INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO

TECNOLOGÍA, INNOVACIÓN CULTURA Y VIDA.

HERRAMIENTAS DE SOFTWARE PARA GESTIONAR BASES DE DATOS SQL SERVER

Documento compilado por **Oralia Cortés Grajales**

Actualizado por **Irma Lucía Franco Sepúlveda**

PREÁMBULO

Existen muchas herramientas para gestionar las bases de datos, cada una tiene sus propias características, es en la etapa del análisis de la situación a resolver con bases de datos donde se debe escoger esta herramienta.

La siguiente imagen, representa la comunicación de los usuarios con las bases de datos a través de los Sistemas gestores de bases de datos, los cuales pueden ser entre otros: Access, MySQL, Oracle o SQL Server

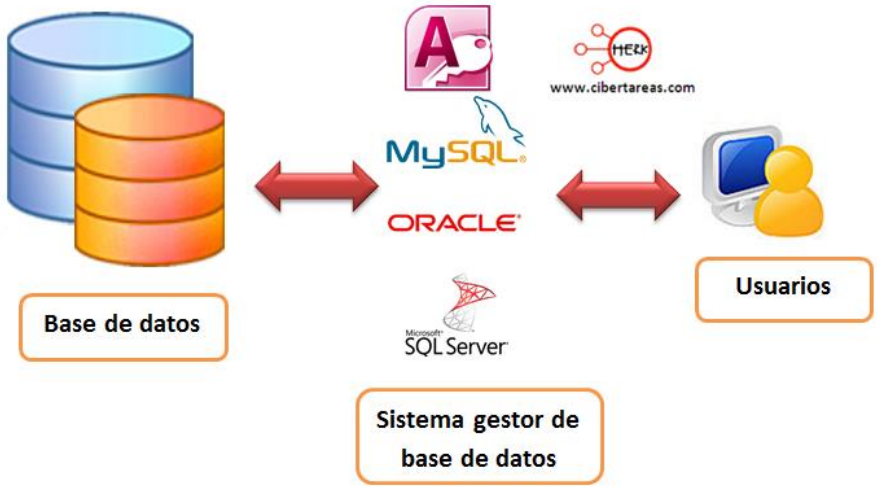


Imagen recuperada de: http://sae-igeh2no2.blogspot.com.co/

Para tener en cuenta

Para trabajar este módulo se toma como referencia SQL Server en versión 2008 o superior.

SQL PARA LA MANIPULACIÓN DE BASES DE DATOS

El lenguaje de consulta estructurado SQL (Strucured Query Language), está compuesto por comandos, cláusulas, operadores y funciones, estos combinados en las instrucciones sirven para crear, actualizar y manipular las bases de datos.

Existen dos tipos de comandos SQL:

1. DDL (Data Definition Language): sirven para crear y definir nuevas bases de daos, campos e índices, estos son: CREATE, DROP, ALTER. Realiza operaciones con tablas, índices y vistas.

2. DML (Data Manipulation Language): sirven para generar consultas, ordenar, filtrar y extraer datos de la base de datos, estos son: INSERT, SELECT, UPDATE, DELETE. Realiza operaciones con registros.

DDL: Lenguaje de definición de datos.

Comando	Descripción
CREATE	Utilizado para **crear** objetos como nuevas bases de datos, tablas, campos e índices
ALTER	Utilizado para **modificar** objetos como las tablas agregando campos o cambiando la definición de los campos.
DROP	Empleado para **eliminar** objetos como tablas e índices, vistas

DML: Lenguaje de manipulación de datos.

Comando	Descripción
INSERT	Se utiliza para **ingresar** los registros en las tablas
SELECT	Utilizado para **consulta**r datos almacenados por una petición del usuario.
UPDATE	Se emplea para **cambiar o actualizar** valores de los campos de la base de datos
DELETE	Empleado para **elimina** registros de las tablas

La siguiente tabla relaciona comandos, clausulas y operadores SQL

Concepto	Definición	Ejemplo
Lenguaje de Manipulación de Datos	Denominado por sus siglas como: DML (Data Manipulation Language), permite a los usuarios finales realizar operaciones de manipulación sobre el contenido de la base de datos.	SELECT NOMBRE,valor_hora + valor_hora * 0.3 AS PORCENTAJE FROM TBLtrabajador
INSERT	Para Ingresar los registros en las tablas.	insert into estudiante values (2008131001,'oscar',2895463,'cll 10')
SELECT	Para consultar datos almacenados por una petición del usuario.	Select * from tbloficio
UPDATE:	Para cambiar o actualizar valores de los campos de la base de datos.	UPDATE tblTrabajador SET nombre = 'Pedro' WHERE nombre = 'oscar'
DELETE	Elimina registros de las tablas.	Delete from tblAsignar where Num_Dias <10

Cláusulas del DML

Cláusula	Descripción
SELECT	Utilizada para especificar los campos y datos que se van a mostrar
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico

Operadores lógicos

Operador	Uso
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdadero si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

Operadores relacionales o de comparación

Operador	Uso
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor o Igual que
>=	Mayor o Igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo
In	Utilizado para especificar registros de una base de datos

Tipos de consultas

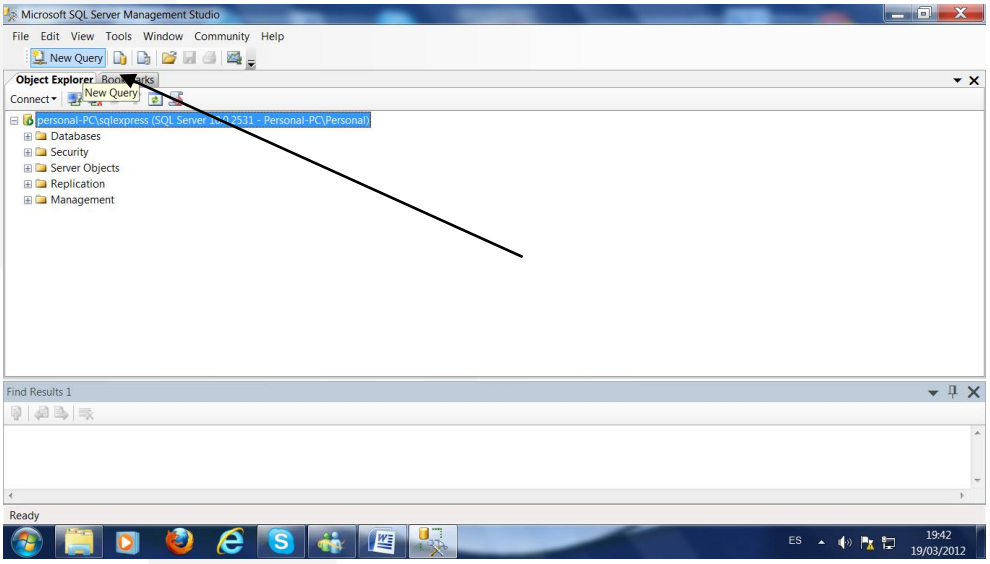
Tipos	**Clausulas**
Selección	FROM
	TOP
	WHERE
	junta interna

	junta externa
	group by
	having,
	order by
Subconsultas	Anidamiento a varios niveles
	diferencia
	comparación con un valor
	comparaciones de varios valores.
Parámetros	Se usan para intercambiar datos entre las funciones y los procedimientos almacenados y la aplicación o la herramienta que llamó a la función o al procedimiento almacenados
Referencias cruzadas	Permite visualizar los datos en filas y en columnas

EJERCICIO 1.

Crear una base de datos llamada "Constructora" **(El script de esta base de datos le fue suministrado por su tutor a través de la plataforma LMS)**

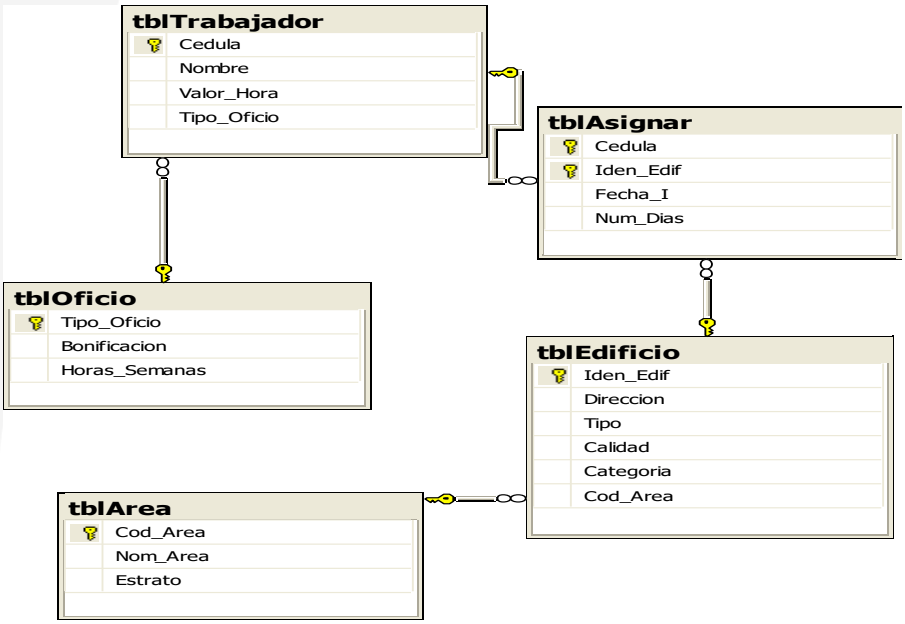
Estando en el ambiente de trabajo SQL server, haga doble clic en "New Query", visualizará un editor donde puede escribir los siguientes comandos



```
/* Crear la base de datos*/
/**********ejecuta esta línea ************/

create database DB_Constructora
/* activa la base de datos construcción*/
/********solo ejecuta esta línea************/

use  DB_Constructora
/* Crear tabla oficio mirar el comando abajo*/

create table tblOficio(Tipo_Oficio varchar(30),
Bonificacion int,
check(Bonificacion >= 30000 and Bonificacion <= 80000),
Horas_Semana int,
check(Horas_Semana >= 30 and Horas_Semana <= 60),
Primary key (Tipo_Oficio))
/**Selecciona todo el bloque de la tabla y ejecuta***/

/*** nota si la tabla le queda mal construida con este comando drop table nom_tabla
la puede borrar luego arregla el codigo y la vuelve a ejecutar****/

/*Crear tabla Trabajador*/
create table tblTrabajador(Cedula int,
Nombre varchar(50) not null,
Valor_Hora int,
/*check es el comando que se utiliza para hacer las validaciones*/
check(Valor_Hora >= 7000 and Valor_Hora <= 20000),
Tipo_Oficio varchar(30),
primary key (Cedula),
foreign key (Tipo_Oficio)references
tblOficio(Tipo_Oficio))
```

```
/*Crear tabla Area*/
create table tblArea(Cod_Area varchar(10),
Nom_Area Varchar(30),
Estrato int,
check(Estrato >= 1 and Estrato <= 7),
primary key (Cod_Area))

/*Crear tabla edificio*/
create table tblEdificio(Iden_Edif int,
Direccion varchar(30),
Tipo varchar (30),
Calidad int,
check (Calidad >= 1 and Calidad <= 5),
Categoria int,
check(Categoria >= 1 and Calidad <= 5),
Cod_Area varchar (10)
primary key (Iden_Edif),
foreign key(Cod_Area)references tblArea(Cod_Area))

/*Crear tabla Asignar*/
create table tblAsignar(Cedula int,
Iden_Edif int,
Fecha_I datetime,
Num_Dias int,
check (Num_Dias > 0),
Primary key(Cedula,Iden_edif),
foreign key(Cedula) references tbltrabajador(cedula),
foreign key(Iden_edif) references tblEdificio(Iden_Edif))
```

Este es el digrama correspondiente a la base de datos DB_Constructora que acabamos de crear.



EJERCICIO 2.

Realiza en el editor SQL server estos ejemplos de consultas de selección creadas en la base de datos **"Constructora"** **(El script de esta base de datos le fue suministrado por su tutor a través de la plataforma LMS)**

Abre el editor del sql server y activa la base de datos **DB_Constructora**

Use **DB_Constructora**

/Para ver los datos de oficio seleccionamos y ejecutamos*******/**

```
Select * from tbloficio
```

Estas consultas se deben elaborar en la base de datos "**Constructora**"

Tabla con los tipos de consulta con su definición y ejemplo.

Concepto	Definición	Ejemplo
Consultas de selección	Permiten recuperar información de la base de datos. Se muestra cómo elegir las columnas de las tablas a recuperar, y cómo aplicar distintos tipos de cláusulas como WHERE, JOIN, GROUP BY, y TOP en la consulta.	**/*Hallar el total de días asignado a cada trabajador si este es mayor que 20*/** SELECT cedula, SUM(num_dias) AS total FROM TBLasignar GROUP BY cedula HAVING (SUM(num_dias) > 20)
Selección de columnas y cláusula	La cláusula select y from siempre están presentes en las consultas, siempre se muestra de una o unas tablas. El la cláusula select se incluyen operaciones y operadores como:	**/*Mostrar el nombre y valor hora de los trabajadores*/** SELECT nombre, valor hora FROM TBLtrabajador
	*: muestra todos los campos de la tabla	**/*Mostrar todos los datos de los trabajadores*/** SELECT * FROM TBLtrabajador
	Distinct: muestra los datos sin repetir valores.	**/*Mostrar los oficios de los trabajadores con Distinct*/** SELECT DISTINCT tipo_oficio FROM TBLtrabajador
	TOP: obtener sólo una cantidad limitada de registros	**/*Mostrar los tres primeros registros de los trabajadores*/** SELECT top 3, nombre, valor hora FROM TBLtrabajador

	Operaciones aritméticas: se pueden multiplicar, sumar, restar y dividir columnas.	**/*Hallar el nombre del trabajador con el valor hora aumentado el 3%*/** SELECT NOMBRE,valor_hora + valor_hora * 0.03 AS PORCENTAJE FROM TBLtrabajador
	As: renombrado de columnas	**/*Mostrar los oficios de los trabajadores renombrado en oficio.t*/** SELECT tipo_oficio as oficio FROM TBLtrabajador
	Funciones de Agregado: se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros. • AVG calcula el promedio de los valores de un campo determinado. • COUNT calcula el número de registros de la selección. • SUM calcula la suma de todos los valores de un campo • MAX devuelve el valor más alto de un campo especificado. • MIN devuelve el valor más bajo de un campo especificado. **Nota:** todas estas funciones requieren de un parámetro que para todos debe ser de tipo numérico excepto para la función **count (*)** que cuenta sobre cualquier columna y da el mismo valor.	**/*Hallar el total el máximo de los valores hora*/** SELECT SUM(valor_hora) AS total, MAX(valor_hora) AS maximo FROM TBLtrabajador

FROM	En la cláusula FROM dice cuáles son las tablas, vistas, funciones, tablas derivadas o expresiones de tablas comunes involucradas en la consulta.	**/*Hallar los nombres de los trabajadores y el Número de días asignado */** SELECT TBLtrabajador.nombre, TBLasignar.num_dias FROM TBLtrabajador INNER JOIN TBLasignar ON TBLtrabajador.cedula =TBLasignar.cedula
	Consulta utilizando tablas temporales o alias para el ejemplo Tra Y do	**/*Hallar los trabajadores que tengan el mismo valor hora*/** SELECT tra.nombre,tra.valor_hora FROM TBLtrabajador tra,TBLtrabajador do where tra.valor_hora=do.valor_hora and tra.cedula<>do.cedula group by tra.valor_hora,tra.nombre
Cláusula WHERE	La cláusula WHERE se utiliza para aplicar filtros al conjunto de resultados; para ello existen operadores lógicos AND, OR, NOT, EXISTS con los condicionantes <, >, =, y BETWEEN. No hay límite para el número de condiciones a establecer. El orden de prioridad de los operadores lógicos es NOT, seguido de AND y OR. Se pueden utilizar paréntesis para suplantar esta prioridad en una condición de búsqueda.	**/*Hallar los nombres de los trabajadores que tienen un valor hora mayor que 10000*/** select nombre from TBLtrabajador where valor_hora>10000
	BETWEEN: Utilizado para especificar un intervalo de valores.	**/*Hallar los nombres de los trabajadores que tienen un valor hora entre 10000 y 30000*/** SELECT nombre, valor_hora FROM TBLtrabajador WHERE (valor_hora BETWEEN 10000 AND 30000)

	LIKE: Utilizado en la comparación de cadenas	**/* Hallar el nombre del trabajador que el nombre tenga la cadena ar*/** select * from TBLtrabajador where nombre like '%ar%'
	IN: Utilizado para especificar registros de una base de datos	**/*Hallar los datos de los trabajadores Nelson Fernando o Gabriel*/** select * from TBLtrabajador where nombre in('Nelson',' Fernando',' Gabriel')
Cláusula Junta interna	Cuando se necesita recuperar información de más de una tabla, se suele especificar cuáles son las filas coincidentes entre ambas tablas (columnas Clave Ajena / Clave Primaria Para ello, hay una serie de operadores que condicionan dicho filtro.	**/*Hallar los nombres de los trabajadores y el Número de días asignado si es mayor que 15*/** SELECT TBLtrabajador.nombre, TBLasignar.num_dias FROM TBLtrabajador INNER JOIN TBLasignar ON TBLtrabajador.cedula =TBLasignar.cedula WHERE TBLasignar.num_dias > 15
Cláusula Junta externa	La sentencia **LEFT JOIN** combina los valores de la primera tabla con los valores de la segunda tabla. Siempre devolverá las filas de la primera tabla, incluso aunque no cumplan la condición	SELECT * FROM tbltrabajador LEFT JOIN tblasignar On tbltrabajado.cedula = tblasignar.cedula
	La sentencia **RIGHT JOIN** combina los valores de la primera tabla con los valores de la segunda tabla. Siempre devolverá las filas de la segunda tabla, incluso aunque no cumplan la condición. En algunas bases de datos, la sentencia RIGHT JOIN es igual a RIGHT OUTER JOIN	SELECT * FROM tbltrabajador RIGHT JOIN tblasignar On tbltrabajado.cedula = tblasignar.cedula
	La sentencia **FULL JOIN** combina los valores de la primera tabla con los valores de la segunda tabla. Siempre devolverá las filas de las dos tablas, aunque no cumplan la condición. La sentencia FULL JOIN es la unión de LEFT JOIN y RIGHT JOIN	SELECT * FROM tbltrabajador FULL JOIN departamentos tblasignar On tbltrabajado.cedula = tblasignar.cedula

láusula GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos.	**/*Hallar la cantidad de trabajadores por cada oficio*/** Select tipo_oficio,count(*)as total from TBLtrabajador group by tipo_oficio
Cláusula Having	Utilizada para expresar la condición que debe satisfacer cada grupo.	**/*Hallar el total de días asignado a cada trabajador si este es mayor que 20*/** SELECT cedula, SUM(num_dias) AS total FROM TBLasignar GROUP BY cedula HAVING (SUM(num_dias) > 20)
Cláusula Order by	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico.	**/*Hallar los datos los trabajadores ordenados por nombre.*/** SELECT * from TBLtrabajador order by nombre

Subconsultas	Son consultas dentro de una consulta, estas están en las clausulas Where o having, después de los operadores de Comparación relacionales, in, all o any. **Nota 1:**Utiliza los operadores in o exists **Nota 2:** estas consultas solo devuelven valores de la tabla de la consulta más externa	**/*Hallar el nombre y el total de días asignado a cada trabajador mostrar si el total de días es mayor que el mínimo valor de num_horas*/** SELECT TBLasignar.cedula,TBLtrabajador.nombre,sum(num_dias) FROM TBLasignar inner join TBLtrabajador on TBLasignar.cedula=TBLtrabajador.cedula GROUP BY TBLasignar.cedula,TBLtrabajador.nombre having sum(num_dias)> (select min(num_horas) from oficio)
Anidamiento a varios niveles	Las subconsultas se anidan según las tablas involucradas.	**/*Hallar los nombres de los trabajadores y el Número de días asignado a los edificios tipo oficina o comercio */** SELECT TBLtrabajador.nombre, TBLasignar.num_dias FROM TBLtrabajador WHERE cedula in(select cedula from tblasignar Where ident_edif in (select ident_edif from tbledificio WHERE tipo in('oficina', 'comercio')))

Diferencia	Con NOT IN	**/*Hallar el nombre y valor_hora de los trabajadores que no están asignados a ningún edificio*/** **Estas consultas con NOT IN se identifican como diferencia** SELECT nombre,valor_hora FROM TBLtrabajador where cedula not in (select cedula from TBLasignar)
	Con NOT exists evalúa si la sub consulta es verdadera o falsa	**/*Hallar los nombres de los trabajadores que no se han asignado a ningún edificio */** **Estas consultas con NOT EXIST se identifican como diferencia** SELECT nombre FROM TBLtrabajador WHERE (NOT EXISTS (SELECT * FROM TBLasignar WHERE TBLtrabajador.cedula = TBLasignar.cedula))

Comparación con un valor	Hay un dato desconocido que el sistema debe hallar en la subconsulta	**/*Mostrar los nombres de los trabajadores que tienen un valor hora mayor que el promedio de todos los valores hora de los trabajadores*/** SELECT nombre FROM TBLtrabajador where valor_hora> (select avg(valor_hora) from TBLtrabajador
Comparación de varios valores	Hay varios datos desconocidos que el sistema debe hallar en la subconsulta.	**/*Hallar el nombre los nombres de los trabajadores que tienen un valor hora mayor que el valor hora de Nelson Fernando o Gabriel*/** SELECT nombre FROM TBLtrabajador where valor_hora>any (select valor_hora from TBLtrabajador where nombre in('nelson','fernando','gabriel'))

Consultas de parámetros	Utiliza variables para recibir los valores de comparación de la consulta	**/*mostrar un mensaje Hola Buen Día*/** declare @nombre varchar(50)-- declare declara una variable -- @nombre es el identificador de la -- variable de tipo varchar set @nombre = 'Hola Buen Día' -- El signo = es un operador print @Nombre -- Imprime por pantalla el valor de @nombre. -- No diferencia mayúsculas ni minúsculas **/* devolver un único registro en variable****/** DECLARE @nombre VARCHAR(50) -- La consulta debe devolver un único registro SET @nombre = (SELECT nombre FROM tbltrabajador WHERE cedula = 1235) PRINT @nombre

Consultas de referencias cruzadas	Intercambia una fila por una columna A veces es necesario hacer girar los resultados, de forma que las columnas se presenten horizontalmente y las filas se presenten verticalmente[1]. Esta instrucción SELECT realiza también el tratamiento de una tabla en la que hay varias filas por cada trimestre. La cláusula GROUP BY combina todas las filas de PPivot de un año determinado en una única fila del resultado. Cuando se realiza la operación de agrupamiento, las funciones CASE de los agregados SUM se aplican de tal forma que los valores Amount de cada trimestre se agregan a la columna adecuada del conjunto de resultados, y se agrega 0 a las columnas del conjunto de resultados del resto de los trimestres.	**Ejemplo tomado de:** Database tool for MS SQL **/******referencias cruzadas****/** Create database referenciascruzadas USE referenciascruzadas GO CREATE TABLE PPivot (Year SMALLINT, Quarter TINYINT, Amount DECIMAL(2,1)) GO INSERT INTO PPivot VALUES (1990, 1, 1.1) INSERT INTO PPivot VALUES (1990, 2, 1.2) INSERT INTO PPivot VALUES (1990, 3, 1.3) INSERT INTO PPivot VALUES (1990, 4, 1.4) INSERT INTO PPivot VALUES (1991, 1, 2.1) INSERT INTO PPivot VALUES (1991, 2, 2.2) INSERT INTO PPivot VALUES (1991, 3, 2.3) INSERT INTO PPivot VALUES (1991, 4, 2.4) GO

		Ésta es la instrucción SELECT que se utiliza para crear resultados girados: SELECT Year, SUM(CASE Quarter WHEN 1 THEN Amount ELSE 0 END) AS Q1, SUM(CASE Quarter WHEN 2 THEN Amount ELSE 0 END) AS Q2, SUM(CASE Quarter WHEN 3 THEN Amount ELSE 0 END) AS Q3, SUM(CASE Quarter WHEN 4 THEN Amount ELSE 0 END) AS Q4 FROM referenciascruzadas.dbo.PPivot GROUP BY Year GO

[1] Tomado de Database tool for MS SQL (s.f.). Recuperado de: http://www.todoexpertos.com/categorias/tecnologia-e-internet/bases-de-datos/sql-server/respuestas/208707/referancias-cruzadas

	Si los resultados de esta instrucción SELECT se usan como entrada de una hoja de cálculo, es fácil para la hoja de cálculo calcular un total para cada año. Cuando se usa SELECT desde una aplicación, puede ser más fácil mejorar la instrucción SELECT para calcular el total anual.	SELECT P1.*, (P1.Q1 + P1.Q2 + P1.Q3 + P1.Q4) AS YearTotal FROM (SELECT Year, SUM(CASE P.Quarter WHEN 1 THEN P.Amount ELSE 0 END) AS Q1, SUM(CASE P.Quarter WHEN 2 THEN P.Amount ELSE 0 END) AS Q2, SUM(CASE P.Quarter WHEN 3 THEN P.Amount ELSE 0 END) AS Q3, SUM(CASE P.Quarter WHEN 4 THEN P.Amount ELSE 0 END) AS Q4 FROM PPivot AS P GROUP BY P.Year) AS P1 GO

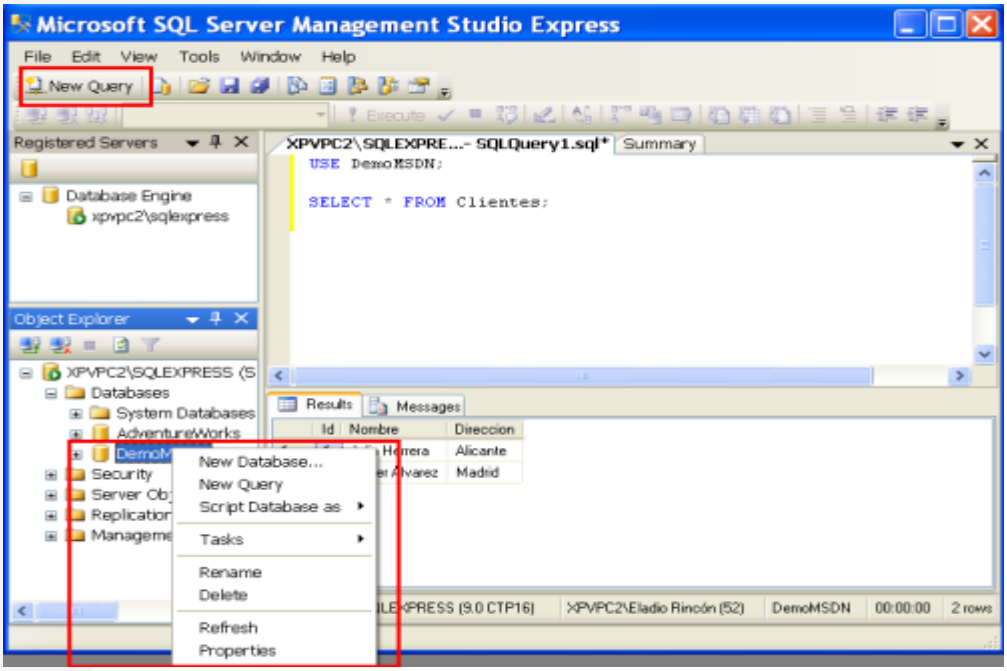
Recomendación: Si se genera código "genérico" que recupera todas las columnas (las columnas serán necesarias unas veces sí y otras no) se estará penalizando el rendimiento del servidor SQL Server Express pero se estará ganando tiempo de desarrollo.

Como Ejecutar bases de datos en el editor SQL

A continuación encuentra una guía[2] de cómo ejecutar cualquier base de datos en el editor de SQL server para este caso en versión 2008.

Para ejecutar una consulta desde SQL Server 2008 Management Studio Express, conectaremos a la base de datos **DemoMSDN**, desde una de las opciones marcada en la siguiente imagen en recuadro rojo:

[2] Microsoft SQL Server Management Studio Express - Community Technology Preview (CTP) (2008). Recuperado en noviembre de 2015 de http://www.desarrollaconmsdn.com/msdn/CursosOnline/Curso_SQL_Server/index.html



A continuación, deberás escribir el siguiente texto para rellenar unas cuantas filas en la tabla clientes (veremos luego la sentencia INSERT):

INSERT Clientes SELECT 'Julia Herrera', 'Alicante';

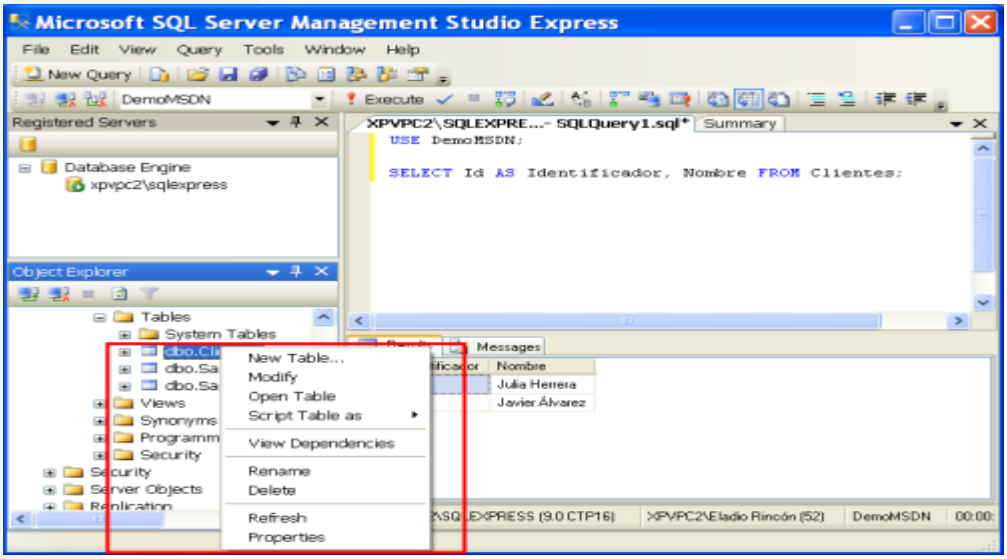
Para ello, copia el texto en la ventana de texto, y pulsa F5, o el botón Execute para ejecutar la sentencia.

Borra el texto de la sentencia, y escribe el siguiente texto:

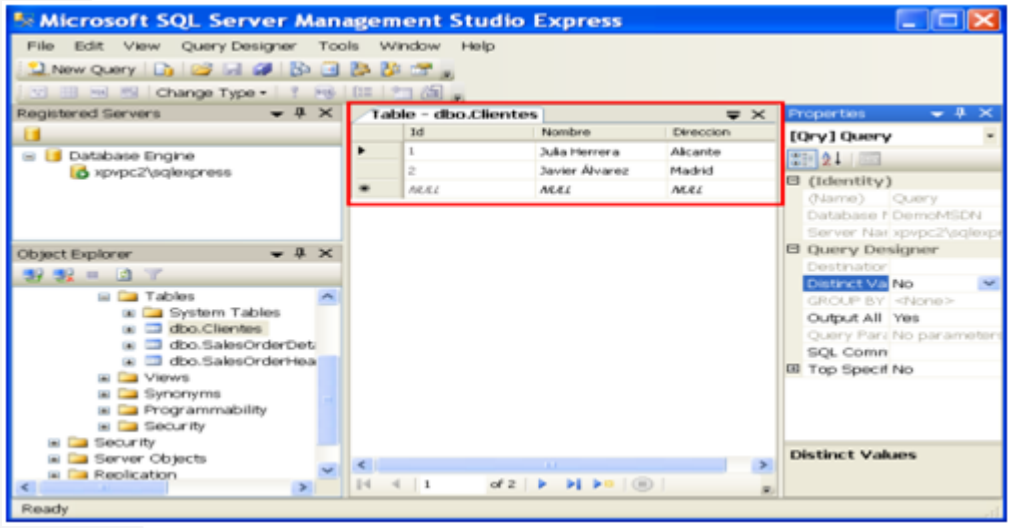
SELECT Id AS Identificador, Nombre FROM Clientes;

Ejecuta la instrucción y verás como resultado todas las filas de la tabla Clientes. Fíjate que la columna de base de datos Id, ahora parece que se llama Identificador. Esto es un alias de columna; habrá ocasiones en las que necesites personalizar el nombre de columna que se muestra.

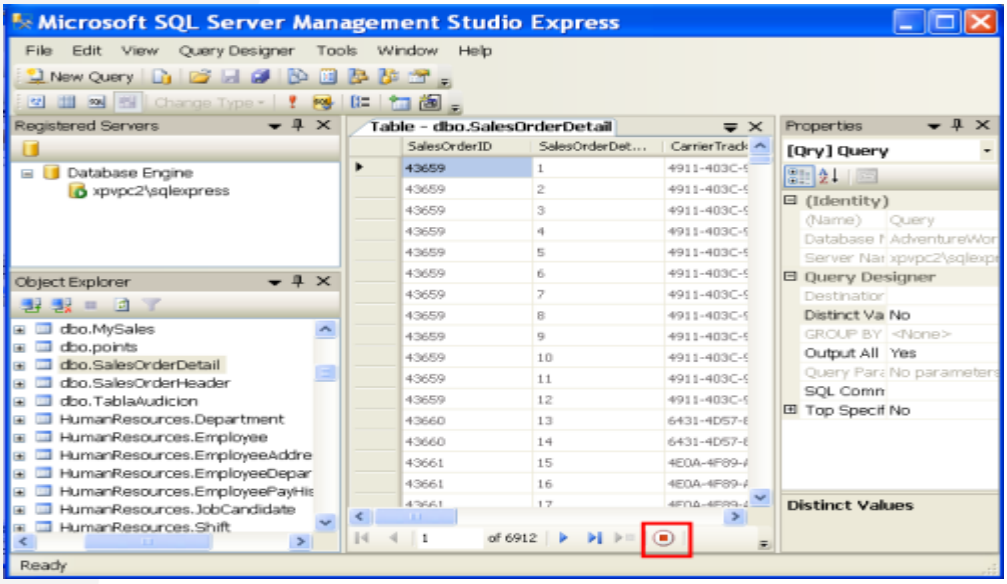
Desde Management Studio, también se puede ver la información de la tabla; para ello, deberás seleccionar la tabla que quieres editar, botón derecho del ratón, y elegir la opción "Open Table" como aparece en la siguiente imagen:



Y como resultado tendremos:



Se debe tener cuidado cuando se abren tablas muy grandes, porque el proceso de carga es más costoso cuanto mayor sea el tamaño de la tabla. Sin embargo, en la versión 2008 de las herramientas administrativas, tenemos la posibilidad de cancelar consultas mientras se está realizando la petición; fíjate en el botón marcado en color rojo en la siguiente imagen; pulsando dicho botón se solicita al servidor que se cancele la consulta.



REFERENCIAS BIBLIOGRÁFICAS

Microsoft SQL Server Management Studio Express - Community Technology Preview (CTP) (2008). Recuperado en noviembre de 2015 de http://www.desarrollaconmsdn.com/msdn/CursosOnline/Curso_SQL_Server/index.html

Database tool for MS SQL (s.f.). Recuperado de: http://www.todoexpertos.com/categorias/tecnologia-e-internet/bases-de-datos/sql-server/respuestas/208707/referancias-cruzadas

INSTITUCIÓN UNIVERSITARIA
PASCUAL BRAVO

Unidad de Educación Virtual