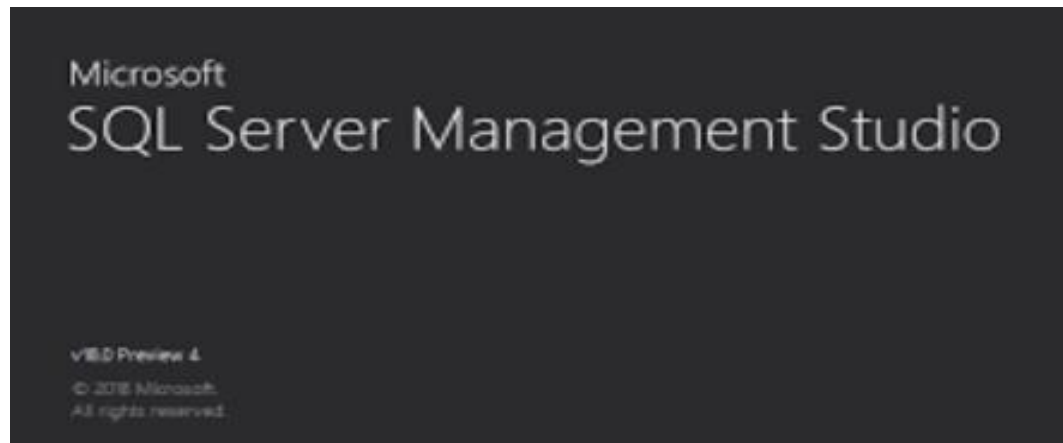


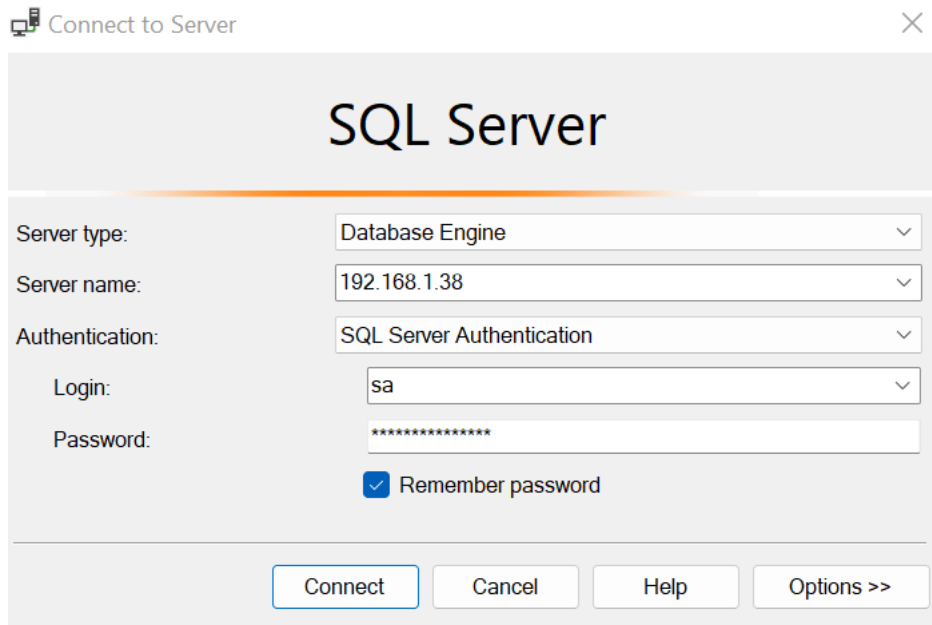
Consultas Básicas SQL

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional.

Para trabajar con las consultas usaremos el **Management Studio** del SQL Server



Al iniciar el programa nos solicitará las credenciales de acceso



Connect to Server

SQL Server

Server type: Database Engine

Server name: 192.168.1.38

Authentication: SQL Server Authentication

Login: sa

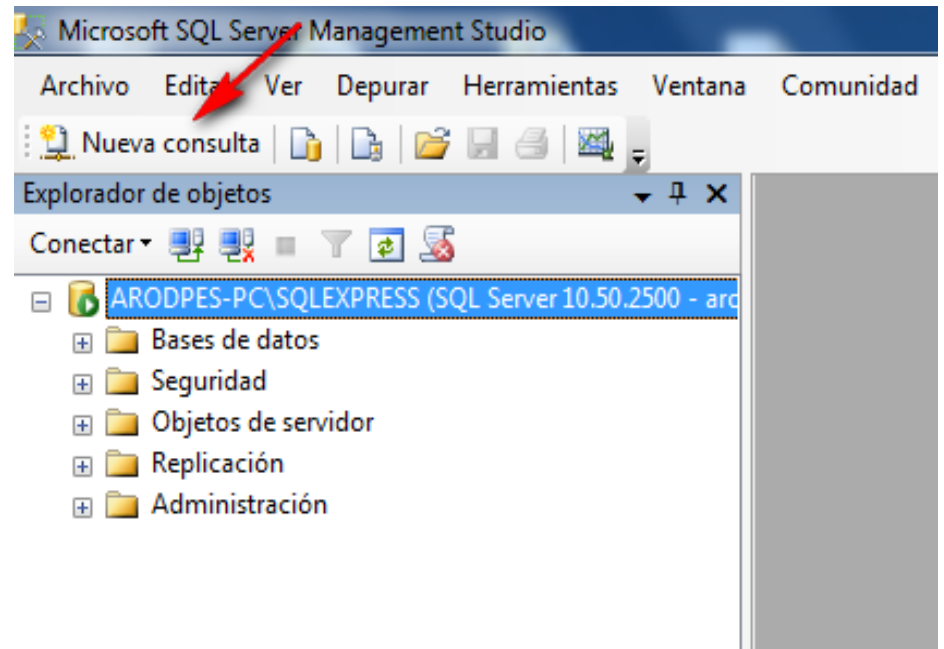
Password: *****

☒ Remember password

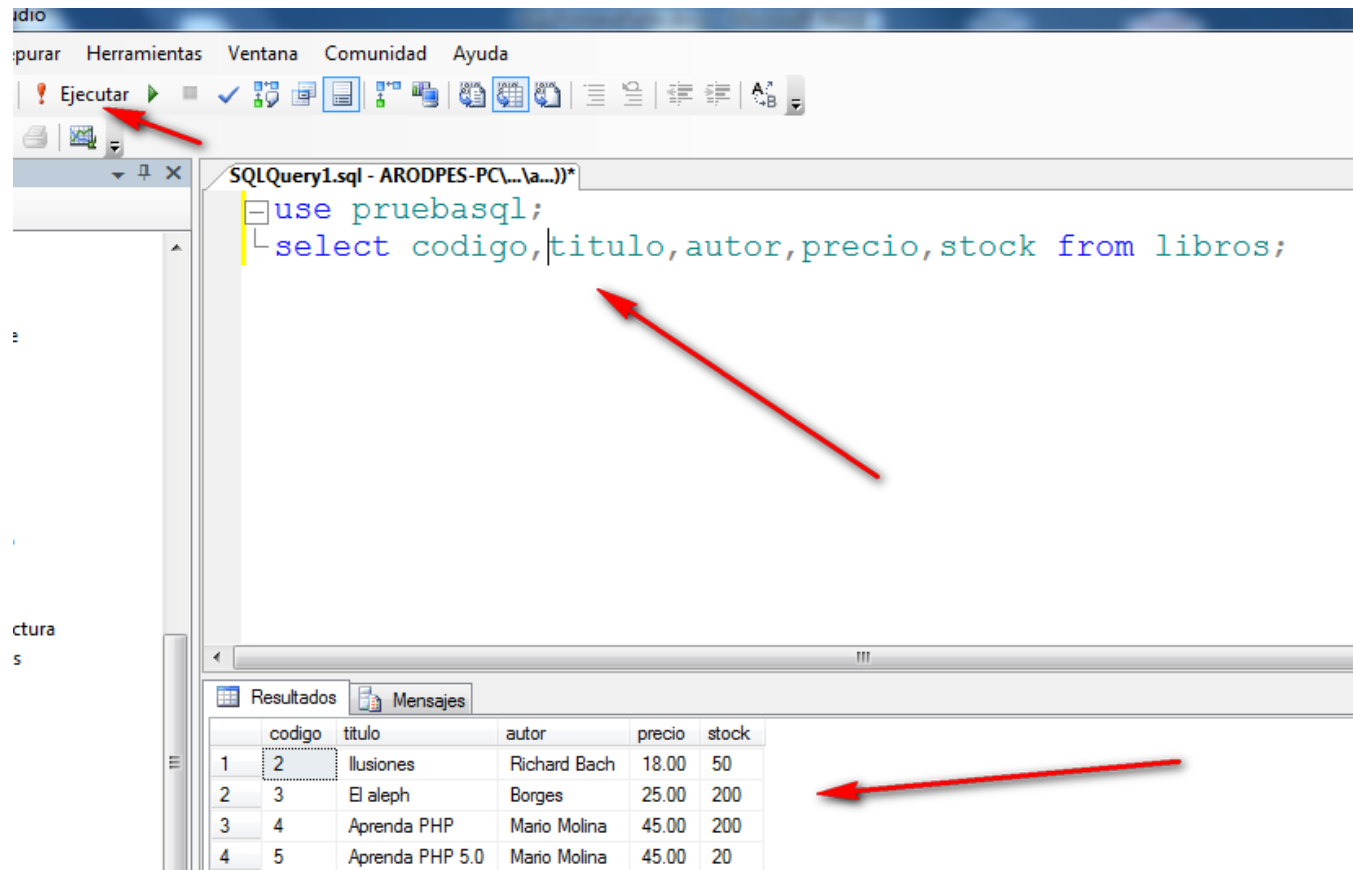
Connect Cancel Help Options >>

Como nuestro servidor está en una máquina virtual y accedemos desde la máquina anfitrión conectaremos con autenticación SQL Server.

Una vez se abra el programa tendremos disponible múltiples opciones de configuración y acceso al SGBD. Para efectuar una consulta haremos clic en Nueva consulta



Se abrirá una página en blanco donde podremos escribir las sentencias SQL



Para ejecutarla haremos clic en el botón Ejecutar, nos dará los resultados en las ventanas de debajo.

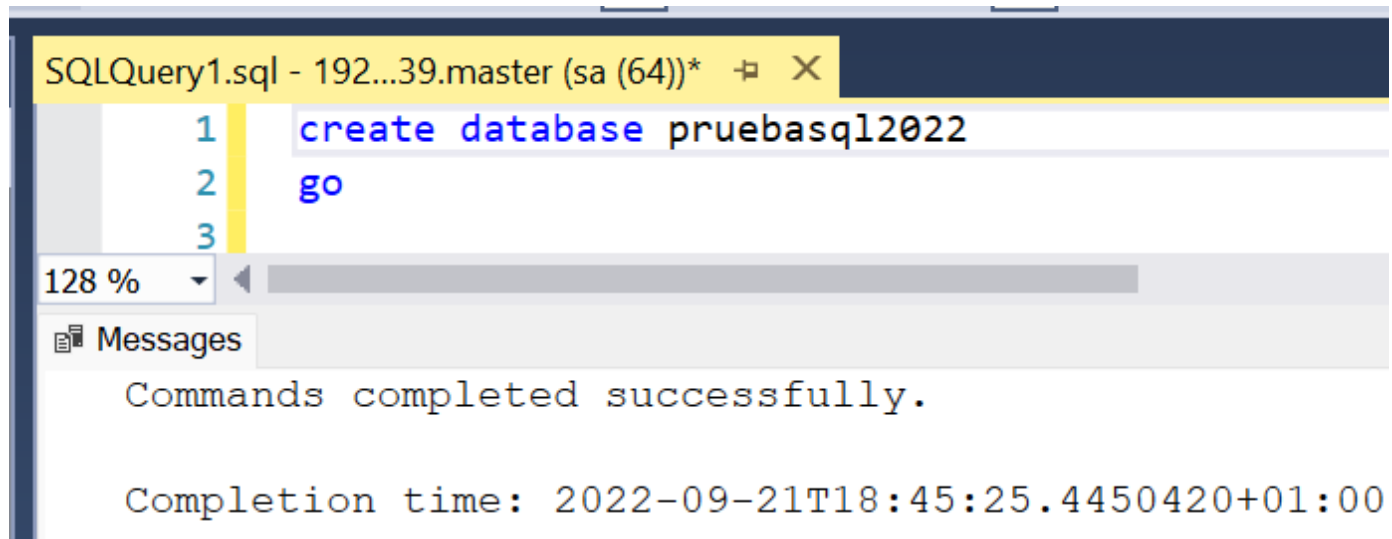
CREAR UNA BASE DE DATOS

Una **base de datos** es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Una Base de datos dentro de SQL Server es una estructura organizada que permite almacenar informaciones (tablas) identificadas por un nombre y otras estructuras y programas que faciliten el acceso a los datos.

Crear la Base de datos:

```
create database pruebasql2022  
go
```



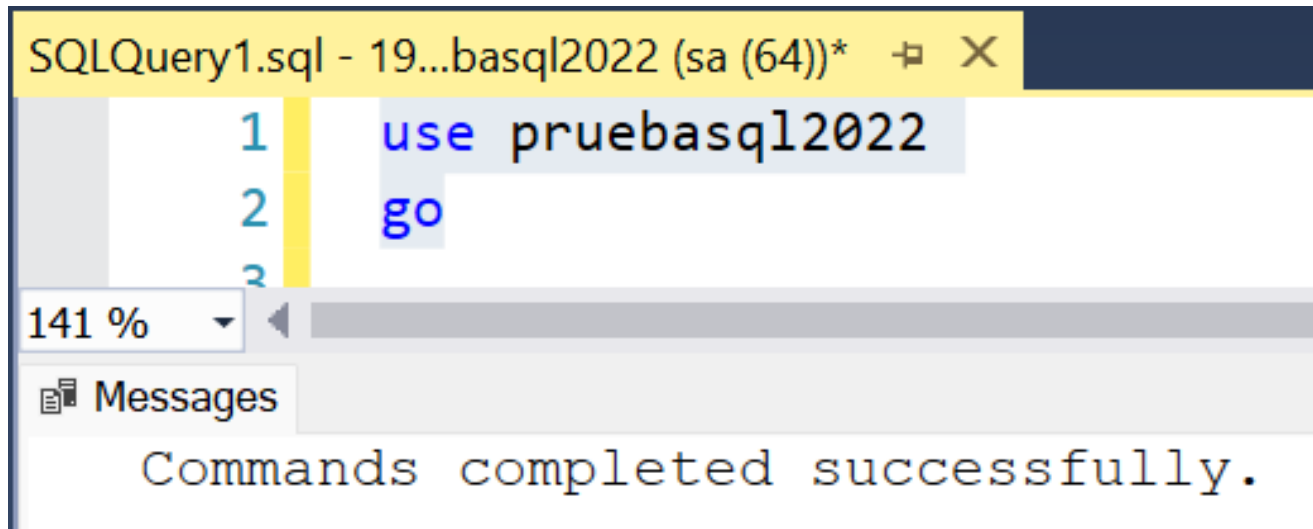
The screenshot shows a SQL Server Enterprise Manager window titled "SQLQuery1.sql - 192...39.master (sa (64))*". The query editor contains the following SQL commands:

```
1 create database pruebasql2022  
2 go  
3
```

Below the query editor, the "Messages" pane displays the following output:

```
Commands completed successfully.  
  
Completion time: 2022-09-21T18:45:25.4450420+01:00
```

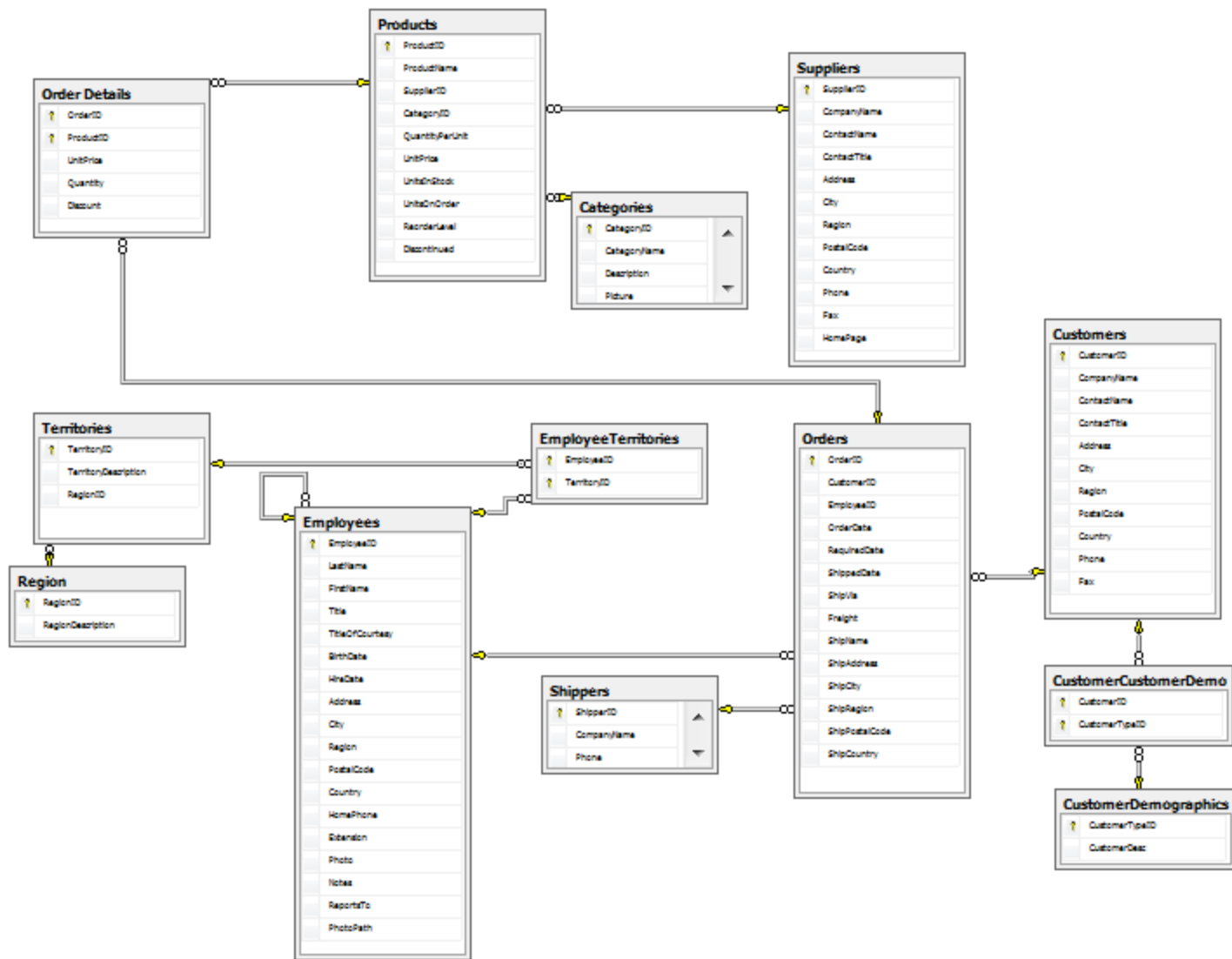
Para asegurarnos que las operaciones sobre tablas se realicen en esa Base de datos tendríamos que indicar siempre la base de datos en la que vamos a trabajar añadiendo en las sentencias:



The screenshot shows a SQL query window titled "SQLQuery1.sql - 19...basql2022 (sa (64))*". The query text is as follows:

```
1 use pruebasql2022
2 go
```

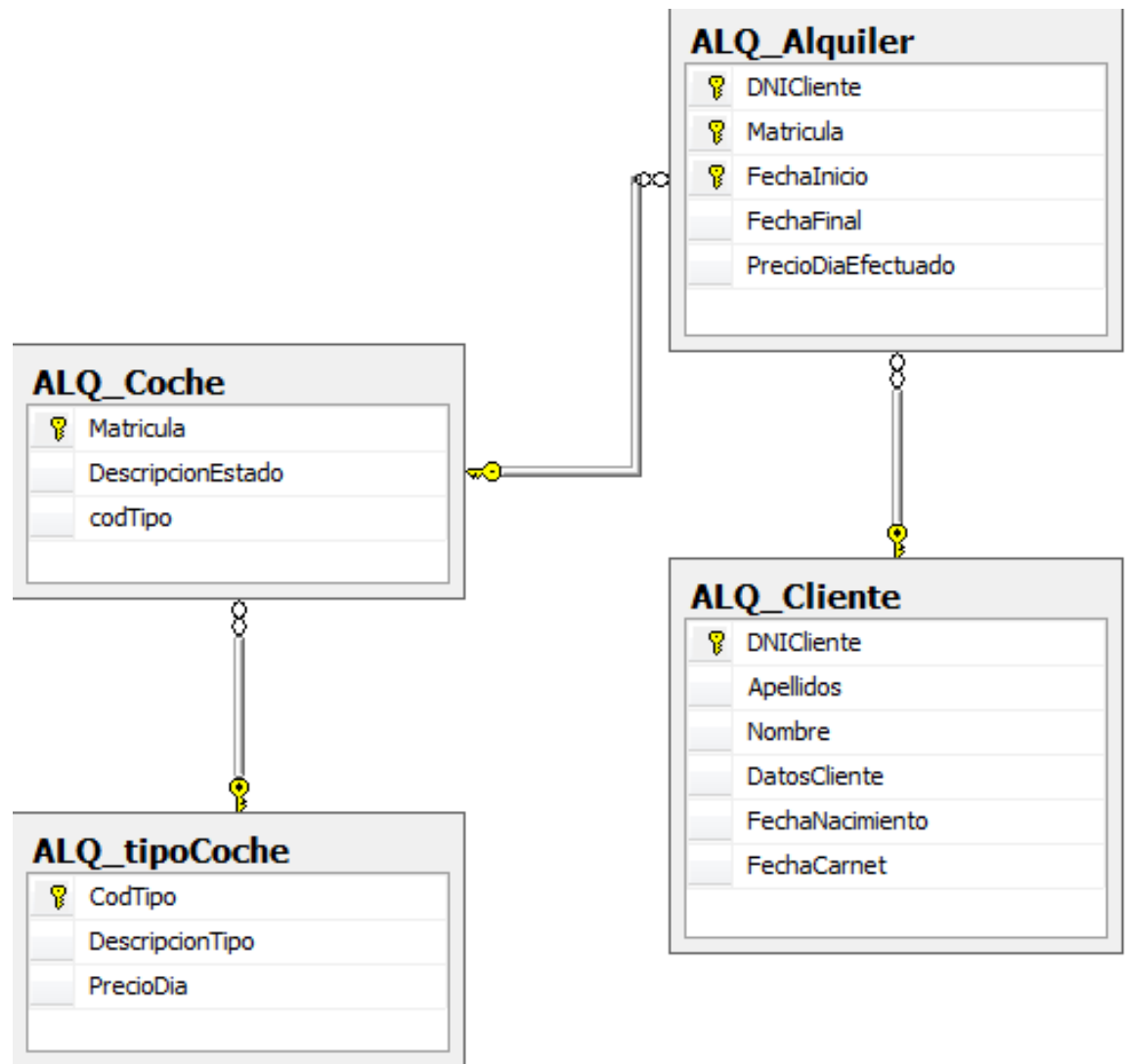
Below the query text, there is a status bar indicating "141 %". At the bottom, a "Messages" pane displays the message "Commands completed successfully."



CREAR UNA TABLA

Una tabla es la estructura dentro de una base de datos donde podemos almacenar información correspondiente a algo, cuya estructura se mantiene igual.

Tendremos que para cada elemento del que vayamos a almacenar datos, los elementos serán siempre los mismos.



Cuando se crea una tabla debemos indicar su nombre y definir al menos un campo con su tipo de dato.

El formato básico de creación de tablas será:

```
create table NOMBRE_DE_LA_TABLA
```

```
(  
  NOMBRE_CAMPO_1 TIPO_CAMPO_1 ,  
  NOMBRE_CAMPO_2 TIPO_CAMPO_2 ,
```

AÑADIREMOS TANTOS CAMPOS COMO NECESITEMOS

```
  NOMBRE_CAMPO_N TIPO_CAMPO_N  
);
```

Creamos la base de datos facturasbasicas la primera vez.

```
create database facturasbasicas  
go
```

La consideramos la base de datos por defecto

```
use facturasbasicas  
go
```

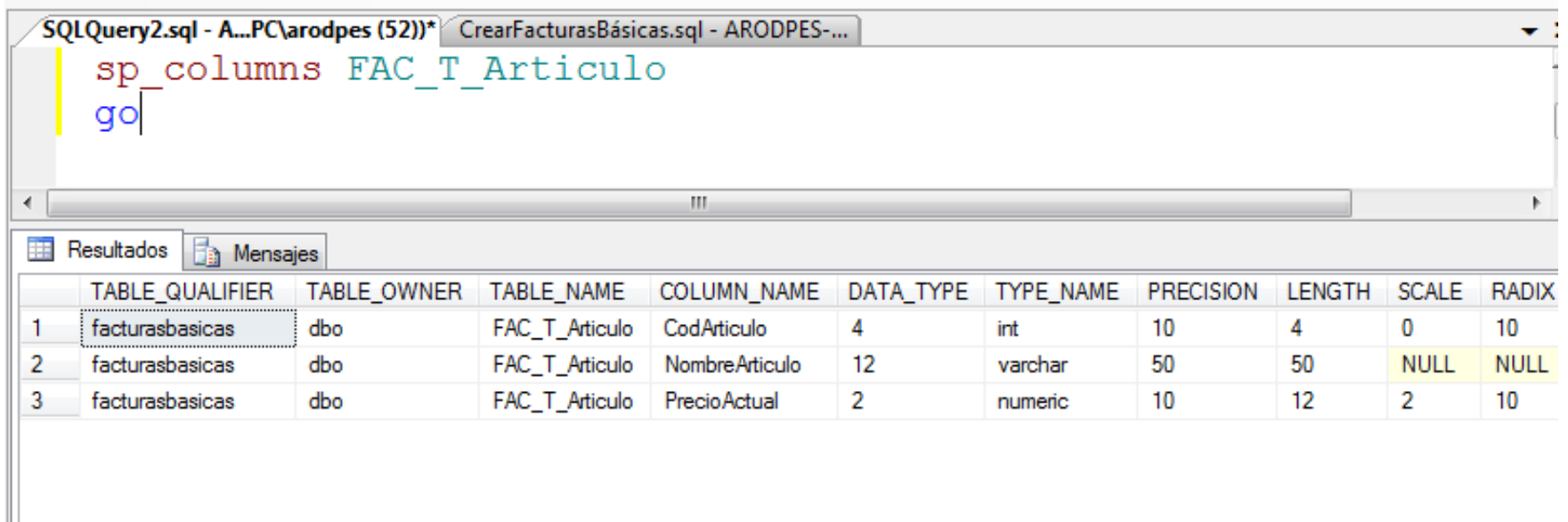
Creamos la tabla FAC_T_Articulo, donde almacenaremos los datos de los artículos de nuestra empresa

```
create table FAC_T_Articulo
(
    CodArticulo           integer,
    NombreArticulo        varchar(50),
    PrecioActual          numeric(10,2)
);
go
```


Creamos la tabla FAC_T_Cliente, donde almacenaremos los datos de los clientes de nuestra empresa

```
create table FAC_T_Cliente
(
    CodCliente          integer,
    NombreCliente       varchar(60),
    DatosCliente        varchar(60),
    FechaAlta           datetime,
    FechaNacimiento     datetime
);
go
```

Ver estructura de una tabla: **sp_columns**
nombre_de_tabla



The screenshot shows a SQL Server Enterprise Manager window with two tabs: "SQLQuery2.sql - A...PC\arodpes (52))*" and "CrearFacturasBásicas.sql - ARODPES-...". The active tab contains the SQL query: `sp_columns FAC_T_Articulo` followed by `go` on a new line. Below the query editor, there are two tabs: "Resultados" (Results) and "Mensajes" (Messages). The "Resultados" tab is active, displaying a table with 11 columns: TABLE_QUALIFIER, TABLE_OWNER, TABLE_NAME, COLUMN_NAME, DATA_TYPE, TYPE_NAME, PRECISION, LENGTH, SCALE, and RADIX. The table contains three rows of data for the "FAC_T_Articulo" table in the "dbo" schema. The first row shows the "CodArticulo" column with an integer data type. The second row shows the "NombreArticulo" column with a varchar data type and NULL values for SCALE and RADIX. The third row shows the "PrecioActual" column with a numeric data type.

	TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME	PRECISION	LENGTH	SCALE	RADIX
1	facturasbasicas	dbo	FAC_T_Articulo	CodArticulo	4	int	10	4	0	10
2	facturasbasicas	dbo	FAC_T_Articulo	NombreArticulo	12	varchar	50	50	NULL	NULL
3	facturasbasicas	dbo	FAC_T_Articulo	PrecioActual	2	numeric	10	12	2	10

Ejercicios:

Crear la base de datos MoviesBasicas en el caso en que no exista.

En esa Base de Datos crear una tabla denominada Peliculas con la estructura siguiente:

Id	entero
Titulo	cadena variable de 100 caracteres
Director	cadena variable de 100 caracteres
Agno	entero
FechaCompra	fecha hora
precio	numérico con 6 dígitos, dos de ellos decimales

En la misma Base de Datos crear otra tabla denominada Socios con la estructura siguiente:

NIFNIE	cadena fija de longitud 9
Apellidos	cadena variable de 50 caracteres
Nombre	cadena variable de 100 caracteres
Direccion	cadena variable de 100 caracteres
Telefono	cadena fija de longitud 9
FechaDeAlta	fecha hora

ELIMINAR TABLA

Para eliminar una tabla usamos **"drop table"** junto al nombre de la tabla a eliminar:

```
drop table FAC_T_Articulo;
```

Las tablas eliminadas desaparecen, se pierde su estructura y todos los datos que tiene almacenados

Si intentamos eliminar una tabla que no existe, aparece un mensaje de error indicando tal situación y la sentencia no se ejecuta. Con esto lo evitamos:

```
if object_id('FAC_T_Articulo') is not null  
    drop table FAC_T_Articulo;  
go
```

Hay que tener en cuenta que este NO es el procedimiento habitual, ya que las tablas usualmente no se borran nunca, ya que perderíamos la información que contienen. En este caso lo haremos para asegurarnos unos resultados en nuestras pruebas de aprendizaje.

Para ver las tablas de trabajo que hay en la base de datos haremos:

```
sp_tables @table_owner='dbo';
```

Ejercicios:

En la Base de Datos facturasbasicas crear las tablas
FAC_T_Cliente y FAC_T_Articulo borrándolas previamente
en caso en que existan.
Verificar qué tablas tenemos y cuál es su estructura.

AÑADIR INFORMACIÓN A UNA TABLA

Añadir información a una tabla

Hay varias formas de añadir registros/filas en una tabla.

Añadiendo la información de un registro para todos los campos/atributos de la tabla y en el orden en que se definieron.

```
insert into nombre_de_la_tabla  
values (valorcampo1, valorcampo2, ..., valorcampoN);
```

También dando el conjunto de campos a los que le daremos valores.

```
insert into nombre_de_la_tabla  
(nombrecampo1, nombrecampo2, ... , nombrecampoN)  
values (valorcampo1, valorcampo2, ..., valorcampoN);
```

Valores de cadenas de caracteres y de fechas entre comillas simples.

Siempre que grabemos un campo fechahora tendremos que determinar el formato con el que lo escribiremos.

Esto se hace en SQLServer mediante:

set dateformat dmy;

Representando en este caso que primero colocaremos el día, después el mes y finalizaremos con el año (**d**ay/**m**onth/**y**ear).

Se pueden elegir otros formatos, como por ejemplo ymd, mdy, ...

```
insert FAC_T_Cliente  
(CodCliente,NombreCliente,DatosCliente,FechaAlta,FechaNacimiento)  
values ( 1,'Antonio','C/uno nº 3','01/03/2012','15/04/1970')
```

```
insert FAC_T_Cliente  
(CodCliente,NombreCliente,DatosCliente,FechaAlta,FechaNacimiento)  
values ( 2,'Juan','C/la hornera nº 7' , '22/05/2012','29/06/1982' )
```

```
insert FAC_T_Cliente  
values ( 3,'María','C/el pino nº 7','22/05/2010','15/06/1960')
```

Ejercicios:

En la tabla Peliculas de la base de datos MoviesBasicas.

Insertar las siguientes informaciones:

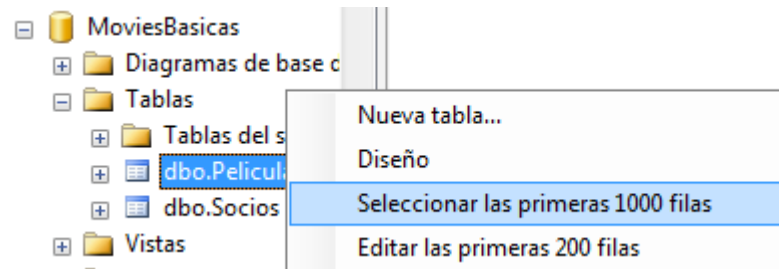
- 1, 'Rashomon','Akira Kurosawa',1951,'01/01/2012'
- 2, 'Forrest Gump','Robert Zemeckis',1994,'01/02/2012'
- 3, 'La Fiera de mi Niña','Howard Hawks',1938,'01/03/2012 '

Insertar en un registro en los campos
Id, Agno,Titulo,Director

los valores

33, 1956,'Moby Dick','John Huston'

Ver el contenido de la tabla.



**RECUPERAR LA INFORMACIÓN
ALMACENADA**

Recuperar la información almacenada

select lista de campos separados por comas from NOMBRETABLA;

Podemos usar un asterisco (*) en la lista de campos lo que indica que se seleccionan todos los campos de la tabla, aunque esto hace que la consulta sea menos eficiente.

Quedaría:

```
select * from NOMBRETABLA
```

Pero esta forma es muy poco eficiente.

Hace que el SGBD tenga que enviar la estructura de la tabla cada vez que se ejecuta. **Por tanto NO se admitirá como válida.**

Ejemplo

Usar el archivo que genera la Base de Datos facturasbasicas, creando y cargando con datos las tablas FAC_T_Articulo y FAC_T_Cliente

Mostrar todos los datos de la tabla FAC_T_Cliente:

```
select  
CodCliente,NombreCliente,DatosCliente,FechaAlta,FechaNacimiento  
from FAC_T_Cliente;
```

Sería equivalente a `select * from FAC_T_Cliente`, pero esta última manera es menos eficiente.

Mostrar el NombreArticulo y PrecioActual de la tabla FAC_T_Articulo:

```
select NombreArticulo,PrecioActual from FAC_T_Articulo;
```


Ejercicio

Usar el archivo que genera la Base de Datos MoviesBasicas, creando y cargando con datos las tablas Peliculas y Socios

Mostrar los datos de la tabla Peliculas

Mostrar el Título y Director de las películas.

Mostrar Apellidos y nombre de todos los Socios.

**RECUPERAR LA INFORMACIÓN
ALMACENADA DE ALGUNOS REGISTROS**

Recuperar la información almacenada de algunos registros

```
select NOMBRECAMPO1, ..., NOMBRECAMPOn  
from NOMBRETABLA  
where CONDICION;
```

Mostrará los campos indicados de las filas que cumplan la condición especificada.

Ejemplo

Mostrar los datos de los artículos con precio igual a 12:

```
select CodArticulo,NombreArticulo,PrecioActual  
from FAC_T_Articulo  
where PrecioActual=12;
```

Mostrar los datos de los Clientes con Fecha de alta 23/01/2010:

```
set dateformat dmy  
select  
CodCliente,NombreCliente,DatosCliente,FechaAlta,FechaNacimi  
ento  
from FAC_T_Cliente  
where FechaAlta='23/01/2010'
```

Ejercicio

Usar el archivo que genera la Base de Datos MoviesBasicas, creando y cargando con datos las tablas Peliculas y Socios

Mostrar todos los datos de la tabla Peliculas del director Francis Ford Coppola.

Mostrar Apellidos y Nombre de los Socios con nombre Juan.

Mostrar Título y Director de las películas de año 1960.

OPERADORES RELACIONALES

Operadores relacionales

Los operadores relacionales (o de comparación) nos permiten comparar dos expresiones, que pueden ser variables, valores de campos, etc.

Los operadores son símbolos que permiten realizar operaciones matemáticas, concatenar cadenas, hacer comparaciones.

SQL Server tiene 4 tipos de operadores:

- relacionales (o de comparación)
- aritméticos
- de concatenación
- lógicos.

Los operadores relacionales son los siguientes:

=	igual
<>	distinto
>	mayor
<	menor
>=	mayor o igual
<=	menor o igual

Ejemplo

Mostrar los datos de los artículos con precio mayor que 12:

```
select CodArticulo,NombreArticulo,PrecioActual  
from FAC_T_Articulo  
where PrecioActual>12;
```

Mostrar los datos de los Clientes con Fecha de alta anterior a 23/01/2010:

```
set dateformat dmy  
select  
CodCliente,NombreCliente,DatosCliente,FechaAlta,FechaNacimiento  
from FAC_T_Cliente  
where FechaAlta<'23/01/2010'
```

Ejemplo

Mostrar los datos de los artículos con precio distinto de 12:

```
select CodArticulo,NombreArticulo,PrecioActual  
from FAC_T_Articulo  
where PrecioActual<>12;
```

Mostrar los datos de los Clientes con Fecha de alta 23/01/2010 o posterior:

```
set dateformat dmy  
select  
CodCliente,NombreCliente,DatosCliente,FechaAlta,FechaNacimiento  
from FAC_T_Cliente  
where FechaAlta>='23/01/2010'
```

Ejercicio

Usar el archivo que genera la Base de Datos MoviesBasicas, creando y cargando con datos las tablas Peliculas y Socios

Mostrar todos los datos de la tabla Peliculas que no sean del director Francis Ford Coppola.

Mostrar Título y Director de las películas del año 1960 o posteriores.

BORRAR REGISTROS

Borrar registros

El comando para eliminar registros de una tabla es el **delete**.

Eliminar registros es borrar de manera irreversible información de la tabla.

Se hará siempre con precaución.

El formato básico es:

delete from NOMBREDETABLA;

Con este comando se eliminarán TODOS los registros de la tabla.

TODOS

Normalmente especificaremos la condición de filtrado de registros a borrar:

**delete from NOMBREDETABLA
where CONDICIÓN**

Vigilando que se corresponda con el conjunto de registros a eliminar.

Ejemplo

Borrar los artículos con precio menor que 12:

```
delete from FAC_T_Articulo  
where PrecioActual < 12;
```

Borrar los clientes Clientes con Fecha de alta posterior a 23/01/2010:

```
set dateformat dmy  
delete from FAC_T_Cliente  
where FechaAlta > '23/01/2010'
```

Ejercicio

Usar el archivo que genera la Base de Datos MoviesBasicas, creando y cargando con datos las tablas Peliculas y Socios.

En todos los casos comprobar las tabas antes y después del borrado.

Borrar las películas del año 1975.

Borrar la película de título Gandhi.

Borrar los Socios con fecha de alta posterior a '31/12/2008'

ACTUALIZAR REGISTROS

Actualizar registros

Actualizar registros es cambiar parte de su contenido, manteniendo el resto de su información invariable.

El comando para actualizar información de los registros de una tabla es el **update**.

Se hará siempre con precaución. Si no filtramos los registros a actualizar mediante el where, la actualización afectará a **TODOS** los registros.

El formato básico es:

```
update NOMBREDETABLA  
set CAMPO=nuevovalor;
```

Con este comando se modificará el valor del CAMPO, asignándole nuevovalor en TODOS los registros de la tabla.

Normalmente especificaremos la condición de filtrado de registros a actualizar:

```
update NOMBREDETABLA  
set CAMPO=nuevovalor  
where CONDICIÓN;
```

Vigilando que se corresponda con el conjunto de registros a actualizar y que el nuevovalor sea el deseado.

Ejemplo

Subir dos euros a los artículos con precio menor que 12:

```
update FAC_T_Articulo  
    set precioactual=precioactual+2  
    where PrecioActual<12;
```

Modificar el cliente de codcliente 7 cambiando su nombre a Ana María

```
update FAC_T_Cliente  
    set nombreciente='Ana María'  
    where codcliente=7
```

Ejercicio

Usar el archivo que genera la Base de Datos MoviesBasicas, creando y cargando con datos las tablas Peliculas y Socios.

En todos los casos comprobar las tabas antes y después de la actualización.

Subir un 10% el precio de las películas de año 1980

Modificar la fecha de compra de la película La Fiera de mi Niña al 15/2/2013.

Cambiar el director de todas las películas de Joseph L. Mankiewicz a **Joseph Leo Mankiewicz**

Subir un euro a las películas de menos de 4 euros.

COMENTARIOS

Podremos usar símbolos en los archivos sql de manera que podamos documentar las operaciones que realizamos pero sin afectar a la funcionalidad de nuestro código.

Lo que coloquemos en una línea después de dos guiones medios no se ejecutará.

```
select Apellidos, Nombre from socios; -- esto es para...
```

Lo que coloquemos entre los símbolos `/*` y `*/` no se ejecutará. Puede ocupar más de una línea.

```
select Apellidos, Nombre /* comenzamos a documentar  
seguimos documentando  
finalizamos comentario */ from socios;
```

VALOR NULL

null significa "dato desconocido" o "valor inexistente". No es lo mismo que un valor "0", una cadena vacía o una cadena literal "null".

Podemos asignarlo directamente, colocando NULL en la lista de valores de un insert o no asignándole valor al no añadir el campo en la lista de campos del insert.

Colocando valor NULL

```
insert FAC_T_Cliente (  
CodCliente,NombreCliente,DatosCliente,FechaAlta,FechaNacimiento )  
values ( 998,'Antonia','C/uno nº 3','01/03/2012',NULL)
```

Prescindiendo del campo FechaNacimiento

```
insert FAC_T_Cliente (  
CodCliente,NombreCliente,DatosCliente,FechaAlta,) values (  
999,'Juana','C/la hornera nº 7' , '22/05/2012')
```

A los campos de una tabla se les puede añadir una especificación que indica si se admiten o no valores null.

```
create table NOMBRETABLA  
(  
    ...  
    campoA tipo null,  
    campoB tipo not null,  
    ...  
)
```

El campoA puede no asignarse (o asignarse a NULL) y el campoB es obligatorio, siempre hay que darle valor y nunca puede ser NULL.

En las sentencias select podemos validar si un campo es null.

```
select Nombrecliente  
from FAC_T_Cliente  
where fechanacimiento is null
```

```
select nombrecliente  
from FAC_T_Cliente  
where DatosCliente is null
```

No es lo mismo NULL que vacío:

```
select nombrecliente  
from FAC_T_Cliente  
where DatosCliente =''
```

No es lo mismo
NULL
' '
' '
"
'NULL'

Para comprobar que tenga valor será mediante:

IS NOT NULL

Aquí vemos que la fechanacimiento tenga valor:

```
select Nombrecliente  
from FAC_T_Cliente  
where fechanacimiento is not null;
```

No valdrá nunca colocar:

<>NULL
= NULL

Ejercicio

Usar el archivo que genera la Base de Datos MoviesBasicas, creando y cargando con datos las tablas Peliculas y Socios.

Películas con director a NULL

Películas con director en blanco

Crear la tabla peliculas2, con la misma estructura que la tabla peliculas, pero sin que permita datos null en título ni en director.

Probar inserciones con valor null y sin él en los campos modificados.