



Creación de tablas

En este capítulo veremos los tipos de datos disponibles en SQL Server 2005, cómo crear tablas, cómo administrar las tablas, y el diseño de la integridad de datos.

Esta página se ha dejado en blanco intencionalmente.

Capítulo 4

Creación de tablas

Contenido

- ❑ *Los tipos de datos SQL Server 2005*
 - ✓ *Tipos de datos numéricos exactos*
 - ✓ *Tipos de datos numéricos aproximados*
 - ✓ *Tipos de datos especiales*
 - ✓ *Tipos de datos de fecha y hora*
 - ✓ *Tipo de dato moneda*
 - ✓ *Tipo de dato timestamp*
- ❑ *Creación de tablas*
 - ✓ **Ejercicio 14:** *Creación de la base de datos ExtPro*
 - ✓ *La instrucción CREATE TABLE*
 - *Especificación de NULL ó NOT NULL*
 - *Valores nulos*
 - ✓ **Ejercicio 15:** *Creación de la tabla Curso*
 - ✓ *El procedimiento sp_help*
 - ✓ **Ejercicio 16:** *Verificación de la definición de la tabla Curso*
 - ✓ **Ejercicio 17:** *Creación de las tablas Alumno, Matricula y Pago*
- ❑ *Modificación de la definición de una tabla*
 - ✓ *La instrucción ALTER TABLE*
 - ✓ **Ejercicio 18:** *Adición de una columna a una tabla*
 - ✓ **Ejercicio 19:** *Adición de una columna NOT NULL a una tabla*

- ❑ *Integridad de datos*
 - ✓ *Niveles de la integridad de datos*
 - *Integridad de dominio*
 - *Integridad de entidad*
 - *Integridad referencial*
- ❑ *Las restricciones (constraints)*
 - ✓ *Tipos de restricciones*
 - ✓ *Creación de clave primaria (PK)*
 - ✓ **Ejercicio 20:** *Creación de la clave primaria de la tabla Curso*
 - ✓ **Ejercicio 21:** *Creación de una clave primaria compuesta en la tabla Matricula*
 - ✓ **Ejercicio 22:** *Creación de la clave primaria de las tablas Alumno y Pago*
 - ✓ *Creación de clave foránea (FK)*
 - ✓ **Ejercicio 23:** *Creación de la clave foránea en la tabla Matricula que referencia a la tabla Curso*
 - ✓ **Ejercicio 24:** *Creación de la clave foránea en la tabla Matricula que referencia a la tabla Alumno*
 - ✓ **Ejercicio 25:** *Creación de la clave foránea en la tabla Pago que referencia a la tabla Matricula*
 - ✓ *Creación de restricción valor no duplicado (UNIQUE)*
 - ✓ **Ejercicio 26:** *Creación de restricción UNIQUE para la columna nombreCurso en la tabla Curso*
 - ✓ *Creación de valor predeterminado (DEFAULT)*
 - ✓ **Ejercicio 27:** *Creación de restricción DEFAULT para la columna numVacantes en la tabla Curso*

- ✓ Creación de regla de validación (CHECK)
- ✓ **Ejercicio 28:** Creación de restricción CHECK para la columna precioCurso de la tabla Curso
- ✓ **Ejercicio 29:** Prueba de la integridad referencial en la base de datos ExtPro
- ✓ Integridad referencial en cascada
- ✓ **Ejercicio 30:** Definición de integridad referencial en cascada para las tablas Matricula y Pago
- Eliminación de tablas (la instrucción DROP TABLE)
 - ✓ **Ejercicio 31:** Eliminación de tablas

Esta página se ha dejado en blanco intencionalmente.

Creación de tablas

Antes de crear una tabla, se debe definir el tipo de datos para cada columna de la tabla. Los tipos de los datos especifican el tipo de información (los caracteres, números, o fechas) que una columna puede almacenar.

Los tipos de datos SQL Server

SQL Server 2005 presenta un nuevo tipo de dato, el tipo **xml**, utilizado para soportar datos XML; otros tipos tienen mejoras.

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.

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... (repetiendo). 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.

Tipos de datos especiales

bit

El tipo de dato **bit** es un tipo de dato lógico que se usa para almacenar información booleana. Los tipos de datos booleanos se utilizan como marcadores para expresar criterios como encendido/apagado, cierto/falso, y sí/no. Los valores se almacenan como 0 o 1.

Las columnas de tipo bit pueden tener el valor NULL (desconocido) y no pueden ser indexadas.

text e image

Los tipos de datos **text** e **image** se usan cuando los requerimientos de almacenamiento exceden al límite de columna de 8,000 caracteres. A menudo, a estos tipos de datos se les hace referencia como BLOBs. Los tipos de datos text e image pueden almacenar hasta 2 GB de datos binarios o de texto.

sql_variant

El tipo de datos **sql_variant** es un tipo de datos especial que le permite almacenar valores de múltiples tipos de datos base en la misma columna. Por ejemplo, en la misma columna puede almacenar valores nchar, valores int y valores decimal.

table

SQL Server también admite un tipo de datos base **table**, que se puede utilizar para almacenar el conjunto de resultados de una instrucción SQL. No se puede utilizar el tipo de datos table en columnas de una tabla. Sólo es posible utilizarlo en variables Transact-SQL y en los valores de retorno de funciones definidas por el usuario.

Tipos de datos de fecha y hora

La fecha y hora pueden almacenarse en un tipo de dato **datetime** o bien en uno **smalldatetime**. La fecha y hora siempre se almacenan juntas en un solo valor.

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 1999.

```
'Abr 15 1999'  
'Abr 15 99'  
'Abr 99 15'  
'15 Abr 99'  
'1999 Abril 15'  
'1999 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 1999.

```
4/15/99 (mm/dd/aa)  
4/99/15 (mm/aa/dd)  
15/99/04 (dd/aa/mm)  
99/15/04 (aa/mm/dd)
```

Los datos almacenados en el tipo de dato `datetime` se guardan hasta el milisegundo.

Se utiliza un total de 8 bytes, entre un intervalo de fechas de 01/01/1753 hasta 31/12/9999.

El tipo de datos `smalldatetime` utiliza un total de 4 bytes. Las fechas almacenadas en este formato son precisas hasta el minuto. Esta se encuentra entre un intervalo de fecha de 01/01/1900 hasta 06/06/2079

Tipo de dato moneda

Hay dos tipos de datos moneda: **money** y **smallmoney**. Ambas tienen una escala de cuatro, lo que significa que almacenan cuatro dígitos a la derecha del punto decimal.

Estos tipos de datos pueden almacenar para uso internacional unidades distintas a dólares, pero no hay disponibles en SQL Server funciones de conversión de moneda.

Al ingresar datos monetarios, debe antecederlos con un signo dólar.

Tipo de dato timestamp

Cada vez que agregue un nuevo registro a una tabla con un campo **timestamp**, se agregarán valores de hora de forma automática; pero no solo esto, timestamp va un poco más allá. Si realiza una actualización a una fila, timestamp se actualizará a sí mismo en forma automática.

El tipo de dato timestamp crea un valor único, generado por SQL Server, que se actualiza automáticamente. Aunque el tipo timestamp luce como un tipo de dato datetime, no lo es. Los tipos de datos timestamp se almacenan como binary(8) para columnas NOT NULL o Varbinary(8) si la columna está marcada para permitir valores nulos.

El siguiente cuadro muestra una relación de los distintos tipos de datos soportados por SQL Server 2005.

Categoría	Tipo de datos	Comentario
Cadena	char(<i>n</i>)	Cadena de longitud fija (n puede ser de 1 a 8000 bytes).
	varchar(<i>n</i> max)	Cadena de longitud variable (n puede ser de 1 a 8000 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.
Entero	bit	Puede tomar el valor 1, 0 ó NULL.
	tinyint	Entero de 1 byte.
	smallint	Entero de 2 bytes.
	int	Entero de 4 bytes
	bigint	Entero de 8 bytes.
Numérico exacto	decimal(<i>p</i> , <i>s</i>) *	p es la precisión, y va de 1 a 38, siendo 18 el valor predeterminado. s es la escala y va de 0 hasta p.
	numeric(<i>p</i> , <i>s</i>) *	p es la precisión, y va de 1 a 38, siendo 18 el valor predeterminado. s es la escala y va de 0 hasta p.
Numérico aproximado	float	- 1.79E + 38 a -2.23E - 38, 0 y 2.23E -38 a 1.79E + 38.
	real	-1.18E - 38, 0 y 1.18E - 38 a 3.40E + 38.

Categoría	Tipo de datos	Comentario
Fecha y hora	datetime	Fecha en el rango del 1 de enero de 1753 al 31 de diciembre de 9999, con una precisión de 3.33 milisegundos.
	smalldatetime	Fecha en el rango del 1 de enero de 1900 al 6 de junio del 2079, con una precisión de 1 minuto.
Binario	binary(<i>n</i>)	Dato binario de longitud fija (n puede ser de 1 a 8000 bytes).
	varbinary(<i>n</i> 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.
Especial	image **	Dato binario de longitud variable que va desde 0 hasta $2^{31} - 1$ bytes.
	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.
	xml(xml_esquema)	Almacena data XML.
	cursor ****	Tipo de dato para variables ó parámetros de salida de procedimientos almacenados.

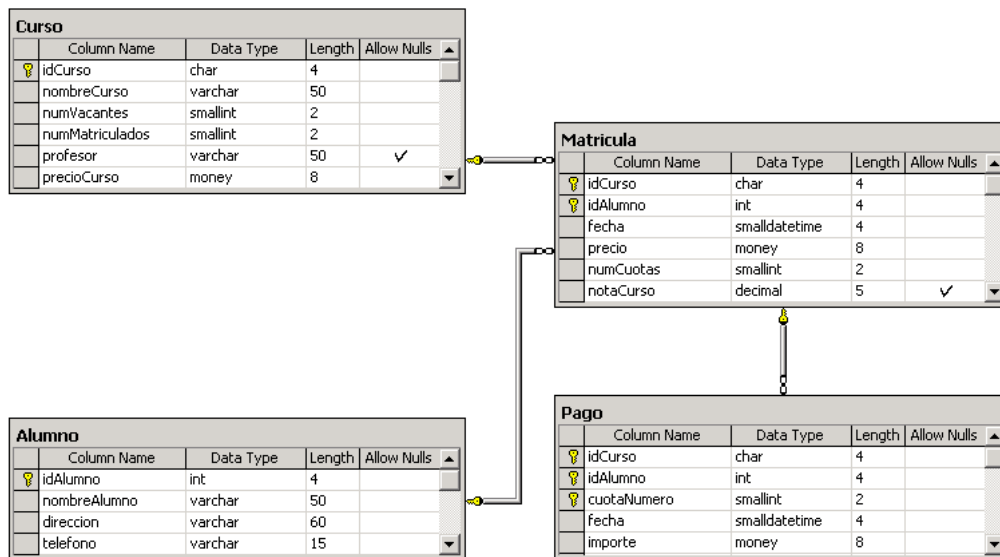
Categoría	Tipo de datos	Comentario
Moneda	money	Valor monetario de 8 bytes.
	smallmoney	Valor monetario de 4 bytes.
Datos UNICODE	nchar(<i>n</i>)	Cadena UNICODE de longitud fija de n caracteres (n puede ser de 1 a 4000 caracteres).
	nvarchar(<i>n</i> max)	Cadena UNICODE de longitud variable (n puede ser de 1 a 4000 caracteres). max indica la longitud máxima que es $2^{31} - 1$ bytes.
	ntext **	Cadena UNICODE de longitud variable con un máximo de $2^{30} - 1$ caracteres.
Generación automática	timestamp ***	Generación automática de números binarios de modo que son únicos dentro de una base de datos.

- * **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.
- ** **ntext**, **text** e **image** no serán considerados en versiones futuras de SQL Server. En lugar de ellas utilice **nvarchar(max)**, **varchar(max)** y **varbinary(max)**, mejoras introducidas en SQL Server 2005.
- *** En SQL ANSI 92, **timestamp** es equivalente a **datetime** de Transact-SQL. En versiones futuras de SQL Server, **timestamp** se modificará para que se comporte como el estándar ANSI 92. El tipo **timestamp** actual será reemplazado por el tipo **rowversion**.
- **** **table** y **cursor** no se pueden utilizar para definir columnas con el comando CREATE TABLE:

Creación de tablas

Para desarrollar este tema, presentaremos un caso ejemplo muy sencillo. Se desea crear la base de datos de una institución que brinda cursos de extensión profesional. Los interesados en los cursos se pueden matricular sin ninguna restricción, y además tienen facilidades para pagar el costo de los cursos.

A continuación se presenta el modelo físico SQL Server de la base de datos a crear. El nombre de la base de datos será **ExtPro**.



Ejercicio 14: Creación de la base de datos ExtPro

1. Conéctese a su servidor SQL abriendo un nuevo query.
2. Ejecute las siguientes instrucciones:

```
USE Master  
go
```

```
CREATE DATABASE ExtPro  
go
```

La instrucción CREATE TABLE

Sintáxis

```
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 ] ,  
    ... )
```

Especificación de NULL ó NOT NULL

Una columna con la propiedad NULL indica que el valor a ingresar en esa columna es opcional (no es un dato obligatorio). Si la columna tiene la propiedad NOT NULL, el ingreso de un valor en dicha columna es obligatorio. Por ejemplo, para los datos de una persona podemos especificar que el registro de su nombre es obligatorio, por lo que definiríamos la columna **nombrePersona** como NOT NULL; sin embargo, para el caso de su número de teléfono celular podemos establecer que el dato no es obligatorio, por lo que la columna **telefonoMovil** podemos definirla como NULL.

Cuando no se especifica si una columna es NULL o NOT NULL, SQL Server determina esta propiedad de la columna en base al estado de la opción **ANSI null default** de la base de datos. Para evitar conflictos debido al cambio de configuración de la base de datos, indique en forma explícita para cada columna si será NULL o NOT NULL.

Valores nulos

Las columnas pueden aceptar ó rechazar los valores nulos. Cuando en una columna de una fila de una tabla se guarda el valor NULL 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 0. Todos estos representan un dato conocido, el valor NULL significa dato desconocido.

Cuando una columna permite valores nulos, aumenta la complejidad de las comparaciones lógicas que usan dicha columna. El estándar SQL ANSI 92 establece que cualquier comparación contra un valor NULL no devuelve ni TRUE (verdadero) ni FALSE (falso). Dicha comparación se evalúa como UNKNOWN (desconocido). Esto establece tres valores posibles como resultado de una comparación.

Ejercicio 15: Creación de la tabla Curso

1. Ejecute las siguientes instrucciones:

```
USE ExtPro
go

CREATE TABLE Curso(
    idCurso          CHAR(4) NOT NULL ,
    nombreCurso      VARCHAR(50) NOT NULL ,
    numVacantes      SMALLINT NOT NULL ,
    numMatriculados  SMALLINT NOT NULL ,
    profesor         VARCHAR(50) NULL ,
    precioCurso      MONEY NOT NULL )
go
```

El procedimiento sp_help

Genera un reporte con información acerca de la definición de un objeto de la base de datos activa. Todos los usuarios de la base de datos pueden ejecutar este procedimiento.

Sintáxis

```
sp_help nombre_objeto_basedatos
```

Ejercicio 16: Verificación de la definición de la tabla Curso

1. En su query, ejecute la siguiente instrucción:

```
sp_help Curso
go
```

Script04-01-Cr...EY2005.ExtPro* Summary						
	Name	Owner	Type	Created_datetime		
1	Curso	dbo	user table	2005-06-22 08:1...		
	Column_name	Type	Computed	Length	Prec	Sc
1	idCurso	char	no	4		
2	nombreCurso	varchar	no	50		
3	numVacantes	smallint	no	2	5	0
4	numMatriculados	smallint	no	2	5	0
5	profesor	varchar	no	50		
6	precioCurso	money	no	8	19	4
	Identity	Seed	Increment	Not For Replicati...		
1	No identity colu...	NULL	NULL	NULL		

Ejercicio 17: Creación de las tablas Alumno, Matricula y Pago

1. Como ejercicio, en su query escriba las instrucciones para crear las tablas **Alumno, Matricula y Pago**

-- Creación de la tabla Alumno

-- Creación de la tabla Matricula

-- Creación de la tabla Pago

Modificación de la definición de una tabla

En una tabla podemos añadir nuevas columnas, eliminar columnas, cambiar las propiedades de una columna, añadir ó eliminar restricciones.

La instrucción ALTER TABLE

Permite modificar la definición de una tabla.

Sintaxis

```
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 añadir una nueva columna a la tabla.
- DROP COLUMN se usa para eliminar una columna.
- ALTER COLUMN permite modificar la definición de una columna.
- ADD CONSTRAINT permite añadir una restricción PRIMARY KEY, UNIQUE, FOREIGN KEY, DEFAULT ó CHECK a la definición de una tabla.
- DROP CONSTRAINT se usa para eliminar una restricción.

Ejercicio 18: Adición de una columna a una tabla

1. Para ver el efecto de la instrucción ALTER TABLE, en la base de datos **ExtPro** crearemos una tabla de nombre **TablaPrueba**, y luego empezaremos a modificar su definición.

```
USE ExtPro
go

CREATE TABLE TablaPrueba(
    columna1 INT NOT NULL ,
    columna2 CHAR(10) NOT NULL ,
    columna3 DATETIME NULL )
go
```

2. Ahora, le añadiremos una columna de tipo MONEY con la propiedad NULL.

```
ALTER TABLE TablaPrueba
    ADD columna4 MONEY NULL
go
```

3. Revisamos la nueva definición de la tabla **TablaPrueba**.

```
sp_help TablaPrueba
go
```

Ejercicio 19: Adición de una columna NOT NULL a una tabla

1. Ahora, trataremos de añadirle una columna de tipo VARCHAR con la propiedad NOT NULL.

```
ALTER TABLE TablaPrueba  
    ADD columna5 VARCHAR(20) NOT NULL  
go
```

2. Obtenemos el siguiente mensaje de error:

```
Msg 4901, Level 16, State 1, Line 1  
ALTER TABLE only allows columns to be added that can  
contain nulls or have a DEFAULT definition specified.  
Column 'columna5' cannot be added to table 'TablaPrueba'  
because it does not allow nulls and does not specify a  
DEFAULT definition.
```

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.

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, primero debemos añadirla como una columna NULL, luego debemos registrar el valor para esa columna de las filas que la tabla ya tiene, y finalmente cambiamos la propiedad NULL de la nueva columna a NOT NULL.

3. Añadimos la columna VARCHAR con la propiedad NULL.

```
ALTER TABLE TablaPrueba  
    ADD column5 VARCHAR(20) NULL  
go
```

4. Ahora, cambiamos la propiedad NULL de dicha columna a NOT NULL.

```
ALTER TABLE TablaPrueba  
    ALTER COLUMN column5 VARCHAR(20) NOT NULL  
go
```

Integridad de datos

La integridad de los datos se refiere a la consistencia y exactitud de los datos que se guardan en una base de datos.

Niveles de la integridad de datos

La integridad de datos se define en los siguientes niveles:

Integridad de dominio

Se conoce como el **dominio** de un atributo al conjunto de valores aceptables para dicho atributo. La integridad de dominio establece qué condiciones deben cumplir los valores a insertar 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 una tabla **Empleado** se puede requerir que el valor mínimo válido para la columna **sueldoBase** sea 420; cualquier valor menor a éste debe ser rechazado por la base de datos.

Integridad de entidad

Una tabla almacena los datos de cada una de las ocurrencias de una 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 **Empleado**, cada fila de la tabla debe representar a solo un empleado; un empleado no puede aparecer registrado en dos filas de la tabla.

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. 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 **Dependencia** y **Empleado**, cada empleado debe pertenecer a solo una dependencia; es decir, cada fila de la tabla **Empleado** debe estar relacionada con una fila de la tabla **Dependencia**, pero una fila de la tabla **Dependencia** puede estar relacionada con muchas filas de la tabla **Empleado** ya que una dependencia puede tener muchos empleados.

Las restricciones (constraints)

Las restricciones son un método declarativo de definición de la integridad de datos ya que ellas se definen al momento de crear la tabla (con la sentencia CREATE TABLE), o al momento de modificar la definición de la tabla (con la sentencia ALTER TABLE).

En otras palabras, una restricción forma parte de la definición de la tabla. Las restricciones son el método preferido para dar fuerza a la integridad de los datos.

Tipos de restricciones

Las restricciones son un método estándar ANSI para forzar la integridad de los datos.

Garantizan que los datos ingresados en las columnas sean valores válidos y que se mantengan las relaciones entre las tablas.

La tabla siguiente describe los diferentes tipos de restricciones.

Tipo de restricción	Descripción
PRIMARY KEY (clave primaria)	Garantiza que cada fila ó registro en una tabla es único(a). 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. Permite valores únicos
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.
DEFAULT (valor predeterminado)	Establece el valor predeterminado para una columna cuando al insertar una fila no se especifica el valor para dicha columna.
CHECK (regla de validación)	Establece la regla que debe cumplir un valor para que sea un valor aceptable en una columna.

Creación de clave primaria (PK)

Sintaxis

```
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 especifica 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 20: Creación de la clave primaria de la tabla

Curso

1. En su query, ejecute las siguientes instrucciones para definir la columna **idCurso** como la clave primaria de la tabla **Curso**.

```
USE ExtPro
go

ALTER TABLE Curso
    ADD CONSTRAINT PK_curso
    PRIMARY KEY( idCurso )
go
```

2. Para verificar el comportamiento de la PK de la tabla **Curso**, vamos a insertar algunas filas.

```

INSERT INTO Curso(idCurso, nombreCurso, numVacantes,
numMatriculados, precioCurso, profesor)
VALUES('PB1A', 'PowerBuilder Nivel I - Grupo A', 15,
0, 250, 'Coronel Castillo, Gustavo')

INSERT INTO Curso(idCurso, nombreCurso, numVacantes,
numMatriculados, precioCurso, profesor)
VALUES('PB1A', 'MS Power Point XP - Grupo A', 15,
0, 150, 'Zegarra Zavaleta, Daniel')
go

```

3. La primera instrucción INSERT se ejecuta sin problemas. La segunda genera el mensaje de error 2627 debido a que se viola la clave primaria de la tabla ya que el valor de la columna **idCurso** se está duplicando (PB1A).

```

(1 row(s) affected)
Msg 2627, Level 14, State 1, Line 1
Violation of PRIMARY KEY constraint 'PK_curso'. Cannot
insert duplicate key in object 'Curso'.
The statement has been terminated.

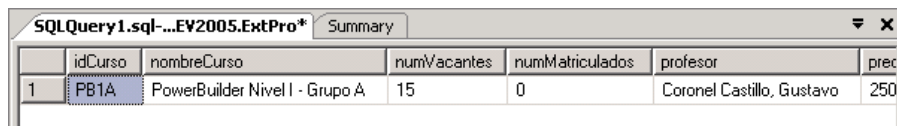
```

4. Para revisar la tabla y verificar la data ingresada ejecute:

```

SELECT * FROM Curso
go

```



	idCurso	nombreCurso	numVacantes	numMatriculados	profesor	precioCurso
1	PB1A	PowerBuilder Nivel I - Grupo A	15	0	Coronel Castillo, Gustavo	250

Ejercicio 21: Creación de una clave primaria compuesta en la tabla Matricula

Según el modelo de la base de datos, en la tabla **Matricula** la clave primaria está formada por las columnas **idCurso** e **idAlumno**, ya que para identificar una matrícula se necesita conocer al alumno matriculado y en qué curso se matriculó.

1. Ejecute la siguiente instrucción para crear la clave primaria de la tabla **Matricula**.

```
ALTER TABLE Matricula
    ADD CONSTRAINT PK_Matricula
    PRIMARY KEY( idCurso, idAlumno )
go
```

Ejercicio 22: Creación de la clave primaria de las tablas Alumno y Pago

1. Escriba las instrucciones para crear la clave primaria de las tablas **Alumno** y **Pago**. Note que según el modelo físico de la base de datos, la clave primaria de la tabla **Pago** es una clave compuesta de tres columnas.

```
-- PK de Alumno
```

```
-- PK de Pago
```

Creación de clave foránea

Sintaxis

```
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 especifica 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 23: Creación de la clave foránea en la tabla Matricula que referencia a la tabla Curso

1. En su query ejecute las siguientes instrucciones para crear la clave foránea de la tabla **Matricula** que referencia a la tabla **Curso**:

```
USE ExtPro
go

ALTER TABLE Matricula
    ADD CONSTRAINT fk_Matricula_Curso
    FOREIGN KEY( idCurso )
    REFERENCES Curso
go
```

La columna **idCurso** de la tabla **Matricula** establece la relación de esta tabla con la tabla **Curso**. La clave foránea en **idCurso** de **Martricula** apunta a la clave primaria en **idCurso** de la tabla **Curso**.

Ejercicio 24: Creación de la clave foránea en la tabla **Matricula** que referencia a la tabla **Alumno**

1. Escriba la instrucción para crear la clave foránea de la tabla **Matricula** que referencia a la tabla **Alumno**. Para ello, observe el modelo físico de datos presentado arriba.

```
-- FK de Matricula que apunta a Alumno
```

Ejercicio 25: Creación de la clave foránea en la tabla **Pago** que referencia a la tabla **Matricula**

1. Escriba la instrucción para crear la clave foránea de la tabla **Pago** que apunta a la clave primaria de la tabla **Matricula**. Note que según el modelo físico de la base de datos, la clave foránea de la tabla **Pago** es una clave compuesta de dos columnas.

```
ALTER TABLE Pago
    ADD CONSTRAINT fk_Pago_Matricula
    FOREIGN KEY( idCurso, idAlumno )
    REFERENCES Matricula
go
```

Creación de restricción valor no duplicado (UNIQUE)

Sintaxis

```
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 especifica 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 26: Creación de restricción UNIQUE para la columna nombreCurso en la tabla Curso

Supongamos que en la institución se ha establecido como regla que el nombre de un curso no se puede duplicar. Para garantizar el cumplimiento de la regla creamos una restricción UNIQUE en la columna **nombreCurso** de la tabla **Curso**.

1. Ejecute las siguientes instrucciones en su query:

```
USE ExtPro
go

ALTER TABLE Curso
    ADD CONSTRAINT U_Curso_nombreCurso
    UNIQUE( nombreCurso )
go
```

2. Ahora, pruebe la restricción ejecutando las siguientes inserciones:

```
INSERT INTO Curso(idCurso, nombreCurso, numVacantes,
    numMatriculados, precioCurso, profesor)
VALUES('PB1B', 'PowerBuilder Nivel I - Grupo B', 15,
    0, 250, 'Matsukawa Maeda, Sergio')
go

INSERT INTO Curso(idCurso, nombreCurso, numVacantes,
    numMatriculados, precioCurso, profesor)
VALUES('PB2A', 'PowerBuilder Nivel I - Grupo B', 15,
    0, 250, 'Matsukawa Maeda, Sergio')
go
```

La segunda inserción falla debido a que el nombre del curso se está duplicando. Se recibe el siguiente mensaje de error:

```
Msg 2627, Level 14, State 1, Line 1
Violation of UNIQUE KEY constraint 'U_Curso_nombreCurso'.
Cannot insert duplicate key in object 'Curso'.
The statement has been terminated.
```

Creación de valor predeterminado (DEFAULT)

Sintaxis

```
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 especifica 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 27: Creación de restricción DEFAULT para la columna numVacantes en la tabla Curso

Para cada curso, el número de vacantes por defecto es 15.

1. Para que lo anterior se cumpla debe crear en la columna **numVacantes** de la tabla **Curso** la siguiente restricción DEFAULT:

```
ALTER TABLE Curso
    ADD CONSTRAINT DF_Curso_numVacantes
    DEFAULT 15 FOR numVacantes
go
```

2. Pruebe la restricción insertando un curso en el que no se establece de modo explícito el número de vacantes:

```
INSERT INTO Curso(idCurso, nombreCurso,
    numMatriculados, precioCurso, profesor)
    VALUES('VB1A', 'Visual Basic .NET Nivel I - Grupo B',
    0, 250, 'Marcelo Villalobos, Ricardo')
go

SELECT * FROM Curso
go
```

Creación de regla de validación (CHECK)

Sintaxis

```
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 28: Creación de restricción CHECK para la columna precioCurso de la tabla Curso

El precio de un curso no puede ser 0 (cero) ni menor que 0.

1. En su query escriba la siguiente instrucción para definir la regla de validación para la columna **precioCurso** de la tabla **Curso**:

```
ALTER TABLE Curso
    ADD CONSTRAINT CK_Curso_precioCurso
    CHECK( precioCurso > 0 )
go
```

2. Ahora, pruebe la regla ejecutando la siguiente inserción:

```
INSERT INTO Curso(idCurso, nombreCurso, numMatriculados,
    precioCurso, profesor)
VALUES('JV1A', 'Java 2 Nivel I - Grupo A', 0,
    0, 'Valencia Morales, Pedro Hugo')
go
```

Se recibe el siguiente mensaje de error que indica que se ha intentado violar una regla de validación:

```
Msg 547, Level 16, State 0, Line 1
INSERT statement conflicted with CHECK constraint
'CK_Curso_precioCurso'. The conflict occurred in database
'ExtPro', table 'Curso', column 'precioCurso'.
The statement has been terminated.
```

Ejercicio 29: Prueba de la integridad referencial en la base de datos ExtPro

```
USE ExtPro
go

SELECT * FROM Alumno
go

-- Registrando a un alumno
INSERT INTO Alumno(idAlumno, nombreAlumno,
    direccion, telefono)
VALUES(1001, 'Castro Sinchi, Oscar',
    'Av. Arica 1301 - Breña', '432-1254')
go

SELECT * FROM Alumno
go

SELECT * FROM Curso
go

-- Registrando una matrícula
INSERT INTO Matricula(idCurso, idAlumno, fecha,
    precio, numCuotas)
VALUES('PB2A', 1001, '12 Feb 2005', 250, 3)
go
-- Violación de la clave foránea FK_Matricula_Curso
-- El curso 'PB2A' no está registrado.
```

Se recibe el siguiente mensaje de error:

```
Msg 547, Level 16, State 0, Line 1
INSERT statement conflicted with FOREIGN KEY constraint
'fk_Matricula_Curso'. The conflict occurred in database
'ExtPro', table 'Curso', column 'idCurso'.
The statement has been terminated.
```

```
-- Registrando otra matrícula
INSERT INTO Matricula(idCurso, idAlumno, fecha,
    precio, numCuotas)
VALUES('PB1A', 1015, '12 Feb 2005', 250, 3)
go
-- Violación de la clave foránea FK_Matricula_Alumno
-- El alumno 1015 no está registrado.
```

Se recibe el mensaje de error siguiente:

```
Msg 547, Level 16, State 0, Line 1
INSERT statement conflicted with FOREIGN KEY constraint
'fk_Matricula_Alumno'. The conflict occurred in database
'ExtPro', table 'Alumno', column 'idAlumno'.
The statement has been terminated.
```

```
-- Registrando una matrícula con datos correctos
INSERT INTO Matricula(idCurso, idAlumno, fecha,
    precio, numCuotas)
VALUES('PB1A', 1001, '12 Feb 2005', 250, 3)
go

SELECT * FROM Matricula
go
```

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 un alumno que tiene matrículas registradas.

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 30: Definición de integridad referencial en cascada para las tablas Matricula y Pago

Para la base de datos **ExtPro** defina que al eliminar una matrícula en la tabla **Matricula** se eliminen también todos los pagos registrados en la tabla **Pago** asociados a dicha matrícula.

1. Primero, registre la matrícula de un alumno nuevo:

```
USE ExtPro
go

-- Matriculando a un alumno nuevo
INSERT INTO Alumno(idAlumno, nombreAlumno,
    direccion, telefono)
VALUES(1002, 'Talavera Osorio, Ana',
    'Jr. Junín 2312 - Pueblo Libre', '2221564')
go

INSERT INTO Matricula(idCurso, idAlumno, fecha,
    precio, numCuotas)
VALUES('VB1A', 1002, '16 Mar 2005', 250, 3)
go
```

2. Ahora, registre los pagos de dicho alumno:

```
-- Registrando sus pagos
INSERT INTO Pago(idCurso, idAlumno, cuotaNumero,
    fecha, importe)
VALUES('VB1A', 1002, 1, '16 Mar 2005', 100)

INSERT INTO Pago(idCurso, idAlumno, cuotaNumero,
    fecha, importe)
VALUES('VB1A', 1002, 2, '30 Mar 2005', 75)

INSERT INTO Pago(idCurso, idAlumno, cuotaNumero,
    fecha, importe)
VALUES('VB1A', 1002, 3, '15 Apr 2005', 75)
go
```

```
SELECT * FROM Matricula
go
```

```
SELECT * FROM Pago
go
```

SQLQuery1.sql-...EV2005.ExtPro* Summary						
	idCurso	idAlumno	fecha	precio	numCuotas	notaCurso
1	PB1A	1001	2005-02-12 00:00:00	250.00	3	NULL
2	VB1A	1002	2005-03-16 00:00:00	250.00	3	NULL

	idCurso	idAlumno	cuotaNumero	fecha	importe
1	VB1A	1002	1	2005-03-16 00:00:00	100.00
2	VB1A	1002	2	2005-03-30 00:00:00	75.00
3	VB1A	1002	3	2005-04-15 00:00:00	75.00

3. Ahora, redefina la clave foránea de la tabla **Pago**:

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

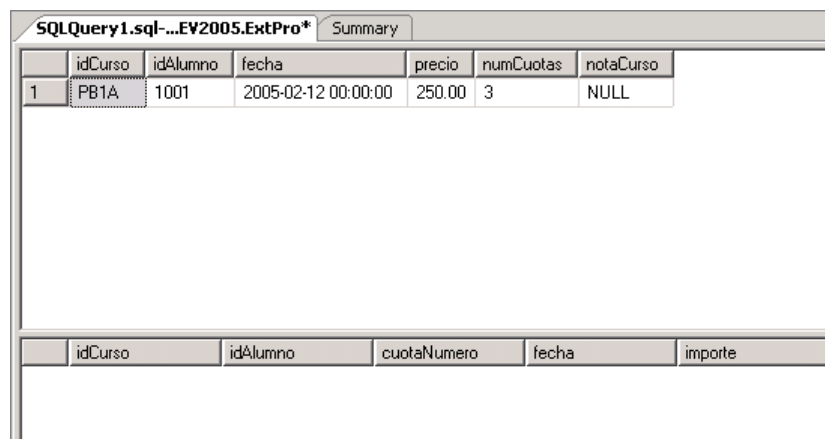
ALTER TABLE Pago
    ADD CONSTRAINT FK_Pago_Matricula
    FOREIGN KEY( idCurso, idAlumno )
    REFERENCES Matricula
    ON DELETE CASCADE
go
```

4. A continuación, elimine la matrícula del alumno **1002**:

```
-- Eliminando la matrícula del alumno 1002
-- para el curso 'VB1A'
DELETE FROM Matricula
    WHERE idAlumno = 1002 AND idCurso = 'VB1A'
go

SELECT * FROM Matricula
go

SELECT * FROM Pago
go
```



SQLQuery1.sql-...EV2005.ExtPro*						
Summary						
	idCurso	idAlumno	fecha	precio	numCuotas	notaCurso
1	PB1A	1001	2005-02-12 00:00:00	250.00	3	NULL

idCurso	idAlumno	cuotaNumero	fecha	importe
---------	----------	-------------	-------	---------

La eliminación de la matrícula del alumno **1002** en el curso **VB1A** condujo a la eliminación automática de todos los pagos registrados por dicha matrícula.

Eliminación de tablas (la instrucción DROP TABLE)

Sintaxis

```
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 31: Eliminación de tablas

1. Ejecute la siguiente instrucción en su query:

```
DROP TABLE Matricula  
go
```

2. Se recibe el siguiente mensaje de error:

```
Msg 3726, Level 16, State 1, Line 2  
Could not drop object 'Matricula' because it is referenced  
by a FOREIGN KEY constraint.
```

La tabla **Matricula** no se puede eliminar porque es referenciada por una clave foránea.

3. Ahora, ejecute la instrucción siguiente:

```
DROP TABLE TablaPrueba  
go
```

La tabla se elimina sin problemas.

Esta página se ha dejado en blanco intencionalmente.