Name: SDawkins

Date: 08/16/2020

Assignment: Module06Lab

GoitHub: <https://github.com/rootrUW/DBFoundations>

Note: By now, you should be able to complete these questions, without pictures of the results, but I have copied and pasted a few rows of results in the assignment06.sql files to guide you.

-- Here is an example of some rows selected from the view:

-- Manager,Employee

-- Andrew Fuller,Andrew Fuller

-- Andrew Fuller,Janet Leverling

-- Andrew Fuller,Laura Callahan

Select

[Manager] = M.EmployeeFirstName + ' ' + M.EmployeeLastName,

[Employees] = E.EmployeeFirstName + ' ' + E.EmployeeLastName

From Employees As E

Inner Join Employees As M

On E.ManagerID = M.EmployeeID

Where E.EmployeeID in (Select E.EmployeeID from Employees Where E.EmployeeFirstName + ' ' + E.EmployeeLastName In ('Laura Callahan', 'Janet Leverling', 'Andrew Fuller'))

Order By 2, 1

Go

1. Explain when you would use a SQL View.

A SQL View is a virtual table, which is based on SQL SELECT query. A view references one or more existing database tables or other views. It is the snap shot of the database whereas a stored procedure is a group of Transact-SQL statements compiled into a single execution plan.

View is simple showcasing data stored in the database tables whereas a stored procedure is a group of statements that can be executed.

A view is faster as it displays data from the tables referenced whereas a store procedure executes sql statements.

1. Explain are the differences and similarities between a View, Function, and Stored Procedure.

**VIEW**

A view is a “virtual” table consisting of a **SELECT** statement, by means of “virtual”

I mean no physical data has been stored by the view -- only the definition of the view is stored inside the database; unless you materialize the view by putting an index on it.

1)     By definition you can not pass parameters to the view

2)     NO DML operations (e.g. INSERT, UPDATE, and DELETE) are allowed inside the view; ONLY SELECT statements.

Most of the time, view encapsulates complex joins so it can be reusable in the queries or stored procedures. It can also provide level of isolation and security by hiding sensitive columns from the underlying tables.

**Stored Procedure**

A stored procedure is a group of Transact-SQL statements compiled into a single execution plan or in other words saved collection of Transact-SQL statements.

Here is a good summary from SQL MVP Hugo Kornelis (was posted in an internet newsgroup few years ago)

\*\*\*\*\*\*\*\*\*

A stored procedure:

\* accepts parameters

\* can NOT be used as building block in a larger query

\* can contain several statements, loops, IF ELSE, etc.

\* can perform modifications to one or several tables

\* can NOT be used as the target of an INSERT, UPDATE or DELETE

statement.

A view:

\* does NOT accept parameters

\* can be used as building block in a larger query

\* can contain only one single SELECT query

\* can NOT perform modifications to any table

\* but can (sometimes) be used as the target of an INSERT, UPDATE or

DELETE statement.

Asking which is faster or better is like comparing the speed of a car with that of

a boat - the speed difference is irrelevant, since you'll always prefer

the boat if you travel over water, and always the car for travels over

land.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Now let’s explain **User Defined Functions**

Functions are subroutines made up of one or more Transact-SQL statements that can be used to encapsulate code for reuse

There are three types (scalar, table valued and inline mutlistatement)  UDF and each of them server different purpose you can read more about  functions or [UDF](http://msdn.microsoft.com/en-us/library/aa175085(SQL.80).aspx) in BOL

UDF has a big limitation; **by definition it cannot change the state of the database. What I mean by this you cannot perform data manipulation operation inside UDF (INSERT, UPDATE , DELETE) etc.**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Questions and Answers \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

'NOTES------------------------------------------------------------------------------------

1) You can use any name you like for you views, but be descriptive and consistent

2) You can use your working code from assignment 5 for much of this assignment

3) You must use the BASIC views for each table after they are created in Question 1

------------------------------------------------------------------------------------------

-- Question 1 (5 pts): How can you create BACIC views to show data from each table in the database.

-- NOTES: 1) Do not use a \*, list out each column!

-- 2) Create one view per table!

-- 3) Use SchemaBinding to protect the views from being orphaned!

--------------------------------- Category Table ---------------------------------

Select CategoryID, CategoryName from Categories

go

Create -- Drop

View vvCategories With SchemaBinding

AS

Select CategoryID, CategoryName from dbo.Categories

go

Select CategoryID, CategoryName from vvCategories;

Go

--Select \* From vvCategories

--Go

--------------------------------- Products Table ---------------------------------

Select ProductID, ProductName, CategoryID, UnitPrice from Products;

Go

Create

View vvProducts With SchemaBinding

AS

Select ProductID, ProductName, CategoryID, UnitPrice from dbo.Products;

Go

Select ProductID, ProductName, CategoryID, UnitPrice

From vvProducts

Go

--------------------------------- Inventory Table ---------------------------------

Select InventoryID, InventoryDate, EmployeeID, ProductID, [Count]

From Inventories

Go

Create

View vvInventories With SchemaBinding

AS

Select InventoryID, InventoryDate, EmployeeID, ProductID, [Count]

From dbo.Inventories

go

Select InventoryID, InventoryDate, EmployeeID, ProductID, [Count]

From vvInventories

--------------------------------- Employees Table ---------------------------------

Select EmployeeID, EmployeeFirstName, EmployeeLastName, ManagerID

From Employees

Go

Create

View vvEmployees

As Select EmployeeID, EmployeeFirstName, EmployeeLastName, ManagerID

From dbo.Employees

go

Select EmployeeID, EmployeeFirstName, EmployeeLastName, ManagerID

From vvEmployees

Go

-- Question 2 (5 pts): How can you set permissions, so that the public group CANNOT select data

-- from each table, but can select data from each view?

Deny Select On dbo.Catergories to Public

Grant Select On vvCategories to Public

Deny Select On dbo.Products to Public

Grant Select On vvProducts to Public

Deny Select On dbo.Inventories to Public

Grant Select On vvInventories to Public

Deny Select On dbo.Employees to Public

Grant Select On vvEmployees to Public

Select \* From vvCategories

Select \* From vvProducts

Select \* From vvInventories

Select \* From vvEmployees

-- Question 3 (10 pts): How can you create a view to show a list of Category and Product names,

-- and the price of each product?

-- Order the result by the Category and Product!

-- Here is an example of some rows selected from the view:

-- CategoryName,ProductName,UnitPrice

-- Beverages,Chai,18.00

-- Beverages,Chang,19.00

-- Beverages,Chartreuse verte,18.00

--Select

-- C.CategoryName,

-- P.ProductName,

-- P.UnitPrice

-- From vvCategories As C

-- Inner Join vvProducts As P

-- On C.CategoryID = P.ProductID

-- Order by 1, 2

--Go

go

Create --Drop

View vvProductsByCategories

As

Select Top 100 Percent

C.CategoryName,

P.ProductName,

P.UnitPrice

From vvCategories As C

Inner Join vvProducts As P

On C.CategoryID = P.CategoryID

Order by 1, 2

Go

Select \* From vvProductsByCategories;

Go

-- Question 4 (10 pts): How can you create a view to show a list of Product names

-- and Inventory Counts on each Inventory Date?

-- Order the results by the Product, Date, and Count!

-- Here is an example of some rows selected from the view:

--ProductName,InventoryDate,Count

--Alice Mutton,2017-01-01,15

--Alice Mutton,2017-02-01,78

--Alice Mutton,2017-03-01,83

Create

View vvInventoriesByProductsByDates

As

Select Top 1000000000

P.ProductName,

I.InventoryDate,

I.[Count]

From vvProducts as P

Inner Join vvInventories As I

On P.ProductID = I.ProductID

Order By

InventoryDate,

ProductName,

[Count]

go

Select \* From vvInventoriesByProductsByDates

Go

-- Question 5 (10 pts): How can you create a view to show a list of Inventory Dates

-- and the Employee that took the count?

-- Order the results by the Date and return only one row per date!

-- Here is an example of some rows selected from the view:

-- InventoryDate,EmployeeName

-- 2017-01-01,Steven Buchanan

-- 2017-02-01,Robert King

-- 2017-03-01,Anne Dodsworth

Create

View vvInventoriesByEmployeesByDates

As

Select Distinct Top 100000000

I.InventoryDate,

[EmployeeName] = E.EmployeeFirstName + ' ' + E.EmployeeLastName

From vvInventories As I

Inner Join vvEmployees As E

On I.EmployeeID = E.EmployeeID

Order By InventoryDate, EmployeeName

Go

Select \* From vvInventoriesByEmployeesByDates

Go

-- Question 6 (10 pts): How can you create a view show a list of Categories, Products,

-- and the Inventory Date and Count of each product?

-- Order the results by the Category, Product, Date, and Count!

-- Here is an example of some rows selected from the view:

-- CategoryName,ProductName,InventoryDate,Count

-- Beverages,Chai,2017-01-01,72

-- Beverages,Chai,2017-02-01,52

-- Beverages,Chai,2017-03-01,54

Create --Drop

View vvInventoriesByProductsByCategories

As

Select Top 1000000000

C.CategoryName,

P.ProductName,

I.InventoryDate,

I.[Count]

From vvInventories As I

Inner Join vvProducts As P

On I.ProductID = P.ProductID

Inner Join vvCategories As C

On C.CategoryID = P.CategoryID

Order By 1, 2, 3, 4

Go

Select \* From vvInventoriesByProductsByCategories

Go

-- Question 7 (10 pts): How can you create a view to show a list of Categories, Products,

-- the Inventory Date and Count of each product, and the EMPLOYEE who took the count?

-- Order the results by the Inventory Date, Category, Product and Employee!

-- Here is an example of some rows selected from the view:

-- CategoryName,ProductName,InventoryDate,Count,EmployeeName

-- Beverages,Chai,2017-01-01,72,Steven Buchanan

-- Beverages,Chang,2017-01-01,46,Steven Buchanan

-- Beverages,Chartreuse verte,2017-01-01,61,Steven Buchanan

Create

View vvInventoriesByProductsByEmployees

AS

Select Top 1000000000

C.CategoryName,

Order By 1,2,3,4;

go

-- Now we use the view an apply

Go

P.ProductName,

I.InventoryDate,

I.[Count],

[EmployeeName] = E.EmployeeFirstName + ' ' + E.EmployeeLastName

From Employees As E

Inner Join vvInventories As I

On I.EmployeeID = E.EmployeeID

Inner Join vvProducts As P

On I.ProductID = P.ProductID

Inner Join vvCategories As C

On P.CategoryID = C.CategoryID

Order By 3, 1, 2, 4

Go

Select \* From vvInventoriesByProductsByEmployees

Go

-- Question 8 (10 pts): How can you create a view to show a list of Categories, Products,

-- the Inventory Date and Count of each product, and the Employee who took the count

-- for the Products 'Chai' and 'Chang'?

-- Here is an example of some rows selected from the view:

-- CategoryName,ProductName,InventoryDate,Count,EmployeeName

-- Beverages,Chai,2017-01-01,72,Steven Buchanan

-- Beverages,Chang,2017-01-01,46,Steven Buchanan

-- Beverages,Chai,2017-02-01,52,Robert King

Create

View vInventoriesForChaiAndChangByEmployees

As

Select Top 10000000000

C.CategoryName,

P.ProductName,

I.InventoryDate,

I.[Count],

[EmployeeName] = E.EmployeeFirstName + ' ' + E.EmployeeLastName

From Inventories As I

Inner Join vvEmployees As E

On I.EmployeeID = E.EmployeeID

Inner Join vvProducts As P

On P.ProductID = I.ProductID

Inner Join vvCategories As C

On C.CategoryID = P.CategoryID

Where I.ProductID in (Select ProductID from vvProducts Where ProductName In ('Chai', 'Chang'))

Order By 3, 1, 2, 4

Go

Select \* from vvInventoriesForChaiAndChangByEmployees

Go

-- Question 9 (10 pts): How can you create a view to show a list of Employees and the Manager who manages them?

-- Order the results by the Manager's name!

-- Here is an example of some rows selected from the view:

-- Manager,Employee

-- Andrew Fuller,Andrew Fuller

-- Andrew Fuller,Janet Leverling

-- Andrew Fuller,Laura Callahan

Select

[Manager] = M.EmployeeFirstName + ' ' + M.EmployeeLastName,

[Employees] = E.EmployeeFirstName + ' ' + E.EmployeeLastName

From vvEmployees As E

Inner Join vvEmployees As M

On M.EmployeeID = E.ManagerID

Order By 1, 2

Go

Create -- Drop

View vvEmployeesByManager

As

Select Top 1000000

[Manager] = M.EmployeeFirstName + ' ' + M.EmployeeLastName,

[Employees] = E.EmployeeFirstName + ' ' + E.EmployeeLastName

From vvEmployees As E

Inner Join vvEmployees As M

On M.EmployeeID = E.ManagerID

Order By 1, 2

Go

Select \* From vvEmployeesByManager

Go

-- Question 10 (10 pts): How can you create one view to show all the data from all four

-- BASIC Views?

Create View

vvInventoriesByProductsByCategoriesByEmployees

As

Select Top 10000000

C.CategoryID,

C.CategoryName,

P.ProductID,

P.ProductName,

P.UnitPrice,

I.InventoryID,

I.InventoryDate,

I.[Count],

E.EmployeeID,

[Manager] = M.EmployeeFirstName + ' ' + M.EmployeeLastName,

[Employees] = E.EmployeeFirstName + ' ' + E.EmployeeLastName

From vvInventories As I

Inner Join vvProducts As P

On I.ProductID = P.ProductID

Inner Join vvCategories As C

On C.CategoryID = P.CategoryID

Inner Join vvEmployees As E

On I.EmployeeID = E.EmployeeID

Inner Join vvEmployees As M

On E.ManagerID = M.EmployeeID

Order By 1, 3, 6, 9

Select \* From [dbo].[vvInventoriesByProductsByCategoriesByEmployees]

-------------------------------------------------------------------------------------------------------

Select \* From [dbo].[vvCategories]

Select \* From [dbo].[vvProducts]

Select \* From [dbo].[vvInventories]

Select \* From [dbo].[vvEmployees]

Select \* From [dbo].[vvProductsByCategories]

Select \* From [dbo].[vvInventoriesByProductsByDates]

Select \* From [dbo].[vvInventoriesByEmployeesByDates]

Select \* From [dbo].[vvInventoriesByProductsByCategories]

Select \* From [dbo].[vvInventoriesByProductsByEmployees]

Select \* From [dbo].[vvInventoriesForChaiAndChangByEmployees]

Select \* From [dbo].[vvEmployeesByManager]

Select \* From [dbo].[vvInventoriesByProductsByCategoriesByEmployees]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

-- Here is an example of some rows selected from the view:

-- CategoryID,CategoryName,ProductID,ProductName,UnitPrice,InventoryID,InventoryDate,Count,EmployeeID,Employee,Manager

-- 1,Beverages,1,Chai,18.00,1,2017-01-01,72,5,Steven Buchanan,Andrew Fuller

-- 1,Beverages,1,Chai,18.00,78,2017-02-01,52,7,Robert King,Steven Buchanan

-- 1,Beverages,1,Chai,18.00,155,2017-03-01,54,9,Anne Dodsworth,Steven Buchanan

Select

[Manager ID] = IsNull(Mgr.EmployeeId, 0)

,[Manager] = IIF(IsNull(Mgr.EmployeeId, 0) = 0, 'General Manager', Emp.EmployeeFirstName + ' ' + Mgr.employeeLastName)

,[Employee ID] = Emp.EmployeeID

,[Employee Name] = Emp.EmployeeFirstName + ' ' + Emp.EmployeeLastName

From vvEmployees as Emp

Left Join vvEmployees Mgr

On Emp.ManagerID = Mgr.EmployeeID

Order By 1,3;

go