Lista de Exercícios VI
Implementação de Funções e Gatilhos em PL/pgSQL

João Batista de Sousa Paula, 11911BCC008
John Vitor da Silva Cunha, 11821BCC005
Laís Saloum Deghaide, 11821BCC001
Vinícius Calixto Rocha, 11911BCC039
Yan Stivaletti e Souza, 11821BCC002

1)

- CREATE OR REPLACE FUNCTION ehoras(employee.ssn%TYPE)
  RETURNS DECIMAL(6,2) AS $$
  DECLARE myhoras DECIMAL(6,2);
  BEGIN SELECT SUM(hours) FROM works_on INTO myhoras WHERE essn=$1;
  RETURN myhoras;
  END;
  $$ LANGUAGE plpgsql;

  SELECT * FROM ehoras('123456789');

  RESPOSTA
  "ehoras"
  40.00

- CREATE OR REPLACE FUNCTION does_employee_exist (employee.ssn%TYPE)
  RETURNS bool AS $$
  DECLARE key ALIAS FOR $1; myemployee employee%ROWTYPE;
  BEGIN SELECT INTO myemployee * FROM employee WHERE ssn=key;
  IF NOT FOUND THEN RETURN false;
  END IF;
  RETURN true;
  END;
  $$ LANGUAGE plpgsql;

  SELECT * FROM does_employee_exist ('123456789');

  RESPOSTA

does_employee_exist
VERDADEIRO

- CREATE OR REPLACE FUNCTION vemp()
  RETURNS SETOF employee AS $$
  DECLARE
  wrow employee%rowtype;
  BEGIN

```
FOR wrow IN SELECT * FROM employee
LOOP
RETURN NEXT wrow;
END LOOP;
RETURN;
END;
$ LANGUAGE plpgsql;

SELECT * FROM vemp();
```

RESPOSTA

fname	minit	lname	ssn	bdate	address	sex	salary	superssn	dno
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000.00	333445555	5
Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000.00	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire ,TX	F	43000.00	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice,	F	25000.00	333445555	5

					Houston,TX				
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000.00	987654321	4

- CREATE OR REPLACE FUNCTION tgenero(company.employee.ssn%TYPE)
  RETURNS TEXT AS $$
  DECLARE
  myrow company.employee%ROWTYPE;
  mysexo TEXT DEFAULT ' ';
  BEGIN
  SELECT * INTO myrow FROM company.employee WHERE ssn=$1;
  IF myrow.sex = 'F' THEN mysexo := 'Feminino';
  ELSE IF myrow.sex = 'M' THEN mysexo := 'Masculino';
  END IF;
  END IF;
  RETURN myrow.fname || ' ' || myrow.minit || '. ' || myrow.lname || ' - ' || mysexo;
  END;
  $$ LANGUAGE plpgsql;

  SELECT * FROM tgenero('123456789');

  RESPOSTA

"tgenero"
John B. Smith - Masculino

- CREATE TABLE company.materializeddsummary
  AS SELECT dno as Dno, count(*) as NroEmp,
  sum(salary) as TotalS, avg(salary) as AverageS
  FROM company.employee GROUP BY dno;

  CREATE TABLE company_original.visoesmaterializadas (
  mvname varchar(50) PRIMARY KEY, -- armazenar nome da "visão"
  mvquery varchar(500) -- armazenar comando que insere dados
  );

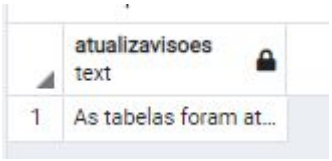
  INSERT INTO company_original.visoesmaterializadas(mvname, mvquery)

```
VALUES ('company_original.materializeddsummary',
'SELECT dno as Dno, count(*) as NroEmp,
sum(salary) as TotalS, avg(salary) as AverageS
FROM company_original.employee GROUP BY dno');

INSERT INTO company_original.visoesmaterializadas(mvname, mvquery)
VALUES ('company_original.materializeddsummary',
'SELECT dno as Dno, count(*) as NroEmp,
sum(salary) as TotalS, avg(salary) as AverageS
FROM company_original.employee GROUP BY dno');

CREATE OR REPLACE FUNCTION atualizavisoes() RETURNS TEXT AS $$
DECLARE
mviews RECORD;
BEGIN
FOR mviews IN SELECT * FROM company_original.visoesmaterializadas
ORDER BY mvname
LOOP
EXECUTE 'DELETE FROM ' || mviews.mvname;
 EXECUTE 'INSERT INTO '
|| mviews.mvname || ' ' || mviews.mvquery;
END LOOP;
RETURN 'As tabelas foram atualizadas!!';
END;
$$ LANGUAGE plpgsql;
SELECT * FROM atualizavisoes();
```

RESPOSTA



2)

- update employee
  set salary=0
  where ssn='123456789'

RESPOSTA

ERROR: ERRO: John Smith : Salario !> 0?
CONTEXT: função PL/pgSQL emp_stamp() linha 7 em RAISE

RESULTADO DO SELECT APÓS O UPDATE

	fname character varying (15)	minit character (1)	lname character varying (15)	ssn [PK] character (9)	bdate date	address character varying (30)	sex character (1)
1	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M
2	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M
3	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M
4	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F
5	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire,TX	F
6	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M
7	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston,TX	F
8	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M
9	joao batista	J	sousa	190560405	1997-05-19	638 Voss, Houston, TX	M
10	mariana	a	amorim	190560410	1997-05-19	638 Voss, Houston, TX	M

- select * from dependent
  where relationship='SPOUSE';

  RESPOSTA

essn	dependent_name	sex	bdate	relationship
333445555	Joy	F	1958-05-03	SPOUSE
987654321	Abner	M	1942-02-28	SPOUSE
123456789	Elizabeth	F	1967-05-05	SPOUSE

- CREATE OR REPLACE FUNCTION spouse ()
  RETURNS TRIGGER AS $$
  BEGIN
  IF NEW.relationship = 'SPOUSE'
  THEN PERFORM * FROM dependent
  WHERE essn=NEW.essn
  AND dependent_name <> NEW.dependent_name
  AND relationship='SPOUSE';
  IF FOUND THEN RAISE EXCEPTION 'Employee % still have a spouse',
  NEW.essn;
  END IF;
  END IF;
  RETURN NEW;
  END;
  $$ LANGUAGE plpgsql;

  CREATE TRIGGER spouse BEFORE INSERT OR UPDATE ON dependent FOR EACH ROW EXECUTE PROCEDURE spouse();

  INSERT INTO employee

VALUES ('joao batista','J','sousa','190560405',DATE '1997-05-19', '638 Voss, Houston, TX','M', 40000, '888665555', 5);

INSERT 0 1

RESPOSTA
Query returned successfully

- CREATE OR REPLACE FUNCTION fnewemp () RETURNS TRIGGER AS $$
DECLARE
mycomand TEXT;
mydsummary materializeddsummary%ROWTYPE;
mynroemp materializeddsummary.nroemp%TYPE; mytotals
materializeddsummary.totals%TYPE;
myaverages materializeddsummary.averages%TYPE;
BEGIN
IF NEW.dno ISNULL THEN
RAISE EXCEPTION 'dno cannot be NULL value';
END IF;
IF NEW.salary ISNULL THEN
RAISE EXCEPTION '% cannot have NULL salary', NEW.fname;
END IF;

IF NEW.salary <= 0 THEN
RAISE EXCEPTION '% cannot have a negative salary',
NEW.fname;
END IF;
SELECT * FROM materializeddsummary
INTO mydsummary WHERE dno=NEW.dno;
mynroemp = mydsummary.nroemp + 1;
mytotals = mydsummary.totals + NEW.salary;
myaverages = ((mydsummary.averages * mydsummary.nroemp) +
NEW.salary) / mynroemp;

mycomand := 'UPDATE ' || 'materializeddsummary ' || 'SET nroemp = '
|| quote_literal(mynroemp) || ', totals = ' || quote_literal(mytotals)
|| ', averages = ' || quote_literal(myaverages)
|| ' ' || 'WHERE dno = ' || quote_literal(NEW.dno);
EXECUTE mycomand;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tnewemp BEFORE INSERT ON employee
FOR EACH ROW EXECUTE PROCEDURE fnewemp();

SELECT * FROM materializeddsummary

RESPOSTA

dno	nroemp	totals	averages
4	3	93000.00	31.000.000.000.000.000
1	1	55000.00	55.000.000.000.000.000
5	4	133000.00	33.250.000.000.000.000

- INSERT INTO employee
  VALUES ('mariana','a','amorim','190560410',DATE '1997-05-19', '638 Voss, Houston,TX',
  'M', 40000, '888665555', 5);

  RESPOSTA
  INSERT 0 1

  Query returned successfully

3) SET SEARCH_PATH TO company;

CREATE OR REPLACE FUNCTION checkvisoes () RETURNS TRIGGER AS $$
BEGIN
IF NEW.dno = 4 THEN
RAISE EXCEPTION 'O Departamento ja foi alocado na visão';
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER checkvisoes
BEFORE INSERT OR UPDATE ON materializeddsummary
FOR EACH ROW EXECUTE PROCEDURE checkvisoes();
Antes do update:
SELECT * FROM materializeddsummary

RESPOSTA
"dno","nroemp","totals","averages"
4,3,93000.00,31000.000000000000
1,1,55000.00,55000.000000000000

5,4,133000.00,33250.000000000000

Depois do update:
UPDATE materializeddsummary SET dno = 4 WHERE nroemp = 3;

RESPOSTA
ERROR: ERRO: O Departamento ja foi alocado na visão
CONTEXT: função PL/pgSQL checkvisoes() linha 4 em RAISE

4)
```
CREATE OR REPLACE FUNCTION modcatgen(see.categoria.idcat%TYPE)
RETURNS TEXT AS $$
DECLARE
myrow RECORD;
mysexo TEXT DEFAULT ' ';
BEGIN
        SELECT * INTO myrow FROM (see.categoria NATURAL JOIN
        see.modalidade) WHERE idcat=$1;
        IF myrow.generocat = 'F' THEN mysexo := 'Feminino';
        ELSE IF myrow.generocat = 'M' THEN mysexo := 'Masculino';
                END IF;
        END IF;
        RETURN myrow.nomecat || '  ' || myrow.nomemod || '  ' || mysexo;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM modcatgen('32')
```

RESPOSTA
"modcatgen"
"Peso Mosca Boxe Masculino"

5)
```
CREATE VIEW vencedores AS
(SELECT a1.idcompetidor, nomemod, nomecat, escore, unidade
FROM atleta a1, pessoa p1, participacao pa1, competicao c1,
categoria, modalidade
WHERE a1.idpes = p1.idpes
AND a1.idcompetidor = pa1.idcompetidor
AND pa1.idcompeticao = c1.idcompeticao
AND c1.idcat = categoria.idcat
AND categoria.idmod = modalidade.idmod
AND fasecom = 'F'
AND NOT EXISTS
(SELECT *
FROM atleta a2, pessoa p2,
```

```
participacao pa2, competicao c2
WHERE a2.idpes = p2.idpes
AND a2.idcompetidor = pa2.idcompetidor
AND pa2.idcompeticao = c2.idcompeticao
AND c1.idcompeticao = c2.idcompeticao
AND ((pa2.escore < pa1.escore
AND pa2.unidade IN
('segundos', 'milisegundos'))
OR
(pa2.escore > pa1.escore
AND pa2.unidade IN
('metros' ,'centimetros' ,'milimetros'
,'pontos','sets', 'gols'))))
UNION
SELECT a1.idcompetidor, nomemod, nomecat, escore, unidade
FROM equipe a1, participacao pa1, competicao c1,
categoria, modalidade
WHERE a1.idcompetidor = pa1.idcompetidor
AND pa1.idcompeticao = c1.idcompeticao
AND c1.idcat = categoria.idcat
AND categoria.idmod = modalidade.idmod
AND tipocat = 'C'
AND fasecom = 'F'
AND NOT EXISTS
(SELECT *
FROM equipe a2, participacao pa2, competicao c2
WHERE a2.idcompetidor = pa2.idcompetidor
AND pa2.idcompeticao = c2.idcompeticao
AND c1.idcompeticao = c2.idcompeticao
AND ((pa2.escore < pa1.escore
AND pa2.unidade IN
('segundos', 'milisegundos'))
OR
(pa2.escore > pa1.escore
AND pa2.unidade IN
('metros' ,'centimetros' ,'milimetros'
,'pontos','sets', 'gols'))))
)

CREATE OR REPLACE FUNCTION patresult(see.empresa.cnpj%TYPE)
RETURNS TEXT AS $$
DECLARE
qtd INT;
BEGIN
```

```
SELECT COUNT(idcompetidor) FROM empresa NATURAL JOIN patrocinio
NATURAL JOIN vencedores INTO qtd
WHERE cnpj=$1;
RETURN qtd;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM patresult('20222222000133');
```

RESPOSTA

```
"patresult"
"1"
```

6)

```
CREATE OR REPLACE FUNCTION compcheck() RETURNS TRIGGER AS $$
BEGIN
        PERFORM * FROM competicao AS comp
                WHERE NEW.idloc = comp.idloc
                        AND NEW.datacom = comp.datacom
                        AND (comp.horacom + interval  '2 hour') >= NEW.horacom
                        AND (comp.horacom - interval  '2 hour') <= NEW.horacom;

                IF FOUND

                        THEN RAISE EXCEPTION 'ERRO intervalo de horários
                        registrados menor que duas horas para o mesmo local para
                        competicao';

                END IF;

        RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER compcheck

        BEFORE INSERT OR UPDATE ON competicao

                FOR EACH ROW EXECUTE PROCEDURE compcheck();



BEGIN TRANSACTION;
INSERT INTO see.competicao(idcompeticao,fasecom,datacom,horacom,idcat,idloc)
VALUES (default, '0', '2006-07-10', '13:00', '1', '3');
```

END TRANSACTION;

RESPOSTA

“ERRO: ERRO intervalo de horários registrados menor que duas horas para o mesmo local para competicao CONTEXT: função PL/pgSQL compcheck() linha 9 em RAISE”

7)
```
CREATE OR REPLACE FUNCTION atlcheck() RETURNS TRIGGER AS $$
DECLARE
        tupla1 RECORD;
        tupla2 RECORD;
BEGIN
        FOR tupla1 IN SELECT * FROM pessoa, equipe NATURAL JOIN inscricao
NATURAL JOIN categoria
                WHERE NEW.idpes=pessoa.idpes AND NEW.idequ=equipe.idequ
                LOOP
                        IF tupla1.sexo = tupla1.generocat
                                THEN RETURN NEW;
                        END IF;
                        IF tupla1.sexo <> tupla1.generocat
                                THEN RAISE EXCEPTION 'ERRO gênero da
        categoria da equipe não vale para o atleta';
                        END IF;
                END LOOP;
        FOR tupla1 IN SELECT * FROM pessoa WHERE pessoa.idpes=NEW.idpes
                LOOP
                FOR tupla2 IN SELECT * FROM pessoa NATURAL JOIN atlequ
        WHERE atlequ.idequ=NEW.idequ
                        LOOP
                                IF tupla2.sexo <> tupla1.sexo
                                        THEN RAISE EXCEPTION 'ERRO gênero da
        equipe não vale para o atleta';
                                END IF;
                        END LOOP;
                END LOOP;
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER atlcheck
        BEFORE INSERT OR UPDATE ON atlequ
                FOR EACH ROW EXECUTE PROCEDURE atlcheck();
```

```
BEGIN TRANSACTION;
        INSERT INTO see.atlequ (idpes, idequ) VALUES ('4', '5');
END TRANSACTION;
```

RESPOSTA

"ERRO: ERRO gênero da categoria da equipe não vale para o atleta CONTEXT: função PL/pgSQL atlcheck() linha 13 em RAISE"

```
BEGIN TRANSACTION;
        INSERT INTO see.atlequ (idpes, idequ) VALUES ('4', '12');
END TRANSACTION;
```

RESPOSTA

"ERRO: ERRO gênero da equipe não vale para o atleta CONTEXT: função PL/pgSQL atlcheck() linha 21 em RAISE"

8)

```
CREATE OR REPLACE FUNCTION newcompetidor() RETURNS TRIGGER AS
$$
        DECLARE
                competidor RECORD;
        BEGIN
                FOR competidor IN SELECT idcompetidor FROM atleta WHERE
NEW.idcompetidor IN
                (SELECT idcompetidor FROM equipe)
                        LOOP
                                IF NEW.idcompetidor=competidor.idcompetidor
THEN
                                        RAISE EXCEPTION 'Ja existe um competidor
com esse idcompetidor!';
                                END IF;
                END LOOP;
                FOR competidor IN SELECT idcompetidor FROM equipe WHERE
NEW.idcompetidor IN
                (SELECT idcompetidor FROM atleta)
                        LOOP
                                IF NEW.idcompetido=competidor.idcompetidor THEN
                                        RAISE EXCEPTION 'Ja existe um competidor
com esse idcompetidor!';
                                END IF;
                END LOOP;
```

```
                RETURN NEW;
        END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER newcompetidor
        BEFORE INSERT OR UPDATE ON competidor
                FOR EACH ROW EXECUTE PROCEDURE newcompetidor();
```

RESPOSTA

ERROR: ERRO: Ja existe um competidor com esse idcompetidor!

CONTEXT: função PL/pgSQL newcompetidor() linha 9 em RAISE