Clase 04. Sublenguaje SQL DDL

Concepto

Conocido como SQL por sus siglas en inglés (Structured Query Language), es un lenguaje de consultas estructuradas.

IBM desarrolló la versión original de SQL, originalmente denominado Sequel, como parte del proyecto System R a principios de 1970. El lenguaje Sequel ha evolucionado desde entonces y su nombre ha pasado a ser SQL (Structured Query Language, lenguaje estructurado de consultas). Hoy en día, numerosos productos son compatibles con el lenguaje SQL y se ha establecido como el lenguaje estándar para las bases de datos relacionales.

Se usa para describir conjuntos de datos que pueden ayudar a responder preguntas. La sintaxis SQL se basa en el idioma inglés y usa muchos de los mismos elementos que la sintaxis del lenguaje Visual Basic para desarrollo de aplicaciones.

Este lenguaje no solo sirve para obtener información de los datos almacenados, sino también para conocer la metadata de la base de datos. La metadata permite conocer el origen y la estructura de cada uno de los objetos que componen la base de datos, desde una columna en una tabla, hasta el contenido de un store procedure.

El lenguaje se implementa por medio de un sistema de gestión de base de datos. De estos existen muchas tecnologías en la actualidad, como por ejemplo SQL, Teradata, Oracle, etc.

Objetos de una base de datos

Una base de datos se compone por elementos, con los cuales los usuarios pueden operar, y de esta forma generar el almacenamiento, modificación y eliminación de los datos.

Base de datos

Cada instancia de servidor de base de datos, puede contener una o varias bases de datos. En una base de datos, hay uno o varios grupos de propiedad de objetos denominados esquemas. Dentro de cada esquema hay objetos de base de datos como tablas, vistas y procedimientos almacenados. Algunos objetos, como certificados y claves asimétricas, se encuentran en la base de datos, pero no dentro de un esquema.

Cuando los usuarios obtienen acceso a un servidor de base de dato, se identifican como un inicio de sesión. Cuando los usuarios obtienen acceso a una base de datos, se identifican como un usuario de base de datos. Un usuario de base de datos puede estar basado en un inicio de sesión. Si están habilitadas las bases de datos independientes, se puede crear un usuario de base de datos que no esté basado en un inicio de sesión.

A un usuario que tiene acceso a una base de datos se le puede conceder permiso para acceder a los objetos de la base de datos. Aunque los permisos se pueden conceder a usuarios individuales, se recomienda crear roles de base de datos, agregar usuarios de base de datos a los roles y, a continuación, conceder permiso de acceso a los roles. La concesión de permisos a roles en vez de a usuarios facilita la coherencia y la comprensión de los permisos a medida que el número de usuarios aumenta y cambia continuamente.

Sentencias

También denominadas comandos o cláusulas, son las palabras reservadas que componen el lenguaje para ejecutar acciones sobre la base de datos. Gracias a estas los usuarios pueden operar en las bases de datos, y se caracterizan porque al usarlas en la redacción, la fuente adquiere colores distintos al estándar.

Sub lenguaje DDL

Lenguaje de definición de datos, DDL por sus siglas en inglés. Este es el conjunto de sentencias que se encargan de la definición de la base de datos y sus objetos.

Los esquemas de las bases de datos se especifican mediante un conjunto de definiciones expresadas mediante un lenguaje especial denominado lenguaje de definición de datos (LDD). El LDD también se usa para especificar más propiedades de los datos.
La estructura de almacenamiento y los métodos de acceso usados por el sistema de bases de datos se especifican mediante un conjunto de instrucciones en un tipo especial de LDD denominado lenguaje de almacenamiento y definición de datos. Estas instrucciones definen los detalles de implementación de los esquemas de las bases de datos, que suelen ocultarse a los usuarios.

Los valores de los datos almacenados en la base de datos deben satisfacer ciertas restricciones de consistencia. Por ejemplo, supóngase que el saldo de una cuenta no debe caer por debajo de 10.000 pesos. El LDD proporciona facilidades para especificar tales restricciones. Los sistemas de bases de datos las comprueban cada vez que se modifica la base de datos.

En general, las restricciones pueden ser predicados (condiciones de filtrado) arbitrarios relativos a la base de datos. No obstante, los predicados arbitrarios pueden resultar costosos de comprobar. Por tanto, los sistemas de bases de datos se concentran en las restricciones de integridad que pueden comprobarse con una sobrecarga mínima:

1. Restricciones de dominio. Se debe asociar un dominio de valores posibles a cada atributo (por ejemplo, tipos enteros, tipos de carácter, tipos fecha/hora). La declaración de un atributo como parte de un dominio concreto actúa como restricción de los valores que puede adoptar. Las restricciones de dominio son la forma más elemental de restricción de integridad. El sistema las comprueba fácilmente siempre que se introduce un nuevo elemento de datos en la base de datos.
2. Integridad referencial. Hay casos en los que se desea asegurar que un valor que aparece en una relación para un conjunto de atributos dado aparece también para un determinado conjunto de atributos en otra relación (integridad referencial). Las modificaciones de la base de datos pueden causar violaciones de la integridad referencial. Cuando se viola una restricción de integridad, el procedimiento normal es rechazar la acción que ha causado esa violación.
3. Asertos. Un aserto es cualquier condición que la base de datos debe satisfacer siempre. Las restricciones de dominio y las restricciones de integridad referencial son formas especiales de asertos. No obstante, hay muchas restricciones que no pueden expresarse empleando únicamente esas formas especiales. Por ejemplo: “Cada préstamo tiene como mínimo un cliente tenedor de una cuenta con un saldo mínimo de 1.000.000 de pesos” debe expresarse en forma de aserto. Cuando se crea un aserto, el sistema comprueba su validez. Si el aserto es válido, cualquier modificación futura de la base de datos se permite únicamente si no hace que se viole ese aserto.
4. Autorización. Puede que se desee diferenciar entre los usuarios en cuanto al tipo de acceso que se les permite a diferentes valores de los datos de la base de datos. Estas diferenciaciones se expresan en términos de autorización, cuyas modalidades más frecuentes son: autorización de lectura, que permite la lectura pero no la modificación de los datos; autorización de inserción, que permite la inserción de datos nuevos, pero no la modificación de los datos ya existentes; autorización de actualización, que permite la modificación, pero no la eliminación, de los datos; y la autorización de eliminación, que permite la eliminación de datos. A cada usuario se le pueden asignar todos, ninguno o una combinación de estos tipos de autorización

El LDD, al igual que cualquier otro lenguaje de programación, obtiene como entrada algunas instrucciones y genera una salida. La salida del LDD se coloca en el diccionario de datos, que contiene metadatos, es decir, datos sobre datos. El diccionario de datos se considera un tipo especial de tabla, a la que sólo puede tener acceso y actualizar el propio sistema de bases de datos (no los usuarios normales). El sistema de bases de datos consulta el diccionario de datos antes de leer o modificar los datos reales.

Sentencias y sintaxis

CREATE

La cláusula create permite crear los objetos en la base de datos: tablas, vistas, funciones y stored procedures, y la misma base Para la creación de tablas, se requiere la definición de cada columna.

La sentencia CREATE DATABASE se utiliza para crear bases de datos.

Sintaxis CREATE DATABASE:

CREATE DATABASE nombreBaseDatos

Ejemplo CREATE DATABASE

CREATE DATABASE mibasededatos

Tabla

Las tablas son objetos de base de datos que contienen todos sus datos. En las tablas, los datos se organizan con arreglo a un formato de filas y columnas, similar al de una hoja de cálculo. Cada fila representa un registro único y cada columna un campo dentro del registro. Por ejemplo, en una tabla que contiene los datos de los empleados de una compañía puede haber una fila para cada empleado y distintas columnas en las que figuren detalles de los mismos, como el número de empleado, el nombre, la dirección, el puesto que ocupa y su número de teléfono particular.

El número de tablas de una base de datos se limita solo por el número de objetos admitidos en una base (2.147.483.647). Una tabla definida por el usuario estándar puede tener hasta 1.024 columnas (aunque esto puede cambiar con la tecnología del servidor). El número de filas de la tabla sólo está limitado por la capacidad de almacenamiento del servidor.

Puede asignar propiedades a la tabla y a cada columna de la tabla para controlar los datos admitidos y otras propiedades. Por ejemplo, puede crear restricciones en una columna para no permitir valores nulos o para proporcionar un valor predeterminado si no se especifica un valor, o puede asignar una restricción de clave en la tabla que exige la unicidad o definir una relación entre las tablas.

Tipos de datos de las columnas

Permiten establecer las reglas de contenido de cada uno de los campos en las tablas.
En SQL cada columna tiene un tipo de datos relacionado. Un tipo de datos es un atributo que especifica el dominio que el objeto puede contener: datos enteros, datos de caracteres, datos de moneda, datos de fecha y hora, cadenas binarias, etc.

El tipo de datos debe ser definido al momento de la creación de la tabla, sin un tipo de datos asignado, el SGBD no permite la creación de la tabla, ya que presenta error de sintaxis.

Los principales tipos de datos

Los tipos de datos más usados son los siguientes:

Tipo	SQL	Ejemplo
Enteros	**Int**	1000
Decimales	**Decimal (18,2)**	10,51
Texto	**Text (n)**	Argentina
Alfanumérico	**Varchar (n)**	L0332154
Fecha	**Datetime**	01/10/2019
Fecha y hora	**Timestamp**	01/10/2019 12:15:00

Sin embargo existe una amplia variedad de datos, y generalmente son usados en procesos avanzados de almacenamiento de datos, en los cuales la especificación de tipos de datos cambia de acuerdo a la fuente de datos.

Grupo	Tipo de dato	Intervalo	Almacenamiento
Numéricos exactos	bigint	De -2^{63} (-9.223.372.036.854.775.808) a $2^{63} - 1$ (9.223.372.036.854.775.807)	8 bytes
	int	De -2^{31} (-2.147.483.648) a $2^{31} - 1$ (2.147.483.647)	4 bytes
	smallint	De -2^{15} (-32.768) a $2^{15} - 1$ (32.767)	2 bytes
	tinyint	De 0 a 255	1 byte
	bit	Tipo de datos entero que puede aceptar los valores 1, 0 ó NULL	2 bytes

	decimal, numeric, decimal (p, s)	• *p* (precisión): el número total máximo de dígitos decimales que se puede almacenar, tanto a la izquierda como a la derecha del separador decimal. La precisión debe ser un valor comprendido entre 1 y la precisión máxima de 38. La precisión predeterminada es 18. • *s* (escala): el número máximo de dígitos decimales que se puede almacenar a la derecha del separador decimal. La escala debe ser un valor comprendido entre 0 y *p*. Sólo es posible especificar la escala si se ha especificado la precisión. La escala predeterminada es 0. Con precisión máxima 10^{38} +1 y 10^{38} - 1	Precisión 1 - 9: 5 bytes
	money	Tipos de datos que representan valores monetarios o de moneda: de -922.337.203.685,4775808 a 922.337.203.685,4775807	8 bytes
	smallmoney	De - 214,7483648 a 214,7483647	4 bytes
Numéricos aproximados	float	De - 1,79E+308 a -2,23E-308, 0 y de 2,23E-308 a 1,79E+308	Depende del valor de *n*
	real	De - 3,40E + 38 a -1,18E - 38, 0 y de 1,18E - 38 a 3,40E + 38	4 Bytes
Fecha y hora	datetime	Del 1 de enero de 1753 hasta el 31 de diciembre de 9999	
	smalldatetime	Del 1 de enero de 1900 hasta el 6 de junio de 2079	
Cadenas de caracteres	char (n)	Caracteres no Unicode de longitud fija, con una longitud de *n* bytes. *n* debe ser un valor entre 1 y 8.000	n bytes
	varchar (n)	Caracteres no Unicode de longitud variable. *n* indica que el tamaño de almacenamiento máximo es de 2^{31} - 1 bytes	n bytes (aprox.)
	text	En desuso, sustituido por *varchar*. Datos no Unicode de longitud variable con una longitud máxima de 2^{31} - 1 (2.147.483.647) caracteres	max bytes (aprox.)
Cadenas de caracteres unicode	nchar (n)	Datos de carácter Unicode de longitud fija, con *n* caracteres. *n* debe estar comprendido entre 1 y 4.000	2 * n bytes

	nvarchar (n)	Datos de carácter Unicode de longitud variable. *n* indica que el tamaño máximo de almacenamiento es 2^{31} - 1 bytes	2 * n bytes + 2 bytes
	ntext (n)	En desuso, sustituido por *nvarchar*. Datos Unicode de longitud variable con una longitud máxima de 2^{30} - 1 (1.073.741.823) caracteres	2 * n bytes
Cadenas binarias	binary (n)	Datos binarios de longitud fija con una longitud de *n* bytes, donde *n* es un valor que oscila entre 1 y 8.000	*n* bytes
	varbinary (n)	Datos binarios de longitud variable. n indica que el tamaño de almacenamiento máximo es de 2^{31} - 1 bytes	n bytes
	image	En desuso, sustituido por *varbinary*. Datos binarios de longitud variable desde 0 hasta 2^{31} - 1 (2.147.483.647) bytes	
Otros tipos de datos	cursor	Tipo de datos para las variables o para los parámetros de resultado de los procedimientos almacenados que contiene una referencia a un cursor. Las variables creadas con el tipo de datos *cursor* aceptan NULL	
	timestamp	Tipo de datos que expone números binarios únicos generados automáticamente en una base de datos. El tipo de datos *timestamp* es simplemente un número que se incrementa y no conserva una fecha o una hora	8 bytes
	sql_variant	Tipo de datos que almacena valores de varios tipos de datos aceptados en SQL Server, excepto *text*, *ntext*, *image*, *timestamp* y *sql_variant*	
	uniqueidentifier	Es un GUID (Globally Unique Identifier, Identificador Único Global)	16 bytes
	table	Es un tipo de datos especial que se puede utilizar para almacenar un conjunto de resultados para su procesamiento posterior. *table* se utiliza principalmente para el almacenamiento temporal de un conjunto de filas devuelto como el conjunto de resultados de una función con valores de tabla	

	xml	Almacena datos de XML. Puede almacenar instancias de xml en una columna o una variable de tipo xml	

La sentencia CREATE TABLE se utiliza para crear una tabla en una base de datos existente.

Sintaxis CREATE TABLE

```
CREATE TABLE nombretabla
(
nombrecolumna1 tipodato1,
nombrecolumna2 tipodato2,
nombrecolumna3 tipodato3,
..
)
```

Ejemplo CREATE TABLE

```
CREATE TABLE personas
(
nombre varchar(255),
apellido1 varchar(255),
apellido2 varchar(255),
dep int
)
```

Esta sentencia creará la base de datos 'personas' con 4 columnas.

Las columnas 'nombre', 'apellido1' y 'apellido2' son de tipo 'varchar', es decir, acepta valores alfanuméricos hasta una longitud máxima de 255 caracteres.

La columna 'dep' es de tipo 'int', es decir, acepta solo números.

Existen diferentes tipos de datos, algunos son iguales en todas las bases de datos (MySQL, ORACLE, DB2, ..) y otros pueden ser particulares para ser usados únicamente en alguna de estas bases de datos.

Vistas

Una vista es una tabla virtual cuyo contenido está definido por una consulta. Al igual que una tabla, una vista consta de un conjunto de columnas y filas de datos con un nombre. Sin embargo, a menos que esté indizada, una vista no existe como conjunto de valores de datos almacenados en una base de datos. Las filas y las columnas de datos proceden de tablas a las que se hace referencia en la consulta que define la vista y se producen de forma dinámica cuando se hace referencia a la vista.

Una vista actúa como filtro de las tablas subyacentes a las que se hace referencia en ella. La consulta que define la vista puede provenir de una o de varias tablas, o bien de otras vistas de la base de datos actual u otras bases de datos. Asimismo, es posible utilizar las consultas distribuidas para definir vistas que utilicen datos de orígenes heterogéneos. Esto puede resultar de utilidad, por ejemplo, si desea combinar datos de estructura similar que proceden de distintos servidores, cada uno de los cuales almacena los datos para una región distinta de la organización.

Las vistas suelen usarse para centrar, simplificar y personalizar la percepción de la base de datos para cada usuario. Las vistas pueden emplearse como mecanismos de seguridad, que permiten a los usuarios obtener acceso a los datos por medio de la vista, pero no les conceden el permiso de obtener acceso directo a las tablas base subyacentes de la vista.

La sentencia CREATE VIEW se utiliza para crear una vista.

Sintaxis CREATE VIEW

```
CREATE VIEW nombrevista AS
SELECT nombrecolumna(s)
FROM nombretabla
WHERE condición
```

Ejemplo CREATE VIEW

```
CREATE VIEW personas_que_se_llaman_ANTONIO AS
SELECT nombre, apellido1, apellido2,edad
FROM personas
WHERE nombre = 'ANTONIO'
```

Para consultar los datos de una vista

SELECT * FROM personas_que_se_llaman_ANTONIO

Esquema

El esquema es una base dentro de la base de datos, cuando se crea una base de datos, el SGBD de forma automática le asigna el nombre dbo al esquema principal.
El objetivo principal de los esquemas es poder establecer restricciones de integridad y seguridad de la información.

Un esquema de base de datos representa la configuración lógica de todo o parte de una base de datos relacional. Puede existir de dos formas: como representación visual y como un conjunto de fórmulas conocidas como restricciones de integridad que controlan una base de datos. Estas fórmulas se expresan en un lenguaje de definición de datos, tal como SQL. Como parte de un diccionario de datos, un esquema de base de datos indica cómo las entidades que conforman la base de datos se relacionan entre sí, incluidas las tablas, las vistas, los procedimientos almacenados y mucho mas

Típicamente, un diseñador de bases de datos crea un esquema de base de datos para ayudar a los programadores cuyo software interactúa con la base. Al proceso de crear un esquema de base de datos se le llama modelado de datos. Al seguir el enfoque de tres esquemas para el diseño de bases de datos, este paso seguiría la creación de un esquema conceptual. Los esquemas conceptuales se enfocan en las necesidades informativas de una organización, más que en la estructura de una base de datos.

Hay dos tipos principales de esquemas de bases de datos:

Un esquema lógico de base de datos expresa las restricciones lógicas que se aplican a los datos almacenados. Puede definir las restricciones de integridad, las vistas y las tablas.
Un esquema físico de base de datos dispone cómo se almacenan los datos físicamente en un sistema de almacenamiento en términos de archivos e índices.

En el nivel más básico, un esquema de base de datos indica qué tablas o relaciones componen la base de datos, así como los campos incluidos en cada tabla. Por lo tanto, los términos diagrama de esquema y diagrama de relaciones de entidades con frecuencia son intercambiables

Stored Procedure

Es un conjunto de comandos que pueden ser ejecutados directamente en el servidor, es decir, será ejecutado por el servidor de Base de Datos y no por el programa cliente que lo accede, permitiendo la ejecución de una acción o conjunto de acciones específicas y que puede o no devolver un parámetro de salida.

Los procedimientos se asemejan a las construcciones de otros lenguajes de programación, porque pueden:

Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al programa que realiza la llamada.

Contener instrucciones de programación que realicen operaciones en la base de datos. Entre otras, pueden contener llamadas a otros procedimientos.

Devolver un valor de estado a un programa que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores, y el motivo de estos.

ALTER

La cláusula alter permite modificar la estructura de una tabla u objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de datos un campo, modificar la clave primaria, etc.

Sintaxis SQL ALTER

Para añadir una nueva columna a una tabla

ALTER TABLE nombretabla
ADD nombrecolumna tipodatocolumna

Para borrar una columna de una tabla

ALTER TABLE nombretabla
DROP COLUMN nombrecolumna

Para modificar el tipo de dato de una columna de una tabla

ALTER TABLE nombretabla
ALTER COLUMN nombrecolumna tipodatocolumna

Ejemplos de SQL ALTER

per	nombre	apellido1	apellido2
1	ANTONIO	PEREZ	GOMEZ
2	ANTONIO	GARCIA	RODRIGUEZ
3	PEDRO	RUIZ	GONZALEZ

Dada la siguiente tabla de 'personas', queremos añadir una nueva columna, denominada 'fechadenacimiento'

ALTER TABLE personas

ADD fechadenacimiento date

per	nombre	apellido1	apellido2	fechadenacimiento
1	ANTONIO	PEREZ	GOMEZ	
2	ANTONIO	GARCIA	RODRIGUEZ	
3	PEDRO	RUIZ	GONZALEZ	

Si queremos borrar la columna 'fechadenacimiento', y dejarlo igual que al principio

ALTER TABLE personas

DROP COLUMN fechadenacimiento

per	nombre	apellido1	apellido2
1	ANTONIO	PEREZ	GOMEZ
2	ANTONIO	GARCIA	RODRIGUEZ
3	PEDRO	RUIZ	GONZALEZ

DROP

La sentencia DROP se utiliza para borrar definitivamente un índice, tabla o base de datos.

DROP INDEX

Sintaxis DROP INDEX

DROP INDEX nombretabla.nombreindice

DROP TABLE

Se utiliza DROP TABLE para borrar definitivamente una tabla

DROP TABLE nombretabla

DROP DATABASE

Se utiliza para borrar una base de datos definitivamente

DROP DATABASE nombrebasededatos