LABORATÓRIO 2

FUNÇÕES, STORED PROCEDURES E TRIGGERS

COMPLEMENTOS DE BASES DE DADOS

Licenciatura em Engenharia Informática

Grupo Nº 6

Alexandre Coelho, Nº 190221093

Sérgio Veríssimo, Nº 190221128

Índice

Etapa 1

a) Criar a função fnTotalVendasCliente que calcule o valor total das vendas para umdeterminado cliente.

```
CREATE OR ALTER FUNCTION
[SalesLT].fnTotalVendasCliente (@idUser INT)
RETURNS FLOAT
AS BEGIN
    DECLARE @Vendas FLOAT
    SELECT @Vendas = SUM(soh.TotalDue)
    FROM [SalesLT].SalesOrderHeader soh
    WHERE soh.CustomerID = @idUser
    RETURN @Vendas
END
```

b) Utilizando a função anterior, faça uma função fnTotalVendas que calcule o total de vendas efetuado;

```
CREATE OR ALTER FUNCTION [SalesLT].fnTotalVendas ()
RETURNS FLOAT
AS BEGIN
    DECLARE @Vendas FLOAT
    SELECT @Vendas = SUM(soh.TotalDue)
    FROM [SalesLT].SalesOrderHeader soh
    RETURN @Vendas
END
```

c) Criar o procedimento spClientesCidade que recebe uma cidade (ex: Las Vegas) e lista os clientes residentes na respetiva cidade.

```
CREATE OR ALTER PROCEDURE spClientesCidade @City
varchar(30)
AS
  SELECT *
  FROM [SalesLT].Customer c
  JOIN [SalesLT].CustomerAddress ca
  ON ca.CustomerID=c.CustomerID
  JOIN [SalesLT].Address a
  ON a.AddressID=ca.AddressID
  WHERE a.City = @City
```

d) Criar o procedimento spListaCompras, que liste para um dado cliente, as compras com o respetivo detalhe. Utilize cursores e a instrução print para gerar o output.

```
CREATE OR ALTER PROCEDURE spListaCompras @Client
int
AS
  DECLARE @SalesOrderNumber varchar(10),
            @OrderDate date,
            @DueDate date,
            @ShipDate date,
            @Status int,
            @PurchaseOrderNumber varchar(20),
            @AccountNumber varchar(20),
            @ShipMethod varchar(50),
            @Freight float,
            @TotalDue float
  DECLARE checkCursor CURSOR FOR
       SELECT CAST(soh.SalesOrderNumber AS
       varchar), CAST(soh.OrderDate AS date),
       CAST(soh.DueDate AS date), CAST(soh.ShipDate
       AS date), CAST(soh.Status AS int),
       CAST(soh.PurchaseOrderNumber AS varchar),
       CAST(soh.AccountNumber AS varchar),
       CAST(soh.ShipMethod AS varchar),
       CAST(soh.Freight AS float),
       CAST(soh.TotalDue AS float)
       FROM [SalesLT].Customer c
       JOIN [SalesLT].SalesOrderHeader soh
       ON soh.CustomerID=c.CustomerID
       WHERE c.CustomerID=@Client
  OPEN checkCursor
  While @@Fetch_Status = 0 Begin
  PRINT 'SalesOrderNumber: ' +
  CAST(@SalesOrderNumber AS varchar) +
  ' | OrderDate: ' + CAST(@OrderDate AS varchar) +
  ' | DueDate: ' + CAST(@DueDate AS varchar) +
  ' | ShipDate: ' + CAST(@ShipDate AS varchar) +
  ' |  Status: ' + CAST(@Status AS varchar) +
```

```
' | PurchaseOrderNumber: ' +
CAST(@PurchaseOrderNumber AS varchar) +
' | AccountNumber: ' + CAST(@AccountNumber AS
varchar) +
' | ShipMethod: ' + CAST(@ShipMethod AS varchar) +
' |    Freight: ' + CAST(@Freight AS varchar) +
' |    TotalDue: ' + CAST(@TotalDue AS varchar)
  FETCH NEXT FROM checkCursor INTO
  @SalesOrderNumber, @OrderDate, @DueDate,
  @ShipDate, @Status, @PurchaseOrderNumber,
  @AccountNumber, @ShipMethod, @Freight, @TotalDue
  END
  CLOSE checkCursor
  DEALLOCATE checkCursor
```

Etapa 2

a) Criar uma função fnCodificaPassword que codifica uma password em SHA1 (utilize a função HASHBYTES). A função recebe a password e retorna a sua codificação.

```
CREATE OR ALTER FUNCTION
[SalesLT].fnCodificaPassword (@Password varchar)
RETURNS varbinary(200)
AS BEGIN
    DECLARE @HashPassword varbinary(200)
  SELECT @HashPassword = HASHBYTES('SHA1',
  CONVERT(nvarchar,@Password))
    RETURN @HashPassword
END
```

b) Crie a tabela "CustomerPW", para guardar a password de novos clientes.

```
CREATE TABLE [SalesLT].CustomerPW (
       CustomerPwID int not null,
       PasswordHash varbinary(250),
       NameStyle bit,
       Title nvarchar(8) null,
       FirstName nvarchar(50) not null,
       MiddleName nvarchar(50) null,
```

```
        LastName nvarchar(50) not null,
        Suffix nvarchar(10) null,
        CompanyName nvarchar(128) null,
        SalesPerson nvarchar(256) null,
        EmailAddress nvarchar(50) null,
        Phone nvarchar(25) null,
        ModifiedDate datetime not null
    );
```

c) Criar o procedimento spNovoCliente, que recebe os dados obrigatórios para um novo cliente (ver tabela Customer) mais a nova password, e faz o registo nas tabelas Customer e CustomerPW. A password é guardada em SHA1.

```
CREATE OR ALTER PROCEDURE [SalesLT].spNovoCliente (
@CustomerID int,
@CustomerPW varchar(250),
@FirstName nvarchar(50),
@LastName nvarchar(50),
@EmailAdress nvarchar(50))
AS
BEGIN
  DECLARE @pw varchar(300)
  select @pw =
  [SalesLT].fnCodificaPassword(@CustomerPW)
  INSERT INTO SalesLT.CustomerPW(CustomerID,
  CustomerPW,FirstName,LastName,EmailAddress,Modifi
  edDAte)
  Values
  (@CustomerID,convert(varbinary(250),@pw),@FirstNa
  me,@LastName,@EmailAdress,GETDATE());
END
```

d) Verifique a autenticação (EmailAddress/Password) de um utilizador através de uma função fnAutenticar, que devolve o ID do utilizador se a autenticação é válida ou 0 se inválida.

```
CREATE OR ALTER FUNCTION
[SalesLT].fnAutenticar(@email varchar(60),
@password varchar(60))
RETURNS int
BEGIN
```

```
    DECLARE @CustomerIdentification int
    SELECT @CustomerIdentification = CustomerPwID
    FROM SalesLT.CustomerPW
    WHERE EmailAddress = @email
    AND CustomerPW =
    [SalesLT].fnCodificaPassword(@password)
    RETURN isnull(@CustomerIdentification, 0)
END
```

Etapa 3

a) Crie a tabela CustomerLog e os triggers necessários, de modo a implementar um mecanismo de auditoria sobre a tabela Customer. A tabela CustomerLog para além dos atributos da tabela Customer, devem ser adicionados os seguintes atributos:

- **Log_Data: timestamp da alteração;**
- **Log_Operacao: 'U' – update, 'D' – delete**

```
CREATE TABLE [SalesLT].CustomerLog(
        Log_data datetime not null,
        Log_Operacao char(1) check(Log_Operacao IN
        ('U','D')) not null,
        CustomerID int not null,
        NameStyle bit,
        Title nvarchar(8) null,
        FirstName nvarchar(50) not null,
        MiddleName nvarchar(50) null,
        LastName nvarchar(50) not null,
        Suffix nvarchar(10) null,
        CompanyName nvarchar(128) null,
        SalesPerson nvarchar(256) null,
        EmailAddress nvarchar(50) null,
        Phone nvarchar(25) null,
        PasswordHash varchar(128) not null,
        PasswordSalt varchar(10) not null,
        rowguid uniqueidentifier not null,
        ModifiedDate datetime not null
);
```

- **Quando se altera ou se apaga um registo da tabela Customer, deve ser executada uma cópia do registo que sofreu as alterações para a tabela de log, adicionando o rowversion e o tipo de operação.**

```
CREATE TRIGGER CustomerLogs ON
[SalesLT].Customer FOR DELETE, UPDATE
AS
BEGIN
  IF EXISTS (select 0 FROM DELETED)
  BEGIN
    IF EXISTS (SELECT 0 FROM Inserted)
       INSERT INTO
       SalesLT.CustomerLog(Log_data,Log_Operacao,Cu
       stomerID,NameStyle,Title,FirstName,MiddleNam
       e,LastName,Suffix,CompanyName,SalesPerson,Em
       ailAddress,Phone,PasswordHash,PasswordSalt,r
       owguid,ModifiedDate)
       SELECT
       CURRENT_TIMESTAMP,'U',U.CustomerID,U.NameSty
       le,U.Title,U.FirstName,U.MiddleName,U.LastNa
       me,U.Suffix,U.CompanyName,U.SalesPerson,U.Em
       ailAddress,U.Phone,U.PasswordHash,U.Password
       Salt,U.rowguid,U.ModifiedDate from DELETED U
     ELSE
       INSERT INTO
       SalesLT.CustomerLog(Log_data,Log_Operacao,Cu
       stomerID,NameStyle,Title,FirstName,MiddleNam
       e,LastName,Suffix,CompanyName,SalesPerson,Em
       ailAddress,Phone,PasswordHash,PasswordSalt,r
       owguid,ModifiedDate)
       SELECT
       CURRENT_TIMESTAMP,'D',D.CustomerID,D.NameSty
       le,D.Title,D.FirstName,D.MiddleName,D.LastNa
       me,D.Suffix,D.CompanyName,D.SalesPerson,D.Em
       ailAddress,D.Phone,D.PasswordHash,D.Password
       Salt,D.rowguid,D.ModifiedDate from DELETED D
  END
END
```

b) Crie os triggers necessários para implementar as seguintes funcionalidades:

- **Quando se atualiza a password de um cliente, esta deve ser guardada na tabela CustomerPW com codificação em SHA1;**

```
GO
```

```
CREATE TRIGGER TrgNewPW ON
[SalesLT].Customer AFTER UPDATE
AS
BEGIN
  IF UPDATE(PasswordHash)
  BEGIN
       INSERT INTO
       [SalesLT].CustomerPW(CustomerID,CustomerPW,N
       ameStyle,Title,FirstName,MiddleName,LastName
       ,Suffix,CompanyName,SalesPerson,EmailAddress
       ,Phone,ModifiedDate)
       SELECT
       U.CustomerID,hashbytes('SHA1',U.PasswordHash
       ),U.NameStyle, U.Title,U.FirstName,
       U.MiddleName, U.LastName, U.Suffix,
       U.CompanyName, U.SalesPerson,
       U.EmailAddress, U.Phone, GETDATE()
       FROM DELETED U
  END
END
GO
```

- **Verifique a restrição de não poderem existir utilizadores com o mesmo login (EmailAddress).**

```
GO
CREATE TRIGGER TrgExists
ON [SalesLT].CustomerPW
FOR INSERT
AS
BEGIN
  DECLARE @Email varchar(60)
  SELECT @Email = EmailAddress FROM inserted
  IF((select count(*) FROM [SalesLT].CustomerPW
WHERE EmailAddress = @Email)>1)
  BEGIN
       ROLLBACK
       RAISERROR('Email Duplicado!', 16, 1)
  END
END
GO
```

Etapa 4

a) Utilizando a tabela CustomerLog, faça um procedimento que reponha o estado inicial da tabela Customer antes das alterações.

```
CREATE OR ALTER PROCEDURE [SalesLT].estadoInicial
AS
  DELETE c
  FROM [SalesLT].Customer c
  JOIN [SalesLT].CustomerLog cl
  ON cl.CustomerID = c.CustomerID
  WHERE cl.Log_data>=c.ModifiedDate

  SET IDENTITY_INSERT [SalesLT].Customer ON

  INSERT INTO [SalesLT].Customer(
            CustomerID,
            NameStyle,
            Title,
            FirstName,
            MiddleName,
            LastName,
            Suffix,
            CompanyName,
            SalesPerson,
            EmailAddress,
            Phone,
            PasswordHash,
            PasswordSalt,
            rowguid,
            ModifiedDate
       )
  (SELECT  CustomerID,
            NameStyle,
            Title,
            FirstName,
            MiddleName,
```

```
            LastName,
            Suffix,
            CompanyName,
            SalesPerson,
            EmailAddress,
            Phone,
            PasswordHash,
            PasswordSalt,
            rowguid,
            ModifiedDate
FROM [SalesLT].CustomerLog )
SET IDENTITY_INSERT [SalesLT].Customer OFF

DELETE
FROM [SalesLT].CustomerLog
```