

Планировщик заданий

Графеева Н.Г.

2020

Введение

- СУБД Oracle — большой и сложный механизм, требующий выполнения определенных плановых работ, таких как сбор статистики о хранимых объектах или сбор/очистка внутренней информации.
- Первый механизм планового запуска появился в версии 7 для поддержки автоматических обновлений снимков (snapshots), как поначалу именовались нынешние материализованные виртуальные таблицы (materialized views). В версии 8 этот механизм был открыт для обычных пользователей через посредство некоторых параметров СУБД, таблиц словаря, а также пакета DBMS_JOB. Пакет DBMS_JOB позволял (и позволяет) запускать хранимую процедуру, или же неименованный блок PL/SQL в моменты времени, вычисляемые по указанной пользователем формуле.
- К версии 10 такое устройство имевшегося планировщика заданий было сочтено слишком примитивным, и в ней появился новый планировщик DBMS_SCHEDULER, значительно более проработанный.

Основные понятия планировщика DBMS_SCHEDULER

- *Schedule* (расписание)
- *Program* (программа)
- *Job* (плановое задание = расписание + программа)
- *Chain* (последовательность заданий)

Объекты словаря данных

- таблицы словаря LIKE '%SCHEDULER_%':
- DBA_SCHEDULER_JOBS
DBA_SCHEDULER_JOB_LOG
- **USER_SCHEDULER_PROGRAMS**
- **USER_SCHEDULER_JOBS**
- и прочие

Типы заданий (и программ)

- PL/SQL – процедура (STORED_PROCEDURE)
- PL/SQL - блок (PLSQL_BLOCK)
- external OS-program (EXECUTABLE)

Пример (простейшее задание с PLSQL блоком)

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB
( job_name  => 'simple_job',
  job_type  => 'PLSQL_BLOCK',
  job_action => 'UPDATE emp SET sal = sal +1;',
  enabled    => TRUE
);
END;
```

Пример (простейшее задание с вызовом хранимой процедуры)

```
CREATE PROCEDURE updatesal AS BEGIN UPDATE emp SET sal = sal - 1; END;  
/
```

```
BEGIN  
DBMS_SCHEDULER.CREATE_JOB  
( job_name => 'simple_job',  
  job_type => 'STORED_PROCEDURE',  
  job_action => 'updatesal',  
  enabled => TRUE  
);  
END;  
/
```

Пример (задание параметров задания с ВНЕШНИМ ВЫЗОВОМ)

```
BEGIN  
DBMS_SCHEDULER.CREATE_JOB  
( job_name => 'simple_job',  
  job_type => 'EXECUTABLE',  
  job_action => 'cmd.exe /C dir > \temp\out.txt',  
  enabled => TRUE  
);  
END;
```


Параметры для указания запуска заданий

- Следующие параметры процедуры CREATE_JOB задают начало, конец и периодичность выполнения заданий:
- start_date => SYSTIMESTAMP + INTERVAL '10' SECOND
- end_date => SYSTIMESTAMP + INTERVAL '100' SECOND
- repeat_interval => 'FREQ=MONTHLY; BYDAY=SUN, -1 SAT'
(В результате задание будет исполняться ежемесячно по воскресеньям и последним субботам месяца)

Примеры (использование специализированного языка для запуска заданий)

- `FREQ=HOURLY;INTERVAL=4` каждые 4 часа;
- `FREQ=MINUTELY;INTERVAL=5` каждые 5 минут
- `FREQ=HOURLY;INTERVAL=4;BYMINUTE=10;BYSECOND=30` каждые 4 часа на 10-й минуте, 30-й секунде;
- `FREQ=YEARLY;BYYEARDAY=-276` каждое 31-е марта;
- `FREQ=YEARLY;BYMONTH=MAR;BYMONTHDAY=31` каждое 31-е марта;

Как проверить правильность составленного выражения?

```
DECLARE
  next_run_date TIMESTAMP;
BEGIN
  DBMS_SCHEDULER.EVALUATE_CALENDAR_STRING
  (
    'FREQ=MINUTELY;INTERVAL=4',
    SYSTIMESTAMP,
    NULL,
    next_run_date
  );
  DBMS_OUTPUT.PUT_LINE ( 'next_run_date: ' || next_run_date );
END;
```

Информация о заданиях

- Если указать план запуска, задание появится в словаре уже надолго. Удалить его при необходимости можно будет так:
- **EXECUTE DBMS_SCHEDULER.DROP_JOB ('simple_job', TRUE)**
- Информацию об имеющихся заданиях пользователь может посмотреть запросом:
- **SELECT job_name, state, enabled**
- **FROM user_scheduler_jobs;**
- Более подробную информацию можно обнаружить в таблицах **USER_SCHEDULER_%**, а более общую – в обычной таблице **USER_OBJECTS**.

Скомпонованное задание

Пакет DBMS_SCHEDULER имеет и более продвинутые возможности для компоновки заданий из независимых элементов: программы и расписания. Характерная особенность в том, что оба эти элемента самостоятельны; их можно комбинировать в разных заданиях и изменять, не внося изменений в определения заданий.

Пример (создание программы)

```
BEGIN
DBMS_SCHEDULER.CREATE_PROGRAM
( program_name => 'simple_program',
  program_type => 'STORED_PROCEDURE' ,
  program_action => 'updatesal',
  enabled      => TRUE
);
END;
```

Примечания

- Информация об имеющихся программах для планировщика присутствует в представлениях:
`DBA/ALL/USER_SCHEDULER_PROGRAMS`.
- Другими значениями параметра `PROGRAM_TYPE` могут быть `'PLSQL_BLOCK'` и `'EXECUTABLE'` (как и типов заданий).

Пример (процедуры с параметрами)

```
CREATE PROCEDURE salary ( deer NUMBER ) AS  
BEGIN  
    UPDATE emp SET sal = sal - deer;  
END;  
/
```

```
BEGIN  
DBMS_SCHEDULER.CREATE_PROGRAM  
( program_name      => 'simple_program1',  
  program_type      => 'STORED_PROCEDURE',  
  program_action     => 'salary',  
  enabled            => FALSE,  
  number_of_arguments => 1  
);  
END;  
/
```


Пример(уточнение фактических параметров вызова программы)

```
BEGIN
DBMS_SCHEDULER.DEFINE_PROGRAM_ARGUMENT
( program_name    => 'simple_program1',
  argument_position => 1,
  argument_name    => 'DELTA',
  argument_type    => 'NUMBER'
);
END;
/

BEGIN DBMS_SCHEDULER.ENABLE ( 'simple_program1' ); END
```

Пример(уточнение фактических параметров вызова программы)

```
BEGIN
DBMS_SCHEDULER.DEFINE_PROGRAM_ARGUMENT
( program_name    => 'simple_program1',
  argument_position => 1,
  argument_name    => 'DELTA',
  argument_type    => 'NUMBER',
  default_value    => 8
);
END;
/
```

```
BEGIN DBMS_SCHEDULER.ENABLE ( 'simple_program1' ); END
```

Пример (создание расписания)

```
BEGIN
  DBMS_SCHEDULER.CREATE_SCHEDULE
    ( schedule_name => 'simple_schedule',
      start_date    => SYSTIMESTAMP,
      repeat_interval => 'FREQ=WEEKLY; BYDAY=MON, TUE, WED, THU, FRI',
      end_date      => SYSTIMESTAMP + INTERVAL '1' MONTH
    );
END;
```

Информация о расписаниях

- Информация об имеющихся расписаниях для планировщика находится в представлениях словаря:
- DBA/ALL/USER_SCHEDULER_SCHEDULES

Пример (скомпонованное задание для программы без параметров)

```
BEGIN
```

```
  DBMS_SCHEDULER.CREATE_JOB
```

```
    ( job_name      => 'compound_job',
```

```
      program_name => 'simple_program',
```

```
      schedule_name => 'simple_schedule',
```

```
      enabled       => TRUE
```

```
    );
```

```
END;
```

Пример (скомпонованное задание для программы с параметрами)

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB
    ( job_name    => 'compound_job1',
      program_name => 'simple_program1',
      schedule_name => 'simple_schedule',
      enabled      => FALSE
    );
END;
/

BEGIN
  DBMS_SCHEDULER.SET_JOB_ANYDATA_VALUE
    ( job_name    => 'compound_job1',
      argument_name => 'DELTA',
      argument_value => ANYDATA.CONVERTNUMBER ( 3 )
    )
END;
/
```

```
BEGIN DBMS_SCHEDULER.ENABLE ( 'compound_job1' ); END
```

Домашнее задание 11(10 баллов)

!!!!!!! Именно из-за этого задания был скандал с АРЕХ

Создайте приложение, которое позволяет:

- 1.Выполнять запуск задания (что-то типа кнопки Start).
- 2.Исполнять задание 20 -30 раз (не больше!!!!). В журнале debug_log записывать каждый вызов задания.
- 3.Задание должно создавать во вспомогательной таблице точку с двумерными координатами.
- 4.Приложение должно отображать динамику порождение новых точек как точек на графике (точечная диаграмма или диаграмма – радар).
- 5.В приложении должна быть предусмотрена кнопка для очистки вспомогательной таблицы (и графика).

Примечания:

- Генерация случайных чисел в заданном диапазоне:
dbms_random.VALUE(min_val,max_val).
- Настоятельно советую зафиксировать масштаб графика с самого начала, например точками с минимальными и максимальными значениями.