Procedimentos (Procedures)

PROFESSOR: VINICIUS CAMPOS

profvinicampos@gmail.com

Introdução

- Contribui com a tarefa de programar;
- Favorece a estruturação do programa;
- Facilita a modificação do programa;
- Melhora a legibilidade do programa;
- Divide o problema a ser resolvido em partes (modularização).

Procedimentos (Procedures)

Vantagens:

- Diferentes partes dos programas podem ser independentes de forma que possam ser escritas e testadas separadamente;
- Trechos dos programas podem ser escritos de forma a serem reusados em diferentes partes do programa;
- Permite que programas complexos possam ser montados a partir de unidades menores já prontas e testadas.

Procedimentos (Procedures)

- Uma forma de criar um subprograma;
- Ao contrário de funções não retornam um valor explicitamente;
- Quando um determinado conjunto de instruções tem que ser repetido dentro da solução de um problema, é conveniente colocá-lo dentro de um procedimento;

Procedure

Procedures tem facilidades para controlar o fluxo de comandos como:

- Sentenças IF : efetua um seletivo controle de ações baseado em condições;
- Repetições (Loops): para repetições de ações sem uma condição imposta;

Procedure

Procedures tem facilidades para controlar o fluxo de comandos como:

- Sentenças FOR(Loops): controla as repetições de ações utilizando um contador;
- Sentenças While (Loops): Controla as repetições baseado em condições.

Procedure

Para criar acesse:

- FILE → NEW→ PROGRAM WINDOW→ BLANK.

Procedure

```
CREATE OR REPLACE PROCEDURE PCEXEMPLO ( PCOD IN NUMBER)

IS
  VNOME VARCHAR2(100);
BEGIN

 SELECT A.ENAME INTO VNOME
 FROM EMP A
 WHERE A.EMPNO = PCOD;
 COMMIT;
END;
```

Procedure

A sintaxe básica de uma procedure é:

CREATE [OR REPLACE] PROCEDURE [schema.]nome_da_procedure
[(parâmetro1 [modo1] tipodedado1,
parâmetro2 [modo2] tipodedado2)]
IS|AS
Bloco PL/SQL

Procedure

- Mode → Identifica o tipo de argumento (IN/OUT/IN OUT);
- Tipo Argumento → Tipo do dado (Varchar2, Integer);
- IS ou AS→ Essas cláusulas são equivalentes, pode-se utilizar tanto uma quanto a outra. Por convenção utilizamos IS;

Procedure

- Bloco PL/SQL→ É o corpo da procedure que define as ações que serão executadas quando a procedure for executada;

- OBS: A cláusula **REPLACE** é utilizada quando a procedure já existe.

Procedure - Exemplo

```
CREATE OR REPLACE PROCEDURE PROXIMONUM(NUMERO OUT NUMBER)
IS

QTDE NUMBER;
BEGIN

      SELECT MAX(COD_CARDAPIO) INTO QTDE FROM TBCADCARDAPIOS;
      IF NVL(QTDE,0) <> 0 THEN
        NUMERO := QTDE +1 ;
      ELSE
         NUMERO := 1;
      END IF;

END PROXIMONUM;
```

Procedure - Exemplo

```
CREATE OR REPLACE PROCEDURE aumenta_sal (p_empno IN emp.empno%TYPE)
IS
BEGIN
  UPDATE
    scott.emp
  SET
    sal = sal * 1.10
  WHERE
    empno = p_empno;
END aumenta_sal;
```

Trocando valores entre diferentes ambientes através de argumentos

- In argumento→ passa o valor do ambiente chamador para a procedure (default);
- Out argumento→ retorna um valor da procedure para o ambiente chamador;
- IN OUT→ Passa um valor do ambiente chamador para a procedure, e a procedure retorna um valor para o ambiente chamador.

Criando procedure com argumento IN.

```
CREATE OR REPLACE PROCEDURE novos_empregados
(v_emp_no          in emp.empno%type,
 v_emp_name        in emp.ename%type,
 v_emp_job         in emp.job%type,
 v_mgr_no          in emp.mgr%type,
 v_emp_hiredate    in emp.hiredate%type,
 v_emp_sal         in emp.sal%type,
 v_emp_comm        in emp.comm%type,
 v_dept_no         in emp.deptno%type)

IS
BEGIN
  insert into emp (empno,ename, job, mgr, hiredate, sal, comm, deptno)
  values (v_emp_no,v_emp_name,v_emp_job,v_mgr_no,v_emp_hiredate,v_emp_sal,v_emp_comm, v_dept_no);
 commit;
END novos_empregados;
```

Criando procedure com argumento OUT.

```
CREATE OR REPLACE PROCEDURE pesquisa_empregados
(v_emp_no           in emp.empno%type,
 v_emp_name         out emp.ename%type,
 v_emp_sal          out emp.sal%type,
 v_emp_comm         out emp.comm%type)

IS
BEGIN
 select ename,sal,comm
 into v_emp_name, v_emp_sal,v_emp_comm
 from emp
 where empno = v_emp_no;
END pesquisa_empregados;
```

Passando um valor do ambiente chamador para a procedure, e a procedure retorna um valor para o ambiente chamador.

```
CREATE OR REPLACE PROCEDURE formata_telefone
(v_fone_no in out varchar2)

IS
BEGIN
 v_fone_no:= SUBSTR(v_fone_no,1,4)||'-'||
             SUBSTR(v_fone_no,5,4);
END formata_telefone;
```

O atributo %Type

```
CREATE OR REPLACE PROCEDURE pesquisa_empregados
(v_emp_no          in emp.empno%type,
 v_emp_name        out emp.ename%type,
 v_emp_sal         out emp.sal%type,
 v_emp_comm        out emp.comm%type)
```

O atributo %Type

- Utiliza pra herdar a característica de um campo;
- Monta uma consulta para buscar um único campo;

O atributo %Type

- Herda os valores de um campo da tabela;
- Esse atributo elimina a necessidade de alterar seu programa sempre que uma coluna for alterada, ele sempre terá o mesmo tipo de uma coluna da tabela.

O atributo %RowType

```
CREATE OR REPLACE PROCEDURE Insere

IS
 reg_fun emp%rowtype;
BEGIN
   select * into reg_fun
   from emp
   where empno = 1010;
END insere;
```

O atributo %RowType

- Herda a característica de vários campos de uma tabela uma única vez;
- Com este atributo, teremos uma variável com exatamente a mesma estrutura de uma tabela.

Gerenciando exceções em tempo de execução

- Pode-se gerenciar qualquer tipo de exceção em tempo de execução permitindo propagar para o ambiente chamador ou tomar ações quando essas acontecerem.

- RAISE_APPLICATION_ERROR (numero_erro, texto_erro).

numero_erro→ É o número do erro definido pelo usuário. Deve estar entre -20000 e -20999.

Gerenciando exceções em tempo de execução

```
CREATE OR REPLACE PROCEDURE exclui_funcionario
(v_emp_no in  emp.empno%type)

IS
BEGIN
 delete from emp
 where empno = v_emp_no;
  IF SQL%notfound then
   raise_application_error(-20000,'Funcionario não existe');
  end if;
  commit;

END exclui_funcionario;
```

```
CREATE OR REPLACE PROCEDURE exclui_funcionario2
(v_emp_no          in emp.empno%type,
 v_emp_name        in emp.ename%type,
 v_emp_job         in emp.job%type,
 v_mgr_no          in emp.mgr%type,
 v_emp_sal         in emp.sal%type)

IS
 v_emp_hiredate    emp.hiredate%type;
 v_emp_comm        emp.comm%type;
 v_dept_no         emp.deptno%type;

BEGIN

 select deptno
 into v_dept_no
 from emp
 where empno = v_mgr_no;

  insert into emp (empno,ename, job, mgr, hiredate, sal, comm, deptno)
  values (s_empno.nextval,v_emp_name,v_emp_job,v_mgr_no,v_emp_hiredate,v_emp_sal,v_emp_comm, v_dept_no);
 commit;

 exception
  when no_data_found then
   raise_application_error(-20000,'Gerente n?o ? um empregado v?lido');
END exclui_funcionario2;
```

Gerenciando Procedures

- Exibir todas as procedures e funções:

```
SELECT A.OBJECT_NAME, A.OBJECT_TYPE
FROM USER_OBJECTS A
WHERE A.OBJECT_TYPE IN ('PROCEDURE', 'FUNCTION')
ORDER BY A.OBJECT_NAME
```

Testando os Procedimentos

Teste Procedure

- FILE → NEW→ TEST WINDOW.

```
-- Created on 29/07/2015 by VINI
declare
   -- Local variables here
   i integer;
begin
   -- Test statements here

end;
```

Exercícios

1) Criar um procedimento para buscar na tabela de empregados o ENAME, JOB, SAL E COMM do empregado onde o EMPNO = 7839 e inserir esses dados na tabela BONUS.

2) Criar um procedimento para alterar a procedure acima dando UPDATE na tabela BONUS e alterando o valor do campo ENAME para seu nome.

Exercícios

3) Criar um procedimento para inserir seus dados na tabela EMP.

4) Criar um procedimento para inserir na tabela SALGRADE os dados 6, 4000 e 1600.

5) Criar um procedimento para deletar o registro da tabela EMP onde o EMPNO = 1010.

Bibliografia

Elmasri, Ramez
Sistemas de banco de dados/ Ramez Elmasri e Shamkant B. Navathe; revisor técnico Luis Ricardo de Figueiredo. –São Paulo: Pearson Addison Wesley, 2005.

Rob, Peter
Sistemas de banco de dados : projeto, implementação e gerenciamento / Peter Rob,

http://www.devmedia.com.br/pl-sql-functions-e-procedures/29882