Capítulo

1

INTRODUCCIÓN A LAS BASES DE DATOS RELACIONALES

1.1 CONCEPTOS

Una *base de datos relacional* es un conjunto de una o más tablas estructuradas en registros (líneas) y campos (columnas), que se vinculan entre sí por campos en común que poseen las mismas características en ambas tablas, como por ejemplo el nombre de campo, tipo y longitud. A estos campos generalmente se les denomina campos identificadores (ID) o campos clave. A esta forma de construir bases de datos se le denomina modelo relacional.

Toda base de datos relacional está formada por uno o varios bloques de información llamados TABLAS (inicialmente denominados *ficheros* o *archivos*) que normalmente tendrán alguna característica en común.

Una *tabla* o archivo de datos es un conjunto conexo de información del mismo tipo, por ejemplo en una base de datos de una biblioteca, una tabla estará constituida por la información relativa a todos los libros de la misma, otra tabla contendrá información sobre los lectores, etc.

Cada tabla está formada por *registros*. Un registro es la unidad elemental de información de la tabla o fichero (en un archivo clásico no automatizado un registro se corresponde con lo que suele llamarse ficha). En la tabla o fichero de libros, un registro estaría constituido por la información correspondiente a cada libro concreto, con su título, autor, área, editorial, etc. Cada registro está formado por uno o más elementos llamados *campos*. Un campo es cada una de las informaciones que interesa almacenar en cada registro, y es por tanto la unidad elemental de información del registro. En el ejemplo anterior, un campo sería el título del libro, otro campo su autor, etc.

Gracias a la aparición de los llamados programas de usuario (sistemas gestores de base de datos) es posible realizar la gestión de tablas de una base de datos, sin tener que realizar programas que procesen esos datos, facilitando todas las operaciones de creación, actualización, consulta y creación de informes con los datos recogidos. La forma de almacenar

los ficheros, registros y campos es distinta, desde el punto de vista técnico, dependiendo de las características de la información que se quiere tratar. Sea cual sea la estructura de la base de datos, siempre se pueden incluir ficheros de índices secundarios para facilitar una respuesta eficaz a determinados tipos de acceso.

Históricamente el almacenamiento de la información ha experimentado una evolución muy rápida y siempre paralela a la disponibilidad y avances en el software y el hardware. Comenzó almacenándose la información en ficheros planos, para ir avanzando hasta tecnologías muy desarrolladas actualmente como es el Data warehouse. No obstante, las bases de datos, a través de sus sistemas gestores ocupan la parcela más importante en el campo del almacenamiento y administración de la información.

Actualmente, los sistemas de gestión de base de datos (abreviado mediante SGBD o DBMS) organizan y estructuran los datos de tal modo que puedan ser recuperados y manipulados por usuarios y programas de aplicación. Las estructuras de los datos y las técnicas de acceso proporcionadas por un DBMS particular se denominan su *modelo de datos.* El modelo de datos determina la «personalidad» de un DBMS, y las aplicaciones para las cuales está particularmente bien conformado.

Existe un tipo de lenguaje estándar normalizado para trabajar con bases de datos relacionales denominado SQL (*Structured Query Language*). SQL es un lenguaje de base de datos para bases de datos relacionales, y utiliza *el modelo de datos relacional.*

Las bases de datos relacionales pasan por un proceso al que se le conoce como *normalización*, que es entendido como el proceso necesario para que una base de datos sea utilizada de manera óptima.

Entre las ventajas más importantes del modelo relacional podemos citar que garantiza herramientas para evitar la duplicidad de registros a través de campos clave o llaves, garantiza también la integridad referencial (por ejemplo, al eliminar un registro elimina todos los registros relacionados dependientes) y favorece la normalización para que el trabajo con la base de datos resulte más comprensible y aplicable.

1.2 MODELOS DE DATOS

Desde un punto de vista simple podríamos definir un modelo de datos como la descripción pormenorizada de una base de datos. Típicamente un modelo de datos permite describir las estructuras de datos de la base, su tipo, descripción, la forma en que se relacionan, restricciones de integridad y otras características de los datos. Es factible pensar que un modelo de datos permite describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí. No hay que perder de vista que una Base de Datos siempre está orientada a resolver un problema práctico determinado. En la modelización se representa el problema realizando múltiples abstracciones

con el fin de asimilar toda la información del mismo. De esta forma se generará un mapa donde estén identificados todos los objetos de la base de datos.

La modelización habitualmente suele estar en manos de analistas informáticos que no suelen ser expertos en el campo o dominio del problema a resolver. Será, por tanto, muy necesario contar en la fase de modelado con la colaboración de personal experto en la materia no necesariamente con conocimientos informáticos, pero sí conocedor de los pormenores del negocio en el que se está realizando la modelización.

Por otro lado, en la modelización no debemos alejarnos de los estándares en la materia, lo que hará posible aprovechar las herramientas de software ya disponibles en el mercado.

Hay que tener presente que una base de datos representa la información contenida en algún dominio del mundo real y el diseño de la base de datos persigue extraer todos los datos relevantes relativos a un problema real que definirá los datos esenciales de la base de datos. Para extraer estos datos se debe realizar un análisis en profundidad de dominio del problema que identifique los datos esenciales para la base de datos. Extraídos estos datos, comienza el proceso de modelización, que nos llevará a construir un esquema que exprese con toda exactitud los datos que el problema requiere almacenar. El proceso de modelización suele basarse en herramientas de diseño de bases de datos preexistentes.

Así mismo, el sistema gestor de bases de datos (SGBD) que se utilizará debe ser seleccionado para que cubra todas las características de la modelización, de tal forma que abarque nuestro problema, pero que tampoco sea sobredimensionado. Ello exige conocer el estado y características de los sistemas gestores de bases de datos más habituales en el mercado actualmente. En los párrafos que siguen se presenta un análisis de los SGBD más habituales actualmente.

1.3 CARACTERÍSTICAS DE LOS SISTEMAS GESTORES DE BASE DE DATOS DEL MERCADO PARA LA MODELIZACIÓN

Cuando se realiza la modelización es necesario tener muy en cuenta las características de los sistemas gestores de bases de datos disponibles en el mercado. Dicha modelización se adaptará a un sistema gestor determinado según la naturaleza del problema que se modela y las prestaciones y coste del SGBD. A continuación se presenta una comparativa entre los sistemas gestores de base de datos más utilizados actualmente en lo relativo a información general, sistema operativo con el que trabajan, características fundamentales, uso de tablas y vistas, índices y particionamiento.

1.3.1 Información general

Sistema Gestor de BD	Creador	Fecha de la primera versión pública	Última versión estable	Licencia de software
Adaptive Server Anywhere	Sybase/iAnywhere	1992	10.0	Propietario
Adaptive Server Enterprise	Sybase Inc	1987	15.0	Propietario
ANTs Data Server	ANTs Software	1999	3.6	Propietario
DB2	IBM	1982	9	Propietario
Firebird	Firebird Foundation	25 de julio de 2000	2.1	Licencia Pública InterBase
Informix	Informix Software	1985	10.0	Propietario
HSQLDB	Hsqldb.Org	2001	1.9	Licencia BSD
Ingres	Berkeley University, Computer Associates	1980	2006	CA-TOSL
InterBase	Borland	1985	7.5.1	Propietario
SapDB	SAP AG		7.4	GPL con drivers LGPL
MaxDB	MySQL AB, SAP AG		7.7	GPL o propietario
Microsoft SQL Server	Microsoft	1989	2008	Propietario
MySQL	MySQL AB	Noviembre de 1996	5.0	GPL o propietario
Oracle	Oracle Corporation	1977	11g R 2	Propietario
PostgreSQL	PostgreSQL Global Development Group	Junio de 1989	9.0	Licencia BSD
SmallSQL	SmallSQL	16 de abril de 2005	0.12	LGPL
SQLite	D. Richard Hipp	17 de agosto de 2000	3.6.16	Dominio público

1.3.2 Soporte del sistema operativo

	Windows	Mac OS X	Linux	BSD	Unix	z/OS
Adaptive Server Enterprise	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✗ No
ANTs Data Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	?
DB2	✓ Sí	✗ No	✓ Sí	✗ No	✓ Sí	✓ Sí
Firebird	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	Quizás
HSQLDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Informix	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✗ No
Ingres	✓ Sí		✓ Sí		✓ Sí	Quizás
InterBase	✓ Sí	✗ No	✓ Sí	✗ No	✓ Sí (Solaris)	✗ No
SapDB	✓ Sí	✗ No	✓ Sí	✗ No	✓ Sí	
MaxDB	✓ Sí	✗ No	✓ Sí	✗ No	✓ Sí	
Microsoft SQL Server	✓ Sí	✗ No	✗ No	✗ No	✗ No	✗ No
MySQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	Quizás
Oracle	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✗ No
SmallSQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
SQLite	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	Quizás

1.3.3 Características fundamentales

	Características ACID	Integridad referencial	Transacciones	Unicode
Adaptive Server Enterprise	✓ Sí	✓ Sí	✓ Sí	✓ Sí
ANTs Data Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí
DB2	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Firebird	✓ Sí	✓ Sí	✓ Sí	✓ Sí
HSQLDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Informix	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Ingres	✓ Sí	✓ Sí	✓ Sí	✓ Sí
InterBase	✓ Sí	✓ Sí	✓ Sí	✓ Sí
SapDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí
MaxDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Microsoft SQL Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí
MySQL	Depende	Depende	Depende	✓ Sí
Oracle	✓ Sí	✓ Sí	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí
SQLite	✓ Sí	✗ No	Básico	✓ Sí

En bases de datos se denomina ACID a un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. En concreto ACID es un acrónimo de *Atomicity*, *Consistency*, *Isolation and Durability* (Atomicidad, Consistencia, Aislamiento y Durabilidad en español).

- Atomicidad: es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- Consistencia: es la propiedad que asegura que solo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
- Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.
- Durabilidad: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

1.3.4 Tablas y vistas

	Tabla temporal	Vista materializada
Adaptive Server Enterprise	✓ Sí	✓ Sí
ANTs Data Server	✓ Sí	✓ Sí
DB2	✓ Sí	✓ Sí
Firebird	✓ Sí	✗ No
HSQLDB	✓ Sí	✗ No
Informix	✓ Sí	✓ Sí
Ingres	✓ Sí	✗ No
InterBase	✓ Sí	✗ No
SapDB	✓ Sí	✗ No
MaxDB	✓ Sí	✗ No
Microsoft SQL Server	✓ Sí	Similar
MySQL	✓ Sí	✗ No
Oracle	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✗ No
SQLite	✓ Sí	✗ No

1.3.5 Índices

	Árbol R-/R+	Hash	Expresión	Parcial	Reversa	Mapa de bits
Adaptive Server Enterprise	✗ No	✗ No	✓ Sí	✗ No	✓ Sí	✗ No
ANTs Data Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
DB2	✗ No	?	✗ No	✗ No	✓ Sí	✓ Sí
Firebird	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No
Informix	✓ Sí	✓ Sí	✓ Sí	✗ No	✗ No	✗ No
Ingres	✓ Sí	✓ Sí	✗ No	✗ No	✗ No	✗ No
InterBase			✗ No	✗ No	✗ No	✗ No
SapDB			✗ No	✗ No	✗ No	✗ No
MaxDB			✗ No	✗ No	✗ No	✗ No
Microsoft SQL Server			✗ No	✗ No	✗ No	✗ No
MySQL	Tablas MyISAM solo	Tablas HEAP solo	✗ No	✗ No	✗ No	✗ No
Oracle	Edición EE solo		✓ Sí	✗ No	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✗ No	✗ No
SQLite	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No

1.3.6 Otros objetos

	Dominio	Cursor	Trigger	Funciones	Procedimiento	Rutina externa
Adaptive Server Enterprise	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
ANTs Data Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
DB2	✗ No	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Firebird	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
HSQLDB		✗ No	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Informix		✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Ingres	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	
InterBase	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
SapDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	
MaxDB	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	
Microsoft SQL Server	✗ No	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
MySQL	✗ No	✓ Sí	✓ Sí	✓ Sí	✓ Sí [3]	✓ Sí
Oracle	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí	✓ Sí
SQLite	✗ No	✗ No	✓ Sí	✗ No	✗ No	✓ Sí

Funciones y Procedimiento se refieren a las rutinas internas escritas en SQL o lenguajes procedurales como PL/SQL. Rutina externa se refiere a la escritura en los lenguajes anfitriones como C, Java, Cobol, etc. Procedimiento almacenado es un término comúnmente usado para ese tipo de rutinas. Sin embargo, su definición varía entre diferentes vendedores de bases de datos.

1.3.7 Particionamiento

	Rango	Hash	Compuesto (Rango+Hash)	Lista
Adaptive Server Enterprise	AA	AA	AA	AA
ANTs Data Server	✓ Sí	✓ Sí	✓ Sí	✓ Sí
DB2	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Firebird	✗ No	✗ No	✗ No	✗ No
Ingres	✓ Sí	✓ Sí	✓ Sí	✓ Sí
InterBase	✗ No	✗ No	✗ No	✗ No
Microsoft SQL Server	✓ Sí	✗ No	✗ No	✗ No
MySQL	✓ Sí	✓ Sí	✓ Sí	✓ Sí
Oracle	✓ Sí	✓ Sí	✓ Sí	✓ Sí
PostgreSQL	✓ Sí	✗ No	✗ No	✓ Sí
SQLite	✓ Sí	✓ Sí	✓ Sí	✓ Sí

La tabla presenta información acerca de qué métodos de particionamiento son soportados nativamente por los diferentes SGBD.

Una partición es una división de una base de datos lógica o sus elementos constituyentes en partes independientes. La partición de bases de datos se hace normalmente por razones de mantenimiento, rendimiento o manejo. Se utiliza habitualmente en Sistemas de Administración de Bases de Datos Distribuidas. Cada partición puede ser extendida hasta múltiples nodos, y los usuarios en el nodo pueden hacer transacciones locales en la partición. Esto aumenta el rendimiento en sitios que tienen transacciones regularmente involucrando ciertas vistas de datos, y manteniendo la disponibilidad y la seguridad. Esta partición puede hacerse creando bases de datos más pequeñas separadas (cada una con sus propias tablas, índices, y registros de transacciones) o dividiendo elementos seleccionados, por ejemplo, solo una tabla.

1.4 TIPOS DE MODELOS DE DATOS

Una opción bastante utilizada a la hora de clasificar los modelos de datos es hacerlo de acuerdo al nivel de abstracción que presentan. Teniendo presente este criterio de abstracción podríamos clasificar los modelos de datos de la forma siguiente:

- *Modelos de Datos Conceptuales*. Se trata de los modelos orientados a la descripción de estructuras de datos y restricciones de integridad. Se usan fundamentalmente durante la etapa de análisis de un problema dado y están orientados a representar los elementos que intervienen en ese problema y sus relaciones. El ejemplo más típico es el *Modelo Entidad-Relación*. Un diagrama o modelo entidad-relación (a veces denominado por sus siglas, E-R "Entity relationship", o, "DER" Diagrama de Entidad Relación) es una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información así como sus interrelaciones y propiedades tal y como veremos en un capítulo posterior (Figura 1.1).

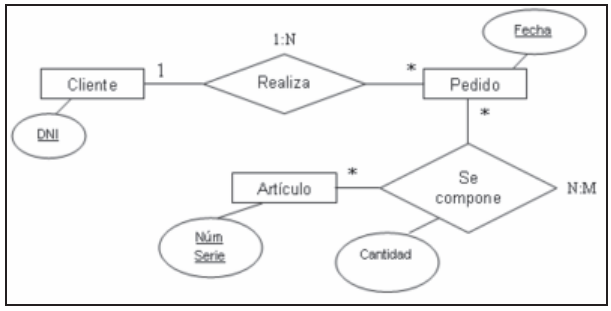


Figura 1.1

- *Modelos de Datos Lógicos*. Son modelos orientados a las operaciones más que a la descripción de una realidad. Usualmente están implementados en algún Gestor de Base de Datos. El ejemplo más típico es el *Modelo Relacional*, que cuenta con la particularidad de contar también con buenas características conceptuales (*Normalización de bases de datos*). El modelo relacional para la gestión de una base de datos es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de «relaciones». Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar

Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar, esto es, pensando en cada relación como si fuese una tabla que está compuesta por registros (cada fila de la tabla sería un registro o tupla), y columnas (también llamadas campos).

- *Modelos de Datos Físicos*. Son estructuras de datos a bajo nivel implementadas dentro del propio gestor de base de datos. Ejemplos típicos de estas estructuras son los Árboles B+, las estructuras de Hash, etc.

Un *árbol-B* es un tipo de estructura de datos de árboles. Representa una colección de datos ordenados de manera que se permite una inserción y borrado eficientes de elementos. Es un índice, multinivel, dinámico, con un límite máximo y mínimo en el número de claves por nodo. Su nombre se deriva de la inicial del nombre de sus creadores: Byron Pérez y PlusEmma Mora. Ademas el signo "+" hace mención a la primera implementación de esta estructura: la cual fue utilizada por primera vez en el lenguaje C++ en un curso (llamado Algoritmos y Estructuras de Datos 2) mientras los creadores del árbol sacaban su título de Licienciatura. Un árbol-B+ es una variación de un árbol-B. En un árbol-B+, en contraste respecto a un árbol-B, toda la información se guarda en las hojas. Los nodos internos solo contienen claves y punteros. Todas las hojas se encuentran en el mismo, más bajo nivel. Los nodos hoja se encuentran unidos entre sí como una lista enlazada para permitir búsqueda secuencial. El número máximo de claves en un registro es llamado el orden del árbol-B+. El mínimo número de claves por registro es la mitad del máximo número de claves. Por ejemplo, si el orden de un árbol-B+ es n, cada nodo (exceptuando la raíz) debe tener entre n/2 y n claves. El número de claves que pueden ser indexadas usando un árbol-B+ está en función del orden del árbol y su altura. Para un árbol-B+ de orden *n*, con una altura *h* tenemos:

- Número máximo de claves es: n^h
- Número mínimo de claves es: $2(n / 2)^{h-1}$

En la Figura 1.2 se representa un árbol B+ simple (una variación del árbol B) que enlaza los elementos 1 al 7 a valores de datos d1-d7.

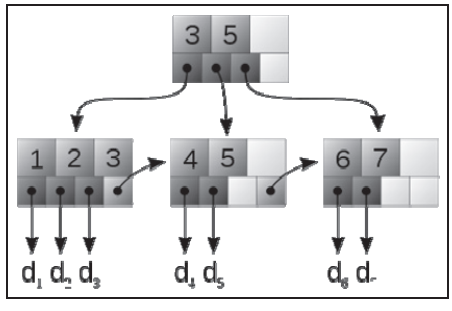


Figura 1.2

Una *estructura de hash* (Figura 1-3) se refiere a una función o método para generar claves o llaves que representen de manera casi unívoca a un documento, registro, archivo, etc., resumir o identificar un dato a través de la probabilidad, utilizando una función hash o algoritmo hash. Un hash es el resultado de dicha función o algoritmo. Una función de hash es una función para resumir o identificar probabilísticamente un gran conjunto de información, dando como resultado un conjunto imagen finito generalmente menor (un subconjunto de los números naturales por ejemplo). Varían en los conjuntos de partida y de llegada y en cómo afectan a la salida similitudes o patrones de la entrada. Una propiedad fundamental del hashing es que si dos resultados de una misma función son diferentes, entonces las dos entradas que generaron dichos resultados también lo son. Es posible que existan claves resultantes iguales para objetos diferentes, ya que el rango de posibles claves es mucho menor que el de posibles objetos a resumir (las claves suelen tener en torno al centenar de bits, pero los ficheros no tienen un tamaño límite). Son usadas en múltiples aplicaciones, como los arrays asociativos, criptografía, procesamiento de datos y firmas digitales, entre otros. Una buena función de hash es una que experimenta pocas colisiones en el conjunto esperado de entrada; es decir, que se podrán identificar unívocamente las entradas (ver función inyectiva). Muchos sistemas relacionados con la seguridad informática usan funciones o tablas hash.

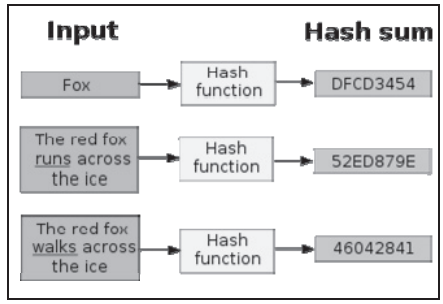


Figura 1.3

1.5 MODELOS DE DATOS Y TIPOS DE SISTEMAS GESTORES DE BASES DE DATOS

Cada Sistema Gestor de Bases de Datos (SGBD) está basado en el uso de un modelo de datos y en el uso de su teoría para la descripción y manipulación de los datos. De esta forma tenemos SGBD jerárquicos, en red, relacionales, orientados a objetos, etc., los cuales están basados y se apoyan en los modelos de datos correspondientes: jerárquico, en red, relacional (algebraicos o predicativos), orientados a objetos, etc.

1.5.1 Modelo de datos jerárquico

Básicamente podemos distinguir tres tipos de estructuras de Bases de Datos: Jerárquica, en Red y Relacional. La *organización jerárquica*, que es la que primero se utilizó, se basa en el establecimiento de jerarquías o niveles entre los distintos campos de los registros, basándose en el criterio de que los campos de mayor jerarquía sean los más genéricos, y tiene una estructura arborescente, donde los nodos del mismo nivel corresponden a los campos y cada rama a un registro (Figura 1.4). Para acceder a un campo que se encuentra en un determinado nivel, es preciso localizarlo partiendo del nivel superior y descendiendo por las ramas hasta llegar al mismo.

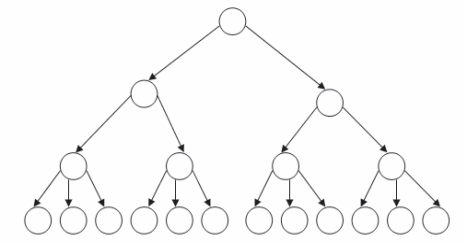


Figura 1.4

Tomemos como ejemplo la base de datos de una biblioteca y situemos como nivel superior el código de materia y el título de la materia, y supongamos que tomamos como campo maestro el título de la materia. En un segundo nivel de la jerarquía se incluyen las tablas de autores asociados a cada materia; en un tercer nivel tendríamos, por ejemplo, las tablas correspondientes a los autores y sus respectivos títulos de libros, y así hasta el último nivel en el que incluimos título, referencia, año de edición, número de ejemplares, o cualquier otro dato individual referente al libro buscado.

En la Figura 1.5 se muestra una organización jerarquizada. Para acceder a un dato del último nivel, por ejemplo año de edición de un determinado libro, hay que recorrer todos los niveles desde el más alto (materia, etc.). Esta forma de organización puede hacer lenta la obtención de determinadas informaciones, ya que para acceder a un nodo (campo) hay que recorrer toda la rama, partiendo de la raíz (nodo de mayor jerarquía), es decir, todos los campos precedentes en el registro. No obstante, existen estructuras arborescentes más sofisticadas que incluyen índices y que permiten acelerar el resultado de las consultas.

Uno de los sistemas de gestión de bases de datos jerárquicas más populares fue el *Information Management System* (IMS) de IBM, introducido en 1968. Este sistema presentaba las siguientes ventajas:

- *Estructura simple*: la organización de una base de datos IMS es fácil de entender. La jerarquía de la base de datos se asemejaba al diagrama de organización de una empresa o a un árbol familiar.

- *Organización padre/hijo*: una base de datos IMS era excelente para representar relaciones padre/hijo, tales como "A es pieza de B", "A es propiedad de B", "el título A es del autor B", etc.

- *Rendimiento*: IMS almacenaba las relaciones padre/hijo como punteros físicos de un registro de datos a otro, de modo que el movimiento a través de la base de datos era rápido. Puesto que la estructura era sencilla, IMS podía colocar los registros padre e hijo cercanos unos a otros en el disco, minimizando la cuota entrada/salida del disco.

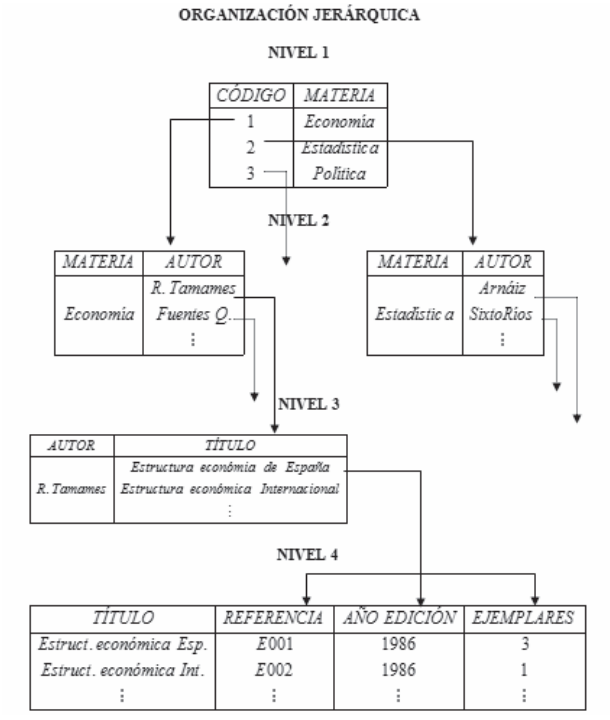


Figura 1.5

1.5.2 Modelo de datos en red

Para resolver el problema de lentitud de la organización jerárquica se utiliza la *organización en Red* que corresponde a una estructura de grafo, donde existe más de una conexión entre los nodos de diversos niveles, de forma que puedan recorrerse por distintos caminos sin necesidad de acudir cada vez a la raíz, con lo cual la búsqueda es más flexible,

desapareciendo el concepto de "jerarquía" entre campos, pues un campo puede ser descendiente de su antecesor por un camino de la red y ascendente por otro.

Si se crean conexiones entre nodos de igual nivel, el acceso a campos de determinado nivel se logrará más rápidamente, así por ejemplo, en el caso de la base de datos de la biblioteca, se podrían listar los títulos de los distintos libros, a partir de un título dado sin acceder cada vez a los autores.

El modelo de datos en red extiende el modelo jerárquico permitiendo que un registro participe en múltiples relaciones padre/hijo. Estas relaciones se conocen como *conjuntos* en el modelo de red.

El inconveniente esencial de esta estructura es la necesidad de utilizar mucha más cantidad de memoria, al tener que almacenar en cada nodo las posiciones de los campos siguientes, mediante apuntadores.

En la figura 1-6 se representa un grafo similar al del modelo jerárquico, pero adecuado al modelo en red. En este grafo desaparece el concepto de jerarquía entre campos, pues un campo puede ser descendiente de su antecesor por un camino de la red y ascendente por otro.

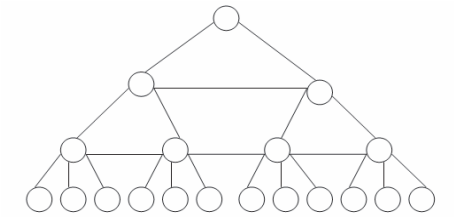


Figura 1.6

En 1971 la Conferencia sobre Lenguajes de Sistemas de Datos publicó un estándar oficial para bases de datos en red, al que se conoció con el nombre de CODASYL. IBM nunca desarrolló un DBMS en red, eligiendo en su lugar extender el IMS a lo largo de los años. Pero durante los años setenta, compañías de software independientes se apresuraron a adoptar el modelo en red, creando productos tales como el IDMS de Cullinet, el Total de Cincom y el DBMS Adabas que se hizo muy popular.

Para un programador, acceder a una base de datos en red era muy similar a acceder a una base de datos jerárquica. Un programa de aplicación podía:

- Hallar un registro padre específico mediante clave (como por ejemplo un número de cliente en un procesamiento de pedidos).

- Descender al primer hijo en un conjunto particular (el primer pedido remitido por este cliente).

- Moverse lateralmente de un hijo al siguiente dentro del conjunto (la orden siguiente remitida por el mismo cliente).

- Ascender desde un hijo a su padre en otro conjunto (el vendedor que aceptó el pedido).

Una vez más el programador tenía que recorrer la base de datos registro a registro, especificando esta vez qué relación recorrer además de indicar la dirección.

Las bases de datos en red tenían varias ventajas entre las que se pueden destacar las siguientes:

- *Flexibilidad*: las múltiples relaciones padre/hijo permitían a una base de datos en red representar datos que no tuvieran una estructura jerárquica sencilla.

- *Normalización*: el estándar CODASYL reforzó la popularidad del modelo de red, y los vendedores de minicomputadores tales como Digital Equipment Corporation y Data General implementaron bases de datos en red.

- *Rendimiento*: a pesar de su superior complejidad, las bases de datos en red reforzaron el rendimiento aproximándolo al de las bases de datos jerárquicas. Los conjuntos se representaron mediante punteros a registros de datos físicos, y en algunos sistemas, el administrador de la base de datos podía especificar la agrupación de datos basada en una relación de conjunto.

Las bases de datos en red tenían sus desventajas también. Igual que las bases de datos jerárquicas resultaban muy rígidas. Las relaciones de conjunto y la estructura de los registros tenían que ser especificadas de antemano. Modificar la estructura de la base de datos requería generalmente la reconstrucción de la base de datos completa. Tanto las bases de jerárquicos como las bases en red eran herramientas para programadores.

1.5.3 Modelo de datos relacional

Quizás, el problema fundamental que suele plantearse al realizar una base de datos real, formada por varias tablas, es la repetición de datos, es decir, campos repetidos en diferentes tablas (redundancia), lo cual va a dificultar su gestión, es decir, la actualización, inserción, modificación, eliminación, consulta, etc.

Para resolver estos problemas es necesario que exista integración entre las distintas tablas y que esté controlada la repetición de datos. Así surgen los llamados Sistemas de Gestión de Bases de Datos relacionales que, en el caso de los microordenadores, están concebidos como un conjunto de programas de propósito general que permiten controlar el acceso y la utilización de las bases de datos de forma que satisfagan las necesidades del usuario (programas de usuario) y que actúen con independencia de los datos, y con ellos las

llamadas *Bases de Datos Relacionales* que pueden resolver, mejor que otras organizaciones, las dificultades de redundancia y no integración de los datos. En este tipo de bases de datos se suprimen las jerarquías entre campos, pudiéndose utilizar cualquiera de ellos como clave de acceso.

La teoría relacional se basa en el concepto matemático de relación. Se debe a E.F. Codd, quien ha desarrollado una sólida fundamentación teórica. Aunque dicha teoría requiere para su completa implantación que el acceso a la memoria sea por contenido y no por dirección, como ocurre en los actuales ordenadores, puede adecuarse y de hecho se está implantando y desarrollando en la mayoría de los equipos.

Las principales ventajas de la utilización de Bases de Datos relacionadas son:

- Actúan sobre las tablas en su conjunto, en lugar de hacerlo sobre los registros como ocurre en otros sistemas.
- Se pueden realizar consultas complejas que utilizan varias tablas de forma simple.
- Son fáciles de utilizar (la organización física de los datos es independiente de su tratamiento lógico).

La organización relacional se caracteriza porque las tablas de la base de datos tienen estructura de matriz o tabla bidimensional, donde las filas son los registros y las columnas los campos (Figura 1.7).

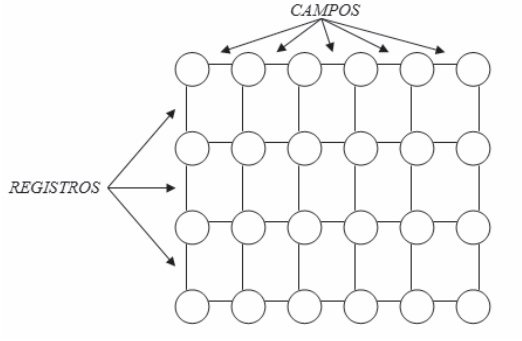
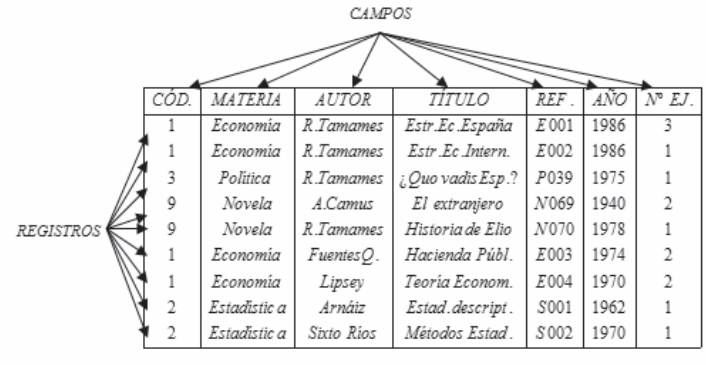


Figura 1.7

Las tablas son tratadas como conjuntos matemáticos, obtenidas como subconjuntos del producto cartesiano de los rangos de posibles valores de los distintos campos que la forman. Cada tabla dispone de una cabecera que es un registro especial donde figuran los nombres de los campos y una serie de registros (filas) donde se describen los objetos.

El esquema que se muestra en la Figura 1.8 representa la tabla del ejemplo de la base de datos de la biblioteca adaptado al modelo relacional. La tabla tiene todos sus campos relacionados, de forma que es posible tomar, por ejemplo, como entrada, la materia y averiguar el número de ejemplares, o bien obtener todos los libros editados en 1986 sin más que consultar el campo correspondiente a "año de edición".



CÓD.	MATERIA	AUTOR	TÍTULO	REF.	AÑO	Nº EJ.
1	Economía	R.Tamames	Estr.Ec.España	E001	1986	3
1	Economía	R.Tamames	Estr.Ec.Intern.	E002	1986	1
3	Política	R.Tamames	¿Quo vadis Esp.?	P039	1975	1
9	Novela	A.Camus	El extranjero	N069	1940	2
9	Novela	R.Tamames	Historia de Elio	N070	1978	1
1	Economía	FuentesQ.	Hacienda Públ.	E003	1974	2
1	Economía	Lipsey	Teoría Econom.	E004	1970	2
2	Estadística	Arnáiz	Estad.descript.	S001	1962	1
2	Estadística	Sixto Ríos	Métodos Estad.	S002	1970	1

Figura 1.8

Para que la estructura de las tablas cumpla las leyes de la teoría relacional deben satisfacerse las siguientes condiciones:

- Todos los registros de la tabla deben tener el mismo número de campos, aunque alguno de ellos esté vacío, deben ser registros de longitud fija.

- Cada campo tiene un nombre o etiqueta que hay que definir previamente a su utilización. No obstante, una vez creado el fichero se podrá ampliar o disminuir el número de campos, mediante el SGBD.

- La base de datos estará formada por muchas tablas, una por cada tipo de registro. En el ejemplo de la biblioteca podríamos definir otras tablas. Por ejemplo con los nombres de campo AUTOR, NACIONALIDAD, PROFESION.

- Dentro de una tabla cada nombre de campo debe ser distinto, por ejemplo en la de materias podría haber Autor 1, Autor 2: pero no puede haber dos campos con el nombre Autor, pues al referirnos con el gestor al nombre de campo AUTOR, no sabría cual utilizar.

- Los registros de una misma tabla tienen que diferenciarse, al menos, en el contenido de alguno de sus campos, no puede haber dos registros "idénticos".

- Los registros de una tabla pueden estar dispuestos en cualquier orden.

- El contenido de cada campo está delimitado por un rango de valores posibles. En el ejemplo del campo AÑO DE EDICIÓN no puede ponerse MIL o M, ni cualquier otro carácter alfabético, e incluso ningún año mayor que 2000, por no estar dentro del rango definido de los años posibles.

- Permite la creación de nuevas tablas a partir de las ya existentes, relacionando campos de distintas tablas anteriores. Esta condición es la esencia de las bases de datos relacionales, formando lo que se llama un fichero "virtual" (temporalmente en memoria).

El primer punto a abordar en la generación de las tablas es establecer una tabla "vacía". Esta tarea se realiza, bien mediante un asistente, o bien mediante el comando CREATE TABLE de SQL (lenguaje típico para el trabajo con bases de datos relacionales). Posteriormente, se pueden almacenar registros en la tabla, bien mediante asistentes proporcionados por las aplicaciones de base de datos o bien ejecutando la sentencia INSERT de SQL. Veremos cómo se utilizan los asistentes del SGBD de Microsoft Access en próximos capítulos, limitándonos aquí a citar su existencia y utilidad.

La sentencia CREATE TABLE crea un "objeto" (la tabla) dentro del sistema. Hay otros objetos importantes en el sistema, como los índices, que se establecen ejecutando la sentencia CREATE INDEX de SQL. La mayoría de los sistemas gestores de bases de datos permiten crear tablas e índices. Sin embargo, cada sistema contiene también otros objetos.

Existen diferencias en las versiones completas de los distintos SGBD, porque cada una tiene cláusulas diferentes que referencian objetos específicos del sistema. Afortunadamente, los usuarios y los programadores de aplicaciones no necesitan, más que en raras ocasiones, estar familiarizados con estos objetos, ya que existe una estandarización que hace familiar cualquier sistema gestor actual a cualquier usuario.

Los objetivos más importantes de la creación de tablas mediante asistentes o mediante la sentencia CREATE TABLE de SQL) podrían enumerarse como sigue:

- Establecer nuevas tablas en la base de datos y asignarles un nombre.

- Asignar nombre a todas las columnas de cada tabla y definir sus *tipos de datos*.

- Especificar la secuencia u orden de columnas por defecto.

- Especificar qué columnas no pueden aceptar *valores nulos* (valores que faltan, son desconocidos o no son aplicables). El valor nulo no debe confundirse con el valor cero, pues cero no es un valor nulo.

- Especificar la *clave primaria* (columna o combinación de columnas cuyos valores identifican unívocamente cada fila en la tabla, es decir, la clave primaria tiene un

valor único, diferente para cada fila de una tabla, de modo que no hay dos filas de una tabla con clave primaria que sean duplicados exactos la una de la otra).

- Especificar las *claves secundarias*, *externas o foráneas* (una columna de una tabla cuyo valor coincide con la clave primaria de alguna otra tabla de la base de datos se denomina una clave secundaria, externa o foránea, de modo que una tabla puede contener más de una clave secundaria si está relacionada con más de una tabla adicional de la base de datos). Definir las *relaciones* entre los campos de las tablas de la base de datos es una tarea fundamental.

- Especificar las *restricciones de integridad* para preservar la consistencia y corrección de los datos almacenados. Se especificarán los datos requeridos en los campos y registros y chequeos de validez para los tipos de datos a introducir en los campos.

- Especificar los *índices* para los campos o grupos de campos de las tablas de la base de datos. Los índices son estructuras internas que el sistema gestor de la base de datos puede utilizar para encontrar uno o más registros de una tabla de forma rápida.

1.5.4 Modelo de datos orientado a objetos

Las aplicaciones de las bases de datos en áreas como el diseño asistido por computadora, la ingeniería de software y el procesamiento de documentos no se ajustan al conjunto de suposiciones que se hacen en los modelos de datos vistos hasta ahora. El modelo de datos orientado a objetos se ha propuesto para tratar algunos de estos nuevos tipos de aplicaciones.

El modelo de bases de datos orientado a objetos se basa en el concepto de encapsulamiento de datos y código que opera sobre estos en un objeto. Los objetos estructurados se agrupan en clases. El conjunto de clases está estructurado en sub y superclases. Puesto que el valor de un dato en un objeto también es un objeto, es posible representar el contenido del objeto dando como resultado un objeto compuesto.

El modelo orientado a objetos se basa en encapsular código y datos en una única unidad, llamada objeto. El interfaz entre un objeto y el resto del sistema se define mediante un conjunto de mensajes.

Un objeto tiene asociado:

- Un conjunto de variables que contienen los datos del objeto. El valor de cada variable es un objeto.

- Un conjunto de mensajes a los que el objeto responde.

- Un método, que es un trozo de código para implementar cada mensaje. Un método devuelve un valor como respuesta al mensaje.

El término mensaje en un contexto orientado a objetos no implica el uso de un mensaje físico en una red de computadoras, sino que se refiere al paso de solicitudes entre objetos sin tener en cuenta detalles específicos de implementación. La capacidad de modificar la definición de un objeto sin afectar al resto del sistema está considerada como una de las mayores ventajas del modelo de programación orientado a objetos.

En una base de datos existen objetos que responden a los mismos mensajes, utilizan los mismos métodos y tienen variables del mismo nombre y tipo. Sería inútil definir cada uno de estos objetos por separado por lo tanto se agrupan los objetos similares para que formen una clase, a cada uno de estos objetos se le llama instancia de su clase. Todos los objetos de su clase comparten una definición común, aunque difieran en los valores asignados a las variables.

Así que básicamente las bases de datos orientadas a objetos tienen la finalidad de agrupar aquellos elementos que sean semejantes en las entidades para formar una clase, dejando por separado aquellas que no lo son en otra clase.

Las clases en un sistema orientado a objetos se representan en forma jerárquica, de modo que las propiedades o características de un elemento las contendrán o heredarán otros elementos, dando lugar así al concepto de *herencia*. Se pueden crear muchas agrupaciones (clases) para simplificar un modelo así que una jerarquía (en forma gráfica) puede quedar muy extensa, en estos casos tenemos que tener bien delimitados los elementos que intervienen en una clase y aquellos objetos que los heredan.

Los *lenguajes de programación orientados a objetos* requieren que toda la interacción con objetos se realice mediante el envío de mensajes. Así, un *lenguaje de consultas para un sistema de bases de datos orientado a objetos* debe incluir tanto el modelo de pasar el mensaje de objeto a objeto como el modelo de pasar el mensaje de conjunto en conjunto.

En una base de datos orientadas a objetos se pueden realizar adiciones de nuevas clases y eliminación de clases. Para realizar la adición de una nueva clase, la nueva clase debe colocarse en la jerarquía de clase o subclase cuidando las variables o métodos de herencia correspondientes. Para realizar la eliminación de una clase se requiere la realización de varias operaciones, debiendo cuidar los elementos que se han heredado de esa clase a otras y reestructurar la jerarquía.

La estructuración de modelos orientados a objetos simplifica una estructura evitando elementos o variables repetidas en diversas entidades, sin embargo el precio de esto es dedicarle un minucioso cuidado a las relaciones entre las clases cuando el modelo es complejo. La dificultad del manejo de objetos radica en la complejidad de las modificaciones y eliminaciones de clases, ya que de tener variables que heredan otros objetos se tiene que realizar una reestructuración que involucra una serie de pasos no triviales.