Grupo05

Lista de exercícios

Semana 02

Eberton Chaves Dias – 11621BSI215

Elisângela Rithiely Evaristo de Souza - 11811BSI251

Gabriel Miranda Silva - 11621BSI222

```
--É preciso atualizar a informação do saldo do cliente na tabela cliente.
--para este propósito devemos levar em conta o saldo dos depósitos menos os
--saldos de empréstimos. o cálculo final deve ser armazenado na tabela
conta.

--CLIENTES QUE POSSUEM APENAS DEPOSITOS
--Q01:
SELECT     NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA,
        SUM(SALDO_DEPOSITO) AS TOTAL
FROM DEPOSITO
WHERE NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA NOT IN
      (SELECT NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA FROM EMPRESTIMO)
GROUP BY NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA


--CLIENTES QUE POSSUEM APENAS EMPRESTIMOS
--Q02:
SELECT     NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA,
        -1*SUM(VALOR_EMPRESTIMO) AS TOTAL
FROM EMPRESTIMO
WHERE NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA NOT IN
      (SELECT NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA FROM DEPOSITO)
GROUP BY NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA


--CLIENTES QUE POSSUEM DEPOSITO E EMPRESTIMOS (AMBOS)
--Q03:
SELECT     E.NOME_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA,
      SUM(D.SALDO_DEPOSITO)-SUM(E.VALOR_EMPRESTIMO) AS TOTAL
FROM EMPRESTIMO AS E, DEPOSITO AS D
WHERE E.NOME_CLIENTE = D.NOME_CLIENTE
AND E.NOME_AGENCIA = D.NOME_AGENCIA
```

```
AND E.NUMERO_CONTA = D.NUMERO_CONTA
GROUP BY E.NOME_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA


--CLIENTES QUE POSSUEM DEPOSITOS, EMPRESTIMOS OU AMBOS
--Q04:
(SELECT    NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA, --SOMENTE DEPOSITOS
        SUM(SALDO_DEPOSITO) AS TOTAL
FROM DEPOSITO
WHERE NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA NOT IN
     (SELECT NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA FROM EMPRESTIMO)
GROUP BY NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA)
UNION
(SELECT    NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA, --SOMENTE EMPRESTIMOS
        -1*SUM(VALOR_EMPRESTIMO) AS TOTAL
FROM EMPRESTIMO
WHERE NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA NOT IN
     (SELECT NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA FROM DEPOSITO)
GROUP BY NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA)
UNION
(SELECT    E.NOME_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA, -- AMBOS
     SUM(D.SALDO_DEPOSITO)-SUM(E.VALOR_EMPRESTIMO) AS TOTAL
FROM EMPRESTIMO AS E, DEPOSITO AS D
WHERE E.NOME_CLIENTE = D.NOME_CLIENTE
AND E.NOME_AGENCIA = D.NOME_AGENCIA
AND E.NUMERO_CONTA = D.NUMERO_CONTA
GROUP BY E.NOME_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA)


--Agora utilizamos a Q04 como relatório para o comando de atualização
```

```
UPDATE CONTA SET SALDO_CONTA = 0;
SELECT NOME_CLIENTE, SALDO_CONTA FROM CONTA ORDER BY SALDO_CONTA DESC;


--Q05:
UPDATE CONTA SET SALDO_CONTA = R.TOTAL
FROM (
     (SELECT    NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA, --SOMENTE
DEPOSITOS
          SUM(SALDO_DEPOSITO) AS TOTAL
     FROM DEPOSITO
     WHERE NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA NOT IN
          (SELECT NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA FROM
EMPRESTIMO)
     GROUP BY NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA)
     UNION
     (SELECT    NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA, --SOMENTE
EMPRESTIMOS
          -1*SUM(VALOR_EMPRESTIMO) AS TOTAL
     FROM EMPRESTIMO
     WHERE NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA NOT IN
          (SELECT NOME_CLIENTE || NOME_AGENCIA || NUMERO_CONTA FROM
DEPOSITO)
     GROUP BY NOME_CLIENTE, NOME_AGENCIA, NUMERO_CONTA)
     UNION
     (SELECT    E.NOME_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA, -- AMBOS
          SUM(D.SALDO_DEPOSITO)-SUM(E.VALOR_EMPRESTIMO) AS TOTAL
     FROM EMPRESTIMO AS E, DEPOSITO AS D
     WHERE E.NOME_CLIENTE = D.NOME_CLIENTE
     AND E.NOME_AGENCIA = D.NOME_AGENCIA
     AND E.NUMERO_CONTA = D.NUMERO_CONTA
     GROUP BY E.NOME_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA)
) AS R
```

```
WHERE   CONTA.NOME_CLIENTE = R.NOME_CLIENTE
AND   CONTA.NOME_AGENCIA = R.NOME_AGENCIA
AND   CONTA.NUMERO_CONTA = R.NUMERO_CONTA


--
-- PERCEBAM QUE ESTA CONSULTA AINDA NÃO FEZ UM RELATÓRIO COMPLETO
-- DOS CLIENTES DO BANCO PORQUE EXISTEM CLIENTES QUE NÃO FIZERAM
-- NEM EMPRESTIMOS E NEM DEPÓSITOS, MAS AINDA ASSIM POSSUEM UMA
-- CONTA NO BANCO.




-- EXISTE UM MODO MAIS FÁCIL DE REALIZAR ESTAS OPERAÇÕES?




-- COM A UTILIZAÇÃO DE JUNÇÕES, ESCREVEMOS CONSULTAS MENORES, MAIS
-- SIMPLES, DETERMINANDO COMO AS LINHAS DE CADA TABELA DEVEM SER
-- RETORNADAS DE ACORDO COM A EXISTÊNCIA OU NÃO DE UMA LINHA
-- CORRESPONDENTE NA OUTRA TABELA

--ANTES DE FAZER A CONSULTA COMPLETA, VAMOS GRADUAR A COMPLEXIDADE:
--COMEÇAMOS COM A CONSULTA PARA RETORNAR CLIENTES QUE POSSUEM
--EMPRESTIMOS E DEPOSITOS (AO MESMO TEMPO)
```

```
--Q06:
SELECT     E.NOME_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA,
      SUM(D.SALDO_DEPOSITO)-SUM(E.VALOR_EMPRESTIMO) AS TOTAL
FROM EMPRESTIMO AS E INNER JOIN DEPOSITO AS D
ON    E.NOME_CLIENTE = D.NOME_CLIENTE
AND         E.NOME_AGENCIA = D.NOME_AGENCIA
AND         E.NUMERO_CONTA = D.NUMERO_CONTA
GROUP BY E.NOME_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA




-- UTILIZANDO A CLÁUSULA "NATURAL", ENTÃO A CLÁUSULA "ON" NÃO
-- NECESSITA SER UTILIZADA E A COMPLEXIDADE DA CONSULTA SERÁ
-- REDUZIDA COMPARANDO COM A CONSULTA INICIAL, NA QUAL TODAS
-- AS CHAVES DEVIAM SER COMPARADAS AOS PARES.
--REESCRITA POR MEIO DE JUNÇÕES. NESTE CASO TEMOS:
--Q07:
SELECT E.NOME_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA,
      SUM(D.SALDO_DEPOSITO)-SUM(E.VALOR_EMPRESTIMO) AS TOTAL
FROM EMPRESTIMO AS E NATURAL INNER JOIN DEPOSITO AS D
GROUP BY E.NOME_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA

--CONCLUSÃO:
--É MELHOR FAZER USO DA CLÁUSULA NATURAL QUANDO POSSÍVEL.




-- QUER-SE RETORNAR DADOS EXCLUSIVOS DA TABELA CLIENTE EM UM JOIN
```

```
-- COM A TABELA DEPOSITO. CASO EXISTAM DEPÓSITO DEVE SER MOSTRADO
-- E CASO NÃO EXISTAM DEPÓSITOS MOSTRAR APENAS OS DADOS DO CLIENTE

--Q08:

SELECT     C.NOME_CLIENTE, C.CIDADE_CLIENTE, D.NOME_AGENCIA,
        D.NUMERO_CONTA, SUM(D.SALDO_DEPOSITO) AS DEP
FROM CLIENTE AS C NATURAL LEFT OUTER JOIN DEPOSITO AS D
GROUP BY C.NOME_CLIENTE, C.CIDADE_CLIENTE, D.NOME_AGENCIA, D.NUMERO_CONTA
ORDER BY C.NOME_CLIENTE, C.CIDADE_CLIENTE, D.NOME_AGENCIA, D.NUMERO_CONTA

-- AS LINHAS DA TABELA À ESQUERDA QUE ESTIVEREM FORA DO JOIN
-- SERÃO RETORNADAS, PORÉM COM CONTEÚDO NULO NOS DADOS NÃO
-- AGRUPADOS.


--Q09: O MESMO ENTRE CLIENTE E EMPRESTIMO

SELECT     C.NOME_CLIENTE, C.CIDADE_CLIENTE, E.NOME_AGENCIA,
         E.NUMERO_CONTA, SUM(E.VALOR_EMPRESTIMO) AS EMP
FROM CLIENTE AS C NATURAL LEFT OUTER JOIN EMPRESTIMO AS E
GROUP BY C.NOME_CLIENTE, C.CIDADE_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA
ORDER BY C.NOME_CLIENTE, C.CIDADE_CLIENTE, E.NOME_AGENCIA, E.NUMERO_CONTA

-- ISSO NÃO É UMA FALHA, MAS UMA VIRTUDE DO JOIN PORQUE NOS PERMITE
-- RESPONDER À SEGUINTE PERGUNTA:

--************************************************************
-- "RETORNE TODOS OS CLIENTES DO BANCO COM SUAS RESPECTIVAS SOMAS
-- "DE DEPÓSITOS E EMPRESTIMOS CASO EXISTAM"
--************************************************************
```

```
--Q10a:
SELECT     C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA,
      SUM(D.SALDO_DEPOSITO) AS TOTAL_DEP,
      SUM(E.VALOR_EMPRESTIMO) AS TOTAL_EMP
FROM CONTA AS C NATURAL LEFT OUTER JOIN
           (EMPRESTIMO AS E NATURAL LEFT OUTER JOIN DEPOSITO AS D)
GROUP BY C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA


--Q10b:
SELECT     C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA,
      SUM(D.SALDO_DEPOSITO) AS TOTAL_DEP,
      SUM(E.VALOR_EMPRESTIMO) AS TOTAL_EMP
FROM CONTA AS C NATURAL LEFT JOIN
           (EMPRESTIMO AS E NATURAL FULL JOIN DEPOSITO AS D)
GROUP BY C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA



--Agora a atualizacao do saldo da conta dos clientes fica
--mais simples do que a Q05
--Q10c ou Q05b:
UPDATE CONTA SET SALDO_CONTA = RELATORIO.TOTAL
FROM(
      SELECT     C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA,
           SUM(D.SALDO_DEPOSITO) - SUM(E.VALOR_EMPRESTIMO) AS TOTAL
      FROM CONTA AS C NATURAL LEFT JOIN
                 (EMPRESTIMO AS E NATURAL FULL JOIN DEPOSITO AS D)
      GROUP BY C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA
) AS RELATORIO
```

```
WHERE  CONTA.NOME_CLIENTE = RELATORIO.NOME_CLIENTE
AND    CONTA.NOME_AGENCIA = RELATORIO.NOME_AGENCIA
AND    CONTA.NUMERO_CONTA = RELATORIO.NUMERO_CONTA



--SERÁ QUE AGORA ESTÁ CERTO?
SELECT NOME_CLIENTE, SALDO_CONTA FROM CONTA ORDER BY SALDO_CONTA DESC;



--Q10d
SELECT     C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA,
      COALESCE(SUM(D.SALDO_DEPOSITO  ),0) AS TOTAL_DEP,
      COALESCE(SUM(E.VALOR_EMPRESTIMO),0) AS TOTAL_EMP
FROM CONTA AS C NATURAL LEFT JOIN
           (EMPRESTIMO AS E NATURAL FULL JOIN DEPOSITO AS D)
GROUP BY C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA



--Q10e
SELECT     C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA,
      COALESCE(SUM(D.SALDO_DEPOSITO),0) -
COALESCE(SUM(E.VALOR_EMPRESTIMO),0) AS TOTAL
FROM CONTA AS C NATURAL LEFT JOIN
           (EMPRESTIMO AS E NATURAL FULL JOIN  DEPOSITO AS D)
GROUP BY C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA
```

```
--Q10f ou Q05c
UPDATE CONTA SET SALDO_CONTA = RELATORIO.TOTAL
FROM(
     SELECT    C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA,
          COALESCE(SUM(D.SALDO_DEPOSITO),0) -
COALESCE(SUM(E.VALOR_EMPRESTIMO),0) AS TOTAL
     FROM CONTA AS C NATURAL LEFT JOIN
               (EMPRESTIMO AS E NATURAL FULL JOIN DEPOSITO AS D)
     GROUP BY C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA
) AS RELATORIO
WHERE CONTA.NOME_CLIENTE = RELATORIO.NOME_CLIENTE
AND   CONTA.NOME_AGENCIA = RELATORIO.NOME_AGENCIA
AND   CONTA.NUMERO_CONTA = RELATORIO.NUMERO_CONTA


--SERÁ QUE AGORA ESTÁ CERTO?
SELECT NOME_CLIENTE, SALDO_CONTA FROM CONTA ORDER BY SALDO_CONTA DESC;


update conta set saldo_conta=0;
--PRIMEIRO CONFERIMOS COMO ESTá A SITUAçãO DAS CONTAS
select * from conta;

--EXECUTAMOS OUTRA CONFERENCIA PARA VER COMO SERãO OS RETORNOS DA PESQUISA
select getliquido(numero_conta, nome_agencia, nome_cliente) from conta;

--DEPOIS ATUALIZAMOS ...
update conta set saldo_conta=getliquido(numero_conta, nome_agencia,
nome_cliente);

--... E CONFERINDO DE NOVO:
```

```
select * from conta where nome_agencia = 'PUC';


--A DEFINIÇÃO DA FUNÇÃO GETLIQUIDO
CREATE OR REPLACE FUNCTION getliquido(p_numero_conta integer,
p_nome_agencia character varying, p_nome_cliente character varying)
  RETURNS float AS
$BODY$
DECLARE
    saldo_liquido float;
    soma_deposito float;
    soma_emprestimo float;
    cursor_relatorio CURSOR FOR SELECT SUM(D.SALDO_DEPOSITO) AS TOTAL_DEP,
           SUM(E.VALOR_EMPRESTIMO) AS TOTAL_EMP
           FROM CONTA AS C NATURAL LEFT OUTER JOIN
           (EMPRESTIMO AS E NATURAL FULL JOIN DEPOSITO AS D)
     WHERE C.NOME_CLIENTE=p_nome_cliente AND C.NOME_AGENCIA=p_nome_agencia
AND C.NUMERO_CONTA=p_numero_conta
     GROUP BY C.NOME_CLIENTE, C.NOME_AGENCIA, C.NUMERO_CONTA;
BEGIN

    OPEN cursor_relatorio;
        saldo_liquido=0;
        FETCH cursor_relatorio INTO soma_deposito, soma_emprestimo;
        RAISE NOTICE 'O valor de DEP é % e EMP é %', soma_deposito,
soma_emprestimo;
        IF FOUND THEN
          IF soma_deposito IS NULL THEN soma_deposito=0; END IF;
          IF soma_emprestimo IS NULL then soma_emprestimo=0; END IF;
             saldo_liquido = soma_deposito - soma_emprestimo ;
        END IF;
    CLOSE cursor_relatorio;
    RETURN saldo_liquido;
```

```
END
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION getliquido(integer, character varying, character varying)
  OWNER TO aluno;



--PRIMEIRO CONFERIMOS COMO ESTá A SITUAçãO DAS CONTAS
select * from emprestimo where nome_agencia = 'PUC';


--EXECUTAMOS OUTRA CONFERENCIA PARA VER COMO SERãO OS RETORNOS DA PESQUISA
select conta.*, update_valor_emprestimo1(numero_conta, nome_agencia,
nome_cliente) from conta where nome_agencia = 'PUC';


--A DEFINIÇÃO DA FUNÇÃO update_valor_emprestimo1
CREATE OR REPLACE FUNCTION update_valor_emprestimo1(  p_numero_conta
integer,
                                      p_nome_agencia character varying,
                                      p_nome_cliente character varying
                                  )
  RETURNS float AS
$BODY$
DECLARE
    l_valor_emprestimo float;
    l_valor_juros float;
    cursor_relatorio CURSOR FOR SELECT VALOR_EMPRESTIMO, JUROS_EMPRESTIMO
     FROM EMPRESTIMO
     WHERE NOME_CLIENTE=p_nome_cliente AND NOME_AGENCIA=p_nome_agencia AND
NUMERO_CONTA=p_numero_conta;
BEGIN
    OPEN cursor_relatorio;
```

```
     FETCH cursor_relatorio INTO l_valor_emprestimo, l_valor_juros;
     IF FOUND THEN
          l_valor_emprestimo = l_valor_emprestimo *
(1+((l_valor_juros)/100));
          UPDATE EMPRESTIMO SET VALOR_EMPRESTIMO = l_valor_emprestimo
WHERE
          NOME_CLIENTE=p_nome_cliente AND NOME_AGENCIA=p_nome_agencia AND
NUMERO_CONTA=p_numero_conta;
     END IF;
    CLOSE cursor_relatorio;
    RETURN l_valor_emprestimo;
END
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION update_valor_emprestimo1(integer, character varying,
character varying)
  OWNER TO aluno;


--Lista as quantidades de emprestimos realizados por cada cliente
select nome_cliente, count(1) from emprestimo group by nome_cliente order
by count(1) desc;


--Lista apenas um deles
select * from emprestimo where nome_cliente = 'Reinaldo Pereira da Silva';


--Agora atualiza apenas os valores de empréstimos desta pessoa
select update_valor_emprestimo2('Reinaldo Pereira da Silva');


--A DEFINIÇÃO DO PROCEDIMENTO update_valor_emprestimo2
CREATE OR REPLACE FUNCTION update_valor_emprestimo2(p_nome_cliente
character varying )
  RETURNS void as
```

```
$BODY$
DECLARE
     l_valor_emprestimo float;
     l_valor_juros float;
     l_numero_emprestimo integer;
    cursor_relatorio CURSOR FOR SELECT VALOR_EMPRESTIMO, JUROS_EMPRESTIMO,
NUMERO_EMPRESTIMO
                      FROM EMPRESTIMO
                      WHERE NOME_CLIENTE=p_nome_cliente;
BEGIN
   OPEN cursor_relatorio;
     LOOP
           FETCH cursor_relatorio INTO l_valor_emprestimo, l_valor_juros,
l_numero_emprestimo;
           IF FOUND THEN
                l_valor_emprestimo = l_valor_emprestimo * (1+
((l_valor_juros)/100));
                UPDATE EMPRESTIMO SET VALOR_EMPRESTIMO = l_valor_emprestimo
WHERE
                NUMERO_EMPRESTIMO=l_numero_emprestimo;
           END IF;
           IF not FOUND THEN EXIT;
           END IF;
     END LOOP;
   CLOSE cursor_relatorio;
   RETURN;
END
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION update_valor_emprestimo2(character varying)
  OWNER TO aluno;
```

```
--IMPLEMENTE UM GATILHO (TRIGGER) QUE ATUALIZE A TABELA CONTA, PARA O CAMPO SALDO_CONTA,
--SEMPRE QUE UMA NOVA LINHA FOR INSERIDA NA TABELA DE DEPÓSITO OU EMPRÉSTIMO.


CREATE OR REPLACE FUNCTION atualiza_saldo_conta()
	RETURNS TRIGGER AS
$BODY$
DECLARE
	local_total_deposito FLOAT;
	local_total_emprestimo FLOAT;
	local_numero_conta_deposito INTEGER;
	local_numero_conta_emprestimo INTEGER;
	local_nome_agencia_deposito CHARACTER VARYING;
	local_nome_agencia_emprestimo CHARACTER VARYING;
	cursor_deposito CURSOR FOR SELECT C.NUMERO_CONTA, C.NOME_AGENCIA,
	COALESCE(SUM(D.SALDO_DEPOSITO),0) AS TOTAL_DEPOSITO
	FROM CONTA AS C NATURAL LEFT OUTER JOIN DEPOSITO AS D
	GROUP BY C.NUMERO_CONTA, C.NOME_AGENCIA
	ORDER BY C.NUMERO_CONTA;
	cursor_emprestimo CURSOR FOR SELECT C.NUMERO_CONTA, C.NOME_AGENCIA,
	COALESCE(SUM(E.VALOR_EMPRESTIMO),0) AS TOTAL_EMPRESTIMO
	FROM CONTA AS C NATURAL LEFT OUTER JOIN EMPRESTIMO AS E
	GROUP BY C.NUMERO_CONTA, C.NOME_AGENCIA
	ORDER BY C.NUMERO_CONTA;
BEGIN
	RAISE NOTICE 'GALVÃO! FALA TINO! SENTIU!';
	OPEN cursor_deposito;
	LOOP
```

```
          FETCH cursor_deposito INTO local_numero_conta_deposito,
local_nome_agencia_deposito, local_total_deposito;
          IF FOUND THEN
               FOR emprestiminho in cursor_emprestimo
               LOOP
                    IF
local_numero_conta_deposito=emprestiminho.NUMERO_CONTA
                    AND
local_nome_agencia_deposito=emprestiminho.NOME_AGENCIA THEN
                         UPDATE CONTA SET SALDO_CONTA =
local_total_deposito - emprestiminho.TOTAL_EMPRESTIMO
                         WHERE NUMERO_CONTA = local_numero_conta_deposito
                         AND NOME_AGENCIA = local_nome_agencia_deposito;
                    EXIT;
                    END IF;
               END LOOP;
          END IF;
          IF NOT FOUND THEN EXIT; END IF;
     END LOOP;
     CLOSE cursor_deposito;
     RETURN NULL;
END
$BODY$
LANGUAGE plpgsql VOLATILE COST 100;
ALTER FUNCTION atualiza_saldo_conta() OWNER TO aluno;



CREATE TRIGGER TRIGGER_atualiza_saldo_conta_from_deposito
AFTER UPDATE ON DEPOSITO
FOR EACH STATEMENT EXECUTE PROCEDURE atualiza_saldo_conta();
```

```
CREATE TRIGGER TRIGGER_atualiza_saldo_conta_from_deposito_insert
AFTER INSERT ON DEPOSITO
FOR EACH STATEMENT EXECUTE PROCEDURE atualiza_saldo_conta();

CREATE TRIGGER TRIGGER_atualiza_saldo_conta_from_emprestimo
AFTER UPDATE ON EMPRESTIMO
FOR EACH STATEMENT EXECUTE PROCEDURE atualiza_saldo_conta();

CREATE TRIGGER TRIGGER_atualiza_saldo_conta_from_emprestimo_insert
AFTER INSERT ON EMPRESTIMO
FOR EACH STATEMENT EXECUTE PROCEDURE atualiza_saldo_conta();
```