PL-pgSQL – Triggers

Exemplo :

Tabelas usadas no exemplo:

```
create table alterada(
      cod serial primary key,
      valor varchar(50));

create table log(
        cod serial primary key,
         data date,
         autor varchar(20),
         alteracao varchar(6));
```

PL-pgSQL – Triggers

Exemplo (cont):

Criando a função para a trigger:

```
create function gera_log() returns trigger as
$$
Begin
        insert into log (data, autor, alteracao) values (now(), user, TG_OP);
       return new;
end;
$$ language 'plpgsql';
```

Criando a trigger:

```
create trigger tr_gera_log after insert or update or delete on alterada  for each row execute
     procedure gera_log();
```

PL-pgSQL – Triggers

Outro exemplo:

```
CREATE TABLE funcionarios (
    ID_FUNCIONARIO SERIAL,
    NOME VARCHAR(255),
    IDADE INTEGER,
    FUNCAO VARCHAR(150),
    PRIMARY KEY(ID_FUNCIONARIO)
);

CREATE TABLE FUNCIONARIOS_LOG (
    COD_ALTERACAO SERIAL,
    USUARIO VARCHAR(150) NOT NULL,
    TIPO_ACAO VARCHAR(25) NOT NULL,
    DATA_ALTERACAO TIMESTAMP NOT NULL,
    PRIMARY KEY (COD_ALTERACAO)
);
```

PL-pgSQL – Triggers

Outro exemplo:

```
CREATE FUNCTION valida_dados_funcionario() RETURNS TRIGGER AS $valida_dados_funcionario$
BEGIN
   IF NEW.NOME IS NULL THEN
      RAISE EXCEPTION 'Por Favor, digite o nome do funcionario!';
   END IF;

   IF NEW.IDADE IS NULL THEN
      RAISE EXCEPTION 'Por favor, informe a idade!';
   END IF;

   IF NEW.IDADE < 0 THEN
      RAISE EXCEPTION 'Desculpe, o funcionario não pode ter % anos', NEW.IDADE;
   END IF;

   IF NEW.FUNCAO IS NULL THEN
      RAISE EXCEPTION 'Por favor, informe qual a função do funcinário!';
   END IF;

   -- Aqui iremos gravar no log o tipo de ação que foi realizada
   INSERT INTO FUNCIONARIOS_LOG (USUARIO, TIPO_ACAO, DATA_ALTERACAO) VALUES (CURRENT_USER, TG_OP, CURRENT_TIMESTAMP);

   RETURN NEW;
END;
$valida_dados_funcionario$
LANGUAGE plpgsql;
```

PL-pgSQL – Triggers

Outro exemplo:

Vincular a trigger a uma tabela:

```
CREATE TRIGGER validacao_insert BEFORE INSERT OR UPDATE ON funcionarios
FOR EACH ROW EXECUTE PROCEDURE valida_dados_funcionario();
```

Alterar o nome de uma trigger;

```
ALTER TRIGGER validacao_insert ON funcionarios RENAME TO a_validacao_insert
```

Para desbilitar uma trigger:

```
ALTER TABLE  funcionarios DISABLE TRIGGER a_validacao_insert
```

Ou alterar todas as triggers de uma tabela:

```
ALTER TABLE  funcionarios DISABLE TRIGGER ALL
```

PL-pgSQL –Triggers

Vincular a trigger a uma tabela:

```
CREATE TRIGGER validacao_insert BEFORE INSERT OR UPDATE ON funcionarios
FOR EACH ROW EXECUTE PROCEDURE valida_dados_funcionario();
```

Alterar o nome de uma trigger;

```
ALTER TRIGGER validacao_insert ON funcionarios RENAME TO a_validacao_insert
```

Para desbilitar uma trigger:

```
ALTER TABLE  funcionarios DISABLE TRIGGER a_validacao_insert
```

Ou alterar todas as triggers de uma tabela:

```
ALTER TABLE  funcionarios DISABLE TRIGGER ALL
```

DROP TRIGGER a_validacao_insert ON funcionarios -> remove a trigger

PL-pgSQL – Trabalho T2

Exercício: Considerando as tabelas abaixo

```
create table tr_emp
(idEmp integer primary key,
nome varchar(50) not null,
maxSal decimal(9,2) not null check (maxsal > 0),
dth_inc timestamp,
usu_inc varchar(20),
dth_atu timestamp,
usu_atu varchar(20));
```

PL-pgSQL – Trabalho T2

```
create table tr_sal_emp
(idSal_emp integer primary key,
idEmp integer references tr_emp(idEmp) not null,
dtIni date not null,
dtFim date null check (dtFim > dtIni),
sal decimal(9,2) not null,
usu_inc varchar(20),
dth_inc timestamp,
usu_atu varchar(20),
dth_atu timestamp);

create table tr_promovido
( anoMes integer primary key,
qtd integer check (qtd > 0));
```

PL-pgSQL – Trabalho T2

1) Escreva gatilhos para:

a) Verificar se tr_sal_emp.SAL é inferior um igual ao maior salário permitido para um funcionário (emp.maxSal);

b) Garantir que a data de início de um novo salário não esteja entre qualquer outro período de salário para o empregado;

c) Se for uma inclusão com data de início posterior a dezembro de 2010 a data de fim NÃO pode ser preenchida; Porém, se a data de início informada for inferior a Janeiro de 2012 e o empregado possuir os códigos 10, 20 ou 30 então a data de fim deverá ser igual a 31/12/2012.

d) Preencher automaticamente os campos (dependendo da operação):

- **usu_inc / usu_atu := current_user**
- **dth_inc / dth_atu:= current_timestamp**

e) Quando um funcionário receba aumento, ou seja, um novo registro é inserido na tabela TR_SAL_EMP automaticamente um registro deve ser incluído na tabela TR_PROMOVIDO (ou atualizado, se já existir).

- **OBS: A tabela TR_PROMOVIDO possui a quantidade de empregados que receberam aumento em um determinado ano/mês. Esse ano/mês é obtido da data de inclusão da tabela TR_SAL_EMP.**

PL-pgSQL – Trabalho T2

2) Escreva um trigger para auditar as operações efetuadas na tabela TR_EMP.

a) Criar tabela de auditoria composta, no mínimo, dos seguintes atributos:

- **Operação realizada**
- **Usuário que efetuou a operação**
- **Data e hora da ocorrência da operação**
- **MaxSal ANTES e DEPOIS quando de alteração da linha**

- **b) Criar trigger para popular a tabela de auditoria conforme são realizadas as operações na tabela TR_EMP;**