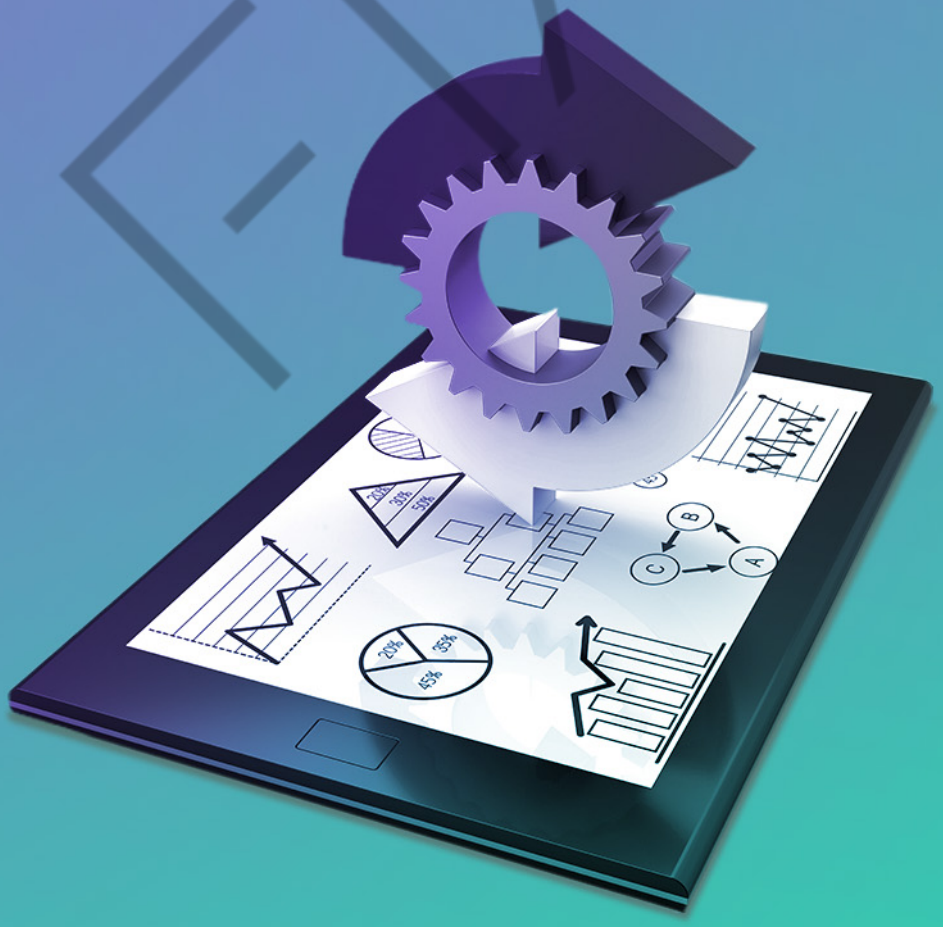
DATABASE PROGRAMMING

MUITO A **PROCESSAR ANTES DE PERSISTIR**



7

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

1 MUITO A PROCESSAR ANTES DE PERSISTIR

Chegou o momento em que vamos otimizar alguns processos, deixando-os mais rápidos e, por meio de *stored procedures,* conseguimos ler, manipular e atualizar os dados de uma só vez. Temos muito a processar antes de persistir!

1.1 Bloco PL/SQL nomeado

Os programas que temos desenvolvido até agora são chamados de blocos PL/SQL anônimos. Têm essa denominação porque não estão associados a nenhum nome. Normalmente, são salvos em arquivos-texto e se precisarmos executá-los novamente, será necessário carregá-los a partir do arquivo-texto e recompilá-los.

Ao usarmos blocos PL/SQL nomeados, passamos a ter uma série de vantagens. Por exemplo: procedimentos e funções são armazenados no banco de dados em formato compilado. Caso ocorram modificações em seus objetos dependentes, haverá necessidade de recompilar o código.

Outra vantagem de armazenarmos os procedimentos e funções no banco de dados em formato compilado e nomeado é que outras aplicações e/ou usuários podem executá-los, desde que possuam os privilégios e as autorizações para tanto. Além disso, podemos passar parâmetros para os programas e, no caso das funções, podemos obter um retorno.

Para Feuerstein e Pribyl (2014), quando criamos e compilamos um procedimento, o banco de dados Oracle armazena automaticamente uma série de informações, como o nome do objeto, o código-fonte, o pseudocódigo e os códigos de erro.

Para Puga, França *e* Goya (2015), um procedimento ou PROCEDURE é um conjunto de instruções que realizam determinada tarefa. Podem ser executadas a partir do SQL*Plus, de outros procedimentos, das funções, de outros aplicativos pré-compilados ou de ferramentas, como o SQL*Developer.

Para Dillon *et al.* (2013), o nome do objeto identifica um procedimento. O nome do procedimento, pacote, função ou corpo do pacote pode ser definido por meio dos

comandos CREATE PROCEDURE, CREATE FUNCTION, CREATE PACKAGE ou CREATE PACKAGE BODY.

Para Feuerstein e Pribyl (2014), o compilador PL/SQL analisa o código-fonte que digitou e produz uma representação da análise do código-fonte. Essa análise é chamada de PARSE TREE ou árvore de análise.

Para a Oracle (2016), o pseudocódigo ou P-CODE é gerado pelo compilador PL/SQL baseado no código analisado ou PARSED CODE. O PL/SQL executa o P-CODE quando solicitamos a execução do procedimento, função ou pacote.

Durante a compilação de um pacote, procedimento ou função, podem ocorrer erros. Esses erros são exibidos para o desenvolvedor por meio de mensagens denominadas de mensagens de erro ou ERROR MESSAGES.

Tanto o P-CODE quanto a PARSE TREE de um procedimento, função ou pacote são armazenados no banco de dados para evitar que sejam recompilados desnecessariamente.

O fato de o P-CODE estar armazenado no banco de dados permite que seja copiado para a memória CACHE do servidor em uma área denominada de SHARED POOL. A SHARED POOL faz parte de uma área de memória denominada SGA ou SYSTEM GLOBAL AREA.

Estando em memória, o código pode ser executado rapidamente e permite que seja executado por vários usuários, desde que tenham permissão para isso. Na primeira vez que o código é executado, é lido do disco e armazenado na memória. Na próxima vez que um usuário precisar desse código, estará na memória e poderá ser acessado mais rapidamente do que tivesse que ser acessado em disco.

A versão compilada do código permanece em memória baseada em um algoritmo de LRU, ou LEAST RECENTLY USED, em tradução direta, MENOS RECENTEMENTE USADO. Esse algoritmo garante que os códigos que não estão sendo usados serão descartados da memória.

Os códigos-fonte P-CODE e PARSE TREE são armazenados no dicionário de dados do banco de dados. O dicionário de dados fica armazenado no TABLESPACE SYSTEM do banco de dados. Um TABLESPACE é um nome lógico para um ou mais arquivos físicos do banco de dados.

1.2 O procedimento

Para Puga, França e Goya (2015), similar a outras linguagens de programação, um procedimento ou PROCEDURE envolve basicamente os passos de identificação do procedimento, definição dos parâmetros ou parâmetro, definição do conjunto de instruções do procedimento e submissão do código ao SGBDR.

Após a execução desses passos, o código-fonte é armazenado no dicionário de dados e o procedimento é compilado.

É importante notar que, se a compilação é bem-sucedida, o P-Code é armazenado no dicionário de dados e só pode ser consultado pelo SGBDR. O usuário final não tem acesso ao P-Code.

É sempre bom lembrar que, caso ocorram erros de compilação, serão armazenados no USER_ERRORS e podem ser consultados pelo usuário final.

```
CREATE [ OR REPLACE] PROCEDURE nome_procedimento
[parâmetro [{in, out, in out}] tipo_parâmetro,
  ...
{IS ou AS}

BEGIN
corpo_do_procedimento

END [nome_procedimento];
/
```

Código-fonte 1 – Sintaxe da criação de uma PROCEDURE
Fonte: ORACLE (2016)

CREATE OR REPLACE é a instrução para a criação ou a substituição do procedimento.

nome_procedimento é o nome que será dado ao procedimento.

parâmetro [parâmetro [{in, out, in out}] é nome do parâmetro que poderá ser de entrada, saída ou entrada e saída.

tipo_parâmetro é o tipo de dado que o parâmetro poderá aceitar. Os parâmetros podem ser IN, OUT ou IN OUT.

IS ou AS têm a mesma função e indicam o bloco que estará associado ao procedimento, substitui a palavra reservada DECLARE.

BEGIN corpo_do_procedimento END são, respectivamente, o início do bloco, o conjunto de instruções do procedimento e o final do bloco.

Vejamos um exemplo simples de criação de procedimento ou PROCEDURE:

```
SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE quadrado
(p_num IN NUMBER :=0)
IS
BEGIN
DBMS_OUTPUT.PUT_LINE (p_num*p_num );
END quadrado;
/

EXECUTE quadrado(5);
```

Código-fonte 2 – Exemplo de criação e execução de um procedimento
Fonte: Elaborado pelo autor (2017)

O exemplo acima cria um procedimento denominado QUADRADO, que recebe um número qualquer e exibe o resultado desse número multiplicado por ele mesmo, ou seja, calcula o quadrado do número informado. No exemplo, testamos o procedimento com o valor 5, que deve retornar o valor de 25.

ATENÇÃO: É preciso fazer a criação da PROCEDURE antes de executá-la. Caso receba o erro PLS-00201: *identifier* 'QUADRADO' *must be declared*, reexecute a criação da PROCEDURE.

1.3 Parâmetros

Antes de começarmos a usar os parâmetros, é sempre bom relembrarmos os conceitos de parâmetros.

Para Puga, França e Goya (2015), parâmetro é um valor constante ou variável passado de uma rotina chamadora para uma rotina executora. Uma **rotina chamadora** é um algoritmo que usa as funcionalidades da *rotina executora*. Trataremos de dois tipos de parâmetros: os formais e os reais.

Um **parâmetro formal** são as variáveis da rotina executora que recebem os valores da rotina chamadora, isto é, recebem os parâmetros reais. Normalmente, as

variáveis dos parâmetros formais não devem ter tamanho ou precisão predeterminados.

Um parâmetro real são os valores, constantes ou reais, passados da rotina chamadora para a **rotina executora**.

1.3.1 Utilizar parâmetros de entrada

Para a Oracle (2016), por padrão, os parâmetros de um procedimento são do tipo entrada ou IN, isto é, são utilizados para a entrada de valores que serão utilizados internamente pelo procedimento.

Veja o exemplo abaixo:

```
CREATE OR REPLACE PROCEDURE reajuste
(v_codigo_emp IN emp.empno%type,
v_porcentagem IN number)
IS
BEGIN
UPDATE emp
     SET sal = sal + (sal *( v_porcentagem / 100 ) )
WHEREempno = v_codigo_emp;
  COMMIT;
END reajuste;
/
```

Código-fonte 3 – Exemplo de uso de parâmetros de entrada
Fonte: Oracle (2016), adaptado pelo autor (2017)

No exemplo, o procedimento REAJUSTE recebe os parâmetros V_CÓDIGO e V_PORCENTAGEM, cujos valores serão utilizados para a alteração de um registro na tabela emp. Vamos testar o nosso procedimento:

```
SELECT empno, sal
  FROM emp
 WHERE empno = 7839;

EXECUTE reajuste(7839, 10);

SELECT empno, sal
  FROM emp
 WHERE empno = 7839;
```

Código-fonte 4 – Teste do procedimento REAJUSTE
Fonte: Oracle (2016), adaptado pelo autor (2017)

O nosso teste começa exibindo o salário do funcionário 7839. Em seguida, o procedimento REAJUSTE é executado com os parâmetros de entrada 7839 e 10. O valor 7839 é atribuído a V_CODIGO_EMP e o valor 10 é atribuído a V_PORCENTAGEM. O procedimento, então, atualiza o salário do funcionário 7839 em 10%. Após a execução do procedimento, consultamos o salário atualizado do funcionário 7839.

Os parâmetros de entrada ou IN podem receber valores-padrão ou DEFAULT. Vejamos o mesmo programa acima, mas, desta vez, com um valor-padrão para o parâmetro V_PORCENTAGEM:

```
CREATE OR REPLACE PROCEDURE reajuste
(v_codigo_emp IN emp.empno%type,
v_porcentagem IN number DEFAULT 25)
IS

BEGIN
     UPDATE emp
            SET sal = sal + (sal *( v_porcentagem / 100 ) )
            where empno = v_codigo_emp;
            COMMIT;
END reajuste;
/
```

Código-fonte 5 – Exemplo de uso de parâmetros de entrada com valor-padrão
Fonte: Oracle (2016)

O programa continua recebendo dois valores de entrada, mas, desta vez, V_PORCENTAGEM tem o valor-padrão de 25. É importante notar que parâmetros de entrada ou IN recebem valores-padrão. O parâmetro de saída ou OUT e o de entrada e saída ou IN OUT não devem receber valores-padrão. Vamos testar o nosso procedimento alterado:

```
SELECT empno, sal
  FROM emp
 WHERE empno = 7839;

EXECUTE reajuste(7839);

SELECT empno, sal
  FROM emp
 WHERE empno = 7839;
```

Código-fonte 6 – Teste do procedimento REAJUSTE com valores-padrão
Fonte: Oracle (2016), adaptado pelo autor (2017)

O nosso teste começa exibindo o salário do funcionário 7839. Em seguida, o procedimento REAJUSTE é executado com os parâmetros de entrada 7839. Note que, desta vez, não informamos o valor do percentual de reajuste. O valor 7839 é atribuído a V_CODIGO_EMP e o valor 25 é assumido para V_PORCENTAGEM. O procedimento, então, atualiza o salário do funcionário 7839 em 25%. Após a execução do procedimento, consultamos o salário atualizado do funcionário 7839. Perceba que o programa assumiu o valor de 25, porque nenhum valor foi informado para V_PORCENTAGEM no momento da chamada do programa e assumiu o valor-padrão para esse parâmetro.

1.3.2 Utilizar parâmetros de saída

Para a Oracle (2016), os parâmetros de saída ou, simplesmente, OUT são utilizados para a saída de valores processados para o ambiente de chamada. Vejamos em um exemplo simples:

```
CREATE OR REPLACE PROCEDURE consulta_emp
(p_id IN emp.empno%TYPE,
 p_nome OUT emp.ename%TYPE,
 p_salario OUT emp.sal%TYPE)
IS
BEGIN
    SELECT ename, sal INTO
           p_nome, p_salario
      FROM emp
     WHERE empno  = p_id;
END consulta_emp;
/
```

Código-fonte 7 – Exemplo de uso de parâmetros de saída
Fonte: ORACLE (2016), adaptado pelo autor (2017)

O exemplo do procedimento CONSULTA_EMP recebe o parâmetro de entrada P_ID, que será utilizado na condição da consulta para recuperar o registro de um funcionário. O parâmetro de saída P_SALÁRIO será usado para devolver o valor, resultado de uma operação interna do procedimento, para a *rotina chamadora*. Vamos testar nosso procedimento usando um bloco PL/SQL anônimo:

```
SET SERVEROUTPUT ON

DECLARE
   v_nome    emp.ename%TYPE;
   v_salario emp.sal%TYPE;
BEGIN
   consulta_emp(7839, v_nome, v_salario);
   DBMS_OUTPUT.PUT_LINE(v_nome);
   DBMS_OUTPUT.PUT_LINE(v_salario);
END;
/
```

Código-fonte 8 – Teste do procedimento CONSULTA_EMP
Fonte: ORACLE (2016), adaptado pelo autor (2017)

O nosso bloco PL/SQL anônimo define duas variáveis, V_NOME e V_SALARIO, que receberão o valor de saída do procedimento CONSULTA_EMP. Executa o procedimento passando o valor de entrada 7839 e recebe o valor dos parâmetros de saída nas duas variáveis previamente definidas. Em seguida, exibe o valor atual das variáveis.

1.3.3 Utilizar parâmetros de entrada e saída

Para a Oracle (2016), os parâmetros IN OUT ou de entrada e saída são utilizados para a entrada de valores, que poderão ser processados. O parâmetro poderá ser alterado e o seu valor pode ser devolvido para o ambiente de chamada.

O parâmetro é de entrada quando passa para o procedimento o valor que será utilizado no processamento e é de saída quando recebe o resultado do processamento e devolve o resultado à *rotina chamadora*. Note que, normalmente, o valor de entrada é alterado pelo procedimento antes de retornar para a *rotina chamadora.* Vejamos isso em um exemplo simples:

```
CREATE OR REPLACE PROCEDURE formata_fone
(p_fone IN OUT VARCHAR2)
IS
BEGIN
    p_fone := ' (' || SUBSTR(p_fone, 1, 3) || ') ' ||
SUBSTR(p_fone, 4, 4) || '- ' || SUBSTR(p_fone, 8);
END formata_fone;
/
```

```
```

Código-fonte 9 – Exemplo de uso de parâmetros de entrada e saída
Fonte: ORACLE (2016), adaptado pelo autor (2017)

O procedimento FORMATA_FONE recebe o parâmetro P_FONE como entrada. Essa entrada representa uma cadeia de caracteres. Esse parâmetro recebe o resultado das operações do procedimento e devolve esse valor à *rotina chamadora*. Perceba que o parâmetro P_FONE foi usado como entrada e saída. Vamos testar nosso procedimento usando um bloco PL/SQL anônimo:

```
SET SERVEROUTPUT ON

DECLARE
   v_fone VARCHAR2(30) := '01138858010';
BEGIN
   Formata_fone(v_fone);
   DBMS_OUTPUT.PUT_LINE(v_fone);
END;
/
```

Código-fonte 10 – Teste do procedimento FORMATA_FONE
Fonte: ORACLE (2016), adaptado pelo autor (2017)

O nosso bloco PL/SQL anônimo define a variável V_FONE com o valor inicial de 01138858010. O procedimento FORMATA_FONE recebe essa variável, formata o valor recebido e devolve o valor formatado para a mesma variável, que é exibida em seguida.

1.4 Visualização de erros de compilação

Segundo Puga, França e Goya (2015), existem várias técnicas para exibir os erros de compilação, dependendo do ambiente de trabalho do desenvolvedor. Uma forma simples de exibirmos os erros de compilação é pelo comando SHOW ERRORS.

```
SHOW ERRORS

Erros para PROCEDURE REAJUSTE:

LINE/COL ERROR
-------- -----------------------------------------
8/2      PL/SQL: SQL Statementignored
```

```
10/26    PL/SQL: ORA-00904: nome inválido de coluna
```

Código-fonte 11 – Exemplo de uso do comando SHOW ERRORS
Fonte: Elaborado pelo autor (2017)

No exemplo, ao executarmos o comando SHOW ERRORS, foram exibidas duas linhas com os erros de compilação do programa. Existem outras formas de consultar os erros. A VIEW USER_ERROS também pode ser consultada para verificá-los.

Vejamos um exemplo de uso, para efeito de demonstração, criaremos um procedimento simples com erro.

```
CREATE OR REPLACE PROCEDURE errotst AS
    v_conta NUMBER;
BEGIN
    v_conta := 7
END errotst;
/


SELECT line, position, text
  FROM user_errors
 WHERE name = 'ERROTST'
 ORDER BY sequence;

 LINE POSITION TEXT
----- -------- ------------------------------------------
---------------
    5        1 PLS-00103: Encountered the symbol "END" when
expecting one of the following:
                  * & = - + ; < / > at in is mod remainder
not rem <an exponent (**)> <> or != or ~= >= <= <> and or
like LIKE2_ LIKE4_ LIKEC_ between || multiset member
SUBMULTISET_ The symbol ";" was substituted for "END" to
continue.
```

Código-fonte 12 – Exemplo de uso da VIEW USER_ERRORS
Fonte: Elaborado pelo autor (2017)

No exemplo acima, o procedimento ERROTST foi criado com erro propositalmente para que possamos demonstrar o uso da VIEW USER_ERRORS. Durante a digitação, faltou um ponto e vírgula após o número 7. O compilador encontrou o comando END e concluiu que houve um erro. O erro foi armazenado na VIEW e pode ser visualizado por meio de uma consulta.

1.5 Passagem de parâmetros

Segundo Puga, França e Goya (2015), as formas mais comuns de passagem de parâmetros são: por posição, por identificação ou combinada.

- **Por posição**: neste caso, os parâmetros reais são listados de acordo com a ordem dos parâmetros formais.
- **Por identificação**: neste caso, os parâmetros reais são precedidos da identificação do parâmetro formal, podendo ser listados arbitrariamente.
- **Combinada:** neste caso, os parâmetros passados por posição devem ocupar os primeiros lugares da lista.

Vejamos um exemplo simples para demonstrar a passagem de parâmetros:

```
CREATE OR REPLACE PROCEDURE incluir_dept
(p_cod  IN dept.deptno%TYPE DEFAULT '50',
 p_nome IN dept.dname%TYPE DEFAULT 'FIAP',
 p_loc  IN dept.loc%TYPE DEFAULT 'SP')
IS
BEGIN
  INSERT INTO dept(deptno, dname, loc)
  VALUES(p_cod, p_nome, p_loc);
END incluir_dept;
/
```

Código-fonte 13 – Exemplo de procedimento com entrada de parâmetros
Fonte: Oracle (2016), adaptado pelo autor (2017)

O procedimento acima pode ser usado para incluir dados na tabela DEPT. Note que os valores-padrão para os campos foram definidos como 50, ‘FIAP’ e ‘SP’. Lembre-se, quando estiver executando seus testes, lembre-se de que o número do departamento é chave primária e não admite dados em duplicidade. Vamos demonstrar a passagem de parâmetros, usando um bloco anônimo para executar o procedimento INCLUIR_DEPT.

```
BEGIN
   incluir_dept;
END;
/
```

Código-fonte 14 – Teste de passagem de parâmetros para o procedimento INCLUIR_DEPT
Fonte: Oracle (2016), adaptado pelo autor (2017)

Neste exemplo, não estamos passando valores para os parâmetros. Neste caso, os valores-padrão serão incluídos na tabela.

```
BEGIN
   incluir_dept (55, 'Onze', 'SC');
END;
/
```

Código-fonte 15 – Teste de passagem de parâmetros
posicional para o procedimento INCLUIR_DEPT
Fonte: Oracle (2016), adaptado pelo autor (2017)

Neste exemplo, estamos passando valores de forma posicional. Os valores definidos quando o procedimento for executado serão recebidos segundo a posição em que os parâmetros foram definidos no procedimento. No caso, o número 55 será atribuído para P_COD; ‘Onze’ será atribuído para P_NOME; e ‘SC’ será atribuído para P_LOC.

```
BEGIN
   incluir_dept (p_cod => 60, p_nome => 'Doze', p_loc =>
'RJ');
END;
/
```

Código-fonte 16 – Teste de passagem de parâmetros
por identificação para o procedimento INCLUIR_DEPT
Fonte: Oracle (2016), adaptado pelo autor (2017)

Neste exemplo, estamos passando valores por identificação. O sinal de atribuição de valores é representado por ‘**=>**’. Os valores são atribuídos no momento da chamada do procedimento. No caso, o número 60 será atribuído para P_COD; ‘Doze’ será atribuído para P_NOME; e ‘RJ’ será atribuído para P_LOC.

```
BEGIN
   incluir_dept (65, p_nome => 'Treze');
END;
/
```

Código-fonte 17 – Teste de passagem de parâmetros
por combinação para o procedimento INCLUIR_DEPT
Fonte: Oracle (2016), adaptado pelo autor (2017)

Neste exemplo, estamos passando valores por combinação das técnicas posicional e por identificação. No caso, o número 65 será atribuído para P_COD; ‘Treze’ será atribuído para P_NOME; e P_LOC assumirá o valor-padrão.

CONCLUSÃO

Na prática, tudo no banco de dados acaba se tornando um procedimento ou função, pois blocos anônimos não são práticos de utilizar ou armazenar. O procedimento fica pronto para uso e a possibilidade de passar e receber parâmetros torna o procedimento flexível o bastante para ser reaproveitado sempre que for necessário realizar um procedimento similar.

REFERÊNCIAS

DILLON, S.; BECK, C.; KYTE, T.; KALLMAN, J.; ROGERS, H. **Beginning Oracle Programming**. USA: Apress, 2013.

FEUERSTEIN, S.; PRIBYL, B. **Oracle Pl/SqlProgramming**. 6 ed. California, USA: O'Reilly Media, 2014.

ORACLE. **Oracle Database:** PL/SQL Language Reference 12c Release 2 (12.2) B28370-05. USA: Oracle Press, 2016.

PUGA, S.; FRANÇA, E.; GOYA, M. **Banco de dados**. São Paulo: Pearson, 2015.