'DAILYJOB.BILLINGPROC'
number_of_arguments =>0,
enabled => TRUE,
comments => 'DAILY BILLING JOB'
```

```
);
END;
```

3. Create job:

A job defines the schedule name and the program name.

```
DBMS_SCHEDULER.CREATE_JOB (
job_name => 'DAILYBILLINGJOB_RUN',
program_name => 'DAILYBILLINGJOB',
schedule_name => 'DAILYBILLINGJOB_SCHED',
enabled => FLASE,
comments => 'daily billing job'
);
END;
```

Instead of creating scheduler,job and program separately, we can create the scheduler job with below commad directly.

Simple command to create scheduler job:

```
BEGIN
   DBMS_SCHEDULER.CREATE_JOB (
          job_name => '"HWS"."MV_REF_DBA_DATA"',
          job_type => 'PLSQL_BLOCK',
          job_action => 'dbms_refresh.refresh(""HWS"."STC_NEXT_DBA_MV_DATA"");',
          number_of_arguments => 0,
          start_date => NULL,
          repeat_interval => 'FREQ=DAILY;BYHOUR=8;BYMINUTE=00;BYSECOND=00',
          end_date => NULL,
          enabled => FALSE,
          auto_drop => FALSE,
          comments => 'Converted_dba_jobs');

   DBMS_SCHEDULER.enable(
           name => '"HWS"."MV_REF_FTTH_DATA"');
END;
```

Note: if the job_type is procedure, then use job_type='STORED_PROCEDURE'

4. View schedule details of all schedulers:

```
set pagesize 200
set lines 299
```

```
col START_DATE for a45
col REPEAT_INTERVAL for a45
col schedule_name for a34
select schedule_name, schedule_type, start_date, repeat_interval from dba_scheduler_schedules;
```

5.Enable a job

EXECUTE DBMS_SCHEDULER.ENABLE('SCOTT.MONTHLYBILLING');

6.Disable a job

EXECUTE DBMS_SCHEDULER.DISABLE('SCOTT.MONTHLYBILLING');

7.Stop a running job

EXECUTE DBMS_SCHEDULER.STOP_JOB('SCOTT.MONTHLYBILLING');

8.Drop a running job

EXECUTE DBMS_SCHEDULER.DROP_JOB('SCOTT.MONTHLYBILLING');

9. Run a job immediately

EXECUTE DBMS_SCHEDULER.RUN_JOB('SCOTT.MONTHLYBILLING');

10. Drop a schedule:

```
BEGIN
DBMS_SCHEDULER.DROP_SCHEDULE(
schedule_name => 'DAILYBILLINGJOB_SCHED',
force => TRUE
);
END;
```

11. Drop a scheduler job:

DBMS_SCHEDULER.drop_job (job_name => 'SCOTT.MONTHLYBILLING');

12. Scheduler shell script in dbms_scheduler:

— This feature in available from oracle 12c onward

Create a credential store:

```
BEGIN
dbms_credential.create_credential (
CREDENTIAL_NAME => 'ORACLEOSUSER',
USERNAME => 'oracle',
PASSWORD => 'oracle@98765',
DATABASE_ROLE => NULL,
WINDOWS_DOMAIN => NULL,
COMMENTS => 'Oracle OS User',
ENABLED => true
);
END;
/
```

Then create the job:

```
exec dbms_scheduler.create_job(-
job_name=>'myscript4',-
job_type=>'external_script',-
job_action=>'/export/home/oracle/ttest.2.sh',-
enabled=>true,-
START_DATE=>sysdate,-
REPEAT_INTERVAL =>'FREQ=MINUTELY; byminute=1',-
auto_drop=>false,-
credential_name=>'ORACLEOSUSER');
```

13. Monitor scheduler jobs:

— Monitor currently running jobs

SELECT job_name, session_id, running_instance, elapsed_time, FROM dba_scheduler_running_jobs;

— View the job run details

select * from DBA_SCHEDULER_JOB_RUN_DETAILS;

— View the job-related logs:

select * from DBA_SCHEDULER_JOB_LOG;

14. Get DDL of a scheduler job:

select dbms_metadata.get_ddl('PROCOBJ','DUP_ACC','SCOTT') from dual;

15. Copy scheduler job from one user to another :

exec dbms_scheduler.copy_job('SCOTT.MY_JOB_2','DBACLASS.MY_JOB_2');

16. Get log information of scheduler jobs:

```
set pagesize 299
set lines 299
col job_name for a24
col log_date for a40
col operation for a19
col additional_info a79
select job_name,log_date,status,OPERATION,ADDITIONAL_INFO from
dba_scheduler_job_log order by log_date desc;
```

17. History of all scheduler job runs:

```
set pagesize 299
set lines 299
col JOB_NAME for a24
col actual_start_date for a56
col RUN_DURATION for a34
select job_name,status,actual_start_date,run_duration from
DBA_SCHEDULER_JOB_RUN_DETAILS order by ACTUAL_START_DATE desc;
```

18. Managing scheduler credentials:

— Create a credential:

```
BEGIN
dbms_credential.create_credential (
CREDENTIAL_NAME => 'ORACLEOSUSER',
USERNAME => 'oracle',
PASSWORD => 'oracle@123',
DATABASE_ROLE => NULL,
WINDOWS_DOMAIN => NULL,
COMMENTS => 'Oracle OS User',
ENABLED => true
);
END;
/
```

— Drop a credential

exec dbms_scheduler.drop_credential('ORACLEOSUSER');

— View credential details

select owner,CREDENTIAL_NAME,USERNAME,ENABLED from DBA_CREDENTIALS;

— Change username and password in a credentials :

exec DBMS_SCHEDULER.SET_ATTRIBUTE(name=>'ORACLEOSUSER',attribute=>'password',value=>'oracle');

19. View and manage auto task jobs in database:

```
set lines 180 pages 1000
col client_name for a40
col attributes for a60
select client_name, status,attributes,service_name from dba_autotask_client
/
BEGIN
  DBMS_AUTO_TASK_ADMIN.disable(
    client_name => 'auto space advisor',
    operation   => NULL,
    window_name => NULL);
END;
/

BEGIN
  DBMS_AUTO_TASK_ADMIN.disable(
    client_name => 'sql tuning advisor',
    operation   => NULL,
    window_name => NULL);
END;
/

BEGIN
  DBMS_AUTO_TASK_ADMIN.disable(
    client_name => 'auto optimizer stats collection',
    operation   => NULL,
    window_name => NULL);
END;
/
```

F. Conclusion:

By following this SOP and leveraging DBMS_SCHEDULER, you can effectively automate tasks in your Oracle Database, streamlining operations and ensuring timely execution of critical routines. Remember to prioritize security, testing, and monitoring for successful job scheduling.

Additional Notes:

- This SOP provides a basic overview. Refer to the Oracle documentation for advanced scheduling options and troubleshooting guidelines.
- Consider including specific examples and use cases relevant to your environment for a more practical guide.
- Regularly update the SOP to reflect changes in DBMS_SCHEDULER functionality and best practices.