Architecture and Design of Oracle Scheduler

Jimin Liu
Computer & Communication College, Beijing Information Technology College, Beijing, China
Corresponding author’s e-mail: liujm@bitc.edu.cn

Abstract. Oracle scheduler focuses on the automatic management, maintenance and monitoring of Oracle database jobs. Through it, many routine database tasks can be automatically executed, and human intervention can be reduced, so as to improve the efficiency of system operation and implement more flexible and complex task management. This paper introduces the concept and classification of Oracle scheduler, analyzes each component of Oracle scheduler, explains its overall architecture and tasks of different components, summarizes the problems needing attention in design according to design experience, and illustrates its application through the creation of Oracle scheduler object instance, job scheduling and evaluation of execution frequency.

Keywords. Oracle scheduler, job coordinator, architecture, job object

1. Concept and classification of oracle scheduler
Oracle scheduler, as a logical entity for scheduling, managing and executing jobs, can automate your business processes. The so-called logical entity means that after the schedule is created, this object exists in the database, but it is used to describe the execution cycle of the job. It can also help you control the use of resources and can prioritize jobs in the database. Traditional limitation of DBMS_jobs is that it can only schedule jobs based on PL/SQL, and cannot be used to schedule executable files or scripts of the operating system.

Oracle scheduler is an enterprise-level task scheduler, which used to schedule hundreds of tasks through DBMS_Scheduler package procedures and functions to achieve. The scheduler can control when and where to perform what tasks. The task scheduling of scheduler is more flexible. It schedules based on time point, event and dependency relationship between different tasks. The most commonly used scheduling method based on time point. At the same time, the scheduler also supports scheduling the consumption of system resources according to the priority of tasks and tasks, and supports the execution of tasks in local or remote destinations as well. The specific tasks can be database scripts and stored procedures, or operating system shell scripts and batch files [1].

Oracle scheduler, as a logical entity for scheduling, managing and executing jobs, can automate your business processes. The so-called logical entity means that after the schedule is created, this object exists in the database, but it is used to describe the execution cycle of the job. It can also help you control the use of resources and can prioritize jobs in the database. Traditional DBMS_jobs limitation
of jobs is that it can only schedule jobs based on PL / SQL, and cannot be used to schedule executable files or scripts of the operating system [2].

2. Oracle scheduler component

The scheduler consists of the following basic components: jobs, schedules, procedures, chains, and events.

2.1. JOB

A job is a plan to perform one or more tasks, which is the essence of scheduling tasks. As the core of the whole scheduler job, job includes three parts: when, where and how. When is also the meaning of “plan” in a scheduled task. Specify a certain time to complete a certain operation or event, where, exactly where the job is completed, and the job can be completed on a local or remote host. Of course, agent deployment is required. How to define multiple task implementation methods [3].

2.2. Schedule

A schedule is a description of the time and frequency of a job performed by the database. If some jobs are running at roughly the same time, a schedule can be used to simplify the creation and management of these jobs. Examples are as follows:

SQL> begin
  2  dbms_scheduler.create_schedule(
  3  schedule_name=>'test_schedule',
  4  start_date=> systimestamp,
  5  repeat_interval=>'FREQ=MINUTELY; INTERVAL=30',
  6  end_date=> systimestamp+3,
  7  comments=>'Every 30 minute');
  8  end;
  9 /

2.3. Program

The program includes metadata about a scheduler job. Program includes program name, program type and program action. It is the actual name of a procedure or executable script. Examples are as follows:

SQL> begin
  2  dbms_scheduler.create_program(
  3  program_name=>'test_program',
  4  program_type=>'STORED_PROCEDURE',
  5  program_action=>'auto_archive_emp',
  6  enabled=>true,
  7  comments=>'use to execute the procedure of auto_archive_emp');
  8  end;
  9 /

2.4. Chain

The concept of scheduler chain can be used to link related programs together. Therefore, the successful running of a program may be based on the successful running of another program; it can also be based on a key rather than the program starting the job. Successive positions in a chain are called “steps” in a chain, and each step points to another chain, program, or event. Because the chain uses Oracle streams rules engine. Therefore, users must have create job and rules engine permissions to create a chain [4]. Examples are as follows:

SQL> begin
  2  dbms_rule_adm.grant_system_privilege(
  3  dbms_rule_adm.create_rule_obj,'ECIQINTF');
4 dbms_rule_adm.grant_system_privilege(
5 dbms_rule_adm.create_rule_set_obj,'ECIQINTF ');
6 dbms_rule_adm.grant_system_privilege(
7 dbms_rule_adm.create_evaluation_context_obj,'ECIQINTF ');
8 end;
9 /

2.5. Event
The scheduler uses Oracle stream advanced queuing to trigger events and start event based database jobs. An event is a message sent by an application or process when it notices an event or action. There are two types of events: events raised by the scheduler and events raised by applications. Events raised by the scheduler are caused by changes in the scheduler running. For example, the successful completion of the scheduler job is an event. Events raised by an application are used or consumed by the scheduler to start a job.

2.6. The relationship among job, program and schedule
The same program can be called by multiple jobs. Similarly, multiple jobs can share the same schedule. As shown in Figure 1.

![Figure 1. The relationship among job, program and schedule](image)

3. Oracle scheduler architecture
The Oracle scheduler architecture is shown in the Figure 2.

![Figure 2. Oracle scheduler architecture](image)

Job table. Each database has a table to store job data, using package DBMS. The procedures and functions in scheduler can maintain the relevant data of the job table, such as DBMS_SCHEDULER.CREATE_Job is a package DBMS. In the process of creating a job, scheduler will add
a job data to the job table. Through the view of DBA_SCHEDULER_Jobs queries all job lists in the current system.

The job Coordinator (job Coordinator) is a special type of Oracle background process (process name is cjq0) used to schedule jobs. The process information can be viewed from V$process. There is a job coordinator process for each database instance. The process starts and stops as needed. When the database gets started, the process does not start. When the database detects that there are jobs to run, the process will start. As long as there are jobs to run, the process will continue to run. The process stops only if there are no jobs to run in the future [5].

The main tasks of the job coordinator including: controlling and deriving the job slave process (the process name is j001, which actually executes the job); querying the job table data; extracting job information and passing it to the job slave process; and cleaning up the job slave process.

The number of job slave processes derive from the job coordinator, which determined by the instance initialization parameter job_ QUEUE. The parameters can be modified according to the actual situation.

Job execution. The job slave process is used to actually execute the job. When a job needs to be run, the job slave will be awakened by the job coordinator. When a job is ready to run, the job slave process needs to perform the following tasks: collect some metadata information about the running job, such as permissions and parameters; start a database session with the job owner user, start a transaction, and execute the job; after the job is executed, the job slave commits the transaction; and closes the session.

The job is completed. When the job is completed, the job slave process performs the following operations: rescheduling the job if necessary; updating the status of the job table; inserting data into the job log table; updating the number of runs; and finally cleaning up [6].

4. Problems to be considered in job design

4.1. Resource release during job execution
Each time a job is executed, a new database session will be opened. That is to say, each job is executed with different Sid and sieral. The use of variables, cursors and dblink in the PL/SQL code executed by the session will release resources when the session is closed.

4.2. Database transaction problems
Database transaction problems. After all PL/SQL codes in the job are executed, the job slave will implicitly commit the transaction. If there are uncommitted DML statements after the job is executed, the job slave process will help commit the transaction. Normally, in the PL/SQL code executed by the job, we will explicitly control the transaction commit or rollback, and will not use the job slave process to implicitly commit the transaction.

4.3. How to determine the next running time of the job
After the job starts to start, the job parameter repeat_interval is immediately used to determine the next run time of the job. When the next running time arrives, if the job instance is still running, the new job instance cannot start to execute and wait until the currently running job instance is completed. Therefore, for the same job, it is a job instance followed by a job instance in a serial manner, so there will be no concurrent execution of job instances [7].

5. Application of oracle scheduler

5.1. Create job
Using DBMS_SCHEDULER.CREATE_JOB can create a new job, create Job procedures can be called in many overloaded ways. Users with create job permission can create jobs in their own schema, and users with create any job permission can create jobs in any schema except sys. A job has three key elements: when, what task and where. JOB_Name is the job name, job_Type and job_Action defines
what task to perform, start_ DATE、REPEAT_ Inter Val and end_ Date defines when the task is executed, destination_ Name defines where to execute the task, and enable defines whether the job can run. In the following example, job_ Type defines job_ Action is database PL / SQL block script, start_ Date definition starts from the current time, end_ Date definition is executed indefinitely, repeat_ Inter Val defines 0, 15, 30 and 45 minutes per hour. Enable defines that the job can run automatically after it is created. Destination is not defined_ Name means running locally. [8]

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (job_name => 'ECIQINTF_MAIN_JOB', job_type => 'PLSQL_BLOCK', job_action => 'BEGIN INTF_MAIN.PC_MAIN(); END;', start_date => SYSTIMESTAMP, repeat_interval => 'freq=hourly; byminute=0,15,30,45; bysecond=0;', end_date => NULL, enabled => TRUE, Comments => 'timed tasks of customs clearance verification platform');
END;
```

5.2. Oracle scheduler object

Through the above example, you can easily create a job to complete the work. If there are a large number of jobs, you may need to reuse them_ Action parameter or repeat_ Inter Val parameter, so that the job can be modular management.

Through the job_ Type and job_ The action is encapsulated as a program object, as shown below.

```
BEGIN
DBMS_SCHEDULER.CREATE_PROGRAM (program_name => 'ECIQINTF_MAIN_JOB_PROGRAM', program_type => 'PLSQL_BLOCK', program_action => 'BEGIN INTF_MAIN.PC_MAIN(); END;', enabled => TRUE);
END;
```

By setting start_ DATE、REPEAT_ Inter Val and end_ Date is encapsulated as a schedule object, as shown below.

```
BEGIN
DBMS_SCHEDULER.CREATE_SCHEDULE (schedule_name => 'ECIQINTF_MAIN_JOB_SCHEDULE', start_date => SYSTIMESTAMP, end_date => NULL, repeat_interval => 'freq=hourly; byminute=0,15,30,45; bysecond=0;', comments => 'Every 15 minutes');
END;
```

Implement the previous job using the program object and schedule object again, as shown below.

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (job_name => 'ECIQINTF_MAIN_JOB', program_name => 'ECIQINTF_MAIN_JOB_PROGRAM', schedule_name => 'ECIQINTF_MAIN_JOB_SCHEDULE', enabled => TRUE, Comments => 'timed tasks of customs clearance verification platform');
END;
```

5.3. Evaluation of job execution frequency
By setting the repeat for a job object or schedule object, the interval parameter, which is an expression, adjusts the frequency of job execution. If you need to evaluate the execution frequency without actually executing the job, you can use DBMS_SCHEDULER.EVALUATE_CALENDAR_STRING process output repeat. The actual frequency of execution of the interval parameter is shown below [9].

```sql
DECLARE
    start_date TIMESTAMP;
    return_date_after TIMESTAMP;
    next_run_date TIMESTAMP;
BEGIN
    --start_date := to_timestamp_tz('2019-11-21 16:32:15','YYYY-MM-DD HH24:MI:SS');
    start_date := systimestamp;
    return_date_after := start_date;
    FOR i IN 1..100 LOOP
        DBMS_SCHEDULER.EVALUATE_CALENDAR_STRING(
            'freq=hourly; byminute=0,15,30,45; bysecond=0',
            start_date, return_date_after, next_run_date);
        DBMS_OUTPUT.PUT_LINE('next_run_date: ' || next_run_date);
        return_date_after := next_run_date;
    END LOOP;
END;
```

next_run_Date: 09-12-19 09.00.00.600000 am
next_run_Date: 09-12-19 09.15.00.600000 am
next_run_Date: 09-12-19 09.30.00.600000 am
next_run_Date: 09-12-19 09.45.00.600000 am
next_run_Date: 09-12-19 10.00.00.600000 am
next_run_Date: 09-12-19 10.15.00.600000 am
next_run_Date: 09-12-19 10.30.00.600000 am
......

6. Summary

Through the previous discussion, we can see the powerful function of Oracle schedule. In the cloud era, resource management becomes very important, and Oracle scheduler can closely combine with Oracle's own resource management to improve the efficiency and effectiveness of resource management. So as to improve the system performance and provide faster and more stable service.

References

[1] Zhang Junhua, Wu Xiaoyue, and Liu Shenjun, Oracle high performance system architecture: Peking University Press, vol. 1, 2019, pp. 264-272.
[2] Yang Jianrong, Oracle DBA work notes: operation and maintenance, data migration and performance tuning, China Railway Press, vol. 2, 2016, pp. 86-103.
[3] Luo Luo, Oracle query optimization rewriting skills and case 2.0 (produced by blog point of view), Electronic Industry Press, vol. 2, 2018, pp. 193-201.
[4] Zhang Junhua, Oracle high performance system architecture practical compendium, Peking University Press, vol. 1, 2020, pp. 265-278.
[5] Luo Bingsen, Huang Chao, and Zhong Qi, SQL optimization core idea (asynchronous book production), people's Posts and Telecommunications Press, vol. 1, 2018, pp. 203-214.
[6] Gan Changchun, Zhang Jianjun, Oracle database storage management and performance optimization, China Railway Press, vol. 1, 2020, pp. 167-180.
[7] Zhang Jie, Zhao Yumei, Shi Zhe, “Research and application of GeoEast database optimization technology”, computer age, vol. 5, 2019, pp. 12-15, DOI: 10.16644/j.cnki.cn33-1094/tp.2019.05.004

[8] Huang Jiadong, she Jun, “Research on adaptive adjustment and optimization of database resources”, information and communication, vol. 7, 2017, pp. 120-122.

[9] Cheng Miao,”Processing of abnormal dispatching command module caused by a database server failure”, railway communication signal, vol. 12017, pp. 77-78 + 82, DOI: 10.13879/j.issn1000-7458.2017-01.16501