11

PL/SQL: Procedures, Functions, Triggers

Objetivos

- Descrever o formato básico de Blocos PL/SQL
  - Dbms_output, If, loops, raise
- Functions
- Procedures
- Triggers

PL/SQL

- PL/SQL: **Procedural Language extension to SQL**
  - linguagem programática para DB Oracle

- Benefícios:
  - Integração com SQL
  - Portabilidade (independente de plataforma)
  - Reuso de código; Sobrecarga
  - Agrupamento de código
  - Manutenção
  - Simplificação de aplicações
  - Ofuscamento
  - Tratar erros

PL/SQL

- Estrutura básica de um bloco PL/SQL anônimo:

```
DECLARE
      declaração de variáveis
BEGIN
      corpo (sql & pl/sql statements)
END;
/
```

PL/SQL

- Estrutura básica de um bloco PL/SQL anônimo:

```
DECLARE
    v_teste      number;
BEGIN
    v_teste := 100;
    dbms_output.put_line( v_teste );
END;
/
```

```
Statement processed.
100
```

PL/SQL

- Estrutura básica de um bloco PL/SQL anônimo:
  - Comandos podem estar em +1 linha
  - Final de cada comando possui ;

```
DECLARE
    v_teste
          number;
BEGIN
    v_teste :=
          100;
    dbms_output.put_line(
                v_teste );
END;
/
```

```
Statement processed.
100
```

Caracteríticas Importantes: Select dentro de PL/SQL

```
DECLARE
   v_nome varchar2(50);
BEGIN
   select nome
   from   rh.funcionarios
   where  codfunc=10;
END;
/
```

- PLS-00428: é esperada uma cláusula INTO nesta instrução SELECT

Caracteríticas Importantes: Select dentro de PL/SQL

```
DECLARE
   v_nome varchar2(50);
BEGIN
   select nome
   into   v_nome
   from   rh.funcionarios
   where  codfunc=10;
END;
/
```

Procedimento PL/SQL concluído com sucesso.

Caracteríticas Importantes: Select dentro de PL/SQL

- Dentro de um bloco **PL/SQL** (BEGIN … END;)
  - Todo SELECT deve
    - retornar uma única linha!
      caso contrário: erro NO DATA FOUND
    - guardar os valores das colunas retornadas em variáveis

Caracteríticas Importantes: Select dentro de PL/SQL

```
DECLARE
   v_nome varchar2(50);
BEGIN
   select  nome
   into    v_nome
   from    rh.funcionarios;
END;
/
```

- ORA-01422: a extração exata retorna mais do que o número solicitado de linhas
- ORA-01422: exact fetch returns more than requested number of rows"

Caracteríticas Importantes: Select dentro de PL/SQL

```
DECLARE
   v_nome varchar2(50);
BEGIN
   select  nome
   into    v_nome
   from    rh.funcionarios
   where   codfunc=10;
END;
/
```

Procedimento PL/SQL concluído com sucesso.

Caracteríticas Importantes: DDLs e DCLs

- Dentro de um **PL/SQL:**
  - ✓ Permitido Consulta, DMLs ou Constrole de Transações
  - ✗ Não permitido DDLs ou DCLs

```
BEGIN
   alter trigger teste_trg disable;
END;
/
```

- ORA-06550: linha 2, coluna 5:
- PLS-00103: Encontrado o símbolo "ALTER" quando um dos seguintes símbolos era esperado:

IF

```
DECLARE
    v_qtd  number;
BEGIN
    select count(*)
    into   v_qtd
    from   hr.employees;

    if v_qtd > 100 then
        dbms_output.put_line( 'Mais de 100 funcionarios');
    else
        dbms_output.put_line( 'Menos de 100 funcionarios');
    end if;
END;
/

Statement processed.
Mais de 100 funcionarios
```

FOR 1 .. n

```
DECLARE
    v_qtd  number;
BEGIN
    for x in 1..100 loop
        select count(*)
        into   v_qtd
        from   hr.employees
        where   department_id = x;

        dbms_output.put_line( 'Funcionarios no DEPT('
                              || x || ') = ' || v_qtd);
    end loop;
END;
/
```

```
Funcionarios no DEPT(49) = 0
Funcionarios no DEPT(50) = 45
Funcionarios no DEPT(51) = 0
Funcionarios no DEPT(52) = 0
Funcionarios no DEPT(53) = 0
Funcionarios no DEPT(54) = 0
Funcionarios no DEPT(55) = 0
Funcionarios no DEPT(56) = 0
Funcionarios no DEPT(57) = 0
Funcionarios no DEPT(58) = 0
Funcionarios no DEPT(59) = 0
Funcionarios no DEPT(60) = 5
```

FOR in (select)

```
DECLARE
    v_qtd  number;
BEGIN
    for x in (select distinct department_id from hr.employees) loop
        select count(*)
        into   v_qtd
        from   hr.employees
        where   department_id = x.department_id;

        dbms_output.put_line( 'Funcionarios no DEPT(' ||
                        x.department_id || ') = ' || v_qtd);
    end loop;
END;
/
```

```
Statement processed.
Funcionarios no DEPT(50) = 45
Funcionarios no DEPT(40) = 1
Funcionarios no DEPT(110) = 2
Funcionarios no DEPT(90) = 3
Funcionarios no DEPT(30) = 6
Funcionarios no DEPT(70) = 1
Funcionarios no DEPT() = 0
Funcionarios no DEPT(10) = 1
Funcionarios no DEPT(20) = 2
Funcionarios no DEPT(60) = 5
Funcionarios no DEPT(100) = 6
Funcionarios no DEPT(80) = 34
```

RAISE ERROR

```
DECLARE
    v_qtd number;
BEGIN
    select  count(*)
    into    v_qtd
    from    hr.employees;

    if v_qtd < 100 then
        dbms_output.put_line ( 'OK/qtde de funcionários menor que 100.' );
    else
        raise_application_error( -20101, 'ERRO: qtde funcionários maior que 100' );
    end if;
END;
/
```

```
ORA-20101: ERRO: qtde funcionários maior que 100
ORA-06512: at line 11
```

PL/SQL Anonimos vs Nomeados

- Bloco PL/SQL anônimo: BEGIN … END;

- Blocos PL/SQL nomeados:
  - FUNCTION
  - PROCEDURE
  - TRIGGER
  - PACKAGE

Blocos Nomeados
FUNCTION

```
create or replace FUNCTION <nome>
   ( parâmetros )
RETURN <tipo_dados>
IS
   -- variáveis
BEGIN
   -- instruções
   RETURN <valor>;
END;
/
```

Blocos Nomeados
PROCEDURE

```
create or replace PROCEDURE <nome>
  ( parâmetros )
AS
  -- variáveis
BEGIN
  -- instruções
END;
/
```

Blocos Nomeados
TRIGGER

```
create or replace TRIGGER <nome>
AFTER|BEFORE   INSERT|UPDATE|DELETE
  [of <COLUNAS>] on <TABELA>
FOR EACH ROW
DECLARE
  -- variáveis
BEGIN
  -- instruções
END;
/
```

Exemplo: FUNCTION

```
create or replace FUNCTION qtd_func_in_dept
( p_dept_id number )
return number
is
    v_qtd_funcs  number;
begin
    select  count(1)
    into    v_qtd_funcs
    from    hr.departments
    where   department_id = p_dept_id;

    return v_qtd_funcs;
end;
/

Function created.
```

Exemplo: FUNCTION

```
select  qtd_func_in_dept( 30 ) from dual;
```

QTD_FUNC_IN_DEPT(30)
6

```
select  department_id, qtd_func_in_dept( department_id )
from    hr.departments;
```

DEPARTMENT_ID	QTD_FUNC_IN_DEPT(DEPARTMENT_ID)
10	1
20	2
30	6
40	1
50	45

Exemplo: PROCEDURE

```
create or replace procedure P_DEPTARTMENTS_I
    (   p_department_id    hr.departments.department_id%TYPE ,
        p_department_name  hr.departments.department_name%TYPE ,
        p_manager_id       hr.departments.manager_id%TYPE := NULL,
        p_location_id      hr.departments.location_id%TYPE := NULL
    )
as
begin
    insert into hr.departments (department_id, department_name, manager_id, location_id)
    values (p_department_id, p_department_name, p_manager_id, p_location_id);
end;
/
```

Exemplo: PROCEDURE

```
select * from hr.departments;
```

250	Retail Sales	-	1700
260	Recruiting	-	1700
270	Payroll	-	1700

```
begin
     P_DEPTARTMENTS_I( 280, 'Infraestrutura' );
end;
/
```

```
PL/SQL procedure successfully completed.
```

```
select * from hr.departments;
```

```
DEPARTMENT_ID DEPARTMENT_NAME                MANAGER_ID LOCATION_ID
------------- ------------------------------ ---------- -----------
          230 IT Helpdesk                                      1700
          240 Government Sales                                 1700
          250 Retail Sales                                     1700
          260 Recruiting                                       1700
          270 Payroll                                          1700
          280 Infraestrutura
```

Exemplo: TRIGGER

```
create table FUNCIONARIOS(
    cod_empr     number          NOT NULL PRIMARY KEY,
    nome         varchar2(100)   NOT NULL,
    cpf          varchar2(14)    NOT NULL,
    data_nasc    date            NOT NULL,
    nro_fone     varchar2(20)    NOT NULL,
    tipo_fone    char(3)         NOT NULL  check
                          (tipo_fone in ('FIX', 'MOB')),
    salario      number(10,2) default 0   NOT NULL,
    foto         BLOB,
    curriculo    CLOB
);
```

```
create sequence SEQ_FUNCIONARIOS;

Sequence created.
```

Exemplo: TRIGGER

```
create or replace trigger TRG_FUNCIONARIOS_SEQ
before insert on FUNCIONARIOS
for each row
BEGIN
        :new.cod_empr := SEQ_FUNCIONARIOS.nextval;
END;
/

Trigger created.
```

```
insert into FUNCIONARIOS values ( null, 'Alberto', sysdate, 100, 100000);
```

Exemplo: PACKAGE

```
create or replace PACKAGE p1
is
        function f_abc return number;
        procedure p_xyz;
end;
/

create or replace PACKAGE BODY p1
is
        function f_abc ( ... )
        return number
        is
        begin
                ...
        end;

        procedure p_xyz ( ... )
        is
        begin
                ...
        end;
end;
/
```

```
select  p1.f_abc( ... )
from ...  ;


begin
        p1.p_xyz( ... );
end;
/
```

Resumo e Dúvidas

- Dúvidas ou comentários ... ?