Introducción

La creciente necesidad de gestionar la información en toda clase de sistemas, desde una empresa que vende productos en su web hasta una gran multinacional con millones de clientes en todo el mundo, pasando por aplicaciones científicas que trabajan con experimentos con miles de parámetros y mediciones, ha potenciado el desarrollo desde hace más de cuatro décadas, tanto de distintas técnicas de diseño y modelado como de software de sistemas de bases de datos.

En la presente obra pretendemos cubrir con cierto detalle los distintos modelos de datos predominantes en el mercado, así como los sistemas de software de bases de datos que permiten su implementación física.

En primer lugar, veremos los sistemas de almacenamiento para estudiar después el modelo relacional como ejemplo de modelado que más se ha impuesto desde su creación en los años 70.

Después, pasaremos a ver cómo se tratan los datos utilizando un gestor o software de bases de datos bastante extendido y bien documentado, como es MySQL, y el lenguaje genérico (independiente del software) SQL, que permite llevar a cabo las distintas operaciones (consulta, inserción, modificación y borrado de datos).

Posteriormente, examinaremos en detalle el proceso clásico de desarrollo de bases de datos, desde su concepción mediante un modelo conceptual hasta su implementación en un sistema informático.

Por último, veremos un ejemplo de software de bases de datos avanzado que utiliza conceptos de la orientación a objetos. Se trata de Oracle, un sistema gestor de bases objeto-relacionales que incorpora conceptos de bases de datos y de objetos, tal como se definieron según el estándar SQL1999.

Todo ello con numerosos ejemplos y complementado con una serie de apéndices que amplían y completan los conceptos explicados.

1 Almacenamiento de la información

OBJETIVOS DEL CAPÍTULO

- ✓ Conocer características principales y tipos de ficheros.
- ✓ Entender el origen de las bases de datos como alternativa a los sistemas de ficheros.
- ✓ Describir los componentes y funciones principales de un Sistema Gestor de Bases de Datos (SGBD).
- ✓ Conocer las diferencias entre SGBD libres y comerciales.
- ✓ Introducir los conceptos relacionados con bases de datos distribuidas.

1.1 ALMACENAMIENTO DE LA INFORMACIÓN

Todas las aplicaciones informáticas trabajan en última instancia con datos o información que deben ser almacenados en un medio físico, como discos duros, memorias *flash* o DVD. Estos medios forman una jerarquía que distingue entre tres niveles de almacenamiento: primario, secundario e intermedio.

Almacenamiento primario

Se refiere a aquellos medios sobre los que la CPU del ordenador puede acceder directamente y, por tanto, más rápidamente. Son la memoria principal o memoria RAM y las memorias caché de primer y segundo nivel, más pequeñas pero más rápidas.

Almacenamiento secundario

Se refiere a dispositivos más lentos, pero de mayor capacidad, como los discos ópticos y magnéticos o las cintas. Para acceder a los datos la CPU debe copiarlos previamente en el almacenamiento primario.

El almacenamiento secundario de más amplio uso es el disco, aunque las cintas se usan sobre todo para copias de seguridad por su estabilidad, capacidad y durabilidad.

En un disco duro normalmente se agrupan varios discos ópticos, cada uno de los cuales se divide en pistas o círculos concéntricos. En ellas se almacena la información. La agrupación de pistas de todos los discos se denomina cilindro. Es importante que los datos a los que se suelen acceder simultáneamente estén en el mismo cilindro ya que se leen con mayor rapidez.

Cada pista, al contener gran cantidad de información, se subdivide en bloques o sectores de un tamaño fijo, determinado por el sistema operativo al inicializarlo con un sistema de archivos dado. Los bloques, también llamados páginas, suelen tener un tamaño entre 512 y 4.096 bytes.

La transferencia de información entre memoria y disco tiene lugar en unidades de bloque. Cuando produce una orden de lectura se copia uno o varios bloques en el llamado *buffer* de la memoria (área reservada de la memoria principal), si se necesita efectuar una escritura se copia el bloque correspondiente al bloque del disco.

Los discos son dispositivos de acceso aleatorio ya que podemos acceder a cualquier bloque de información solamente conociendo su dirección física en el disco, sin necesidad de recorrerlos todos.

Así mismo, las cintas son dispositivos de acceso secuencial, dado que los bloques se almacenan de manera contigua y para acceder a uno de ellos necesitamos leer todos los anteriores.

Almacenamiento intermedio

Cuando se necesita transferir varios bloques de disco a memoria principal y se conocen todas las direcciones de bloque es posible reservar varias áreas de almacenamiento intermedio o *buffers* dentro de la memoria principal para agilizar la transferencia. De este modo, mientras la CPU procesa datos de un *buffer* puede leer o escribir en otro. El uso de este tipo de almacenamiento es muy común en sistemas de bases de datos.

1.2 SISTEMAS DE ARCHIVOS

En esta sección estudiaremos someramente los conceptos relacionados con archivos tanto en lo relativo a su contenido como a su organización lógica y física.

Registros

La información se almacena en forma de registros, que son colecciones de valores o elementos de información relacionados, cada uno de los cuales corresponde a un campo del registro. Por ejemplo, un registro de alumno incluiría campos como el *nombre*, *fecha de nacimiento* o *teléfono*, cuyos valores para cada alumno forman cada registro. A su vez, cada campo tiene un tipo de dato que especifica el tipo de valores que puede tomar. Estos tipos suelen corresponderse con los de los lenguajes de programación. Así, en nuestro ejemplo tendríamos que para el nombre del alumno usaríamos un tipo carácter o *char*, para la fecha de nacimiento un tipo de fecha y para la edad un tipo numérico entero o *int*. Además, cada campo tiene un tamaño determinado en bytes que puede ser fijo o variable según los requisitos de la aplicación.

La unión de estos campos y sus tipos determina el tipo o formato del registro.

Archivos

Podemos definir un fichero informático como un conjunto de registros, grabados sobre un soporte que pueda ser leído por el ordenador.

Estos registros pueden tener longitud fija si todos los registros son iguales en tamaño, o variable si los registros son de distinto tipo o si, aun siendo iguales en formato tienen campos de tamaño variable u opcionales (campos que no necesariamente tienen un valor para cada registro, por ejemplo, teléfono en el tipo de registro de alumno podría ser un campo opcional).

Los archivos de registros de longitud fija son más fáciles de manipular por parte de los programas, sin embargo desperdician espacio en disco. Por el contrario, para el caso de longitud variable los programas son más complejos ya que los registros no ocupan posiciones fijas, sino que dependen del valor de cada campo.

Normalmente los registros de un archivo se asignan a uno o varios bloques en el sistema de almacenamiento para su manipulación por parte de los programas. El método de asignación elegido y el sistema de almacenamiento de los registros determinarán la eficiencia de este proceso.

Los ficheros son importantes porque son la unidad básica de información utilizada por cualquier programa, incluidos los sistemas gestores de bases de datos. Todos los datos son, en última instancia gestionados mediante ficheros mediante cuatro operaciones básicas: consulta o lectura, inserción, modificación y borrado. Cualquier operación más compleja (como búsqueda, ordenación, etc.) es combinación de dos o más operaciones básicas.

1.2.1 ORGANIZACIÓN PRIMARIA DE ARCHIVOS

El término organización de ficheros se aplica a la forma en que se colocan los datos contenidos en los registros de cada fichero sobre el soporte informático (disco, cinta, etc.) durante su grabación.

Existen dos formas básicas de organización de ficheros: **secuencial** y **relativa**. En la organización secuencial los registros se van grabando unos a continuación de los otros, en el orden que se van dando de alta, mientras que en la organización relativa los registros se graban en las posiciones que les corresponda según el valor que guarden en el campo denominado *clave*, que permite identificar el registro dentro del fichero.

Organización secuencial

Es el tipo más básico de organización. Los registros se colocan secuencialmente uno a continuación del otro y los registros nuevos se añaden al final del fichero.

La inserción es muy eficiente ya que siempre se hace en el último bloque disponible, sin embargo, la búsqueda de datos resulta más complicada ya que se requiere una búsqueda lineal, bloque por bloque hasta llegar al registro buscado.

En cuanto al borrado de datos es muy ineficiente ya que al eliminar registros y no ocupar el espacio liberado quedan huecos en los bloques que, salvo que se elimine el fichero, no serán reutilizados. Aunque hay técnicas para aprovechar estos espacios lo mejor es reorganizar el fichero periódicamente, de modo que se empaqueten los registros eliminando los borrados.

La modificación es más complicada ya que implica la búsqueda del registro y su reescritura. Esto es especialmente problemático en el caso de usar registros de longitud variable ya que puede ocurrir que el nuevo registro no quepa en el bloque, en cuyo caso se debe eliminar del bloque y añadirlo al final del fichero como si fuese una inserción.

Para el caso de archivos de registros de longitud fija el acceso a un registro por su posición dentro del archivo es muy sencillo ya que de este modo el *i-ésimo* registro se encontrará en el bloque resultado de obtener la parte entera de i/fbl (siendo *fbl* el factor de bloque o número de registros por bloque) y, dentro de ese bloque, será el registro número $i \bmod fbl$ (la función módulo *mod* devuelve el resto de dividir *i* entre *fbl*).

Con el fin de mejorar las prestaciones de la organización secuencial surgen una serie de organizaciones que son una variante de ésta y que pueden ser utilizadas con soportes direccionables. Las más empleadas son:

- **La organización secuencial indexada**

  Los registros con los datos se graban en un fichero secuencialmente, pero se pueden recuperar con acceso directo gracias a la utilización de un fichero adicional, llamado índice, que contiene información de la posición que ocupa cada registro en el fichero de datos.

- **La organización secuencial encadenada**

  Permite tener los registros ordenados según un orden lógico diferente del orden físico en el que están grabados gracias a la utilización de unos campos adicionales llamados punteros.

Organización Relativa

En este tipo de archivos los registros se graban en orden según el valor de uno de sus campos llamado *campo de ordenación*. Normalmente se usa un campo especial denominado *campo clave*, cuyos valores son distintos para cada registro.

En estos archivos la lectura es muy eficiente cuando se hace en orden según el campo de ordenación ya que el siguiente registro se encontrará a continuación del actual en el mismo bloque, o en el siguiente, si es el último. Por el mismo motivo las búsquedas son muy rápidas, siempre que la condición de búsqueda incluya el campo de ordenación ya que en tal caso pude usarse la técnica de búsqueda binaria.

Este sistema no ofrece ventajas cuando se trata de acceder a registros de manera aleatoria o de manera ordenada según un campo distinto al de ordenación, en cuyo caso deberemos usar un archivo adicional para ir almacenando los registros ordenados.

La inserción también es costosa, ya que debe mantenerse el orden, lo que puede implicar desplazamiento de registros para insertar uno nuevo en el orden apropiado. En cuanto a la eliminación es menos costosa si se usan registros marcados y se reorganiza el archivo periódicamente.

La modificación no ofrece problemas si no afecta al campo de ordenación, salvo que el registro sea de tamaño variable y no quepa en el bloque con los nuevos valores. Si se quiere modificar el campo de ordenación deberá reinsertarse en la posición correspondiente según el valor de dicho campo.

Dispersión

En este sistema se elige un campo llamado *campo de dispersión*. Al valor de ese campo se le aplica una función llamada *función de aleatorización* o de dispersión que, tomando como entrada dicho valor, devuelve un número que será la dirección del bloque de disco en que se almacenará el registro.

Las técnicas de dispersión o *hashing* se utilizan para acelerar el acceso a los registros cuando se busca un único registro según el llamado campo de dispersión. Normalmente este campo suele ser el *campo clave*.

Existen numerosas funciones de dispersión y su eficacia radica en que distribuyan los registros lo más posible minimizando el número de colisiones (producida cuando dos valores distintos del campo de dispersión producen el mismo valor de dispersión). Para estas situaciones se deben implementar medidas de corrección, como usar una segunda función, buscar la siguiente posición vacía dentro del bloque o usar técnicas de encadenamiento de registros.

1.2.2 MÉTODOS DE ACCESO

El método de acceso se refiere al procedimiento seguido para acceder a uno o más registros determinados de un fichero. Una de las operaciones más costosas y frecuentes es la búsqueda de información, por lo cual se usan sistemas que permiten mejorar su eficiencia. Estos sistemas se basan en el uso de **índices**, que son estructuras de datos que relacionan valores de un campo (normalmente el campo clave) de un registro con su dirección de memoria.

Hay varios tipos de índices y su uso dependiendo del tipo de organización que estemos usando.

Son similares a los índices analíticos de cualquier libro de texto, o sea, son como una lista en orden alfabético de los términos de un libro incluyendo la página o páginas en que se encuentra.

En el caso de ficheros se suele usar un campo como campo de indización. Así, los valores de ese campo junto con las direcciones de los bloques en que se encuentra se usan para crear el índice que, además, estará ordenado según el campo de indización. De este modo, las búsquedas serán mucho más rápidas al poder usar la técnica de búsqueda binaria. Al ser el archivo de índices mucho más pequeño que el de registros estas operaciones serán mucho más rápidas.

Hay varios tipos de índices. Por ejemplo, los *primarios* son índices sobre el campo clave de ordenación del fichero (campos cuyo valor no se repite en ningún otro registro) en ficheros ordenados físicamente por un campo.

En el caso en que el campo de ordenación no sea el campo clave, es decir, que pueda haber valores repetidos, hablamos de *índice de agrupamiento*.

Por último, cuando el campo de indización no es el de ordenación hablamos de *índice secundario*.

Veremos estos tres tipos de índices en la siguiente sección.

Índices primarios

Un índice primario es un archivo ordenado cuyos registros, de longitud fija, tienen dos campos, el campo clave de ordenamiento del fichero de datos y un apuntador a un bloque de disco. Por cada entrada o registro de índice hay un valor del campo clave y un apuntador al bloque que lo contiene. Así, cada entrada de índice apunta a un grupo de registros (bloque) del fichero de datos.

En la siguiente figura vemos un ejemplo gráfico en el que se ha indizado un fichero de registros con datos de jugadores de una liga de baloncesto, según el campo clave *id_jugador*, que identifica a cada jugador por un número secuencial.

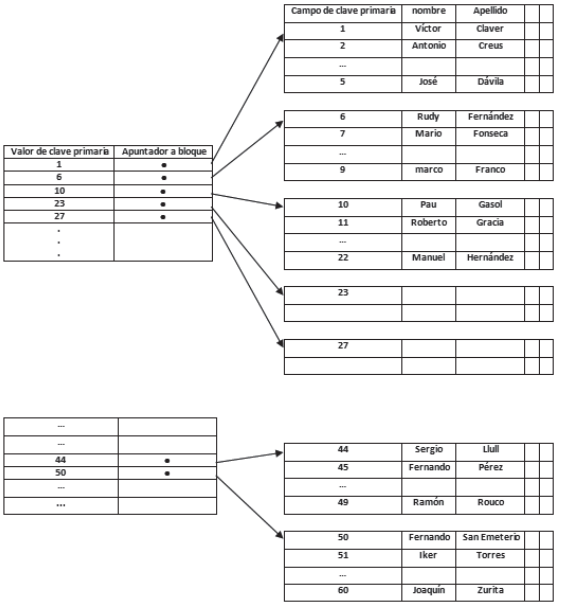

Figura 1.1. *Índice primario según el campo clave, que es también el de ordenación del archivo*

Se observa que cada registro del índice apunta a cada grupo de registros de datos de modo que no todos los identificadores aparecen en el índice, sino solamente los primeros de cada bloque.

Estos índices se consideran no densos, en el sentido de que cada entrada se asocia con varios registros de datos. En los índices densos esta relación es uno a uno.

Los ficheros de índices primarios son mucho más pequeños que los ficheros de datos a los que apuntan dado que solo contienen dos campos de datos y una entrada por bloque.

Índices de agrupamiento

Cuando los registros de un archivo están ordenados físicamente según un campo no clave (no tiene un valor distinto para cada registro de datos), este campo se denomina campo de agrupamiento. Podemos crear un índice sobre este campo llamado *índice de agrupamiento* para acelerar la obtención de registros con el mismo valor en dicho campo.

Este tipo de índice es un fichero formado por el campo de agrupamiento y un apuntador a bloque. Hay una entrada de índice por cada valor distinto del campo de agrupamiento y contiene un apuntador al primer bloque con ese valor en el campo.

Normalmente, se reservan bloques para cada valor distinto del campo de agrupamiento para facilitar las inserciones. Si se precisa más de un bloque para un mismo valor se van enlazando bloques adicionales mediante apuntadores de bloque.

En la figura siguiente observamos un ejemplo en el que se crea un índice de agrupamiento sobre un fichero de jugadores usando el campo *num_equipo* (número de equipo) como campo de agrupamiento.

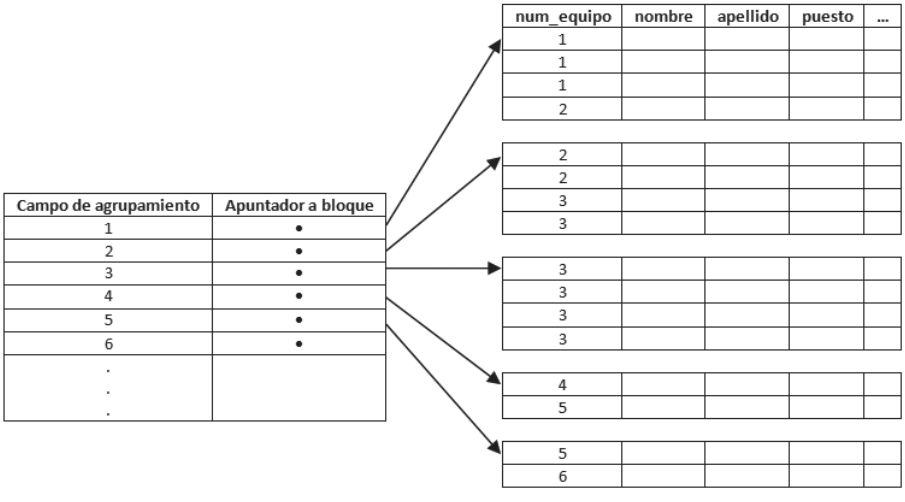

Figura 1.2. *Índice de agrupamiento según el campo de ordenamiento num_equipo*

En este caso el índice es sobre un campo que no es clave y se puede repetir, como es el número de equipo de los jugadores.

Hay una entrada de índice por cada equipo que apunta al bloque en el que se encuentra dicho equipo.

Índices secundarios

Son también archivos ordenados con dos campos. El primero es del mismo tipo que alguno de los campos clave distintos del ordenamiento del archivo de datos y, el segundo, es un apuntador a bloque. Puede haber varios índices secundarios sobre varios campos de un mismo archivo de datos.

Distinguimos entre índices secundarios sobre campos clave, en cuyo caso hablamos de claves secundarias y sobre campos no clave. En el primer caso hay una entrada de índice por cada registro de datos, con lo cual tenemos un índice denso.

En la siguiente figura observamos un ejemplo de este tipo de índices para el caso de una clave secundaria, como puede ser el DNI de un jugador.

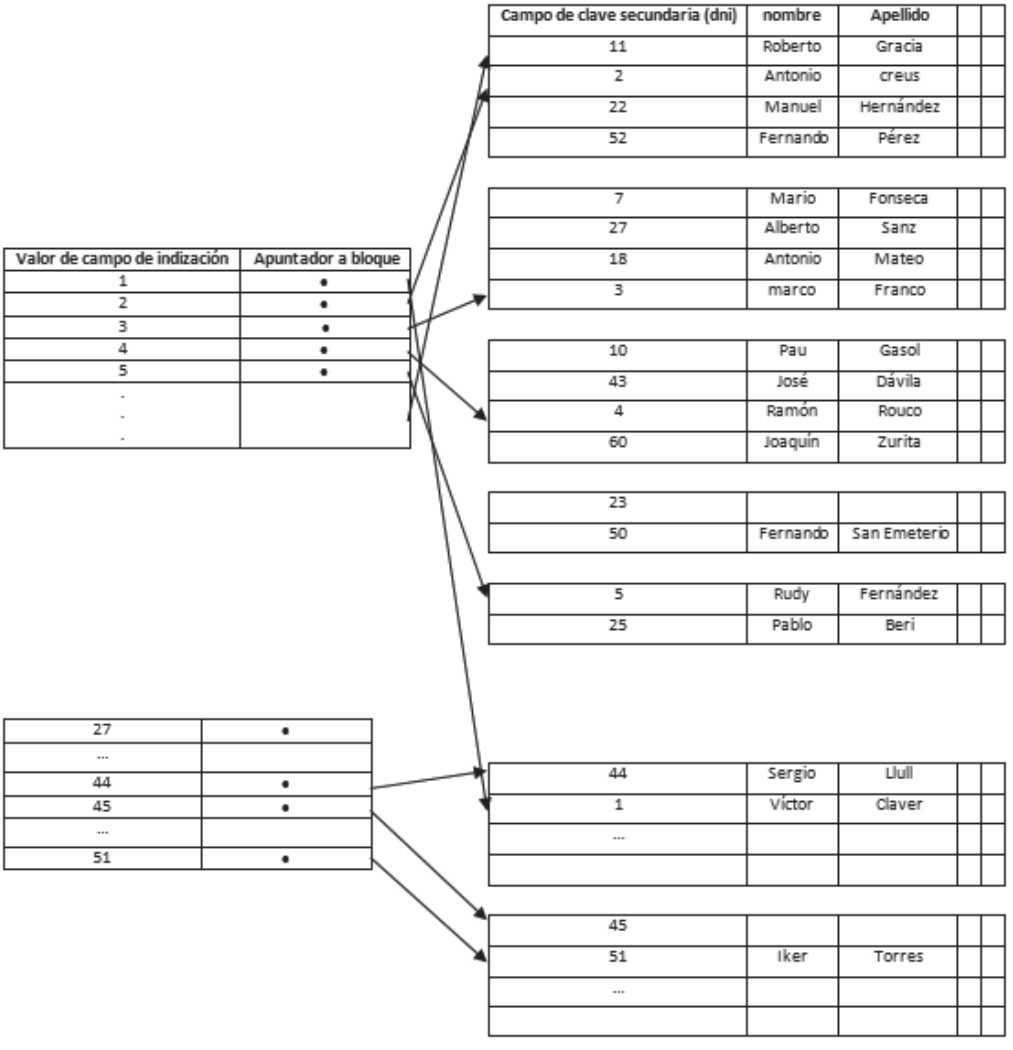

Figura 1.3. *Índice de agrupamiento según un campo clave que no determina el ordenamiento del archivo*

En general, este tipo de índices requieren de más espacio de almacenamiento debido al mayor número de entradas. Sin embargo, mejoran mucho las búsquedas debido a que, al estar ordenado, una búsqueda binaria reduce el tiempo de búsqueda respecto a una lineal en el archivo de datos que no está ordenado.

En las siguientes tablas mostramos un resumen de lo visto hasta ahora:

Tabla 1.1 Tipos de índices

	Campo de ordenación	Campo de no ordenación
Campo clave	Índice primario	Índice secundario
Campo no clave	Índice de agrupamiento	Índice secundario (no clave)

Tabla 1.2 Propiedades de los tipos de índices

	Número de entradas de índice	Denso/no denso
Primario	Número de bloques del archivo de datos	No denso
De agrupamiento	Número de valores distintos del campo índice	No denso
Secundario clave	Número de registros del archivo de datos	Denso
Secundario no clave	Número de registros o número de valores distintos del campo índice	Denso/no denso

En este último caso indicamos además si el índice es denso, es decir, si hay o no una correspondencia uno a uno por cada entrada de índice (una entrada por cada registro de datos).

Otros tipos de índice

Existen estructuras de índice más complejas que quedan fuera del alcance de este libro. A modo de ilustración comentaremos algunos de ellos.

- **Índices multinivel**

  Como hemos visto, una de las principales ventajas de los índices es la mejora sustancial de las búsquedas de registros de datos. Esta mejora se debe, principalmente, al hecho de que estén ordenados y se pueda hacer una búsqueda binaria. Estas búsquedas tienen un coste de $log_2 b$ accesos a bloque para localizar un registro. Sin embargo, se puede mejorar todavía más restringiendo el número de bloques de índice a leer usando para ello un archivo de índice sobre el fichero de índices inicial, creando así uno de dos niveles. Esto se puede extender a más niveles dando lugar a índices multinivel. En ellos el número de accesos es de $log_n b$, donde n es el factor de bloques (número de registros por bloque) del índice.

- **Arboles B y B+**

  Son estructuras de árbol muy utilizadas en programación y en índices de bases de datos. En este caso, un árbol es un fichero en el que hay unos nodos (padres) que apuntan a otros (hijos), los cuales a su vez tienen apuntadores a otros nodos y, así sucesivamente, con tantos nodos como entradas de índice requeridas.

  Los nodos hijo no pueden apuntar a nodos padre, de modo que se forma una estructura con un nodo raíz sin padre cuyos nodos hijos son padres de otros hasta llegar a los nodos hoja, o nodos sin hijos.

  Estos nodos almacenan apuntadores a nodos hijo, apuntadores a registros de datos y valores de campos de los registros de datos y la estructura que forman hace muy eficiente la búsqueda de datos.

- **Índices hash**

  Es posible crear estructuras de acceso de tipo índice usando técnicas de dispersión vistas en la sección anterior. En este caso, cada entrada de índice se organiza como un archivo de dispersión y contiene registros formados por el valor de dispersión y un apuntador al bloque de datos. Así, el acceso a una de ellas requiere aplicar la función *hash* correspondiente al valor buscado y obtener así el valor del apuntador.

- **Índices lógicos**

  Hasta ahora hemos supuesto que las entradas de índice contienen un valor lógico (el valor del campo) y un valor físico, o apuntador, que especifica la dirección física del registro o bloque de datos. Esto tiene la desventaja de que si los registros de datos cambian con frecuencia de lugar físico debe actualizarse el índice en consonancia, lo que puede resultar muy costoso en términos computacionales.

Para solucionar esto se usan índices lógicos en los que cada entrada está formada por un valor de indización secundario y el valor del campo clave, que determina la organización primaria del archivo. De este modo la búsqueda de un registro de datos, según un valor de índice secundario, permitirá obtener el valor correspondiente al campo de ordenación del archivo y realizar la búsqueda directamente en el archivo de datos.

ACTIVIDADES 1.1

- Investigue y explique con sus palabras en qué consiste el método de búsqueda binaria en ficheros.
- Realice un ejemplo de búsqueda secuencial y binaria en clase suponiendo que tiene que acceder a un valor dentro de un conjunto ordenado de valores. Compute y compare el número de lecturas en ambos procesos para varios valores de búsqueda.
- ¿Cuántos índices primarios y de agrupamiento puede tener un fichero ordenado?
- Comente ventajas e inconvenientes respecto a la actualización de datos en ficheros con organización tipo *hash.*
- Si tenemos un archivo de datos de 2.000 jugadores con tamaño fijo de 80 bytes y un disco de tamaño de bloque igual a 1.024 bytes, determine el número de bloques requerido y el coste de una búsqueda binaria en cuanto a número necesario de accesos a bloques para encontrar un registro de datos.
- Suponga que en el ejercicio anterior creamos un índice formado por la clave primaria (5 bytes) y un apuntador de 4 bytes. ¿Cuántas entradas de índice tendremos? ¿Cuántos accesos a bloques de disco necesitaremos ahora para efectuar una búsqueda binaria?
- ¿Qué problemas observa al usar ficheros de índices primarios en ficheros ordenados, respecto a la inserción y eliminación de registros?
- Investigue la diferencia entre una estructura de índice tipo árbol B y B+.

1.3 SISTEMAS DE BASES DE DATOS

Inicialmente, cuando las primeras empresas y organizaciones empezaron a usar sistemas informáticos trabajaban con sistemas de ficheros. Es decir, se trabajaba con programas que manejaban información almacenada en ficheros. Cada equipo trabajaba con sus propios datos y programas y se encargaba de su mantenimiento y gestión. Al principio el sistema funcionó pero con el tiempo y, sobre todo, con el incremento de la cantidad de información así como de los usuarios que la manejaban surgieron problemas (integridad y duplicidad de información, seguridad, etc., que llevaron finalmente a la organización de la información mediante un sistema más ordenado y manejable basado en la centralización de la gestión y la organización de los datos en forma de bases de datos.

Los principales problemas relacionados con el uso de ficheros como sistema de almacenamiento de información fueron los siguientes:

No debemos olvidar que en última instancia todo se almacena en ficheros. La diferencia es que cuando usamos bases de datos no trabajamos directamente con ficheros, sino con estructuras de datos más fáciles de manejar.

- **Separación y aislamiento de los datos**

  Cuando los datos se separan en distintos ficheros es más complicado acceder a ellos, ya que el programador de aplicaciones debe sincronizar el procesamiento de los distintos ficheros implicados para asegurar que se extraen los datos correctos.

- **Duplicación de datos**

  La redundancia de datos existente en los sistemas de ficheros hace que se desperdicie espacio de almacenamiento y, lo que es más importante, puede llevar a que se pierda la consistencia de los datos. Se produce una inconsistencia cuando copias de los mismos datos no coinciden.

- **Dependencia de datos**

  Ya que la estructura física de los datos (la definición de los ficheros y de los registros) se encuentra codificada en los programas de aplicación, cualquier cambio en dicha estructura es difícil de realizar. El programador debe identificar todos los programas afectados por este cambio, modificarlos y volverlos a probar, lo que cuesta mucho tiempo y está sujeto a que se produzcan errores. A este problema, tan característico de los sistemas de ficheros, se le denomina también falta de independencia de datos lógica-física.

- **Formatos de ficheros incompatibles**

  Ya que la estructura de los ficheros se define en los programas de aplicación, es completamente dependiente del lenguaje de programación. La incompatibilidad entre ficheros generados por distintos lenguajes hace que los ficheros sean difíciles de procesar de modo conjunto.

- **Consultas fijas y proliferación de programas de aplicación**

  Desde el punto de vista de los usuarios finales, los sistemas de ficheros fueron un gran avance comparados a los sistemas manuales. A consecuencia de esto, creció la necesidad de realizar distintos tipos de consultas de datos. Sin embargo, los sistemas de ficheros son muy dependientes del programador de aplicaciones: cualquier consulta o informe que se quiera realizar debe ser programado por él. En algunas organizaciones se conformaron con fijar el tipo de consultas e informes, siendo imposible realizar otro tipo de consultas que no se hubieran tenido en cuenta a la hora de escribir los programas de aplicación.

- **Control de concurrencia**

  El acceso de varios clientes al mismo fichero genera inconsistencias, ya que un cliente puede consultar un fichero mientras otro lo está modificando.

- **Autorizaciones**

  Los errores en los permisos de ficheros pueden hacer que un mismo cliente pueda modificar un dato en un fichero y no en otro en el que esté repetido.

- **Catálogo**

  Resulta complicado saber dónde están los distintos datos. No existe un esquema general que muestre la organización de la información.

  En otras organizaciones hubo una proliferación de programas de aplicación para resolver todo tipo de consultas, hasta el punto de desbordar al equipo de proceso de datos, que no daba abasto para validar, mantener y documentar dichos programas.

  Para trabajar de un modo más efectivo, surgieron las bases de datos como modelo de organización de los datos y, con ellas, los sistemas de gestión de bases de datos (SGBD) como herramienta que permite la implementación y gestión de bases de datos.

Definición

Una base de datos es un conjunto de datos almacenados entre los que existen relaciones lógicas y ha sido diseñada para satisfacer los requerimientos de información de una empresa u organización.

La base de datos es un conjunto de datos organizados en estructuras que se definen una sola vez y que se utilizan al mismo tiempo por muchos equipos y usuarios. En lugar de almacenarse en ficheros desconectados y de manera redundante, los datos en una base de datos están centralizados y organizados, de forma que se minimice la redundancia y se facilite su gestión. La base de datos no pertenece a un equipo, se comparte por toda la organización. Además, la base de datos no solo contiene los datos de la organización, también almacena una descripción de dichos datos. Esta descripción es lo que se denomina *metadatos*, se almacena en el diccionario de datos o catálogo que, en muchos casos, se organiza en otra base de datos.

1.3.1 ARQUITECTURA DE SISTEMAS DE BASES DE DATOS

En 1975, el comité ANSI-SPARC (*American National Standard Institute - Standards Planning and Requirements Committee*) propuso un estándar para la creación de sistemas de bases de datos basado en una arquitectura de tres niveles, que resulta muy útil a la hora de conseguir estas tres características:

El objetivo de la arquitectura de tres niveles es el de separar en niveles de abstracción el esquema de una base de datos. Son tres formas distintas de ver o representar una misma base de datos.

Nivel interno

Se describe la estructura física de la base de datos mediante un esquema interno. Este esquema describe todos los detalles para el almacenamiento de la base de datos, así como los métodos de acceso. Se habla de ficheros, discos, directorios, etc.

Nivel global

Se describe la estructura de toda la base de datos para una comunidad de usuarios (todos los de una empresa u organización) mediante un esquema conceptual. Este esquema oculta los detalles de las estructuras de almacenamiento y se concentra en describir entidades, atributos, relaciones (tablas) y restricciones.

Nivel externo

Se describen varios esquemas externos o vistas de usuario. Cada esquema externo describe la parte de la base de datos que interesa a un grupo de usuarios determinados y oculta a ese grupo el resto de la base de datos. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar los esquemas. Ese es el que percibe el usuario final mediante el uso de aplicaciones. Por ejemplo, cuando accedo a la página web de la liga de baloncesto estoy consultando una parte de los datos de sus bases de datos. Son lo que denominamos *vistas*.

Conviene recalcar que los tres niveles no son más que descripciones de los mismos datos, pero en distintos niveles de abstracción. Los únicos datos que existen realmente están a nivel físico, almacenados en un dispositivo, como puede ser un disco.

La arquitectura de tres niveles es útil para explicar el concepto de independencia de datos, que podemos definir como *la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior*. Se pueden definir dos tipos de independencia de datos:

- **La independencia lógica**

  Es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la base de datos o para reducirla. Si, por ejemplo, se reduce la base de datos eliminando una entidad, los esquemas externos que no se refieran a ella no deberán verse afectados.

- **La independencia física**

  Es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual (o los externos). Por ejemplo, puede ser necesario reorganizar ciertos ficheros físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización de datos. Dado que la independencia física se refiere solo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento, es más fácil de conseguir que la independencia lógica.

1.3.2 MODELOS DE DATOS

La base de datos consiste entonces en los datos concretos referentes a un sistema o parte del mundo que hemos modelado (por ejemplo, nuestra base de datos de la liga de baloncesto incluye todos los datos relativos a jugadores, equipos, partidos, etc.). Estos datos son sencillos de manejar cuando son unos pocos, pero cuando su volumen crece se requiere el uso de distintos modelos para facilitar el diseño de las mismas (si solo necesitamos registrar los nombres de jugadores y equipos no es necesario recurrir a ningún modelo).

Existen muchos modelos de distinto tipo para tal fin. A continuación los describiremos comenzando por su definición.

Definición

Un modelo de datos es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos y sus restricciones.

Un modelo nos proporciona mecanismos de abstracción para representar una parte del mundo cuyos datos nos interesan. Dicha representación, realizada en términos de un modelo dado, recibe el nombre de esquema y el conjunto de datos que representa es la base de datos.

1.3.3 TIPOS DE MODELOS

La arquitectura de tres niveles nos obliga a modelar nuestros datos en cada nivel. En este libro nos centraremos en los dos primeros (global y externo), ya que los del nivel interno son específicos del software utilizado.

En este sentido distinguimos tres tipos de modelos:

- ✓ Modelos conceptuales.
- ✓ Modelos lógicos tradicionales.
- ✓ Modelos lógicos avanzados.

Modelos conceptuales

Se usan para describir datos en el nivel global. Con este modelo representamos los datos de forma parecida a como nosotros los captamos en el mundo real. Este tipo de modelos tienen una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Existen diferentes modelos de este tipo, pero el más utilizado por su sencillez y eficiencia es el modelo Entidad-Relación.

Denominado por sus siglas (y a partir de ahora en este libro) como MER, este modelo representa la realidad a través de entidades, que son objetos que existen y que se distinguen de otros por sus características. Por ejemplo, un jugador es una **entidad** con características como su altura, nombre, etc.

Estas características de las entidades en base de datos se llaman **atributos**. A su vez, una entidad se puede asociar o relacionar con más entidades a través de **relaciones**. Así un jugador está vinculado o relacionado con un equipo en virtud de la relación pertenece (jugador pertenece a equipo).

Este tipo de modelos utiliza una simbología para representar cada elemento. Lo estudiaremos en detalle en el Capítulo 6.

Modelos lógicos tradicionales

Fueron lo primeros en usarse aunque su uso hoy en día, especialmente el relacional, está muy extendido.

Se utilizan para describir datos en el nivel global, pero de un modo más lógico (más cercano a la máquina).

Estos modelos utilizan tablas de registros para representar los objetos modelados y sus relaciones. A diferencia de los modelos de datos conceptuales, se usan para especificar la estructura lógica global de las bases de datos y para proporcionar una descripción más estructurada y cercana a la implementación.

Los tres modelos de datos más ampliamente aceptados son:

- ✓ Modelo Relacional.
- ✓ Modelo de Red.
- ✓ Modelo Jerárquico.

■ **Modelo relacional**

En este modelo se representan los datos y las relaciones entre estos, a través de una colección de tablas, en las cuales las filas (*tuplas*) equivalen a cada uno de los registros que contendrá la base de datos y las columnas corresponden a las características (atributos) de cada registro localizado en la *tupla*.

También sirven para representar el nivel externo (vistas) de una base de datos.

■ **Modelo de red**

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

■ **Modelo jerárquico**

Éstas son modelos de bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado *raíz*, y a los nodos que no tienen hijos se les conoce como nodos *hoja*.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos, permitiendo crear estructuras estables y de gran rendimiento a la hora de acceder a los mismos.

Es similar al modelo de red en cuanto a las relaciones y datos, ya que estos se representan por medio de registros vínculos entre los mismos. La diferencia radica en que están organizados por gráficos de tipo árbol en lugar de gráficos arbitrarios.

Modelos lógicos avanzados

Son modelos de datos relativamente recientes y cada vez más utilizados, sobre todo en aplicaciones específicas que manejan nuevos y más complejos tipos de datos.

✓ Modelos de datos orientados a objetos

Estos modelos son utilizados, sobre todo, en aplicaciones programadas bajo el paradigma de la orientación a objetos. Tratan de almacenar en la base de datos no solo los datos (estado), sino también la funcionalidad asociada (comportamiento). De este modo, una base de datos está formada por objetos relacionados entre sí, siendo los objetos entidades con un estado o datos asociados y un comportamiento o funcionalidad determinada.

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos como son:

- **Encapsulación**: propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- **Herencia**: propiedad a través de la cual los objetos o tablas heredan atributos y métodos (comportamiento) de otros situados en un nivel superior según una jerarquía de clases.
- **Polimorfismo**: hace referencia a métodos que pueden aplicarse a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada *función*) se especifica en dos partes. La interfaz (o *signatura*) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o *parámetros*). La implementación (o *método*) de la operación se especifica separadamente y puede modificarse sin afectar a la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

✓ Modelos de datos declarativos

Estos modelos se dividen en *deductivos* y *funcionales*.

Suelen usarse para bases de conocimiento, que no son más que bases de datos con mecanismos de consulta en los que el trabajo de extracción de información a partir de los datos recae en realidad sobre el sistema informático, en lugar de sobre el usuario. Estos mecanismos de consulta exigen que la información esté distribuida de manera que haga eficiente las búsquedas de los datos, ya que normalmente las consultas de este tipo requieren acceder una y otra vez a los datos en busca de patrones que se adecúen a las características de los datos que ha solicitado el usuario.

1.4 SISTEMAS GESTORES DE BASES DE DATOS

Los modelos nos permiten representar nuestra información de un modo sencillo y en un lenguaje común.

Sin embargo, necesitamos un software que nos permita llevarlo a cabo o implementar dichos modelos.

Este software es lo que se denominaremos SGBD. Lo definiremos a continuación.

1.4.1 DEFINICIÓN Y OBJETIVOS

Definición

El sistema de gestión de la base de datos (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener la base de datos y proporciona acceso controlado a la misma. Es una herramienta que sirve de interfaz entre el usuario y las bases de datos.

En el siguiente esquema se representa el funcionamiento de un sistema de información en el que los usuarios acceden a la información usando aplicaciones (por ejemplo, un formulario web) que, a su vez, se comunican con sistemas gestores, que son los que en última instancia acceden a los datos almacenados en las bases de datos mediante la interacción con el sistema operativo:

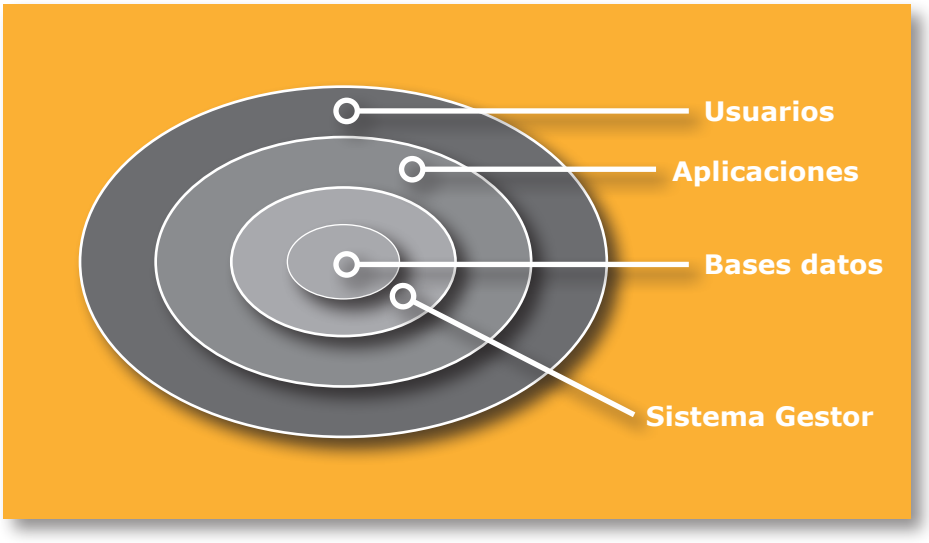

Figura 1.4. *Sistema gestor como interfaz entre usuarios y bases de datos*

Como ya hemos comentado, el organismo ANSI estableció los tres niveles de abstracción (*externo*, *lógico* y *físico*) como requisito en los sistemas gestores, tal y como queda reflejado en la siguiente figura:

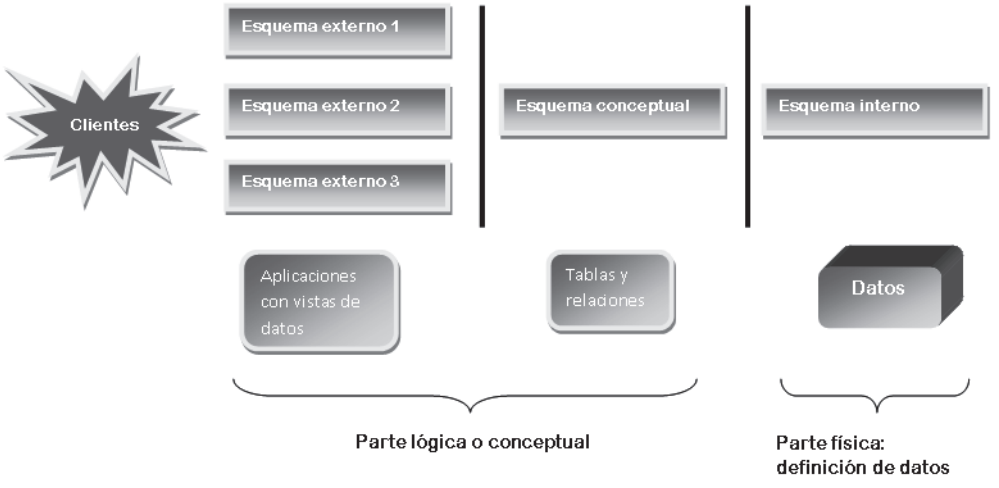

Figura 1.5. *Esquema de los niveles de abstracción en un SGBD*

Se observa como en el nivel externo se encuentran los usuarios finales que tienen acceso a distintos esquemas, los cuales a su vez se derivan del esquema o representación lógica que, a su vez, se traduce a un esquema físico o forma de almacenamiento de los datos.

Objetivos

Los objetivos de un SGBD se pueden resumir en los siguientes:

- Asegurar los tres niveles de abstracción: *físico*, *lógico* y *externo*.
- Permitir la independencia física y lógica de los datos.
- Garantizar la consistencia de los datos, ya que puede haber datos duplicados o derivados que deben mantener sus valores de forma coherente.
- Ofrecer seguridad de acceso a los datos por parte de usuarios y grupos.
- Gestión de transacciones de forma que se garantice la ejecución de un conjunto de operaciones críticas como una sola operación.
- Permitir la concurrencia de usuarios sobre los mismos datos mediante bloqueos que mantienen la integridad de los mismos.

1.4.2 FUNCIONES DEL SISTEMA GESTOR DE BASE DE DATOS (SGBD)

Para la consecución de los objetivos comentados en la sección anterior la mayoría de SGBD comerciales y libres incorporan las siguientes características y funciones:

Un catálogo

Donde se almacenen las descripciones de los datos y sea accesible por los usuarios. Este catálogo es lo que se denomina *diccionario de datos* y contiene información que describe los datos de la base de datos (*metadatos*). Normalmente, un diccionario de datos describe entre otras cosas:

- Nombre, tipo y tamaño de los datos.
- Relaciones entre los datos.
- Restricciones de integridad sobre los datos.
- Usuarios autorizados a acceder a los objetos de base de datos.
- Estadísticas de utilización, tales como la frecuencia de las transacciones y el número de accesos realizados a los objetos de la base de datos.

Garantizar la integridad

Disponer de un mecanismo que garantice que todas las actualizaciones correspondientes a una determinada transacción se realicen o que no se realice ninguna. Una **transacción** es un conjunto de acciones que cambian el contenido de la base de datos. Una transacción en el sistema informático de la empresa inmobiliaria sería dar de alta a un jugador o eliminar un inmueble. Una transacción un poco más complicada sería eliminar un jugador y reasignar sus inmuebles a otro jugador. En este caso hay que realizar varios cambios sobre la base de datos. Si la transacción falla durante su realización, por ejemplo, porque falla el hardware, la base de datos quedará en un estado inconsistente. Algunos de los cambios se habrán hecho y otros no, por tanto, los cambios realizados deberán ser deshechos para devolver la base de datos a un estado consistente.

Permitir actualizaciones

Asegurar que la base de datos se actualice correctamente cuando varios usuarios la están actualizando concurrentemente. Uno de los principales objetivos de los SGBD es el de permitir que varios usuarios tengan acceso concurrente a los datos que comparten. El acceso concurrente es relativamente fácil de gestionar si todos los usuarios se dedican a leer datos, ya que no pueden interferir unos con otros. Sin embargo, cuando dos o más usuarios están accediendo a la base de datos y al menos uno de ellos está actualizando datos, pueden interferir produciendo inconsistencias en la base de datos. El SGBD debe garantizar que no se produzcan, es decir, que los datos estén en todo momento en un estado consistente.

Recuperación de datos

Permitir recuperar las bases de datos en caso de que ocurra algún suceso imprevisto que afecte o destruya la base de datos. Como se ha comentado antes, cuando el sistema falla en medio de una transacción, la base de datos se debe devolver a un estado consistente. Esta falta puede ser a causa de un fallo en algún dispositivo hardware o un error del software, que hagan que el SGBD aborte, o puede ser a causa de que el usuario detecte un error durante la transacción y la aborte antes de que finalice. También pude ser que simplemente se pierdan los datos por cualquier motivo.

En todos estos casos, el SGBD debe proporcionar mecanismos capaces de recuperar y llevar la base de datos consistente lo más cercano posible en el tiempo al momento del fallo.

Integración

Ser capaz de integrarse con algún software de comunicación. Muchos usuarios acceden a la base de datos desde terminales. Algunas veces estos terminales se encuentran conectados directamente a la máquina sobre la que funciona el SGBD. En otras ocasiones, los terminales están en lugares remotos, por lo que la comunicación con la máquina que alberga al SGBD se debe hacer a través de una red. En cualquiera de los dos casos, el SGBD recibe peticiones en forma de mensajes y responde de modo similar. Todas estas transmisiones de mensajes las maneja el gestor de comunicaciones de datos. Aunque este gestor no forma parte del SGBD, es necesario que el SGBD se pueda integrar con él para que el sistema sea comercialmente viable.

Cumplir restricciones

Proporcionar los medios necesarios para garantizar que, tanto los datos de la base de datos como los cambios que se realizan sobre estos datos, sigan ciertas reglas. Se puede considerar como otro modo de proteger la base de datos pero, además de tener que ver con la seguridad, tiene otras implicaciones.

La integridad se ocupa de la calidad de los datos. Normalmente se expresa mediante restricciones, que son una serie de reglas que la base de datos no puede violar. Por ejemplo, en una base de datos de una liga de baloncesto se puede establecer la restricción de que cada equipo no puede tener asignados más de veinte jugadores. En este caso sería deseable que el SGBD controlara que no se sobrepase este límite cada vez que se asigne un jugador a un equipo.

Herramientas de administración

Proporcionar herramientas que permitan administrar la base de datos de modo efectivo, lo que implica un diseño óptimo de las mismas, garantizar la disponibilidad e integridad de los datos, controlar el acceso al servidor y a los datos, monitorizar el funcionamiento del servidor y optimizar su funcionamiento. Muchas de ellas van integradas en el sistema gestor, otras son creadas por terceros o por el propio administrador según sus requerimientos.

1.4.3 COMPONENTES DE UN SGBD

Son los elementos que deben proporcionar los servicios comentados en la sección anterior. No se pueden generalizar ya que varían mucho según la tecnología. Sin embargo, normalmente todo SGBD incluye los siguientes:

Lenguajes de datos

Son lenguajes para la manipulación de datos, tanto desde el punto de vista de su acceso y modificación como del control y seguridad de los mismos. Normalmente se distinguen tres tipos según su funcionalidad.

- **Lenguaje de definición de datos** (**DDL**, *Data Definition Language*)

  Sencillo lenguaje artificial para definir y describir los objetos de la base de datos, su estructura, relaciones y restricciones.

  Permite, entre otras cosas, la creación, eliminación y modificación de las estructuras de la base de datos, es decir, de la definición de las tablas, así como de índices y restricciones.

- **Lenguaje de control de datos** (**DCL**, *Data Control Language*)

  Encargado del control y seguridad de los datos (privilegios y modos de acceso, etc.).

  Este lenguaje permite especificar los permisos sobre los objetos de las bases de datos (tablas, vistas, procedimientos, etc.) así como la creación y eliminación de usuarios y cuentas.

- **Lenguaje de manipulación de datos** (**DML**, *Data Manipulation Language*)

  Es el lenguaje encargado de la manipulación del contenido de las bases de datos.

  Permite la inserción, actualización, eliminación y consulta de datos en las tablas de las bases de datos.

Para todos estos lenguajes se usa principalmente el lenguaje SQL (*Structured Query Language*). Incluye instrucciones para los tres tipos de lenguajes comentados y por su sencillez y potencia se ha convertido en el lenguaje estándar de los **SGBD relacionales**.

Diccionario de datos

Esquemas que describen el contenido del **SGBD** incluyendo los distintos objetos con sus propiedades.

Objetos

- Tablas base y vistas (tablas derivadas).
- Consultas.
- Dominios y tipos definidos de datos.
- Restricciones de tabla y dominio y aserciones.
- Funciones y procedimientos almacenados.
- Disparadores o *triggers*.

Herramientas para...

- **Seguridad**: de modo que los usuarios no autorizados no puedan acceder a la base de datos.
- **Integridad**: que mantiene la integridad y la consistencia de los datos.
- **El control de concurrencia**: que permite el acceso compartido a la base de datos.
- **El control de recuperación**: que restablece la base de datos después de que se produzca un fallo del hardware o del software.
- **Gestión del diccionario de datos** (o catálogo): accesible por el usuario que contiene la descripción de los datos de la base de datos.
- **Programación de aplicaciones**.
- **Importación/exportación de datos** (migraciones).
- **Distribución de datos**.
- **Replicación** (arquitectura maestro-esclavo).
- **Sincronización** (de equipos replicados).

Optimizador de consultas

Para determinar la estrategia óptima para la ejecución de las consultas.

Gestión de transacciones

Este módulo realiza el procesamiento de las transacciones.

Planificador (*scheduler*)

Para programar y automatizar la realización de ciertas operaciones y procesos.

Copias de seguridad

Para garantizar que la base de datos se puede devolver a un estado consistente en caso de que se produzca algún fallo o error grave.

No todos los SGBD presentan la misma funcionalidad, depende de cada producto. En general, los grandes SGBD multiusuario ofrecen todas las funciones que se acaban de citar y muchas más. Los sistemas modernos son conjuntos de programas extremadamente complejos y sofisticados, con millones de líneas de código y con una documentación consistente en varios volúmenes. Los SGBD están en continua evolución, tratando de satisfacer los requerimientos de todo tipo de usuarios. Desde permitir el trabajo con nuevos objetos multimedia a sistemas de minería de datos capaces de detectar patrones de datos, pasando por sistemas de datos distribuidos entre múltiples equipos. Sin duda, a medida que pase el tiempo irán surgiendo nuevos requisitos que serán incorporados paulatinamente.

1.4.4 USUARIOS DE LOS SGBD

Generalmente, distinguimos cuatro grupos de usuarios de sistemas gestores de bases de datos: los usuarios administradores, los diseñadores de la base de datos, los programadores y los usuarios de aplicaciones que interactúan con las bases de datos.

Administradores

Trabajan en el nivel de abstracción físico relacionado con el almacenamiento.

Distinguimos los administradores del propio sistema gestor encargados de la instalación y configuración del sistema, del control de acceso a los recursos, de la seguridad y de la monitorización y optimización del sistema gestor.

Por su parte, los administradores de bases de datos se encargan del diseño físico de la misma, implementación y mantenimiento de la base de datos.

Diseñadores de la base de datos

Realizan el diseño lógico de la base de datos, debiendo identificar los datos, las relaciones entre datos y las restricciones sobre los datos y sus relaciones. El diseñador de la base de datos debe tener un profundo conocimiento de los datos de la empresa y también debe conocer sus reglas de negocio. Las reglas de negocio describen las características principales de los datos tal y como los ve la empresa. Para obtener un buen resultado, el diseñador de la base de datos debe implicar en el desarrollo del modelo de datos a todos los usuarios de la base de datos, tan pronto como sea posible. El diseño lógico de la base de datos es independiente del SGBD concreto que se vaya a utilizar, es independiente de los programas de aplicación, de los lenguajes de programación y de cualquier otra consideración física.

Programadores

Tanto de aplicaciones que, mediante API de lenguajes de programación interactúan con las bases de datos como de objetos de la base de datos, como rutinas almacenadas o disparadores. Estas aplicaciones servirán a los usuarios finales para, de una forma amigable, poder consultar datos, insertarlos, actualizarlos y eliminarlos.

Usuarios finales

Trabajan en el nivel externo mediante vistas o porciones de las bases de datos. Son clientes de las bases de datos que hacen uso de ellas sin conocer en absoluto su funcionamiento y organización interna. Son personas con pocos o nulos conocimientos de informática.

1.4.5 MODELO ANSI/X3/SPARC

El organismo ANSI ha marcado la referencia para la construcción de SGBD. El modelo definido por el grupo de trabajo SPARC se basa en estudios anteriores en los que se definían los tres niveles de abstracción necesarios para describir una base de datos. ANSI profundiza más en esta idea y define cómo debe ser el proceso de creación y utilización de estos niveles.

En el modelo ANSI se indica que hay tres distintos tipos: *externo*, *conceptual* e *interno*.

Los esquemas externos reflejan la información preparada (filtrada en forma de vistas) para el usuario final, el esquema conceptual refleja los datos y relaciones de la base de datos (el nivel lógico) y el esquema interno determina la organización física de los datos (sistemas, de ficheros, estructura de directorios, espacios de almacenamiento, índices, etc.).

Por decirlo de una manera más cercana al "recién llegado", podemos hacer un paralelismo entre una casa y un SGBD. En este caso el nivel externo sería lo que vemos nosotros como usuarios de la casa (enchufes, luces, paredes, etc.). El nivel lógico o conceptual correspondería a los planos detallados de la casa y, el nivel físico, a cómo están hechas realmente las obras según el terreno, orientación y otros condicionantes físicos.

En definitiva, el modelo ANSI es una propuesta teórica sobre las características deseables en un sistema gestor de bases de datos.

1.4.6 TIPOS DE SGBD

Existen numerosos SGBD en el mercado que podemos clasificar según los siguientes criterios:

- **Modelo lógico en el que se basan**
  - Jerárquico.
  - En red.
  - Relacional.
  - Objeto-relacional.
  - Orientado a objetos.
- **Número de usuarios**
  - **Monousuario**: solo permiten un usuario.
  - **Multiusuario**: permiten la conexión de varios usuarios.

- **Número de sitios**
  - **Centralizados**: en un solo servidor o equipo.
  - **Distribuidos**: en varios equipos que pueden ser homogéneos y heterogéneos.
- **Ámbito de aplicación**
  - **Propósito General**: orientados a toda clase de aplicaciones.
  - **Propósito Específico**: centradas en un tipo específico de aplicaciones.
- **Tipos de datos**
  - **Sistemas relacionales estándar**: manejan tipos básicos (*int*, *char*, etc.).
  - **XML**: para el caso de bases de datos que trabajan con documentos *xml*.
  - **Objeto-relacionales**: para bases relacionales que incorporan tipos complejos de datos.
  - **De objetos**: para bases de datos que soportan tipos de objeto con datos y métodos asociados.
- **Lenguajes soportados**
  - **SQL estándar**.
  - **NoSQL o nuevo lenguaje de consulta**: menos estructurado y orientado a bases documentales o de tipo clave-valor. Es muy útil para manejar consultas de grandes cantidades de datos distribuidos en *clusters* de servidores.

Dentro de todos ellos los que con gran diferencia se han impuesto en casi todos los ámbitos del desarrollo han sido los basados en el modelo **relacional**. Ello se ha debido principalmente a su flexibilidad y sencillez de manejo. Igualmente, conviene destacar la amplia implantación del lenguaje SQL, que se ha convertido en un estándar para el manejo de datos en el modelo relacional, lo que ha supuesto una ventaja adicional para su desarrollo ya que, además, ha incorporado (en su versión SQL1999) aspectos importantes de la orientación a objetos.

Cabe destacar, sin embargo, la creciente popularidad de otros lenguajes como NoSQL (*Not Only SQL*), que se han desarrollado para adaptarse a datos masivos y dispersos en muchos servidores.

1.4.7 SISTEMAS GESTORES DE BASE DE DATOS COMERCIALES Y LIBRES

Con el advenimiento de Internet, el software libre se ha consolidado como alternativa, técnicamente viable y económicamente sostenible al software comercial, contrariamente a lo que a menudo se piensa, convirtiéndose el software libre como otra alternativa para ofrecer los mismos servicios a un coste cada vez más reducido.

Sin embargo, debe notarse que lo que comúnmente se denomina software libre no significa que sea gratuito ya que muchas empresas ofrecen servicios de soporte y mantenimiento para productos (en ocasiones de pago como *Red Hat* o *Suse*) pero cuyo código sigue siendo accesible.

Estas alternativas se encuentran tanto para herramientas de ofimática como Libreoffice, Openoffice frente a Microsoft Office, como herramientas mucho más avanzadas y de de propósito general, como MySQL frente SQL Server o a un nivel superior en cuanto a potencialidad, como PostgreSQL frente a Oracle, entre otros.

No obstante, cada vez más los fabricantes ofrecen versiones gratuitas (que no libres), aunque con limitaciones en su funcionalidad, de sus productos que permiten probarlos y aprender sus interioridades. Estas versiones se suelen denominar de tipo *express*.

Con independencia de la versión de producto también disponemos, tanto en los productos de software libre como en los de pago, de cada vez más documentación en línea que facilita enormemente su aprendizaje aunque en este sentido los productos de software libre destacan ya que los usuarios y comunidades generan una gran cantidad de documentación que está permanentemente actualizada.

Otro factor importante es el humano. Usar software libre a menudo implica un mayor conocimiento del producto y, por tanto, personal más cualificado. Por el contrario, esto permite un mayor control sobre el mismo y sobre las aplicaciones a desarrollar, además de una gran independencia con respecto al proveedor del mismo, por no hablar del ahorro en cuanto a licencias y costes de mantenimiento y soporte. Los sistemas comerciales, por el contrario, suelen ser más cerrados y rígidos. En todo caso todas las tecnologías nombradas disponen de servicios de soporte y documentación de gran calidad.

En definitiva, usar software libre o no es una elección del consumidor. Debe considerar estos y otros factores de la manera que mejor se adapte a sus necesidades.

No se trata tanto de defender "qué es mejor en sí mismo", sino de tener la información y poder elegir "qué es mejor según mis circunstancias".

A la hora de decidirse entre un sistema libre o comercial, un factor importante es si disponemos de personal cualificado, en cuyo caso un sistema libre es más barato y potente y nos dará más posibilidades y flexibilidad.

ACTIVIDADES 1.2

- Averigüe y explique el significado del término *ACID compliant* en el contexto de los sistemas gestores de bases de datos.
- ¿Qué se entiende por diseño físico de una base de datos? ¿Qué usuarios son los responsables del mismo?
- Averigüe en qué consiste y para qué sirve la minería de datos.
- ¿Qué lenguaje específico usa SQL Server para implementar el lenguaje SQL?
- Busque al menos 2 sistemas gestores libres (*Open Source*) y 2 comerciales e investigue sus ventajas e inconvenientes.
- Haga una investigación sobre las diferencias fundamentales entre los SGBD orientados a modelos relacionales de datos y los basados en *NoSQL*, como *CouchDB, Redis* o *Cassandra*.

1.5 BASES DE DATOS CENTRALIZADAS Y DISTRIBUIDAS

En un sistema de bases de datos centralizado todos los componentes (software, datos y soportes físicos) residen en un único lugar físico. Los clientes (aplicaciones, funciones, programas cliente, usuarios) acceden al sistema a través de distintas interfaces que se conectan al servidor. Sin embargo, existe otra arquitectura cada vez más extendida en la que se opta por un esquema distribuido en el que los componentes se distribuyen en distintos computadores comunicados a través de una red de cómputo. Dichos sistemas se conocen con el nombre de Sistemas Gestores de Bases de Datos Distribuidas o DDMGS.

Una base de datos distribuida es una colección de datos que pertenece lógicamente al mismo sistema pero que se encuentra físicamente almacenada en distintas máquinas conectadas por una red.

El surgimiento de las mismas se debe a varias razones entre las que destacan las siguientes:

- **Mejora de rendimiento**

  Cuando grandes bases de datos se distribuyen en varios sitios las consultas y transacciones que afectan a un solo sitio son más rápidas al ser más pequeña la base de datos local. Los sitios no se ven congestionados por muchas transacciones, ya que éstas se dispersan entre varios sistemas. De este modo, cuando una transacción requiera acceso a más de un sitio, estos pueden hacerse en paralelo, de forma que se reduce el tiempo de respuesta.

- **Fiabilidad**

  Definida como la probabilidad de que un sistema esté activo en un tiempo dado. Al distribuirse los datos es más fácil asegurar la fiabilidad del sistema, ya que si falla algún sitio es poco probable que afecte a la mayoría de transacciones.

- **Disponibilidad**

  Es la probabilidad de que un sistema esté activo de manera continuada durante un tiempo. Se ve mejorada por las mismas razones expuestas anteriormente. Esta característica se ve mejorada especialmente por la replicación de datos en varios sistemas.

- **Tipos de aplicaciones**

  Muchas aplicaciones tienen un marcado carácter distribuido al estar sus componentes dispersos en distintos sitios. Por ejemplo, una cadena de supermercados puede tener distintas sedes en distintas ciudades de forma que los clientes puedan acceder solamente a sitios y datos de su ciudad.

Por el contrario, distribuir implica una mayor complejidad en el diseño, implementación y gestión de los datos, para lo cual un buen SGBDD debe incorporar, además de los componentes habituales en un SGBD centralizado:

- ✓ Tener acceso a sitios remotos e intercambiar información con los mismos para la ejecución de consultas y transacciones.

- ✓ Disponer de un catálogo con información sobre cómo están distribuidos y replicados los datos del sistema distribuido.
- ✓ Poder optimizar consultas y transacciones sobre datos que estén en más de un sitio.
- ✓ Mantener la integridad en los permisos sobre datos replicados.
- ✓ Mantener la consistencia de las copias de un elemento replicado.
- ✓ Garantizar la recuperación del sistema en una caída.

Todo ello eleva enormemente la complejidad de tal SGBDD. Debemos añadir además la dificultad en el diseño óptimo de bases de datos distribuidas, especialmente en cuanto a las dos decisiones importantes: qué datos distribuir y qué datos replicar.

1.5.1 ARQUITECTURA DE UN DDBMS

En general, en un sistema distribuido disponemos de varias formas o modelos para organizar el software. Se habla de la arquitectura cliente-servidor consistente en dividir el software (la funcionalidad) entre equipos que hacen de clientes y otros que hacen de servidores. De esta manera podemos tener desde una máquina ligera sin disco, que se conecta a un servidor, de forma que todo ocurre en el servidor mientras que el cliente hace de interfaz, hasta un esquema en el que varios servidores trabajan de forma independiente con un esquema común de datos.

A partir de aquí surgen numerosas posibilidades de organización que se caracterizan por tres parámetros:

Autonomía

Tiene que ver con quién tiene el control sobre qué datos del SGBDD, es decir, determina el nivel de independencia de los SGBD. Distinguimos entre:

- **Integración fuerte**: un equipo hace de coordinador enviando las solicitudes de información a los equipos donde resida la información.
- **Sistema semiautónomo**: los SGBD son independientes pero participan en el conjunto pudiendo compartir parte de sus datos.
- **Sistema aislado**: el SGBD no tienen constancia de que existan otros gestores.

Distribución

Hace referencia a cómo se distribuyen los datos a lo largo del sistema:

- **Distribución cero**: no hay distribución de datos.
- **Cliente-servidor**: los datos residen en los servidores mientras que los clientes proveen una interfaz de acceso a los mismos además de otras posibles funcionalidades como caché de consultas.
- **Servidores colaborativos**: no se distingue entre servidores y clientes, cada máquina tiene toda la funcionalidad del SGBDD.

Heterogeneidad

Se refiere a la heterogeneidad de los componentes del sistema en distintos niveles como:

- Hardware.
- Comunicaciones.
- Sistema operativo.

Cualquier combinación de estos parámetros determina un modelo arquitectónico distinto para nuestro SGBDD. En la siguiente figura se observan las diferentes posibilidades según el peso de uno u otro aspecto:

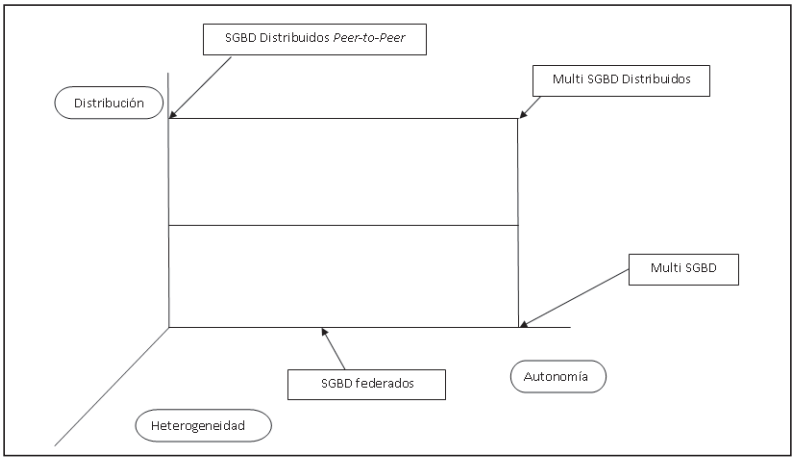

Figura 1.6. *Posibles arquitecturas de SGBDD*

Por otra parte, un SGBDD debe proporcionar lo que se conoce como transparencia en la distribución, lo que implica que cuando un usuario accede a la base todo lo que ocurre es transparente y queda al margen de cualquier detalle de cómo se ejecuta la transacción, es decir, al usuario se le presenta un esquema que no incluye información sobre la distribución de los datos. Es el software cliente quien consulta al catálogo del SGBDD para conocer la ubicación de los datos y poder así descomponer la consulta y distribuirla entre los distintos servidores. En particular, la transparencia debe serlo en cuanto dónde están los datos (transparencia de red), que datos están replicados (transparencia de replicación) y qué datos están fragmentados (transparencia de fragmentación).

1.5.2 TÉCNICAS DE FRAGMENTACIÓN, REPLICACIÓN Y DISTRIBUCIÓN

Diseñar bases de datos distribuidas implica decidir sobre cómo fragmentarlas, cómo replicarlas y cómo distribuirlas, además del diseño previo de la base de datos en sí.

La información sobre cómo se hace este proceso se almacena en el catálogo global del sistema al que tiene acceso el cliente cuando quiere acceder a la información.

Veremos ahora distintas técnicas para hacer lo anterior.

Fragmentación

Consideraremos a nuestra base de datos formada por unidades lógicas que son las relaciones o tablas. Supongamos que en nuestro ejemplo de banca electrónica queremos separar a nuestros clientes por regiones de España para facilitar el acceso. Entonces necesitaríamos dividir la tabla de clientes en tres partes almacenando cada una en un computador, es lo que denominamos *fragmentación horizontal* que divide las *tuplas* según una o varias condiciones sobre distintos atributos, en este caso sobre el campo *región* en la tabla *clientes*.

En otra situación podríamos querer dividir la información de un cliente en sus campos más accedidos y ligeros (nombre, apellidos, dirección) y almacenarla en un servidor más potente, mientras que otros campos más pesados (currículum, foto, historial, etc.) podrían almacenarse en otro servidor con menos capacidad de proceso pero más almacenamiento. Es lo que denominamos fragmentación vertical y debe hacerse con cuidado, ya que tenemos que incluir un atributo común, usualmente una clave, en ambos fragmentos para poder recuperar la información completa cuando sea necesario.

Por supuesto, existe también la posibilidad de una fragmentación mixta que incluya los dos casos comentados.

Debemos ser conscientes de que fragmentar es un proceso complejo que solo debe hacerse cuando se considere necesario por motivos de eficiencia, ya que al hacerlo hacemos también más complejo (más costoso en términos de CPU y consumo de recursos en general) el acceso a los datos por parte de los clientes.

La información sobre la fragmentación se guarda en lo que se denomina un esquema de fragmentación, que es una definición de todos los atributos de la base de datos y cómo están fragmentados. Un esquema de reparto permite definir dónde repartiremos los fragmentos. Si se da el caso de repartirlos en más de un sitio diremos que está replicado.

Fragmentar permite acercar los datos al consumidor, reducir el tráfico de red y mejorar la disponibilidad de los datos.

Replicación

Es fundamentalmente útil para mejorar la disponibilidad de los datos. El caso más extremo es la replicación en todos los sitios de la misma copia de la base de datos, que también es el caso que ofrece mayor disponibilidad, aunque puede reducir considerablemente la eficiencia en operaciones de actualización dado que cada operación debe realizarse en cada base de datos y esto puede traer problemas de integridad y consistencia de los mismos. En el extremo opuesto tenemos la no replicación, en la que cada fragmento está en un sitio en cuyo caso son disjuntos salvo las claves en fragmentos verticales; es lo que se denomina *reparto no redundante*. Entre ambos extremos se da una amplia gama de posibilidades que queda descrita en lo que se denomina *esquema de replicación*. Todo ello se describe en el catálogo de replicación donde se indica qué objetos son replicados, dónde está la réplica y cómo se propagan las actualizaciones. El catálogo es un conjunto de tablas guardadas en cada sitio en el que hay réplicas. Las configuraciones posibles son variadas y dependen mucho de los requisitos de las aplicaciones.

La elección de un sitio para un fragmento depende de los objetivos de rendimiento y disponibilidad para el sistema y de los tipos y frecuencia de transacciones. También depende de si la base es más orientada a consultas o, por el contrario, hay un alto grado de operaciones de escritura.

La replicación, como hemos señalado, distribuye la carga, acelera consultas y mejora la disponibilidad pero, sobre todo, es óptima para sistemas con muchas lecturas. Sin embargo, requiere muchas más actualizaciones con el consiguiente consumo de almacenamiento.

ACTIVIDADES 1.3

- Investigue sobre SGBD, comerciales o libres, que soporten replicación y fragmentación de datos.
- ¿Cuál es el principal problema que tiene la replicación de datos?
- Analice la conveniencia de replicar y fragmentar datos en los siguientes sistemas:
  a. Base de datos de películas *on line*.
  b. Base de datos de un banco.

RESUMEN DEL CAPÍTULO

En este capítulo introductorio hemos repasado las principales características de los sistemas de almacenamiento, así como las organizaciones primarias de archivos. Hemos estudiado el concepto de índice y los tipos más habituales. También se han analizado los distintos modelos de datos, así como los sistemas gestores más importantes que nos permiten implementar dichos modelos.

Así mismo, hemos hecho una pequeña comparativa entre sistemas usados en la actualidad, tanto libres como comerciales. En este sentido hemos visto que hay una gran diversidad de sistemas de distinta complejidad y potencialidad, desde los básicos sistemas monousuario a los grandes sistemas capaces de tomar decisiones o almacenar datos distribuidos. Dicha complejidad se ha traducido también en un repertorio cada vez más amplio y potente que facilita y mejora las funciones del administrador, permitiéndole hacer tareas cada vez más complejas y eficientes.

Por último, hemos descrito los principales conceptos de bases de datos distribuidas entre dos o más servidores.

EJERCICIOS PROPUESTOS

1. ¿Qué es un sistema de información?

2. Investigue en qué consisten las bases de datos XML.

3. Indique al menos tres ventajas e inconvenientes de usar bases de datos frente a los tradicionales sistemas de ficheros

4. Cuando accedemos a información de una página web como Amazon, ¿en qué nivel, dentro de la arquitectura de 3 niveles, nos encontramos? Explíquelo.

5. Comente qué se entiende por software libre considerando aspectos como:
- Gratuidad.
- Código fuente.
- Uso comercial.

6. ¿Qué tiene que ver la administración de un SGBSD con el diseño de bases de datos?

7. Enumere al menos tres objetos típicos de una base de datos indicando su función.

8. ¿Para qué sirve un disparador en un SGBD?

9. Explique con sus palabras qué es el diccionario de datos en un SGBD.

10. En una base de datos como la de YouTube, ¿qué puede ser más conveniente para mejorar su funcionamiento, fragmentar o replicar los datos?

TEST DE CONOCIMIENTOS

1 ¿Qué es una base de datos?
- **a)** Un programa para organizar datos.
- **b)** Un software que facilita la gestión de datos.
- **c)** Un conjunto de datos organizados.
- **d)** Todas las respuestas anteriores son correctas.

2 ¿Cuál es el significado de GPL en el contexto informático?
- **a)** *General Public Library*.
- **b)** *Great Politic Licence*.
- **c)** *General Public Licence*.

3 Un índice de agrupamiento:
- **a)** Permite ordenar ficheros.
- **b)** Es un tipo de archivo ordenado.
- **c)** Sirve para fragmentar bases de datos.
- **d)** Facilita el acceso a datos según un campo.

4 ¿Qué se quiere decir cuando se habla de nivel conceptual?
- **a)** Lo que percibe el usuario.
- **b)** La imagen de la base de datos vista por el ordenador.
- **c)** El código para crear la base de datos.
- **d)** Una representación de la base de datos independiente de la implementación física.

5 Las bases de datos son:
- a) Relacionales.
- b) Relacionales o jerárquicas.
- c) Primero eran en red y ahora son relacionales.
- d) La mayoría son relacionales.

6 Un modelo es:
- a) Una forma de representar información.
- b) Un programa para dibujar cajas y flechas.
- c) Una forma de representar un sistema.
- d) Una representación de un conjunto de datos.

7 Los sistemas gestores:
- a) Permiten gestionar bases de datos.
- b) Controlan el acceso a los datos.
- c) Incluyen un diccionario de datos.
- d) Todas las respuestas anteriores son correctas.

8 ¿Qué es cierto respecto a los SGBD y bases de datos?
- a) No hay diferencia.
- b) El primero hace referencia a un software mientras las bases de datros no tienen realidad física.
- c) Las bases de datos se crean necesariamente con un SGBD.
- d) Un SGBD es una herramienta CASE.

9 Los sistemas libres:
- a) Son más potentes y mejores que los comerciales.
- b) Son más baratos.
- c) Son más difíciles.
- d) Ninguno de los anteriores necesariamente.

10 La independencia física:
- a) Hace que podamos acceder a los datos desde cualquier equipo.
- b) Permite modificar los modelos independientemente de su almacenamiento.
- c) Evita problemas de redundancia.
- d) Hace que podamos usar las bases de datos independientemente del sistema operativo.