Laboratorio #3

Definición de SQL DDL y DML

i

¿Qué es SQL y por qué es importante conocerlo?

SQL (Structured Query Language o Lenguaje de Consulta Estructurado) es un lenguaje de programación estándar utilizado para la manipulación de bases de datos relacionales, que permite realizar diversos tipos de operaciones en ellas. En este laboratorio aprenderá DDL (Lenguaje de Definición de Datos) y DML (Lenguaje de Manipulación de Datos).

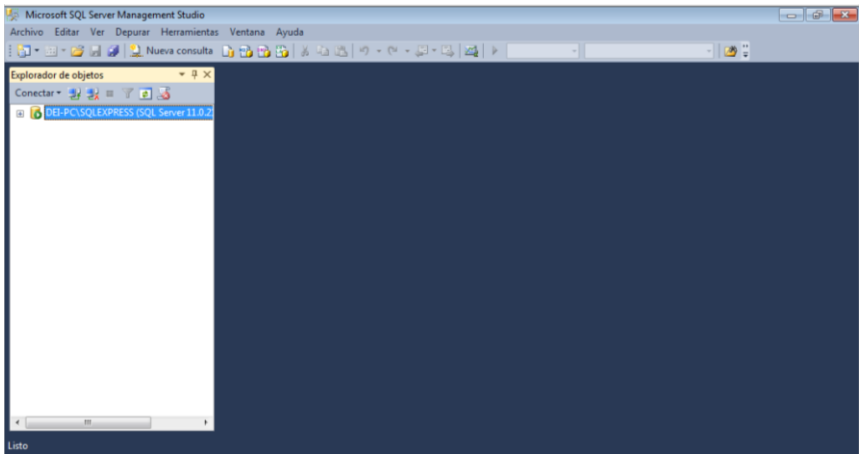
Conozcamos el IDE

SQL Server Management Studio (SSMS) es un ambiente de trabajo para estructuras SQL, provee de herramientas para configurar, monitorear y administrar bases de datos.

Encienda la máquina virtual y busque este icono en el escritorio.

Al entrar en la aplicación, aparecerá esta pantalla, dar al botón “Conectar”.

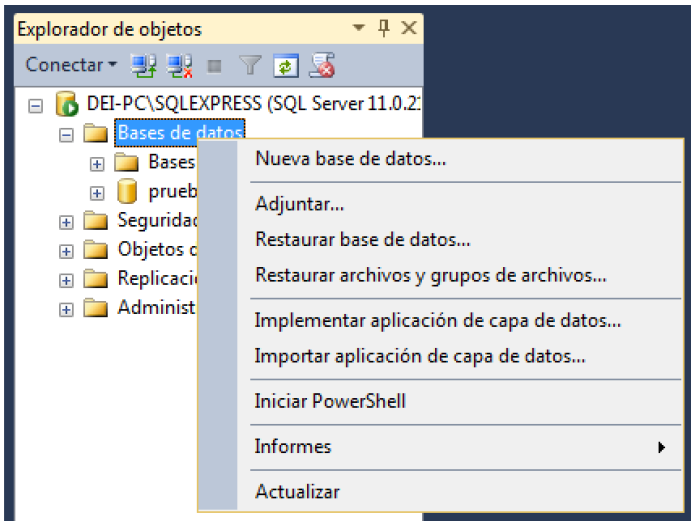
Lo que verá es una pantalla así:



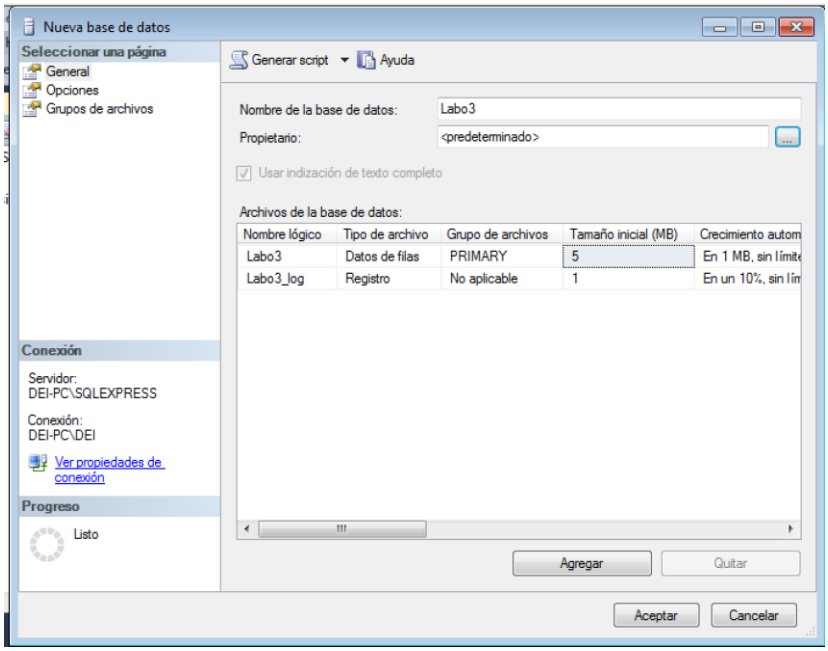
En los laboratorios anteriores se han realizado los proceso iniciales de creación de una base de datos, el siguiente paso después de tener el diagrama totalmente normalizado, es el de crear la base datos, junto a todas sus tablas y definir sus atributos.

Creación de nueva base de datos:

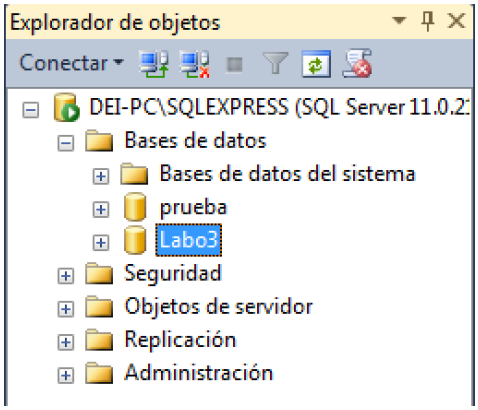
Hay dos opciones para crear una nueva base, la primera es solo con el ratón: desplegar la instancia creada al iniciar la aplicación y buscar la carpeta Bases de datos, dar con el botón derecho sobre la carpeta y elegir la opción “Nueva base de datos”.



Desplegará otra pantalla en la que requerirá de un nombre para la nueva base, puede ponerle el nombre que desee, en esta ocasión se nombra “Labo3”, luego dar click al botón “Aceptar”, por el momento los parámetros de configuración de esta ventana se dejarán con sus valores por defecto.



En el panel de navegación lateral debería aparecer la base de datos recién creada:



La otra opción para crear una base es utilizando sentencias DDL SQL.

DDL

Las sentencias DDL se utilizan para crear, modificar la estructura, de las tablas y objetos de la base de datos. Por ejemplo:

Sentencia	Definición	Ejemplos de uso
CREATE	Utilizada para crear objetos en la base de datos	CREATE DATABASE <db_name> CREATE TABLE <table_name>
ALTER	Es utilizada para agregar, borrar o modificar columnas en una tabla existente	ALTER TABLE <name> ADD COLUMN <name and type>
RENAME	Utilizado para cambiar el nombre a una tabla	RENAME TABLE <old name> TO <modified name>
DROP	Borra objetos de la base de datos	DROP DATABASE <db_name> DROP TABLE <table_name>
TRUNCATE	Elimina todos los registros de la tabla, incluyendo los espacios asignados a los registros	TRUNCATE <table_name>

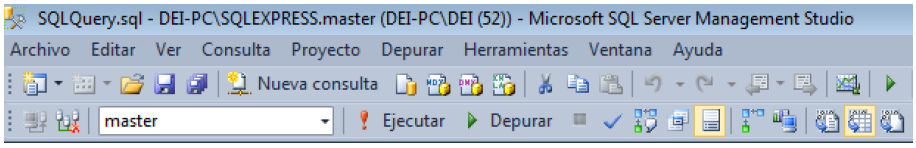
DML

Las sentencias DML son utilizadas para gestionar datos dentro de las tablas. Por ejemplo:

Sentencia	Definición	Ejemplos de uso
SELECT	Se utiliza para seleccionar datos de la base de datos	SELECT <field_came> FROM <table_name>
INSERT	Es utilizada para agregar nuevos registros en una tabla	INSERT INTO <table_name> (<column>) VALUES (<data>)
UPDATE	Utilizado para modificar un dato dentro de una tabla	UPDATE <table_name> SET <column> = <new_data> WHERE <condition>
DELETE	Elimina un registro de la base de datos	DELETE FROM <table_name> WHERE <condition>

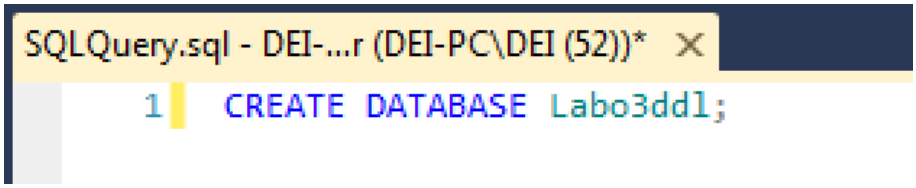
Los pasos para crear la base con comandos son los siguientes:

Crear un nuevo archivo de consultas, dando click al siguiente icono

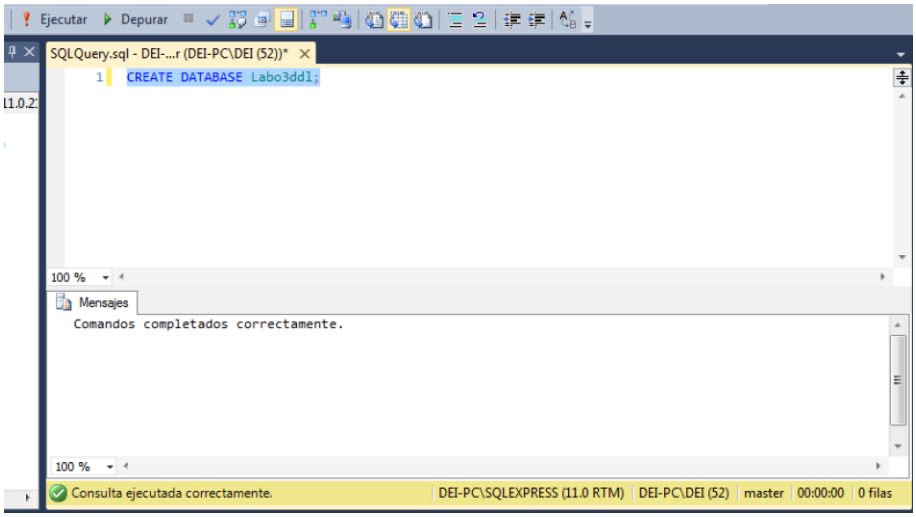


Preguntará a que servidor queremos relacionar la consulta, se utiliza el servidor por defecto y dar click al botón “Conectar”

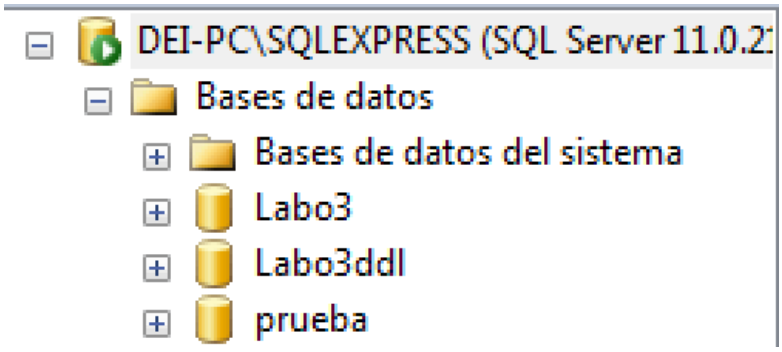
Tendremos una página en blanco en la que escribiremos las sentencias para la creación de la nueva Base de Datos. Se utiliza el comando CREATE



Sombrear el comando CREATE y en la barra superior, buscar el botón con un signo de admiración que dice “Ejecutar”, presionarlo y esperar el mensaje “Comandos completados correctamente” en la ventana Mensajes

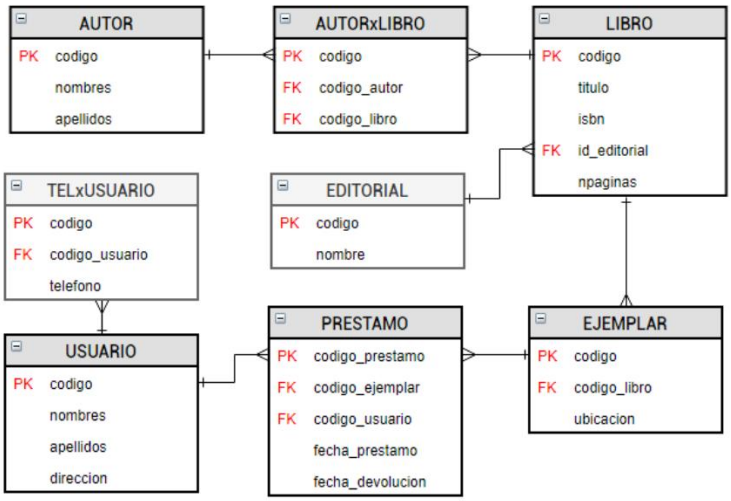


Presionar botón derecho sobre el servidor activo (Con el icono de play verde) y buscar la opción “Actualizar” en el menú, debería aparecer la nueva base de datos en la lista disponible.

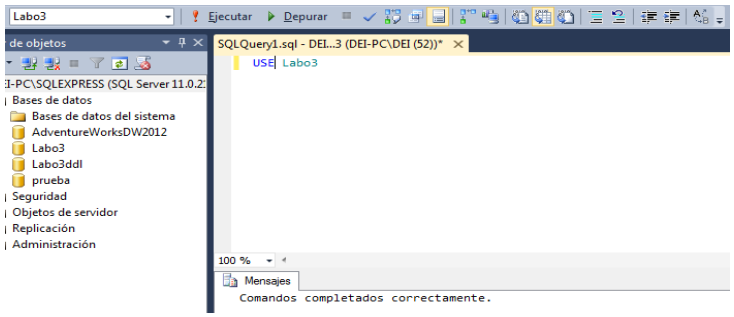


¡Listo! la base de datos ha sido creada.

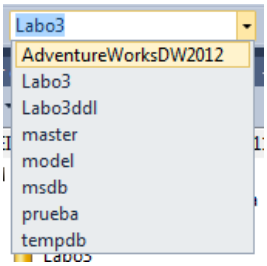
Se procede a crear las tablas y definir los atributos para cada una, se utilizará el siguiente diagrama relacional normalizado como ejemplo:



Antes de empezar con la creación de todos los objetos que utilizaremos, debemos indicarle al IDE que usaremos nuestra base de datos Labo3 con la instrucción USE:



O bien podemos usarlo por el indicador del IDE:



Creación de objetos de bases de datos

Tablas

Utilizaremos el comando CREATE TABLE donde colocaremos los campos a utilizar.

Ejemplo:

```
CREATE TABLE LIBRO(
    codigo int primary key not null,
    titulo varchar(50),
    isbn varchar(30),
    npaginas int,
    id_editorial int
);

CREATE TABLE EDITORIAL(
    codigo int primary key not null,
    nombre varchar(50)
);
```

Note que podemos colocar la llave primaria desde la creación de la tabla usando las palabras reservadas “primary key” en el atributo clave, y colocamos también la condición *‘not null’*. (En clase se han visto dos formas más para configurar las llaves primarias ¿Cuales son?)

Crearemos todas las tablas necesarias para nuestro ejemplo de la biblioteca UCA.

Si tienes consultas de los tipos de datos, ve al **apéndice 1.**

Llaves Primarias y Foráneas

Las llaves primarias son el atributo clave de la tabla, sin ella, los datos estarían desordenados y sin un índice que los identifique, de esto se puede asumir que la llave primaria **no puede ser nula**. Para crear llaves primarias desde la creación de la tabla se usará el siguiente ejemplo:

```
CREATE TABLE AUTORXLIBRO (
    codigo int identity primary key not null,
    codigo_autor int,
    codigo_libro int
)
```

Las llaves foráneas son importantes para la relación de tablas en la base de datos ya que nos permitirán hacer consultas cruzando tablas y obteniendo campos de ambas partes. En el siguiente ejemplo veremos la creación de las llaves foráneas desde la creación de la tabla, pero es recomendable hacerlo en un comando aparte, con un CONSTRAINT pues le asignamos un nombre a esta relación en caso de querer borrarla luego:

```
CREATE TABLE AUTORXLIBRO (
    codigo int identity not null,
    codigo_autor int,
    codigo_libro int FOREIGN KEY (codigo_libro) REFERENCES LIBRO (codigo)
)
```

NOTA: No es necesario que los campos referencia y referenciado tengan el mismo nombre, pero sí es requisito que dispongan del mismo tipo de dato.

Commented [1]: Las Notas irán con el formato sugerido por el modelo del guía, cuando se cree el documento final.

ALTER TABLE

En el caso de haber cometido un error al crear la base de datos, olvidar algún atributo, llave primaria o tipo de dato erróneo, puede modificar las propiedades de la tabla con esta instrucción.

En el caso de haber olvidado colocar la llave primaria, se puede solventar con el siguiente ejemplo:

```
ALTER TABLE AUTORXLIBRO
ADD PRIMARY KEY (codigo)
```

También puede agregar la llave foránea después de haber creado las tablas::

```
ALTER TABLE AUTORXLIBRO
ADD CONSTRAINT FK_AUTORXLIBRO_AUTOR
FOREIGN KEY (codigo_autor) REFERENCES AUTOR (codigo);
```

Al olvidar colocar algún campo, puede ser resuelto de la siguiente forma:

```
ALTER TABLE AUTORXLIBRO
ADD campo_olvidado char(5)
```

Si se colocó el tipo de dato erróneo en una columna:

```
ALTER TABLE AUTORXLIBRO
ALTER COLUMN campo_olvidado int
```

INSERT

Ahora insertamos datos a las tablas que creamos con la sentencia INSERT y los nombres de las columnas que queremos insertar:

```
INSERT INTO LIBRO (codigo, titulo, isbn, npaginas,id_editorial)
            VALUES (1,'Asedio y Tormenta','010-1',544,002),
                   (2,'Siempre Alice','011-2',300,004),
                   (3,'Corazon de Tinta','012-3',500,005);

INSERT INTO EDITORIAL(codigo,nombre)
            VALUES  (002,'Hidra'),
                    (004,'Ediciones B'),
                    (005,'Scholastic Corporation');
```

Cuando se insertan los datos, hay que respetar los tipos, en los campos cuyo tipo es varchar se tienen que agregar los datos encerrados en comillas 'simples'.

También puedes agregar datos a tus tablas aunque no especifiques los campos, siempre y cuando insertes todos los campos en el orden en el que están guardados en la tabla. Por ejemplo, agregaremos los datos de un nuevo libro:

```
INSERT INTO LIBRO VALUES (5,'La Huesped','010-4',644,002);
```

SELECT

Para verificar que las tablas fueron modificadas, utilizamos la sentencia SELECT, en este ejemplo le decimos que nos muestre todos los datos:

```
SELECT * FROM LIBRO;
```

El resultado será el siguiente:

	codigo	titulo	isbn	npaginas	id_editorial
1	1	Asedio y Tormenta	010-1	544	2
2	2	Siempre Alice	011-2	300	4
3	3	Corazon de Tinta	012-3	500	5
4	5	La Huesped	010-4	644	2

Los SELECT también pueden mostrarnos columnas específicas, solo se sustituye el asterisco por el nombre de la columna que se quiere consultar.

```
SELECT titulo FROM LIBRO;
```

El resultado debe verse algo así:

	titulo
1	Asedio y Tormenta
2	Siempre Alice
3	Corazon de Tinta
4	La Huesped

UPDATE

Esta sentencia se utiliza para actualizar un registro de una tabla, es preferible utilizar la sentencia WHERE junto a la llave primaria para especificar qué registro queremos actualizar. Actualicemos el libro con código 2 y cambiemos el número de páginas a 800:

```
UPDATE LIBRO SET npaginas=800
        WHERE codigo=2;
```

NOTA: ¿Qué pasaría si no utilizamos el WHERE?

DELETE

Esta sentencia se utiliza para borrar un registro de una tabla, para esto necesitamos saber la llave primaria del registro a eliminar, puede utilizarse otro campo pero no es recomendado pues podemos borrar muchos datos de una vez, para este ejemplo borraremos el libro "Siempre Alice", cuya llave primaria es el número 2:

```
DELETE FROM LIBRO WHERE codigo=2;
```

Si realizamos un SELECT a la tabla LIBROS veremos que el campo ha sido eliminado.

	codigo	titulo	isbn	npaginas	id_editorial
1	1	Asedio y Tormenta	010-1	544	2
2	3	Corazon de Tinta	012-3	500	5
3	5	La Huesped	010-4	644	2

NOTA: Es importante que no olvidemos la sentencia WHERE ¿Qué pasaría si no la escribimos?

DROP

Para terminar de estudiar las sentencias, utilizaremos el comando DROP para borrar la base de datos que no estamos usando actualmente y que creamos al principio del laboratorio.

```
DROP DATABASE Labo3ddl;
```

Actualizamos el servidor activo, siguiendo los mismos pasos que se utilizaron para ver la nueva base cuando fue creada.

Ejercicio

Elegir una pareja de tablas del Diagrama Relacional Normalizado del ejemplo, escribir los comandos para: crear las tablas, crear las fk, llenarlas con al menos 3 registros cada una, realizar un update y realizar un delete.

Apéndice 1: Tipos de Datos

Hay que tener en cuenta los tipos de datos que usaremos para cada campo, así que se listaran los más utilizados para tener algunas nociones:

Numéricos:

Tipo de dato	Rango	Tamaño	Declaración
bigint	[-2^63, 2^63-1]	8 Bytes	bigint
int	[2^31, 2^31-1]	4 Bytes	int
smallint	[2^15, 2^15-1]	2 Bytes	smallint
tinyint	[0, 225]	1 Byte	tinyint
decimal o numeric	Precisión hasta 38 decimales	17 Bytes Max	decimal(p,s) numeric(p,s)

Al tipos de dato int se le puede colocar un modificador **identity** con el cual podemos hacer un autoincremento de valor. Podemos usarlo en las llaves primarias por ejemplo.

¿Qué tipo de dato utilizaría usted para la edad de una persona?

Cadenas de texto:

Tipo de dato	Rango	Declaración
char	[1, 8000]	char()
varchar	[1, 8000]	varchar()
text	[1, 2^31-1]	text

¿Cuál es la diferencia entre Char y Varchar?

Fecha:

Tipo de dato	Formato	Tamaño	Declaración
date	YYYY-MM-DD	3 Bytes	date
datetime2	YYYY-MM-DD hh:mm:ss	8 Bytes Max	datetime2

investigar que mejoras tienen DATETIME2 con respecto a DATETIME