Introducción

El diseño físico de bases de datos relacionales implica una serie de diferentes procesos que pretenden, principalmente, traducir el modelo lógico obtenido a través de las técnicas hasta ahora estudiadas, atendiendo a las características técnicas del sistema gestor de bases de datos relacional que se ha seleccionado para la implementación y administración de la base de datos. Una vez traducido, diseñando las tablas base, la representación de los datos y el diseño de las restricciones generales se optimiza la organización de los archivos y de los índices, analizando las transacciones, seleccionando la mejor organización de los ficheros, seleccionando qué índices crear y estimando qué requisitos de espacio de disco son necesarios. Una vez resueltos estos procesos, hay que diseñar las vistas de usuario, sus mecanismos de seguridad, el control de redundancia y la monitorización del sistema final.

En esta unidad se profundiza en la primera fase: **traducción del modelo lógico al modelo físico,** usando como sistema gestor de bases de datos Oracle y un lenguaje universal para su gestión, SQL. Los comandos ***CREALDRO*** permiten gestionar los objetos de una base de datos. Estos objetos pueden ser desde tablas hasta vistas, pasando por cuentas de usuarios, espacios de tablas, privilegios, roles, clústeres, etc. En esta unidad se abordarán los aspectos asociados con la creación de los objetos de una base de datos. Los comandos principales para tales acciones son CREATE, ALTER y DROP. La gestión se basa en estos procesos principales: el alta, la modificación o actualización y la eliminación de objetos. La gestión de datos no es más que el conjunto de acciones que permiten el alta de los datos, su modificación y su posible baja.

Los comandos ***INUPDEL*** posibilitan introducir, modificar y eliminar datos en los objetos. Los comandos que se usarán son INSERT INTO, UPDATE SET WHERE y DELETE WHERE. En la inserción de registros es usual insertar valores autonuméricos en las columnas que son claves primarias. Las secuencias son unas estructuras que permiten crear secuencias numéricas en Oracle.

La creación de tablas conlleva la reserva de espacio físico en los sistemas de almacenamiento. La **gestión del almacenamiento** es una tarea muy importante para amortizar los sistemas de almacenamiento y repartir los recursos para aquellas tablas que lo necesitan más que otras. Además, el fin último es que los accesos a los archivos sean lo más eficientes posible, con el fin de acelerarlos, teniendo en cuenta su disposición lógica y física. Aunque esta tarea la llevan a cabo los administradores de bases de datos, es importante que los programadores de aplicaciones conozcan sus aspectos básicos.

4.1. Arranque del diseño físico. El lenguaje SQL

El primer paso que tomar es decidir qué sistema gestor de bases de datos se adapta a las necesidades del problema, atendiendo a múltiples criterios: ¿la base de datos es relacional?, ¿es de tamaño pequeño?, ¿los datos estarán centralizados o distribuidos?, ¿existe un número pequeño de agentes que accedan a la base de datos?, ¿las

aplicaciones que explotan la base de datos son orientadas a objetos? Se podrían listar una gran cantidad de preguntas que, atendiendo a su itinerario de respuestas, determinaría cuál es la elección óptima en relación a los SGBD que existen actualmente en el mercado.

Una vez elegido el SGBD óptimo, se comienza la traducción del modelo lógico con el fin de obtener el esquema relacional de la base de datos. Las relaciones o tuplas que se han obtenido en la fase anterior se convertirán en tablas, sus atributos en columnas y las restricciones serán definidas atendiendo a las inherentes y no inherentes de las existentes en las posibilidades que ofrece el SGBD. Para crear objetos se debe contar con una cuenta de usuario y una contraseña. Cualquier agente que quiera comunicarse con la base de datos la necesita, ya sea para acceder a los datos a través de aplicaciones diseñadas para ello, administrarlos a través de mecanismos para la administración del sistema gestor de bases de datos o para operaciones de copias de seguridad, restauración o cualquier otra función que se preste.

Los desarrolladores de software, para la conexión a la base de datos, necesitan disponer de una cuenta de usuario y contraseña.

Cualquier usuario que se conecta a una base de datos debe conocer cierta información sobre el servidor y ella, como el nombre de host/dirección IP, el puerto de comunicación, el nombre de la base de datos, etcétera.

Figura 4.1. *La conexión a la base de datos requiere de una cuenta de usuario y contraseña, y los valores necesarios para la conexión a la máquina que provee del servicio.*

De forma general, los comandos usados en la presente obra son comandos ejecutados en el sistema gestor de bases Oracle. En la mayoría de los casos, estos comandos son muy parecidos con respecto a otros sistemas gestores de bases de datos, pero, a veces, se hace necesario conocer sus particularidades, ya que pueden ser implementados con algunas diferencias.

4.1.1. Normas de escritura en SQL

Cuando se escribe un comando en SQL se debe tener en cuenta una serie de reglas de escritura que pueden afectar a su correcta ejecución. Todos los comandos terminan en un punto y coma. Hasta que el **parser** no encuentra el símbolo ';' el comando no se ejecuta. Por tanto, se puede escribir los siguientes comandos que funcionan exactamente igual:

Ejemplo 1

```
Select *
From profesores
Where nombre='Antonio';
```

Ejemplo 2

```
Select * from profesores
Where nombre='Antonio';
```

Ejemplo 3

```
Select * from profesores
where nombre='Antonio'
;
```

Como se puede observar en los tres ejemplos anteriores, cualquier comando puede escribirse en diferentes saltos de líneas y usar tabuladores para mejorar la legibilidad. El comando se podría haber escrito incluso en una sola línea, ya que el procesador del lenguaje, parser, no ejecuta el comando hasta que no encuentra el símbolo ';'.

Los comandos se pueden escribir en minúsculas o en mayúsculas, pero hay que prestar mucha atención a los datos en sí. Estos sí son diferentes si están escritos de una forma u otra. Por ejemplo, los siguientes comandos son exactamente iguales:

```
Select * from Profesores where dni=22200500;
SELECT * FROM Profesores WHERE DNI=44200500;
SeLECt * from PROFesores wheRE dni=44200500;
```

Incluso cuando se hace referencia al nombre del objeto, por ejemplo, la tabla Profesor, se puede escribir tanto en minúsculas como en mayúsculas. Esto tiene una excepción: cuando se accede al diccionario de datos, ya que los valores de las tablas y vistas que almacenan los metadatos están escritos en su mayoría en mayúsculas.

Las siguientes consultas son diferentes, pues hacen referencia a valores distintos:

```
Select * from profesores where nombre='Antonio';
Select * from profesores where nombre='antonio';
Select * from profesores where nombre='antoniO';
```

Los comandos para la definición de elementos de la base de datos no pueden ser anulados, es decir, no se puede usar el comando `ROLLBACK` para volver atrás cuando se ha creado cualquier objeto con el comando `CREATE`. Una vez creada, solo puede ser eliminada con el comando `DROP`. A los elementos de una base de datos se les suele llamar **objetos de la base de datos.** Estos objetos pueden ser tablas, columnas, restricciones, sinónimos, índices, vistas, vistas actualizables, vistas materializadas, disparadores, espacios de tablas, segmentos, extensiones, procedimientos, funciones, cursores, usuarios, roles, privilegios, perfiles y un largo etcétera.

Ten en cuenta

Es muy normal escribir en procesadores de texto un texto entrecomillado. Estos programas suelen usar dos símbolos para entrecomillar. Por ejemplo, ‘Esto es un ejemplo’, tiene dos caracteres, comienzo de entrecomillado,‘, y cierre, ’, y son diferentes. En el sistema gestor de bases de datos solo hay un símbolo ('). Igual ocurre con el carácter doble comilla, ".

También existen unas reglas para nombrar a cualquier objeto de una base de datos, ya sea una tabla, una columna, una restricción, un usuario o cualquier otro tipo. Las siguientes son reglas para los sistemas gestores de bases de datos de Oracle:

- Deben comenzar con una letra en el rango A..Z o a..z, nunca por un número.
- Debe contener como máximo 30 caracteres y no debe contener símbolos especiales (´, &, % , @, |, <, >, +, ^, [,], *, (,), =, ?, ¿,...). Sí se permite el guion bajo y otros como $ y #, pero no se recomienda su uso.
- Ningún objeto se puede llamar igual que otro siempre que sean del mismo tipo. Por ejemplo, dos tablas nunca se podrían llamar ambas Profesor, pero sí podría haber una cuenta de usuario llamada Profesor y una tabla con el mismo nombre.
- Si el nombre tiene espacios, se debe escribir entre comillas simples con el carácter '.
- Se puede usar comillas dobles, ", para indicar que dos objetos son diferentes: "Profesores" y "PROFESORES" serían dos tablas diferentes. Este uso debería evitarse.

Nota técnica

Un **parser**, analizador o analizador sintáctico, es un programa informático cuya finalidad es el procesamiento carácter a carácter de un texto escrito de forma plana, es decir, el entendimiento de caracteres alfanuméricos representados en la tabla ASCII. Con este mecanismo se consigue obtener un nivel de abstracción muy bajo para el programador, pero muy alto para la máquina. El objetivo final de este programa es «leer» el texto y mandar órdenes más abstractas a otro programa que actúa como interfaz y es el que realmente ejecuta las órdenes.

4.1.2. Los tipos de datos

Cuando se describe las columnas de una tabla, los sistemas gestores de bases de datos necesitan conocer su dominio, es decir, el conjunto posible de valores que el atributo correspondiente a dicha columna puede tomar. La declaración de estos valores se puede hacer definiendo su dominio asociado, explicitando cada uno de los valores del conjunto o indicando un tipo. El tipo hace alusión a si es numérico, alfabético, alfanumérico, fecha y sus límites posibles. Por ejemplo, el tipo de dato de apellido puede ser alfabético y de 50 caracteres de longitud, el número de coches, un número natural menor que cien, etcétera.

En realidad, las aplicaciones de desarrollo necesitan saber el número de bits que un dato va a necesitar cuando tome un valor. En vez de indicar el número de bits, se indica el tipo de datos. Cada tipo tiene asociado un número de bits dependiendo de si es natural, un entero, un número real, etc. Existen diferencias entre los tipos de datos que usa un SGBD y otro, y, por tanto, es necesario analizar las diferencias. Los tipos de datos se clasificarán en numéricos, de textos, de fecha y hora y de gran tamaño. En las tablas 4.1, 4.2, 4.3 y 4.4 se detallan los que se usan en Oracle:

Tabla 4.1. Tipos de datos numéricos en Oracle

Tipos de datos numéricos		
Tipo	**Descripción**	**Ejemplos**
NUMBER(E,D)	Puede contener tanto valores enteros como decimales. Se indica el número de dígitos que compone el número, E. Si tiene decimales, de ese número, con D se indica cuántos dígitos son decimales.	`Sueldo NUMBER(5)`: 5 dígitos enteros, desde 0 hasta 99999 `Planta NUMBER(1)`: 1 dígito entero. Los valores pueden ir desde 0 hasta 9. `Precio NUMBER(7,3)`: siete dígitos numéricos, de los cuales 3 son para la parte decimal. Los valores pueden ir desde el 0 al 9999,999
BINARY_FLOAT	Puede contener valores numéricos de coma flotante de hasta 32 bits.	precio `BINARY_FLOAT`. Desde 1.17E-38F hasta 3.40+38F.
BINARY_DOUBLE	El doble que BINARY_FLOAT, es decir, hasta 64 bits.	capital `BINARY_DOUBLE`

Tabla 4.2. Tipos de datos de texto en Oracle

Tipos de texto		
Tipo	**Descripción**	**Ejemplos**
VARCHAR2(N)	Puede contener una cadena de hasta N caracteres alfanuméricos. Puede contener cualquier carácter imprimible de la tabla ASCII. El tamaño máximo es de 32 KB.	`Nombre VARCHAR2(10)`: hasta 10 caracteres alfanuméricos.
CHAR(N)	Contiene siempre N caracteres alfanuméricos. Los no introducidos se rellenan con espacios en blanco. El tamaño máximo es de 2000 bytes.	`codigo CHAR(10)`:10 caracteres alfanuméricos.
LONG	Admite cadenas de hasta 2 gigabytes.	`Foto LONG`

Tabla 4.3. Tipos de datos fecha y hora en Oracle

Tipos de datos fecha y hora		
Tipo	**Descripción**	**Ejemplos**
DATE	Almacena la fecha y/u hora.	`fecha DATE`
TIMESTAMP[(n)]	Almacena con N dígitos la fecha y hora.	`Fecha TIMESTAMP`
INTERVAL YEAR(N) TO MONTH(M)|DAY TO SECOND(S)	Puede ser año y mes con n y m dígitos o día a segundo con s dígitos.	`Duracion INTERVAL YEAR TO MONTH;` `Momento INTERVAL DAY(4) TO SECOND(6);`

Tabla 4.4. Otros tipos de datos

Otros tipos		
Tipo	**Descripción**	**Ejemplos**
BOOLEAN	Puede ser true, false o null.	`Titulado BOOLEAN`
BLOB, CLOB, BFILE	Datos de gran tamaño.	`Foto BLOB`