INTRODUCCIÓN AL SQL. TRANSACT-SQL

SQL SERVER 2005

Manual de Referencia para usuarios

Salomón Ccance
CCANCE WEBSITE

INTRODUCCIÓN AL SQL. TRANSACT SQL

CONCEPTOS BÁSICO DE SQL

SQL (Structured Query Language), Lenguaje Estructurado de Consulta es el lenguaje utilizado para definir, controlar y acceder a los datos almacenados en una base de datos relacional.

Como ejemplos de sistemas gestores de bases de datos que utilizan SQL podemos citar DB2, SQL Server, Oracle, MySql, Sybase, PostgreSQL o Access.

El SQL es un lenguaje universal que se emplea en cualquier sistema gestor de bases de datos relacional. Tiene un estándar definido, a partir del cual cada sistema gestor ha desarrollado su versión propia.

En SQL Server la versión de SQL que se utiliza se llama TRANSACT-SQL.

EL SQL en principio es un lenguaje orientado únicamente a la definición y al acceso a los datos por lo que no se puede considerar como un lenguaje de programación como tal ya que no incluye funcionalidades como son estructuras condicionales, bucles, formateo de la salida, etc. (aunque veremos que esto está evolucionando).

Se puede ejecutar directamente en modo interactivo, pero también se suele emplear embebido en programas escritos en lenguajes de programación convencionales. En estos programas se mezclan las instrucciones del propio lenguaje (denominado anfitrión) con llamadas a procedimientos de acceso a la base de datos que utilizan el SQL como lenguaje de acceso. Como por ejemplo en Visual Basic, Java, C#, PHP .NET, etc.

Las instrucciones SQL se clasifican según su propósito en tres grupos:

- ✓ El DDL (Data Description Language) Lenguaje de Descripción de Datos.
- ✓ El DCL (Data Control Language) Lenguaje de Control de Datos.
- ✓ El DML (Data Manipulation Language) Lenguaje de Manipulación de Datos.

- El DDL, es la parte del SQL dedicada a la definición de la base de datos, consta de sentencias para definir la **estructura** de la base de datos, permiten crear la base de datos, crear, modificar o eliminar la estructura de las tablas, crear índices, definir reglas de validación de datos, relaciones entre las tablas, etc. Permite definir gran parte del nivel interno de la base de datos. Por este motivo estas sentencias serán utilizadas normalmente por el administrador de la base de datos.

- El DCL (Data Control Language) se compone de instrucciones que permiten:

  - Ejercer un control sobre los datos tal como la asignación de privilegios de acceso a los datos (GRANT/REVOKE).
  - La gestión de transacciones (COMMIT/ROLLBACK).

  Una transacción se puede definir como un conjunto de acciones que se tienen que realizar todas o ninguna para preservar la integridad de la base de datos.

  Por ejemplo supongamos que tenemos una base de datos para las reservas de avión. Cuando un usuario pide reservar una plaza en un determinado vuelo, el sistema tiene que comprobar que queden plazas libres, si quedan plazas reservará la que quiera el usuario generando un nuevo billete y marcando la plaza como ocupada. Aquí tenemos un proceso que consta de dos operaciones de actualización de la base de datos (crear una nueva fila en la tabla de billetes y actualizar la plaza reservada en el vuelo, poniéndola como ocupada) estas dos operaciones se tienen que ejecutar o todas o ninguna, si después de crear el billete no se actualiza la plaza porque se cae el sistema, por ejemplo, la base de datos quedaría en un estado inconsistente ya

que la plaza constaría como libre cuando realmente habría un billete emitido para esta plaza. En este caso el sistema tiene el mecanismo de transacciones para evitar este error. Las operaciones se incluyen las dos en una misma transacción y así el sistema sabe que las tiene que ejecutar las dos, si por lo que sea no se pueden ejecutar las dos, se encarga de deshacer los cambios que se hubiesen producido para no ejecutar ninguna.

Las instrucciones que gestionan las autorizaciones serán utilizadas normalmente por el administrador mientras que las otras, referentes a proceso de transacciones serán utilizadas también por los programadores.

No todos los sistemas disponen de ellas.

- El DML se compone de las instrucciones para el manejo de los datos, para insertar nuevos datos, modificar datos existentes, para eliminar datos y la más utilizada, para recuperar datos de la base de datos. Veremos que una sola instrucción de recuperación de datos es tan potente que permite recuperar datos de varias tablas a la vez, realizar cálculos sobre estos datos y obtener resúmenes.

  El DML interactúa con el nivel externo de la base de datos por lo que sus instrucciones son muy parecidas, por no decir casi idénticas, de un sistema a otro, el usuario sólo indica lo que quiere recuperar no cómo se tiene que recuperar, no influye el cómo están almacenados los datos.
  Es el lenguaje que utilizan los programadores y los usuarios de la base de datos.

  A lo largo del curso se explicarán cada una de las formas de explotación de la base de datos. Dependiendo de tu perfil profesional (programador o administrador) o de tu interés personal te resultará más útil un bloque u otro.

INTRODUCCIÓN AL TRANSACT-SQL

Como hemos dicho, el sistema gestor de base de datos SQL-Server 2005 utiliza su propia versión del lenguaje SQL, el TRANSACT-SQL.

TRANSACT-SQL es un lenguaje muy potente que nos permite definir casi cualquier tarea que queramos efectuar sobre la base de datos. En este tema veremos que TRANSACT-SQL va más allá de un lenguaje SQL cualquiera ya que incluye características propias de cualquier lenguaje de programación, características que nos permiten definir la lógica necesaria para el tratamiento de la información:

- Tipos de datos.
- Definición de variables.
- Estructuras de control de flujo.
- Gestión de excepciones.
- Funciones predefinidas.

Sin embargo no permite:

- Crear interfaces de usuario.
- Crear aplicaciones ejecutables, sino elementos que en algún momento llegarán al servidor de datos y serán ejecutados.

Debido a estas restricciones se emplea generalmente para crear procedimientos almacenados, triggers y funciones de usuario.

Puede ser utilizado como cualquier SQL como lenguaje embebido en aplicaciones desarrolladas en otros lenguajes de programación como Visual Basic, C, Java, etc. Y por supuesto los lenguajes incluidos en la plataforma .NET.

También lo podremos ejecutar directamente de manera interactiva, por ejemplo desde el editor de consultas de SSMS (SQL Server Management Studio) el entorno de gestión que ya conocemos. Esta es la forma en que lo utilizaremos nosotros.

CARACTERISTICAS GENERALES DEL TRANSACT-SQL

El lenguaje SQL se creó con la finalidad de ser un lenguaje muy potente y a la vez muy fácil de utilizar, se ha conseguido en gran medida ya que con una sola frase (instrucción) podemos recuperar datos complejos (por ejemplo datos que se encuentran en varias tablas, combinándolos, calculando resúmenes), y utilizando un lenguaje muy cercano al lenguaje hablado (¡suponiendo que hablamos inglés, claro!).

Por ejemplo:

SELECT codigo, nombre FROM Clientes WHERE localidad='Valencia';

Esta instrucción nos permite SELECCIONAR el código y nombre DE los Clientes CUYA localidad sea Valencia.

La sencillez también radica en que lo que indicamos es lo que queremos obtener, no el cómo lo tenemos que obtener, de eso se encargará el sistema automáticamente.

Las sentencias SQL además siguen todas el mismo patrón:

- Empiezan por un verbo que indica la acción a realizar,
- completado por el objeto sobre el cual queremos realizar la acción,
- seguido de una serie de cláusulas (unas obligatorias, otras opcionales) que completan la frase, y proporcionan más detalles acerca de lo que se quiere hacer.

Si sabemos algo de inglés nos será más fácil interpretar a la primera lo que quiere decir la instrucción, y de lo contrario, como el número de palabras que se emplean es muy reducido, enseguida nos las aprenderemos.

- Por ejemplo en el DDL (acciones sobre la definición de la base de datos), tenemos 3 verbos básicos:

  CREATE (Crear)
  DROP (Eliminar)
  ALTER (Modificar)

  Completados por el tipo de objeto sobre el que actúan y el objeto concreto:

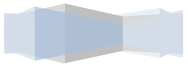
  CREATE DATABASE mibase .....;

  Permite crear una base de datos llamada mibase, a continuación escribiremos las demás cláusulas que completarán la acción, en este caso dónde se almacenará la base de datos, cuánto ocupará, etc...

  CREATE TABLE mitabla (.....);

  Permite crear una nueva tabla llamada mitabla, entre paréntesis completaremos la acción indicando la definición de las columnas de la tabla.

  CREATE INDEX miindex...;



Lo mismo para crear un índice (¿a qué lo habíais adivinado?).

DROP DATABASE mibase;

Permite borrar, eliminar la base de datos mibase.

DROP TABLE mitabla;

Elimina la tabla mitabla.

ALTER TABLE mitabla.....;

Permite modificar la definición de la tabla mitabla.

- En el DML (acciones sobre los datos almacenados) utilizaremos los verbos:

  INSERT (Crear, es decir, insertar una nueva fila de datos)
  DELETE (Eliminar filas de datos)
  UPDATE (Modificar filas de datos)
  SELECT (Seleccionar, obtener)

  Por ejemplo:

  INSERT INTO mitabla ..... Inserta nuevas filas en mitabla

  DELETE FROM mitabla Eliminar filas de mitabla

  UPDATE mitabla ..... Actualiza filas de mitabla

  Como ejemplo de cláusula dentro de una instrucción tenemos:

  ```
  SELECT codigo, nombre
  FROM Clientes
  WHERE localidad='Valencia';
  ```

  En esta sentencia nos aparecen dos cláusulas, la cláusula FROM que nos permite indicar de dónde hay que coger los datos y la cláusula WHERE que permite indicar una condición de selección.

- Otra característica de una sentencia SQL es que acaba con un punto y coma (;) originalmente éste era obligatorio y servía para indicar el fin de la instrucción, pero ahora se puede omitir, aunque se recomienda su uso.

- En una sentencia utilizaremos palabras reservadas (las fijas del lenguaje), y nombres de objetos y variables (identificadores).

  Las palabras reservadas no se pueden utilizar para otro propósito, por ejemplo una tabla no se puede llamar FROM, y los nombres (los identificadores) siguen las reglas detalladas en el punto siguiente.

- Nombres cualificados. En ocasiones deberemos utilizar nombres cualificados, por ejemplo cuando se escribe un nombre de tabla, SQL presupone que se está refiriendo a una de las tablas de la base de datos activa, si queremos hacer referencia a una tabla de otra base de datos utilizamos su nombre cualificado *nombrebasedatos.nombredeesquema.nombretabla*, utilizamos el punto para separar el nombre del objeto y el nombre de su contenedor.

O por ejemplo si en una consulta cuyo origen son dos tablas, queremos hacer referencia a un campo y ese nombre de campo es un nombre de campo en las dos tablas, pues utilizaremos su nombre cualificado *nombretabla.nombrecampo*.

- El valor NULL.

  Puesto que una base de datos es un modelo de una situación del mundo real, ciertos datos pueden inevitablemente faltar, ser desconocidos o no ser aplicables, esto se debe de indicar de alguna manera especial para no confundirlo con un valor conocido pero que sea cero por ejemplo, SQL tiene para tal efecto el valor NULL que indica precisamente la ausencia de valor.

  Por ejemplo: no es lo mismo que el alumno no tenga nota a que tenga la nota cero, esto afectaría también a todos los cálculos que se pueden realizar sobre la columna nota.

REGLAS DE FORMATO DE LOS IDENTIFICADORES

Los identificadores son los nombres de los objetos de la base de datos. Cualquier elemento de Microsoft SQL Server 2005 puede tener un identificador: servidores, bases de datos, tablas, vistas, columnas, índices, desencadenadores, procedimientos, restricciones, reglas, etc.

Las reglas de formato de los identificadores normales dependen del nivel de compatibilidad de la base de datos, que se establecía con el parámetro sp_dbcmptlevel pero que ahora Microsoft aconseja no utilizar ya que desaparecerá en versiones posteriores en vez de eso se tiene que utilizar la cláusula SET COMPATIBILITY_LEVEL de la instrucción ALTER TABLE. Cuando el nivel de compatibilidad es **90**, (el asignado por defecto) se aplican las reglas siguientes para los nombres de los identificadores:

- No puede ser una palabra reservada.
- El nombre debe tener entre 1 y 128 caracteres, excepto para algunos tipos de objetos en los que el número es más limitado.
- El nombre debe empezar por:
  - Una letra, como aparece definida por el estándar Unicode 3.2. La definición Unicode de letras incluye los caracteres latinos de la "a" a la "z" y de la "A" a la "Z".
  - El carácter de subrayado (_), arroba (@) o número (#).
- Ciertos símbolos al principio de un identificador tienen un significado especial en SQL Server. Un identificador que empieza con el signo de arroba indica un parámetro o una variable local. Un identificador que empieza con el signo de número indica una tabla o procedimiento temporal. Un identificador que empieza con un signo de número doble (##) indica un objeto temporal global.
- Algunas funciones de Transact-SQL tienen nombres que empiezan con un doble signo de arroba (@@). Para evitar confusiones con estas funciones, se recomienda no utilizar nombres que empiecen con @@.
- No se permiten los caracteres especiales o los espacios incrustados.

Si queremos utilizar un nombre que no siga estas reglas, normalmente para poder incluir espacios en blanco, lo tenemos que escribir encerrado entre corchetes [ ] (también se pueden utilizar las comillas pero recomendamos utilizar los corchetes).

TIPOS DE DATOS

En SQL Server 2005, cada columna, expresión, variable y parámetro está asociado a un tipo de datos. Un tipo de datos, realmente define el conjunto de valores válidos para los campos definidos de ese tipo. Indica si el campo puede contener: datos numéricos, de caracteres, moneda, fecha y hora, etc.

SQL Server proporciona un conjunto de tipos de datos del sistema que define todos los tipos de datos

que pueden utilizarse. También podemos definir nuestros propios tipos de datos en Transact-SQL o Microsoft .NET Framework.

Los tipos de datos más utilizados son:

- Los numéricos: int, decimal, money
- Los de fecha y hora: datetime
- Y las cadenas de caracteres: varchar

LAS CONSTANTES

Una constante es un valor específico o un símbolo que representa un valor de dato específico. El formato de las constantes depende del tipo de datos del valor que representan. En este apartado veremos las más utilizadas.

- Las constantes numéricas se escriben mediante una cadena de números, con la consideración de que el separador decimal es un punto, no una coma, y que si se trata de un valor monetario deberemos incluir la moneda al inicio de la constante. Por ejemplo: 85.90 y €85.90, el primero sería un valor decimal y el segundo un valor money. De forma predeterminada, los valores serán positivos. Para indicar lo contrario escribimos el signo - al principio.
- Las constantes de fecha y hora van entre comillas simples y con un formato de fecha y hora adecuado. Por ejemplo: '03/10/90'.
- Y las constantes en cadenas de caracteres van entre comillas simples. Por ejemplo: 'Juan García López'.

Para indicar valores negativos y positivos añadimos el prefijo **+** o **-** según sea el valor positivo o negativo. Sin prefijo se entiende que el valor es positivo.

LAS EXPRESIONES

Una expresión es una combinación de símbolos y operadores que el motor de base de datos de SQL Server evalúa para obtener un único valor. Una expresión simple puede ser una sola constante, variable, columna o función escalar. Los operadores se pueden usar para combinar dos o más expresiones simples y formar una expresión compleja.

Dos expresiones pueden combinarse mediante un operador si ambas tienen tipos de datos admitidos por el operador y se cumple al menos una de estas condiciones:

- Las expresiones tienen el mismo tipo de datos.
- El tipo de datos de menor prioridad se puede convertir implícitamente al tipo de datos de mayor prioridad.
- La función CAST puede convertir explícitamente el tipo de datos con menor prioridad al tipo de datos con mayor prioridad o a un tipo de datos intermedio que pueda convertirse implícitamente al tipo de datos con la mayor prioridad.

Tipos de operadores:
- Operadores numéricos:

suma	+
resta	-
multiplicación	*
división	/
módulo (resto de una división)	%

- Operadores bit a bit: realizan manipulaciones de bits entre dos expresiones de cualquiera de los tipos de datos de la categoría del tipo de datos entero.

AND	&
OR	|
OR exclusivo	^

- Operadores de comparación:

Igual a	=
Mayor que	>
Menor que	<
Mayor o igual que	>=
Menor o igual que	<=
Distinto de	<>
No es igual a	!=
No menor que	!<
No mayor que	!>

- Operadores lógicos:
Aquí sólo los nombraremos ya que en el tema de consultas simples los veremos en detalle.

ALL	IN
AND	LIKE
ANY	NOT
BETWEEN	OR
EXISTS	SOME

- Operadores de cadenas:

Concatenación	+

Resultados de la expresión

- Si se combinan dos expresiones mediante operadores de comparación o lógicos, el tipo de datos resultante es booleano y el valor es uno de los siguientes: TRUE, FALSE o UNKNOWN.

- Cuando dos expresiones se combinan mediante operadores aritméticos, bit a bit o de cadena, el operador determina el tipo de datos resultante.

Las expresiones complejas formadas por varios símbolos y operadores se evalúan como un resultado formado por un solo valor. El tipo de datos, intercalación, precisión y valor de la expresión resultante se determina al combinar las expresiones componentes de dos en dos, hasta que se alcanza un resultado final. La prioridad de los operadores de la expresión define la secuencia en que se combinan las expresiones.

FUNCIONES

SQL Server 2005 proporciona numerosas funciones integradas y permite crear funciones definidas por el usuario.

Existen diferentes tipos de funciones:

- Funciones de conjuntos de filas: devuelven un objeto que se puede utilizar, en instrucciones Transact-SQL, en lugar de una referencia a una tabla.
- Funciones de agregado (también llamadas funciones de columna): Operan sobre una colección de valores y devuelven un solo valor de resumen. Por ejemplo, la función de suma sobre la columna importe para conocer el importe total: SUM(importe)
- Funciones de categoría: Devuelven un valor de categoría para cada fila de un conjunto de filas, por ejemplo devuelve el número de la fila, el ranking de la fila en una determinada ordenación, etc.
- Funciones escalares: Operan sobre un valor y después devuelven otro valor. Son las funciones que estamos acostumbrados a utilizar. Las funciones escalares se clasifican según el tipo de datos de sus operandos

LAS VARIABLES

En Transact-SQL podemos definir variables, que serán de un tipo de datos determinado, como tipos de datos podemos utilizar los propios de la base de datos SQL-SERVER, pero también podemos utilizar tipos propios del lenguaje que no pueden ser utilizados en DDL. El tipo Cursor y el tipo Table son dos de estos tipos.

Las variables se definen utilizando la instrucción DECLARE con el siguiente formato:

DECLARE @nbvariable tipo

El nombre de la variable debe empezar por el símbolo @, este símbolo hace que SQL interprete el nombre como un nombre de variable y no un nombre de objeto de la base de datos.

Por ejemplo: DECLARE @empleados INT

Con esto hemos definido la variable @empleados de tipo entero.

Para asignar un valor a una variable, la asignación se realiza con la palabra SELECT y el signo igual con el formato:

SELECT @nbvariable = valor

El valor puede ser cualquier valor constante, otro nombre de variable, una expresión válida o algo más potente, parte de una sentencia SELECT de SQL.

Por ejemplo:

```
SELECT @empleados = 0;
SELECT @empleados = @otra * 100;
SELECT @EMPLEADOS = COUNT(numemp) FROM empleados;
```

El valor almacenado en la variable se puede visualizar mediante la orden PRINT. o SELECT

PRINT @nbvariable o SELECT @nbvariable

El valor almacenado en la variable se visualizará en la pestaña de resultados. También se puede usar para escribir mensajes:

PRINT 'Este es el mensaje'

OTROS ELEMENTOS DEL LENGUAJE

Comentarios. Como en cualquier otro lenguaje de programación, debemos utilizar comentarios destinados a facilitar la legibilidad del código. En SQL se insertan comentarios con los signos:

/* */	Varias líneas	/* Esto es un comentario en varias líneas */
--	Una única línea	-- Esto es un comentario en una única línea.

USE. Cambia el contexto de la base de datos al de la base de datos especificada.
USE nbBaseDeDatos

Hace que la base de datos activa pase a ser la base de datos indicada en la instrucción, las consultas que se ejecuten a continuación se harán sobre tablas de esa base de datos si no se indica lo contrario. Es una instrucción útil para asegurarnos de que la consulta se ejecuta sobre la base de datos correcta.

GO

GO no es una instrucción Transact-SQL, sino un comando reconocido por las utilidades **sqlcmd** y **osql**, así como por el Editor de código de SQL Server Management Studio.

Las utilidades de SQL Server interpretan GO como una señal de que deben enviar el lote actual de instrucciones Transact-SQL a una instancia de SQL Server. El lote actual de instrucciones está formado por todas las instrucciones específicadas desde el último comando GO o desde el comienzo de la sesión o script si se trata del primer comando GO.

Por ejemplo si queremos crear una consulta para crear una base de datos y sus tablas, después del CREATE DATABASE…; tenemos que poner GO antes del primer CREATE TABLE para que el sistema efectúe la primera operación y la base de datos esté creada antes de ejecutar el primer CREATE TABLE.

BEGIN...END

Encierra un conjunto de instrucciones Transact-SQL de forma que estas instrucciones formen un bloque de instrucciones.