1. **Tarefa:**

   a. Verificar se tr_sal_emp.sal é inferior ou igual ao maior salário permitido para um funcionário (tr_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 receber 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.

2. **Realizar uma auditoria para as operações efetuadas na tabela TR_EMP, para isso:**

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

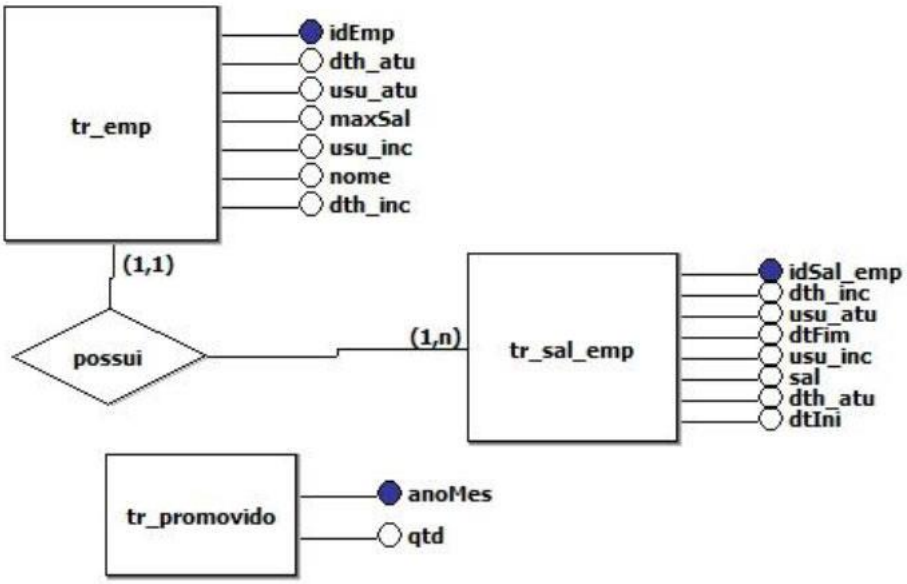
      - Operação realizada
      - Usuário que efetuou a operação
      - Data e hora da ocorrência da operação
      - MaxSal ANTES e DEPOIS quando de alteração da linha

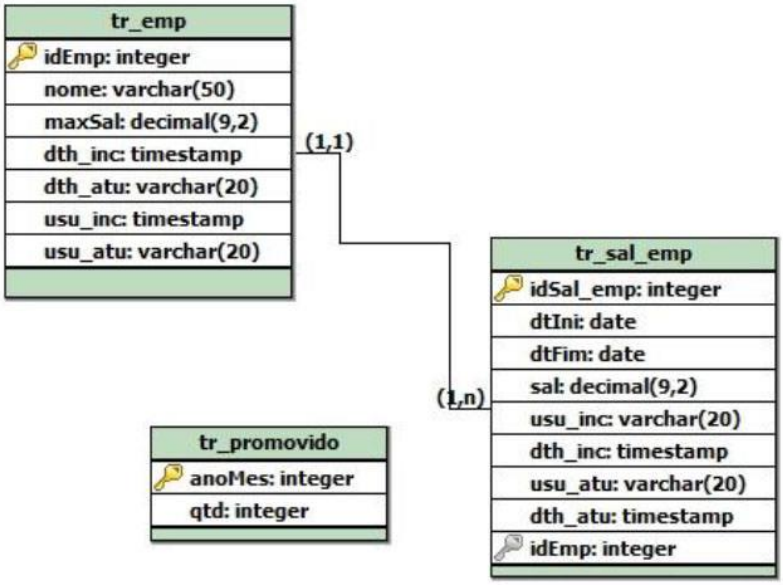
   b. Criar trigger para popular a tabela de auditoria conforme são realizadas as operações na tabela TR_EMP;

3. Definição das Tabelas:

OBS.: Em **.sql** no mesmo diretório.

4. ER das Tabelas:





1.

```
CREATE OR REPLACE FUNCTION
fn_01() RETURNS TRIGGER AS $$

DECLARE

_MAX_SAL decimal(9,2);
_DATA_FIM date;
_EXISTE int;
_TOTAL int;

BEGIN
_MAX_SAL =      (       select  maxSal
                        from    tr_emp
                        where   idEmp = NEW.idEmp                       );

_DATA_FIM =     (       select  dtFim
                        from    tr_sal_emp
                        where   idEmp = NEW.idEmp
                        order by dtFim desc
                        limit 1                                         );

_EXISTE =       (       select  anoMes
                        from    tr_promovido, tr_sal_emp
                        where   anoMes  = cast(to_char(dth_inc, 'YYYYMM') as int)
                        limit 1                                         );

IF NEW.sal > _MAX_SAL THEN
        RAISE EXCEPTION 'Salário Superior ao Permitido pelo Sistema';
ELSEIF TG_OP = 'INSERT' AND NEW.dtIni < DATA_FIM THEN
        RAISE EXCEPTION 'Data de Início entre a data de outro Salário';
END IF;

IF TG_OP = 'INSERT' AND (to_char(NEW.dtIni, 'YYYY-MM')) < '2012-01' AND NEW.idEmp IN (10,20,30) THEN
        NEW.dtFim = '2012-12-31';
ELSEIF TG_OP = 'INSERT' AND (to_char(NEW.dtIni, 'YYYY-MM')) > '2010-12' THEN
        NEW.dtFim = null;
END IF;

IF TG_OP = 'INSERT' THEN
        NEW.usu_inc = CURRENT_USER;
        NEW.dth_inc = CURRENT_TIMESTAMP;
        NEW.usu_atu = CURRENT_USER;
        NEW.dth_atu = CURRENT_TIMESTAMP;
ELSEIF TG_OP = 'UPDATE' THEN
        NEW.usu_atu = CURRENT_USER;
        NEW.dth_atu = CURRENT_TIMESTAMP;
END IF;

GET DIAGNOSTICS _TOTAL = ROW_COUNT;

IF TG_OP = 'INSERT' THEN
        IF _EXISTE IS NULL THEN
                insert into tr_promovido (anoMes, qtd) values (cast(to_char(now(), 'YYYYMM') as int), _TOTAL);
        ELSE
                update tr_promovido set qtd = qtd + 1 where _EXISTE = anoMes;
        END IF;
END IF;

RETURN NEW;

END;
$$ LANGUAGE PLpgSQL;

-- TRIGGER:

CREATE TRIGGER tg_01
BEFORE INSERT OR UPDATE OR DELETE ON tr_sal_emp
FOR EACH ROW EXECUTE PROCEDURE fn_01();
```

2.

```
CREATE OR REPLACE FUNCTION
fn_02_log() RETURNS TRIGGER AS $$

BEGIN

IF TG_OP = 'UPDATE' THEN

        insert into tr_emp_log (operacao, usuario, dataHora, maxSalAnt, maxSalDep)
        values (TG_OP, CURRENT_USER, now(), OLD.maxSal, NEW.maxSal);

ELSEIF TG_OP = 'INSERT' THEN

        insert into tr_emp_log (operacao, usuario, dataHora, maxSalDep)
        values (TG_OP, CURRENT_USER, now(), NEW.maxSal);

ELSEIF TG_OP = 'DELETE' THEN

        insert into tr_emp_log (operacao, usuario, dataHora, maxSalAnt)
        values (TG_OP, CURRENT_USER, now(), OLD.maxSal);

END IF;

RETURN NEW;

END;
$$ LANGUAGE PLpgSQL;


-- TRIGGER:

CREATE TRIGGER tg_02_log
AFTER INSERT OR UPDATE OR DELETE ON tr_emp
FOR EACH ROW EXECUTE PROCEDURE fn_02_log();
```