Lab 2 - Funções, Stored Procedures, Triggers e Metadados

Nome: Nuno Reis Número: 202000753

Etapa 1

--a)

create or alter function dbo.fnTotalVendasProduto (@product\_id int)

returns int

AS

begin

declare @total int;

select @total=sum(OrderQty\*UnitPrice)

from SalesLT.SalesOrderDetail

where ProductID = @product\_id;

return @total;

end;

select dbo.fnTotalVendasProduto(976) as Total;

--b)

select Name, dbo.fnTotalVendasProduto(ProductID) as Total

from SalesLT.Product

where dbo.fnTotalVendasProduto(ProductID) is not null;

--c)

create or alter function dbo.fnTotalVendas ()

returns int

as

begin

declare @total int =0;

select @total += dbo.fnTotalVendasProduto(ProductID)

from SalesLT.Product

where dbo.fnTotalVendasProduto(ProductID) is not null

return @total;

end;

select dbo.fnTotalVendas() as Total;

--d)

create or alter procedure dbo.spClientesCidade @city nvarchar(30)

as

select \*

from SalesLT.Address

where SalesLT.Address.City=@City;

go

exec dbo.spClientesCidade @City = "Las Vegas";

--e)

create or alter procedure dbo.spListaCompra @SalesOID int

as

declare @costumerEmail nvarchar(50);

declare @orderID int;

declare @date datetime;

declare @total int;

--Costumer

set @costumerEmail = (select c.EmailAddress

from [SalesLT].[SalesOrderHeader] as h

left join [SalesLT].[Customer] as c

on h.CustomerID=c.CustomerID

where h.SalesOrderID=@SalesOID);

--Order

set @orderID = (select h.SalesOrderID

from [SalesLT].[SalesOrderHeader] as h

where h.SalesOrderID=@SalesOID);

--Date

set @date = (select h.OrderDate

from [SalesLT].[SalesOrderHeader] as h

where h.SalesOrderID=@SalesOID);

--Total

set @total = (select sum(OrderQty\*UnitPrice)

from SalesLT.SalesOrderDetail as h

where h.SalesOrderID=@SalesOID);

--Print do cabeçalho

print '--------------------------------------------'

print 'Costumer: ' + @costumerEmail;

print 'Order: SO' + cast(@orderID as varchar(10));

print 'Date: ' + CONVERT( VARCHAR(24), @date, 121);

print 'Total: ' + cast(@total as varchar(10));

--Print da lista de produtos

print '--------------------------------------------'

declare @salesOrderDetailID int;

declare SalesOrderDetail\_Cursor cursor

for select d.SalesOrderDetailID

from [SalesLT].[SalesOrderDetail] as d

where d.SalesOrderID=@SalesOID;

open SalesOrderDetail\_Cursor

fetch next from SalesOrderDetail\_Cursor into @salesOrderDetailID;

while (@@FETCH\_STATUS = 0)

begin

declare @productName nvarchar(50);

declare @saleOrderQty int;

declare @saleUnitPrice float;

declare @saleUnitPriceDiscount float;

declare @saleLineTotal float;

--Product name

set @productName = (select p.Name

from [SalesLT].[SalesOrderDetail] as d

left join [SalesLT].Product as p

on d.ProductID=p.ProductID

where d.SalesOrderDetailID=@salesOrderDetailID);

--Sale Order Qty

set @saleOrderQty = (select d.OrderQty

from [SalesLT].[SalesOrderDetail] as d

left join [SalesLT].Product as p

on d.ProductID=p.ProductID

where d.SalesOrderDetailID=@salesOrderDetailID);

--Sale Unit Price

set @saleUnitPrice = (select d.UnitPrice

from [SalesLT].[SalesOrderDetail] as d

left join [SalesLT].Product as p

on d.ProductID=p.ProductID

where d.SalesOrderDetailID=@salesOrderDetailID);

--Sale Unit Price Discount

set @saleUnitPriceDiscount = (select d.UnitPriceDiscount

from [SalesLT].[SalesOrderDetail] as d

left join [SalesLT].Product as p

on d.ProductID=p.ProductID

where d.SalesOrderDetailID=@salesOrderDetailID);

--Sale Line Total

set @saleLineTotal = (select d.LineTotal

from [SalesLT].[SalesOrderDetail] as d

left join [SalesLT].Product as p

on d.ProductID=p.ProductID

where d.SalesOrderDetailID=@salesOrderDetailID);

print 'Product: ' + @productName + ' / OrderQty: ' + cast(@saleOrderQty as varchar(10)) + ' / UnitPrice: ' + cast(@saleUnitPrice as varchar(10)) + ' / UnitPriceDiscount: ' + cast(@saleUnitPriceDiscount as varchar(10)) + ' / Line Total: ' + cast(@saleLineTotal as varchar(10));

fetch next from SalesOrderDetail\_Cursor into @salesOrderDetailID;

end

close SalesOrderDetail\_Cursor;

deallocate SalesOrderDetail\_Cursor;

go

exec dbo.spListaCompra @SalesOID = 71863;

Etapa 2

--a)

create or alter function dbo.fnCodificaPassword (@password nvarchar(10))

returns nvarchar(255)

AS

begin

declare @newPassword nvarchar(255);

set @newPassword = hashbytes('SHA1',@password);

return @newPassword;

end;

select dbo.fnCodificaPassword('123abc.') as Password;

--b

CREATE TABLE CustomerPW (

ID int,

Password nvarchar(255)

);

--c

create or alter procedure dbo.spNovoCliente @ID int, @NameStyle bit, @FirstName nvarchar(50), @LastName nvarchar(50), @email nvarchar(50), @Password nvarchar(10)

as

SET IDENTITY\_INSERT SalesLT.Customer ON

insert into SalesLT.Customer (CustomerID, NameStyle, FirstName, LastName, EmailAddress, PasswordHash, PasswordSalt) values (@ID, @NameStyle, @FirstName, @LastName, @email, ' ', ' ');

SET IDENTITY\_INSERT SalesLT.Customer off

insert into dbo.CustomerPW (ID, Password) values (@ID, (select dbo.fnCodificaPassword(@Password)));

go

exec dbo.spNovoCliente @ID=8, @NameStyle=0, @FirstName='Nuno', @LastName='Reis', @email='nunoreis294@gmail.com', @Password='123abc.';

--d

create or alter function dbo.fnAutenticar (@email nvarchar(50), @password nvarchar(10))

returns int

AS

begin

declare @id int;

set @id = (select c.CustomerID

from SalesLT.Customer as c

join dbo.CustomerPW as cp

on c.CustomerID=cp.ID

where c.EmailAddress=@email and cp.Password=(select dbo.fnCodificaPassword(@password)));

return @id;

end;

select dbo.fnAutenticar('nunoreis294@gmail.com', '123abc.');

Etapa 3

--a)

create schema Logs;

/\* Log\_Data timestamp,

Log\_Operacao char\*/

create table Logs.CustomerLog

as select \*

from SalesLT.Customer;

drop table Logs.CustomerLog

select \*

from Logs.CustomerLog

Crie a tabela CustomerLog no schema “Logs” (deve ser criado previamente) 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

• 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.

--b)

create trigger trig\_name

on all server

{ for| after }

logon

as { sql\_statement [ ; ] }

go

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;

• Verifica a restrição de não poderem existir utilizadores com o mesmo login

(EmailAddress).

Etapa 4

--a)

Crie um conjunto de queries que para uma determinada tabela (e.g., Customer) permita:

• Visualizar para todas as colunas, o nome e se essa coluna tem a restrição NOT NULL;

• Visualizar o atributo IDENTITY;

• Visualizar a(s) coluna(s) que constituem a chave primária;

• Visualizar para as chaves estrangeiras, qual o nome da coluna e a tabela/coluna que é

referenciada;

--b)

Crie o stored procedure sp\_disable\_FK que recebe como argumento o nome de uma tabela

(@p\_table\_name), e gera como saída um script (i.e., lista de comandos SQL) que permite

fazer disable a todas as chaves estrangeiras que fazem referência à tabela.