MEETUP #1 - PROCESSAMENTO PARALELO COM DBMS_SCHEDULER

PL/SQL CAMP – 24/05/2018

CONTEÚDO DA APRESENTAÇÃO

- O cenário alvo.
- Exemplo de um processo não paralelizado.
- Execução paralela com o DBMS_SCHEDULER.
- Controle das sessões
- Pontos de atenção.
- Conclusão.

O CENÁRIO ALVO

- Processos que fazem algum tipo de fechamento ou processamento em lote.
- Processos que possuem um prazo máximo para serem concluídos e não possuem restrição de uso de MEMÓRIA e CPU.
  - Processos que a partir de uma lista de ID's (dados em geral) devem fazer cálculos complexos ou demorados para gerar uma saída para cada registro do cursor. Ex: Folha de pagamento.
  - (ou) Processos que possuem várias fases que podem ser executadas simultaneamente.

EXEMPLO DE UM PROCESSO NÃO PARALELIZADO.

```
DECLARE
  vStartHour DATE;
BEGIN
  /*
  output to show the processing time
  */
  vStartHour := SYSDATE;
  dbms_output.put_line('Start: ' ||
                       TO_CHAR(vStartHour, 'dd/mm/yyyy hh24:mi:ss'));
  -- Call the procedure
  pk_payroll_processing.pr_calculate_month_payroll(ppayrollmonth => to_date('01/01/2005',
                                                                            'dd/mm/yyyy'));
  /*
  output to show the processing time
  */
  dbms_output.put_line('End: ' ||
                       TO_CHAR(SYSDATE, 'dd/mm/yyyy hh24:mi:ss') || ' - ' ||
                       ROUND((SYSDATE - vStartHour) * 24 * 60 * 60, 2) || 's');
END;
```

```
Start: 15/05/2018 08:04:28
End: 15/05/2018 08:05:25 - 57s
```

Resumo do processo:

Obter a lista de funcionários

Processar folha de Pagamento

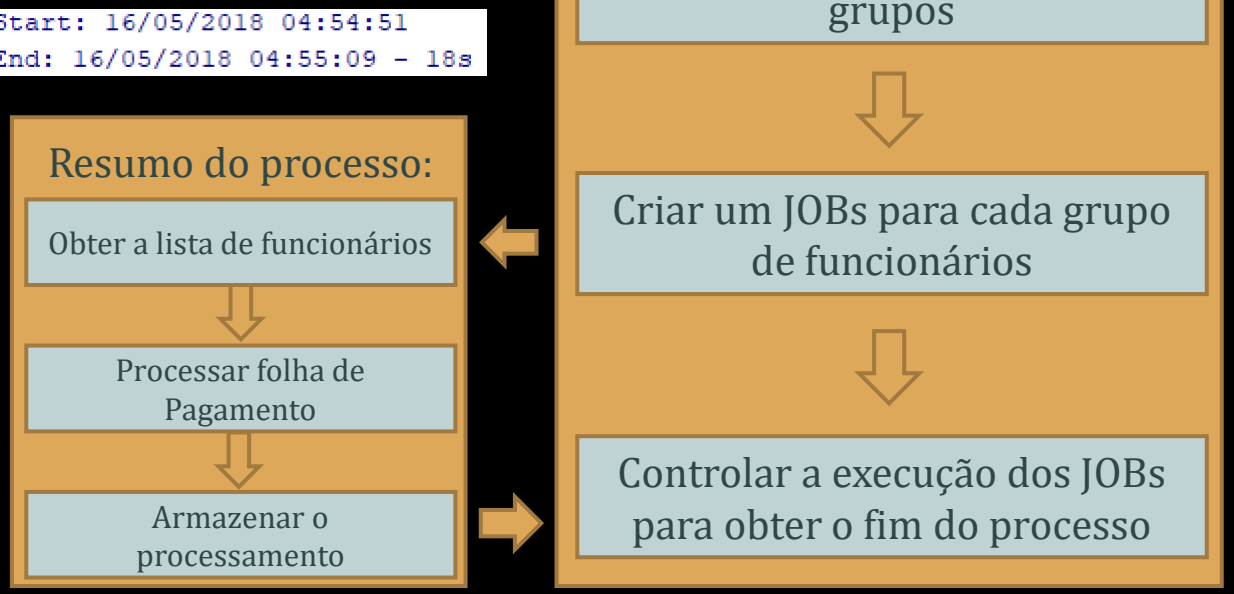
Armazenar o processamento

Para esse exemplo foi criada uma package representando um processamento de folha de pagamento.

(Criada no Schema HR da Oracle)

```
DECLARE
  vStartHour DATE;
BEGIN
  /*
  output to show the processing time
  */
  vStartHour := SYSDATE;
  dbms_output.put_line('Start: ' ||
                       TO_CHAR(vStartHour, 'dd/mm/yyyy hh24:mi:ss'));
  -- Call the procedure
  pk_payroll_parallel_ctrl.pr_calculate_month_payroll(ppayrollmonth => to_date('01/01/2005',
                                                                               'dd/mm/yyyy'));
  /*
  output to show the processing time
  */
  dbms_output.put_line('End: ' ||
                       TO_CHAR(SYSDATE, 'dd/mm/yyyy hh24:mi:ss') || ' - ' ||
                       ROUND((SYSDATE - vStartHour) * 24 * 60 * 60, 2) || 's');
END;
```

```
Start: 16/05/2018 04:54:51
End: 16/05/2018 04:55:09 - 18s
```



EXECUÇÃO PARALELA COM O DBMS_SCHEDULER.

- A package original foi alterada para aceitar um intervalo de IDs que servirão de filtro para o processamento.
- Uma nova package foi criada para controlar a criação/execução dos JOBs.

EXECUÇÃO PARALELA COM O DBMS_SCHEDULER. (CONTINUAÇÃO)

Package original

```
PROCEDURE pr_Calculate_month_payroll(pPayrollMonth IN DATE);
```

Package alterada para execução paralela

```
PROCEDURE pr_Calculate_Month_Payroll(pPayrollMonth     IN DATE,
                                     pJobNum           IN PLS_INTEGER,
                                     pEmployeeIDStart  IN NUMBER,
                                     pEmployeeIDEnd    IN NUMBER);
```

```
CURSOR cEmployees(pMonth IN TO_MONTH_INFO) IS
  SELECT e.employee_id,
         e.first_name || ' ' || e.last_name full_name,
         greatest(e.hire_date, pMonth.firstDay) month_start_date,
         e.salary,
         j.JOB_ID,
         j.JOB_TITLE
    FROM EMPLOYEES e, jobs j
   WHERE e.hire_date <= pMonth.lastDay
     AND j.JOB_ID = e.JOB_ID;
```

```
CURSOR cEmployees(pMonth           IN TO_MONTH_INFO,
                  pEmployeeIDStart IN NUMBER,
                  pEmployeeIDEnd   IN NUMBER) IS
  SELECT e.employee_id,
         e.first_name || ' ' || e.last_name full_name,
         greatest(e.hire_date, pMonth.firstDay) month_start_date,
         e.salary,
         j.JOB_ID,
         j.JOB_TITLE
    FROM EMPLOYEES e, jobs j
   WHERE e.hire_date <= pMonth.lastDay
     AND e.employee_id BETWEEN pEmployeeIDStart AND pEmployeeIDEnd
     AND j.JOB_ID = e.JOB_ID;
```

- Package original necessita de poucas alterações.

Teste realizado com 4 JOBs:

Original	4 JOBs
57 Segundos	18 Segundos

CONTROLE DAS SESSÕES

- Como acompanhar se os JOBs terminaram?
  - DBMS_PIPE;
  - DBMS_ALERT;
  - Manualmente (usando uma tabela de controle).

PONTOS DE ATENÇÃO.

- Avaliar a necessidade de controlar LOCKs de recursos/tabelas.
  - Teremos várias sessões acessando simultaneamente os mesmos recursos.

- A solução precisa ser avaliada em conjunto com o DBA.
  - O tempo total do processo tende a diminuir, porém, será exigido mais do servidor durante a execução, o que pode comprometer a performance geral do banco de dados.

MATERIAL DE APOIO

- DBMS_PIPE – https://docs.oracle.com/database/121/ARPLS/d_pipe.htm#ARPLS038D
- DBMS_ALERT - https://docs.oracle.com/database/121/ARPLS/d_alert.htm#ARPLS351
- DBMS_SCHEDULER - https://docs.oracle.com/database/121/ARPLS/d_sched.htm#ARPLS72235
- DBMS_LOCK - https://docs.oracle.com/cd/B19306_01/appdev.102/b14258/d_lock.htm#i1002309

- Fontes dos exemplos disponíveis em: https://github.com/RodrigoEdson/PL-SQL-CAMP

AGRADECEMOS A ATENÇÃO DE TODOS!!!

RODRIGO EDSON FERNANDES

- E-mail: rodrigoedson@gmail.com
- LinkedIn: https://www.linkedin.com/in/rodrigo-edson-fernandes/

FAUSTO SARKIS MIRA

- E-mail: fausto.mira@gmail.com
- LinkedIn: https://www.linkedin.com/in/faustosarkismira/

Grupo no Linkedin para o Meetup → PL/SQL CAMP