

# Tablas en base de datos

Las tablas son objetos de base de datos que contienen todos sus datos y se organizan como un arreglo a un formato de filas y columnas. 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 estos, como el número de empleado, el nombre, la dirección, el puesto que ocupa y su número de teléfono particular.

En este documento usted encontrará:

Tipos de tablas

Creación y manipulación de tablas

Sintaxis para crear una tabla en SQL Server

Tipos de datos

Crear tablas con el Diseñador de tablas

Usar Transact-SQL para la creación de tablas

Modificación de una tabla

Eliminación de una tabla

Restricciones de integridad

Índices

Relaciones entre tablas

Establecer un campo como autonumérico o identity

Ejercicio propuesto

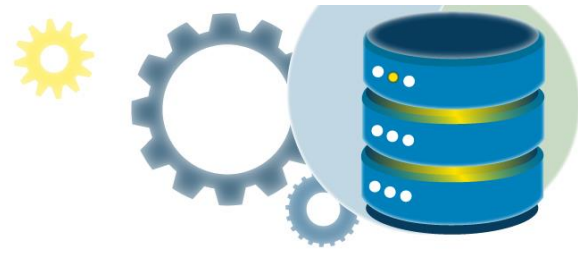
Solución

Referencias



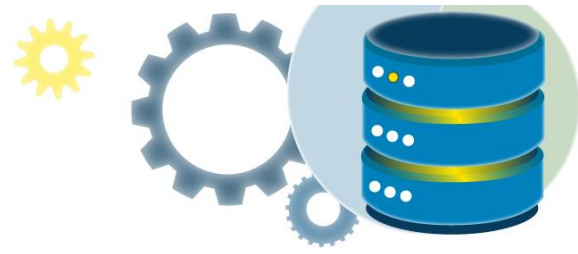
Instituto  
Nacional de  
Aprendizaje

Llave del Progreso



## MANIPULACIÓN DE DATOS CON SQL





## Tipos de tablas

Además del rol estándar de las tablas básicas definidas por el usuario, SQL Server proporciona los siguientes tipos de tabla que permiten llevar a cabo objetivos especiales en una base de datos:

### Tablas con particiones

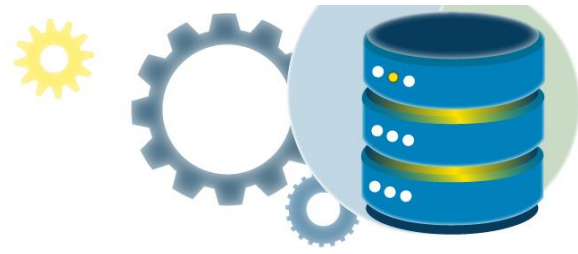
Las tablas con particiones son tablas cuyos datos se han dividido horizontalmente entre unidades que pueden repartirse por más de un grupo de archivos de una base de datos. Las particiones facilitan la administración de índices y tablas grandes al permitir el acceso y la administración de subconjuntos de datos rápidamente y con eficacia, mientras se mantiene la integridad de la colección global. De forma predeterminada, SQL Server admite hasta 15.000 particiones.

### Tablas temporales

Las tablas temporales se almacenan en tempdb. Hay dos tipos de tablas temporales: locales y globales. Se diferencian entre sí por los nombres, la visibilidad y la disponibilidad. Las tablas temporales locales tienen como primer carácter de sus nombres un solo signo de número (#); solo son visibles para el usuario de la conexión actual y se eliminan cuando el usuario se desconecta de la instancia de SQL Server. Las tablas temporales globales presentan dos signos de número (##) antes del nombre; son visibles para cualquier usuario después de su creación y se eliminan cuando todos los usuarios que hacen referencia a la tabla se desconectan de la instancia de SQL Server.

### Tablas del sistema

SQL Server almacena los datos que definen la configuración del servidor y de todas sus tablas en un conjunto de tablas especial, conocido como tablas del sistema. Los usuarios no pueden consultar o actualizar directamente las tablas del sistema. La información de las tablas del sistema está disponible a través de las vistas del sistema.



## Tablas anchas

Las tablas anchas usan las columnas dispersas para aumentar hasta 30 000 el número total de columnas permitidas. Las columnas dispersas son columnas normales que disponen de un almacenamiento optimizado para los valores NULL. Este tipo de columnas reducen los requisitos de espacio de los valores NULL a costa de una mayor sobrecarga a la hora de recuperar valores no NULL.

## Creación y manipulación de tablas

Las tablas se componen de dos estructuras:

- **Campo:** Corresponde al nombre de la columna. Debe ser único y además de tener un tipo de dato asociado a los registros que se insertaran.
- **Registro:** Corresponde a cada fila que compone la tabla. Allí se componen los datos y los registros. Eventualmente pueden ser nulos en su almacenamiento.

Cada registro contiene un dato por cada columna de la tabla.

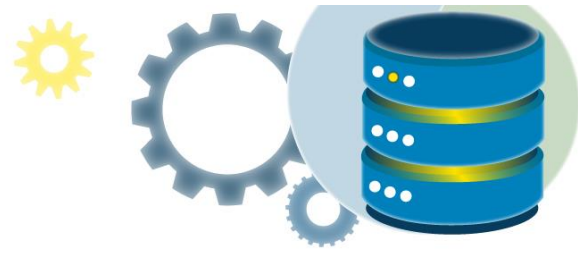
Cada campo (columna) debe tener un nombre. El nombre del campo hace referencia a la información que almacenará, por ejemplo: Nombre, para almacenar el nombre de una persona.

Cada campo (columna) también debe definir el tipo de dato que almacenará.

La tarea de creación de tablas requiere el permiso CREATE TABLE en la base de datos y el permiso ALTER en el esquema en que se crea la tabla.

## Sintaxis para crear una tabla en SQL Server

La sentencia **CREATE TABLE** se utiliza para crear una tabla en una base de datos existente.



```
CREATE TABLE nombre_tabla  
(  
  nombrecolumna1 tipodato1,  
  nombrecolumna2 tipodato2,  
  nombrecolumna3 tipodato3,  
  ..  
);
```

**CREATE TABLE:** este comando se utiliza para indicar que una tabla se va a crear, seguidamente especificamos el nombre de la tabla.

Los paréntesis () indican que dentro de ellas se agregaran las columnas de la tabla indicando el inicio y fin de la estructura, las columnas van con la siguiente estructura:

### Resumen de propiedades de las columnas de tablas

**Permitir valores NULL:** Por defecto si no se especifica NOT NULL, la columna podrá soportar nulos.

**Primary Key:** Indica si la columna será la llave primaria de la tabla.

**Unique:** Indica que la columna permitirá valores únicos, no se puede repetir un mismo valor en varias filas (row).

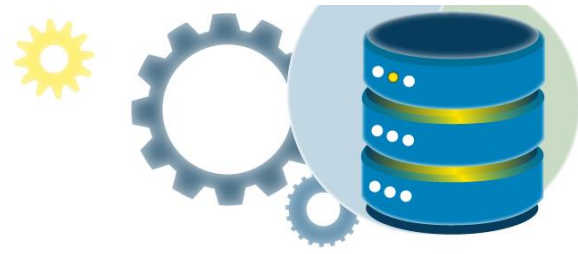
**Default:** Se indica el valor default que tendrá la columna cuando en la inserción no este contemplado este campo.

**Identity:** La columna será auto incrementable por defecto inicia en 1.

**Description:** Una breve descripción a la columna.

### Tipos de datos

En SQL Server, cada columna, variable local, expresión y parámetro tiene un tipo de dato relacionado. Un tipo de dato es un atributo que especifica qué valor puede contener un



atributo: datos de enteros, datos de caracteres, datos de moneda, datos de fecha y hora, cadenas binarias, etc.

## Categorías de tipos de datos

Los tipos de datos de SQL Server se organizan en las siguientes categorías:

- Numéricos exactos
- Cadenas de caracteres Unicode
- Numéricos aproximados
- Cadenas binarias
- Fecha y hora
- Otros tipos de datos
- Cadenas de caracteres

En SQL Server, según las características de almacenamiento, algunos tipos de datos están designados como pertenecientes a los siguientes grupos:

- Tipos de datos de valores grandes: **varchar(max)** y **nvarchar(max)**.
- Tipos de datos de objetos grandes: **text**, **ntext**, **imEdad**, **varbinary(max)** y **xml**.

### *Numéricos exactos*

[bigint](#)

[numeric](#)

[bit](#)

[smallint](#)

[decimal](#)

[smallmoney](#)

[int](#)

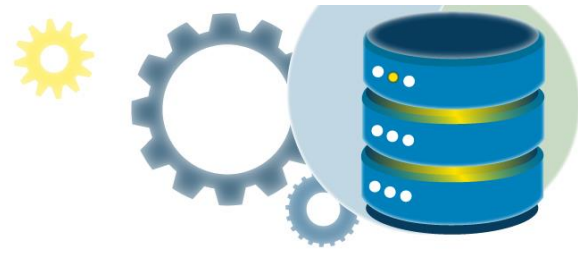
[tinyint](#)

[money](#)

### *Numéricos aproximados*

[float](#)

[real](#)



*Fecha y hora*

[date](#)  
[datetimeoffset](#)  
[datetime2](#)  
[smalldatetime](#)  
[datetime](#)  
[time](#)

*Cadenas de caracteres*

[char](#)  
[varchar](#)  
[text](#)

*Cadenas de caracteres Unicode*

[nchar](#)  
[nvarchar](#)  
[ntext](#)

*Cadenas binarias*

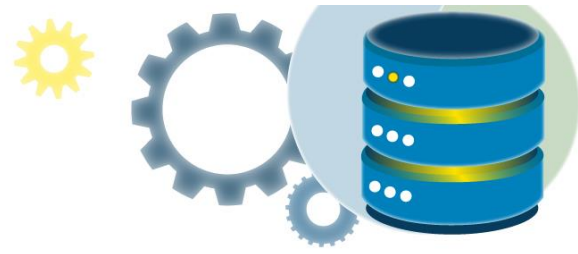
[binary](#)  
[varbinary](#)  
[imEdad](#)

*Otros tipos de datos*

[cursor](#)  
[rowversion](#)  
[hierarchyid](#)  
[uniqueidentifier](#)  
[sql\\_variant](#)  
[xml](#)  
[Tipos de geometría espacial](#)  
[Tipos de geografía espacial](#)  
[table](#)

## Crear tablas con el Diseñador de tablas

1. En SSMS, en el **Explorador de objetos**, conéctese a la instancia de Motor de base de datos que contiene la base de datos que se va a modificar.

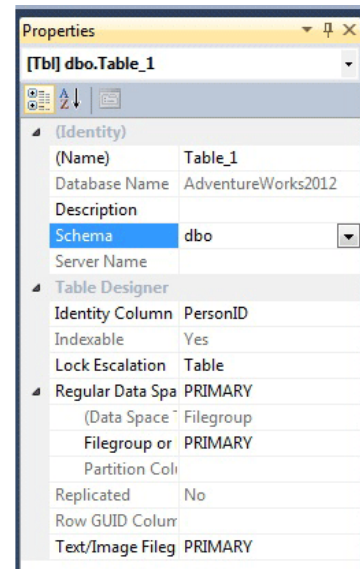


2. En el **Explorador de objetos**, expanda el nodo **Bases de datos** y, a continuación, expanda la base de datos que contendrá la nueva tabla.
3. En el Explorador de objetos, haga clic con el botón derecho en el nodo **Tablas** de la base de datos y, después, haga clic en **Nueva tabla**.
4. Escriba los nombres de columna (ID, Nombre, Apellido, FechaModificación), elija los tipos de datos y elija si desea permitir valores NULL para cada columna como se muestra en la ilustración siguiente:

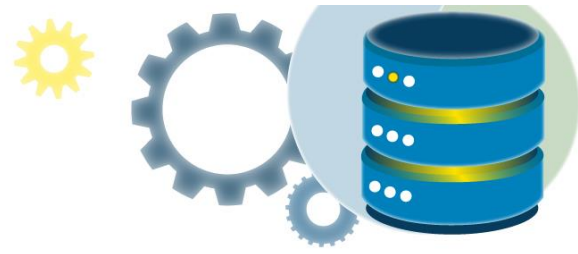
Column Name	Data Type	Allow Nulls
ID	int	<input checked="" type="checkbox"/>
Nombre	nvarchar(30)	<input checked="" type="checkbox"/>
Apellido	nvarchar(50)	<input checked="" type="checkbox"/>
FechaModificación	smalldatetime	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Más adelante veremos como especificar más propiedades para cada columna, como la identidad o valores de columna calculada, llave primaria, llave foránea, relaciones y otras restricciones.

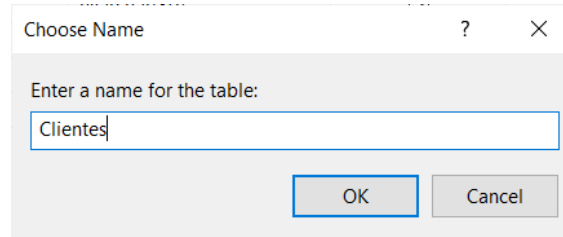
5. De forma predeterminada, la tabla está contenida en el esquema **dbo**. Para especificar un esquema diferente para la tabla, haga clic con el botón derecho en el panel Diseñador de tablas y seleccione **Propiedades** como se muestra en la ilustración siguiente. En la lista desplegable **Esquema**, seleccione el esquema adecuado.







6. En el menú **Archivo**, seleccione **Guardar** nombre de tabla.
7. En el cuadro de diálogo **Elegir nombre**, escriba un nombre para la tabla y haga clic en **Aceptar**.



8. Para ver la nueva tabla, en el **Explorador de objetos**, expanda el nodo **Tablas** y presione **F5** para actualizar la lista de objetos. La nueva tabla se mostrará en la lista de tablas.

## Usar Transact-SQL para la creación de tablas

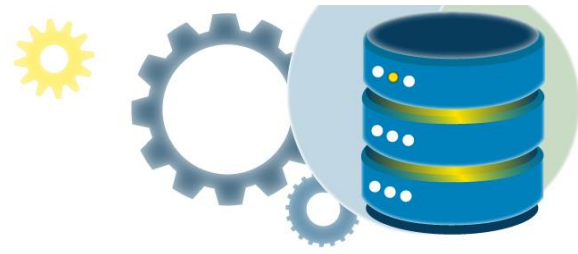
1. En el **Explorador de objetos**, conéctese a una instancia del Motor de base de datos.
2. En la barra de Estándar, haga clic en **Nueva consulta**.
3. Copie y pegue el siguiente ejemplo en la ventana de consulta y haga clic en **Ejecutar**.

```
CREATE TABLE [dbo].[Clientes](
  [ID] [int] NULL,
  [Nombre] [nvarchar](30) NULL,
  [Apellido] [nvarchar](50) NULL,
  [FechaModificación] [smalldatetime] NULL
) ON [PRIMARY]
GO
```

## Modificación de una tabla

Una vez que se crea la tabla en la base de datos, hay muchas ocasiones donde uno puede desear cambiar la estructura de la tabla. Los casos típicos incluyen los siguientes:

- Agregar una columna.



- Eliminar una columna.
- Cambiar el nombre de una columna.
- Cambiar el tipo de datos para una columna.

Por favor note que lo anterior no es una lista exhaustiva. Hay otras instancias donde **ALTER TABLE** se utiliza para cambiar la estructura de la tabla, tales como cambiar la especificación de la clave primaria o agregar una restricción única para una columna.

La sintaxis SQL para **ALTER TABLE** es la siguiente:

```
ALTER TABLE "nombre_tabla"  
[modificar especificación];
```

[**modificar especificación**] depende del tipo de modificación que deseamos realizar. Para los usos mencionados anteriormente, las instrucciones [modificar especificación] son:

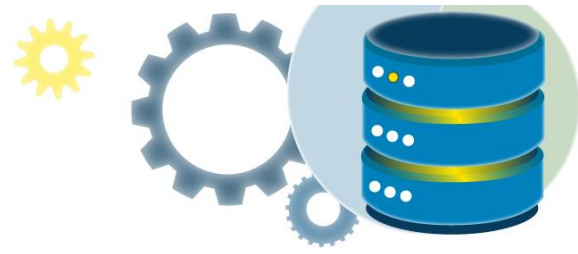
- Agregar una columna: ADD "columna 1" "tipos de datos para columna 1"
- Eliminar una columna: DROP "columna 1"
- Cambiar el nombre de una columna: CHANGE "nombre antiguo de la columna" "nuevo nombre de la columna" "tipos de datos para la nueva columna".
- Cambiar el tipo de datos para una columna: MODIFY "columna 1" "nuevo tipo de datos"

Recorramos ejemplos para cada uno de lo anteriormente mencionado, utilizando la tabla "clientes".

Tabla CLIENTES

Nombre de Columna	Tip de Datos
ID	int
Nombre	char(50)
Apellido	char(50)
FechaModificacion	char(50)

Primero, deseamos agregar una columna denominada "NoClienteFrecuente" a esta tabla. Para hacerlo, realizamos el siguiente enunciado:



```
ALTER TABLE [dbo].[Clientes]
    ADD [NoClienteFrecuente] [int] NULL
)
GO
```

Estructura de la tabla resultante:

Tabla CLIENTES

Nombre de Columna	Tip de Datos
ID	int
Nombre	char(50)
Apellido	char(50)
FechaModificacion	char(50)
NoClienteFrecuente	int

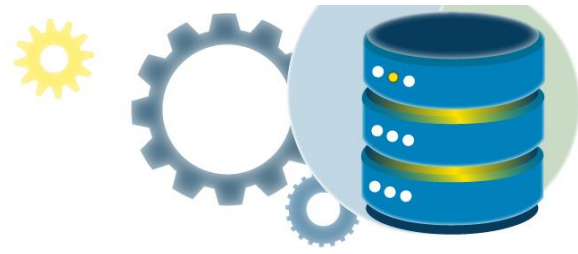
Si, deseamos eliminar la columna “FechaModificacion”. Para hacerlo, ingresamos,

```
ALTER TABLE [dbo].[Clientes]
    DROP [FechaModificacion] [int] NULL
)
GO
```

Estructura de la tabla resultante:

Tabla CLIENTES

Nombre de Columna	Tip de Datos
ID	int
Nombre	char(50)
Apellido	char(50)
NoClienteFrecuente	int



## Eliminación de una tabla

A veces podemos decidir que necesitamos eliminar una tabla en la base de datos por alguna razón. De hecho, sería problemático si no podemos hacerlo ya que esto crearía una pesadilla de mantenimiento para DBA. Afortunadamente, SQL nos permite hacerlo, ya que podemos utilizar el comando **DROP TABLE**. La sintaxis para **DROP TABLE** es

```
DROP TABLE [dbo].[Clientes]
GO
```

## Restricciones de integridad

Las restricciones de integridad de SQL se utilizan para limitar el tipo de datos que pueden incluirse en una tabla. Esto asegura la precisión y confiabilidad de los datos en la tabla. Si hay alguna violación entre la restricción y la acción de datos, la acción se cancela.

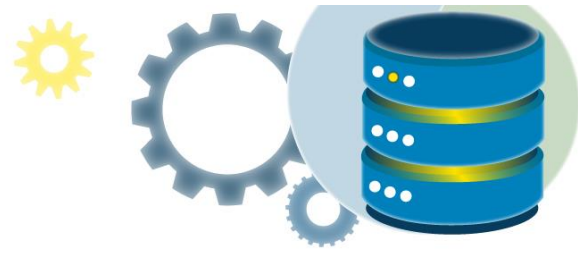
Las restricciones pueden ser de nivel de columna o de tabla. Las restricciones de nivel de columna se aplican a una columna y las restricciones de nivel de tabla se aplican a toda la tabla.

Las siguientes restricciones se usan comúnmente en SQL:

- [NOT NULL](#): Asegura que una columna no pueda tener un valor NULO.
- [UNIQUE](#): Asegura que todos los valores en una columna sean diferentes
- [PRIMARY KEY](#): Es la llave primaria que identifica de forma única cada fila o registro de la tabla. Implica una combinación de una **NOT NULL** y **UNIQUE**.
- [FOREIGN KEY](#): Evita acciones que destruirían enlaces entre tablas. Especifica la integridad referencial, es decir, cómo se relacionan las tablas y sus datos entre sí.
- [CHECK](#): Asegura que los valores en una columna satisfagan una condición específica.
- [DEFAULT](#): Establece un valor predeterminado para una columna si no se especifica ningún valor.
- [INDEX](#): Se utiliza para crear y recuperar datos de la base de datos muy rápidamente.

### PRIMARY KEY

Una llave primaria identifica un único registro en una tabla. Deben contener valores únicos y no pueden contener valores NULOS.



Una tabla solo puede tener UNA clave primaria; y en la tabla, esta clave principal puede constar de una o varias columnas (campos).

### *Uso de SQL Server Management Studio para crear una clave principal*

1. En el Explorador de objetos, haga clic con el botón derecho en la tabla a la que quiere agregar una restricción UNIQUE y haga clic en **Diseño**.
2. En el **Diseñador de tablas**, haga clic en el selector de filas de la columna de base de datos que desea definir como clave principal. Si desea seleccionar varias columnas, mantenga presionada la tecla CTRL mientras hace clic en los selectores de fila de las otras columnas.
3. Haga clic con el botón derecho en el selector de fila para la columna y seleccione **Establecer clave principal (Set primary key)**.

### *Usar Transact-SQL para crear una llave primaria*

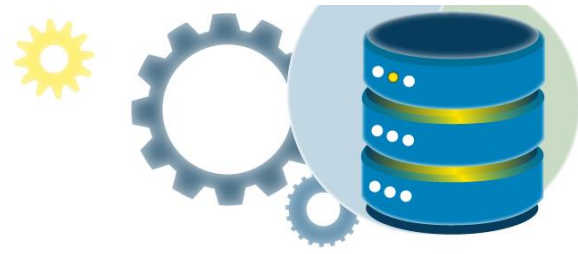
El siguiente SQL crea un **PRIMARY KEY** en la columna "ID" cuando se crea la tabla "Clientes":

```
CREATE TABLE Clientes (  
    ID int NOT NULL PRIMARY KEY,  
    Apellido varchar(255) NOT NULL,  
    Nombre varchar(255),  
    Edad int  
);
```

Para crear una **PRIMARY KEY** en la columna "ID" cuando la tabla ya está creada, use el siguiente SQL:

```
ALTER TABLE Clientes  
ADD PRIMARY KEY (ID);
```

En caso de que desee que la llave primaria esté compuesta por varios campos puede establecerlo de la siguiente forma:



```
ALTER TABLE Clientes  
ADD CONSTRAINT PK_Person PRIMARY KEY (ID,Apellido);
```

**Nota:** Si usa **ALTER TABLE** para agregar una clave primaria, las columnas de la clave primaria deben haber sido declaradas para no contener valores NULL (cuando se creó la tabla por primera vez).

*Uso de SQL Server Management Studio para eliminar una restricción de clave principal mediante el Explorador de objetos*

1. En el Explorador de objetos, expanda la tabla que contiene la clave principal y, a continuación, expanda **Claves (keys)**.
2. Haga clic con el botón derecho en la clave y seleccione **Eliminar**.
3. En el cuadro de diálogo **Eliminar objeto**, compruebe que se ha especificado la clave correcta y haga clic en **Aceptar**.

*Para eliminar una restricción de clave principal mediante el Diseñador de tablas*

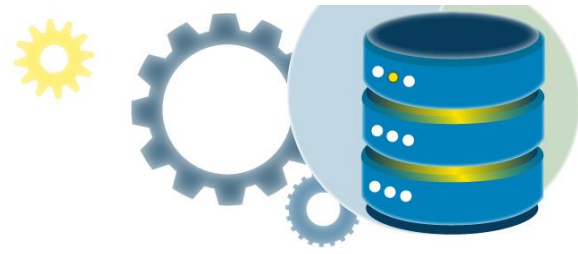
1. En el Explorador de objetos, haga clic con el botón derecho en la tabla con la clave principal y, después, haga clic en **Diseño**.
2. En la cuadrícula de la tabla, haga clic con el botón derecho en la fila que contiene la clave principal y elija **Quitar clave principal** para desactivar el valor.

**Nota:** Para deshacer esta acción, cierre la tabla sin guardar los cambios. Si se elimina una clave principal, no se podrá deshacer la acción sin perder todos los demás cambios realizados en la tabla.

3. En el menú **Archivo**, haga clic en **\*Guardar nombre de tabla**.

*Usar Transact-SQL para eliminar una llave primaria*

```
ALTER TABLE Clientes  
DROP CONSTRAINT PK_Person;
```



## NOT NULL

Por defecto una columna en una tabla puede aceptar valores NULOS.

La restricción NOT NULL, fuerza a la columna a no aceptar valores nulos. Esto se asegura de que el campo tenga un valor y no puede insertar o actualizar un registro si dicho campo no tiene un valor diferente de NULL.

En el siguiente ejemplo, se asegura de que los campos ID, Apellido y Nombre no acepten valores nulos cuando tabla Clientes es creada:

```
CREATE TABLE Clientes (  
    ID int NOT NULL,  
    Apellido varchar(255) NOT NULL,  
    Nombre varchar(255) NOT NULL,  
    Edad int  
);
```

Para crear una restricción NOT NULL en una tabla ya existente, podemos ejecutar la instrucción ALTER TABLE. Por ejemplo, deseamos que el campo edad en la tabla CLIENTES no contenga un valor nulo:

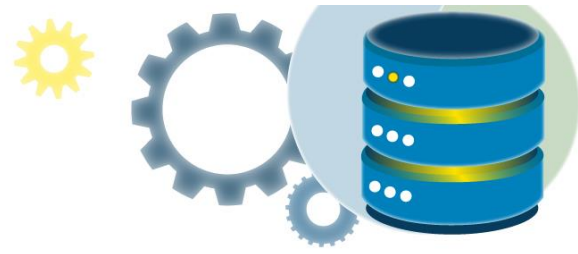
```
ALTER TABLE Clientes  
MODIFY edad int NOT NULL;
```

## UNIQUE

La restricción UNIQUE asegura que todos los valores de una columna sean diferentes.

Tanto las restricciones UNIQUE y PRIMARY KEY proporcionan una garantía de la unicidad para una columna o conjunto de columnas. Una llave primaria tiene automáticamente una restricción UNIQUE.

Sin embargo, puede tener muchas restricciones UNIQUE por tabla, pero solo una llave primaria por tabla.



### Uso de SQL Server Management Studio

1. En el **Explorador de objetos**, haga clic con el botón derecho en la tabla a la que quiera agregar una restricción UNIQUE y haga clic en **Diseño**.
2. En el menú **Diseñador de tablas**, haga clic en **Índices o claves**.
3. En el cuadro de diálogo **Índices o claves**, haga clic en **Agregar**.
4. En la cuadrícula situada debajo de **General**, haga clic en **Tipo** y elija **Clave UNIQUE** en el cuadro de lista desplegable situado a la derecha de la propiedad y, a continuación, haga clic en **Cerrar**.
5. En el menú **Archivo**, haga clic en **\*Guardar nombre de tabla**.

### Usar Transact-SQL

El siguiente SQL crea una restricción **UNIQUE** en la columna "ID" cuando se crea la tabla "Clientes":

```
CREATE TABLE Clientes (  
    ID int NOT NULL UNIQUE,  
    Apellido varchar(255) NOT NULL,  
    Nombre varchar(255),  
    Edad int  
);
```

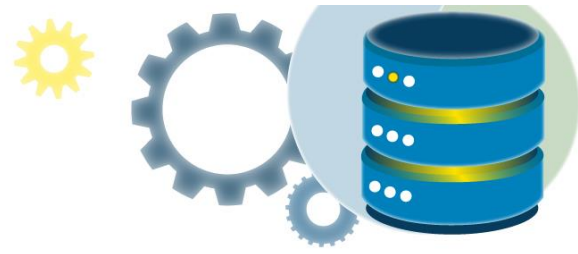
Para nombrar una restricción UNIQUE y definir en varias columnas, use la siguiente sintaxis SQL:

```
CREATE TABLE Clientes (  
    ID int NOT NULL,  
    Apellido varchar(255) NOT NULL,  
    Nombre varchar(255),  
    Edad int,  
    CONSTRAINT UC_Person UNIQUE (ID,Apellido)  
);
```

Para crear una restricción UNIQUE en la columna "ID" cuando la tabla ya está creada, use el siguiente SQL:

```
ALTER TABLE Clientes  
ADD UNIQUE (ID);
```





Para nombrar una restricción UNIQUE y definir una restricción en varias columnas, use la siguiente sintaxis SQL:

```
ALTER TABLE Clientes  
ADD CONSTRAINT UC_Person UNIQUE (ID,Apellido);
```

Para eliminar una restricción UNIQUE, use el siguiente SQL:

```
ALTER TABLE Clientes  
DROP CONSTRAINT UC_Person;
```

## CHECK

La restricción CHECK se utiliza para limitar el rango de valores que se puede colocar en una columna.

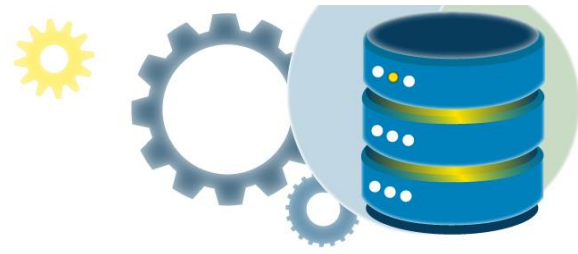
*Uso de SQL Server Management Studio para crear una restricción CHECK nueva*

1. En el **Explorador de objetos**, expanda la tabla a la que quiera agregar una restricción CHECK , haga clic con el botón derecho en **Restricciones** y haga clic en **Nueva restricción**.
2. En el cuadro de diálogo **Comprobar restricciones**, haga clic en el campo **Expresión** y luego en los puntos suspensivos(...).
3. En el cuadro de diálogo **Expresión de restricción CHECK**, escriba expresiones SQL para la restricción CHECK. Por ejemplo, para limitar las entradas de la columna SellEndDate de la tabla Product a un valor que sea mayor o igual que la fecha de la columna SellStartDate o que sea un valor NULL, escriba:

```
SellEndDate >= SellStartDate OR SellEndDate IS NULL
```

O bien, para exigir que las entradas que se escriben en la columna zip tengan 5 dígitos, escriba:

```
zip LIKE '[0-9][0-9][0-9][0-9][0-9]'
```



**Nota:** Asegúrese de que escribe los valores de restricción no numéricos entre comillas sencillas (').

4. Haga clic en **Aceptar**.
5. En la categoría **Identidad**, puede cambiar el nombre de la restricción CHECK y agregar una descripción (propiedad extendida) para la restricción.
6. En la categoría **Diseñador de tablas**, puede definir cuándo debe exigirse la restricción.
7. Haga clic en **Cerrar**.

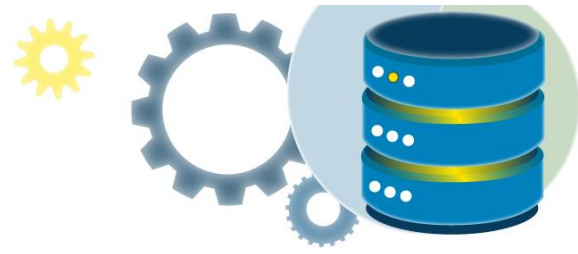
### Usar Transact-SQL

El siguiente SQL crea una restricción CHECK en la columna "Edad" cuando se crea la tabla "Clientes". La restricción CHECK asegura que la edad de una persona debe ser mayor de 18 años:

```
CREATE TABLE Clientes (  
    ID int NOT NULL,  
    Apellido varchar(255) NOT NULL,  
    Nombre varchar(255),  
    Edad int CHECK (Edad>=18)  
);
```

Para permitir el nombramiento de una restricción CHECK y para definir una restricción CHECK en varias columnas, utilice la siguiente sintaxis SQL:

```
CREATE TABLE Clientes (  
    ID int NOT NULL,  
    Apellido varchar(255) NOT NULL,  
    Nombre varchar(255),  
    Edad int,  
    City varchar(255),  
    CONSTRAINT CHK_Person CHECK (Edad>=18 AND City='Sandnes')  
);
```



Para crear una restricción CHECK en la columna "Edad" cuando la tabla ya está creada, use el siguiente SQL:

```
ALTER TABLE Clientes  
ADD CHECK (Edad>=18);
```

Para permitir el nombramiento de una restricción CHECK y para definir una restricción CHECK en varias columnas, utilice la siguiente sintaxis SQL:

```
ALTER TABLE Clientes  
ADD CONSTRAINT CHK_PersonEdad CHECK (Edad>=18 AND City='Sandnes');
```

Para eliminar una restricción CHECK, use el siguiente SQL:

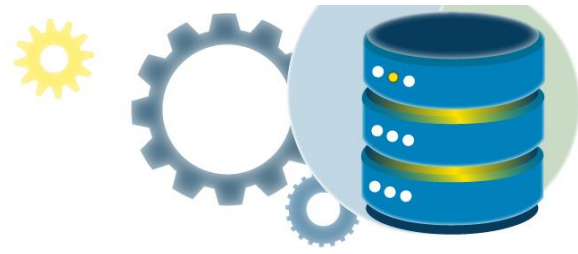
```
ALTER TABLE Clientes  
DROP CONSTRAINT CHK_PersonEdad;
```

## DEFAULT

La restricción DEFAULT se utiliza para establecer un valor predeterminado para una columna. El valor predeterminado se agregará a todos los registros nuevos, si no se especifica ningún otro valor.

El siguiente SQL establece un valor por defecto para la columna "Ciudad" cuando se crea la tabla "Clientes":

```
CREATE TABLE Clientes (  
    ID int NOT NULL,  
    Apellido varchar(255) NOT NULL,  
    Nombre varchar(255),  
    Edad int,  
    City varchar(255) DEFAULT 'San José'  
);
```



La restricción DEFAULT también se puede usar para insertar valores del sistema, usando funciones como GETDATE()

```
CREATE TABLE Ordenes (  
    ID int NOT NULL,  
    OrderNumber int NOT NULL,  
    OrderDate date DEFAULT GETDATE()  
);
```

Para crear una restricción DEFAULT en la columna "Ciudad" cuando la tabla ya está creada, use el siguiente SQL:

```
ALTER TABLE Clientes  
ADD CONSTRAINT df_Ciudad  
DEFAULT 'San José' FOR Ciudad;
```

Para eliminar una restricción DEFAULT, use el siguiente SQL:

```
ALTER TABLE Clientes  
ALTER COLUMN Ciudad DROP DEFAULT;
```

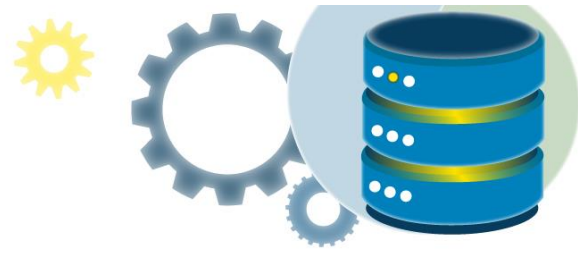
## Índices

Los índices se utilizan para recuperar datos de la base de datos más rápidamente que de otra manera. Los usuarios no pueden ver los índices, solo se utilizan para acelerar las búsquedas / consultas.

**Nota:** Actualizar una tabla con índices lleva más tiempo que actualizar una tabla sin ellos (porque los índices también necesitan una actualización). Por lo tanto, solo cree índices en columnas en las que se buscarán con frecuencia.

### Sintaxis CREATE INDEX

Crea un índice en una tabla. Se permiten valores duplicados:



```
CREATE INDEX index_name  
ON table_name (column1, column2, ...);
```

### Sintaxis CREATE UNIQUE INDEX

Crea un índice único en una tabla. No se permiten valores duplicados:

```
CREATE UNIQUE INDEX index_name  
ON table_name (column1, column2, ...);
```

**Nota:** la sintaxis para crear índices varía entre las diferentes bases de datos. Por lo tanto, verifique la sintaxis para crear índices en su base de datos.

### Ejemplo de CREAR ÍNDICE

La siguiente declaración SQL crea un índice llamado "idx\_Apellido" en la columna "Apellido" en la tabla "Clientes":

```
CREATE INDEX idx_Apellido  
ON Clientes (Apellido);
```

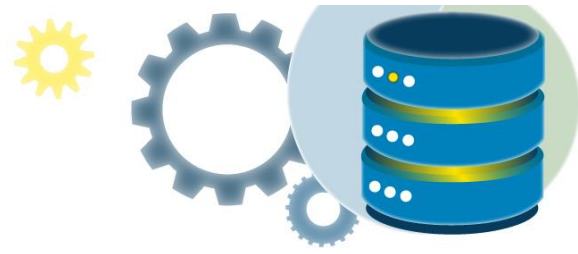
Si desea crear un índice en una combinación de columnas, puede enumerar los nombres de las columnas entre paréntesis, separados por comas:

```
CREATE INDEX idx_pname  
ON Clientes (Apellido, Nombre);
```

### Declaración DROP INDEX

**DROP INDEX** se usa para eliminar un índice en una tabla.

```
DROP INDEX table_name.index_name;
```



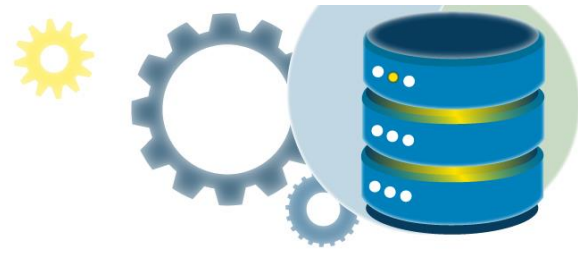
## Relaciones entre tablas

### Límites y restricciones

- Una restricción de clave externa no tiene que estar vinculada solo a una restricción de clave principal en otra tabla. Las claves externas también se pueden definir para hacer referencia a las columnas de una restricción UNIQUE de otra tabla.
- Si se especifica un valor distinto de NULL en la columna de una restricción FOREIGN KEY, el valor debe existir en la columna a la que se hace referencia. De lo contrario, se devuelve un mensaje de error de infracción de clave externa. Para asegurarse de que todos los valores de la restricción de clave externa compuesta se comprueben, especifique NOT NULL en todas las columnas que participan.
- Las restricciones FOREIGN KEY solo pueden hacer referencia a las tablas de la misma base de datos en el mismo servidor. La integridad referencial entre bases de datos debe implementarse a través de desencadenadores. Para obtener más información, vea [CREATE TRIGGER](#).
- Las restricciones FOREIGN KEY pueden hacer referencia a otra columna de la misma tabla, lo que se conoce como autorreferencia.
- Una restricción FOREIGN KEY especificada en el nivel de columna solo puede incluir una columna de referencia. Esta columna debe tener el mismo tipo de datos que la columna en la que se define la restricción.
- Una restricción FOREIGN KEY especificada en el nivel de tabla debe tener el mismo número de columnas de referencia que la lista de columnas de la restricción. El tipo de datos de cada columna de referencia debe ser también el mismo que el de la columna correspondiente de la lista de columnas.
- El Motor de base de datos no tiene un límite predefinido en el número de restricciones FOREIGN KEY que puede contener una tabla y que hacen referencia a otras tablas. El Motor de base de datos tampoco limita el número de restricciones FOREIGN KEY que pertenecen a otras tablas y que hacen referencia a una tabla concreta.
- Las restricciones FOREIGN KEY no se exigen en tablas temporales.
- Una columna de tipo **varchar(max)** solo puede participar en una restricción FOREIGN KEY si la clave principal a la que hace referencia se define también como tipo **varchar(max)**.

### Uso de SQL Server Management Studio para crear llaves foráneas

1. En el Explorador de objetos, haga clic con el botón derecho en la tabla que va a estar en el lado de la clave externa de la relación y, después, haga clic en **Diseño**.



La tabla se abre en el [Diseñador de tablas](#).

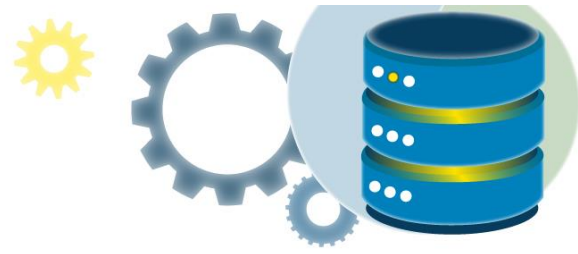
2. En el menú **Diseñador de tablas** , haga clic en **Relaciones**.
3. En el cuadro de diálogo **Relaciones de clave externa** , haga clic en **Agregar**.

La relación aparece en la lista **Relación seleccionada** con un nombre proporcionado por el sistema con el formato `FK_<tablename>_<tablename>`, donde *tablename* es el nombre de la tabla de clave externa.

4. Haga clic en la relación en la lista **Relación seleccionada** .
5. Haga clic en **Especificaciones de tablas y columnas** en la cuadrícula situada a la derecha y, después, haga clic en los puntos suspensivos ( ... ) situados a la derecha de la propiedad.
6. En el cuadro de diálogo **Tablas y columnas** , en la lista desplegable **Clave principal** , elija la tabla que estará en el lado de la clave principal de la relación.
7. En la cuadrícula situada debajo, elija las columnas que contribuyen a la clave principal de la tabla. En la celda de la cuadrícula adyacente situada a la derecha de cada columna, elija la columna de clave externa correspondiente de la tabla de clave externa.

El **Diseñador de tablas** sugerirá un nombre para la relación. Para cambiar este nombre, edite el contenido del cuadro de texto **Nombre de la relación** .

8. Elija **Aceptar** para crear la relación.
9. Cierre la ventana del diseñador de tablas y **guarde** los cambios para que el cambio de relación de clave externa surta efecto.



## Usar Transact-SQL en la creación de FOREIGN KEY

A **FOREIGN KEY** o llave foránea, es un campo (o colección de campos) en una tabla, que hace referencia a **PRIMARY KEY** en otra tabla.

La tabla con la clave externa se denomina tabla secundaria y la tabla con la clave principal se denomina tabla principal o referenciada.

Mire las dos tablas siguientes:

### Tabla de Clientes

ID	Apellido	Nombre	Edad
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

### Tabla de pedidos

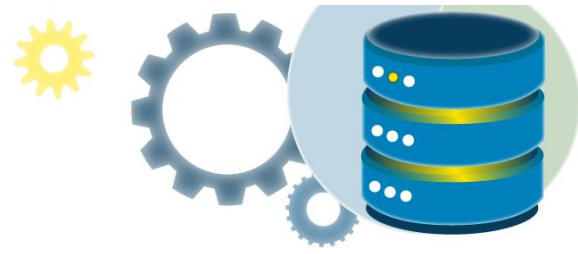
OrderID	OrderNumber	IDCliente
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Observe que la columna "IDCliente" de la tabla "Pedidos" apunta a la columna "ID" de la tabla "Clientes".

La columna "ID" de la tabla "Clientes" es la **PRIMARY KEY** de la tabla "Clientes".

La columna "IDCliente" de la tabla "Pedidos" es una **FOREIGN KEY** en la tabla "Pedidos".





Una **FOREIGN KEY** evita que se inserten datos no válidos en la columna de clave externa, porque tiene que ser uno de los valores contenidos en la tabla principal. Es decir, en el ejemplo anterior, no se puede crear un pedido para un cliente que no existe en la tabla de clientes.

Además, una **FOREIGN KEY** crea integridad referencial y en caso de que existan pedidos asociados a un cliente, no puede borrar los datos de dicho cliente de la tabla Clientes.

#### *SQL FOREIGN KEY en CREATE TABLE*

El siguiente SQL crea un **FOREIGN KEY** en la columna "IDCliente" cuando se crea la tabla "Pedidos":

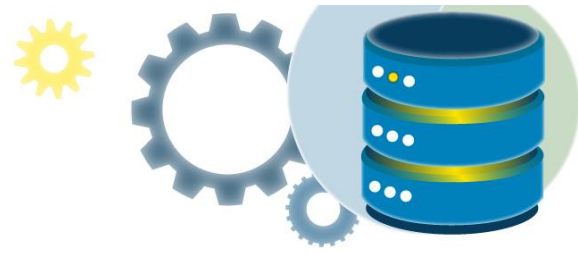
```
CREATE TABLE Pedidos (  
    OrderID int NOT NULL PRIMARY KEY,  
    OrderNumber int NOT NULL,  
    IDCliente int FOREIGN KEY REFERENCES Clientes(ID)  
);
```

Para permitir el nombramiento de una **FOREIGN KEY** y para definirla en varias columnas, utilice la siguiente sintaxis SQL:

```
CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    ID int,  
    PRIMARY KEY (OrderID),  
    CONSTRAINT FK_PersonOrder FOREIGN KEY (ID)  
    REFERENCES Clientes(ID)  
);
```

#### *SQL FOREIGN KEY en ALTER TABLE*

Para crear una **FOREIGN KEY** en la columna "ID" cuando la tabla "Pedidos" ya está creada, use el siguiente SQL:



```
ALTER TABLE Pedidos  
ADD CONSTRAINT FK_PersonOrder  
FOREIGN KEY (IDCliente) REFERENCES Clientes(ID);
```

#### Eliminar una llave foránea

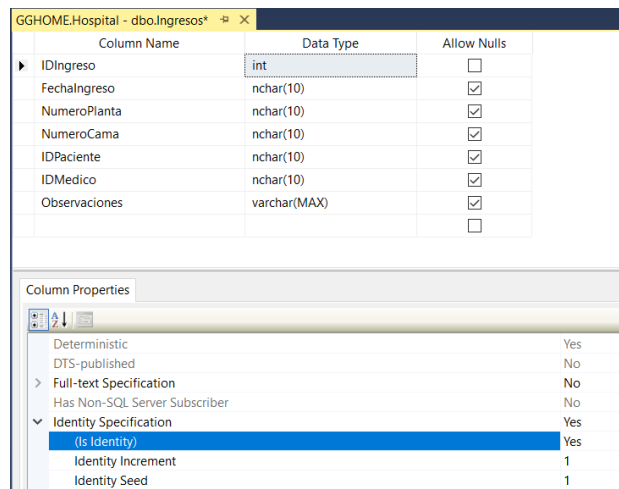
Para eliminar una **FOREIGN KEY**, use el siguiente SQL:

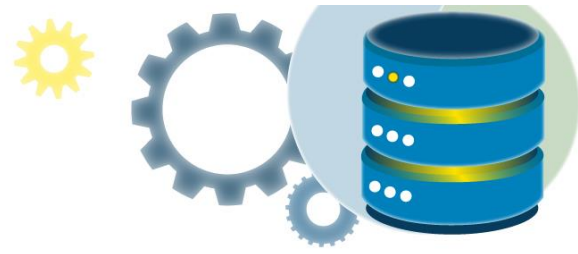
```
ALTER TABLE Orders  
DROP CONSTRAINT FK_PersonOrder;
```

#### Establecer un campo como autonumérico o identity

El incremento automático permite generar automáticamente un número único cuando se inserta un nuevo registro en una tabla. A menudo, este es el campo de clave principal que nos gustaría que se creara automáticamente cada vez que se inserta un nuevo registro.

Para establecer un campo como autonumérico, seleccionamos el campo y en el Diseñador de tablas, podemos crear y editar la tabla, seleccionamos el campo que deseamos establecer como autonumérico y en las propiedades de la columna, variamos la propiedad Especificación de Identidad o Identity Specification, para activarla, seleccionamos SI y digitamos los intervalos que deseamos que tenga, si queremos que incremente de uno en uno, colocamos un 1 en este campo, tal como se muestra en la siguiente imagen.





## Establecer identity con Transact-SQL

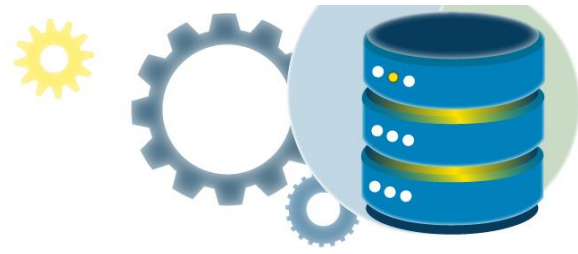
La siguiente instrucción SQL define la columna "Personid" como un campo de clave primaria de incremento automático en la tabla "Personas":

```
CREATE TABLE Persons (  
    Personid int IDENTITY(1,1) PRIMARY KEY,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int  
);
```

MS SQL Server utiliza la palabra **IDENTITY** clave para realizar una función de incremento automático.

En el ejemplo anterior, el valor inicial de **IDENTITY** es 1 y se incrementará en 1 para cada nuevo registro.

Para especificar que la columna "Personid" debe comenzar en el valor 10 y aumentar en 5, cámbielo a **IDENTITY(10,5)**.



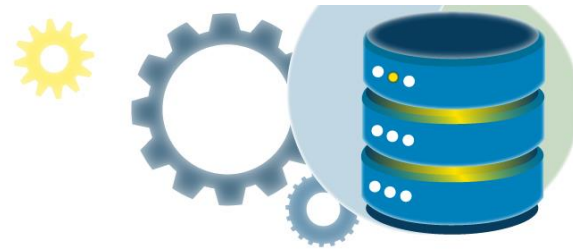
## Ejercicio propuesto

En el siguiente ejercicio, se continuará con la creación física de la base de datos y las tablas para el ejercicio del Hospital. Se han creado unos cambios en los campos de las tablas a petición de la administración del hospital.

Los campos se definen a continuación:

Estructura de la tabla Pacientes			
Nombre del campo	Tipo de campo	Descripción	Tamaño
IDSeguroSocial	Texto	Número de afiliación del paciente a la seguridad social	15
OrdenPatronal	Texto	Número de orden patronal vinculada con la afiliación del paciente	17
Nombre	Texto	Nombre del paciente	50
PrimerApellido	Texto	Primer Apellido del paciente	30
SegundoApellido	Texto	Segundo apellido del paciente	30
FechaNacimiento	Fecha	Fecha de nacimiento del paciente	
Domicilio	Texto	Domicilio del paciente	100
Telefono	numero	Telefono de contacto	8
IDProvincia	Numero	Provincia del domicilio	Entero
IDCanton	Numero	Canton del domicilio	Entero
IDDistrito	Numero	Distrito del domicilio	Entero
Observaciones	Texto	Datos referentes al paciente	MAX

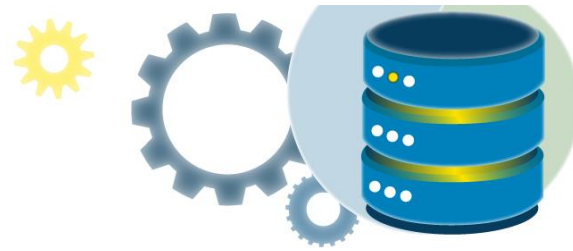
Estructura de la tabla Ingresos			
Nombre del campo	Tipo de campo	Descripción	Tamaño
IDIngreso	Autonumérico	Identificador del ingreso del paciente	Entero
FechaIngreso	Fecha	Fecha de ingreso del paciente	fecha
NumeroPlanta	Numero	Numero de planta en que ingresó el paciente	Entero



Estructura de la tabla Ingresos			
Nombre del campo	Tipo de campo	Descripción	Tamaño
NumeroCama	Numero	Numero de cama del paciente	Entero
IDPaciente	Texto	Número de afiliación del paciente a la seguridad social	15
IDMedico	Texto	Código de identificación del médico responsable	6
Diagnóstico	Texto	Diagnóstico por el que ingresa el paciente	Max
Costo del tratamiento	Moneda	Coste diario del tratamiento del paciente	
Observaciones	Texto	Datos referentes al ingreso	Max

Estructura de la tabla Médicos			
Nombre del campo	Tipo de campo	Descripción	Tamaño
Código del médico	Texto	Código de identificación del médico responsable	6
Nombre	Texto	Nombre del paciente	50
PrimerApellido	Texto	Primer Apellido del paciente	30
SegundoApellido	Texto	Segundo apellido del paciente	30
CodigoEspecialidad	Numero	Id de la especialidad del médico	entero
Cargo	Texto	Cargo del médico	30
Observaciones	Texto	Datos referentes al médico	MAX





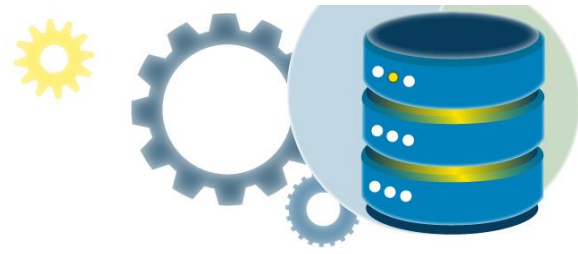
Estructura de la tabla Provincias			
Nombre del campo	Tipo de campo	Descripción	Tamaño
IDProvincia	autonumérico	Identificador de la provincia	Entero
NombreProvincia	Texto	Nombre de la provincia	30

Estructura de la tabla Cantones			
Nombre del campo	Tipo de campo	Descripción	Tamaño
IDCanton	Autonumérico	Identificador del cantón	Entero
IDProvincia	Número	Identificador de la provincia	Entero
NombreCanton	Texto	Nombre del cantón	50

Estructura de la tabla Distritos			
Nombre del campo	Tipo de campo	Descripción	Tamaño
IDDistrito	Autonumérico	Identificador de distrito	Entero
IDCanton	Numérico	Identificador de Cantón	Entero
IDProvincia	Numérico	Identificador de provincia	Entero
Nombre	Texto	Nombre del distrito	50

Estructura de la tabla Especialidades			
Nombre del campo	Tipo de campo	Descripción	Tamaño
IDEspecialidad	Autonumérico	Identificador de la especialidad	Entero
NombreEspecialidad	Texto	Nombre de la especialidad	50



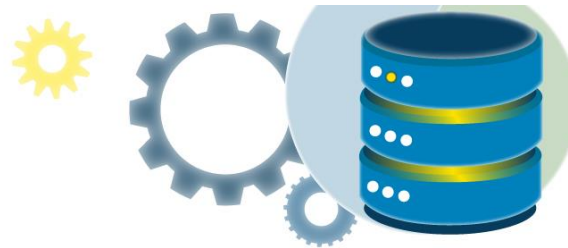


### Instrucciones:

Realice las siguientes actividades antes de revisar la solución al ejercicio, esto le preparará para los proyectos asignados.

1. Cree la base de datos con el nombre Hospital.
2. Cree las tablas de la base de datos.
3. Cree el diagrama de la base de datos y establezca las relaciones entre las tablas.
4. Genere los scripts de creación de las tablas.

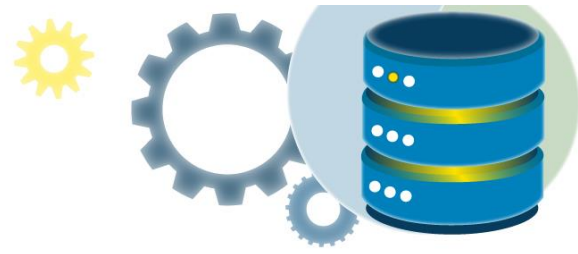
### Solución



### Diagrama de base de datos







## Scripts de creación de las tablas

### *Tabla Provincias*

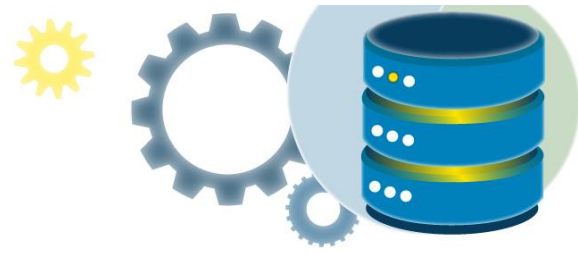
```
USE [Hospital]
GO
```

```
/****** Object: Table [dbo].[Provincias]    Script Date: 22/9/2021 06:53:31 *****/
DROP TABLE [dbo].[Provincias]
GO
```

```
/****** Object: Table [dbo].[Provincias]    Script Date: 22/9/2021 06:53:31 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Provincias](
    [IDProvincia] [int] IDENTITY(1,1) NOT NULL,
    [NombreProvincia] [varchar](30) NULL,
    CONSTRAINT [PK_Provincias] PRIMARY KEY CLUSTERED
(
    [IDProvincia] ASC
) ON [PRIMARY]
)
GO
```



*Tabla Cantones*

```
USE [Hospital]
GO
```

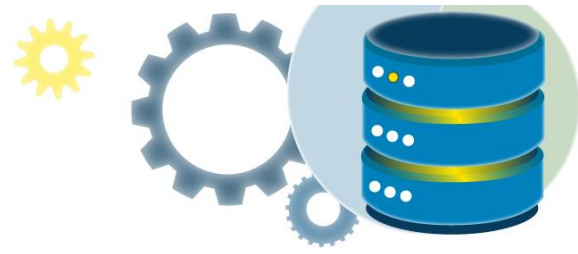
```
/****** Object: Table [dbo].[Cantones]    Script Date: 22/9/2021 07:01:04 *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Cantones](
    [IDCanton] [int] IDENTITY(1,1) NOT NULL,
    [IDProvincia] [int] NOT NULL,
    [NombreCanton] [varchar](50) NULL,
    CONSTRAINT [PK_Cantones] PRIMARY KEY CLUSTERED
(
    [IDCanton] ASC,
    [IDProvincia] ASC
) ON [PRIMARY]
)
GO
```

```
ALTER TABLE [dbo].[Cantones] WITH CHECK ADD CONSTRAINT [FK_Cantones_Provincias] FOREIGN
KEY([IDProvincia])
REFERENCES [dbo].[Provincias] ([IDProvincia])
GO
```

```
ALTER TABLE [dbo].[Cantones] CHECK CONSTRAINT [FK_Cantones_Provincias]
GO
```



*Tabla Distritos*

```
USE [Hospital]
GO
```

```
/****** Object: Table [dbo].[Distritos]    Script Date: 22/9/2021 07:01:55 *****/
```

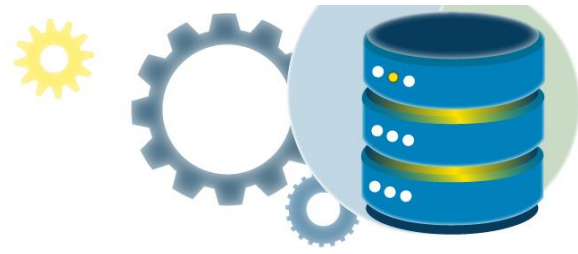
```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Distritos](
    [IDDistrito] [int] IDENTITY(1,1) NOT NULL,
    [IDCanton] [int] NOT NULL,
    [IDProvincia] [int] NOT NULL,
    [NombreDistrito] [varchar](50) NULL,
    CONSTRAINT [PK_Distritos] PRIMARY KEY CLUSTERED
(
    [IDDistrito] ASC,
    [IDCanton] ASC,
    [IDProvincia] ASC
) ON [PRIMARY]
)
GO
```

```
ALTER TABLE [dbo].[Distritos] WITH CHECK ADD CONSTRAINT [FK_Distritos_Cantones] FOREIGN
KEY([IDCanton], [IDProvincia])
REFERENCES [dbo].[Cantones] ([IDCanton], [IDProvincia])
GO
```

```
ALTER TABLE [dbo].[Distritos] CHECK CONSTRAINT [FK_Distritos_Cantones]
GO
```



*Tabla Especialidades*

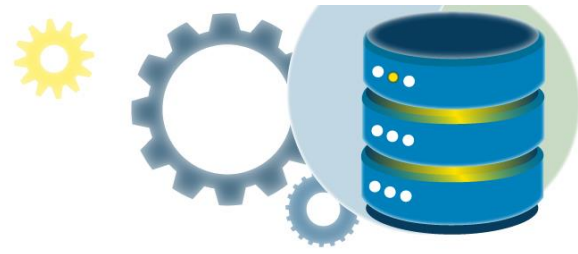
```
USE [Hospital]
GO
```

```
/****** Object: Table [dbo].[Especialidades]    Script Date: 22/9/2021 07:02:46 *****/
```

```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Especialidades](
    [IDEspecialidad] [int] IDENTITY(1,1) NOT NULL,
    [NombreEspecialidad] [varchar](50) NULL,
    CONSTRAINT [PK_Especialidades] PRIMARY KEY CLUSTERED
(
    [IDEspecialidad] ASC
) ON [PRIMARY]
)
GO
```



*Tabla Pacientes*

```
USE [Hospital]
GO
```

```
/****** Object: Table [dbo].[Pacientes]    Script Date: 22/9/2021 07:03:25 *****/
```

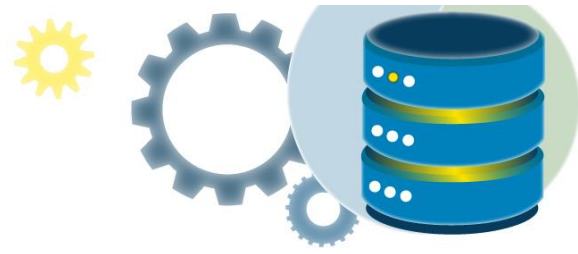
```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Pacientes](
    [IDSeguroSocial] [nchar](15) NOT NULL,
    [OrdenPatronal] [nchar](17) NULL,
    [Nombre] [nvarchar](30) NULL,
    [PrimerApellido] [nvarchar](50) NULL,
    [SegundoApellido] [nvarchar](50) NULL,
    [FechaNacimiento] [datetime] NULL,
    [Domicilio] [varchar](max) NULL,
    [Telefono] [nchar](8) NULL,
    [IDProvincia] [int] NULL,
    [IDCanton] [int] NULL,
    [IDDistrito] [int] NULL,
    [Observaciones] [varchar](max) NULL,
    CONSTRAINT [PK_Pacientes] PRIMARY KEY CLUSTERED
(
    [IDSeguroSocial] ASC
) ON [PRIMARY]
)
GO
```

```
ALTER TABLE [dbo].[Pacientes] WITH CHECK ADD CONSTRAINT [FK_Pacientes_Distritos]
FOREIGN KEY([IDDistrito], [IDCanton], [IDProvincia])
REFERENCES [dbo].[Distritos] ([IDDistrito], [IDCanton], [IDProvincia])
GO
```

```
ALTER TABLE [dbo].[Pacientes] CHECK CONSTRAINT [FK_Pacientes_Distritos]
GO
```



*Tabla médicos*

```
USE [Hospital]
GO
```

```
/****** Object: Table [dbo].[Médicos]    Script Date: 22/9/2021 07:04:36 *****/
```

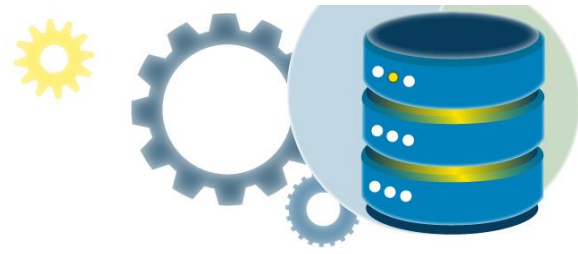
```
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Médicos](
    [CodigoMedico] [nchar](6) NOT NULL,
    [Nombre] [varchar](50) NULL,
    [PrimerApellido] [varchar](30) NULL,
    [SegundoApellido] [varchar](30) NULL,
    [CodigoEspecialidad] [int] NULL,
    [Cargo] [varchar](30) NULL,
    [Observaciones] [varchar](max) NULL,
    CONSTRAINT [PK_Médicos] PRIMARY KEY CLUSTERED
(
    [CodigoMedico] ASC
) ON [PRIMARY]
)
GO
```

```
ALTER TABLE [dbo].[Médicos] WITH CHECK ADD CONSTRAINT [FK_Médicos_Especialidades]
FOREIGN KEY([CodigoEspecialidad])
REFERENCES [dbo].[Especialidades] ([IDEspecialidad])
GO
```

```
ALTER TABLE [dbo].[Médicos] CHECK CONSTRAINT [FK_Médicos_Especialidades]
GO
```



### Tabla Ingresos

```
USE [Hospital]
GO

/***** Object: Table [dbo].[Ingresos]    Script Date: 22/9/2021 07:05:17 *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

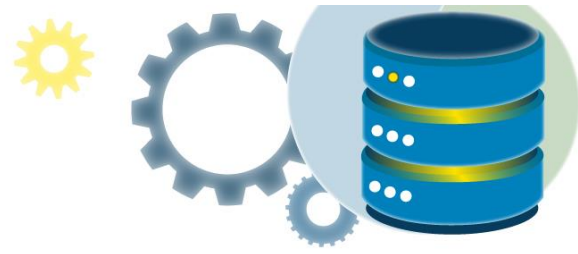
CREATE TABLE [dbo].[Ingresos](
    [IDIngreso] [int] IDENTITY(1,1) NOT NULL,
    [FechaIngreso] [datetime] NULL,
    [NumeroPlanta] [int] NULL,
    [NumeroCama] [int] NULL,
    [IDPaciente] [nchar](15) NULL,
    [IDMedico] [nchar](6) NULL,
    [Diagnostico] [varchar](max) NULL,
    [CostoTratamiento] [float] NULL,
    [Observaciones] [varchar](max) NULL,
    CONSTRAINT [PK_Ingresos] PRIMARY KEY CLUSTERED
(
    [IDIngreso] ASC
) ON [PRIMARY]
)
GO

ALTER TABLE [dbo].[Ingresos] WITH CHECK ADD CONSTRAINT [FK_Ingresos_Médicos] FOREIGN
KEY([IDMedico])
REFERENCES [dbo].[Médicos] ([CodigoMedico])
GO

ALTER TABLE [dbo].[Ingresos] CHECK CONSTRAINT [FK_Ingresos_Médicos]
GO

ALTER TABLE [dbo].[Ingresos] WITH CHECK ADD CONSTRAINT [FK_Ingresos_Pacientes] FOREIGN
KEY([IDPaciente])
REFERENCES [dbo].[Pacientes] ([IDSeguroSocial])
GO

ALTER TABLE [dbo].[Ingresos] CHECK CONSTRAINT [FK_Ingresos_Pacientes]
GO
```



## Referencias

Microsoft. (13 de 09 de 2017). *Tipos de datos (Transact-SQL)*. Obtenido de <https://docs.microsoft.com/es-es/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15>

Microsoft. (08 de 09 de 2019). *Documentación Microsoft*. Obtenido de <https://docs.microsoft.com/es-es/sql/relational-databases/tables/tables?view=sql-server-ver15>