MANUAL DEL SQL SERVER 2008

UNIVERSIDAD NACIONAL DE CAJAMARCA

FACULTAD DE INGENIERÍA

ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA DE SISTEMAS

BASE DE DATOS II

ALUMNO:

VÁSQUEZ SAMÁN, ALEX DAVID

DOCENTE:

CARLOS APARICIO

Contenido

INTRODUCCIÓN. 3
INSTALACIÓN DEL SQL SERVER 2008....4
INGRESAR AL SQL SERVER 2008....19
CONSULTAS EN LA BASE DATOS NORTHWIND 22
1. Crear una consulta 22
2. Ejecutar consulta....23
3. SELECT 24
4. FROM 24
5. WHERE....24
6. ORDERDATE: 26
7. DATEPART....26
8. ORDER BY 27
9. LIKE....28
10. TOP 28
Ejercicios 30
11. INSERT 33
12. UPDATE....34
13. DELETE 36
14. DISTINCT 37
15. INNER JOIN....37
16. UNION 40
17. SELECT INTO 41
18. SUBCONSULTAS....41
19. GROUP BY 41
Ejercicios 43
PROCEDIMIENTOS ALMACENADOS 46
REPORTING SERVICES 60
RESTRICCIONES EN SQL....127
FUNCIONES 140
TRIGGERS 157
Ejercicios 163

MANUAL DEL SQL SERVER 2008

INTRODUCCIÓN.

SQL son las siglas de "Structured Query Language" que quiere decir "Lenguajes de Solicitud Estructurado". SQL SERVER es un Gestor de Base Datos de Microsoft, esta versión 2008, muestra grandes avances con respecto a sus predecesora (SQL Server 2005); comienza ser un serio competidor para sistemas como ORACLE, orientados a base de datos de gran tamaño.

SQL Server 2008 incluye una gran cantidad de nuevas características que permiten una gestión más racional y eficaz del mismo, aumentan el rendimiento, la escalabilidad y la estabilidad del servidor y, permiten una configuración avanzada a nivel de servicios, seguridad del servidor, etc.

Entre las nuevas características, quizás la más llamativa sea el soporte para compatibilidad con **.NET**.

Esto permite la programación de ensamblados en C# o Visual Basic 2008 y su ejecución en SQL Server, lo que abre un gran abanico de posibilidades complementando la funcionalidad que proporciona T-SQL (TRANSACT - SQL)

Todas estas características se agrupan dentro de SQL Server 2008 clasificadas según sus distintas ediciones:

- **Express**: Esta edición es la evolución del antiguo MSDE, la versión gratuita de SQL Server 2000. Sigue siendo gratuita y, aunque limitada, incorpora un pequeño entorno gráfico de administración y permite un máximo de 50 conexiones concurrentes (suficiente para cualquier entorno pequeño).

- **Workgroup**: Está diseñada para entornos y departamentos pequeños y medianos. Posee muchas de las características de SQL Server, pero no contiene las de alto nivel.

- **Standard**: Esta versión está destinada al entorno medio. Contiene prácticamente todas las características, como los Servicios de Análisis, o los Servicios de Integración, pero elimina las opciones de alta disponibilidad, como particionado o indexación online.

- **Developer**: Esta versión contiene todas las opciones, pero al ser una versión destinada a entornos de prueba y laboratorio, contiene limitaciones en cuanto a CPUs soportadas y a licencias.

- **Enterprise**: Esta es la versión completa, la más potente, escalable y robusta y, por supuesto, la más cara. Está destinada al entorno empresarial de tamaño medio-grande, donde el rendimiento, la alta disponibilidad y la escalabilidad son cruciales.

INSTALACIÓN DEL SQL SERVER 2008

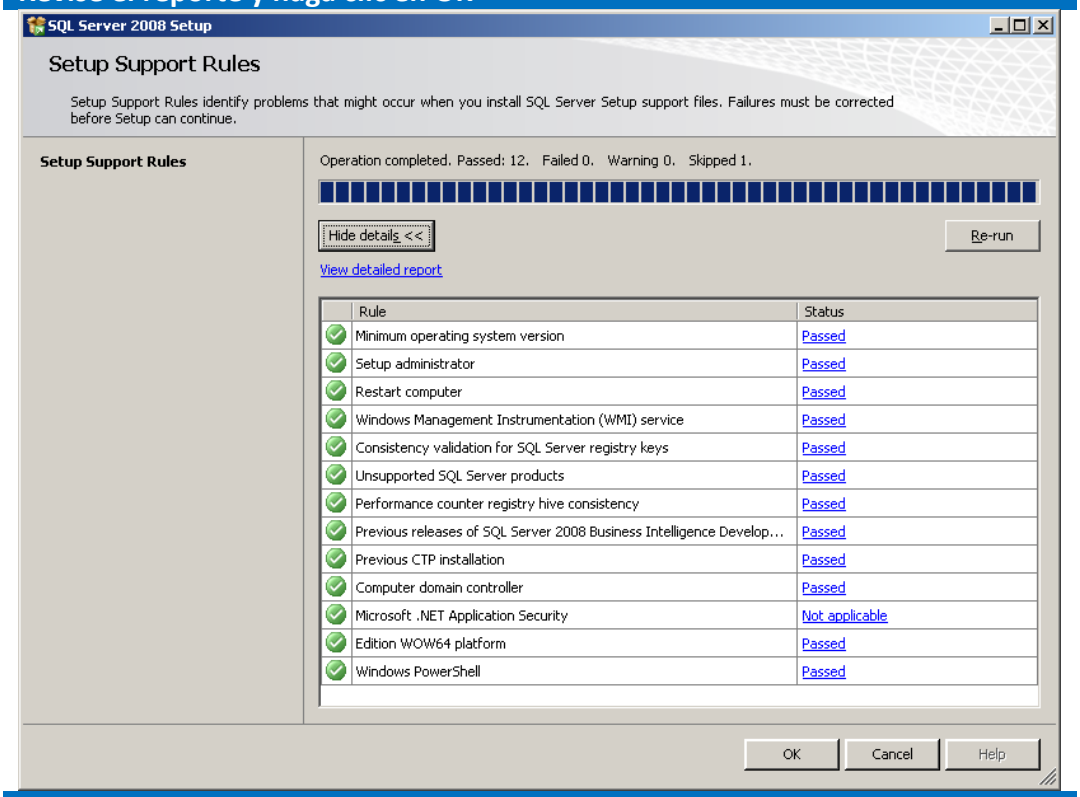
- Antes de instalar es importante que se tenga en cuenta algunas recomendaciones:

- Revisar que cumpla con los requerimientos de hardware y software necesarios para instalar SQL Server 2008, la información al respecto se encuentra en la documentación del producto, al final de este documento se presenta el link hacia dicha documentación
- Crear cuentas para los servicios de SQL Server, estas cuentas deben ser creadas con privilegios mínimos ya que durante el proceso de instalación, el asistente les asignará los permisos necesarios para ejecutar los respectivos servicios. ***La creación de estas cuentas de servicio NO es obligatoria para poder instalar SQL Server, pero es una buena práctica de seguridad***

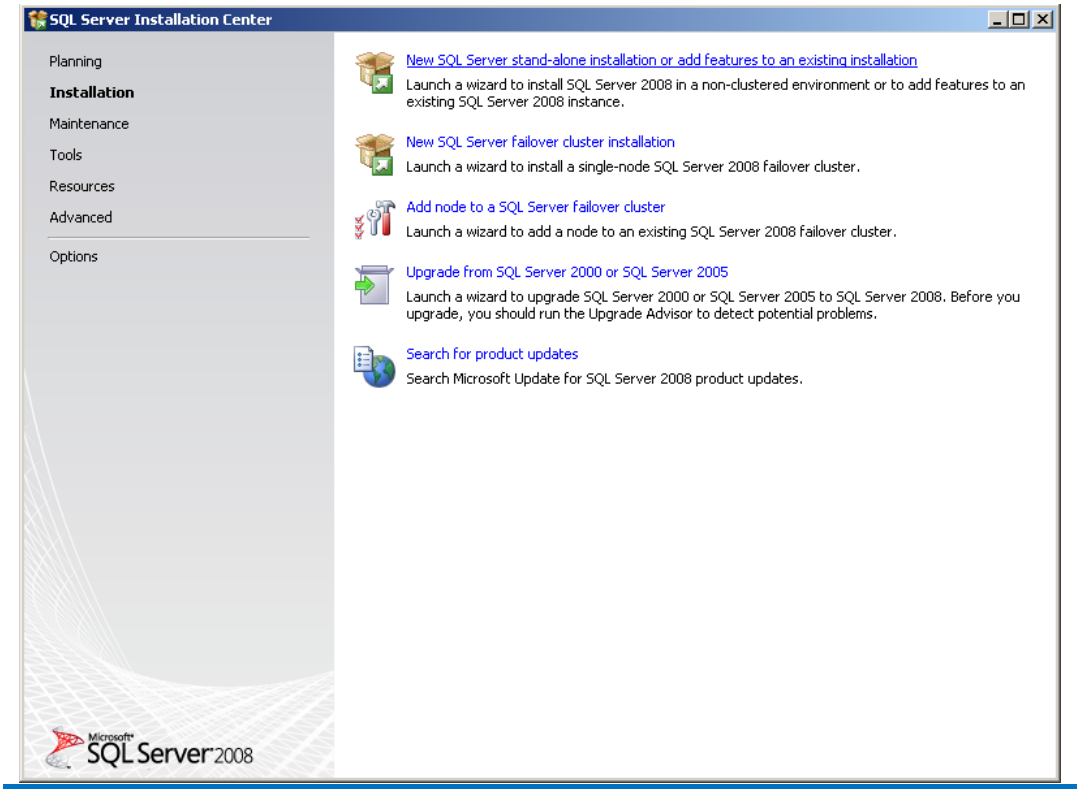
A continuación se muestra los pasos de la instalación:

"En SQL Server Installation Center" es posible revisar información detallada acerca de requerimientos para la instalación, recomendaciones de seguridad y adicionalmente realizar un chequeo de la configuración del sistema. Haga clic en "System Configuration Checker"

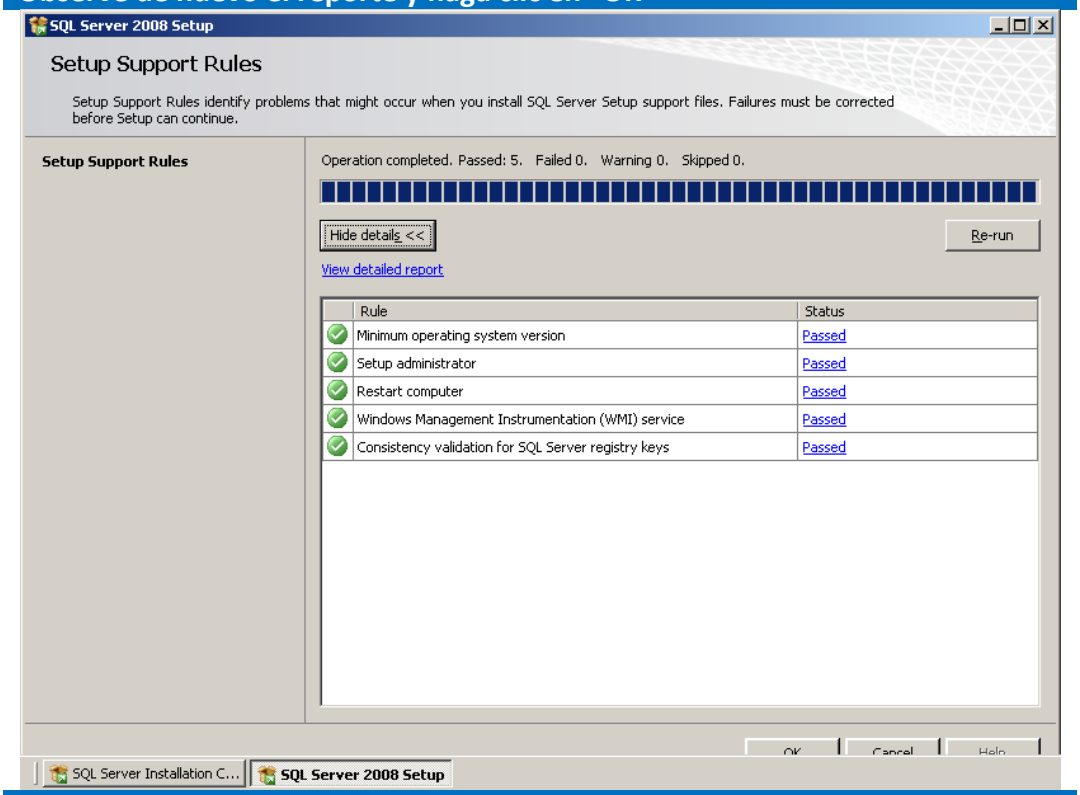
Revise el reporte y haga clic en OK



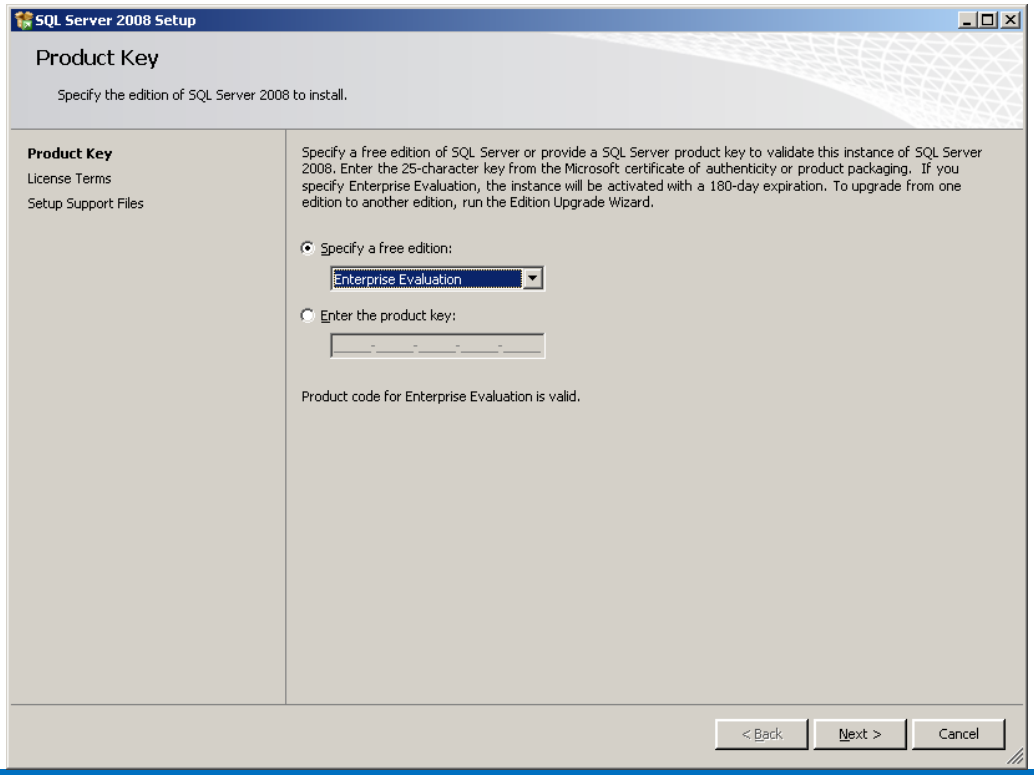
Ahora, vaya al tab "Installation", y allí seleccione la opción "New SQL Server stand alone installation or add features to an existing installation"



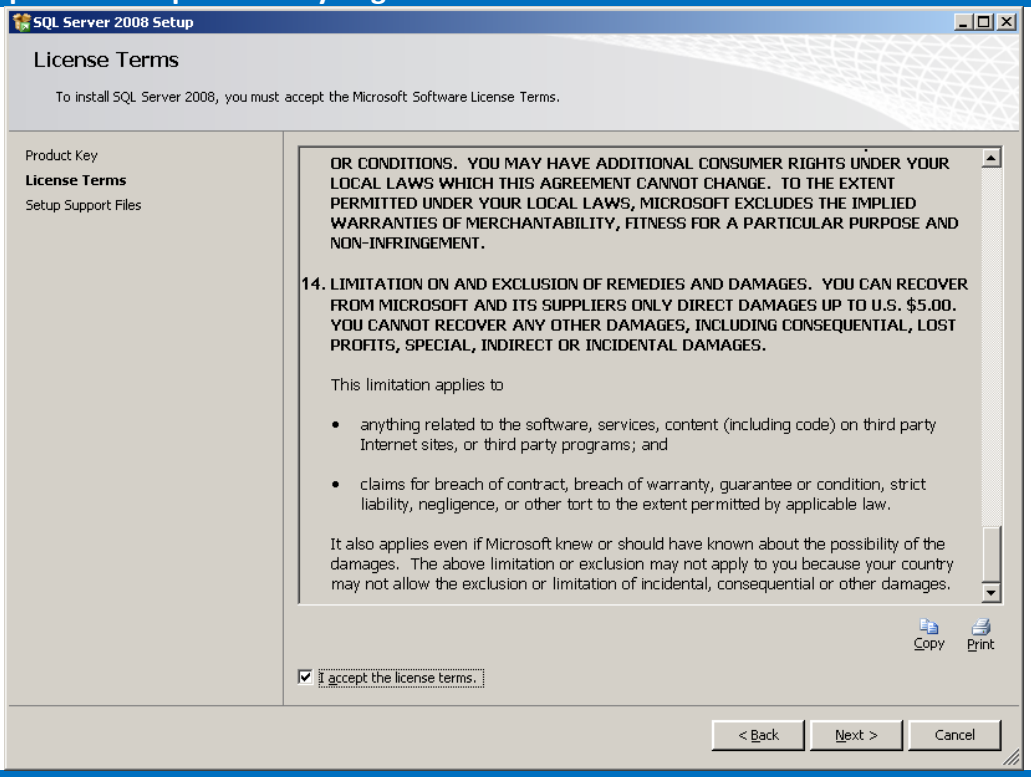
Observe de nuevo el reporte y haga clic en "OK"



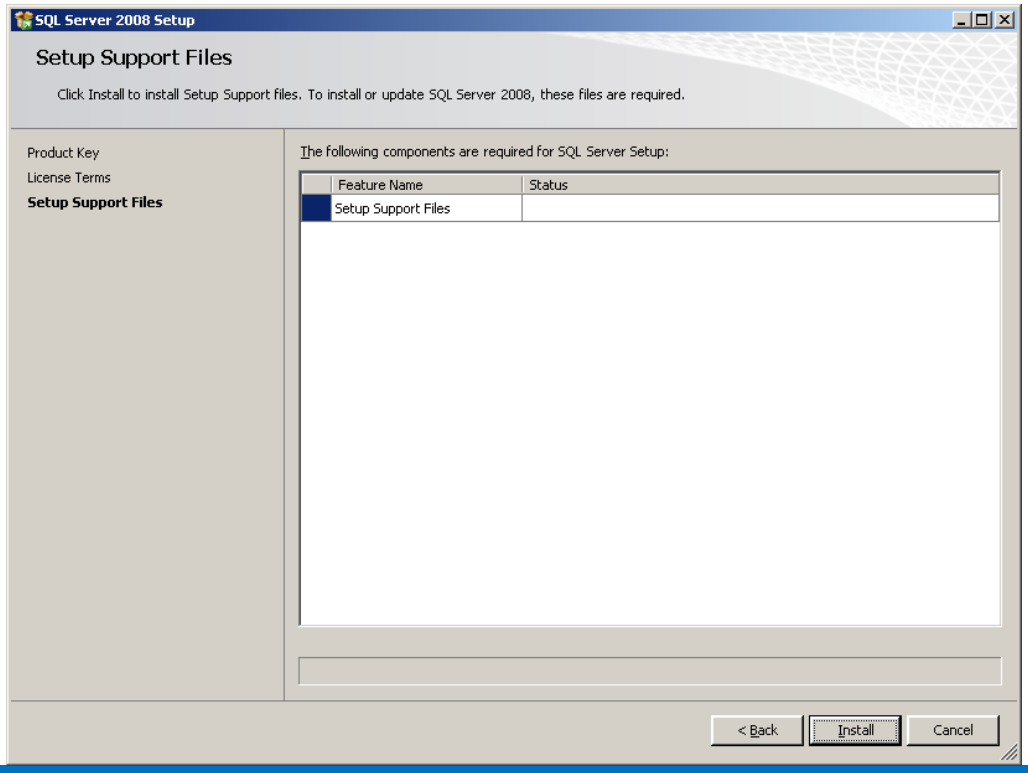
Si está instalando una versión de pruebas (cómo en este ejemplo) de SQL Server, podrá seleccionar la opción correspondiente para la edición que desee; en una instalación diferente, agregue la clave de producto y haga clic en "Next"



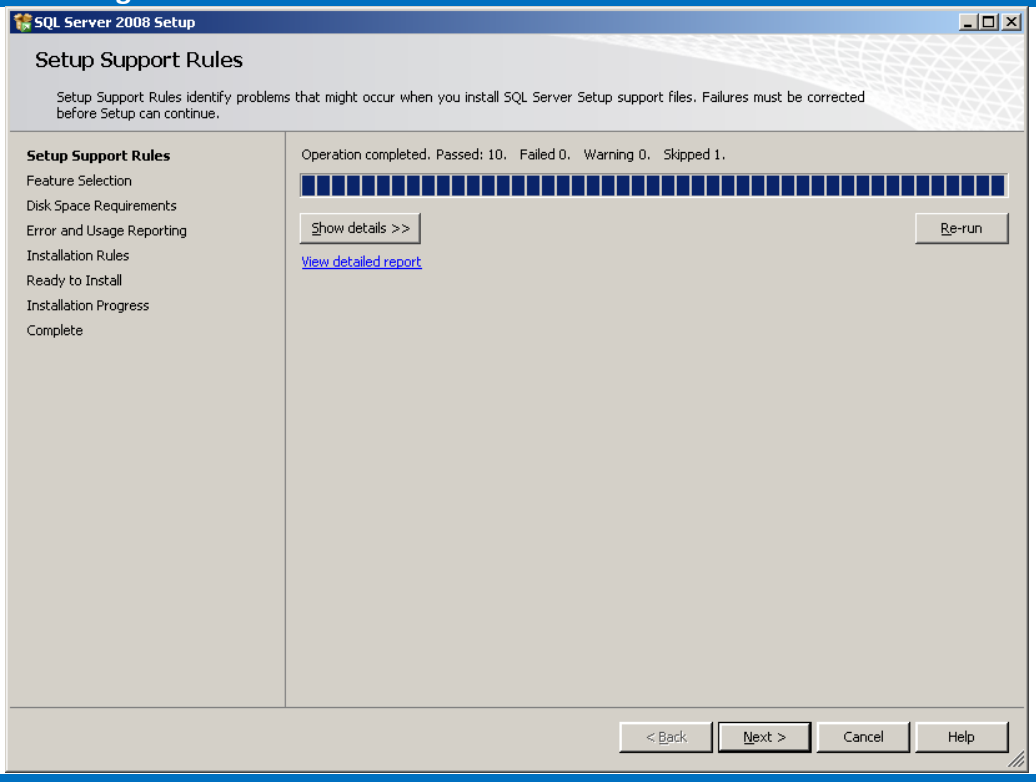
Ahora, lea los términos de licencia y luego, si está de acuerdo seleccione la opción correspondiente y haga clic en "Next"



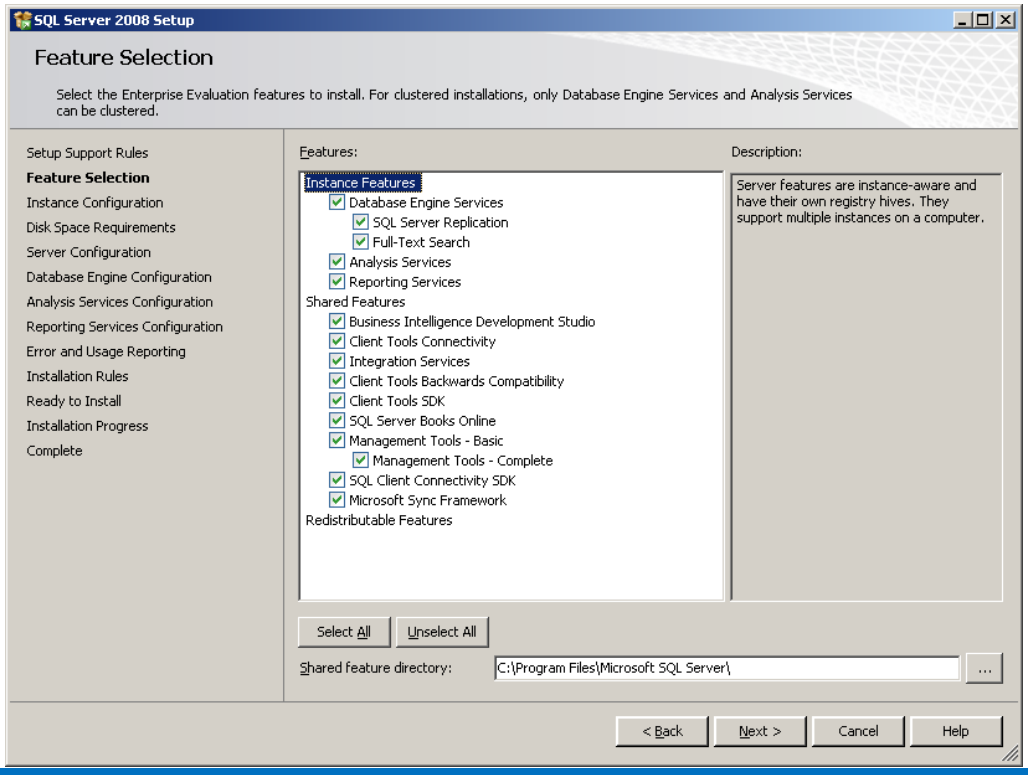
A continuación, se instalan componentes de soporte necesarios para la instalación, haga clic en "Install" para instalarlos



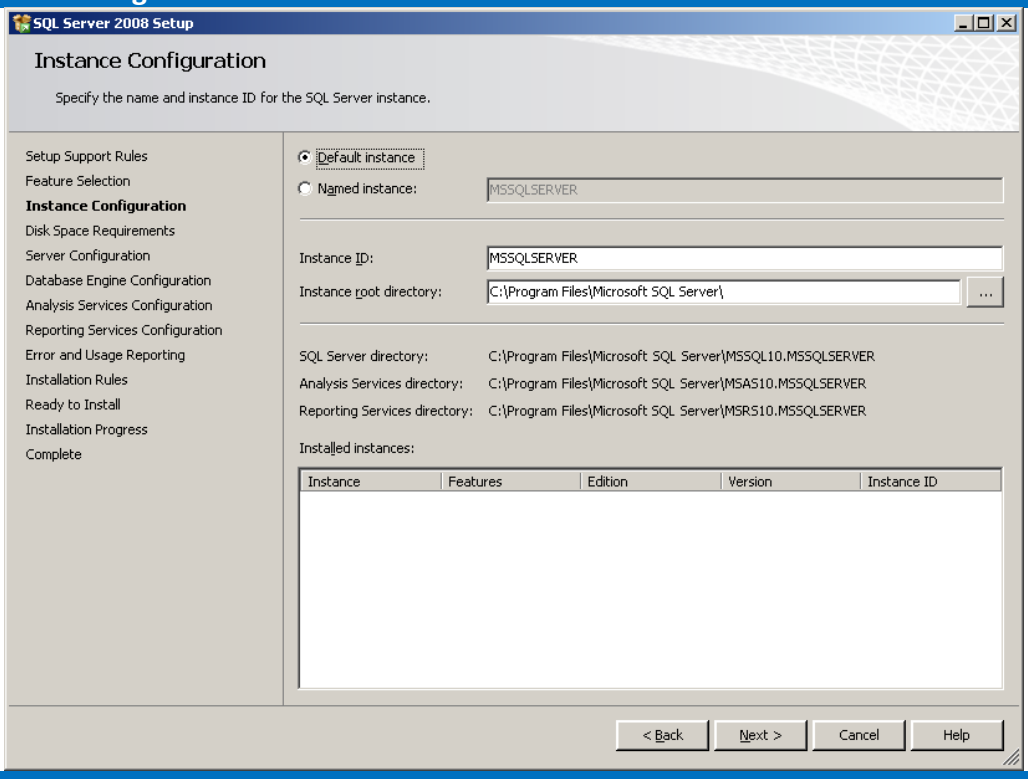
Ahora haga clic en "Next"



Ahora, deberá seleccionar las características de SQL server 2008 que desea instalar; Asegúrese de instalar los servicios que en algún momento vaya a utilizar, si está totalmente seguro que no va a usar un servicio específico cómo Analysis Services, limpie la casilla de verificación junto a él, y haga clic en "Next"

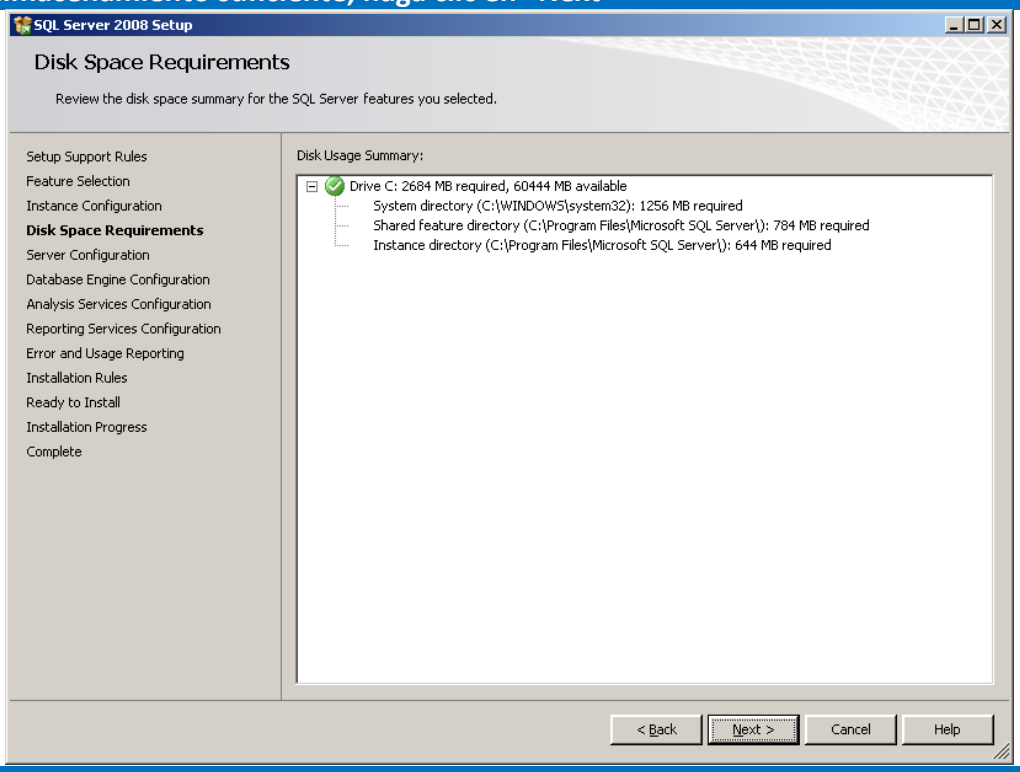


A continuación tendrá que decidir si la instancia que va instalar es una instancia por defecto o nombrada, en el segundo caso tendrá que asignar a esta un nombre con el cual la reconocerá a futuro; si la instancia es creada por defecto, la forma de conectarse a esta desde servidores o equipos clientes remotos, será por medio del nombre de la máquina o de la dirección ip de la misma. Haga clic en "Next"

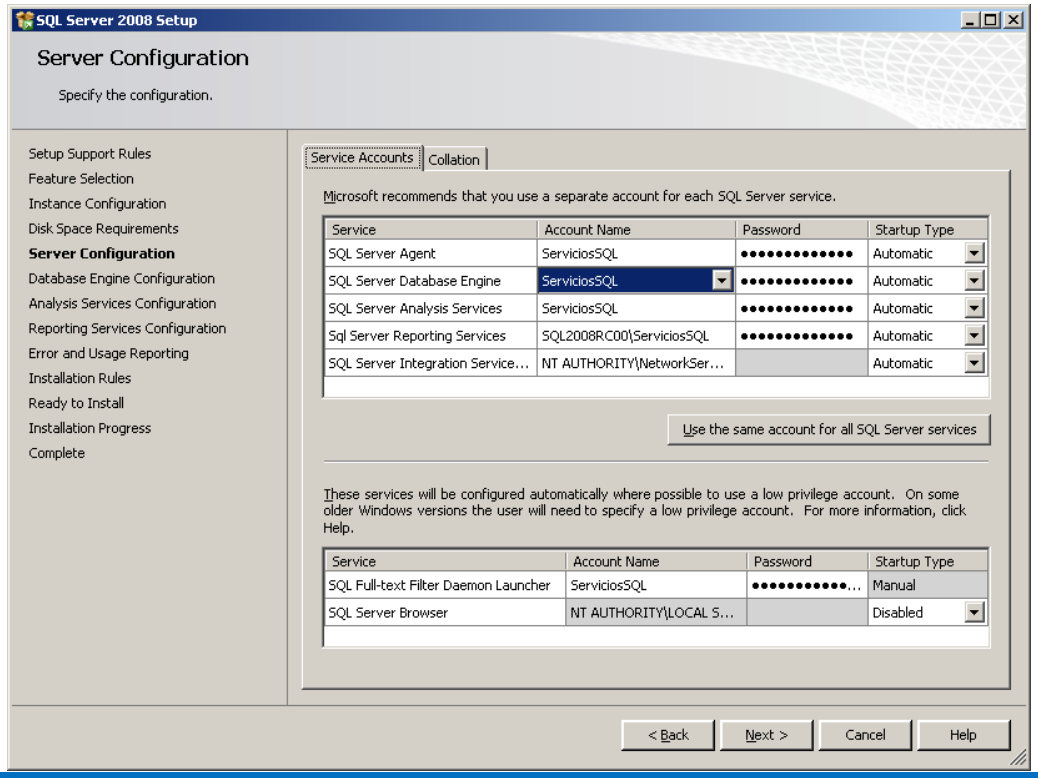


En la siguiente ventana, se encuentra un análisis de requerimientos de

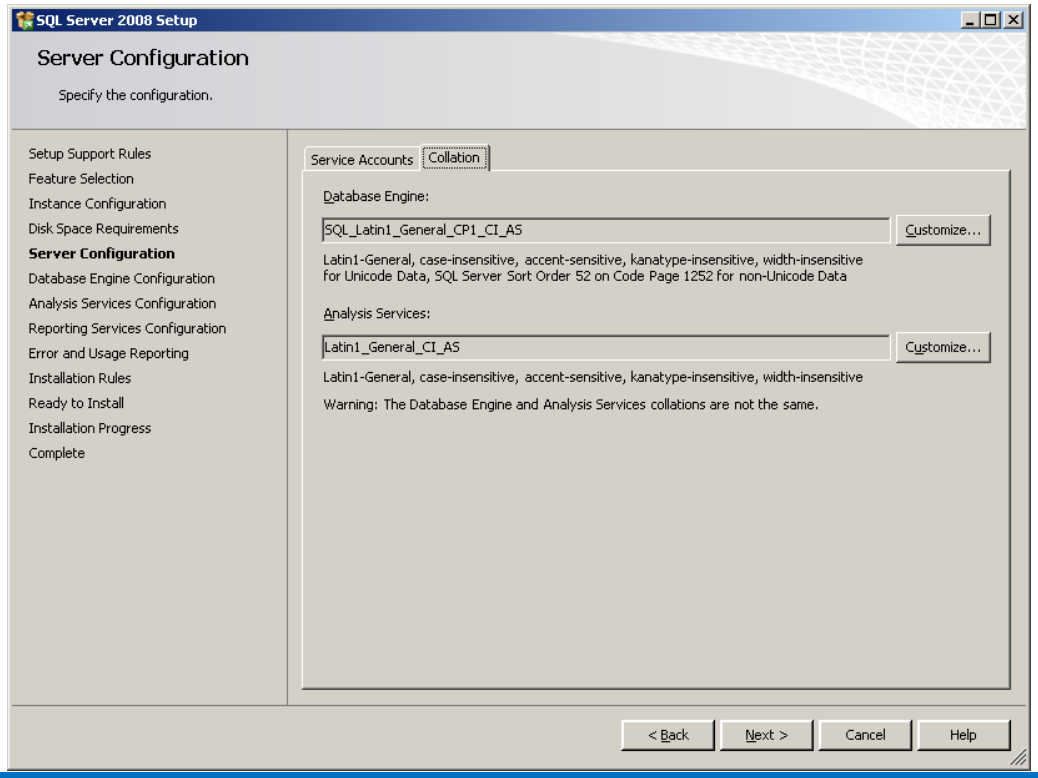
espacio, cuando se haya comprobado que cuenta con el espacio de almacenamiento suficiente, haga clic en "Next"



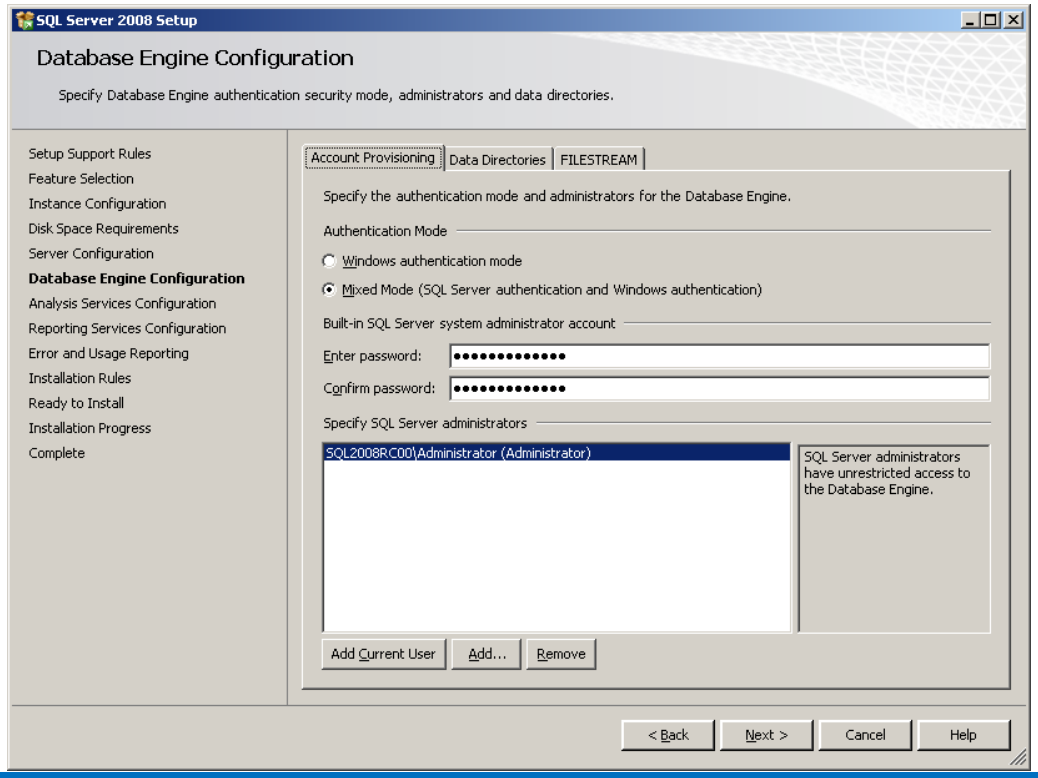
Ahora, usted deberá configurar las cuentas con las cuales se ejecutará el servicio; la recomendación es utilizar diferentes cuentas, sin embargo, en la imagen de la derecha usted puede observar cómo una cuenta es utilizada para ejecutar más de un servicio, en la parte inferior podría seleccionar la opción para utilizar la misma cuenta para todos los servicios, en cuyo caso solamente tendrá que escribir credenciales una vez, pero no estará cumpliendo con buenas prácticas de seguridad. Después de configurar las cuentas, haga clic en el tab "Collation"



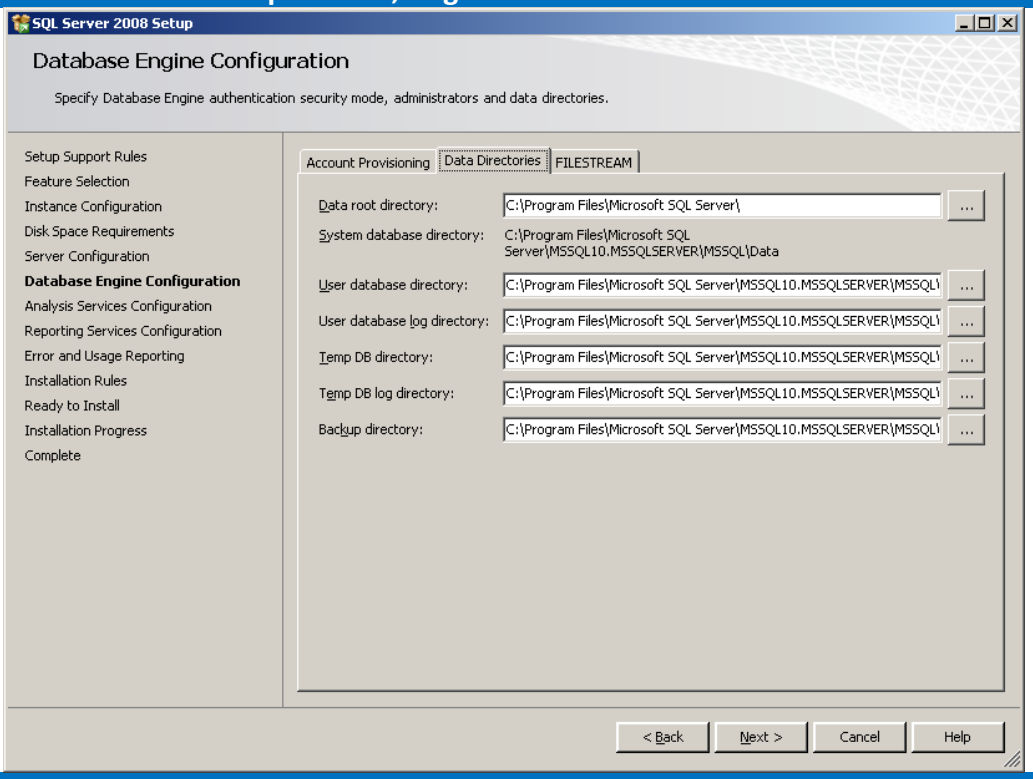
En Collation, observe los métodos de ordenamiento que van a ser utilizados tanto para SQL Server cómo para Analysis Services; es importante que tenga un cuenta si existen regulaciones en su organización acerca del tipo de ordenamiento a utilizar, y de no ser así, busque que tanto las bases de datos, cómo Analysis Services tengan modelos de ordenamiento similares para evitar problemas cuando estos dos componentes se conecten entre sí. Haga clic en Next



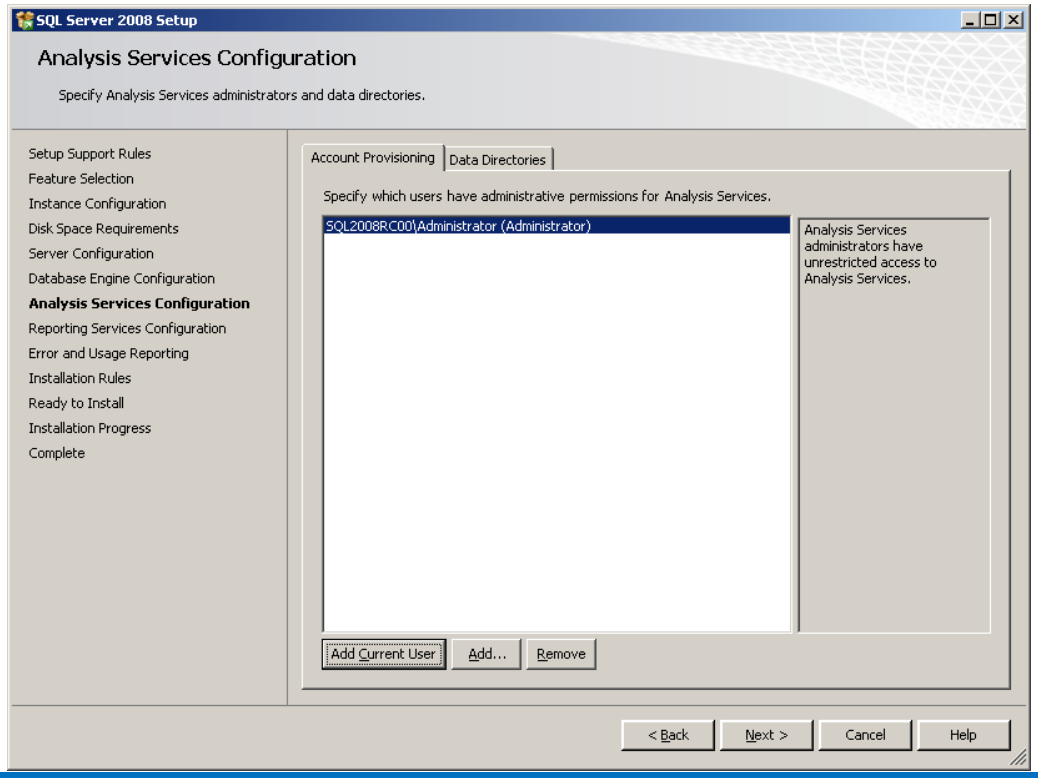
Ahora, tendrá que definir si va a utilizar un modelo de autenticación Windows o Mixto, y si especifica un modelo mixto deberá escribir una contraseña para el usuario administrador tipo SQL; Recuerde que el modo mixto permite la utilización de inicios de sesión tipo SQL (usuarios que no hacen parte de Windows) y es utilizada para dar acceso a SQL Server desde aplicaciones, entre otras cosas. De todas maneras se recomienda por razones de seguridad y mientras sea posible, utilizar el modo de autenticación tipo Windows. Agregue también como administrador a cualquier usuario que vaya a cumplir con dicha tarea, por ejemplo el usuario que está ejecutando la instalación (Add current User) Haga clic en "Data Directories"



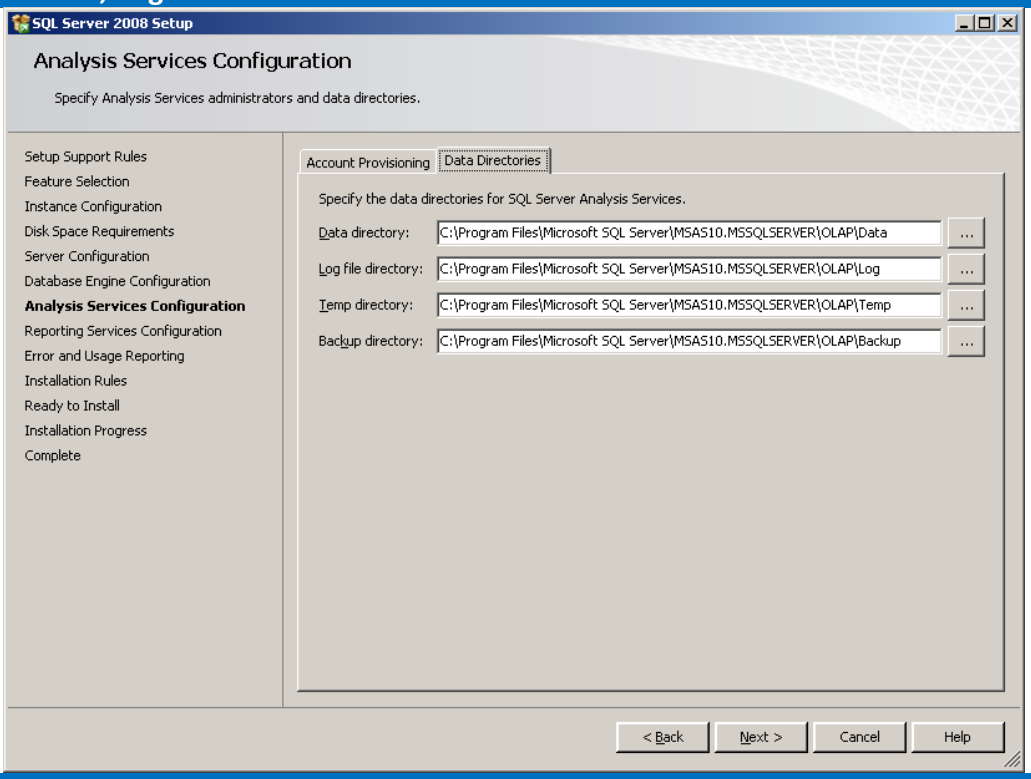
Ahora revise las ubicaciones físicas donde va a quedar instalado SQL Server y cada uno de sus componentes, Haga clic en Next



Agregue los usuarios que van a ser administradores de Análisis Services, puede agregar al usuario con el que está ejecutando la instalación o a cualquier otro usuario, vaya a "Data Directories"

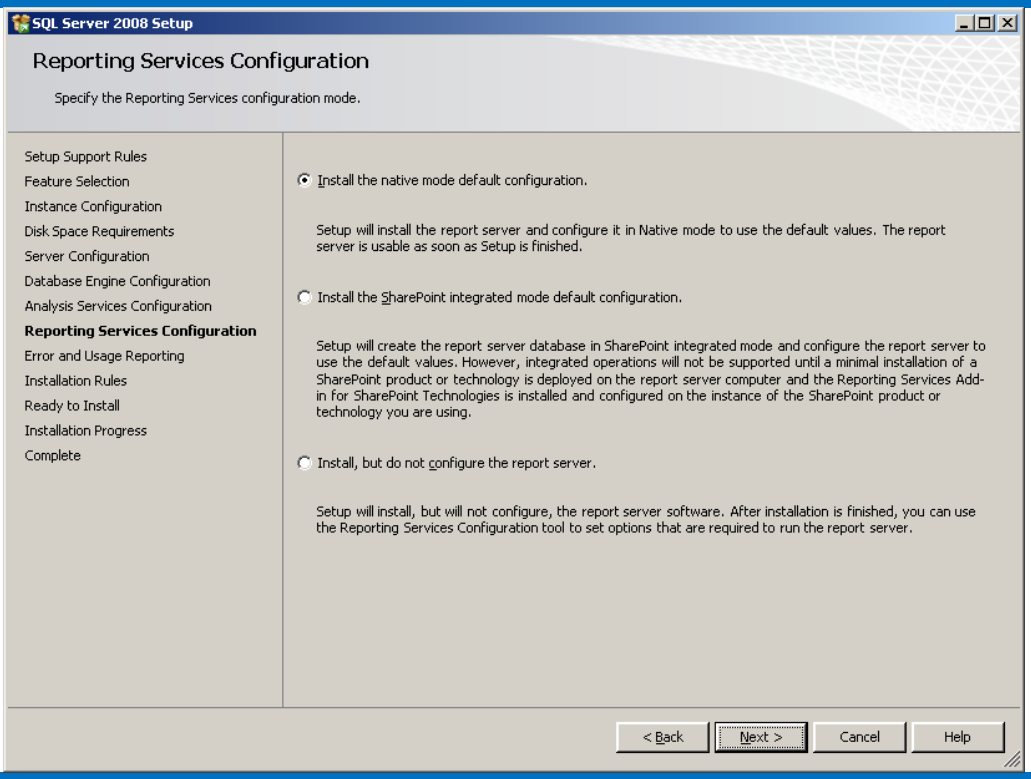


Revise la ubicación donde va a quedar almacenada la información de Analysis Services, haga clic en "Next"

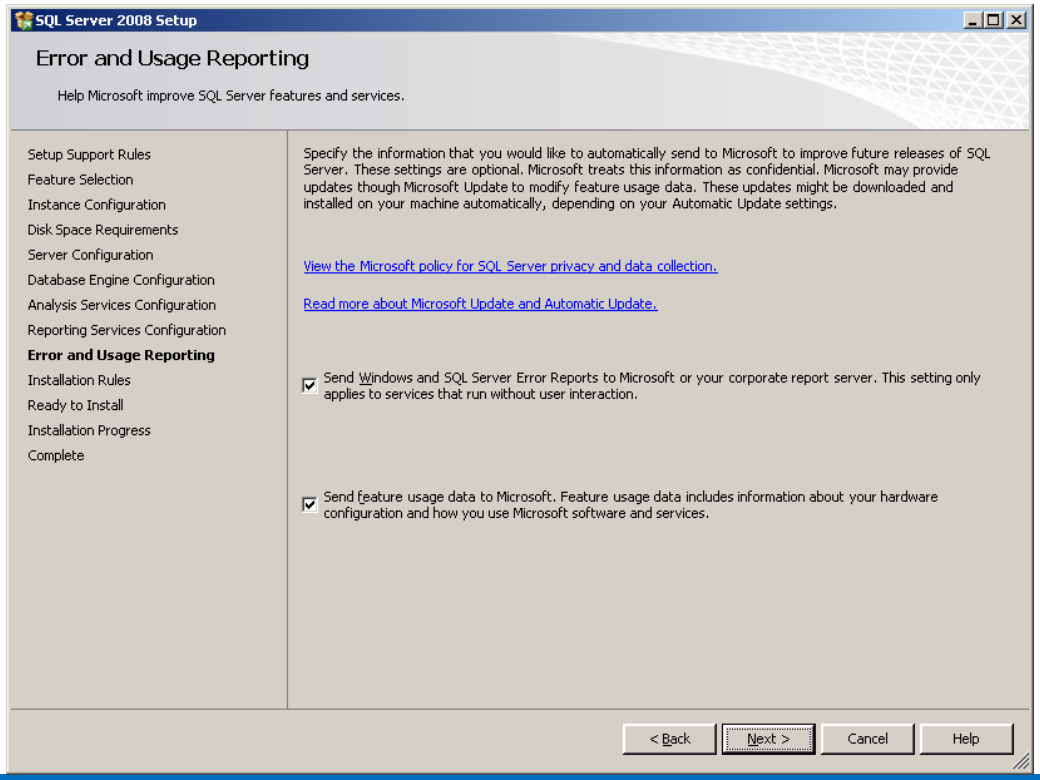


Defina en qué modo va a instalar reporting services, puede instalarlo en el modo nativo (para que pueda usar reporting services una vez termine la instalación sin necesidad de SharePoint), otro es el modo integrado con SharePoint (que almacenara sus reportes en una librería de reportes de

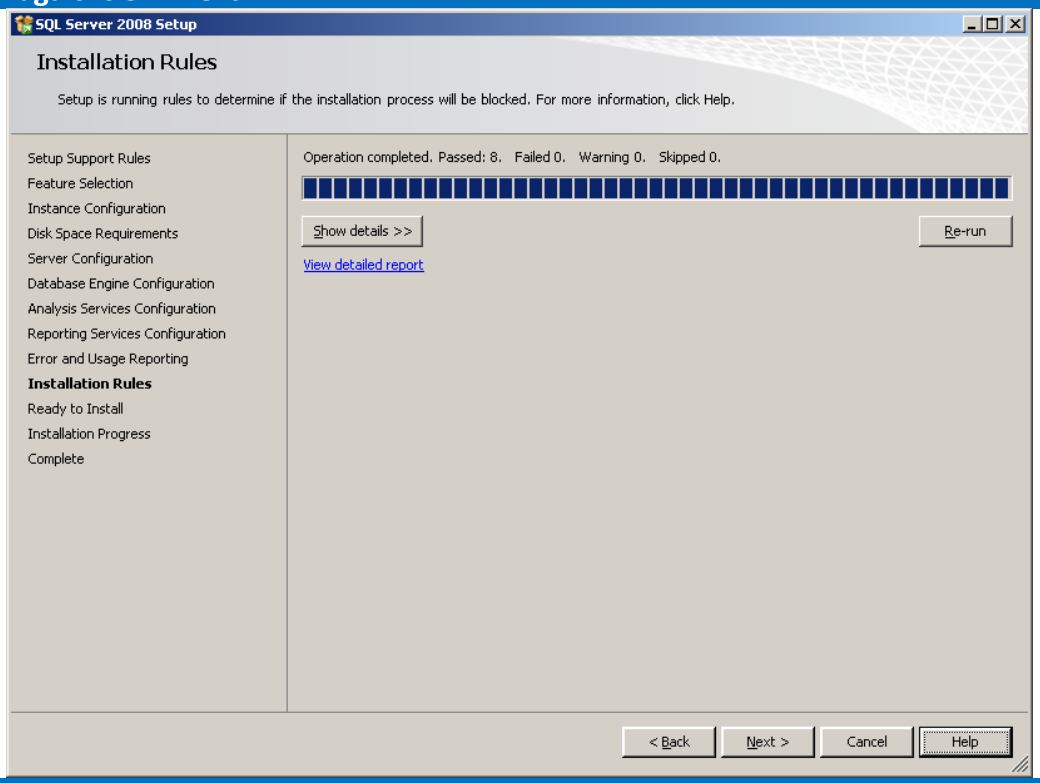
SharePoint) o bien puede instalar Reporting Services pero no configurarlo, lo cual implica que luego debería realizar dicha configuración (esta opción se utilizaría si piensa realizar una configuración escalada de Reporting Services) Para efectos de este ejemplo, seleccionamos el modo nativo. Haga clic en "Next"



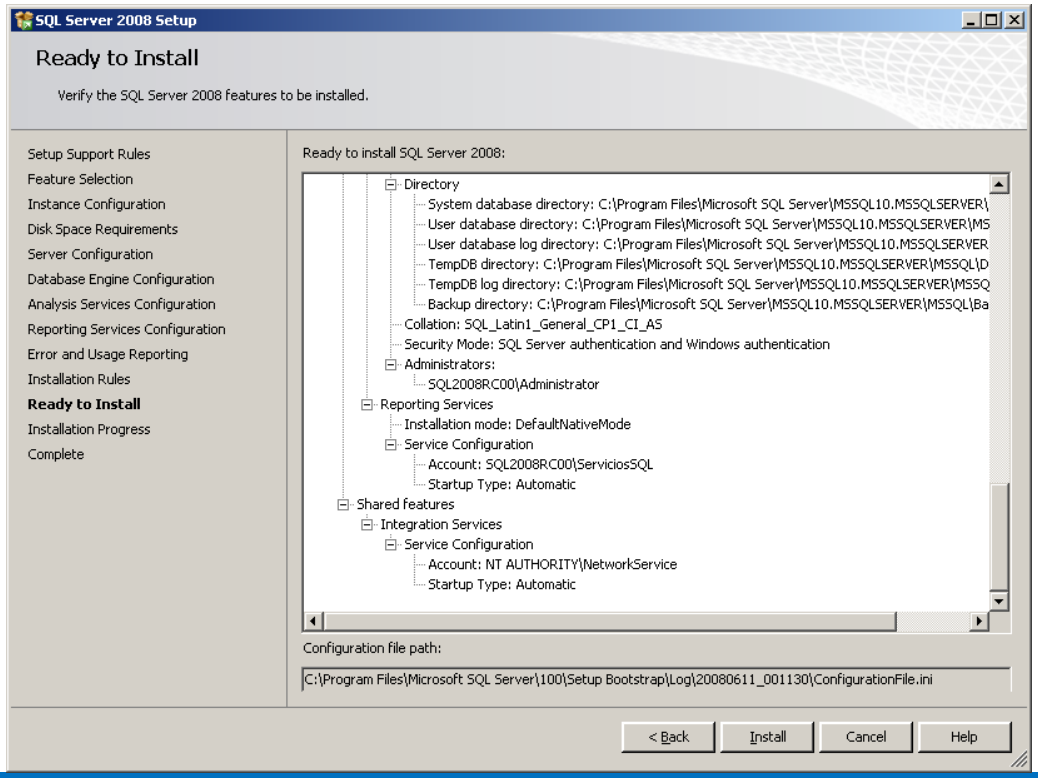
Ahora, seleccione las opciones para que se envíen reportes de errores y de uso de características hacia Microsoft y haga clic en "Next"



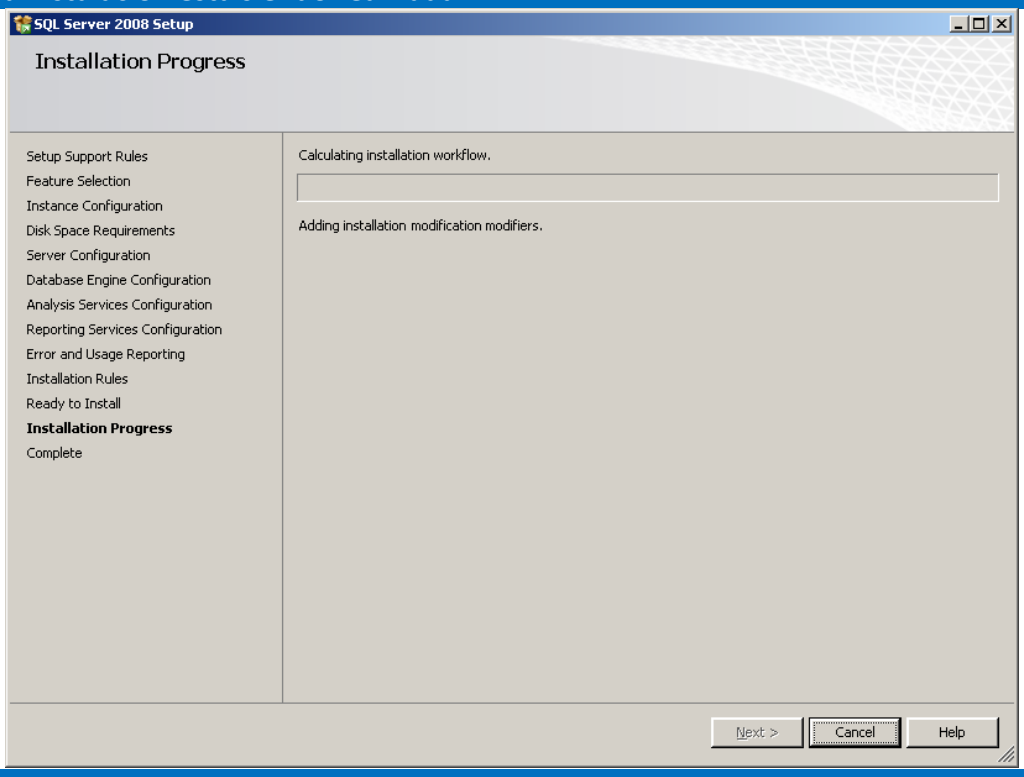
Haga clic en "Next"



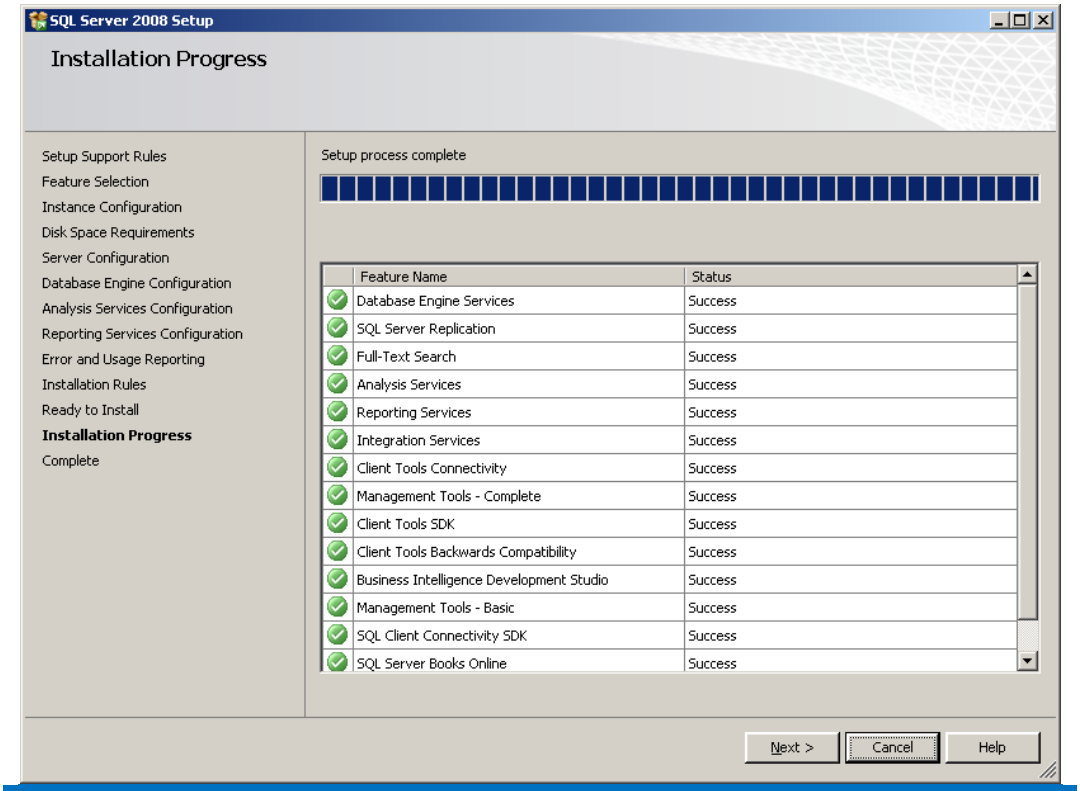
Revise el resumen y haga clic en "Install"



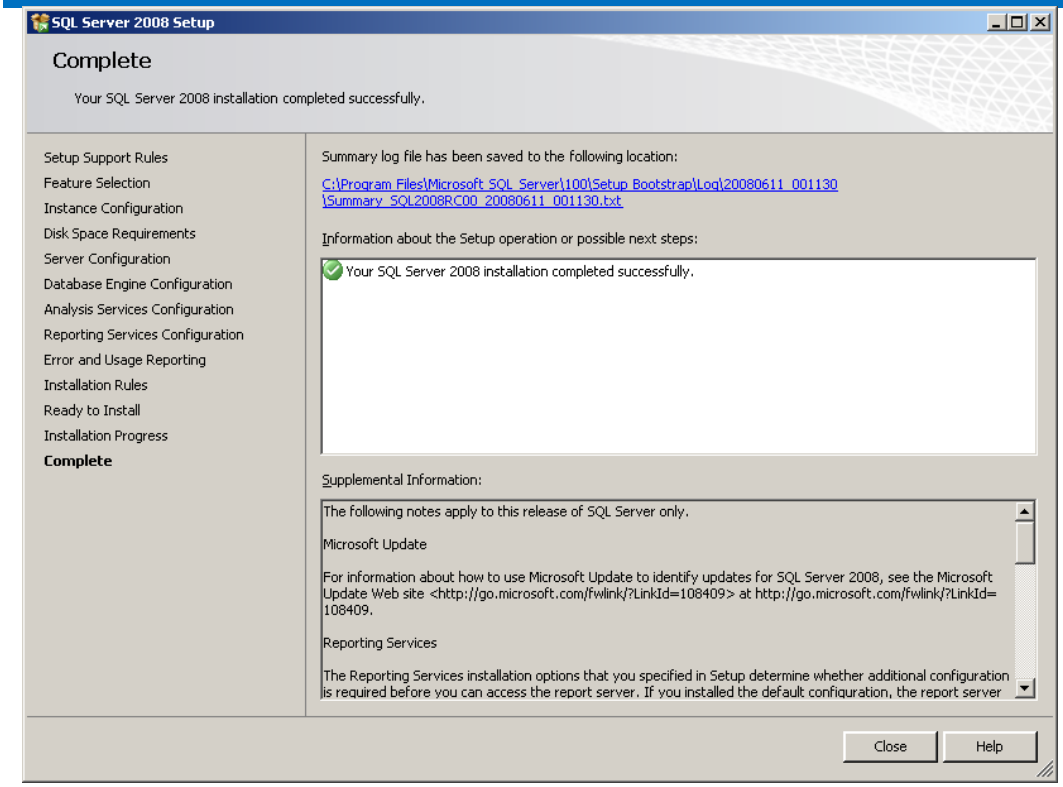
La instalación está siendo realizada



La instalación ha sido completada



Si desea ver un resumen de la instalación, aquí encuentra un link hacia dicho registro de resumen; Haga clic en "Close" para salir, la instalación ha sido terminada.

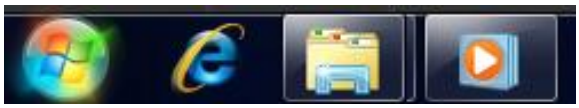


La instalación de todas las ediciones y componentes de SQL Server es similar al ejemplo que se mostró anteriormente, recuerde que puede instalar todos los componentes o solo algunos de en un equipo; por ejemplo, si se deseara instalar únicamente los componentes cliente para que desde allí pueda conectarse a un servidor de SQL Server ubica do en un lugar remoto, bastaría con seguir el mismo proceso de instalación y en la página de selección de componentes el seccionar únicamente los componentes cliente; de la misma manera podría instalarse únicamente la documentación (libros en pantalla), para tener una buena fuente de información y capacitación en cualquier máquina .

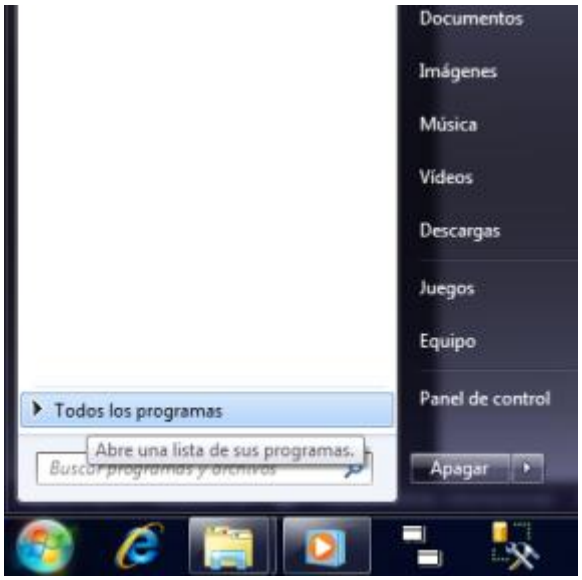
Nota: El SQL no distingue las mayúsculas de minúsculas así que para este programa es lo mismo "**hola**" que "**HOLA**"

INGRESAR AL SQL SERVER 2008

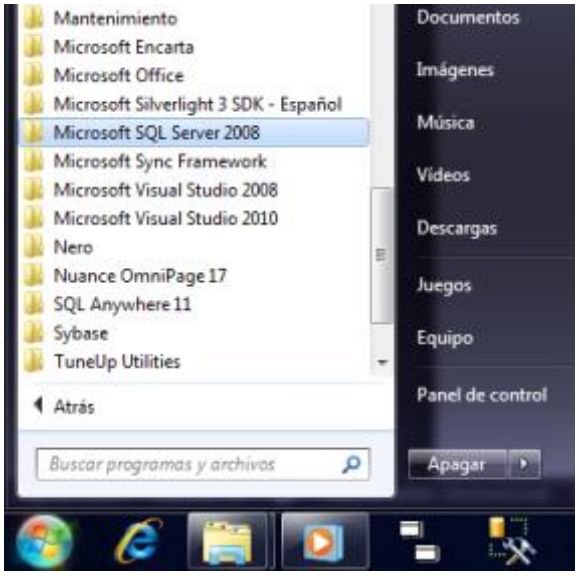
1. Botón inicio



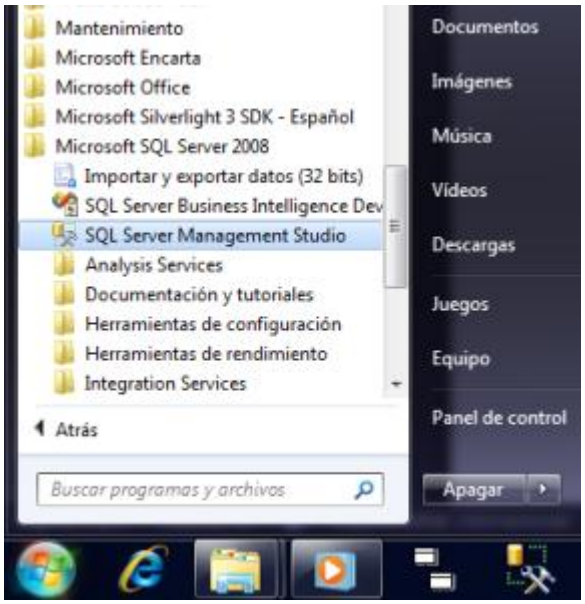
2. Todos los programas



3. Microsoft SQL Server 2008



4. SQL Server Management Studio

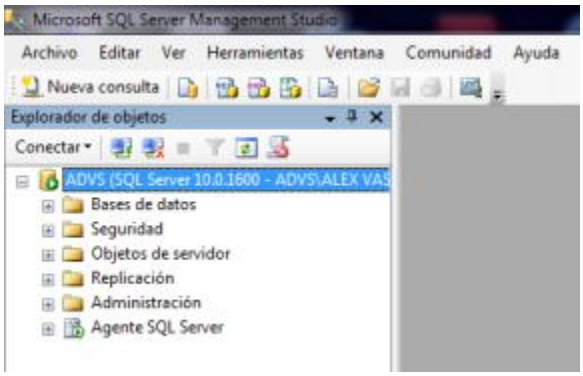


5. Ingresando

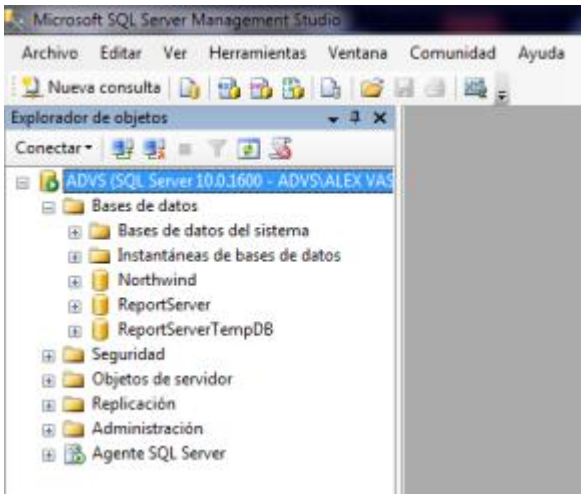
6. Eliges la opción ***Autenticación de Windows***

Cuando ingresas por ***Autenticación de Windows*** se refiere al usuario que tiene por defecto la PC, en cambio sí eliges ***Autenticación de SQL Server*** es cuando tienes otro usuario, el cual se le da al instalar el ***SQL Server 2008***

7. Luego conectar, veras…



8. Despliegas ***Base Datos***

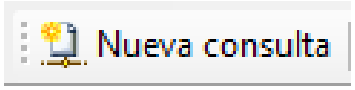


En ella veras las base datos que se instalan con el SQL Server y la base datos que vas creando.
Nosotros trabajaremos con la base datos ***Northwind***

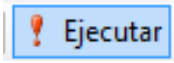
CONSULTAS EN LA BASE DATOS NORTHWIND

1. Crear una consulta

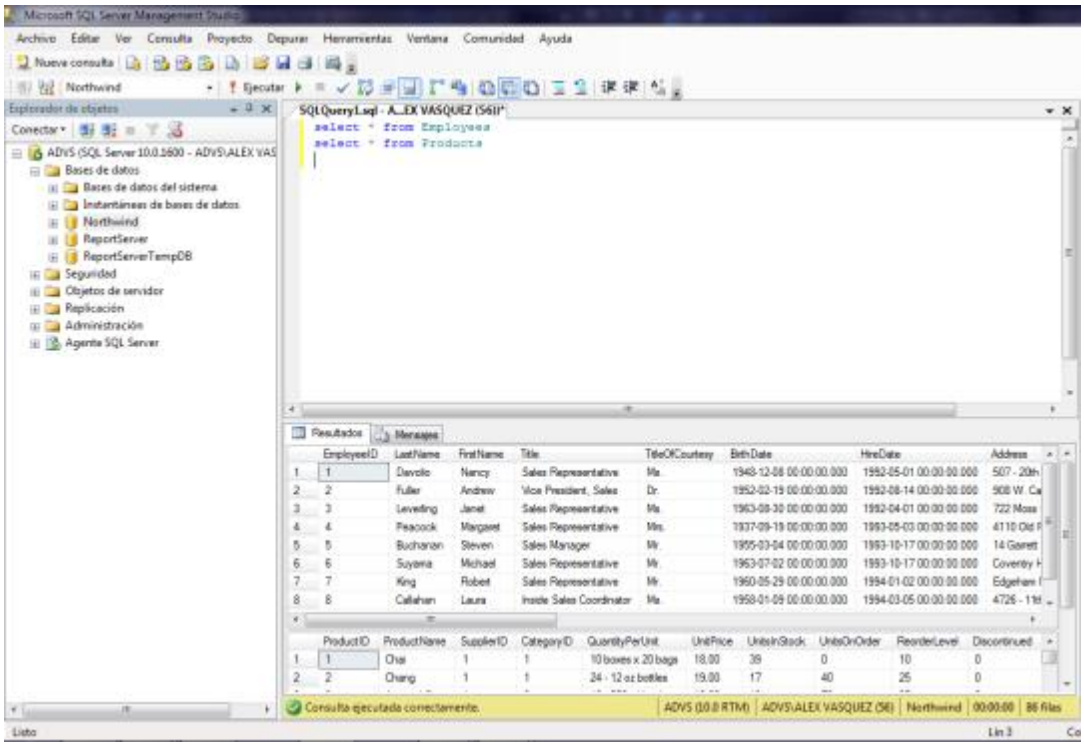
Para crear una nueva consulta, primero debes seleccionar la base datos que vas a trabajar en este caso es Northwind, luego haces clic derecho y nueva consulta; o después de seleccionar la base datos haces clic en el icono nueva consulta.



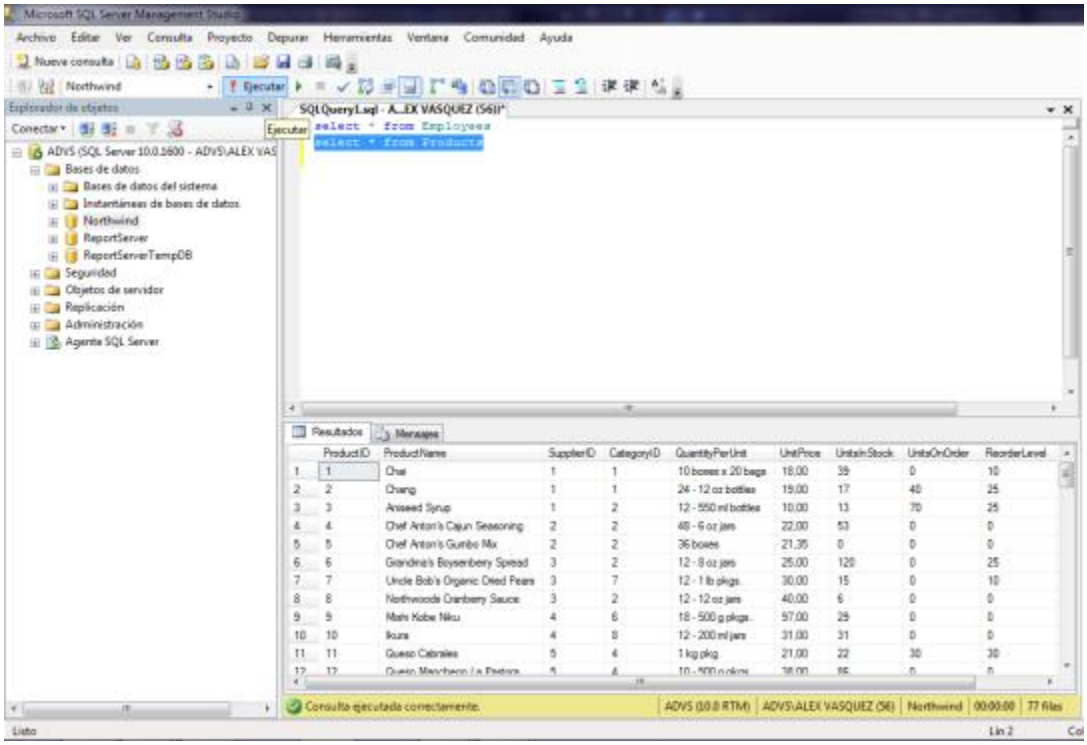
2. Ejecutar consulta

Para ejecutar una consulta hacemos clic en el icono ejecutar 

o solo presionamos la tecla **F5**.
Si tenemos más de una consulta y lo ejecutamos directo se ejecutaran todas las consultas.



En cambio si queremos ejecutar una consulta específica primero lo seleccionamos y luego lo ejecutamos.



3. SELECT

Esta sentencia me permitirá seleccionar los campos de una o varias tablas, si queremos toda la información colocaremos asterisco (*), de lo contrario mencionamos los campos.
Si estamos utilizando varias tablas y el mismo nombre del campo, tenemos que especificar a qué tabla pertenece el campo que queremos, para no generar una ambigüedad.

4. FROM

Con esta sentencia indicaremos en que tabla buscamos información

Ejemplos:

```
select * from Employees
select * from Products
```

5. WHERE

Esta sentencia filtra registros que cumplan la condición dada, verdadera o falsa.
Si quisiera unir más condiciones utilizare los operadores lógicos **AND** u **OR**, esto será de acuerdo a la información de lo que se desea obtener.

Ejemplos:

a) Una condición

```
select * from Products
where CategoryID = 2
```

b) Varias condiciones utilizando AND

```
select * from Products
where CategoryID = 6 and CategoryID=4
```

c) Cuando utilizamos varias condiciones con 2 de un **AND** utilizaremos la sentencia **BETWEEN**. Pero para utilizarlo debe cumplir:

- Debe tener dos condiciones, que uno sea mayor igual (>=) y el otro menos igual (<=)
- Tiene que estar unidas con **AND**
- Que el campo sea el mismo

Ejemplo 1:

```
select ProductID, ProductName,UnitPrice
from Products
where UnitPrice between 30 and 60
```

Ejemplo 2:

```
select ProductID, ProductName,UnitPrice,UnitsInStock
from Products
where UnitsInStock between 0 and 10
```

d) Varias condiciones utilizando **OR**

```
select * from Products
where CategoryID = 6 or CategoryID=4
```

e) Varias condiciones utilizando más de un **OR**

```
select * from Products
where CategoryID = 6 or CategoryID=4
or CategoryID=8
```

f) Cuando tenemos varias condiciones con más de un **OR** utilizamos la sentencia IN.
IN: Reduce el código, asignado a consultas con condiciones **OR**. Comando que sirve para búsquedas en varios campos
Pero para utilizar esta sentencia **IN** tiene que cumplir:

- Que todas las condiciones estén unidas con el **OR**.
- Que pertenezcan al mismo campo.
- Que los operadores sean iguales (la misma condición)

Ejemplos 1:

```
select * from Products
where CategoryID in (4,6,8,2)
```

Ejemplo 2:

```
select * from Products
where SupplierID in (3,5,8)
```

Ejemplo 3:

```
select ProductName, UnitPrice,UnitsInStock
from Products
where SupplierID in (2,3,5,8)
```

Ejemplo 4:

```
select * from Customers
where Country in ('USA','UK')
```

6. ORDERDATE:

Permite seleccionar una parte de la fecha de un registro, y establecer un valor de comparación a este, estableciendo verdadero o falso.

Ejemplo 1:

```
select *
from dbo.Orders
WHERE YEAR (OrderDate)='1996'
```

7. DATEPART

Permite establecer argumentos de comparación de un determinado campo, siendo este día (dd), mes (mm), año (yyyy), etc.

Ejemplo 1:

```
select *
from Orders
```

```
where EmployeeID in (2,3,4,5) and
    CustomerID like '[A-G]%' AND
    DATEPART (MM,OrderDate) ='07' and

    DATEPART (DD,OrderDate)='31'
```

8. ORDER BY

Si queremos ordenar la información tiene que estar explícito en el **SELECT**. Como ya mencionemos sirve para ordenar uno o más campos, ya sea en forma ascendente como descendente. Cuando ordenamos y no lo aclaramos la forma como queremos que se ordene por defecto lo hace ascendentemente.

Forma ascendente

Ejemplo 1:

```
select ProductID, ProductName,UnitPrice,UnitsInStock
from Products
where UnitsInStock between 0 and 10
order by ProductName
```

Ejemplo 2:

```
select ProductID, ProductName,UnitPrice,UnitsInStock
from Products
where UnitsInStock between 0 and 10
order by ProductName asc
```

Forma descendente

Ejemplo 1:

```
select ProductID, ProductName,UnitPrice,UnitsInStock
from Products
where UnitsInStock between 0 and 10
order by ProductName desc
```

Ejemplo 2:

```
select ProductID, ProductName,UnitPrice,UnitsInStock
from Products
where UnitsInStock between 0 and 10
order by UnitPrice desc
```

Ejemplo 1:

```
select ProductID, ProductName,UnitPrice,UnitsInStock, CategoryID
from Products
where UnitsInStock between 0 and 10
order by CategoryID desc
```

9. LIKE

```
select ProductID, ProductName
from Products
where ProductName='chang'
```

Se utiliza para comparar una expresión de cadena

```
select ProductID, ProductName
from Products
where ProductName like 'chang'
```

Si queremos buscar por una palabra, letra o frase; entra a tallar un comodín que es el signo de porcentaje (**%**)

En este caso todos los nombres que empiecen con C

```
select ProductID, ProductName
from Products
where ProductName LIKE 'c%'
```

En este caso la palabra que en su composición tenga C

```
select ProductID, ProductName
from Products
where ProductName LIKE '%c%'
```

En este caso todos los que terminen en S

```
select ProductID, ProductName
from Products
where ProductName LIKE '%s'
```

10. TOP

Devuelve un determinado número de registros. Esta sentencia se guía por la ordenación.

Select top n * from tabla

Donde "n" es la cantidad de registros quiere

Ejemplo 1:

```
select top 3*
from Products
```

Ejemplo 2: Obtener los 15 primeros registros de detalle de la orden q tengan mayor cantidad vendida

```
Select top 15 OrderId,ProductID,Quantity
from Order_Details
order by Quantity desc
```

Ejercicios

Para hacer comentarios en el SQL SERVER es con dos guiones (--) para una línea, si queremos comentar más de una línea utilizaremos (/* comentario */) para abrir y cerrar el comentario. Usando la Base Datos Northwind, realizar las siguientes consultas:

```
select * from Products
where Discontinued = 'true'
```

==========

```
select * from Products
where Discontinued <> 'FALSE'
```

==========

```
select * from Orders
where YEAR( OrderDate) ='1996'
```

==========

Ejercicio 1: Seleccionar todos los campos de la tabla clientes, ordenado por nombre del contacto de la compañía, alfabéticamente.

```
select *from Customers
order by ContactName
```

Ejercicio 2: Seleccionar todos los campos de la tabla órdenes, ordenados por fecha de la orden, descendentemente.

```
select *from Orders
order by OrderDate desc
```

Ejercicio 3: Seleccionar todos los campos de la tabla detalle de la orden, ordenada por cantidad pedida, ascendentemente.

```
select *from [Order Details] --cuando tiene espacio
--select *from Order_Details
order by Quantity
```

Ejercicio 4: Obtener todos los productos, cuyo nombre comienzan con la letra P y tienen un precio unitario comprendido entre 10 y 120.

```
select *from Products
where (ProductName like 'p%')and
```

```
        (UnitPrice between 10 and 20)
```

Ejercicio 5: Obtener todos los clientes de los países de: USA, Francia y UK.

```
select *from Customers
where Country in('USA','Francia','UK')
```

Ejercicio 6: Obtener todos los productos descontinuados y sin stock, que pertenecen a las categorías 1, 3, 4 y 7.

```
select *from Products
where (Discontinued=1)and
      (UnitsInStock=0)and
      (CategoryID in (1,3,4,7))
```

Ejercicio 7: Obtener todas las ordenes hechas por el empleado con código: 2, 5 y 7 en el año 1996

```
select *from Orders
where (EmployeeID in (2,5,7)) and
      (YEAR(OrderDate)='1996')

--otra forma

select *from Orders
where EmployeeID in (2,5,7) and
    datepart (yyyy,OrderDate)='1996'
    --datepart (lo que se quiere sacar, de qué lugar)
```

Ejercicio 8: Seleccionar todos los clientes que cuenten con FAX

```
select *from Customers
where Fax <>'isnull'
-- where Fax is not null --otra forma
```

Ejercicio 9: Seleccionar todos los clientes que no cuenten con FAX, del país de USA

```
select *from Customers
where (Fax is null) and
       (Country='USA')
```

Ejercicio 10: Seleccionar todos los empleados que cuentan con un jefe.

```
select *from Employees
```

```
where ReportsTo is not null
```

Ejercicio 11: Seleccionar todos los campos del cliente, cuya compañía empiece con la letra de A hasta la D y pertenezcan al país de USA, ordenarlos por la dirección.

```
select *from Customers
where (CompanyName like '[A-D]%') and
        (Country='USA')
order by Address
```

Ejercicio 12: Seleccionar todos los campos del proveedor, cuya compañía no comience con las letras de la B a la G, y pertenezca al país de UK, ordenarlos por nombre de la compañía.

```
select *FROM Suppliers
where (CompanyName like'[^B-G]%') and
        (Country='UK' )
--where (CompanyName not like'[B-G]%') and
--     (Country='UK')
order by CompanyName
```

Ejercicio 13: Seleccionar los productos vigentes cuyos precios unitarios están entre 35 y 250, sin stock en almacén. Pertenecientes a las categorías 1, 3, 4, 7 y 8, que son distribuidos por los proveedores 2, 4, 6, 7 y 9.

```
select *FROM Products
where (Discontinued=0)and
        (UnitPrice between 35 and 250)and
        (UnitsInStock=0)and
        (CategoryID in (1,3,4,7,8))and
        (SupplierID in (2,4,6,7,9))
```

Ejercicio 14: Seleccionar todos los campos de los productos descontinuados, que pertenezcan a los proveedores con códigos: 1, 3, 7, 8 y 9, que tengan stock en almacén, y al mismo tiempo que sus precios unitarios estén entre 39 y 190, ordenados por código de proveedor y precio unitario de manera ascendente.

```
select *FROM Products
where (Discontinued=1) and
        (SupplierID in (1,3,7,8,9))and
        (UnitsInStock<>0)and
        (UnitPrice >39 and UnitPrice<190)
order by  SupplierID, UnitPrice asc
```

Ejercicio 15: Seleccionar los 7 productos con precios más caros, que cuenten con stock en almacén.

```
SELECT TOP 7 *FROM Products
where UnitsInStock<>0
ORDER BY UnitPrice desc
```

Ejercicio 16: Seleccionar los 9 productos, con menos stock en almacén, que pertenezcan a la categoría 3, 5 y 8.

```
SELECT TOP 9
*FROM Products
where CategoryID in (3,5,8)
ORDER BY UnitsInStock
```

Ejercicio 17: Seleccionar las órdenes de compra, realizadas por el empleado con código entre el 2 y el 5, además de los clientes con códigos que comienzan con las letras de la A hasta la G, del 31 de Julio de cualquier año.

```
select *from Orders
where (EmployeeID between 2 and 5)and
        (CustomerID like '[A-G]%') and
       (DATEPART(mm,OrderDate)='07') and
      (DATEPART(dd,OrderDate)='31')
```

Ejercicio 18: Seleccionar las órdenes de compra, realizadas por el empleado con código 3, de cualquier año pero solo de los últimos 5 meses (agosto - Diciembre).

```
select *from Orders
where (EmployeeID=3) and
        (MONTH (OrderDate)in(8,9,10,11,12))
    -- (MONTH (OrderDate) between 8 and12))
order by MONTH(OrderDate)
```

Ejercicio 19: Seleccionar los detalles de las órdenes de compra, que tengan un monto de cantidad pedida entre 10 y 250.

```
--[Solo si me piden la cantidad esto es correcto]
select OrderId, Quantity
from [Order Details]
where Quantity between 10 and 250

--[Si me piden el total lo correcto es]
select OrderId, Quantity,(UnitPrice*Quantity) as total
from [Order Details]
where UnitPrice * Quantity between 10 and 250
```

11. INSERT

Esta sentencia nos permite insertar nuevos registros.
Sintaxis para llenar algunos campos:

Insert into Nombre_Tabla(campo1,campo2,campo3,..)
Values (valor1, valor2, valor3,…)

Sintaxis para llenar todos campos:

Insert into Nombre_Tabla
Values (valor1, valor2, valor3,…)

Ejemplo 1: Se creó la tabla alumno, con los siguientes campos: código, nombre, apellido.

```
SELECT *
FROM ALUMNO

insert into ALUMNO
values ('A0001','Alex','Vasquez')


INSERT INTO ALUMNO (CODIGO, NOMBRES, APELLIDOS)
VALUES('A0002','IRIS','SALAZAR')

INSERT INTO ALUMNO (NOMBRES,CODIGO,APELLIDOS)
VALUES('PAUL','A0003','RODAS')
```

Ejemplo 2: Crear una tabla artículos, créala con identidad (es decir el código se genera solo a partir de una determinada cantidad y no acepta valores nulos):

```
SELECT *
FROM ARTICULOS

INSERT INTO ARTICULOS (NOMBRE,PRECIO,STOCK)
VALUES('LAPTOP HP',1200,10)

INSERT INTO ARTICULOS (NOMBRE,PRECIO,STOCK)
VALUES('MOUSE',12,100)
```

12. UPDATE

Modifica o actualiza los valores de una o más campos seleccionados.

Ejemplo 1:

```
update Products
set UnitsInStock=50
where ProductID=1
```

SET: Permite asignar el valor al campo

Ejemplo 2: Cambiar el nombre del país de UK por el de Perú. Suppliers and Customers (cambiado)

```
update Customers
set Country='Peru'
where Country='UK'

update Suppliers
set Country='Peru'
where Country='UK'
```

Ejemplo 3: Cambiar el nombre de la categoría 5 por juguetes

```
select CategoryName from Categories
where CategoryID=5

update Categories
set CategoryName='juguetes'
where CategoryId=5
```

Ejemplo 4: Cambiar y colocar el número de fax 'o' a los que son nulos

```
select fax from Suppliers
where Fax is null

update Suppliers
set fax='0'
where Fax is null
```

Ejemplo 5: Cambiar el nombre y apellido del empleado de código 6, con su nombre y apellido.

```
select lastName,FirstName from Employees
where EmployeeID=6

update Employees
set LastName='Vasquez',FirstName='Alex'
where EmployeeID=6
```

Observación: convertir a cascada hacer un nuevo diagrama solo lo que necesitas marcar la relación, vas a especificación de **insert y update** (**insert** no existe debería ser **delete** eso por la traducción) se cambia a cascada el **update** y luego dices sí, sí. Para evitar algunos errores ya que están relacionados.

Ejemplo 6: Cambiar el código del cliente 'ALFIKI' por el codigo 'UNCFI'.

```
select * from Customers
where CustomerID='ALFKI'

update Customers
set CustomerID='UNCFI'
where CustomerID='ALFKI'
```

13. DELETE

Esta sentencia sirve para eliminar los campos de una tabla específica.

Ejemplo 1:

```
delete
from Customers
```

Ejemplo 2: Eliminar la orden de compra 10255:

```
delete
from [Order Details]
where OrderID=10255

delete
from Orders
where OrderID=10255
```

Ejemplo 3: Eliminar todos los productos discontinuados:

```
delete
from [Order Details]
where ProductID in (5,9,17,24,28,29,42,53)

delete
from Products
where Discontinued=1
```

Otra forma:

```
delete
```

```
from [Order Details]
where ProductID in (select *
                    from dbo.Products
                    where Discontinued=1)


delete
from Products
where Discontinued=1
```

Observación: Recordar que si queremos eliminar un campo que esté relacionado, preferentemente pasarlo a una relación en cascada; para tratar de evitar errores al eliminar los campos.

14. DISTINCT

Sirve para que no se repitan los campos, es decir, omite los registros cuyos campos seleccionados coincidan totalmente

15. INNER JOIN

Vincula o combina varias tablas, siempre y cuando posean un campo en común. Pero para que se pueda mostrar la combinación se utiliza **ON**

Sintaxis:

```
Select * from nombre_tabla_1
Inner join nombre_tabla_2
On campo.nombre_tabla_1=campo.nombre_tabla_2
```

Pero para la combinación de tablas se puede hacer de dos formas una es con alias y otra sin ella.

Ejemplos sin alias

```
SELECT CompanyName, ProductName,UnitPrice
from Suppliers inner join Products
on Suppliers.SupplierID=Products.SupplierID
```

Ejemplos con alias

```
SELECT CompanyName, ProductName,UnitPrice
from Suppliers as b inner join Products as a
on b.SupplierID=a.SupplierID
```

En este tipo de combinaciones tenemos que mencionar a que tabla pertenece el campo que requerimos, para evitar la ambigüedad.

Ejemplo generando ambigüedad.

```
SELECT CompanyName, ProductName,UnitPrice, SupplierID
from Suppliers as b inner join Products as a
on b.SupplierID=a.SupplierID
```

En este caso **SupplierID** está generando la ambigüedad, ya que no sabemos a qué tabal pertenece.

Ejemplo evitando la ambigüedad.

```
SELECT CompanyName, ProductName,UnitPrice, b.SupplierID
from Suppliers as b inner join Products as a
on b.SupplierID=a.SupplierID
```

Ejercicio 1: El codigo de la orden de compra, la fecha de la orden de compra, el codigo del producto,el nombre del producto y la cantidad pedida

```
SELECT od.OrderId, OrderDate,p.ProductID,ProductName,Quantity
from Orders as o
inner join [Order Details] as od on o.OrderID=od.OrderID
inner join Products as p on od.ProductID=p.ProductID
```

Ejercicio 2: Mostrar: código de la categoría, el nombre de la categoría, cod. Producto, nombre del producto y precio.

```
select c.CategoryID, c.CategoryName, p.ProductID, p.ProductName, p.UnitPrice
from Products as p inner join Categories as c on p.CategoryID = c.CategoryID
```

Ejercicio 3: Mostrar: número de la orden, fecha de la orden, código del producto, cantidad, precio y flete de la orden

```
select o.OrderID, o.OrderDate, p.ProductID,
    p.QuantityPerUnit, p.UnitPrice, o.Freight
from Orders as o
inner join [Order Details] as od on o.OrderID = od.OrderID
inner join  Products as p on od.ProductID = p.ProductID
```

Ejercicio 4: Mostrar: código, nombre, ciudad y país de proveedor, código, nombre, precio, stock del producto

```
select s.SupplierID, s.CompanyName, s.City,
    s.Country, p.ProductID, p.ProductName,
    p.UnitPrice, p.UnitsInStock
```

```
from Suppliers as s
inner join Products as p on s.SupplierID = p.SupplierID
```

Ejercicio 5: Mostrar: código y nombre de la categoría, código, nombre, precio y stock de los productos, código, nombre de los proveedores

```
select c.CategoryID, c.CategoryName, p.ProductID,
    p.ProductName, p.UnitPrice, p.UnitsInStock,
    s.SupplierID, s.CompanyName
from Categories as c
inner join Products as p on c.CategoryID = p.CategoryID
inner join Suppliers as s on s.SupplierID = p.SupplierID
```

Ejercicio 6: Mostrar: núm. de la orden, fecha, nombre del producto, nombre de la categoría, nombre del proveedor

```
select o.OrderID, o.OrderDate, p.ProductName, c.CategoryName,
    s.CompanyName
from dbo.Orders as o
inner join [Order Details] as od on o.OrderID = od.OrderID
inner join dbo.Products as p on od.ProductID = p.ProductID
inner join Categories as c on p.CategoryID = c.CategoryID
inner join Suppliers as s on p.SupplierID = s.SupplierID
```

Ejercicio 7: Mostrar: núm. de la orden, fecha, nombre y dirección del cliente, nombre y apellidos del empleado. Nombre del producto comprado y nombre del proveedor

```
select o.OrderID, o.OrderDate, c.CompanyName, c.[Address],
    e.LastName, e.FirstName
from dbo.Orders as o
inner join dbo.Employees as e on o.EmployeeID = e.EmployeeID
inner join dbo.Customers as c on o.CustomerID = c.CustomerID
```

Ejercicio 8: Modificar el ejercicio 2: solo de los productos de la categorías 2, 4, 5, 7

```
select c.CategoryID, c.CategoryName, p.ProductID, p.ProductName,
    p.UnitPrice
from Products as p
inner join Categories as c on p.CategoryID = c.CategoryID
where c.CategoryID in (2, 4, 5, 7)
```

Ejercicio 9: Modificar el ejercicio 3 solo las órdenes del mes de enero de 1997

```
select o.OrderID, o.OrderDate, p.ProductID,
      p.QuantityPerUnit, p.UnitPrice, o.Freight
from Orders as o
inner join [Order Details] as od on o.OrderID = od.OrderID
inner join  Products as p on od.ProductID = p.ProductID
where year (o.OrderDate) = '1997'
```

Ejercicio 10: Modificar el ejercicio 4 solo las productos con stock cero

```
select s.SupplierID, s.CompanyName, s.City, s.Country,
     p.ProductID, p.ProductName, p.UnitPrice, p.UnitsInStock
from Suppliers as s
inner join Products as p on s.SupplierID = p.SupplierID
where p.UnitsInStock = '0'
```

Ejercicio 11: Modificar el ejercicio 5 solo con precios entre 50 y 100

```
select c.CategoryID, c.CategoryName, p.ProductID,
     p.ProductName, p.UnitPrice, p.UnitsInStock,
     s.SupplierID, s.CompanyName
from Categories as c
inner join Products as p on c.CategoryID = p.CategoryID
inner join Suppliers as s on s.SupplierID = p.SupplierID
where p.UnitPrice between 50 and 100
```

Ejercicio 12: Modificar el ejercicio 6 solo del primer trimestre del año 1996

```
select o.OrderID, o.OrderDate, p.ProductName, c.CategoryName,
     s.CompanyName
from dbo.Orders as o
inner join [Order Details] as od on o.OrderID = od.OrderID
inner join dbo.Products as p on od.ProductID = p.ProductID
inner join Categories as c on p.CategoryID = c.CategoryID
inner join Suppliers as s on p.SupplierID = s.SupplierID
where (month (o.OrderDate) between 1 and 3) and
    (YEAR (o.OrderDate) = '1996')
```

16. UNION

Esta sentencia sirve para unir varias consultas, pero para esto debe cumplir:

- Deben ser del mismo tipo y campo
- En la sentencia **SELECT** tiene que ser el mismo número.

Ejemplo:

```
select (FirstName+' '+LastName) as name, City, PostalCode
from Employees
union
select CompanyName, City, PostalCode
from Customers
```

17. SELECT INTO

Esta sentencia nos permite crear una nueva tabla a partir de un conjunto de resultados.

Ejemplo:

```
select ProductName as Products,
    UnitPrice as Price,
    (UnitPrice*1.1)as tax
into #pricetable
     -- el # lo oculta => into pricetable –- no lo oculta
from Products
```

18. SUBCONSULTAS

Una subconsulta es una sentencia **SELECT** que aparece dentro de otra sentencia **SELECT**, que llamaremos consulta principal.

Una subconsulta tiene la misma sintaxis que una sentencia **SELECT** normal exceptuando que aparece encerrada entre paréntesis, no puede contener la cláusula **ORDER BY**, ni puede ser la **UNION** de varias sentencias **SELECT**, además tiene algunas restricciones en cuando a número de columnas según lugar donde aparece en la consulta principal. Se aconseja no utilizar campos calculados en las subconsultas.

Ejemplo 1: Eliminar los productos discontinuados

```
delete from [Order Details]
where ProductID in (select ProductID from Products where Discontinued=1)
delete from Products where Discontinued=1
```

Ejemplo 2: Mostrar los productos cuyo precio es mayor al promedio de todos los productos

```
select* from Products
where UnitPrice>(select AVG(UnitPrice)from Products)
```

19. GROUP BY

Esta sentencia agrupa todos los registros iguales en uno solo y los únicos campos que existen son los que sobreviven y van en el **SELECT**; también pueden sumar, sacar promedio, desviación estándar, mínimo, máximo.

Ejemplos:

```
SELECT sum(Quantity)  - - SUM = suma todos los iguales
FROM [Order Details]
GROUP BY Quantity
```

Función de agregado	Descripción
AVG	Promedio de valores en una expresión numérica
COUNT	Número de valores en una expresión
COUNT (*)	Número de filas seleccionadas
MAX	Valor más alto en la expresión
MIN	Valor más bajo en la expresión
SUM	Valores totales en una expresión numérica
STDEV	Desviación estadística de todos los valores
STDEVP	Desviación estadística para la población
VAR	Varianza estadística de todos los valores
VARP	Varianza estadística de todos los valores para la población

Count (*) = cuenta todos los registros
Count (campo) = cuenta todos los not NULL
HAVING: va al resultado después de agrupar
WHERE: va a la tabla original

Ejercicios

Ejercicio 1: Visualizar el máximo y el mínimo precio de los productos por categoría, mostrar el nombre de la categoría.

```
select c.CategoryID, c.CategoryName, max (p.UnitPrice) as maximo,
     MIN (p.UnitPrice) as minimo
from dbo.Categories as c
inner join dbo.Products as p on c.CategoryID = p.CategoryID
group by  c.CategoryID, c.CategoryName
```

Ejercicio 2: Visualizar el máximo y mínimo precio de los productos por proveedor, mostrar el nombre de la compañía proveedora.

```
select s.SupplierID, s.CompanyName, max (p.UnitPrice) as maximo,
     MIN (p.UnitPrice) as minimo
from Suppliers as s
inner join dbo.Products as p on s.SupplierID = p.SupplierID
group by  s.SupplierID, s.CompanyName
```

Ejercicio 3: Seleccionar las categorías que tengan más de 5 productos. Mostrar el nombre de la categoría y el número de productos.

```
select c.CategoryID, c.CategoryName, COUNT (p.QuantityPerUnit)
from dbo.Categories as c
inner join dbo.Products as p on c.CategoryID = p.CategoryID
group by c.CategoryID, c.CategoryName
having COUNT (p.QuantityPerUnit) > '5'
```

Ejercicio 4: Calcular cuántos clientes existe en cada país.

```
select Country, COUNT (Country)as Cantidad
from dbo.Customers
group by Country
```

Ejercicio 5: Calcular cuántos clientes existen en cada ciudad.

```
select City, COUNT (City) as Total
from dbo.Customers
group by City
```

Ejercicio 6: Calcular cuántos proveedores existen en cada ciudad y país.

```
select City, Country, COUNT (City) as [cant_client ciudad],
     COUNT (Country) as [cant_client pais]
from dbo.Customers
```

```
group by City, Country
```

Ejercicio 7: Calcular el stock total de los productos por cada categoría. Mostrar el nombre de la categoría y el stock por categoría.

```
select c.CategoryName, COUNT (p.UnitsInStock) as cant_categoria
from dbo.Categories as c
inner join dbo.Products as p on c.CategoryID = p.CategoryID
group by CategoryName
```

Ejercicio 8: Calcular el stock total de los productos por cada categoría. Mostrar el nombre de la categoría y el stock por categoría. Solamente las categorías 2, 5 y 8.

```
select  c.CategoryName, COUNT (p.UnitsInStock) as cant_categoria
from dbo.Categories as c
inner join dbo.Products as p on c.CategoryID = p.CategoryID
where c.CategoryID in (2, 5, 8)
group by c.CategoryName
```

Ejercicio 9: Obtener el nombre del cliente, nombre del proveedor, nombre del empleado y el nombre de los productos que están en la orden 10250.

```
select c.CompanyName, s.CompanyName,
     (e.LastName + ' ' + e.FirstName) as empleado,
     p.ProductID, o.OrderID
from Customers as c
inner join Orders as o on c.CustomerID = o.CustomerID
inner join Employees as e on o.EmployeeID = e.EmployeeID
inner join [Order Details] as od on o.OrderID = od.OrderID
inner join Products as p on od.ProductID = p.ProductID
inner join Suppliers as s on p.SupplierID = s.SupplierID
where o.OrderID = 10250
```

Ejercicio 10: Mostrar el número de órdenes realizadas de cada uno de los clientes por año.

```
select CompanyName, YEAR (OrderDate) as año,
     COUNT (CompanyName) as total
from Orders as o
inner join Customers as c on o.CustomerID = c.CustomerID
group by CompanyName, YEAR (OrderDate)
```

Ejercicio 11: Mostrar el número de órdenes realizadas de cada uno de los empleados en cada año.

```
select (e.LastName + ' ' + e.FirstName) as empleado,
    YEAR (OrderDate) as año, COUNT (e.EmployeeID) as total
from Orders as o
inner join Employees as e on o.EmployeeID = e.EmployeeID
group by (e.LastName + ' ' + e.FirstName), YEAR (OrderDate)
```

Ejercicio 12: Mostrar el número de órdenes realizadas de cada uno de los clientes por cada mes y año.

```
select CompanyName, MONTH (OrderDate) as mes,
    YEAR (OrderDate) as año, COUNT (CompanyName) as total
from Orders as o
inner join Customers as c on o.CustomerID = c.CustomerID
group by CompanyName, MONTH (OrderDate), YEAR (OrderDate)
```

Ejercicio 13: Contar el número de órdenes que se han realizado por años y meses.

```
select MONTH (OrderDate) as mes, YEAR (OrderDate) as año,
    COUNT (OrderID) as cant_orden
from Orders
group by  MONTH (OrderDate), YEAR (OrderDate)
```

PROCEDIMIENTOS ALMACENADOS

Para crear un procedimiento iniciara con **CREATE** luego ira **PROC** o **PROCEDUTE** y al final darle el nombre del procedimiento.
Sintaxis:

```
create proc hol
```

Si se desea modifica se reemplazara **CREATE** por **ALTER**, este último nos permitirá guardar las modificación en cuerpo del procedimiento.
Sintaxis:

```
alter proc hol
```

Luego si va o no los parámetros.

Parámetros: es una variable que establece el ámbito de los datos específicos devueltos cuando se ejecuta un informe. Puede proporcionar un valor predeterminado, o la persona que ejecuta el informe puede seleccionar un valor o un conjunto de valores. Para agregar un parámetro al informe.
Primero va el signo arroba (@), luego el nombre del parámetro y luego el tipo.
Sintaxis:

```
@hola int
```

Enseguida va la cláusula AS.

AS: La cláusula **AS** se puede usar para cambiar el nombre de una columna de conjunto de resultados o para asignar un nombre a una columna derivada.
Cuando se define una columna del conjunto de resultados mediante una referencia a la columna de una tabla o vista, el nombre de la columna del conjunto de resultados es el mismo que el nombre de la columna a la que se hace referencia. La cláusula **AS** se puede usar para asignar un nombre distinto, o alias, a la columna del conjunto de resultados. Esto se puede hacer para mejorar la comprensión.

```
as
```

Después si se desea se declara o no variables. Para declarar una variable se tiene que anteponer DECLARE, el signo arroba (@), nombre de la variable y el tipo.

Y por último va el cuerpo del procedimiento.
Para ejecutar el procedimiento es necesario escribir EXEC, el nombre del procedimiento y si le damos o no parámetros.
Sintaxis con parámetros:

```
exec hol 1
```

Ejemplo 1:

```
create proc hol
@hola int
as
```

```
select * from Products
where ProductID=@hola

exec hol 1 --ejecuta el procedimiento
```

Ejemplo 2: Crear un SP que ingrese registros a la tabla productos (código, nombre, precio, stock) el código se generara automáticamente.

Muestra. Código P0001, P0002, …

```
create proc produ_L
@nom varchar(50),
@pre MONEY,
@sto int
as
declare @nr as int,
    @cod as char(5)
select @nr=COUNT(*)
from productos
if @nr is null set @nr=0;
        set @nr=@nr+1;
if @nr<=9
        set @cod='P000'+ltrim(STR(@nr))
else
        If @nr<=99
                set @cod='P00'+ltrim(STR(@nr))
        else
                if @nr<=999
                        set @cod='P0'+ltrim(STR(@nr))
        else
                set @cod='P'+ltrim(STR(@nr))

insert into productos(codigo,nombre,precio,stock)
values (@cod,@nom,@pre,@sto)

EXEC PRODU_L 'azucar',1.3,202
```

	codigo	nombre	precio	stock
1	P0001	arroz	1,30	202
2	P0002	azucar	1,80	22
3	P0003	gelaina	1,10	28

Ejemplo 3: Crear una tabla CLIENTES con los siguientes campos: código (7), nombre (30), apellidos (30), dirección (30) y correo (30). Crear un SP que inserte a los clientes, y el código se generará con las dos primeras letras de su nombre y apellido, y un número correlativo.

```
create  proc clients
@n varchar (30),
@a varchar (30),
@d varchar (30),
@c varchar (30)
As
Declare @cod as char(7),@nr int
Set @cod=left(@n,2)+left(@a,2)

Select @nr=count(*)+1
From clientes
If @nr <=9
        Set @cod =rtrim(@cod)+'00'+ltrim(str(@nr))
If @nr <=99
        Set @cod =rtrim(@cod)+'0'+ltrim(str(@nr))
Else
        Set @cod =rtrim(@cod)+ltrim(str(@nr))
        Insert into clientes(codigo,nombre,apellidos,direccion,correo)
        Values (@cod,@n,@a,@d,@c)

exec clients 'davis','saman','CUCARDAS 250','SSHI@HOMAIL.COM'
```

	codigo	nombre	apellidos	direccion	correo
1	ALVA001	ALEX	VASQUEZ	CUCARDAS 250	SSHI@HOMAIL.COM
2	ANSA005	ANTHONY	SAMAN	CUCARDAS 250	SSHI@HOMAIL.COM
3	dasa002	davis	saman	CUCARDAS 250	SSHI@HOMAIL.COM
4	DAVA004	DAVID	VASQUEZ	CUCARDAS 250	SSHI@HOMAIL.COM
5	PEMA003	PETER	MARTINES	CUCARDAS 250	SSHI@HOMAIL.COM

Ejemplo 4: Crear un SP que ingrese registros a la tabla PERSONA (código, nombre, apellido, dirección, correo), el código se generara automáticamente.

Muestra.

CODIGO	NOMBRE	APELLIDOS	DIRECCION	CORREO
LUMA001	LUIS	MARTINEZ	CUEVA 89	JOL@HOMAIL.COM
LUMA002	LUIZA	MARQUEZ	CUEVA 89	JOL@HOMAIL.COM
ANMA001	ANA	MARIA	CUEVA 89	JOL@HOMAIL.COM
ANMA002	ANA	MARIÑES	CUEVA 89	JOL@HOMAIL.COM

```
create proc personita
@nom varchar(50),
@ape varchar(50),
@dir varchar(50),
@core varchar(50)
as
declare @nr as int,
```

```
    @cod as char(7)

set @cod=LEFT(@nom,2)+LEFT(@ape,2)

select @nr=COUNT(*)+1
from persona
where substring(codigo,1,4)=@cod

if @nr<=9
        set @cod=rtrim(@cod)+'00'+LTRIM(STR(@nr))
if @nr<=99
        set @cod=rtrim(@cod)+'0'+LTRIM(STR(@nr))
if @nr<=999
        set @cod=rtrim(@cod)+LTRIM(STR(@nr))

insert into persona(codigo,nombre,apellidos,direccion,correo)
values (@cod,@nom,@ape,@dir,@core)

exec personita 'lus','malvinas','cueva 58','89@hotmail.com'
```

	codigo	nombre	apellidos	direccion	correo
1	anma001	anita	marques	cueva 58	89@hotmail.com
2	anma002	ana	matines	cueva 58	89@hotmail.com
3	luma001	luir	matines	cueva 58	89@hotmail.com
4	luma002	lus	malvinas	cueva 58	89@hotmail.com

Ejemplo 4: Crear un SP, obtenga todos los productos de un proveedor

```
create proc prod_prov
        @cod int
as
        select *
        from Products
        where SupplierID= @cod

execute prod_prov 5

-- otra forma de llamar al procedimiento
exec prod_prov 5

--otra forma de llamar al procedimiento
prod_prov 5
```

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLeve
1	11	Queso Cabrales	5	4	1 kg pkg.	21,00	22	30	30
2	12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38,00	86	0	0

Ejemplo 5: Hacer un procedimiento almacenado que elimine una orden de compra

```
create proc elimina_orden
        @cod int
as
        delete from [Order Details]
        where OrderID=@cod
        delete from Orders
        where OrderID=@cod

exec elimina_orden 10570
```

Llamamos:

```
select OrderID from [Order Details]
where OrderID=10570
union select OrderID from Orders
where OrderID=10570
```

OrderID

Ejemplo 6: crear un SP para cambiar el stock de un producto

```
create proc actualizar
        @nuevo smallint,
        @id int
as
        update Products
        set UnitsInStock= @nuevo
        where ProductID= @id

        select *from Products
        where ProductID=@id

exec actualizar 100,1
```

Observación: Tiene que seguir el orden de los parámetros, caso contrario saldrá un error.

Ejemplo 7: Crear un SP cuantos productos ha sido vendido por los empleados en el año 1996

```
create proc empleados_unid
@anio int
as
        select YEAR(Orders.OrderDate) as anio,
            (Employees.LastName+' '+Employees.FirstName)as empleado,
            SUM([Order Details].Quantity) as total
        from Employees
        inner join Orders on Employees.EmployeeID=Orders.EmployeeID
        inner join [Order Details] on Orders.OrderID=[Order Details].OrderID
        where YEAR(Orders.OrderDate)=@anio
        group by Employees.LastName,Employees.firstName,
              YEAR(Orders.OrderDate)
        order by Employees.LastName,Employees.FirstName

exec empleados_unid 1996
```

	anio	empleado	total
1	1996	Buchanan Steven	743
2	1996	Callahan Laura	782
3	1996	Davolio Nancy	1440
4	1996	Dodsworth Anne	532
5	1996	Fuller Andrew	867
6	1996	King Robert	463
7	1996	Leverling Janet	867
8	1996	Peacock Margaret	1992
9	1996	Suyama Michael	868

Ejemplo 8: Productos más vendidos, en un determinado año.

```
create proc topventas
        @num int,
        @anio int
as
        select top (@num) p.ProductName,
              SUM(od.Quantity * p.UnitPrice) as monto
        from [Order Details] as od
        inner join Orders as o on od.OrderID=od.OrderID
        inner join Products as p on od.ProductID=p.ProductID
        where year(o.OrderDate)=@anio
        group by p.ProductName
        order by (SUM(od.Quantity * p.UnitPrice)) desc  --order by total desc
```

```
exec topventas 2,1996
```

	ProductName	monto
1	Côte de Blaye	24952396,00
2	Raclette Courdavault	12506560,00

Ejemplo 9: Productos no vendidos en un determinado año (como sabemos que el producto no existe en otra tabla)

```
create proc prod_novendido
        @anio int
as
        select ProductId,ProductName from Products
        where ProductID not in (select od.ProductID
                                from [Order Details] as od
                                inner join Orders as o on   o.OrderID=od.OrderID
                                where YEAR(o.OrderDate)=@anio)

exec prod_novendido 1998
```

	ProductId	ProductName
1	15	Genen Shouyu

Ejemplo 10: Buscar frase mostrar el empleado, cantidad, nombre producto

```
create proc frase
        @fr nchar(40)
as
        select e.LastName, SUM(od.Quantity)as cantidad,
                p.ProductName
        from Employees as e
        inner join Orders as o on e.EmployeeID=o.EmployeeID
        inner join [Order Details] as od on o.OrderID=od.OrderID
        inner join Products as p on od.ProductID=p.ProductID
        where p.ProductName like @fr
        group by e.LastName,p.ProductName
        order by e.LastName,p.ProductName

exec frase '%queso%'
```

	LastName	cantidad	ProductName
1	Buchanan	52	Queso Cabrales
2	Callahan	100	Queso Cabrales
3	Callahan	70	Queso Manchego La Pastora
4	Davolio	85	Queso Cabrales
5	Davolio	68	Queso Manchego La Pastora
6	Dodsworth	75	Queso Cabrales
7	Fuller	160	Queso Cabrales
8	King	32	Queso Cabrales
9	King	140	Queso Manchego La Pastora
10	Leverling	60	Queso Cabrales
11	Leverling	31	Queso Manchego La Pastora
12	Peacock	82	Queso Cabrales

Consulta ejecutada correctamente.

Ejemplo 11: Mostrar las cantidades de ordenes en un determinado año menor a los días transcurridos.

```
create proc diastrascur
        @d int,
        @anio int
as
        select  count(OrderId)as cantord,
                DATEDIFF(DAY,ShippedDate,RequiredDate)as dias_tras
        from Orders
        where (DATEDIFF (DAY,ShippedDate,RequiredDate)<@d)and
            YEAR(RequiredDate)=@anio
        group by DATEDIFF (DAY,ShippedDate,RequiredDate)

exec diastrascur 3,1996
```

	cantord	dias_tras
1	1	-6
2	1	-2
3	4	-1
4	1	2

Ejemplo 12: Generar un SP que genere un numero al azar y al comparar con los empleados, y muestre al ganador.

```
CREATE PROC GANADOR
AS
        DECLARE @AZAR INT
        DECLARE @AUX INT

        SET @AUX = (SELECT COUNT(*)
                    FROM Employees)

        SET @AZAR = cast((rand ()*@AUX) AS int)
```

```
SELECT LastName
FROM Employees
WHERE EmployeeID = @AZAR
```

RAND(): Devuelve un valor (de tipo **float)** aleatorio de 0 a 1.

Ejemplo 13: Generar un SP que muestre al ganador de los clientes.

```
CREATE PROC GANADOR_CLIENTE
AS
        SELECT TOP 1 CustomerID,ContactName,NEWID()
        FROM Customers
        ORDER BY NEWID()
```

NEWID(): Genera números hexadecimales sin repetirse

Ejemplo 14: Crear un SP que actualice los precios de los productos, solo de los productos de la categoría ingresada (ingresar el porcentaje de aumento de los precios) solamente se actualizan los productos vigentes.

```
create PROC LOP
@PC FLOAT,
@CAT INT
AS
SELECT PRODUCTNAME,UNITPRICE,CATEGORYID
FROM Products
UPDATE Products
SET UnitPrice=UnitPrice+(UnitPrice*@PC)
WHERE (CategoryID=@CAT)and
        (Discontinued=0)


EXEC LOP 1.5,1
```

Antes de ejcutar el SP

	PRODUCTNAME	UNITPRICE	CATEGORYID
1	Chai	45,00	1
2	Chang	47,50	1
3	Aniseed Syrup	10,00	2
4	Chef Anton's Cajun Seasoning	22,00	2
5	Chef Anton's Gumbo Mix	21,35	2
6	Grandma's Boysenberry Spread	25,00	2
7	Uncle Bob's Organic Dried Pears	30,00	7
8	Northwoods Cranberry Sauce	40,00	2
9	Mishi Kobe Niku	97,00	6
10	Ikura	31,00	8
11	Queso Cabrales	21,00	4
12	Queso Manchego La Pastora	38,00	4

Despues de ejecutar el SP

	PRODUCTNAME	UNITPRICE	CATEGORYID
1	Chai	112,50	1
2	Chang	118,75	1
3	Aniseed Syrup	10,00	2
4	Chef Anton's Cajun Seasoning	22,00	2
5	Chef Anton's Gumbo Mix	21,35	2
6	Grandma's Boysenberry Spread	25,00	2
7	Uncle Bob's Organic Dried Pears	30,00	7
8	Northwoods Cranberry Sauce	40,00	2
9	Mishi Kobe Niku	97,00	6
10	Ikura	31,00	8
11	Queso Cabrales	21,00	4
12	Queso Manchego La Pastora	38,00	4

Ejemplo 15: Crear un SP que cuando se elimine un registro, la numeración continúe.

```
alter proc inser_alum
        @nom varchar(50),
        @ape varchar(50),
        @Fn date,
        @g char
as
        declare @nr int,
        @COD CHAR(9)
        select @nr=MAX(RIGHT(CODIGO,3))+1
        from alumno
        IF @nr<=9
                set @cod='P0000000'+LTRIM(STR(@NR))
        ELSE
                IF @nr<=99
                        set @cod='P000000'+LTRIM(STR(@NR))
                ELSE
                        IF @nr<=999
                        set @cod='P000000'+LTRIM(STR(@NR))
        insert into alumno (CODIGO,NOMBRE,APELLIDO,FECh_NA,GENERO)
        values(@cod,@nom,@ape,@fn,@g)

EXEC inser_alum 'ANA','LOPEZ','06/05/1985','m'
```

Ejemplo 16: Crear una tabla con los siguientes campos: código (7), nombre, apellidos, género. El código se generara con los dos primeros caracteres del nombre, con los dos primeros caracteres del apellido; y el resto será completado por un contador, según el género.

Muestra:

CÓDIGO	NOMBRE	APELLIDO	GENERO
JUEPE001	JUAN	PÉREZ	M
VIRA002	VÍCTOR	RAMÍREZ	M
VISA001	VIOLETA	SÁNCHEZ	F
SAVI002	SARA	VILLAR	F

```
CREATE proc PERSONITA 'MANUEL','VILLAR','M'
@nom varchar(30),
```

```
@ape varchar(30),
@SX CHAR(1)
as
declare @nr int,
      @cod char(7)

select @nr=COUNT(*)+1
from PERSONA
WHERE genero=@SX
if @nr<=9
            set @cod=LEFT(@nom,2)+left(@ape,2)+'00'+LTRIM(STR(@nr))
else if @nr<=99
            set @cod=LEFT(@nom,2)+left(@ape,2)+'0'+LTRIM(STR(@nr))
            else
                        set @cod=LEFT(@nom,2)+left(@ape,2)+LTRIM(STR(@nr))

insert into PERSONA(codigo,nombre,apellido,genero)
values(@cod,@nom,@ape,@SX
```

Ejemplo 17: Crear una tabla con los siguientes campos: código (9), nombre, apellidos, fecha nacimiento. El código se generara con el primer carácter del nombre, del apellido, el año de nacimiento; y el resto será completado por un contador.

```
CREATE proc alumnito
@nom varchar(50),
@ape varchar(50),
@fn date
as
declare @nr int,@cod char(9)

select @nr=COUNT(*)+1
from alum
if @nr<=9
      set @cod=LEFT(@nom,1)+left(@ape,1)+LTRIM(str(YEAR(@fn)))+'00'+LTRIM(STR(@nr))
else if @nr<=99
                        set @cod=LEFT(@nom,1)+left(@ape,1)+LTRIM(str(YEAR(@fn)))+'0'+LTRIM(STR(@nr))
            else
                        set @cod=LEFT(@nom,1)+left(@ape,1)+LTRIM(str(YEAR(@fn)))+LTRIM(STR(@nr))

insert into alum(codigo,nombre,apellido,fnacimiento)
values(@cod,@nom,@ape,@fn)

exec alumnito 'ALEX','VASQUEZ','06/08/1985'
```

	CODIGO	NOMBRE	APELLIDO	FNACIMIENTO
1	AV1985002	ALEX	VASQUEZ	1985-08-06
2	DS1987001	DAVID	SAMAN	1987-08-06

Ejemplo 18: Crear un Stored Procedure que muestre los productos de la misma categoría que tiene el producto más caro (ir modificando el producto más caro para realizar las pruebas)

```
Alter proc pcaro
as
select * from Products
WHERE CategoryID =(select top 1 CategoryID
            From Products
            Orderby UnitPrice desc)
Orderby UnitPrice desc

Select * from Products

--Pruebas
Update Products
Set UnitPrice=9642
Where ProductName='Tofu'

Update Products
Set UnitPrice=10000
Where ProductName='Geitost
```

Ejemplo 19: Crear un Stored Procedure que muestre el número de la orden, el año de la orden, el total de productos comprados en esa orden, el monto total pagado por la orden. Solo mostrar las ordenes según el año ingresado al Sored Procedure.

```
Create proc ordenar
@año int
as
select o.OrderID,year(o.OrderDate)as año,
     sum(od.Quantity)as tproductos,
     sum(od.UnitPrice*od.Quantity*2.81)as monto
from Products as p
innerjoin [Order Details]as od on p.ProductID=od.ProductID
innerjoin Orders as o on od.OrderID=o.OrderID
where year(OrderDate)=@año
group by o.OrderID,YEAR(o.OrderDate)

exec ordenar 1998
```

Ejemplo 20: Crear un Stored Procedure que seleccione aleatoriamente a 10 clientes diferentes, los 10 clientes seleccionados se almacenaran temporalmente en la tabla GANADORES hasta que se ejecute nuevamente el SP. Los campos que se almacenaran en la tabla ganadores son: código del cliente, nombre de la compañía, nombre del contacto, dirección, ciudad y país del cliente.

```
Createproc gane
as
select top 10 CustomerID as codigo, CompanyName as compañia,   ContactName as contacto,Address
as direccion, City as ciudad, Country as pais
into ganador
```

```
from Customers
orderby NEWID()
select *from ganador
drop table ganador
```

Ejemplo 21: Crear un procedimiento almacenado que copie los registros de productos a la tabla ProductosNuevos (código, nombre, precio, stock) colocando el stock a cero a todos los productos y el precio actualizarlo según:

Menor o igual a 10	-->	200%
Mayor que 10 y menor o igual a 23	-->	100%
Mayor a 23 y menor o igual a 110	-->	80%
Mayor a 110	-->	70%

```
Alterproc nuevo
as
select ProductID as codigo,
ProductName as nombre,
UnitPrice as precio,
UnitsInStock as stock
Into productonuevo
From Products

Update productonuevo
Set stock=0

Update productonuevo
Set precio=precio+precio*0.7
Where precio>110

Update productonuevo
Set precio=precio+precio*0.8
Where precio>23 and precio<=110

Update productonuevo
Set precio=precio+precio
Where precio>10 and precio<=23

Update productonuevo
Set precio=precio+precio*2
Where precio<=10

Select * from productonuevo
drop table productonuevo

-- Pruebas
Select ProductID,ProductName,UnitPrice,UnitsInStock
From Products
Update Products
Set UnitPrice=20
Where ProductName='Chang'
```

Ejemplo 22: Crear una tabla personas, con los campos: DNI, nombres, apellidos, sexo. Ingresar por lo menos 10 registros.
Implementar un Stored Procedure que permita seleccionar una pareja de personas, conformada por una mujer y un hombre de manera aleatoria.

```
create proc Paleatorio
as
select * from (select top 1 DNI, Nombre, Apellido , Sexo
                    from Persona
                    where Sexo = 'M'
                    order by NEWID()) as tabla1
UNION ALL
select * from (select top 1 DNI, Nombre, Apellido , Sexo
          from Persona
          where Sexo = 'F'
          order by NEWID()) as tabla2
```

Ejemplo 23: Crear un Stored Procedure que nos permita insertar un registro a la tabla postulantes (código, nombres, apellidos, fecha de nacimiento, sexo), internamente se debe autogenerar el código para posteriormente ingresar el registro completo, el código se genera de la siguiente forma:

4 caracteres del año	1 carácter del sexo (F: femenino M: masculino)	3 números correlativo en función del sexo y año	1 carácter nombre

```
Create proc ingreso
@nom varchar(30),
@ape varchar(30),
@fn date,
@sx char
as
declare @nr int, @cod char(9)
select @nr = COUNT(*)+1
from postulante
where YEAR(@fn)=YEAR(fnacimieno) and @sx=sexo
if @nr<9
set @cod=ltrim(str(YEAR(@fn)))+LTRIM(@sx)+'00'+ltrim(str(@nr))+LEFT(@nom,1)
else
if @nr<99
set @cod=ltrim(str(YEAR(@fn)))+LTRIM(@sx)+'0'+ltrim(str(@nr))+LEFT(@nom,1)
else
set
@cod=ltrim(str(YEAR(@fn)))+LTRIM(@sx)+ltrim(str(@nr))+LEFT(@nom,1)

Insert into postulante(codigo,nombre,apellido,fnacimieno,sexo)
values (@cod,@nom,@ape,@fn,@sx)

exec ingreso 'JUAN','VELASQUEZ','1988/05/12','M'

SELECT * FROM POSTULANTE
```

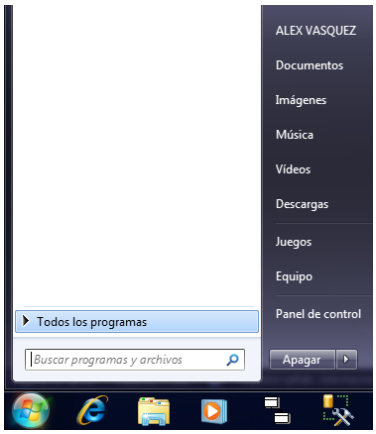
REPORTING SERVICES

Para crear un reporte vamos utilizar una herramienta del SQL Server que es el Visual Studio 2008.

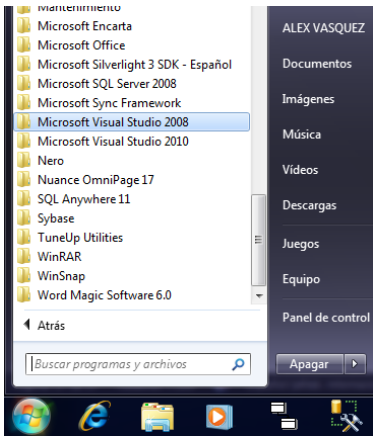
Iniciar Visual Studio 2008

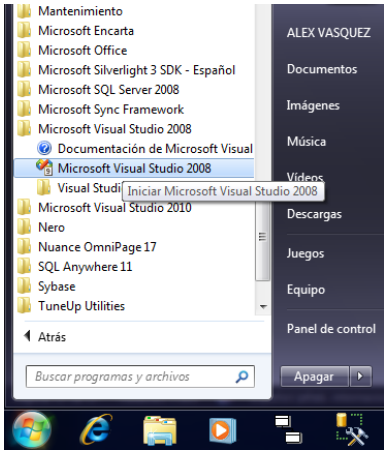
1. Clic en el botón inicio

2. Todos los programas

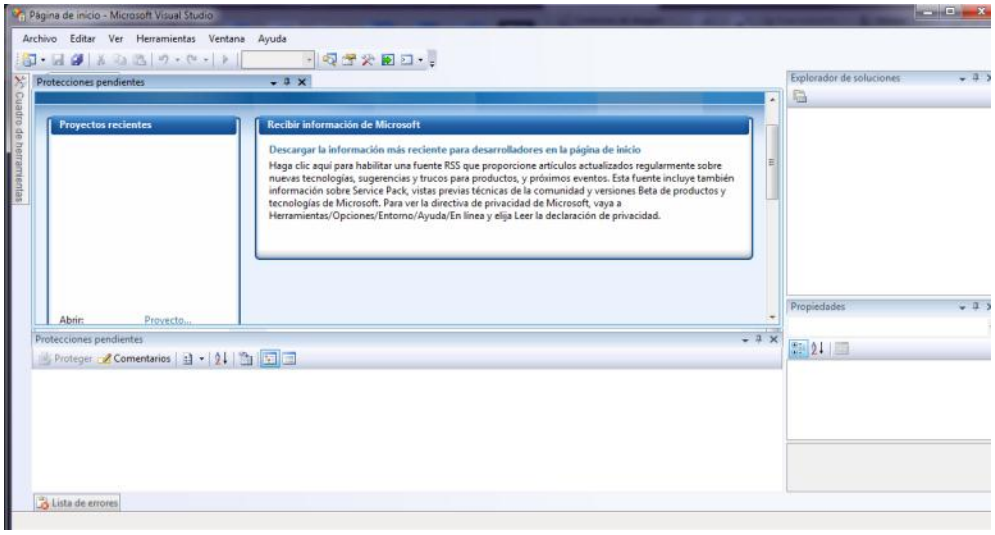


3. Microsoft Visual Studio 2008

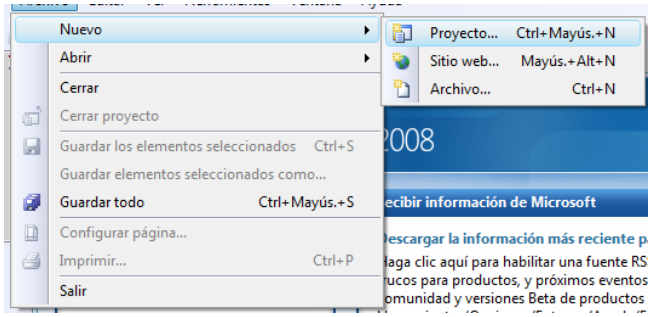




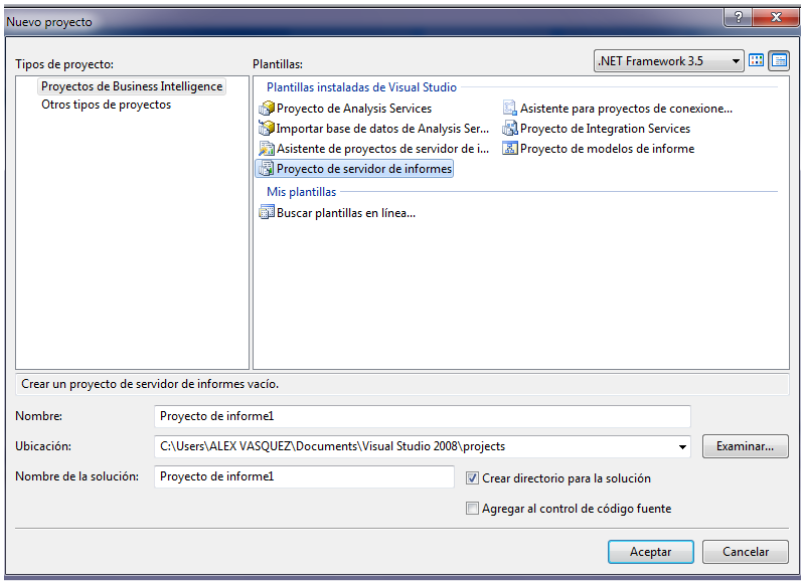
4. Iniciando Visual Studio 2008



5. Creando un primer informe
   5.1. Clic en el menú **archivo > Nuevo > Proyecto**

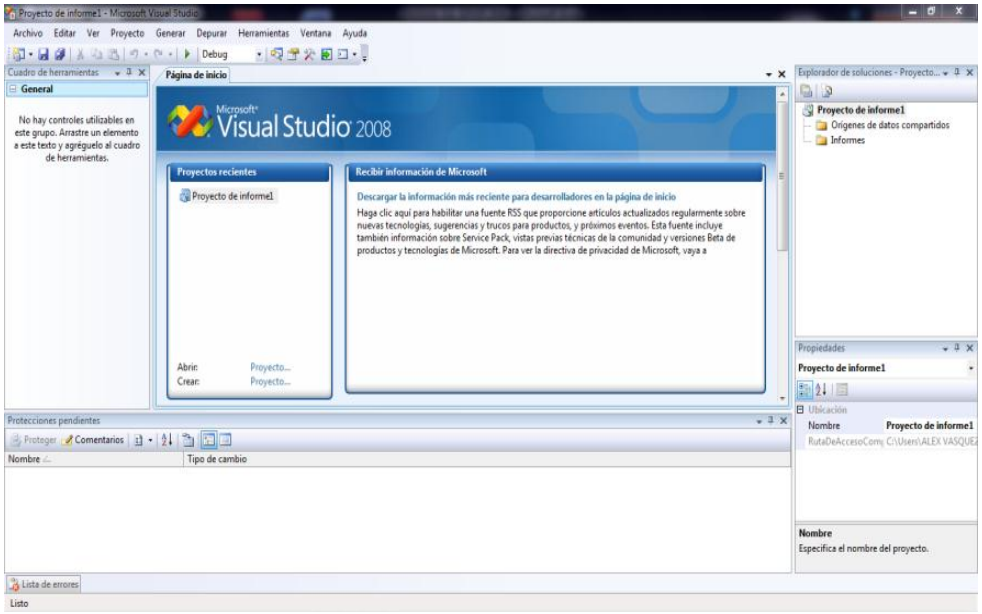


   5.2. Se abre una ventana

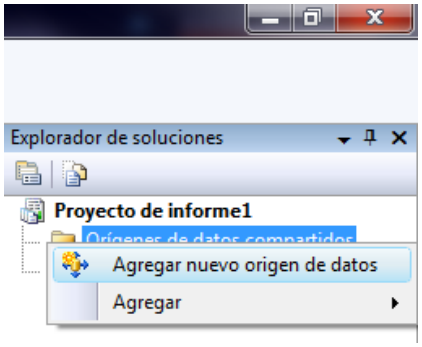


- En la ventana seleccionamos **Proyecto de servidor de informes**.
- En la parte inferior de la ventana aparece la opción **Nombre**, en el cual podrá cambiar el nombre por el que guste; es recomendable que le asigne el nombre según el trabajo que vaya haciendo.
- En la opción **Ubicación**, es la ruta donde se va a guardar su aplicación. Si desea cambiarla haga clic en **examinar** y busque donde quiere guardarla
- **Nombre de la solución**, debe coincidir con el nombre que se le asigne

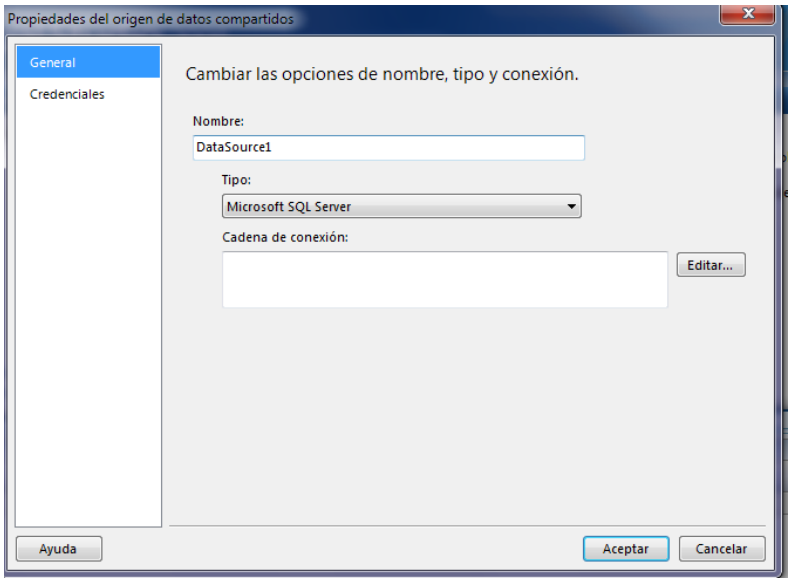
Una vez cambiada las opciones hacer clic en **aceptar.**



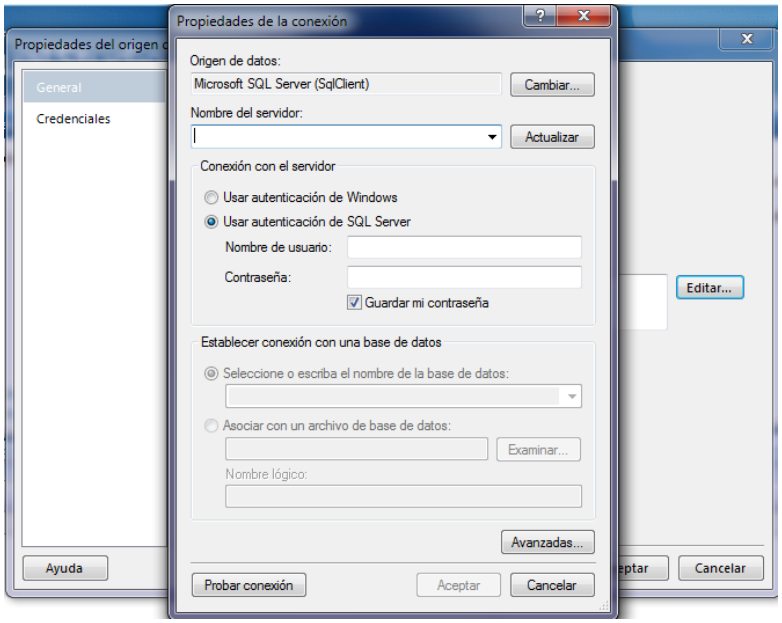
5.3. Hacer un primer informe. En la parte de la derecha en el explorador de soluciones hacemos clic derecho en **orígenes de datos compartidos > Agregar Nuevo origen de datos**



5.4. La ventana que aparece selecciona **General**

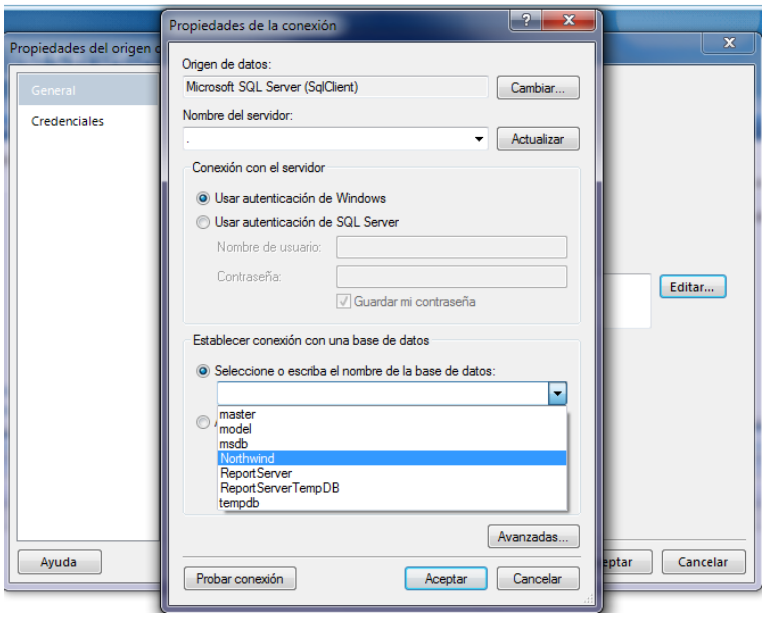


En la opción **nombre** le asigna un nombre cualesquiera, luego hacemos clic en la opción **editar**.



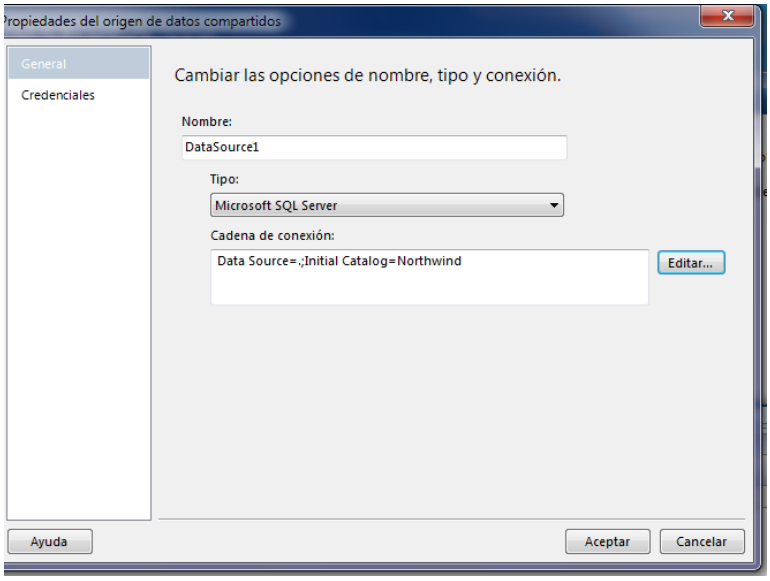
Donde está la opción **Nombre del servidor** buscara el servidor que esté trabajando, en nuestro caso estamos con Autentificación de Windows y allí vamos a colocar el punto (.)

Si está trabajando con una **Autenticación de SQL server** colocara el nombre del *usuario* y su *contraseña* respectiva.

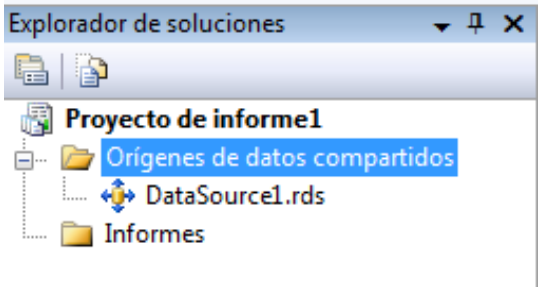


Después de colocar el punto (**.**), se activa la opción **Seleccione o escriba el nombre de la base de datos**, vamos elegir la base de datos que estamos trabajando, en este caso es **Northwind**.

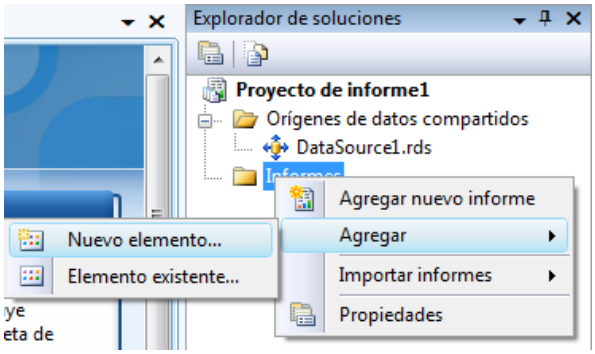
Y damos clic en **Aceptar**.



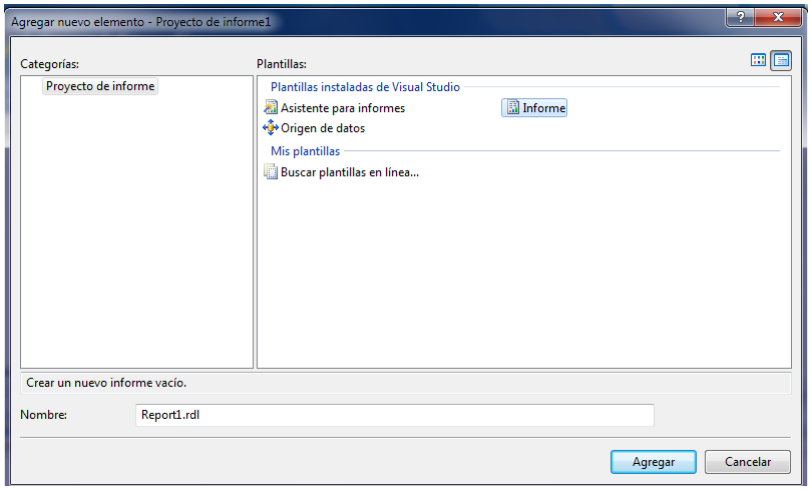
Aceptar

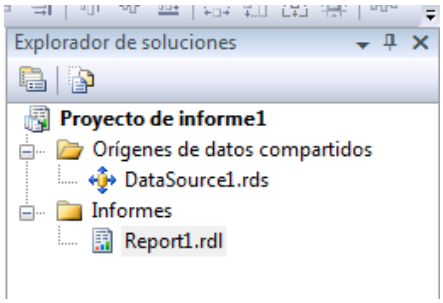


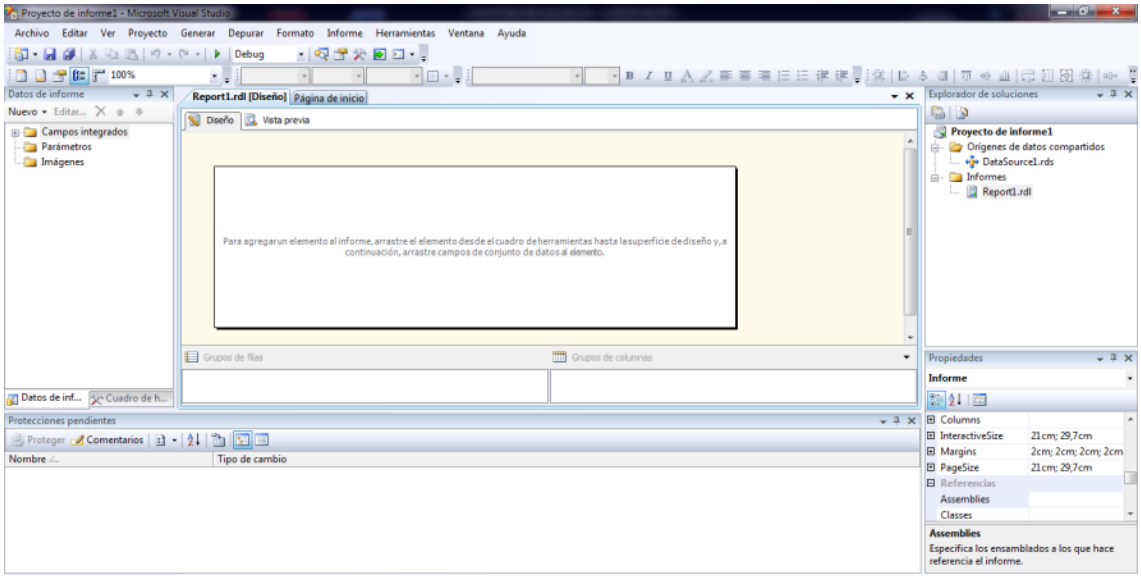
5.5. Creamos el primer informe
Clic derecho **Informes > Agregar > Nuevo elemento**



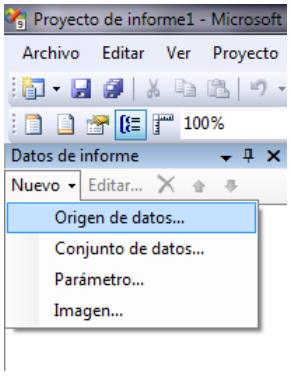
Seleccionamos **informe** y si queremos cambiamos el **nombre** que aparece por defecto y luego clic en **Agregar**

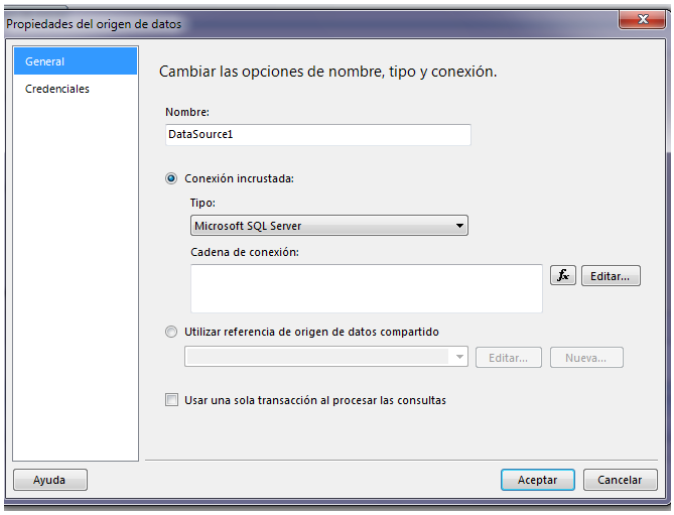




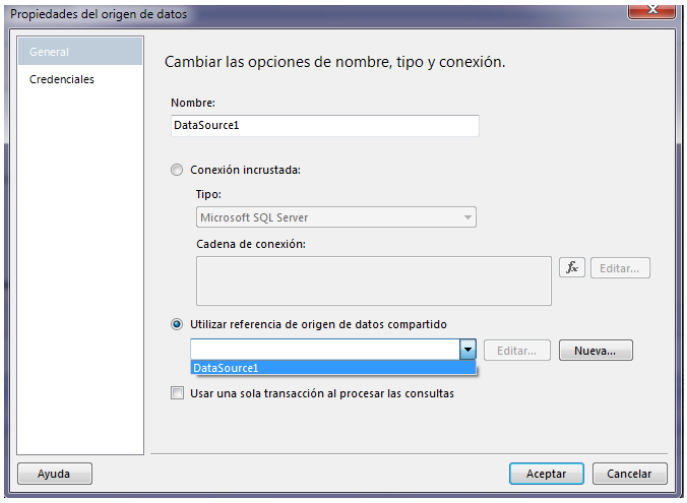


5.6. Luego en la parte izquierda **Datos de Informe** hacemos clic **Nuevo > Origen de Datos**

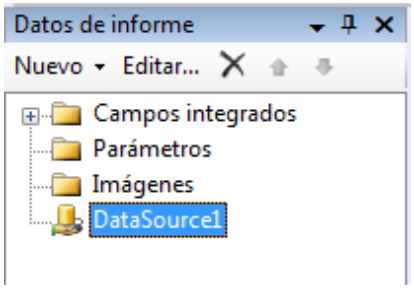




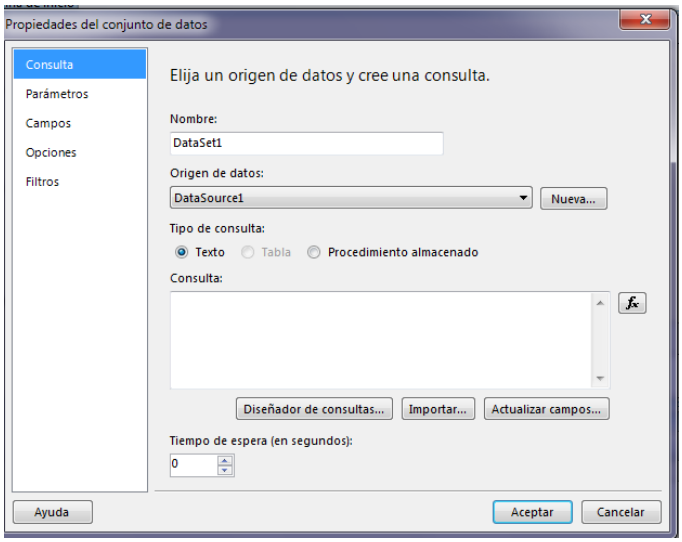
Pero como ya hemos creado el origen de datos seleccionamos la opción Utilizar referencia de **origen de datos compartidos** > Seleccionamos el que ya hemos creado (en este caso **DataSource1**).



Aceptar

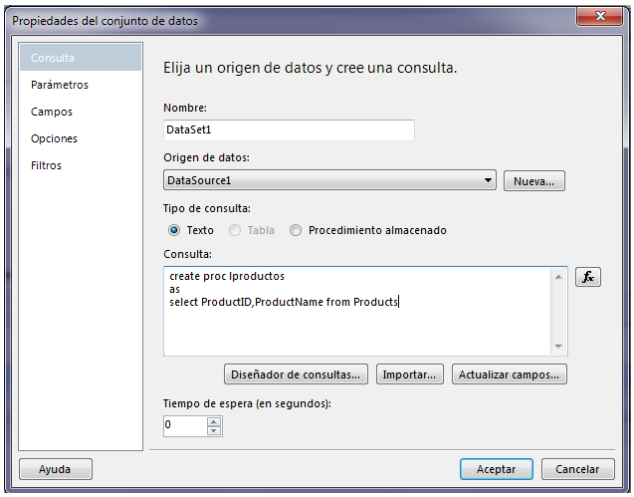


Seleccionamos y hacemos clic derecho **DataSource1** (de acuerdo al nombre que le hayas asignado) > **Agregar conjuntos de datos**.



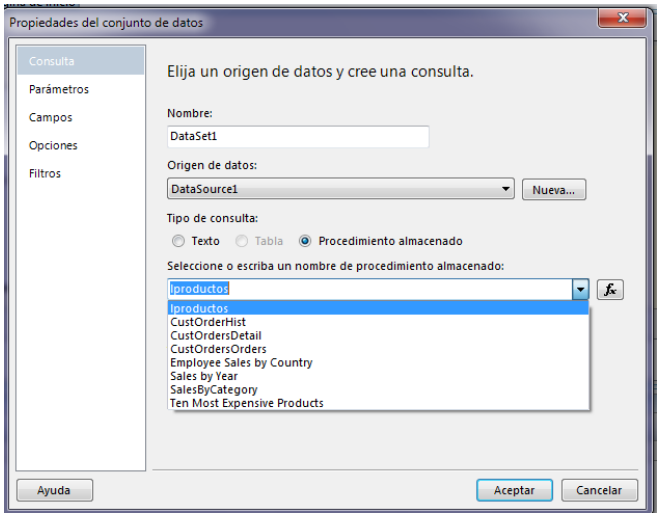
Podemos asignarle un **nombre**, elegimos el **origen de datos** (**DataSource1**) que hemos creado y en la consulta hay dos opciones:

a. **Texto**: Es cuando escribes directamente la consulta.

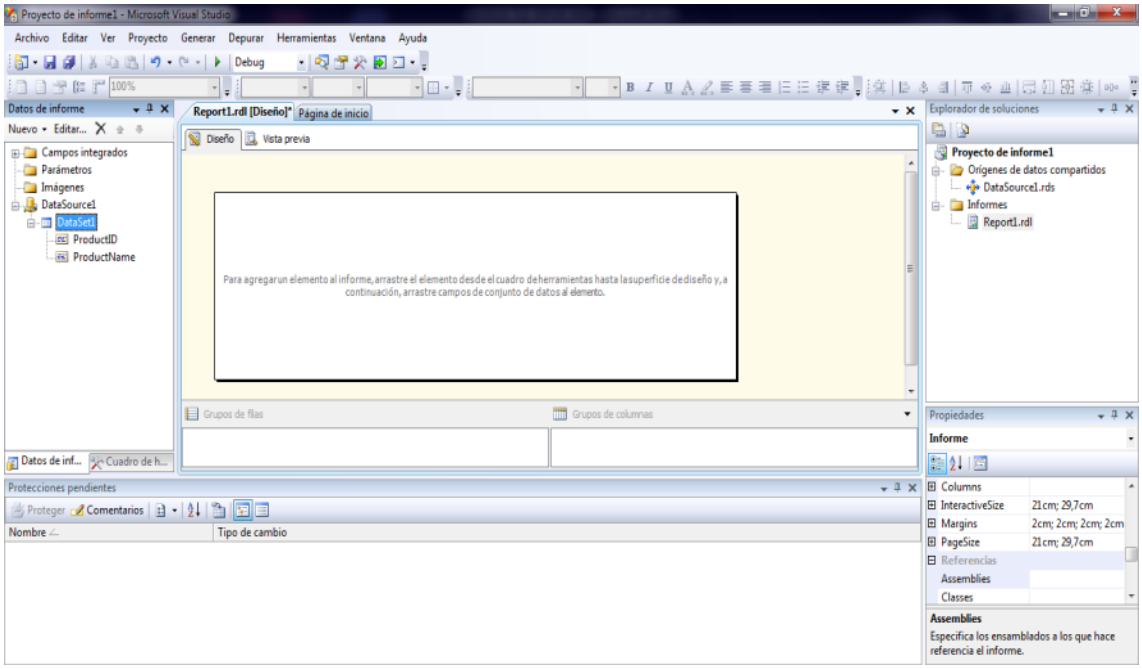


b. **Procedimiento Almacenado**: Es cuando ya tengas un procedimiento creado en el SQL Server, lo que hace esta opción es jalar el procedimiento del SQL Server.

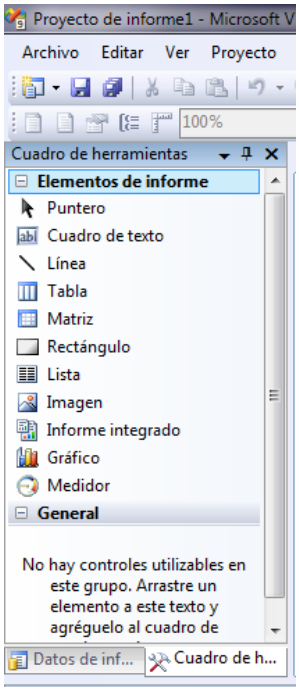
```
create proc lproductos
as
select ProductID,ProductName from Products
```



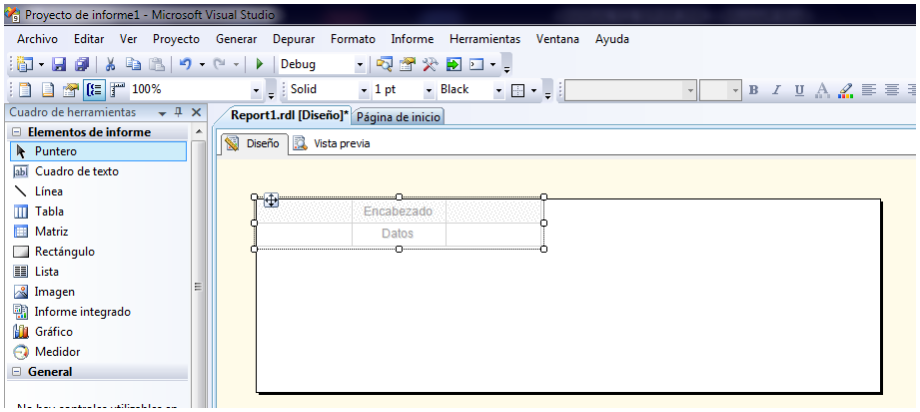
Buscamos el nombre del procedimiento que se ha creado en el SQL Server. Luego Aceptamos



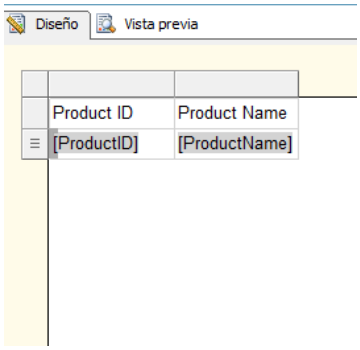
5.7. Creamos la tabla del informe
En la parte izquierda seleccionamos el cuadro de herramientas



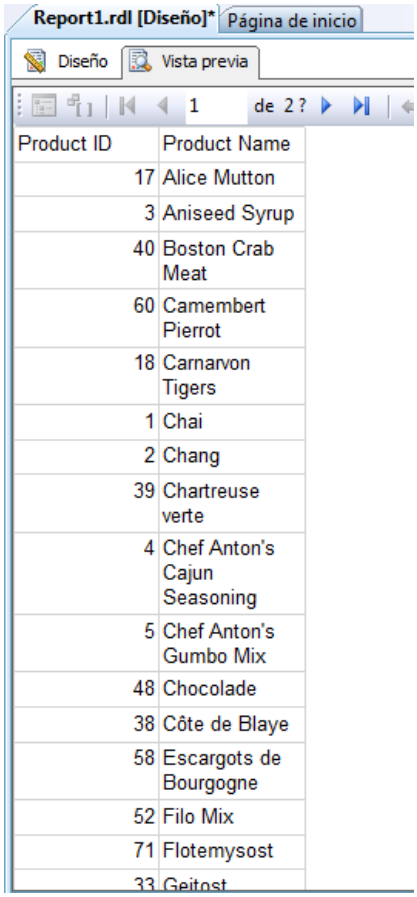
Luego damos doble clic o arrastramos la opción tabla hacia la ventana de diseño



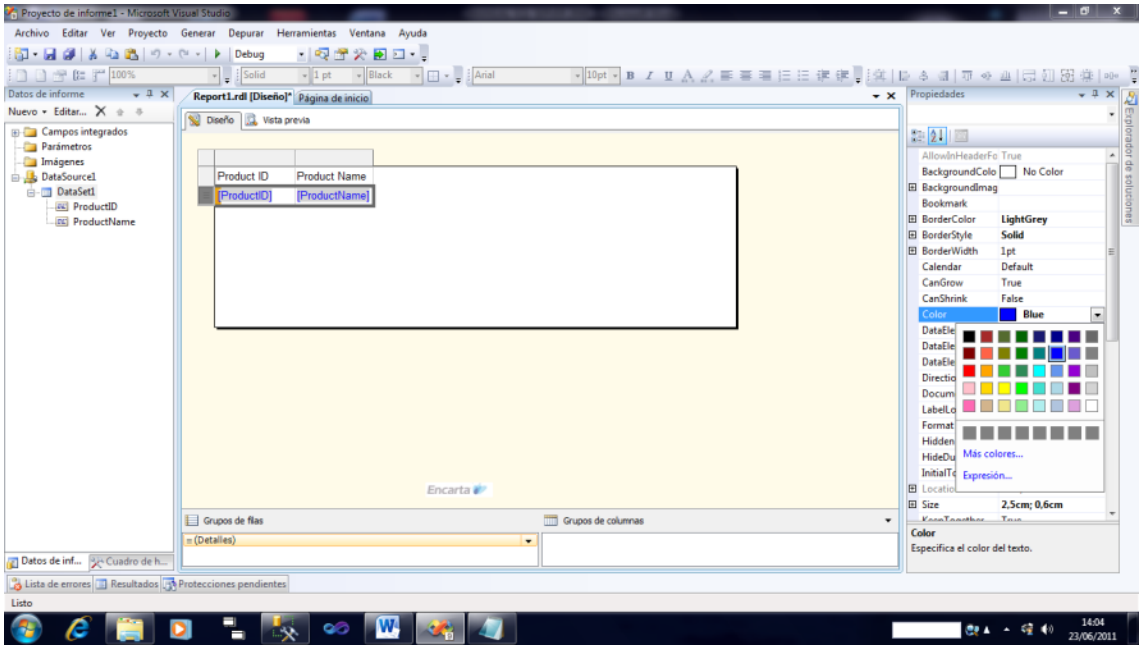
Luego volvemos a la ventana origen de datos
Y arrastramos los elementos del **DataSource1** hacia las columnas que se desean.



Estamos trabajando en la pestaña diseño. Luego en la pestaña **vista previa**, le damos clic.

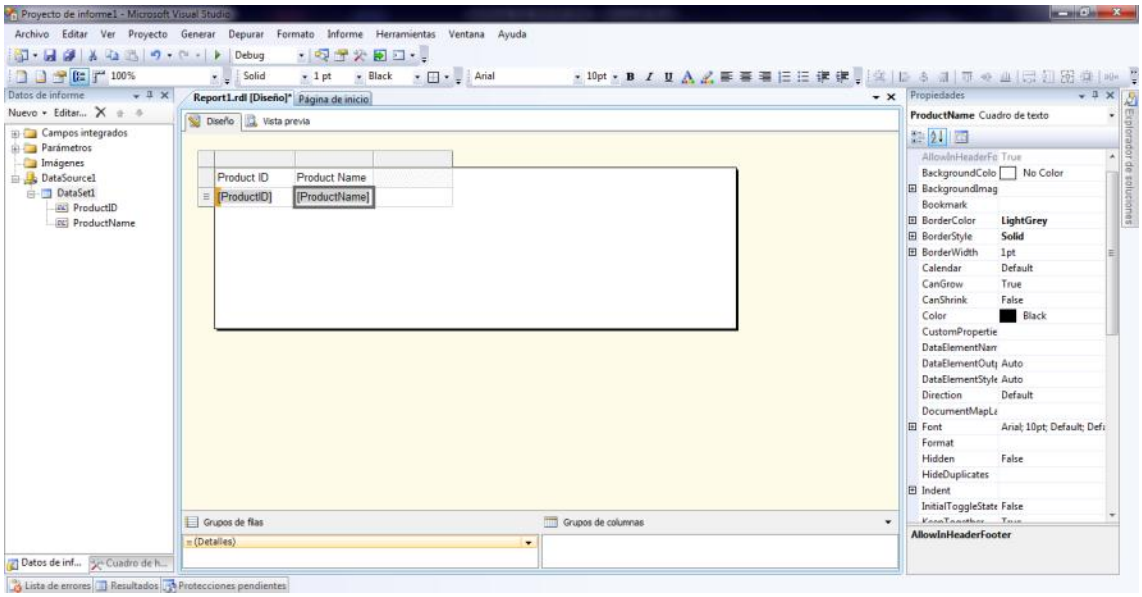


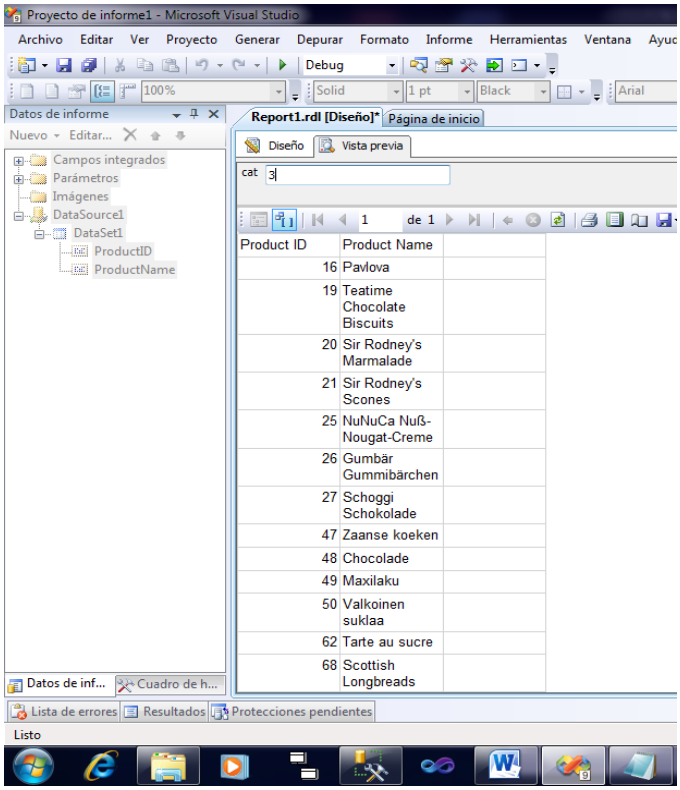
Ahora si queremos darle toques de color, texto u otras, lo hacemos en la pestaña de diseño, con la ayuda de sus propiedades, que aparece en la parte derecha.



Ejemplo 1: Crear un reporte en el cual se muestre el PoductID, ProductName. Filtrado por un parámetro.

```
alter proc lproductos
@cat int
as
select ProductID,ProductName from Products
where CategoryID=@cat
```

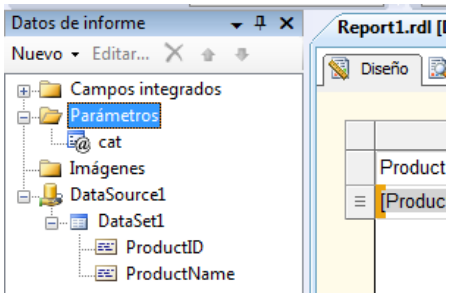




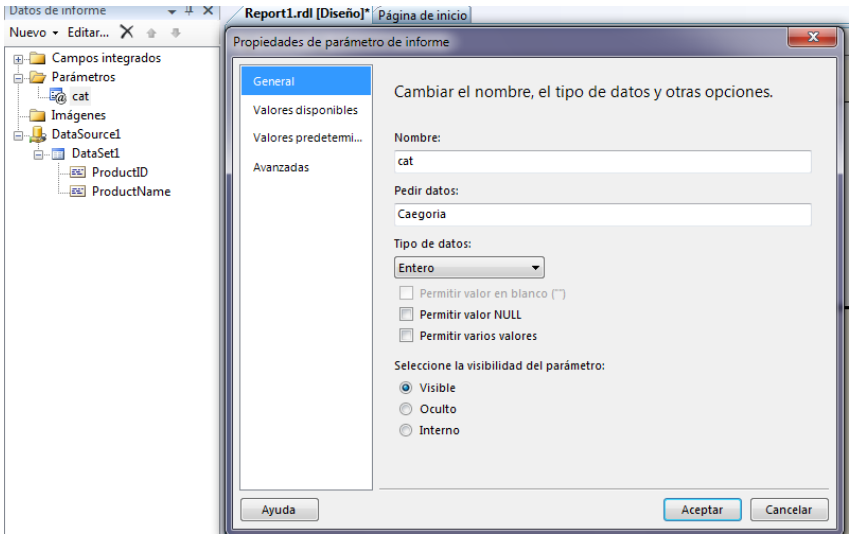
Cuando estamos en la pestaña **vista previa**, aparece una caja por el cual vamos a filtrar, este es el parámetro creado en procedimiento. En este caso es por **CategoryID.**

Ahora si queremos que no aparezca **CAT**, podemos cambiarlo:

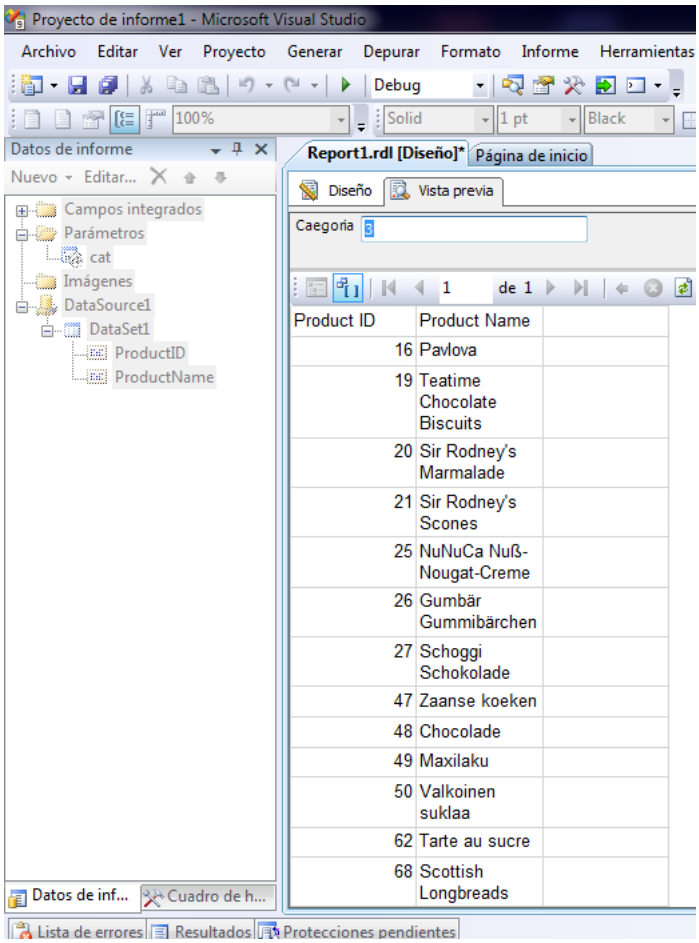
- Volvemos a la pestaña de diseño
- En la ventana Datos de **Informe**, desplegamos en el signo más donde dice **parámetros**.



- Damos doble clic en el nombre del parámetro en este caso **@cat.**



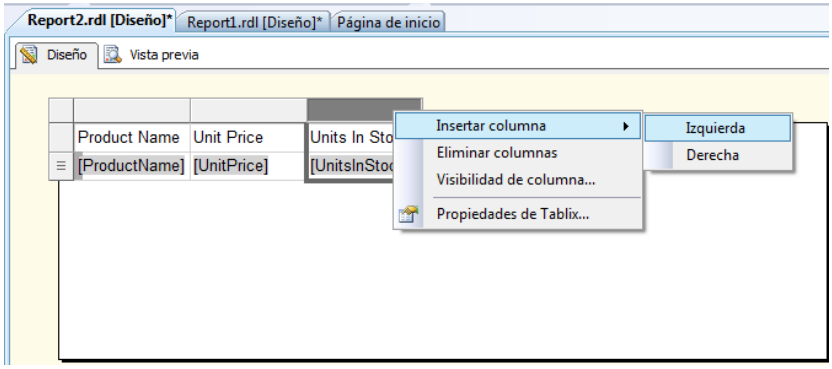
- En la opción **Pedir Datos** cambiamos el nombre, en la opción **Nombre** No se cambia nada porque es el nombre del parámetro, si lo haces generarías un error; ese nombre se le asignó al hacer el procedimiento.

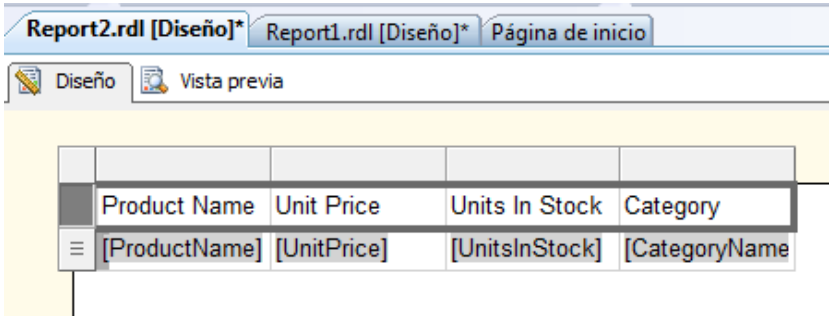


Ejemplo 2:

```
create proc informe
as
select ProductName,UnitPrice,UnitsInStock,CategoryName
from Products
inner join Categories on
Categories.CategoryID=Products.CategoryID
```

Nota: si queremos agregar más columnas es muy sencillo, lo primero es seleccionar cualquiera de las columnas clic derecho, insertar columna, ya sea derecha o izquierda. Similar es para las filas.





Report2.rdl [Diseño]* | Report1.rdl [Diseño]* | Página de inicio

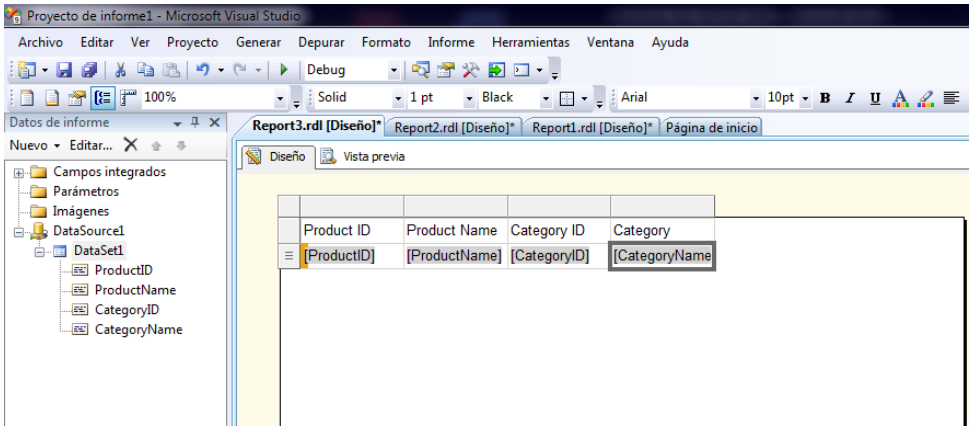
Diseño | Vista previa

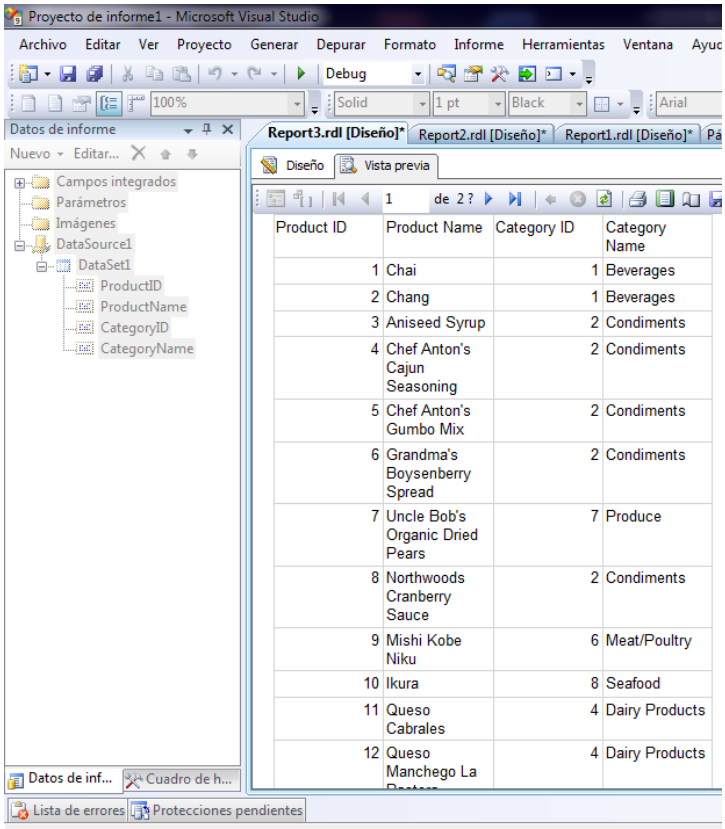
1 de 2 ?

Product Name	Unit Price	Units In Stock	Category Name
Chai	18,0000	39	Beverages
Chang	19,0000	17	Beverages
Aniseed Syrup	10,0000	13	Condiments
Chef Anton's Cajun Seasoning	22,0000	53	Condiments
Chef Anton's Gumbo Mix	21,3500	0	Condiments
Grandma's Boysenberry Spread	25,0000	120	Condiments
Uncle Bob's Organic Dried Pears	30,0000	15	Produce
Northwoods Cranberry Sauce	40,0000	6	Condiments
Mishi Kobe Niku	97,0000	29	Meat/Poultry
Ikura	31,0000	31	Seafood
Queso Cabrales	21,0000	22	Dairy Products
Queso Manchego La	38,0000	86	Dairy Products

Ejemplo 3:

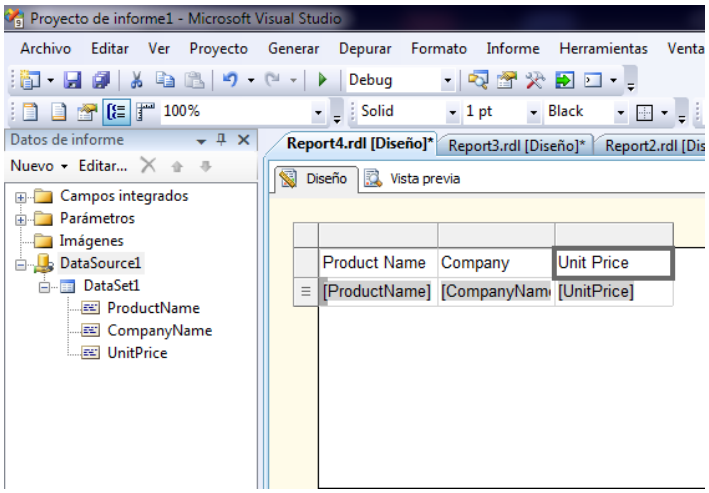
```
create proc informe2
as
select ProductID,ProductName,c.CategoryID,CategoryName
from Products
inner join Categories as c on
C.CategoryID=Products.CategoryID
```

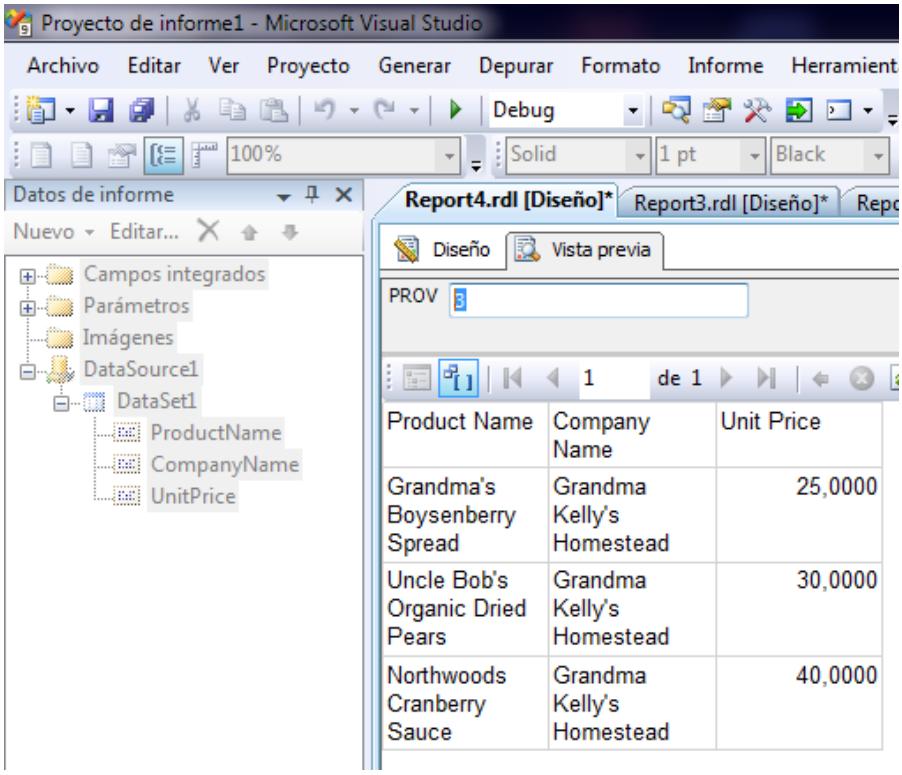




Ejemplo 4: Crear un reporte usando reporting services, que muestre el nombre del producto, nombre de la empresa proveedor, precio seleccionar el nombre de la empresa y mostrar los productos del proveedor seleccionado

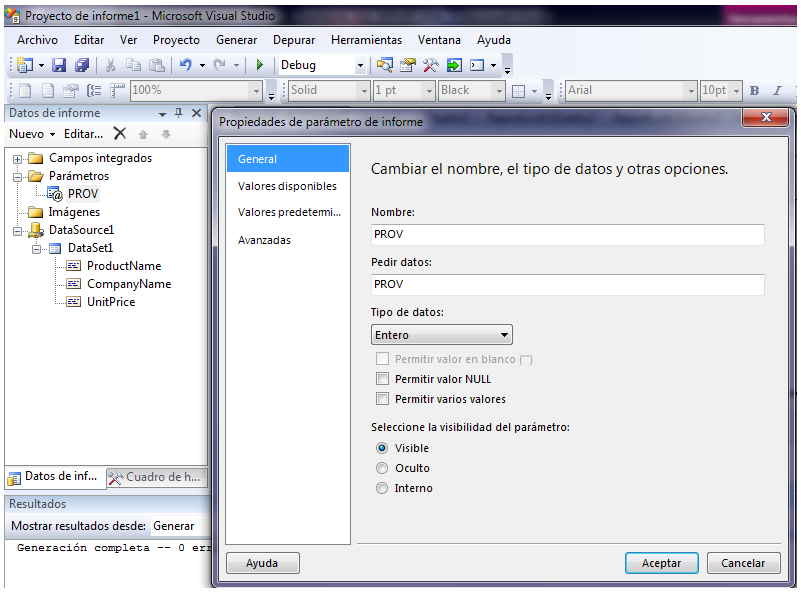
```
CREATE PROC INFO_PRO
 @PROV INT
 AS
 SELECT ProductName,S.CompanyName,UnitPrice
 FROM Products AS P
 INNER JOIN Suppliers AS S ON P.SupplierID=S.SupplierID
 WHERE S.SupplierID=@PROV
```



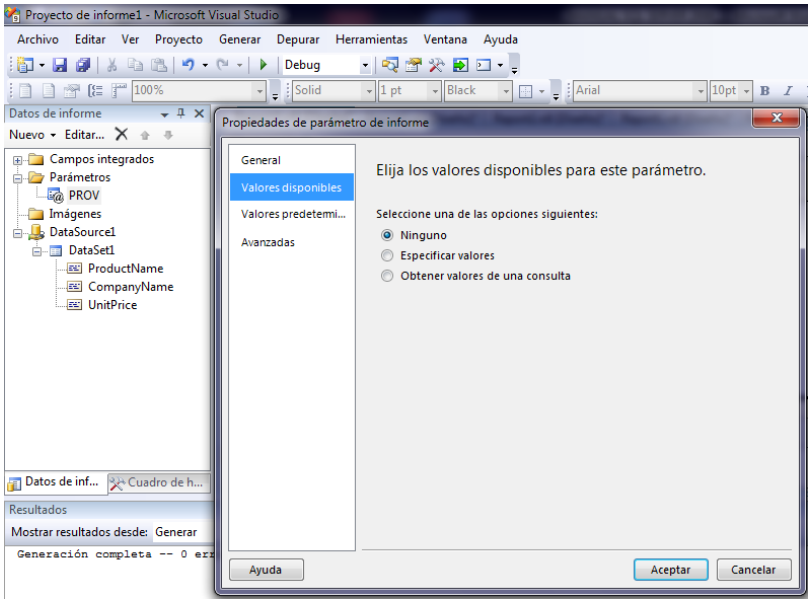


Nota: Podemos cambiar la caja que pide los valores.

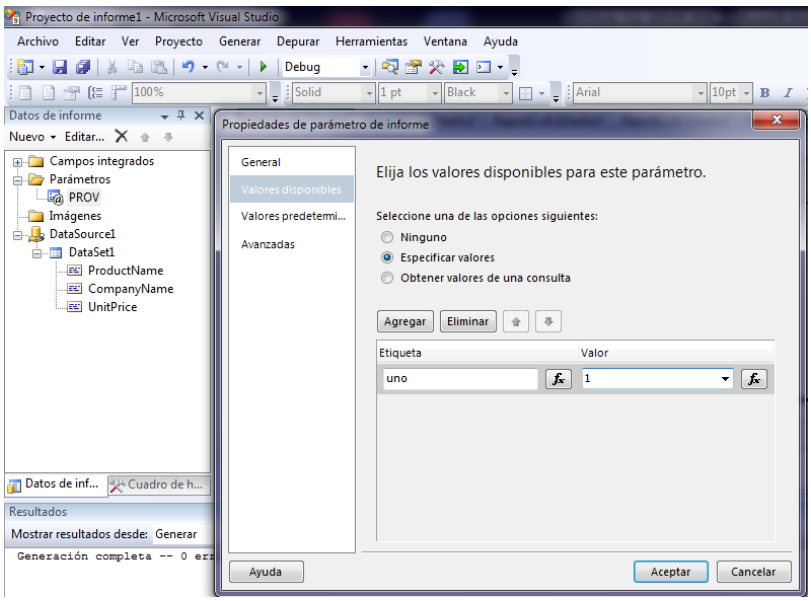
- Hacemos lo mismo que hacíamos para cambiar el nombre de pedido.



- Vamos a la opción valores disponibles.
- Nos aparece tres opciones



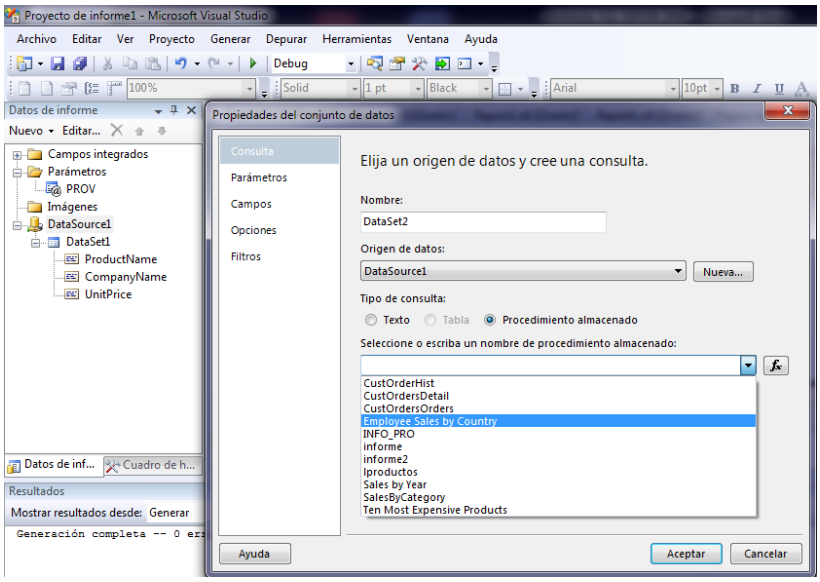
- La primera opción **Ninguno**, sale la opción dela caja, donde tenemos que escribir el valor
- La segunda opción **Especificar valores**, es cuando conocemos los valores, vamos agregando uno a uno. Lo cual no es tan recomendable si son varias.



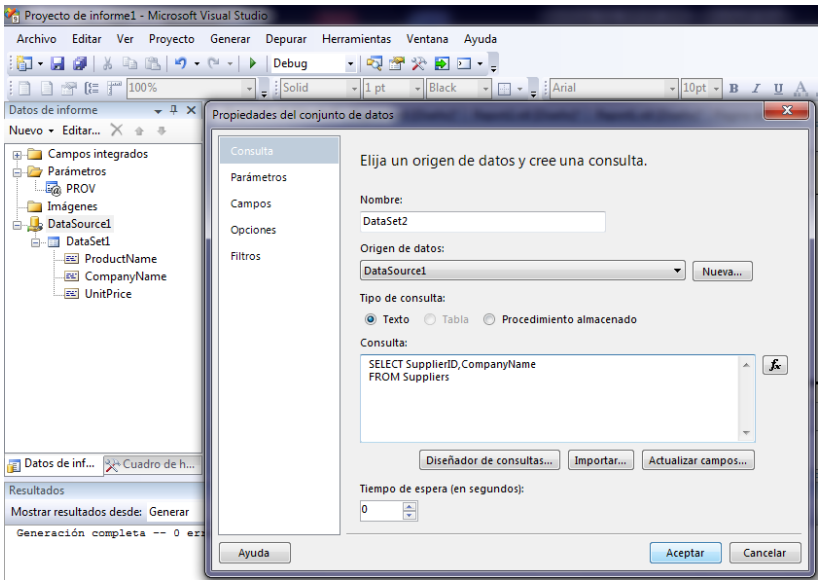
En donde sale etiqueta es el nombre con el que se va a mostrar y el valor es con el que se va a comparar.

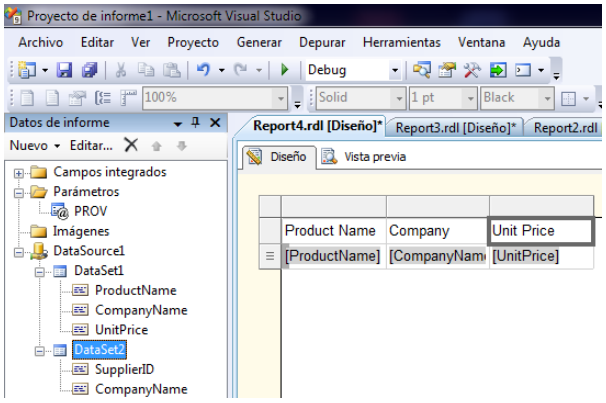
- La tercera opción **Obtener valores de una consulta**, esta es cuando usamos una consulta, esta puede ser creada como procedimiento en el SQL o también directo. Lo que vimos al inicio cuando hacíamos la consulta.

Primero crear la otra consulta

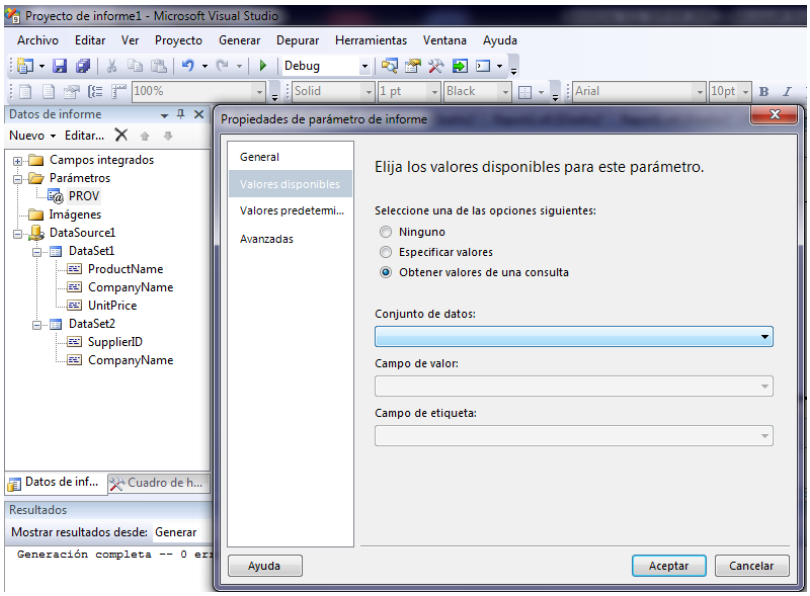


O también

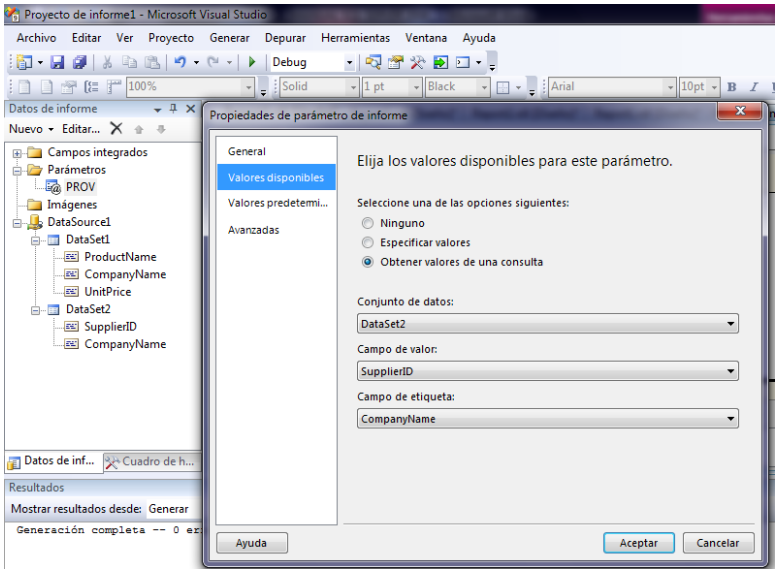




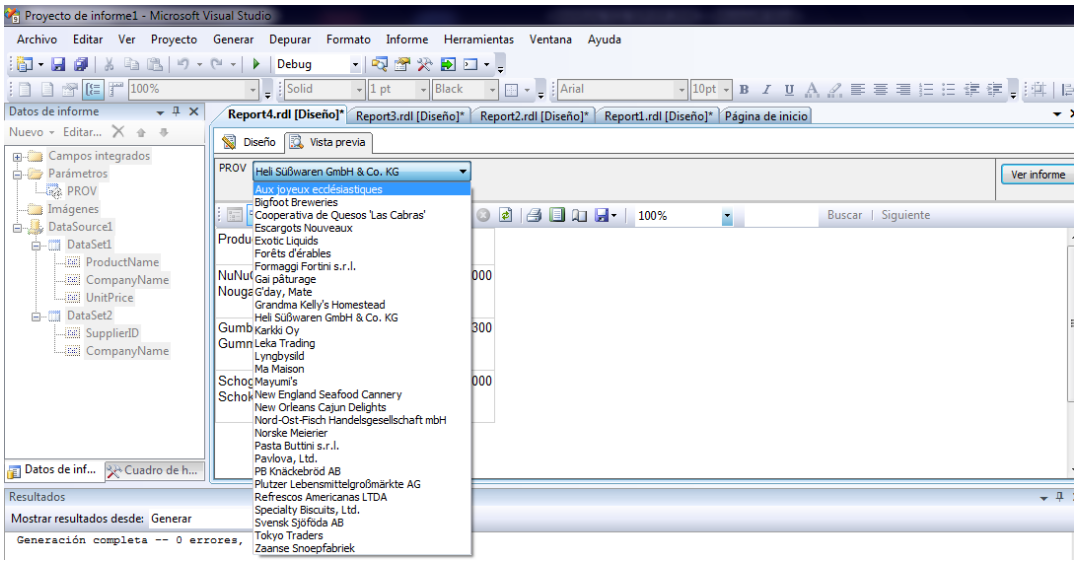
Luego vamos a **Propiedades del parámetro**, dándole doble clic, elegimos **Valores disponibles** y checkamos la tercera opción.



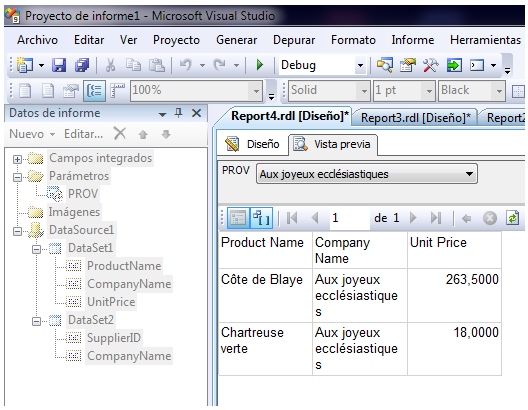
Buscamos el nombre de la segunda consulta (**DataSet2**) en **Conjunto de datos**.
En el **Campo de Valor** es con el cual se va filtrar, y **Campo de etiqueta** es el nombre con el que va aparecer.



Aceptas y vas a la pestaña **vista previa**.



Seleccionas la opción que deseas y le das clic en **Ver informe** o un enter.



Ejemplo 5: Mostrar el número de productos vendidos en cada mes por cada empleado

Creamos el procedimiento

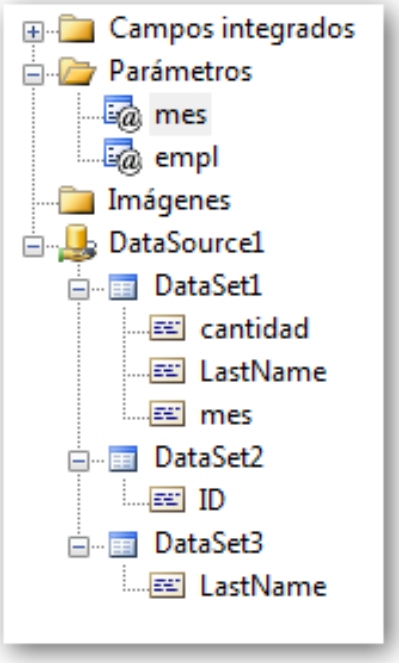
```
CREATE PROC vendidos1
 @mes INT,
 @empl varchar(35)
  AS
 SELECT count(Quantity)as cantidad,
      LastName,MONTH(OrderDate) as mes
 FROM Products AS P
 INNER JOIN [Order Details] as od on p.ProductID=od.ProductID
 inner join Orders as o on o.OrderID=od.OrderID
 inner join Employees as e on e.EmployeeID=o.EmployeeID
 where MONTH(OrderDate)=@mes and e.LastName=@empl
 group by MONTH(OrderDate),LastName
```

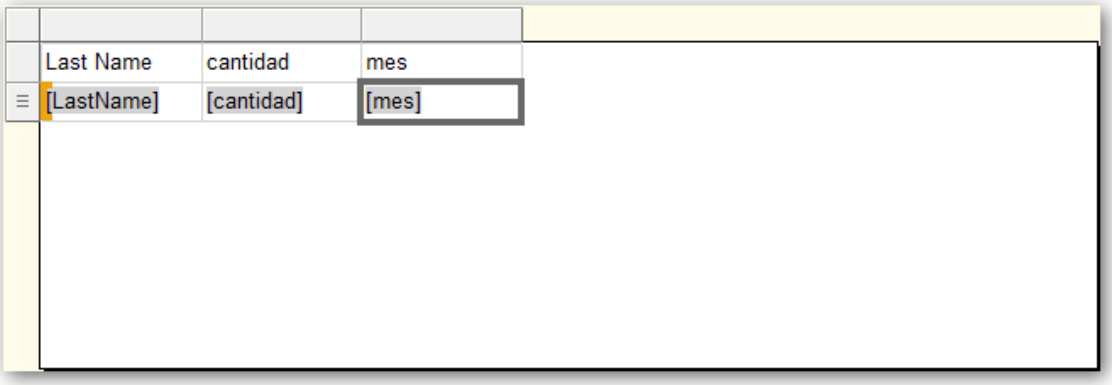
Procedimiento para mostrar el mes

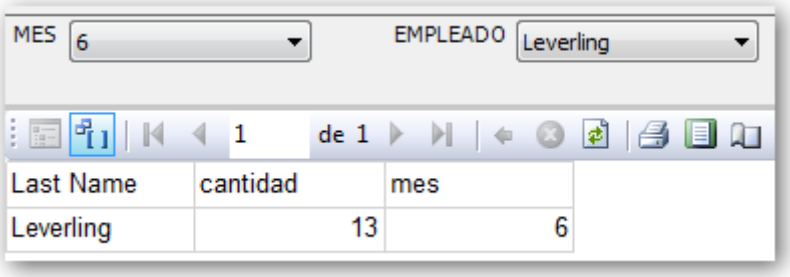
```
 ALTER PROC MES
AS
SELECT DISTINCT MONTH(OrderDate)
FROM Orders
ORDER BY MONTH(OrderDate)
```

Procedimiento para mostrar al empleado

```
 CREATE PROC EMPLEADO
AS
SELECT LastName
FROM Employees
ORDER BY LastName
```







Ejemplo 6: Mostrar el nombre del cliente, nombre del empleado, filtrar por el empleado que hizo la orden

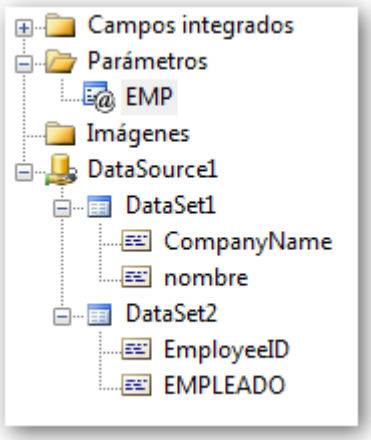
Creamos el procedimiento

CREATE PROC ORDENAR

```
@EMP INT
AS
SELECT CompanyName,(LastName+' '+FirstName)as nombre
FROM Customers as c
inner join Orders as o
on o.CustomerID=c.CustomerID
inner join Employees as e
on e.EmployeeID=o.EmployeeID
where e.EmployeeID=@EMP
```

Procedimiento para el empleado

```
CREATE PROC EMPLEADO1
AS
SELECT EmployeeID, (FirstName+' '+LastName) AS EMPLEADO
FROM Employees
```



CLIENTE	EMPLEADO
[CompanyName]	[nombre]

EMPLEADO Janet Leverling

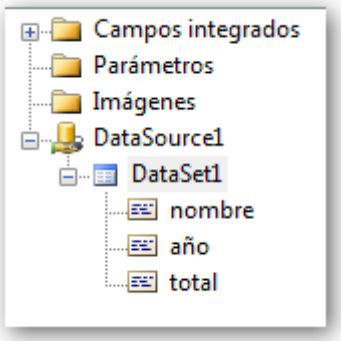
1 de 2 ?

CLIENTE	EMPLEADO
Victuailles en stock	Leverling Janet
Hanari Carnes	Leverling Janet
Wellington Importadora	Leverling Janet
Wartian Herkku	Leverling Janet
QUICK-Stop	Leverling Janet
LILA-Supermercado	Leverling Janet
Hungry Owl All-Night Grocers	Leverling Janet
Island Trading	Leverling Janet
LILA-Supermercado	Leverling Janet
Mère Paillarde	Leverling Janet

Ejemplo 7: Crear una matriz

Creando el procedimiento

```
create proc empleado_anio
as
select FirstName+''+LastName as nombre,
        YEAR(orderdate)as año,SUM(Quantity)as total
from Employees as e
inner join Orders as o
on o.EmployeeID=e.EmployeeID
inner join [Order Details] as od
on od.OrderID=o.OrderID
group by FirstName+''+LastName,
        YEAR(orderdate)
```



nombre	[año]
[nombre]	[Sum(total)]

nombre	1997	1998	1996
AndrewFuller	2604	2366	1085
NancyDavolio	3877	2315	1620
StevenBuchanan	1471	787	778
LauraCallahan	2843	2147	923
AnneDodsworth	955	1140	575
MichaelSuyama	1738	826	963
MargaretPeacock	5273	2313	2212
RobertKing	2292	1877	485
JanetLeverling	4436	2476	940

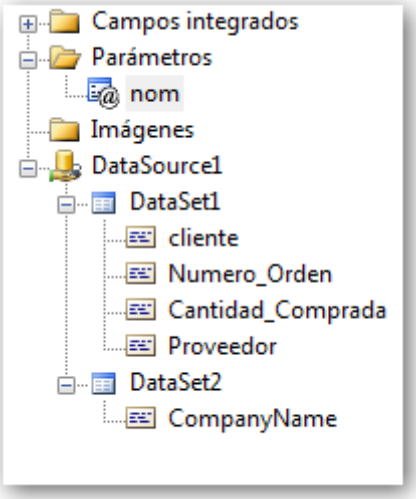
Ejemplo 8: Mostrar en un objeto table, el nombre del cliente, el nro. de orden y cantidad comprada, además del nombre del proveedor mostrar el nombre del cliente en un combox o drop down, para filtrar la información.

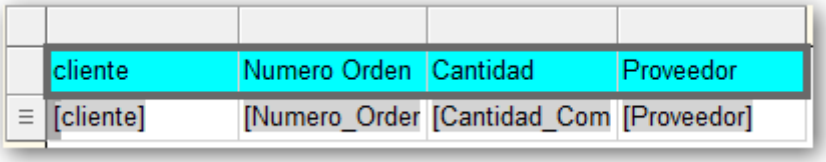
Creando el procedimiento

```
alter proc orden
        @nom varchar(30)
as
select c.CompanyName as cliente,od.OrderID as [Numero Orden],
    od.Quantity as [Cantidad Comprada] ,s.CompanyName as Proveedor
from Customers as c
inner join Orders as o
on c.CustomerID = o.CustomerID
inner join [Order Details] as od
on o.OrderID = od.OrderID
inner join Products as p
on od.ProductID = p.ProductID
inner join Suppliers as s
on p.SupplierID = s.SupplierID
where c.CompanyName = @nom
```

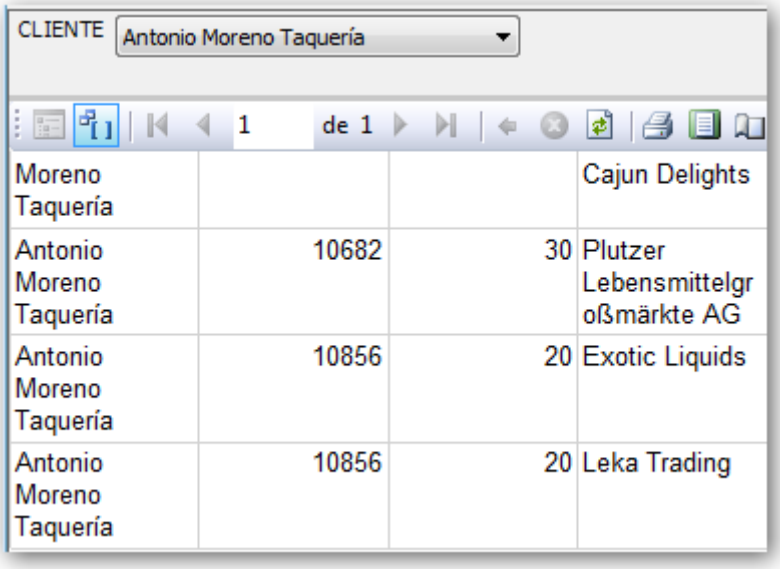
Procedimiento para el cliente

```
alter proc cliente
as
select distinct CompanyName
from Customers
```





cliente	Numero Orden	Cantidad	Proveedor
[cliente]	[Numero_Order	[Cantidad_Com	[Proveedor]

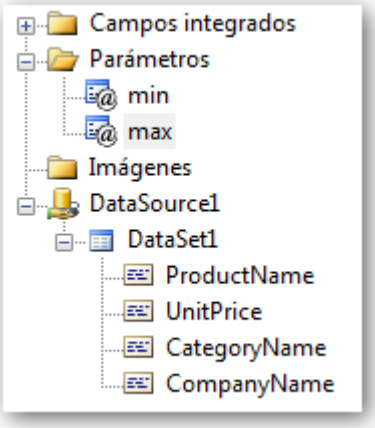


Moreno Taquería			Cajun Delights
Antonio Moreno Taquería	10682	30	Plutzer Lebensmittelgroßmärkte AG
Antonio Moreno Taquería	10856	20	Exotic Liquids
Antonio Moreno Taquería	10856	20	Leka Trading

Ejemplo 9: Mostrar: nombre del producto, unitprice, category name nombre del proveedor del producto entre dos precios.

Creando el procedimiento

```
alter proc entre
        @min int,
        @max int
as
select p.ProductName,p.UnitPrice,c.CategoryName,s.CompanyName
from Products as p inner join Categories as c
on p.CategoryID = c.CategoryID inner join Suppliers as s
on p.SupplierID = s.SupplierID
where p.UnitPrice between @min and @max
order by p.UnitPrice
```



Product Name	Unit Price	Category	Company
[ProductName]	[UnitPrice]	[CategoryName	[CompanyNam

Precio Minimo 13 Precio Maximo 15

1 de 1

Product Name	Unit Price	Category Name	Company Name
Original Frankfurter grüne Soße	13,0000	Condiments	Plutzer Lebensmittelgroßmärkte AG
Escargots de Bourgogne	13,2500	Seafood	Escargots Nouveaux
Laughing Lumberjack Lager	14,0000	Beverages	Bigfoot Breweries
NuNuCa Nuß-Nougat-Creme	14,0000	Confections	Heli Süßwaren GmbH & Co. KG
Sasquatch Ale	14,0000	Beverages	Bigfoot Breweries
Singaporean Hokkien Fried Mee	14,0000	Grains/Cereals	Leka Trading
Outback Lager	15,0000	Beverages	Pavlova, Ltd.
Röd Kaviar	15,0000	Seafood	Svensk Sjöföda AB

Ejemplo 10: Mostrar: productname stock del producto, categoryname, company name del proveedor. Filtrar por proveedor y nombre de la categoría.

Creando el procedimiento

```
create proc ejercicio3
        @prov varchar(30),
        @cate varchar(30)
as
select p.ProductName,p.UnitsInStock,c.CategoryName,s.CompanyName
from Products as p inner join Categories as c
on p.CategoryID = c.CategoryID inner join Suppliers as s
on p.SupplierID = s.SupplierID
where s.CompanyName = @prov and c.CategoryName = @cate
```

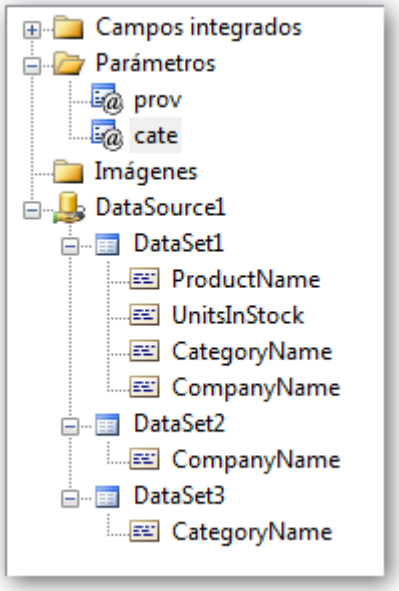
Procedimiento para el proveedor

```
create proc proveedor
as
select CompanyName
```

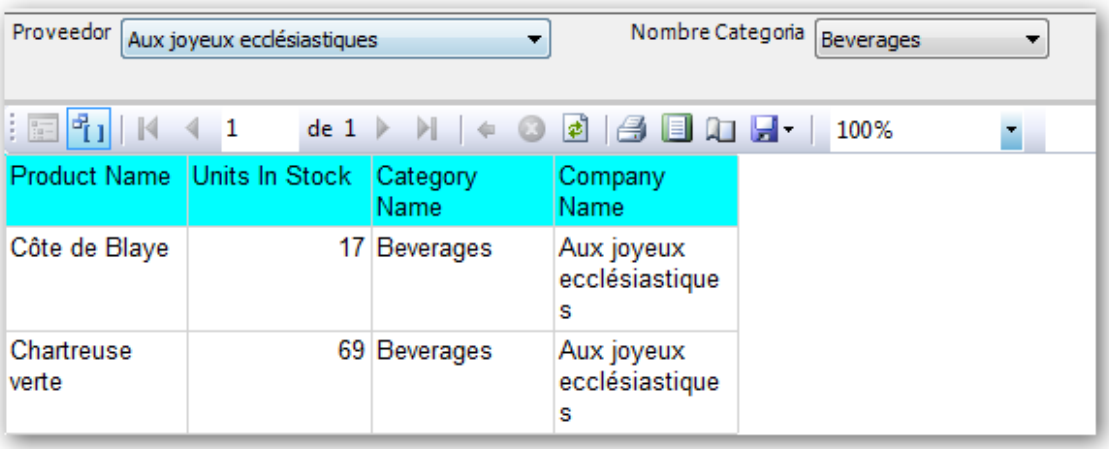
```
from Suppliers
```

Procedimiento para el nombre de la categoría

```
create proc nom_cate
as
select CategoryName
from Categories
```



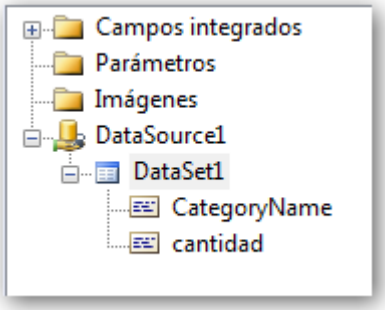
Product Name	Units In Stock	Category	Company
[ProductName]	[UnitsInStock]	[CategoryName	[CompanyNam

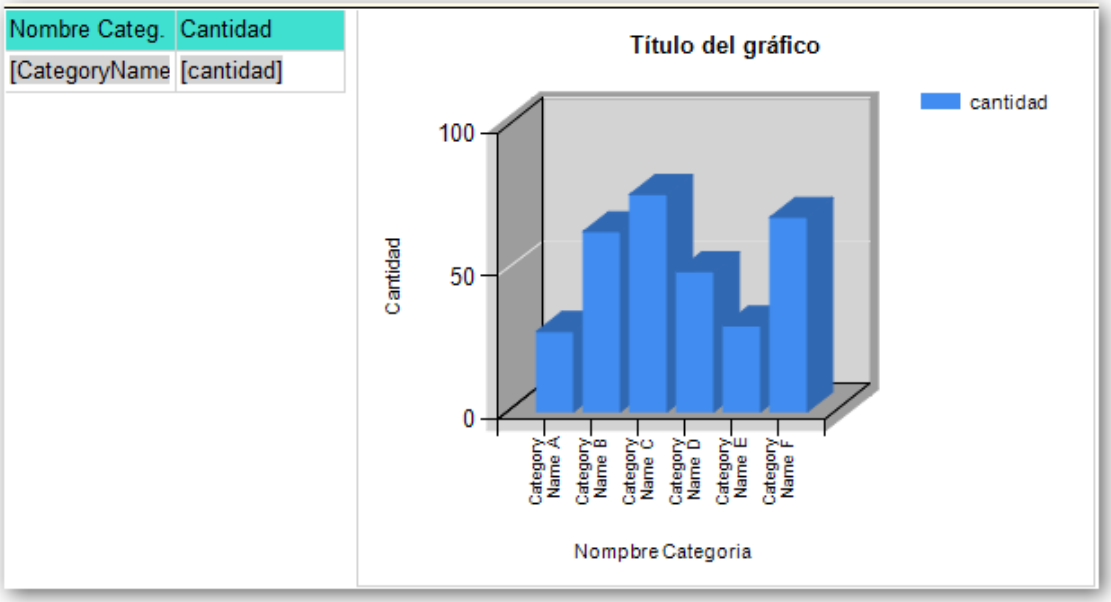


Ejemplo 11: Mostrar en un gráfico y en una tabla el número de productos y categorías.

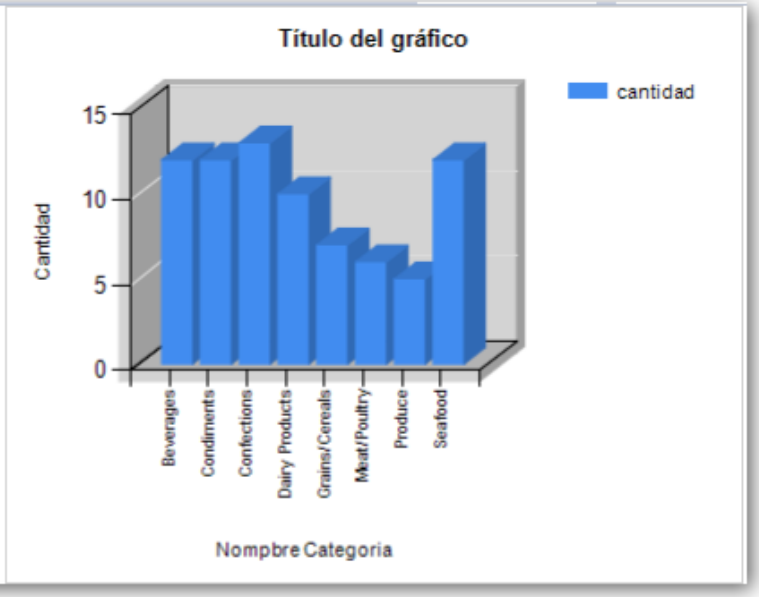
Creando el procedimiento

```
create proc ejer4
as
select c.CategoryName,count(p.ProductID) as cantidad
from Products as p inner join Categories as c
on p.CategoryID = c.CategoryID
group by c.CategoryName
```





Nombre Categ.	Cantidad
Beverages	12
Condiments	12
Confections	13
Dairy Products	10
Grains/Cereals	7
Meat/Poultry	6
Produce	5
Seafood	12



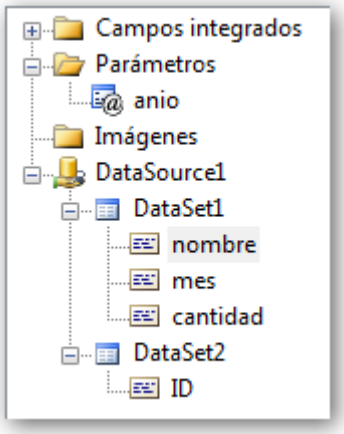
Ejemplo 12: Mostrar empleados nombre y apellido, numero de ordenes realizadas en un mes del año especificado, representarlo a través de un gráfico.

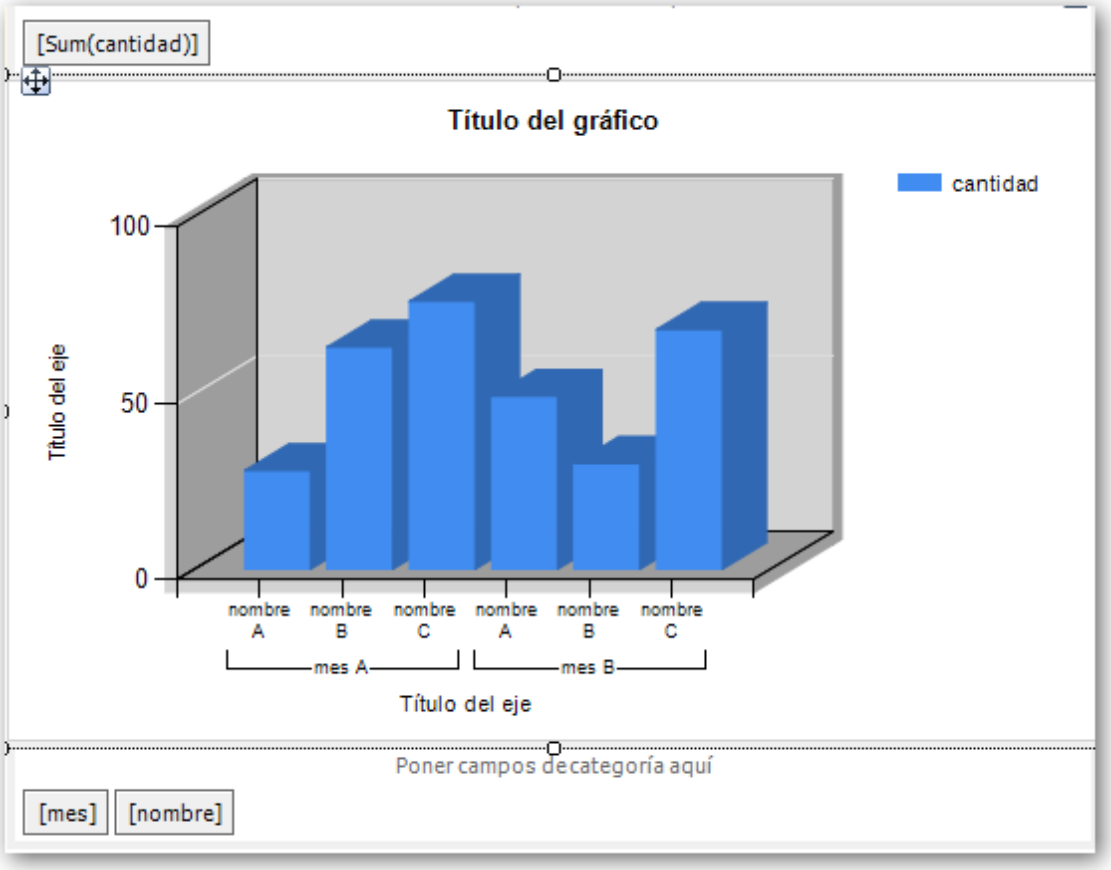
Creamos el procedimiento.

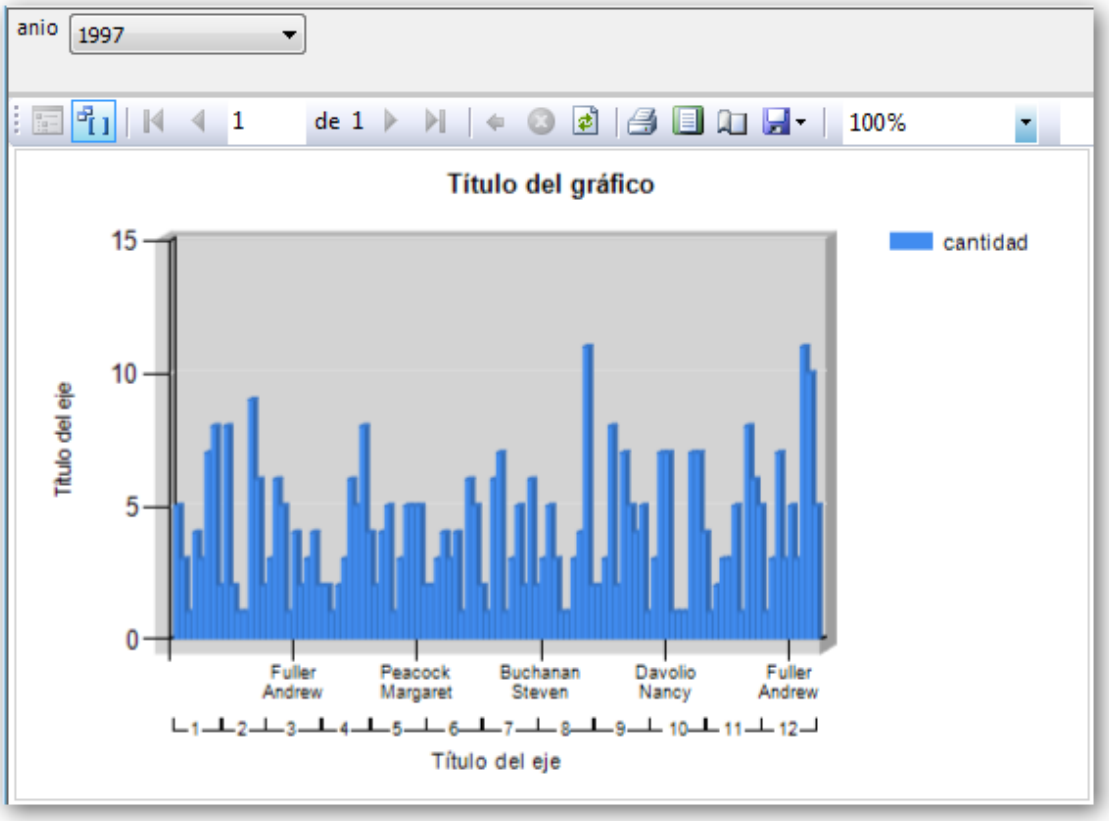
```
alter proc ejercicio5
        @anio int
as
select (LastName+' '+ FirstName) as nombre,
    MONTH(OrderDate) as mes,COUNT(OrderID)as cantidad
from Employees as e
inner join Orders as o
on e.EmployeeID = o.EmployeeID
where (YEAR(OrderDate) = @anio)
group by (LastName+' '+ FirstName),MONTH(OrderDate)
```

Procedimiento para el año

```
create proc anio1
as
select YEAR(OrderDate)
from Orders
group by YEAR(OrderDate)
order by YEAR(OrderDate)
```

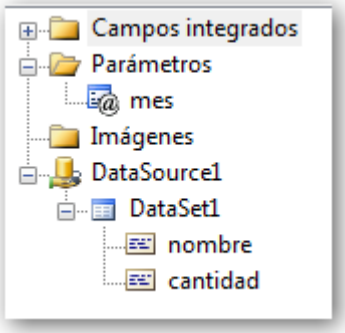

Campos integrados
Parámetros
anio
Imágenes
DataSource1
DataSet1
nombre
mes
cantidad
DataSet2
ID

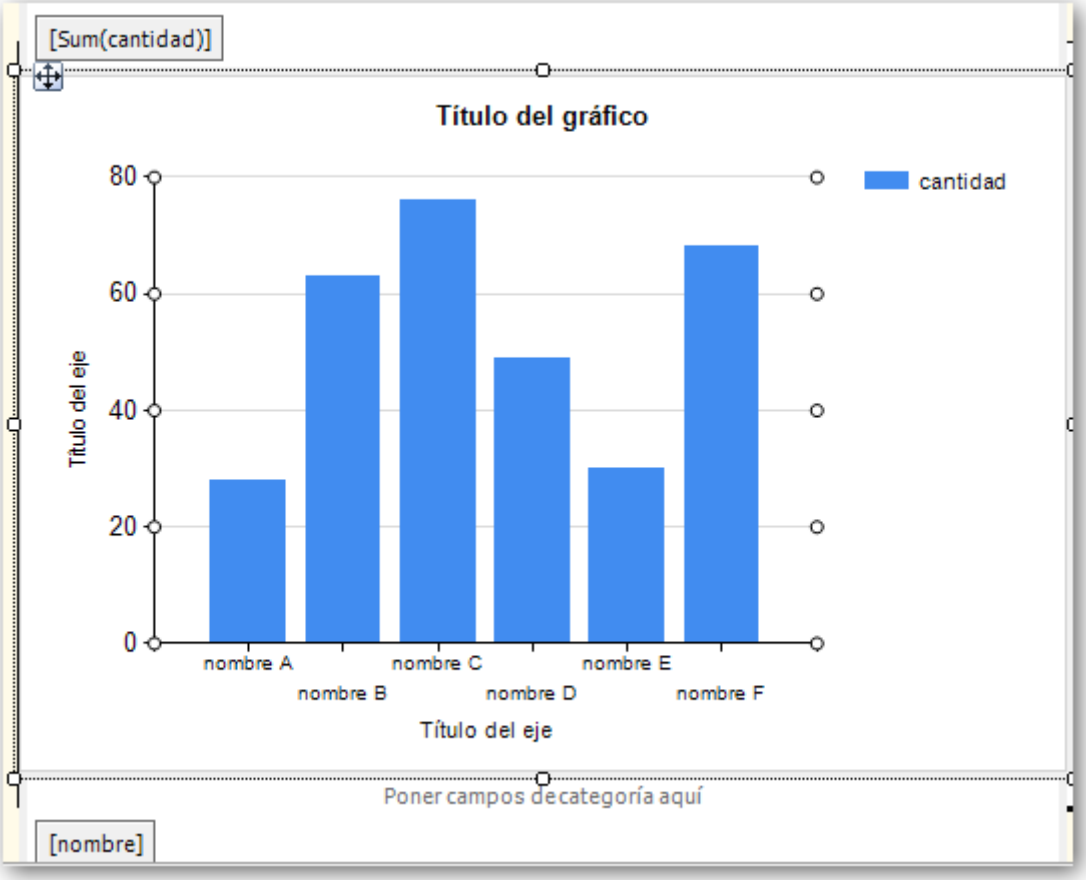

[Sum(cantidad)]
Título del gráfico
cantidad
100
50
0
Título del eje
nombre A
nombre B
nombre C
nombre A
nombre B
nombre C
mes A
mes B
Título del eje
Poner campos de categoría aquí
[mes]
[nombre]

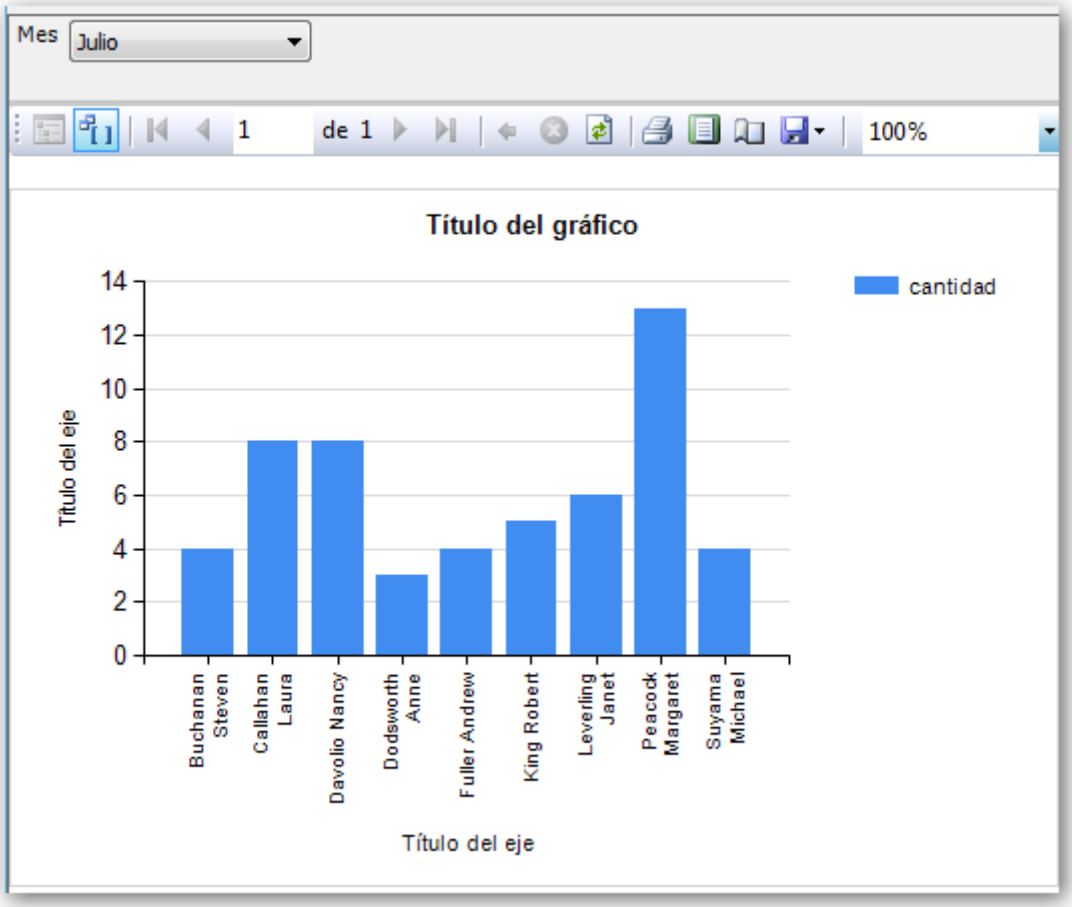


Ejemplo 13: Parámetros mes, tener en cuenta el nombre de los empleados, numero de ordenes realizadas, mostrar en un gráfico.

```
create proc ejercicio6
        @mes int
as
select (LastName+' '+ FirstName) as nombre,COUNT(OrderID)as cantidad
from Employees as e inner join Orders as o
on e.EmployeeID = o.EmployeeID
where MONTH(OrderDate) = @mes
group by LastName+' '+ FirstName,MONTH(OrderDate)
```



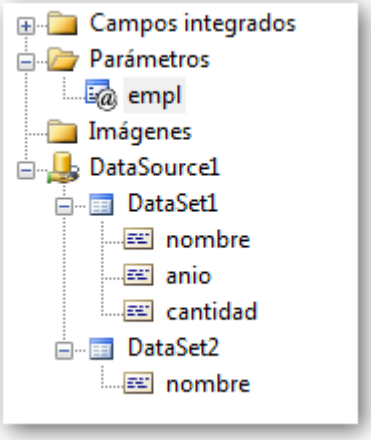

[Sum(cantidad)]
Título del gráfico
80
60
40
20
0
Título del eje
cantidad
nombre A
nombre B
nombre C
nombre D
nombre E
nombre F
Título del eje
Poner campos de categoría aquí
[nombre]

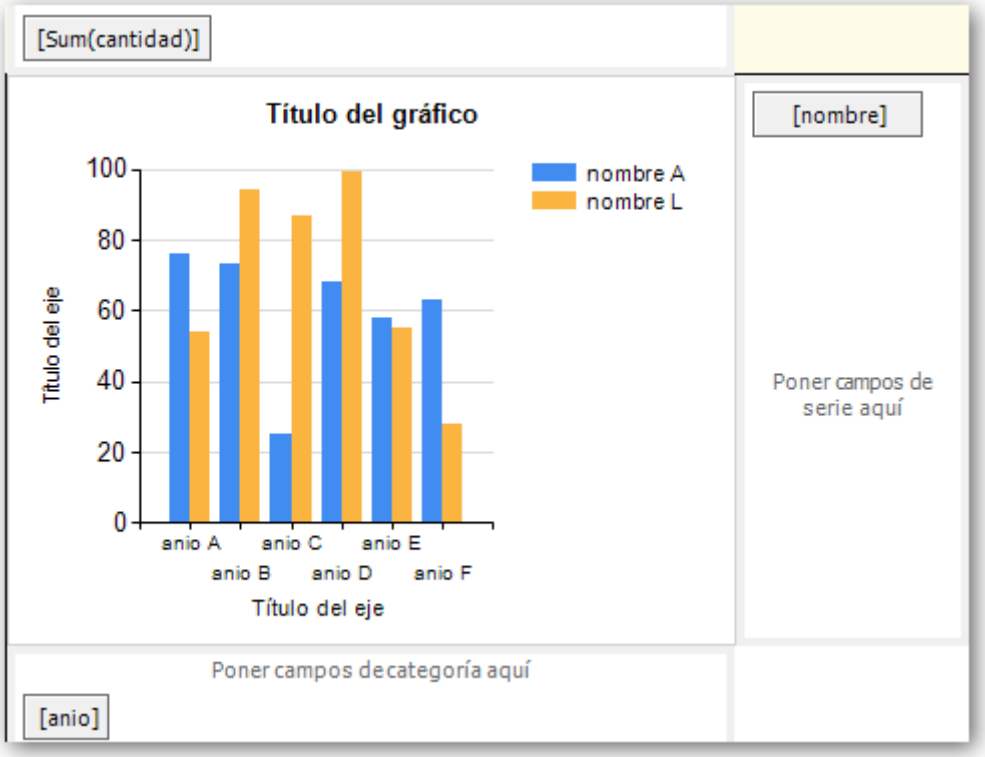


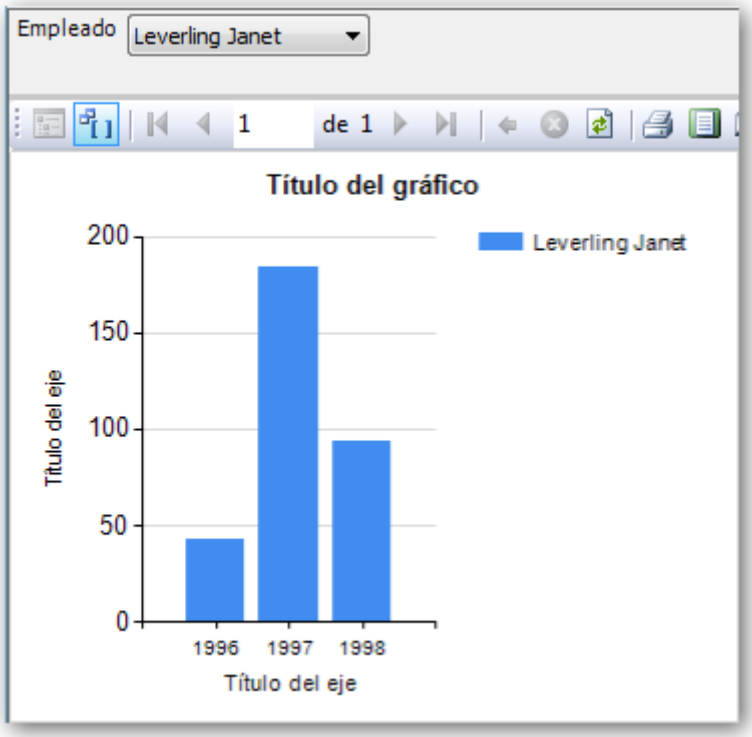
Ejemplo 14: Parámetro empleado, ver el año y el monto total vendido por el empleado mostrar en un gráfico.

```
alter proc ejercicio7
        @empl varchar(30)
as
select (LastName +' '+ FirstName) as nombre,YEAR(OrderDate) as anio,
COUNT(Quantity) as cantidad
from Employees as e inner join Orders as o
on e.EmployeeID = o.EmployeeID inner join [Order Details] as od
on o.OrderID = od.OrderID
where (LastName +' '+ FirstName) = @empl
group by (LastName +' '+ FirstName),YEAR(OrderDate)


select (LastName +' '+ FirstName) as nombre
from Employees
```


Campos integrados
Parámetros
empl
Imágenes
DataSource1
DataSet1
nombre
anio
cantidad
DataSet2
nombre

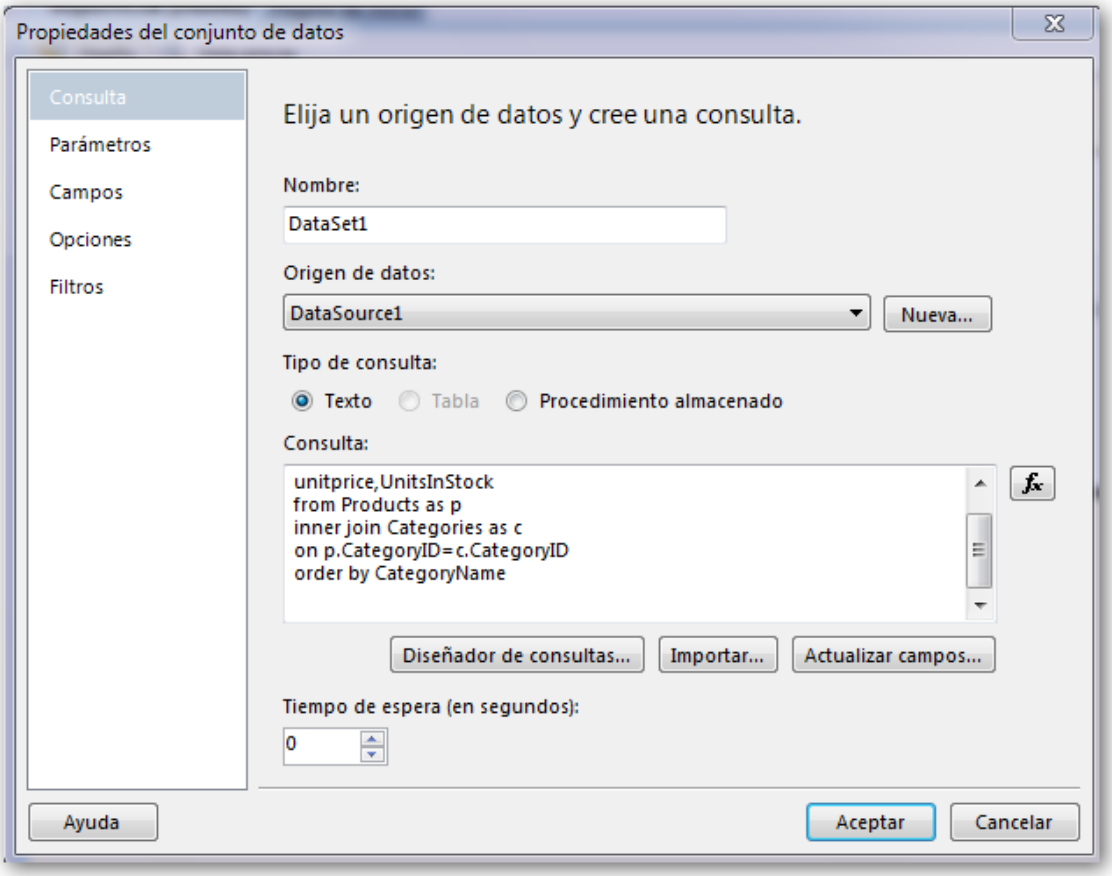

[Sum(cantidad)]
Título del gráfico
100
80
60
40
20
0
Título del eje
nombre A
nombre L
anio A
anio B
anio C
anio D
anio E
anio F
Título del eje
[nombre]
Poner campos de serie aquí
Poner campos decategoría aquí
[anio]



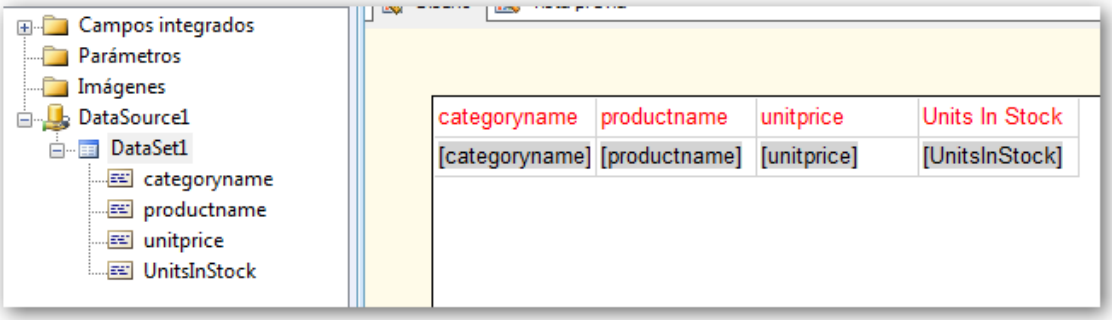
Ejemplo 15: Vamos a crear un grupo en el Visual 2008, para la cual vamos utilizar la siguiente consulta:

```
select categoryname, productname,
unitprice,UnitsInStock
from Products as p
inner join Categories as c
on p.CategoryID=c.CategoryID
order by CategoryName
```

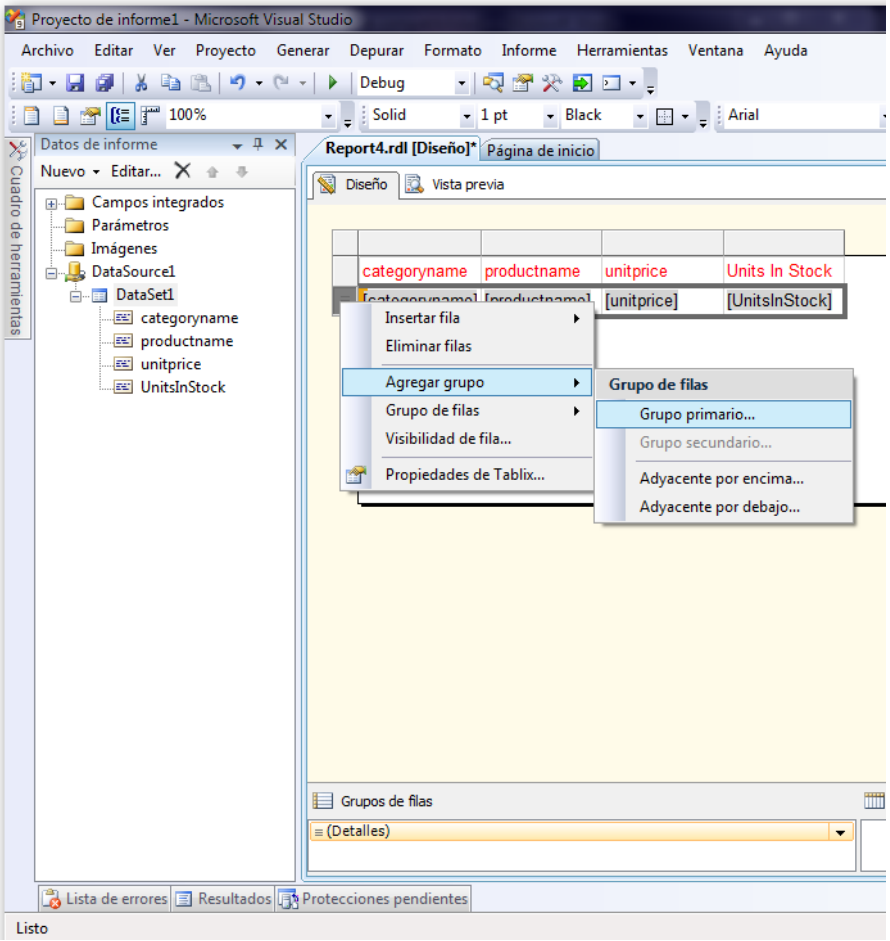
Directamente



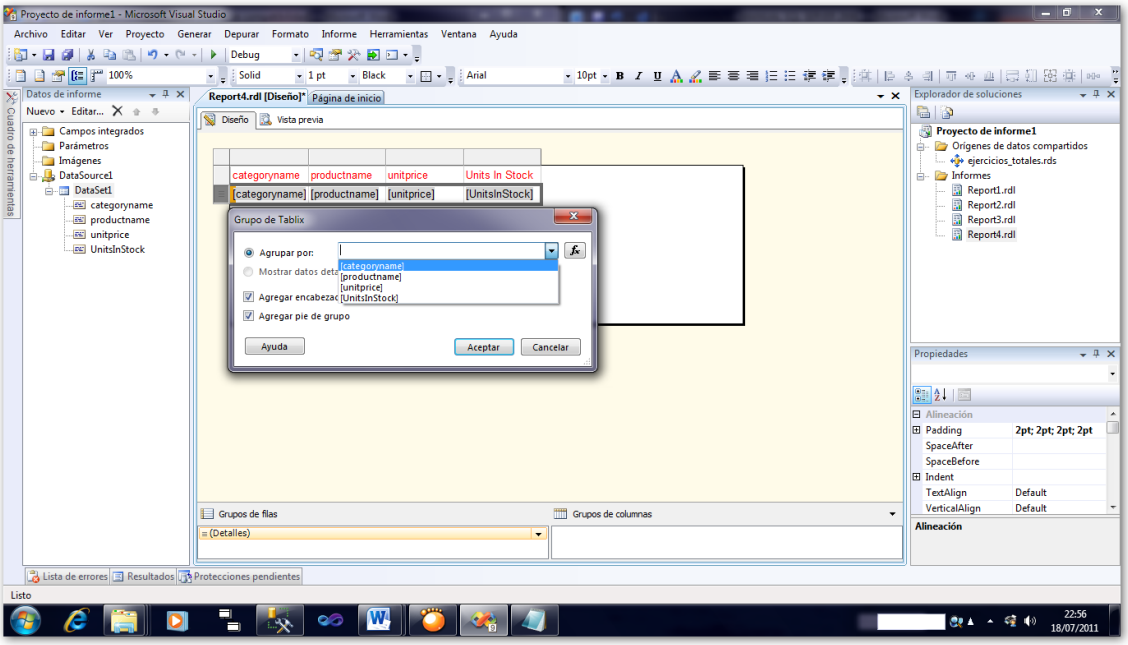
Luego vamos a crear una tabla con los datos



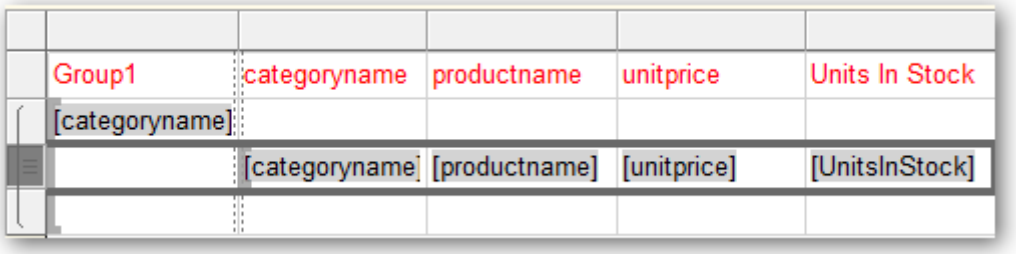
Seleccionamos la fila para crear un grupo
Clic derecho > Agregar Grupo > Grupo primario



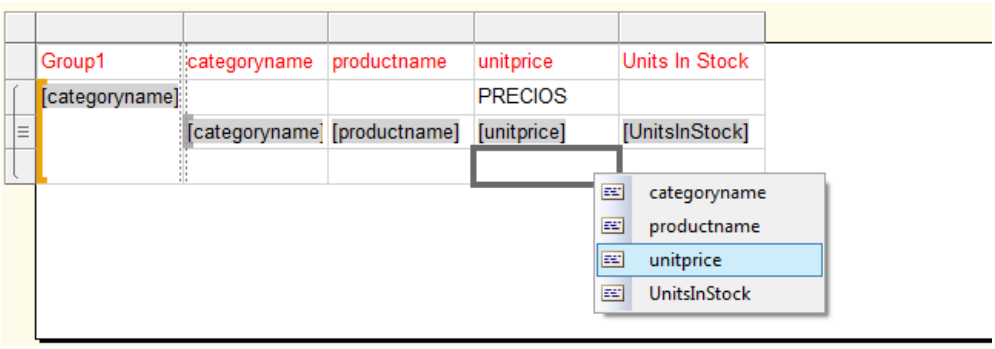
Se abre una ventana en el cual vamos agrupar en nuestro caso por categoryname. A la vez checamos las dos casillas: "**Agregar encabezado de grupo**", "**Agregar pie de grupo**"



Aceptamos



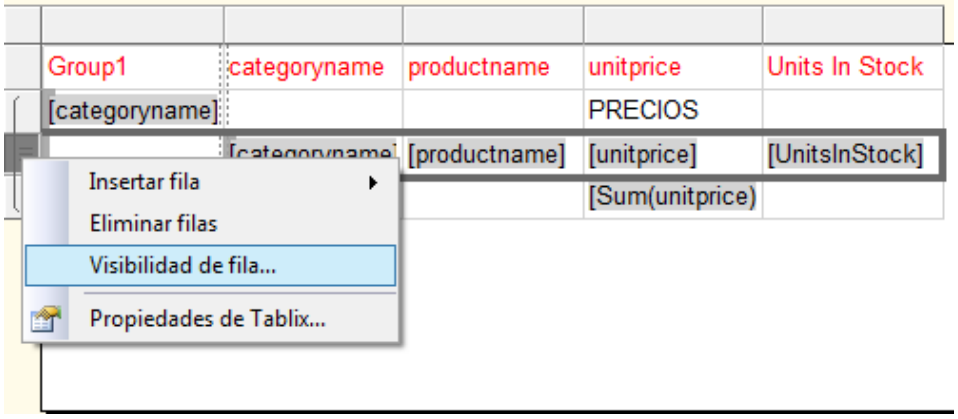
Vemos que han aparecido dos filas más, una encima y otra debajo eso es debido a las casillas checkadas, mencionadas anteriormente. En la fila de encima puedes tomarla como para un título por el estilo dependerá de tu trabajo; mientras que la fila que está debajo podrías utilizarlo para una operación, repito todo será cuestión de tu trabajo. Nosotros vamos a utilizar para la suma total del precio.



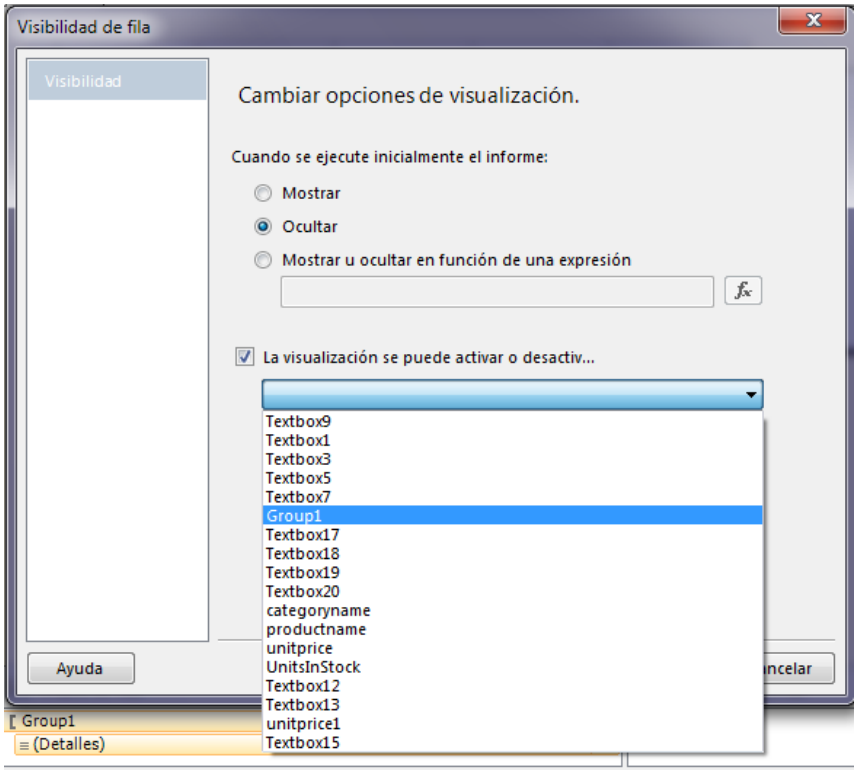
Vemos como se muestra en la vista previa.

Group1	categoryname	productname	unitprice	Units In Stock
Beverages			PRECIOS	
	Beverages	Chai	18,0000	39
	Beverages	Chang	19,0000	17
	Beverages	Guaraná Fantástica	4,5000	20
	Beverages	Sasquatch Ale	14,0000	111
	Beverages	Steeleye Stout	18,0000	20
	Beverages	Côte de Blaye	263,5000	17
	Beverages	Chartreuse verte	18,0000	69
	Beverages	Ipoh Coffee	46,0000	17
	Beverages	Laughing Lumberjack Lager	14,0000	52
	Beverages	Outback Lager	15,0000	15
	Beverages	Rhönbräu Klosterbier	7,7500	125
	Beverages	Lakkalikööri	18,0000	57
			455,7500	
Condiments			PRECIOS	
	Condiments	Aniseed Syrup	10,0000	13
	Condiments	Chef Anton's Cajun	22,0000	53

Se ha creado el grupo que tiene por nombre **Group1**.
Ahora bien vamos a ocultar en función del grupo creado.



Al hacer clic en esta **Visibilidad de fila**. Se habre una ventana en la cual vamos checkar la opción de **ocultar** y en la casilla **La visualización se puede activar o desactivar** vamos a buscar el nombre del grupo por el cual se quiere ocultar.



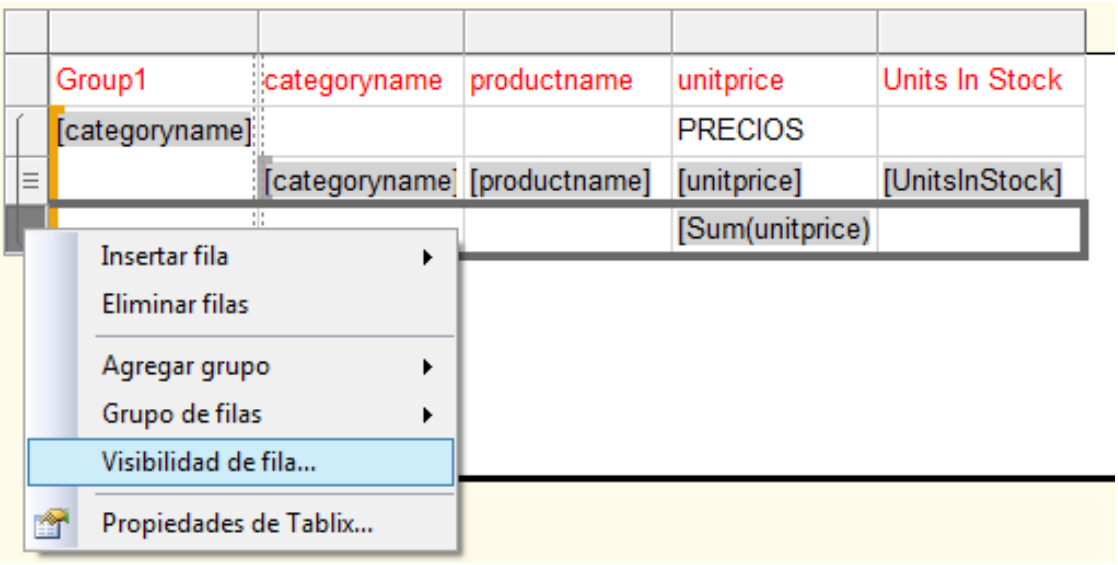
Y ahora vamos a la pestaña de vista previa y vemos el cambio.

Group1	categoryname	productname	unitprice	Units In Stock
⊞ Beverages			PRECIOS	
			455,7500	
⊞ Condiments			PRECIOS	
			276,7500	
⊞ Confections			PRECIOS	
			327,0800	
⊞ Dairy Products			PRECIOS	
			287,3000	
⊞ Grains/Cereals			PRECIOS	
			141,7500	
⊞ Meat/Poultry			PRECIOS	
			324,0400	
⊞ Produce			PRECIOS	
			161,8500	
⊞ Seafood			PRECIOS	
			248,1900	

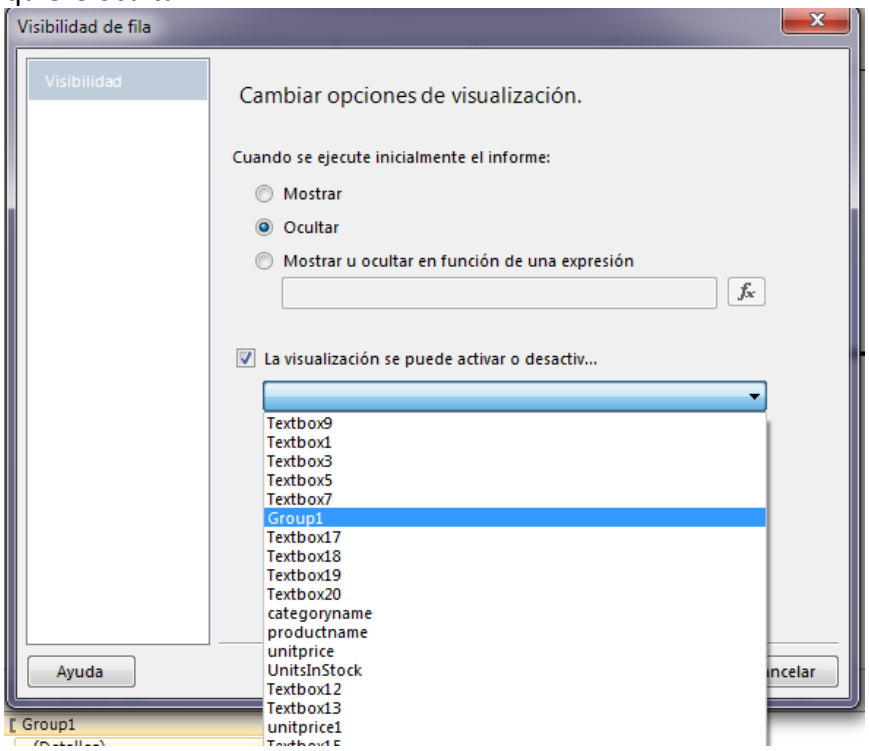
Al lado de cada grupo ha aparecido el signo más (+), esto quiere decir que están ocultos ciertos datos y para verlos solo hay que darle clic.

Group1	categoryname	productname	unitprice	Units In Stock
⊟ Beverages			PRECIOS	
	Beverages	Chai	18,0000	39
	Beverages	Chang	19,0000	17
	Beverages	Guaraná Fantástica	4,5000	20
	Beverages	Sasquatch Ale	14,0000	111
	Beverages	Steeleye Stout	18,0000	20
	Beverages	Côte de Blaye	263,5000	17
	Beverages	Chartreuse verte	18,0000	69
	Beverages	Ipoh Coffee	46,0000	17
	Beverages	Laughing Lumberjack Lager	14,0000	52
	Beverages	Outback Lager	15,0000	15
	Beverages	Rhönbräu Klosterbier	7,7500	125
	Beverages	Lakkalikööri	18,0000	57
			455,7500	
⊞ Condiments			PRECIOS	
			276,7500	
⊞ Confections			PRECIOS	
			327,0800	

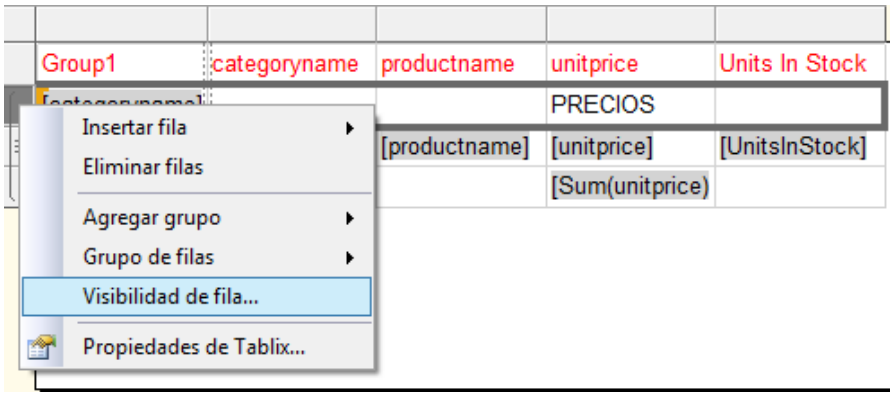
Pero si nos damos cuenta no se han ocultado las dos filas creadas anteriormente como encabezado y pie de grupo. Pero haciendo el mismo procedimiento podemos ocultarlo.



Aparece la misma ventana en el cual vamos a ocultar y elegir el grupo por el cual se quiere ocultar.



Lo mismo es para ocultar el encabezado del grupo.



Ahora vemos cual ha sido el cambio.

Group1	categoryname	productname	unitprice	Units In Stock
⊞ Beverages	Beverages	Chai	18,0000	39
⊞ Condiments	Condiments	Aniseed Syrup	10,0000	13
⊞ Confections	Confections	Pavlova	17,4500	29
⊞ Dairy Products	Dairy Products	Queso Cabrales	21,0000	22
⊞ Grains/Cereals	Grains/Cereals	Gustaf's Knäckebröd	21,0000	104
⊞ Meat/Poultry	Meat/Poultry	Mishi Kobe Niku	97,0000	29
⊞ Produce	Produce	Uncle Bob's Organic Dried Pears	30,0000	15
⊞ Seafood	Seafood	Ikura	31,0000	31

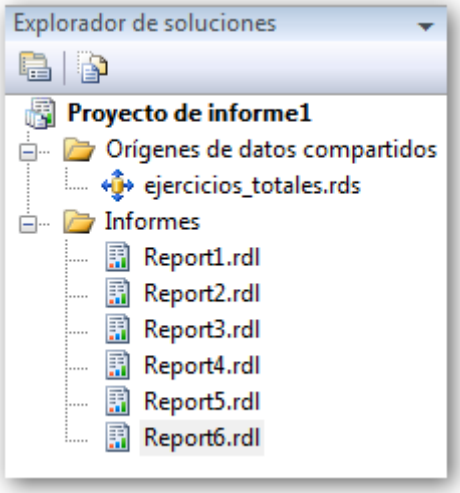
Group1	categoryname	productname	unitprice	Units In Stock
⊟ Beverages			PRECIOS	
	Beverages	Chai	18,0000	39
	Beverages	Chang	19,0000	17
	Beverages	Guaraná Fantástica	4,5000	20
	Beverages	Sasquatch Ale	14,0000	111
	Beverages	Steeleye Stout	18,0000	20
	Beverages	Côte de Blaye	263,5000	17
	Beverages	Chartreuse verte	18,0000	69
	Beverages	Ipoh Coffee	46,0000	17
	Beverages	Laughing Lumberjack Lager	14,0000	52
	Beverages	Outback Lager	15,0000	15
	Beverages	Rhönbräu Klosterbier	7,7500	125
	Beverages	Lakkalikööri	18,0000	57
			455,7500	
⊞ Condiments	Condiments	Aniseed Syrup	10,0000	13
⊞ Confections	Confections	Pavlova	17,4500	29
⊞ Dairy Products	Dairy Products	Queso Cabrales	21,0000	22

Ejemplo 16: Vamos a generar un informe integrado, pero para esto no debe tener ningún parámetro.

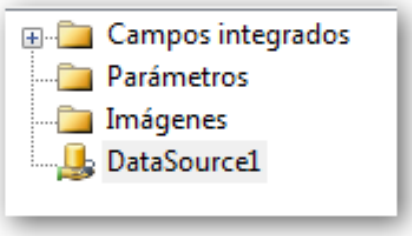
En este caso vamos utilizar la consulta del ejemplo anterior (ejemplo 16) y esta consulta.

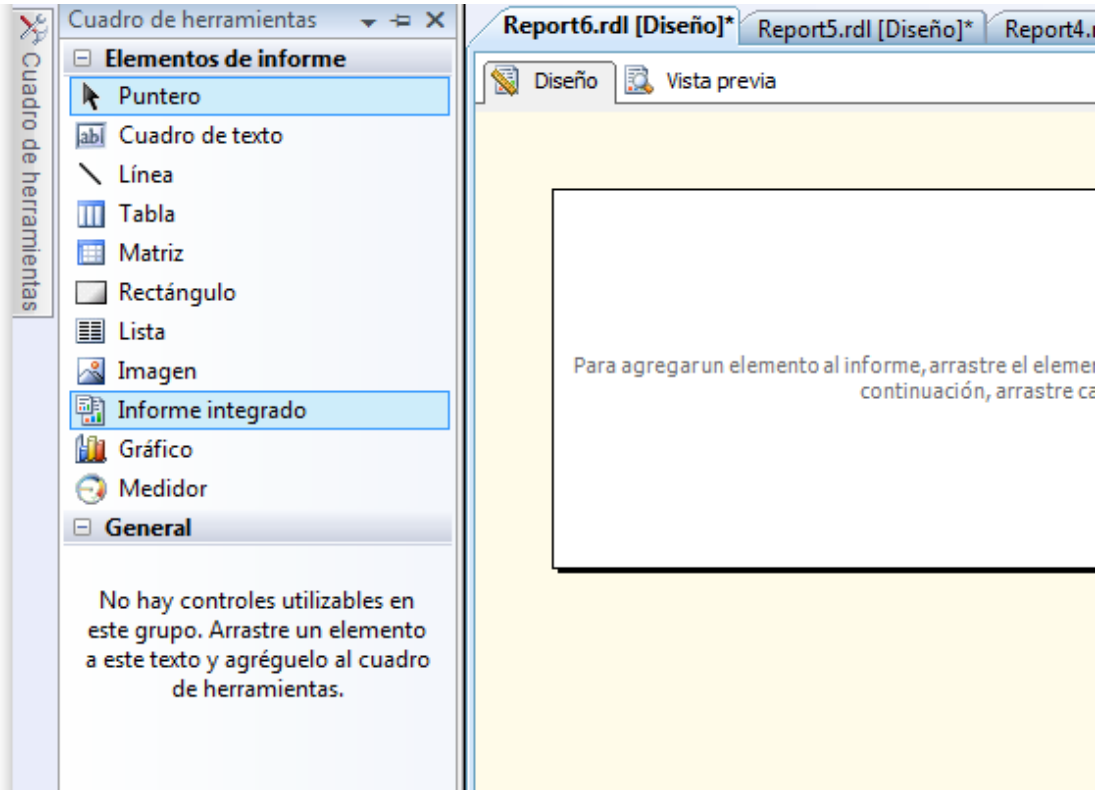
```
select CategoryName, COUNT(*) as total
from Products as p
inner join Categories as c
on p.CategoryID=c.CategoryID
group by CategoryName
```

Hacemos los mismos pasos para crear el informe, nos detenemos cuando queremos jalar una herramienta del cuadro de herramientas. Porque allí vamos elegir la herramienta informe integrado.
Es decir los dos reportes tienen que existir por si solas ya que la herramienta solo las va juntar. Eso lo vemos en el **explorador de soluciones**.

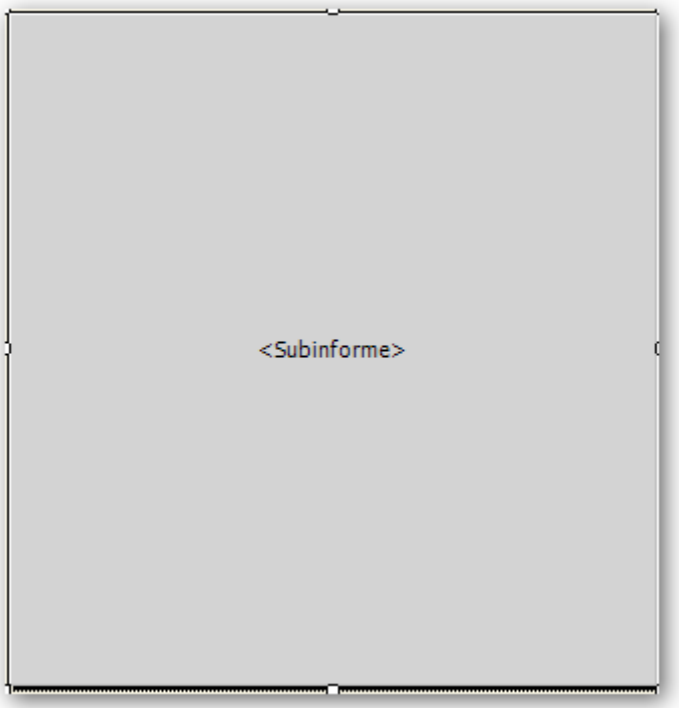


Para nuestro ejemplo se está utilizando el Report4 y Report5, en el Report6 va ser donde se unan los dos Report mencionados.

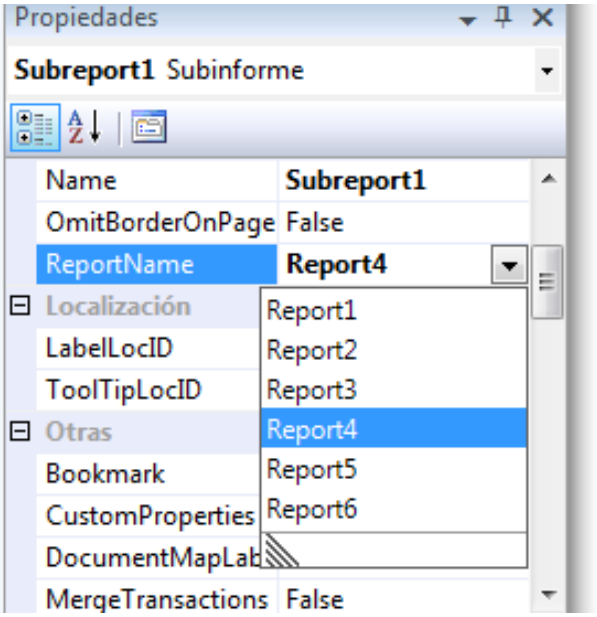


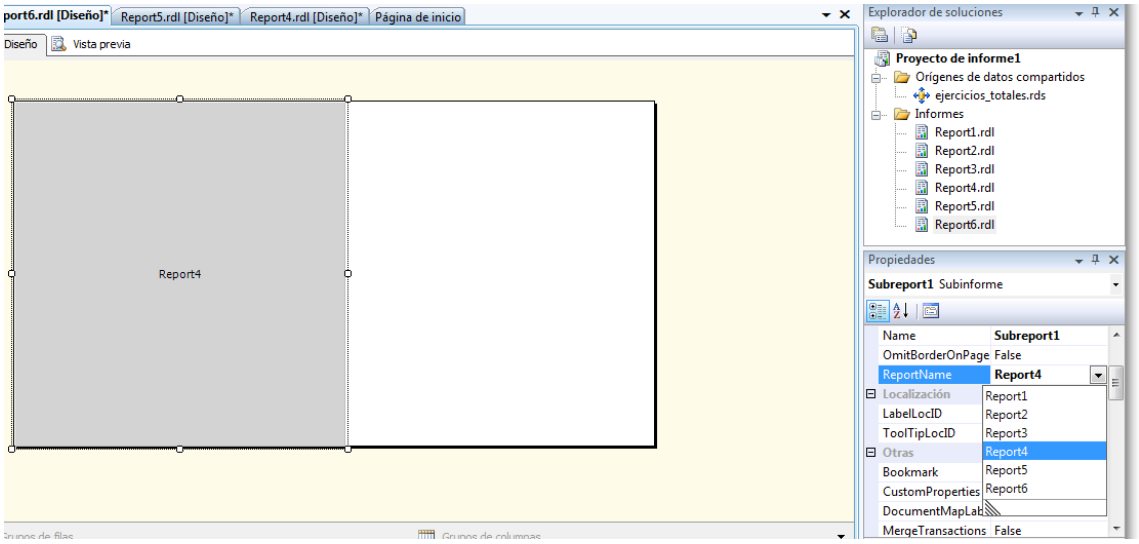


Seleccionamos el **Informe integrado**, arrastramos al área de trabajo o hacemos doble clic.

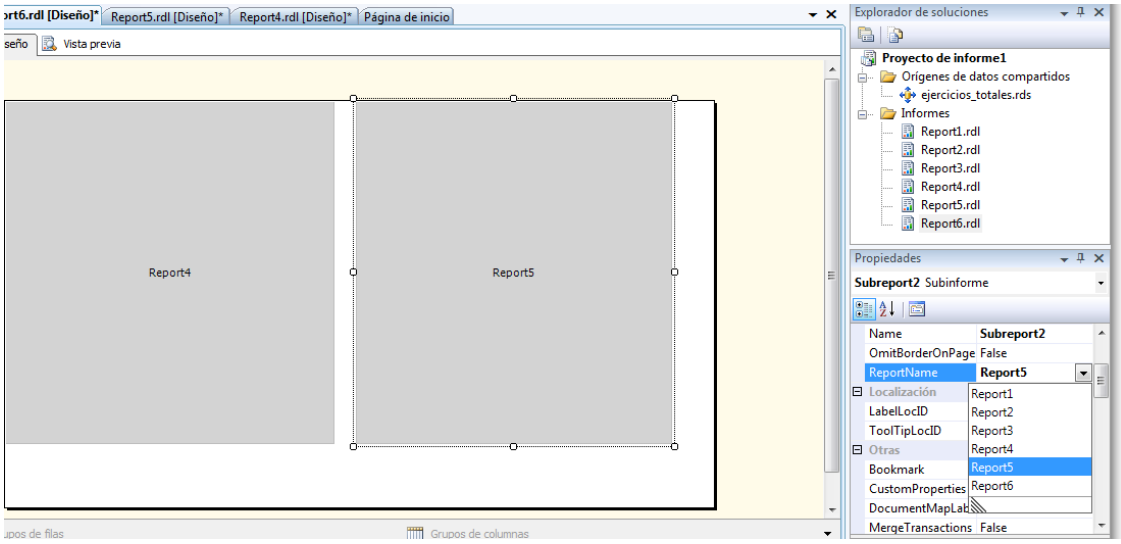


En la ventana de propiedades elegimos con cual **Report** se va enlazar, en este caso es el **Report4**. Propiedad **ReportName**.

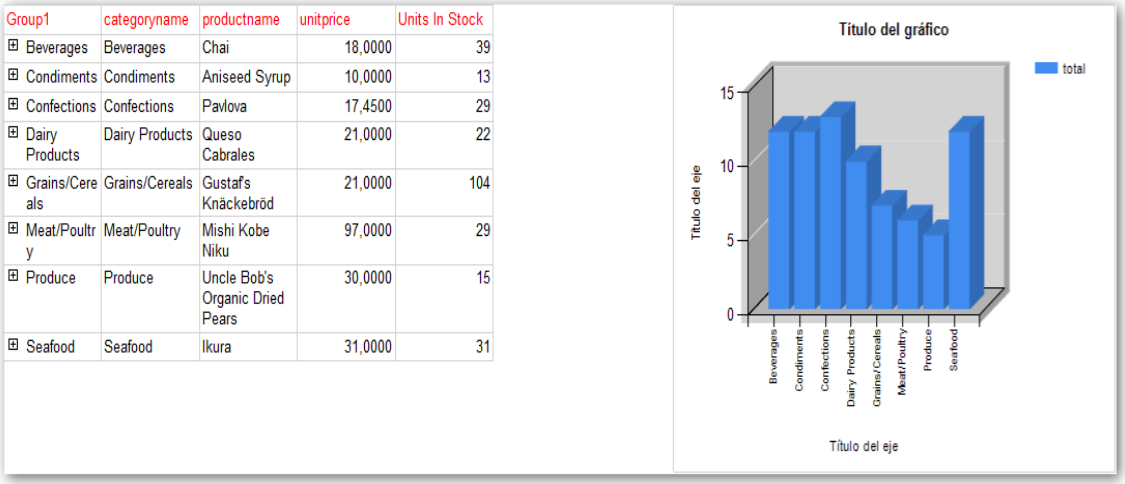




Del mismo modo se hace para el siguiente Informe integrado.



Y al final queda de la siguiente manera



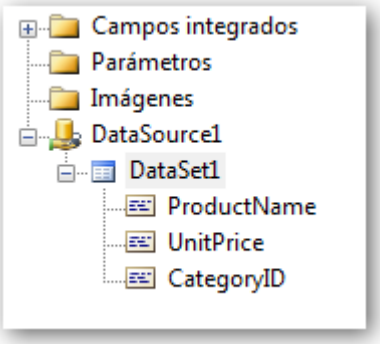
Group1	categoryname	productname	unitprice	Units In Stock
Beverages	Beverages	Chai	18,0000	39
Condiments	Condiments	Aniseed Syrup	10,0000	13
Confections	Confections	Pavlova	17,4500	29
Dairy Products	Dairy Products	Queso Cabrales	21,0000	22
Grains/Cereals	Grains/Cereals	Gustaf's Knäckebröd	21,0000	104
Meat/Poultry	Meat/Poultry	Mishi Kobe Niku	97,0000	29
Produce	Produce	Uncle Bob's Organic Dried Pears	30,0000	15
Seafood	Seafood	Ikura	31,0000	31

Ejemplo 17: Vamos a crear un campo calculado.
Para esto hacemos la siguiente consulta y vamos a crear dos columnas en el cual van hacer los descuentos en porcentaje como en soles de acuerdo a la siguiente regla:
Si pertenece a la categoría:

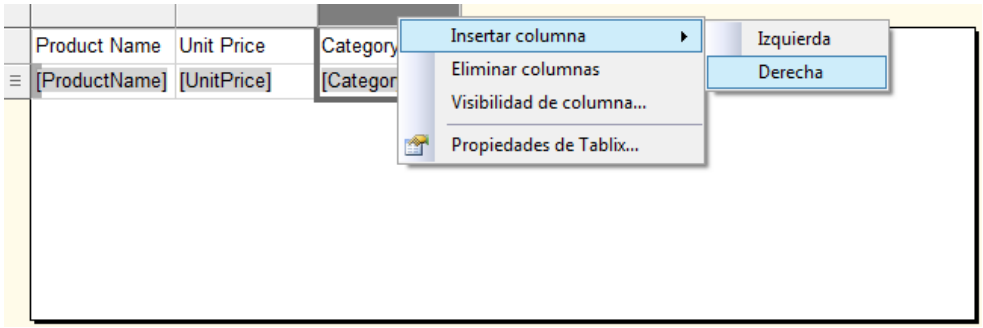
1	el descuento es de	10%
2	el descuento es de	15%
3	el descuento es de	23%
4	el descuento es de	30%
Resto	el descuento es de	0%

La consulta es:

```
select ProductName,UnitPrice,CategoryID
from Products
```

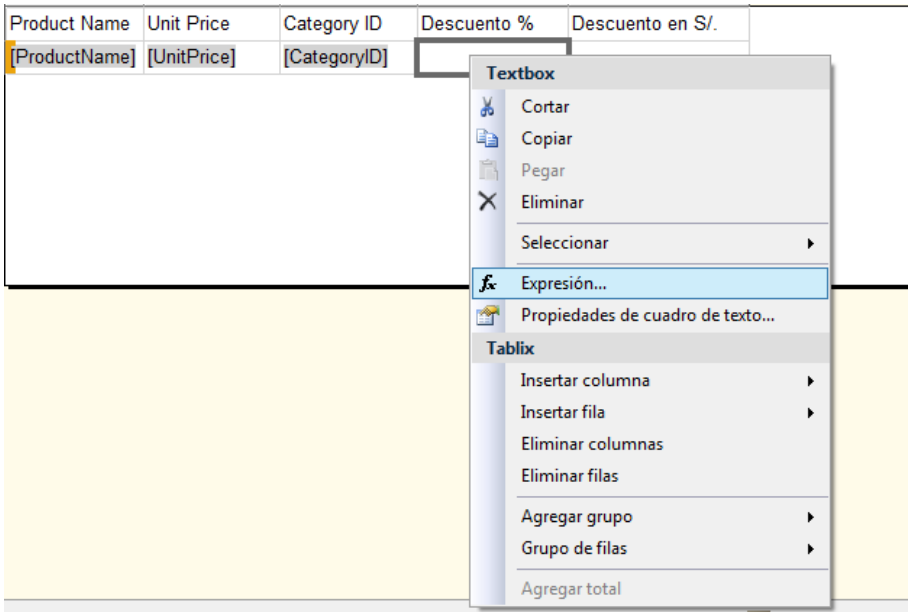


Insertamos la tabla y también insertamos las dos columnas adicionales.



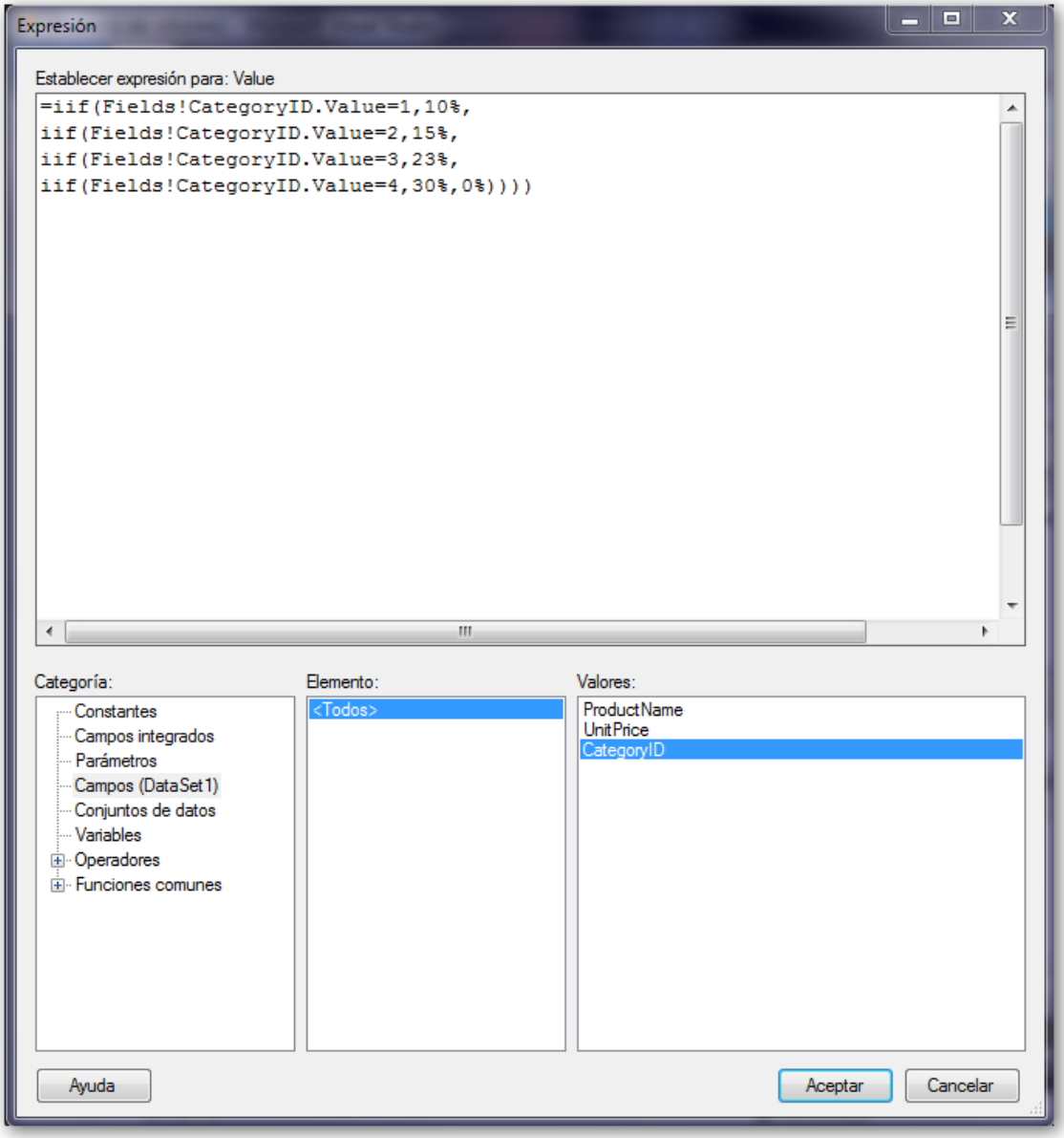
Product Name	Unit Price	Category ID	Descuento %	Descuento en S/.
[ProductName]	[UnitPrice]	[CategoryID]		

Ahora seleccionamos la casilla del **Descuento %**, le damos clic derecho y vamos a expresión.



Se abre una ventana en el cual digitaremos al condición.

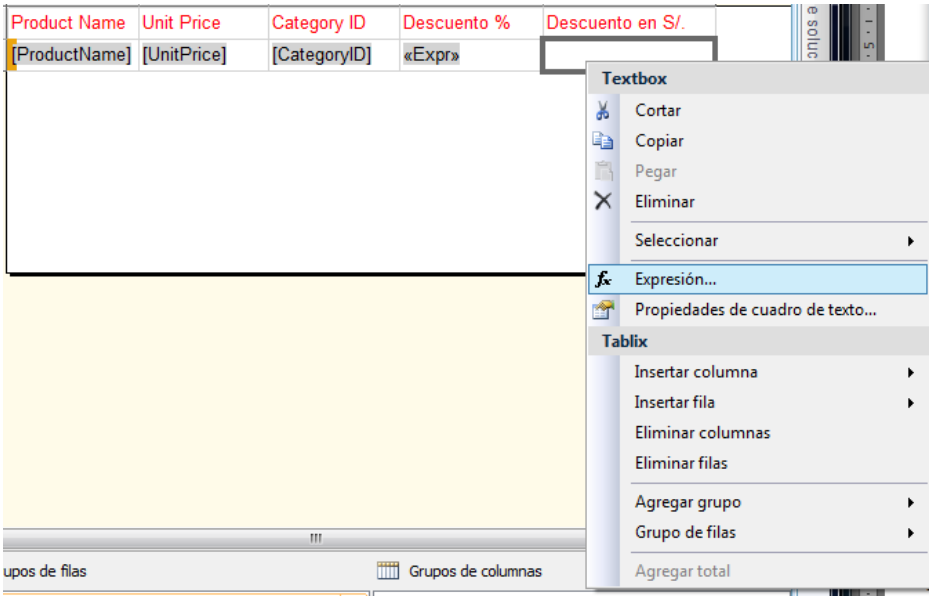
=iif(Fields!CategoryID.Value=1,10%,
iif(Fields!CategoryID.Value=2,15%,
iif(Fields!CategoryID.Value=3,23%,
iif(Fields!CategoryID.Value=4,30%,0%))))



Vamos viendo el campo calculado.

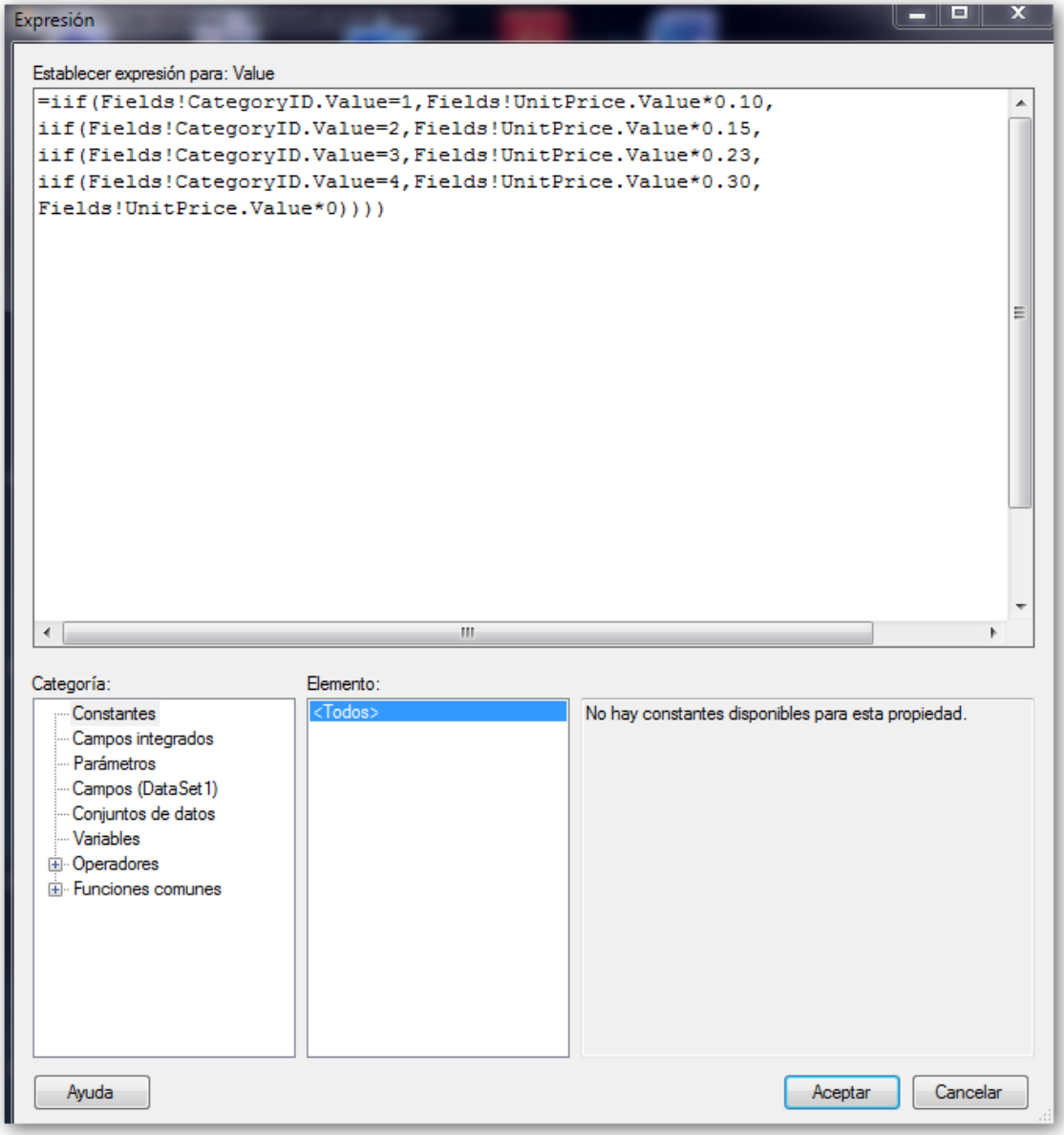
Product Name	Unit Price	Category ID	Descuento %	Descuento en S/.
Chai	18,0000	1	10	
Chang	19,0000	1	10	
Aniseed Syrup	10,0000	2	15	
Chef Anton's Cajun Seasoning	22,0000	2	15	
Chef Anton's Gumbo Mix	21,3500	2	15	
Grandma's Boysenberry Spread	25,0000	2	15	
Uncle Bob's Organic Dried Pears	30,0000	7	0	
Northwoods Cranberry Sauce	40,0000	2	15	
Mishi Kobe Niku	97,0000	6	0	
Ikura	31,0000	8	0	
Queso Cabrales	21,0000	4	30	

Ahora hacemos algo similar para calcula el **Descuento en S/**.



Cuya condición es:

**=iif(Fields!CategoryID.Value=1,Fields!UnitPrice.Value*0.10,
iif(Fields!CategoryID.Value=2,Fields!UnitPrice.Value*0.15,
iif(Fields!CategoryID.Value=3,Fields!UnitPrice.Value*0.23,
iif(Fields!CategoryID.Value=4,Fields!UnitPrice.Value*0.30,
Fields!UnitPrice.Value*0))))**

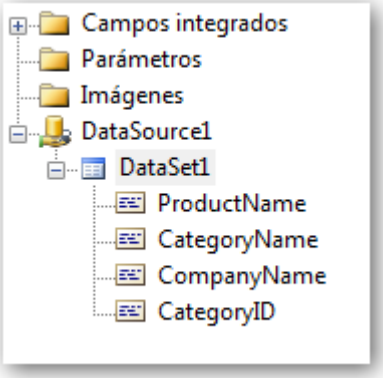


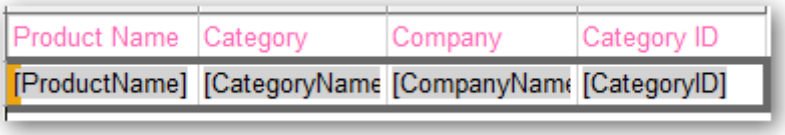
De tal modo que va quedando:

Product Name	Unit Price	Category ID	Descuento %	Descuento en S/.
Chai	18,0000	1	10	1,8
Chang	19,0000	1	10	1,9
Aniseed Syrup	10,0000	2	15	1,5
Chef Anton's Cajun Seasoning	22,0000	2	15	3,3
Chef Anton's Gumbo Mix	21,3500	2	15	3,2025
Grandma's Boysenberry Spread	25,0000	2	15	3,75
Uncle Bob's Organic Dried Pears	30,0000	7	0	0,0000
Northwoods Cranberry Sauce	40,0000	2	15	6
Mishi Kobe Niku	97,0000	6	0	0,0000
Ikura	31,0000	8	0	0,0000
Queso Cabrales	21,0000	4	30	6,3

Ejemplo 18: Mostrar el nombre del producto, nombre de la categoría y nombre de la empresa proveedora. Mostrar toda la fila de un registro en color amarillo si son de las categorías 1 y 2, en color verde las categorías 3 y 4, en color azul las categorías 5 y 6, y en color rojo el resto de categorías.

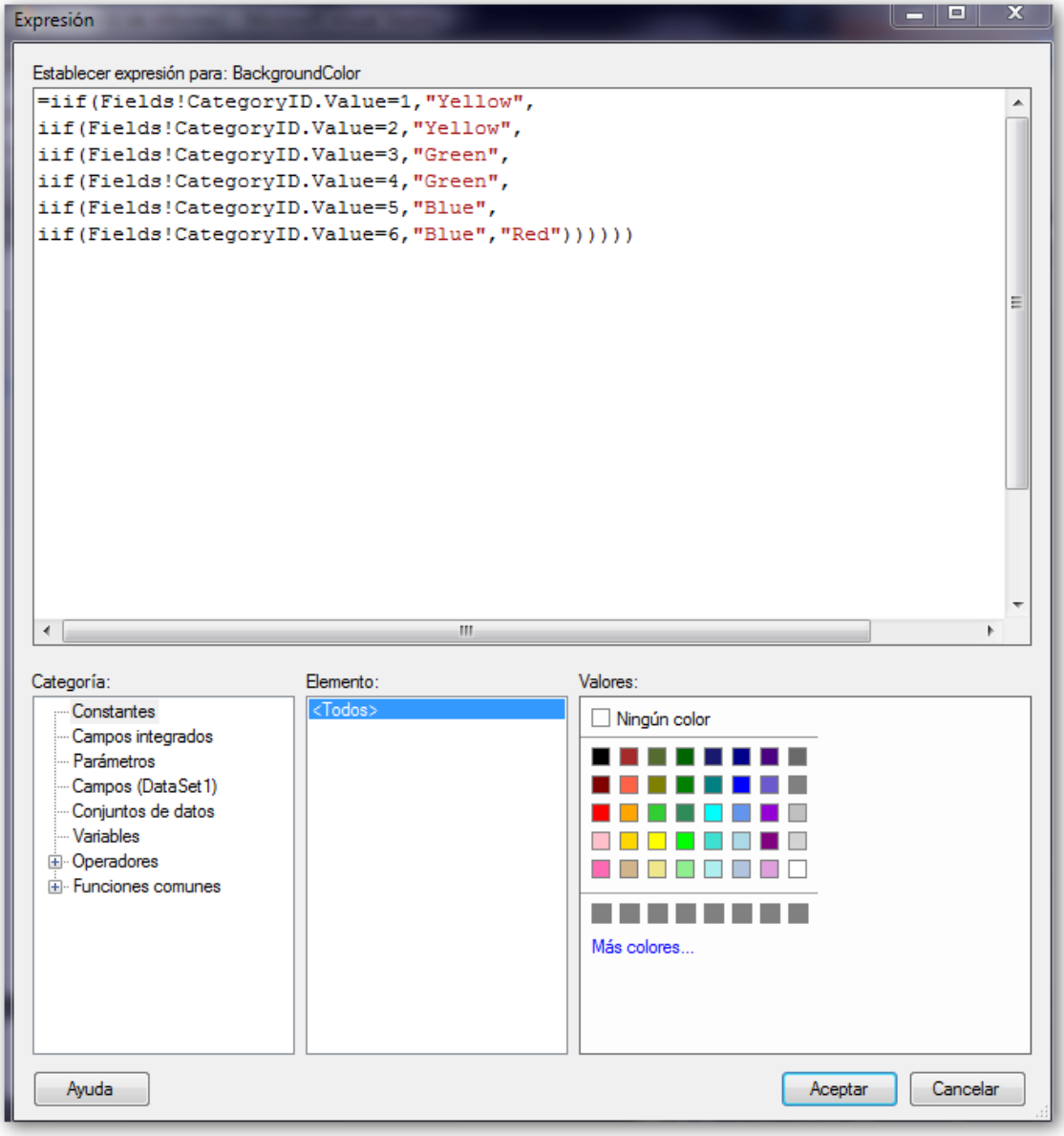
```
create proc ejercicio8
as
select ProductName,c.CategoryName,s.CompanyName,c.CategoryID
from Products as p
inner join [Order Details] as od
on p.ProductID=od.ProductID
inner join Orders as o
on od.OrderID=o.OrderID
inner join Categories as c
on c.CategoryID=p.CategoryID
inner join Suppliers as s
on s.SupplierID=p.SupplierID
```





Seleccionamos la fila como se muestra en la imagen, luego vamos a las propiedades y elegimos la propiedad **BackgroundColor**, le damos clic y seleccionamos expresión. Digitamos la condición:

```
=iif(Fields!CategoryID.Value=1,"Yellow",
iif(Fields!CategoryID.Value=2,"Yellow",
iif(Fields!CategoryID.Value=3,"Green",
iif(Fields!CategoryID.Value=4,"Green",
iif(Fields!CategoryID.Value=5,"Blue",
iif(Fields!CategoryID.Value=6,"Blue","Red"))))))
```


Expresión
Establecer expresión para: BackgroundColor
=iif(Fields!CategoryID.Value=1,"Yellow",
iif(Fields!CategoryID.Value=2,"Yellow",
iif(Fields!CategoryID.Value=3,"Green",
iif(Fields!CategoryID.Value=4,"Green",
iif(Fields!CategoryID.Value=5,"Blue",
iif(Fields!CategoryID.Value=6,"Blue","Red"))))))
Categoría:
Constantes
Campos integrados
Parámetros
Campos (DataSet1)
Conjuntos de datos
Variables
Operadores
Funciones comunes
Elemento:
<Todos>
Valores:
Ningún color
Más colores...
Ayuda
Aceptar
Cancelar

Y se ve así:

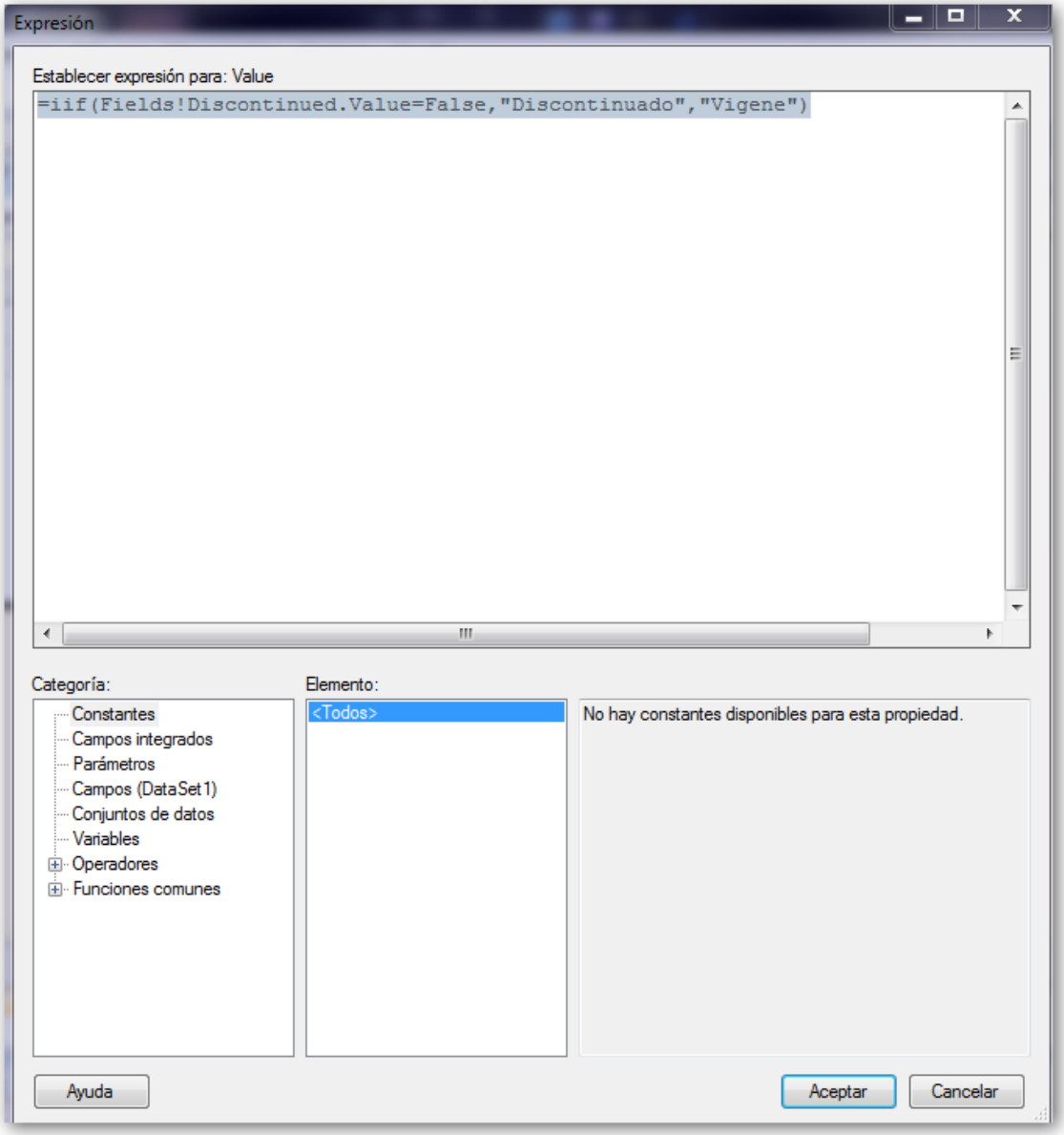
Northwoods Cranberry Sauce	Condiments	Grandma Kelly's Homestead	2
Northwoods Cranberry Sauce	Condiments	Grandma Kelly's Homestead	2
Northwoods Cranberry Sauce	Condiments	Grandma Kelly's Homestead	2
Mishi Kobe Niku	Meat/Poultry	Tokyo Traders	6
Mishi Kobe Niku	Meat/Poultry	Tokyo Traders	6
Mishi Kobe Niku	Meat/Poultry	Tokyo Traders	6
Mishi Kobe Niku	Meat/Poultry	Tokyo Traders	6
Mishi Kobe Niku	Meat/Poultry	Tokyo Traders	6
Ikura	Seafood	Tokyo Traders	8
Ikura	Seafood	Tokyo Traders	8
Ikura	Seafood	Tokyo Traders	8
Ikura	Seafood	Tokyo Traders	8

Ejemplo 19: Mostrar la siguiente tabla, añadiendo que en el campo Discontinuado si el valor es 1 salga discontinuado y si sale 0 se escriba como vigente.

```
CREATE PROC EJERCICIO9
AS
SELECT ProductID,ProductName,UnitPrice,Discontinued
FROM Products
```

Condición.

=iif(Fields!Discontinued.Value=False,"Discontinuado","Vigene")


Expresión
Establecer expresión para: Value
=iif(Fields!Discontinued.Value=False,"Discontinuado","Vigene")
Categoría:
Constantes
Campos integrados
Parámetros
Campos (DataSet1)
Conjuntos de datos
Variables
Operadores
Funciones comunes
Elemento:
<Todos>
No hay constantes disponibles para esta propiedad.
Ayuda
Aceptar
Cancelar

Sale:

Product ID	Product Name	Unit Price	Discontinued
1	Chai	18,0000	Discontinuado
2	Chang	19,0000	Discontinuado
3	Aniseed Syrup	10,0000	Discontinuado
4	Chef Anton's Cajun Seasoning	22,0000	Discontinuado
5	Chef Anton's Gumbo Mix	21,3500	Vigene
6	Grandma's Boysenberry Spread	25,0000	Discontinuado
7	Uncle Bob's Organic Dried Pears	30,0000	Discontinuado
8	Northwoods Cranberry Sauce	40,0000	Discontinuado
9	Mishi Kobe Niku	97,0000	Vigene
10	Ikura	31,0000	Discontinuado
11	Queso Cabrales	21,0000	Discontinuado
12	Queso Manchego La Pastora	38,0000	Discontinuado

Ejemplo 20: Mostrar el código y nombre de los empleados, luego enlazar los informes de tal manera que al dar click en el nombre del empleado, se abra otro informe en el cual se pueda visualizar una gráfica con las ventas del empleado por año, mostrándose el nombre del empleado.

La primera consulta es:

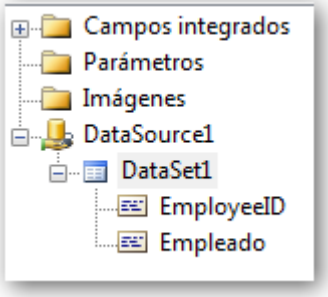
```
alter proc informe1
as
SELECT EmployeeID, (FirstName + ' '+ LastName) as Empleado
FROM Employees
order by EmployeeID
```

La segunda consulta es:

```
alter proc informe2
@ID INT
AS
```

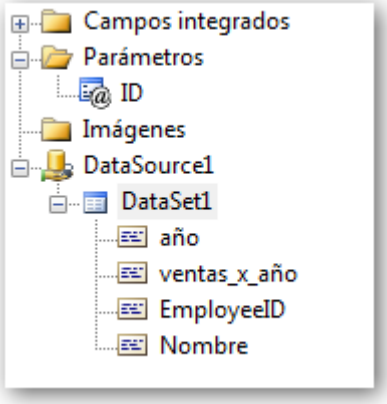
```
SELECT YEAR(O.OrderDate) as año, COUNT(*) as ventas_x_año, e.EmployeeID,
(E.firstname +' '+e.Lastname) as Nombre
FROM Products AS P
INNER JOIN [Order Details] AS OD
ON P.ProductID=OD.ProductID
INNER JOIN Orders AS O
ON O.OrderID=OD.OrderID
INNER JOIN Employees AS E
ON E.EmployeeID=O.EmployeeID
WHERE E.EmployeeID=@ID
GROUP BY YEAR(O.OrderDate), E.EmployeeID,E.FirstName,E.LastName
ORDER BY YEAR(O.OrderDate)
```

Creamos un primer informe con la primera consulta.



Employee ID	Empleado
[EmployeeID]	[Empleado]

Ahora bien queremos que cuando hagamos clic en empleado nos lleve a una segunda consulta, pero antes debemos crearla. Usamos la segunda consulta.

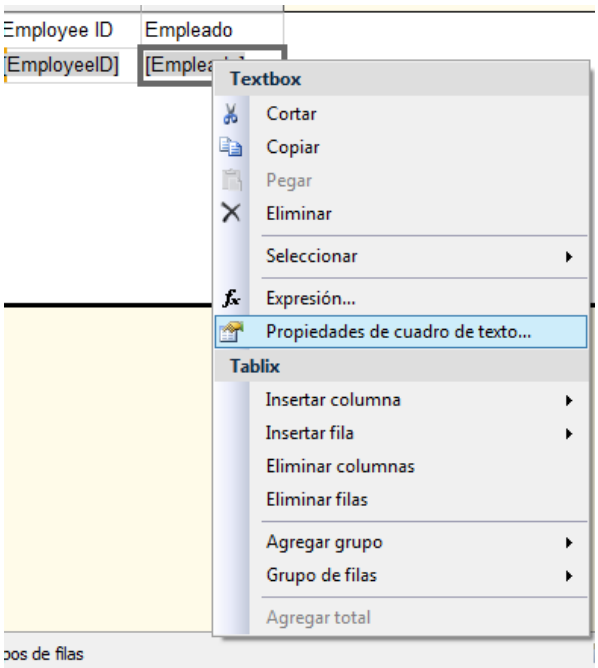


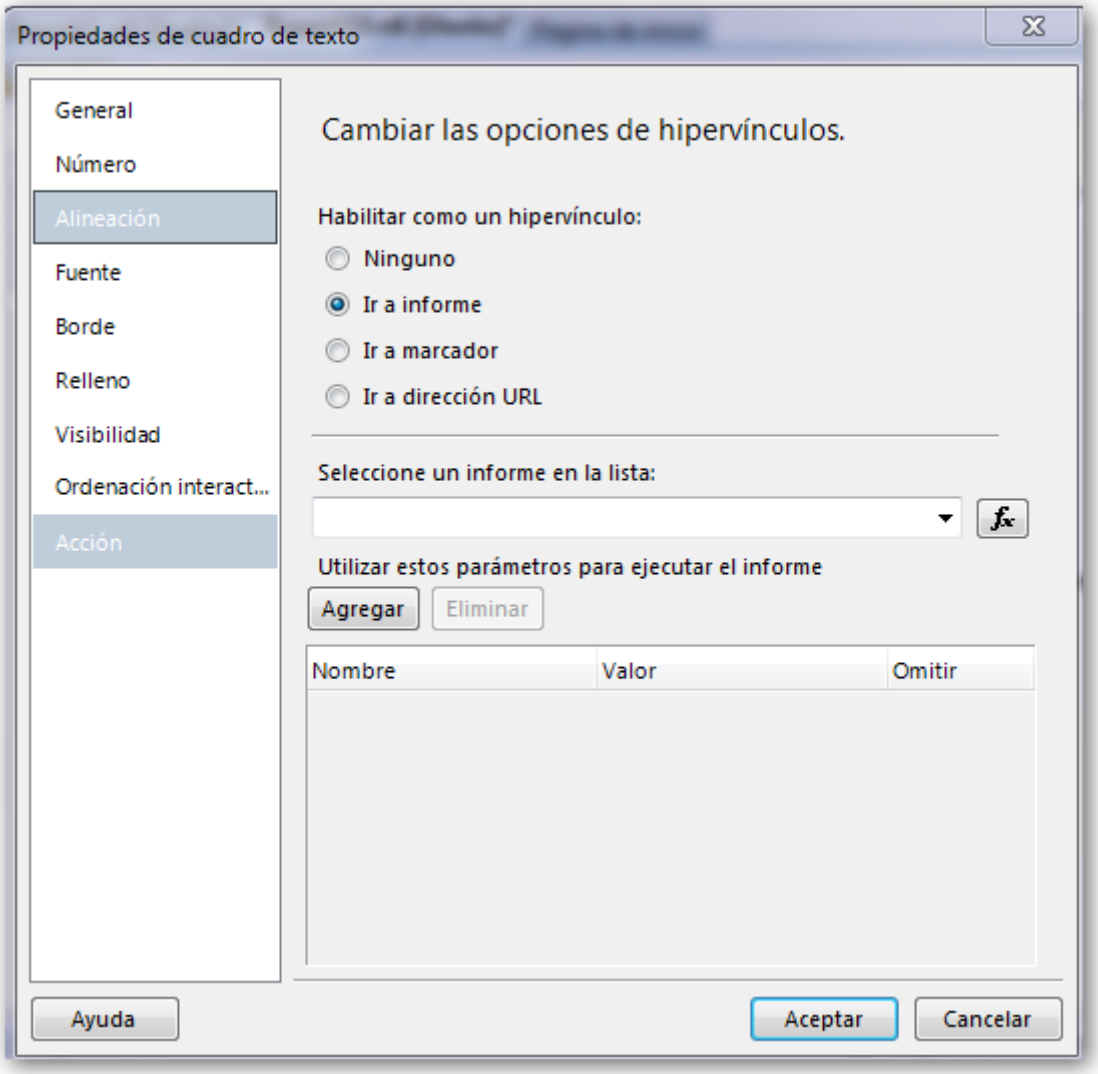
Employee ID	Nombre	año	ventas x año
[EmployeeID]	[Nombre]	[año]	[ventas_x_año]

Una vez creada el segundo informe, regresamos al **informe1**, ya que es el empleado que se va enlazar al hacer clic, con el **infome2**.

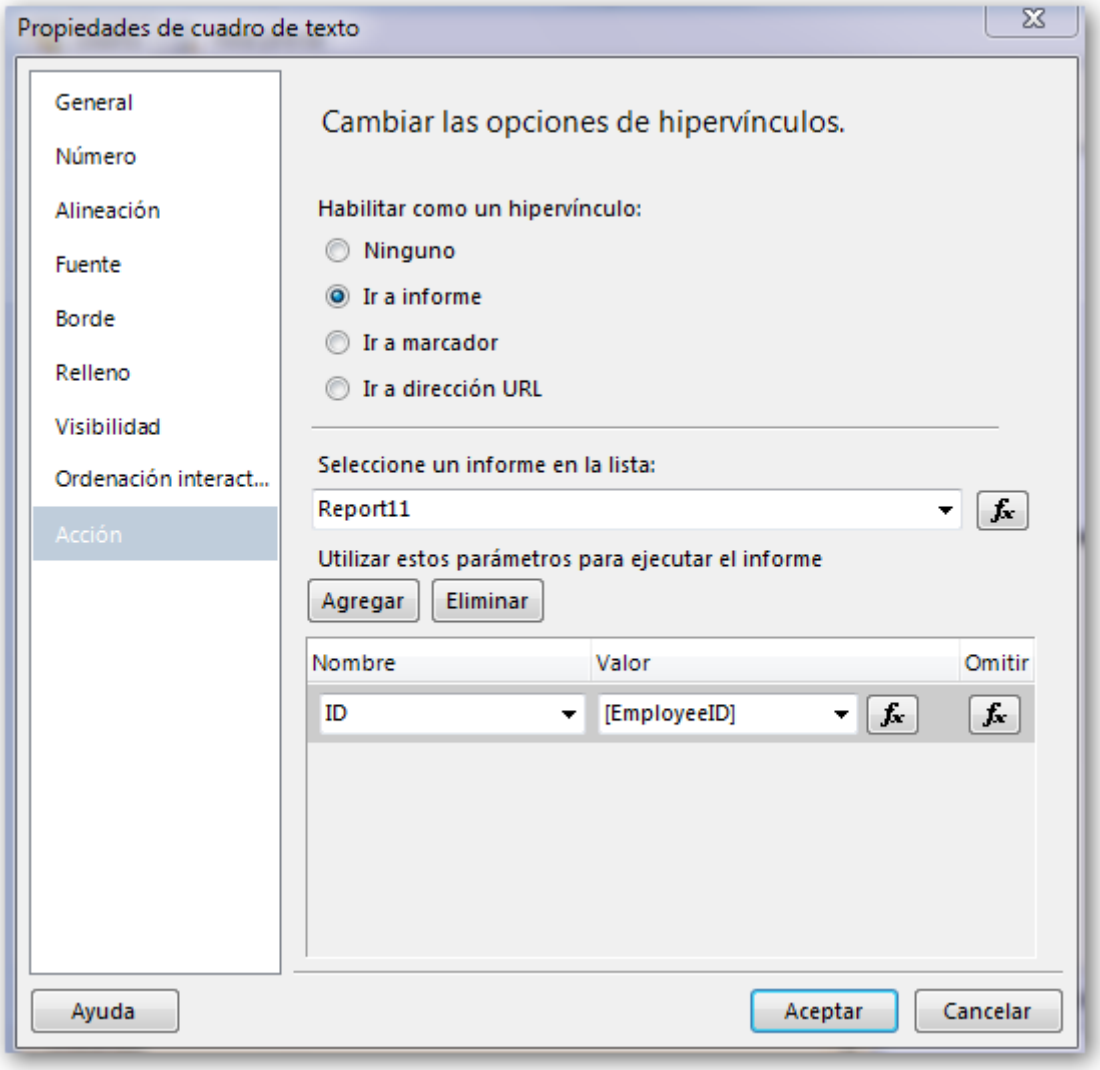
Employee ID	Empleado
[EmployeeID]	[Empleado]

Seleccionamos la celda donde dice **[Empleado]**, le damos clic derecho y hacemos clic en **Propiedades de cuadro de texto**, luego vamos a **Acciones** y elegimos la opción **Ir a informe**.





Seleccionamos el Report que queremos que se enlace en este caso es el **Repor11** (Seleccione un informe en la lista – en este ejemplo estamos trabajando con el **Report10** y el **Report11**). Luego agregamos los parámetros con los que está trabajando el **Report11**, y lo filtramos.



Aceptamos y ponemos a prueba

Employee ID	Empleado
1	Nancy Davolio
2	Andrew Fuller
3	Janet Leverling
4	Margaret Peacock
5	Steven Buchanan
6	Michael Suyama
7	Robert King
8	Laura Callahan
9	Anne Dodsworth

Cuando pasamos el cursor sobre alguno de los empleados, se convierte en manito, esto quiere decir que tiene un Hipervinvulo. Al hacerle clic nos lleva al informe enlazado. Vamos darle clic en **Jane Leverling**.

ID 3

1 de 1

Employee ID	Nombre	año	ventas x año
3	Janet Leverling	1996	43
3	Janet Leverling	1997	184
3	Janet Leverling	1998	94

Nos ha llevado al informe enlazado, filtrándolo rápidamente.

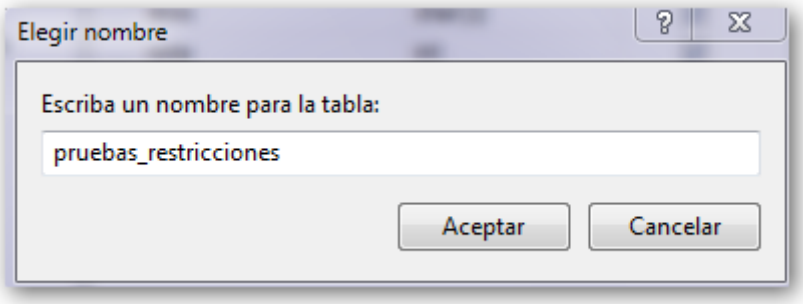
RESTRICCIONES EN SQL

En nuestro caso vamos a crear una nueva tabla llamada pruebas_restricciones, con los siguientes campos:

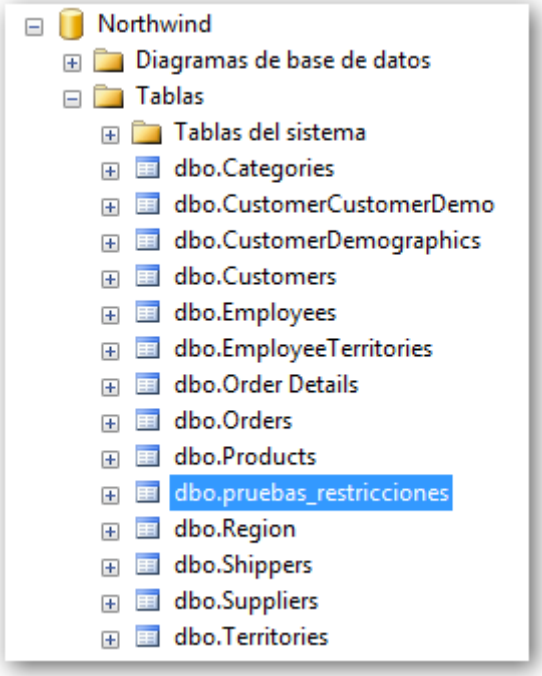
Primarykey	Código	Char(5)
	Nombres	Varchar(30)
	Correo	Varchar(30)
	Celular	Varchar(11)
	Sexo	Char(1)
	Nota	Int
	Sueldo	money

Como vemos en las imágenes.

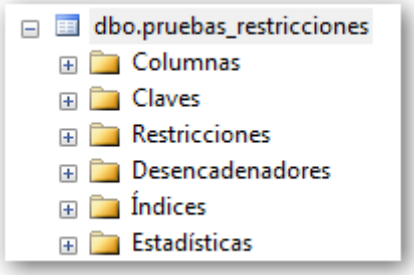
Nombre de columna	Tipo de datos	Permitir v...
codigo	char(5)	☐
nombres	varchar(30)	☑
correo	varchar(30)	☑
celular	varchar(11)	☑
sexo	char(1)	☑
nota	int	☑
sueldo	money	☑



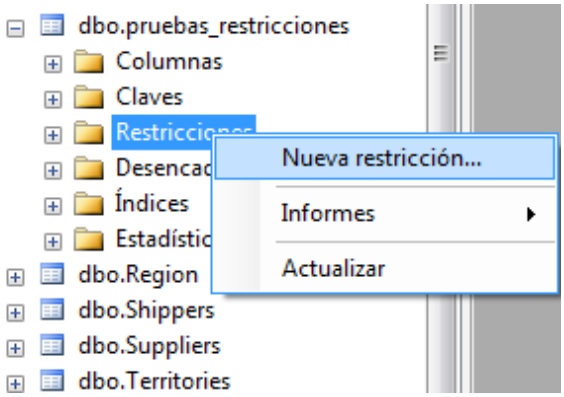
Lo verificamos en la base datos Northwind.



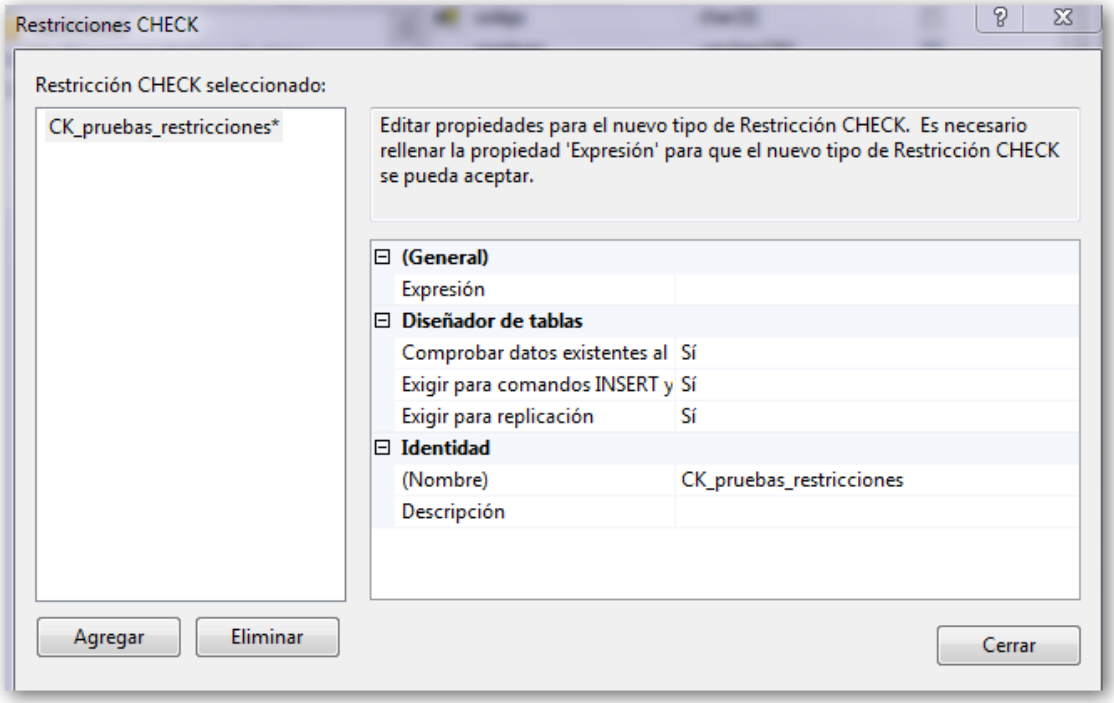
Desplegamos en el signo más (+) de la tabla creada.



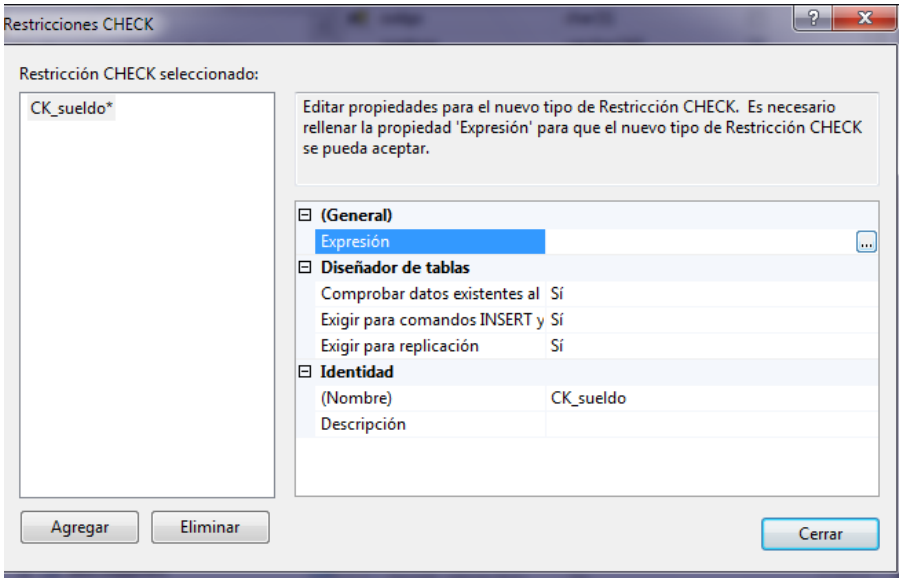
Seleccionamos la opción **Restricciones**, le damos clic derecho y elegimos **Nueva restricción**.



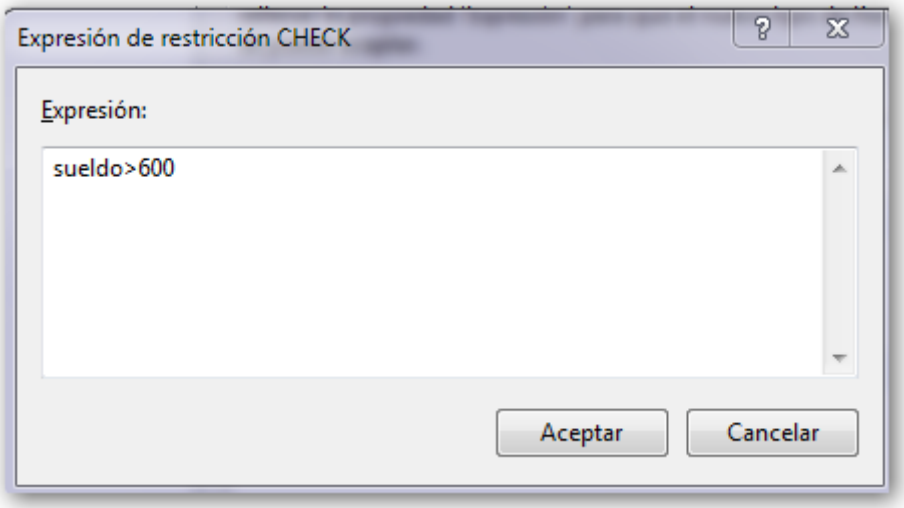
Luego se abre la ventana en donde vamos a dar la restricción. En este caso le cambiamos el nombre.



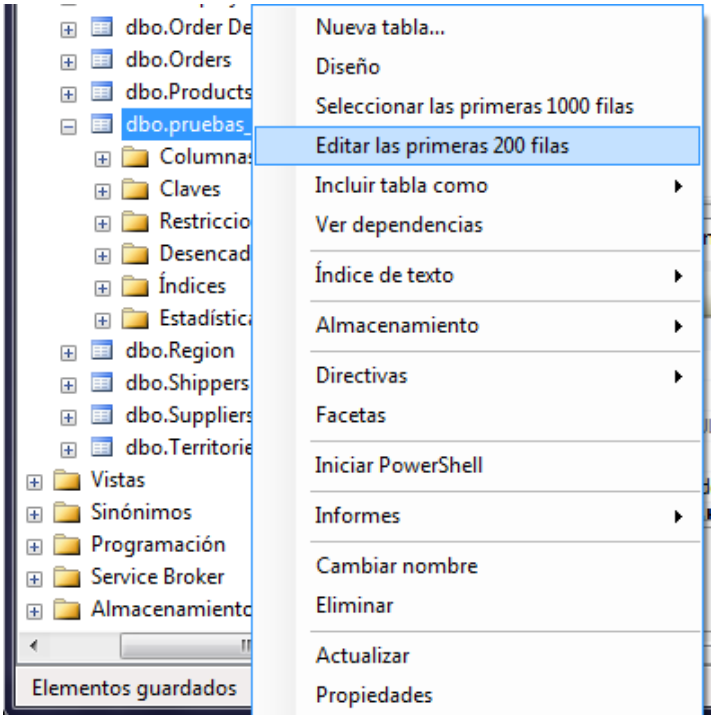
- Vamos a restringir al sueldo que sea mayor a 650, si cumple se podrá ingresar el dato caso contrario no.
  Nombre de la restricción: **CK_sueldo**
  Nos vamos a la opción expresión y condicionamos.



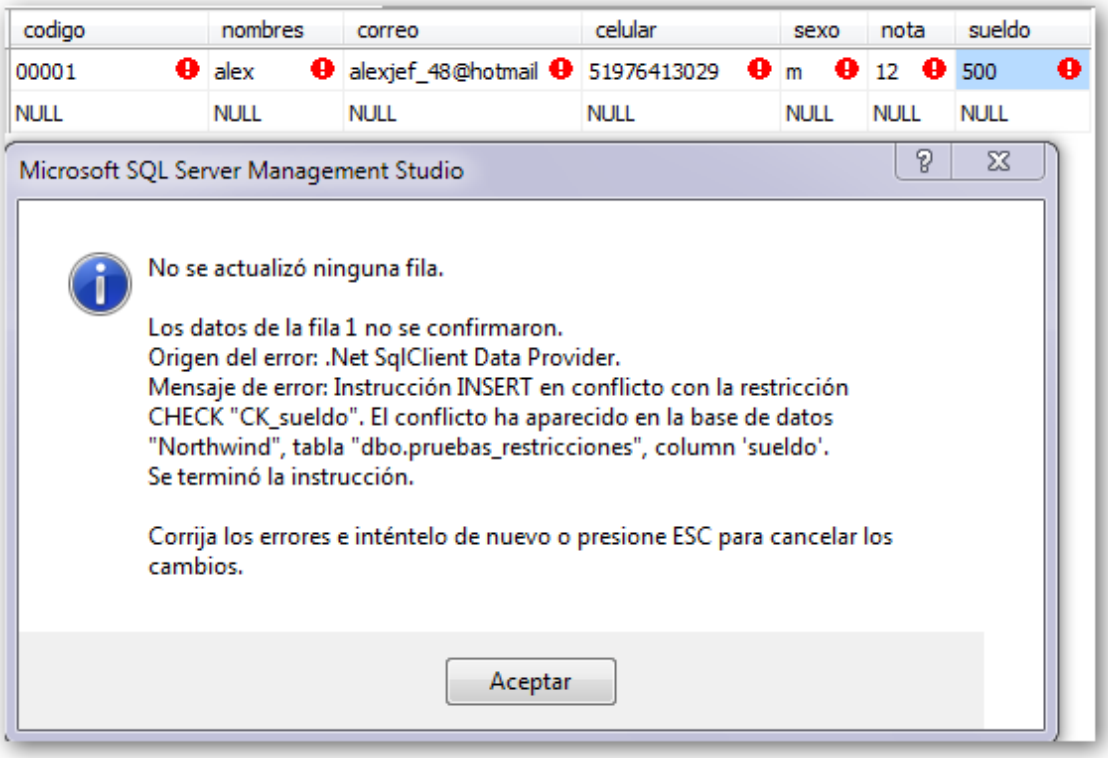
Cuando ingresamos a la expresión y poder condicionarlo debemos tener en cuenta el nombre del campo.



Acepamos todos los cambios guardamos los cambios, para que haga efecto en la tabla. Y luego insertamos algunos datos.



Como la restricción no dice que no permitirá ingresar datos en el campo sueldo menores a 600.



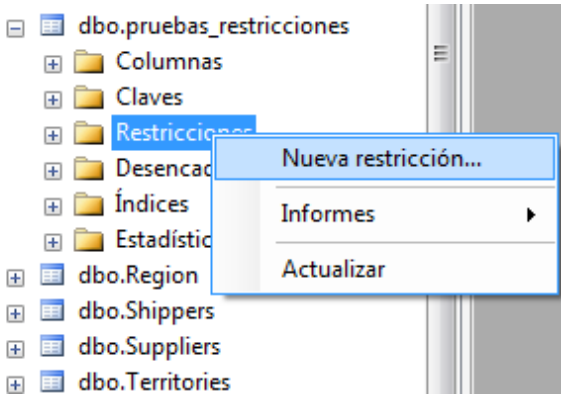
Y ahora si ingresamos en el campo sueldo mayor a 600, eso sí nos permite.

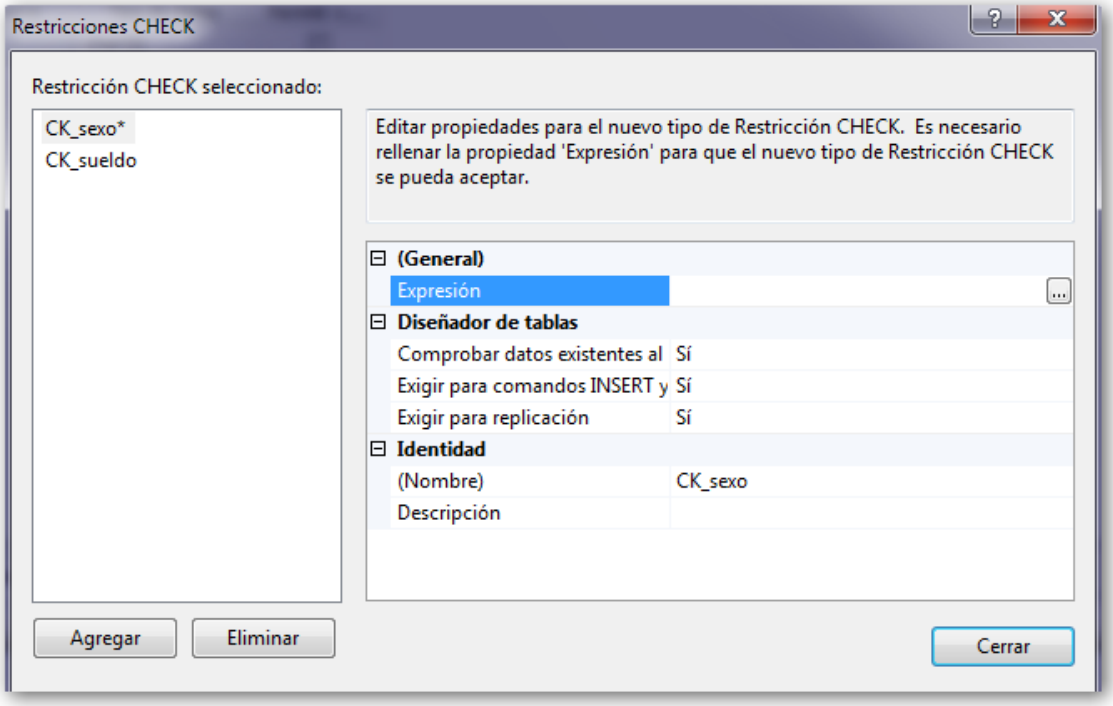
codigo	nombres	correo	celular	sexo	nota	sueldo
00001	alex	alexjef_48@hotmail	51976413029	m	12	700,0000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

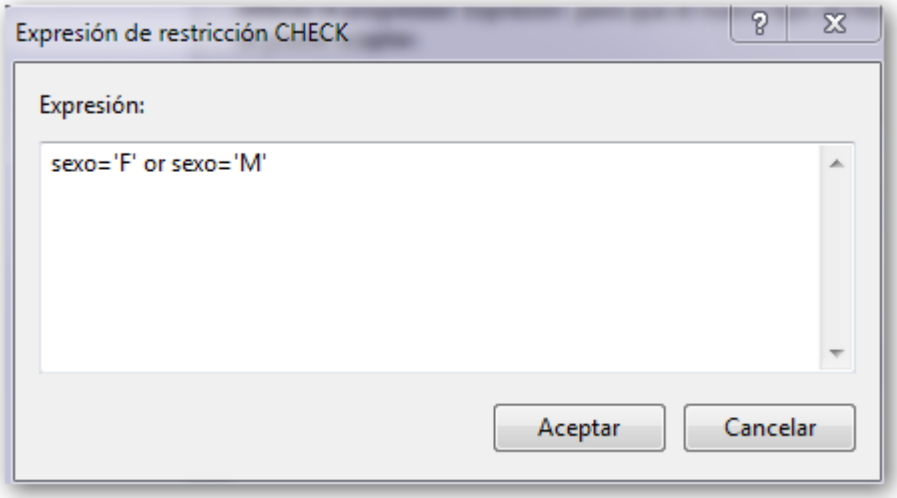
- Vamos a restringir al sexo que solamente sea **M** o **F** y no otra letra.
  Para lo cual se procede del mismo modo que se ha mencionado, para crear la restricción.

  Nota: Ahora bien si tenemos datos ingresados y queremos colocar la restricción, el dato ingresado debe cumplir con restricción que se va hacer, de lo contrario no se va poder hacer la restricción. De ser así se tiene que eliminar los datos. En nuestro caso vamos eliminar.

codigo	nombres	correo	celular	sexo	nota	sueldo
NULL	NULL	NULL	NULL	NULL	NULL	NULL







También puedes utilizarse: **sexo in ('F','M')**
Aceptamos y guardamos para que haga efecto.

codigo	nombres	correo	celular	sexo	nota	sueldo
00001	Alex	alexjef_48@hotmail..	51948754667	T	12	700
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Microsoft SQL Server Management Studio

No se actualizó ninguna fila.

Los datos de la fila 1 no se confirmaron.
Origen del error: .Net SqlClient Data Provider.
Mensaje de error: Instrucción INSERT en conflicto con la restricción CHECK "CK_sexo". El conflicto ha aparecido en la base de datos "Northwind", tabla "dbo.pruebas_restricciones", column 'sexo'.
Se terminó la instrucción.

Corrija los errores e inténtelo de nuevo o presione ESC para cancelar los cambios.

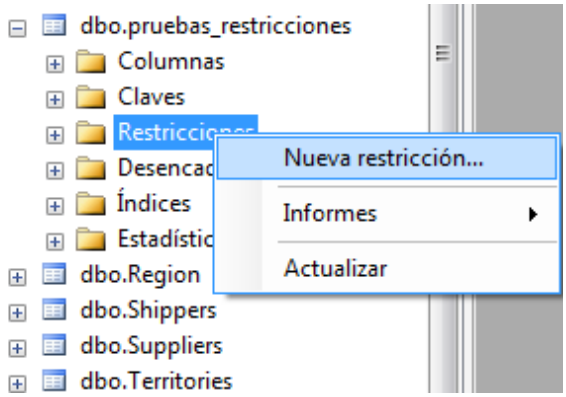
Aceptar

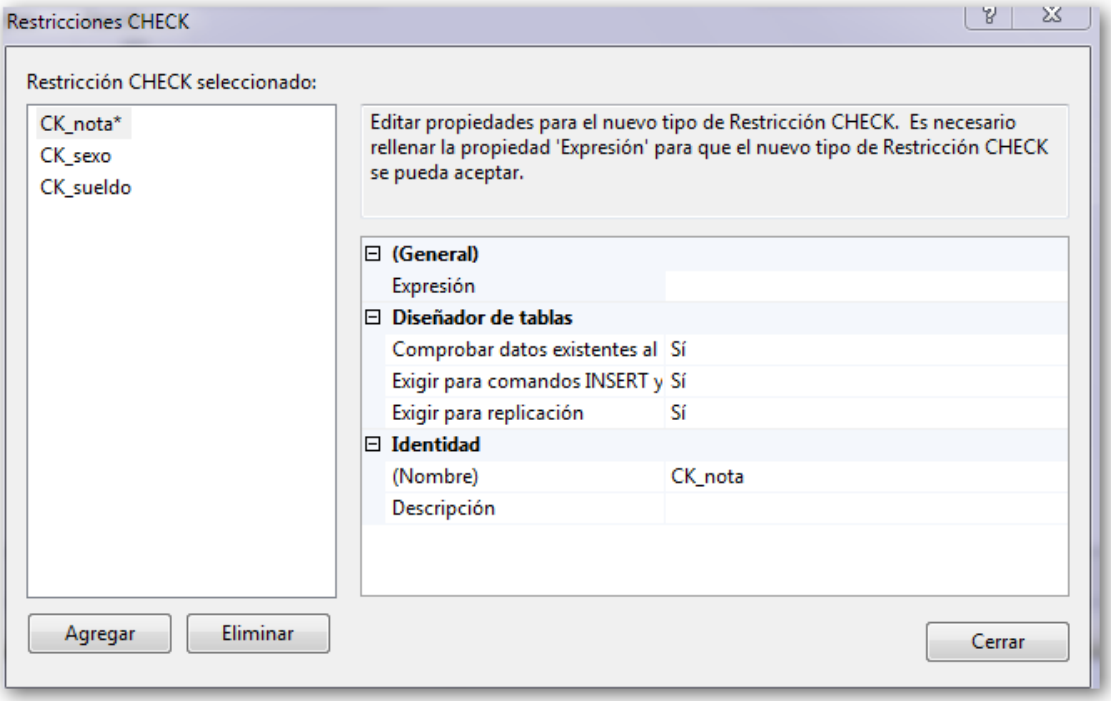
Nos genera un error por lo que está restringido el campo seso. Y en cambio sí colocamos según la restricción, si nos acepta.

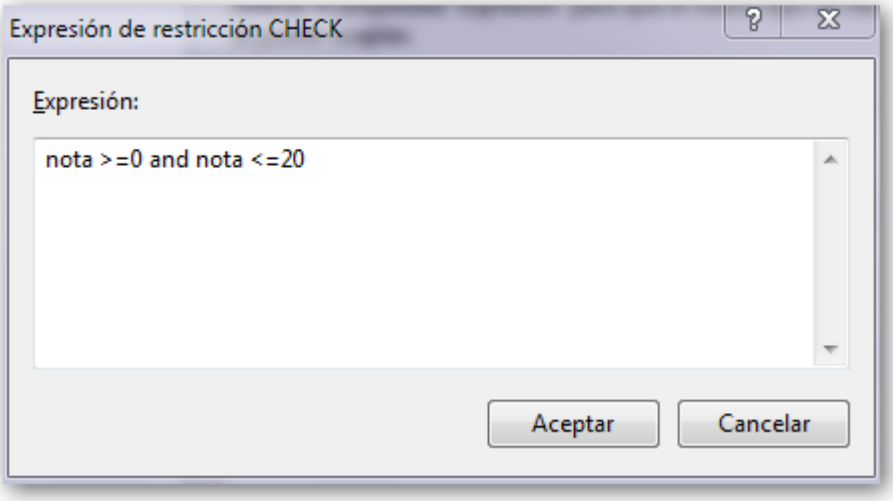
codigo	nombres	correo	celular	sexo	nota	sueldo
00001	Alex	alexjef_48@hotmail....	51948754667	M	12	700,0000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Vamos a restringir la nota que se de 0 – 20.

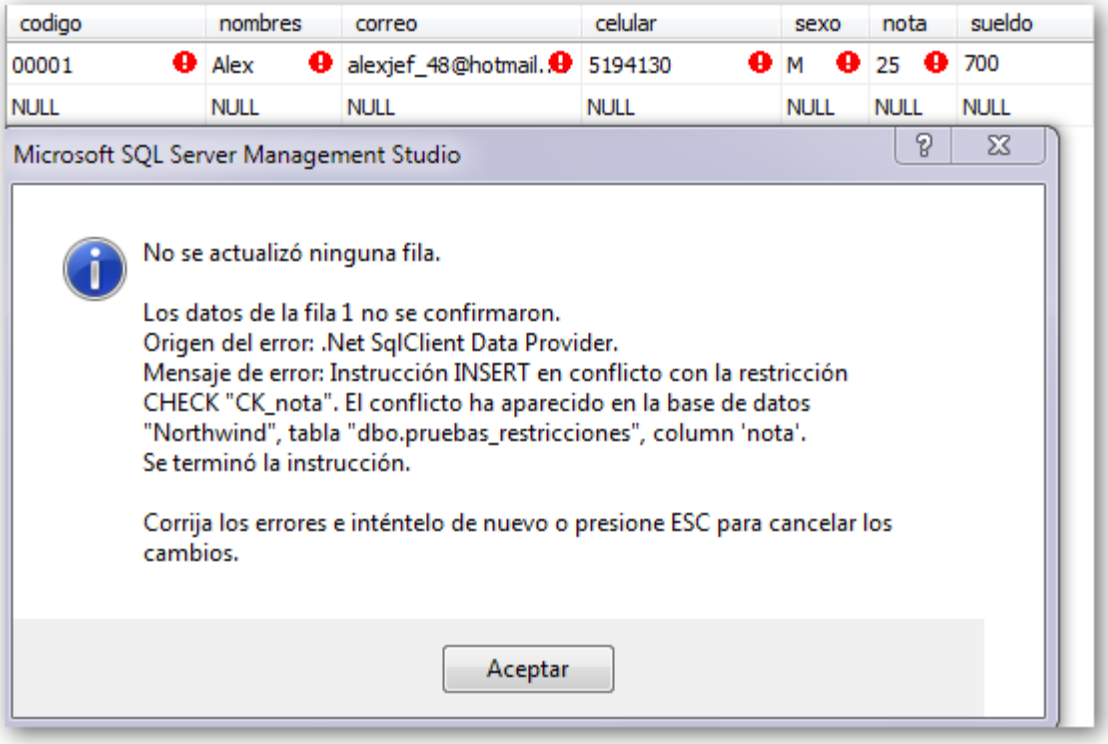
codigo	nombres	correo	celular	sexo	nota	sueldo
NULL	NULL	NULL	NULL	NULL	NULL	NULL

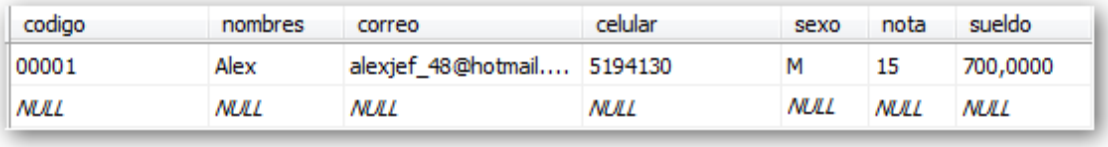

dbo.pruebas_restricciones
Columnas
Claves
Restricciones
Desencad
Índices
Estadístic
dbo.Region
dbo.Shippers
dbo.Suppliers
dbo.Territories
Nueva restricción...
Informes
Actualizar


Restricciones CHECK
Restricción CHECK seleccionado:
CK_nota*
CK_sexo
CK_sueldo
Editar propiedades para el nuevo tipo de Restricción CHECK. Es necesario rellenar la propiedad 'Expresión' para que el nuevo tipo de Restricción CHECK se pueda aceptar.
(General)
Expresión
Diseñador de tablas
Comprobar datos existentes al Sí
Exigir para comandos INSERT y Sí
Exigir para replicación Sí
Identidad
(Nombre) CK_nota
Descripción
Agregar
Eliminar
Cerrar

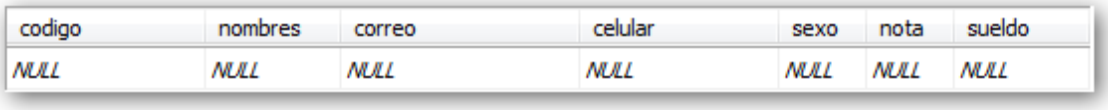

Expresión de restricción CHECK
Expresión:
nota >=0 and nota <=20
Aceptar
Cancelar

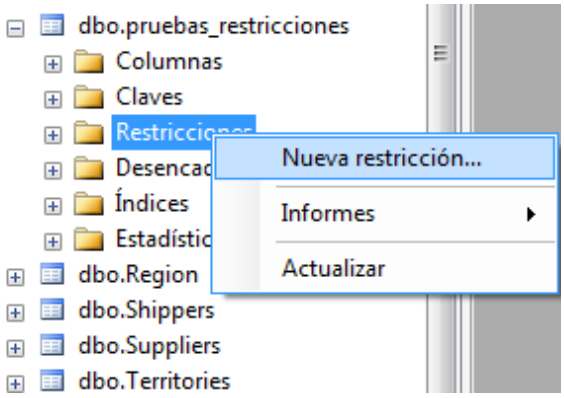
También puede ser: **nota between 0 and 20**

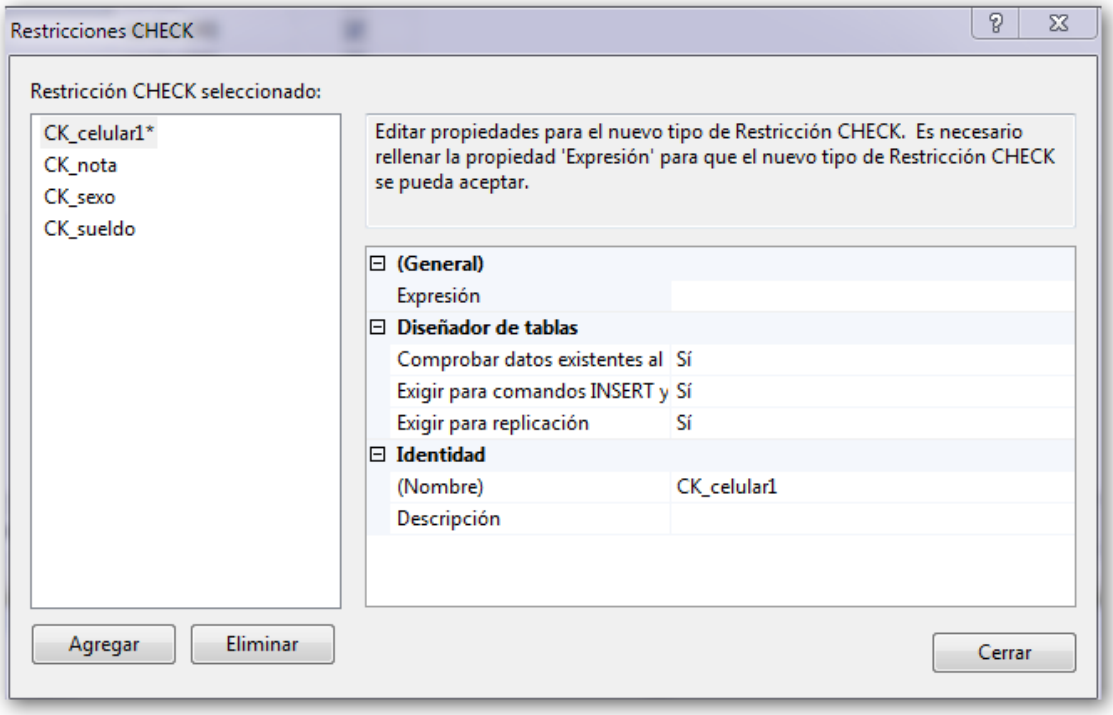


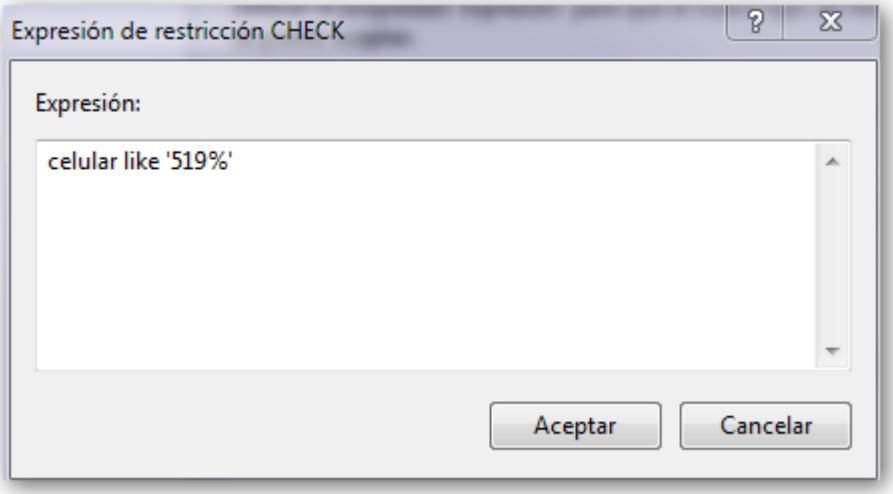


- Vamos a restringir al campo celular, como es de Perú va iniciar 519….






Restricciones CHECK
Restricción CHECK seleccionado:
CK_celular1*
CK_nota
CK_sexo
CK_sueldo
Editar propiedades para el nuevo tipo de Restricción CHECK. Es necesario rellenar la propiedad 'Expresión' para que el nuevo tipo de Restricción CHECK se pueda aceptar.
(General)
Expresión
Diseñador de tablas
Comprobar datos existentes al Sí
Exigir para comandos INSERT y Sí
Exigir para replicación Sí
Identidad
(Nombre) CK_celular1
Descripción
Agregar
Eliminar
Cerrar


Expresión de restricción CHECK
Expresión:
celular like '519%'
Aceptar
Cancelar

codigo	nombres	correo	celular	sexo	nota	sueldo
00001	Alex	alex@htmail.com	976413029	M	15	700
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Microsoft SQL Server Management Studio

No se actualizó ninguna fila.

Los datos de la fila 1 no se confirmaron.
Origen del error: .Net SqlClient Data Provider.
Mensaje de error: Instrucción INSERT en conflicto con la restricción CHECK "CK_celular1". El conflicto ha aparecido en la base de datos "Northwind", tabla "dbo.pruebas_restricciones", column 'celular'.
Se terminó la instrucción.

Corrija los errores e inténtelo de nuevo o presione ESC para cancelar los cambios.

Aceptar

codigo	nombres	correo	celular	sexo	nota	sueldo
00001	Alex	alex@htmail.com	519764	M	15	700,0000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Ahora también se puede hacer mediante código

```
alter table dbo.pruebas_restricciones
add constraint ck_celular1
check
( celular like '519%')
```

- Vamos a restringir al campo celular para que tenga los 11 caracteres, aunque los puedo llenar de espacios en blanco el cual no se le permitirá.

```
alter table dbo.pruebas_restricciones
add constraint ck_celular2
check
( len(rtrim(celular))=11)
```

codigo	nombres	correo	celular	sexo	nota	sueldo
1	alex	alex@hotmail.com	519	M	15	700
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Microsoft SQL Server Management Studio

No se actualizó ninguna fila.

Los datos de la fila 1 no se confirmaron.
Origen del error: .Net SqlClient Data Provider.
Mensaje de error: Instrucción INSERT en conflicto con la restricción CHECK "ck_celular2". El conflicto ha aparecido en la base de datos "Northwind", tabla "dbo.pruebas_restricciones", column 'celular'.
Se terminó la instrucción.

Corrija los errores e inténtelo de nuevo o presione ESC para cancelar los cambios.

Aceptar

No nos permite porque hay espacios en blando en el campo celular y solo hay 7 caracteres. Lo cual no es permitido.

codigo	nombres	correo	celular	sexo	nota	sueldo
1	alex	alex@hotmail.com	519413029	M	15	700
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Microsoft SQL Server Management Studio

No se actualizó ninguna fila.

Los datos de la fila 1 no se confirmaron.
Origen del error: .Net SqlClient Data Provider.
Mensaje de error: Instrucción INSERT en conflicto con la restricción CHECK "ck_celular2". El conflicto ha aparecido en la base de datos "Northwind", tabla "dbo.pruebas_restricciones", column 'celular'.
Se terminó la instrucción.

Corrija los errores e inténtelo de nuevo o presione ESC para cancelar los cambios.

Aceptar

No permite porque aun fala caracteres por completar.

codigo	nombres	correo	celular	sexo	nota	sueldo
1	alex	alex@hotmail.com	51976413029	M	15	700,0000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Allí si están completos los caracteres del campo celular.

- d

FUNCIONES

(1) UPPER

Convierte a mayúsculas

```
select UPPER(CompanyName)
from Suppliers
```

```
select UPPER(CompanyName)
from Suppliers
```

Resultados | Mensajes

	(Sin nombre de columna)
1	AUX JOYEUX ECCLÉSIASTIQUES
2	BIGFOOT BREWERIES
3	COOPERATIVA DE QUESOS 'LAS CABRAS'
4	ESCARGOTS NOUVEAUX
5	EXOTIC LIQUIDS
6	FORÊTS D'ÉRABLES
7	FORMAGGI FORTINI S.R.L.
8	GAI PÂTURAGE
9	G'DAY, MATE
10	GRANDMA KELLY'S HOMESTEAD

(2) LOWER

Convierte a minúsculas

```
select lower (CompanyName)
from Suppliers
```

```
select lower (CompanyName)
from Suppliers
```

Resultados | Mensajes

	(Sin nombre de columna)
1	aux joyeux ecclésiastiques
2	bigfoot breweries
3	cooperativa de quesos 'las cabras'
4	escargots nouveaux
5	exotic liquids
6	forêts d'érables
7	formaggi fortini s.r.l.
8	gai pâturage

(3) **LTRIM**

Borra los espacios de la izquierda

```
select LTRIM('  Alex')
```

```
select LTRIM('  Alex')
```

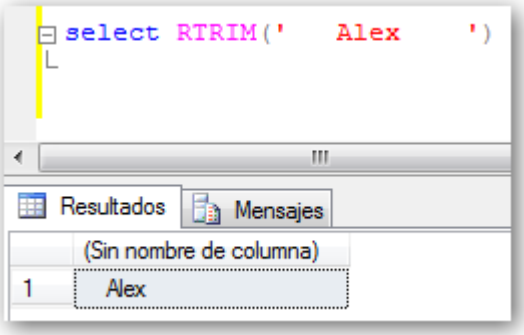
Resultados | Mensajes

	(Sin nombre de columna)
1	Alex

(4) **RTRIM**

Borra los espacios de la derecha

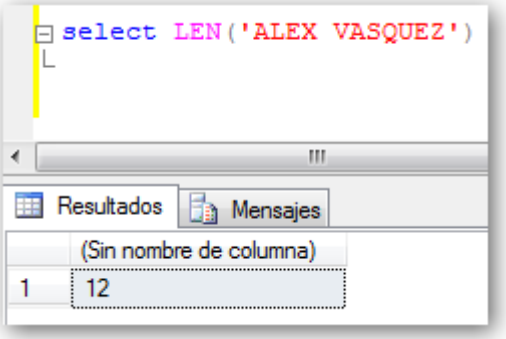
```
select RTRIM('  Alex')
```



(5) **LEN**

Cuenta la cantidad de caracteres

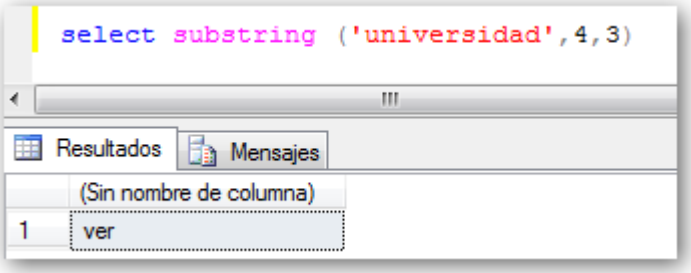
```
select LEN('ALEX VASQUEZ')
```



(6) **SUBSTRING**

Substrae un subcadena de una cadena a partir de una posición,

```
select substring ('universidad',4,3)
```



(7) **CHARINDEX**

Muestra la primera posición donde se encuentra ubicado el carácter dentro de una cadena, la cual empieza a buscar de una posición dada.

CHARINDEX (lo que busca, cadena donde buscar, desde que posición)

```
select CHARINDEX ('v','universidad',1)
```

```
select CHARINDEX ('v','universidad',1)
```

Resultados | Mensajes

	(Sin nombre de columna)
1	4

(8) GETDATE

Muestra la fecha actual del sistema

```
select GETDATE()
```

```
select GETDATE()
```

Resultados | Mensajes

	(Sin nombre de columna)
1	2011-07-19 16:47:02.743

(9) DATEADD

Añade o disminuye cierta cantidad al año (yy), mes (mm), día (dd)

```
select DATEADD(YY,1,GETDATE())
```

```
select DATEADD(YY,1,GETDATE())
```

Resultados | Mensajes

	(Sin nombre de columna)
1	2012-07-19 16:50:56.640

```
select DATEADD(YY,1,GETDATE())
```

```
select DATEADD(YY,-1,GETDATE())
```

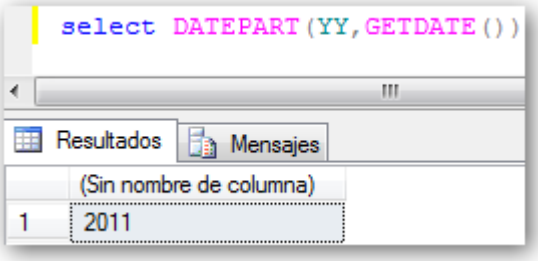
Resultados | Mensajes

	(Sin nombre de columna)
1	2010-07-19 16:55:18.060

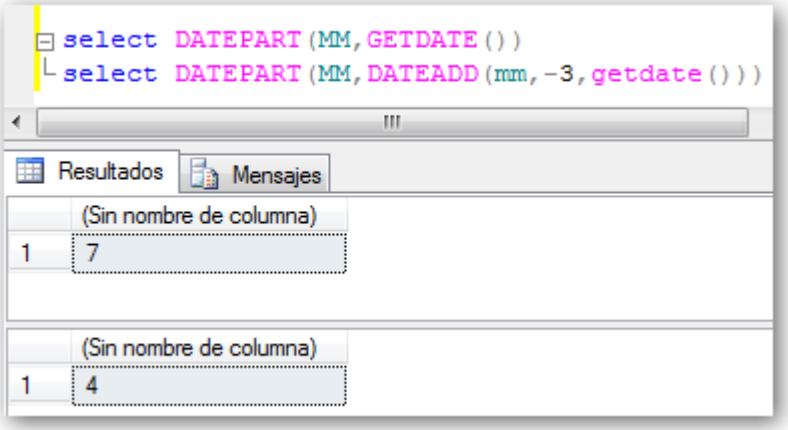
(10) DATEPART

Saca parte de la fecha ya sea el día, mes, año.

```
select DATEPART(YY,GETDATE())
```




```
select DATEPART(MM,GETDATE())
select DATEPART(MM,DATEADD(mm,-3,getdate()))
```



(11) DATEDIFF

Devuelve la diferencia entre dos fechas como días, meses, años.

```
select DATEDIFF (YY,'19/04/1996',GETDATE())
select DATEDIFF (MM,'19/04/1996',GETDATE())
select DATEDIFF (DD,'19/04/1996',GETDATE())
```

```
select DATEDIFF (YY,'19/04/1996',GETDATE())
select DATEDIFF (MM,'19/04/1996',GETDATE())
select DATEDIFF (DD,'19/04/1996',GETDATE())
```

Resultados | Mensajes

	(Sin nombre de columna)
1	15

	(Sin nombre de columna)
1	183

	(Sin nombre de columna)
1	5569

(12) DATENAME

Devuelve el nombre de un mes, día.

```
select DATENAME (MM,GETDATE())
select DATENAME (DW,GETDATE())
```

```
select DATENAME (MM,GETDATE())
select DATENAME (DW,GETDATE())
```

Resultados | Mensajes

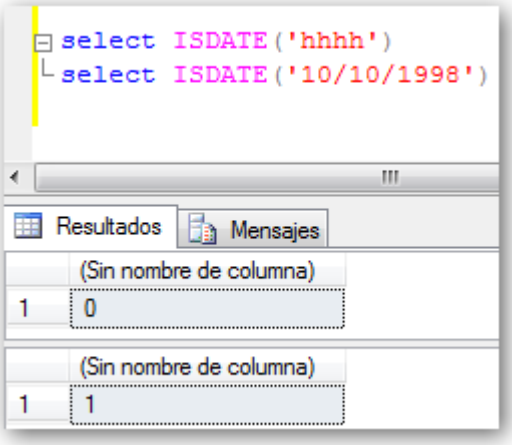
	(Sin nombre de columna)
1	Julio

	(Sin nombre de columna)
1	Martes

(13) ISDATE

Devuelve uno (1) cuando se trata de una fecha y cero (0) cuando no lo es.

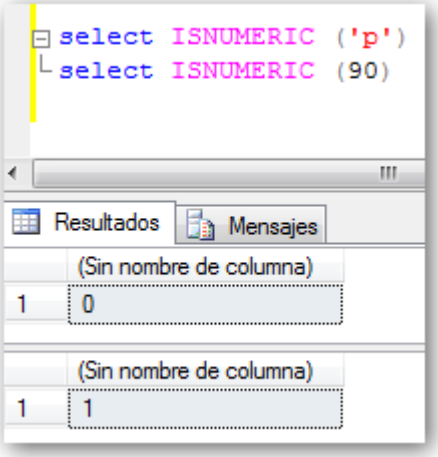
```
select ISDATE('hhhh')
select ISDATE('10/10/1998')
```



(14) ISNUMERIC

Devuelve uno (1) cuando se trata de un número y cero (0) cuando no lo es.

```
select ISNUMERIC ('p')
select ISNUMERIC (90)
```



(15) @@ROWCOUNT

Devuelve la cantidad de campos afectados con la última acción. Es decir guarda temporalmente hasta que se ejecute una siguiente instrucción.
Vamos actualizar el stock de los productos de la categoría 1

```
update Products
set UnitsInStock=200
where CategoryID=1

select @@ROWCOUNT
```

Resulta

```
update Products
set UnitsInStock=200
where CategoryID=1

select @@ROWCOUNT
```

Resultados | Mensajes

	(Sin nombre de columna)
1	12

Ahora si ejecutamos

```
select @@ROWCOUNT
```

Obtenemos.

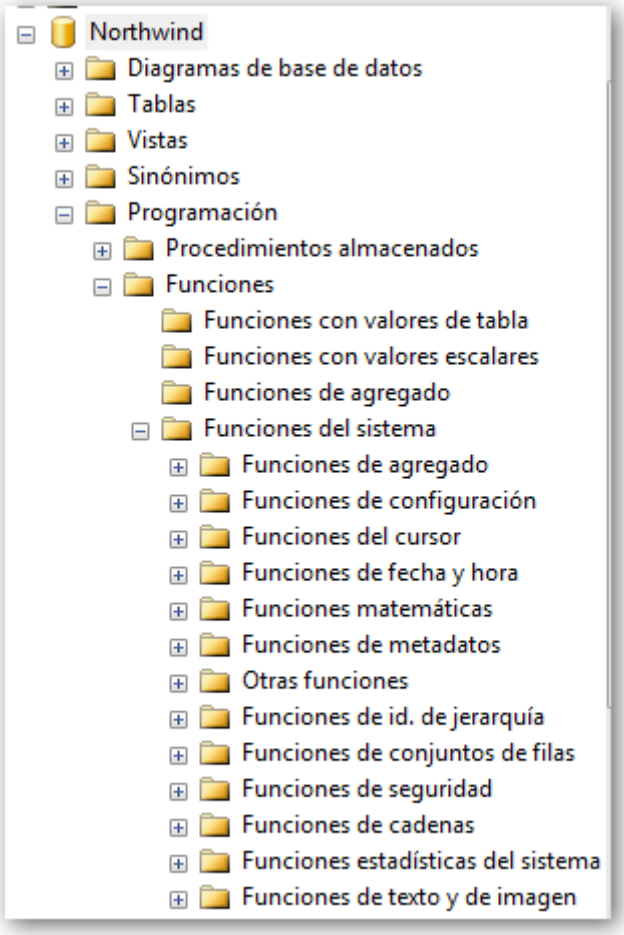
```
select @@ROWCOUNT
```

Resultados | Mensajes

	(Sin nombre de columna)
1	1

La última acción ha afectado a una fila.
Los que comienzan con @@ son variables a nivel de servidor

NOTA: Todas las funciones se los encuentra



(I) Funciones escalares

- Es similar a una función integrada
- La cláusula RETURNS especifica el tipo dato
- La función se define en un bloque BEGIN y END
- El tipo de devolución puede ser cualquier tipo de datos, excepto text, ntext, cursor o timeestamp
- Uso de SELECT y WHERE

Ejemplo 1: Hacer una función para calcular el IGV.

```
create function fn_igv
-- parametros
(@monto money)
returns money -- define el tipo de dato a devolver
as
        begin
                declare @igv money
                set @igv=@monto * 0.19
                return @igv
```

```
        end

--probamos la funcion
select UnitPrice, dbo.fn_igv(UnitPrice)as IGV
from Products
```

	UnitPrice	IGV
1	18,00	3,42
2	19,00	3,61
3	10,00	1,90
4	22,00	4,18
5	21,35	4,0565
6	25,00	4,75
7	30,00	5,70
8	40,00	7,60
9	97,00	18,43
10	31,00	5,89
11	21,00	3,99
12	38,00	7,22
13	6,00	1,14

Ejemplo 2: Crear una función el cual borre los espacios tanto de la derecha como de la izquierda.

```
create function fn_espacios
-- parametros
(@frase varchar(30))
returns varchar(30) -- define el tipo de dato a devolver
as
        begin
                declare @cadena varchar(30)
                set @cadena=ltrim(rtrim(@frase))
                return @cadena
        end

--probamos la funcion
select dbo.fn_espacios('  Alex   ')as cadena
```

	cadena
1	Alex

Ejemplo 3: Crear una función que devuelva el precio en soles

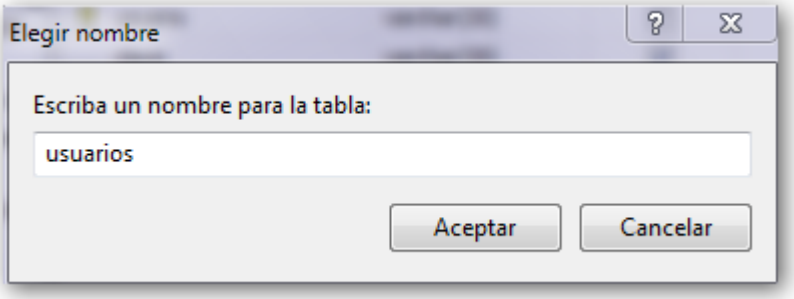
```
create function fn_soles
```

```
(@monto_dolar money,@tipo_cambio money)
returns money
as
begin
        declare @monto_soles money
        set @monto_soles= @monto_dolar*@tipo_cambio
        return @monto_soles
end

select UnitPrice,dbo.fn_soles(UnitPrice,2.85) as soles
from Products
```

Ejemplo 4: Crear una función que verifique dentro de una tabla si los usuarios son correctos o no. De tal modo que me devuelva uno (1) cuando sea correcto y cero (0) se no lo es.

Nombre de columna	Tipo de datos	Permitir v...
usuario	varchar(30)	☐
clave	varchar(30)	☑
nombre	varchar(30)	☑
apellidos	varchar(30)	☑
		☐



usuario	clave	nombre	apellidos
alex	alex	alex	vasquez
david	david	david	saman

```
create function fn_usuarios
(@usu varchar(30),@pass varchar(30))
returns int
begin
        declare @nr int
        select @nr=count(*)
        from usuarios
```

```
        where usuario=@usu and clave=@pass

        return @nr
end

---probamos
select dbo.fn_usuarios('david','alex')
select dbo.fn_usuarios('alex','alex')
```

	(Sin nombre de columna)
1	0

	(Sin nombre de columna)
1	1

Ejemplo 5: Siguiendo el ejemplo 4 vamos a cambiar la contraseña del usuario.

```
select usuario,clave
from usuarios
```

Resultados | Mensajes

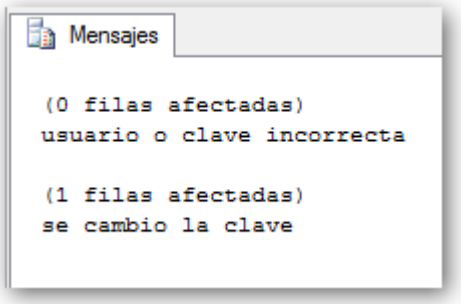
	usuario	clave
1	alex	alex
2	david	david

Como función no se puede cambiar la clave así que nos ayudaremos de un procedimiento.

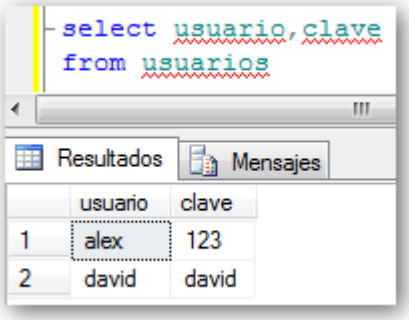
```
create proc usuarios_cambi_clave
(@usu varchar(30),
 @pass varchar(30),
 @pass_nueva varchar(30))
as
begin
        update usuarios
        set clave=@pass_nueva
        where usuario=@usu and clave=@pass

        if @@ROWCOUNT=1
                raiserror('se cambio la clave',10,1)
        else
                raiserror('usuario o clave incorrecta',10,1)
end
```

```
---probamos
exec usuarios_cambi_clave 'alex','david','123'
exec usuarios_cambi_clave 'alex','alex','123'
```



Verificamos el cambio de clave



Ejemplo 6: Crear una función que seleccione aleatoriamente un proveedor
Para la solución de este ejemplo vamos apoyarnos en una vista.

Primero creamos la vista

```
--crear una vista que almacene el numero aleatorio
create view v_rand
as
select rand() as number
```

Ahora si queremos ver la vista

Select cast (number*5 as int) from v_rand
Esto quiere decir que los números generados aleatoriamente van hacer de tipo **int** desde cero a 4.

Select cast (number*5 + 1 as int) from v_rand
El número aleatoria va desde cero a 5.

Seguidamente la función

```
-- FUNCION

create function fn_aleatorio()
returns varchar(20)
as
begin
        declare @nreg int,@reg varchar(20),@rnd float

        -- cuenta la cantidad de registros
        select @nreg=count(*)
        from suppliers

    -- hacemos el llamado a la vista y almacenamos
    -- el numero aleatorio generado en @rnd
        set @rnd=(select number from v_rand)

        select @reg=companyname
        from suppliers
        where supplierid=cast((@rnd*@nreg+1) as int)

        return @reg

end
```

Y probamos

```
select dbo.fn_aleatorio()
```

	(Sin nombre de columna)
1	Ma Maison

(II) Funciones con valores de tabla de varias instrucciones

- Contenido como un procedimiento almacenado
- Se hace referencia como una vista
- Se hace el llamado desde un FROM y el RETURN no devuelve
- La lógica es llenar la tabla
- Puede tener INSERT, UPDATE, lo que en funciones escalares no permite

Ejemplo 1: Crear una función que ingrese "C" (corto) si se desea visualizar solo el ID y apellido del Empleado, y "L" (Largo) si se desea visualizar id, nombre y apellido del Empleado.

```
CREATE FUNCTION fn_tabla_
```

```
(@tipo char(1))
--es la tabla temporal
RETURNS @fn_empleados TABLE
--son los campos de la tabla temporal @fn_empleados
  (ID int PRIMARY KEY NOT NULL,
   nombre nvarchar(61) NOT NULL)
AS
BEGIN
        --convierte a mayuscula
        -- condiciona
  IF upper(@tipo) = 'C'
         --se inserta datos en la tabla temporal
    INSERT @fn_Empleados
    --utiliza la tabla empleados, es decir
    --pasa datos a la tabla temporal
    SELECT EmployeeID, LastName
    FROM Employees
  ELSE IF upper(@tipo) = 'L'
    INSERT @fn_Empleados
    --utiliza la tabla empleados, es decir
    --pasa datos a la tabla temporal
    SELECT EmployeeID,
          (FirstName + ' ' + LastName) as Nombre
          FROM Employees
RETURN
END

--como vemos la tabla temporal tiene dos campos
--quiere decir que tambien se va a coger dos campos
--de las tablas

select * from dbo.fn_tabla_('c')
```

	ID	nombre
1	1	Davolio
2	2	Fuller
3	3	Leverling
4	4	Peacock
5	5	Buchanan
6	6	Suyama
7	7	King
8	8	Callahan
9	9	Dodsworth

```
select * from dbo.fn_tabla_('l')
```

	ID	nombre
1	1	Nancy Davolio
2	2	Andrew Fuller
3	3	Janet Leverling
4	4	Margaret Peacock
5	5	Steven Buchanan
6	6	Michael Suyama
7	7	Robert King
8	8	Laura Callahan
9	9	Anne Dodsworth

Ejemplo 2: Crear una función que nos permita seleccionar aleatoriamente "N" registro de la tabla Products.

Creamos la vista primero

```
create view v_r
as
select rand() as number
```

Luego la función

```
create  function fn_productos
(@num int)
RETURNS @prod TABLE
  (codigo int,
  nombre nvarchar(40)
  )
AS
BEGIN
  declare @reg_azar int, @i int, @j int, @nr int
  set @i=0
  -- cuenta el numero de registros
  select @nr=count(*) from products
  while @i<@num
          begin
                  set @reg_azar= cast((select number from v_r)*@nr+1 as int)
                  select @j= count (*) from @prod     where codigo = @reg_azar
                  if @j=0
                          begin
                                  set @i=@i+1
                                  INSERT @prod
                                  SELECT productid,productname
                                  from products
```

```
                    where productid=@reg_azar
                end
        end
    RETURN
END
```

Probamos

```
SELECT * FROM dbo.fn_productos(5)
```

	codigo	nombre
1	70	Outback Lager
2	18	Camarvon Tigers
3	74	Longlife Tofu
4	59	Raclette Courdavault
5	15	Genen Shouyu

(III) Funciones con valores de tabla en línea

- Similar a una vista con parámetros
- Devuelve una tabla como el resultado de una instrucción SELECT única.

Ejemplo 1: Crear una función que me muestre el ID, cliente. Filtradas por la región.

```
CREATE FUNCTION fn_CustomerNamesInRegion
  ( @RegionParameter varchar(30) )
RETURNS table
AS
RETURN (
  SELECT CustomerID, CompanyName
  FROM Customers
  WHERE Region = @RegionParameter
  )
```

Probamos.

```
SELECT * FROM fn_CustomerNamesInRegion('WA')
```

	CustomerID	CompanyName
1	LAZYK	Lazy K Kountry Store
2	TRAIH	Trail's Head Gourmet Provisioners
3	WHITC	White Clover Markets

TRIGGERS

Es un procedimiento que se ejecuta automáticamente en el momento que se produce una acción.

Es un desencadenador, el cual se desencadena frente un INSERT, UPDATE, DELETE.
El trigger no puede llamarse, se ejecuta cuando se haga una instrucción, mencionados anteriormente (INSERT, UPDATE, DELETE), sobre la tabla relacionada con el trigger,
Si queremos eliminar el trigger seguimos la sintaxis

DROP TRIGGER NOMBRE_TRIGGER

Estas instrucciones cuando las ejecutamos se almacenan en una tabla temporal

DELETE ----- DELETED
INSERT ----- INSERTED
UPDATE ----- UPDATED

Ejemplo 1: Permitir eliminar un solo registro de la tabla order details.

Creamos el trigger

```
CREATE TRIGGER BorrarOrd
ON [order details]
FOR DELETE
AS

IF (SELECT COUNT(*) FROM Deleted) > 1
BEGIN
  -- MUESTRA MENSAJE
  RAISERROR('Solo puede eliminar un registro a la vez', 16, 1)
  -- DESHACE LA INSTRUCCION,DEJANDO SIN EFECTO LA INSTRUCCION
  ROLLBACK TRANSACTION
END
```

Probamos

```
delete from [Order Details]
where OrderID = 10529
```

Nos muestra

```
Mens 50000, Nivel 16, Estado 1, Procedimiento BorrarOrd, Línea 9
Solo puede eliminar un registro a la vez
Mens. 3609, Nivel 16, Estado 1, Línea 1
La transacción terminó en el desencadenador. Se anuló el lote.
```

Verificamos en la tabla detalle de la orden.

```
select *
from [Order Details]
where OrderID = 10529
```

	OrderID	ProductID	UnitPrice	Quantity	Discount
1	10529	55	24,00	14	0
2	10529	68	12,50	20	0
3	10529	69	36,00	10	0

El trigger nos condiciona que debemos borrar uno por uno, y no en conjunto.

Ejemplo 2: Cuando se inserte un registro en la tabla order details, la cantidad pedida se debe restar en el campo unitsinstock de la tabla products.

Creamos el trigger

```
CREATE TRIGGER disminuir_stock
ON [ORDER DETAILS]
FOR INSERT
AS
        UPDATE P
        SET UNITSINSTOCK = (P.UNITSINSTOCK - I.QUANTITY)
        FROM PRODUCTS AS P
        INNER JOIN INSERTED AS I
        ON P.PRODUCTID = I.PRODUCTID
```

Vemos la tabla producto.

```
select productid,productname,unitprice,unitsinstock
from products
where productid=11
```

	productid	productname	unitprice	unitsinstock
1	11	Queso Cabrales	21,00	12

Insertamos en la tabla detalle de la orden.

```
insert into [Order Details]
values (10260,11,14,5,0)
```

Mensajes

```
(1 filas afectadas)

(1 filas afectadas)
```

Verificamos nuevamente la tabla producto.

```
select productid,productname,unitprice,unitsinstock
from products
where productid=11
```

	productid	productname	unitprice	unitsinstock
1	11	Queso Cabrales	21,00	7

Como vemos a disminuido.

Ejemplo 3: Cuando se borre un detalle de la orden, el producto que estaba pedido en la orden se actualizara a descontinuado en la tabla products.

Creamos el trigger

```
CREATE TRIGGER discontinuado_produ
ON [ORDER DETAILS]
FOR DELETE
AS
        UPDATE PRODUCTS
        SET DISCONTINUED = 1
        FROM PRODUCTS AS P
        INNER JOIN DELETED AS D
        ON P.PRODUCTID = D.PRODUCTID
```

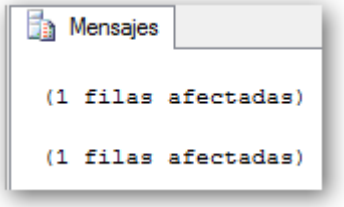
Verificamos la tabla productos

```
select ProductID,ProductName,Discontinued
from products
WHERE PRODUCTID = 11
```

	ProductID	ProductName	Discontinued
1	11	Queso Cabrales	0

Eliminamos de tabla detalle de la orden

```
DELETE FROM [ORDER DETAILS]
WHERE ORDERID = 10248 AND PRODUCTID = 11
```



Verificamos nuevamente la tabla productos

```
select ProductID,ProductName,Discontinued
from products
WHERE PRODUCTID = 11
```

	ProductID	ProductName	Discontinued
1	11	Queso Cabrales	1

Ejemplo 4: Cuando se quiera actualizar el nombre de la compañía del proveedor, no debe permitirlo.

Creamos el trigger

```
CREATE TRIGGER NOM_PROVEEDOR
ON SUPPLIERS
FOR UPDATE
AS
        IF UPDATE(COMPANYNAME)
        BEGIN
                RAISERROR ('NO PUEDE CAMBIAR EL NOMBRE DE LA COMPAÑIA',10,1)
                ROLLBACK TRANSACTION
        END
```

Probamos

```
UPDATE SUPPLIERS
SET COMPANYNAME ='JUAN'
WHERE SUPPLIERID = 2
```

```
Mensajes
NO PUEDE CAMBIAR EL NOMBRE DE LA COMPAÑIA
Mens. 3609, Nivel 16, Estado 1, Línea 1
La transacción terminó en el desencadenador. Se anuló el lote.
```

Ejemplo 5: Crear una tabla histórico que vaya guardando el nombre de la compañía (actual nombre y nuevo nombre) de los proveedores. Tabla: histórico (id, actual, nuevo) el id es identity.

Para esto debe existir la tabla HISTORICO.

Creamos el triggers

```
CREATE TRIGGER historico_
ON SUPPLIERS
FOR UPDATE
AS
        IF UPDATE(COMPANYNAME)
        BEGIN
                DECLARE @C_ACT VARCHAR(50)
                DECLARE @C_NEW VARCHAR(50)

                SELECT @C_ACT=COMPANYNAME FROM DELETED
                SELECT @C_NEW=COMPANYNAME FROM INSERTED

                INSERT INTO HISTORICO (ACTUAL,NUEVO)
                VALUES (@C_ACT,@C_NEW)
        END
```

Probamos

```
UPDATE SUPPLIERS
SET COMPANYNAME ='JUAN'
WHERE SUPPLIERID = 2
```

```
Mensajes
(1 filas afectadas)

(1 filas afectadas)
```

Verificamos

```
SELECT *
FROM HISTORICO
```

	ACTUAL	NUEVO
1	New Orleans Cajun Delights	JUAN

Ejercicios

1. Crear una función que permita cambiar el primer carácter de cada nombre y apellido ingresado a mayúsculas y el resto en minúsculas.

   Ejemplo:
   Ingresa: JUAN PABLO DÍAZ TORRES
   Sale: Juan Pablo Díaz Torres

   SOLUCIÓN

```
CREATE function mayus
(@texto varchar(50))
returns varchar(50)
as
begin
    declare @i int
    declare @c int
    --declare @txt varchar(50)
    declare @tx char(1)
    declare @frase varchar(50)

    set @i=len(@texto)
    set @tx=substring(@texto,1,1)
    set @frase=upper(@tx)
    set @c=2
    while @c<=@i
            begin
            set @tx=substring(@texto,@c,1)
            if @tx=' '
                    begin
                    set @frase=@frase+' '+upper(substring(@texto,@c+1,1))
                    set @c=@c+2
                    end
            else
                    begin
                    set @frase=@frase+LOWER(@tx)
                    set @c=@c+1
                    end

            end
    return @frase
end

--PROBAMOS
```

```
select dbo.mayus('alex david vasquez saman')

select dbo.mayus('ALEX DAVID VASQUEZ SAMAN')

select dbo.mayus(CompanyName)AS NOMBRES
from Customers
```

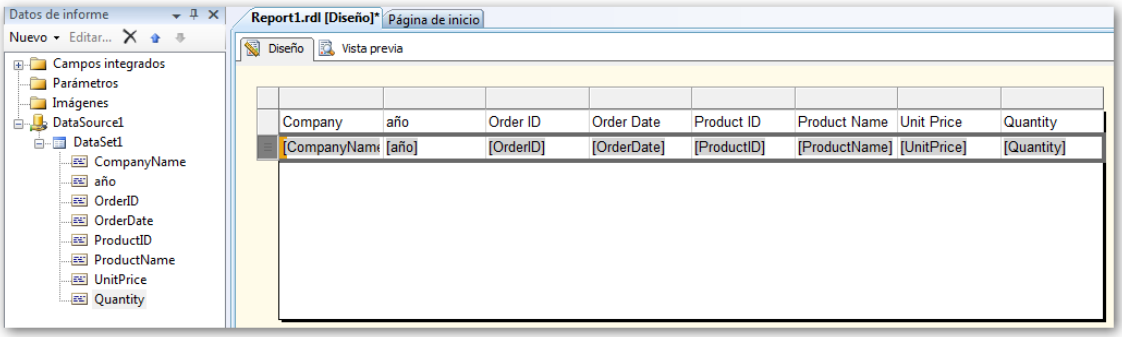
2. Crear un informe en Reporting Services
   + Cliente
      + Año
         ODERID, ORDERDATE, PRODUCTID, PRODUCTNAME, UNITPRICE, QUANTITY

SOLUCIÓN

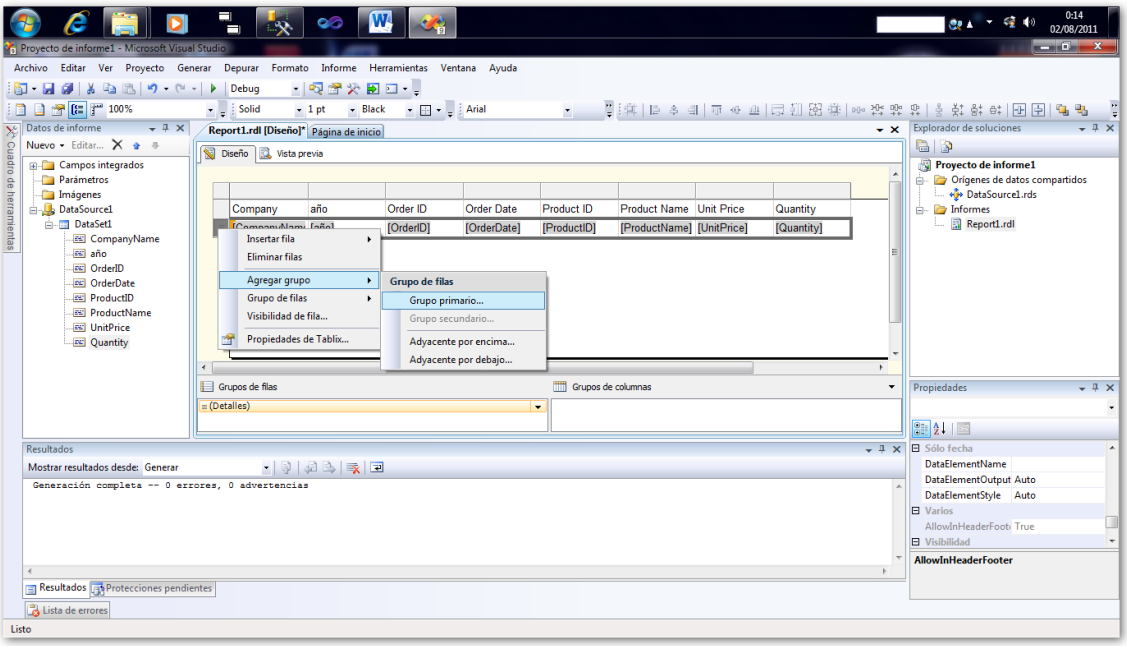
1º Creamos el SP que nos muestre lo mencionado

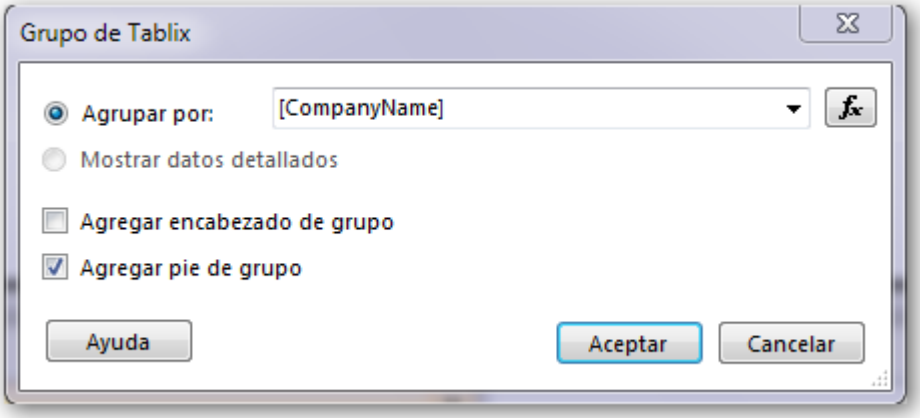
```
create proc ejercicio2
as
select c.CompanyName,YEAR(o.OrderDate)as año,o.OrderID,o.OrderDate,
   p.ProductID,p.ProductName,p.UnitPrice,od.Quantity
from Products as p
inner join [Order Details] as od
on od.ProductID=p.ProductID
inner join Orders as o
on o.OrderID=od.OrderID
inner join Customers as c
on o.CustomerID=c.CustomerID
```

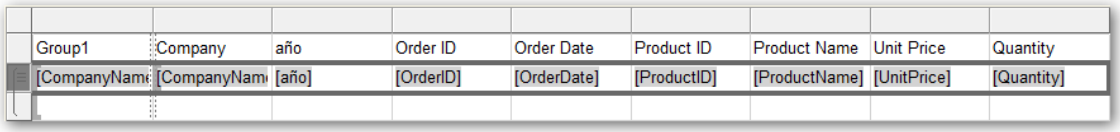
2º Creamos el Reporting

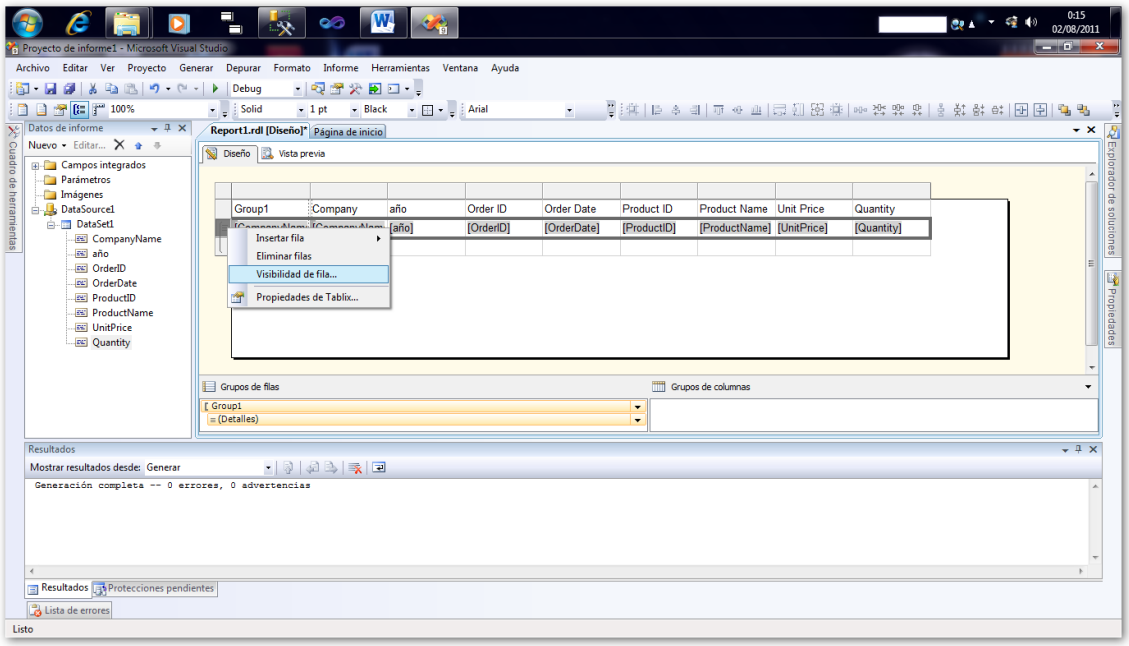


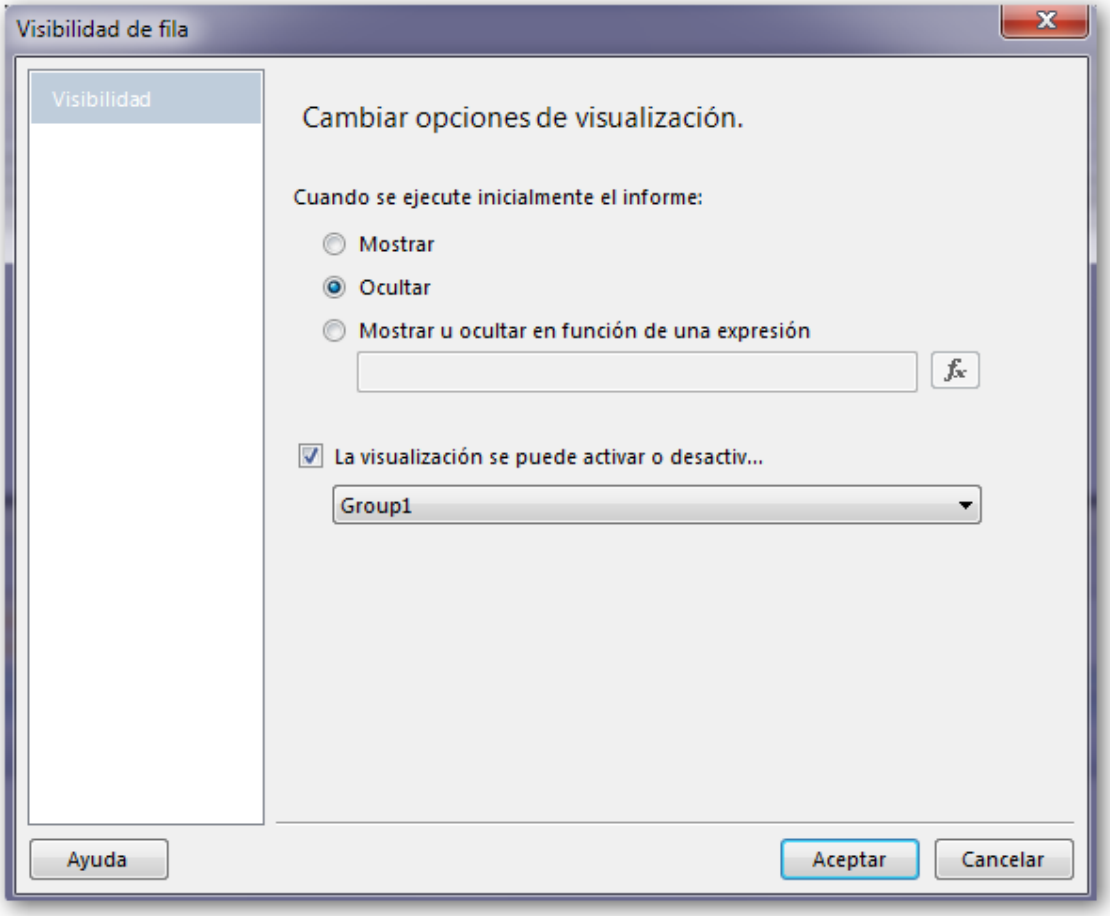
3º Creamos el grupo primario y ocultamos las filas


Proyecto de informe1 - Microsoft Visual Studio
Archivo Editar Ver Proyecto Generar Depurar Formato Informe Herramientas Ventana Ayuda
Datos de informe
Campos integrados
Parámetros
Imágenes
DataSource1
DataSet1
CompanyName
año
OrderID
OrderDate
ProductID
ProductName
UnitPrice
Quantity
Report1.rdl [Diseño]*
Página de inicio
Diseño
Vista previa
Company
año
Order ID
Order Date
Product ID
Product Name
Unit Price
Quantity
[OrderID]
[OrderDate]
[ProductID]
[ProductName]
[UnitPrice]
[Quantity]
Insertar fila
Eliminar filas
Agregar grupo
Grupo de filas
Visibilidad de fila...
Propiedades de Tablix...
Grupo de filas
Grupo primario...
Grupo secundario...
Adyacente por encima...
Adyacente por debajo...
Grupos de filas
Grupos de columnas
(Detalles)
Explorador de soluciones
Proyecto de informe1
Orígenes de datos compartidos
DataSource1.rds
Informes
Report1.rdl
Propiedades
Resultados
Mostrar resultados desde: Generar
Generación completa -- 0 errores, 0 advertencias
Resultados
Protecciones pendientes
Lista de errores
Listo

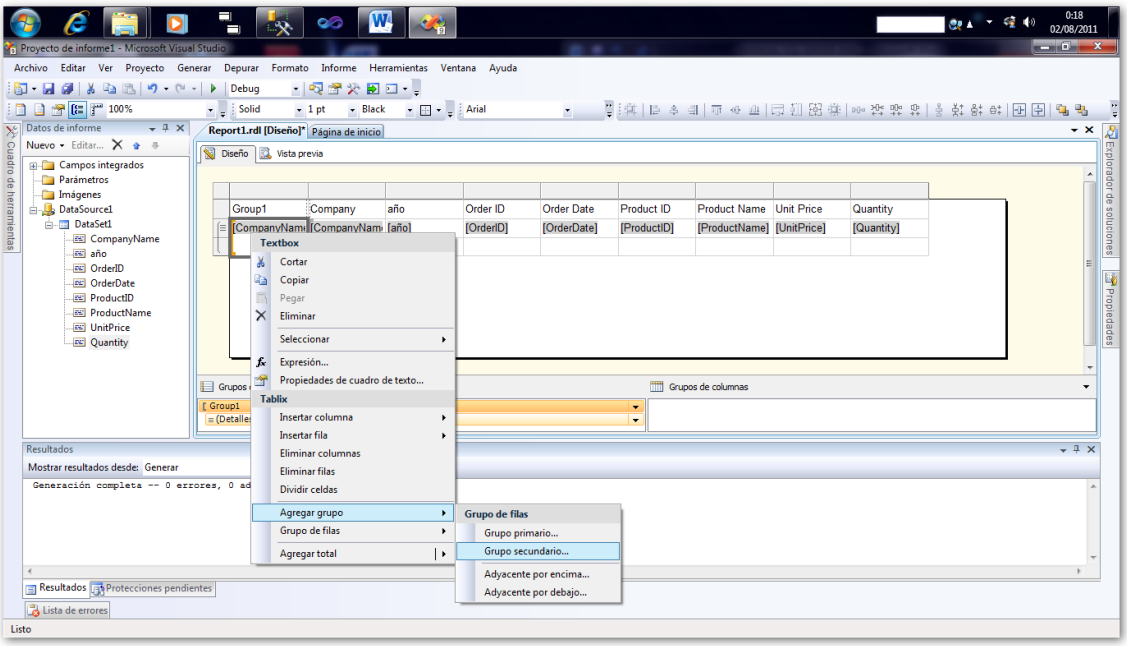

Grupo de Tablix
Agrupar por:
[CompanyName]
Mostrar datos detallados
Agregar encabezado de grupo
Agregar pie de grupo
Ayuda
Aceptar
Cancelar

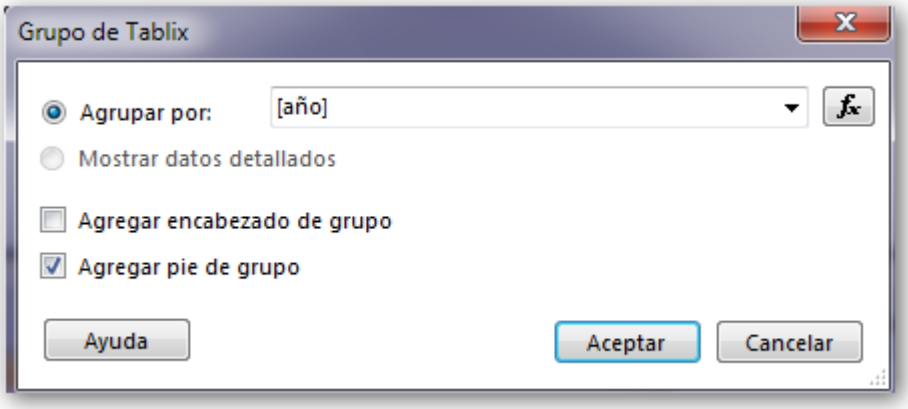

Group1
Company
año
Order ID
Order Date
Product ID
Product Name
Unit Price
Quantity
[CompanyNam
[CompanyNam
[año]
[OrderID]
[OrderDate]
[ProductID]
[ProductName]
[UnitPrice]
[Quantity]

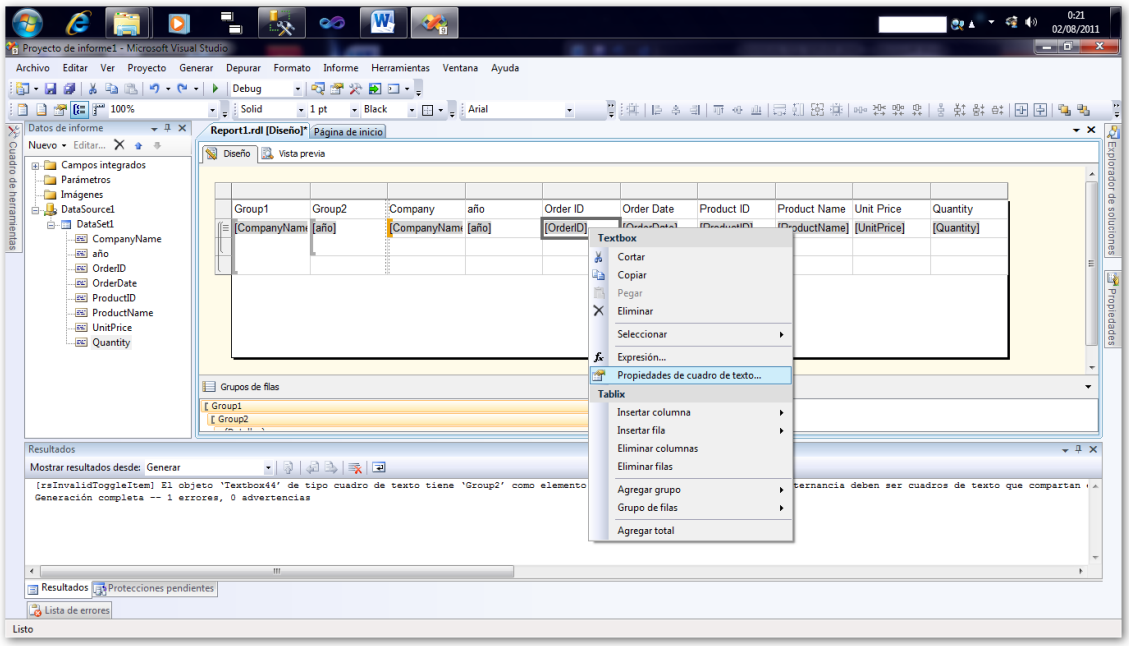


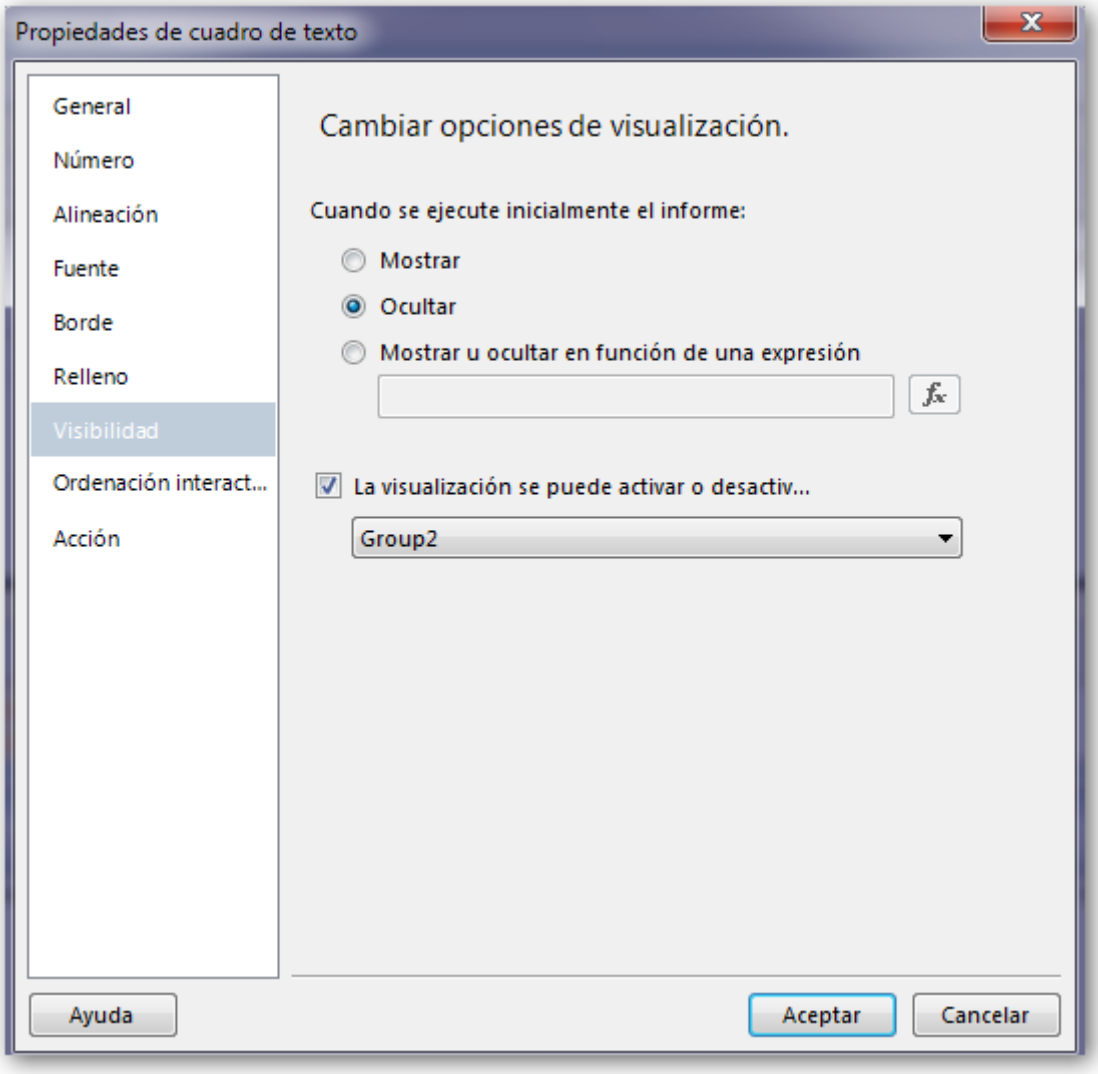


4º Creamos el grupo secundario y ocultamos las demás, una cada una.


Proyecto de informe1 - Microsoft Visual Studio
Archivo Editar Ver Proyecto Generar Depurar Formato Informe Herramientas Ventana Ayuda
Datos de informe
Nuevo Editar...
Campos integrados
Parámetros
Imágenes
DataSource1
DataSet1
CompanyName
año
OrderID
OrderDate
ProductID
ProductName
UnitPrice
Quantity
Report1.rdl [Diseño]*
Página de inicio
Diseño
Vista previa
Group1
Company
año
Order ID
Order Date
Product ID
Product Name
Unit Price
Quantity
[OrderID]
[OrderDate]
[ProductID]
[ProductName]
[UnitPrice]
[Quantity]
Textbox
Cortar
Copiar
Pegar
Eliminar
Seleccionar
Expresión...
Propiedades de cuadro de texto...
Tablix
Insertar columna
Insertar fila
Eliminar columnas
Eliminar filas
Dividir celdas
Agregar grupo
Grupo de filas
Agregar total
Grupo de filas
Grupo primario...
Grupo secundario...
Adyacente por encima...
Adyacente por debajo...
Grupos de columnas
Resultados
Mostrar resultados desde: Generar
Generación completa -- 0 errores, 0 ad
Resultados
Protecciones pendientes
Lista de errores
Listo

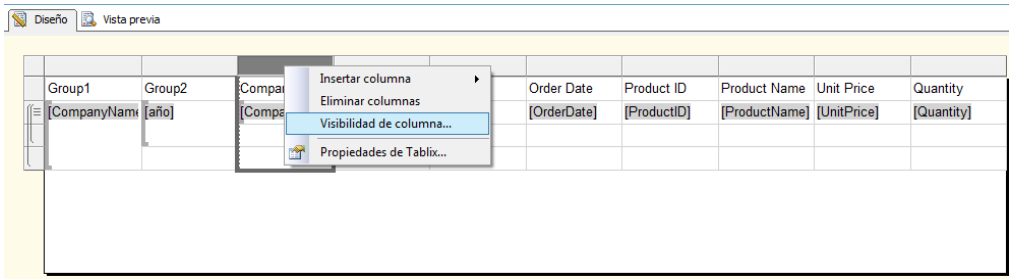

Grupo de Tablix
Agrupar por:
[año]
Mostrar datos detallados
Agregar encabezado de grupo
Agregar pie de grupo
Ayuda
Aceptar
Cancelar

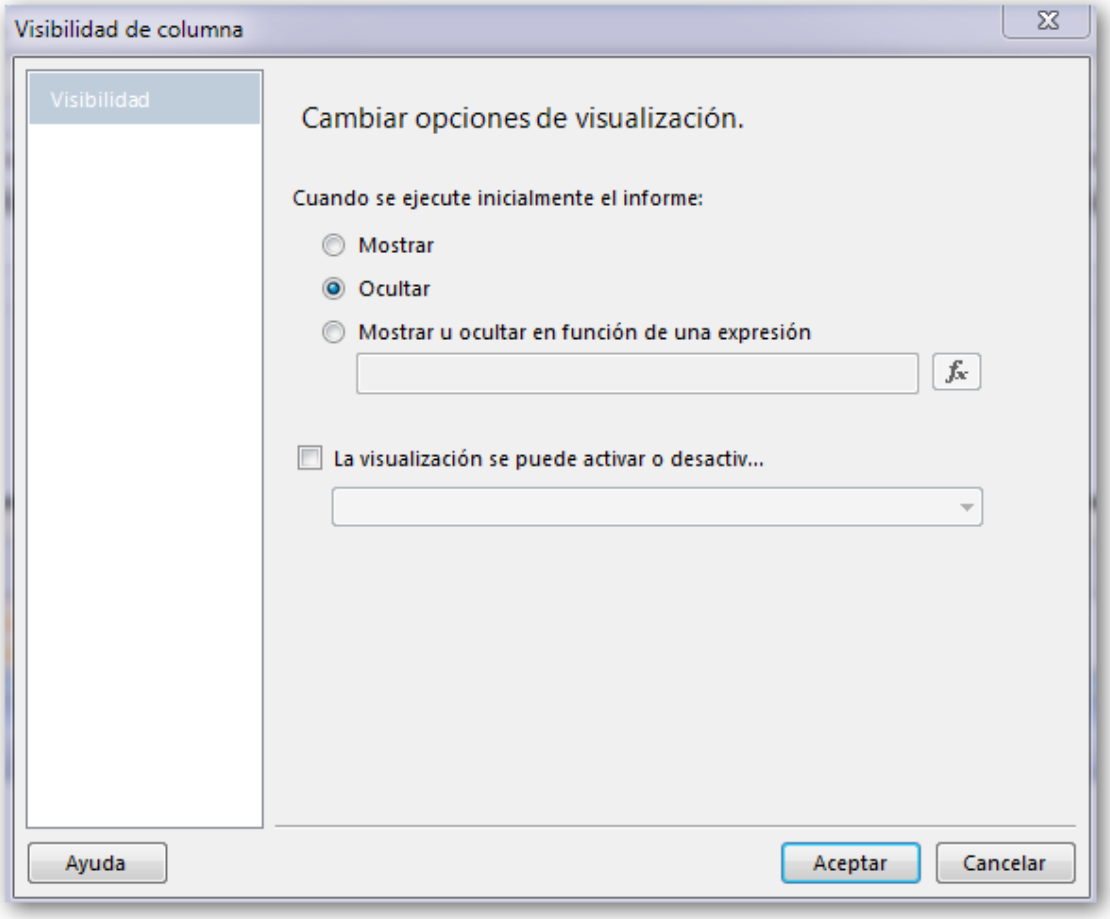

Proyecto de informe1 - Microsoft Visual Studio
Archivo Editar Ver Proyecto Generar Depurar Formato Informe Herramientas Ventana Ayuda
Datos de informe
Campos integrados
Parámetros
Imágenes
DataSource1
DataSet1
CompanyName
año
OrderID
OrderDate
ProductID
ProductName
UnitPrice
Quantity
Report1.rdl [Diseño]*
Página de inicio
Diseño
Vista previa
Group1
Group2
Company
año
Order ID
Order Date
Product ID
Product Name
Unit Price
Quantity
Textbox
Cortar
Copiar
Pegar
Eliminar
Seleccionar
Expresión...
Propiedades de cuadro de texto...
Tablix
Insertar columna
Insertar fila
Eliminar columnas
Eliminar filas
Agregar grupo
Grupo de filas
Agregar total
Grupos de filas
Resultados
Mostrar resultados desde: Generar
Generación completa -- 1 errores, 0 advertencias
Resultados
Protecciones pendientes
Lista de errores
Listo


Propiedades de cuadro de texto
General
Número
Alineación
Fuente
Borde
Relleno
Visibilidad
Ordenación interact...
Acción
Cambiar opciones de visualización.
Cuando se ejecute inicialmente el informe:
Mostrar
Ocultar
Mostrar u ocultar en función de una expresión
La visualización se puede activar o desactiv...
Group2
Ayuda
Aceptar
Cancelar

Así sucesivamente hasta el último.
Ojo que no se está ocultando toda la columna.

5º Ocultamos las columnas Company y año, y cambiamos de nombres al Group1 y Group2.





Del mismo modo para el año, esto para no generar que se muestre doble vez esas dos columnas.

Diseño | Vista previa

Group1	Group2	Company	año	Order ID	Order Date	Product ID	Product Name	Unit Price	Quantity
[CompanyNam	[año]	[CompanyNam	[año]	[OrderID]	[OrderDate]	[ProductID]	[ProductName]	[UnitPrice]	[Quantity]

Cliente	Año	Company	año	Order ID	Order Date	Product ID	Product Name	Unit Price	Quantity
[CompanyNam	[año]	[CompanyNam	[año]	[OrderID]	[OrderDate]	[ProductID]	[ProductName]	[UnitPrice]	[Quantity]

6º Quedando de la siguiente manera.

Cliente	Año	Order ID	Order Date	Product ID	Product Name	Unit Price	Quantity
⊞ Alfreds Futterkiste							
⊞ Ana Trujillo Emparedados y helados							
⊞ Antonio Moreno Taquería							
⊞ Around the Horn							
⊞ Berglunds snabbköp							

Damos clic en el signo más (+) del cliente

Cliente	Año	Order ID	Order Date	Product ID	Product Name	Unit Price	Quantity
⊟ Alfreds Futterkiste	⊞ 1997						
	⊞ 1998						
⊞ Ana Trujillo Emparedados y helados							

Damos clic en el signo más (+) del año

Cliente	Año	Order ID	Order Date	Product ID	Product Name	Unit Price	Quantity
⊟ Alfreds Futterkiste	⊟ 1997	10643	25/08/1997 0:00:00	28	Rössle Sauerkraut	45,6000	15
		10643	25/08/1997 0:00:00	39	Chartreuse verte	18,0000	21
		10643	25/08/1997 0:00:00	46	Spegesild	12,0000	2
		10692	03/10/1997 0:00:00	63	Vegie-spread	43,9000	20
		10702	13/10/1997 0:00:00	3	Aniseed Syrup	10,0000	6
		10702	13/10/1997 0:00:00	76	Lakkalikööri	18,0000	15
	⊞ 1998						

Cliente	Año	Order ID	Order Date	Product ID	Product Name	Unit Price	Quantity
⊟ Alfreds Futterkiste	⊟ 1997	10643	25/08/1997 0:00:00	28	Rössle Sauerkraut	45,6000	15
		10643	25/08/1997 0:00:00	39	Chartreuse verte	18,0000	21
		10643	25/08/1997 0:00:00	46	Spegesild	12,0000	2
		10692	03/10/1997 0:00:00	63	Vegie-spread	43,9000	20
		10702	13/10/1997 0:00:00	3	Aniseed Syrup	10,0000	6
		10702	13/10/1997 0:00:00	76	Lakkalikööri	18,0000	15
	⊟ 1998	10835	15/01/1998 0:00:00	59	Raclette Courdavault	55,0000	15
		10835	15/01/1998 0:00:00	77	Original Frankfurter grüne Soße	13,0000	2
		10952	16/03/1998 0:00:00	6	Grandma's Boysenberry Spread	25,0000	16
		10952	16/03/1998 0:00:00	28	Rössle Sauerkraut	45,6000	2

Así se mostraría, los restantes también cumplen.

3. Crear un Trigger que cada vez que cambie un precio del producto, el antiguo precio (el que se va a reemplazar, el precio anterior al cambio es el quese va a guardar). Se almacene en una tabla "histórico precio".

TABLA: Historico_Precio
ID: Int Identi Primary Key
FECHA_CAMBIO
COD_PRODUCT
PRECIO

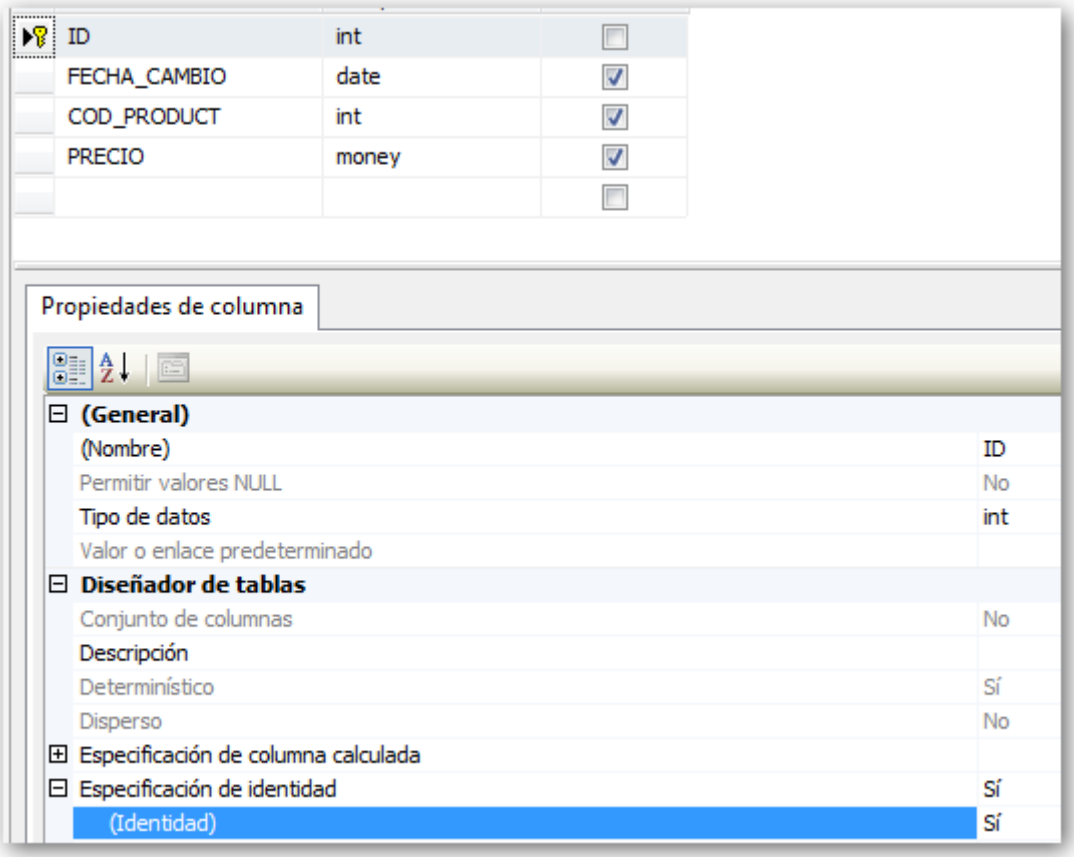
Update product
Set unitprice=100
Where productid=5

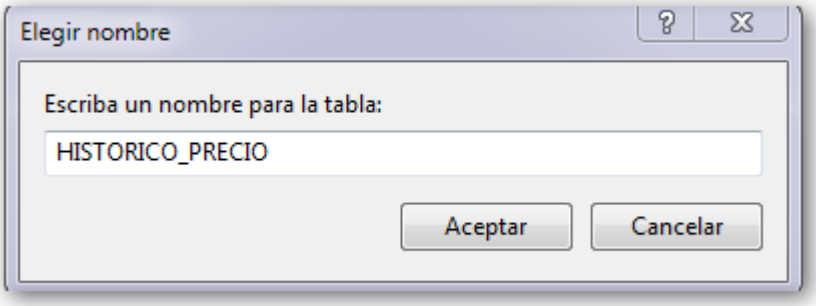
HISTORICO_PRECIO

ID	FECHA_CAMBIO	COD_PRODUCT	PRECIO
1	20/07/11	5	80

SOLUCIÓN

1º Debemos crear la tabla HISTORICO_PRECIO





2º Creamos el Triggers

```
ALTER TRIGGER historico_
ON Products
FOR UPDATE
AS
        IF UPDATE(UnitPrice)
        BEGIN
                DECLARE @PRE_ACT money
                DECLARE @FECHA DATE
                DECLARE @PROD INT

                SELECT @PRE_ACT = UnitPrice FROM DELETED
```

```
                SET @FECHA = GETDATE()
                SELECT @PROD = ProductID FROM INSERTED

                INSERT INTO
HISTORICO_PRECIO(FECHA_CAMBIO,COD_PRODUCT,PRECIO)
                VALUES (@FECHA,@PROD,@PRE_ACT)
        END

-- PROBAMOS

DELETE HISTORICO_PRECIO

-- 1º ACTUALIZAMOS

Update products
Set unitprice=258
Where productid=2

-- 2º MOSTRAMOS LA TABLA

SELECT *
FROM HISTORICO_PRECIO
```

	ID	FECHA_CAMBIO	COD_PRODUCT	PRECIO
1	1	2011-08-02	2	19,00
2	2	2011-08-02	2	8888,00

```
-- 3º VERIFICAMOS

SELECT ProductID,UNITPRICE
FROM PRODUCTS
Where productid=2
```

2011

UNIVERSIDAD NACIONAL DE CAJAMARCA

Administración de Usuarios en SQL SERVER 2008

Google APPS

Alumno

- Vásquez Samán, Alex David.

INGENIERÍA DE SISTEMAS

Administración de Usuarios en SQL SERVER 2008

1. USUARIOS

Los inicios de sesión de usuario

La seguridad es lo más importante cuando hablamos de sistemas, en SQL Server también se trabaja con la parte de seguridad. Todo inicia por la configuración de un "Inicio de Sesión".

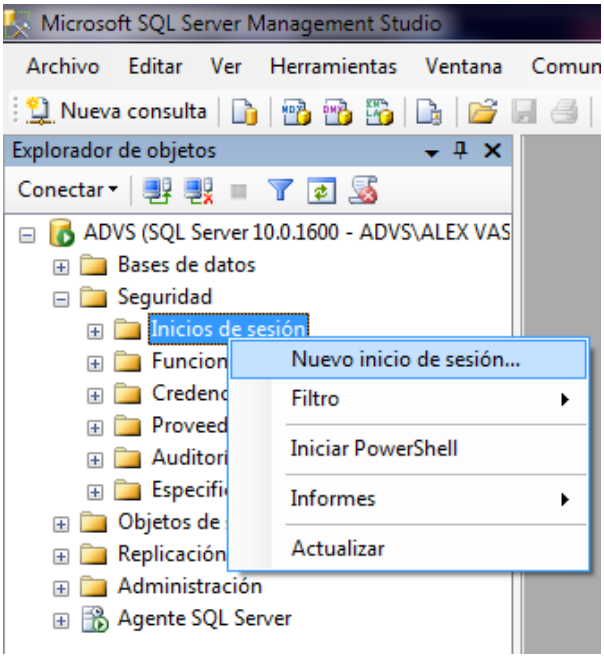
SQL Server permite la creación de inicios de sesión de usuario. Cada persona que necesita tener acceso a SQL Server se puede administrar su propia cuenta de usuario.

Cuando el administrador configura estos inicios de sesión de usuario, se le puede asignar a cualquier número de roles y esquemas, en función del acceso que el individuo tiene derecho.

Para crear un inicio de sesión nuevo usuario

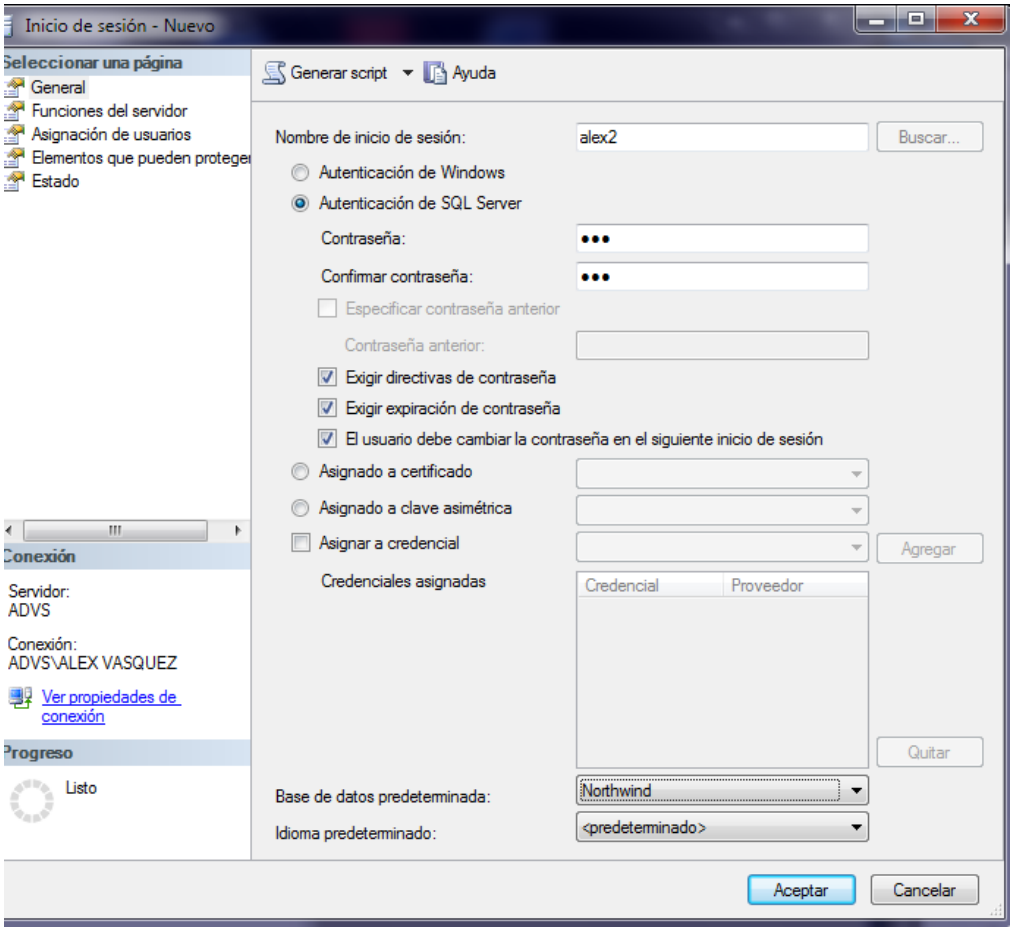
1. Utilizando SQL Server Management Studio, expanda la opción **Seguridad** y haga clic derecho sobre **Inicios de sesión**.

2. Haga clic en **Nuevo inicio de sesión**.



3. Completar las propiedades de entrada en la pestaña **General** al proporcionar un nombre para el inicio de sesión, la elección del método de autenticación (que proporciona una contraseña si usted elige **autenticación de SQL Server**), y la selección de la base de datos para utilizar como predeterminado. Si usted no elige un idioma, se utilizará el valor predeterminado para la instalación actual de SQL Server.

   Si recibe un error que dice "La opción **MUST_CHANGE** no es compatible con esta versión de Microsoft Windows", simplemente desactive la casilla **"El usuario debe cambiar la contraseña en el siguiente inicio de sesión**" opción. El error se produce porque el sistema operativo no es compatible con esta opción.



Hacemos clic en el botón de búsqueda (**Buscar)** y seleccionamos la cuenta de Windows que deseamos agregar como Login o simplemente digitamos. En algunos casos lo digitaremos en el formato **<domino o computadora>\<usuario>**, teniendo en cuenta que debe existir.

Como ya veníamos mencionando existen dos tipos de Inicios de sesión, hablaremos muy brevemente de ellos:

a) **Autenticación de Windows.**

Cuando un usuario se conecta a través de una cuenta de Windows NT o de usuario de Windows 2000, SQL Server revalida el nombre de cuenta y contraseña llamando de nuevo a Windows NT o Windows 2000 para la información.

SQL Server logra la integración de seguridad de inicio de sesión con Windows NT o Windows 2000 con los atributos de seguridad de un usuario de la red para controlar el acceso de inicio de sesión. Los atributos de un usuario de la red de seguridad se establezcan en el momento de inicio de sesión de red y es validado por un controlador de dominio de Windows. Cuando un usuario de la red intenta conectar, SQL Server utiliza las instalaciones basadas en Windows para determinar el nombre de usuario validado. SQL Server verifica que la persona es quien dice ser, y permite o deniega el acceso de inicio de sesión basada en el nombre de usuario de red solo, sin necesidad de un nombre de usuario y contraseña aparte

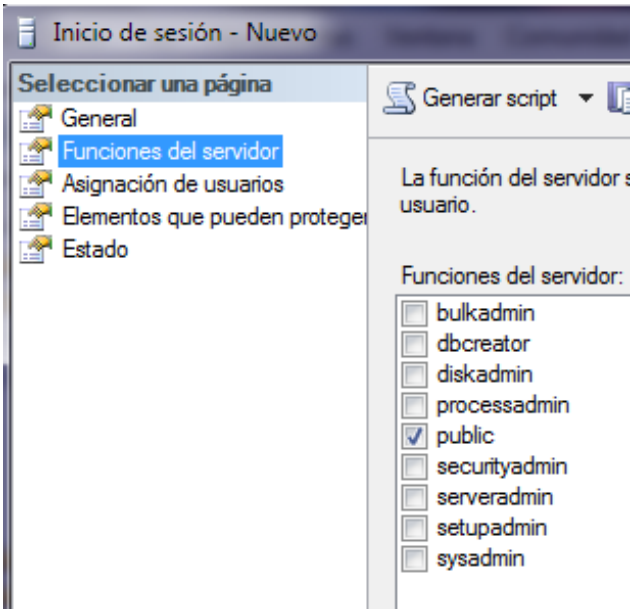
b) **Autenticación de SQL Server.**

Cuando un usuario se conecta con un nombre de usuario y la contraseña de una conexión no son de confianza, SQL Server realiza la autenticación y la comprobación para ver si una cuenta de inicio de sesión de SQL Server se ha establecido y si la contraseña especificada coincide con el grabado previamente. Si SQL Server no tiene configurada una cuenta de acceso, la autenticación falla y el usuario recibe un mensaje de error.

Autenticación de SQL Server se proporciona para compatibilidad con versiones anteriores ya que las aplicaciones escritas para SQL Server versión 7.0 o anterior pueden requerir el uso de los inicios de sesión de SQL Server y las contraseñas. Además, autenticación de SQL Server es necesaria cuando una instancia de SQL Server se ejecuta en Windows 98 porque el modo de autenticación de Windows no es compatible con Windows 98. Por lo tanto, SQL Server utiliza de modo mixto cuando se ejecuta en Windows 98 (pero sólo es compatible con autenticación de SQL Server).

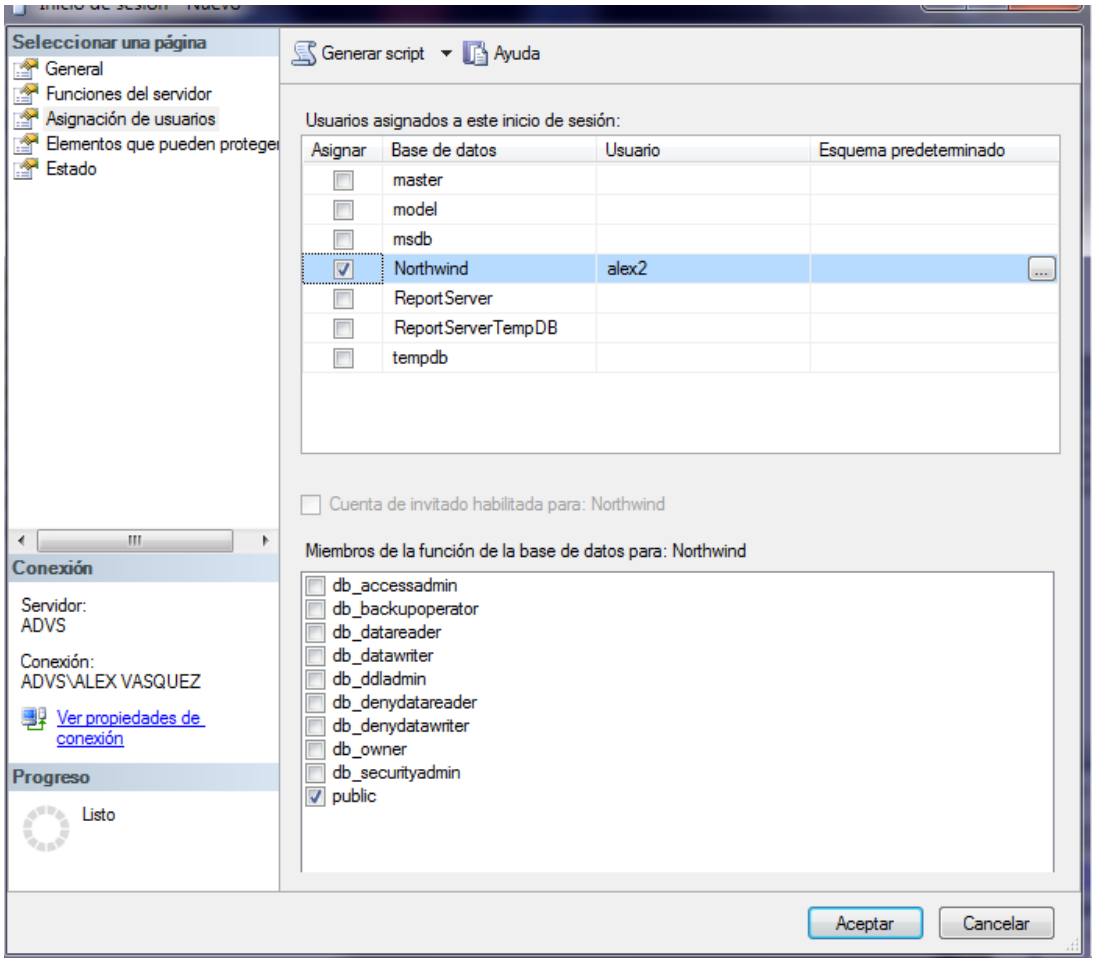
A pesar de que la autenticación de Windows, se recomienda, autenticación de SQL Server puede ser necesaria para las conexiones con otros clientes de Windows NT y Windows 2000, sino que también puede ser necesario para las aplicaciones heredadas.

4. Haga clic en **Funciones del servidor** ficha si es necesario aplicar los privilegios de seguridad a nivel de servidor.

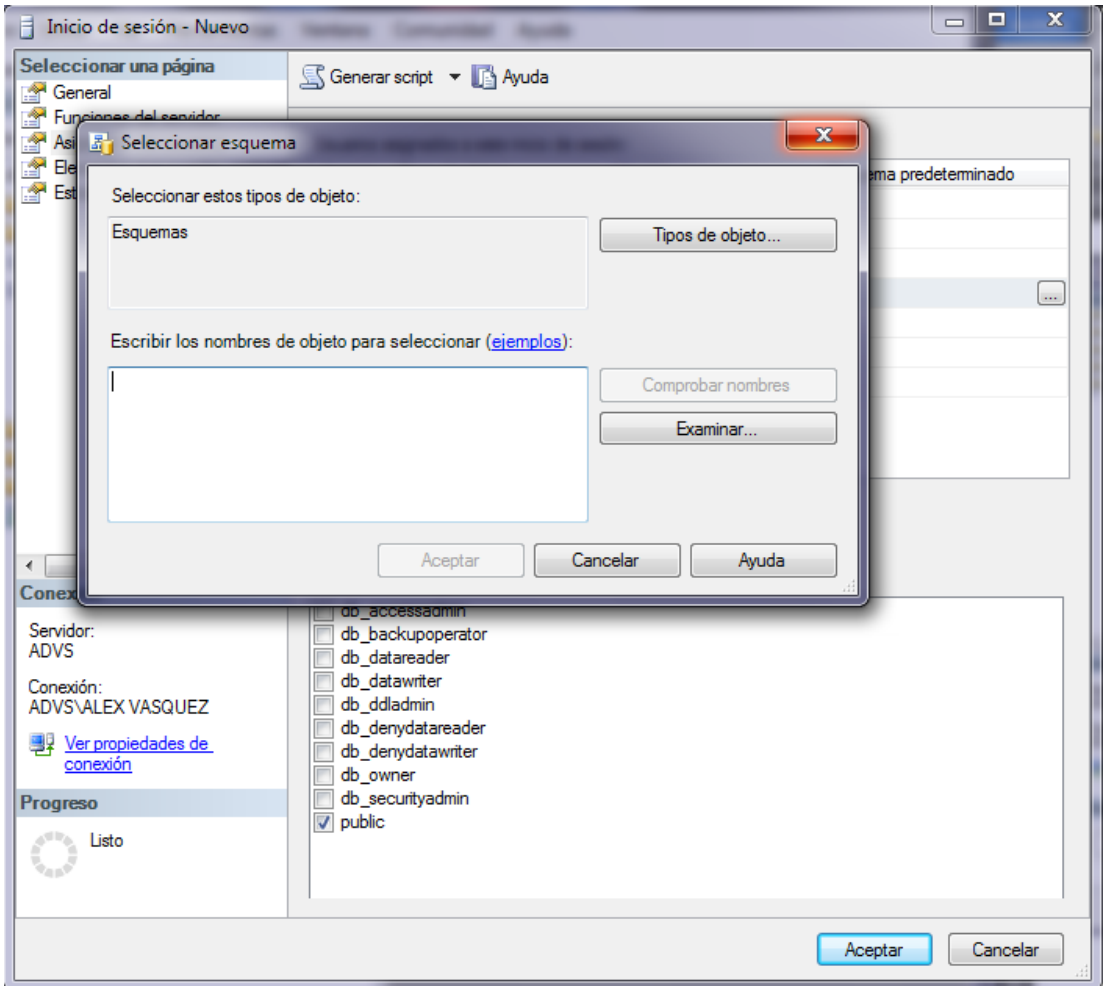


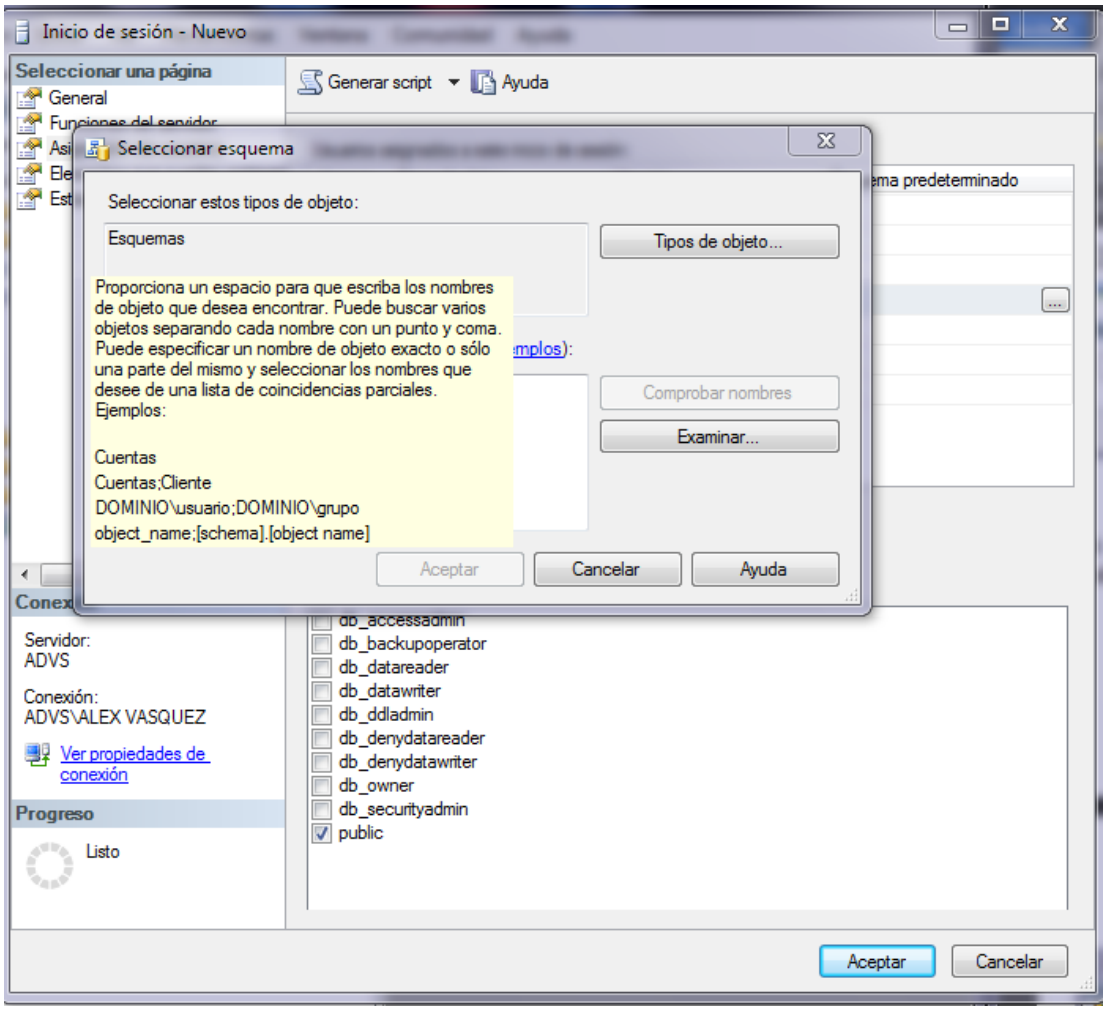
5. Haga clic en la **Asignación de usuarios** ficha para especificar qué bases de datos de esta cuenta de usuario se le permite el acceso. De forma predeterminada, la entrada será asignado a la "**pública"** papel, que prevé el inicio de sesión con acceso básico. Si el inicio de sesión necesita más el acceso en una o más bases de datos, puede ser asignado a otra función con mayores privilegios.

   Tenga en cuenta que estas funciones son **la base de datos Funciones** y son diferentes a las funciones de servidor en la ficha anterior. Las funciones de servidor son para la administración de SQL Server. Funciones de base de datos se crean dentro de cada base de datos y especificar lo que la entrada se puede hacer dentro de esa base de datos.

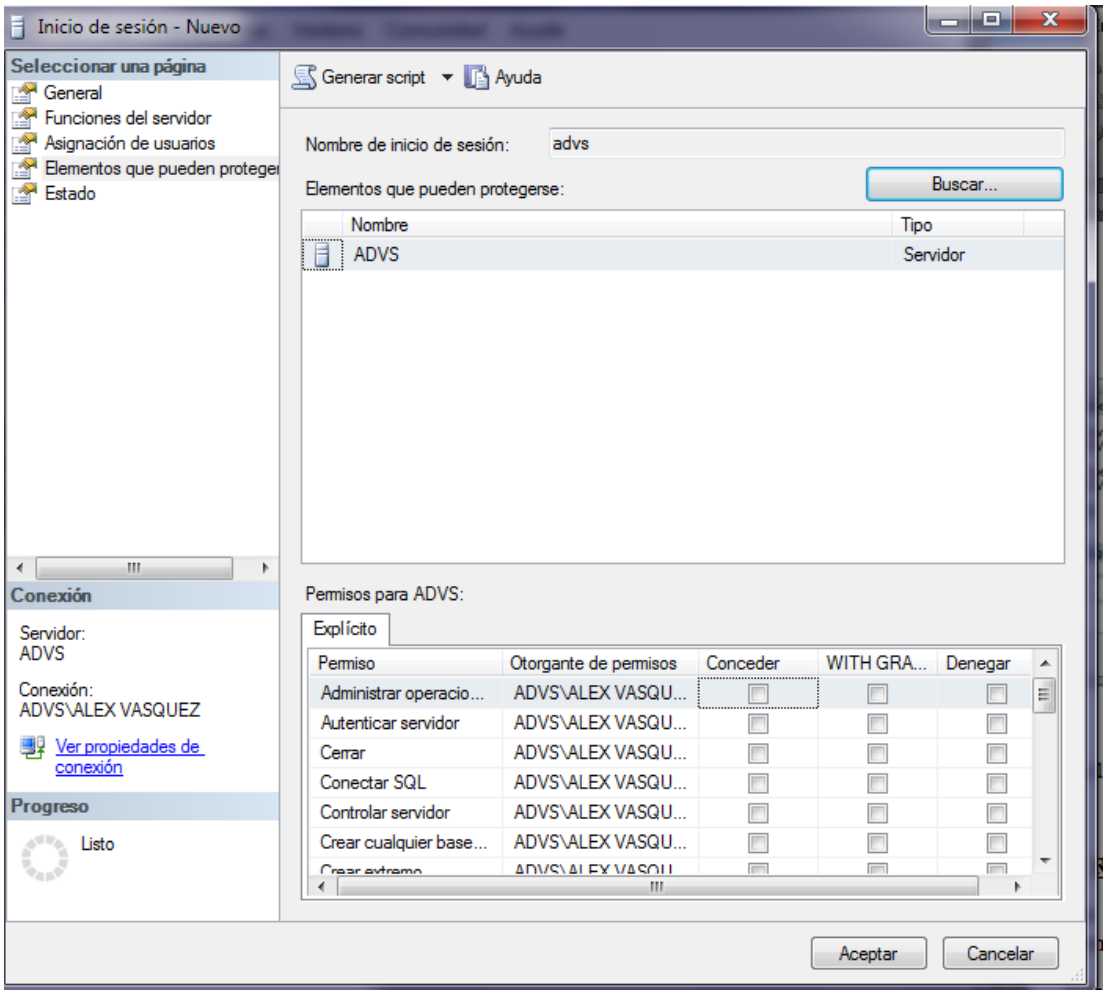


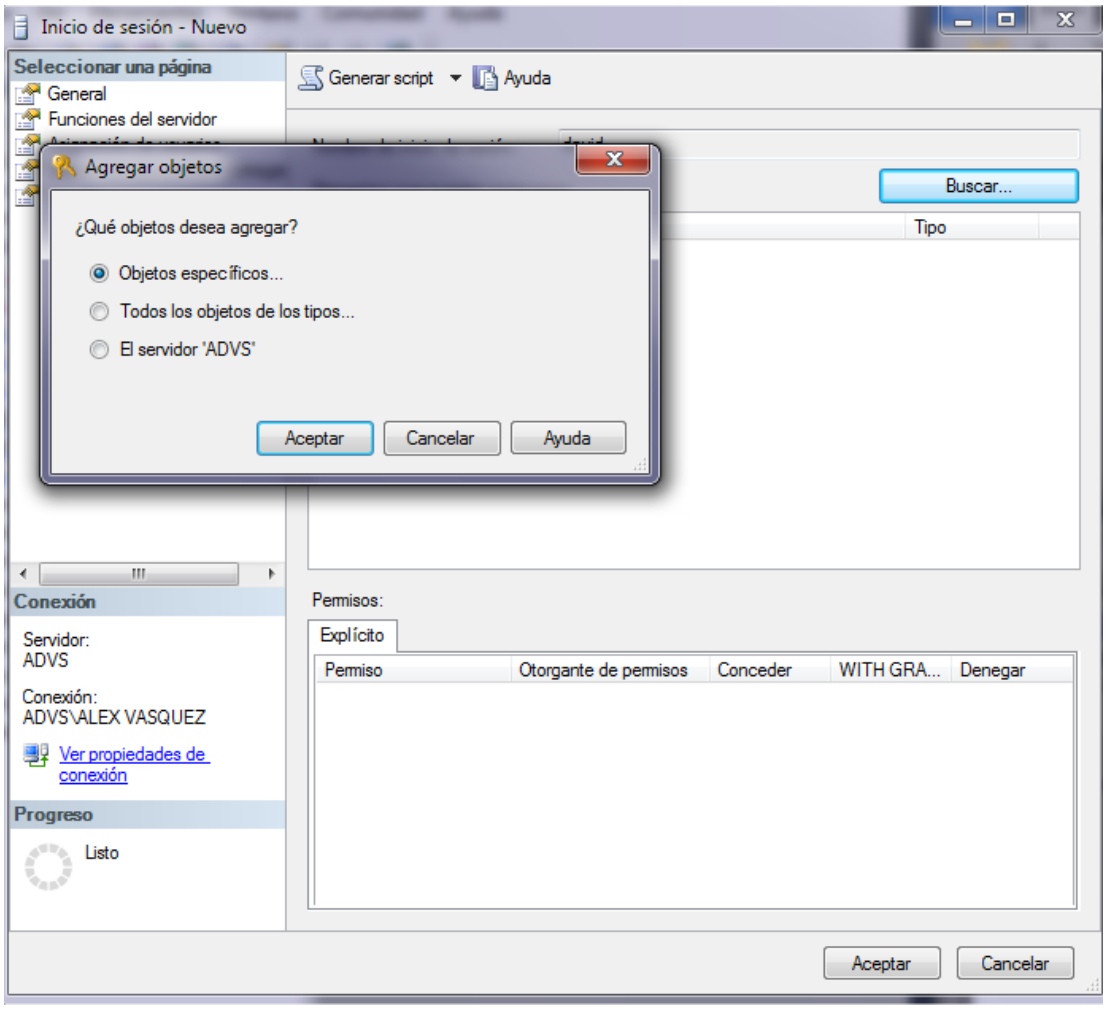
En la parte derecha de la imagen (Esquema predeterminado), hay un botón donde Ud. podrá buscar objetos.

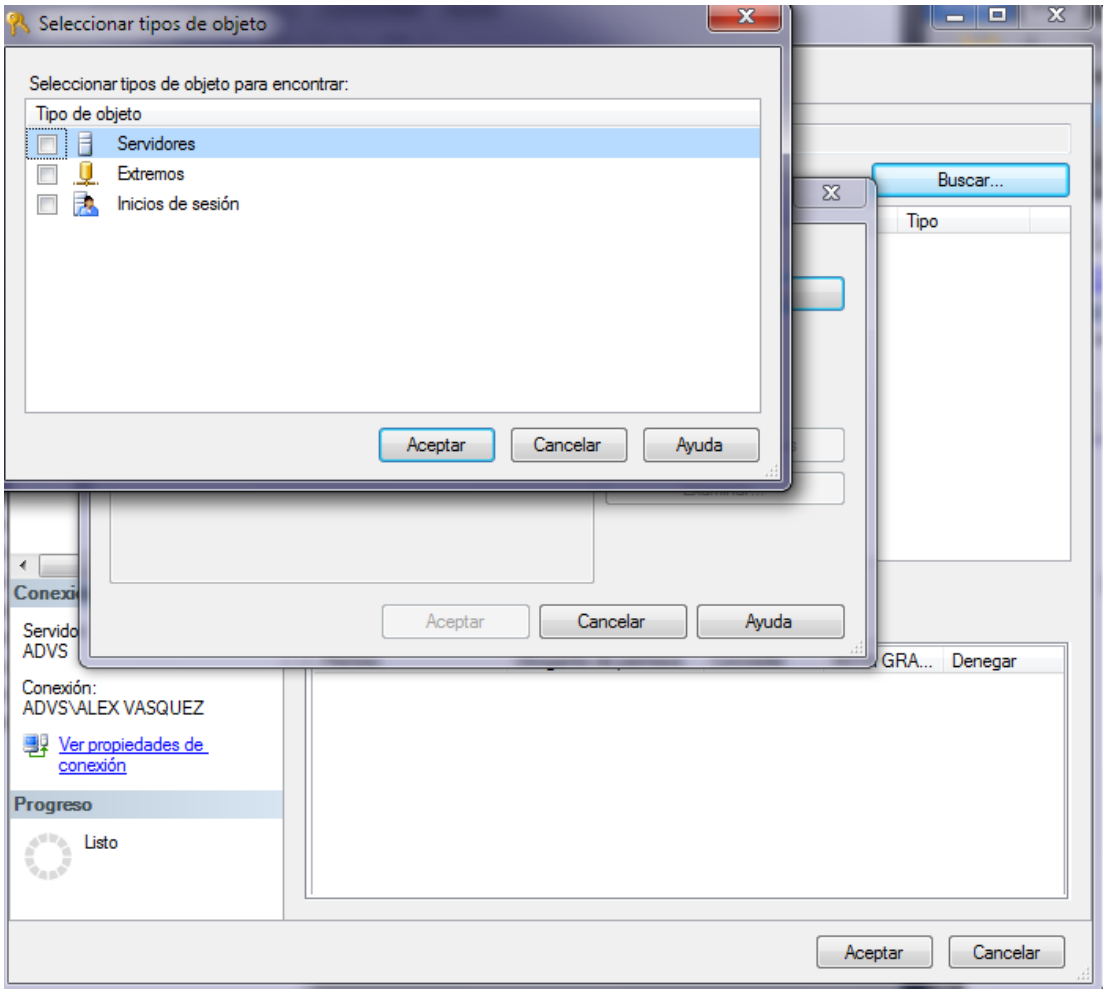

Inicio de sesión - Nuevo
Seleccionar una página
General
Generar script
Ayuda
Seleccionar esquema
Seleccionar estos tipos de objeto:
Esquemas
Tipos de objeto...
Escribir los nombres de objeto para seleccionar (ejemplos):
Comprobar nombres
Examinar...
Aceptar
Cancelar
Ayuda
Servidor:
ADVS
Conexión:
ADVS\ALEX VASQUEZ
Ver propiedades de conexión
Progreso
Listo
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
public
Aceptar
Cancelar



6. En la opción **Elementos que puede proteger** podrá elegir los elementos que desea proteger

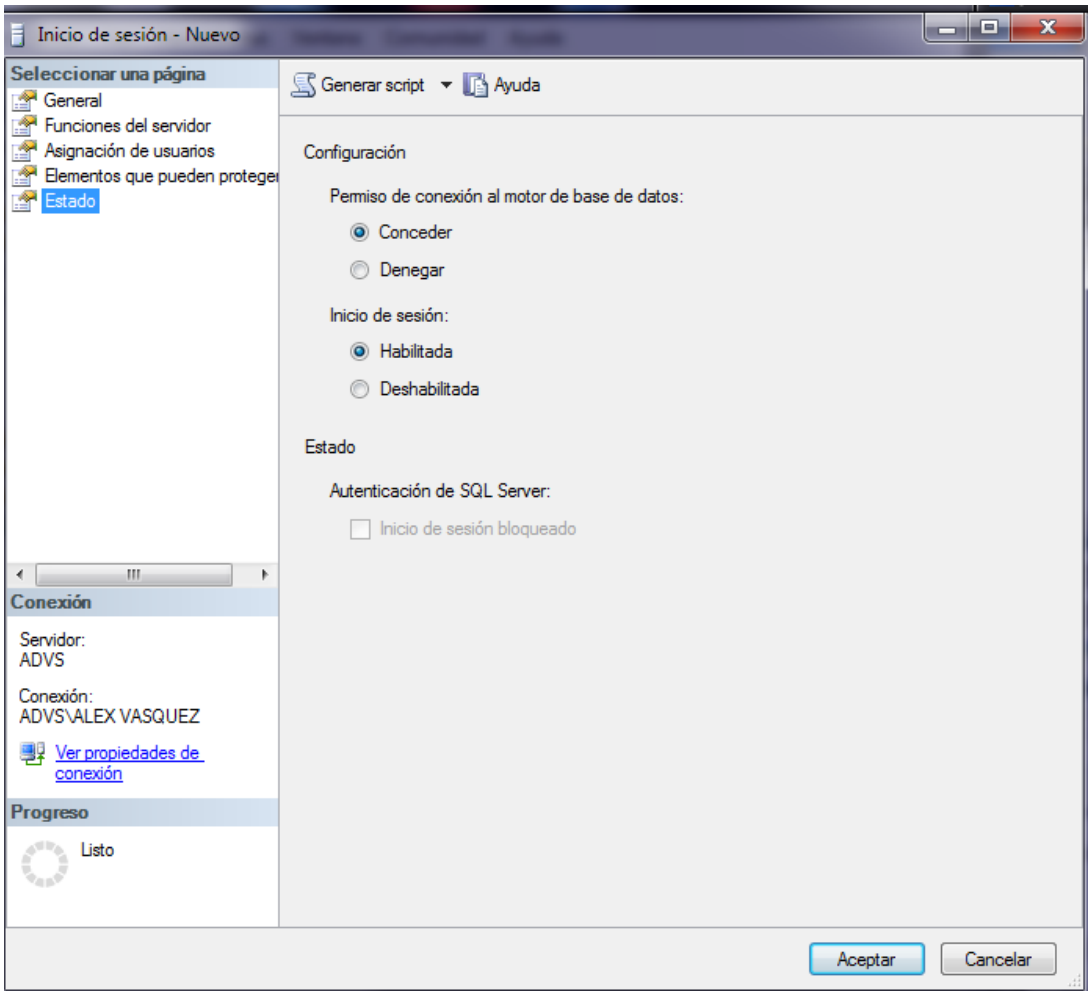

Inicio de sesión - Nuevo
Seleccionar una página
General
Funciones del servidor
Asignación de usuarios
Elementos que pueden protege
Estado
Generar script
Ayuda
Nombre de inicio de sesión:
advs
Elementos que pueden protegerse:
Buscar...
Nombre
Tipo
ADVS
Servidor
Conexión
Servidor:
ADVS
Conexión:
ADVS\ALEX VASQUEZ
Ver propiedades de conexión
Progreso
Listo
Permisos para ADVS:
Explícito
Permiso
Otorgante de permisos
Conceder
WITH GRA...
Denegar
Administrar operacio...
ADVS\ALEX VASQU...
Autenticar servidor
ADVS\ALEX VASQU...
Cerrar
ADVS\ALEX VASQU...
Conectar SQL
ADVS\ALEX VASQU...
Controlar servidor
ADVS\ALEX VASQU...
Crear cualquier base...
ADVS\ALEX VASQU...
Aceptar
Cancelar


Inicio de sesión - Nuevo
Seleccionar una página
General
Funciones del servidor
Generar script
Ayuda
Agregar objetos
¿Qué objetos desea agregar?
Objetos específicos...
Todos los objetos de los tipos...
El servidor 'ADVS'
Aceptar
Cancelar
Ayuda
Buscar...
Tipo
Conexión
Servidor:
ADVS
Conexión:
ADVS\ALEX VASQUEZ
Ver propiedades de conexión
Progreso
Listo
Permisos:
Explícito
Permiso
Otorgante de permisos
Conceder
WITH GRA...
Denegar
Aceptar
Cancelar



Si elegimos **Servers**, nos permitirá ver la lista de servidores (instancias) instaladas en la computadora o red, de tal forma podremos gestionar los permisos que se tendrán sobre el servidor, si se puede apagar, si tiene control sobre el mismo, entre muchos otros. En cambio sí elegimos **Extremos** podemos seleccionar si podemos alterarlo, conectarnos, controlarlo, ser propietario o ver la definición del mismo. En caso de que seleccionemos **Inicio de Sesión** nos permitirá, permitir o denegar los servicios a alterar, controlar, utilizar suplantación de personalidad, o ver la definición del mismo.

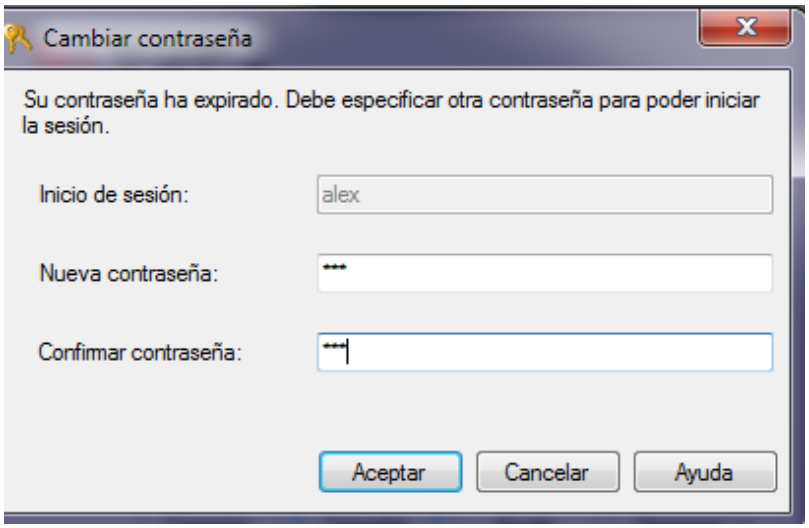
7. En la opción **Estado** podrá configurar el inicio de sesión. Podemos indicar si el Inicio de Sesión tiene permisos para conectarse a SQL Server y si está activado. Esto lo dejaremos como viene por defecto **Habilitado/Enabled**.



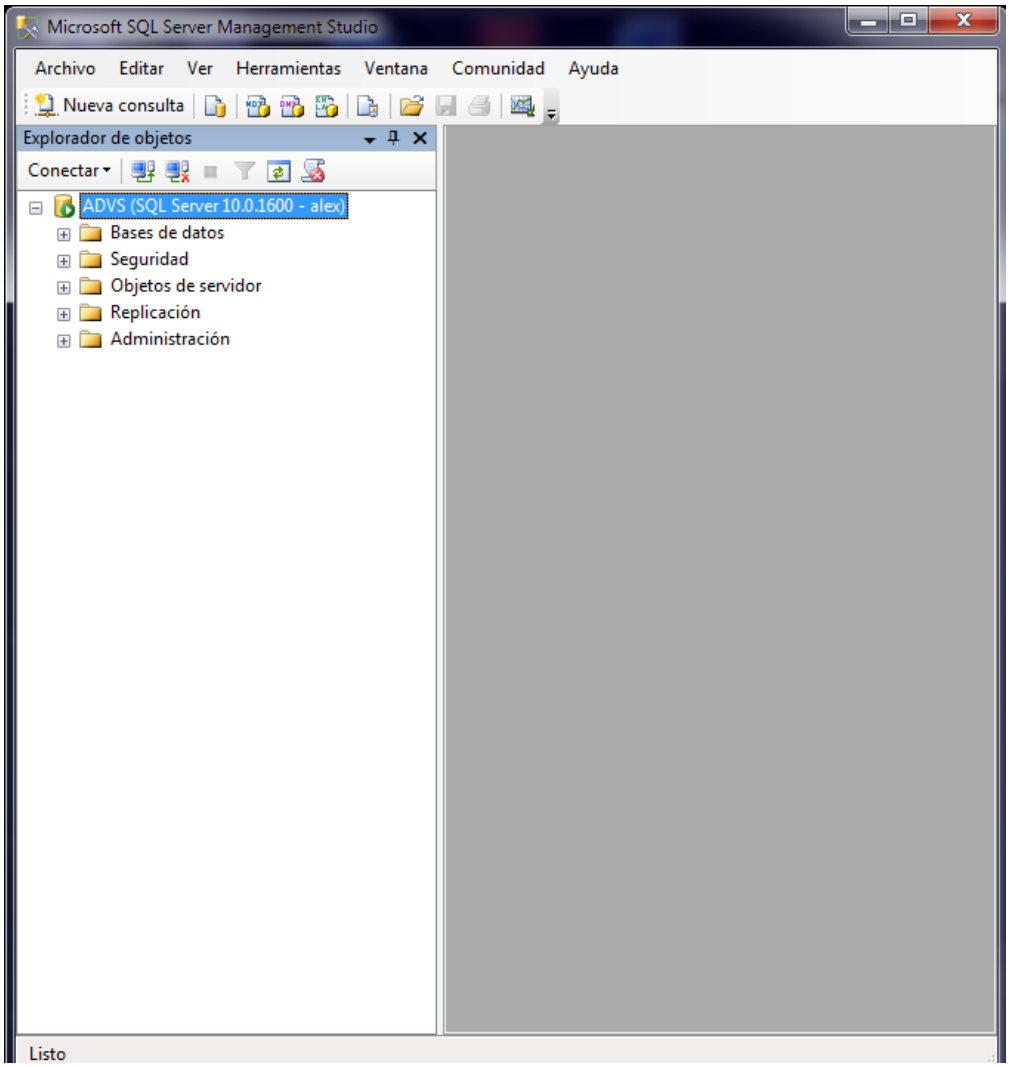
Hacemos clic en **Aceptar** y si hemos hecho todo correctamente ya tendremos nuestro nuevo **Inicio de Sesión** creado.

8. Ingresar con la nueva sesión.

Bueno en mi caso me pide cambiar la contraseña. Como ya mencionábamos por compatibilidad.



Luego de aceptar, ingresamos al programa.



Se percataran que aparece el nombre de sesión con el que se ha ingresado.

2. ROLES Y PRIVILEGIOS(PERMISOS)

Los roles son agrupaciones de usuarios en SQL Server donde se puede asignar permisos (y negar permisos también).

Esto te permite organizar usuarios más fácilmente simplemente añadiéndolos a un rol o moviéndolos de rol a rol sin necesidad de dar permisos individuales.

SQL server tiene un grupo de roles que son creados en el momento de la instalación que corresponden a funciones comunes de administradores.

Los roles de SQL Server cumplen la misma función que un rol de Windows, esto es, agrupar usuarios que comparten los mismos permisos. Dichos permisos son otorgados al rol y los usuarios heredan los permisos de los roles.

❖ **Roles de Servidor**

db_owner	Los miembros de la función de base de datos fija **db_owner** pueden realizar todas las actividades de configuración y mantenimiento en la base de datos y también pueden quitar la base de datos.
db_securityadmin	Los miembros de la función de base de datos fija **db_securityadmin** pueden modificar la pertenencia a funciones y administrar permisos. Si se agregan entidades de seguridad a esta función, podría habilitarse un aumento de privilegios no deseado.
db_accessadmin	Los miembros de la función de base de datos fija **db_accessadmin** pueden agregar o quitar el acceso a la base de datos para inicios de sesión de Windows, grupos de Windows e inicios de sesión de SQL Server.
db_backupoperator	Los miembros de la función de base de datos fija **db_backupoperator** pueden crear copias de seguridad de la base de datos.
db_ddladmin	Los miembros de la función de base de datos fija **db_ddladmin** pueden ejecutar cualquier comando del lenguaje de definición de datos (DDL) en una base de datos.
db_datawriter	Los miembros de la función de base de datos fija **db_datawriter** pueden agregar, eliminar o cambiar datos en todas las tablas de usuario.
db_datareader	Los miembros de la función de base de datos fija **db_datareader** pueden leer todos los datos de todas las tablas de usuario.
db_denydatawriter	Los miembros de la función de base de datos fija **db_denydatawriter** no pueden agregar, modificar ni eliminar datos de tablas de usuario de una base de datos.
db_denydatareader	Los miembros de la función de base de datos fija **db_denydatareader** no pueden leer datos de las tablas de usuario dentro de una base de datos.
db_datareader	Los miembros de la función de base de datos fija **db_datareader** pueden leer todos los datos de todas las tablas de usuario.

❖ **Roles de Base de Datos**

Cada base de datos tiene un conjunto definido de roles de base de datos, al cual los usuarios de base de datos pueden ser adicionados. Estos roles de base de datos son únicos dentro de la base de datos. Mientras los permisos de los roles de base de datos no pueden ser alterados, nuevos roles de base de datos pueden ser creados.

db_accessadmin	Se le conceden: ALTER ANY USER, CREATE SCHEMA
db_accessadmin	Se le concede con la opción GRANT: CONNECT
db_backupoperator	Se le conceden: BACKUP DATABASE, BACKUP LOG, CHECKPOINT
db_datareader	Se le concede: SELECT
db_datawriter	Se le conceden: DELETE, INSERT, UPDATE
db_denydatareader	Se le deniega: SELECT
db_owner	Se le concede con la opción GRANT: CONTROL

Nota:

- **Roles**

  Create role (nombre) authorization (nombre_rol)

  create role hola authorization db_owner

  Alter role (nombre) with name=(Nuevo_nombre)

  alter role hola with name= olas

  Drop role (nombre)

  drop role olas

  Asignar un role a un usuario

  sp_addrolemember hola,davidd

- **Login**: Un "login" es un nombre y contraseña para acceder a un servidor. Los usuarios son de base de datos y se le asignan "logins".

  Create login: crear un inicio de sesión

  Create login (nombre) with password= 'contraseña'

  Drop login: eliminar el inicio de sesión

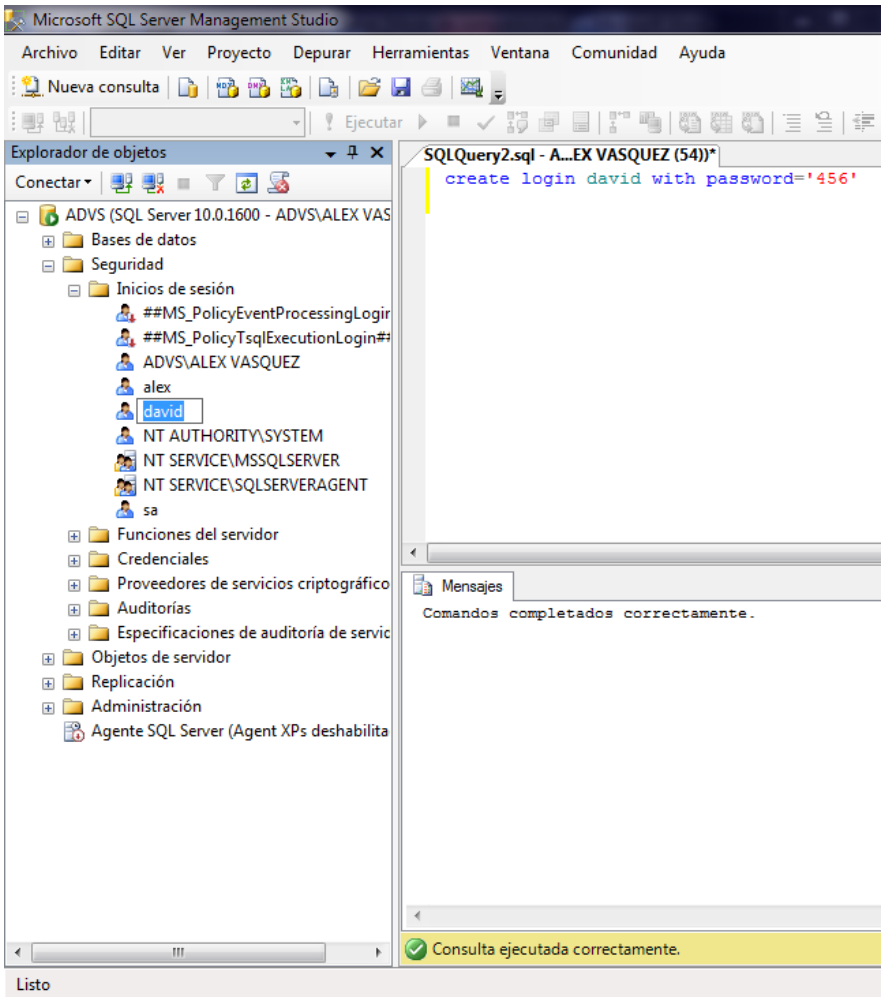
  Alter login: modificar el inicio de sesión

  Ejemplo:

  create login david with password='456'

  alter login david with password='123'

  drop login david



- **Usuario**: Cada "login" debe tener un usuario asociado. Si un "login" no posee usuario mapeado busca el usuario GUEST (deshabilitado por defecto). Un usuario sin "login" asociado se lo denomina un usuario huérfano.

  Guest: Permite a usuarios que no tienen cuenta en la Base Datos, que accedan a ella, pero hay que darle permiso explícitamente

  Créate user: crea el usuario

  Alter user: modifica el usuario

  Drop user: elimina el usuario, no si es propietario de objetos

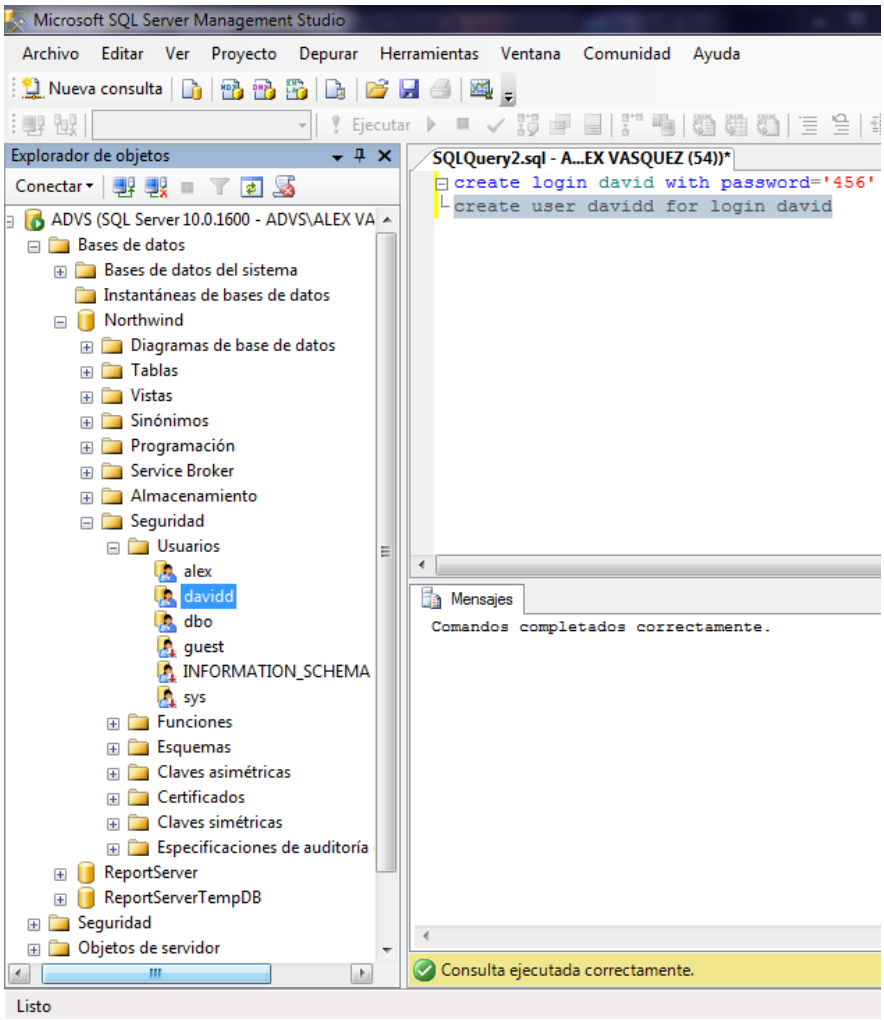
  Create user (nombre) for login (nombre_login)

  Ejemplo:

  create user davidd for login david

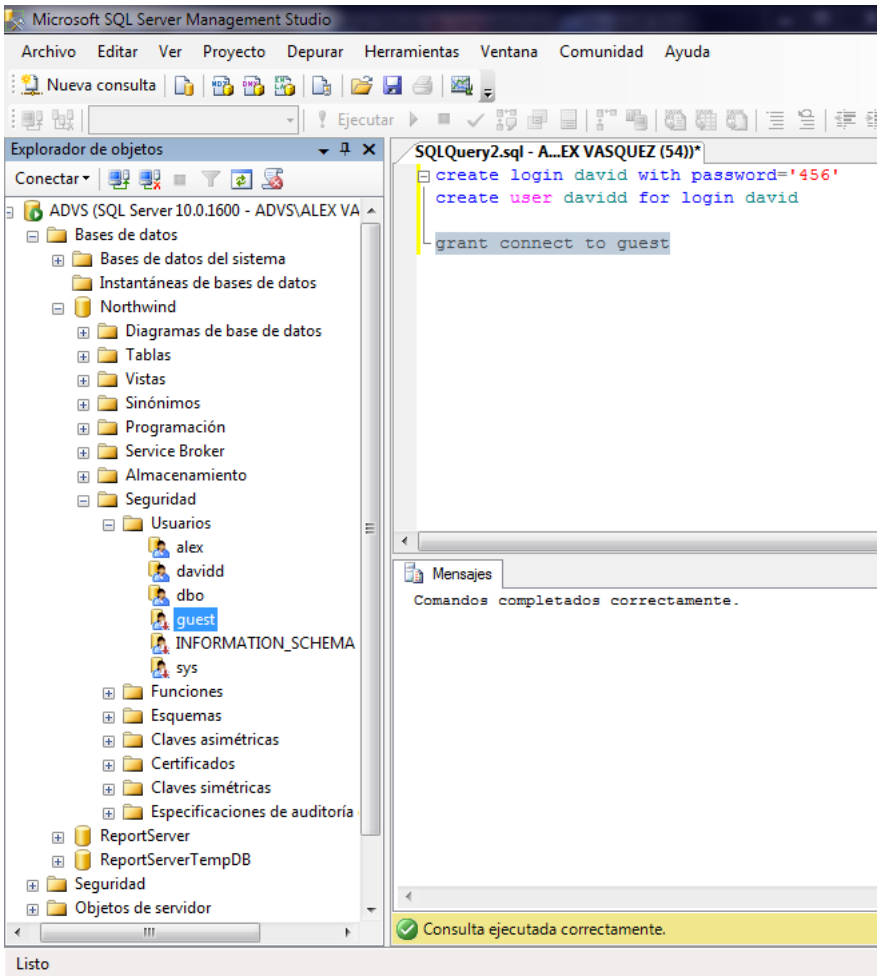
  alter user davidd with name= saman

  drop user saman



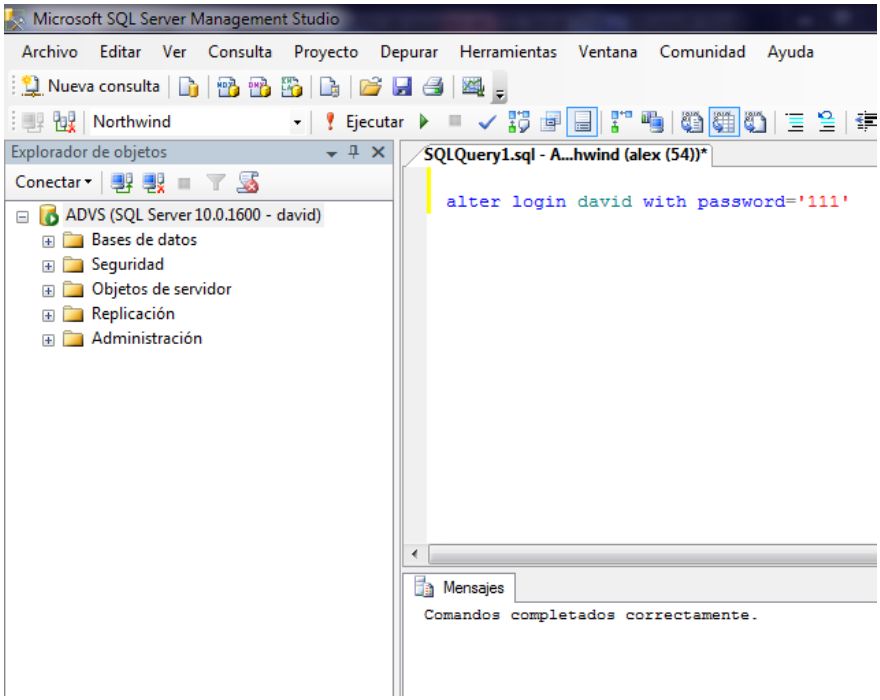
Cuando utilizamos el Guest

grant connect to guest



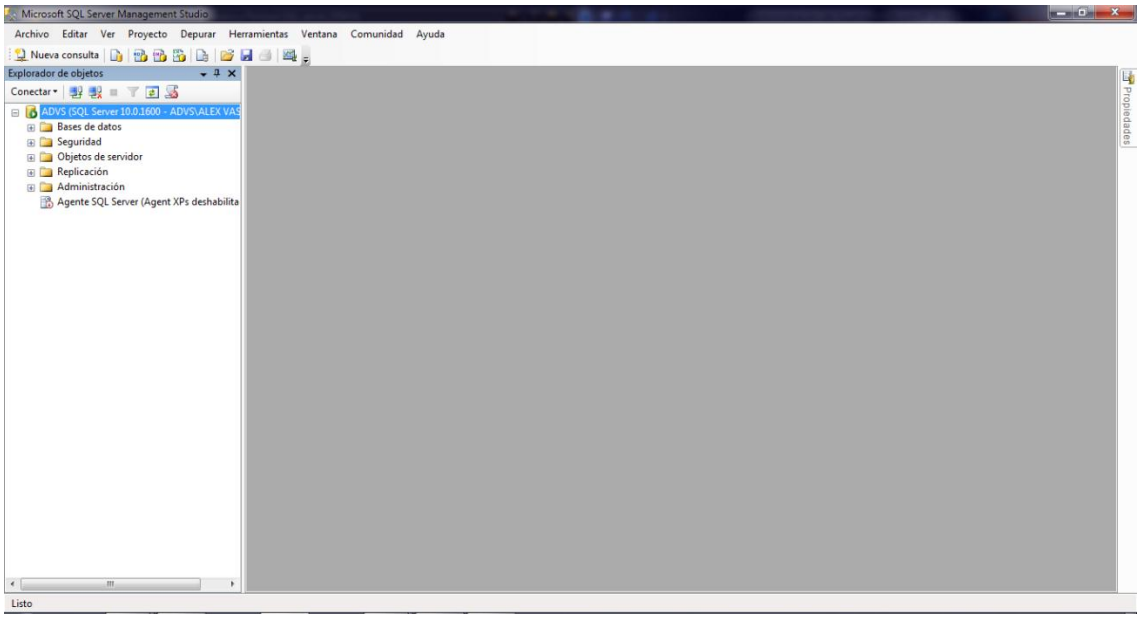
Para cambiar la contraseña de inicio de sesión:

alter login david with password='111'

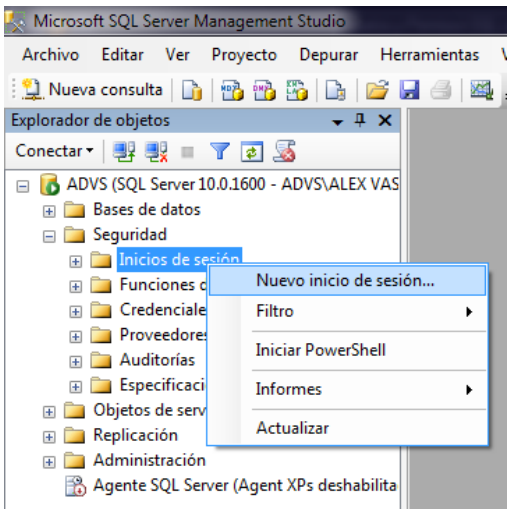


Ejercicio (*).

1. Ingresamos al SQL SERVER 2008



2. Vamos a la opción **seguridad**, desplegamos el signo (+), en **Inicios de sesión** damos clic derecho y damos clic en Nuevo Inicio de Sesión.

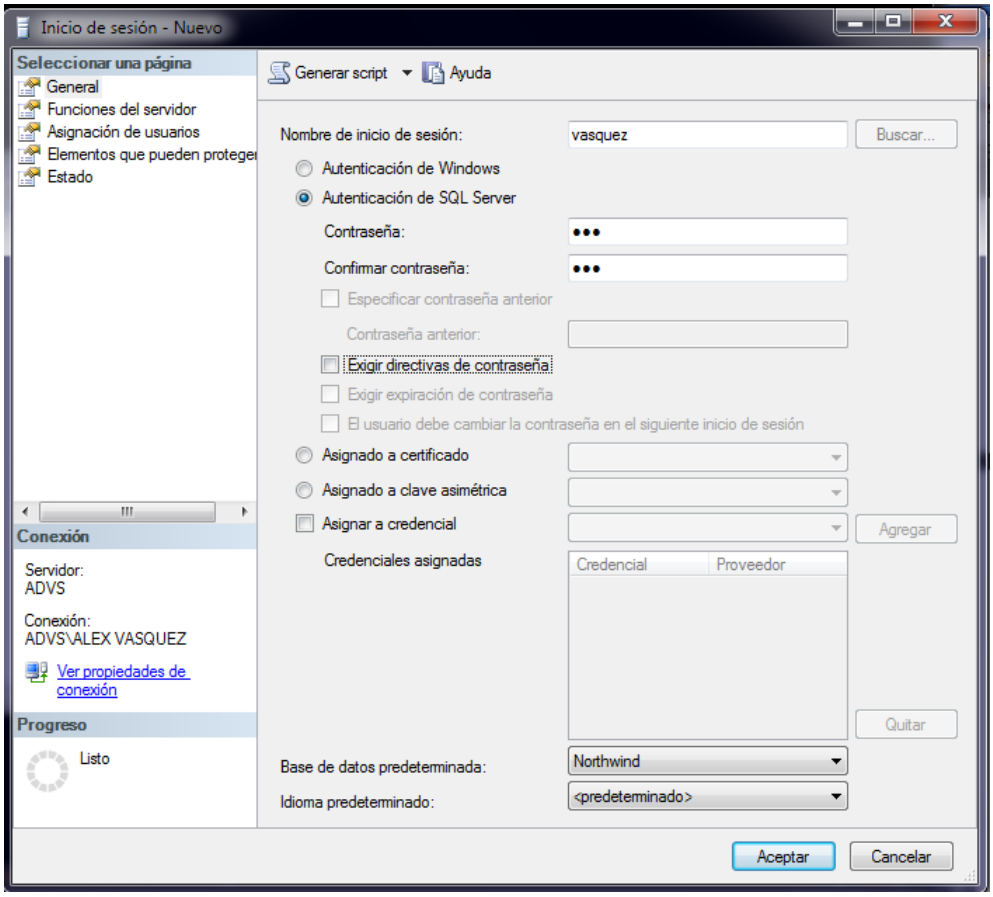


3. Se abrirá una nueva ventana en el cual designara el nombre de la sesión, hacemos que se autentifique con SQL Server, la contraseña que Ud. Desee y elegimos la base de datos que por defecto va estar asignado este usuario.

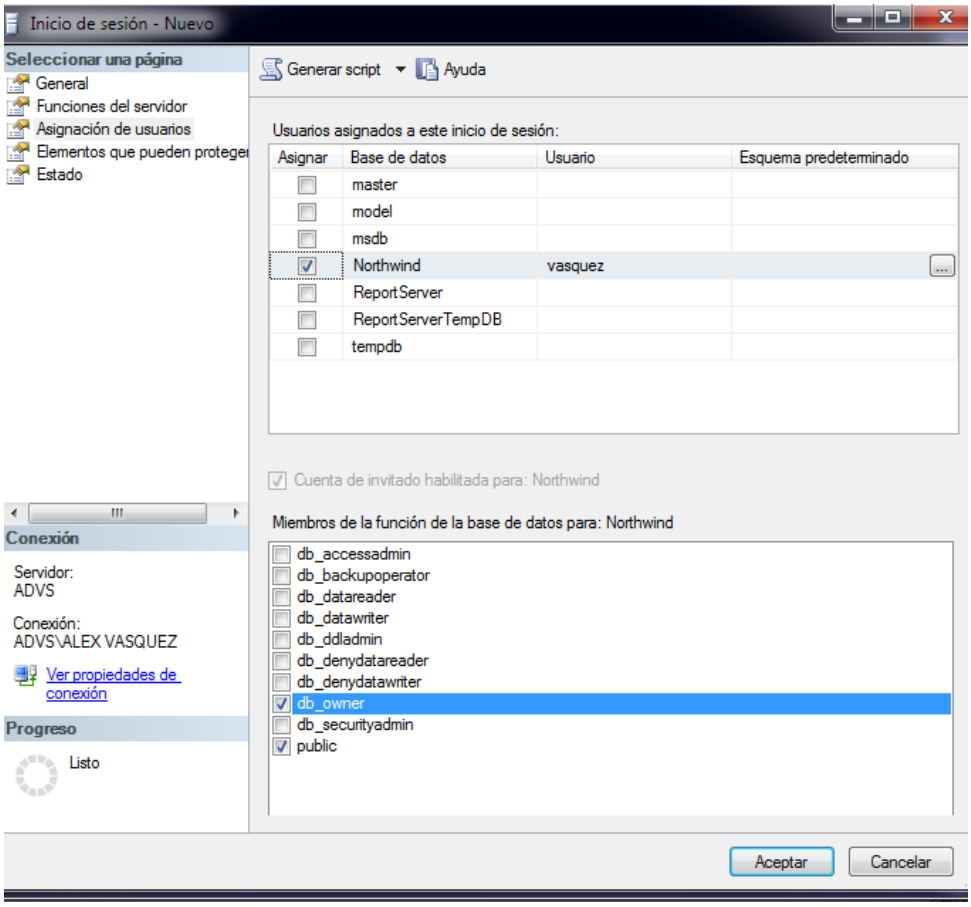
   Nombre de inicio de sesión: vasquez

   Contraseña: 987

   Base de datos por defecto: Northwind



4. Vamos a la Asignación de usuarios en la cual seleccionamos la base de datos a la cual el usuario va poder aceptar. En nuestro caso es la base de datos Nortwind.



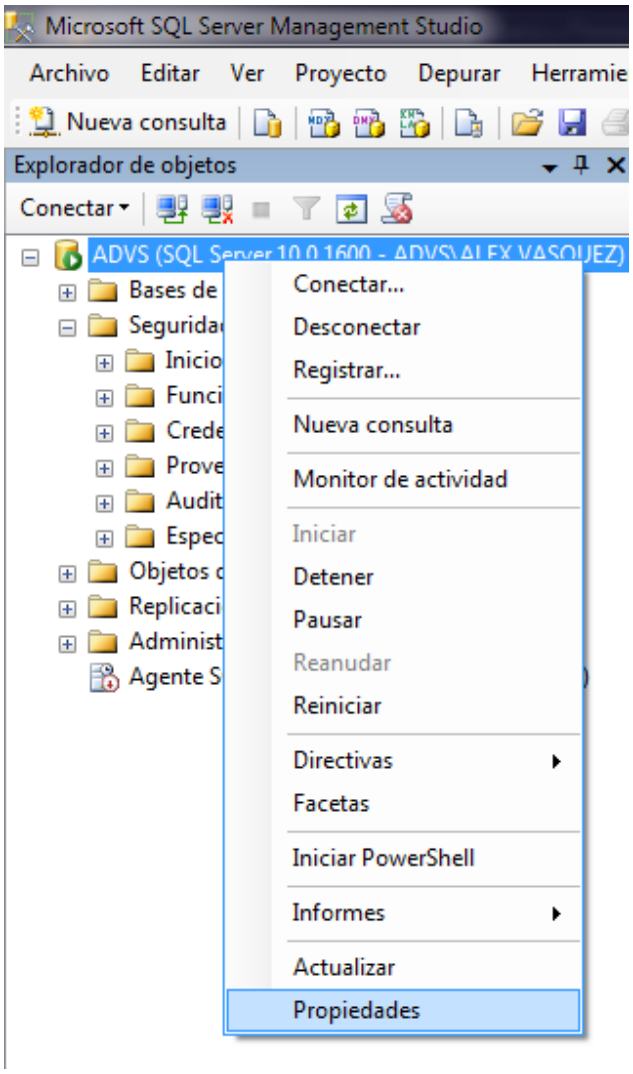
5. Antes de autenticarnos tenemos que verificar la autentificación con SQL SERVER. Para eso damos clic derecho en la primera opción en mi caso es: **ADVS (SQL SERVER 10.0.1600 – ADVS\ALEX VASQUEZ)**. Es así porque tengo un usuario en Windows.

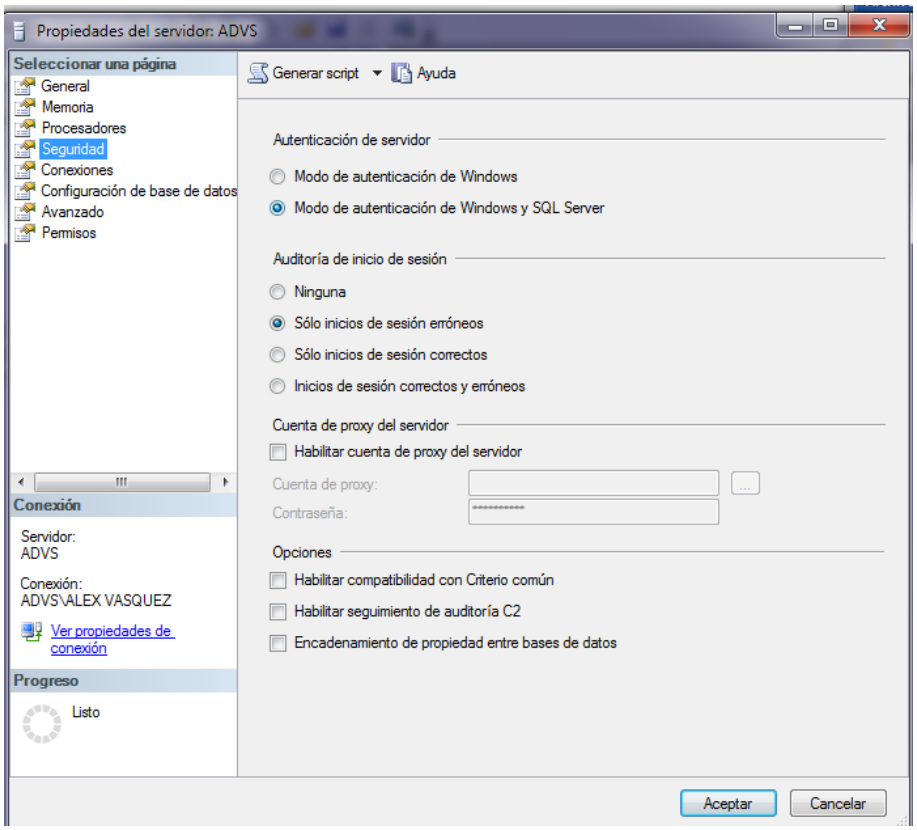
   Lo común seria:

   PC (SQL SERVER 10.0.1600 – PC\Admin) o

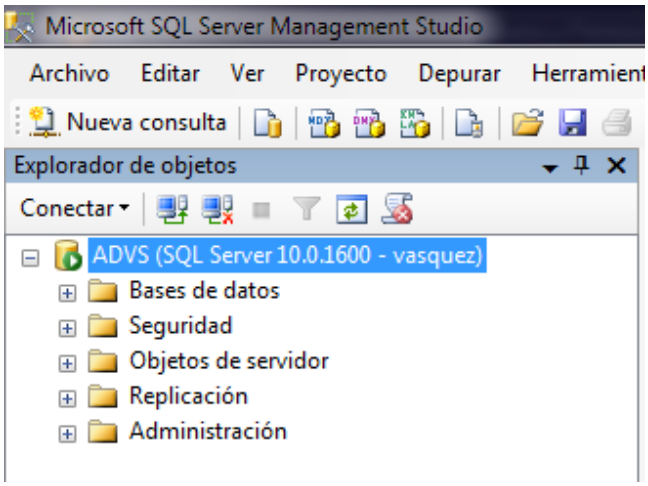
   PC (SQL SERVER 10.0.1600 – PC)

   Siguiendo hacemos clic en **propiedades**, luego en **seguridad**. Verificamos que este "checada" **Modo de autentificación de Windows y SQL Server**


Microsoft SQL Server Management Studio
Archivo
Editar
Ver
Proyecto
Depurar
Herramie
Nueva consulta
Explorador de objetos
Conectar
ADVS (SQL Server 10.0.1600 - ADVS\ALEX VASQUEZ)
Bases de
Seguridad
Inicio
Funci
Crede
Prove
Audit
Espec
Objetos d
Replicaci
Administ
Agente S
Conectar...
Desconectar
Registrar...
Nueva consulta
Monitor de actividad
Iniciar
Detener
Pausar
Reanudar
Reiniciar
Directivas
Facetas
Iniciar PowerShell
Informes
Actualizar
Propiedades



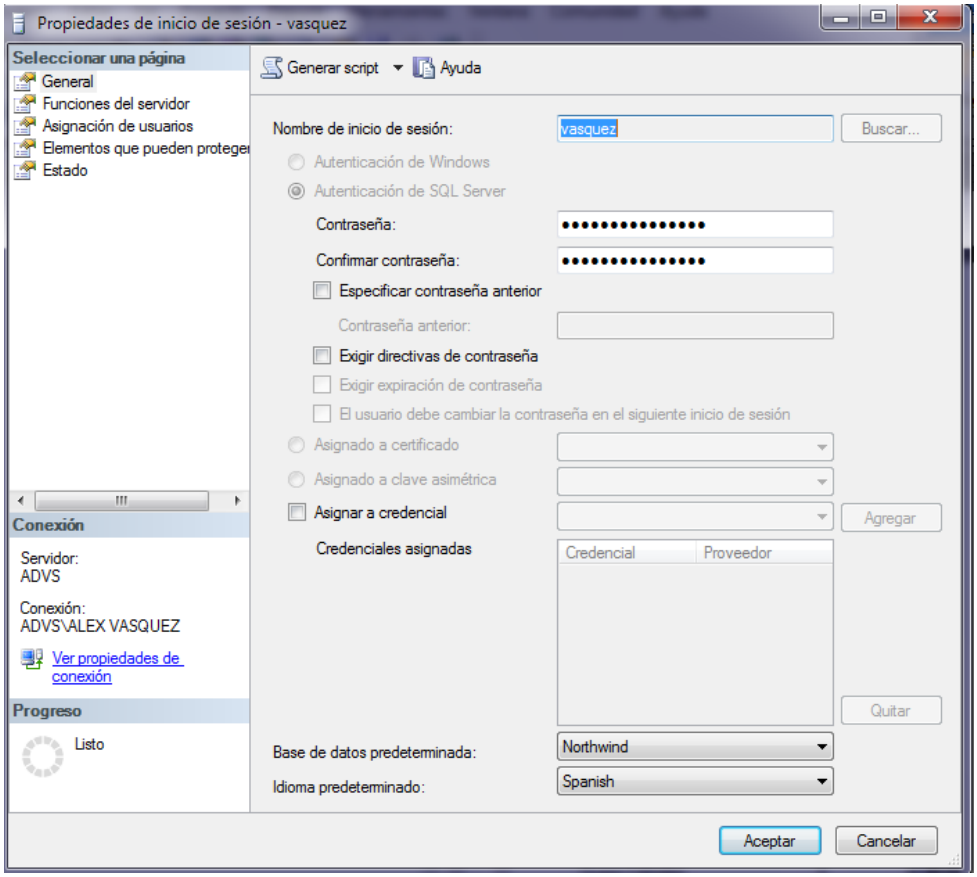
6. Vamos hacer una primera prueba vamos acceder con el nuevo usuario. Nos desconectamos y volvemos a conectarnos pero con Autentificación de SQL Server, en las casillas digitamos el inicio de sesión y la contraseña.



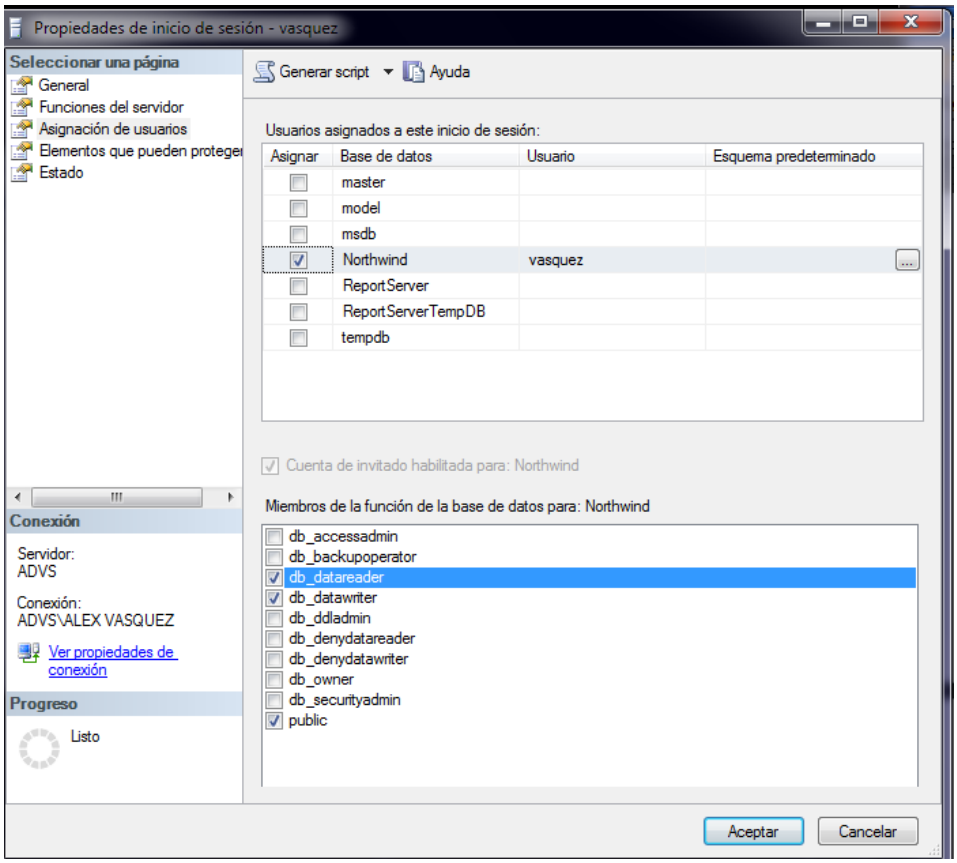
7. Vamos asignarle los permisos respectivos para lo cual volvemos a la Autentificación con Windows.

   Vamos a la pestaña seguridad>Inicio de sesión>vasquez

   Le damos doble clic y se abre nuevamente la ventana



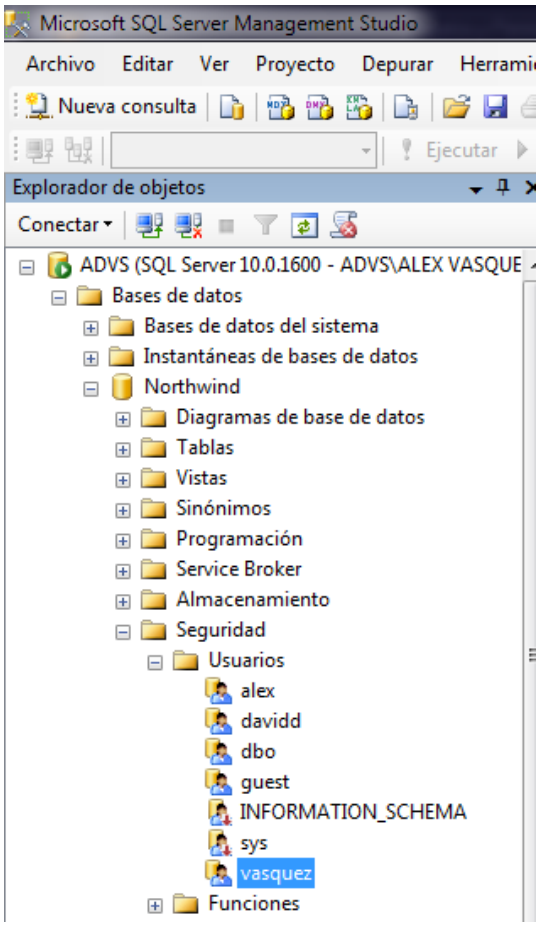
8. Vamos a la asignación de usuarios y checamos las opciones db_datareader y db_datawriter; es decir que el usuario podrá leer y escribir datos sobre la BD.



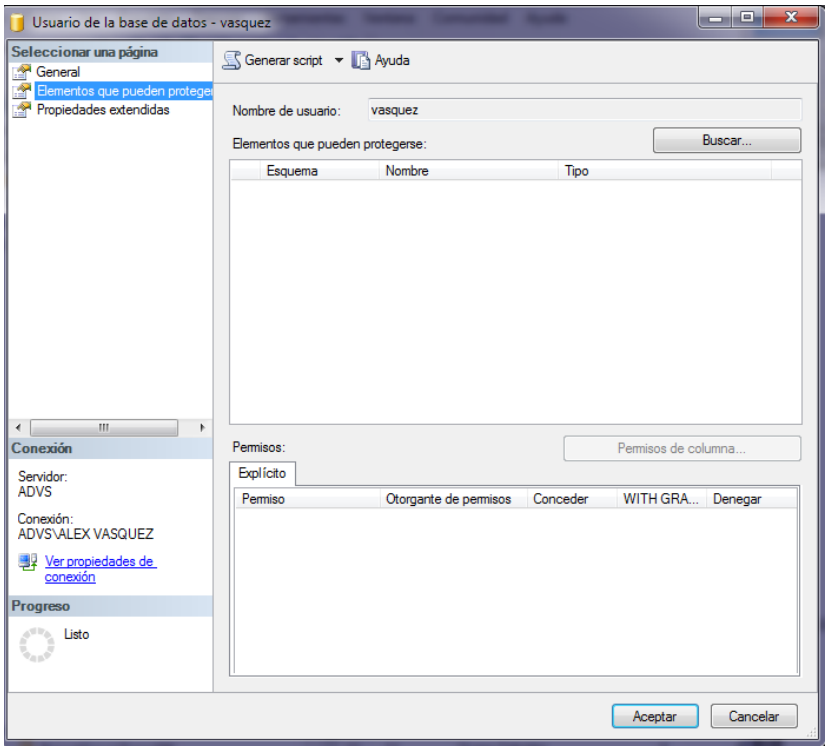
9. Luego vamos a la base de datos a la que se tiene permiso:

   Base de datos > Northwind > Seguridad > usuarios

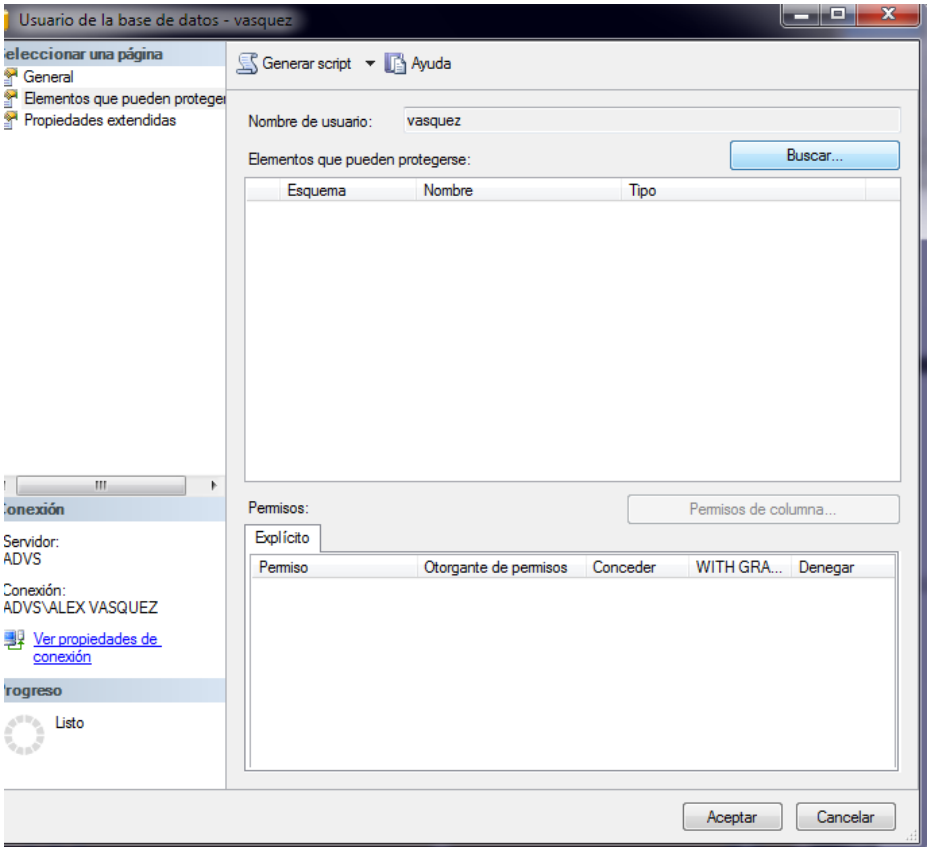
   Y damos doble clic en el usuario que se ha creado

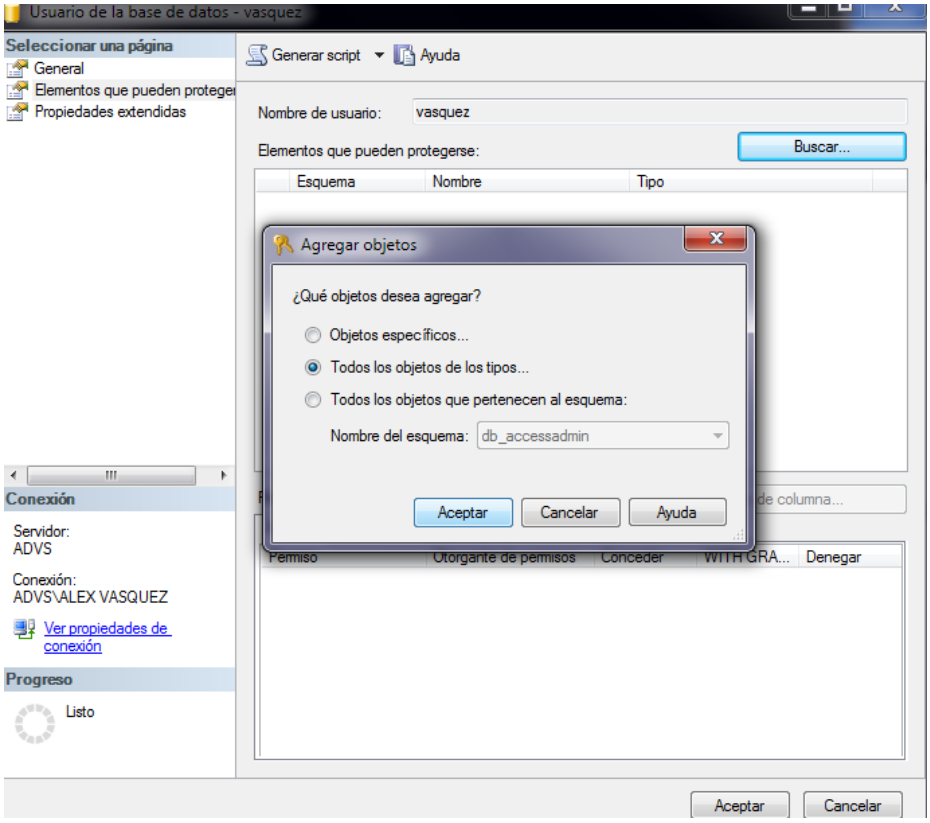


10. De allí vamos a la opción elementos que pueden protegerse



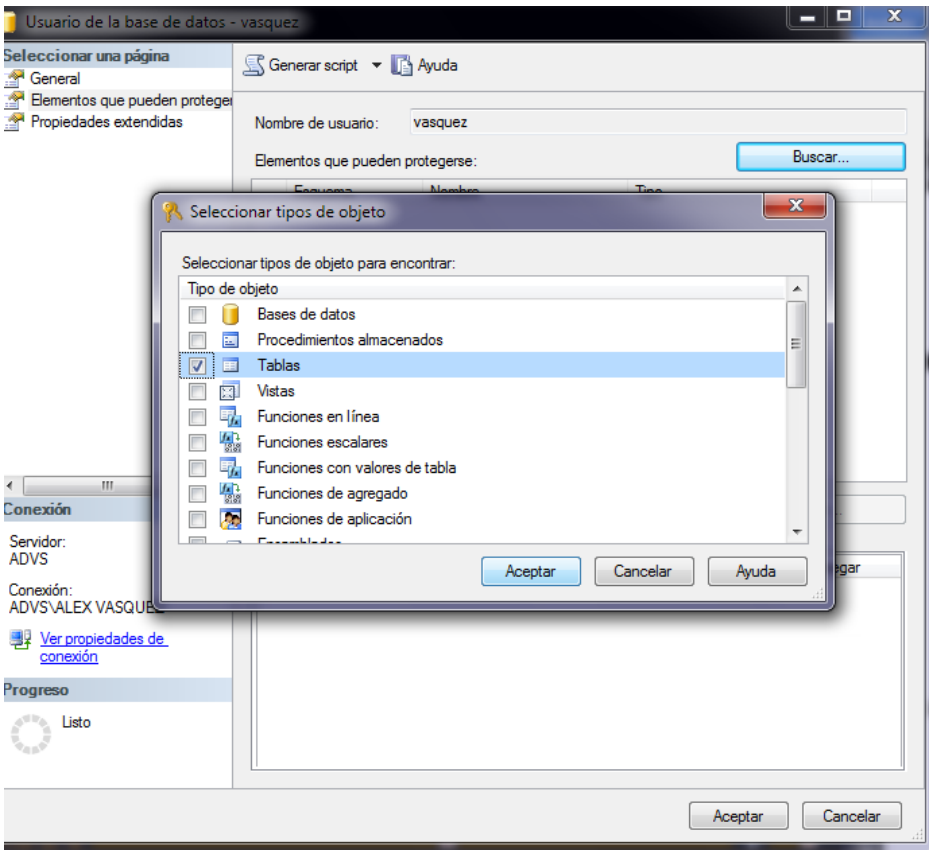
Clic en buscar, checamos la opción Todos los objetos de los tipos y acepamos



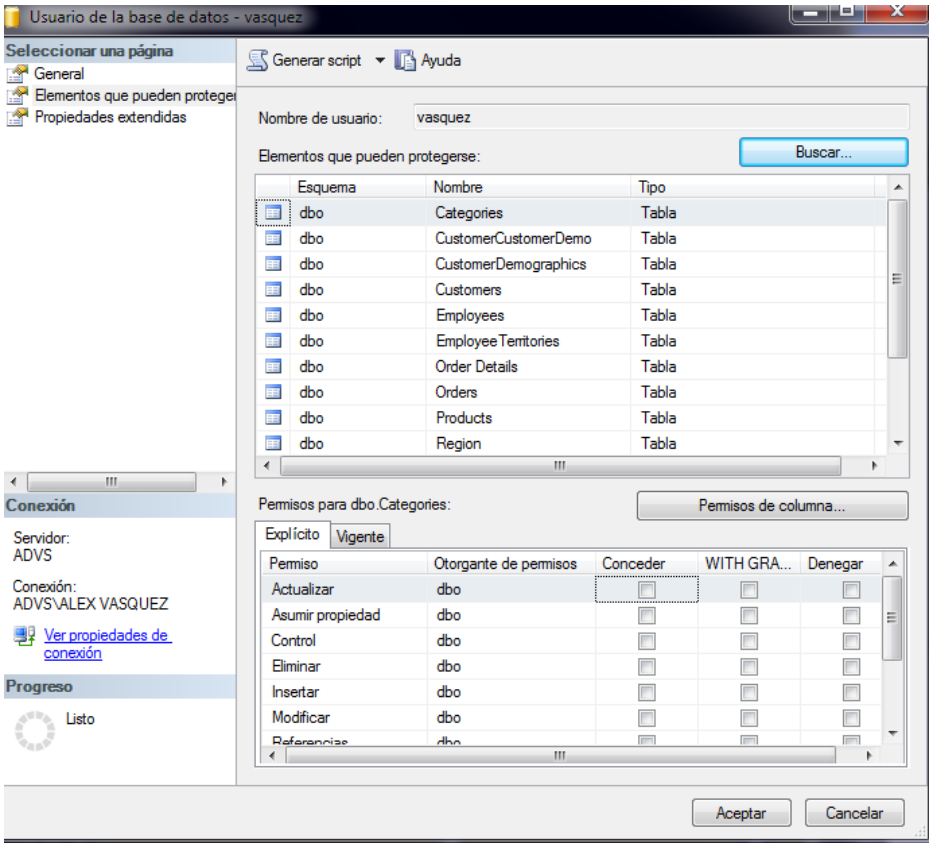


Luego seleccionamos lo que se quiere proteger.

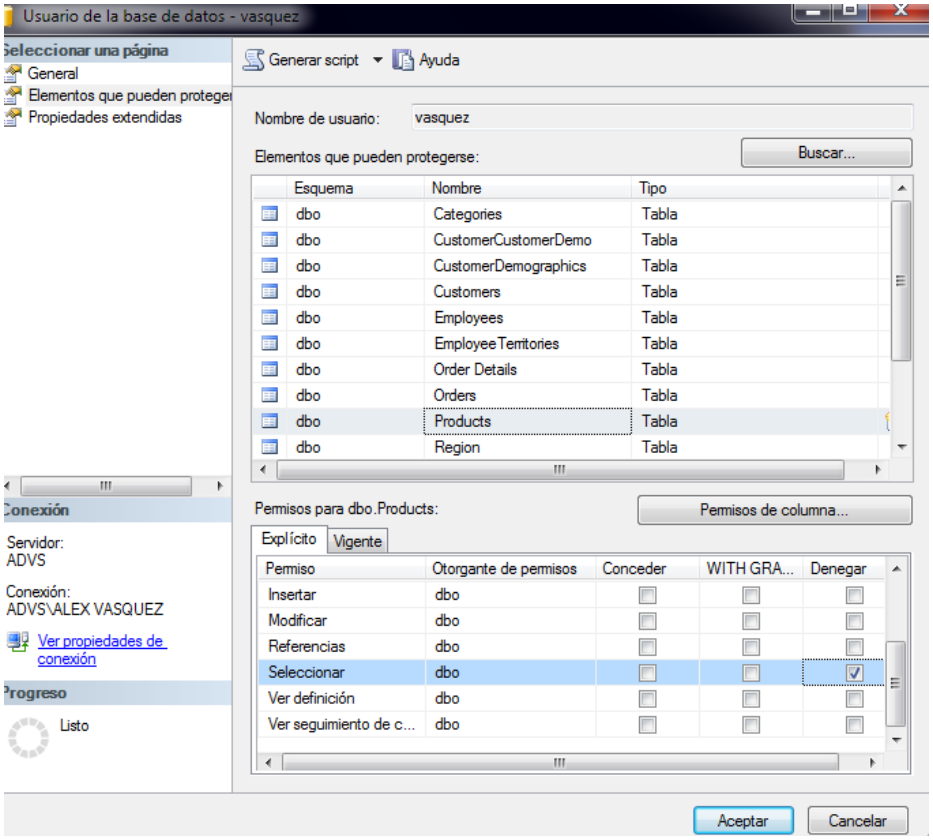
En este caso solo será tablas



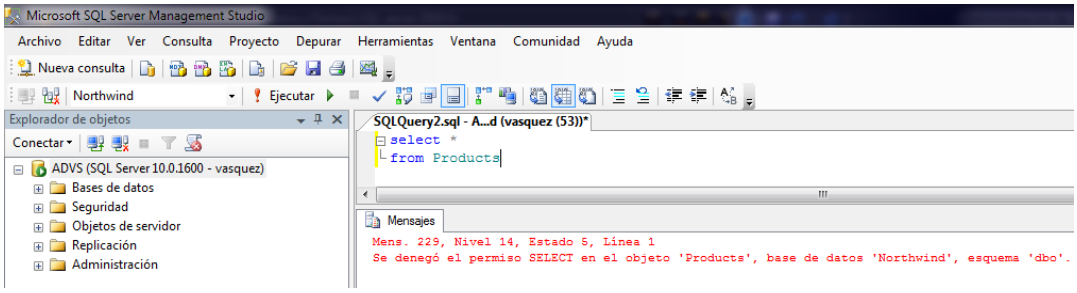
Al aceptar aparecerán todas las tablas en el cual podrá restringir o no, lo que Ud. prefiera:



En nuestro caso por ejemplo vamos a restringir la tabla **Products**, no se le va a poder hacer un **select**.



Al no tener el permiso me va arrojar un error



La consulta:

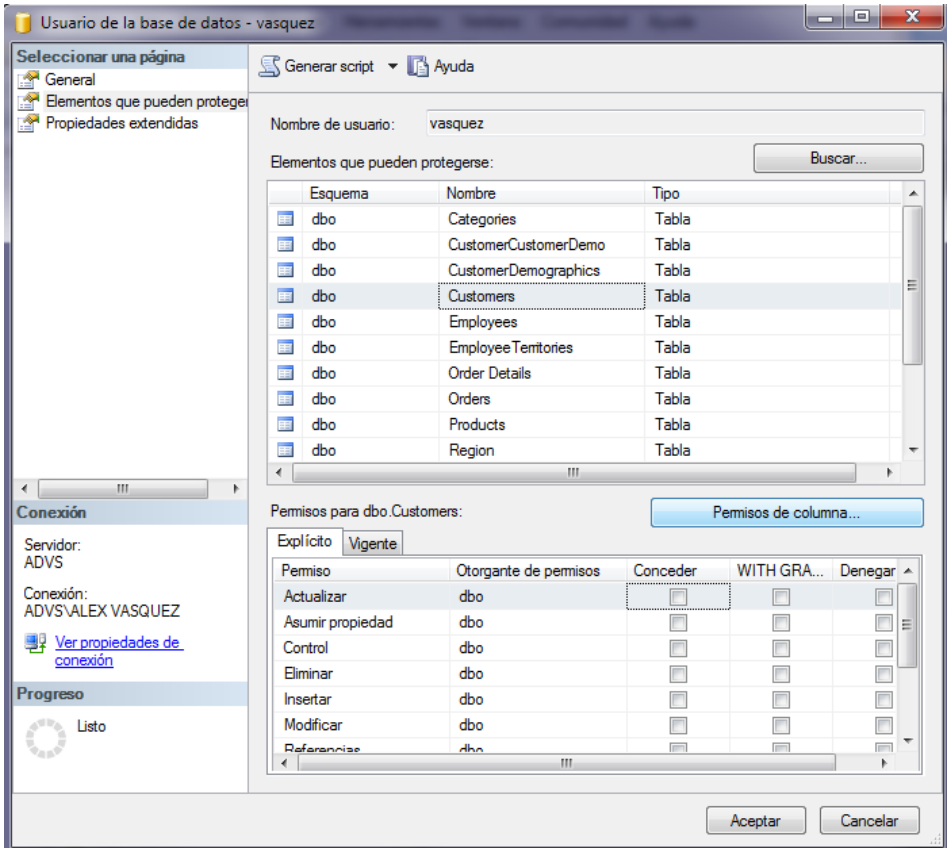
```
select *
from Products
```

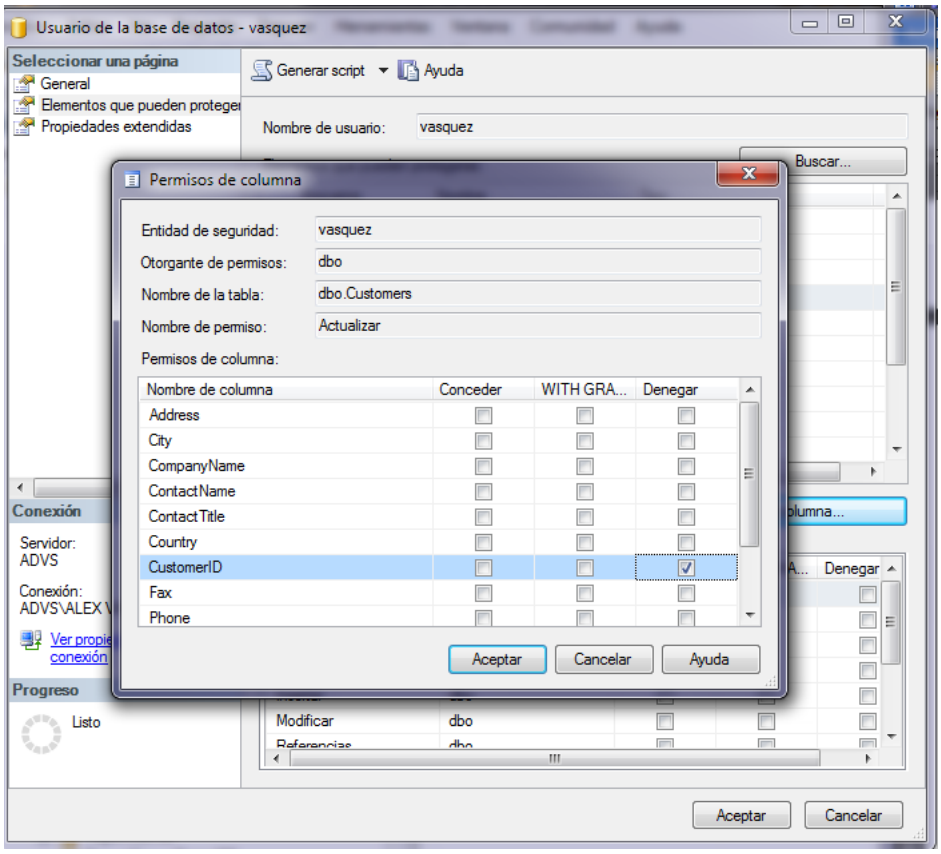
El mensaje:

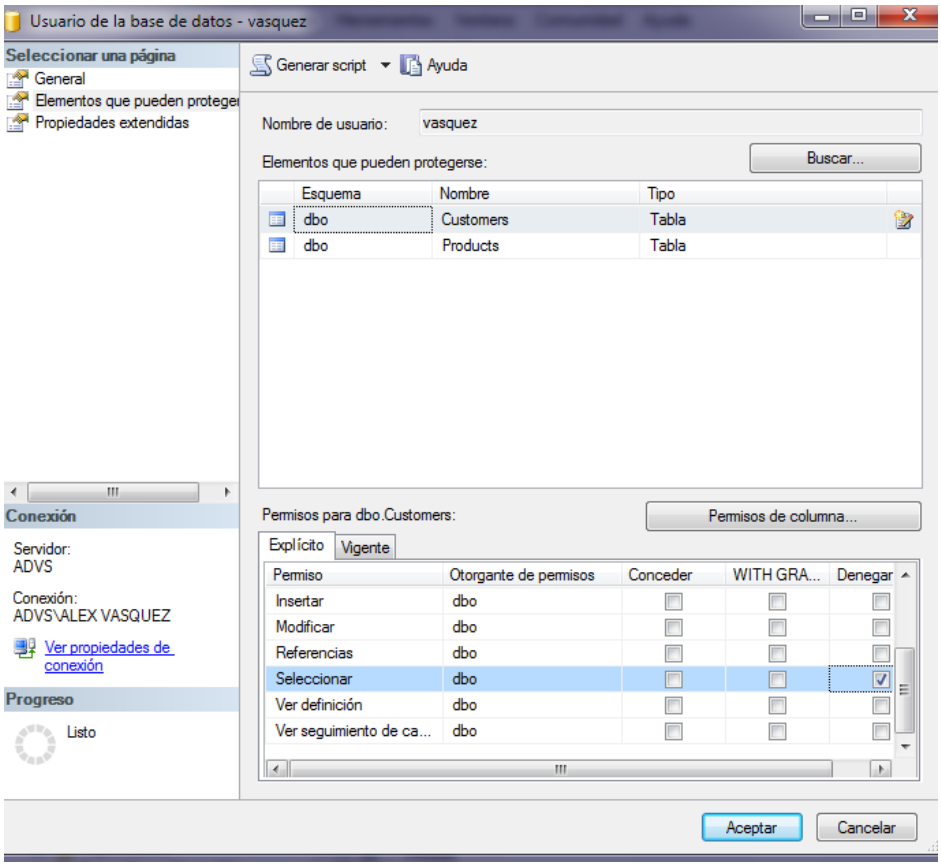
Mens. 229, Nivel 14, Estado 5, Línea 1
Se denegó el permiso SELECT en el objeto 'Products', base de datos 'Northwind', esquema 'dbo'.

A la vez si queremos podemos denegar el permiso al campo o columna.

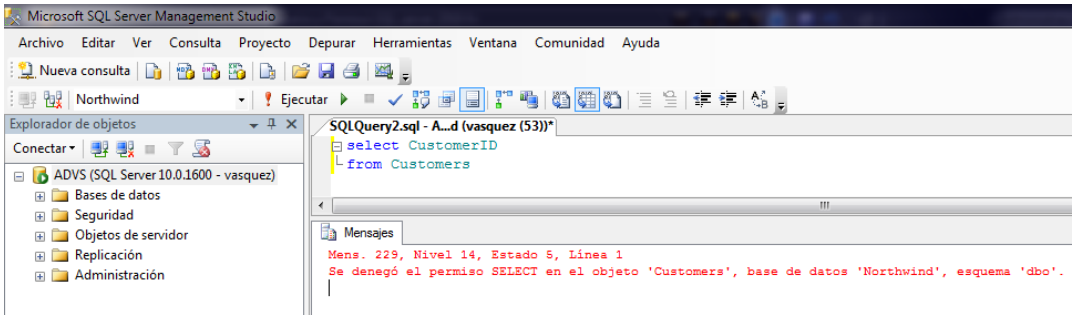
En este caso vamos a denegar el **select** al campo **CustomerID** de la tabla **Customers**




Usuario de la base de datos - vasquez
Seleccionar una página
General
Elementos que pueden proteger
Propiedades extendidas
Generar script
Ayuda
Nombre de usuario:
vasquez
Buscar...
Permisos de columna
Entidad de seguridad:
vasquez
Otorgante de permisos:
dbo
Nombre de la tabla:
dbo.Customers
Nombre de permiso:
Actualizar
Permisos de columna:
Nombre de columna
Conceder
WITH GRA...
Denegar
Address
City
CompanyName
ContactName
ContactTitle
Country
CustomerID
Fax
Phone
Aceptar
Cancelar
Ayuda
Conexión
Servidor:
ADVS
Conexión:
Progreso
Listo
Modificar
dbo
Aceptar
Cancelar


Usuario de la base de datos - vasquez
Seleccionar una página
General
Elementos que pueden proteger
Propiedades extendidas
Generar script
Ayuda
Nombre de usuario:
vasquez
Elementos que pueden protegerse:
Buscar...
Esquema
Nombre
Tipo
dbo
Customers
Tabla
dbo
Products
Tabla
Conexión
Servidor:
ADVS
Conexión:
ADVS\ALEX VASQUEZ
Ver propiedades de conexión
Progreso
Listo
Permisos para dbo.Customers:
Permisos de columna...
Explícito
Vigente
Permiso
Otorgante de permisos
Conceder
WITH GRA...
Denegar
Insertar
dbo
Modificar
dbo
Referencias
dbo
Seleccionar
dbo
Ver definición
dbo
Ver seguimiento de ca...
dbo
Aceptar
Cancelar

Nos va ha generar un error



Consulta 1:

```
select CustomerID
from Customers
```

Consulta 2:

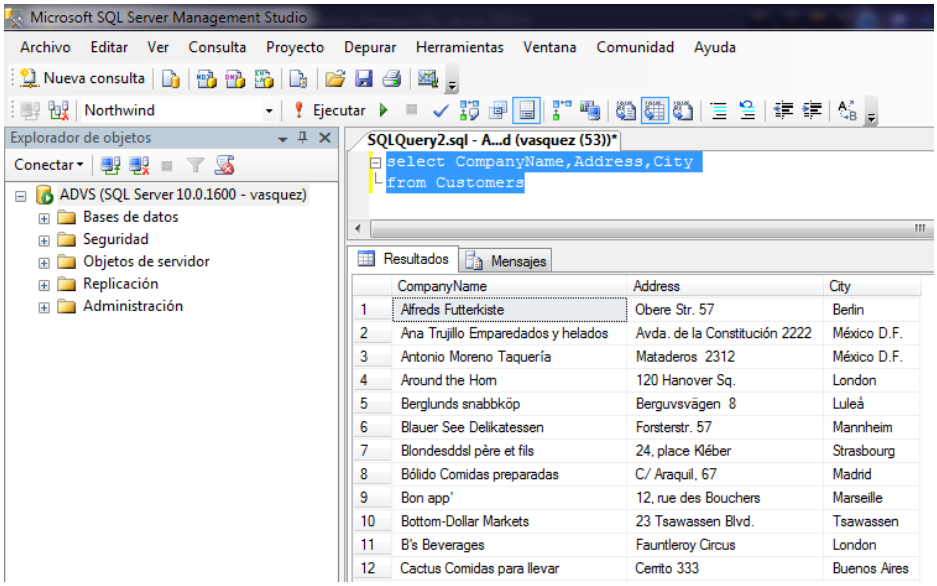
```
select *
from Customers
```

El mensaje:

Mens. 229, Nivel 14, Estado 5, Línea 1

Se denegó el permiso SELECT en el objeto 'Customers', base de datos 'Northwind', esquema 'dbo'.

Consulta 3: En esta consulta no mencionamos el campo restringido así que no va a generar error



3. Usuarios (Autenticación de Windows)

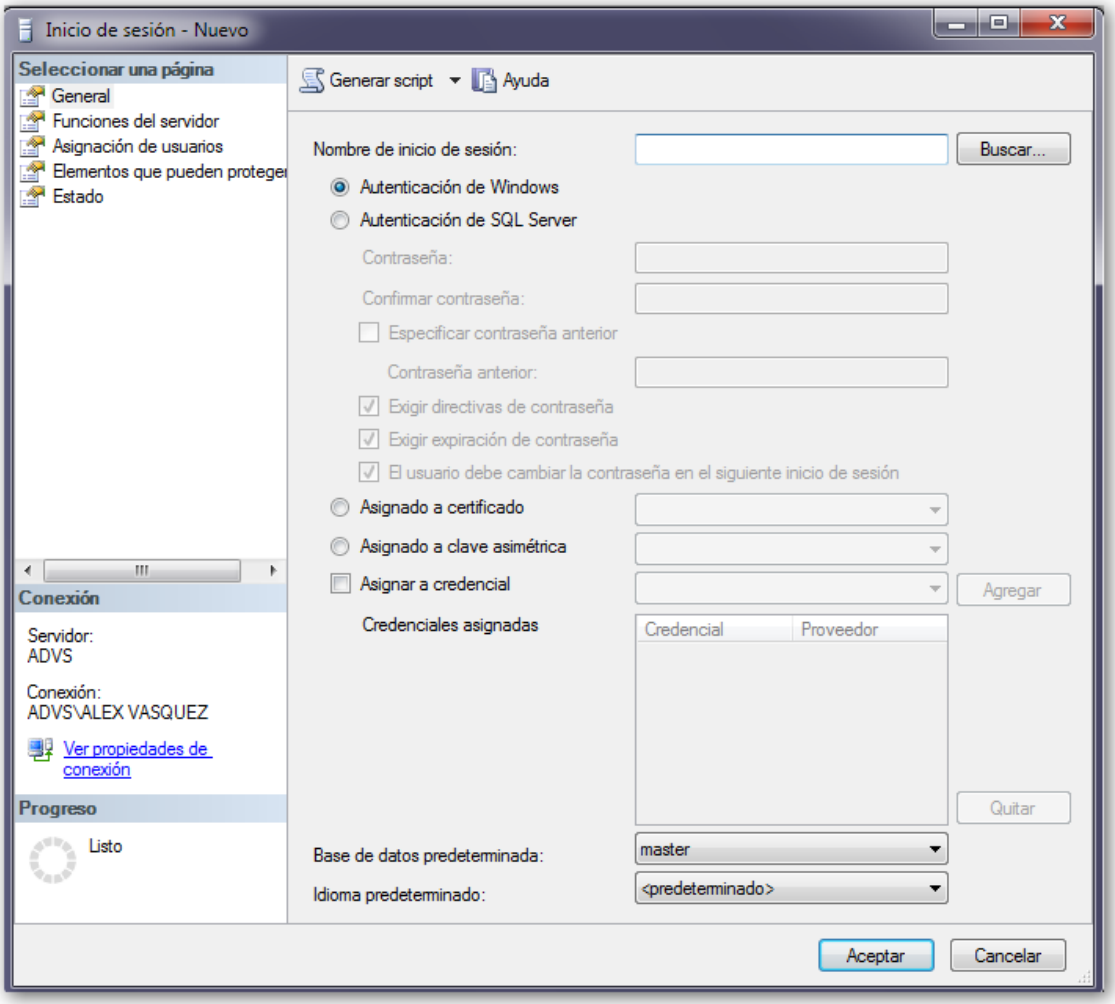
Lo anteriormente realizado es bajo la AUTENTICACIÓN DE SQL SERVER, para este caso lo que cambia es que tienes que crear un usuario y los permisos y soles son iguales. Ahora lo único que haremos es crear solo el usuario y el funcionamiento como si fuera un Windows server, nada más. Repito los permisos y los roles se hacen de la misma manera líneas arriba.

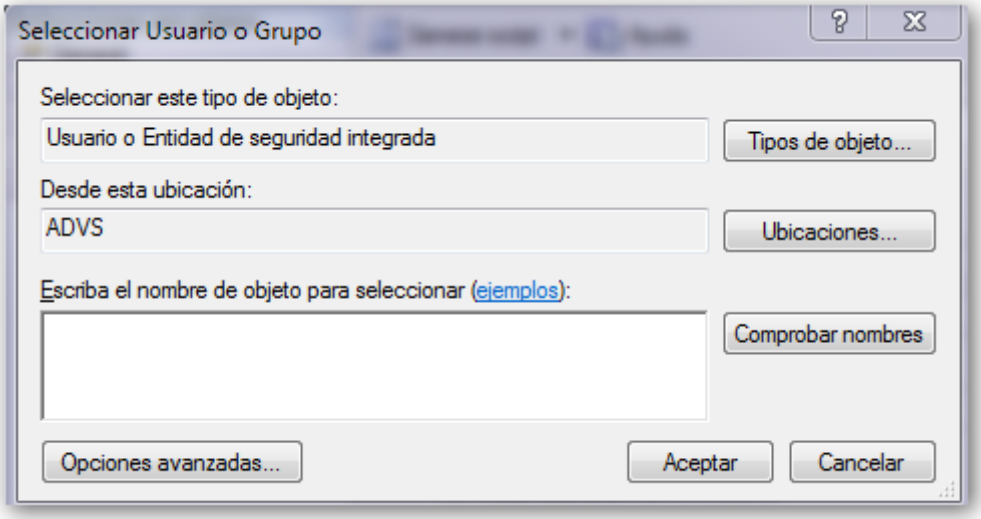
Inicio > Panel de Control > Cuentas de Usuario… > Agregar o quitar cuentas de usuario > Crear una nueva cuenta

Se le da un nombre cualesquiera

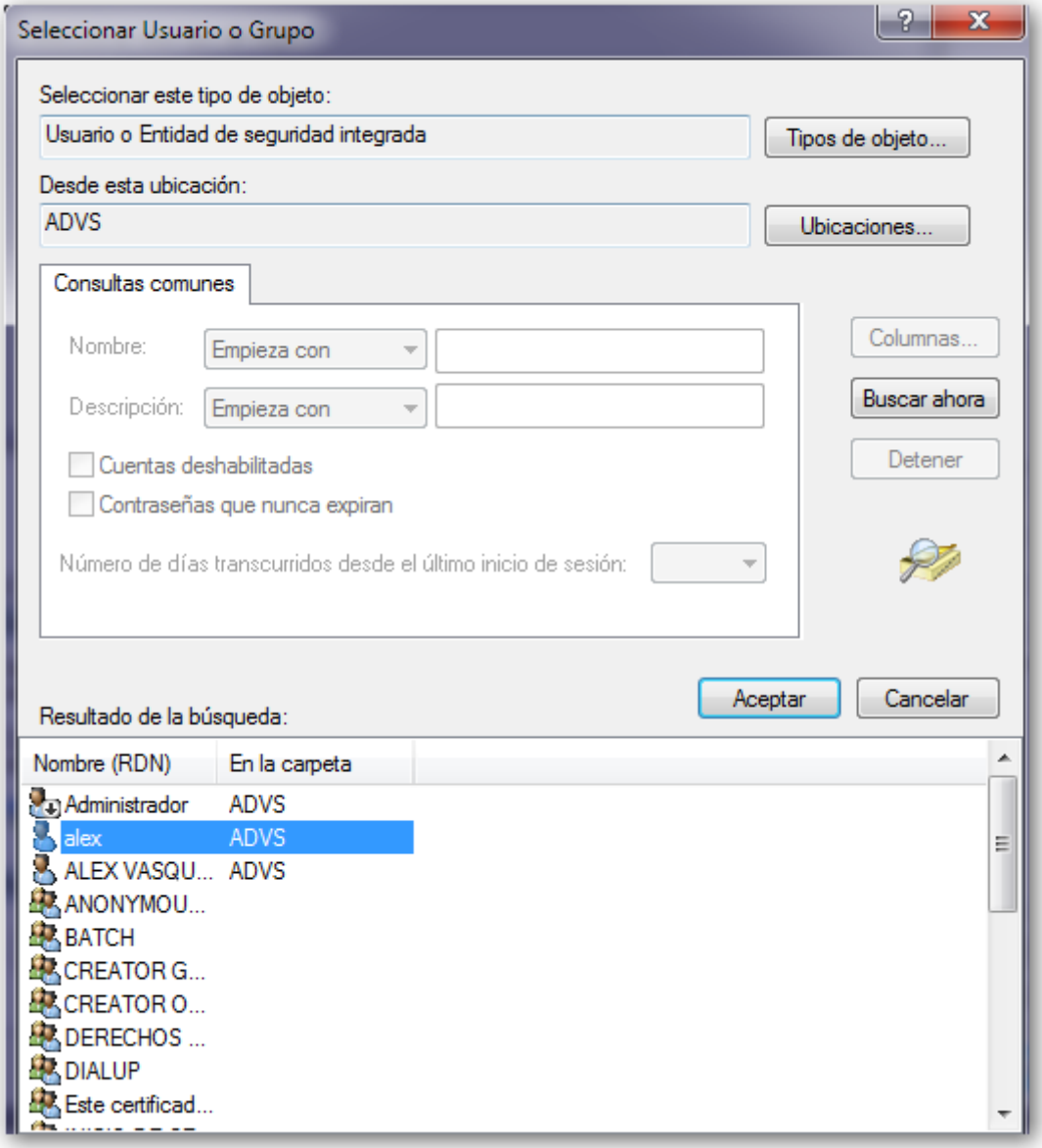
Se selecciona un usuario estándar. Para no tener dos administradores.

Creamos el inicio de sesión en el SQL SERVER

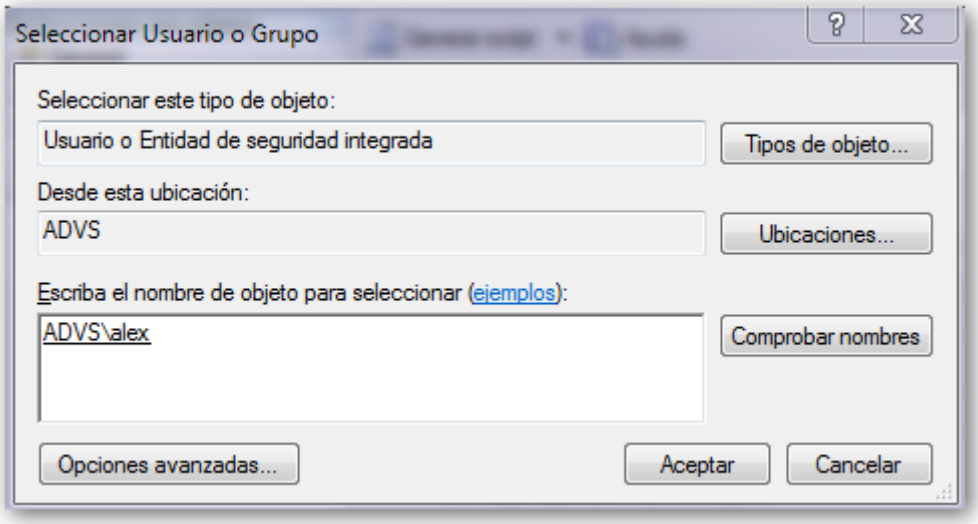




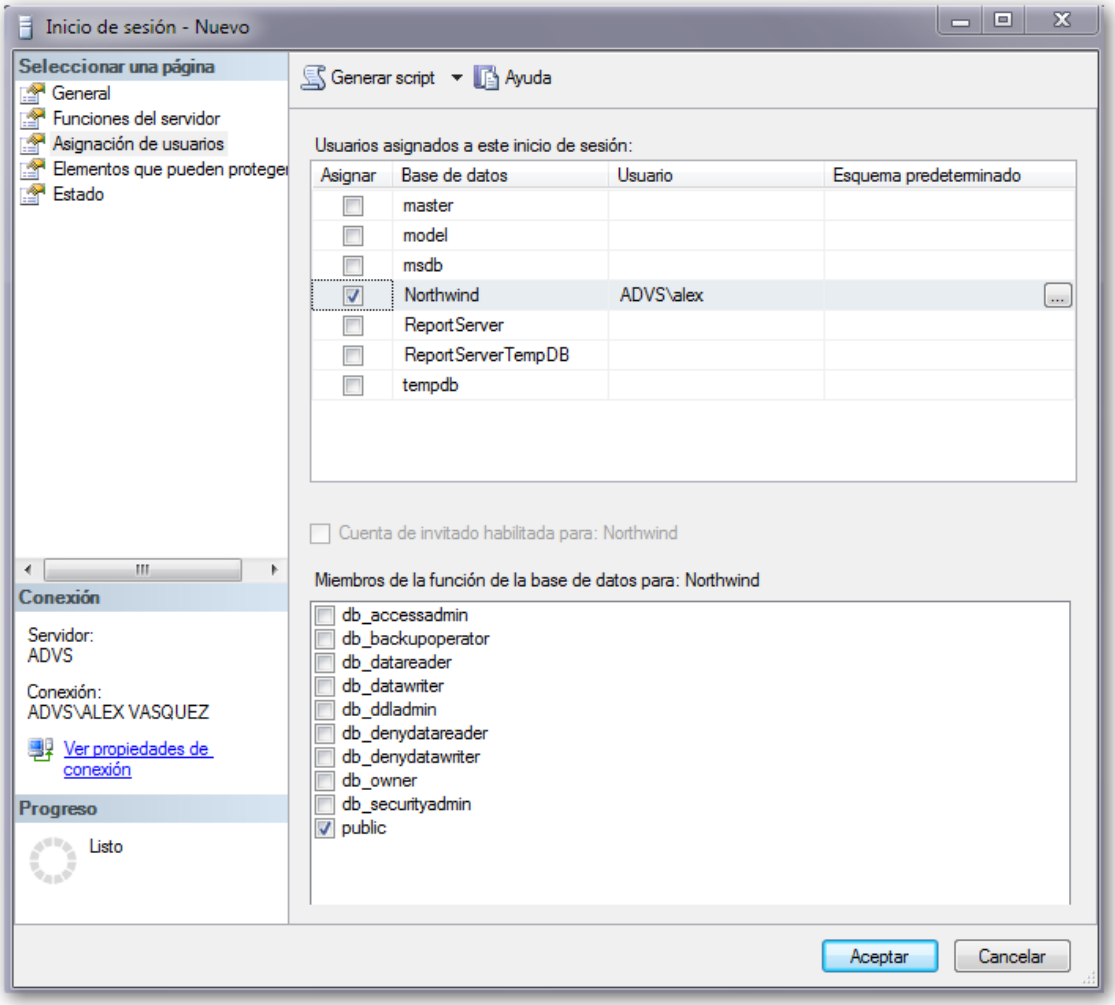
Hacemos clic en **Opciones avanzadas**.



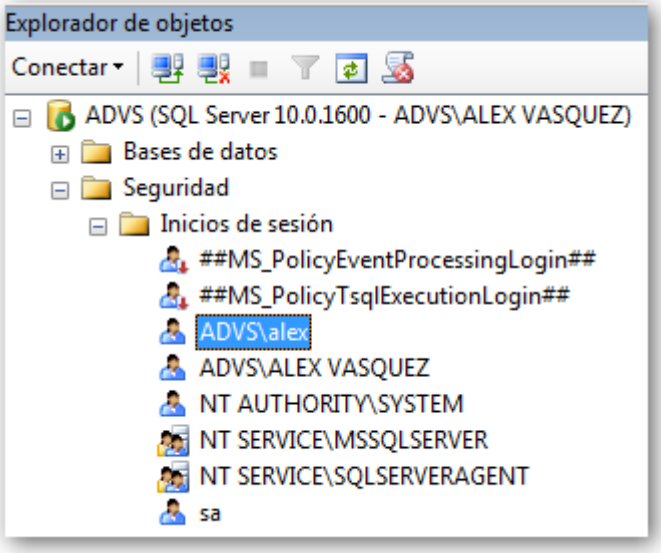
Selecciono el usuario creado y acepto



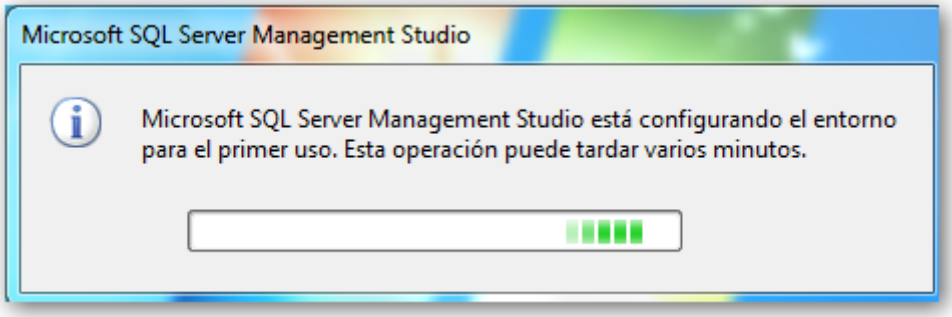
Selecciono la base de datos.

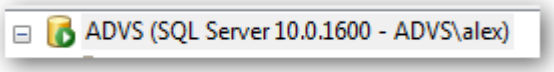


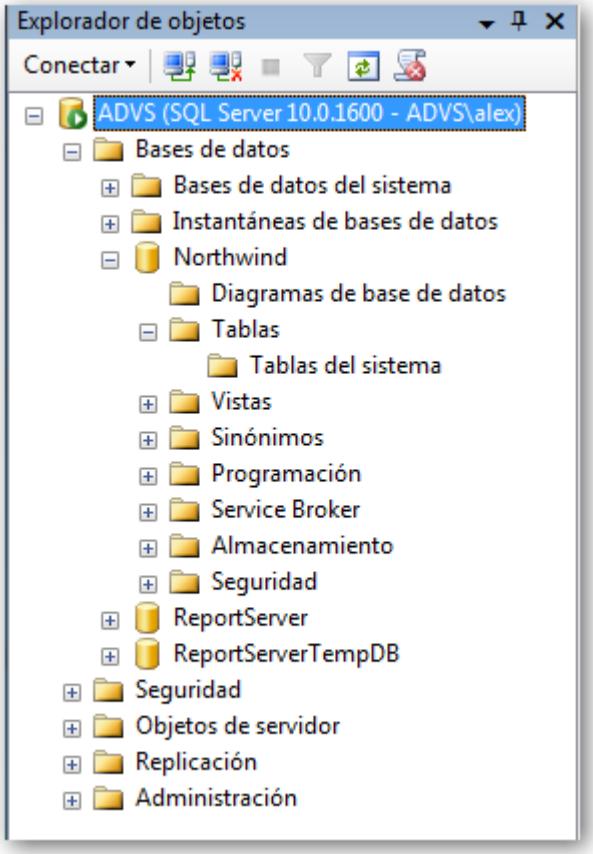
Verifico que el inicio de sesión este creada



Iniciamos sesión SQL SERVER con el usuario creado.



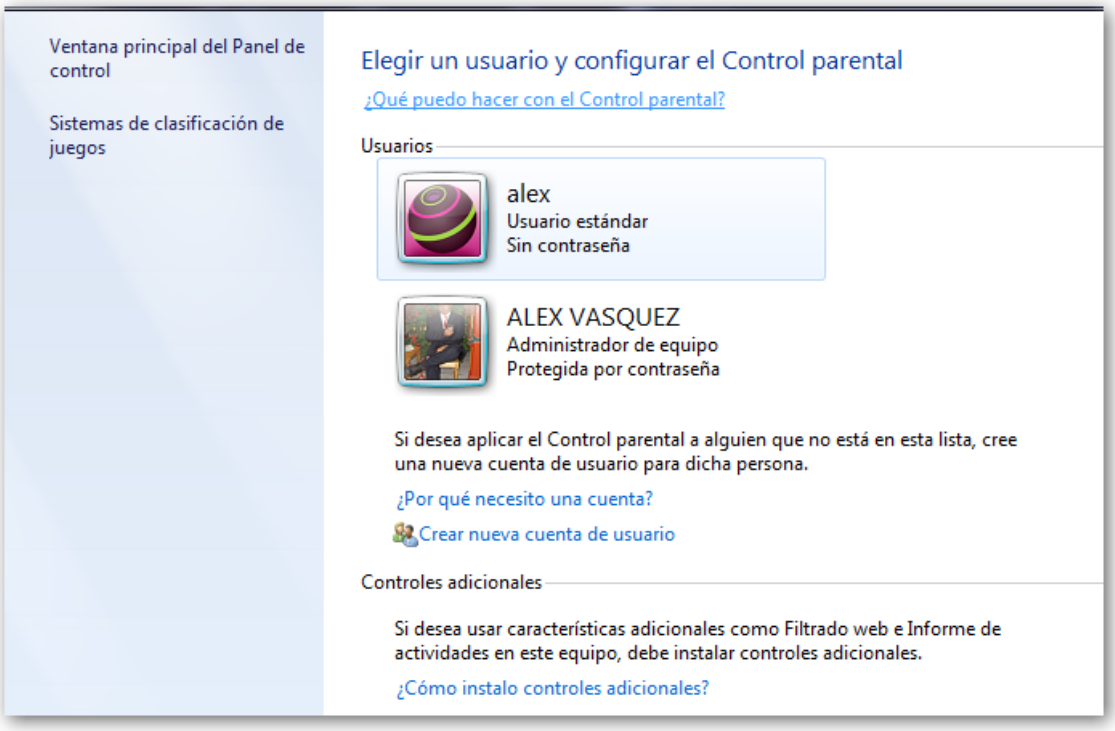




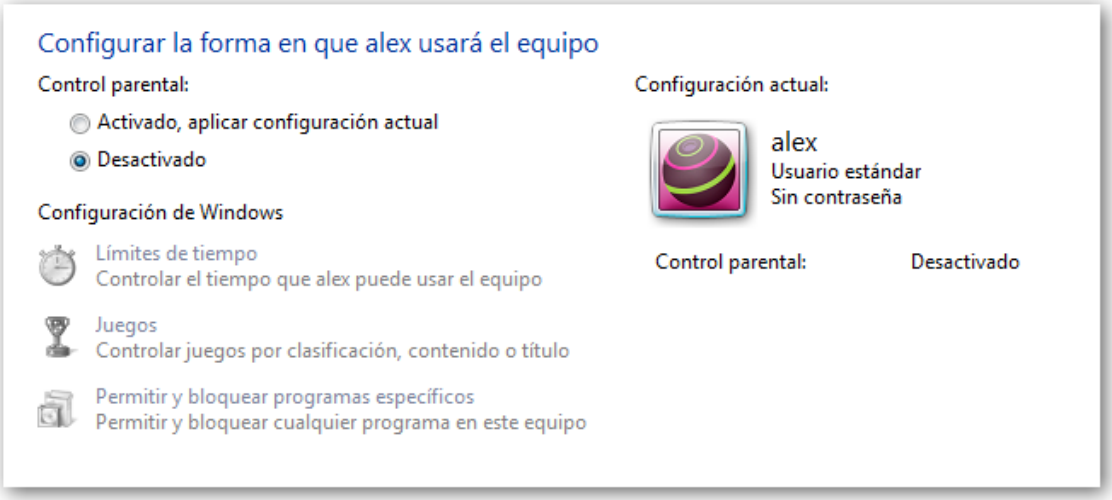
Ahora no aparece ninguna tabla porque aún falta dar los permisos eso se hace del mismo modo que para la AUTENTICACIÓN DE SQL SERVER.

Ahora le vamos a configurar como si fuese un Windows Server.

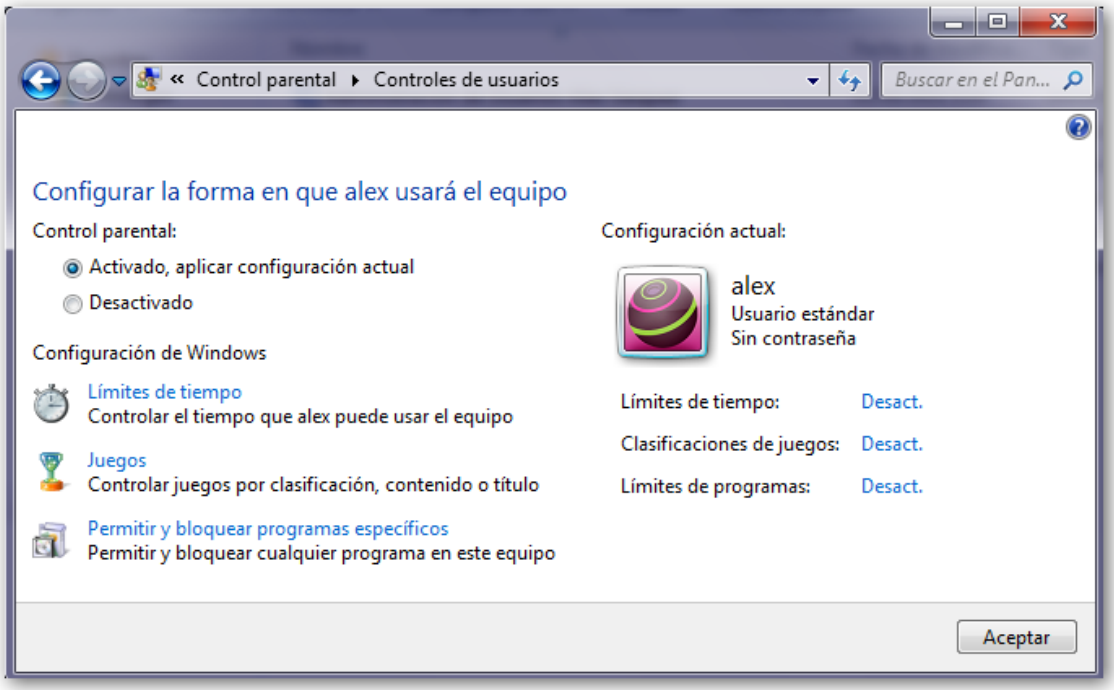
Vamos a la cuenta recién creada, le damos clic en **configurar control parental**, me lleva a una ventana donde aparecen todas las cuentas que tiene el usuario, hago clic en la recién creada.



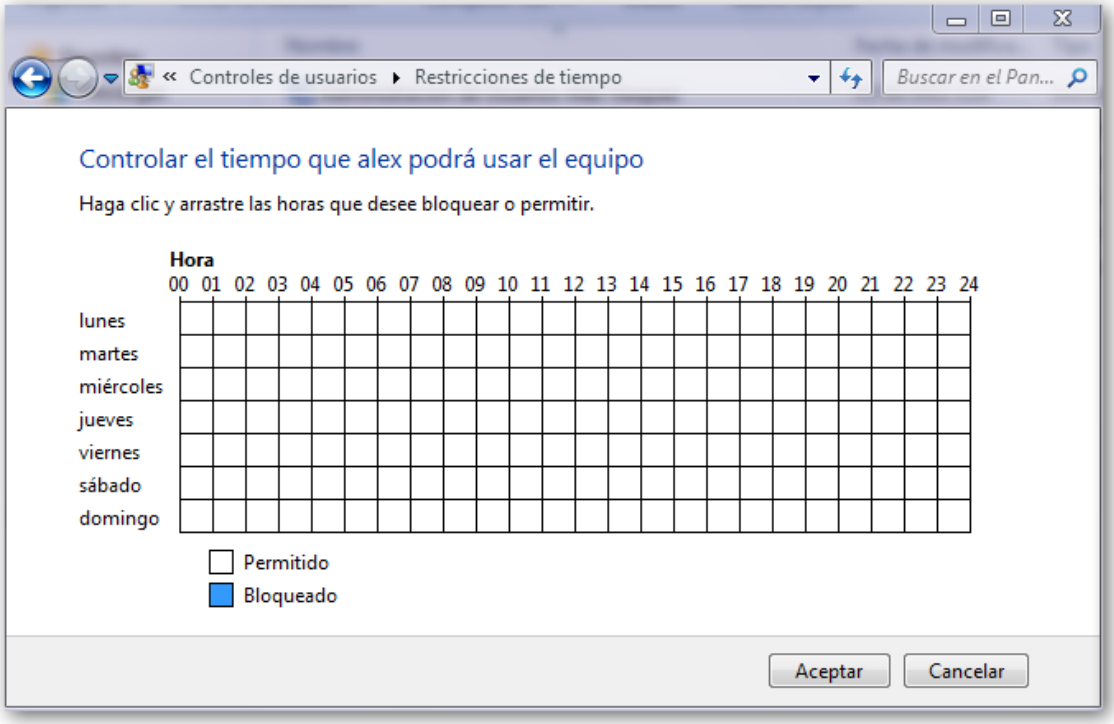
En mi caso la cuenta recién creada es **alex**. Le doy clic y me lleva a una ventana, donde me muestra.



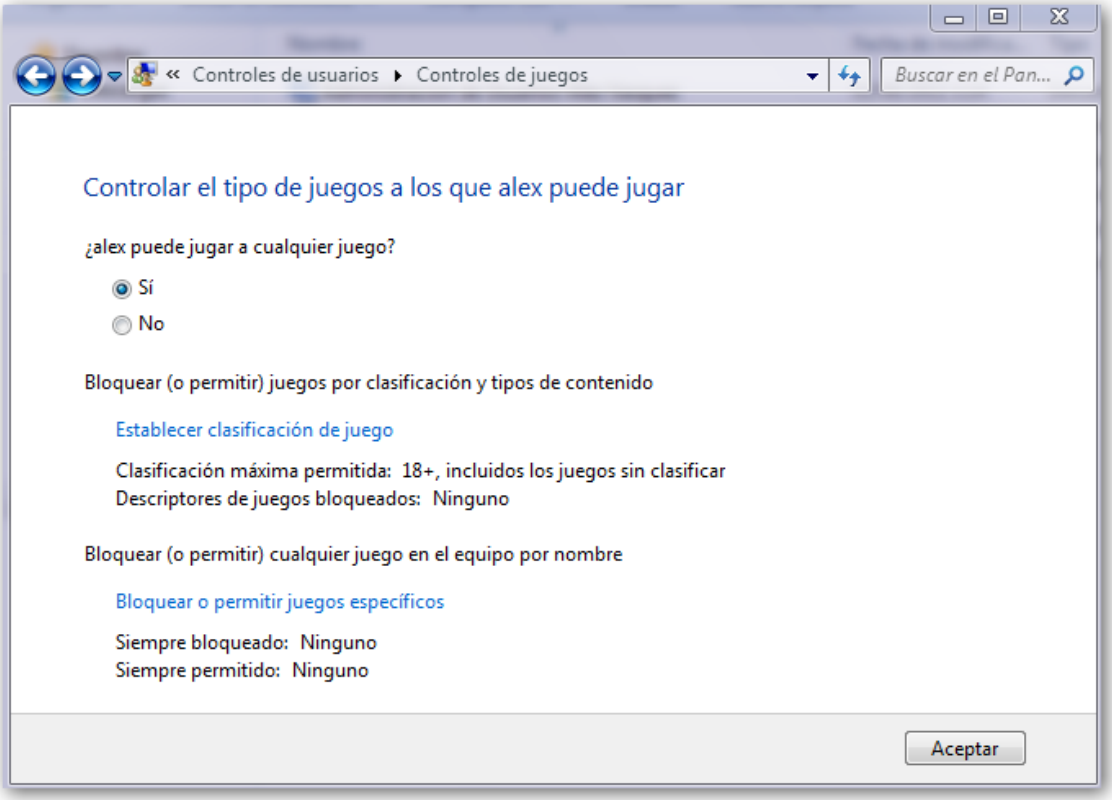
En esta ventana al checar la opción **Activado**, me permitirá configurar al usuario, mas no al SQL SERVER.

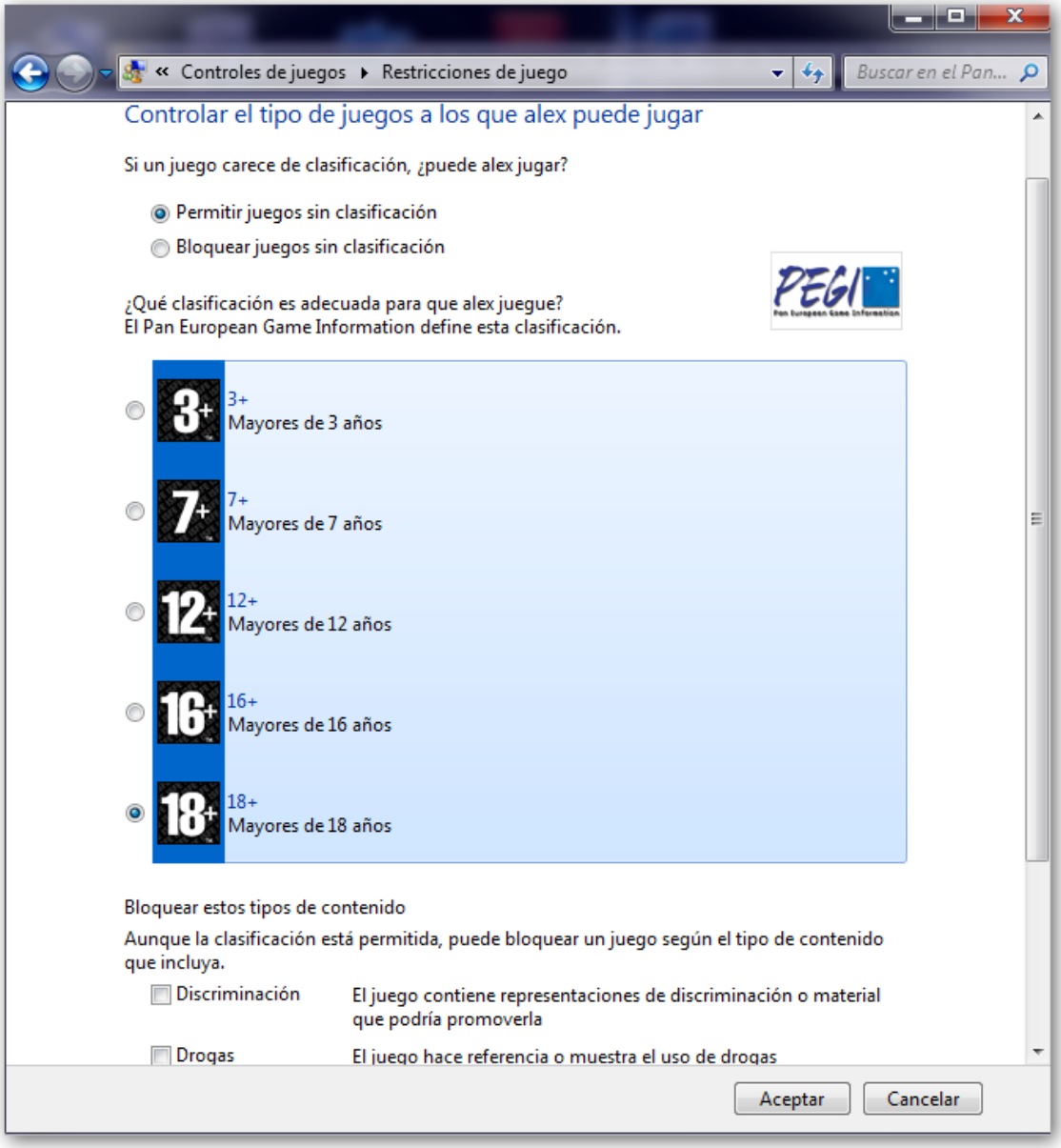


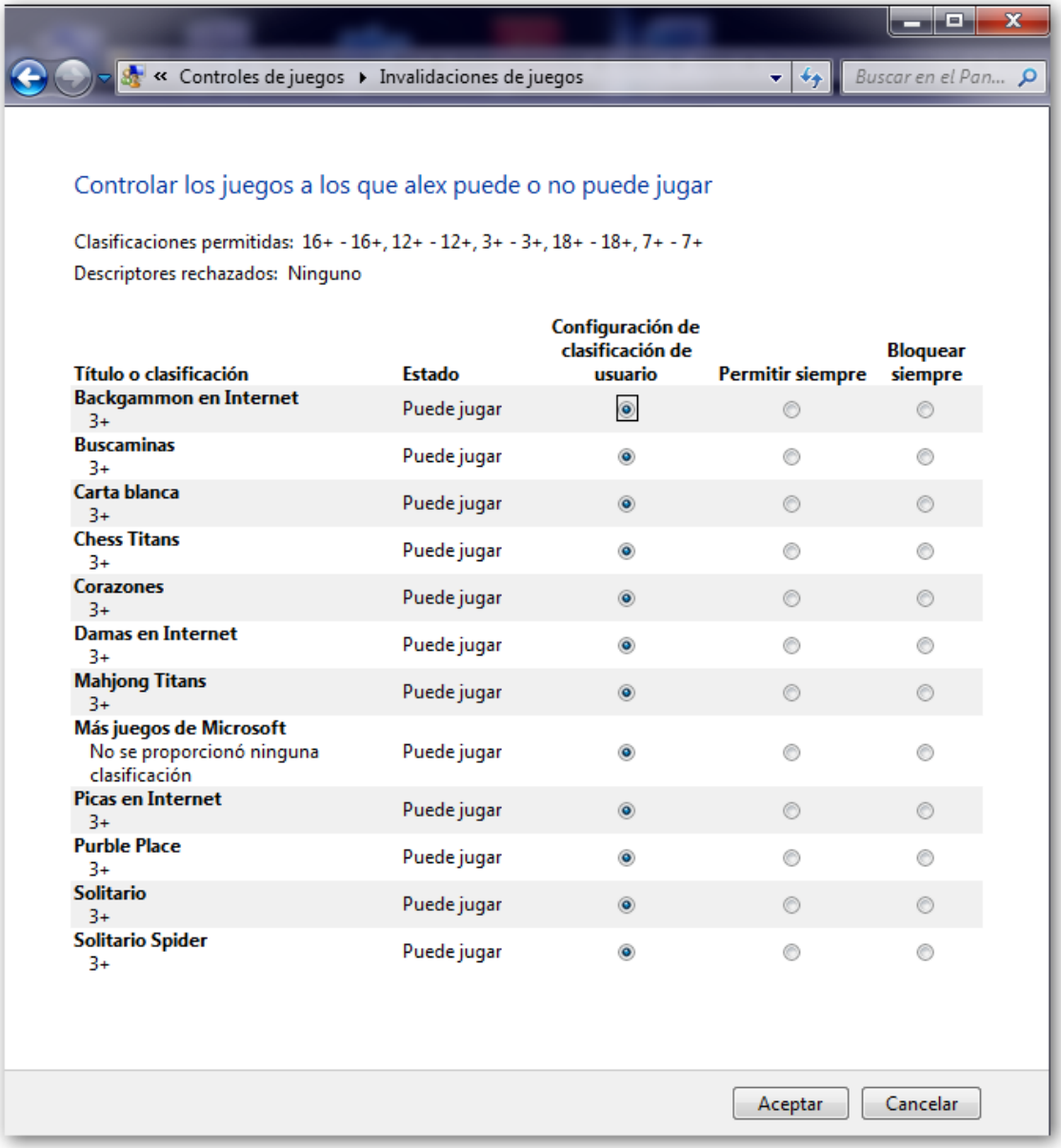
Se podría dar límites de tiempo, es decir que solo tenga acceso por horas, días.



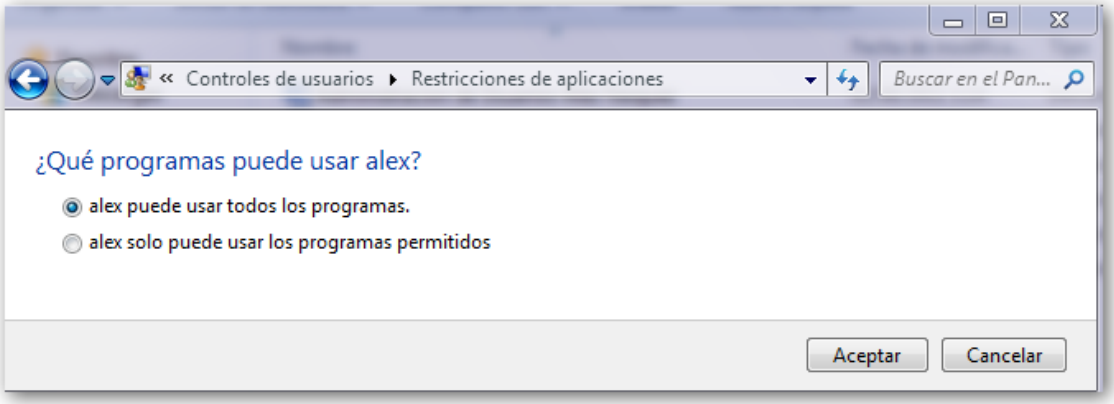
Se podría dar acceso a algunos juegos.

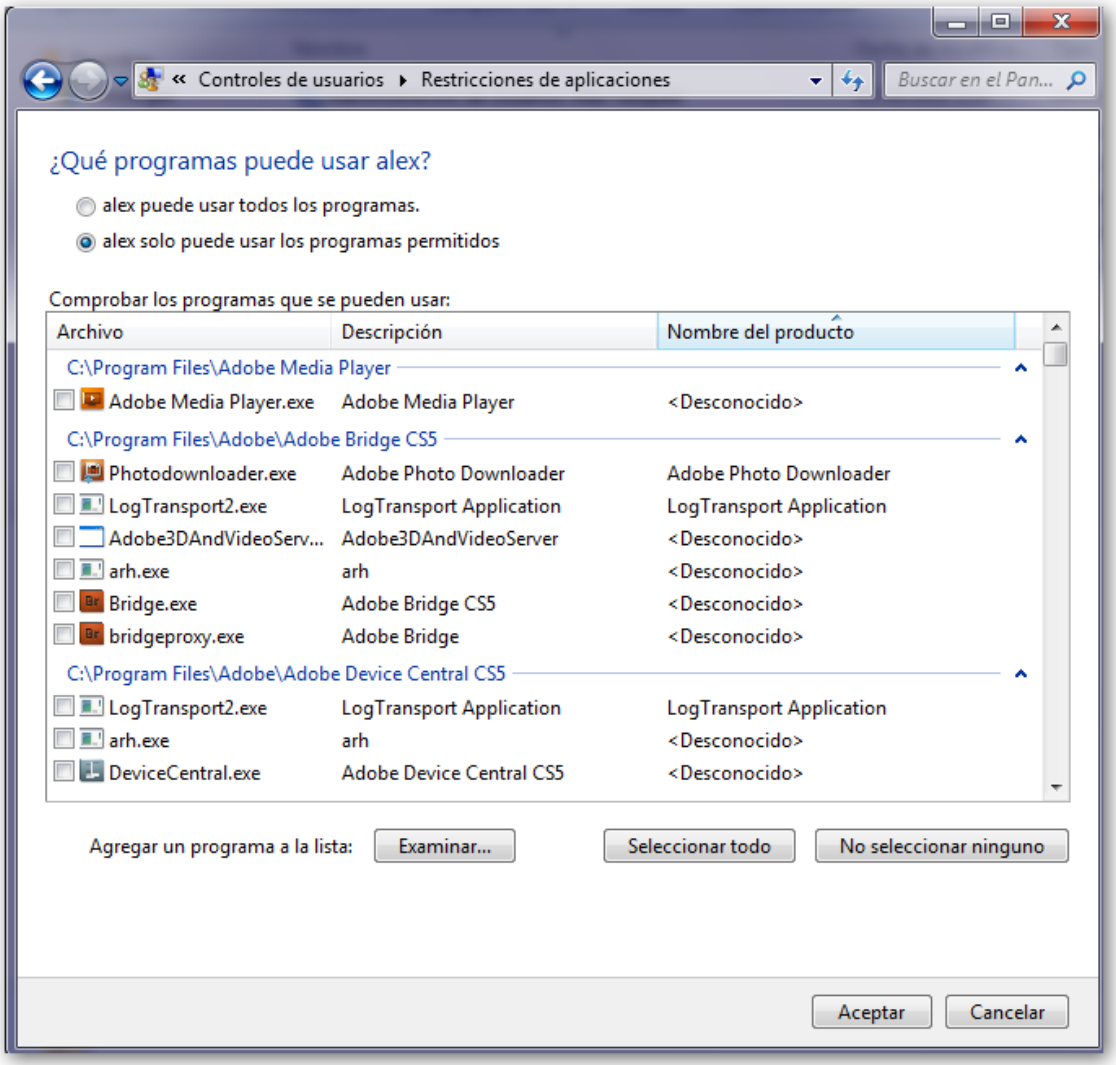

« Controles de usuarios ▸ Controles de juegos
Buscar en el Pan...
Controlar el tipo de juegos a los que alex puede jugar
¿alex puede jugar a cualquier juego?
Sí
No
Bloquear (o permitir) juegos por clasificación y tipos de contenido
Establecer clasificación de juego
Clasificación máxima permitida: 18+, incluidos los juegos sin clasificar
Descriptores de juegos bloqueados: Ninguno
Bloquear (o permitir) cualquier juego en el equipo por nombre
Bloquear o permitir juegos específicos
Siempre bloqueado: Ninguno
Siempre permitido: Ninguno
Aceptar


« Controles de juegos ▸ Restricciones de juego
Buscar en el Pan...
Controlar el tipo de juegos a los que alex puede jugar
Si un juego carece de clasificación, ¿puede alex jugar?
Permitir juegos sin clasificación
Bloquear juegos sin clasificación
¿Qué clasificación es adecuada para que alex juegue?
El Pan European Game Information define esta clasificación.
PEGI
3+
Mayores de 3 años
7+
Mayores de 7 años
12+
Mayores de 12 años
16+
Mayores de 16 años
18+
Mayores de 18 años
Bloquear estos tipos de contenido
Aunque la clasificación está permitida, puede bloquear un juego según el tipo de contenido que incluya.
Discriminación
El juego contiene representaciones de discriminación o material que podría promoverla
Drogas
El juego hace referencia o muestra el uso de drogas
Aceptar
Cancelar



Se podría seleccionar los programas que si tener acceso.





Nota: Basándonos en la realidad la similitud seria: Cuando tú vas a clases y el vigilante no te permita el ingreso, y el profesor no se entera de eso. De la misma manera es como funcionaria los permisos de usuarios. (Ítem 3)

El las restricciones son al usuario y no al SQL Server.

4. **Google Apps**

Google Apps es un conjunto de aplicaciones que incorpora un gestor de correo basado en gmail, un calendario, gdocs, sites, y la posibilidad de añadir aplicaciones de terceros.

Esto también permite elegir una garantía de seguridad y de conformidad para la mensajería electrónica existente. Para Educación, el programa es gratuito.

Google Apps ofrece herramientas eficaces para la manipulación, gestión y personalización de utilidades para dominios o nombres de Internet. Es decir, Google Apps te permite gestionar el correo electrónico de tu dominio (a través de Gmail), mensajería instantánea entre miembros de tu organización o red (Google Talk),

calendario en línea (Google Calendar), edición de Documentos también en línea (Google Docs) y creación de sitios web profesionales (Google Sites).

En el cual es a tiempo real.

La integración con Google Apps implicará que los mails no serán administrados por nosotros sino por Google, por lo que resulta fundamental decidir si desean utilizar el servicio.

Fallas

La desventaja fundamental de este excelente software de colaboración radica en su principal ventaja: que sea una aplicación basada en web, es decir, se accede a ella mediante **sesiones online** que requieren contraseña. Y el problema es que, debido a su éxito, Google hace que sea diabólicamente fácil confundir las contraseñas, las sesiones y otros datos web.

¿Por qué?

Todos tienen una cuenta Google, tu negocio, tus empleados, así que es indispensable que todos tengan identificadas perfectamente la dirección web correcta y los inicios de sesión correspondientes para evitar el desastre.

Por ejemplo, si yo usaba el buscador Internet Explorer para iniciar sesión en mi cuenta de correo personal Gmail desde la página principal de Google (http://www.google.com/) me dirigía a mi cuenta Esta dirección electrónica está protegida contra spambots. Es necesario activar JavaScript para visualizarla. Pero si quería ingresar a la cuenta que tengo en la empresa, también hospedada en Google, tenía que recordar no ir a la página principal, sino abrir (www.google.com/a). Esa pequeña **"a"** no es un error, es la letra crítica que establece la diferencia entre mi cuenta laboral y la cuenta que todo mundo usa en Google

Google usa la misma interfaz de e-mail para Gmail y para hospedar las cuentas de Google Apps. Así que es fácil confundirse si no estás atento.

La confusión empeora cuando se comparten computadoras, pues a veces Google Talk se queda con el nombre de la persona que inicia sesión por primera vez, y sigue achacándote esa identidad aunque entres a Google Apps con tu propia sesión. Para solucionarlo tuvimos que limpiar el cache.

El desorden también aparece a veces en Google Calendar, que intercambia agendas laborales y personales. En ocasiones recibo invitaciones dirigidas a Dan, un interno de la empresa, por alguna razón Google las remite a mi cuenta.

Los desarrolladores del software han reconocido que el manejo de identidades puede ser un problema, pero añaden que los beneficios pesan más: funcionar a bajo costo, acceder a tu información desde cualquier parte, y dejar que todo se almacene remotamente.

"Los usuarios tienen que sentirse cómodos usando aplicaciones basadas en la Web. Eso no vale sólo para Google, sino para todo tipo de buscadores. Cuando se usa la ***cloud computing*** se necesita cambiar ciertas cosas de la conducta laboral, pero las ventajas son enormes" apunta ***Matt Glotzbach***, director de productos en Google Apps.

Es barato y fácil de usar, pero requiere mayor gestión y mantenimiento. Justo ahora, la empresa está revisando las identidades de las cuentas empleado por empleado, lo que nos quita mucho tiempo. Google Apps es un software atractivo, pero tiende a crear confusión. Google Apps puede ser una poderosa herramienta, y es atractiva. Podría optar por la versión Premier, que viene con ayuda adicional y más funciones.

Estoy iniciando en esto así que les pido disculpas por algunos errores que se presenten en este manual.