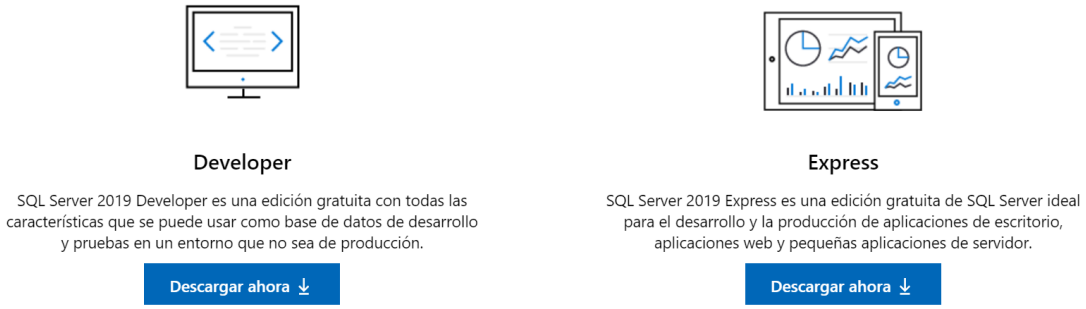
Capacitación Bases de Datos SQL Server 2019

1. Descargar la versión de la base de datos a utilizar para este caso vamos a trabajar con la versión de SQL Server 2019 Express la cual se puede descargar del siguiente Link.

   https://www.microsoft.com/es-es/sql-server/sql-server-downloads



2. Posterior mente realizaremos la descarga e instalación del SQL Managment Studio la cual pueden descargar desde la misma pagina.

   A partir de la instalación de estas herramientas iniciaremos el desarrollo de la capacitación con las siguientes temáticas.

3. Que es SQL Server y TransacSQL?

   ¿Qué es SQL? SQL Structured Query Language en español Lenguaje de consulta Estructurado, es un lenguaje específico utilizado en programación, diseñado para administrar y recuperar información de sistemas de gestión de bases de datos relacionales.

   Una de sus principales características es el manejo de álgebra y el cálculo relacional para realizar consultas y obtener información de forma sencilla, y además para realizar cambios en ella.

   ¿Qué es T-SQL? T-SQL (Transact-SQL) es la manera en que se comunican las instrucciones de manipulación de datos que gestiona el usuario con el Servidor; las cuales permiten realizar operaciones claves en SQL Server, como creación y modificación de esquemas de base de datos, inserción y modificación de datos y además la administración del propio Servidor de Base de Datos. Esto se realiza mediante el envío de sentencias e instrucciones en T-SQL que son procesadas por el servidor y los resultados regresan a la aplicación cliente.

   Transact-SQL incluye características propias de cualquier lenguaje de programación que nos permiten definir:

• Tipos de datos.

- Definición de variables.
- Estructuras de control de flujo.
- Gestión de excepciones.
- Funciones predefinidas.

Sin embargo, no nos permite:

- Crear interfaces de usuario.
- Crear aplicaciones ejecutables.

Microsoft SQL Server Es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft. El lenguaje utilizado (que puede ser ejecutado por línea de comandos o mediante la interfaz gráfica de Management Studio) es Transact-SQL T-SQL

4. Que son las Tablas, Campos y registros

Las tablas son objetos de base de datos que contienen todos sus datos. En las tablas, los datos se organizan con arreglo a un formato de filas y columnas, similar al de una hoja de cálculo. Cada fila representa un registro único y cada columna un campo dentro del registro. Por ejemplo, en una tabla que contiene los datos de los empleados de una compañía puede haber una fila para cada empleado y distintas columnas en las que figuren detalles de los mismos, como el número de empleado, el nombre, la dirección, el puesto que ocupa y su número de teléfono particular.

Origen y estructura

Las tablas se componen de dos elementos principales que vamos a utilizar muy seguido cuando hablemos de bases de datos y de lenguaje SQL, ellos son los campos y los registros:

Campo: es la denominación que se le da a la columna, el mismo tiene un nombre único, un tipo de datos específico y un formato con una longitud o ancho de campo.

Registro: así se llama a cada fila que compone una tabla, es decir los renglones de la tabla, y están compuestos por cada uno de los campos de la tabla.

Dentro de una base de datos las "Tablas", son objetos, de vital importancia, y se podría decir que constituyen el arreglo principal que pueden tener los datos dentro de la misma, ya que viene siendo la forma más común de guardar y organizar los mismos. Su estructura general es comparable a una hoja de cálculo de Excel, es decir que la podríamos imaginar como un conjunto de celdas donde tenemos un campo al lado del otro, los cuales todos juntos conforman un registro y así luego tenemos un registro debajo del otro, completando la estructura final de la tabla.

A continuación, mostraremos un ejemplo.

Campos de la Tabla (Columnas - amarillo)

ID	Nombre	Descripción	Precio
1	Chaqueta	Chaqueta para dama tipo jean	100000
2	Camiseta	Tipo polo marca ACME	30000
3	Pantalón	Para caballero tipo Jean	80000
4	Camisa	Para Caballero	

Registros de la tabla (Filas - azul)

Propiedades especiales

A los campos se les puede asignar, además, propiedades especiales que afectan a los registros insertados, como ser que no se admitan valores nulos, para lo cual dicho campo deberá tener al menos un valor, o bien que sean campos índice, los cuales se usarán como criterio principal a la hora de ordenar los datos de la tabla y demás acciones que veremos más adelante, y por último pueden ser campos autonuméricos o autoincrementables los cuales básicamente sirven como para dar un único identificador o Id a los registros de la tabla.

Todas estas propiedades no son las únicas y las veremos más en detalle cuando veamos cómo se crea y construye una tabla pero como vimos se resumen en:

- Admitir valores Nulos
- Indexación
- Campo autoincrementable

El espacio de una tabla

Una tabla es una forma o modelo para guardar y organizar los datos dentro de una base de datos, son uno de los objetos principales de dichas bases, y existen como tales desde el momento de crearse hasta que sean eliminadas, permanecen guardadas invariables hasta que se les realice alguna modificación y no varían sus datos a no ser que sean actualizados, se inserten registros nuevos, se eliminen registros o bien se elimine la tabla. También se les puede agregar campos nuevos o eliminar alguno de los mismos.

Con todo esto también queremos explicar que la tabla ocupa un espacio dentro de la base de datos, este espacio está determinado por el total de la cantidad de registros y estos a su vez por el total de campos que los conforman.

5. Que es un Tipo de Dato

   En ciencias de la computación, un tipo de dato informático o simplemente tipo, es un atributo de los datos que indica al ordenador (y/o al programador/programadora) sobre la clase de datos que se va a manejar. Esto incluye imponer restricciones en los datos, como qué valores pueden tomar y qué operaciones se pueden realizar.

   Los tipos de datos más comunes son: números enteros, números con signo (negativos), números de coma flotante (decimales), cadenas alfanuméricas (y unicodes), estados, etc.

   Un tipo de dato es, un espacio en memoria con restricciones. Por ejemplo, el tipo "int" representa, generalmente, un conjunto de enteros de 32 bits cuyo rango va desde el -2.147.483.648 al 2.147.483.647, así como las operaciones que se pueden realizar con los enteros, como son la suma, la resta, y la multiplicación. Los colores, por su parte, se representan como tres bytes denotando la cantidad de rojo, verde y azul, y una cadena de caracteres representando el nombre del color (en este caso, las operaciones permitidas incluyen la adición y la sustracción, pero no la multiplicación).

   Este es un concepto propio de la informática, y más específicamente de los lenguajes de programación, aunque también se encuentra relacionado con nociones similares de la matemática y la lógica.

   En un sentido amplio, un tipo de datos define un conjunto de valores y las operaciones sobre esos valores. Casi todos los lenguajes de programación explícitamente incluyen la notación del tipo de datos, aunque lenguajes diferentes pueden usar terminologías diferentes. La mayor parte de los lenguajes de programación permiten al programador definir tipos de datos adicionales, normalmente combinando múltiples elementos de otros tipos y definiendo las operaciones del nuevo tipo de dato. Por ejemplo, un programador puede crear un nuevo tipo de dato llamado "Persona", contemplando que el dato interpretado como Persona incluiya un nombre y una fecha de nacimiento.

   Un tipo de dato puede ser también visto como una limitación impuesta en la interpretación de los datos en un sistema de tipificación, describiendo la representación, la interpretación y la estructura de los valores u objetos almacenados en la memoria del ordenador. El sistema de tipificación usa información de los tipos de datos para comprobar la verificación de los programas que acceden o manipulan los datos.

   A continuación, se muestran cuáles son los tipos de datos que se pueden utilizar en SQL Server.

   Repaso de Unidades de Medida en Informática

   Bit = 0 ó 1
   Byte = 8 bits
   Kilobyte = 1024 bytes

Megabyte = 1024 kilobytes
Gigabyte = 1024 megabytes
Terabyte = 1024 gigabytes
Petabyte = 1024 terabytes

Tipos de datos - sql server

TIPOS DE DATOS STANDARD (Más utilizados)

- Numéricos
  - Enteros BIT, TINYINT, SMALLINT, INT, BIGINT
  - Decimales MONEY, DECIMAL
- Texto y Binarios
  - CHAR, VARCHAR, NCHAR, NVARCHAR
  - BINARY, VARBINARY
- Fecha y Hora
  - DATE, TIME, DATETIME, SMALLDATETIME

Descripción de Tipos de Dato y sus tamaños posibles
BIT 1 byte, 0 ó 1 (True ó False)

TINYINT 1 byte: 0 a 255

SMALLINT 2 bytes: -2^15 (-32,768) HASTA 2^15-1 (32,767)

INT 4 bytes: -2^31 (-2,147,483,648) HASTA 2^31-1 (2,147,483,647)

BIGINT 8 bytes: -2^63 (-9,223,372,036,854,775,808) HASTA 2^63-1 (9,223,372,036,854,775,807)

MONEY 8 bytes: -922,337,203,685,477.5808 HASTA 922,337,203,685,477.5807

DECIMAL(10,2) Precisión y Escala
1 a 9: 5 bytes
10 a 19: 9 bytes
20 a 28: 13 bytes
29 a 38: 17 bytes

CHAR 1 byte por carácter UTF-8 = 1 HASTA 8000

VARCHAR 1 byte por carácter variable UTF-8= 1 HASTA 8000

MAX HASTA 2^31-1 bytes (2 GB)

NCHAR 2 bytes por caracter UTF-16: 1 HASTA 4000

NVARCHAR 2 bytes por caracter UTF-16: 1 HASTA 4000
MAX 2^30-1 characters (2 GB de espacio) (1GB de datos)

BINARY 1 byte por valor 1 HASTA 8000

VARBINARY 1 byte por valor variable: 1 HASTA 8000
2^31-1 bytes (2GB)

DATE 3 bytes: YYYY-MM-DD 01/01/0001 HASTA 31/12/9999

TIME 5 bytes: hh:mm:ss:nnnnnnn, 00:00:00.0000000 HASTA 23:59:59.9999999

DATETIME 8 bytes: YYYY-MM-DD hh:mm:ss:nnn 01/01/1753 HASTA 31/12/9999
00:00:00 HASTA 23:59:59.997

SMALLDATETIME 4 bytes: YYYY-MM-DD hh:mm:ss 01/01/1900 HASTA 06/06/2079
00:00:00 HASTA 23:59:59

6. Restricciones y Propiedades de los campos

PRIMARY KEY Definiciones y reglas generales

- ✓ La clave primaria o primary key, identifica de manera unívoca (única) a cada registro de una tabla.
- ✓ El valor que contiene la columna definida como primary key, debe ser único.
- ✓ El valor debe ser NOT NULL (no permitirá valores nulos)
- ✓ Una tabla puede tener más de un campo PK, a la que llamaremos CLAVE COMPUESTA
- ✓ Sea SIMPLE o COMPUESTA, cada tabla solo podrá tener una clave primaria (PRIMARY KEY)
- ✓ Es decir que sólo habrá una única clave primaria de ese tipo. No se podrá crear otra combinación.

IdPaciente ▽	nombre	apellido	email	idpais
1	Jorge	Rodriguez	a@a.com	MEX
2	Marcelo	Lopez Llano	a@a.com	MEX
3	Kari	Lopreta	a@a.com	COL
4	Juan Manuel	Loperfano	a@a.com	ARG
5	Juan Manuel	Perez Lozano	a@a.com	ESP
6	Karim	Berragas	a@a.com	PER
7	Saul	Lopez Gomez	a@a.com	CHI

7. FOREING KEY Definiciones y reglas generales

- ✓ La clave foránea o foreign key, debe ser del mismo tipo de dato que su campo relacionado.
- ✓ El valor del campo definido como FK puede ser NULL
- ✓ Una tabla puede tener más de un campo FK

IdPaciente ▽	nombre	apellido	email	Idpais (FK) ▽
1	Jorge	Rodriguez	a@a.com	MEX
2	Marcelo	Lopez Llano	a@a.com	MEX
3	Kari	Lopreta	a@a.com	COL
4	Juan Manuel	Loperfano	a@a.com	ARG
5	Juan Manuel	Perez Lozano	a@a.com	ESP
6	Karim	Berragas	a@a.com	PER
7	Saul	Lopez Gomez	a@a.com	CHI

Idpais ▽	pais
MEX	México
COL	Colombia
ARG	Argentina

8. Fundamentos y Diseño de Bases de Datos

Modelo entidad-relación ER

El modelo entidad-relación ER es un modelo de datos que permite representar cualquier abstracción, percepción y conocimiento en un sistema de información formado por un conjunto de objetos denominados entidades y relaciones, incorporando una representación visual conocida como diagrama entidad-relación.

Conceptos del modelo ER

Registros: guardan una serie de características similares o que pueden ser agrupados o clasificados dadas sus características comunes en grupos bien delimitados, en términos de abstracción como la extensión de la base de datos. Por ejemplo, es la lista de usuarios de una biblioteca, la lista de productos con sus características, la lista de tipos de documentos y su definición.

Entidad(Tabla). La entidad es cualquier clase de objeto o conjunto de elementos presentes o no, en un contexto determinado dado por el sistema de información o las funciones y procesos que se definen en un plan de automatización. Dicho de otra forma, las entidades las constituyen las tablas de la base de datos que permiten el almacenamiento de los ejemplares o registros del sistema, quedando recogidos bajo la denominación o título de la tabla o entidad. Por ejemplo, la entidad usuarios guarda los datos personales de los usuarios de la biblioteca, la entidad catalogo registra

todos los libros catalogados, la entidad circulación todos los libros prestados y devueltos y así sucesivamente con todos los casos.

Atributos - Intención. Son las características, rasgos y propiedades de una entidad, que toman como valor una instancia particular. Es decir, los atributos de una tabla son en realidad sus campos descriptivos, el predicado que permite definir lo que decimos de un determinado sujeto. Por ejemplo, de una entidad o tabla catálogo, se pueden determinar los atributos título, subtítulo, título paralelo, otras formas del título, autor principal, otras menciones de responsabilidad, edición, mención de edición, editorial, lugar de publicación, fecha de publicación, ...

Relación. Vínculo que permite definir una dependencia entre los conjuntos de dos o más entidades. Esto es la relación entre la información contenida en los registros de varias tablas. Por ejemplo, los usuarios suelen clasificarse según una lista de tipos de usuarios, ya sean profesores, alumnos o investigadores. De esta forma es posible emitir la relación entre el usuario Jorge Martínez como alumno y Enrique Valtierra como profesor. Las relaciones son definidas de forma natural en un diagrama relacional para expresar un modelo cognitivo que dará lugar posteriormente a las interrelaciones de las entidades.

Interrelación. Las interrelaciones las constituyen los vínculos entre entidades, de forma tal que representan las relaciones definidas en el esquema relacional de forma efectiva. Esto no sólo la relación de los registros sino de sus tablas y de las características de la interrelación entre las entidades, a través de un campo clave que actúa como código de identificación y referencia para relacionar (es decir, como nexo de unión y articulación de la relación). Los tipos de interrelaciones entre entidades o tablas se realizan aplicando las reglas de cardinalidad y modalidad.

Entidades fuertes. Lo constituyen las tablas principales de la base de datos que contienen los registros principales del sistema de información y que requieren de entidades o tablas auxiliares para completar su descripción o información. Por ejemplo, la tabla usuario es una entidad fuerte en relación a la tabla tipos de usuarios, que es una entidad débil dada su condición auxiliar para clasificar a los usuarios registrados en la biblioteca.

Entidades débiles. Son entidades débiles a las tablas auxiliares de una tabla principal a la que completan o complementan con la información de sus registros relacionados. Por ejemplo, también son consideradas entidades débiles las tablas intermedias que sirven para compartir información de varias tablas principales.

Modelo Entidad – Relación	Objeto de la Base de Datos	Ejemplo
Registros - Conjuntos-Extensiones	Registros de una Tabla	Conjunto-Usuarios{Jorge Martínez(1|alumno), Enrique Valtierra(2|profesor), Miguel dos Santos(3|investigador)}
Entidad	Tabla de la base de datos	Tabla usuarios
Atributos – Intención	Campos de una tabla	id, nombre, apellidos, tipo de usuario, dni, dirección, teléfono.
Relación	Vínculo entre conjuntos	Jorge Martínez es investigador

Interrelación	Relación entre tablas	Tabla Usuarios relacionada con Tabla Tipo de usuarios
Entidades fuertes	Tabla principal	Tabla Usuarios
Entidades débiles	Tabla auxiliar	Tabla Tipo de usuarios

Restricciones del MER

- ✓ Un único constructor (la relación).
- ✓ La relación es un conjunto (en el sentido matemático) y por lo
- ✓ tanto:
    - o No pueden existir tuplas duplicadas
    - o Toda relación debe tener una llave primaria 1
    - o No hay noción de orden entre las tuplas o entre los atributos.
    - o Los atributos de la llave primaria NO pueden ser nulos

Llaves Primarias

- ✓ **Llave**: Atributo o conjunto de atributos cuyo valor es único y diferente para cada tupla, una tabla puede poseer más de una llave.
- ✓ **Llave Candidata**: Es una llave tal que:
    - ✓ Es única (i.e., es una llave).
    - ✓ Es irreducible: No se pueden eliminar componentes de la llave sin destruir la unicidad.

Las llaves no se identifican dependiendo del estado o la instancia de la base de datos.

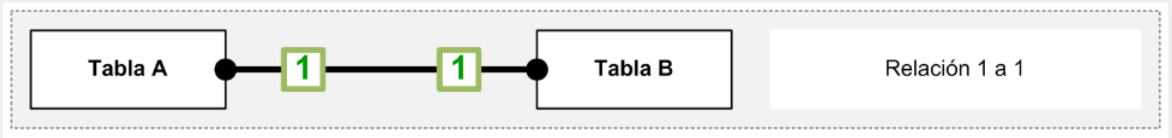
- ✓ **Llave Compuesta**: Es una llave conformada por más de un atributo.
- ✓ **Llave Primaria**: Llave candidata que se escoge en el modelo para identificar cada tupla.
- ✓ **Llaves alternas**: Las llaves candidatas que no fueron escogidas como llave primaria
- ✓ **Llave Foránea:** atributo(s) de una relación r1 que hacen referencia a otra relación r2 Las FK permiten representar relaciones entre las entidades. r1 y r2 pueden ser la misma relación. Por ejemplo: personas (cc, nombre, apellido, teléfono, dirección, ciu_id) ciudades(id, nombre).
  El atributo ciu_id en personas es una llave foránea que referencia al atributo id en la tabla ciudades.

Integridad referencial. Se denomina integridad referencial al tipo de interrelación que se produce entre tablas mediante un campo clave que deberá contener la cadena alfanumérica exacta al identificador de la tabla auxiliar para poder realizar la relación entre los registros. En caso contrario no se produce la relación. Además, se trata de un mecanismo que evita duplicidades e incorrecciones ya que la propiedad de integridad referencial conmina a que los datos de un usuario además de su identificador ID sean distintos al de los demás. Dicho de otra forma, no pueden existir dos registros iguales con los mismos datos.

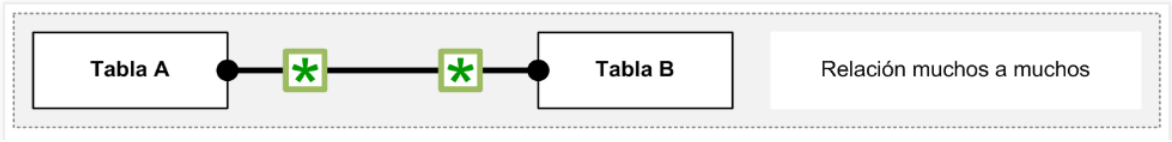
Tipos de relaciones

Según cardinalidad. La cardinalidad se representan en un diagrama ER como una etiqueta que se ubica en ambos extremos de la línea de relación de las entidades y que puede contener diversos valores entre los que destacan comúnmente el 1 y el *, obteniendo los siguientes tipos:

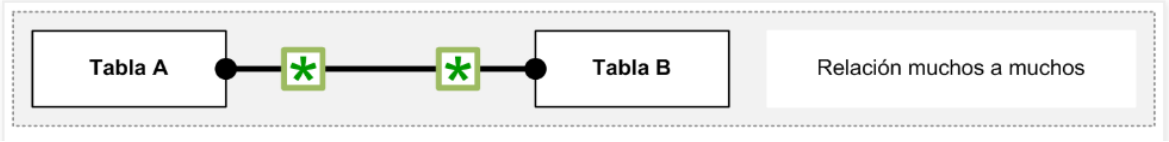
Relación 1 a 1. La relación uno a uno, define que un único registro de la tabla puede estar relacionado con un único registro de la tabla relacionada.



Relación 1 a *. La relación de uno a varios, define que un registro dado de una tabla auxiliar o secundaria sólo puede estar vinculado con un único registro de la tabla principal con la que está relacionada.



Relación * a *. La relación de varios a varios, define que un registro de una tabla puede estar relacionado con varios registros de la tabla relacionada y viceversa.



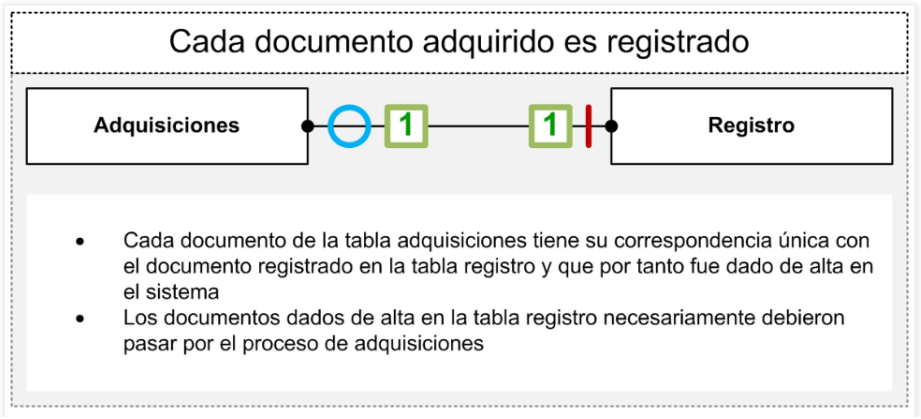
Según modalidad

- ✓ **Optativa**. La relación entre un registro de una tabla y varios de la tabla relacionada, puede existir o no.
- ✓ **Obligatoria.** La relación entre un registro de una tabla y otro de la tabla relacionada es obligada, debe existir siempre.

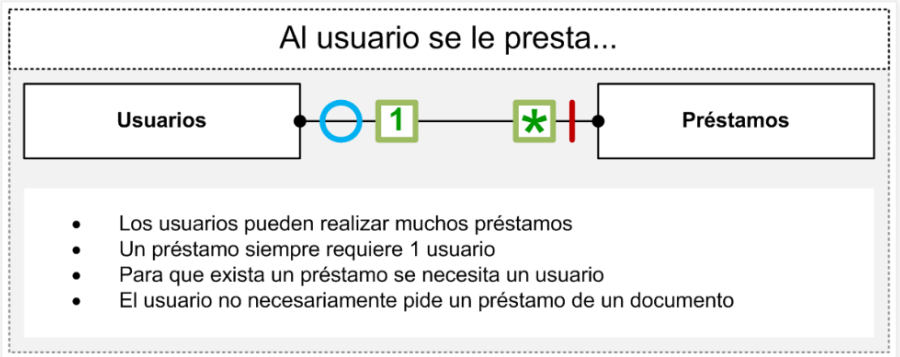
Ejemplos

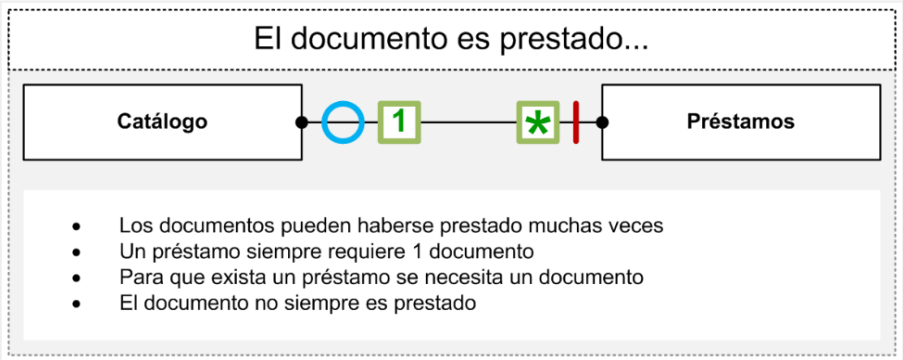
Relación 1 a 1. Cada documento adquirido es registrado podría equipararse a la cardinalidad 1 a 1. Esto significa que cada documento que se introduce en el módulo de adquisiciones (y por ende en su tabla) tiene su correspondencia con los documentos que finalmente se reciben en la biblioteca para ser dados de alta en la tabla registro. De esta forma, puede haber o no documentos en proceso

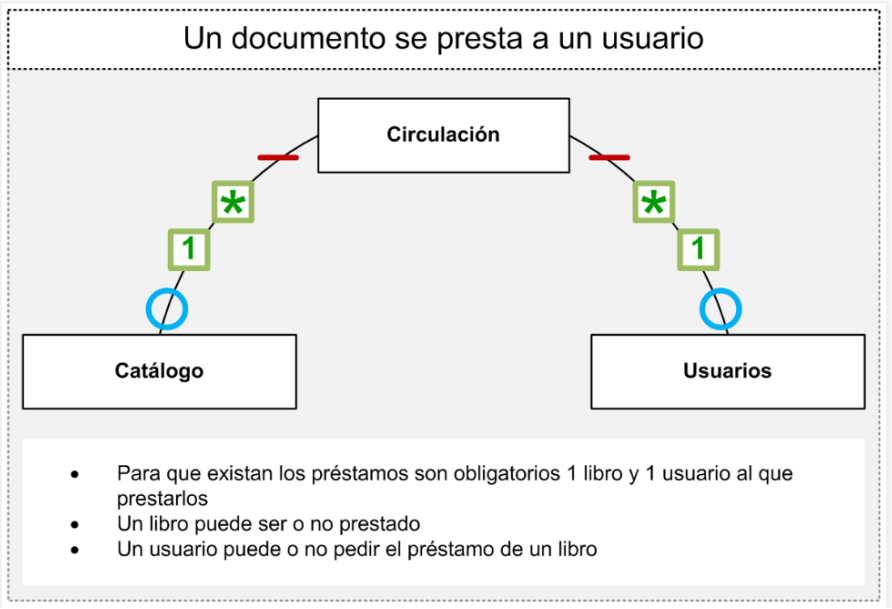
de adquisición (relación optativa). En cambio, la tabla registro se encarga de registrar los documentos que se reciben por lo que su relación es de obligatoriedad (todo documento registrado está presente en la tabla de adquisiciones). Todo ello no implica que todos los documentos en fase de adquisición tengan que estar registrados. Pueden existir documentos en fase de adquisición que no hubieran sido registrados.



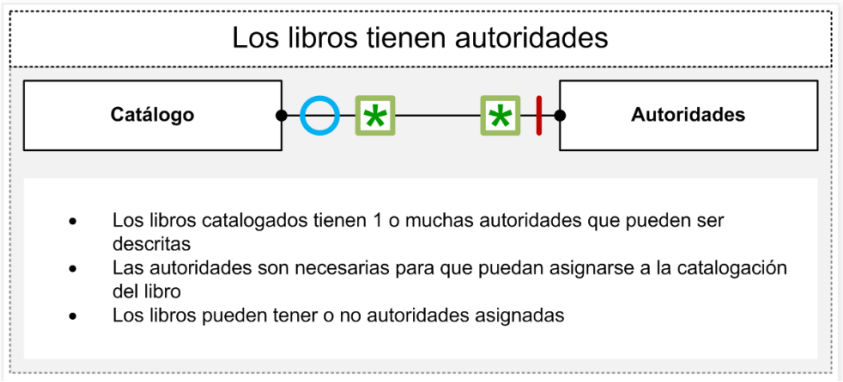
Relación 1 a *. Los usuarios de una biblioteca suelen solicitar préstamos, por lo tanto, la relación que se produce es de uno a muchos. Un usuario puede pedir o no el préstamo de uno o varios libros o documentos. Por lo tanto, pueden existir uno o ningún usuario solicitando el préstamo, pero para que exista la relación con la tabla préstamos, ésta debe registrar al usuario, su fecha de préstamo y devolución. Véase figura6. Por otra parte, el préstamo se compone no sólo del usuario que lo solicita, sino del documento u objeto que le es prestado, que es el caso que se expone en la figura7. Al igual que en la relación del usuario con la tabla préstamo, un documento puede o no ser prestado. Un documento puede haber sido prestado y devuelto muchas veces y para que exista la relación entre la tabla préstamos y catálogo, debe existir un registro que integre su información (de aquí su modalidad de relación obligatoria). Todo ello conduce a una relación más compleja que puede observarse en la figura8. Lo que en un principio se consideraba una relación de 1 a muchos, termina convirtiéndose en una relación de muchos a muchos gracias a una tabla débil o intermedia que almacena la información necesaria del usuario y del documento para poder efectuar el préstamo correspondiente. Por lo tanto, la tabla préstamos, relaciona muchos usuarios con muchos libros en múltiples conjuntos o registros que pueden estar activos o finalizados. Recuérdese que los libros una vez devueltos vuelven a estar disponibles para dar servicio a terceros usuarios. Por ello se concluye que para que un préstamo tenga lugar, deberá estar presente el identificador del usuario y del documento siendo obligatorios en todo proceso de circulación.







Relación * a *. Cuando se catalogan los documentos en una biblioteca, al seguir las indicaciones de las normas de catalogación, se advierte un apartado de suma importancia; las autoridades. Éstas definen la responsabilidad intelectual, artística, cognitiva, editorial, administrativa, introductoria, del documento. Por ello no es de extrañar que en la catalogación los campos de autoridades sean repetibles, dado que pueden existir 1 o más autoridades. Esta relación es la que se advierte en la figura9. Cada libro puede tener o no 1 o muchas autoridades. Por lo tanto, una autoridad puede estar presente en varios libros o formar parte de varias responsabilidades en el mismo.



9. Definición de un campo IDENTITY

La propiedad Identity se puede establecer a uno o más campos de una tabla determinada. Un campo con propiedad Identity activada, hará que su valor se incremente automáticamente a medida que se inserten registros en la tabla.

Es por ello que, para que un campo pueda ser Identity, su tipo de dato debe ser numérico. Por lo general establecemos Identity a un campo que es Primary Key. Con esto logramos que su valor no se duplique, generando registros unívocos (únicos).

Argumentos de la propiedad Identity

Existe un argumento requerido llamado Seed.

- ✓ El argumento Seed define desde que valor comienza a incrementar su valor.
- ✓ Si el valor de Seed es 1, comenzará a incrementarse desde 1, Si el valor de Seed es 5, comenzará a incrementarse desde 5.

También existe un argumento de la propiedad Identity, llamado Increment.

- ✓ Si el valor de Increment es 1, el valor del campo se guardará con valores correlativos ej: 1,2,3,4
- ✓ Si el valor de Increment es 2, el valor del campo se guardará con valores correlativos ej: 1,3,5,7

Para definir un campo con propiedad Identity (Seed=1, Increment=2) desde T-SQL colocamos:

Ej: [nombreCampo] [int] IDENTITY(1,2)
Ejemplo de tabla con propiedad IDENTITY(1,2)

Idpaciente (PK)	nombre	IdPais	FechaNacimiento
1	Juan Carlos Ruber	ESP	01-12-1990
3	Carlos Andrés Montoya	MEX	15-05-1978
5	Juan Sanchez	MEX	20-03-1992
7	Juan Sanchez	MEX	16-07-1992

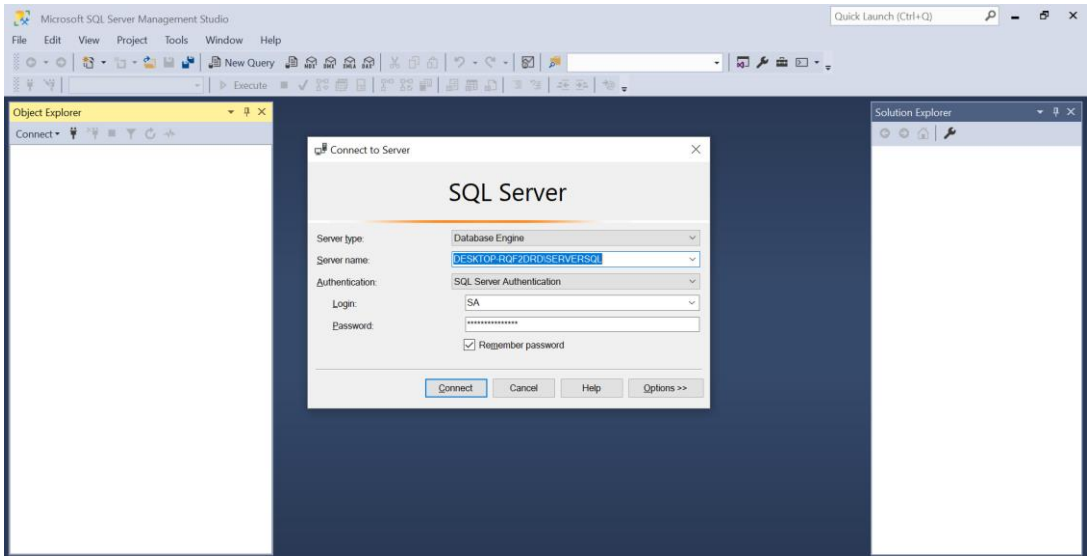
Ejemplo de tabla con propiedad IDENTITY(10,1)

Idpaciente (PK)	nombre	IdPais	FechaNacimiento
10	Juan Carlos Ruber	ESP	01-12-1990
11	Carlos Andrés Montoya	MEX	15-05-1978
12	Juan Sanchez	MEX	20-03-1992
13	Juan Sanchez	MEX	16-07-1992

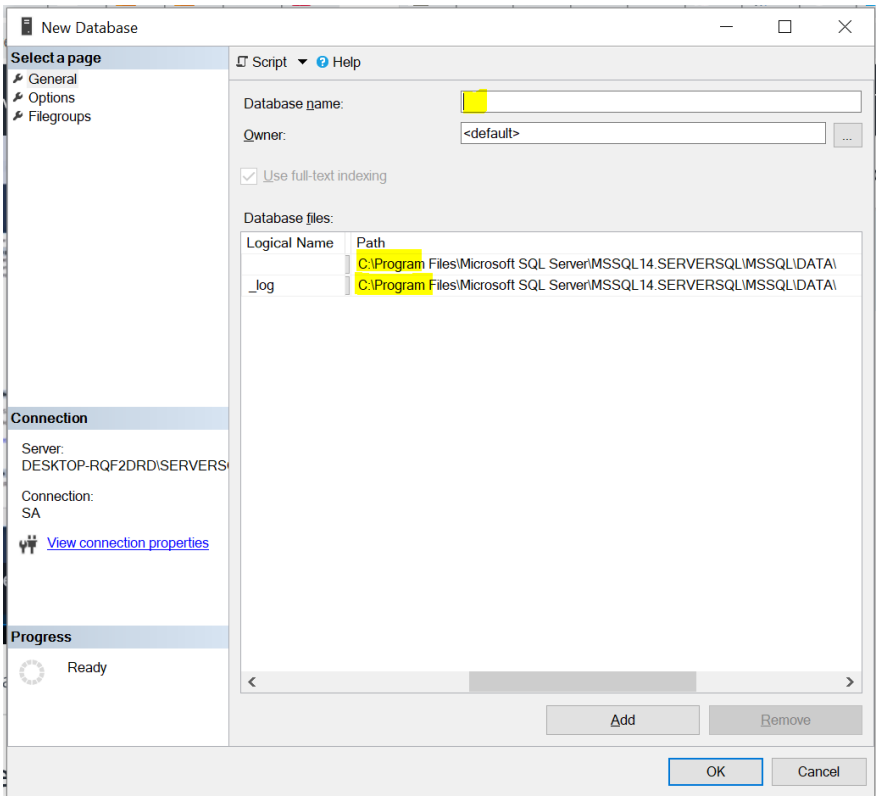
10. Practica 1: Creando el Archivo de Bases de Datos

Para el desarrollo de esta práctica vamos a realizar las siguientes actividades:

- ✓ Abrir el Microsoft Managment Studio, una vez iniciado vamos a poner la contraseña para el usuario SA que hallamos puesto durante el proceso de la instalación, clic en la opción connect.



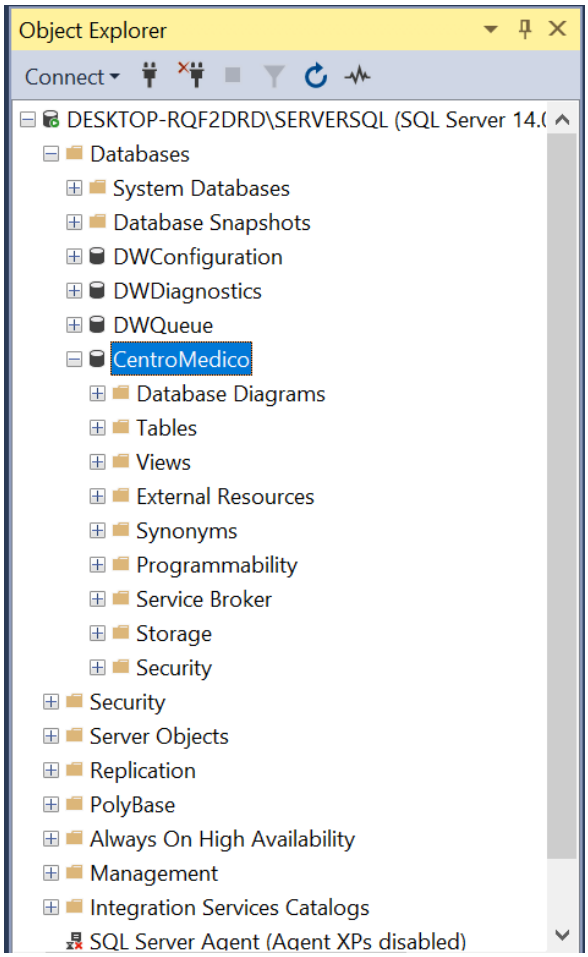
- ✓ Una vez conectado a nuestro motor el siguiente paso será seleccionar la opción **Databases**, dar clic derecho y elegimos la opción nueva base de datos.



En el campo Database name: puedes poner el nombre a nuestra base de datos, aunque lo importante en este caso es tener presente en qué lugar exactamente será almacenada nuestra base de Datos, la ruta especifica tiene dos archivos uno en formato MDF y el otro en formato LDF, el primero es el archivo de la base de datos y el segundo es su log.

El nombre que utilizaremos para nuestra base de datos será: CentroMedico
Recomendación: no guardar la base de datos en la misma partición C donde está el Sistema operativo por seguridad, en caso de falla del sistema operativo la base de datos no se vaya a perder en caso de tener que formatear el disco.

✓ Una vez creada la base datos ya podemos ir a navegar sobre la opción databases, nos debe mostrar la base de datos recientemente creada y al dar clic en las propiedades de la base de datos podemos observar los diferentes objetos que podremos crear posteriormente sobre ella.



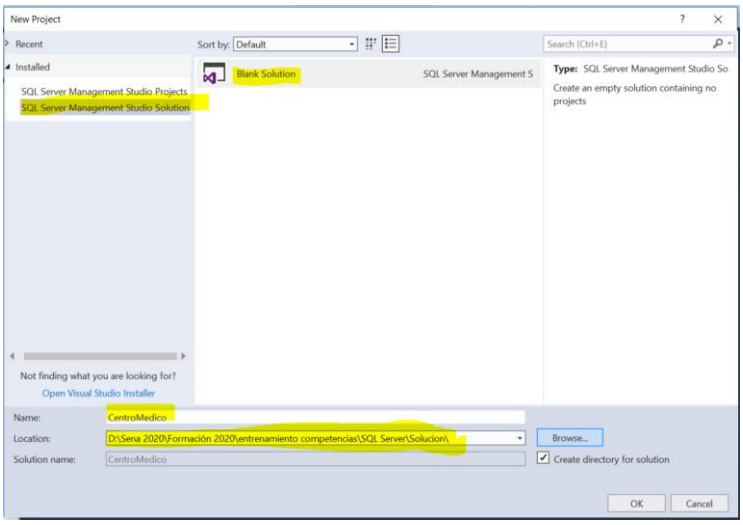
Podemos observar algunos de los objetos como son: Diagramas de bases de datos, Tablas, vistas, programación (donde se almacenan los procedimientos almacenados) entre otros.

11. Crear Proyectos y Soluciones en SQL Server

Cada vez que necesitemos trabajar con una base de datos en la instancia que tenemos en nuestro servidor se va a requerir crear una solución y proyecto por ejemplo para nuestra preparación utilizaremos una base de datos para un centro médico por lo que crearemos la solución para esta base de datos.

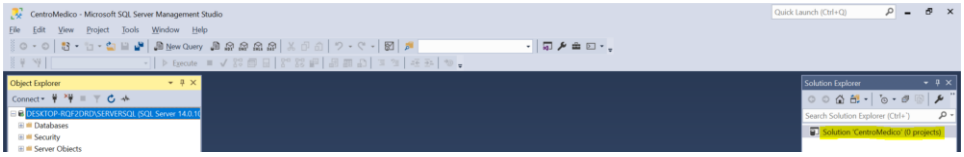
Una solución de SQL Server Management Studio es una colección de uno o más proyectos relacionados. Los proyectos son contenedores que los desarrolladores de software usan para organizar los archivos relacionados, como conjuntos de scripts de uso común.

Inicialmente lo que haremos será ir al menú File, New y Proyecto, vamos a encontrar en la ventana dos alternativas una para crear el proyecto y otra para crear la solución.

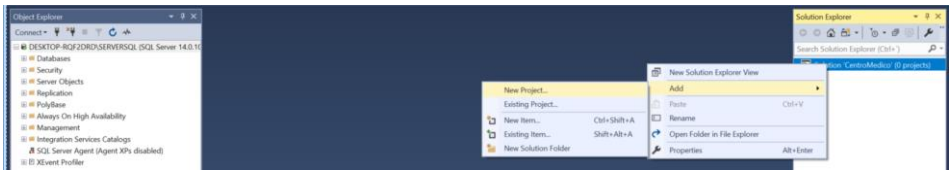


De esta manera se crea la solución al dar clic en el botón OK.

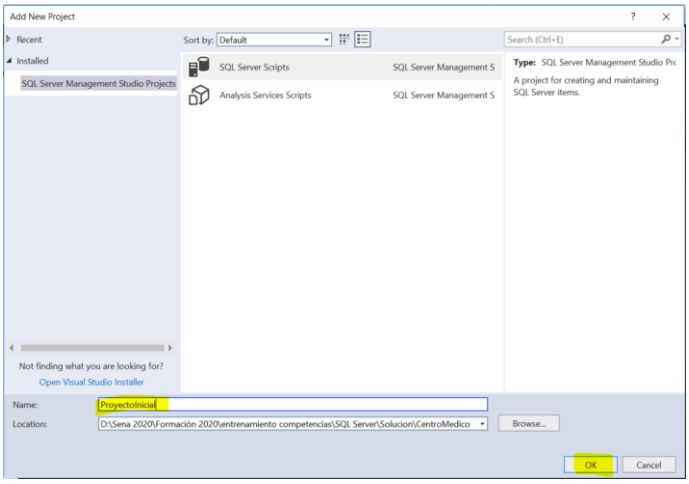
Ahora una solución puede contener uno o varios proyectos, todo dependerá del tipo de solución que estemos trabajando, una vez creada la solución en el panel izquierdo del Microsoft Managment Studio debe mostrarnos la solución que hemos acabado de crear.



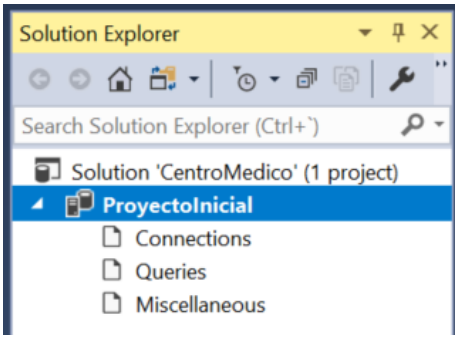
El paso siguiente será crear nuestro proyecto, par esto vamos a dar clic derecho en el nombre de la solución, ADD, NEW Project.



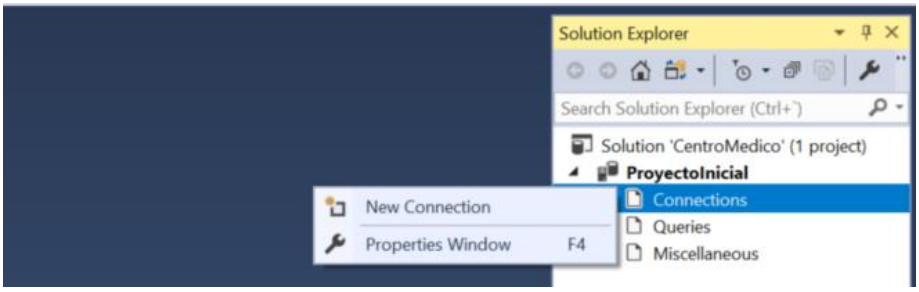
De esta manera creamos nuestro proyecto



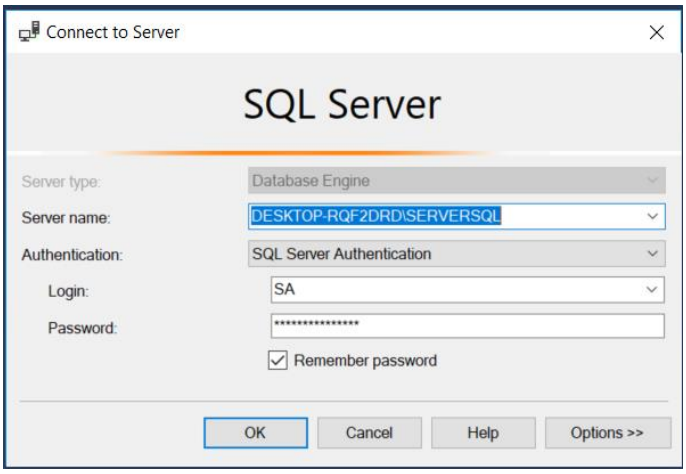
De esta manera se crea nuestro primer proyecto el cual se almacena dentro de la ruta donde fue creada la solución.



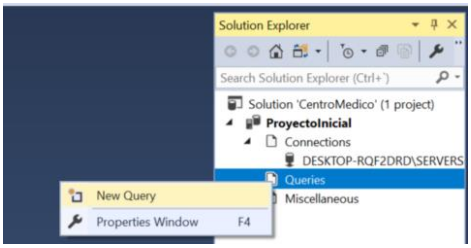
El paso siguiente es crear una conexión al servidor de base de datos de SQL, para ello damos clic en la opción connections del proyecto.



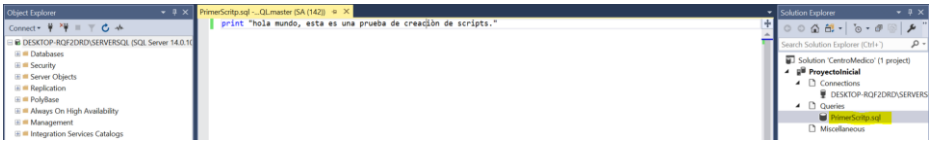
Nos muestra una ventana donde haremos la conexión a la instancia del servidor local o remoto donde se encuentra la base de datos con la instancia a conectar.



Esta estructura de trabajar con soluciones y proyectos nos permite organizar el área de trabajo de tal manera que haya un estándar organizativo, por ejemplo si para mi solución y/o proyecto requiero generar algunos script, para ello debo ir al proyecto clic derecho en la opción Queris y de esta manera podremos ir generando y guardando nuestros scripts.



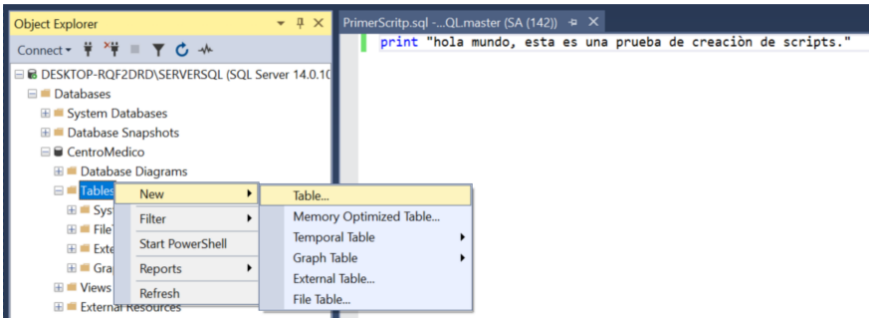
Para nuestro caso crearemos un script que simplemente haga uso de la función print y mostrar en pantalla un mensaje de texto, pero lo importante es saber que de esta manera se pueden ir creando, modificando y gestionando los scripts que vayamos generando para nuestro proyecto y solución.



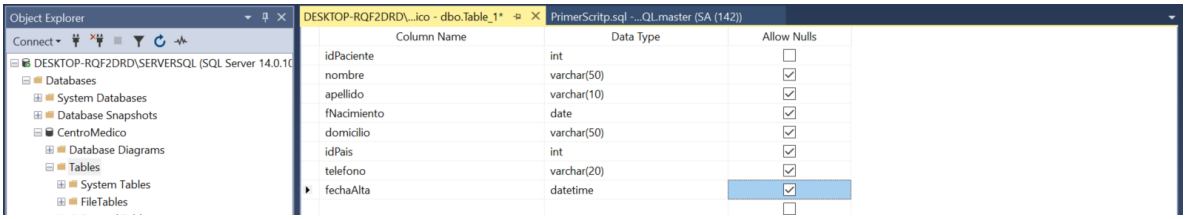
12. Crear una tabla

Desde Microsoft Managment studio podemos crear las tablas de dos maneras, una es haciendo uso de la interfaz gráfica y la otra es haciendo el script para que desde consola se vayan creando las tablas requeridas.

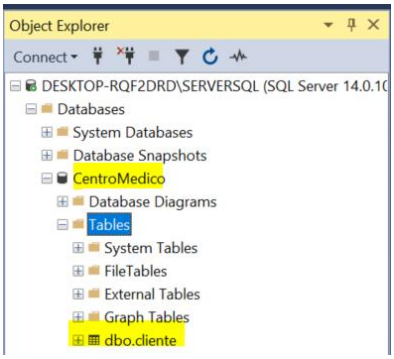
Para la primera alternativa nos ubicamos en nuestra base de datos creada, luego seleccionamos la opción tables, damos clic derecho y luego en nuevo.



El siguiente paso será definir cuales atributos serán los que harán parte de la Tabla, para nuestro ejemplo vamos a crear una tabla llamada paciente de tal manera que en el diseño es donde debemos estar atentos a definir qué campos y de que tipos de datos serán los que conformarán la tabla a crear.



Finalmente damos clic en guardar y listo tenemos ya creada nuestra primera tabla, si actualizamos el Explorador de objetos vemos como la tabla ya quedo creada.



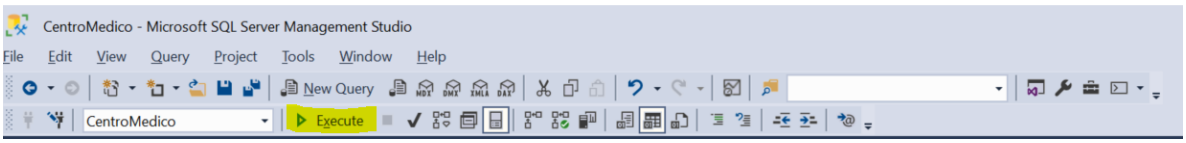
La segunda alternativa es crear la tabla mediante un script.

Para ellos haremos uso del SQL a manera de ejemplo simplemente porque ya entraremos a conocer y utilizar cada una de las alternativas del lenguaje SQL como lo son las sentencias de tipo DML, DDL y DCL.

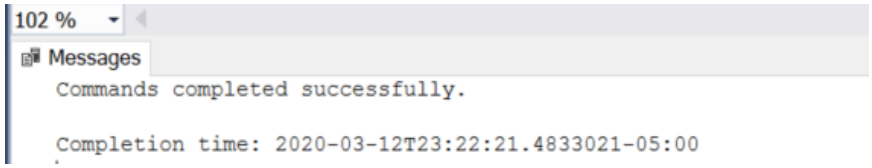
Lo que vamos a hacer es crear un nuevo script en el proyecto en la opción de Queries al que llamaremos CreatePaciente.sql

```
CREATE TABLE paciente1(
	idPaciente int NOT NULL,
	nombre varchar(50) NOT NULL,
	apellido varchar(50) NOT NULL,
	fNacimiento date NULL,
	idPais INT NOT NULL,
	telefono varchar(20) NULL,
	email varchar(30) NULL,
	observacion varchar(1000) NULL,
	fechaAlta datetime NOT NULL,
	CONSTRAINT PK_idpaciente PRIMARY KEY (idPaciente)
)
```

De esta manera podemos crear nuestra tabla desde la consola utilizando el script acá solo faltaría ejecutar el script y de esa manera podremos validar la creación de la tabla, presionamos la tecla F5 o simplemente damos clic en el botón ejecutar.

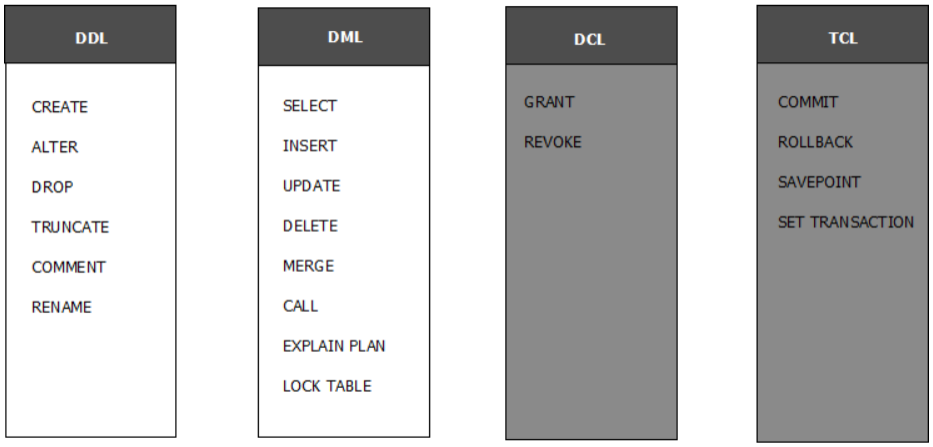


Si todo esta OK, al ejecutar el script nos debe mostrar un mensaje indicando que la tabla fue creada exitosamente y que no hubo inconvenientes.



13. Comandos de SQL

Los comandos de SQL son sentencias que se pueden utilizar para diferentes tareas y están divididos en cuatro grupos: DDL, DML, DCL y TCL.



Iniciaremos el estudio de los dos primeros las sentencias o comandos DDL y DML.

14. DDL

Comandos DDL (**Data Definition Language** – Lenguaje Definición de Datos) para creación, modificación y eliminación de la estructura y objetos de la base de datos.

CREATE – Utilizado para crear objetos (tablas, vistas, índices) en la base de datos

Ejemplo: CREATE DATABASE coleccion;

CREATE TABLE colección;

Al crear una tabla de base de datos debemos considerar la siguiente estructura:

CREATE TABLE nombre_tabla (

nombre_columna tipo_columna [cláusula_defecto] [vínculos_de_columna],

nombre_columna tipo_columna [cláusula_defecto] [vínculos_de_columna]

nombre_columna tipo_columna [cláusula_defecto] [vínculos_de_columna]

[vínculo_de tabla] …])

Nombre del campo deben empezar por un carácter alfabético y ser lo más comprensible posible para entender que datos se almacenan en ese campo.

Tipo del campo (lista de los más importante):

Numérico: bigint, int, smallint, tinyint, numeric, bit, decimal, money, smallmoney.

Cadena de caracteres: char, varchar, text.

Fecha y hora: date, datetime, time, timestamp.

Cadenas binarias: binary, varbinary, image.

Cláusula defecto se le asigna valor por defecto al campo si no se le indica en el momento que se inserta una fila. Puede iniciarse con un valor o nulo.

DEFAULT { valor | NULL }

Vínculos de integridad que se aplica a cada campo

NOT NULL no permite valor NULL

PRIMARY KEY para indicar que es la clave primaria de la tabla. Puede formarse por más de un campo, directamente debe almacenar valor único y que no sea NULL

FOREIGN KEY indica la clave foránea haciendo referencia a otra tabla, estableciendo la relación. Tiene las cláusulas ON DELETE y ON UPDATE indican que acción debe ejecutarse en el caso que la clave foránea (a quién hace referencia) es eliminada o borrada. Las acciones pueden ser:

o **CASCADE**: elimina o modifica la tupa que tiene el campo referenciado

o **SET DEFAULT**: asigna valor por defecto a la columna referenciada

o **SET NULL**: asigna valor NULL a la columna referenciada

Control de valor permite asignar o no un valor a la columna dependiendo del resultado de la condición

CHECK { expresión_condicional }

Vínculos de integridad que se pueden aplicar a más campos de la tabla

Clave primaria PRIMARY KEY (columna1 [, columna2 …])

Clave foránea FOREIGN KEY (columna1 [, columna2 …])

Ejemplos de creación de una tabla

```
CREATE TABLE consola (id_consola bigint IDENTITY(1,1),
nombre varchar (50),
tipo varchar (15) DEFAULT 'Sobremesa',
marca varchar (50),
PRIMARY KEY (id_consola)
)


CREATE TABLE juego(id_juego bigint IDENTITY(1,1),
titulo varchar(50),
genero varchar(50),
PRIMARY KEY (id_juego)
)


CREATE TABLE consola_juego(id_consola bigint,
id_juego bigint,
PRIMARY KEY (id_consola, id_juego),
FOREIGN KEY (id_consola) REFERENCES consola(id_consola) ON UPDATE CASCADE ON DELETE
CASCADE,
FOREIGN KEY (id_juego) REFERENCES juego(id_juego) ON UPDATE CASCADE ON DELETE
CASCADE
)
```

También podemos crear índices, se usan para encontrar un dato más rápidamente mejorando la eficacia de la consulta

CREATE INDEX index_marca

ON consola (marca)

CREATE UNIQUE INDEX index_nombre_marca

ON consola (nombre, marca)

15. ALTER

Utilizado para modificar las tablas agregando o cambiando la definición de los campos

Ejemplos:

```
ALTER TABLE consola ALTER COLUMN marca char;
ALTER TABLE consola ADD fecha_lanzamiento date;
ALTER TABLE consola DROP COLUMN fecha_lanzamiento;
```

¿Qué pasaría si creamos el campo fecha con valores por defecto y después intentamos eliminar el propio campo?

```
ALTER TABLE consola ADD fecha_lanzamiento DATE NOT NULL DEFAULT '2017-01-01'
```

```
ALTER TABLE consola DROP COLUMN fecha_lanzamiento
```

Msg 5074, Level 16, State 1, Line 1 The object 'DF__consola__fecha_l__276EDEB3' is dependent on column 'fecha_lanzamiento'.

Msg 4922, Level 16, State 9, Line 1 ALTER TABLE DROP COLUMN fecha_lanzamiento failed because one or more objects access this column.

Este mensaje nos indica que no podemos borrar un campo de una table cuando esta tiene ya datos almacenados.

16. DROP

Utilizado para eliminar objetos en la base de datos, Podemos eliminar un índice

Ejemplos:

```
DROP INDEX consola.index_nombre_marca
```

O la tabla entera

```
DROP TABLE consola
```

17. TRUNCATE

Borra todo el contenido de una tabla (Los registros que estén ingresados en la Tabla.)

Ejemplos:

```
TRUNCATE TABLE consola
```

Al intentar borrar todo el contenido de la tabla SQL Server nos muestra una advertencia como la siguiente: Msg 4712, Level 16, State 1, Line 1 Cannot truncate table 'consola' because it is being referenced by a FOREIGN KEY constraint.

En este caso no nos lo permitirá al haber una clave foránea que hace referencia a la tabla consola

18. COMMENT

Agregar comentarios al diccionario de datos sobre un objeto de la base de datos

Ejemplo:

COMMENT ON COLUMN consola.marca IS 'desarrolladora del producto';

19. RENAME

Renombrar la tabla de la base de datos

Ejemplo: `ALTER TABLE consola RENAME TO consola1`

Si modificamos el nombre, los script podrían dejar de funcionar. Hay que tener en cuenta también las relaciones que hay en la tabla, en Oracle por ejemplo si una tabla tiene una clave foránea, generará error.

20. DML

Data Manipulation Language – Lenguaje Manipulación de Datos) para recuperar y trabajar con datos.

21. SELECT

Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado

Sintaxis:

```
SELECT [ ALL | DISTINCT ] lista_elementos_selección
FROM lista_referencias_tabla
[ WHERE expresión_condicional ]
[ GROUP BY lista_columnas ]
[ HAVING expresión_condicional ]
[ ORDER BY lista_columnas ]
```

Se puede buscar todos los valores de un campo o varios campos de una tabla, o con DISTINCT eliminar aquellas filas cuyo campo se repite. Si no se indica ALL | DISTINCT por defecto se realizará ALL. Nunca se debe usar DISTINCT en la clave primaria, ni en las candidatas ya que de por sí son únicas

WHERE se utiliza para generar una condición que debe cumplirse, aquellos campos que no la cumplan no se seleccionarán.

GROUP BY para agrupar en una sola línea el campo o campos especificados.

HAVING del resultado de GROUP BY elimina las que no cumplan su expresión condicional.

ORDER BY para ordenar el orden de las columnas de formas ascendente (ASC) o descendente (DESC). Por defecto el orden se hace de forma ascendente.

Además de la sintaxis, podemos encontrarnos con consultas que utilicen las siguientes funciones de agregado: COUNT, SUM, AVG, MAX, MIN.

Operadores lógicos: AND, OR, NOT.

Operadores de comparación: >, <, , =, =, !=, !, BETWEEN para intervalo de valores, LIKE para comparación entre cadenas de caracteres con los pattern '%' y '_'.

IN para especificar registros de una base de datos.

Para ver un ejemplo, vamos a crear una nueva base de datos llamada VideoJuegos, para ello vamos a utilizar las siguientes tablas:

CONSOLA

ID_CONSOLA	NOMBRE	TIPO	MARCA
00001	PS4	Sobremesa	Sony
00002	XBOX ONE	Sobremesa	Microsoft
00003	3DS	Portátil	Nintendo

JUEGO

ID_JUEGO	TITULO	GENERO
0001	Darks Souls III	ROL
0002	Project Cars	Conducción
0003	Metal Gear Solid V	Acción
0004	Killer Instinct	Lucha
0005	Metroid Prime	Acción
0006	Dragon Quest VII	ROL

CONSOLA_JUEGO

ID_CONSOLA	ID_JUEGO	STOCK
0001	0001	45
0001	0002	30
0001	0003	15
0002	0003	15
0002	0004	19
0003	0005	34
0003	0006	23

Ejercicios:

- Consulta: Mostrar los tipos de consolas que hay registrados

```
SELECT DISTINCT tipo FROM consola
```

- Nombre de los juegos que sólo están en 3DS

```
SELECT titulo FROM juego
WHERE juego.id_juego IN(
SELECT id_juego
FROM consola_juego
WHERE consola_juego.id_consola IN(
SELECT id_consola
FROM consola
WHERE consola.nombre='3DS')
)
```

- Stock total de aquellos juegos cuyo género sea de Acción

```
SELECT SUM(consola_juego.stock)
FROM consola_juego
WHERE consola_juego.id_juego IN(
SELECT id_juego
FROM juego
WHERE juego.genero='accion')
```

- Título de aquellos juegos con stock total que supere las 20 unidades

```
SELECT juego.titulo
FROM juego
WHERE juego.id_juego IN(
SELECT id_juego
FROM consola_juego
GROUP BY id_juego
HAVING SUM(consola_juego.stock) > 20)
```

-