

# Планировщик заданий

Графеева Н.Г.

2017

# Введение

- СУБД Oracle — большой и сложный механизм, требующий выполнения определенных плановых работ, таких как сбор статистики о хранимых объектах или сбор/очистка внутренней информации. Необходимость осуществлять плановый запуск работ могут испытывать и пользователи БД.
- 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

- *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 (простое 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 (простое 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 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;



# Возможности для указания запуска заданий

- Следующие параметры процедуры CREATE\_JOB:
- 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 воскресениям и последним субботам месяца)

# Примеры (использование 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

- Более развитая возможность DBMS\_SCHEDULER позволяет скомпоновать задание 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(5 баллов)

Создайте приложение, которое позволяет:

- 1.Выполнять запуск задания.
- 2.Исполнять задание 10 -20 раз (не больше!!!!!!).
- 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 временной таблицы (и графика).

Генерация случайных чисел в заданном диапазоне:

`dbms_random.VALUE(min_val,max_val).`

Результат отправьте по адресу [N.Grafeeva@spbu.ru](mailto:N.Grafeeva@spbu.ru). Тема письма – `DB_Application_2017_job11`.

*Примечание: задание должно быть отправлено в течение 2 недель. За более позднее отправленние будут сниматься штрафные баллы ( по баллу за каждые 3 недели).*