PL SQL

Prof. Lúcio Camilo
Email: luciocamilo@gmail.com
http://www.itnerante.com.br/profile/LucioCamilo

ITnerante

PROVAS DE TI
TUDO PARA VOCÊ PASSAR

Lúcio Camilo

- Resumo – CV
- Analista de Sistemas TCE/RJ
- Pós Graduado em Engenharia de Software
- Autor do Livro “Android para Desenvolvedores”, Editora Brasport
- MBA Gerenciamento de Projetos
- Certificações Profissionais:
  - SCJP, OCWD, OCJA Part I
  - RHSA, Big IP Essentials e Advanced

Contatos:

- luciocamilo@gmail.com
- www.itnerante.com.br/profile/luciocamilo

Conteúdo

- Histórico;
- Estrutura da Linguagem;
- Abstração de Dados;
- Variáveis e constantes;
- Cursores Manipulação de erros;
- Estruturas de Controle
- Subprogramas
- Códigos e mais códigos..

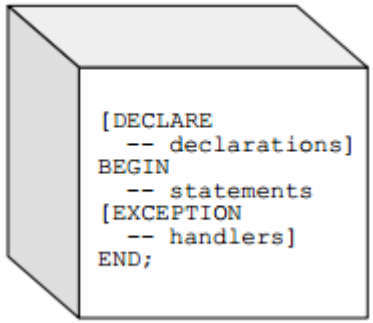
PL/SQL Oracle

- “Procedural Language extensions to the Structured Query Language”;
- Iniciativa da Oracle para sobrepor algumas limitações do padrão SQL;
- Padrão dos Bancos de Dados Oracle – “Write once, run everywhere”;
- Linguagem de fácil aprendizado e leitura;
- Linguagem Embutida;
- Performance e integração;

Histórico

- Oracle Version 6.0 – “Procedural Option”
- SQL Forms v3.0
- Restrições Iniciais;
- Fraquezas
  - Portabilidade
  - Autoridade de Execução
- Melhorias Constantes;

Estrutura de Bloco PL/SQL




```
<< label >> (optional)
DECLARE     -- Declarative part (optional)
  -- Declarations of local types, variables, & subprograms

BEGIN       -- Executable part (required)
  -- Statements (which can use items declared in declarative part)

[EXCEPTION -- Exception-handling part (optional)
  -- Exception handlers for exceptions (errors) raised in executable part]
END;
```

Estrutura de Bloco PL/SQL

- Declaration
  - Tipos de Dados
  - Estruturas
  - Variáveis
- Execution
  - Instruções
  - PL/SQL e SQL
- Exception
  - Tratamento de erros;
  - Manipulação de exceções;

Exemplo de Código

```
1  declare
2     v_string_texto varchar2(256) := 'Hello World';
3  begin
4     dbms_output.put_line(v_string_texto);
5  end;
```

```
SQL>  declare
  2      v_string_texto varchar2(256) := 'Hello World';
  3   begin
  4      dbms_output.put_line(v_string_texto);
  5   end;
  6   /
Hello World
PL/SQL procedure successfully completed.
SQL>
```

Variáveis e Constantes

- Declarando variáveis:

```
SQL>  DECLARE
  2      numero      NUMBER(6);          --SQL
  3      nome            VARCHAR2(20);   --SQL
  4      tem_estoque     BOOLEAN;        --PL/SQL
  5      preco           NUMBER(6,2)     --SQL
  6      descricao       VARCHAR2(50)    --SQL
  7  BEGIN
  8      NULL;
  9  END;
```

- Atribuição de valores:

```
SQL>  DECLARE
  2      numero          NUMBER(6) :=40;          --SQL
  3      nome            VARCHAR2(20) := 'Nome'; --SQL
  4      tem_estoque     BOOLEAN;                 --PL/SQL
  5      preco           OUT NUMBER(6,2)          --SQL
  6      descricao       VARCHAR2(50)             --SQL
  7  BEGIN
  8      SELECT valor into preco
  9         From Produtos
 10            WHERE id_produto=73;
 11  END;
```

- Constante: limite_de_credito CONSTANT := 5000.00;

Abstração de Dados

- CURSOR
  - Área Privada para armazenamento de informações;
- Atributo %TYPE
  - Associa o tipo de dado ao da coluna referenciada;
- Atributo %ROWTYPE
  - Representa um conjunto de tipos de dados de uma tabela;
- Collections
  - Permite declarar tipos de dados semelhantes a arrays, sets e hash tables;
- Records
  - Composição de vários tipos de dados diferentes;
- Object Types
  - Permite definir objetos, tais como subprogramas e tipos de dados;

Cursores Implícitos

- Criados automaticamente pelo sistema;
- Atributo:
  - %FOUND – Um DML alterou linhas?
  - %ISOPEN – Cursor está aberto ou fechado?
  - %NOTFOUND – Um DML falhou em alterar linhas?
  - %ROWCOUNT – Quantas linhas foram afetadas até agora?

Cursores Explícitos

- Declaração: CURSOR nome_cursor IS comando_select;
- Abrir: OPEN nome_cursor;
- Buscando dados: FETCH nome_cursor INTO nome_variavel;
- Fechando: CLOSE nome_cursor;

```
1  DECLARE
2      cpf      number(11);
3      nome     varchar2(50);
4      CURSOR emp IS SELECT cpf, nome from empregado;
5
6  BEGIN
7      OPEN emp;
8      LOOP
9          FETCH emp INTO cpf, nome;
10         DBMS_OUTPUT.PUTLINE('CPF: '|| cpf || '- NOME: '||nome);
11     END LOOP;
12     CLOSE emp;
13 END;
```

Cursores Explícitos

- Semelhantes aos implícitos, porém ao invés de precedermos com o SQL%, precedemos com o nome do Cursor;
- %FOUND – Alguma linha foi recuperada?
- %ISOPEN – Cursor está aberto?

```
IF c1%ISOPEN = FALSE THEN -- cursor fechado
    OPEN c1;
END IF;
```

- %NOTFOUND – Fetch falhou?
- %ROWCOUNT – Quantas linhas?

```
BEGIN
    OPEN c1;
    LOOP
        FETCH c1 INTO my_ename, my_salary;
        IF c1%NOTFOUND THEN -- falhou fetch, sair do loop
        -- "EXIT WHEN c1%NOTFOUND OR c1%NOTFOUND IS NULL;"
        END IF;
    END LOOP;
EXIT
```

```
LOOP
    FETCH c1 INTO name;
    EXIT WHEN c1%NOTFOUND OR c1%NOTFOUND IS NULL;
    DBMS_OUTPUT.PUT_LINE(c1%ROWCOUNT || '. ' || name);
    IF c1%ROWCOUNT = 5 THEN
        DBMS_OUTPUT.PUT_LINE('--- Fetched 5th record ---');
    END IF;
END LOOP;
```

Estados de um Cursor

Momento	%FOUND	%ISOPEN	%NOTFOUND	%ROWCOUNT
Antes OPEN	EXCEÇÃO	FALSE	EXCEÇÃO	EXCEÇÃO
DEPOIS OPEN	NULL	TRUE	NULL	0
ANTES 1º FETCH	NULL	TRUE	NULL	0
DEPOIS 1º FETCH	TRUE	TRUE	FALSE	1
ANTES CADA FETCH	TRUE	TRUE	FALSE	1
DEPOIS CADA FETCH	TRUE	TRUE	TRUE	DEPENDE
ANTES ÚLTIMO FETCH	TRUE	TRUE	FALSE	DEPENDE
APÓS ÚLTIMO FETCH	FALSE	TRUE	TRUE	DEPENDE
ANTES CLOSE	FALSE	TRUE	TRUE	DEPENDE
DEPOIS CLOSE	EXCEÇÃO	FALSE	EXCEÇÃO	EXCEÇÃO

Abstração de Dados

Atributo %TYPE

```
1.
credito      PLS_INTEGER RANGE 1000..25000;
debito       credito%TYPE;

2.
nome         VARCHAR2(20) := "Lucio";
sobrenome    nome%TYPE;

3.
v_usuario    usuario.nome%TYPE;
```

Atributo %ROWTYPE

```
1.
empregado    empregado%ROWTYPE;

2.
cursor C1 is
    select id_depto, nome_depto
        from departamentos;
depto_reg    c1%ROWTYPE;

3.
CURSOR c2 IS
    SELECT employee_id, email, employees.manager_id, location_id
        FROM employees, departments
        WHERE employees.department_id = departments.department_id;
join_rec c2%ROWTYPE;

4.
dept_rec departments%ROWTYPE;
BEGIN
    SELECT * INTO dept_rec
    FROM departments
    WHERE department_id = 30
    AND ROWNUM < 2;
END;
```

Manipulação de Erros

- NO_DATA_FOUND
- OTHERS

- Sintaxe:

  WHEN tratador_exceções THEN
    comando1;
    comando2;
    comandon;

Exemplo:

```
-- bloco PL/SQL que imprime dados sobre empregado que tem o maior salario

DECLARE
        v_nome empregado.nome%TYPE;
        v_endereco empregado.endereco%TYPE;
        v_salario empregado.salario%TYPE;

BEGIN
        SELECT nome, endereco, salario
        INTO v_nome, v_endereco, v_salario
        FROM empregado
        WHERE salario = (SELECT MAX (salario) FROM empregado);
        DBMS_OUTPUT.PUT_LINE (v_nome, v_endereco, v_salario );

EXCEPTION
        WHEN OTHERS
        DBMS_OUTPUT.PUT_LINE ('Erro Detectado');
END;
```

Manipulação de Erros

- DBMS:

```
IF Valor IS NULL THEN
    DBMS_STANDARD.Raise_application_error(-20201, 'Valor não pode ser nulo!');
END IF;
```

- RAISE:

```
DECLARE
  comm_missing   EXCEPTION;
.
.
.
IF comissao IS NULL THEN
  RAISE comm_missing;
ELSE
  (código)
END IF;
EXCEPTION
  WHEN comm_missing THEN
    DBMS_OUTPUT.PUT_LINE(' Esse empregado não recebeu comissão.');
    comissao := 0;
  WHEN OTHERS THEN
    NULL;
END;
```

1ª Bateria de Questões

ITnerante

PROVAS DE TI
TUDO PARA VOCÊ PASSAR

Questão 01 – 2010 – FCC – TRE/RS

A sigla PL/SQL significa:

A) Procedural Language extensions to SQL

B) Plataform Language extensions to SQL

C) Plataform Library extensions to SQL

D) Procedural Library extensions to SQL

E) Portable Language extensions to SQL

Questão 01 – 2010 – FCC – TRE/RS

A sigla PL/SQL significa:

A) Procedural Language extensions to SQL

B) Plataform Language extensions to SQL

C) Plataform Library extensions to SQL

D) Procedural Library extensions to SQL

E) Portable Language extensions to SQL

Questão 02 – 2010 – FCC – TRT20

A sigla PL/SQL significa uma linguagem

A) padrão SQL para o SGBD ORACLE.

B) padrão SQL para qualquer SGBD.

C) procedural do ORACLE, que substitui a linguagem SQL.

D) procedural do ORACLE, que estende o padrão SQL.

E) procedural, que estende o padrão SQL para qualquer SGBD.

Questão 02 – 2010 – FCC – TRT20

A sigla PL/SQL significa uma linguagem

A) padrão SQL para o SGBD ORACLE.

B) padrão SQL para qualquer SGBD.

C) procedural do ORACLE, que substitui a linguagem SQL.

D) procedural do ORACLE, que estende o padrão SQL.

E) procedural, que estende o padrão SQL para qualquer SGBD.

Questão 03 – 2012 - CESPE – TRE/RJ

A unidade básica em PL/SQL é um bloco com a seguinte estrutura: DECLARE, que é a seção para declaração de variáveis, tipos e subprogramas locais; BEGIN — única seção do bloco que é indispensável e obrigatória —, que é a seção executável, na qual ficam as instruções procedimentais e SQL; EXCEPTION, que é a seção/setor onde ficam as instruções de tratamento de erro; e END.

Certo

Errado

Questão 03 – 2012 - CESPE – TRE/RJ

A unidade básica em PL/SQL é um bloco com a seguinte estrutura: DECLARE, que é a seção para declaração de variáveis, tipos e subprogramas locais; BEGIN — única seção do bloco que é indispensável e obrigatória —, que é a seção executável, na qual ficam as instruções procedimentais e SQL; EXCEPTION, que é a seção/setor onde ficam as instruções de tratamento de erro; e END.

➡ Certo

Errado

Questão 04 – 2012 - FCC– TJ/PE

Um bloco PL/SQL tem três partes: uma parte declarativa, uma parte executável e uma parte de tratamento de exceção que lida com as condições de erro. No bloco é necessária, no mínimo, a presença

A) das três partes.

B) da parte declarativa.

C) da parte executável.

D) da parte executável e da declarativa.

E) da parte executável e a de tratamento de exceção.

Questão 04 – 2012 - FCC– TJ/PE

Um bloco PL/SQL tem três partes: uma parte declarativa, uma parte executável e uma parte de tratamento de exceção que lida com as condições de erro. No bloco é necessária, no mínimo, a presença

A) das três partes.

B) da parte declarativa.

C) da parte executável.

D) da parte executável e da declarativa.

E) da parte executável e a de tratamento de exceção.

Questão 06 – 2010 – FEPESE – SEFAZ/SC

Sobre a linguagem PL/SQL do Oracle, é **correto** afirmar:

A) Um bloco de programa PL/SQL deve conter três seções: declarativa (para declaração de variáveis, por exemplo), executável (comandos) e uma seção de tratamento de exceções.

B) Apresenta comandos condicionais e de repetição, como CASE, IF e REVOKE.

C) É uma linguagem orientada a objetos destinada ao desenvolvimento de aplicações que acessam bancos de dados.

D) É uma linguagem declarativa, ou seja, apresenta apenas instruções para consulta e atualização de dados.

E) Um bloco de programa PL/SQL pode conter uma seção declarativa (para declaração de variáveis, por exemplo) e uma seção de tratamento de exceções.

Questão 06 – 2010 – FEPESE – SEFAZ/SC

Sobre a linguagem PL/SQL do Oracle, é **correto** afirmar:

A) Um bloco de programa PL/SQL deve conter três seções: declarativa (para declaração de variáveis, por exemplo), executável (comandos) e uma seção de tratamento de exceções.

B) Apresenta comandos condicionais e de repetição, como CASE, IF e REVOKE.

C) É uma linguagem orientada a objetos destinada ao desenvolvimento de aplicações que acessam bancos de dados.

D) É uma linguagem declarativa, ou seja, apresenta apenas instruções para consulta e atualização de dados.

➡ E) Um bloco de programa PL/SQL pode conter uma seção declarativa (para declaração de variáveis, por exemplo) e uma seção de tratamento de exceções.

Questão 07 – 2012 - FCC– TRT19

Em relação à criação de um bloco PL/SQL, está INCORRETO:

A) A seção de Declaração é opcional somente quando o bloco não utilizar constantes ou variáveis.

B) Cada variável ou constante deve ser especificada, obrigatoriamente, com seu nome, tipo e valor inicial; todas as linhas devem terminar com ponto e vírgula.

C) A linguagem permite a declaração de variáveis e constantes que podem ser usadas em comandos SQL contidos em *procedures* e *funções.*

D) A declaração de uma constante é parecida com a de uma variável, diferenciando-se, apenas, pela palavra chave *CONSTANT.*

E) As variáveis podem ter qualquer tipo de *datatype* válido pela linguagem SQL e *ORACLE.*

Questão 07 – 2012 - FCC– TRT19

Em relação à criação de um bloco PL/SQL, está INCORRETO:

A) A seção de Declaração é opcional somente quando o bloco não utilizar constantes ou variáveis.

B) Cada variável ou constante deve ser especificada, obrigatoriamente, com seu nome, tipo e valor inicial; todas as linhas devem terminar com ponto e vírgula.

C) A linguagem permite a declaração de variáveis e constantes que podem ser usadas em comandos SQL contidos em *procedures* e *funções.*

D) A declaração de uma constante é parecida com a de uma variável, diferenciando-se, apenas, pela palavra chave *CONSTANT.*

E) As variáveis podem ter qualquer tipo de *datatype* válido pela linguagem SQL e *ORACLE.*

Questão 08 – 2011 - FCC– INFRAERO

A seção do bloco PL/SQL executável e dentro da qual ficam instruções procedimentais e SQL é a

A) EXCEPTION.

B) SELECTION.

C) EXECUTE.

D) DECLARE.

E) BEGIN.

Questão 08 – 2011 - FCC– INFRAERO

A seção do bloco PL/SQL executável e dentro da qual ficam instruções procedimentais e SQL é a

A) EXCEPTION.

B) SELECTION.

C) EXECUTE.

D) DECLARE.

E) BEGIN.

Questão 09 – 2010 – CESPE – MPU

Um bloco de programa PL/SQL inclui partes bem distintas, como declaração (*declaration*) de variáveis e objetos, módulo executável (*executable*) e módulo de excessões (*exception*).

Certo Errado

Questão 09 – 2010 – CESPE – MPU

Um bloco de programa PL/SQL inclui partes bem distintas, como declaração (*declaration*) de variáveis e objetos, módulo executável (*executable*) e módulo de excessões (*exception*).

➡ Certo

Errado

Questão 10 – 2011 - FCC– INFRAERO

Considere a linha de comando PL/SQL:

aux_salario emp.sal%type;

O parâmetro *%type*

A) cria um registro completo, utilizando as caracte- rísticas de uma tabela.

B) cria um registro completo, utilizando as características das colunas retornadas de um cursor.

C) faz com que uma variável ou constante assuma o formato dos valores de uma coluna da base de dados.

D) faz com que uma variável ou constante assuma o formato dos valores de uma tupla.

E) cria uma coluna completa, utilizando as características da coluna de origem.

Questão 10 – 2011 - FCC– INFRAERO

Considere a linha de comando PL/SQL:

aux_salario emp.sal%type;

O parâmetro *%type*

A) cria um registro completo, utilizando as caracte- rísticas de uma tabela.

B) cria um registro completo, utilizando as características das colunas retornadas de um cursor.

C) faz com que uma variável ou constante assuma o formato dos valores de uma coluna da base de dados.

D) faz com que uma variável ou constante assuma o formato dos valores de uma tupla.

E) cria uma coluna completa, utilizando as características da coluna de origem.

Questão 11 – 2010 – CESPE – MPU

O trecho de programa em PL/SQL a seguir possibilita remover os efeitos de determinada transação no banco de dados, sempre que um erro ocorrer durante a execução.

```
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line(Erro na atualização do Salário)
    ROLLBACK
```

Certo Errado

Questão 11 – 2010 – CESPE – MPU

O trecho de programa em PL/SQL a seguir possibilita remover os efeitos de determinada transação no banco de dados, sempre que um erro ocorrer durante a execução.

```
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line(Erro na atualização do Salário)
    ROLLBACK
```

Certo ➡ Errado

Questão 12 – 2012 - CESPE – TRE/RJ

CURSOR é uma área de trabalho temporária criada na memória do sistema quando um comando SQL é executado.

Certo Errado

Questão 12 – 2012 - CESPE – TRE/RJ

CURSOR é uma área de trabalho temporária criada na memória do sistema quando um comando SQL é executado.

➡ Certo

Errado

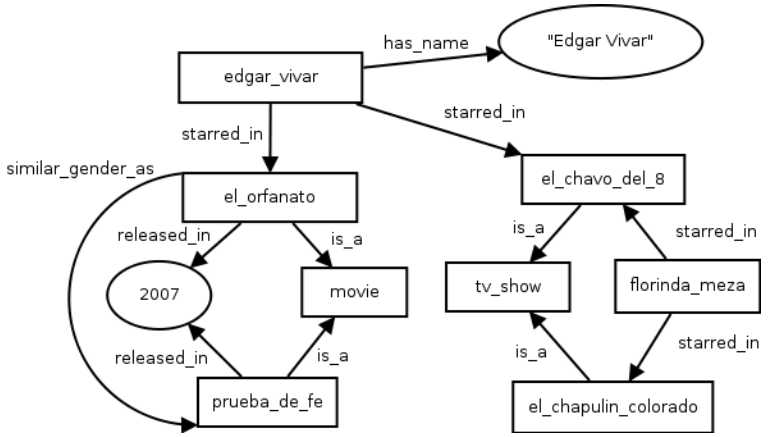
1ª Bateria - Gabarito

1 – A		9 – CERTO
2 – D	6 – E	10 – C
3 – CERTO	7 – B	11 – ERRADO
4 – C	8 – E	12 – CERTO

Estruturas de Controle

- Consideradas as estensões mais importantes da linguagem;
- Permite manipular e processar dados do Banco de Dados;

1. Controles Condicionais
2. Controles Iterativos
3. Controles Sequenciais



Controles Condicionais

- IF-THEN-ELSE

```
IF sal < 3500 THEN
    sal_raise := .10;
ELSE
    sal_raise := .07;
END IF;
```

- CASE

```
BEGIN
    SELECT job_id, salary INTO jobid, sal
    FROM employees
    WHERE employee_id = empid;

CASE
    WHEN jobid = 1 THEN
        sal_raise := .10; --aumento de 10%
    WHEN jobid = 2 THEN
        sal_raise := .11; --aumento de 11%
    WHEN jobid = 3 THEN
        sal_raise := .07;
    ELSE
        BEGIN
            DBMS_OUTPUT.PUT_LINE('Sem aumento para este cargo: ' || jobid);
        END;
    END CASE;
END;
```

Controles Condicionais

- IF-THEN

```
IF salario > 10000 THEN
    UPDATE emp set salario = salario * .10
    WHERE emp_id = emp_id;
END IF;
```

- IF-THEN-ELSIF

```
IF salario > 10000 THEN
    bonus : = 500;
ELSIF salario > 5000 THEN
    bonus := 300;
ELSE
    bonus := 100;
END IF;
```

- ELSIF Estendido

```
IF nota = 'A' THEN
    DBMS_OUTPUT.PUT_LINE('Excelente');
ELSIF nota = 'B' THEN
    DBMS_OUTPUT.PUT_LINE('Muito Bom');
ELSIF nota = 'C' THEN
    DBMS_OUTPUT.PUT_LINE('Bom');
ELSIF nota = 'D' THEN
    DBMS_OUTPUT.PUT_LINE('RUIM');
ELSE
    DBMS_OUTPUT.PUT_LINE('Não Avaliado');
END IF;
```

Controles Condicionais

- CASE

```
CASE nota
    WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE('Excelente');
    WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE('Muito Bom');
    WHEN 'C' THEN DBMS_OUTPUT.PUT_LINE('Bom');
    WHEN 'D' THEN DBMS_OUTPUT.PUT_LINE('Ruim');
    ELSE DBMS_OUTPUT.PUT_LINE('Não Avaliado');
END CASE;
```

- ELSE RAISE CASE_NOT_FOUND
- <<nome_do_case>>

Controles Iterativos

- LOOP

```
LOOP
    --ação que deverá ser executada
END LOOP;
```

- FOR–LOOP

```
FOR i in 1..100 LOOP
    dbms_output.put_line(i);
END LOOP;
```

- WHILE–LOOP

```
WHILE sal <= 10000 LOOP
    --atualiza a tabela com o aumento de salário
END LOOP;
```

- EXIT–WHEN

```
LOOP
    --ação que deverá ser executada
    EXIT WHEN valor > 100;
END LOOP;
```

- CONTINUE_WHEN

```
LOOP
    --ação que deverá ser executada
    CONTINUE WHEN valor > 100;
END LOOP;
```

Controles Sequenciais

- GOTO

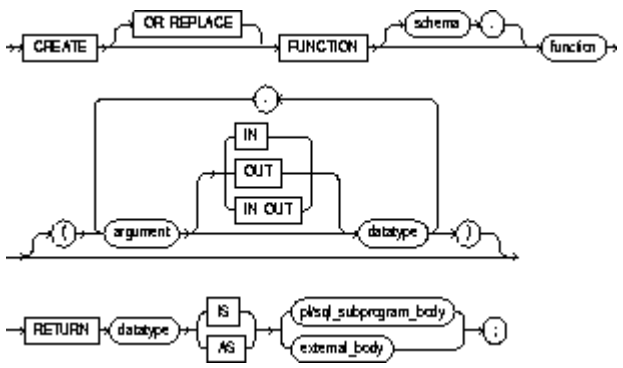
```
<<calculo_salario>>
salario := salario *.10

IF salario > 10000 THEN
    GOTO exibe_salario;
ELSE
    GOTO calculo_salario;
END IF;

<<exibe_salario>>
dbms_output.put_line(salario);
```

Subprogramas

- Permitem estender a linguagem PL/SQL;
- Possibilitam quebrar um programa em pedaços menores e bem definidos;
- Promovem a reusabilidade;
- Promovem a manutenabilidade;
- Tipos de Subprogramas:
  - Procedure
  - Functions
  - Triggers
  - Packages



Parameter Mode

- IN (DEFAULT)
  - Permite passar um valor para o subprograma.
- OUT
  - Retonar um valor para quem chamou o subprograma.
- IN OUT
  - Permite passar um valor e este ser atualizado e devolvido.

Análise de Código

```
DECLARE
    emp_num NUMBER(6) := 120;
    bonus NUMBER(6) := 50;
    emp_last_name VARCHAR2(25);
PROCEDURE raise_salary (emp_id IN NUMBER, amount IN NUMBER,
                        emp_name OUT VARCHAR2) IS
    BEGIN
        UPDATE employees SET salary =
            salary + amount WHERE employee_id = emp_id;
        SELECT last_name INTO emp_name
            FROM employees
            WHERE employee_id = emp_id;
    END raise_salary;
BEGIN
    raise_salary(emp_num, bonus, emp_last_name);
    DBMS_OUTPUT.PUT_LINE('Salario foi atualizado para o funci: ' || emp_last_name);
END;
```

Procedure

```
1. CREATE [OR REPLACE] PROCEDURE [schema.]nome_da_procedure
2. [(parâmetro1 [modo1] tipodedado1,
3.         parâmetro2 [modo2] tipodedado2,
4.         ...)]
5. IS|AS
6. Bloco PL/SQL
```

- REPLACE – caso a procedure já exista será substituída;
- Bloco PL/SQL – toda o código existente dentro do bloco begin – end (ou end “nome da procedure”);
- Modo – parameter mode

```
1.  CREATE OR REPLACE PROCEDURE aumenta_sal (p_empno IN emp.empno%TYPE) IS
2.  BEGIN
3.          UPDATE
4.                  scott.emp
5.          SET
6.                  sal = sal * 1.10
7.          WHERE
8.                  empno = p_empno;
9.  END aumenta_sal;
10. /
```

Function

```
1. CREATE [OR REPLACE] FUNCTION nome_da_função
2. [( parameter1 [ mode1] datatype1,
3. parameter2 [ mode2] datatype2,
4. . . .)]
5. RETURN tipo_de_dado
6. IS|AS
7. Bloco PL/SQL;
```

- Em uma função deverá ter pelo menos uma claúsula RETURN, caso contrário será lançada uma exceção;

```
1.  CREATE OR REPLACE FUNCTION pega_sal
2.         (p_id IN emp.empno%TYPE)
3.         RETURN NUMBER
4.      IS
5.         v_sal emp.sal%TYPE :=0;
6.      BEGIN
7.         SELECT sal
8.           INTO v_sal
9.           FROM scott.emp
10.         WHERE empno = p_id;
11.     RETURN v_sal;
12. END pega_sal;
13. /
```

Análise de Código

```
CREATE OR REPLACE FUNCTION SOMA (num1 NUMBER, num2 NUMBER)
    RETURN NUMBER AS
BEGIN
    RETURN num1+num2;
END SOMA;

CREATE OR REPLACE PROCEDURE PLOOP (limite IN INTEGER) IS
BEGIN
    FOR i IN 2..limit LOOP
        DBMS_OUTPUT.PUT_LINE('Dentro do loop: ' || i);
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Fora do loop: ' || TO_CHAR(limite));
END;
```

Triggers

- Subprograma armazenado associado a tabela, view ou evento;
- Pode ser associada a um DML, DDL ou operação;
- Acionamento:
  - Antes ou Após a declaração ser executada;
  - Antes ou depois de cada linha afetada;
- Estados:
  - Enabled ou Disabled;
- Vantagens:
  - Impor regras de negócio;
  - Fornecer registro de eventos;
  - Fornecer auditoria...entre outros;

- Trigger não é um mecanismo de segurança

Triggers

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
   Declaration-statements
BEGIN
   Executable-statements
EXCEPTION
   Exception-handling-statements
END;
```

- Create [Or Replace] – cria ou substitui uma trigger que já existe.
- {Before | After | Instead Of} – especifica quando a trigger deverá ser executada.
- {Insert [Or] | Update [Or] | Delete} – especifica a operação DML.
- [Of col_name] – especifica a coluna que terá que sofrer modificação.
- [Referencing OLD AS o NEW AS n] – permite que referencia novos e antigos valores para as declarações DML.
- [For Each Row] – especifica o nível do trigger.
- When (condition) – provê uma condição para as linhas que o trigger será acionado.

Análise de Código

```
CREATE OR REPLACE TRIGGER t
  BEFORE
    INSERT OR
    UPDATE OF salary, department_id OR
    DELETE
  ON employees
BEGIN
  CASE
    WHEN INSERTING THEN
      DBMS_OUTPUT.PUT_LINE('Inserting');
    WHEN UPDATING('salary') THEN
      DBMS_OUTPUT.PUT_LINE('Updating salary');
    WHEN UPDATING('department_id') THEN
      DBMS_OUTPUT.PUT_LINE('Updating department ID');
    WHEN DELETING THEN
      DBMS_OUTPUT.PUT_LINE('Deleting');
  END CASE;
END;
/
```

Pseudo Registros

Triggering Statement	OLD.*field* Value	NEW.*field* Value
`INSERT`	`NULL`	Post-insert value
`UPDATE`	Pre-update value	Post-update value
`DELETE`	Pre-delete value	`NULL`

Análise de Código

```
CREATE OR REPLACE TRIGGER audit_sal
    AFTER UPDATE OF salary
        ON employees
            FOR EACH ROW
BEGIN
    INSERT INTO emp_audit
        VALUES(:old.employee_id, SYSDATE, :new.salary, :old.salary);
END;
```

```
CREATE OR REPLACE TRIGGER Print_salary_changes
    BEFORE DELETE OR INSERT OR UPDATE ON emp
        FOR EACH ROW
            WHEN (NEW.EMPNO > 0)
DECLARE
    sal_diff number;
BEGIN
    sal_diff := :NEW.SAL - :OLD.SAL;
    dbms_output.put('Old salary: ' || :OLD.sal);
    dbms_output.put(' New salary: ' || :NEW.sal);
    dbms_output.put_line(' Difference ' || sal_diff);
END;
```

Package

- Agrupa logicamente elementos de programação PL/SQL;
- Disponibiliza várias packages pré-definidas;
- Composta por duas partes:
  - Especificação (CREATE PACKAGE);
  - Corpo (CREATE PACKAGE BODY);

Vantagens

- Modularidade
- Fácil Design de Aplicações
- Esconder Informações
- Adicionar Funcionalidades
- Melhor Performance

Análise de Código

```
CREATE OR REPLACE PACKAGE emp_actions AS

    PROCEDURE hire_employee (
        employee_id NUMBER,
        last_name VARCHAR2,
        first_name VARCHAR2,
        email VARCHAR2,
        phone_number VARCHAR2,
        hire_date DATE,
        job_id VARCHAR2,
        salary NUMBER,
        commission_pct NUMBER,
        manager_id NUMBER,
        department_id NUMBER
    );

    PROCEDURE fire_employee (emp_id NUMBER);

    FUNCTION num_above_salary (emp_id NUMBER) RETURN NUMBER;
END emp_actions;
```

Análise de Código

```
CREATE OR REPLACE PACKAGE BODY emp_actions AS

    PROCEDURE hire_employee (employee_id NUMBER,last_name VARCHAR2,
        first_name VARCHAR2,email VARCHAR2,phone_number VARCHAR2,
        hire_date DATE,job_id VARCHAR2,salary NUMBER,commission_pct NUMBER,
        manager_id NUMBER,department_id NUMBER
    ) IS
    BEGIN
        INSERT INTO employees
            VALUES (employee_id,last_name,first_name,
            email,phone_number,hire_date,job_id,salary,
            commission_pct,manager_id,department_id);
    END hire_employee;

    PROCEDURE fire_employee (emp_id NUMBER) IS
    BEGIN
        DELETE FROM employees WHERE employee_id = emp_id;
    END fire_employee;

    FUNCTION num_above_salary (emp_id NUMBER) RETURN NUMBER IS
        emp_sal NUMBER(8,2);
        num_count NUMBER;
    BEGIN
        SELECT salary INTO emp_sal FROM employees
            WHERE employee_id = emp_id;
        SELECT COUNT(*) INTO num_count FROM employees
            WHERE salary > emp_sal;

        RETURN num_count;
    END num_above_salary;
END emp_actions;
```

Chamando um subprograma de uma package

```
CALL emp_actions.hire_employee (300, 'Belden', 'Enrique',
    'EBELDEN', '555.111.2222',
    '31-AUG-04', 'AC_MGR', 9000,
    .1, 101, 110);
```

```
SQL> BEGIN
  2   DBMS_OUTPUT.PUT_LINE
  3       ('Number of employees with higher salary: ' ||
  4        TO_CHAR(emp_actions.num_above_salary(120)));
  5
  6   emp_actions.fire_employee(300);
  7 END;
  8 /

Number of employees with higher salary: 34

PL/SQL procedure successfully completed.
```

Packages Pré-Definidos

- DBMS_ALERT – disparar um alerta quando algum valor for alterado;
- DBMS_OUTPUT – exibir saídas dos blocos pl/sql;
- DBMS_PIPE – passagem de informações para processos;
- DBMS_CONNECTION_POOL – gerenciar pools de conexões residentes;
- HTF E HTP – possiblitam a geração de tags HTML;
- UTL_FILE – manipulação de arquivos;
- UTL_HTTP – chamadas HTTP;
- UTL_SMTP – envio de emails;

2ª Bateria de Questões

ITnerante

PROVAS DE TI
TUDO PARA VOCÊ PASSAR

Questão 13 – 2013 - FEPESE – JUCESCTRE/RJ

A palavra reservada utilizada em PL/SQL para referenciar campos específcos que devem disparar uma trigger de UPDATE é:

A) IN

B) OF

C) ON

D) FOR

E) AMONG

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
   Declaration-statements
BEGIN
   Executable-statements
EXCEPTION
   Exception-handling-statements
END;
```

Questão 13 – 2013 - FEPESE – JUCESCTRE/RJ

A palavra reservada utilizada em PL/SQL para referenciar campos específcos que devem disparar uma trigger de UPDATE é:

A) IN

B) OF

C) ON

D) FOR

E) AMONG

Questão 13 – 2013 - FEPESE – JUCESCTRE/RJ

A palavra reservada utilizada em PL/SQL para referenciar campos específcos que devem disparar uma trigger de UPDATE é:

A) IN

B) OF

C) ON

D) FOR

E) AMONG

Questão 14 – 2013 - FCC – TRT9

A linguagem PL/SQL, utilizada no gerenciador de banco de dados ORACLE, possui o conjunto de comandos SQL e acrescenta recursos de programação utilizados em outras linguagens de programação.

Considere a seguinte subrotina PL/SQL:

```
CREATE OR REPLACE PROCEDURE decisao (a IN REAL, b IN REAL) IS x REAL;
  BEGIN
    x := a + b;
       IF (x > 10)
      THEN DBMS_OUTPUT.put_line (x);
      ELSE DBMS_OUTPUT.put_line ('Valor inferior ao limite');
  END IF;
 END;
/
```

Tendo sido esta rotina implementada no Oracle Database 10g *Express Edition no Windows*, já executada a linha de comandos SQL que dá acesso ao banco de dados, é correto afirmar que

A) o resultado de EXEC decisao (5.5, 4.5); é 10.0.

B) a *stored procedure* decisao recebe 2 parâmetros reais e apresenta apenas o resultado da adição dos valores cuja soma seja maior ou igual a 10.

C)o comando de decisão IF utilizado na *stored procedure* apresenta erros de sintaxe.

D)o resultado de EXEC decisao (5.5, 6.5); é 12.0.

E)não existe ELSE no comando de decisão IF; o certo seria usar ELSEIF.

Questão 14 – 2013 - FCC – TRT9

A linguagem PL/SQL, utilizada no gerenciador de banco de dados ORACLE, possui o conjunto de comandos SQL e acrescenta recursos de programação utilizados em outras linguagens de programação.

Considere a seguinte subrotina PL/SQL:

```
CREATE OR REPLACE PROCEDURE decisao (a IN REAL, b IN REAL) IS x REAL;
  BEGIN
    x := a + b;
       IF (x > 10)
      THEN DBMS_OUTPUT.put_line (x);
      ELSE DBMS_OUTPUT.put_line ('Valor inferior ao limite');
  END IF;
 END;
/
```

Tendo sido esta rotina implementada no Oracle Database 10g *Express Edition no Windows*, já executada a linha de comandos SQL que dá acesso ao banco de dados, é correto afirmar que

A) o resultado de EXEC decisao (5.5, 4.5); é 10.0.

B) a *stored procedure* decisao recebe 2 parâmetros reais e apresenta apenas o resultado da adição dos valores cuja soma seja maior ou igual a 10.

C)o comando de decisão IF utilizado na *stored procedure* apresenta erros de sintaxe.

D)o resultado de EXEC decisao (5.5, 6.5); é 12.0.

E)não existe ELSE no comando de decisão IF; o certo seria usar ELSEIF.

Questão 15 – 2012 - CESPE – ANAC

Procedure e function são objetos PL/SQL que armazenam blocos de códigos PL/SQL. Destes dois, o objeto function permite que se retorne um valor a partir do comando Return.

CERTO ERRADO

Questão 15 – 2012 - CESPE – ANAC

Procedure e function são objetos PL/SQL que armazenam blocos de códigos PL/SQL. Destes dois, o objeto function permite que se retorne um valor a partir do comando Return.

➡ CERTO

ERRADO

Questão 16 – 2011 - FCC – TRT19

Considere o bloco PL/SQL abaixo:

```
CREATE OR REPLACE TRIGGER department_maiusc
BEFORE INSERT OR UPDATE ON department
FOR EACH ROW
DECLARE
Dup_flag INTEGER;
BEGIN
:NEW.dept_name := UPPER(:NEW.dept_name);
END;
```

É INCORRETO afirmar que este bloco contém comandos para

A) criar o *trigger* department_maiusc.

B) substituir o trigger existente de nome department_maiusc.

C) disparar o *trigger* sempre que houver a inclusão de uma nova linha na respectiva tabela do banco de dados.

D) acionar o *trigger* sempre que houver mudança num registro da respectiva tabela do banco de dados.

E) forçar o nome do departamento a ser colocado em letras minúsculas.

Questão 16 – 2011 - FCC – TRT19

Considere o bloco PL/SQL abaixo:

```
CREATE OR REPLACE TRIGGER department_maiusc
BEFORE INSERT OR UPDATE ON department
FOR EACH ROW
DECLARE
Dup_flag INTEGER;
BEGIN
:NEW.dept_name := UPPER(:NEW.dept_name);
END;
```

É INCORRETO afirmar que este bloco contém comandos para

A) criar o *trigger* department_maiusc.

B) substituir o trigger existente de nome department_maiusc.

C) disparar o *trigger* sempre que houver a inclusão de uma nova linha na respectiva tabela do banco de dados.

D) acionar o *trigger* sempre que houver mudança num registro da respectiva tabela do banco de dados.

➡ E) forçar o nome do departamento a ser colocado em letras minúsculas.

Questão 17 – 2011 - FCC – TRT19

Sobre as formas de execução de funções (functions) do PL/SQL, considere:

I. Pode-se executar uma função como parte de uma instrução *SELECT.*

II. Pode-se atribuir o valor de uma função a uma variável.

III. Não é possível passar parâmetros para uma função quando ela é executada dentro de um *trigger*.

É correto o que consta APENAS em

A) I e II.
B) I e III.
C) II e III.
D) I.
E) III.

Questão 17 – 2011 - FCC – TRT19

Sobre as formas de execução de funções (functions) do PL/SQL, considere:

I. Pode-se executar uma função como parte de uma instrução *SELECT.*

II. Pode-se atribuir o valor de uma função a uma variável.

III. Não é possível passar parâmetros para uma função quando ela é executada dentro de um *trigger*.

É correto o que consta APENAS em

A) I e II.
B) I e III.
C) II e III.
D) I.
E) III.

Questão 18 – 2011 - FCC – TRE/AP

Em PL/SQL é INCORRETO afirmar que *triggers* são executados quando

A) ocorre operações de instruções de DML em um objeto *schema* específico.

B) ocorre operações de instruções de DDL feitos em um *schema* ou numa base de dados.

C) ocorre erros de servidor.

D) invocados explicitamente pelo usuário.

E) ocorre eventos de *Login/Logoff* do usuário.

Questão 18 – 2011 - FCC – TRE/AP

Em PL/SQL é INCORRETO afirmar que *triggers* são executados quando

A) ocorre operações de instruções de DML em um objeto *schema* específico.

B) ocorre operações de instruções de DDL feitos em um *schema* ou numa base de dados.

C) ocorre erros de servidor.

D) invocados explicitamente pelo usuário.

E) ocorre eventos de *Login/Logoff* do usuário.

Questão 19 – 2011 - FCC – TRE/RS

Considere a procedure PL/SQL abaixo:

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

O bloco PL/SQL que executará as ações da *procedure* inicia-se em

A) CREATE.

B) IS.

C) BEGIN.

D) UPDATE.

E) SET.

Questão 19 – 2011 - FCC – TRE/RS

Considere a procedure PL/SQL abaixo:

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

O bloco PL/SQL que executará as ações da *procedure* inicia-se em

A) CREATE.

B) IS.

C) BEGIN.

D) UPDATE.

E) SET.

Se uma consulta PL/SQL no Oracle retornar mais do que uma tupla, então, para receber o retorno da consulta, será necessário usar um

A) while

B) cursor.

C) procedure.

D) declare.

E) for.

Questão 20 – 2010 - CESGRANRIO – IBGE

Se uma consulta PL/SQL no Oracle retornar mais do que uma tupla, então, para receber o retorno da consulta, será necessário usar um

A) while

B) cursor.

C) procedure.

D) declare.

E) for.

2ª Bateria - Gabarito

13 – B	17 – A	
14– D	18 – D	
15 – CERTO	19 – C	
16 – E	20 –B	