Implementación de la integridad de datos

Contenido

Notas para el instructor

Este módulo proporciona a los alumnos una introducción a los conceptos relativos a la integridad de los datos, incluidos los métodos disponibles para exigir dicha integridad. El módulo presenta a continuación las restricciones, que constituyen el método principal para asegurar la integridad de los datos. También se ilustran varios tipos de restricciones. La creación y la implementación de las restricciones se describe en detalle, así como los medios para deshabilitar las restricciones cuando sea necesario.

Los valores predeterminados y las reglas se describen como métodos alternativos para implementar la integridad de los datos, aunque el énfasis se mantiene en las restricciones. El módulo concluye con la comparación de los diferentes métodos para conseguir la integridad de los datos.

En la práctica, los alumnos definen restricciones DEFAULT, CHECK, PRIMARY KEY y FOREIGN KEY.

Después de completar este módulo, los alumnos serán capaces de:

- Describir los tipos de integridad de datos.
- Describir los métodos para implementar la integridad de datos.
- Determinar qué restricción utilizar y crear restricciones.
- Definir y utilizar las restricciones DEFAULT, CHECK, PRIMARY KEY, UNIQUE y FOREIGN KEY.
- Deshabilitar restricciones.
- Describir y utilizar valores predeterminados y reglas.
- Determinar los métodos que se van a utilizar para implementar la integridad de los datos.

Introducción

Objetivo del tema
Proporcionar una introducción a los temas y objetivos del módulo.

Explicación previa
En este módulo aprenderá acerca de los distintos tipos de integridad de datos y las características que los posibilitan.

- Tipos de integridad de datos
- Exigir la integridad de los datos
- Definición de restricciones
- Tipos de restricciones
- Deshabilitación de restricciones
- Uso de valores predeterminados y reglas
- Decisión del método de implementación que va a utilizar

Este módulo comienza con una introducción a los conceptos relativos a la integridad de los datos, incluidos los métodos disponibles para exigir dicha integridad. El módulo presenta a continuación las restricciones, que constituyen el método clave para asegurar la integridad de los datos, y los diversos tipos de restricciones. La creación y la implementación de las restricciones se describe en detalle, así como los medios para deshabilitar las restricciones cuando sea necesario.

Los valores predeterminados y las reglas se describen como métodos alternativos para implementar la integridad de los datos, aunque el énfasis se mantiene en las restricciones. El módulo concluye con la comparación de los diferentes métodos para conseguir la integridad de los datos.

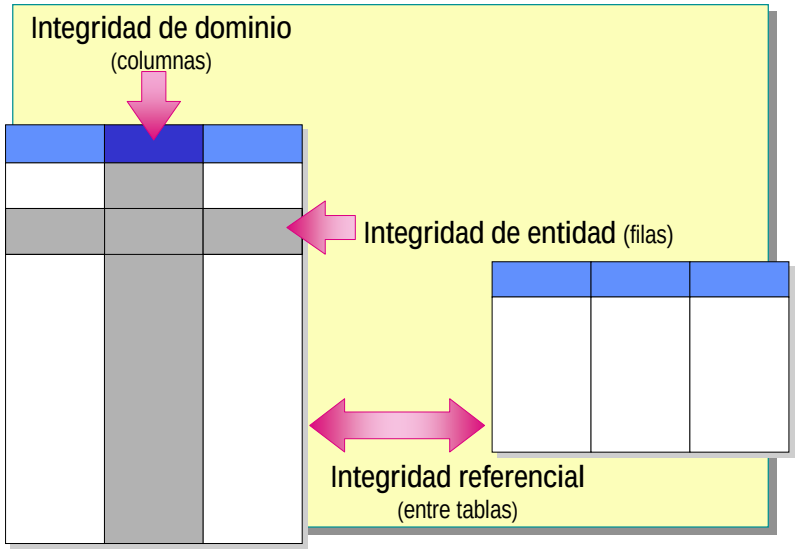
Después de realizar esta práctica, el alumno será capaz de:

- Describir los tipos de integridad de datos.
- Describir los métodos para implementar la integridad de datos.
- Determinar qué restricción utilizar y crear restricciones.
- Definir y utilizar las restricciones DEFAULT, CHECK, PRIMARY KEY, UNIQUE y FOREIGN KEY.
- Deshabilitar la comprobación de restricciones.
- Describir y utilizar valores predeterminados y reglas.
- Determinar los métodos que se van a utilizar para implementar la integridad de los datos.

Tipos de integridad de datos

Objetivo del tema
Presentar los distintos tipos de integridad de datos.

Explicación previa
Un paso importante en el diseño de una base de datos es decidir la mejor forma de implementar la integridad de los datos. La integridad de los datos se clasifica en tres categorías.



Un paso importante en el diseño de una base de datos es decidir la mejor forma de implementar la integridad de los datos. La integridad de los datos hace referencia a la coherencia y la precisión de los datos que están almacenados en una base de datos. Los diferentes tipos de integridad de datos son los siguientes.

Sugerencia
Los tipos de integridad de datos a los que se hace referencia aquí representan el diseño básico de las bases de datos relacionales. Explique los conceptos de forma general. Puede aportar un ejemplo con la base de datos Northwind.

Integridad de dominio

La integridad de *dominio* (o columna) especifica un conjunto de valores de datos que son válidos para una columna y determina si se permiten valores nulos. La integridad de dominio se suele implementar mediante el uso de comprobaciones de validez y, también, mediante la restricción del tipo de datos, el formato o el intervalo de los valores posibles permitidos en una columna.

Integridad de entidad

La integridad de *entidad* (o tabla) requiere que todas las filas de una tabla tengan un identificador exclusivo, conocido como *clave principal*. El que se pueda modificar el valor de la clave principal o eliminar la fila entera depende del nivel de integridad requerido entre la clave principal y cualquier otra tabla.

Integridad referencial

La integridad *referencial* asegura que siempre se mantienen las relaciones entre las claves principales (en la tabla a la que se hace referencia) y las *claves externas* (en las tablas que hacen referencia). No se puede eliminar una fila de una tabla a la que se hace referencia, ni se puede modificar la clave principal, si una clave externa hace referencia a la fila, salvo que se permita la acción en cascada. Puede definir relaciones de integridad referencial dentro de la misma tabla o entre tablas diferentes.

Exigir integridad de los datos

Objetivo del tema
Presentar la forma en que SQL Server implementa la integridad de los datos.

Explicación previa
Puede conseguir la integridad de los datos mediante dos métodos.

- Integridad de datos declarativa
  - Los criterios se definen en la definición del objeto
  - Asegurada automáticamente por SQL Server
  - Implementada mediante restricciones, valores predeterminados y reglas
- Integridad de datos procedimental
  - Los criterios se definen en una secuencia de comandos
  - Asegurada mediante secuencia de comandos
  - Implementada mediante desencadenadores y prodedimientos almacenados

Puede conseguir la integridad de los datos mediante dos métodos: integridad de datos declarativa o integridad de datos procedimental.

Integridad de datos declarativa

Con la integridad *declarativa*, se definen los criterios que los datos tienen que cumplir como parte de la definición de un objeto y, después, Microsoft® SQL Server™ versión 2000 asegura automáticamente que los datos cumplan dichos criterios. El método preferido para implementar la integridad de datos básica es la integridad declarativa. Tenga en cuenta los hechos siguientes acerca del método declarativo:

- La integridad declarativa se declara como parte de la definición de la base de datos, mediante el uso de restricciones declarativas que se definen directamente en las tablas y las columnas.
- Implemente la integridad declarativa mediante la utilización de restricciones, valores predeterminados y reglas.

Sugerencia
Mencione que en este módulo sólo se tratan las características de la integridad declarativa: restricciones, valores predeterminados y reglas. Los desencadenadores y los procedimientos almacenados se tratan en módulos siguientes.

Integridad de datos procedimental

Con la integridad *procedimental* se escriben secuencias de comandos que definen los criterios que los datos tienen que cumplir y que aseguran que dichos criterios se cumplen. Debe limitar el uso de la integridad procedimental a situaciones excepcionales y a aquellas con una lógica complicada. Por ejemplo, utilice la integridad procedimental cuando desee implementar una eliminación en cascada. Los hechos siguientes se aplican a la integridad procedimental:

- La integridad procedimental se puede implementar en el cliente o en el servidor mediante otros lenguajes y herramientas de programación.
- Implemente la integridad procedimental utilizando desencadenadores y procedimientos almacenados.

Definición de restricciones

Objetivo del tema
Presentar la implementación de la integridad de datos con restricciones.

Explicación previa
Las restricciones son el método más adecuado para implementar la integridad de los datos.

- Determinación del tipo de restricción que se va a utilizar
- Creación de restricciones
- Consideraciones para el uso de restricciones

Las restricciones son el método más adecuado para conseguir la integridad de los datos. En esta sección se describe cómo determinar el tipo de restricción que se tiene que utilizar, qué tipo de integridad de datos implementa cada tipo de restricción y cómo definir las restricciones.

Determinación del tipo de restricción que se va a utilizar

Objetivo del tema
Presentar los distintos tipos de restricciones y cómo utilizarlas para implementar la integridad de datos.

Explicación previa
Diferentes tipos de restricciones aseguran que los valores que se escriban en los datos de las columnas son válidos y que se mantienen las relaciones entre las tablas.

Tipo de integridad	Tipo de restricción
Dominio	DEFAULT
	CHECK
	REFERENTIAL
Entidad	PRIMARY KEY
	UNIQUE
Referencial	FOREIGN KEY
	CHECK

Punto clave
Haga énfasis en que las restricciones cumplen el estándar ANSI.

Las restricciones son un método estándar ANSI para implementar la integridad de los datos. Cada tipo de integridad de datos (dominio, entidad y referencial) se implementa con tipos de restricciones diferentes. Las restricciones aseguran que los datos que se escriben en las columnas sean válidos y que se mantengan las relaciones entre las tablas. La tabla siguiente describe los diferentes tipos de restricciones.

Tipo de integridad	Tipo de restricción	Descripción
Dominio	DEFAULT	Especifica el valor que se proporciona para la columna cuando no se especifica explícitamente en una instrucción INSERT.
	CHECK	Especifica los valores de los datos que se aceptan en una columna.
	REFERENTIAL	Especifica los valores de datos que se aceptan como actualización en función de los valores de una columna de otra tabla.
Entidad	PRIMARY KEY	Identifica de forma exclusiva cada una de las filas; asegura que los usuarios no escriban valores duplicados y que se cree un índice para aumentar el rendimiento. No se permiten valores nulos.
	UNIQUE	Impide la duplicación de claves alternativas (no principales) y asegura que se cree un índice para aumentar el rendimiento. Se permiten valores nulos.
Referencial	FOREIGN KEY	Define una columna o combinación de columnas cuyos valores coinciden con la clave principal de la misma u otra tabla.
	CHECK	Especifica los valores de los datos que se aceptan en una columna en función de los valores de otras columnas de la misma tabla.

Creación de restricciones

Objetivo del tema
Presentar la sintaxis para definir restricciones.

Explicación previa
Las restricciones se implementan mediante la instrucción CREATE TABLE o ALTER TABLE.

- Utilizar CREATE TABLE o ALTER TABLE
- Puede agregar restricciones a una tabla con datos existentes
- Puede aplicar restricciones a una sola columna o a varias columnas
  - Una sola columna, se llama restricción de columna
  - Varias columnas, se llama restricción de tabla

Las restricciones se crean mediante la instrucción CREATE TABLE o ALTER TABLE.

Sugerencia
Destaque que el término "restricción de tabla" hace referencia a cualquier restricción de varias columnas.

Puede agregar restricciones a una tabla con datos existentes y puede aplicar restricciones a una sola columna o a varias columnas:

- Si la restricción se aplica a una sola columna, se llama restricción de *columna*.
- Si la restricción hace referencia a varias columnas, se llama restricción de *tabla*, incluso si no hace referencia a todas las columnas de la tabla.

Sintaxis parcial

```
CREATE TABLE Tabla
    ( { < definiciónColumna >
     | < restricciónTabla > } [ ,...n ])

< definiciónColumna > ::= { columnatipoDeDatos }
    [ [ DEFAULT expresiónConstante ]
    [ < restricciónColumna > ] [ ,..n]

< restricciónColumna > ::=
  [ CONSTRAINT nombreRestricción ]
        | [ { PRIMARY KEY | UNIQUE }
            [ CLUSTERED | NONCLUSTERED ] ]

    | [ [ FOREIGN KEY ]
            REFERENCES tablaRef [ ( columnaRef ) ]
            [ ON DELETE { CASCADE | NO ACTION } ]
            [ ON UPDATE { CASCADE | NO ACTION } ]]
        | CHECK ( expresiónLógica ) }
```

Sugerencias
Resalte que la sintaxis se divide en restricciones de columna y de tabla.

Aconseje a los alumnos que creen primero la tabla base y después agreguen las restricciones, lo que simplifica el proceso de definición de las tablas.

```
< restricciónTabla > ::=
  [ CONSTRAINT nombreRestricción ]
   { [ { PRIMARY KEY | UNIQUE }
       [CLUSTERED | NONCLUSTERED]
       { ( columna [ ASC | DESC ] [ ,...n ] ) } ]
   | FOREIGN KEY
       [ ( columna [ ,...n ] ) ]
       REFERENCES tablaRef [ ( columnaRef [ ,...n ] ) ]
       [ ON DELETE { CASCADE | NO ACTION } ]
       [ ON UPDATE { CASCADE | NO ACTION } ]
   | CHECK ( condicionesBúsqueda ) }
```

Ejemplo

Este ejemplo crea la tabla **Products**, define columnas y define restricciones de columna y de tabla.

```
USE northwind
CREATE TABLE dbo.Products
(
   ProductID      int IDENTITY (1,1) NOT NULL,
   ProductName    nvarchar (40) NOT NULL,
   SupplierID     int      NULL,
   CategoryID     int      NULL,
   QuantityPerUnit nvarchar (20) NULL,
   UnitPrice      money    NULL     CONSTRAINT DF_Products_UnitPrice  DEFAULT(0),
   UnitsInStock   smallint NULL     CONSTRAINT DF_Products_UnitsInStock DEFAULT(0),
   UnitsOnOrder   smallint NULL     CONSTRAINT DF_Products_UnitsOnOrder DEFAULT(0),
   ReorderLevel   smallint NULL     CONSTRAINT DF_Products_ReorderLevel DEFAULT(0),
   Discontinued   bit      NOT NULL CONSTRAINT DF_Products_Discontinued DEFAULT(0),

   CONSTRAINT PK_Products PRIMARY KEY CLUSTERED (ProductID),

   CONSTRAINT FK_Products_Categories FOREIGN KEY (CategoryID)
      REFERENCES dbo.Categories (CategoryID) ON UPDATE CASCADE,
   CONSTRAINT FK_Products_Suppliers  FOREIGN KEY (SupplierID)
      REFERENCES dbo.Suppliers  (SupplierID) ON DELETE CASCADE,

   CONSTRAINT CK_Products_UnitPrice CHECK (UnitPrice >= 0),
   CONSTRAINT CK_ReorderLevel       CHECK (ReorderLevel >= 0),
   CONSTRAINT CK_UnitsInStock       CHECK (UnitsInStock >= 0),
   CONSTRAINT CK_UnitsOnOrder       CHECK (UnitsOnOrder >= 0)
)
GO
```

Consideraciones para el uso de restricciones

Objetivo del tema
Describir algunas de las consideraciones para utilizar restricciones.

Explicación previa
Considere estos hechos cuando implemente o modifique restricciones.

- Pueden cambiarse sin volver a crear una tabla
- Requieren comprobación de errores en aplicaciones y transacciones
- Comprueban los datos existentes

Sugerencia
Demuestre que SQL Server crea nombres de restricción complicados, generados por el sistema.

Considere estos hechos cuando implemente o modifique restricciones:

- Puede crear, modificar y eliminar restricciones sin tener que eliminar y volver a crear una tabla.
- Tiene que generar lógica de control de errores en sus aplicaciones y transacciones para probar si se ha infringido una restricción.
- SQL Server comprueba los datos existentes cuando se agrega una restricción a una tabla.

Tiene que especificar los nombres de las restricciones cuando las cree, puesto que SQL Server proporciona nombres complicados, generados por el sistema. Los nombres tienen que ser exclusivos para el propietario del objeto de la base de datos y seguir las reglas de los identificadores de SQL Server.

Para obtener ayuda acerca de las restricciones, ejecute el procedimiento almacenado del sistema **sp_helpconstraint** o **sp_help**, o consulte las vistas del esquema de información, como **check_constraints**, **referential_constraints** y **table_constraints**.

Las tablas del sistema siguientes almacenan las definiciones de las restricciones: **syscomments**, **sysreferences** y **sysconstraints**.

◆ Tipos de restricciones

Objetivo del tema
Describir los tipos de restricciones.

Explicación previa
Esta sección describe los tipos de restricciones.

- Restricciones DEFAULT
- Restricciones CHECK
- Restricciones PRIMARY KEY
- Restricciones UNIQUE
- Restricciones FOREIGN KEY
- Integridad referencial en cascada

Esta sección describe los tipos de restricciones. Cada restricción viene definida por su sintaxis, ejemplos y consideraciones de uso.

Restricciones DEFAULT

Objetivo del tema
Presentar la restricción DEFAULT.

Explicación previa
La restricción DEFAULT implementa la integridad de dominio.

- Sólo se aplica a las instrucciones INSERT
- Sólo una restricción DEFAULT por columna
- No se puede utilizar con la propiedad IDENTITY o el tipo de datos rowversion
- Permite que se especifiquen algunos valores proporcionados por el sistema

```
USE Northwind
ALTER TABLE dbo.Customers
ADD
CONSTRAINT DF_contactname DEFAULT 'UNKNOWN'
FOR ContactName
```

La restricción DEFAULT escribe un valor en una columna cuando no se especifica en las instrucciones INSERT. Las restricciones DEFAULT implementan la integridad de dominio.

Sintaxis parcial

[CONSTRAINT *nombreRestricción*]
DEFAULT *expresiónConstante*

Ejemplo

En este ejemplo se agrega una restricción DEFAULT que inserta el valor UNKNOWN en la tabla **dbo.Customers** si no se proporciona el nombre de contacto.

```
USE Northwind
ALTER TABLE dbo.Customers
ADD
CONSTRAINT DF_contactname DEFAULT 'UNKNOWN' FOR ContactName
```

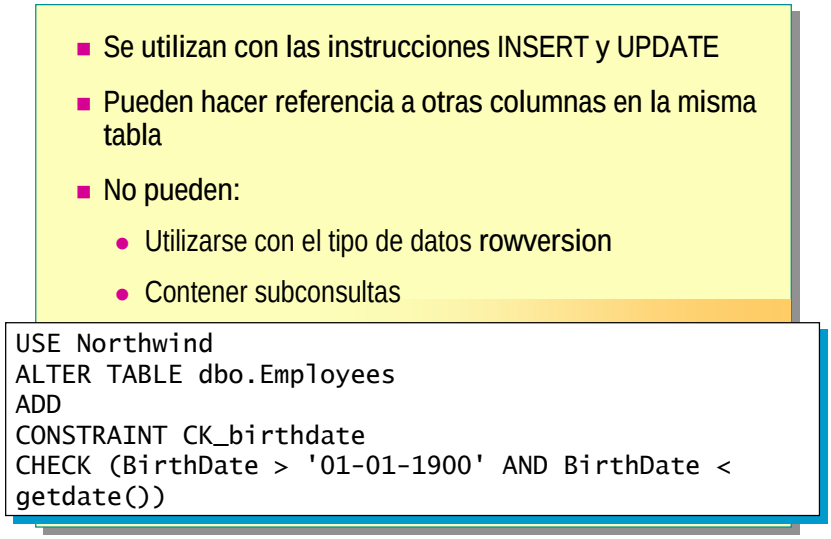
Considere los hechos siguientes cuando aplique una restricción DEFAULT:

- La restricción comprueba los datos existentes en la tabla.
- Sólo se aplica a las instrucciones INSERT.
- Sólo se puede definir una restricción DEFAULT por cada columna.
- No se puede aplicar a columnas con la propiedad **Identity** o a columnas con el tipo de datos **rowversion**.
- Permite que se especifiquen algunos valores proporcionados por el sistema (USER, CURRENT_USER, SESSION_USER, SYSTEM_USER o CURRENT_TIMESTAMP) en lugar de valores definidos por el usuario. Dichos valores proporcionados por el sistema pueden ser útiles para obtener un registro de los usuarios que insertan los datos.

Restricciones CHECK

Objetivo del tema
Presentar la restricción CHECK.

Explicación previa
Una restricción CHECK restringe los datos escritos en una columna a valores específicos.



La restricción CHECK restringe los datos que los usuarios pueden escribir en una columna particular a unos valores específicos. Las restricciones CHECK son similares a las cláusulas WHERE donde se pueden especificar las condiciones bajo las que se aceptan los datos.

Sintaxis parcial

[CONSTRAINT *nombreRestricción*]
CHECK **(***expresiónLógica***)**

Ejemplo

Este ejemplo agrega una restricción CHECK para garantizar que una fecha de nacimiento cumpla un intervalo aceptable de fechas.

```
USE Northwind
ALTER TABLE dbo.Employees
ADD
CONSTRAINT CK_birthdate
CHECK (BirthDate > '01-01-1900' AND BirthDate < getdate())
```

Considere los hechos siguientes cuando aplique una restricción CHECK:

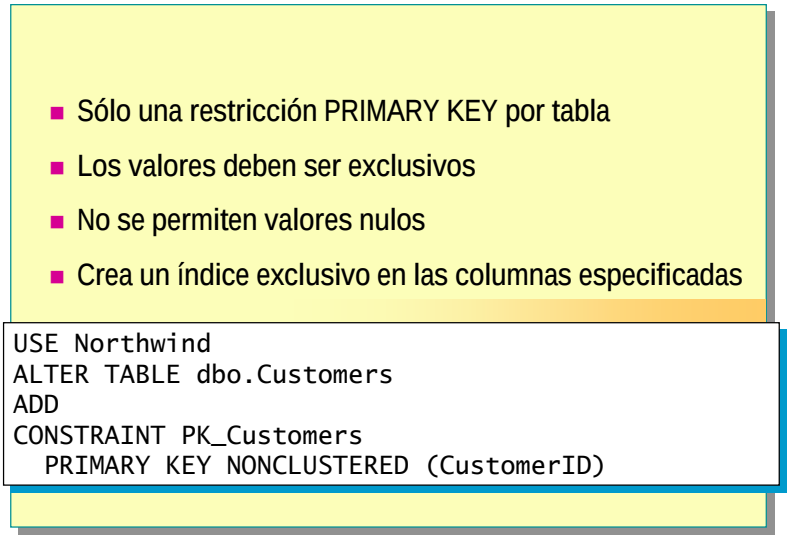
- La restricción comprueba los datos cada vez que se ejecuta una instrucción INSERT o UPDATE.
- Puede hacer referencia a otras columnas de la misma tabla.

  Por ejemplo, una columna **salary** podría hacer referencia a un valor de una columna **job_grade**.
- No se puede aplicar a columnas con el tipo de datos **rowversion**.
- No puede contener subconsultas.
- Si alguno de los datos infringe la restricción CHECK, puede ejecutar la instrucción DBCC CHECKCONSTRAINTS para ver las filas infractoras.

Restricciones PRIMARY KEY

Objetivo del tema
Presentar las restricciones PRIMARY KEY.

Explicación previa
Las restricciones PRIMARY KEY implementan la integridad de entidad.



La restricción PRIMARY KEY define una clave principal en una tabla para identificar de forma exclusiva cada una de sus filas. Implementa la integridad de entidad.

Sintaxis parcial

[CONSTRAINT *nombreRestricción*]
PRIMARY KEY [CLUSTERED | NONCLUSTERED]
{ (*columna*[,...*n*]) }

Ejemplo

En este ejemplo se agrega una restricción que especifica que la clave principal de la tabla **dbo.Customers** es la identificación del cliente e indica que se va a crear un índice no agrupado para implementar la restricción.

```
USE northwind
ALTER TABLE dbo.Customers
ADD
CONSTRAINT PK_Customers
   PRIMARY KEY NONCLUSTERED (CustomerID)
```

Puntos clave
La restricción PRIMARY KEY siempre es exclusiva y no permite valores nulos.

La restricción PRIMARY KEY siempre crea un índice.

Considere los hechos siguientes cuando aplique una restricción PRIMARY KEY:

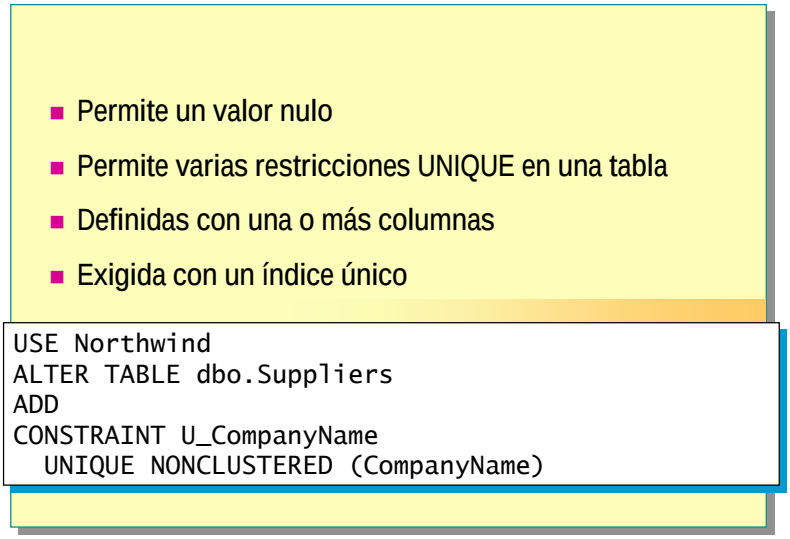
- Sólo se puede definir una restricción PRIMARY KEY por tabla.
- Los valores escritos tienen que ser exclusivos.
- No se permiten valores nulos.
- Crea un índice exclusivo en las columnas especificadas. Puede especificar un índice agrupado o un índice no agrupado (el agrupado es el tipo predeterminado si no existe anteriormente).

Nota El índice creado para la restricción PRIMARY KEY no se puede eliminar directamente. Se elimina cuando se quita la restricción.

Restricciones UNIQUE

Objetivo del tema
Presentar las restricciones UNIQUE.

Explicación previa
La restricción UNIQUE especifica que dos filas de una columna no pueden tener el mismo valor.



La restricción UNIQUE especifica que dos filas de una columna no pueden tener el mismo valor. Esta restricción implementa la integridad de entidad con un índice único.

La restricción UNIQUE es útil cuando ya se tiene una clave principal, como un número de empleado, pero se desea garantizar que otros identificadores, como el número del permiso de conducir de un empleado, también sean exclusivos.

Sintaxis parcial

```
[CONSTRAINT nombreRestricción]
    UNIQUE [CLUSTERED | NONCLUSTERED]
        { ( columna[,...n] ) }
```

Ejemplo

Este ejemplo crea una restricción UNIQUE sobre la columna **company name** de la tabla **dbo.Suppliers**.

```
USE northwind
ALTER TABLE dbo.Suppliers
ADD
CONSTRAINT U_CompanyName
   UNIQUE NONCLUSTERED (CompanyName)
```

Considere los hechos siguientes cuando aplique una restricción UNIQUE:

- Puede permitir un valor nulo.
- Puede aplicar varias restricciones UNIQUE en una misma tabla.
- Puede aplicar la restricción UNIQUE a una o varias columnas que tengan que tener valores exclusivos y no sean la clave principal de una tabla.
- La restricción UNIQUE se implementa mediante la creación de un índice exclusivo en la columna o columnas especificadas.

Restricciones FOREIGN KEY

Objetivo del tema
Presentar la restricción FOREIGN KEY.

Explicación previa
La restricción FOREIGN KEY implementa la integridad referencial.

- Deben hacer referencia a una restricción PRIMARY KEY o UNIQUE
- Proporcionan integridad referencial de una o de varias columnas
- No crean índices automáticamente
- Los usuarios deben tener permisos SELECT o REFERENCES en las tablas a las que se hace referencia
- Usa sólo la cláusula REFERENCES en la tabla de ejemplo

```
USE Northwind
ALTER TABLE dbo.Orders
ADD CONSTRAINT FK_Orders_Customers
  FOREIGN KEY (CustomerID)
  REFERENCES dbo.Customers(CustomerID)
```

La restricción FOREIGN KEY implementa la integridad referencial. La restricción FOREIGN KEY define una referencia a una columna con una restricción PRIMARY KEY o UNIQUE en la misma o en otra tabla.

Sintaxis parcial

[CONSTRAINT *nombreRestricción*]
[FOREIGN KEY] [**(***columna*[,…*n*]**)**]
REFERENCES *tablaRef* [**(***columnaRef* [,…*n*]**)**].

Ejemplo

Este ejemplo utiliza una restricción FOREIGN KEY para garantizar que la identificación del cliente de la tabla **dbo.Orders** esté asociada con una identificación válida en la tabla **dbo.Customers**.

```
USE northwind
ALTER TABLE dbo.Orders
ADD CONSTRAINT FK_Orders_Customers
   FOREIGN KEY (CustomerID)
   REFERENCES dbo.Customers(CustomerID)
```

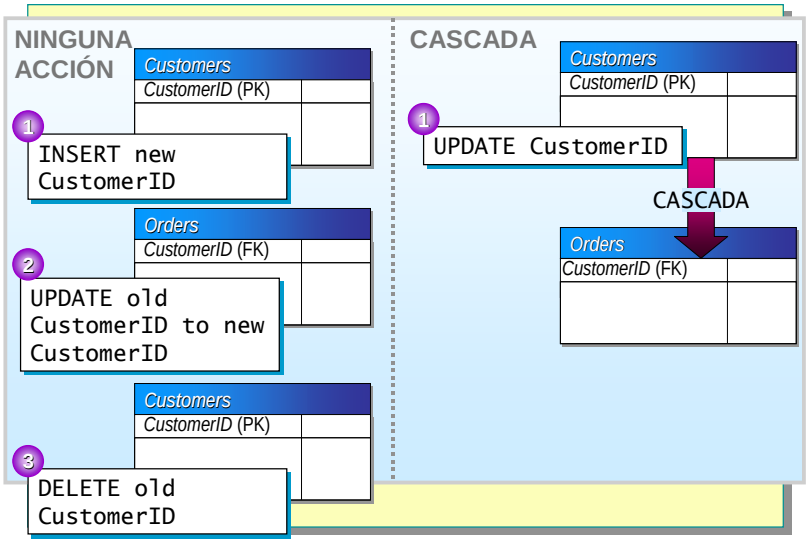
Considere los hechos y recomendaciones siguientes cuando aplique una restricción FOREIGN KEY:

- Proporciona integridad referencial de una o de varias columnas. El número de columnas y los tipos de datos que se especifican en la instrucción FOREIGN KEY tienen que coincidir con el número de columnas y los tipos de datos de la cláusula REFERENCES.
- Al contrario que las restricciones PRIMARY KEY o UNIQUE, las restricciones FOREIGN KEY no crean índices automáticamente. Sin embargo, si la base de datos utiliza muchas combinaciones, tiene que crear un índice para que FOREIGN KEY aumente el rendimiento en las combinaciones.
- Para modificar los datos, los usuarios deben tener permisos SELECT o REFERENCES en las tablas a las que se hace referencia en la restricción FOREIGN KEY.
- Sólo se puede utilizar la cláusula REFERENCES sin la cláusula FOREIGN KEY cuando se hace referencia a una columna de la misma tabla.

Integridad referencial en cascada

Objetivo del tema
Describir la integridad referencial en cascada.

Explicación previa
La integridad referencial en cascada propaga automáticamente los cambios realizados a la base de datos.



La restricción FOREIGN KEY incluye la opción CASCADE que permite que cualquier cambio en un valor de columna que define una restricción UNIQUE o PRIMARY KEY se propague automáticamente al valor de clave externa. Esta acción se conoce como *integridad referencial en cascada*.

Las cláusulas REFERENCES de las instrucciones CREATE TABLE y ALTER TABLE aceptan cláusulas ON DELETE y ON UPDATE. Estas cláusulas le permiten especificar la opción CASCADE o NO ACTION.

Sintaxis parcial

```
[CONSTRAINT nombreRestricción]
    [FOREIGN KEY] [(columna[,…n])]
        REFERENCES tablaRef [(columnaRef [,…n])].
        [ ON DELETE { CASCADE | NO ACTION } ]
        [ ON UPDATE { CASCADE | NO ACTION } ]
```

NO ACTION especifica que cualquier intento de eliminar o actualizar una clave a la que hagan referencia claves externas en otras tablas genere un error y el cambio se deshaga. NO ACTION es la opción predeterminada.

Si se selecciona CASCADE y se cambia una fila en la tabla primaria, la fila correspondiente se cambia entonces en la tabla que hace referencia.

Por ejemplo, en la base de datos **Northwind**, la tabla **Orders** tiene una relación referencial con la tabla **Customers**; específicamente, la clave externa **Orders.CustomerID** hace referencia a la clave principal **Customers.CustomerID**.

Si se ejecuta una instrucción UPDATE sobre la columna **CustomerID** de la tabla **Customers** y se especifica una acción ON UPDATE CASCADE para **Orders.CustomerID**, SQL Server busca una o varias filas dependientes en la tabla **Orders**. Si encuentra alguna, actualiza las filas dependientes de la tabla **Orders**, así como la fila a la que se hace referencia en la tabla **Customers**.

Considere estos factores al aplicar la opción CASCADE:

- Es posible combinar CASCADE y NO ACTION en tablas que mantengan relaciones referenciales. Si SQL Server encuentra NO ACTION, concluye y deshace las acciones CASCADE relacionadas.

  Cuando una instrucción DELETE provoca la combinación de las acciones CASCADE y NO ACTION, todas las acciones CASCADE se aplican antes de que SQL Server busque cualquier acción NO ACTION.
- CASCADE no se puede especificar para ninguna columna de clave externa o principal que se haya definido con una columna **rowversion**.

Deshabilitación de restricciones

Objetivo del tema
Describir los métodos de deshabilitación de restricciones.

Explicación previa
Por motivos de rendimiento, algunas veces resulta aconsejable deshabilitar restricciones.

- Deshabilitación de la comprobación de las restricciones en los datos existentes
- Deshabilitación de la comprobación de las restricciones al cargar datos nuevos

Por motivos de rendimiento, algunas veces resulta aconsejable deshabilitar restricciones. Por ejemplo, es más conveniente permitir que se procesen grandes operaciones por lotes antes que habilitar restricciones. Esta sección describe el modo de deshabilitar la comprobación de restricciones, tanto si va a crear una nueva restricción o deshabilitar una existente.

Deshabilitación de la comprobación de las restricciones en los datos existentes

Objetivo del tema
Presentar cómo se deshabilitan las restricciones.

Explicación previa
Puede deshabilitar la comprobación de restricciones al agregar una restricción a una tabla.

- Se aplica a las restricciones CHECK y FOREIGN KEY
- Utilice la opción WITH NOCHECK cuando agregue una restricción nueva
- Utilizar si los datos existentes no cambian
- Se pueden cambiar los datos existentes antes de agregar restricciones

```
USE Northwind
ALTER TABLE dbo.Employees
WITH NOCHECK
  ADD CONSTRAINT FK_Employees_Employees
  FOREIGN KEY (ReportsTo)
  REFERENCES dbo.Employees(EmployeeID)
```

Cuando se define una restricción en una tabla que ya contiene datos, SQL Server los comprueba automáticamente para confirmar que cumplen los requisitos de la restricción. Sin embargo, puede deshabilitar la comprobación de restricciones en los datos existentes al agregar una restricción a una tabla.

Considere las recomendaciones siguientes para deshabilitar la comprobación de restricciones en los datos existentes:

- Sólo puede deshabilitar las restricciones CHECK y FOREIGN KEY. El resto de las restricciones se tienen que eliminar y volver a agregar.
- Para deshabilitar la comprobación de restricciones cuando se agrega una restricción CHECK o FOREIGN KEY a una tabla con datos existentes, incluya la opción WITH NOCHECK en la instrucción ALTER TABLE.
- Utilice la opción WITH NOCHECK si los datos existentes no van a cambiar. Los datos tienen que cumplir las restricciones CHECK si van a ser actualizados.
- Asegúrese de que deshabilitar la comprobación de la restricción es una acción apropiada. Puede ejecutar una consulta para cambiar los datos existentes antes de decidir agregar una restricción.

Sintaxis parcial

ALTER TABLE *tabla*
[WITH CHECK | WITH NOCHECK]
ADD CONSTRAINT *restricción*

[FOREIGN KEY] [**(***columna*[,…*n*]**)**]
REFERENCES *tablaRef* [**(***columnaRef* [,…*n*]**)**].
[CHECK **(***condicionesBúsqueda***)**]

Ejemplo

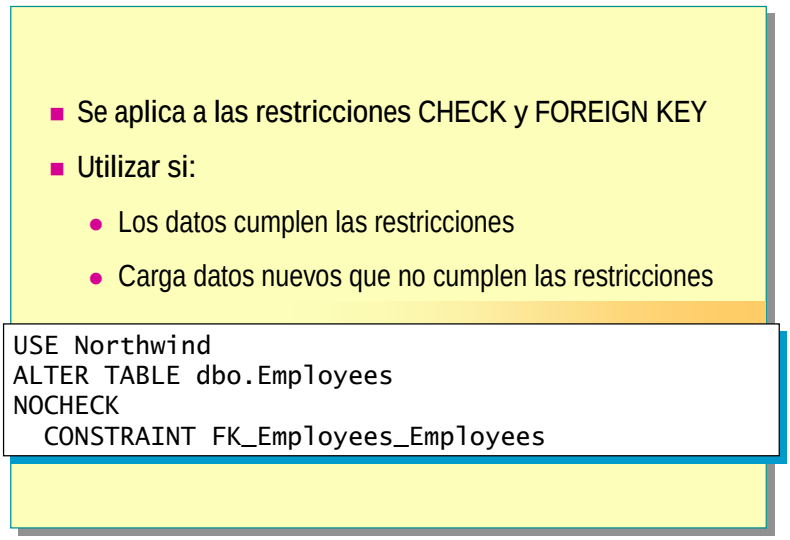
En este ejemplo, se agrega una restricción FOREIGN KEY que comprueba que todos los empleados están asociados a un director válido. La restricción no se implementa en los datos existentes en el momento en que se agrega.

```
USE northwind
ALTER TABLE dbo.Employees
WITH NOCHECK
   ADD CONSTRAINT FK_Employees_Employees
   FOREIGN KEY (ReportsTo)
   REFERENCES dbo.Employees(EmployeeID)
```

Deshabilitación de la comprobación de las restricciones al cargar datos nuevos

Objetivo del tema
Describir cómo se deshabilita la comprobación de restricciones al cargar datos nuevos.

Explicación previa
Esta característica se limita a las restricciones CHECK y FOREIGN KEY.



Se puede deshabilitar la comprobación de restricciones CHECK y FOREIGN KEY existentes, de forma que sea posible modificar o agregar datos a una tabla sin comprobar la restricción.

Para evitar los costos de la comprobación de las restricciones, puede que le interese deshabilitar las restricciones cuando:

- Ya esté seguro de que los datos cumplen las restricciones.
- Desee cargar datos que no cumplan las restricciones. Posteriormente, puede ejecutar consultas para modificar los datos y volver a habilitar las restricciones.

Importante La deshabilitación de restricciones en una tabla no afecta a las restricciones de otras tablas que hagan referencia a la tabla original. Las actualizaciones de una tabla siguen pudiendo generar errores de infracción de restricciones.

La habilitación de una restricción que ha estado deshabilitada requiere la ejecución de otra instrucción ALTER TABLE que contenga una cláusula CHECK o CHECK ALL.

Sintaxis parcial

```
ALTER TABLE tabla
  {CHECK | NOCHECK} CONSTRAINT
  {ALL | restricción[,...n]}
```

Ejemplo

Este ejemplo deshabilita la restricción **FK_Employees_Employees**. Se puede volver a habilitar si se ejecuta otra instrucción ALTER TABLE con la cláusula CHECK.

```
USE northwind
ALTER TABLE dbo.Employees
NOCHECK
   CONSTRAINT FK_Employees_Employees
```

Para determinar si una restricción está habilitada o deshabilitada en una tabla, ejecute el procedimiento almacenado del sistema **sp_help** o utilice la propiedad **CnstIsDisabled** de la función OBJECTPROPERTY.

Uso de valores predeterminados y reglas

Objetivo del tema
Especificar cómo se crean valores predeterminados y reglas.

Explicación previa
Los valores predeterminados y las reglas son dos métodos adicionales para implementar la integridad de datos.

- Como objetos independientes:
  - Se definen una vez
  - Pueden vincularse a una o más columnas o a tipos de datos definidos por el usuario

```
CREATE DEFAULT phone_no_default
  AS '(000)000-0000'
GO
EXEC sp_bindefault phone_no_default,
  'Customers.Phone'
```

```
CREATE RULE regioncode_rule
  AS @regioncode IN ('IA', 'IL', 'KS', 'MO')
GO
EXEC sp_bindrule regioncode_rule,
  'Customers.Region'
```

Los valores predeterminados y las reglas son objetos que se pueden asociar con una o varias columnas o tipos de datos definidos por el usuario, lo que permite que se definan una sola vez y se utilicen varias veces. Un inconveniente del uso de valores predeterminados y reglas es que no cumplen el estándar ANSI.

Creación de un valor predeterminado

Si no se especifica un valor cuando se insertan datos, el valor predeterminado especifica el valor de la columna a la que el objeto está asociado. Considere estos hechos antes de crear valores predeterminados:

Punto clave
No puede usar una restricción de valor predeterminado en una columna con un tipo de datos definido por el usuario si ya hay un valor predeterminado asociado al tipo de datos o a la misma.

- Toda regla asociada a la columna y los tipos de datos valida el valor de un valor predeterminado.
- Cualquier restricción CHECK sobre la columna debe validar el valor de un valor predeterminado.
- No se puede crear una restricción DEFAULT en una columna que esté definida con un tipo de datos definido por el usuario si dicha columna o tipo de datos ya tuvieran asociado un valor predeterminado.

Sintaxis

CREATE DEFAULT *predeterminado*
  AS *expresiónConstante*

Asociación de un valor predeterminado

Después de crear un valor predeterminado, tiene que asociarlo a una columna o a un tipo de datos definido por el usuario mediante la ejecución del procedimiento almacenado del sistema **sp_bindefault**. Para deshacer dicha asociación, ejecute el procedimiento almacenado del sistema **sp_unbindefault**.

Ejemplo

En este ejemplo se inserta un marcador de posición con el formato correcto para el número de teléfono, hasta que se proporcione el número de teléfono real.

```
USE Northwind
GO
CREATE DEFAULT phone_no_default
 AS '(000)000-0000'
GO
EXEC sp_bindefault phone_no_default, 'Customers.Phone'
```

Creación de una regla

Las reglas especifican los valores aceptables que se pueden insertar en una columna. Garantizan que los datos se encuentran dentro de un intervalo de valores especificado, coinciden con un patrón concreto o con las entradas de una lista especificada. Tenga en cuenta estos hechos acerca de las reglas:

- La definición de una regla puede contener cualquier expresión válida para una cláusula WHERE.
- Una columna o un tipo de datos definido por el usuario sólo puede tener asociado una regla.

Sintaxis

CREATE RULE *regla*
 AS *condición*

Asociación de una regla

Después de crear una regla, tiene que asociarla a una columna o a un tipo de datos definido por el usuario mediante la ejecución del procedimiento almacenado del sistema **sp_bindrule**. Para deshacer la asociación de la regla, ejecute el procedimiento almacenado del sistema **sp_unbindrule**.

Ejemplo

En este ejemplo, la regla asegura que sólo se acepten los estados especificados.

```
USE Northwind
GO
CREATE RULE regioncode_rule
 AS @regioncode IN ('IA', 'IL', 'KS', 'MO')
GO
EXEC sp_bindrule regioncode_rule, 'Customers.Region'
```

Eliminación de un valor predeterminado o una regla

La instrucción DROP quita un valor predeterminado o una regla de la base de datos.

Sintaxis

DROP DEFAULT *predeterminado* [,...*n*]

Sintaxis

DROP RULE *regla* [, ...*n*]

Decisión del método de implementación que va a utilizar

Objetivo del tema
Mostrar las ventajas y los inconvenientes de los distintos componentes de integridad de datos.

Explicación previa
Debe tener en cuenta la funcionalidad y los costos de rendimiento cuando determine qué métodos va a utilizar para implementar la integridad de datos.

Componente de integridad de datos	Funcionalidad	Costos de rendimiento	Antes o después de la transacción
Restricciones	Media	Baja	Antes
Valores predeterminados y reglas	Baja	Baja	Antes
Desencadenadores	Alta	Medio-alto	Después
Tipos de datos, Null/Not Null	Baja	Baja	Antes

Puntos clave
Utilice restricciones si es posible.
Utilice valores predeterminados y reglas si necesita objetos independientes.
Utilice desencadenadores sólo cuando requiera una lógica compleja.

Debe tener en cuenta la funcionalidad y los costos de rendimiento cuando determine qué métodos va a utilizar para implementar la integridad de datos.

- Lo mejor es utilizar la integridad declarativa para la lógica de integridad fundamental, como al determinar los valores válidos y al mantener las relaciones entre las tablas.
- Si desea mantener datos redundantes complejos que no forman parte de una relación de claves principales o externas, debe utilizar desencadenadores o procedimientos almacenados.

  Sin embargo, como los desencadenadores no se activan hasta que tiene lugar una modificación, la comprobación de errores ocurre después de que la instrucción se termina. Cuando un desencadenador detecta una infracción, tiene que deshacer los cambios.

Componente de integridad de datos	Efecto	Funcionalidad	Costos de rendimiento	Antes o después de la modificación
Restricciones	Se definen con la tabla y validan los datos antes de empezar la transacción, lo que mejora el rendimiento.	Media	Baja	Antes
Valores predeterminados y reglas	Implementan la integridad de los datos como objetos independientes que se pueden asociar con una o varias tablas.	Baja	Baja	Antes
Desencadenadores	Proporcionan funcionalidad adicional, como cascadas y lógica más compleja. Las modificaciones se tienen que deshacer.	Alta	Medio alto	Después (excepto para desencadenadores INSTEAD OF)
Tipos de datos, Null/Not Null	Proporcionan el nivel más bajo de integridad de datos. Se implementan para cada columna cuando se crea una tabla. Los datos se validan antes de que comience una transacción.	Baja	Baja	Antes