Universidad Tecnológica de Panamá
Facultad de Ingeniería de Sistemas Computacionales
Sistemas de Base de Datos I
Laboratorio N°2

Facilitador: Víctor A. Fuentes T.
Estudiante: Johel Heraclio Batista Cárdenas
Cédula: 8-914-587
Grupo: 1IF-131

A. TÍTULO DE LA EXPERIENCIA:

Laboratorio N°2. Creación de una Base de Datos en modo gráfico y por consola

B. TEMAS:

1.1 Creación de Base de Datos
- 1.1.1 Usando el Modo gráfico
- 1.1.2 Usando línea de comandos
- 1.1.3 Introducción a la sentencia Create (DDL)

C. OBJETIVO(S):

- Dominar la creación de base de datos en el ambiente del sistema de gestión de base de datos instalado previamente.
- Conocer los mecanismos gráficos y de sentencias para la creación de bases de datos a través del uso de ambos métodos en un SGBD.

D. METODOLOGÍA:

Utilizar los recursos provistos en la guía del laboratorio para el desarrollo de los entregables finales.

Debe tener la precaución de documentar el proceso que se presenta a través de capturas de pantalla y que debe colocar en la sección de resultados de este documento.

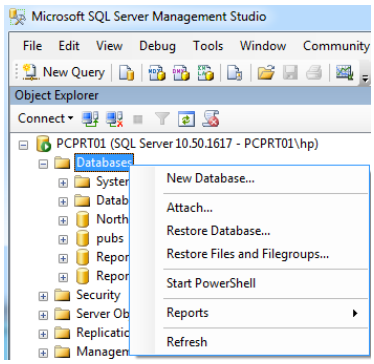
Trabajar de forma individual para lograr el desarrollo completo del laboratorio, sin embargo, pueden usar sus grupos de trabajo para el desarrollo de este laboratorio, tomando en cuenta que puede que algunos no hayan completado el proceso de instalación del SGBD.

El entregable de este laboratorio será este mismo documento con las evidencias respectivas a modo de diferentes capturas de pantalla.

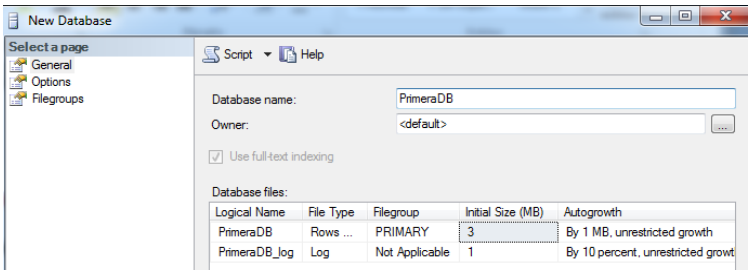
E. PROCEDIMIENTO O ENUNCIADO DE LA EXPERIENCIA:

A. CREACIÓN DE BASE DE DATOS

A.1 A través del modo gráfico (primera BD):



- Seleccione **Databases** en el explorador
- Pulsar botón derecho y desplegar menú
- Tomar **New Database** (Nueva Base de Datos)
- Se despliega una nueva pantalla
- Colocar en esta el nombre que le daremos a nuestra Base de Datos; en este caso **PrimeraBD.**



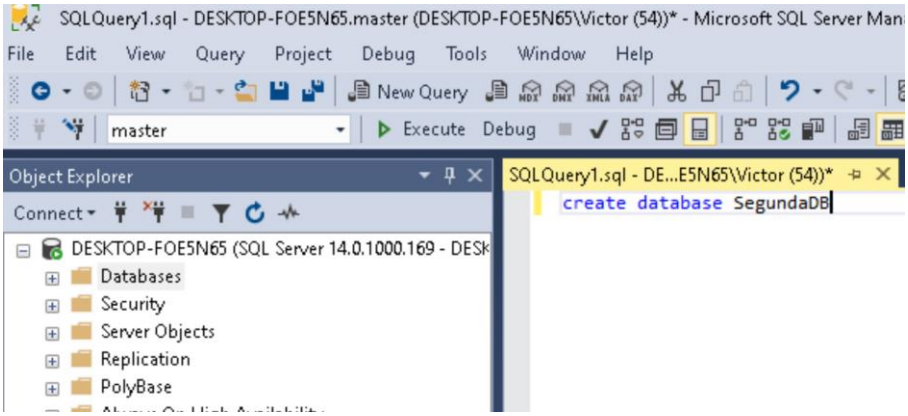
- Note que se crean automáticamente dos archivos de base de datos con el mismo nombre. En el primero se guardarán todos los datos y el segundo (el log), almacenará todas las operaciones que realice sobre la base de datos y que no han sido guardadas. Esto permite hacer rollback o garantiza la recuperación de información en caso por ejemplo de alguna falla eléctrica.
- Estos archivos tienen definido un tamaño que puede alterar de acuerdo con el tamaño que usted estime ocuparán los datos que almacenarán. También encontrará la ruta donde se creará y almacenarán dichos archivos.
- Tome la opción **ACEPTAR.** Note que ahora su base de datos debe aparecer listada en Databases (puede usar la opción de actualizar en caso de que no aparezca).

A.2 A través de sentencias SQL (Segunda BD)

Escriba la siguiente sentencia en el analizador de consulta:

```
create database SegundaDB
```

Donde:
Create especifica que está creando un objeto usando el DDL
Database indica que se crea una Base de datos
SegundaBD es el nombre de la Base de Datos



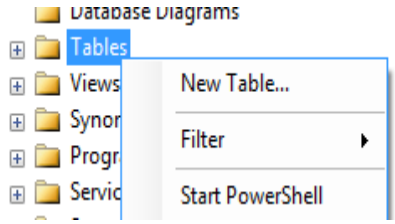
B. CREACIÓN DE LAS TABLAS

Se crearán las tablas Departamento y Empleado. A cada taba se le asignan los atributos o columnas respectivos y se indica cuál de los atributos es la llave principal. Estos conceptos serán profundizados en el curso.

Departamento (No_depto, Nombre_depto)
PK

Empleado (id_empleado, nombre, Depto_labora)
PK **FK**

B.1 A través del modo gráfico en Base de datos Primera DB:

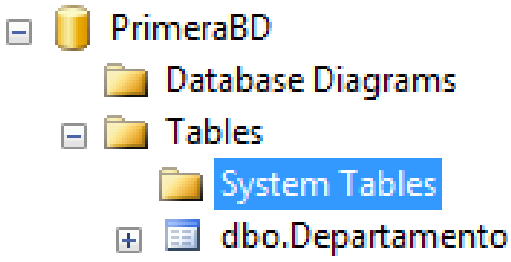


Creación de Tabla Departamento que sólo tiene llave primaria:

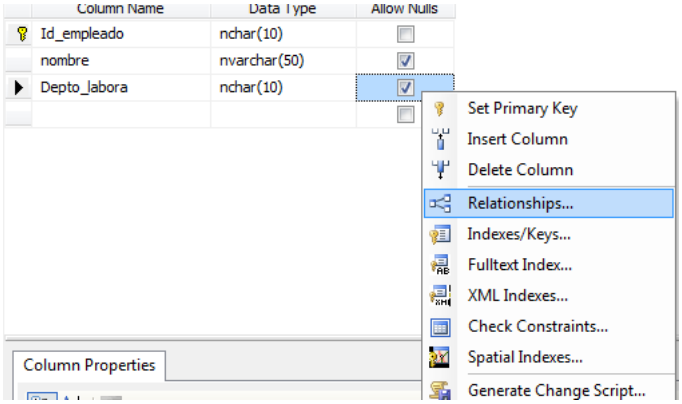
- Abrir la base de datos PrimeraDB, y dé clic sobre **Tables (Tablas)**.
- Pulsar botón derecho y desplegar menú.
- Seleccionar **New Table (Nueva tabla)**
- Se despliega una nueva pantalla que le permite insertar las columnas, los tipos de datos y definir los tipos de llave (posicionarse sobre el campo y botón derecho).
- Vamos a crear la tabla DEPARTAMENTO, para ello defina No_depto como llave primaria, tomando la opción de Primary Key que aparece al desplegar menú con botón derecho. Deje sin activar la columna **Allow null** (permir valores nulos); pues NO queremos que permita nulos.

Nombre de columna	Tipo de datos	Permitir valores NULL
No_depto	nchar(10)	☐
Nombre_depto	nvarchar(50)	☐
		☐

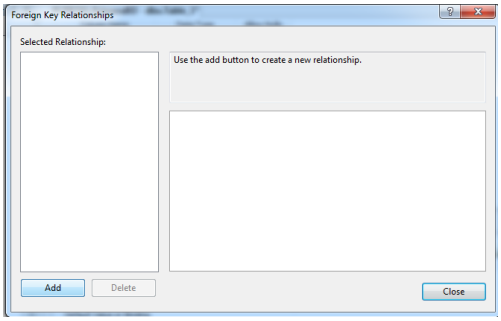
- Cierre la ventana; indique que quiere salvarla como parte de la base de datos llamada PrimeraBD (Sí / Yes), dé nombre a la tabla y salve la tabla con el nombre **Departamento.** (note que se creó esta tabla como parte de la BD PrimeraBD)

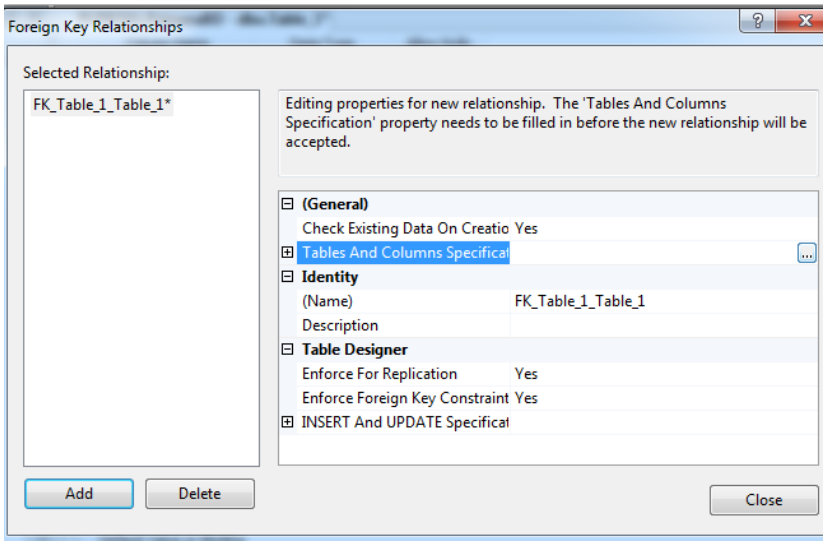


Creación de Tabla Empleado que tiene su llave primaria y un FK (foreign key – llave foránea o secundaria) que hace referencia a una llave PK (llave primaria) de la tabla Departamento.

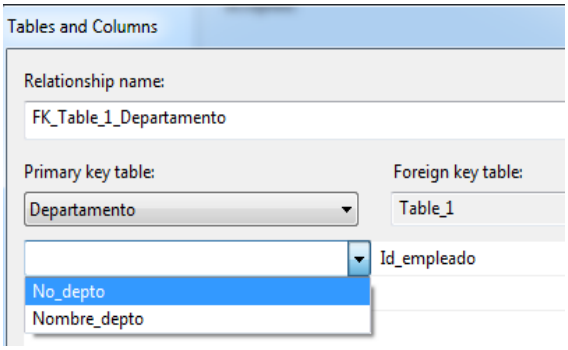


- Ahora creará la tabla empleado, y defina ID_empleado como llave primaria (pulse botón derecho, seleccione primary key).
- Defina **nombre** con su tipo de dato y not null.
- El siguiente campo es muy importante, ya que este es una llave foránea. Por lo tanto, debe corresponderse en tipo y cantidad con la definida en la tabla padre como llave primaria. Una vez definido su tipo y largo, pulse botón derecho y seleccione Relationships (Relaciones).
- Aparece la siguiente pantalla. Tome en esta la opción **Add.**
- Se despliega esta información sobre la pantalla, tome los puntos suspensivos que aparecen al hacer clic sobre **Tables and Columns Specifications**

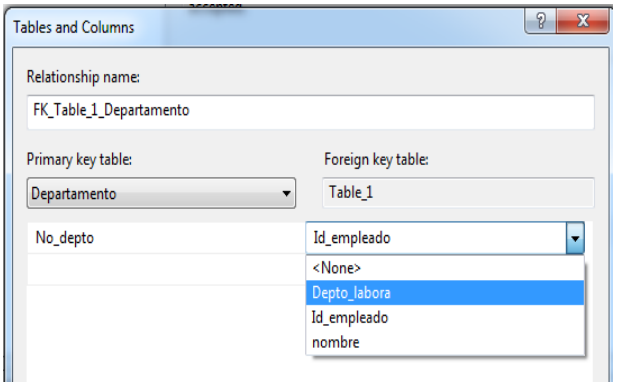




- Seleccione (a la izquierda de esta pantalla) la tabla donde tenemos la primary key (Departamento) con el cual se relaciona el campo que estamos creando. Automáticamente sale otro renglón abajo para definir qué atributo de esta tabla es el que deseamos establecer como enlace. Seleccionamos No_depto.



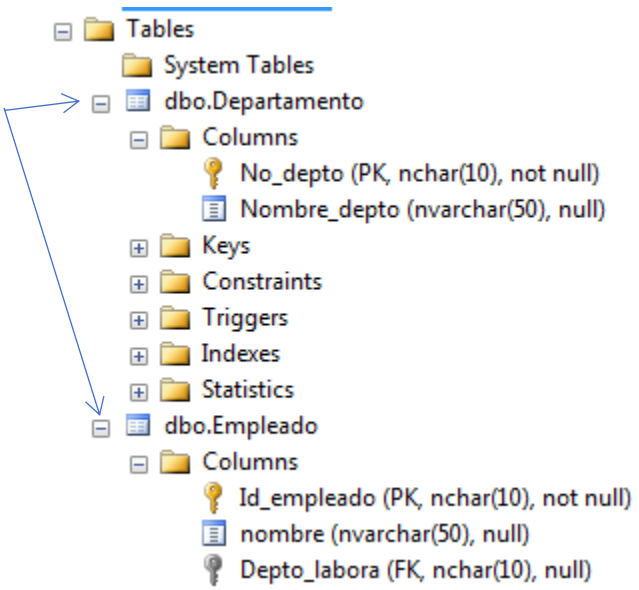
- Seleccione el campo de la tabla donde va a ser foreing Key, es decir (Depto_labora). La tabla Empleado hasta este momento no le hemos dado el nombre por lo que es Table_1 (nombre puesto por el sistema).



- Pulse Ok y Close.

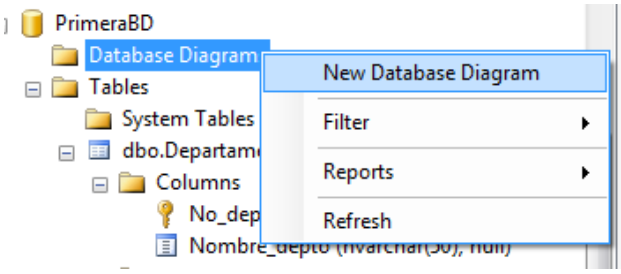
- Ahora salve la tabla cerrando la pantalla de creación (X), sale el mensaje si quiere que sea parte de **PrimeraDB**, diga Sí / Yes y luego le coloca el nombre Empleado y salvamos (YES y YES). Este último Yes relacionó las dos tablas, por lo que están conectadas a través de un atributo en común que es del mismo tipo.

- Note que se tiene dos tablas creadas. Observe que, la tabla departamento tiene una llave primaria: No_depto, en tanto que la Tabla Empleado, tiene una llave primaria Id_empleado y como llave foránea al Depto_labora

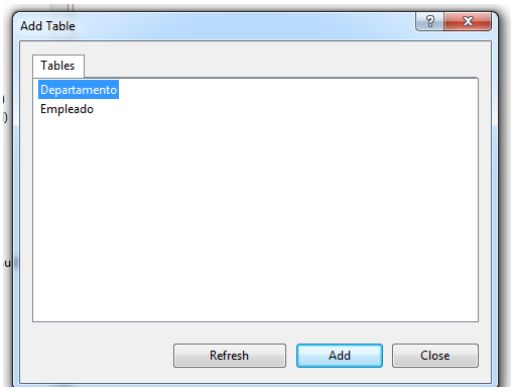


Cómo ver el Diagrama Relacional de lo que hemos creado

- Para poder ver el diagrama Relacional de lo que ha realizado, se debe ubicar sobre Database Diagram (Diagrama de Base de Datos), pulsando el botón derecho, tomando New Database Diagram (Nuevo diagrama de BD).



- Aparece la siguiente pantalla con el nombre de las tablas que ya creó.



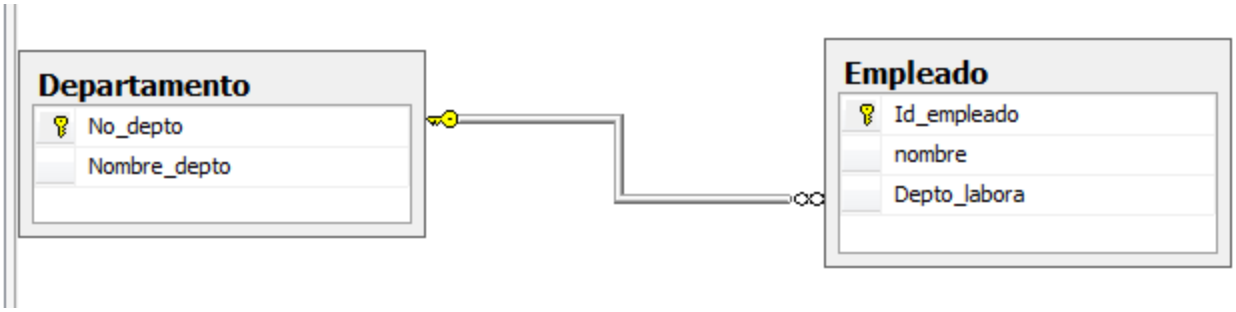
- Pulse Add - Agregar e irá adicionándolas una a una al área de trabajo.

- Al final cierre y arrastre su Modelo relacional al lugar donde lo vea mejor. Deben aparecer las relaciones de llaves primarias y foráneas definidas. Usted puede mover, quebrar las líneas de conexión según lo desee.

- Del lado de la Tabla Padre aparece una llave pequeña indicando la llave primaria. Del lado de la llave foránea (tabla hijo) le sale un símbolo de infinito o cadena. Esto denota que la llave primaria de la tabla Departamento, se relaciona con un Fk (llave foránea) en Empleado.

- Estos conceptos son abarcados con detenimiento en este curso en el capítulo de diseño de una base de datos.

- El diagrama de la base de datos puede aparecer como sigue:



Formato General para Creación de Tablas

```
CREATE TABLE    NombreTabla
(
    Nombre_columna1  tipoDato    [PRIMARY KEY] [NOT NULL]
          [UNIQUE][DEFAULT    defaultOption]

          [REFERENCES ParentTableName] [CHECK searchCondition] [,...]
          [PRIMARY KEY (listOfColumns),]
          [UNIQUE (listOfColumns),] […,]
          [FOREIGN KEY (listOfFKColumns) REFERENCES ParentTableName
                [(listOfCKColumns)],
           [CHECK (searchCondition)] [,…]  ,
    Nombre_columna2….
)
```

Las restricciones usan algunas sentencias para verificar que se cumplen con los formatos. Algunas de estas sentencias son not null, unique, primary key, foreign key, check y default. Se pueden hacer a nivel de columnas o a nivel de tablas.

- A nivel de columnas: Hace referencia a una columna y se define dentro de la especificación de la misma

  Nombre_columna [CONSTRAINT nombre_restricción] tipo_restricción

- A nivel de Tablas: Hace referencia a una o más columnas y es definido separadamente de la definición de las columnas de la tabla. Se puede definir cualquier restricción excepto NOT NULL. (ver anexo al final para mayor detalle)

NOTA: Se recomienda que los nombres de los constraint sean:

NOMBRE-DE-TABLA_ NOMBRE-CAMPO_TIPO-RESTRICCION

NN, PK,FK

Procederemos a crear estas tablas en la BD SegundaDB

Departamento1 (**Num_departamento**, Nombre_departamento)
PK
Empleado1 (**Identif_Empl**; Nombre_Emp, **Departamento_lab**)
PK **FK**

Creación de tabla Departamento1.

A nivel de columnas: use el siguiente script o sentencias para colocarlos en el analizador de consultas:

```
use SegundaDB
create table Departamento1
(
  Num_departamento  nchar(10)
       Constraint  Departamento1_num_depto_pk primary key,
  Nombre_departamanto  nchar(50) not null

)
```

Crea la tabla Departamento1 como parte de la base de datos SegundaDB. Por eso se activa con el comando **use SegundaDB**.

Así, se define la primera columna de esta tabla ***Num_departamento*** y seguido le indica a través del **constraint,** que la llave primaria es el atributo Num_departamento, luego defina la siguiente columna con su tipo de dato e indique que no debe ser nula. (Como no ha colocado nombre a esta restricción not null, en caso de querer o necesitar cambiarla, tendrá que ver que nombre le puso el sistema.)

Si la llave primaria es compuesta, esta no puede ser definida a nivel de columna; por lo que es recomendable hacerlo una vez definidas todas las columnas de la tabla, para que la restricción sea a nivel de tabla y poder colocar las columnas que necesite, ya que han sido definidas previamente (esto se profundiza en el capítulo respectivo).

A nivel de Tablas. No es necesario que se ejecute el siguiente script porque ya tiene creada la tabla Departamento1.

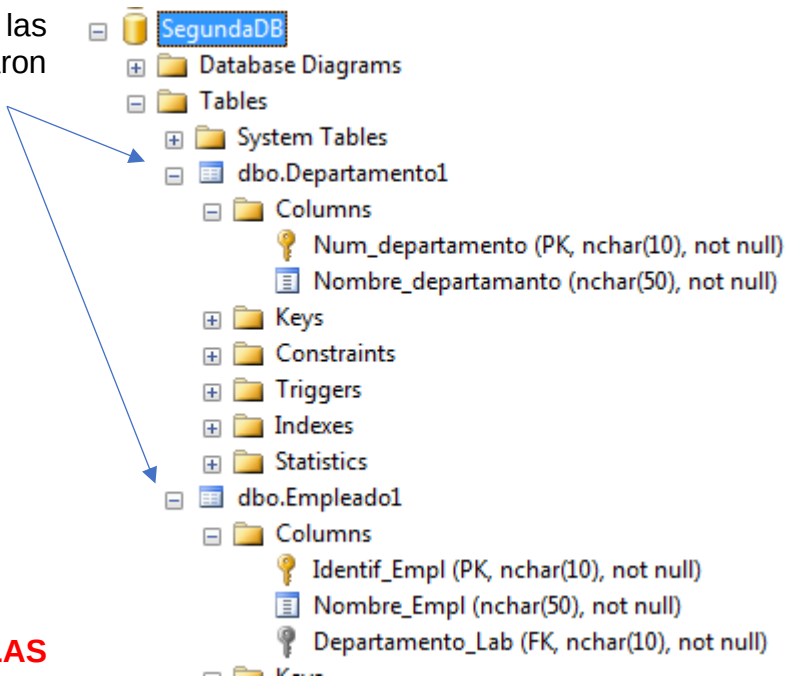
```
use SegundaDB
create table Departamento1
(
   Num_departamento  nchar(10),
   Nombre_departamento  nchar(50) not null,
      Constraint  Departamento1_Nombre_departamento_pk
            primary key  (Num_departamento),
)
```

Creación de tabla Empleado1.

OBSERVE; Esta es otra forma de poder definir las restricciones, sin definir nombre a las restriccciones (constraint); pero no es lo recomendado, sin embargo lo puede encontrar de esta forma.

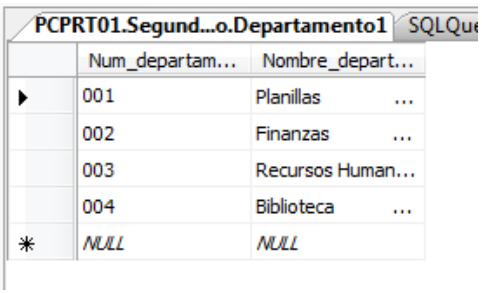
```
use SegundaDB
create table Empleado1
(
   Identif_Empl  nchar(10),
   Nombre_Empl  nchar(50) not null,
   Departamento_Lab nchar(10) not null ,
         primary key (Identif_Empl),
         foreign key (Departamento_Lab)
               References Departamento1 (Num_departamento)
   )
```

- Note las tablas creadas y las restricciones con las que se crearon las mismas,



C. INSERTANDO VALORES A LAS TABLAS

C.1 A través del modo gráfico



A través del modo gráfico es sumamente sencillo insertar valores a las tablas.

- Solo hay que ubicarse sobre la tabla a la que desea insertar datos,
- Botón derecho
- Tomar Edit top 200 row (editar las primeras 200 filas)
- Aparece una tabla para ir adicionando valores a las columnas.
- **Inserte algunos valores usando esta opción.**

C.2 A través de sentencias SQL

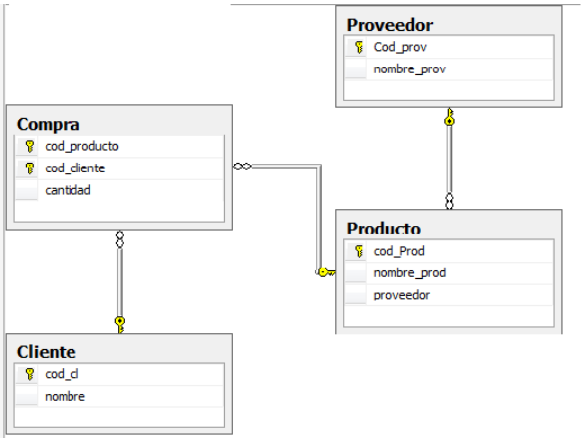
```
insert into NombreTabla
      values (valorcolum1,valorcolum2….,valorcolumN)
```

```
insert into Departamento1
  values ('005','UTP')
```

Observación: Se colocan entre comilla simple porque son tipo Char

D. Ahora se crea una tabla con llave compuesta, en donde cada parte de la llave compuesta es a su vez un foreign key

D.1. Creacion. Para ello use el siguiente script para la creación de la base de datos respectiva y sus correspondientes tablas.




```
Create Database  VentaDB

use  VENTADB

CREATE TABLE Cliente
(  cod_cl  nchar(4)
     Constraint Cliente_cod_cl_pk primary
key,
   nombre nvarchar(30) not null

)

CREATE TABLE Proveedor
(  Cod_prov  char(5)
     Constraint Proveedor_cod_prov_pk  primary key,
   nombre_prov varchar(30) not null

)

CREATE TABLE Producto
(  cod_Prod  char(4)
     Constraint Producto_cod_prod_pk  primary key,
   nombre_prod  varchar(20),
   proveedor char(5),
    constraint Producto_cod_prod_fk  foreign key (proveedor)
       references Proveedor(Cod_prov)
)
```

Tabla donde es llave primaria y el atributo Llave Primaria

Atributo que es foreign key en la tabla que estamos creando

```
create table Compra
  (
    cod_producto  char(4),
    cod_cliente  nchar(4),
```

```
    cantidad  smallint,
     constraint Compra_cod_producto_cod_cliente_PK
         primary key (cod_producto, cod_cliente),
     constraint Compra_cod_producto_fk
         foreign key (cod_producto) references Producto (cod_Prod),
     constraint Compra_cod_cliente_fK
         foreign key (cod_cliente) references Cliente(cod_cl)
  )
```

D.2. Eliminación.

Para Eliminar una tabla creada se usa la sentencia DROP. En el entorno de una Base de Datos es importante diferencias entre DROP y DELETE. A medida que avance, conocerá las diferencias. Drop permite eliminar la estructura completa. Si usa Drop junto al nombre de una base de datos, la está eliminando por completo.

Drop table nombreTabla

Ejemplo: Drop table Cliente *--eliminará la tabla cliente*

Usando Comentarios

Existen dos formas de realizar comentarios en SQL Server:
- - Esto es un comentario que cabe en una línea y su identificador son dos guiones seguidos

/* Esto es un comentario que abarca más de una línea.
Inicia con una barra y luego un asterisco y termina con un asterisco y luego una barra tal y como se muestra en este ejemplo.
*/

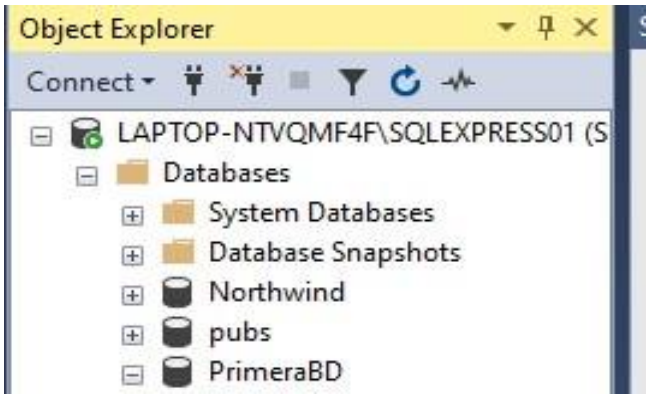
F. RECURSOS:

- Computador con acceso a internet, acceso a plataforma ecampus.utp.ac.pa/moodle, curso Tecnología de Base de Datos.
- MS SQL Server
- Guía de laboratorio

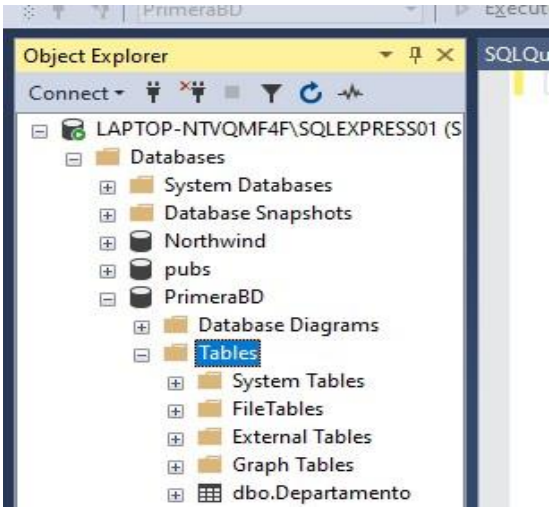
G. RESULTADOS:

Desarrollo del Laboratorio #2

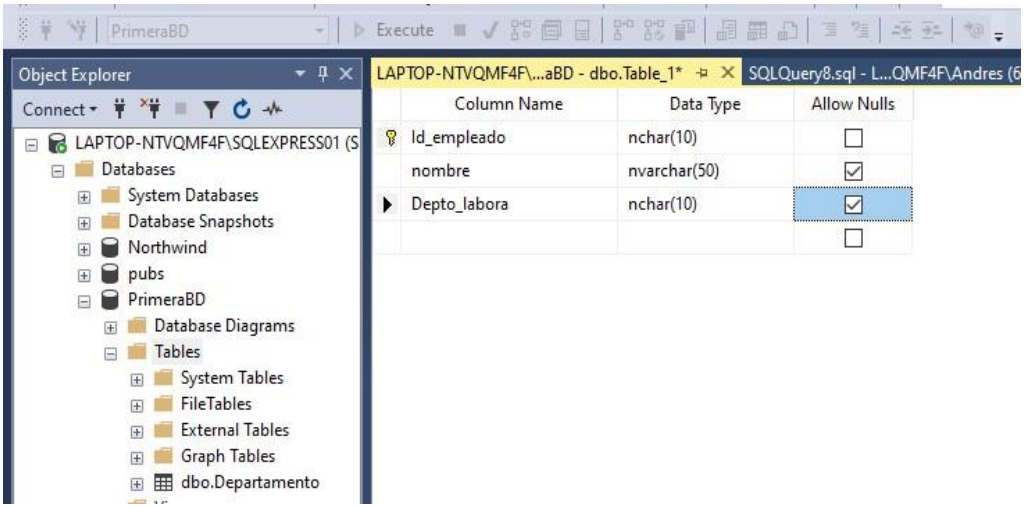
1. A través del Modo Gráfico, procederemos a crear una nueva Base de Datos, a la cual le daremos el nombre de PrimeraBD



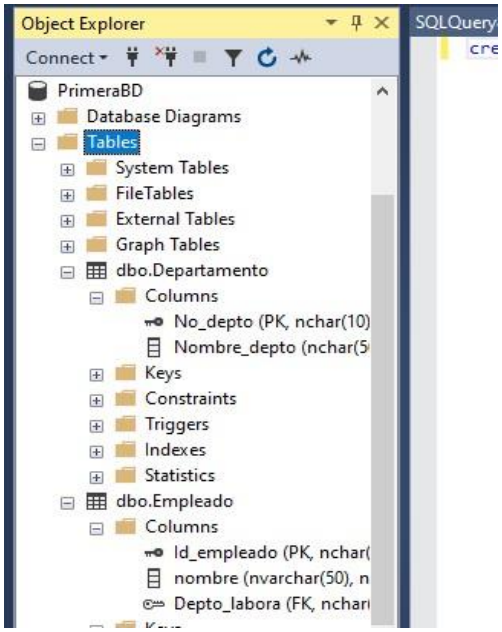
2. Ahora crearemos la primera tabla de la Base de Datos a la cual vamos a denominar Departamento, en la que nos encontramos a No_depto como la llave primaria de la misma.



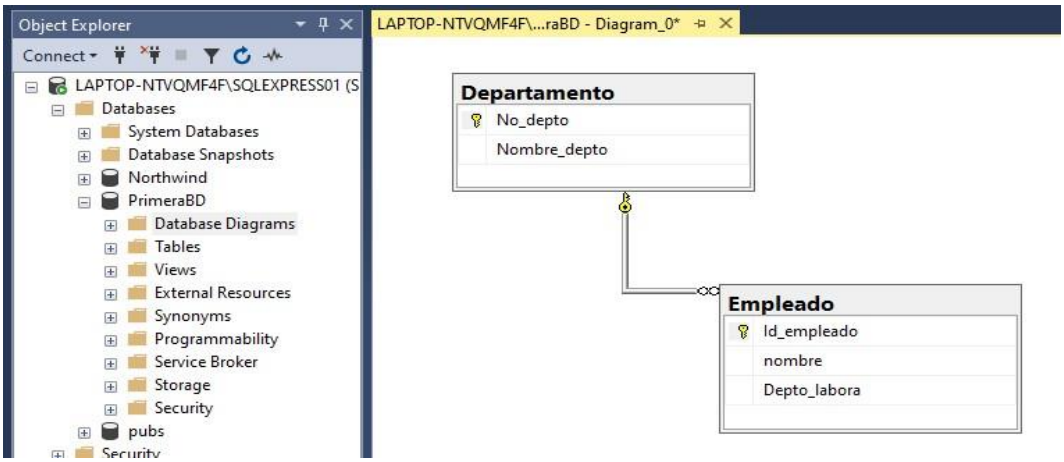
3. Posterior a ello, crearemos una nueva tabla, únicamente que ahora se llamará Empleado, donde utilizaremos la Función Relationships, para establecer una comunicación (Relación) con la Tabla de Departamento dentro de nuestro esquema.



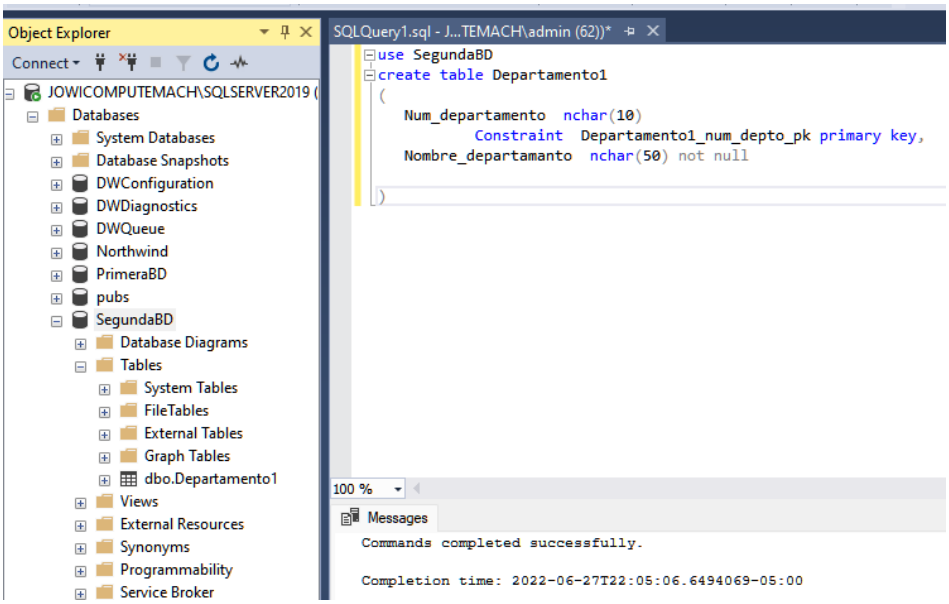
4. Ahora, una vez hemos asignado las Relationships que hay dentro de nuestro esquema de Base de Datos Relacional, podemos encontrar a la Tabla Departamento y la Tabla Empleado con sus respectivas columnas.



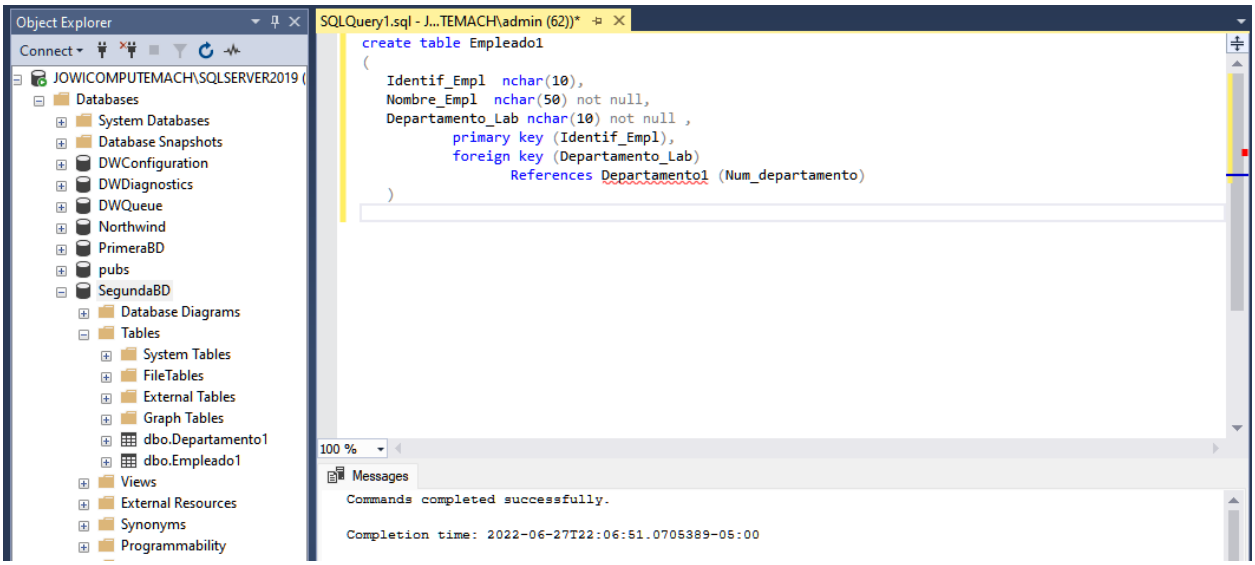
5. Una función sumamente interesante de SQL Server Management Studio es que puede crearnos el Diagrama Relacional que existe en nuestro Modelo de Bases de Datos, de manera que podamos observar la creación de las tablas, sus respectivas columnas y las relaciones de llaves primarias, secundarias o compuestas que existen entre ellas, como es el caso que se muestra en este ejemplo entre la tabla Departamento y tabla Empleado.



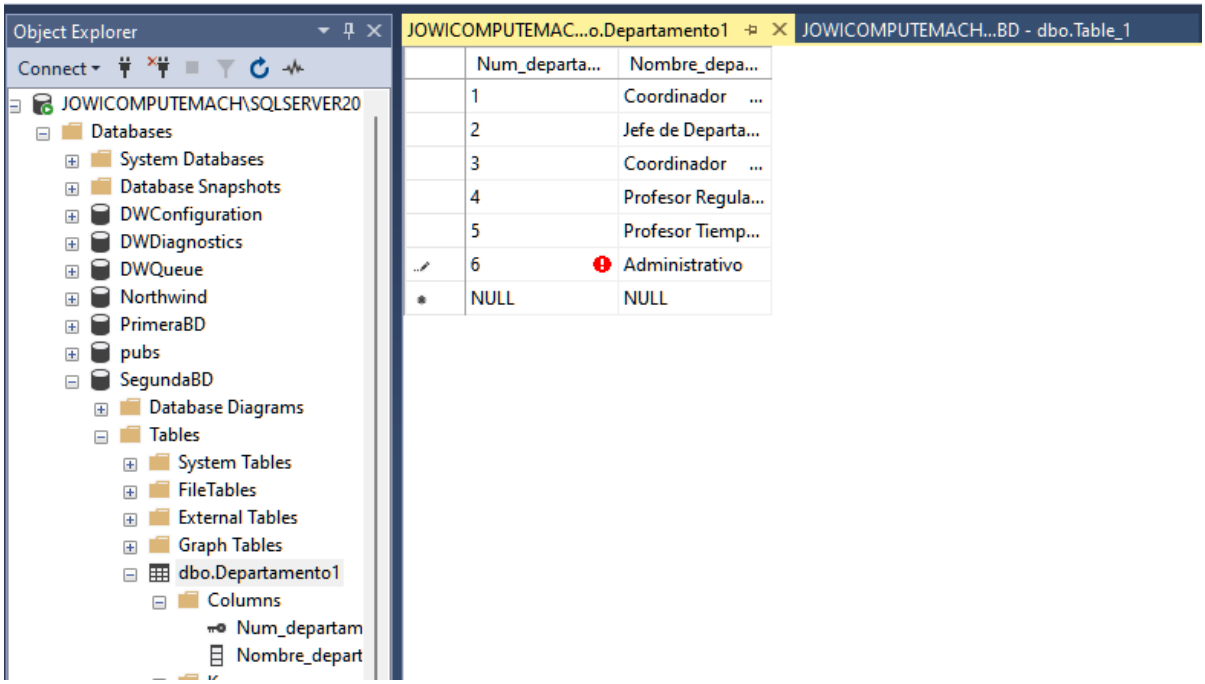
6. Para el desarrollo de nuestra segunda base de datos, a la cual vamos a llamar SegundaBD, usaremos el lenguaje SQL, es decir que será creada, tanto ella como sus tablas, columnas, llaves primarias y secundarias a través de línea de comandos, lo cual, en cierta medida con la técnica del programador, se vuelve de muchísima ayuda para desarrollar posteriores consultas de manera rápida.



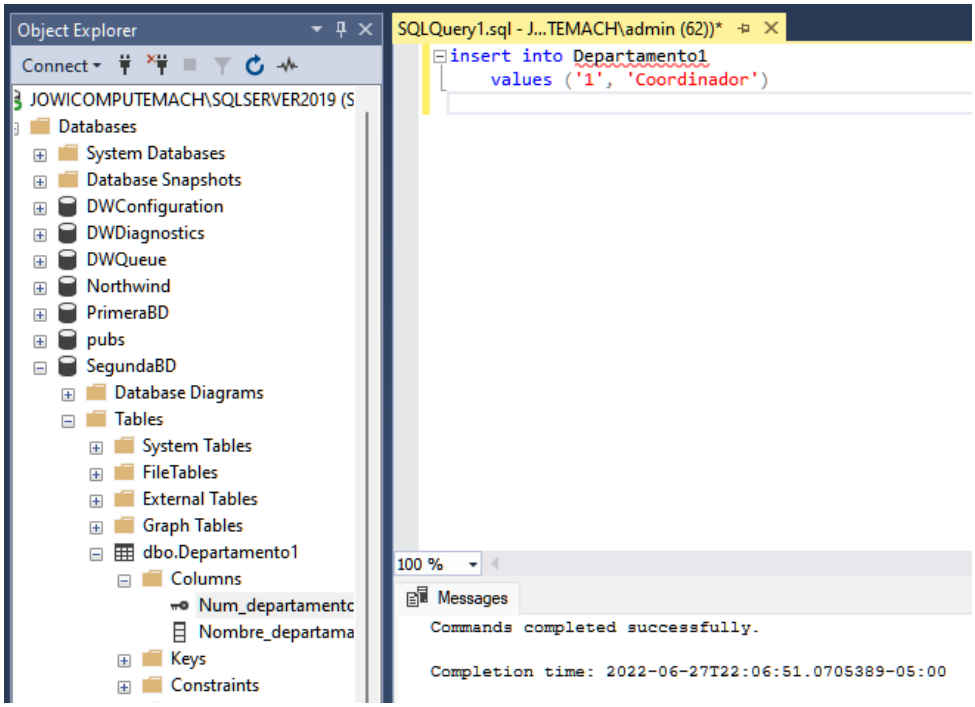
7. Demostrando que existen múltiples maneras de crear una tabla en SQL Server Management Studio, procederemos a crear la tabla Empleado1 a través del uso de sentencias SQL en líneas de comando, incluyendo sus respectivas restricciones.



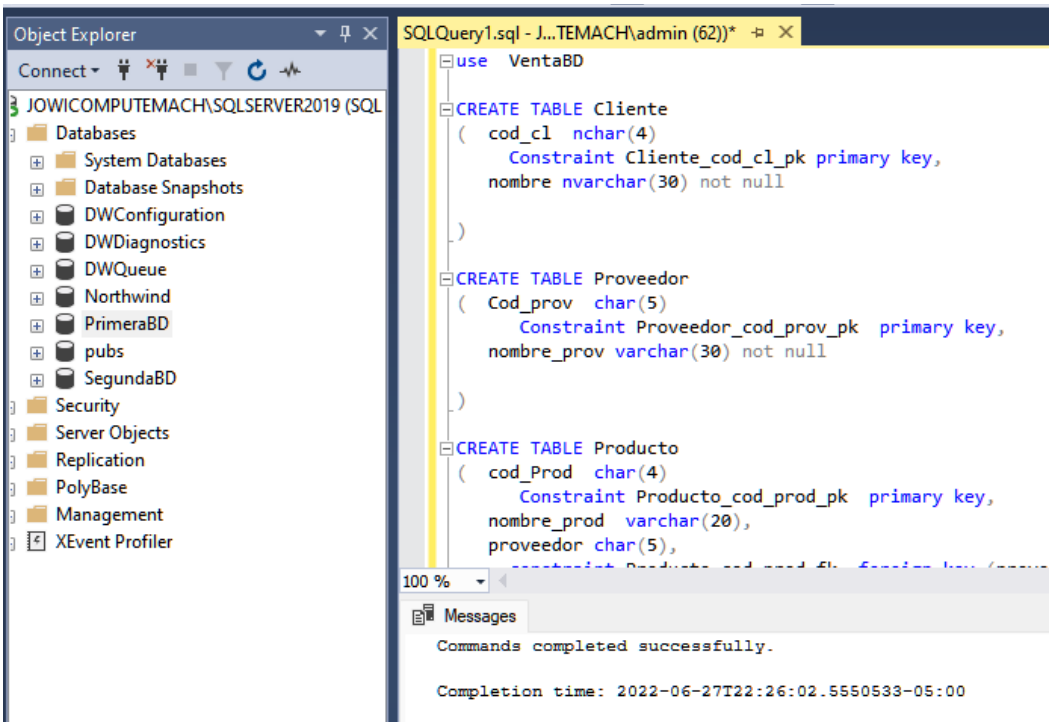
8. A continuación, tomaremos la Tabla Departamento1 e insertaremos valores a través del Modo Gráfico para demostrar la dualidad de SGBD.



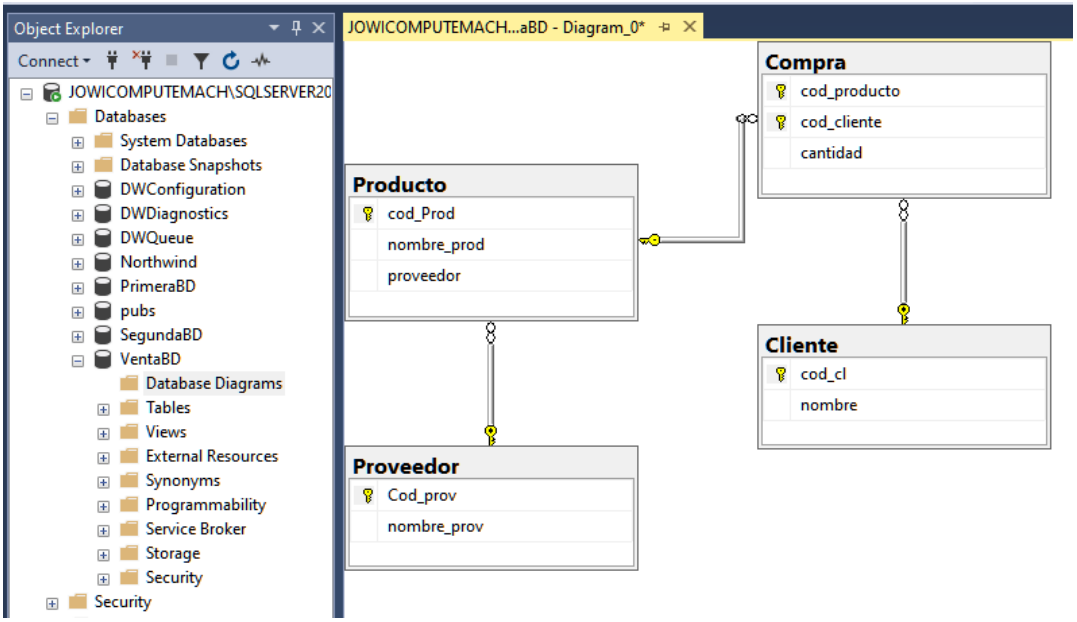
9. Ahora procederemos a realizar la inserción de datos en las tablas, únicamente que utilizando sentencias SQL, es decir a través de línea de comandos.



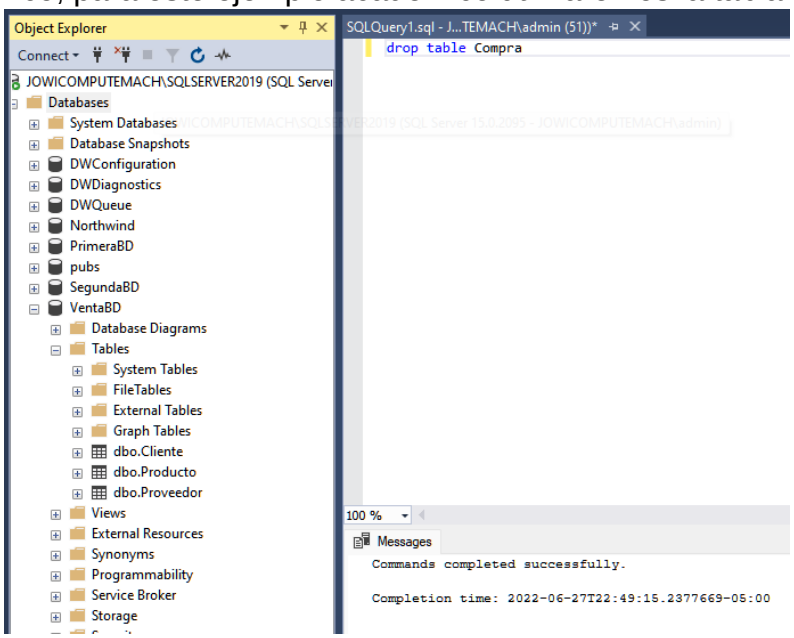
10. Procederemos a crear una tabla en la que se pueda contar con una llave compuesta, donde tengamos que cada parte de la llave compuesta, a su vez es una llave foránea (Foreign Key, FK) correspondiente a otra tabla. Todo esto será realizado a través del uso de sentencias SQL.



11. A continuación, nuevamente utilizaremos la función Database Diagrams, para crear nuestro Diagrama del Modelo Relacional de la Base de Datos que recién creamos, llamada VentaBD.



12. Lo último que haremos en este laboratorio, será realizar utilizar la función Drop, la cual se encarga de borrar todos lo que se encuentre dentro de una tabla en específico, para este ejemplo académico utilizaremos la tabla Compra.



H. CONSIDERACIONES FINALES:

El aprendizaje más grande que tal vez me lleve de este laboratorio es la eficiencia que tiene SQL Server Management Studio para el desarrollo de proyectos de Bases de Datos, ya sea a nivel macro o a nivel micro, como es este caso con fines académicos; ya que cuenta con la capacidad de diseñarlas sin escribir una sola línea de código utilizando el “Modo Gráfico” o utilizando sentencias SQL que nos permitirán ver a fondo todo lo que se nos presente, especialmente el uso de las llaves primarias (PK) y las llaves foráneas (FK) para la interconexión entre tablas.

I. BIBLIOGRAFÍA:

- Ver la bibliografía del curso.

J. RÚBRICA:

- Este laboratorio evalúa la completitud de la actividad, con una puntuación base de 100.
- El laboratorio debe ser entregado en el tiempo estipulado para el desarrollo de este (no se aceptan informes de laboratorio enviados al correo electrónico).
- Debe contener todos los elementos que se solicitan en la forma descrita el inicio de la guía de laboratorio.
- Uso de la sección de resultados de la experiencia.