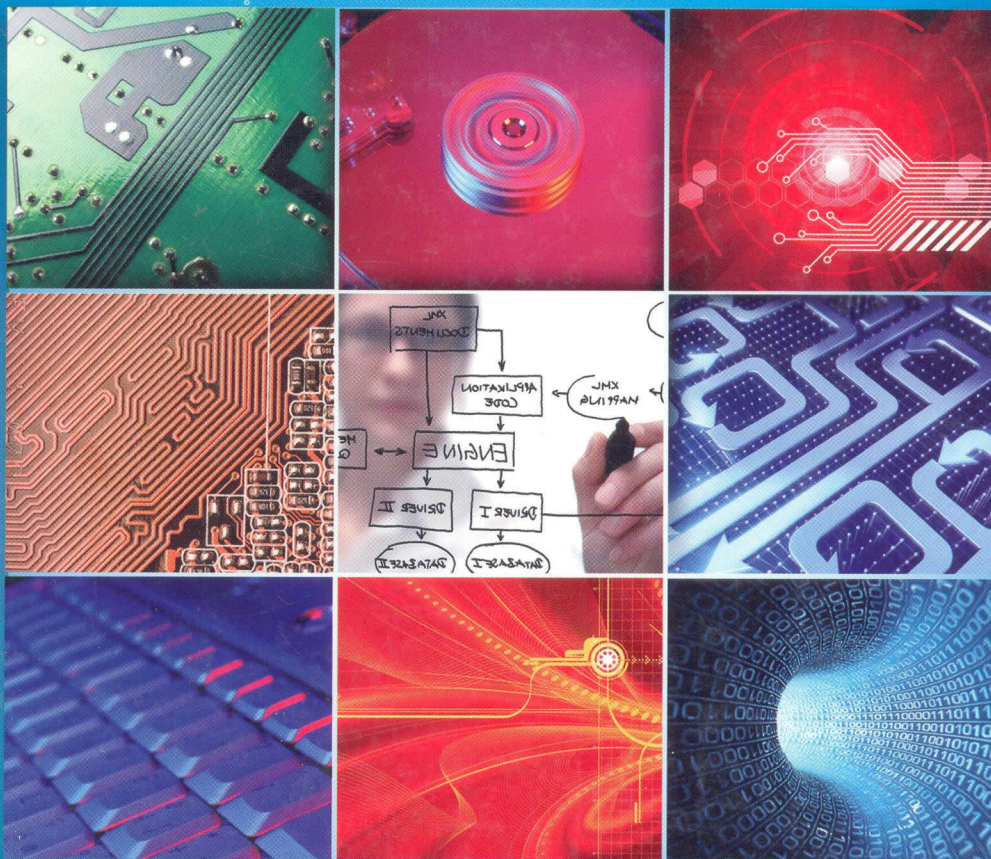


CORONEL / MORRIS / ROB

BASES DE DATOS

Diseño, implementación
y administración



NOVENA EDICIÓN

- Los valores V_CODE existentes en la tabla PRODUCT deben tener (y tienen) un valor correspondiente en la tabla VENDOR para asegurar integridad referencial.
- Algunos productos suministrados son directos de fábrica, otros se hacen en casa y el resto pueden haber sido adquiridos en una venta de almacén. En otras palabras, un producto no es necesariamente suministrado por un vendedor. Por tanto, VENDOR es opcional a PRODUCT.

Pocas de las condiciones que acabamos de describir se hicieron para ilustrar funciones específicas de SQL. Por ejemplo, los valores nulos en V_CODE se usaron en la tabla PRODUCT para ilustrar (más adelante) cómo se les puede dar seguimiento con SQL.

7.2.2 CREACIÓN DE LA BASE DE DATOS

Antes que pueda usar un nuevo RDBMS, usted debe completar dos tareas: primero, crear la estructura de base de datos y, segundo, crear las tablas que contendrán los datos de usuario final. Para completar la primera tarea, el RDBMS crea los archivos físicos que contendrán la base de datos. Cuando usted cree una nueva base de datos, el RDBMS automáticamente crea las tablas de datos del diccionario en las que se guardarán los metadatos y crea un administrador predeterminado de base de datos. Crear los archivos físicos que contendrán la base de datos significa interactuar con el sistema operativo y los sistemas de archivo soportados por éste. Por tanto, crear la estructura de la base de datos es la función que tiende a distinguir sustancialmente un RDBMS respecto de otro. La buena noticia es que es relativamente fácil crear una estructura de base de datos, cualquiera que sea el RDBMS que se use.

Si usted emplea Microsoft Access, crear la base de datos es sencillo: inicie Access, seleccione *File* → *New* → *Blank Database*, especifique la carpeta en la que desea guardar la base de datos y después aplique nombre a ésta. No obstante, si trabaja en un ambiente de base de datos que por lo general se usa en organizaciones más grandes, es probable que use un RDBMS empresarial como Oracle, SQL Server, MySQL o DB2. Dados sus requisitos de seguridad y mayor complejidad, esos productos requieren un proceso más elaborado para la creación de una base de datos. (Véase en el Apéndice N una ilustración de instrucciones específicas para crear una estructura de base de datos en Oracle.)

Sentirá alivio al descubrir que, *salvo el proceso de creación de la base de datos*, la mayoría de los vendedores de RDBMS usan SQL que se desvía poco del SQL estándar del ANSI. Por ejemplo, casi todos los RDBMS requieren que cada uno de los comandos de SQL termine con un punto y coma, pero algunas implementaciones no usan ese signo. En los recuadros de Nota se destacan aquí importantes diferencias de sintaxis entre implementaciones.

Si usted está usando un RDBMS empresarial, antes de empezar a crear tablas debe ser autenticado por el RDBMS. La **autenticación** es el proceso mediante el cual el DBMS verifica que sólo usuarios registrados pueden tener acceso a la base de datos. Para estar autenticado, es necesario registrarse en el RDBMS usando una ID de usuario y una contraseña creada por el administrador de la base de datos. En un RDBMS empresarial, la ID de todo usuario está asociada con un esquema de base de datos.

7.2.3 EL ESQUEMA DE BASE DE DATOS

En el ambiente de SQL, un **esquema** es un grupo de objetos de base de datos, por ejemplo tablas e índices, que están relacionados entre sí. Por lo general, el esquema pertenece a un solo usuario o aplicación. Una sola base de datos puede contener múltiples esquemas que pertenecen a diferentes usuarios o aplicaciones. Considere un esquema como agrupamiento lógico de objetos de bases de datos, por ejemplo tablas, índices y vistas. Los esquemas son útiles en el sentido que agrupan tablas por propietario (o función) y aplican un primer nivel de seguridad al permitir que cada usuario vea sólo las tablas que le pertenecen.

Las normas del ANSI para SQL definen un comando para crear un esquema de base de datos:

```
CREATE SCHEMA AUTHORIZATION {creador};
```

Por tanto, si el creador es JONES, use el comando:

```
CREATE SCHEMA AUTHORIZATION JONES;
```

Casi todos los RDBMS empresariales soportan ese comando, pero éste se usa raras veces directamente, es decir, de la línea de comando. (Cuando se crea un usuario, el DBMS automáticamente le asigna un esquema.) Cuando el DBMS se usa, el comando CREATE SCHEMA AUTHORIZATION debe ser emitido por el usuario que posee el esquema. Esto es, si usted se registra como JONES, sólo puede usar CREATE SCHEMA AUTHORIZATION JONES.

Para casi todos los RDBMS, CREATE SCHEMA AUTHORIZATION es opcional. Esa es la razón por la que este capítulo se concentra en los comandos del ANSI para SQL requeridos para crear y manipular tablas.

7.2.4 TIPOS DE DATOS

En el diccionario de datos de la tabla 7.3, observe particularmente los tipos seleccionados de datos. Recuerde que la selección del tipo de datos está por lo general dictada por la naturaleza de los datos y por el uso que se pretende. Por ejemplo:

- P_PRICE claramente requiere alguna clase de tipo numérico de datos; definirlo como campo de caracteres no es aceptable.
- Con igual claridad, el nombre de un vendedor es un candidato obvio para un tipo de datos de caracteres. Por ejemplo, VARCHAR2(35) se ajusta bien porque los nombres de vendedores son filas de caracteres de "longitud variable" y, en este caso, esas filas pueden ser de hasta 35 caracteres de largo.
- A primera vista, pudiera parecer lógico seleccionar un tipo numérico de datos para V_AREACODE porque contiene sólo dígitos, pero agregar y restar códigos postales no da resultados significativos. Por tanto, seleccionar un tipo de datos de caracteres es más apropiado. Esto es cierto para muchos atributos comunes que se encuentran en modelos de datos de negocios. Por ejemplo, aun cuando los códigos postales contienen todos los dígitos, deben estar definidos como datos de caracteres porque algunos códigos postales empiezan con un dígito cero (0) y un tipo numérico de datos causaría que se cancelara el cero inicial.
- Las abreviaturas de cada estado en Estados Unidos son siempre de dos caracteres, de modo que CHAR(2) es una selección lógica.
- Seleccionar P_INDATE para ser un campo DATE (juliano), en lugar de campo de caracteres, es deseable porque las fechas julianas permiten hacer comparaciones sencillas de fechas y efectuar aritmética de fechas. Por ejemplo, si usted ha usado campos DATE, puede determinar cuántos días hay entre ellos.

Si usted usa campos DATE, también puede determinar cuál será la fecha, por ejemplo, en 60 días desde una P_INDATE determinada con sólo usar P_INDATE + 60. O puede usar la fecha del sistema del RDBMS, SYSDATE en Oracle, GETDATE() en MS SQL Server y Date() en Access, para determinar la respuesta a preguntas como ¿cuál será la fecha dentro de 60 días? Por ejemplo, podría usar SYSDATE + 60 (en Oracle), GETDATE() + 60 (en MS SQL Server), o Date() + 60 (en Access).

La aptitud aritmética de fechas es particularmente útil en facturaciones. Quizás usted desea que su sistema empiece a cargar intereses sobre el saldo de un cliente a partir de 60 días de generar la factura. Esa sencilla aritmética de fechas sería imposible si se usa tipo de datos de caracteres.

A veces la selección de tipo de datos requiere juicio profesional. Por ejemplo, usted debe tomar una decisión acerca del tipo de datos de V_CODE como sigue:

- Si desea que la computadora genera nuevos códigos de vendedor al agregar 1 al máximo código registrado de vendedor, debe clasificar V_CODE como atributo numérico. (No puede efectuar procedimientos matemáticos en datos de caracteres.) La designación INTEGER asegura que pueden usarse sólo los números naturales (enteros). Casi todas las implementaciones de SQL también permiten el uso de SMALLINT para valores enteros de hasta seis dígitos.
- Si usted no desea efectuar procedimientos matemáticos basados en V_CODE, debería clasificarlo como atributo de caracteres aun cuando esté compuesto totalmente de números. Los datos de caracteres son más "rápidos" para usarse en consultas. Por tanto, cuando no haya necesidad de efectuar procedimientos matemáticos en el atributo, guárdelo como atributo de caracteres.

La primera opción se usa para demostrar los procedimientos de SQL en este capítulo.

Diccionario de datos para la base de datos CH07_SALECO

TABLA 7.3

NOMBRE DE TABLA	NOMBRE DE ATRIBUTO	CONTENIDO	TIPO	FORMATO	INTERVALO*	SE REQUIERE	PK O FK	TABLA REFERIDA POR LA FK
PRODUCT	P_CODE	Código de producto	CHAR(10)	XXXXXXXXXX	NA	Y	PK	
	P_DESCRIPT	Descripción de producto	VARCHAR(35)	XXXXXXXXXXXX	NA	Y		
	P_INDATE	Fecha de abastecimiento	DATE	DD-MON-YYYY	NA	Y		
	P_QOH	Unidades disponibles	SMALLINT	####	0-9999	Y		
	P_MIN	Unidades mínimas	SMALLINT	####	0-9999	Y		
	P_PRICE	Precio de producto	NUMBER(8,2)	#####	0.00-9999.00	Y		
	P_DISCOUNT	Tasa de descuento	NUMBER(5,2)	0.##	0.00-0.20	Y		
	V_CODE	Código de vendedor	INTEGER	###	100-999		FK	VENDOR
VENDOR	V_CODE	Código de vendedor	INTEGER	###	1000-9999	Y	PK	
	V_NAME	Nombre de vendedor	CHAR(35)	XXXXXXXXXXXXXX	NA	Y		
	V_CONTACT	Contacto	CHAR(25)	XXXXXXXXXXXXXX	NA	Y		
	V_AREACODE	Código postal	CHAR(3)	999	NA	Y		
	V_PHONE	Número telefónico	CHAR(8)	999-9999	NA	Y		
	V_STATE	Estado	CHAR(2)	XX	NA	Y		
	V_ORDER	Pedido previo	CHAR(1)	X	Y or N	Y		

FK = Llave foránea

PK = Llave primaria

CHAR = Datos de longitud fija de 1 a 255 caracteres

VARCHAR = Datos de longitud variable 1 a 2 000 caracteres. VARCHAR se convierte automáticamente en VARCHAR2 en Oracle

NUMBER = Datos numéricos. NUMBER(9,2) se usa para especificar números con dos lugares decimales y hasta nueve dígitos de largo, incluidos los lugares decimales.

Algunos RDBMS permiten el uso de un tipo de datos MONEY o CURRENCY.

INT = Sólo valores enteros

SMALLINT = Sólo valores enteros pequeños

Los formatos DATE varían. Los comúnmente aceptados son: 'DD-MES-AAAA', 'DD-MES-AA', 'MM/DD/AAAA' y 'MM/DD/AA'

* No todos los intervalos (rangos) mostrados aquí se ilustran en este capítulo, pero se pueden usar estas restricciones para practicar la escritura de sus propias restricciones.

Cuando usted defina el tipo de datos de un atributo, debe poner mucha atención al uso esperado de éste para fines de orden y obtención de datos. Por ejemplo, en una aplicación de bienes raíces, un atributo que representa los números de cuartos de baño en una casa (H_BATH_NUM) podría ser asignado al tipo de datos CHAR(3) porque es poco probable que la aplicación haga alguna adición, multiplicación o división con el número de cuartos de baño. Con base en la definición del tipo de datos CHAR(3), los valores válidos para H_BATH_NUM serían '2', '1', '2.5', '10'. No obstante, esta decisión de tipo de datos crea potenciales problemas. Por ejemplo, si una aplicación ordena las casas por número de cuartos de baño, una consulta sería "ver" el valor '10' como menor a '2', que es claramente incorrecto. Entonces, se debe pensar en el uso esperado de los datos para definir en forma correcta el tipo de dato de atributo.

El diccionario de datos de la tabla 7.3 contiene algunos de los tipos de datos soportados por SQL. Para fines de enseñanza, la selección de tipos de datos está limitada a asegurar que casi cualquier RDBMS se puede usar para implementar los ejemplos. Si el RDBMS del lector está por completo apegado al ANSI para SQL, soportará muchos más tipos de datos que los que se ven en la tabla 7.4. Y numerosos RDBMS soportan tipos de datos además de los especificados en esa norma.

TABLA 7.4 Algunos tipos comunes de datos de SQL

TIPO DE DATOS	FORMATO	COMENTARIOS
Númérico	NUMBER(L,D)	La declaración NUMBER(7,2) indica números que se guardarán con dos lugares decimales y pueden ser hasta de siete dígitos de largo, incluido el signo y el lugar decimal. Ejemplos: 12.32,-134.99.
	INTEGER	Puede ser abreviado como INT. Los enteros son números naturales (enteros), de modo que no se pueden usar si se desea guardar números que requieren lugares decimales.
	SMALLINT	Igual que INTEGER pero limitado a valores enteros hasta de seis dígitos. Si sus valores enteros son relativamente pequeños, use SMALLINT en lugar de INT.
	DECIMAL(L,D)	Igual que la especificación NUMBER, pero la longitud de almacenamiento es una especificación mínima. Esto es, longitudes mayores son aceptables, no así las menores. DECIMAL(9,2), DECIMAL(9) y DECIMAL son todos aceptables.
Carácter	CHAR(L)	Datos de caracteres de longitud fija hasta para 255 caracteres. Si usted guarda cadenas ordenadas que no sean tan largas como el valor de parámetro CHAR, los espacios restantes se dejan sin usar. Por tanto, si usted especifica CHAR(25), las cadenas como Smith y Katzenjammer se guardan cada una como de 25 caracteres. No obstante, un código postal en Estados Unidos es siempre de tres dígitos de largo, de modo que CHAR(3) sería apropiado si se desea guardar esos códigos.
	VARCHAR(L) o VARCHAR2(L)	Datos de caracteres de longitud variable. La designación VARCHAR2(25) le permitirá guardar cadenas hasta de 25 caracteres de largo. Sin embargo, VARCHAR no dejará espacios sin usar. Oracle convierte automáticamente VARCHAR en VARCHAR2.
Fecha	DATE	Guarda fechas en el formato juliano de fechas.

Además de los tipos de datos que se muestran en la tabla 7.4, SQL soporta varios tipos de datos, incluido TIME, TIMESTAMP, REAL, DOUBLE, FLOAT e intervalos como INTERVAL DAY TO HOUR. Muchos RDBMS también han expandido la lista para incluir otros tipos de datos, por ejemplo, LOGICAL, CURRENCY, AutoNumber (Access) y sequence (Oracle), pero, en vista que este capítulo está diseñado para introducir aspectos básicos de SQL, la discusión se limita a los tipos de datos resumidos en la tabla 7.4.