PL SQL Functions e Procedures

Neste artigo trataremos da utilização das functions e das stored procedures no PL/SQL. Abordaremos suas principais características e apresentaremos alguns exemplos práticos e com isso apresentar as diferenças existentes entre eles.

Uma function é um bloco PL/SQL muito semelhante a uma procedure. O que podemos entender de início entre esses dois tipos de blocos é que os blocos functions retornam valores e as procedures podem ou não retornar um valor. As functions tem duas características que diferem das procedures, as quais não podemos deixar de tratar:

- As functions sempre retornam valores
- Functions são usadas como parte de uma expressão.

e para isso precisamos entender a sua sintaxe básica. Vejamos a seguir na **Listagem 1**.

Listagem 1. Estrutura básica de uma function.

```
CREATE [OR REPLACE] FUNCTION function_name
  [(parameter_name [IN | OUT | IN OUT] type [, ...])]
  RETURN return_datatype
  {IS | AS}
  BEGIN
     < function_body >
  END [function_name];
```

Vejamos então o que cada um desses termos representa:

- CREATE [OR REPLACE] FUNCTION: Caso uma function já exista com o mesmo nome, ela será reescrita devido ao termo ‘replace’. Caso contrário, ela será criada de acordo com o termo ‘create’.
- Function_name: Será o nome atribuído para essa função.
- Parameters: a lista opcional de parâmetros contém os nomes, os modos e os tipos que esses parâmetros terão. O IN representa o valor que será passado de fora, já o OUT representa que este parâmetro será utilizado para retornar um valor de fora do procedimento.
- Return_datatype: é o tipo de retorno que será utilizado, sendo este SQL ou PL/SQL. Podemos neste caso utilizar referências como o %TYPE ou %ROWTYPE se necessário, ou mesmo utilizar qualquer tipo de dados escalar ou composto.
- IS/AS: por convecção, temos o ‘is’ para a criação de funções armazenadas e o ‘as’ quando criamos pacotes (packages).
- function_body: contém o bloco PL/SQL que inicia com a claúsula BEGIN e finaliza com END [function_name], e executa neste momento todas as instruções necessárias.

Para facilitar o entendimento, vamos ver um exemplo de utilização que será recuperar o primeiro nome do funcionário de id igual a 90. Veja a **Listagem 2.**

```
  CREATE OR REPLACE FUNCTION primeiro_nome_func
  RETURN VARCHAR(20);
 IS
  emp_name VARCHAR(20);
BEGIN
SELECT primeiro_nome INTO emp_name
 FROM funcionarios_tbl WHERE ID = 90;
 RETURN emp_name;
END;
/
```

Neste exemplo estamos apenas recuperando o primeiro nome do funcionário com id 90 para a variável emp_name. O tipo de retorno dessa informação é do tipo Varchar2 e o seu retorno também será do mesmo tipo.

Este é um exemplo fictício de tabela de uma base de dados, mas que demonstra fielmente o que realmente é necessário entender sobre a criação de uma function.

Como executar a nossa function?

Podemos executar a função de várias formas. No caso da função retornar um valor, podemos declará-lo como uma variável, como da seguinte forma:

```
func_nome :=  primeiro_nome_func;
```

Ou então como parte de uma instrução select:

```
SELECT primeiro_nome_func FROM dual;
```

Ou também como uma instrução PL/SQL:

```
dbms_output.put_line(primeiro_nome_func);
```

a um novo exemplo que será responsável por definir, declarar e chamar uma simples função PL/SQL que irá computar e retornar o máximo entre dois números. Vejamos no código da **Listagem 3.**

Listagem 3. Exemplo de valor máximo entre dois números inteiros.

```
SET serveroutput ON;

DECLARE
  a NUMBER;
  b NUMBER;
  c NUMBER;

  FUNCTION max_entre_numeros(
      x IN NUMBER,
      y IN NUMBER)
    RETURN NUMBER
  IS
    z NUMBER;
  BEGIN
    IF x > y THEN
      z := x;
    ELSE
      Z:= y;
    END IF;
    RETURN z;
  END;
  BEGIN
    a := 23;
    b := 45;
    c := max_entre_numeros (a, b);
    dbms_output.put_line('Valor máximo obtido entre os valores ' || a ||
     ' e ' || b || ' foi ' || c);
END;
  /
```

Quando executamos esta função o valor retornado é 45, como é mostrado pelo prompt SQL:

```
bloco anônimo concluído
Valor máximo obtido entre os valores: 23 e 45 foi 45
```

realizar consultas sem a necessidade de extração de dados.

Funções recursivas PL/SQL

Muitas vezes precisamos utilizar um bloco de instruções dentro de outro bloco de instruções. Quando isso acontece, nos referimos a esta função como sendo uma função recursiva. Vamos realizar este procedimento com um exemplo que vai calcular o fatorial de um número qualquer, de acordo com a **Listagem 4**.

Listagem 4. Representação de uma função recursiva para calcular o fatorial de um número.

```
DECLARE
  num NUMBER;
  valor NUMBER;
  fatorial NUMBER;
  FUNCTION fact(
      x NUMBER)
    RETURN NUMBER
  IS
    f NUMBER;
  BEGIN
    IF x =0 THEN
      f := 1;
    ELSE
      f := x * fact(x-1);
    END IF;
    RETURN f;
  END;
  BEGIN
    num       := &valor;
    factorial := fact(num);
    dbms_output.put_line(' O valor fatorial de '|| num
     || ' é ' || fatorial);
  END;
  /
```

Quando executamos esta instrução num prompt SQL, primeiramente seremos questionados quanto ao valor do ‘num’ que será passado pelo usuário e após isso será retornado o

passar o valor 5 e termos como resultado o seguinte:

```
    bloco anônimo concluído
O valor fatorial de 5 é 120
```

Stored procedures PL/SQL

Uma stored procedure é um bloco de instruções PL/SQL que executa uma ou mais tarefas específicas. Elas são bem similares com as procedures de outras linguagens de programação.

Uma procedure normalmente possui um cabeçalho e um corpo. O cabeçalho consiste do nome e de parâmetros ou variáveis que serão passadas para a procedure. Já o corpo consiste da declaração de uma seção, execução de uma seção e uma seção de exceções muito similar a um bloco geral da PL/SQL. Uma procedure pode ou não ter um valor de retorno. Normalmente as procedures são criadas dentro de pacotes ou em blocos PL/SQL.

Podemos passar os parâmetros para uma procedure de três maneiras:

1. **Parâmetros IN** – passamos o valor na própria procedure.
2. **Parâmetros OUT** – recebemos o valor a partir da chamada de blocos externos.
3. **Parâmetros IN OUT** – passamos um valor inicial para a procedure e recebemos de volta uma atualização.

Criando stored procedures

Vejamos a sintaxe geral da criação de uma procedure para uma melhor compreensão, como segue na **Listagem 5**.

Listagem 5. Estrutura básica de criação de uma procedure.

```
    Declaration section
BEGIN
    Execution section
EXCEPTION
  Exception section
END;
```

O 'Is' marca o início do corpo de uma procedure e é bem similar ao DECLARE de um bloco anônimo PL/SQL. O código criado entre o IS e o BEGIN forma a seção de declaração da procedure.

A sintaxe entre os colchetes [] indica que é opcional. Mas de igual forma a uma função, a utilização do CREATE é para criar uma procedure e o REPLACE irá sobrepor uma procedure existente com novas informações de código.

Para melhorarmos o aprendizado para utilização deste recurso, vamos a um exemplo prático. O exemplo da **Listagem 6** será criar uma procedure chamada de 'detalhes_dos_funcionarios' que irá nos retornar informações relativas aos funcionários.

Listagem 6. Criação da procedure detalhes_dos_funcionarios.

```
CREATE OR REPLACE
PROCEDURE detalhes_dos_funcionarios
IS
  CURSOR emp_cur
  IS
    SELECT first_name, last_name, salary FROM emp_tbl;
  emp_rec emp_cur%rowtype;
BEGIN
  FOR emp_rec IN sales_cur
  LOOP
    dbms_output.put_line('Nome do funcionário: ' || emp_cur.first_name ||
     '. Sobrenome do funcionário: ' ||emp_cur.last_name ||
     '. Salário do funcionário:
     ' ||emp_cur.salary);
  END LOOP;
END;
/
```

uma base de dados de exemplo da Oracle.

Vejamos agora um novo exemplo que será utilizado para retornar o valor mínimo entre dois números inteiros. Observe a **Listagem 7.**

Listagem 7. Criação de uma procedure para retornar o número mínimo utilizando o IN OUT.

```
DECLARE
   a number;
   b number;
   c number;

PROCEDURE valorMinimo(x IN number, y IN number, z OUT number) IS
BEGIN
   IF x < y THEN
      z:= x;
   ELSE
      z:= y;
   END IF;
END;

BEGIN
   a:= 23;
   b:= 45;
   valorMinimo(a, b, c);
 dbms_output.put_line('O valor mínimo obtido entre os valores ' || a
 || ' e ' || b || ' foi ' || c);
END;
/
```

Como executamos uma strored procedure? Para este ponto, a resposta é bem simples, pois temos duas maneiras de chamar uma stored procedure:

1. Usando a palavra-chave EXECUTE;
2. Chamando o nome da procedure de um bloco PL/SQL.

No primeiro caso, executamos da seguinte forma:

```
EXECUTE detalhes_dos_funcionarios;
```

```
BEGIN
detalhes_dos_funcionarios
END;
/
```

Deletando uma stored procedure

Para deletarmos uma procedute, utilizamos a instrução DROP PROCEDURE. Vejamos a seguir a sintaxe para a exclusão:

```
DROP PROCEDURE nome_da_procedure;
```

Então podemos utilizar esta instrução também dentro de um bloco PL/SQL e realizar sua exclusão.

Com isso finalizamos este artigo, onde pudemos avaliar conceitos importantes com relação ao uso das functions e também ao uso das stored procedures, além de visualizar na prática estes recursos e ter uma boa noção de sua utilização e de suas diferenças.

Até a próxima!

Publicado no Canal Banco de dados

por Edson José
Banco de dados & BigData expert

Post aqui sua dúvida ou comentário que nossa equipe responderá o mais rápido possível.

Alane Freitas.
Edson parabéns pelo post, bem objetivo e fácil entendimento.

há +1 ano

[autor] Edson Dionisio
Obrigado Alane pelo contato, fico feliz por você ter gostado.

há +1 ano

Allan Silvestre
Estou criando minha procedure com Join, mas esta com erro.
Criei IP. IT E PT, acredito que isso que esta errado.

```
CREATE OR REPLACE PROCEDURE SP_LISTA_ITENS_PEDIDO
(
CURSOR_OUT OUT Types.ref_cursor
)
IS
BEGIN
OPEN CURSOR_OUT FOR
SELECT IT.I_ITEMID, IT.I_DESCRICAO, IT.I_VALOR, IP.I_ITEMID
FROM ITEM_TABLE IT
INNER JOIN ITENS_PEDIDO IP
ON IT.I_ITEMID = IP.I_ITEMID
INNER JOIN PEDIDO TABLE PT
ON IP.P_ORDERID = PT.P_ORDERID
WHERE PT.ORDERID = ?;
END;
```

há +1 mês

Douglas
Opa Allan, tudo bem?

Contactamos o autor e estamos verificando o seu código.
O seu tópico será respondido em até 24 horas.

Um forte abraço.

há +1 mês

Estamos aqui de olho no post! Não esquecemos de você.
Ainda estamos analisando a sua dúvida.

Um forte abraço.

há +1 mês

Julio Sampaio CONSULTOR

Olá Allan,

E qual o erro que dá, poderia postar aqui?

Abraço

há +1 mês

Adicionar um comentário...

André Santos

Boa noite,

gostaria de saber o por que no seu tem ponto e virgula (RETURN VARCHAR(20);) e quando eu faço igual no meu da erro.
quando eu tiro esse ponto e virgula o erro some.

há +1 mês

Douglas Claudio

Olá André, obrigado pelo seu comentário.

Enviamos sua solicitação ao Edson e estamos no aguardo de um feedback do mesmo.

Um abraço.

há +1 mês

[autor] Edson Dionisio

Olá André,
você está utilizando algum software como o JDeveloper para criar os exemplos? ou está utilizando o SQL*PLus?
Porquê isso pode ser questão do programa em uso, pois não vejo razão para não funcionar. Outra coisa, de acordo com a própria documentação, faz parte da estrutura o ponto-e-virgula..não sei porquê pra você não está pegando.

há +1 mês

Alexandre Abreu

Olá. estou precisando de uma ajuda.

Quando execute a procedure abaixo .

```
CREATE OR REPLACE PROCEDURE P_SUM
IS
xml CLOB;
comp sys.dbms_xmlgen.ctxHandle;
txSQL varchar2(800);
trans char(1);
BEGIN
trans:='A';

txSQL := 'SELECT
MA.NOM_MARCA,NOM_COR,CO.DSC_COMBUSTIVEL,DSC_TRANSMISSAO
FROM VEICULO V
INNER JOIN MODELO M ON V.COD_MODELO = M.COD_MODELO
INNER JOIN CARRO C ON V.COD_CARRO = C.COD_CARRO
INNER JOIN MARCA MA ON C.COD_MARCA = MA.COD_MARCA
INNER JOIN COMBUSTIVEL CO ON V.COD_COMBUSTIVEL = CO.COD_COMBUSTIVEL
INNER JOIN COR COR ON V.COD_COR = COR.COD_COR
INNER JOIN TRANSMISSAO T ON T.COD_TRANSMISSAO = V.COD_TRANSMISSAO
INNER JOIN LOJA L ON V.COD_LOJA = L.COD_LOJA
WHERE MA.COD_MARCA = 9 AND COR.COD_COR IN(1,2)
AND V.COD_COMBUSTIVEL = 4 AND DSC_TRANSMISSAO = `A`';
comp := sys.dbms_xmlgen.newContext(txSQL);
xml := sys.dbms_xmlgen.getXml(comp);
dbms_output.put_line(xml);
END P_SUM;
```

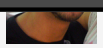
da o seguinte erro

```
Relatório de erros -
ORA-19202: Ocorreu um erro no processamento XML
ORA-00911: caractere inválido
ORA-06512: em "SYS.DBMS_XMLGEN", line 7
ORA-06512: em "SYS.DBMS_XMLGEN", line 147
ORA-06512: em "ALEXANDRE.P_SUM", line 22
ORA-06512: em line 1
19202. 00000 - "Error occurred in XML processing%s"
*Cause: An error occurred when processing the XML function
*Action: Check the given error message and fix the appropriate problem
```

e eu já tentei tudo e não consigo resolver. Poderia me ajudar?

há +1 ano

[autor] Edson Dionisio



há +1 ano

Adicionar um comentário...

Mais posts

Microartigo

Introdução à criptografia no MySQL

Artigo

Comandos básicos em SQL - insert, update, delete e select

Guia

Guia de Referência NoSQL com MongoDB

Vídeo aula Microcurso

Configurando o serviço - Curso Trabalhando com dois servidores Firebird - Aula 3

Vídeo aula Microcurso

Configurando o ambiente - Curso Trabalhando com dois servidores Firebird - Aula 2

Listar mais conteúdo