Mario Martínez

Bases de datos y tablas

Creación con SQL

100499. Cuando concluyas la lectura de este documento sabrás como:

- Crear bases de datos.
- Crear tablas y sus campos.

¿De dónde vienen las bases de datos con que la mayoría de los usuarios tienen que trabajar?. Este documento explica esto desde la perspectiva del SQL. Te explica como empieza la existencia de una base de datos. Para este propósito debes conocer las sentencias de CREATE.

create database

La mayoría de los sistemas SQL actuales incluyen un menú, o un botón para crear bases de datos. Muestran un cuadro de dialogo para que el usuario escriba el nombre de la nueva base de datos, el nombre de usuario y sus password. Evidentemente, quien quiera crear una nueva base de datos debe tener los privilegios para hacerlo. eso es cuestión del administrador del sistema, él es quien gestiona los privilegios de los usuarios.

Cuando el sistema SQL no tiene una interfaz visual para crear bases de datos, puede usarse la sentencia CREATE DATABASE, que tiene la siguiente sintaxis:

```
CREATE DATABASE nombre_de_la_base_de_datos;
```

Por ejemplo:

```
CREATE DATABASE Pagos;
```

Existen algunas consideraciones a tomarse cuando se crea una base de datos.

Primero, no todos los sistemas dan soporte a esta sentencia de la misma forma, algunos ni siquiera la toman en cuenta y devuelven un mensaje de error. es muy probable que aquellos sistemas que no acepten esta sentencia tengan una forma más visual de construir una base de datos, por ejemplo: InterBase 4.2.

Segundo, muchos sistemas de gestión de bases de datos relacionales permiten especificar un tamaño de base de datos en términos de espacio en disco duro, como los megabytes. Necesitas entender cómo es que tu sistema de bases de datos almacena datos en disco para estimar un tamaño preciso necesario para tu nueva base de datos.

Algunos sistemas gestionan el espacio sin hacer muchas preguntas y te apartan de la necesidad de hacer estas estimaciones.

Tercero, debes diseñar apropiadamente tu base de datos. Antes que nada, tus conocimientos teóricos sobre bases de datos y sobre normalización de bases de datos tienen que estar bien asentados. De no ser así, seguramente tu base de datos fracasara en su desempeño y estructura.

Nota bibliográfica: Un buen libro para novicios es "FUNDAMENTOS DE BASES DE DATOS", tercera edición, de Abraham Silberschatz, Henry F. Korth, S. Sudarshan, editado en español por Mc

Graw Hill.

De cualquier modo, vamos a dar una revisión de aspectos a tomar en cuenta cuando se diseña una base de datos:

1. Seguridad.

2. Espacio disponible en disco.

3. Velocidad en búsquedas.

4. Velocidad en actualización de datos.

5. Velocidad en carga de tablas vinculadas.

6. Soporte del sistema de gestión de bases de datos relacionales para tablas temporales.

La normalización de una base de datos consiste en el seccionamiento de información en componentes con la finalidad de evitar la repetición innecesaria de datos. Cada nivel de normalización reduce la repetición de datos. Normalizar la información es un proceso complejo que bien logrado brinda beneficios importantes al desempeño de consultas de datos y al aprovechamiento del espacio disponible en disco.

Una herramienta útil en el diseño de bases de datos es el Diccionario de Datos.

Diccionario de Datos

Un Diccionario de Datos es una forma de documentación para el diseñador de bases de datos. Su utilidad básica se describe en las siguientes funciones:

1. Describir el propósito de la BD y quienes serán sus usuarios.

2. Documentar las especificaciones detrás de la BD misma: en qué dispositivo estará almacenada, cuál será el tamaño estándar de la BD junto con sus archivos lógicos (aquellos que almacenan información sobre operaciones en algunos sistemas de gestión de BD).

3. Almacenar código fuente de SQL referente a la instalación y desinstalación de la BD, incluida también documentación respecto al uso de herramientas de importar/exportar.

4. Proveer una detallada descripción de cada tabla dentro de la BD y explicar su propósito en términos de procesos de negocios.

5. Documentar la estructura interna de cada tabla, incluyendo todos sus campos y sus tipos de datos con comentarios, todos los índices y todas las vistas.

6. Contener todo el código fuente SQL para todos los procedimientos y triggers.

7. Describir reglas como pueden ser el uso de valores no nulos, valores únicos.

Algunos programas CASE auxilian al diseñador de BD en la creación y mantenimiento de un Diccionario de Datos.

Campos clave

Vamos a revisar algunas definiciones de campos clave.

Clave primaria

1. Cada registro es único dentro de una tabla (ningún otro registro dentro de la tabla tiene todos sus campos dentro de la tabla iguales a los de cualquier otro registro).

2. Para que cada registro sea único, por lo menos un campo no debe repetirse en ningún otro registro. Basándonos en este ultimo punto, el campo único se denomina campo de clave primaria.

Clave foránea

Es un campo creado con la finalidad de vincular o relacionar tablas. Esto se logra usando valores idénticos en la clave primaria de una tabla y los campos foráneos de otras.

CONTRATOS	CUENTAS_BANCARIAS	EMPRESAS
NUMERO_CONTRATO, NUMERIC	CUENTA_ID, NUMERIC	NOMBRE, CHAR(30)
IMPORTE, NUMERIC	TIPO, CHAR(30)	DOMICILIO, CHAR(50)
CUENTA_ID, NUMERIC	BALANCE, NUMERIC	CIUDAD, CHAR(20)
CLAVE_CLIENTE, CHAR(6)	BANCO, CHAR(30)	ESTADO, CHAR(3)
		CLAVE_CLIENTE, CHAR(6)

¿Cuáles campos crees que sean claves primarias y cuáles claves foráneas?.

La clave primaria en EMPRESAS es CLAVE_CLIENTE y tiene relación con el campo del mismo nombre en la tabla CONTRATOS.

La clave primaria en CONTRATOS es CUENTA_ID y tiene relación con el campo del mismo nombre en la tabla CUENTAS_BANCARIAS.

Ahora mira este ejemplo de tabla:

CAMPOS	DESCRIPCIÓN
NOMBRE, CHAR(30)	Nombre de la empresa
IMPORTE, NUMERIC	Importe del contrato en moneda nacional
CUENTA_ID, NUMERIC	Cuenta bancaria
DOMICILIO, CHAR(30)	Domicilio de la empresa
CIUDAD, CHAR(15)	Ciudad de la empresa
ESTADO, CHAR(3)	Estado de la empresa abreviado

Aunque pueda parecer una tabla correctamente planeada considera esto: cada vez que se dé de alta un nuevo contrato para la misma empresa se tendrán que duplicar los valores de DOMICILIO, CIUDAD Y ESTADO. Sí multiplicamos esta duplicación por cientos de miles de registros de empresas que abren más de un contrato y después volvemos a multiplicar por 10 o 20 o 30 tablas planeadas de la misma forma podremos sopesar la importancia de un diseño de BD bien normalizado.

Creación de tablas

Cuando sea el caso de crear físicamente nuevas tablas dentro de la base de datos se debe utilizar la sentencia CREATE TABLE que para el propósito tiene la siguiente sintaxis:

```
CREATE TABLE nombre_de_la_tabla (
campo1 tipo_de_dato [ NOT NULL ],
campo2 tipo_de_dato [ NOT NULL ],
campo3 tipo_de_dato [ NOT NULL ]... )
```

Un ejemplo:

```
SQL> CREATE TABLE CONTRATOS (
2 NOMBRE CHAR(30),
3 IMPORTE NUMERIC,
```

```
4 CUENTA_ID NUMERIC);
```

Esta sentencia construye una tabla denominada CONTRATOS, con un campo NOMBRE que tiene capacidad para 30 caracteres alfanuméricos, y los campos IMPORTE y CUENTA_ID que únicamente almacenan valores numéricos.

Nombres de tablas

Respecto al "bautizo" de tablas, existen ciertas restricciones. Los nombres deben empezar con un caracter entre A y Z, la longitud maxima de caracteres para el nombre no debe exceder cierto número, por ejemplo, Personal Oracle7 admite hasta 30 caracteres, (revisa la ayuda de tu sistema en este sentido).

Además, la mayoría de los sistemas hacen distinción entre mayúsculas y minúsculas por lo que que es un buen consejo convenir en un modo fijo de nominación basado solo en mayúsculas o solo en minúsculas dentro de una base de datos.

Evita usar como nombres palabras que tu sistema SQL tenga reservadas, tales como ALTER y CREATE.

No debes intentar crear tablas con el mismo nombre dentro de la misma base de datos.

Campos y tipos

Los campos tambien tienen sus convenios. Puedes tener nombres de campos duplicados, triplicados y más dentro de la misma base de datos, pero no en la misma tabla, por ejemplo, puedes tener 10 tablas en tu base de datos y cada una puede contener un campo denominado ID.

Existen tipos de campos para almacenar ciertos tipos de datos:

char(n)

es una cadena de caracteres de longitud fija, con una longitud n especificada por el usuario. También se puede usar la palabra completa character.

varchar(n)

es una cadena de caracteres de longitud variable, con una longitud n especificada por el usuario. También se puede utilizar la palabra completa character varying.

int

es un entero, se puede utilizar la palabra completa integer.

smallint

es un entero pequeño.

numeric(p, d)

es un número con precisión decimal, esta precisión la determina el usuario. El número está formado por p dígitos (más el signo), y de esos p dígitos, d pertenecen a la parte decimal. Por ejemplo, numeric(3, 1) permite que el número 55.4 se almacene exactamente, mientras que los números 555.4 y 0.32 no se pueden almacenar exactamente en un campo como éste.

real, double precision

son, respectivamente, números de punto flotante (coma en Europa) de doble precisión, con precisión dependiente de la máquina.

float(n)

es un número en coma flotante, cuya precisión la especifica el usuario y es de al menos n dígitos.

date

es una fecha del calendario, que contiene un año (de cuatro dígitos), un mes y un día del mes.

time

es la hora del día, expresada en horas, minutos y segundos.

Las cadenas caracteres de longitud variable, la fecha y la hora no forman parte de la norma SQL. Encontraremos que no todos los