BASES DE DATOS – INGENIERIA INFORMATICA - LABORATORIO
UNIVERSIDAD NACIONAL DE ITAPUA – 2012
Ejercicios SQL con Ejemplos y un poco de Teoría – Trabajo Práctico #3

Profesor: Asistente Univ. Hisakazu Ishibashi

INTRODUCCION

En este trabajo intentará abarcar la mayor cantidad de conceptos SQL que están disponibles en distintos DBMS, a través de la base de datos de ejemplo de **SQL Server 2008 R2**, **AdventureWorks2008R2**, la cual se encuentra disponible tanto en la instalación en la imagen de virtual-win7 para Virtual Box proveída en el transcurso de las clases, como así también como instalador por separado, ambos en la PC-01 del Laboratorio de FIUNI. Sin embargo, tanto SQL Server y su Transact-SQL tienen muchas más prestaciones de las que se puede estudiar a lo largo de este semestre, atendiendo el programa de estudio y las horas de clase disponibles. Por eso recomendable conocer las prestaciones que puede ofrecernos cada producto DBMS específico en base a los requerimientos del problema específico o los requerimientos de tecnologías de la información que haya que satisfacer.

Para tener una mejor comprensión y entendimiento de los ejercicios propuestos es recomendable tener un diagrama de las entidades, relaciones y vistas de dicha base de datos, el cual se puede obtener entre otras maneras, a través de ingeniería inversa utilizando el programa ER Studio. Si utilizan la versión de ER Studio que está instalada en virtual-win7 (ER Studio 7.5.1) verán que los *schemas* de la BD en cuestión son interpretados como usuarios, y que al generar el modelo se crean todas las tablas y vistas en un *Main Model*. En este sentido es conveniente (para clarificar el panorama) recrear dichos *schemas* a través de la utilización de *submodels*, y luego elegir las entidades y vistas que corresponden a dichos *schemas* haciendo uso de la opción *Apply To Submodels.* Para más información al respecto ver la ayuda del ER Studio buscando a *submodels.* También pueden usar la funcionalidad para crear diagramas de bases de datos proveída en el SQL Server Management Studio el cual también se encuentra en el laboratorio y en el virtual-win7.

El trabajo está dividido en distintas etapas las cuales tienen distintas fechas de entrega. Favor leer atentamente las instrucciones y ante cualquier duda que tengan por favor no duden en consultar a cualquiera de los profesores de la materia.

Antes de empezar con los ejercicios, no se olviden de hacer un backup de la base de datos para poder volver a un estado consistente en caso de que hayan cometido algún error que no pueden reparar.

Todos los ejercicios se pueden realizar con la ayuda de la documentación online del MSDN (Microsoft Developer Network) y/o con la ayuda de Google. Algunos de ellos requerirán un poco de investigación para encontrar la respuesta, pero cabe destacar que en cada sección se darán ejemplos que puedan dar alguna pista sobre como resolver los ejercicios que serán propuestos posteriormente, y se hablará brevemente al respecto de temas relacionados a los mismos. En dichos ejemplos mencionaremos las entidades en inglés, así como están en la BD

Por último, la sección Lecturas Recomendadas contiene enlaces a temas interesantes relacionados a Bases de Datos en general, y a SQL Server y sus bases de datos de ejemplo entre otros.

Por favor no se copien.

Parte 1 - CONSULTAS (SELECT…) Y VISTAS SQL

Como ya lo habíamos visto en clase, en SQL usamos la cláusula **select** para recuperar información de la BD, y las vistas (views) nos permiten almacenar consultas (las cuales se almacenan como tablas dinámicas que se crean cada vez que se las consulta) para entre otras cosas:

- Encapsular la complejidad de la BD o de las tablas que hacen a alguna consulta.
- Proveer cierto mecanismo de seguridad otorgando distintos privilegios a los roles que acceden solo a las vistas y a los que pueden acceder a estas y a las tablas subyacentes.
- Proveer compatibilidad retroactiva para tablas cuyo esquema haya cambiado.

Adicionalmente, hay que tener en cuenta que en SQL Server 2008 R2 y en la mayoría de RDBMSs podemos modificar datos a través de una vista, ya sea a través de INSERT, UPDATE, DELETE, con ciertas restricciones acerca de las tablas subyacentes. En SQL Server 2008 R2 también es posible que una vista sea modificable a través de los INSTEAD OF Triggers.

En SQL Server se pueden crear distintos tipos de vistas como las particionadas y distribuidas, entre otras, las cuales están fuera del alcance de este curso pero pueden verse en la documentación (views).

En cuanto a la cláusula **select** hay que tener en cuenta que la misma es una cláusula de la sentencia **select**, a través de la cual se puede entre otras cosas, pasar sugerencias al optimizador de consulta (Query Optimizer).

A continuación se mostrarán algunos ejemplos muchos de los cuales provienen de la misma BD de ejemplo AventureWorks2008R2. Otros ejemplos pueden verse en SELECT Examples (Transact-SQL)

Ejemplo 1 – Vista HumanResources.vEmployee

A continuación veremos el DDL de la vista HumanResources.vEmployee la cual existe en la instalación de AdventureWorks2008R2. Existen varias formas de ver el DDL o metadata de objetos de la BD. La que yo utilicé es a través del explorador de objetos del SSMS, haciend click derecho sobre el objecto y luego en este caso elegir la opción **Script View as > CREATE To > New Query Editor Window.** Sin embargo, también existen formas de obtener el metadata o DDL de los objetos a través del Transact-SQL.

Al examinar la consulta subyacente podemos apreciar que se trata de una consulta que nos muestra todas las Person.Person que son HumanResources.Employee, junto con información adicional tales como su Address, StateProvince, etc., teniendo en cuenta que la persona puede ser que no tenga PersonPhone, PersonPhoneNumberType ni EmailAddress, por eso los LEFT OUTER JOIN con dichas tablas.

```
USE [AdventureWorks2008R2]
GO

/****** Object:  View [HumanResources].[vEmployee]    Script Date: 05/01/2012 02:56:46
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO



CREATE VIEW [HumanResources].[vEmployee]
```

```
AS
SELECT
e.[BusinessEntityID]
,p.[Title]
,p.[FirstName]
,p.[MiddleName]
,p.[LastName]
,p.[Suffix]
,e.[JobTitle]
,pp.[PhoneNumber]
,pnt.[Name] AS [PhoneNumberType]
,ea.[EmailAddress]
,p.[EmailPromotion]
,a.[AddressLine1]
,a.[AddressLine2]
,a.[City]
,sp.[Name] AS [StateProvinceName]
,a.[PostalCode]
,cr.[Name] AS [CountryRegionName]
,p.[AdditionalContactInfo]
FROM [HumanResources].[Employee] e
INNER JOIN [Person].[Person] p
ON p.[BusinessEntityID] = e.[BusinessEntityID]
INNER JOIN [Person].[BusinessEntityAddress] bea
ON bea.[BusinessEntityID] = e.[BusinessEntityID]
INNER JOIN [Person].[Address] a
ON a.[AddressID] = bea.[AddressID]
INNER JOIN [Person].[StateProvince] sp
ON sp.[StateProvinceID] = a.[StateProvinceID]
INNER JOIN [Person].[CountryRegion] cr
ON cr.[CountryRegionCode] = sp.[CountryRegionCode]
LEFT OUTER JOIN [Person].[PersonPhone] pp
ON pp.BusinessEntityID = p.[BusinessEntityID]
LEFT OUTER JOIN [Person].[PhoneNumberType] pnt
ON pp.[PhoneNumberTypeID] = pnt.[PhoneNumberTypeID]
LEFT OUTER JOIN [Person].[EmailAddress] ea
ON p.[BusinessEntityID] = ea.[BusinessEntityID];
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Employee names and
addresses.' , @level0type=N'SCHEMA',@level0name=N'HumanResources',
@level1type=N'VIEW',@level1name=N'vEmployee'
GO
```

También podemos observar que después de la creación de la vista, se ejecuta el procedimiento almacenado sys.sp_add_extendedproperty. Este SP (*Stored Procedure, por sus siglas en inglés*) sirve como un mecanismo para agregar Extended Properties, los cuales entre otras cosas nos permiten agregar información descriptiva para entre otras cosas documentar la finalidad o el propósito de los objetos de nuestra base de datos, lo cual a su vez puede ser el inicio de un "diccionario de datos". En el ejemplo de arriba podemos ver que se agrega la 'MS_Description' que indica que la vista vEmployee en el schema HumanResources muestra los empleados y sus direcciones (Employee names and addresses). Obviamente no sería de mucha ayuda si no pudiéramos recuperar las extended properties de nuestros objetos, pero por suerte para eso existe la vista sys.extended_properties. Por ejemplo si quisiéramos recuperar la extended_property que acabamos de agregar podemos usar el siguiente ejemplo

Ejemplo 2 – Obtener extended property MS_Description del objeto HumanResources.vEmployee

```
select * from
     sys.extended_properties ep
where
     major_id = OBJECT_ID('HumanResources.vEmployee')
     AND ep.name = N'MS_Description';
```

Ejemplo 3 - Obtener todas las extended properties de la tabla Production.ProductInventory y sus respectivas columnas, haciendo uso de la expresión CASE para escribir 'THIS IS THE TABLE' en caso de que la tupla listada represente a los extended properties de la tabla, y CAST para convertir a nvarchar el valor de la columna value.

```
SELECT
     class,
     class_desc,
     minor_id,
     OBJECT_NAME(ep.major_id) AS [TableName],
     CASE WHEN ep.minor_id>0 THEN COL_NAME(ep.major_id, ep.minor_id) ELSE 'THIS IS
THE TABLE' END AS [ColumnName],
     CAST(Value AS nvarchar(500)) AS [Property]
FROM
     sys.extended_properties AS ep
WHERE
     ep.major_id = OBJECT_ID('Production.ProductInventory')
AND class_desc = N'OBJECT_OR_COLUMN'
```

Ejemplo 4 - Obtener un listado de las columnas de la vista HumanResources.vEmployee de la BD AdventureWorks2008R2 utilizando las vistas de InformationSchema, con una subconsulta para "verificar que la tabla sea una vista".

```
SELECT
     TABLE_NAME,
     COLUMN_NAME,
     COLUMNPROPERTY(OBJECT_ID(TABLE_SCHEMA + '.' + TABLE_NAME), COLUMN_NAME,
'ColumnID') AS COLUMN_ID
FROM
     AdventureWorks2008R2.INFORMATION_SCHEMA.COLUMNS
WHERE
     TABLE_SCHEMA = 'HumanResources' AND TABLE_NAME = 'vEmployee'
     AND TABLE_NAME IN (
          SELECT TABLE_NAME FROM AdventureWorks2008R2.INFORMATION_SCHEMA.VIEWS
          WHERE TABLE_SCHEMA = 'HumanResources')
```

Ejemplo 5 – Mismo resultado que la consulta anterior pero ésta vez utilizando JOIN

```
SELECT
     c.TABLE_NAME,
     c.COLUMN_NAME,
     COLUMNPROPERTY(OBJECT_ID(c.TABLE_SCHEMA + '.' + c.TABLE_NAME), c.COLUMN_NAME,
'ColumnID') AS COLUMN_ID
FROM
     AdventureWorks2008R2.INFORMATION_SCHEMA.COLUMNS c
     INNER JOIN AdventureWorks2008R2.INFORMATION_SCHEMA.VIEWS v
          ON c.TABLE_NAME = v.TABLE_NAME and c.TABLE_SCHEMA = v.TABLE_SCHEMA
WHERE
 c.TABLE_SCHEMA = 'HumanResources' AND c.TABLE_NAME = 'vEmployee'
```

Ejemplo 6 – Vista Production.vProductAndDescription

Lo interesante de este ejemplo es que usa la opción WITH SCHEMABINDING, que impide que la tabla o tablas subyacentes sean modificadas de manera tal a que afecte la definición de la vista, además la propiedad extendida MS_Description nos dice que hay un *clustered index* en la vista.

```
CREATE VIEW [Production].[vProductAndDescription]
WITH SCHEMABINDING
AS
SELECT
p.[ProductID]
,p.[Name]
,pm.[Name] AS [ProductModel]
,pmx.[CultureID]
,pd.[Description]
FROM [Production].[Product] p
INNER JOIN [Production].[ProductModel] pm
ON p.[ProductModelID] = pm.[ProductModelID]
INNER JOIN [Production].[ProductModelProductDescriptionCulture] pmx
ON pm.[ProductModelID] = pmx.[ProductModelID]
INNER JOIN [Production].[ProductDescription] pd
```

```
ON pmx.[ProductDescriptionID] = pd.[ProductDescriptionID];

GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Clustered index on
the view vProductAndDescription.' , @level0type=N'SCHEMA',@level0name=N'Production',
@level1type=N'VIEW',@level1name=N'vProductAndDescription',
@level2type=N'INDEX',@level2name=N'IX_vProductAndDescription'
GO

EXEC sys.sp_addextendedproperty @name=N'MS_Description', @value=N'Product names and
descriptions. Product descriptions are provided in multiple languages.' ,
@level0type=N'SCHEMA',@level0name=N'Production',
@level1type=N'VIEW',@level1name=N'vProductAndDescription'
GO
```

Ejemplo 7 – Obtener una lista de todos los Person que no son Employee mostrando solo los campos de Person en el resultado.

```
select
     pp.*
from
     Person.Person pp left join HumanResources.Employee he
          on pp.BusinessEntityID = he.BusinessEntityID
where
     he.BusinessEntityID is null
```

Ejemplo 8 - Obtener la cantidad de Customer que son solo Person y la cantidad de los que son solo Store. En este ejemplo vemos como podemos usar subconsultas en el select_list
Obs: Hay 635 personas que son Customer de tipo Store y Person al mismo tiempo

```
select
     (select COUNT(*) from Sales.Customer where PersonID is null) as [CustomerStore],
     (select COUNT(*) from Sales.Customer where StoreID is null) as [CustomerPerson]
```

Ejemplo 9 - Para obtener la cantidad de clientes que tienen al menos una venta podemos tener los distintos clientes que tienen ventas y luego obtener su cantidad. Usamos para eso una subconsulta en la especificación de FROM.

```
select COUNT(*)
from
     (select distinct CustomerID from Sales.SalesOrderHeader) as ClientesVentas
```

Ejemplo 10 - Obtener la suma de todos los Pagos a los Employee agrupados por departamento y convertidos a guaraníes al tipo de cambio 4270. El resultado debe mostrar la una Columna con el Nombre del departamento y otra con dicha suma.

```
select
     Name, SUM(rate*4270)as [suma]
from
     HumanResources.Employee e inner join HumanResources.EmployeePayHistory h
          on e.BusinessEntityID = h.BusinessEntityID
     inner join HumanResources.EmployeeDepartmentHistory dh
          on e.BusinessEntityID = dh.BusinessEntityID
     inner join HumanResources.Department d
          on dh.DepartmentID = d.DepartmentID
group by
     Name
order by
     suma desc
```

Ejemplo 11 - Mostrar todos los Employee que trabajan en Department q cuyo nombre contiene la palabra Production, y que tienen mas de un HumanResources.EmployeePayHistory en un listado mostrando las columnas BusinessEntityID, LastName seguido de una coma y luego FirstName en

una columna llamada NombreCompleto, su departamento actual, su ultimo rate, la diferencia entre su menor y su mas alto rate, su rate promedio (truncado a solo 2 posiciones decimales), su ultimo RateChangeDate y la cantidad en años a partir del ultimo RateChangeDate a la fecha actual. Ordenar las filas por RatePromedio en forma descendente y Año del último rate y nombre completo en forma ascendente.

Para hacer este ejemplo quise hacer uso de la vista HumanResources.vEmployeeDepartment, pero raramente dicha vista no traía ningún resultado, cuando debería estar trayendo el departamento actual de cada empleado. Al mirar el DDL de la vista y ejecutar la consulta subyacente me di cuenta de que lo que estaba haciendo que el resultado no tenga filas es la condición de búsqueda. A continuación se muestra el DDL de dicha vista.

```
CREATE VIEW [HumanResources].[vEmployeeDepartment]
AS
SELECT
e.[BusinessEntityID]
,p.[Title]
,p.[FirstName]
,p.[MiddleName]
,p.[LastName]
,p.[Suffix]
,e.[JobTitle]
,d.[Name] AS [Department]
,d.[GroupName]
,edh.[StartDate]
FROM [HumanResources].[Employee] e
INNER JOIN [Person].[Person] p
ON p.[BusinessEntityID] = e.[BusinessEntityID]
INNER JOIN [HumanResources].[EmployeeDepartmentHistory] edh
ON e.[BusinessEntityID] = edh.[BusinessEntityID]
INNER JOIN [HumanResources].[Department] d
ON edh.[DepartmentID] = d.[DepartmentID]
WHERE GETDATE() BETWEEN edh.[StartDate] AND ISNULL(edh.[EndDate], GETDATE());
```

Lo que pasa es que la función ISNULL() está truncando GETDATE(), removiendo la parte que contiene la información de hora, minuto, segundo, etc., dejando solo la fecha. Si se ejecuta la siguiente consulta

```
select EndDate, ISNULL(EndDate, GETDATE()) AS FechaISNULL, GETDATE() AS FechaActual
 from HumanResources.EmployeeDepartmentHistory
```

se podrá ver las implicancias que tiene ISNULL() sobre GETDATE() y como esto finalmente afecta la condición de búsqueda de la vista en cuestión, ya que la fecha actual con hora, minuto, segundo, etc., no está entre el StartDate y la fecha actual sin información de hora, minuto ni segundo.

Entonces para poder usar esta vista, a través de ALTER VIEW agregaremos un día a GETDATE() adentro de ISNULL() en la condición de búsqueda, así como está a continuación:

```
ALTER VIEW [HumanResources].[vEmployeeDepartment]
AS
SELECT
e.[BusinessEntityID]
,p.[Title]
,p.[FirstName]
,p.[MiddleName]
,p.[LastName]
,p.[Suffix]
,e.[JobTitle]
,d.[Name] AS [Department]
,d.[GroupName]
,edh.[StartDate]
FROM [HumanResources].[Employee] e
INNER JOIN [Person].[Person] p
ON p.[BusinessEntityID] = e.[BusinessEntityID]
INNER JOIN [HumanResources].[EmployeeDepartmentHistory] edh
ON e.[BusinessEntityID] = edh.[BusinessEntityID]
INNER JOIN [HumanResources].[Department] d
ON edh.[DepartmentID] = d.[DepartmentID]
WHERE GETDATE() BETWEEN edh.[StartDate] AND ISNULL(edh.[EndDate],
dateadd(day,1,GETDATE()));
```

```
GO
```

Luego la consulta finalmente queda como sigue:

```
Select
     ph.BusinessEntityID,
     ed.LastName+','+ed.FirstName as NombreCompleto,
     ed.Department,
     (
          select ph1.rate from HumanResources.EmployeePayHistory ph1
          where ph1.RateChangeDate >= all (
               select RateChangeDate from HumanResources.EmployeePayHistory ph2
               where ph2.BusinessEntityID = ph1.BusinessEntityID
          ) and ph1.BusinessEntityID = ph.BusinessEntityID
     ) as UltimoRate,
     round(AVG(ph.Rate),2,1) as RatePromedio,
     MAX(ph.rate)- MIN(ph.rate) DiferenciaRate,
     substring(cast(max(ph.RateChangeDate) as nvarchar),0, charindex('12',
cast(max(ph.RateChangeDate) as nvarchar), 3)) as FechaUltimoRate,
     (
          select DATEDIFF(YEAR, max(ph1.RateChangeDate), GETDATE()) from
HumanResources.EmployeePayHistory ph1
               where ph1.BusinessEntityID = ph.BusinessEntityID
     ) as AnhosUltimoRate
from
     HumanResources.EmployeePayHistory ph inner join HumanResources.vEmployeeDepartment ed
          on ph.BusinessEntityID = ed.BusinessEntityID
where
     ed.Department like '%Production%'
group by
     ph.BusinessEntityID, ed.LastName+','+ed.FirstName, ed.Department
Having
     COUNT(*)>1
order by RatePromedio desc, AnhosUltimoRate, NombreCompleto asc
```

Aquí podemos destacar entre otras cosas la utilización de los alias de las tablas tanto en la consulta *externa* como así también en las subconsultas para crear algo similar a variables de tipo puntero a las distintas tuplas que se evalúan en la consulta. La utilización de funciones de *string*, funciones de fecha, la función *cast* para convertir de un tipo de dato a otro y como podemos usar having con una función distinta a la que está en el *select_list*.

Ejemplo 13 - Obtener el SalesPerson que tuvo el mayor promedio de ventas entre los años fiscales 2006, 2007, y 2008, en su respectivo SalesTerritory, mostrando las columnas:

- SalesPersonID,
- FullName (FirstName, MiddleName y LastName del SalesPerson, concatenados con espacios entre sí),
- JobTitle,
- SalesTerritory,
- Columnas respectivas a los años 2006, 2007 y 2008 en las cuales se mostrarán la suma de los SubTotal de los SalesOrderHeader correspondiente al SalesPersonID en dichos años.
- Por ùltimo, la columna que contendrá el promedio de venta de los 3 años mencionados (la suma de los 3 años divido 3).

En este caso, quise usar la vista Sales.vSalesPersonSalesByFiscalYears, pero la misma está "desactualizada" con respecto a los datos que están en la BD, ya que la consulta subyacente solo muestra las ventas de los años 2002, 2003, y 2004, y las OrderDate de los SalesOrderHeader van desde le 2005 en adelante. De todas formas, he hecho uso de la consulta subyacente de dicha vista y la he modificado acorde a los requerimientos de este ejemplo.

```
SELECT
```

```
     pvt.[SalesPersonID]
     ,pvt.[FullName]
     ,pvt.[JobTitle]
     ,pvt.[SalesTerritory]
     ,pvt.[2006]
     ,pvt.[2007]
     ,pvt.[2008]
     ,(COALESCE(pvt.[2006],0)+COALESCE(pvt.[2007],0)+COALESCE(pvt.[2008],0))/3 as
[Promedio]
     FROM (SELECT
          soh.[SalesPersonID]
          ,p.[FirstName] + ' ' + COALESCE(p.[MiddleName], '') + ' ' + p.[LastName] AS
[FullName]
          ,e.[JobTitle]
          ,st.[Name] AS [SalesTerritory]
          ,st.TerritoryID
          ,soh.[SubTotal]
          ,YEAR(DATEADD(m, 6, soh.[OrderDate])) AS [FiscalYear]
          FROM [Sales].[SalesPerson] sp
          INNER JOIN [Sales].[SalesOrderHeader] soh
          ON sp.[BusinessEntityID] = soh.[SalesPersonID]
          INNER JOIN [Sales].[SalesTerritory] st
          ON sp.[TerritoryID] = st.[TerritoryID]
          INNER JOIN [HumanResources].[Employee] e
          ON soh.[SalesPersonID] = e.[BusinessEntityID]
          INNER JOIN [Person].[Person] p
          ON p.[BusinessEntityID] = sp.[BusinessEntityID]
     ) AS soh
     PIVOT
     (
          SUM([SubTotal])
          FOR [FiscalYear]
          IN ([2006], [2007], [2008])
     ) AS pvt
     where not exists (
          select ss.SalesPersonID, sum(SubTotal)
          from Sales.SalesOrderHeader ss join Sales.SalesPerson sp on ss.SalesPersonID =
sp.BusinessEntityID
          where  YEAR(DATEADD(m, 6, ss.OrderDate))in (2006,2007,2008) and pvt.TerritoryID
= sp.TerritoryID
          group by ss.SalesPersonID
          having pvt.[2006]+pvt.[2007]+pvt.[2008]<SUM(SubTotal)
     )
go
```

En este ejemplo hay que destacar que si no hubiéramos usado PIVOT hubiéramos tenido que escribir 3 sub consultas en la select_list para traer las sumas correspondiente a los años solicitados. También la función COALESCE, la cual es necesaria para devolver valores en reemplazo de NULL, por ejemplo en la operación en donde se concatenan los distintos campos que hacen al *FullName*.

Ejercicios – Parte 1

Instrucciones

1) Como se ha mencionado en la INTRODUCCION, se usará la BD AdventureWorks2008R2.
2) Crear un *schema* su nombre y apellido. Ej.: CREATE SCHEMA JuanPerez … Esto significa que todos los ejercicios entregados por alumno deben ser parte del *schema* creado por el mismo.
3) Todas las vistas requeridas a continuación deberán ser creadas dentro de dicho *schema.*
4) Los resultados deberán incluir exactamente las columnas que se solicitan. Por favor atender bien este punto y consultar en caso de ser necesario.
5) Para más información ver **CREATE SCHEMA (Transact-SQL).**
6) Las vistas deberán tener como profijo la letra v (minúscula), tal cual lo tienen las vistas que actualmente existen en la BD en cuestión.
7) Los nombres de las vistas deben ser representativos en cuanto al resultado que proveen. Ej.:
8) Los ejercicios deberán ser entregados en un archivo cuyo nombre tiene que tener el formato nombre_apellido_tp3p1_bd2012.sql, el 16 de mayo de 2012 antes de las 23:59 a kazutron@gmail.com escribiendo como subject (sujeto) del mensaje BD2012 – TP#3.

Ejercicios

Crear las siguientes vistas:

1) Vista que muestre los 3 productos más vendidos mostrando el ProductID, Name, ProductModel, y la cantidad de ventas de dicho producto. Las ventas se registran en las tablas Sales.SalesOrderHeader y Sales.SalesOrderDetail. Tengan cuidado con los CultureID si van a usar la vista Production.vProductandDescription
2) Crear una vista que muestre los 3 productos más vendidos como la anterior pero esta vez agrupados por año. La columna año tiene que ser la primera del resultado. Ordenar por año en forma descendente, cantidad en forma ascendente y ProductID en forma ascendente.
3) Vista que muestre los empleados que trabajaron en más de un departamento mostrando su BusinessEntityID, Nombre Completo (Title, FirstName, MiddleName y LastName separados por espacio), y la cantidad de cambios que tuvieron.
4) Vendedores que menos vendieron en el 2do trimestre del año 2008, mostrando su FirstName, LastName, el nombre de su Department actual, y la cantidad en dólares de sus respectivas ventas en dicho período de tiempo.
5) Que muestre la diferencia entre el importe de las ventas Sales.SalesOrderHeader.SubTotal y de las compras Purchasing.PurchaseOrderHeader.SubTotal agrupados por año. Las columnas de la tabla resultante deben llamarse Año y Ganancia respectivamente.
6) Cuyo resultado sea una lista de los empleados de menor edad cuyos cumpleaños se festejan entre los meses de Septiembre, Octubre y Noviembre. El reporte deberá mostrar las columnas Mes (en español), Nombre(FirstName), Apellido(LastName), Edad. Los meses deberán escribirse en español, para lo cual se puede usar la sentencia CASE. La edad debe escribirse como la edad en números seguido de la palabra años, ej.: 20 años.
7) Que muestre los productos que contengan la palabra aluminun en su Description (case-insensitve), y que contengan la palabra mountain(también case-insensitve) ya sea en su Name, ProductModel o Description, mostrando las columnas Name, ProductModel, Description, su correspondiente Production.ProductCostHistory.StandardCost actual (el que no tiene EndDate). Consejo: Ver o usar la vista Production.vProductAndDescription.
8) Que muestre los meses (en español) con sus respectivos importes de ventas totales correspondientes a cada año en el que haya ventas registradas. El reporte debe contener las columnas Año, Mes, Importe Mensual, Porcentaje Año (cuyo valor es relativo con respecto al total de las ventas del año en cuestión). El resultado sería por ejemplo (*no incluye todos los mese ni todos los años*):

Año	Mes	Importe Mensual	Porcentaje Año
2006	Enero	75500	80%
2006	Febrero	18875	20%
2007	Enero	299898	100%

9) Vista un poco similar a la anterior pero organizada de la siguiente manera (*no se muestran todos los meses, ni todos los años*)

Año	Enero	Febrero	Marzo	Abril	Mayo, etc…
2006	10%	20%	5%	3.5%	8%
2007	21%	2%	33%	4%	9%
2009	8%	2%	10%	8%	33%

Este ejercicio es similar al Ejemplo – 13.

10) Que muestre el nombre del producto más vendido, con su respectivo modelo y los primeros 15 caracteres de su descripción, el nombre y apellido del vendedor que más veces lo vendió, su rate actual (EmployePayHistory.rate) más reciente, y la cantidad de veces que vendió dicho producto. Los nombres de las columnas deben ser respectivamente ProductName, ProductModel, ProductShortDesc, SalesName, SalesLastName, CurrRate, ProdSalesCount.

Parte 2 - Más consultas, Procedimientos Almacenados, UDFs, y Triggers.

Próximamente…

Parte 3 - Casi de Todo un Poco – Administración, Backup y Restore, Replicación, Seguridad y otros temas.

Próximamente (Si hay tiempo)…

LECTURAS RECOMENDADAS

- Why do some SQL strings have an 'N' prefix?
  http://databases.aspfaq.com/general/why-do-some-sql-strings-have-an-n-prefix.html

- Fun with Extended Properties in SQL Server 2008
  http://sqlserverperformance.wordpress.com/2011/01/10/fun-with-extended-properties-in-sql-server-2008/

- **Using Extended Properties on Database Objects**
  http://msdn.microsoft.com/en-us/library/ms190243(v=sql.105).aspx

- Diccionario de Datos
  http://en.wikipedia.org/wiki/Data_dictionary

- SELECT Clause (Transact-SQL)
  http://msdn.microsoft.com/en-us/library/ms176104(v=sql.105).aspx

- **CASE (Transact-SQL)**
  http://msdn.microsoft.com/en-us/library/ms181765(v=sql.105).aspx

- **CAST and CONVERT (Transact-SQL)**
  http://msdn.microsoft.com/en-us/library/ms187928(v=sql.105).aspx

- **Copy-Only Backups (SQL Server 2008 R2)**
  http://msdn.microsoft.com/en-us/library/ms191495(v=sql.105).aspx

- Querying the SQL Server System Catalog
  http://msdn.microsoft.com/en-us/library/ms189082.aspx

- Information Schema views (Transact-SQL)
  http://msdn.microsoft.com/en-us/library/ms186778(v=sql.105).aspx

- Displaying Graphical Execution Plans (SQL Server Management Studio)
  http://msdn.microsoft.com/en-us/library/ms178071(v=sql.105).aspx

- Query Plan
  http://en.wikipedia.org/wiki/Query_plan

- **Query Hints (Transact-SQL)**
  http://msdn.microsoft.com/en-us/library/ms181714(v=sql.105).aspx

- **Introduction to Query Processing and Optimization**
  http://www.cs.iusb.edu/technical_reports/TR-20080105-1.pdf

- **Query Processing in DBMS (Slides muy bien ilustrados y explicacion bien sencilla)**
  http://www.slideshare.net/koolkampus/ch13

- Query Performance
  http://msdn.microsoft.com/en-us/library/ms190610(v=sql.105).aspx

- Optimizing Server Performance
  http://msdn.microsoft.com/en-us/library/ms188284(v=sql.105).aspx

- **DBCC FREEPROCCACHE (Transact-SQL) – Borra elementos del PLAN CACHE**
  http://msdn.microsoft.com/en-us/library/ms188284(v=sql.105).aspx

- **SQL CLR – SQL Common Language Runtime**
  http://en.wikipedia.org/wiki/SQL_CLR

- **ORDER BY Clause (Transact-SQL)**
  http://msdn.microsoft.com/en-us/library/ms188385(v=sql.105).aspx

- Ranking Functions (Transact-SQL)
  http://msdn.microsoft.com/en-us/library/ms189798(v=sql.105).aspx