EXEMPLOS COM PL/PGSQL

1) Verificando se existe suporte a PL/pgSQL no banco de dados em uso.

   a) Via sql: SELECT true::BOOLEAN FROM pg_language WHERE lanname='plpgsql';

2) Instalando a PL/pgSQL no meu banco de dados(usuário precisa ser superuser).

   a) Por linha de comando: createlang plpgsql nome_banco_de_dados
   b) Via sql(conectado ao banco de dados): create language plpgsql;

3) Função que retorna o maior valor entre dois valores inteiros passados.

```
CREATE OR REPLACE FUNCTION min(integer, integer) RETURNS integer AS $body$
   SELECT CASE WHEN $1 < $2 THEN $1 ELSE $2 END
$body$ LANGUAGE SQL;
```

USANDO PLPGSQL

4) Calculando a taxa:

```
CREATE FUNCTION taxa_venda(real) RETURNS real AS $$
     DECLARE total ALIAS FOR $1; 12
     BEGIN
          RETURN total * 0.06;
     END;
$$
LANGUAGE plpgsql;
```

5) Retorna uma lista de valores utilizando RETURN NEXT.

```
CREATE OR REPLACE FUNCTION getAlunos() RETURNS SETOF aluno AS
$$
DECLARE
   a aluno%rowtype;
BEGIN
   FOR a IN SELECT * FROM aluno
   WHERE codalu > 0
   LOOP
      -- can do some processing here
      IF (MOD(A.CODALU,2) = 0) THEN
         RETURN NEXT a; -- return current row of SELECT

      END IF;
   END LOOP;
   RETURN;
END
$$
LANGUAGE plpgsql;
```

- USANDO A FUNÇÃO
  SELECT * FROM getAlunos();

6) Um exemplo de uma função em PL/pgSQL para remover acentos.

--chamaremos a função de ft_racento(função remove acentos)

```
CREATE OR REPLACE FUNCTION ft_racento(termo VARCHAR) RETURNS VARCHAR AS
$body$
DECLARE
        com_acento
VARCHAR(100):='ÁÀÂÃÄÉÈÊËÌÍÎÏÓÒÔÕÖÚÙÛÜÇáàâãäéèêëíìîïóòôõöúùûüç°ºª';
        sem_acento
VARCHAR(100):='AAAAAEEEEIIIIOOOOOUUUUCaaaaaeeeeiiiioooooouuuucooa';
BEGIN
        return TRANSLATE(termo, com_acento, sem_acento);
END;
$body$
LANGUAGE plpgsql;

-- chamando a função para um exemplo
SELECT ft_racento('JOÃO');

clientDataSet1.commandText := 'SELECT * FROM pessoa
                                WHERE ft_racento(nompes) ilike ft_racento(:termo);';
clientDataSet1.params[0].value := 'joão' ;
clientDataSet1.open;
```

7) Exemplo tratando exception no momento de uma inserção ou update.

```
CREATE TABLE db (a INT PRIMARY KEY, b TEXT);

CREATE FUNCTION merge_db(key INT, data TEXT) RETURNS VOID AS
$body$
BEGIN
   LOOP
      -- first try to update the key
      UPDATE db SET b = data WHERE a = key;
      IF found THEN
         RETURN;
      END IF;
      -- not there, so try to insert the key
      -- if someone else inserts the same key concurrently,
      -- we could get a unique-key failure
      BEGIN
         INSERT INTO db(a,b) VALUES (key, data);
         RETURN;
      EXCEPTION WHEN unique_violation THEN
         -- do nothing, and loop to try the UPDATE again
      END;
   END LOOP;
END;
$body$
LANGUAGE plpgsql;

-- USANDO A FUNÇÃO
SELECT merge_db(1, 'david');
SELECT merge_db(1, 'dennis');
```

8) Exemplo utilizando CURSOR(cursor that encapsulates the query, and then read the query result a few rows at a time).

```
CREATE OR REPLACE FUNCTION update_historico()
RETURNS integer AS
$body$
```

```
DECLARE
   hist    RECORD;
   counter int;
   curs1   refcursor;
BEGIN
   counter := 0;
   OPEN curs1 FOR
        SELECT * FROM historico WHERE MOD(codalu, 2) = 0;

   LOOP
        FETCH curs1 INTO hist;
        EXIT WHEN NOT FOUND;
        RAISE NOTICE 'ROW %  %.',counter, timeofday();
       UPDATE historico SET vlrnot = vlrnot * 1.25 WHERE codalu = hist.codalu and coddis =
hist.coddis;
        counter := counter + 1;
   END LOOP;
   RETURN counter;
END;
$body$ LANGUAGE plpgsql;
```

9) Mais um exemplo utilizando cursor.

```
CREATE OR REPLACE FUNCTION getAlunoNotas(refcursor)
RETURNS refcursor AS
$body$
BEGIN
  OPEN $1 FOR  SELECT A.CODALU, A.NOMALU, D.NOMDIS, H.VLRNOT
                FROM ALUNO AS A
                INNER JOIN HISTORICO AS H ON H.CODALU = A.CODALU
                INNER JOIN DISCIPLINA AS D ON D.CODDIS = H.CODDIS
                ORDER BY A.NOMALU, NOMDIS;
  RETURN $1;
END
$body$
LANGUAGE 'plpgsql';

-- fazendo a chamada
BEGIN;
        SELECT getAlunoNotas('notas');
        FETCH ALL IN notas;
END;
```

10) Um exemplo de auditoria utilizando gatilhos e funções.

```
-- criando a tabela onde serão armazenadas as informações auditadas
CREATE TABLE AUDITORIA  (
        ID SERIAL NOT NULL PRIMARY KEY,
        TABELA VARCHAR(50) NOT NULL,
        USUARIO VARCHAR(50) NOT NULL,
        DATA TIMESTAMP NOT NULL,
        OPERACAO  VARCHAR(1)  NOT  NULL,  -- I  –  INCLUSÃO,  E  –  EXCLUSÃO,  A  -
ALTERAÇÃO
        NEWREG TEXT,
        OLDREG TEXT
);
```

```
-- criando a função genérica de auditoria
CREATE OR REPLACE FUNCTION ft_auditoria() RETURNS TRIGGER AS
$body$
BEGIN
        -- Cria uma linha na tabela AUDITORIA para refletir a operação
        -- realizada na tabela que invoca a trigger.       --
        IF (TG_OP = 'DELETE') THEN
                INSERT INTO auditoria(tabela, usuario, data, operacao,oldreg)
        SELECT TG_RELNAME, user, current_timestamp, 'E', OLD::text;
                RETURN OLD;
        ELSIF (TG_OP = 'UPDATE') THEN
                INSERT INTO auditoria(tabela, usuario, data, operacao,newreg,oldreg)
                SELECT TG_RELNAME, user, current_timestamp, 'A',NEW::text,OLD::text;
                RETURN NEW;
        ELSIF (TG_OP = 'INSERT') THEN
                INSERT INTO auditoria(tabela, usuario, data, operacao,newreg)
        SELECT TG_RELNAME, user, current_timestamp, 'I',NEW::text;
                RETURN NEW;
        END IF;
        RETURN NULL; -- o resultado é ignorado uma vez que este é um gatilho AFTER
END;
$body$
LANGUAGE plpgsql;

-- auditando a tabela academico e a tabela uf

CREATE TRIGGER naturalidade_audit AFTER INSERT OR UPDATE OR DELETE ON
naturalidade FOR EACH ROW EXECUTE PROCEDURE ft_auditoria();

CREATE TRIGGER aluno_audit AFTER INSERT OR UPDATE OR DELETE ON aluno FOR
EACH ROW EXECUTE PROCEDURE ft_auditoria();

-- agora basta executar os comandos insert, update e delete sobre estas tabelas, e as mesmas
serão auditadas.
```

11) Exemplo de função de validação de CPF. Fonte do algoritmo: http://postgresqlbr.blogspot.com/2008/06/validao-de-cpf-com-pl-pgsql.html.

```
CREATE OR REPLACE FUNCTION CPF_Validar(par_cpf varchar(11)) RETURNS integer AS
$body$
-- ROTINA DE VALIDAÇÃO DE CPF
-- Conversão para o PL/ PGSQL: Cláudio Leopoldino - http://postgresqlbr.blogspot.com/
-- Algoritmo original: http://webmasters.neting.com/msg07743.html
-- Retorna 1 para CPF correto.
DECLARE

    x real;
    y real; --Variável temporária
    soma integer;
    dig1 integer; --Primeiro dígito do CPF
    dig2 integer; --Segundo dígito do CPF
    len integer; -- Tamanho do CPF
    contloop integer; --Contador para loop
    val_par_cpf varchar(11); --Valor do parâmetro
```

```
BEGIN
    -- Teste do tamanho da string de entrada
    IF char_length(par_cpf) != 11 THEN
        RAISE NOTICE 'Formato inválido: %',$1;
        RETURN 0;
    END IF;
    -- Inicialização
    x := 0;
    soma := 0;
    dig1 := 0;
    dig2 := 0;
    contloop := 0;
    val_par_cpf := $1; --Atribuição do parâmetro a uma variável interna
    len := char_length(val_par_cpf);
    x := len -1;
    -- Loop de multiplicação - dígito 1
    contloop :=1;
    WHILE contloop <= (len -2) LOOP
        y := CAST(substring(val_par_cpf from contloop for 1) AS NUMERIC);
        soma := soma + ( y * x);
        x := x - 1;
        contloop := contloop +1;
    END LOOP;
    dig1 := 11 - CAST((soma % 11) AS INTEGER);
    if (dig1 = 10) THEN dig1 :=0 ; END IF;
    if (dig1 = 11) THEN dig1 :=0 ; END IF;
    -- Dígito 2
    x := 11; soma :=0;
    contloop :=1;
    WHILE contloop <= (len -1) LOOP
        soma := soma + CAST((substring(val_par_cpf FROM contloop FOR 1)) AS REAL) * x;
        x := x - 1;
        contloop := contloop +1;
    END LOOP;
    dig2 := 11 - CAST ((soma % 11) AS INTEGER);
    IF (dig2 = 10) THEN dig2 := 0; END IF;
    IF (dig2 = 11) THEN dig2 := 0; END IF;
    --Teste do CPF
    IF ((dig1 || '' || dig2) = substring(val_par_cpf FROM len-1 FOR 2)) THEN
        RETURN 1;
    ELSE
        RAISE NOTICE 'DV do CPF Inválido: %',$1;
        RETURN 0;
    END IF;
END;
$body$
LANGUAGE plpgsql;
```

12) Geração de CPFs Fictícios com Pl/PgSQL. Fonte do algoritmo: http://we2ajax.blogspot.com/2010/07/geracao-de-cpfs-ficticios-com-pl-pgsql.html(perece estar fora do ar).

```
CREATE OR REPLACE FUNCTION gerar_CPF() RETURNS varchar AS $body$

-- ROTINA DE GERAÇÃO DE CPF SEM LOOP
-- Retorna string com CPF aletório correto.
```

```
DECLARE
    vet_cpf integer [11]; --Recebe o CPF
    soma integer; -- Soma utilizada para o cálculo do DV
    rest integer; -- Resto da divisão
BEGIN
    -- Atribuição dos valores do Vetor
    vet_cpf[0] := cast(substring (CAST (random() AS VARCHAR), 3,1) as integer);
    vet_cpf[1] := cast(substring (CAST (random() AS VARCHAR), 3,1) as integer);
    vet_cpf[2] := cast(substring (CAST (random() AS VARCHAR), 3,1) as integer);
    vet_cpf[3] := cast(substring (CAST (random() AS VARCHAR), 3,1) as integer);
    vet_cpf[4] := cast(substring (CAST (random() AS VARCHAR), 3,1) as integer);
    vet_cpf[5] := cast(substring (CAST (random() AS VARCHAR), 3,1) as integer);
    vet_cpf[6] := cast(substring (CAST (random() AS VARCHAR), 3,1) as integer);
    vet_cpf[7] := cast(substring (CAST (random() AS VARCHAR), 3,1) as integer);
    vet_cpf[8] := cast(substring (CAST (random() AS VARCHAR), 3,1) as integer);

    -- CÁLCULO DO PRIMEIRO NÚMERO DO DV
    -- Soma dos nove primeiros multiplicados por 10, 9, 8 e assim por diante...
    soma:=(vet_cpf[0]*10)+(vet_cpf[1]*9)+(vet_cpf[2]*8)+(vet_cpf[3]*7)+(vet_cpf[4]*6)+
    (vet_cpf[5]*5)+(vet_cpf[6]*4)+(vet_cpf[7]*3)+(vet_cpf[8]*2);
    rest:=soma % 11;
    if (rest = 0) or (rest = 1) THEN
        vet_cpf[9]:=0;
    ELSE
        vet_cpf[9]:=(11-rest);
    END IF;
    -- CÁLCULO DO SEGUNDO NÚMERO DO DV
    -- Soma dos nove primeiros multiplicados por 11, 10, 9 e assim por diante...
    soma:= (vet_cpf[0]*11) + (vet_cpf[1]*10) + (vet_cpf[2]*9) + (vet_cpf[3]*8) + (vet_cpf[4]*7) +
    (vet_cpf[5]*6) + (vet_cpf[6]*5) + (vet_cpf[7]*4) + (vet_cpf[8]*3) + (vet_cpf[9]*2);
    rest:=soma % 11;
    if (rest = 0) or (rest = 1) THEN
        vet_cpf[10] := 0;
    ELSE
        vet_cpf[10] := (11-rest);
    END IF;
    --Retorno do CPF
    RETURN trim(trim(to_char(vet_cpf[0],'9')) || trim(to_char(vet_cpf[1],'9')) ||
    trim(to_char(vet_cpf[2],'9')) || trim(to_char(vet_cpf[3],'9')) || trim(to_char(vet_cpf[4],'9')) ||
    trim(to_char(vet_cpf[5],'9')) || trim(to_char(vet_cpf[6],'9')) || trim(to_char(vet_cpf[7],'9'))||
    trim(to_char(vet_cpf[8],'9')) || trim(to_char(vet_cpf[9],'9')) || trim(to_char(vet_cpf[10],'9')));
END;
$body$
LANGUAGE plpgsql;

-- Chamada da função, retornando um CPF aleatório.
SELECT gerar_CPF();

-- Verificando se o CPF gerado é válido utilizando a função de validação.
SELECT gerar_CPF(), CPF_Validar(gerar_CPF());
```