III. CREACION DE BASE DE DATOS

El propósito de una base de datos es almacenar y retornar información, estas acciones no serian posibles sin un gestor de base de datos. Las tablas son la estructura básica de una base de datos, y la elección del diseño afectará la cantidad de espacio consumido en disco, la cantidad de memoria que se consumirá cuando se procesen datos, y las consultas necesarias para manipular los datos a través de una aplicación. Este tema le dará las bases necesarias para diseñar base de datos de altos rendimientos, se iniciará determinando la estructuración de las columnas en una tabla para forzar una estructura que soportará los datos de las reglas de negocio usando el mínimo de recursos de almacenamiento, permitiéndole definir apropiadamente las columnas, y asignar permisos de acceso a las tablas.

Creando una base de datos

Los elementos básicos para la creación de una base de datos son:

- El nombre de la base de datos
- El tamaño de la base de datos
- Los archivos donde residirá la base de datos

Se puede crear una base de datos utilizando el administrador de SQL o utilizando el analizador de consultas, utilizando el comando CREATE DATABASE. El proceso de la creación de una base de datos también crea un archivo para el registro de transacciones.

La información acerca de cada base de datos de SQL es almacenada en las bases de datos de sistema, en la MASTER, es por ello que puede utilizar esta base para ejecutar el comando que creará la base de datos nueva.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

La creación de una base de datos, es el proceso de asignar el nombre de la base, y la determinación del tamaño y ubicación física de la base. Cuando la nueva base es creada, esta es copiada a partir de la plantilla de la base MODEL. Cualquier opción o ambientación en la base MODEL será copiada a las nuevas bases creadas.

La sintaxis para la creación de una base nueva a través del analizador de consultas es:

```
CREATE DATABASE database_name
[ON
        { [PRIMARY] (NAME = logical_file_name,
        Filename = 'os_file_name'
        [, SIZE = size ]
        [, MAXSIZE = [max_size / unlimited } ]
        [, FILEGROWTH = growth_increment ] )
        } [,…n]
        ]
        [ LOG ON
        { (NAME=  'logical_file_name',
                FILENAME = 'os_file_name'
                [, SIZE = size ]
                [, MAXSIZE = {max_size / unlimited}]
                [, FILEGROWTH = growth_increment] )
        } [,….n]
]
```


Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

[COLLATE collation_name]

Cuando se crea una base de datos, se crean los siguientes parámetros:

PRIMARY: este parámetro especifica el archivo del grupo primario, este contiene todas las bases de sistema, también contiene todos los objetos. Cada base de datos contiene un archivo de datos primario, este archivo es el punto de inicio de la base de datos y el punto para el resto de archivos en la base de datos. La extensión del archivo primario es .mdf, sino se especifica la palabra PRIMARY, el primer archivo listado en la sintaxis será el archivo primario.

FILENAME: este parámetro especifica el nombre y ruta para el archivo de sistema operativo. La ruta del archivo de sistema debe especificar un fólder en el servidor donde SQL ha sido instalado.

SIZE: este parámetro especifica el tamaño de los datos o del log de archivo, puede especificar el tamaño en MB, que es el valor por defecto, o KB; el tamaño mínimo es 512 KB para ambos, el archivo de datos y el de log. El tamaño especificado para el archivo primario debe ser como mínimo igual al tamaño de la base de datos MODEL.

MAXSIZE: este parámetro especifica el tamaño máximo que cada archivo puede crear, usted puede asignar el tamaño en MB que es el valor por defecto, sino se asigna un tamaño, este puede crecer hasta llenar el disco.

FILEGROWTH: este parámetro le especifica en que porcentaje o que valor en MB, KB puede ir creciendo el LOG de transacciones.

COLLATION: este parámetro asigna la colación para la base de datos, este determina las reglas para el uso de caracteres para cada lenguaje o alfabeto utilizado.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

El siguiente ejemplo crea una base de datos llama SAMPLE con un archivo primario de datos de 10 MB y 3 MB para el LOG en la instancia por defecto.

```
CREATE DATABASE SAMPLE

ON

    PRIMARY (NAME = SAMPLEDATA,

    FILENAME = 'C:\DATOS_SQL\SAMPLE.MDF',

    SIZE = 10 MB,

    MAXSIZE = 15 MB,

    FILEGROWTH = 20% )

LOG ON

    ( NAME = SAMPLELOG,

    FILENAME = 'C:\DATOS_SQL\SAMPLE.LDF',

    SIZE = 3 MB,

    MAXSIZE = 5 MB,

    FILEGROWTH = 1 MB )

COLLATE SQL_LATIN1_GENERAL1_CP1_CI_AS
```

Como funciona el LOG de transacciones:

- La modificación de los datos es enviada por la aplicación.
- Las páginas de datos son localizadas o la información es leída del buffer de cache y modificada.
- La modificación es grabada en el LOG de transacciones.
- Un punto de chequeo es escrito al finalizar la transacción de la base de datos.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

SQL almacena cada transacción en un LOG de transacciones para mantener la consistencia de la base de datos y poder restablecerla. El LOG es un área de almacenamiento que automáticamente lleva una pista de los cambios realizados a la base de datos. SQL almacena las modificaciones en el LOG, estas solo son almacenada si la las modificaciones han tenido éxito en la base de datos.

Si el sistema falla, el proceso automáticamente se recupera utilizando el LOG de transacciones y deshace todas las transacciones incompletas y lleva el sistema hasta la ultima transacción completada con éxito.

Los marcadores de transacción en el log son utilizados durante una recuperación automática para determinar el punto de inicio y de fin de una transacción. Una transacción es considerada completa cuando el marcador BEGIN TRANSACTION esta asociado con el marcador COMMIT TRANSACTION. Las páginas de datos son escritas al disco cuando un punto de cheque ocurre.

Parametrizando las opciones de base de datos

Después que se ha creado una base de datos, se puede parametrizar la base utilizando el SQL MANAGMENT o utilizando la sentencia en el analizador de consultas ALTER DATABASE.

Puede configurar un número de opciones de la base, pero solo puede realizarlo con una base a la vez. Si desea que las opciones afecten a todas las bases nuevas, cambie los parámetros en la base MODEL y así todas las bases nuevas llevaran esos cambios adoptados en los parámetros.

La siguiente lista muestra las opciones utilizadas frecuentemente:

- AUTO_CREATE_STATISTICS: automáticamente crea cualquier estadística necesaria para la optimización de una consulta, por defecto esta en ON.
- AUTO_UPDATE_STATISTICS: automáticamente se actualiza las estadísticas necesarias para la optimización de una consulta, por defecto esta en ON.
- CURSOR_CLOSE_ON_COMMIT: automáticamente se cierran todos los cursores abiertos cuando una transacción es finalizada. El valor por defecto es OFF, y los cursores permanecen abiertos durante toda la sesión.
- CURSOR_DEFAULT_LOCAL / GLOBAL: el local limita el ámbito de el cursor, si esta localizado en un procedimiento almacenado, disparador; en el área donde fue creado el cursor.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Global es el valor por defecto, el ámbito es en toda la conexión.

- RECOVERY FULL / BULK_LOGGED / SIMPLE: FULL, provee recuperabilidad total si el medio falla, este es el valor por defecto. BULK LOGGED utiliza un menor espacio del LOG, por que el archivo es mínimo, esto genera grandes riesgos de exposición. SIMPLE recupera la base solamente al último respaldo completo o parcial.
- TORN_PAGE_DETECTION: permite que SQL detecte operaciones incompletas de entrada o salida, causadas por fallas de energía u otras fallas del sistema, el valor por defecto es ON.
- ANSI_NULL_DEFAULT: permite el uso de los valores NULL por defecto, el valor por defecto es NOT NULL.
- ANSI_NULLS: cuando esta en ON las comparaciones entre 2 valores nulos es desconocida, cuando esta OFF la comparación devuelve un valor TRUE si los dos valores comparados son NULL, por defecto se encuentra en OFF.
- READ_ONLY / READ_WRITE: el primero determina si una base estará en modo de lectura, esta opción es utilizada para brindar soporte a la base y después regresar a la operación normal que es READ_WRITE.
- SINGLE_USER / RESTRICTED_USER / MULTIUSER: el primero habilita la conexión solo para un usuario, las demás conexiones son dadas de baja, la segunda opción permite que solo usuarios que le dan mantenimiento a la base puedan conectarse como lo son: db_owner, db_creator, sysadmin; el tercero es el valor por defecto, permite que todos los usuarios que tengan permisos puedan conectarse a la base.

CREANDO TABLAS

La analogía de base para una tabla de base de datos es que es como una sola hoja de trabajo dentro de una Hoja de cálculo. Como se trabaja con una hoja de calculo, se introduce la información en filas y columnas, bien diseñada las hojas de cálculo generalmente tienen cabeceras de columna que le dan una idea de la clase de datos que se encuentran en una columna. El problema de trabajar con datos en una hoja de cálculo, es que no garantiza el cumplimiento de cualquier estructura. Puede colocar cualquier tipo de datos en todas las columnas, sin limitación. En las Tablas de base de datos también se almacenan registros de los datos de las columnas y cada una de estas columnas tiene un nombre asociado, pero lo que distingue a una tabla de una base

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

de datos y de una hoja de cálculo de Microsoft Office Excel, es la aplicación restringida de los datos que puede introducir en una columna. SQL Server impone la estructura de datos mediante el uso de tipos de datos, así como las propiedades que se puede añadir al definir una columna. En esta lección, usted aprenderá cómo tomar las mejores decisiones en la definición de tipos de datos y propiedades de las columnas, cómo crear una tabla, la manera de asignar los permisos adecuados para permitir el acceso a una tabla.

Qué función desempeña los tipos de datos

Estos se encargan de limitar el tipo de datos que se puede almacenar en una columna y, en algunos casos, incluso limitar la gama de posibles valores en la columna. El tipo de datos que usted elija de una columna es la decisión más importante que usted haga dentro de su base de datos, si se elige un tipo de datos que es demasiado restrictiva, las aplicaciones no podrán almacenar los datos que se deriven de un proceso, dando lugar a un gran esfuerzo de diseño. Si elige una base de datos muy amplia podría producir el costo de hasta consumir más espacio del necesario en el disco y en la memoria, que puede crear un problema de recursos y de rendimiento. Cuando se selecciona un tipo de datos para una columna, debe escoger el tipo de datos que permite todos los datos de los valores que usted espera que se almacenan al hacerlo en la menor cantidad de espacio posible. Los tipos de datos de SQL Server se dividen en siete categorías generales, que son:

- **Numérico exacto:** almacena números precisos, ya sea con o sin decimales.
- **Numéricos aproximados:** almacena valores numéricos con o sin decimales.
- **Monetarios:** almacena valores numéricos con decimales; utilizado específicamente en los valores de moneda con un máximo de hasta cuatro decimales.
- **Fecha y hora:** almacena información de fecha y hora y permite especiales cronologías de ejecución, tales como el rechazo de un valor 30 de febrero.
- **Carácter:** almacena caracteres basados en los valores de longitud variable.
- **Binario:** amacena datos en estricto binario (0 y 1) la representación.
- **Propósitos especiales:** tipos de datos complejos que requieren tratamiento especializado, como documentos XML únicos a nivel mundial.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Veamos cada una de estas categorías de tipos de datos para ver cómo puede utilizar los diferentes tipos de datos para proporcionar la definición básica de cada columna en una tabla. Puede utilizar estos tipos de datos en la definición permanente de tablas, tablas temporales, y las variables. Hay pocas restricciones a los tipos de datos que se pueden utilizar en los procedimientos almacenados, disparadores, y funciones.

Numérico exacto

Puede utilizar tipos de datos numéricos exactos para almacenar números que tengan cero o más decimales. Puede manipular los números que tenga almacenados en estos tipos de datos utilizando cualquier operación matemática sin necesidad de ninguna manipulación especial. El almacenamiento es definido también con precisión, por lo que los datos almacenados en estos tipos de datos retornan el mismo valor, ya sea en un Intel o un AMD. La siguiente tabla lista los tipos Numérico exacto que SQL Server soporta.

Data Type	Storage	Value Range	Purpose
bigint	8 bytes	−2E63 to 2E63 −1	Stores very large whole numbers that can be positive or negative
int	4 bytes	−2E31 to 2E31 −1	Stores whole numbers that can be positive or negative
smallint	2 bytes	−32,768 to 32,767	Stores whole numbers that can be positive or negative
tinyint	1 byte	0 to 255	Stores a small range of positive whole numbers
decimal(p,s)	5−17 bytes depending on the precision	−10E38 + 1 to 10E38 −1	Stores decimals up to a maximum of 38 places
numeric(p,s)	5−17 bytes depending on the precision	−10E38 + 1 to 10E38 −1	Functionally equivalent to decimal, and can be used interchangeably with decimal

El tipo de datos numérico y decimal acepta parámetros adicionales para completar la definición del tipo de dato. Estos parámetros definen la precisión y la escala para el tipo de datos. Por Ejemplo, un decimal (12,4) define a un valor decimal que puede tener hasta 12 dígitos en total, con cuatro dígitos decimales.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Los tipos de datos más comunes de este grupo son int y decimal. Puede utilizar un tipo de datos decimal para almacenar valores enteros, pero hacerlo requiere muchos bytes de almacenamiento por fila y no deben utilizarse para este fin. Aunque el tipo de datos int puede almacenar tantos números positivos y negativos, la porción negativa es muy raramente utilizada. Si el rango de valores que desea almacenar en una columna no excede de 32767, puede guardar dos bytes para cada fila, utilizando en lugar de int un smallint. Si los valores se encuentran en un rango de 0 a 255, puede guardar 1 bytes para cada fila utilizando Tinyint.

Importancia en la utilización del espacio en disco

El Ahorro de dos o tres bytes de almacenamiento por línea no parece algo mucho en comparación con los 250 GB + unidades de disco duro que usted ahora puede adquirir por unos cientos de dólares, libras, euros, yenes, o lo que sea moneda con la que esté trabajando. Sin embargo, el disco duro de almacenamiento es una preocupación de menor cuantía. Si almacena 1 millón de filas de datos en una tabla, que es muy común, los bytes por fila salvado añadirán hasta 2 o 3 MB. A pesar de que no suena como mucho, considere la posibilidad de que el espacio en la memoria puede agotarse si un usuario ejecuta una consulta que devuelve todos los registros en la tabla. Usted también salvará al procesador de ejecutar miles de ciclos a la vez. El problema de espacio se hace aún mayor cuando se unan dos cuadros juntos. Unir dos columnas INT juntas consume ocho bytes de memoria, así como el cálculo correspondiente a éste, si ambas tablas tienen 1 millón de filas y la necesidad de ser leída por completo, la operación consume alrededor de 8 MB de espacio en la memoria. Si podría tener almacenados los datos en una columna smallint o tinyint en cambio, el ahorro de memoria para esta consulta sería 4-6 MB. y este es el ahorro para una sola consulta. Considere qué pasaría si miles de preguntas que se están tramitando en contra de la Base de datos, y se puede ver cómo uno o dos bytes por fila, la economía basada en el tipo de datos que usted puede hacer puede marcar la diferencia entre un ambiente con buen rendimiento y uno de muy malos resultados.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Tipos de datos numéricos aproximados

La aproximación de los tipos de datos numéricos pueden almacenar valores decimales. Sin embargo, los datos almacenados en un Flota o real, el tipo de datos sólo es exacta a la precisión especificada en la definición de tipo de datos. Cualquier dígito a la derecha no se garantiza que se almacene exactamente. Por ejemplo, si 1.00015454 almacenados en un tipo de datos definido como float (8), la columna se garantiza a que regrese sólo con precisión 1.000154. SQL Server elimina cualquier dígito más a la derecha cuando se almacenan los datos. Por lo tanto, los cálculos de estos tipos de datos compuestos pueden producir errores de redondeo. La transferencia de las bases de datos que contienen tablas con estos tipos de datos entre los procesadores de Intel y AMD también introduce errores. La tabla lista los tipos de datos numéricos aproximados de SQL Server.

Data Type	Storage	Value Range	Purpose
float(p)	4 or 8 bytes	−2.23E308 to 2.23E308	Stores large, floating point numbers that exceed the capacity of a decimal data type
real	4 bytes	−3.4E38 to 3.4E38	Still valid, but replaced by float to meet the SQL-92 standard

Los tipos de datos float aceptan un parámetro en la definición que determina el número de dígitos para almacenar. Por ejemplo, un float (8), la columna almacenará siete dígitos, y cualquier cosa superior estará sujeta a errores de redondeo. Debido a la imprecisión asociada a estos tipos de datos, que son poco utilizadas, usted debería considerar el uso de float únicamente en los casos en los que un tipo de datos numérico exacto no sea lo suficientemente grande para almacenar los valores.

Tipo de dato monetario

Este tipo de datos está diseñado para almacenar el valor de las divisas con cuatro decimales de precisión. La siguiente tabla lista los tipos de datos monetarios de SQL Server.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Data Type	Storage	Value Range	Purpose
money	8 bytes	–922,337,203,685,477.5808 to 922,337,203,685,477.5807	Stores large currency values
smallmoney	4 bytes	–214,748.3648 to 214,748.3647	Stores small currency values

Los tipos de datos smallmoney rara vez se define en las bases de datos, aunque este tipo de datos es la más precisa elección para muchas aplicaciones que se ocupan de productos y pedidos, es mucho más común que el tipo Money en estas bases de datos, debido al espacio de 4 bytes utilizado de almacenamiento y al redondeo de 4 decimales. Aunque el Money y smallmoney, tipos de datos que están diseñados para almacenar valores de las divisas, que rara vez se utiliza en aplicaciones financieras. En lugar de ello, estas aplicaciones utilizan decimal, por que estas necesitan realizar cálculos de 6, 8 y hasta 12 decimales.

Tipo de dato fecha y hora

En el almacenamiento de datos, nada genera más controversia que el modo de almacenar fechas y horas. Algunas aplicaciones necesitan almacenar sólo la fecha, otras aplicaciones necesitan almacenar sólo la hora, y otras aplicaciones se ven en la necesidad de almacenar tanto las fechas y las horas. Lamentablemente, SQL Server almacena este tipo de datos sólo juntos como una fecha y una hora, por ejemplo, 2008-02-14 20:53:36.153, que es la precisión de milisegundos en el reloj del sistema cuando comencé a escribir esta frase. La siguiente tabla lista los tipos de datos de fecha y hora de SQL Server.

Data Type	Storage	Value Range	Purpose
datetime	8 bytes	January 1, 1753, through December 31, 9999, with an accuracy of 3.33 milliseconds	Stores large date and time values.
smalldatetime	4 bytes	January 1, 1900, through June 6, 2079, with an accuracy of 1 minute	Stores a smaller range of date and time values

Los tipos de datos datetime y smalldatetime se almacenan internamente como enteros, el tipo de dato datetime es almacenado como un par de números enteros de cuatro bytes, que en conjunto representan el número de milisegundos desde la medianoche del 1 de enero de 1753. Los primeros cuatro bytes almacena la fecha, y los siguientes cuatro bytes almacenan el tiempo. El

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

tipo de datos smalldatetime es almacenado como un par de enteros de dos bytes, que en conjunto representan el número de minutos desde la medianoche del 1 de enero de 1900. Los dos primeros bytes almacenan la fecha, y los dos bytes restantes almacenan el tiempo.

Tipo de dato Carácter

Para almacenar datos de carácter, debe seleccionar uno de los tipos de datos diseñado para este propósito. Cada uno consume uno o dos bytes de almacenamiento por cada carácter, en función de si el tipo de datos de uso es del Instituto Nacional Americano de Estándares (ANSI) de codificación o codificación Unicode.

Data Type	Storage	Number of Characters	Purpose
char(n)	1–8,000 bytes	Maximum of 8,000 characters	ANSI data type that is fixed width
nchar(n)	2–8,000 bytes	Maximum of 4,000 characters	Unicode data type that is fixed width

Data Type	Storage	Number of Characters	Purpose
varchar(n)	1–8,000 bytes	Maximum of 8,000 characters	ANSI data type that is variable width
varchar(max)	Up to 2 GB	Up to 1,073,741,824 characters	ANSI data type that is variable width
nvarchar(n)	2–8,000 bytes	Maximum of 4,000 characters	Unicode data type that is variable width
nvarchar(max)	Up to 2 GB	Up to 536,870,912 characters	Unicode data type that is variable width
text	Up to 2 GB	Up to 1,073,741,824 characters	ANSI data type that is variable width
ntext	Up to 2 GB	Up to 536,870,912 characters	Unicode data type that is variable width

¿Por qué hay tantos tipos de datos de carácter que parecen ser equivalentes a cada uno de los Otros? Las diferencias en los tipos de datos pueden ser sutiles, pero pueden ser importantes. Char, ya sea ANSI o Unicode, es un ancho fijo de tipo de dato. Por lo tanto, que consume la misma cantidad de espacio de almacenamiento, independientemente del número de caracteres que se almacenan en la columna. Por ejemplo, una columna char (30) consume 30 bytes de espacio de

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

almacenamiento independientemente de si almacena un solo carácter o son 30 caracteres en la columna.

El espacio no utilizado se rellena con espacios en blanco hasta el máximo de almacenamiento especificado de la columna. Sin embargo, una columna varchar (30) consume sólo un byte para cada carácter que se almacena en la columna.

El text y tipo de datos ntext se han diseñado para almacenar grandes cantidades de caracteres de datos. Sin embargo, las columnas ntext y text no son permitidas en muchas operaciones. Por ejemplo, no se puede usar con un operador de igualdad o de unión juntos. Muchas funciones de sistema no pueden utilizar el text y ntext.

Debido a estas limitaciones, SQL Server 2008 presenta el varchar (max) y nvarchar (máximo). Estos tipos de datos combinan las capacidades de ambos text / ntext.. Se pueden almacenar hasta 2 GB de datos y no tienen cualquier restricción de las operaciones que se pueden realizar con ellos o sobre las funciones que se puedan usar con ellos.

Tipo de dato Binary

Hay muchas ocasiones en las que se necesite almacenar datos binarios. Por lo tanto, SQL Server proporciona tres tipos de datos que le permiten almacenar varias cantidades de datos binarios en una tabla. Los tipos de datos binarios que SQL Server permite son:

Data Type	Storage	Purpose
binary(n)	1–8,000 bytes	Stores fixed-size binary data
varbinary(n)	1–8,000 bytes	Stores variable-size binary data
varbinary(max)	Up to 2 GB	Stores variable-size binary data
image	Up to 2 GB	Stores variable-size binary data

Se usan los tipos de datos binarios esencialmente para almacenar archivos dentro de SQL Server. Puede utilizar Binary / varbinary para almacenar archivos pequeños, como un grupo de 4 o 6 KB , archivos que contienen una variedad de datos en el formato nativo.

El tipo de datos más populares dentro de este grupo es el tipo de datos de imagen. Este tipo de datos tiene un nombre desafortunado, no es utilizada exclusivamente para almacenar imágenes,

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

tales como una biblioteca de fotografías de una reciente vacación. Aunque pueda almacenar las imágenes en un tipo dato imagen, también puede utilizar este tipo de datos para almacenar Word, Excel, PDF, y los documentos de Visio. Usted puede almacenar cualquier archivo que sea de 2 GB o menos en un tipo de dato imagen. Una de las más famosas implementaciones de este tipo de datos es el proyecto TerraServer, que es una Base de datos multiterabyte de imágenes terrestres de la que se puede acceder en www.terraserver.com.

El varbinary (max), es nuevo para SQL Server 2008. Se puede almacenar la misma cantidad de datos como el tipo de datos imagen, y puede usarse con todas las operaciones y funciones que se pueden utilizar con los tipos de datos binarios / varbinary.

Tipos de datos Especializados

Además de la anterior norma de los tipos de datos, SQL Server proporciona siete tipos de datos para fines muy específicos. La siguiente lista describe los tipos de datos especializados.

Data Type	Purpose
bit	Stores a 0, 1, or *null*. Used for basic "flag" values. TRUE is converted to 1, and FALSE is converted to 0.
timestamp	An automatically generated value. Each database contains an internal counter that designates a relative time counter not associated with an actual clock. A table can have only one *timestamp* column, which is set to the database timestamp when the row is inserted or modified.
uniqueidentifier	A 16-bit GUID used to globally identify a row across databases, instances, and servers.
sql_variant	Can change the data type based on the data that is stored within it. Stores a maximum of 8,000 bytes.
cursor	Used by applications that declare cursors. Contains a reference to the cursor that can be used for operations. This data type cannot be used in a table.
table	Used to hold a result set for subsequent processing. This data type cannot be used for a column. The only time you use this data type is when declaring table variables in triggers, stored procedures, and functions.
Xml	Stores an XML document of up to 2 GB in size. You can specify options to force only well-formed documents to be stored in the column.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Nulabilidad (NULL)

La segunda característica de cualquier definición de columna es si se necesita un valor que sea almacenado. Las bases de datos tienen un llamado especial para construir un null que se puede utilizar para indicar la ausencia de un valor de algo similar a desconocido; no aplicable; Null no es un valor, ni consume espacio de almacenamiento. La mejor manera de entender esta construcción es ver un ejemplo.

Digamos que usted esta diseñando una tabla para almacenar las direcciones de los clientes de su empresa. Ustedes han decidido que cada dirección puede tener hasta tres líneas de la dirección de calle Cada dirección también puede tener una ciudad, un estado o provincia, código postal, y un país. Así se crea una tabla que contiene siete columnas. No todos los clientes necesitaran más de tres líneas para salvar la dirección, por lo que una o dos de estas columnas no serán necesarias para algunas direcciones. Algunos clientes viven en países que no tienen estados o provincias, por lo que esta columna no es necesaria para cada cliente. Además puede no saber el código postal de determinados clientes, Pero tienen que ser capaces de guardar todos los datos que se conoce. Estas cuestiones crean un dilema básico. Se puede pegar un valor ficticio en el que cualquiera de las columnas no tienen valores o los valores no son conocidos cuando se introdujeron los datos. Sin embargo, la inserción de datos ficticio puede causar aún más problemas debido a que está añadiendo datos no válidos a su tabla de datos que podría ser visto y utilizado por un empleado o cliente. Debido a que los datos no fueron explícitamente especificados, se desconoce o no se aplica. En la base de datos, la columna sería nula para designar este estado desconocido.

Cuando definimos las columnas, puede especificar si se permite nulls. Si se Inhabilita el uso de nulls, un usuario debe especificar un valor para la columna. Tenga en cuenta que debido a que es imposible que la ausencia de algo a la igualdad de la ausencia de algo, en otras palabras, no se pueda utilizar nulos en las comparaciones.

Identity

Usted también tiene la posibilidad de especificar una propiedad de identidad para una sola columna en una tabla. La definición de una columna con la propiedad de identidad causa que SQL

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Server genere un número de crecimiento automático. La propiedad identidad toma dos parámetros: el valor inicial y el incremento. El valor inicial se designa a partir del Valor que SQL Server utilizará. El incremento de valor especifica el número que SQL Server añade a este valor a partir de la generación de cada valor. Esta propiedad es Autonumber equivalente a autoincrement valores equivalentes en otros idiomas.

Puede utilizar la propiedad de identidad con los tipos de datos numéricos exactos: bigint, int, smallint, tinyint, decimal, y numérico. Si utiliza decimal o numérico, con la propiedad identidad, debe definirla con 0 decimales.

Columnas Calculadas

También puede crear un tipo especial de columna llamada columna calculada, en el que figura un cálculo que implica una o varias de las otras columnas de la tabla.

Por defecto, la columna calculada contiene una definición para el cálculo, pero no almacena físicamente los datos por defecto. Cuando los datos se devuelven, el cálculo es aplicado al devolver un resultado.

Sin embargo, puede forzar una columna calculada para almacenar físicamente los datos mediante el uso de la palabra clave PERSISTED. Esta palabra clave hace que el al ocurrir el cómputo cuando una fila es insertada o modificada, el resultado del cálculo se almacenan físicamente en la tabla.

CREANDO UNA TABLA

Ahora que ha visto todos los detalles de la columna se puede definir la estructura de una tabla, usted está listo para crear una tabla, puede crear tres tipos diferentes de tablas en SQL Server: permanentes, temporales, y en el ámbito de variables.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Tablas Permanentes

Para crear una tabla, puede utilizar la sentencia CREATE TABLE en Transact-SQL. La sintaxis general de este comando es la siguiente:

CREATE TABLE

[*database_name* . [*schema_name*] . | *schema_name* .] *table_name*

({ <column_definition> | <computed_column_definition> }

[<table_constraint>] [,...n])

[ON { *partition_scheme_name* (*partition_column_name*) | *filegroup*

| "default" }]

[{ TEXTIMAGE_ON { *filegroup* | "default" }]

[;]

Para ejecutar este comando, su usuario debe contar con los permisos para poder crear tablas en la base de datos correspondiente. Cuando utiliza este comando, se crea una tabla en la base de datos que se puede acceder por cualquier usuario con los permisos apropiados.

La cláusula ON específica donde residirá la tabla en su almacenamiento físico. Si no se especifica un filegroup, SQL Server crea la tabla en el filegroup por defecto.

Usando nuestro ejemplo anterior, puede utilizar el comando CREATE TABLE para crear la tabla CustomerAddress de la siguiente manera:

CREATE TABLE dbo.CustomerAddress

(AddressLine1 varchar(30) NOT NULL,

AddressLine2 varchar(30) NULL,

AddressLine3 varchar(30) NULL,

City varchar(50) NOT NULL,

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

StateProvinceID **int** **NULL,**

PostalCode **char(10)** **NULL,**

CountryID **int** **NULL** **)**

La definición de la tabla determina lo siguiente:

- La tabla será creada bajo el esquema dbo.
- Un mínimo de una línea de dirección que tiene un máximo de 30 caracteres debe ser especificado para cada cliente. El espacio de almacenamiento consumido será igual al número de caracteres en la columna.
- Dos líneas de dirección son opcional, las cuales si contienen datos en la columna, serán las que determinan el espacio consumido, el cual tendrá relación directa con la cantidad de caracteres que se almacene en dicha columna.
- Un registro de cliente debe tener una ciudad; la columna puede contener un valor de hasta 50 caracteres de longitud y consume un almacenamiento igual al número de caracteres de la columna.
- Un registro de cliente es opcional que pueda tener un estado o provincia. La columna consume cuatro bytes de almacenamiento y contiene un valor entero.
- Un registro de cliente es opcional que tenga un código postal, cada registro consumirá 10 bytes de almacenamiento.
- Un registro de cliente es opcional que tenga un identificador de país, la columna consumirá un espacio de cuatro bytes de almacenamiento y contendrá valores enteros.

Aunque la definición de la tabla anterior capta con precisión los datos necesarios, usted ya ha notado algunos problemas. Un cliente puede tener una o más direcciones de origen, una o más direcciones profesionales, y una o más direcciones de envío, también puede designar a una determinada dirección la dirección principal. Así es que usted podría tener la tentación de añadir un montón de columnas adicionales para manejar estas situaciones, pero eso sería pensar en términos de una hoja de cálculo, no en una base de datos. En cambio, puede simplemente añadir

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

una columna a la tabla en la que se designa el tipo de dirección y una columna para designar a la dirección principal, tal como lo muestra el siguiente ejemplo:

```
CREATE TABLE dbo.CustomerAddress
(       AddressType             char(4) NOT NULL,
        PrimaryAddressFlag      bit             NOT NULL,
        AddressLine1            varchar(30)     NOT NULL,
        AddressLine2            varchar(30)     NULL,
        AddressLine3            varchar(30)     NULL,
        City                    varchar(50)     NOT NULL,
        StateProvinceID         int             NULL,
        PostalCode              char(10)        NULL,
        CountryID               int             NULL            )
```

Por ahora, vamos a pasar por alto las cuestiones relativas a las columnas StateProvinceID y CountryID, ya que se cubrirá mas adelante sobre las limitaciones.

Pero todavía hay otro problema con esta definición de tabla. Estamos capturando direcciones, pero no tenemos forma de saber la dirección que corresponde con la de cada cliente. Para completar la estructura de la tabla y permitir que una dirección se asocie con la de un cliente, tenemos que añadir una columna más a la tabla: la columna CustomerAddressID int, que se define con la propiedad de identidad. La definición se completa de la siguiente manera:

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

```
CREATE TABLE dbo.CustomerAddress
(       CustomerAddressID       int             IDENTITY(1,1),
        AddressType             char(4) NOT NULL,
        PrimaryAddressFlag      bit             NOT NULL,
        AddressLine1            varchar(30)     NOT NULL,
        AddressLine2            varchar(30)     NULL,
        AddressLine3            varchar(30)     NULL,
        City                    varchar(50)     NOT NULL,
        StateProvinceID         int             NULL,
        PostalCode              char(10)        NULL,
        CountryID               int             NULL            )
```

Nota: Eliminando tablas

Puede utilizar el comando DELETE para eliminar registros de una tabla. Y para eliminar toda la tabla, debe usar el comando DROP TABLE. Para ejecutar este comando, usted debe tener privilegios en su usuario para poder eliminar tablas de la base de datos.

TABLAS TEMPORALES

Tablas temporales, como su nombre indica, son las estructuras de tabla temporal. Las tablas temporales pueden ser global o local y pueden ser creados por cualquier usuario. Todas las tablas se crean en la base de datos tempdb.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Una tabla temporal local es visible sólo para el usuario que ha creado la tabla y estas son eliminadas cuando el usuario cierra la conexión. Usted puede crear una tabla temporal local utilizando el comando CREATE TABLE y anteponiendo una almohadilla (#) al nombre de tabla.

El siguiente ejemplo muestra como crear una tabla temporal local.

```
CREATE TABLE #CustomerAddress
(       CustomerAddressID       int             IDENTITY(1,1),
        AddressType             char(4) NOT NULL,
        PrimaryAddressFlag      bit             NOT NULL,
        AddressLine1            varchar(30)     NOT NULL,
        AddressLine2            varchar(30)     NULL,
        AddressLine3            varchar(30)     NULL,
        City                    varchar(50)     NOT NULL,
        StateProvinceID         int             NULL,
        PostalCode              char(10)        NULL,
        CountryID               int             NULL            )
```

USO DE VARIABLES DE TABLAS

Las variables de tablas proporcionan una alternativa a las tablas temporales y se pueden utilizar en funciones, disparadores y procedimientos almacenados. En lugar de almacenar la tabla y todos los datos dentro de la base de datos tempdb en el disco, una variable de tabla y todos los datos asociados son almacenados en la memoria. Sin embargo, si la cantidad de datos colocados en la variable tabla causa que se requiera más espacio de almacenamiento del que está disponible en la memoria, el desbordamiento en cola será ubicado en el disco dentro de tempdb.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

Las variables de tabla son locales a la función, disparadores o procedimiento almacenado en el que se crearon, y son automáticamente destruidos cuando el objeto se cierra.

Usted pude crear la tabla Customer Address como una variable de tabla, para declarar la variable de tabla se le indica anteponiéndole al nombre de la tabla el carácter @, de la siguiente manera:

```
DECLARE @CustomerAddress TABLE
(       CustomerAddressID       int             IDENTITY(1,1),
        AddressType             char(4) NOT NULL,
        PrimaryAddressFlag      bit             NOT NULL,
        AddressLine1            varchar(30)     NOT NULL,
        AddressLine2            varchar(30)     NULL,
        AddressLine3            varchar(30)     NULL,
        City                    varchar(50)     NOT NULL,
        StateProvinceID         int             NULL,
        PostalCode              char(10)        NULL,
        CountryID               int             NULL            )
```

Asignando permisos

Ahora que ha creado su tabla, debes dar permisos para que los usuarios puedan acceder a la tabla. SQL Server no proporciona ningún acceso a menos que el permiso haya sido otorgado explícitamente.

Un miembro dentro del rol de servidor sysadmin tiene derechos ilimitados a cualquier objeto dentro de la instancia de SQL Server, por lo que un miembro de este rol puede realizar cualquier

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

operación sobre una tabla. Un miembro de la base de datos con rol propietario se le ha concedido permiso para realizar cualquier operación sobre cualquier objeto en la base de datos que es propietario, por lo que un miembro de este rol puede realizar cualquier operación sobre una tabla. Además, el dueño de una tabla ya se le ha concedido expresamente la autoridad para llevar a cabo cualquier operación. Todos los demás usuarios deberán contar con los permisos asignados para trabajar con una tabla.

Existen 6 permisos que se pueden asignar a una tabla, los cuales son los siguientes:

- CREATE TABLE
- ALTER TABLE
- SELECT
- INSERT
- UPDATE
- DELETE

Usted puede utilizar la palabra clave especial ALL para conceder todos los permisos en la tabla para un rol en específico. Sin embargo, siempre debe explícitamente listar cada permiso que usted permitirá. La declaración general para asignar los permisos es la siguiente:

GRANT { ALL [PRIVILEGES] }

| *permission* [(*column* [,...*n*])] [,...*n*]

[ON [*class* ::] *securable*] TO *principal* [,...*n*]

[WITH GRANT OPTION] [AS *principal*]

La cláusula ON especifica al objeto que usted está otorgando permisos, mientras que la cláusula TO especifica el rol de la base de datos al cual se le asignan los permisos.

Para las tablas, es posible la concesión de permisos en un subconjunto de las columnas en la tabla. No hay servicio de la concesión de permisos para un subconjunto de filas de una tabla.

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)

La opción WITH GRANT le permite conceder permisos a un rol cuyos miembros les pueden conceder los permisos a otros usuarios o roles. Nunca debe utilizar esta opción porque toma el control de la seguridad fuera de las manos del propietario de la tabla.

Para la tabla Customer Address se le pueden conceder los permisos SELECT, INSERT, UPDATE, DELETE a un determinado rol utilizando la sintaxis siguiente:

- GRANT SELECT, INSERT, UPDATE, DELETE ON CustomerAddress TO <database role>

Fuente: Libros en pantalla de SQL Server 2008 de Microsoft Corporation (Edición de Enero de 2009)