	UNIVERSIDAD DON BOSCO **FACULTAD DE ESTUDIOS TECNOLÓGICOS** **ESCUELA DE COMPUTACIÓN**
CICLO 1-2016	**GUÍA DE LABORATORIO Nº 8** **Nombre de la práctica: Consultas a múltiples tablas. Uso de JOIN y SUBCONSULTAS.** **Lugar de ejecución:** Laboratorio de Informática **Tiempo estimado:** 2 horas y 30 minutos **Materia:** Base de datos

I. Objetivos

Que el estudiante sea capaz de:

1. Combinar datos de dos o más tablas por medio de JOINS.

2. Seleccionar información de varias tablas utilizando SUBCONSULTAS.

II. Introducción Teórica

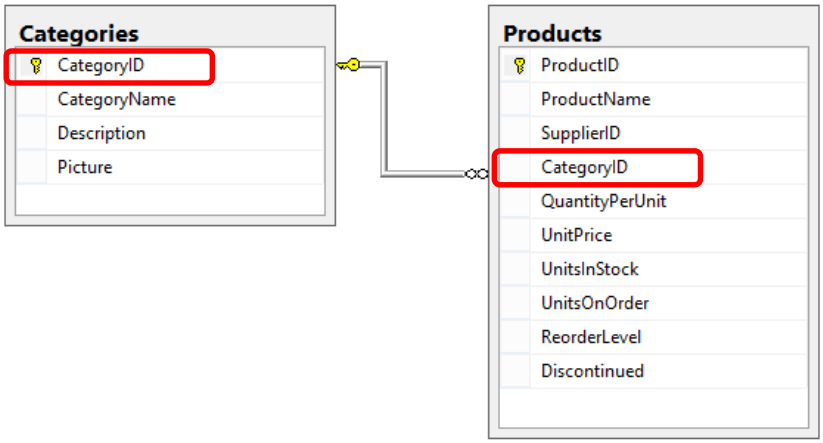
COMBINACIÓN DE TABLAS.

¿Qué se entiende por combinación de tablas?

Una combinación es una operación que permite consultar dos o más tablas para producir un conjunto de resultados que incorpore filas y columnas de cada una de las tablas en cuestión. Las tablas se combinan en función de las columnas que son comunes a ambas tablas.

El objetivo de la combinación de tablas es proporcionar al usuario datos que le permitan un fácil entendimiento de la información que requiere. Esta información, por el uso del modelo entidad – relación, se puede encontrar fragmentada en muchas tablas, y al combinarlas, se puede presentar al usuario la información pertinente de una forma más entendible.

Esta operación es conocida también como unión o vinculación de tablas.



La combinación de campos de tablas distintas sólo es posible cuando se han definido campos relacionados entre tablas. Esto es si existe un campo clave primaria en una tabla que aparece como clave foránea en la otra tabla.

La sentencia JOIN en SQL permite combinar registros de dos o más tablas en una base de datos relacional.

Sentencias JOIN

En el Lenguaje de Consultas Estructurado (SQL) hay tres tipos de JOIN: interno, externo y cruzado.

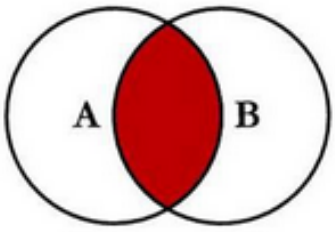
- Combinación interna INNER JOIN
- Cruzada CROSS JOIN
- Combinación externa OUTER JOIN
    - LEFT OUTER JOIN o LEFT JOIN
    - RIGHT OUTER JOIN o RIGHT JOIN
    - FULL OUTER JOIN o FULL OUTER JOIN

La palabra OUTER es opcional y no añade ninguna función

SQL JOINS

INNER JOIN

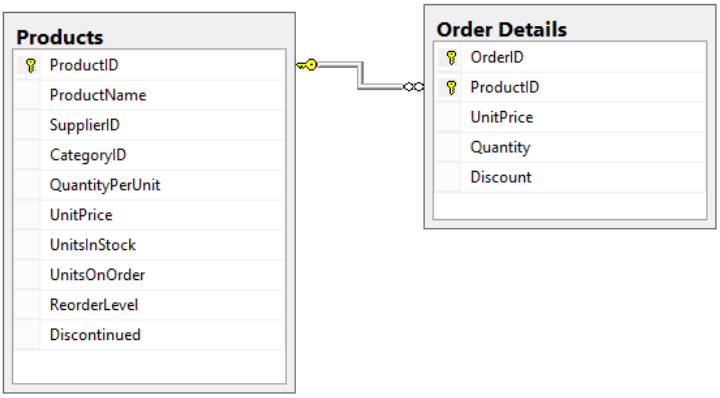
Se utiliza para mostrar los datos coincidentes entre las tablas de donde se quiere mostrar la información:




```
SELECT <lista_campos>
FROM <TablaA A>
INNER JOIN  <TablaB B>
ON A.Key=B.Key
```

Nota: **ON** se utiliza para colocar los nombres de los campos con los cuales se ha realizado la relación entre las tablas.

Ejemplo 1: Se desea conocer todos los productos que se encuentran en una orden



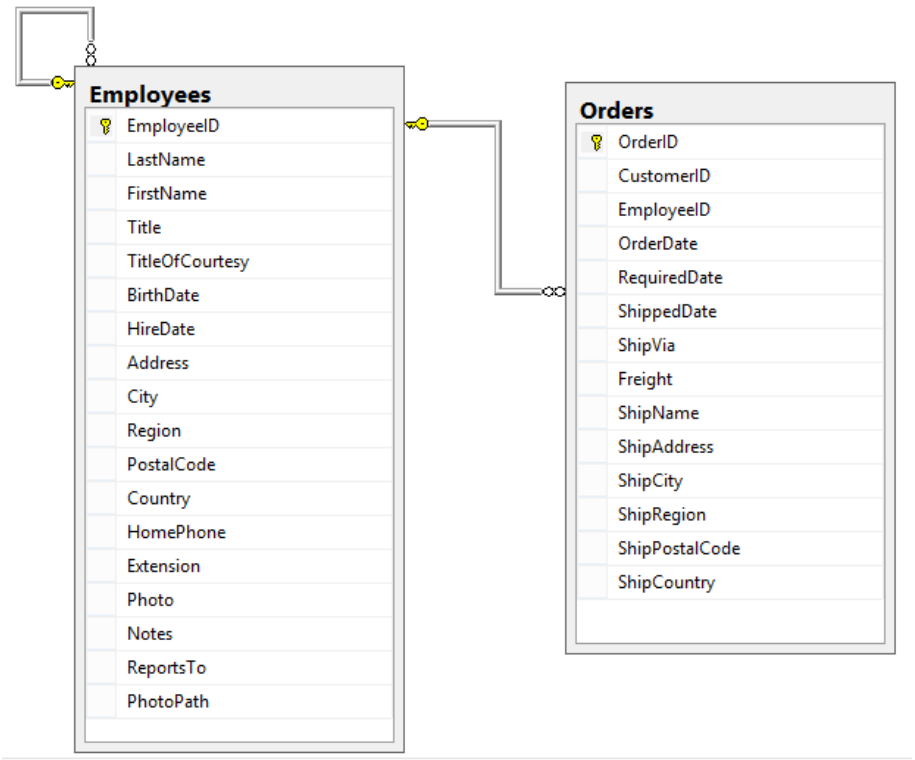
Para obtener los registros coincidentes en ambas tablas habría que realizar la siguiente consulta:

```
SELECT OrderID, P.ProductID, ProductName
FROM Products P
INNER JOIN [Order Details] OD
ON P.ProductID=OD.ProductID
```

Y se obtendría el siguiente resultado:

	OrderID	ProductID	ProductName
1	10285	1	Chai
2	10294	1	Chai
3	10317	1	Chai
4	10348	1	Chai
5	10354	1	Chai
6	10370	1	Chai
7	10406	1	Chai
8	10413	1	Chai
9	10477	1	Chai
10	10522	1	Chai

Ejemplo 2: Se desea conocer los empleados que han atendido una orden y en qué fecha lo hicieron, los registros se deben ordenar por el campo EmployeeID



La consulta SQL es:

```
SELECT LastName, Employees.EmployeeID,OrderDate FROM Orders
INNER JOIN Employees
ON Orders.EmployeeID=Employees.EmployeeID
ORDER BY Employees.EmployeeID
```

Nota: al campo EmployeeID se le coloca el nombre de la tabla de donde queremos sacar los resultados, ya que el nombre de este campo aparece tanto en la tabla Orders y Employees, y si no se utiliza así da error de nombre ambiguo.

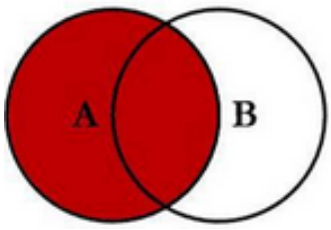
Y se obtienen los siguientes resultados:

	LastName	EmployeeID	OrderDate
1	Davolio	1	1996-07-17 00:00:00.000
2	Davolio	1	1996-08-01 00:00:00.000
3	Davolio	1	1996-08-07 00:00:00.000
4	Davolio	1	1996-08-20 00:00:00.000
5	Davolio	1	1996-08-28 00:00:00.000
6	Davolio	1	1996-08-29 00:00:00.000
7	Davolio	1	1996-09-12 00:00:00.000
8	Davolio	1	1996-09-16 00:00:00.000
9	Davolio	1	1996-09-20 00:00:00.000
10	Davolio	1	1996-09-25 00:00:00.000
11	Davolio	1	1996-09-27 00:00:00.000
12	Davolio	1	1996-10-09 00:00:00.000

Aquí se muestran las primeras 12 filas de 830 filas en total

LEFT JOIN

Muestra los registros de la tabla izquierda más los registros coincidentes con la tabla derecha



SELECT <lista_campos>
FROM <TablaA A>
LEFT JOIN <TablaB B>
ON A.Key=B.Key

Ejemplo: Se desea conocer que empleados han atendido un pedido independientemente si este lo ha realizado o no

La consulta SQL es:

```
SELECT OrderID, E.EmployeeID, Lastname
FROM Employees E
LEFT JOIN Orders O
ON E.EmployeeID=O.EmployeeID
```

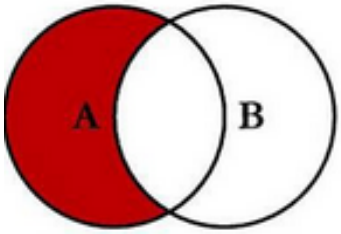
Se obtiene el siguiente resultado:

	OrderID	EmployeeID	Lastname
821	10944	6	Suyama
822	10956	6	Suyama
823	10959	6	Suyama
824	10965	6	Suyama
825	10973	6	Suyama
826	10999	6	Suyama
827	11019	6	Suyama
828	11025	6	Suyama
829	11031	6	Suyama
830	11045	6	Suyama
831	NULL	10	Urrutia

En el último registro se observa que en el campo OrderID tiene un valor NULL, lo cual indica que el empleado Urrutia no ha atendido ningún pedido

LEFT JOIN (IS NULL)

Muestra los registros de la tabla izquierda menos los registros coincidentes con la tabla derecha




```
SELECT <lista_campos>
FROM <TablaA A>
LEFT JOIN  <TablaB B>
ON A.Key=B.Key
WHERE B.Key IS NULL
```

Ejemplo: Se desea conocer los empleados que no han atendido ningún pedido

La consulta SQL es:

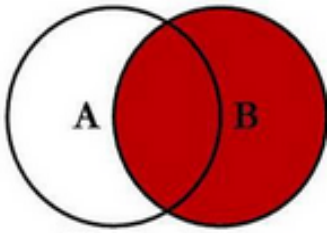
```
SELECT OrderID, E.EmployeeID, Lastname
FROM Employees E
LEFT JOIN Orders O
ON E.EmployeeID=O.EmployeeID
WHERE O.EmployeeID IS NULL
```

Se obtiene el siguiente resultado (un solo registro):

	OrderID	EmployeeID	Lastname
1	NULL	10	Urrutia

RIGHT JOIN

Muestra los registros de la tabla derecha más los registros coincidentes con la tabla izquierda




```
SELECT <lista_campos>
FROM <TablaA A>
RIGHT JOIN  <TablaB B>
ON A.Key=B.Key
```

Ejemplo: Mostrar que productos ofrece cada proveedor independientemente si este lo hace o no

La consulta SQL es:

```
SELECT ProductName, CompanyName, ContactName
FROM Products P
RIGHT JOIN Suppliers S
ON P.SupplierID=S.SupplierID
```

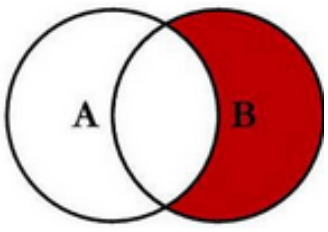
Se obtiene el siguiente resultado:

	ProductName	CompanyName	ContactName
74	Raclette Courdavault	Gai pâturage	Eliane Noz
75	Camembert Pierrot	Gai pâturage	Eliane Noz
76	Sirop d'érable	Forêts d'érables	Chantal Goulet
77	Tarte au sucre	Forêts d'érables	Chantal Goulet
78	NULL	Coca Cola	Iñaky Perez

En el último registro se verifica que el proveedor Coca Cola no ha ofrecido ningún producto

RIGHT JOIN (IS NULL)

Muestra los registros de la tabla derecha menos los registros coincidentes con la tabla izquierda




```
SELECT <lista_campos>
FROM <TablaA A>
RIGHT JOIN  <TablaB B>
ON A.Key=B.Key
WHERE A.Key IS NULL
```

Ejemplo: Mostrar que proveedor no ha ofrecido productos

La consulta SQL es:

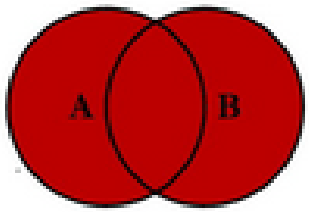
```
SELECT ProductName, CompanyName, ContactName
FROM Products P
RIGHT JOIN Suppliers S
ON P.SupplierID=S.SupplierID
WHERE P.SupplierID IS NULL
```

Se obtiene el siguiente resultado (un solo registro):

	ProductName	CompanyName	ContactName
1	NULL	Coca Cola	Iñaky Perez

FULL JOIN

Muestra los registros de la tabla izquierda y la tabla derecha más los registros coincidentes entre ambas




```
SELECT <lista_campos>
FROM <TablaA A>
FULL JOIN  <TablaB B>
ON A.Key=B.Key
```

Ejemplo: En la siguiente consulta se muestra los productos que tengan o no asignado un proveedor y los proveedores independientemente si estos han ofrecido o no un producto

La consulta SQL es:

```
SELECT ProductName, CompanyName, ContactName
FROM Products P
FULL JOIN Suppliers S
ON P.SupplierID=S.SupplierID
```

Se obtiene el siguiente resultado:

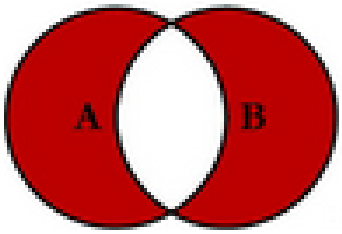
	ProductName	CompanyName	ContactName
1	Producto X	NULL	NULL
2	Chai	Exotic Liquids	Charlotte Cooper
3	Chang	Exotic Liquids	Charlotte Cooper
4	Aniseed Syrup	Exotic Liquids	Charlotte Cooper

	ProductName	CompanyName	ContactName
75	Raclette Courdavault	Gai pâturage	Eliane Noz
76	Camembert Pierrot	Gai pâturage	Eliane Noz
77	Sirop d'érable	Forêts d'érables	Chantal Goulet
78	Tarte au sucre	Forêts d'érables	Chantal Goulet
79	NULL	Coca Cola	Iñaky Perez

Observar que el primer registro indica que el Producto X no tiene proveedor asignado y en el último registro el Proveedor Coca Cola no ha ofrecido ningún producto.

FULL JOIN (IS NULL)

Muestra los registros de la tabla izquierda y la tabla derecha menos los registros coincidentes entre ambas




```
SELECT <lista_campos>
FROM <TablaA A>
FULL JOIN  <TablaB B>
ON A.Key=B.Key
WHERE A.Key IS NULL OR B.Key IS NULL
```

Ejemplo: En la siguiente consulta se muestra los productos que no tienen asignado un proveedor y los proveedores que no han ofrecido un producto

La consulta SQL es:

```
SELECT ProductName, CompanyName, ContactName
FROM Products P
FULL JOIN Suppliers S
ON P.SupplierID=S.SupplierID
WHERE P.SupplierID IS NULL OR S.SupplierID IS NULL
```

Se obtiene el siguiente resultado (dos resgistros):

	ProductName	CompanyName	ContactName
1	Producto X	NULL	NULL
2	NULL	Coca Cola	Iñaky Perez

CROSS JOIN

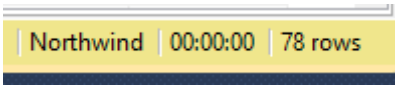
Una combinación cruzada que no tenga una cláusula WHERE genera el producto cartesiano de las tablas involucradas en la combinación.

El tamaño del conjunto de resultados de un producto cartesiano es igual al número de filas de la primera tabla multiplicado por el número de filas de la segunda tabla.

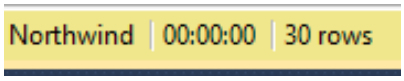
Ejemplo

Ejecutamos las siguientes consultas para conocer la cantidad de filas o registros tienen las siguientes tablas:

```
SELECT * FROM Products
```

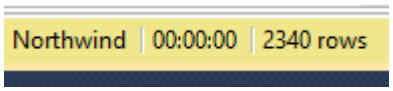



```
SELECT * FROM Suppliers
```



Ahora ejecutamos la siguiente consulta:

```
SELECT ProductName, CompanyName, ContactName
FROM Products P
CROSS JOIN Suppliers S
```



Como resultado tenemos 2340 filas o registros, ya que si multiplicamos las 78 filas de la primera tabla por las 30 filas de la segunda obtenemos ese resultado

Sin embargo, si se agrega una cláusula WHERE, la combinación cruzada se comporta como una combinación interna (INNER JOIN)

```
SELECT ProductName, CompanyName, ContactName
FROM Products P
CROSS JOIN Suppliers S
WHERE P.SupplierID=S.SupplierID
```

Obtenemos el mismo resultado al ejecutar la siguiente consulta:

```
SELECT ProductName, CompanyName, ContactName
FROM Products P
INNER JOIN Suppliers S
ON P.SupplierID=S.SupplierID
```

SUBCONSULTAS.

Una SUBCONSULTA es una consulta T-SQL normal anidada dentro de otra consulta, se crean utilizando paréntesis en una instrucción SELECT que sirve como base para cualquier parte de los datos o de la condición de otra consulta.

Las SUBCONSULTAS devuelven grupos o arreglos de valores los cuales pueden servir de parámetros a otras búsquedas. Esto permite que los valores a buscar no sean estáticos sino dinámicos, y que la consulta no tenga que ser regenerada en vista de los cambios de información que puede haber en la BD.

Normalmente las SUBCONSULTAS se utilizan para satisfacer una o un par de las siguientes necesidades:

- Desglosar una consulta en una serie de pasos lógicos.
- Proporcionar un listado que va a ser el destino de una cláusula WHERE con [IN | EXISTS |ANY | ALL].
- Proporcionar una búsqueda dirigida por cada registro individual de una consulta principal.

Una SUBCONSULTA puede devolver:

- Una sola columna o un solo valor en cualquier lugar en donde pueda utilizarse una expresión de un sólo valor y puede compararse usando los siguientes operadores: =,<,>,<=,>= ,<>,!> y !<.
- Una sola columna o muchos valores que se pueden utilizar con el operador de comparación de listas IN en la cláusula WHERE.
- Muchas filas que pueden utilizarse para comprobar la existencia, usando la palabra EXISTS en la cláusula WHERE.

Sintaxis:

SELECT <lista de seleccion>
FROM <Alguna Tabla>
WHERE <Alguna Columna> **[IN | EXISTS |ANY | ALL]** (**SELECT** <columna>
FROM <Alguna Tabla>
[**WHERE** <condición>]
)

Ejemplos

Ejemplo 1:

Mostrar los campos ProductID, ProductName y UnitPrice de todos los productos con cantidades mayores a 100

```
SELECT ProductID,ProductName,UnitPrice
FROM Products
WHERE ProductID IN (SELECT productid
                    FROM [Order Details]
                    WHERE Quantity>=100)
ORDER BY ProductName
```

Ejemplo 2:

En este ejemplo se utiliza la instrucción EXISTS y obtenemos los mismos resultados de la consulta anterior:

a. Seleccionando un campo en específico en la subconsulta

```
SELECT ProductID,ProductName,UnitPrice
FROM Products
WHERE EXISTS  (SELECT ProductID FROM [Order Details]
               WHERE Quantity>=100
               AND Products.ProductID=[Order Details].ProductID)
ORDER BY ProductID
```

b. Seleccionando todos los campos en la subconsulta:

```
SELECT ProductID,ProductName,UnitPrice
FROM Products
WHERE EXISTS (SELECT * FROM [Order Details]
              WHERE Quantity>=100
              AND Products.ProductID=[Order Details].ProductID)
ORDER BY ProductID
```

SUBCONSULTAs en las instrucciones UPDATE y DELETE

UPDATE

```
-- Actualiza la tabla Products, aumentando en un 20% el precio unitario (UnitPrice)
-- del producto, donde el nombre de la categoria comience con la letra B

UPDATE Products SET UnitPrice=UnitPrice*1.2
WHERE CategoryID IN (SELECT CategoryID FROM Categories
                     WHERE CategoryName LIKE 'B%')
```

Antes de ejecutar la consulta, comprobamos que código tiene la categoría que comienza con la letra B:

	CategoryID	CategoryName	Description	Picture
1	1	Beverages	Soft drinks, coffees, teas, beers, and ales	0x151C2F0002

Los productos que tienen ese código:

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsIr
1	1	Chai	1	1	10 boxes x 20 bags	18.00	39
2	2	Chang	1	1	24 - 12 oz bottles	19.00	17
3	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.00	13

Al ejecutar la consulta obtenemos el siguiente resultado:

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsIr
1	1	Chai	1	1	10 boxes x 20 bags	21.60	39
2	2	Chang	1	1	24 - 12 oz bottles	22.80	17

DELETE

En la siguiente consulta se elimina el o los clientes cuya fecha de pedido sea mayor a 2014-01-01

Al inicio tenemos los siguientes datos:

	CustomerID	CompanyName	ContactName	ContactTitle
78	THECR	The Cracker Box	Liu Wong	Marketing Assistant
79	TIPRE	Tipicos Regional	Alex Pietro	Owner
80	TOMSP	Toms Spezialitäten	Karin Josephs	Marketing Manager

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate
832	12078	TIPRE	4	2014-04-20 00:00:00.000	NULL

Ejecutamos la siguiente consulta:

```
DELETE Customers WHERE
CustomerID IN (SELECT CustomerID FROM Orders
               WHERE OrderDate > '2014-01-01')
```

Obtenemos los siguientes resultados:

	CustomerID	CompanyName	ContactName	ContactTitle
78	THECR	The Cracker Box	Liu Wong	Marketing Assistant
79	TOMSP	Toms Spezialitäten	Karin Josephs	Marketing Manager
80	TORTU	Tortuga Restaurante	Miguel Angel Pa...	Owner

Ya no se encuentra el cliente con ID TIPRE

III. Requerimientos

- Máquina con SQL Server 2012
- Guía Número 8 de base de datos

IV. Procedimiento

Parte 1: Iniciando sesión desde SQL Server Managment Studio

1. Hacer clic en el botón **Inicio**
2. Hacer clic en la opción **Todos los programas** y hacer clic en **Microsoft SQL Server 2012**

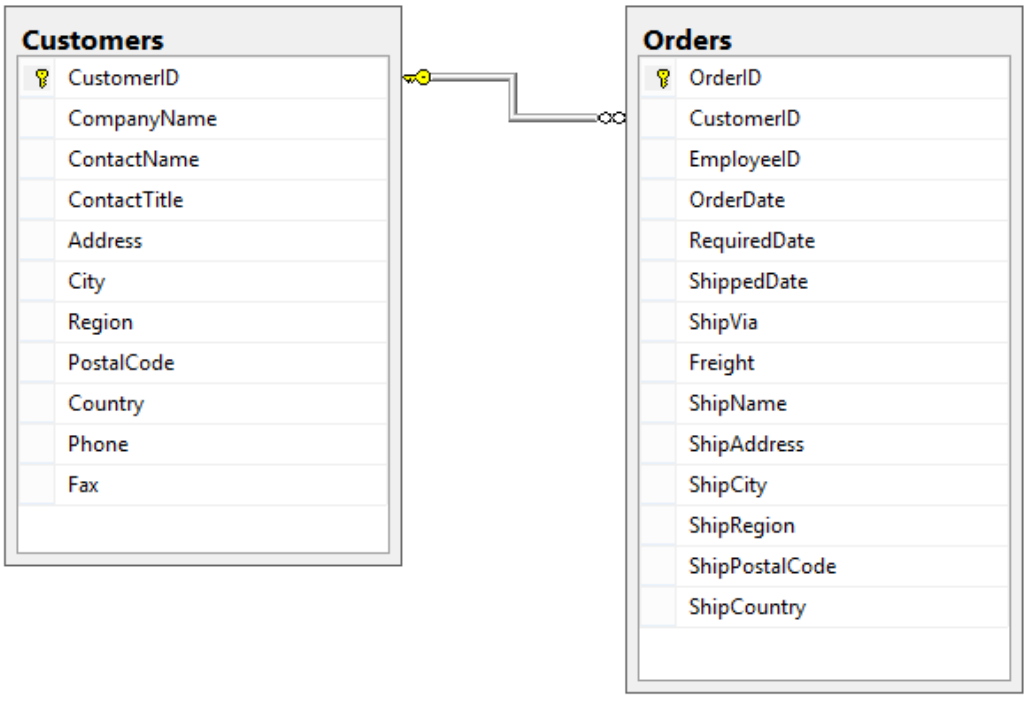
Para conectarse con el servidor de base de datos elija los siguientes parámetros de autenticación:

- **Tipo de servidor:** Database Engine
- **Nombre del servidor:** Colocar el nombre del servidor local, por ejemplo PC**NumMaquina-SALA2**
  Nota: NumMaquina es el número de la maquina local
- **Autenticación**: SQL Server Authentication
- **Login:** sa
- **Password:** 123456

3. Luego, presione Ctrl+N o de clic en **"New Query"** en la barra de trabajo estándar.

Parte 2. Combinación de Tablas

1. Se muestra la relación que existe entre las tablas Customers y Orders



2. Agregar los siguientes datos a cada una de las tablas:

Tabla: **Customers**

	Campos	
	CustomerID	**CompanyName**
1	TIPLE	Típicos Regionales
2	FLOSU	Flores del Sur

Tabla: **Orders**

	Campo
	CustomerID
1	NULL

Ejercicio 1. Uso del INNER JOIN

2. Digitar la siguiente consulta:

```
SELECT OrderID, Customers.CustomerID
FROM Customers
INNER JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
```

3. Ejecutar la consulta para obtener los resultados
4. Construir de nuevo la consulta implementando ALIAS para los nombres de las tablas
5. Ejecutar la consulta
6. Guardar el Script con el nombre: **Guia8_Procedimiento_SuCarnet.sql**

Ejercicio 2. Uso del RIGHT JOIN

1. Digitar la siguiente consulta:

```
SELECT OrderID, Customers.CustomerID
FROM Customers
RIGHT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
```

2. Ejecutar la consulta para obtener los resultados
3. Construir de nuevo la consulta implementando ALIAS para los nombres de las tablas
4. Ejecutar la consulta
5. Guardar los cambios

Ejercicio 3. Uso del RIGHT JOIN (IS NULL)

1. Digitar la siguiente consulta:

```
SELECT OrderID, Customers.CustomerID
FROM Customers
RIGHT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
WHERE Customers.CustomerID IS NULL
```

2. Ejecutar la consulta para obtener los resultados
3. Construir de nuevo la consulta implementando ALIAS para los nombres de las tablas
4. Ejecutar la consulta
5. Guardar los cambios

Ejercicio 4. Uso del LEFT JOIN

1. Digitar la siguiente consulta:

```
SELECT OrderID, Customers.CustomerID
FROM Customers
LEFT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
```

2. Ejecutar la consulta para obtener los resultados
3. Construir de nuevo la consulta implementando ALIAS para los nombres de las tablas
4. Ejecutar la consulta
5. Guardar los cambios

Ejercicio 5. Uso del LEFT JOIN (IS NULL)

1. Digitar la siguiente consulta:

```
SELECT OrderID, Customers.CustomerID
FROM Customers
LEFT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
WHERE Orders.CustomerID IS NULL
```

2. Ejecutar la consulta para obtener los resultados
3. Construir de nuevo la consulta implementando ALIAS para los nombres de las tablas
4. Ejecutar la consulta
5. Guardar los cambios

Ejercicio 6. Uso del FULL JOIN

1. Digitar la siguiente consulta:

```
SELECT OrderID, Customers.CustomerID
FROM Customers
FULL JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
```

2. Ejecutar la consulta para obtener los resultados
3. Construir de nuevo la consulta implementando ALIAS para los nombres de las tablas
4. Ejecutar la consulta
5. Guardar los cambios

Ejercicio 7. Uso del FULL JOIN (IS NULL)

1. Digitar la siguiente consulta:

```
SELECT OrderID, Customers.CustomerID
FROM Customers
FULL JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
WHERE Customers.CustomerID IS NULL OR Orders.CustomerID IS NULL
```

2. Ejecutar la consulta para obtener los resultados
3. Construir de nuevo la consulta implementando ALIAS para los nombres de las tablas
4. Ejecutar la consulta
5. Guardar los cambios

Ejercicio 8. Uso del CROSS JOIN

1. Digitar la siguiente consulta:

```
SELECT OrderID, Customers.CustomerID
FROM Customers
CROSS JOIN Orders
```

2. Ejecutar la consulta para obtener los resultados
3. Construir de nuevo la consulta implementando la cláusula WHERE
4. Ejecutar la consulta
5. Guardar los cambios

Ejercicio 9. Uso de SUBCONSULTAS

SUBCONSULTAS implementadas en la instrucción WHERE

1. Obtener los datos de los clientes que han realizado pedidos mayor o igual a la fecha 1 de Enero de 1998

```
SELECT CustomerID, CompanyName, ContactName, Country
FROM Customers
WHERE CustomerID IN (SELECT CustomerID FROM Orders
                     WHERE OrderDate>='1998-01-01')
```

   Ejecute la consulta

   El resultado de una subconsulta especificada con IN (o con NOT IN) es una lista de cero o más valores que se compara con el campo especificado en la cláusula WHERE

2. Construir la consulta anterior, cambiando la instrucción IN por NOT IN, ejecute la consulta y verifique los resultados.
3. Con el siguiente ejemplo se obtiene los datos del cliente que posee el código del Pedido 10248

```
SELECT CustomerID, CompanyName, ContactName, Country
FROM Customers
WHERE CustomerID = (SELECT CustomerID FROM Orders
                    WHERE OrderID='10248')
```

   Ejecute la consulta

Las subconsultas se pueden presentar con uno de los operadores de comparación (=, < >, >, > =, <, < =).

4. Cuando una subconsulta se especifica con la palabra clave EXISTS, funciona como una prueba de existencia

5. En el siguiente ejemplo se evalúa si existen clientes que han realizado pedidos mayor o igual a la fecha 1 de Enero de 1998 y se muestra el resultado

```
SELECT CustomerID, CompanyName, ContactName, Country
FROM Customers C
WHERE EXISTS (SELECT * FROM Orders O
              WHERE C.CustomerID=O.CustomerID
              AND OrderDate>='1998-01-01')
```

Ejecute la consulta

Verifique que el resultado de la consulta es igual al resultado de la consulta del punto 1

6. Copie de nuevo la consulta pero hoy en lugar de utilizar la instrucción EXISTS utilice la instrucción NOT EXISTS, ejecute la consulta y analice los resultados.

SUBCONSULTAS implementadas en la cláusula SELECT

7. Mostrar el máximo Precio unitario (UnitPrice) por cada pedido

```
SELECT O.OrderID, OrderDate,(SELECT MAX(UnitPrice)
 FROM [Order Details] OD
 WHERE OD.OrderID=O.OrderID) AS MaxPrecioUnitario
 FROM Orders AS O
```

8. Ejecute la consulta

9. Guardar los cambios

Guardar el script con el nombre: **Guia8_Procedimiento.sql**

V. Ejercicio Complementario

Crear los siguientes ejercicios:

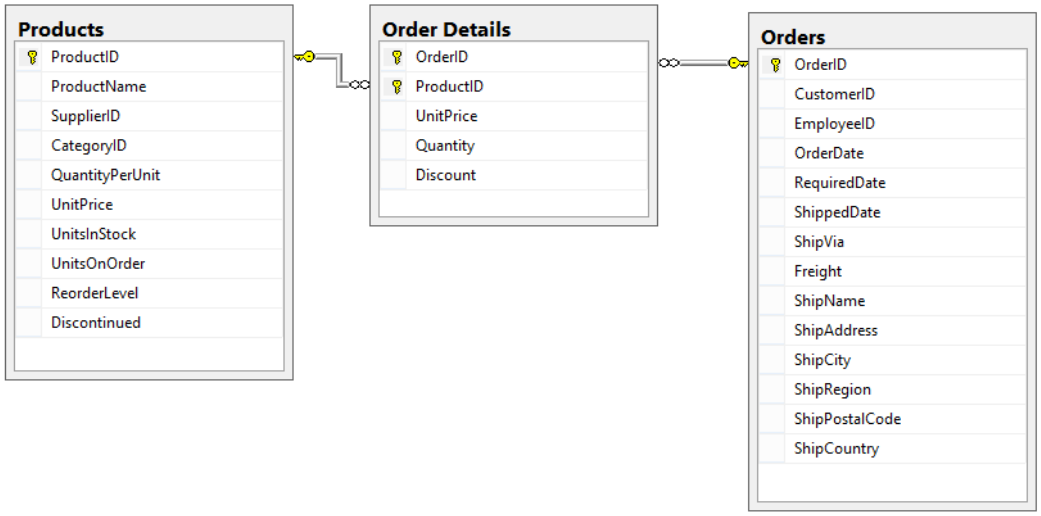
Usando la base de datos Northwind, crear las siguientes consultas:

Ejercicio 1.

Haciendo uso de INNER JOIN mostrar los campos OrderDate y ProductID de las tablas Orders y Order Details donde el dato almacenado en el campo OrderDate sea igual 7 de Julio de 1996

Ejercicio 2.

Se desea mostrar cuantas cantidades de cada producto se han vendido y la fecha de la venta de cada uno de ellos, se debe tomar en cuenta el siguiente diagrama relacional:



Campos a mostrar: ProductID, ProductName, Quantity y OrderDate

Ejercicio 3.

Haciendo uso de SUBCONSULTAS mostrar los campos OrderID, ProductID y Quantity de la tabla Orders Details donde la fecha de pedido (OrderDate de la tabla Orders) sea la más antigua

Ejercicio 4:

Ingrese un nuevo registro a la tabla Employees

Dos nuevos registros a la tabla Orders, en el primer registro en el campo EmployeeID debe agregar el código del nuevo empleado y en el segundo registro debe ingresar para ese campo un valor NULL

Crear consultas que implemente el uso de FULL, LEFT Y RIGHT JOIN

Campos a mostrar: LastName, EmployeeID de la tabla Employees

OrderDate de la tabla Orders

Al final del ejercicio y después de hacer las pruebas necesarias, **elimine de las tablas los datos que agrego.**

Guardar el archivo como: **Guia8_EjercicioComplementario.sql**

VI. Investigación complementaria

TRABAJO INDIVIDUAL

Ejercicio 1.

1. Crear la siguiente Base de datos:

Recuerde que al crear las tablas debe implementar:

- Restricciones UNIQUE, CHECK y DEFAULT en los campos que cree necesarios
- Las instrucciones ON DELETE CASCADE y ON UPDATE CASCADE en la creación de las relaciones entre las tablas

2. Agregue los siguientes datos:

Tabla: BITACORA

	CORRELATIVO	ID	COD_MAQUINA	TIEMPOUSO	LUGAR
1	1	12345678-9	M00001	250	SANTIAGO NONUALCO
2	2	01234567-8	M00002	300	SANTIAGO NONUALCO
3	3	90123456-7	M00003	500	ALEGRIA, USULUTAN
4	4	89012345-6	M00004	300	ALEGRIA, USULUTAN
5	5	90123456-7	M00005	250	SANTIAGO NONUALCO
6	6	01234567-8	M00002	125	SANTIAGO NONUALCO
7	7	12345678-9	M00003	375	ALEGRIA, USULUTAN
8	8	12345678-9	M00004	200	ALEGRIA, USULUTAN

Tabla: MAQUINA

	COD_MAQUINA	DESCRIPCION	MARCA	MODELO	FECHAINGRESO
1	M00001	TALADORA DE ELEMENTOS VARIOS	CATERPILLAR	EVO2000	2006-01-31
2	M00002	APLANADORA DE SUELOS Y OTROS	CATERPILLAR	FLU5000	2006-01-31
3	M00003	PULVERIZADORA DE ELEMENTOS	CATERPILLAR	ASD2001	2006-01-31
4	M00004	CONCRETERA	MG	EDS	2006-05-31
5	M00005	MAQUINA ESPECIAL PARA PROYECTO 10	MG	SFD	2006-05-31
6	M00006	MAQUINA ESPECIAL PARA PROYECTO 30	MG	SFD	2010-12-01

Tabla: EMPLEADO

	ID	NOMBRES	APELLIDOS	EDAD	FECHAINICIO
1	01234567-8	CARLOS FIDEL	ARGUETA MIRANDA	45	2006-08-21
2	12345678-9	JUAN FRANCISCO	VILLALTA ALVARADO	32	2010-02-27
3	78901234-5	RAUL ALEJANDRO	PONCIO VALLADARES	32	2010-02-27
4	89012345-6	MIGUEL EDUARDO	MORALES CLAROS	26	2010-08-21
5	90123456-7	FABRICIO DAVID	ALAS FLORES	30	2008-12-01

3. Crear las siguientes consultas:

    a. Mostrar los empleados que hayan o estén haciendo uso de una maquina
        i. Campos a mostrar: Nombres del empleado, Marca, Modelo y Descripción de la maquina

    b. Mostrar los empleados que todavía no tienen asignada una maquina
        i. Campos a mostrar: Nombres y Apellidos del empleado y Código de la maquina

    c. Mostrar las maquinas que no están asignadas a un proyecto
        i. Campos a mostrar: Nombres y Apellidos del empleado y Descripción de la maquina

Ejercicio 2.

Tomando como ejemplo la base de datos **Control_de_libros** de la Guía 7 (Investigación Complementaria), crear las siguientes consultas:

1. Mostrar el primer nombre, primer apellido de los autores junto con el título de libro que estos han escrito

2. Mostrar el nombre de la editorial y el título del libro

3. Mostrar los títulos de los libros y el nombre de la editorial, donde esta sea del país de Inglaterra

4. Mostrar los nombres de los autores y el título del libro donde el año de edición sea el más actual

5. Mostrar los nombres de los autores y el título del libro donde el año de edición sea el menos actual

6. Agregue los datos necesarios a las tablas, para luego implementar las instrucciones LEFT JOIN, RIGHT JOIN Y FULL JOIN, como por ejemplo autores que no han escrito un libro todavía etc.

VII. Fuente de consulta

1. Microsoft SQL Server 2008: Guía del Administrador
   Madrid, España: ANAYA, 2009
   Autor: William Stanek
   Biblioteca UDB – Clasificación: 005.361 S784 2009

2. Usar combinaciones en SQL Server (Versión 2008)
   https://technet.microsoft.com/es-es/library/ms191472(v=sql.105).aspx

3. Tipos de Subconsultas
   https://technet.microsoft.com/es-es/library/ms175838(v=sql.105).aspx