Capítulo III

Tablas de Microsoft SQL Server 2012

En este capítulo conoceremos qué es una tabla, cuáles son los tipos de datos disponibles en SQL Server, cómo se crea una tabla, cómo se asegura la integridad de datos y qué herramientas nos proporciona SQL Server para definirla.

1. ¿QUÉ ES UNA TABLA?

Una tabla es un objeto de la base de datos que se utiliza como almacenamiento de los datos. Una tabla tiene un formato fila-columna similar a las tablas ó listas de Microsoft Excel. Cada fila es un registro único en la tabla (no existe una fila igual a otra en la misma tabla), y cada columna almacena los valores de propiedades específicas del registro. Por ejemplo, si tenemos una tabla EMPLEADO, cada fila almacena los datos de un único empleado, y en cada columna podemos almacenar su apellido paterno, su apellido materno, su nombre, su fecha de nacimiento, su estado civil y número de hijos, por lo que antes de crear una tabla se debe definir el tipo de dato que almacenará cada columna. Un tipo de dato define el tipo de información (cadena, número o fecha) que la columna almacena.

2. TIPOS DE DATOS EN MICROSOFT SQL SERVER 2012

Los tipos de datos de Microsoft SQL Server 2012 están organizados en las siguientes categorías:

- Numéricos exactos
- Numéricos aproximados
- Fecha y hora
- Cadenas de caracteres
- Cadenas de caracteres UNICODE
- Cadenas binarias
- Tipos especiales

2.1. Tipos de datos numéricos exactos

Los tipos de datos numéricos exactos nos permiten especificar de manera exacta la escala y precisión a utilizar para el dato. Por ejemplo, puede especificar tres dígitos a la derecha del decimal y cuatro a la izquierda. Una consulta siempre devuelve exactamente lo que ingresó. SQL Server soporta dos tipos de datos numéricos exactos compatibles con ANSI: **decimal** y **numeric**.

En general, se usan los datos numéricos exactos para aplicaciones financieras en las que se desea tener los datos de forma consistente, por ejemplo, siempre dos espacios decimales para evitar errores de redondeo.

Tipo de dato	Descripción
Bit	Entero. Solo puede tomar el valor 1, o ó NULL.
Tinyint	Entero de 1 byte. Puede tomar los valores de 0 a 255.
smallint	Entero de 2 bytes. Puede tomar los valores de -2^15 (-32.768) a 2^15-1

	(32.767).
Int	Entero de 4 bytes. Puede tomar los valores de -2^31 (-2.147.483.648) a 2^31-1 (2.147.483.647).
Bigint	Entero de 8 bytes. Puede tomar los valores de -2^63 (-9.223.372.036.854.775.808) a 2^63-1 (9.223.372.036.854.775.807).
decimal(*p* , *s*)*	**p** es la precisión, y va de 1 a 38, siendo 18 el valor predeterminado. **s** es la escala y va de o hasta p.
numeric(*p* , *s*)*	**p** es la precisión, y va de 1 a 38, siendo 18 el valor predeterminado. **s** es la escala y va de o hasta p.
smallmoney	Valor monetario de 4 bytes de - 214.748,3648 a 214.748,3647.
money	Valor monetario de 8 bytes de -922,337,203,685.477,5808 a 922,337,203,685.477,5807.

* **p** es la precisión y determina el número máximo de dígitos tanto a la izquierda como a la derecha del punto decimal. **s** es la escala y determina el número máximo de dígitos a la derecha del punto decimal.

2.2. Tipos de datos numéricos aproximados

Los tipos de datos numéricos aproximados almacenan los datos sin precisión. Por ejemplo, el fragmento 1/3 se representa en un sistema decimal como 0,33333…. El número no puede guardarse con precisión, por lo que se almacena una aproximación del valor.

Se usan en las aplicaciones científicas en las que la cantidad de decimales de un valor suele ser muy grande.

Tipo de dato	Descripción
float(***n*** **)**	Si **n** es de 1 a 24, numérico de 4 bytes con una precisión de 7 dígitos. Si **n** es de 25 a 53, numérico de 8 bytes con una precisión de 15 dígitos. El valor predeterminado de n es 53. Los valores van de - 1,79E+308 a -2,23E-308, o y de 2,23E-308 a 1,79E+308.
real ó **float(24)**	Numérico de 4 bytes. Los valores van de - 3,40E + 38 a -1,18E - 38, o y de 1,18E - 38 a 3,40E + 38.

2.3. Tipos de datos de fecha y hora

La fecha y hora pueden almacenarse en un tipo de dato **datetime** o bien en uno **smalldatetime**. Hasta la versión anterior (SQL Server 2008), la fecha y hora se debían almacenar siempre juntas en un solo valor. Ahora es posible almacenar solo la fecha (tipo de dato **date**) o solo la hora (tipo de dato **time**); estos nuevos tipos de datos, además de los tipos **datetime2** y **datetimeoffset** se alinean con el estándar SQL ANSI.

Los datos de fecha y hora pueden tomar varios formatos diferentes. Puede especificar el mes utilizando el nombre completo o una abreviatura. Se ignora el uso de mayúscula/minúscula y las comas son opcionales.

Los siguientes son algunos de los ejemplos de los formatos alfabéticos para el 15 de abril de 2010.

```
'Abr 15 2010'
'Abr 15 10'
'Abr 10 15'
'15 Abr 10'
'2010 Abril 15'
'2010 15 Abril'
```

También puede especificar el valor ordinal del mes. El valor ordinal de un elemento es el valor posicional dentro de una lista de elementos. En los ejemplos anteriores abril es el cuarto mes del año, así que puede usar el número 4 para su designación.

Los siguientes son algunos ejemplos que usan el valor ordinal para el 15 de abril de 2010.

```
4/15/10 (mm/dd/aa)
4/10/15 (mm/aa/dd)
15/10/04 (dd/aa/mm)
10/15/04 (aa/mm/dd)
```

Tipo de dato	Descripción
datetime	Fecha en el rango del 1 de enero de 1753 al 31 de diciembre de 9999, con intervalo de horas de 00:00:00 a 23:59:59.997. Longitud de 8 bytes.
smalldatetime	Fecha en el rango del 1 de enero de 1900 al 6 de junio del 2079, con intervalo de horas de 00:00:00 a 23:59:59. Longitud de 4 bytes.
Date	Fecha en el rango del 1 de enero del año 1 al 31 de diciembre de 9999. Longitud de 3 bytes.
datetime2(***p*** **)**	Fecha en el rango del 1 de enero del año 1 31 de diciembre de 9999, con intervalo de horas de 00:00:00 a 23:59:59.9999999; **p** va de 0 a 7 dígitos, con una precisión de 100 ns. La precisión predeterminada es 7 dígitos. La longitud es 6 bytes para precisiones inferiores a 3, 7 bytes para precisiones 3 y 4. Todas las demás precisiones requieren 8 bytes.
time(***p*** **)**	Hora en el intervalo de 00:00:00.0000000 a 23:59:59.9999999. **p** es un valor entero de 0 a 7 que especifica el número de dígitos de la parte fraccionaria de los segundos. La precisión predeterminada de las fracciones es 7 (100 ns).
datetimeoffset(***p*** **)**	Fecha y hora con reconocimiento de zona horaria en el rango del 1 de enero del año 1 31 de diciembre de 9999, con intervalo de horas de 00:00:00 a 23:59:59.9999999; **p** va de 0 a 7 dígitos, con una precisión de 100 ns. La precisión predeterminada es 7 dígitos. La longitud es 6 bytes para precisiones inferiores a 3, 7 bytes para precisiones 3 y 4. Todas las demás precisiones requieren 8 bytes.

2.4. Tipos de datos de cadenas de caracteres

Estos tipos de datos se usan para almacenar cadenas de caracteres no Unicode.

Tipo de dato	Descripción
char(***n*** **)** ó **character(** ***n*** **)**	Cadena de longitud fija (**n** puede ser de 1 a 8000 bytes).
varchar(***n*** **| max)** ó	Cadena de longitud variable (**n** puede ser de 1 a 8000

character varying(*n* | max)	bytes). **max** indica la longitud máxima que es 2^31 – 1 bytes.
text *	Cadena de longitud variable con un máximo de 2^31 – 1 caracteres.

* **text** no será considerado en versiones futuras de SQL Server. En lugar de ella utilice **varchar(max)**.

2.5. Tipos de datos de cadenas de caracteres UNICODE

Estos tipos de datos se utilizan para almacenar cadenas de caracteres UNICODE.

Tipo de dato	Descripción
nchar(*n*) ó **national character (*n*)**	Cadena UNICODE de longitud fija (**n** puede ser de 1 a 4000 caracteres). La longitud de almacenamiento es 2n.
nvarchar(*n* | max) ó **national character varying (*n* | max)**	Cadena UNICODE de longitud variable (**n** puede ser de 1 a 4000 caracteres). **max** indica la longitud máxima que es 2^31 – 1 bytes. La longitud de almacenamiento es 2 veces la real + 2 bytes.
ntext ó **national text ***	Cadena UNICODE de longitud variable con un máximo de 2^31 – 1 caracteres.

* **ntext** no será considerado en versiones futuras de SQL Server. En lugar de ella utilice **nvarchar(max)**.

2.6. Tipos de datos de cadenas binarias

Tipos de datos para almacenamiento de datos binarios.

Tipo de dato	Descripción
binary(*n*)	Dato binario de longitud fija (**n** puede ser de 1 a 8000 bytes).
varbinary(*n* | max) ó binary varying(*n* | max)	Dato binario de longitud variable (**n** puede ser de 1 a 8000 bytes). **max** indica la longitud máxima que es 2^31 – 1 bytes.
image *	Dato binario de longitud variable que va desde 0 hasta 2^31 – 1 bytes.

* **image** no será considerado en versiones futuras de SQL Server. En lugar de ella utilice **varbinary(max)**.

2.7. Tipos de datos especiales

Tipo de dato	Descripción
cursor *	Tipo de dato para variables ó parámetros de salida de

	procedimientos almacenados.
hierarchyid	Cadena de longitud variable que representa la posición en una jerarquía.
sql_variant	Almacena un valor que puede ser de cualquier tipo soportado por SQL Server, excepto **text, ntext, image, timestamp,** y **sql_variant**.
table *	Define una tabla temporal para almacenar un conjunto de resultados que será utilizado posteriormente.
rowversion **	Generación automática de números binarios de modo que son únicos dentro de una base de datos.
uniqueidentifier	Genera un GUID de 16 bytes. Se usa para crear identificadores únicos en el sistema.
xml(xml_esquema)	Almacena data XML.

* **table** y **cursor** no se pueden utilizar para definir columnas con el comando CREATE TABLE:

** El tipo **timestamp** de versiones anteriores se ha reemplazado por el tipo **rowversion**.

3. Creación de Tablas en Microsoft SQL Server 2012

Para ilustrar este tema presentaremos la base de datos que usaremos para los ejemplos a lo largo del libro. En este capítulo describiremos el escenario del caso a estudiar, crearemos la base de datos y sus tablas y relaciones.

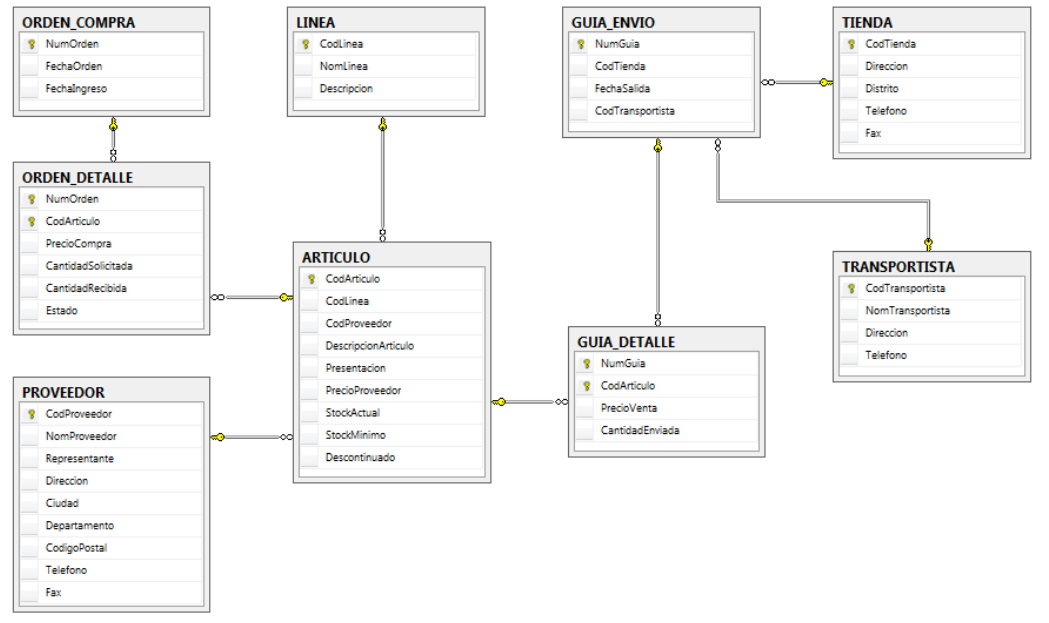
3.1. Presentación del caso a estudiar

QhatuPERU es una empresa dedicada a la comercialización de productos de consumo masivo que cuenta con una cadena de puntos de venta localizados en varias zonas de Lima Metropolitana.

Los distintos locales de la cadena y las áreas administrativas de la empresa se interconectarán mediante una red metropolitana. Se desea diseñar e implantar las aplicaciones comerciales que permitirán gestionar las operaciones de la empresa. Para efectos del presente libro delimitaremos el área de estudio a las operaciones que se ejecutan en el Almacén Central, por lo que la base de datos se diseñará para registrar dichas operaciones, y estará habilitada para que todas las áreas administrativas la utilicen.

Almacén Central controlará todas las operaciones de entrada y salida de artículos (el inventario), y las otras áreas podrán efectuar consultas a la base de datos.

A continuación se presenta el diagrama del modelo de datos propuesto para Almacén Central.



La base de datos **QhatuPERU** estará formada por las siguientes tablas:

TIENDA: relación de los puntos de venta de la empresa. Para cada tienda almacena los siguientes datos:

- CodTienda, identificador de la tienda o punto de venta; dato obligatorio de tipo numérico entero.
- Direccion, ubicación de la tienda; cadena de longitud variable de hasta 60 caracteres.
- Distrito, ubicación de la tienda; cadena de longitud variable de hasta 20 caracteres.
- Telefono, número telefónico de la tienda; cadena de longitud variable de hasta 15 caracteres.
- Fax, número de fax de la tienda; cadena de longitud variable de hasta 15 caracteres.

LINEA: relación de las distintas líneas de artículos que comercializa la empresa. Para cada línea se tiene los siguientes datos:

- CodLinea, identificador de la línea de artículos; dato obligatorio de tipo numérico entero autogenerado.
- NomLinea, nombre de la línea de artículos; dato obligatorio de tipo cadena de longitud variable de hasta 20 caracteres.
- Descripcion, descripción de la línea de artículos; cadena de longitud variable de hasta 40 caracteres.

PROVEEDOR: relación de los proveedores de los artículos que comercializa la empresa. Para cada proveedor se registra los siguientes datos:

- CodProveedor, identificador del proveedor; dato obligatorio de tipo numérico entero autogenerado.
- NomProveedor, nombre del proveedor; dato obligatorio de tipo cadena de longitud variable de hasta 40 caracteres.
- Representante, nombre de la persona de contacto del proveedor; cadena de longitud variable de hasta 30 caracteres.
- Direccion, ubicación del proveedor; cadena de longitud variable de hasta 60 caracteres.
- Ciudad, ubicación del proveedor; cadena de longitud variable de hasta 15 caracteres.
- Departamento, ubicación del proveedor; cadena de longitud variable de hasta 15 caracteres.

- CodigoPostal, ubicación del proveedor; cadena de longitud variable de hasta 15 caracteres.
- Telefono, número telefónico del proveedor; cadena de longitud variable de hasta 15 caracteres.
- Fax, número de fax del proveedor; cadena de longitud variable de hasta 15 caracteres.

ARTICULO: relación de los artículos que comercializa la empresa. Para cada artículo se registra los siguientes datos:

- CodArticulo, identificador del artículo; dato obligatorio de tipo numérico entero autogenerado.
- CodLinea, identificador de la línea a la que pertenece el artículo; dato obligatorio de tipo numérico entero que relaciona al artículo con su correspondiente línea.
- CodProveedor, identificador del proveedor del artículo; dato obligatorio de tipo numérico entero que relaciona al artículo con su proveedor.
- DescripcionArticulo, nombre del artículo; dato obligatorio de tipo cadena de longitud variable de hasta 40 caracteres.
- Presentacion, forma en la que se presenta el artículo; cadena de longitud variable de hasta 30 caracteres.
- PrecioProveedor, precio al que se le compra el artículo al proveedor; dato numérico de tipo dinero (money).
- StockActual, cantidad del artículo existente en el almacén; dato numérico entero pequeño.
- StockMinimo, cantidad mínima que debe existir en el almacén para atender los pedidos de las tiendas; dato numérico entero pequeño.
- Descontinuado, dato de tipo bit que indica si el artículo ha dejado de ser comercializado por la empresa.

ORDEN_COMPRA: relación de los ingresos de artículos al almacén. Para cada orden de compra se registra los siguientes datos:

- NumOrden, identificador de la orden de compra; dato obligatorio de tipo numérico entero.
- FechaOrden, fecha de la orden de compra; dato obligatorio de tipo fecha-hora.
- FechaIngreso, fecha de ingreso de los artículos al almacén; dato de tipo fecha-hora.

ORDEN_DETALLE: relación de los artículos asociados a cada orden de compra. Para cada artículo se registra los siguientes datos:

- NumOrden, número de la orden de compra a la que pertenece el artículo ingresado al almacén; dato obligatorio de tipo numérico entero que relaciona al artículo con su correspondiente orden de compra.
- CodArticulo, identificador del artículo recibido con la orden de compra especificada en NumOrden; dato obligatorio de tipo numérico entero que relaciona al item en la orden de compra con los datos complementarios del artículo.
- PrecioCompra, precio de compra del artículo; dato obligatorio de tipo dinero (money).
- CantidadSolicitada, cantidad que se le solicitó al proveedor en la correspondiente orden de compra; dato obligatorio de tipo numérico entero pequeño.
- CantidadRecibida, cantidad que recibió del proveedor y se ingresó al almacén; dato numérico entero pequeño.
- Estado, estado del artículo en la orden de compra; cadena de longitud variable de hasta 10 caracteres.

TRANSPORTISTA: relación de las personas encargadas de llevar los artículos desde el almacén hasta cada una de las tiendas. Para cada transportista se registra los siguientes datos:

- CodTransportista, identificador del transportista; dato obligatorio de tipo numérico entero.
- NomTransportista, nombre del transportista; dato obligatorio de tipo cadena de longitud variable de hasta 30 caracteres.
- Direccion, dirección del transportista; cadena de longitud variable de hasta 60 caracteres.
- Telefono, número telefónico del transportista; cadena de longitud variable de hasta 15 caracteres.

GUIA_ENVIO: relación de las salidas de artículos al almacén. Para cada guía de envío se registra los siguientes datos:

- NumGuia, identificador de la guia de envío; dato obligatorio de tipo numérico entero.
- CodTienda, identificador de la tienda a la que se envía los artículos; dato obligatorio de tipo numérico entero que relaciona a la guía de envío con la correspondiente tienda.
- FechaSalida, fecha de la guía de envío; dato obligatorio de tipo fecha-hora.
- CodTransportista, identificador del transportista responsable de trasladar el envío; dato obligatorio de tipo numérico entero que relaciona la guía de envío con el transportista a cargo del envío.

GUIA_DETALLE: relación de los artículos asociados a cada guía de envío. Para cada artículo se registra los siguientes datos:

- NumGuia, número de la guía de envío a la que pertenece el artículo que ha salido del almacén; dato obligatorio de tipo numérico entero que relaciona al artículo con su correspondiente guía de envío.
- CodArticulo, identificador del artículo entregado con la guía de envío especificada en NumOrden; dato obligatorio de tipo numérico entero que relaciona al item en la guía de envío con los datos complementarios del artículo.
- PrecioVenta, precio de venta al público del artículo en la tienda; dato obligatorio de tipo dinero (money).
- CantidadEnviada, cantidad que se envió a la tienda desde el almacén; dato obligatorio de tipo numérico entero pequeño.

Ejercicio 3.1: Creación de la base de datos QhatuPERU

1. En **SQL Server Management Studio** conéctese a su servidor SQL y abra una nueva consulta.

2. Ejecute las siguientes instrucciones:

```
USE master
go

CREATE DATABASE QhatuPERU
go
```

3.2. Creación de una tabla – La instrucción CREATE TABLE

Sintáxis básica

```
CREATE TABLE nombre_tabla (
     nombre_columna1 tipo_dato1 [NULL|NOT NULL] ,
     nombre_columna2 tipo_dato2 [NULL|NOT NULL] ,
     nombre_columna3 tipo_dato3 [NULL|NOT NULL] ,
     … )
```

- NULL|NOT NULL establece si el dato a ingresar en la columna es obligatorio u opcional. Por ejemplo, para los datos de un trabajador podemos establecer que su nombre es obligatorio registrarlo en la tabla, por lo tanto definiríamos la columna **nombreTrabajador** con la propiedad NOT NULL. Adicionalmente, como no todos los trabajadores cuentan con un teléfono celular, definiríamos

la columna **telefonoMovil** con la propiedad NULL para indicar que este dato no es obligatorio registrarlo.

Si para una columna no se especifica si es NULL o NOT NULL, el valor predeterminado de la propiedad depende de la opción de configuración **ANSI null default** de la base de datos. Es recomendable establecer de forma explícita el valor de la propiedad; de este modo evitamos conflictos si la configuración de la base de datos cambia.

Significado del valor NULL

Si una columna de una tabla tiene la propiedad NULL, y no se especifica el valor a almacenar en dicha columna, el motor de datos almacena en ella el valor NULL. Con este valor estamos indicando simplemente que para dicha fila el valor de esa columna es desconocido ó no está disponible. Un valor NULL no es lo mismo que una cadena de longitud cero, ó una cadena de espacios, ó el valor o (cero). Todos estos representan un dato conocido, el valor NULL significa dato desconocido.

Ejercicio 3.2: Creación de una tabla especificando solo sus columnas – Creación de la tabla TIENDA

Según el modelo de datos visto anteriormente, la tabla TIENDA tiene la siguiente definición:

- CodTienda, identificador de la tienda o punto de venta; dato obligatorio de tipo numérico entero.
- Direccion, ubicación de la tienda; cadena de longitud variable de hasta 60 caracteres.
- Distrito, ubicación de la tienda; cadena de longitud variable de hasta 20 caracteres.
- Telefono, número telefónico de la tienda; cadena de longitud variable de hasta 15 caracteres.
- Fax, número de fax de la tienda; cadena de longitud variable de hasta 15 caracteres.

Ejecute las siguientes instrucciones:

```
USE QhatuPERU
Go

CREATE TABLE TIENDA (
     CodTienda      int NOT NULL ,
     Direccion      varchar(60) NULL ,
     Distrito       varchar(20) NULL ,
     Telefono       varchar(15) NULL ,
     Fax            varchar(15) NULL )
go
```

3.3. Consulta de la definición de una tabla – El procedimiento sp_help

El procedimiento de sistema **sp_help** genera un reporte que muestra la definición de un objeto de la base de datos activa. Este procedimiento puede ser ejecutado por cualquier usuario de la base de datos.

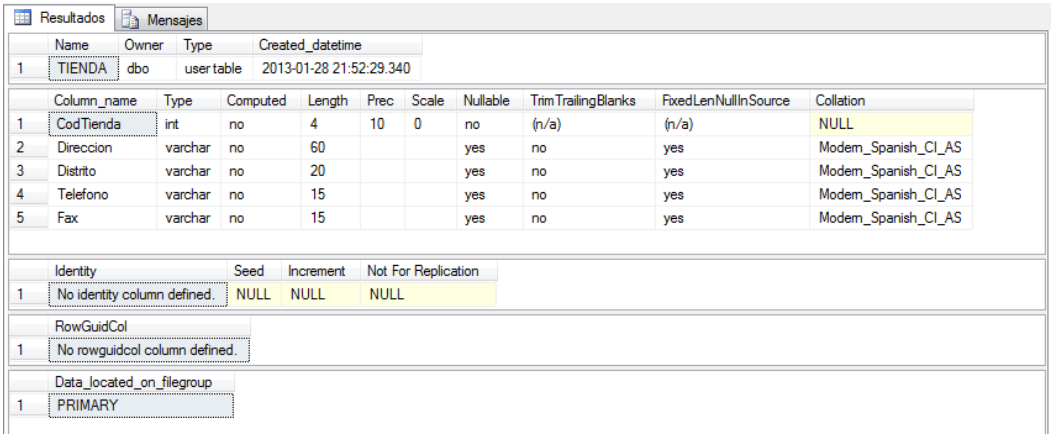
Sintáxis

```
sp_help nombre_objeto_basedatos
```

Ejercicio 3.3: Verificación de la definición de la tabla TIENDA

En la ventana query ejecute la siguiente instrucción:

```
sp_help TIENDA
go
```



Resultados | Mensajes

	Name	Owner	Type	Created_datetime
1	TIENDA	dbo	user table	2013-01-28 21:52:29.340

	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	FixedLenNullInSource	Collation
1	CodTienda	int	no	4	10	0	no	(n/a)	(n/a)	NULL
2	Direccion	varchar	no	60			yes	no	yes	Modern_Spanish_CI_AS
3	Distrito	varchar	no	20			yes	no	yes	Modern_Spanish_CI_AS
4	Telefono	varchar	no	15			yes	no	yes	Modern_Spanish_CI_AS
5	Fax	varchar	no	15			yes	no	yes	Modern_Spanish_CI_AS

	Identity	Seed	Increment	Not For Replication
1	No identity column defined.	NULL	NULL	NULL

	RowGuidCol
1	No rowguidcol column defined.

	Data_located_on_filegroup
1	PRIMARY

El reporte generado muestra:

- Las propiedades principales de la tabla.
- La definición de cada una de sus columnas.
- Si la tabla tiene definida una columna IDENTITY. Una columna IDENTITY es una columna de tipo numérica entera cuyo valor es generado por el sistema en forma secuencial siguiendo una regla de incremento específica. Se verá más adelante.
- Si la tabla tiene definida una columna RowGuidCol. Una columna RowGuidCol es una columna de tipo uniqueidentifier que almacena un GUID (Global Unique Identifier) de 16 bytes como identificador único en el Sistema.
- El grupo de archivos en el que está localizada la tabla. De modo predeterminado, la base de datos y sus objetos se localizan en el grupo de archivos PRIMARY.

4. MODIFICACIÓN DE LA DEFINICIÓN DE UNA TABLA

4.1. La instrucción ALTER TABLE

Sintáxis

```
ALTER TABLE nombre_tabla
       ADD nombre_columna propiedades_columna
       | DROP COLUMN nombre_columna
       | ALTER COLUMN nombre_columna
                    nuevas_propiedades_columna
       | ADD CONSTRAINT nombre_restricción
          PRIMARY KEY… | UNIQUE… | FOREIGN KEY…
          | DEFAULT… | CHECK…
       | DROP CONSTRAINT nombre_restricción
```

- ADD **nombre_columna** permite crear una nueva columna en la tabla.
- DROP COLUMN se utiliza para eliminar una columna.
- ALTER COLUMN permite cambiar las propiedades de una columna.
- ADD CONSTRAINT permite añadir una restricción PRIMARY KEY, UNIQUE, FOREIGN KEY, DEFAULT ó CHECK a la definición de una tabla. Una restricción define las reglas que debe cumplir un dato para ser aceptable.
- DROP CONSTRAINT se utiliza para eliminar una restricción.

Ejercicio 3.4: Adición de una columna a una tabla

1. Para ilustrar el uso de la instrucción ALTER TABLE, en la base de datos **QhatuPERU** vamos a crear una tabla denominada **TablaPrueba**, a la que luego modificaremos su definición.

```
USE QhatuPERU
go

CREATE TABLE TablaPrueba(
     campo1 INT NOT NULL ,
     campo2 CHAR(10) NOT NULL ,
     campo3 DATETIME NULL )
go
```

2. A **TablaPrueba** vamos a añadirle una columna de tipo MONEY con la propiedad NULL.

```
ALTER TABLE TablaPrueba
     ADD campo4 MONEY NULL
go
```

3. Utilizando el procedimiento **sp_help** revisamos la definición de la tabla **TablaPrueba**.

```
sp_help TablaPrueba
go
```

Ejercicio 3.5: Adición de una columna NOT NULL a una tabla

1. Ahora, añadiremos a la tabla **TablaPrueba** una columna de tipo VARCHAR con la propiedad NOT NULL.

```
ALTER TABLE TablaPrueba
     ADD campo5 VARCHAR(20) NOT NULL
go
```

2. La columna se añade sin problemas. A continuación ingresaremos 2 filas de datos a **TablaPrueba**. Para ello ejecute las siguientes instrucciones (más adelante veremos de forma más detallada la instrucción INSERT):

```
INSERT INTO TablaPrueba
     VALUES(1, 'RMM0123456', '19/03/2012',
          1375.50, 'Rosa')
INSERT INTO TablaPrueba
     VALUES(2, 'JCS9876543', '21/09/2012',
          1250.50, 'Juvenal')
go
```

3. Las 2 filas se insertan sin problemas en la tabla **TablaPrueba**. Para ver el contenido de la tabla ejecute la instrucción:

```
SELECT * FROM TablaPrueba
go
```

4. En el panel de **Resultados** se muestra el contenido de la tabla. Ahora, trataremos de añadir otra columna NOT NULL a la tabla; esta nueva columna será de tipo DECIMAL.

```
ALTER TABLE TablaPrueba
     ADD campo6 DECIMAL(12,3) NOT NULL
go
```

5. Obtenemos el siguiente mensaje de error:

```
Mens. 4901, Nivel 16, Estado 1, Línea 1
ALTER TABLE sólo permite agregar columnas que
contengan valores NULL, que tengan la definición
DEFAULT, que la columna que se agrega sea una
columna de identidad o de marca de tiempo, o si
ninguna de las condiciones anteriores se cumplen,
la tabla debe estar vacía para que se pueda
agregar esta columna. La columna 'campo6' no se
puede agregar a la tabla 'TablaPrueba' no vacía
porque no cumple estas condiciones.
```

El mensaje nos dice que ALTER TABLE solo permite añadir columnas que pueden contener valores nulos (columnas NULL) ó que tengan especificado un valor predeterminado, salvo que la tabla esté vacía.

Nota: ¿Qué pasaría en una tabla que ya tiene algunas filas registradas si tratamos de añadirle una nueva columna que no permita valores nulos? ¿Qué valor debería colocar SQL Server en la nueva columna para las filas que ya están registradas?

Es evidente que si la nueva columna no permite valores nulos, SQL Server se vería obligado a ingresar un valor para la nueva columna de las filas ya existentes. El problema es que SQL Server no conoce cuál es ese valor, por lo que se produciría un conflicto entre la definición de la columna y el estado actual de los datos.

Para añadir una columna NOT NULL a una tabla que ya tiene datos, primero debemos añadir la columna con la propiedad NULL; a continuación debemos registrar los valores de esta nueva columna para las filas ya existentes; finalmente cambiamos la propiedad de la columna a NOT NULL.

6. Añadimos la nueva columna como NULL:

```
ALTER TABLE TablaPrueba
     ADD campo6 DECIMAL(12,3) NULL
go
```

7. Establecemos los valores en la columna **campo6** para las 2 filas existentes (más adelante veremos con detalle el uso de la instrucción UPDATE):

```
UPDATE TablaPrueba SET campo6 = 123.456
     WHERE campo1 = 1
UPDATE TablaPrueba SET campo6 = 678.123
     WHERE campo1 = 2
go
```

8. Redefinimos la columna **campo6** como NOT NULL:

```
ALTER TABLE TablaPrueba
     ALTER COLUMN campo6 DECIMAL(12,3) NOT NULL
go
```

5. INTEGRIDAD DE DATOS

Se conoce como integridad de datos al mecanismo utilizado en la base de datos para garantizar la consistencia y exactitud de los datos. La base de datos permite definir un conjunto de herramientas para manejar la integridad de los datos.

5.1. Los niveles de la integridad de datos

La integridad de los datos se define en distintos niveles dependiendo de la parte de la base de datos a la que afecta:

Integridad de entidad

Una tabla almacena los datos de cada una de las ocurrencias ó instancias de una entidad. Por ejemplo, si la entidad es ARTICULO, cada uno de los artículos representa una ocurrencia ó instancia de la entidad.

La entidad (o tabla) requiere que todas sus filas sean únicas. Esto se garantiza definiendo para cada fila de la entidad un identificador único (llave primaria). Por ejemplo, en la tabla ARTICULO, cada fila de la tabla debe representar a solo un artículo; un artículo no puede aparecer registrado en dos filas de la tabla.

Integridad de dominio

Al conjunto de valores aceptables de un atributo (o columna) se le conoce como el dominio del atributo. La integridad de dominio determina las condiciones que deben cumplir los valores a almacenar en una columna.

La integridad de dominio se define mediante reglas de validación, valores predeterminados, conjunto de valores permitidos en la columna (llave foránea), tipo y formato de los datos. Por ejemplo, en la tabla ARTICULO se puede requerir que el valor de la columna **PrecioProveedor** no puede ser negativo; cualquier valor menor a o debe ser rechazado por la base de datos. También podemos establecer que los valores en la columna **DescripcionArticulo** deben ser valores únicos (no duplicados), es decir que no puede haber más de un artículo con la misma descripción.

Integridad referencial

La integridad referencial garantiza que la relación entre la llave primaria (en la tabla referenciada) y la llave foránea (en la tabla de referencia) siempre se mantiene. En otras palabras, que cada una de las filas de la tabla de referencia o tabla secundaria está relacionada con una fila de la tabla referenciada o tabla primaria.

Una fila en una tabla referenciada no puede anularse, ni cambiar su valor de la llave primaria, si una llave foránea se refiere a la fila. Por ejemplo, si tenemos las tablas LINEA y ARTICULO, cada artículo debe pertenecer a solo una línea; es decir, cada fila de la tabla ARTICULO debe estar relacionada con una fila de la tabla LINEA, pero una fila de la tabla LINEA puede estar relacionada con muchas filas de la tabla ARTICULO ya que una línea puede tener muchos artículos.

6. LAS RESTRICCIONES (CONSTRAINTS)

El lenguaje SQL proporciona un método declarativo para definir la integridad de los datos. Este método consiste en la creación de restricciones de datos al momento de crear la tabla con la

instrucción CREATE TABLE, o al momento de modificar su definición con la sentencia ALTER TABLE.

Las restricciones o constraints forman parte de la definición de las tablas, y son el método preferido para forzar la integridad de los datos.

6.1. Tipos de restricciones

Las restricciones o constraints son un método estándar ANSI para forzar la integridad de los datos. Garantizan que los datos ingresados en las columnas de las tablas son valores válidos y que se mantengan las relaciones entre las tablas.

La siguiente es una relación de los diferentes tipos de restricciones que podemos definir y sus descripciones.

PRIMARY KEY (clave primaria)

Garantiza que cada fila ó registro en una tabla es único(a). Se aplica sobre una columna (clave primaria simple) o combinación de columnas (clave primaria compuesta). La columna ó combinación de columnas definida como clave primaria no permite valores duplicados.

UNIQUE (valor no duplicado)

Garantiza que cada valor en una columna es único. Se utiliza para asegurar que una columna que no es la clave primaria contenga valores únicos. Permite valores nulos.

FOREIGN KEY (clave foránea)

Define la columna ó combinación de columnas de una tabla secundaria cuyos valores dependen de la clave primaria de una tabla primaria. Establece la relación entre la tabla primaria y su tabla secundaria.

DEFAULT (valor predeterminado)

Establece el valor predeterminado para una columna cuando al insertar una fila no se especifica el valor para dicha columna. Es útil cuando la columna no permite valores nulos y no se conoce el valor que debe tener dicha columna para la fila que se está insertando.

CHECK (regla de validación)

Establece la regla que debe cumplir un valor para que sea un valor aceptable en una columna. Es útil para implementar reglas de negocios sencillas en la base de datos.

Ejercicio 3.6: Creación de una tabla especificando su clave primaria – Creación de la tabla TRANSPORTISTA

Según el modelo de la base de datos **QhatuPERU,** la tabla TRANSPORTISTA tiene la siguiente definición:

- CodTransportista, identificador del transportista; dato obligatorio de tipo numérico entero.
- NomTransportista, nombre del transportista; dato obligatorio de tipo cadena de longitud variable de hasta 30 caracteres.
- Direccion, dirección del transportista; cadena de longitud variable de hasta 60 caracteres.
- Telefono, número telefónico del transportista; cadena de longitud variable de hasta 15 caracteres.

1. En su ventana Query ejecute las siguientes instrucciones:

```
USE QhatuPERU
go

CREATE TABLE TRANSPORTISTA (
     CodTransportista    int NOT NULL PRIMARY KEY,
     NomTransportista    varchar(30) NOT NULL,
     Direccion           varchar(60) NULL,
     Telefono            varchar(15) NULL )
go
```

2. Se crea en la base de datos **QhatuPERU** la tabla TRANSPORTISTA en la que la columna **CodTransportista** es la clave primaria o identificador de cada transportista. Para probar el efecto de la clave primaria ejecutaremos las siguientes sentencias INSERT:

```
INSERT INTO TRANSPORTISTA
     VALUES(1, 'VELASQUEZ ORTIZ, FRANCISCO',
          'Av. Los Alamos 1234 San Luis',
          '999-123-456')
INSERT INTO TRANSPORTISTA
     VALUES(2, 'ALIAGA VIDAL, JEREMIAS',
          'Jr. Tarma 456 Callao',
          '991-234-456')
INSERT INTO TRANSPORTISTA
     VALUES(2, 'CASTRO SOLIS, JUAN JOSE',
          'Jr. Las Novicias 123 Chorrillos',
          '993-000-111')
go
```

3. La dos primeras instrucciones INSERT se ejecutan sin problemas. La tercera genera el mensaje de error 2627 debido a que se viola la clave primaria de la

tabla ya que el valor de la columna **CodTransportista** se está duplicando (valor 2).

```
(1 filas afectadas)

(1 filas afectadas)
Mens. 2627, Nivel 14, Estado 1, Línea 5
Infracción de la restricción PRIMARY KEY
'PK__TRANSPOR__E42881D5DD5000DD'. No se puede
insertar una clave duplicada en el objeto
'dbo.TRANSPORTISTA'. El valor de la clave
duplicada es (2).
Se terminó la instrucción.
```

4. Observe que SQL Server ha definido la cadena **PK__TRANSPOR__E42881D5DD5000DD** como nombre de la restricción clave primaria. El nombre está formado por el prefijo PK_ seguido de los 8 primeros caracteres del nombre de la tabla a la que pertenece, y luego de una cadena hexadecimal. Para revisar la tabla y verificar la data ingresada ejecute:

```
SELECT * FROM TRANSPORTISTA
go
```

Resultados | Mensajes

	CodTransportista	NomTransportista	Direccion	Telefono
1	1	VELASQUEZ ORTIZ, FRANCISCO	Av. Los Alamos 1234 San Luis	999-123-456
2	2	ALIAGA VIDAL, JEREMIAS	Jr. Tarma 456 Callao	991-234-456

6.2. Creación de una clave primaria en una tabla existente

Si la tabla ya ha sido creada y deseamos añadirle su clave primaria, podemos usar la instrucción ALTER TABLE.

Sintáxis

```
ALTER TABLE nombre_tabla
     ADD CONSTRAINT PK_nombre_tabla
     PRIMARY KEY( columnaX, columnaP, … )
```

- **PK_nombre_tabla** es el nombre de la restricción clave primaria. Se recomienda definir como nombre de la clave primaria, el nombre de la tabla con el prefijo PK_. Si no se especfica el nombre de la restricción, SQL Server le asigna un nombre.
- **columnaX**, **columnaP**, es la columna ó combinación de columnas que se define como clave primaria. Las columnas involucradas no deben permitir valores nulos, y además, no deben tener valores duplicados. En el caso de una combinación de columnas, la combinación vista como una unidad no debe tener valores duplicados.

Ejercicio 3.7: Creación de una clave primaria en una tabla existente – Creación de la clave primaria de la tabla TIENDA

Para la tabla TIENDA creada en el ejercicio 3.2 defina la columna **CodTienda** como su clave primaria.

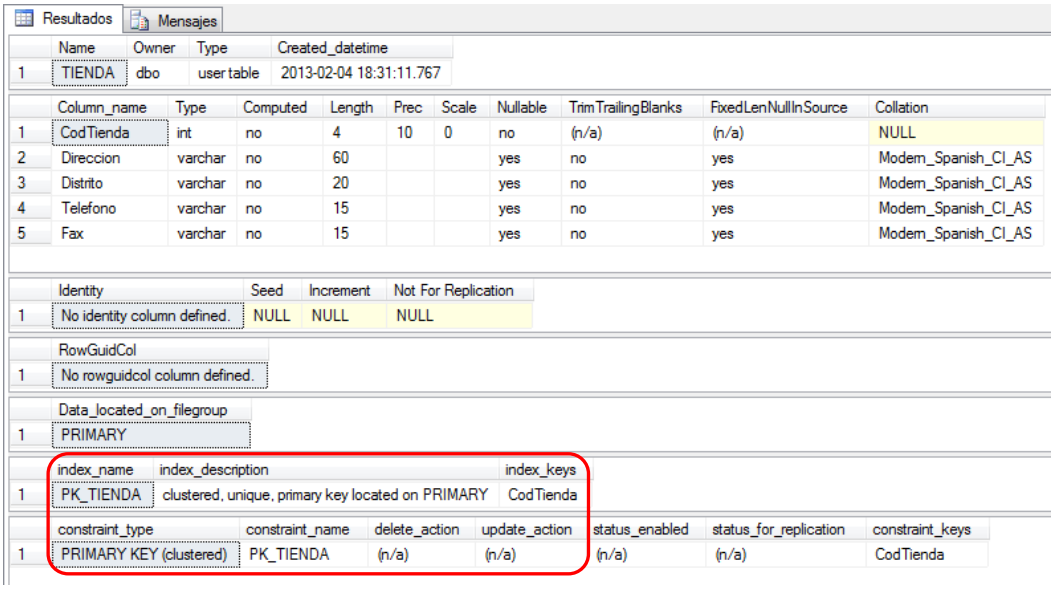
1. Ejecute las siguientes instrucciones:

```
USE QhatuPERU
go

ALTER TABLE TIENDA
     ADD CONSTRAINT PK_TIENDA
     PRIMARY KEY( CodTienda )
go
```

2. Ejecute el procedimiento **sp_help** para verificar la definición de la tabla.

```
sp_help TIENDA
go
```



6.3. Creación de una clave primaria numérica con valores secuenciales autogenerados – Uso de la propiedad IDENTITY

La propiedad IDENTITY permite definir una columna numérica entera en la que el valor de la columna es autogenerado a medida que se va insertando filas. Es útil cuando se desea emplear claves primarias numéricas.

Ejercicio 3.8: Uso de la propiedad IDENTITY – Creación de la tabla LINEA

Según el modelo de datos de la base de datos **QhatuPERU,** la tabla LINEA tiene la siguiente definición:

- CodLinea, identificador de la línea de artículos; dato obligatorio de tipo numérico entero autogenerado.
- NomLinea, nombre de la línea de artículos; dato obligatorio de tipo cadena de longitud variable de hasta 20 caracteres.
- Descripcion, descripción de la línea de artículos; cadena de longitud variable de hasta 40 caracteres.

En la tabla LINEA, la columna **CodLinea** no solo es su clave primaria sino que además su contenido es autogenerado en forma de secuencia a medida que se va insertando las filas.

1. Para crear la tabla LINEA ejecute las siguientes instrucciones:

```
USE QhatuPERU
go
```

```
CREATE TABLE LINEA (
     CodLinea    int IDENTITY(1,1) PRIMARY KEY,
     NomLinea    varchar(20) NOT NULL,
     Descripcion varchar(40) NULL )
go
```

2. Ejecute las siguientes instrucciones para insertar filas en la tabla y vea el efecto de la propiedad IDENTITY:

```
INSERT LINEA
     VALUES('GOLOSINAS',
     'GALLETAS,CHOCOLATES,CARAMELOS,TOFFES')
INSERT LINEA
     VALUES('EMBUTIDOS',
     'JAMONADAS,JAMONES,SALCHICHAS,CHORIZOS')
INSERT LINEA
     VALUES('HIGIENE PERSONAL',
     'JABONES,P.DENTALES,SHAMPOOS,P.H.')
go
```

3. Para ver el contenido de la tabla ejecute:

```
SELECT * FROM LINEA
go
```

En IDENTITY(1,1), el primer argumento conocido como "semilla" establece el primer valor de identidad a generar, y el segundo argumento es el "incremento" a utilizar para generar los siguientes valores.

Ejercicio 3.9: Creación de una clave primaria compuesta – Creación de la tabla ORDEN_DETALLE y de su clave primaria

Según el modelo de la base de datos, la tabla ORDEN_DETALLE tiene la siguiente definición:

- NumOrden, número de la orden de compra a la que pertenece el artículo ingresado al almacén; dato obligatorio de tipo numérico entero que relaciona al artículo con su correspondiente orden de compra.
- CodArticulo, identificador del artículo recibido con la orden de compra especificada en NumOrden; dato obligatorio de tipo numérico entero que relaciona al item en la orden de compra con los datos complementarios del artículo.

- PrecioCompra, precio de compra del artículo; dato obligatorio de tipo dinero (money).
- CantidadSolicitada, cantidad que se le solicitó al proveedor en la correspondiente orden de compra; dato obligatorio de tipo numérico entero pequeño.
- CantidadRecibida, cantidad que recibió del proveedor y se ingresó al almacén; dato numérico entero pequeño.
- Estado, estado del artículo en la orden de compra; cadena de longitud variable de hasta 10 caracteres.

La clave primaria de la tabla ORDEN_DETALLE está formada por las columnas **NumOrden** y **CodArticulo**, ya que para identificar el ingreso de un artículo al Almacén se necesita conocer el artículo ingresado y con cuál orden de compra se ingresó.

1. Para crear la tabla ORDEN_DETALLE ejecute las siguientes instrucciones:

```
USE QhatuPERU
go

CREATE TABLE ORDEN_DETALLE (
     NumOrden             int NOT NULL,
     CodArticulo          int NOT NULL,
     PrecioCompra         money NOT NULL,
     CantidadSolicitada   smallint NOT NULL,
     CantidadRecibida     smallint NULL,
     Estado               varchar(10) NULL )
go
```

2. Para añadir la clave primaria a la tabla ejecute:

```
ALTER TABLE ORDEN_DETALLE
     ADD CONSTRAINT PK_ORDEN_DETALLE
     PRIMARY KEY( NumOrden, CodArticulo )
go
```

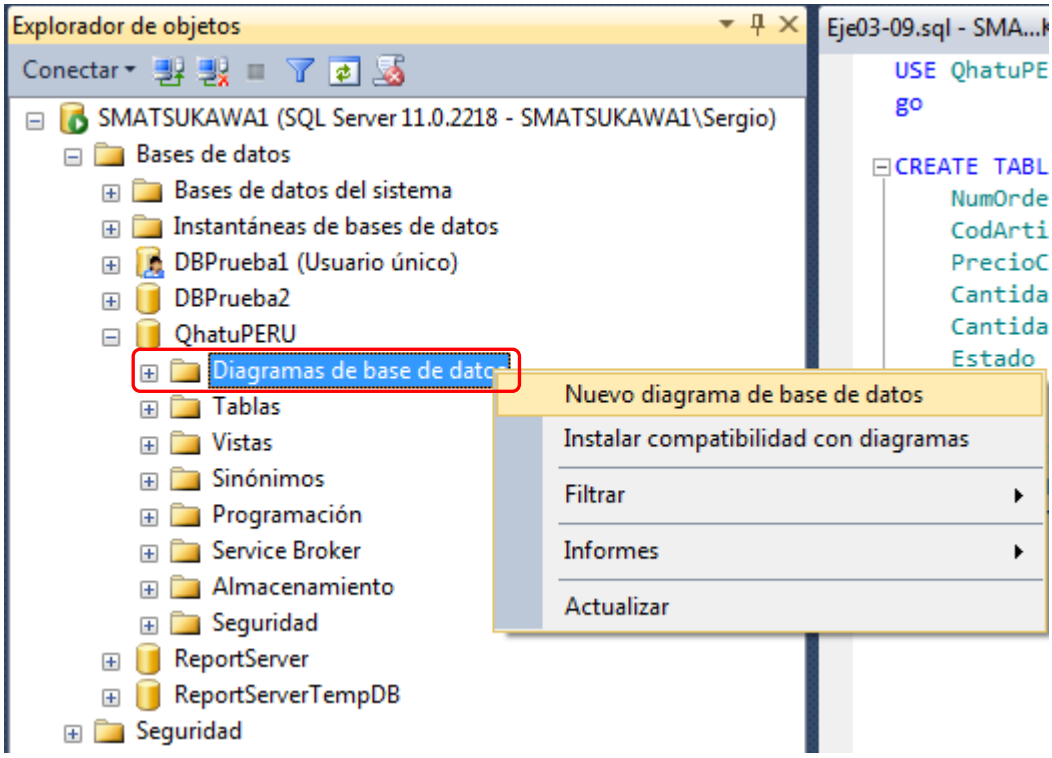
6.4. Creación de un diagrama de base de datos

SQL Server Management Studio posee una herramienta que permite ver gráficamente su modelo de datos, ya sea en una vista parcial o en una vista completa.

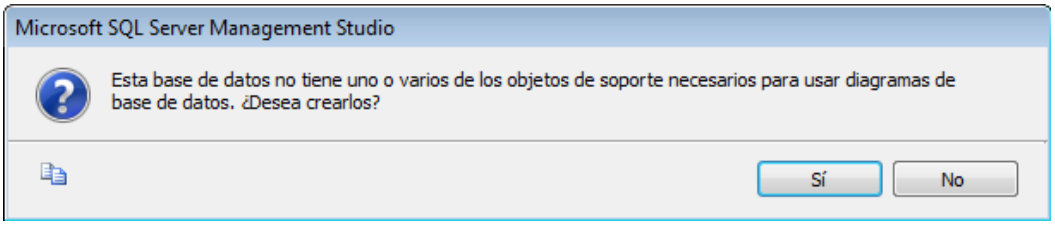
Ejercicio 3.10: Creación de un diagrama de la base de datos

Si deseamos ver gráficamente qué tablas forman parte de nuestra base de datos y cómo se relacionan entre sí, podemos crear un diagrama de la base de datos. Para ello:

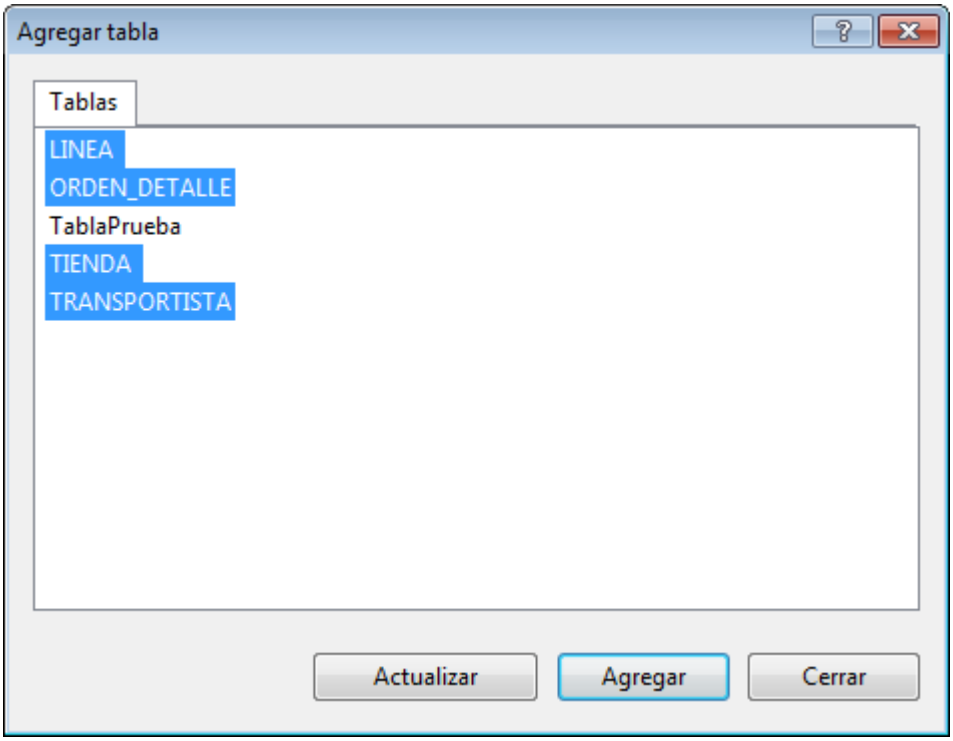
1. En el **Explorador de objetos** de SQL Server Management Studio, expanda el nodo **Bases de datos** de su servidor, luego expanda el nodo de la base de datos **QhatuPERU**, y haga un clic secundario en **Diagramas de base de datos**.



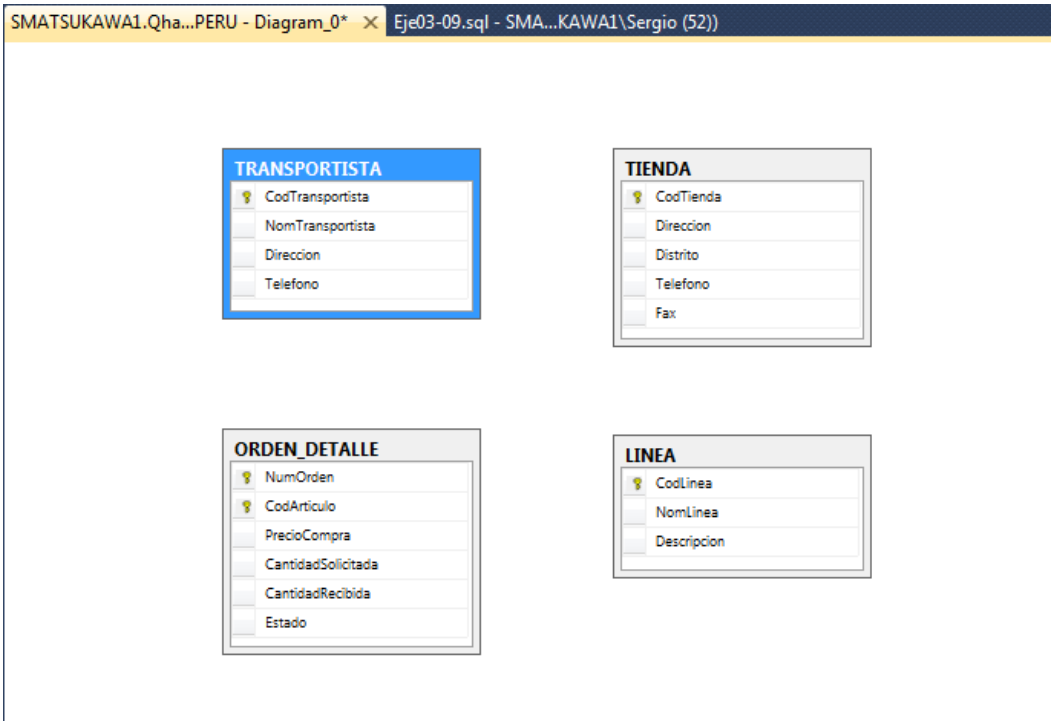
2. En el menú contextual haga clic en **Nuevo diagrama de base de datos**. Si se muestra la ventana de mansajes que se muestra a continuación, haga clic en **Si**.



3. Se muestra la ventana **Agregar tabla** con la lista de tablas presentes en la base de datos.



4. Seleccione las tablas LINEA, ORDEN_DETALLE, TIENDA y TRANSPORTISTA, y luego haga clic en **Agregar**. Luego clic en **Cerrar**. Se crea un diagrama que muestra las tablas añadidas a él. Por el momento, no hay definidas relaciones entre las tablas.



5. Guarde el diagrama bajo el nombre **Diagrama01**.

Ejercicio 3.11: Creación de la tabla ARTICULO y de su clave primaria

El modelo de datos de la base de datos QhatuPERU establece que la tabla ARTICULO tiene la siguiente definición:

- CodArticulo, identificador del artículo; dato obligatorio de tipo numérico entero autogenerado.
- CodLinea, identificador de la línea a la que pertenece el artículo; dato obligatorio de tipo numérico entero que relaciona al artículo con su correspondiente línea.
- CodProveedor, identificador del proveedor del artículo; dato obligatorio de tipo numérico entero que relaciona al artículo con su proveedor.
- DescripcionArticulo, nombre del artículo; dato obligatorio de tipo cadena de longitud variable de hasta 40 caracteres.
- Presentacion, forma en la que se presenta el artículo; cadena de longitud variable de hasta 30 caracteres.
- PrecioProveedor, precio al que se le compra el artículo al proveedor; dato numérico de tipo dinero (money).

- StockActual, cantidad del artículo existente en el almacén; dato numérico entero pequeño.
- StockMinimo, cantidad mínima que debe existir en el almacén para atender los pedidos de las tiendas; dato numérico entero pequeño.
- Descontinuado, dato de tipo bit que indica si el artículo ha dejado de ser comercializado por la empresa.

1. Para crear la tabla ARTICULO ejecute lo siguiente:

```
USE QhatuPERU
Go

CREATE TABLE ARTICULO (
        CodArticulo           int IDENTITY
               PRIMARY KEY,
        CodLinea              int NOT NULL,
        CodProveedor          int NOT NULL,
        DescripcionArticulo   varchar(40)
               NOT NULL,
        Presentacion          varchar(30) NULL,
        PrecioProveedor       money NULL,
        StockActual           smallint NULL,
        StockMinimo           smallint NULL,
        Descontinuado         bit )
go
```

2. Para ver qué tablas tiene la base de datos ejecute la siguiente consulta:

```
SELECT name FROM sys.tables
go
```

6.5. Creación de una clave foránea

Sintáxis

```
ALTER TABLE nombre_tabla
     ADD CONSTRAINT FK_nombre_tabla_tabla_referenciada
     FOREIGN KEY( columnaX, columnaP, … )
     REFERENCES tabla_referenciada
```

- **FK_nombre_tabla_tabla_referenciada** es el nombre de la restricción clave foránea. Se recomienda definir como nombre de la clave foránea, el nombre de

la tabla seguido del nombre de la tabla referenciada, todo con el prefijo **FK_**. Si no se especfica el nombre de la restricción, SQL Server le asigna un nombre.

- **columnaX**, **columnaP**, es la columna ó combinación de columnas que se define como clave foránea.
- **tabla_referenciada** es el nombre de la tabla primaria con la que se relaciona la tabla secundaria que tiene la clave foránea. De modo predeterminado, la clave foránea hace referencia a la clave primaria de la tabla primaria.

Ejercicio 3.12: Creación de la clave foránea en la tabla ARTICULO que referencia a la tabla LINEA

1. En su ventana Query ejecute las siguientes instrucciones para crear la clave foránea de la tabla ARTICULO que referencia a la tabla LINEA:

```
USE QhatuPERU
go

ALTER TABLE ARTICULO
  ADD CONSTRAINT fk_Articulo_Linea
  FOREIGN KEY( CodLinea )
  REFERENCES LINEA
go
```

   La columna **CodLinea** de la tabla ARTICULO establece la relación de esta tabla con la tabla LINEA. La clave foránea en **CodLinea** de ARTICULO apunta a la clave primaria en **CodLinea** de la tabla LINEA.

2. Para probar la integridad referencial (la restricción clave foránea) ejecute las siguientes instrucciones:

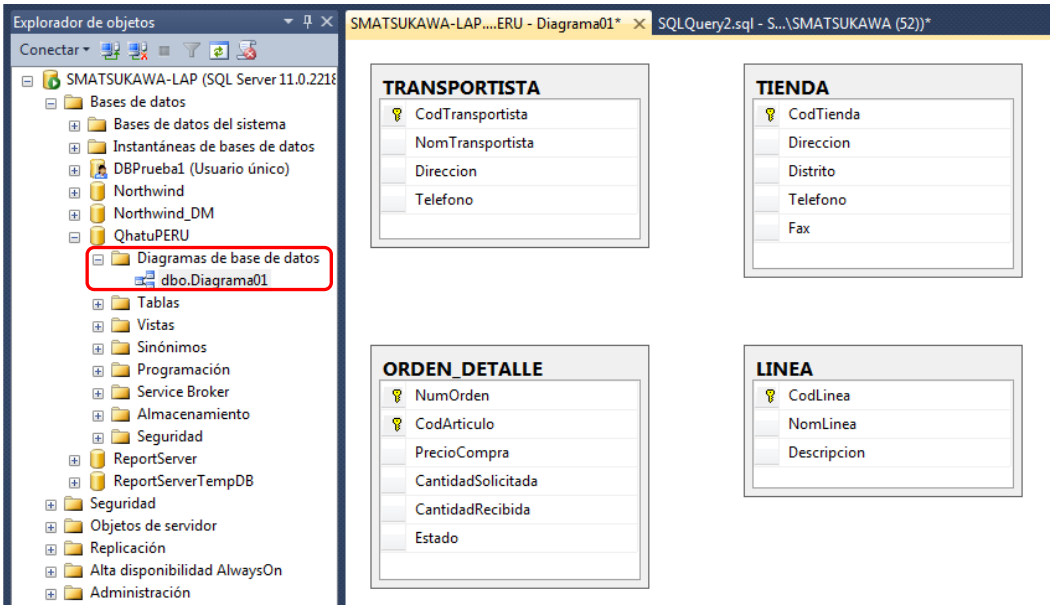
```
INSERT INTO ARTICULO
     VALUES(1,14,
     'CARAMELOS BASTON VIENA ARCOR',
     'PAQUETE 454 GR',1.50,200,50,0)
INSERT INTO ARTICULO
     VALUES(3,5,
     'ACEITE BABY JOHNSONS C/ALOE Y VIT. E',
     'FRASCO 100 ML',3.90,175,100,0)
INSERT INTO ARTICULO
     VALUES(2,1,
     'JAMONADA LAIVE',
     'KILOGRAMO',12.50,80,75,0)
INSERT INTO ARTICULO
```

```
        VALUES(4,1,
        'CREMA DE LECHE LAIVE',
        'ENVASE 160 GR',2.00,250,250,0)
go
```

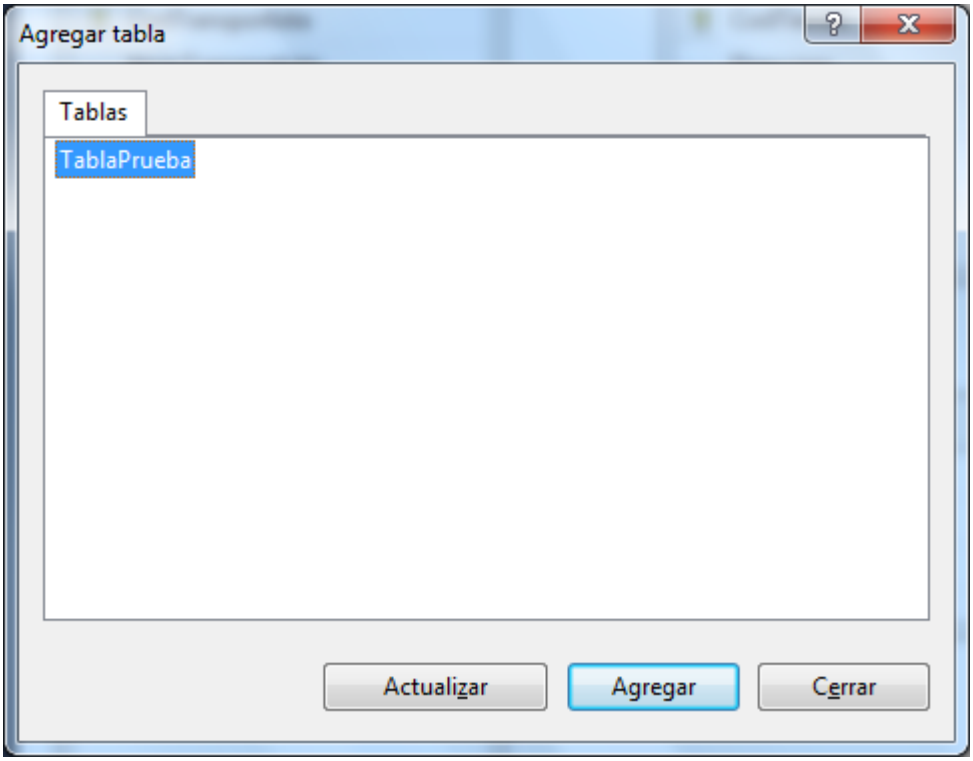
3. Las 3 primeras instrucciones INSERT se ejecutan sin problemas. La última genera un error debido a que la línea cuyo **CodLinea** es **4** aún no se ha registrado.

```
Mens. 547, Nivel 16, Estado 0, Línea 7
Instrucción INSERT en conflicto con la restricción FOREIGN KEY "fk_Articulo_Linea". El conflicto ha aparecido en la base de datos "QhatuPERU", tabla "dbo.LINEA", column 'CodLinea'.
Se terminó la instrucción.
```

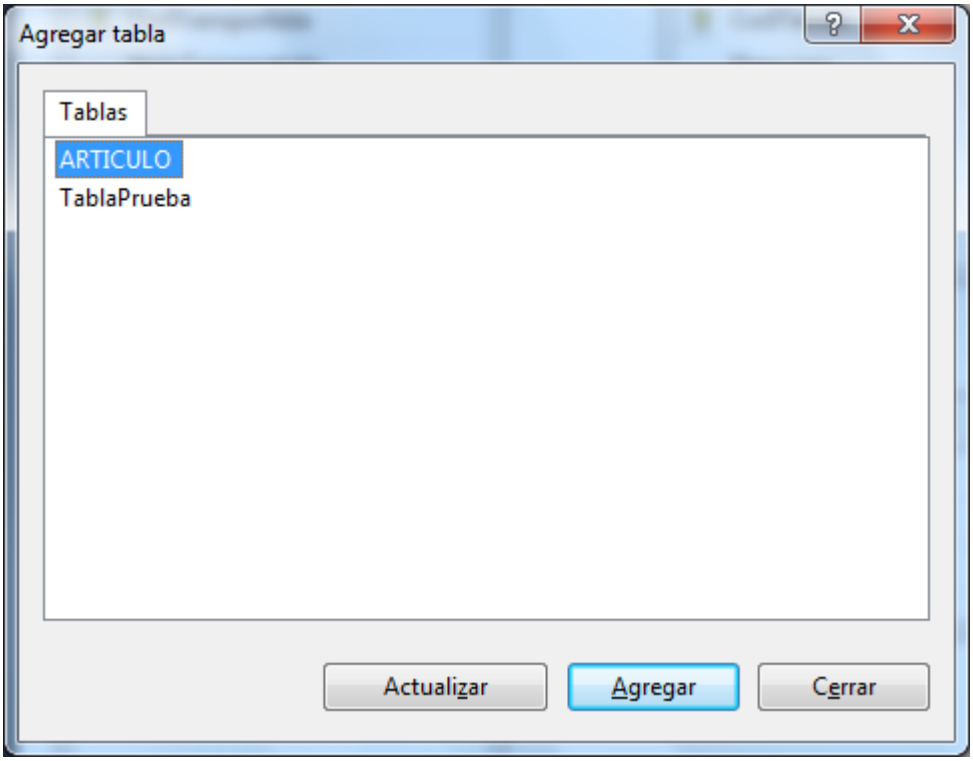
4. Abra el diagrama **Diagrama01** (si no lo tiene abierto) que creó en uno de los ejercicios anteriores. Para ello, en el nodo de la base de datos **QhatuPERU**, expanda **Diagramas de base de datos**, y haga doble clic en **Diagrama0**1.



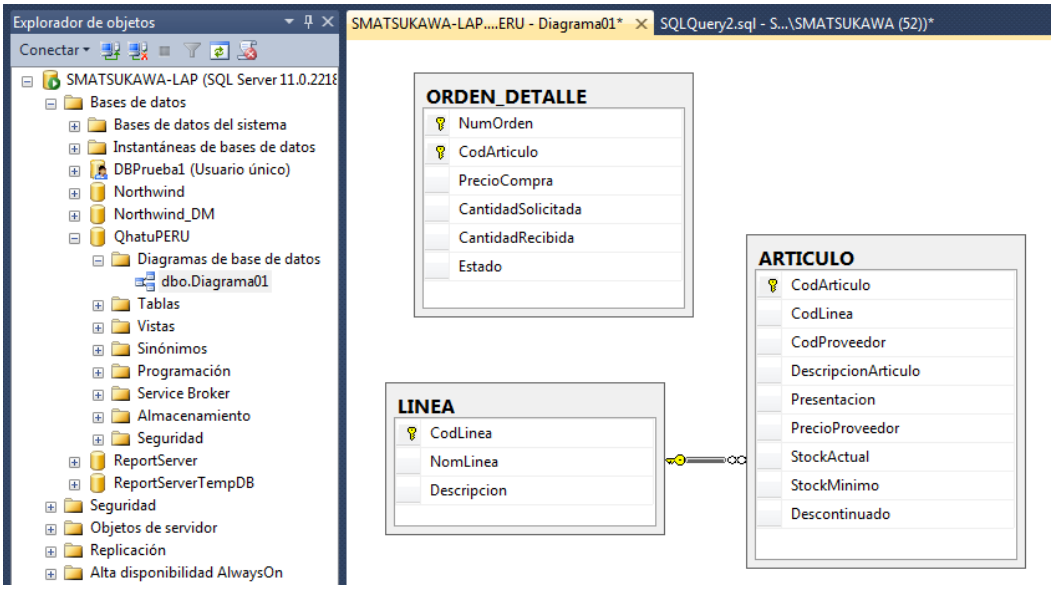
5. Haga clic secundario sobre un área libre del diagrama, y en el menú contextual seleccione **Agregar tabla**. Se abre la ventana **Agregar tabla**.



6. Haga clic en el botón **Actualizar** para que la ventana muestre las tablas creadas después de la creación del diagrama.



7. Seleccione la tabla ARTICULO y haga clic en **Agregar**, y luego en **Cerrar**. La tabla ARTICULO se añade al diagrama mostrando su relación con la tabla LINEA como consecuencia de la creación de su llave foránea.



8. Guarde el diagrama.

6.6. Creación de una restricción UNIQUE (valor no duplicado)

Sintáxis

```
ALTER TABLE nombre_tabla
   ADD CONSTRAINT U_nombre_tabla_nombre_columna
   UNIQUE( columnaX, columnaP, … )
```

- **U_nombre_tabla_nombre_columna** es el nombre de la restricción valor no duplicado ó UNIQUE. Se recomienda definir como nombre de la restricción, el nombre de la tabla seguido del nombre de la columna afectada, todo con el prefijo **U_**. Si no se especfica el nombre de la restricción, SQL Server le asigna un nombre.
- **columnaX**, **columnaP**, es la columna ó combinación de columnas a la que se aplica la restricción.

Ejercicio 3.13: Creación de restricción UNIQUE para la columna NomLinea en la tabla LINEA

Se desea que el nombre de una línea no se duplique. Para garantizar el cumplimiento de esta regla creamos una restricción UNIQUE en la columna **NomLinea** de la tabla LINEA.

1. En su ventana Query ejecute las siguientes instrucciones:

```
USE QhatuPERU
go

ALTER TABLE LINEA
  ADD CONSTRAINT U_Linea_NomLinea
  UNIQUE( NomLinea )
go
```

2. Pruebe la restricción ejecutando la siguiente instrucción:

```
INSERT INTO LINEA
     VALUES('HIGIENE PERSONAL','Por definir')
go
```

3. Se recibe el siguiente mensaje de error debido a que la línea de nombre HIGIENE PERSONAL ya está registrada.

```
Mens. 2627, Nivel 14, Estado 1, Línea 1
```

```
Infracción de la restricción UNIQUE KEY
'U_Linea_NomLinea'. No se puede insertar una
clave duplicada en el objeto 'dbo.LINEA'. El
valor de la clave duplicada es (HIGIENE
PERSONAL).
Se terminó la instrucción.
```

6.7. Creación de una restricción DEFAULT (valor predeterminado)

Sintáxis

```
ALTER TABLE nombre_tabla
   ADD CONSTRAINT DF_nombre_tabla_nombre_columna
   DEFAULT valor_predeterminado FOR columnaX
```

- **DF_nombre_tabla_nombre_columna** es el nombre de la restricción valor predeterminado ó DEFAULT. Se recomienda definir como nombre de la restricción, el nombre de la tabla seguido del nombre de la columna afectada, todo con el prefijo **DF_**. Si no se especfica el nombre de la restricción, SQL Server le asigna un nombre.
- **valor_predeterminado** es el valor que se almacena en columnaX cuando al insertar una fila no se especifica el valor para esa columna.
- **columnaX** es la columna a la que se aplica la restricción.

Ejercicio 3.14: Creación de restricción DEFAULT para la columna Descontinuado en la tabla ARTICULO

Para la columna **Descontinuado** el valor por defecto es **0** (cero).

1. Ejecute en su ventana Query las siguientes instrucciones:

```
USE QhatuPERU
go

ALTER TABLE ARTICULO
  ADD CONSTRAINT DF_Articulo_Descontinuado
  DEFAULT 0 FOR Descontinuado
go
```

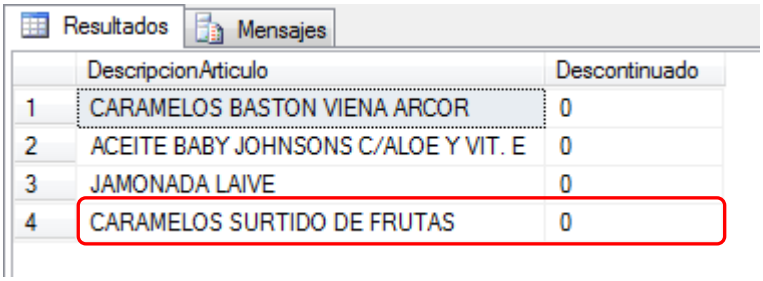
2. Para probar la restricción ejecute la siguiente instrucción:

```
INSERT INTO ARTICULO(CodLinea,
```

```
        CodProveedor, DescripcionArticulo)
        VALUES(1,15,'CARAMELOS SURTIDO DE FRUTAS')
go
```

3. Consulte la tabla para verificar el valor en la columna **Descontinuado** para el artículo CARAMELOS SURTIDO DE FRUTAS.

```
SELECT DescripcionArticulo, Descontinuado
FROM ARTICULO
go
```



	DescripcionArticulo	Descontinuado
1	CARAMELOS BASTON VIENA ARCOR	0
2	ACEITE BABY JOHNSONS C/ALOE Y VIT. E	0
3	JAMONADA LAIVE	0
4	CARAMELOS SURTIDO DE FRUTAS	0

6.8. Creación de una restricción CHECK (regla de validación)

Sintáxis

```
ALTER TABLE nombre_tabla
   ADD CONSTRAINT CK_nombre_tabla_nombre_columna
   CHECK( condición )
```

- **CK_nombre_tabla_nombre_columna** es el nombre de la restricción regla de validación ó CHECK. Se recomienda definir como nombre de la restricción, el nombre de la tabla seguido del nombre de la columna afectada, todo con el prefijo **CK_**. Si no se especifica el nombre de la restricción, SQL Server le asigna un nombre.
- **condición** es la expresión que determina cómo debe ser el valor a ingresar en la columna afectada por la restricción.

Ejercicio 3.15: Creación de restricción CHECK para la columna PrecioProveedor de la tabla ARTICULO

El precio de un artículo no puede ser menor que o (cero)

1. En su ventana Query escriba la siguiente instrucción:

```
USE QhatuPERU
go

ALTER TABLE ARTICULO
  ADD CONSTRAINT CK_Articulo_PrecioProveedor
  CHECK( PrecioProveedor >= 0 )
go
```

2. Pruebe la regla ejecutando la siguiente instrucción:

```
INSERT INTO ARTICULO(CodLinea, CodProveedor,
     DescripcionArticulo, Presentacion,
     PrecioProveedor)
     VALUES(2,12,
     'JAMONADA ESPECIAL LA SEGOVIANA',
     'KILOGRAMO',-15)
go
```

3. La instrucción genera error debido a que el precio no puede ser negativo. Se recibe el siguiente mensaje:

```
Mens. 547, Nivel 16, Estado 0, Línea 1
Instrucción INSERT en conflicto con la
restricción CHECK "CK_Articulo_PrecioProveedor".
El conflicto ha aparecido en la base de datos
"QhatuPERU", tabla "dbo.ARTICULO", column
'PrecioProveedor'.
Se terminó la instrucción.
```

7. INTEGRIDAD REFERENCIAL EN CASCADA

Cuando se intenta eliminar una fila de una tabla a la que apuntan claves foráneas, la eliminación falla debido a que las filas que contienen las claves foráneas no pueden quedar "huérfanas". Por ejemplo, cuando intentamos eliminar a una línea que tiene artículos registrados.

La integridad referencial en cascada permite controlar las acciones que lleva a cabo SQL Server cuando se intenta actualizar o eliminar una clave primaria a la que apuntan claves foráneas existentes. Esto se controla mediante las cláusulas ON DELETE y ON UPDATE en la cláusula REFERENCES de las instrucciones CREATE TABLE y ALTER TABLE.

La cláusula ON DELETE controla las acciones que se llevarán a cabo si intenta eliminar una fila a la que apuntan las claves foráneas existentes. A partir de SQL Server 2005 la cláusula ON DELETE tiene cuatro opciones:

- NO ACTION especifica que la eliminación produce un error.
- CASCADE especifica que también se eliminan todas las filas con claves foráneas que apuntan a la fila eliminada.
- SET NULL especifica que todas las filas con claves foráneas que apuntan a la fila eliminada tendrán el valor NULL en la clave foránea.
- SET DEFAULT especifica que todas las filas con claves foráneas que apuntan a la fila eliminada se configurarán al valor predeterminado en la clave foránea.

La cláusula ON UPDATE define las acciones que se llevarán a cabo si intenta actualizar un valor de clave candidata a la que apuntan las claves foráneas existentes. También acepta las opciones NO ACTION, CASCADE, SET NULL y SET DEFAULT.

Ejercicio 3.16: Definición de integridad referencial en cascada para las tablas LINEA y ARTICULO

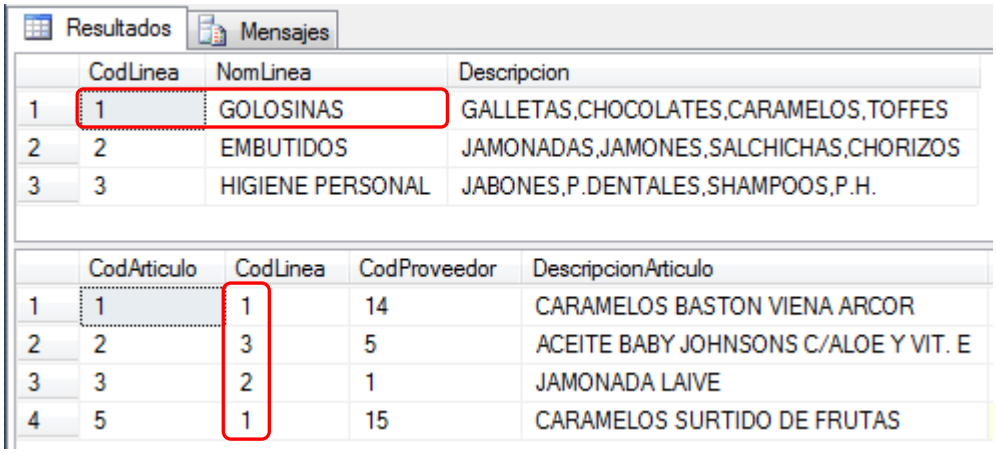
Para la base de datos **QhatuPERU** defina que al eliminar una línea en la tabla LINEA se eliminen también todos los artículos registrados en la tabla ARTICULO asociados a dicha línea.

1. Ejecute las siguientes instrucciones:

```
USE QhatuPERU
go

SELECT * FROM LINEA
SELECT * FROM ARTICULO
go
```

2. Observe que la línea 1 tiene registrados 2 artículos.



Resultados | Mensajes

	CodLinea	NomLinea	Descripcion
1	1	GOLOSINAS	GALLETAS,CHOCOLATES,CARAMELOS,TOFFES
2	2	EMBUTIDOS	JAMONADAS,JAMONES,SALCHICHAS,CHORIZOS
3	3	HIGIENE PERSONAL	JABONES,P.DENTALES,SHAMPOOS,P.H.

	CodArticulo	CodLinea	CodProveedor	DescripcionArticulo
1	1	1	14	CARAMELOS BASTON VIENA ARCOR
2	2	3	5	ACEITE BABY JOHNSONS C/ALOE Y VIT. E
3	3	2	1	JAMONADA LAIVE
4	5	1	15	CARAMELOS SURTIDO DE FRUTAS

3. Procedemos a definir la integridad referencial en cascada. Para ello vamos a eliminar la llave foránea de la tabla ARTICULO que apunta a la tabla LINEA para volverla a definir.

```
-- Eliminando la clave foránea de la
-- tabla ARTICULO para volverla a crear
-- estableciendo la eliminación en cascada
ALTER TABLE ARTICULO
     DROP CONSTRAINT FK_Articulo_Linea
go
```

4. Ahora creamos la clave foránea definiendo la eliminación en cascada:

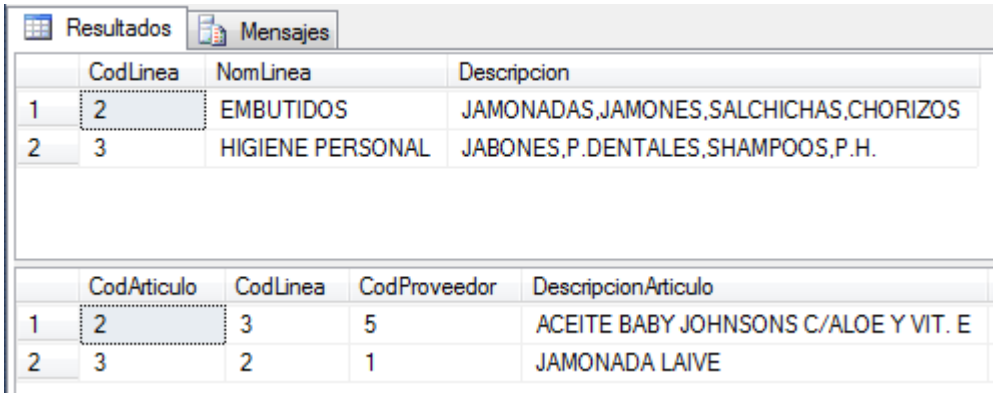
```
ALTER TABLE ARTICULO
     ADD CONSTRAINT FK_Articulo_Linea
     FOREIGN KEY( CodLinea )
     REFERENCES LINEA
     ON DELETE CASCADE
go
```

5. Vamos a eliminar la línea 1 para ver el efecto de la integridad referencial en cascada.

```
-- Eliminando la línea 1
DELETE FROM LINEA
     WHERE CodLinea = 1
go
```

6. Comprobamos que se ha eliminado la línea 1 y además los artículos que tenía registrados.

```
SELECT * FROM LINEA
SELECT * FROM ARTICULO
go
```



Resultados | Mensajes

	CodLinea	NomLinea	Descripcion
1	2	EMBUTIDOS	JAMONADAS,JAMONES,SALCHICHAS,CHORIZOS
2	3	HIGIENE PERSONAL	JABONES,P.DENTALES,SHAMPOOS,P.H.

	CodArticulo	CodLinea	CodProveedor	DescripcionArticulo
1	2	3	5	ACEITE BABY JOHNSONS C/ALOE Y VIT. E
2	3	2	1	JAMONADA LAIVE

8. ELIMINACIÓN DE TABLAS (LA INSTRUCCIÓN DROP TABLE)

Sintáxis

```
DROP TABLE nombre_tabla
```

Elimina la definición de una tabla, y todos los datos, índices, desencadenantes, restricciones y permisos especificados para dicha tabla.

Si la tabla es referenciada por una clave foránea, no se podrá eliminar. Primero debe eliminarse la tabla que contiene la clave foránea, y luego la tabla referenciada.

Ejercicio 3.17: Eliminación de tablas

1. Ejecute la siguiente instrucción en su ventana Query:

```
DROP TABLE LINEA
go
```

2. Se recibe el siguiente mensaje de error:

```
Mens. 3726, Nivel 16, Estado 1, Línea 1
No se puede quitar el objeto 'LINEA'. Hay una
referencia a él en una restricción FOREIGN KEY.
```

3. Ahora, ejecute la siguiente instrucción:

```
DROP TABLE ORDEN_DETALLE
go
```

La tabla ORDEN_DETALLE se elimina sin problemas.

9. EJERCICIO PROPUESTO

Con las especificaciones presentadas en el punto 3.1 de este capítulo escriba todas las instrucciones necesarias para crear la base de datos QhatuPERU y todas sus tablas y relaciones. Previamente deberá eliminar la base de datos creada en este capítulo. Para ello, en la ventana Query ejecute las siguientes instrucciones:

```
USE master
go

DROP DATABASE QhatuPERU
go
```