Sistemas de base de datos

Jaume Raventós Moret

PID_00201453

Universitat Oberta de Catalunya

Índice

Introducción

Las bases de datos son elementos imprescindibles en muchas actividades de la vida cotidiana. Por ejemplo, al ingresar o retirar dinero de la cuenta bancaria, al reservar un vuelo o una habitación de hotel, al acceder al catálogo de una biblioteca o al comprar productos en un supermercado se requiere que alguien interactúe con una base de datos.

Remisiones a otras partes del contenido

En las remisiones a otros contenidos (referencias cruzadas), si no se especifica lo contrario, los apartados referenciados se encuentran en el mismo módulo didáctico.

Desde que se empezaron a introducir los ordenadores para automatizar la gestión de las empresas utilizando un lenguaje de programación, la evolución de los sistemas de información ha tenido una repercusión considerable en la gestión de los datos. Los **sistemas informatizados de gestión de datos** se concibieron por la necesidad de almacenar una gran cantidad de información de una manera rápida, sencilla y fiable para la consulta y utilización posteriores en cualquier momento sin necesidad de desplazarse a estancias dedicadas a archivar documentación.

Las **bases de datos** son una de las herramientas más ampliamente difundidas en la sociedad de la información y un producto estratégico de primer orden, dado que constituyen el **fundamento de los sistemas de información**. Su rápido crecimiento ha dado lugar a una industria específica de productividad excepcional e impacto económico impresionante.

La sociedad de la información se caracteriza por el manejo de un volumen desmedido y aceleradamente creciente de información de todo tipo que supone métodos y técnicas de almacenamiento y acceso a los datos radicalmente diferentes de los utilizados tradicionalmente. En las últimas décadas han aparecido aplicaciones nuevas de sistemas de base de datos que permiten almacenar información audiovisual, extraer y analizar información útil de bases de datos grandes para tomar decisiones, controlar procesos industriales, localizar información a través de Internet, etc.

Nuevas aplicaciones de sistemas de base de datos

Estamos hablando de:

- BD multimedia.
- BD *data warehouse*.
- BD web.

Para poder desarrollar los contenidos expuestos en este módulo y entender los fundamentos de la tecnología de bases de datos, conviene tener claros unos **conceptos básicos** que definiremos brevemente en el primer apartado.

Haremos un repaso de la **evolución de la gestión de datos**, desde los sistemas manuales de fichas en tarjetas de cartulina, pasando por la gestión informatizada con ficheros de datos, hasta los sistemas de base de datos. A continuación, describiremos las **estructuras de datos** más utilizadas en los sistemas de información, que permiten especificar la manera de organizar los datos para obtener un rendimiento razonable en su almacenamiento, tratamiento y recuperación.

Seguidamente, señalaremos los **problemas** que presenta la **utilización de los sistemas de gestión de datos basados en ficheros**, que justifican el uso de bases de datos, de las que enumeraremos los **objetivos** y las **características** que deben cumplir para ser consideradas como tales. A partir de estas características, extraeremos las **ventajas** que presentan las bases de datos frente a los ficheros de datos tradicionales. También comentaremos los **inconvenientes** que supone la utilización de bases de datos y los **casos en los que es mejor utilizar un sistema de ficheros** en lugar de un sistema de base de datos.

Identificaremos los principales **elementos que forman un sistema de base de datos** en su concepción más amplia. Esto es, el contenido (la base de datos), el software (básicamente, el sistema de gestión de bases de datos o SGBD), el hardware y el personal humano. Dada su importancia para el funcionamiento adecuado del SGBD, estudiaremos las distintas **tipologías de usuario** que interactúan con la base de datos. Veremos que los relacionados más directamente con la base de datos son el usuario final, el programador y el administrador de la base de datos.

Más adelante, trataremos los conceptos de **modelo, esquema y estado de la base de datos**. Una característica fundamental de un sistema de base de datos es que proporciona un cierto nivel de abstracción de los datos porque oculta detalles de su almacenamiento. Veremos que un modelo de datos permite conseguir el proceso de abstracción que conduce del mundo real al mundo de los datos, y distinguiremos entre la descripción de la base de datos y la propia base de datos; es decir, entre el esquema y el estado de la base de datos.

La **abstracción** permite considerar la esencia de una realidad escondiendo los detalles o discernir sus elementos para considerarlos aisladamente, de manera que se reduce la complejidad y se facilita la comprensión. Introduciremos los diferentes **tipos de abstracción** que ofrecen los modelos de datos para facilitar la representación de los datos en el diseño de bases de datos y que permiten establecer vínculos entre los elementos de un modelo.

Consideraremos la **arquitectura de los sistemas de gestión de bases de datos** desde diferentes enfoques para entender sus características fundamentales. Trataremos, en especial, la **arquitectura de tres niveles**, que establece una división de la base de datos según la perspectiva desde la que ésta es vista en niveles de abstracción, que se corresponden con tres grupos principales de usuarios. Además, analizaremos las correspondencias entre niveles y el concepto de ligadura.

Daremos una visión integrada de los elementos y las características de un sistema de base de datos entendido como el sistema que integra el SGBD y la base de datos en un sentido amplio en lo que denominamos **estructura global del sistema de base de datos**.

En el apartado siguiente, trataremos el **almacenamiento de bases de datos**, es decir, cómo se organizan los datos en soportes de almacenamiento secun-

dario, que permiten guardar grandes cantidades de información de manera persistente, de modo que el SGBD pueda recuperar, actualizar y procesar los datos cuando sea necesario. Para comprender el **acceso del SGBD a los datos**, veremos el esquema general del flujo de datos y de control que sigue el proceso de ejecución de una consulta realizado por un programa de usuario al SGBD con la ayuda del sistema operativo subyacente.

A partir de los objetivos de las bases de datos, deduciremos las **funciones** que debe ejecutar el SGBD, cuyo análisis nos ayudará a determinar los **componentes** que debe tener el SGBD para cumplirlas.

A continuación, clasificaremos según varios criterios los **lenguajes de base de datos** que permiten manejar la base de datos. Presentaremos los diferentes tipos de **interfaces del SGBD**, muchas de las cuales permiten omitir el uso del lenguaje de base de datos. Estudiaremos los módulos de software que forman el **núcleo del SGBD**, que se ejecutan como procesos transparentes para el usuario. Finalmente, veremos cómo el núcleo, que se puede considerar el intérprete entre el usuario y los datos, se ocupa de las tareas básicas de la base de datos, del trabajo con el lenguaje de base de datos, del diálogo con el sistema operativo, etc.

Objetivos

En el material didáctico de este módulo encontraremos las indicaciones para alcanzar los objetivos siguientes:

1. Aprender los conceptos básicos sobre los sistemas de base de datos.

2. Saber cuáles son los usos más importantes de las bases de datos.

3. Entender cuál es la estructura de los sistemas de gestión de bases de datos.

1. Algunos conceptos básicos

Desde la aparición de los sistemas informáticos, una de sus principales aplicaciones ha sido el almacenamiento y el tratamiento de grandes cantidades de datos para permitir la consulta y la utilización posteriores. Una base de datos, formada por una colección de datos entre las que se establece una relación, es una de las herramientas más utilizadas a la hora de trabajar con grandes volúmenes de información.

El objetivo de este apartado es aclarar una serie de conceptos fundamentales en el entorno de las bases de datos, definiéndolos brevemente. Esto permitirá desarrollar, más adelante, los aspectos siguientes:

- La problemática que presenta la utilización de ficheros de datos, que justifica el uso de sistemas de base de datos.
- Los objetivos, las ventajas y los inconvenientes de las bases de datos.
- Los requisitos que deben cumplir las bases de datos para ser consideradas como tales.
- Los elementos que componen un sistema de base de datos en su concepción más amplia; es decir, incluyendo, junto con los datos y el software que las gestiona, los equipos necesarios y las personas que interactúan con la base de datos.

A continuación, se exponen algunos conceptos básicos sobre bases de datos.

Información

Es la agregación de varios datos junto con sus relaciones o dependencias para acceder a un nivel de comprensión y entendimiento más alto de un fenómeno determinado.

La información es el **recurso básico para la toma de decisiones** y puede ser definida desde múltiples campos de conocimiento que la asocian a problemáticas y ámbitos de estudio muy distintos.

Dato

Es la **unidad mínima básica de información** (o propiedad que caracteriza un objeto, fenómeno o acontecimiento) que, representada en un formato adecuado, se puede transmitir y/o procesar por medios manuales o automáticos.

Un dato puede contener caracteres alfabéticos, magnitudes numéricas, símbolos especiales, colores, imágenes o, incluso, grupos de símbolos no aleatorios que representan cantidades, acciones, objetos, etc.

El concepto de dato en informática (más amplio que el usado en ciencias puras como la física o las matemáticas) es un conjunto de símbolos o caracteres empleados para expresar o representar un valor numérico, un hecho, un objeto o una idea de la manera adecuada para su tratamiento por el ordenador. Los datos se pueden escribir en un teclado, encontrar en la memoria principal, almacenar en un disco, etc.

Jerarquía de los datos

La jerarquía que siguen los datos almacenados en una base de datos se expone en el subapartado "El contenido: los datos" dentro del apartado "Elementos de un sistema de base de datos".

El proceso informacional: datos, información, conocimiento

Es útil recordar la gradación jerárquica que habitualmente se establece entre los conceptos de dato, información, conocimiento y sabiduría o inteligencia en el llamado proceso informacional:

Datos→ Información → Conocimiento → Sabiduría

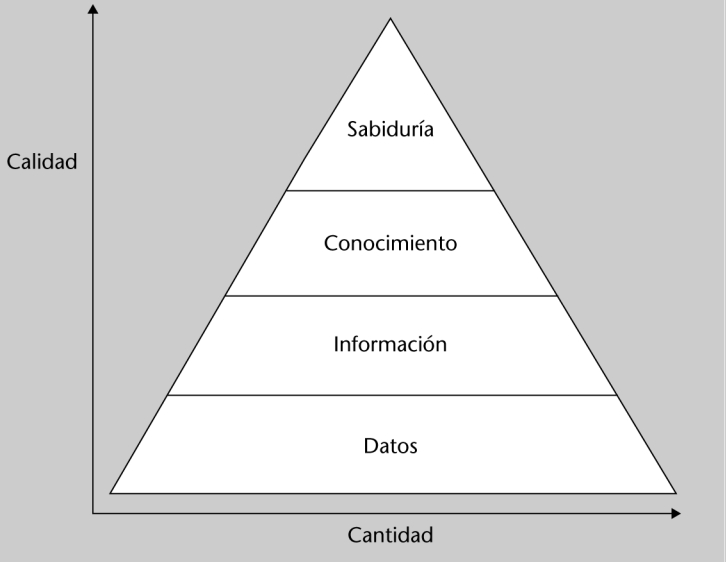
Conocimiento: es el resultado de relacionar informaciones para construir teorías.

Sabiduría: implica un grado más alto de conciencia del alcance y de los límites del conocimiento propio y una capacidad de contextualización más amplia.

El contenido de la información es el conocimiento que aporta la información que se tiene de un fenómeno determinado a través de los datos que se han obtenido y es utilizado en la toma de decisiones.

Este proceso se puede expresar en la llamada *pirámide informacional,* formada por cuatro niveles, que implican una jerarquía de las variables calidad y cantidad:

Figura 1. Pirámide informacional



Desde el punto de vista de la gestión de información, Iraset Páez Urdaneta (1992) plantea la reflexión siguiente:

"En la antigüedad, el hombre occidental quería ser sabio; después, el hombre moderno quiso ser conocedor; el hombre contemporáneo parece contentarse con estar informado (y posiblemente el hombre del futuro no esté interesado en otra cosa que en tener datos)".

Fuente: Iraset Páez Urdaneta (1992). *Gestión de la inteligencia, aprendizaje tecnológico y modernización del trabajo informacional. Retos y oportunidades*. Caracas: Instituto de Estudios del Conocimiento de la Universidad Simón Bolívar/ CONICIT.

Interpretación de datos y conversión en información

Los datos deben ser interpretados (incorporando significado) para que se conviertan en información útil. Por lo tanto, un dato es cualquier elemento de información que no tiene significado fuera de su contexto.

La relación entre estos conceptos se describen con estos ejemplos:

Datos:	1985 (año de nacimiento); X (nombre del alumno)
Interpretación:	año de nacimiento de un alumno determinado
Información:	el alumno X nació en el año 1985
Datos:	65 (peso en kilogramos); Y (nombre del paciente)
Interpretación:	peso en kilogramos de un paciente determinado
Información:	el paciente Y pesa 65 kilogramos

Objeto

Es la parte del mundo real que nos interesa y que percibimos con nuestros sentidos. Un objeto concreto puede ser físico o abstracto.

Ejemplos de objeto

Una persona, un objeto físico, una empresa, un producto, un compuesto químico, un concurso, una etnia, una ideología, un partido político, etc.

Entidad

Es la conceptualización de un objeto del mundo real.

Tipo de entidad (tipo de objeto)

Este concepto por el que una entidad es instancia (ocurrencia) lo podéis consultar en el punto "Ocurrencia y tipo" del subapartado "Modelo de datos" dentro del apartado "Modelo, esquema y estado de la base de datos".

Registro

Es un conjunto de datos correspondientes a un único objeto del mundo real.

Atributo

Es la propiedad de una entidad.

Campo

Es la representación del valor de un atributo.

Clave

Es el atributo o el conjunto de atributos que permiten identificar los objetos (distinguir unos de otros).

Fichero

Un fichero de datos es un conjunto de datos relativos a un tipo de objeto y estructurados en registros, configurado como una unidad de almacenamiento para la búsqueda de uno o más datos individuales.

Dicho de otro modo, es un **conjunto de registros** relativos a un mismo tipo de entidad.

Distinción entre los términos *fichero* y *archivo*

A pesar de que aquí los términos *archivo* y *fichero* se usan indistintamente y, habitualmente, se usan como sinónimos, a veces es conveniente considerar algunas diferencias:

1) Se puede emplear **archivo** en un sentido general, como *fichero de ficheros*, o en un sentido más limitado o específico, como *fichero archivado* (fichero que ha dejado de tener operatividad directa pero que se conserva por varias razones –legales, históricas, etc.– y que, normalmente, implica su traspaso a un soporte más económico al que se accede muy pocas veces).

2) Se considera más apropiado el término ***fichero*** para hacer referencia al *conjunto de datos* relacionados entre sí ubicados en un soporte material accesible, que puede ser objeto de varias operaciones (guardar, recuperar, eliminar, etc.).

Entre otros, es común distinguir los siguientes tipos de fichero:

- **Fichero de datos** o de documento: consiste en datos en forma de texto, números o gráficos. Normalmente, es creado y utilizado por el usuario.
- **Fichero de programa** o de aplicación: contiene la parte ejecutable de un programa. Es utilizado por el ordenador y creado por el programador. Un programa puede estar formado por más de un fichero, y cada uno de ellos contiene instrucciones para ejecutar partes determinadas del funcionamiento global del programa.

Base de datos

De momento, denominamos *base de datos* al **conjunto estructurado de ficheros de datos** que representan objetos del mundo real entre los que se establecen relaciones.

Según la definición elaborada por C. J. Date:

> "Una base de datos es un sistema computacional con el propósito general de albergar información y hacer que sea accesible cada vez que sea necesaria entendiendo por dato cualquier cosa, real o abstracta, que tenga interés para un usuario o una organización".

Sistema de gestión de bases de datos (SGBD)

Es el conjunto del software que permite la gestión automática de una base de datos; generalmente, la creación, el almacenamiento, la manipulación y el control de los datos.

Sistema de base de datos

Es un sistema de información basado en una base de datos.

Sistema de información

Es un conjunto de elementos relacionados entre sí ordenadamente de acuerdo con unas reglas determinadas, que aportan a la organización a la que sirven la información necesaria para el cumplimiento de sus fines.

Normalmente, se hace referencia a un *sistema de información informatizado* (soportado por un sistema informático) simplemente como sistema de información, a pesar de que también puede haber *sistemas de información manuales*.

Elementos y funciones de un sistema de información

Las funciones básicas de un sistema de información son la recogida, el procesamiento y el almacenamiento de datos, así como la elaboración y la presentación de estos datos. Para el cumplimiento de estas funciones, un sistema de información está formado por los elementos siguientes:

- **Contenido**: los datos y su descripción.
- **Equipo lógico**: el sistema de comunicaciones, el sistema de gestión de bases de datos, otros programas que los manipulan, el sistema operativo, etc.
- **Equipo físico**: el ordenador o los ordenadores que soportan todo el sistema.
- **Equipo humano**: el administrador, el usuario final, etc.

El núcleo de todo sistema de información actual es una base de datos.

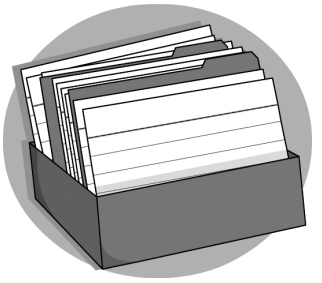
2. Evolución de la gestión de datos

En este apartado se hace un repaso cómo han evolucionado los sistemas de gestión de datos, desde los antiguos sistemas manuales de fichas de cartulina que se guardaban en archivadores, pasando por la gestión informatizada con ficheros de datos, hasta los actuales sistemas de base de datos.

Además, se constata que los sistemas de ficheros ponen el énfasis en el tratamiento de los datos dispersos en ficheros creados para una aplicación concreta, mientras que las bases de datos son conjuntos estructurados de datos no creados para una aplicación específica.

2.1. Sistema manual de fichas de cartulina

Antiguamente, los datos extraídos de fuentes bibliográficas (libros, revistas, periódicos) o no (conceptos, ideas, etc.) se registraban y consultaban mediante **fichas o tarjetas** rectangulares, habitualmente **de cartulina** o papel consistente. Las fichas se ordenaban alfabéticamente y se reunían en **archivadores, catálogos o ficheros manuales**.



Conjunto de fichas o tarjetas

Como criterios de busca y recuperación de los objetos registrados se utilizaban **muy pocos descriptores** (normalmente, tres). Por ejemplo, cada libro se registraba en tres fichas diferentes: la ficha de autor, la ficha de materia y la ficha de título. El contenido de las tres fichas era el mismo, sólo cambiaba la posición de los descriptores: en la primera, el nombre de autor aparecía en primer lugar; en la segunda, la materia aparecía primero, y en la tercera, el título era el que aparecía en primer lugar. De este modo, para cada libro se creaban tres catálogos o ficheros manuales.

2.2. Gestión de datos informatizada

La aparición de los ordenadores, los soportes electrónicos y, especialmente, los ficheros de datos en formato digital cambiaron radicalmente la situación. Además, gracias a las redes de ordenadores y a los sistemas de comunicaciones, los datos no necesitan estar alojados físicamente en el lugar donde son requeridos y pueden ser accesibles desde ubicaciones remotas.

Además, es posible asignar a cada objeto registrado un número elevado de descriptores para mejorar su representación y facilitar y ampliar las posibilidades de búsqueda y recuperación.

Las **ventajas** de un sistema de gestión de datos *informatizado* monousuario respecto al sistema manual clásico de fichas de cartulina o papel son las siguientes:

- **Más consistencia:** agrupación de la información y gran reducción del espacio físico, puesto que se evita la utilización de archivadores voluminosos de papel.

- **Más actualización**: disponibilidad inmediata de información actual y precisa.

- **Más velocidad:** obtención y modificación de los datos por parte del ordenador con mucha más rapidez que el ser humano.

- **Menos laboriosidad:** eliminación del mantenimiento manual pesado de archivadores y fichas, gracias a la mejor y más eficiente realización de las tareas mecánicas por el ordenador.

- **Menos coste:** reducción de los costes asociados al menor consumo de espacio físico gracias a la utilización de soportes electrónicos.

2.3. Sistemas de ficheros: orientados al proceso

Los primeros sistemas informatizados de gestión de datos fueron los ficheros.

Los sistemas de gestión de datos basados en ficheros ponen el énfasis en los tratamientos que reciben los datos, que se almacenan dispersos en varios ficheros generalmente planos, diseñados para una aplicación determinada.

A medida que crecen las necesidades de información, se implementan aplicaciones nuevas independientes entre sí y se crean ficheros de datos nuevos que no se suelen transferir entre programas de aplicación, sino que se duplican, si es necesario, como parte de éstos.

Los principales **inconvenientes** que presentan estos sistemas son los siguientes:

- **Ocupación inútil de memoria** secundaria y aumento de los tiempos de procesamiento porque se repiten los mismos controles y operaciones en los diferentes ficheros (redundancia).

- **Inconsistencia** debido al hecho de que la actualización de los datos no se puede llevar a cabo de manera simultánea en todos los ficheros donde se encuentran.

- **Aislamiento** debido al hecho de que los datos y/o los programas se almacenan en varios formatos, o que programadores diferentes pueden haber usado lenguajes de programación diferentes.

- **Dependencia de los datos respecto al soporte físico y a los programas**, lo que provoca falta de flexibilidad y de adaptabilidad frente a los cambios y repercute negativamente en el rendimiento del sistema.

Integración de datos de ficheros de departamentos diferentes

Si se quieren integrar los datos de diferentes departamentos de una organización y mantenerlos actualizados, se debe crear un programa nuevo. Esto exige un gran esfuerzo para conocer cómo se han almacenado físicamente en cada uno de los programas existentes y cuál debe ser el nuevo sistema de almacenamiento. Este sistema nuevo se solapará con el existente, lo que dificultará la introducción de cambios, puesto que **los datos dependen físicamente del fichero y del lenguaje** con el que fueron creados.

- **Incapacidad para responder a demandas inesperadas de información** fuera del contexto para el que fueron concebidos.

- **Poca fiabilidad**, falta de adecuación a la realidad, mal control de la confidencialidad o privacidad...

Los inconvenientes de los ficheros de datos

Se tratan ampliamente en el apartado "Problemas de la gestión de ficheros de datos" de este mismo módulo.

2.4. Del procesamiento de ficheros a la tecnología de bases de datos

Los programas de los **sistemas tradicionales de procesamiento de ficheros** trataban únicamente los datos de sus ficheros. Como mucho, permitían que otros programas leyeran esta información de la que eran "propietarios", pero sin poder modificarla. Por otro lado, los ordenadores que los controlaban tenían poca potencia, de modo que los datos se trataban en un único ordenador (sistemas monousuario).

Al ampliarse el número de programas de aplicación que podían acceder a los datos y con la aparición de sistemas más potentes que permitían el trabajo multiusuario, se hacía necesario:

- **Independizar los datos** de sus aplicaciones respectivas.

- Crear sistemas que mantuvieran la **coherencia** de los datos para resolver los posibles conflictos originados por la actuación simultánea de varios usuarios.

En el proceso para solucionar estos problemas (de dependencia y de concurrencia) a mediados de los años sesenta nació la **tecnología de bases de datos**.

2.5. Sistemas de base de datos: orientados a los datos

Los datos se organizan y se mantienen en un conjunto estructurado que no está diseñado para una aplicación concreta, sino que pretende satisfacer las necesidades de información, cambiantes y muy variadas, de los usuarios y de toda la organización.

Las características de las bases de datos

Se enumeran en el apartado "Objetivos y características de las bases de datos" de este módulo.

Su **ventaja** principal es la **independencia de los datos** respecto al soporte físico y respecto a los programas.

Las ventajas de las bases de datos

Se detallan en el apartado "Ventajas e inconvenientes de las bases de datos" de este módulo.

3. Estructuras de datos

La información se organiza o se estructura de manera adecuada para obtener un rendimiento razonable en su almacenamiento, su tratamiento y su recuperación. La estructura de datos es el esquema organizativo que se aplica a los datos para facilitar su interpretación o hacer operaciones. Para especificar cómo se organizan los datos de la información, se utiliza el tipo de dato.

El **tipo de dato** es la definición de un conjunto de datos que especifica el intervalo de valores posibles, las operaciones y funciones que se pueden ejecutar sobre los valores y la manera de almacenarlos en memoria. Hay dos categorías de tipos de dato:

1) Tipo de dato elemental o, simplemente, **tipo de dato**

Se puede utilizar para construir tipos de datos más elaborados:

2) Tipo de dato estructurado o **estructura de datos**

Ejemplo de tipo de dato estructurado

Un dato de tipo número complejo (que representa el conjunto de números complejos) es un par ordenado de datos elementales acompañado de un operador. Tiene la forma a+b*i*, donde "a" y "b" son números reales e "*i*" es el valor constante *raíz cuadrada de –1* (denominado unidad imaginaria).

Se compone de una serie de datos de tipo elemental y con alguna relación existente entre ellos. Normalmente, suele ser una relación de orden, pero también la hay de otras clases.

Espacio de memoria según el tipo de dato

Siempre que se utiliza un dato en un programa, su tipo de dato debe estar determinado para que el programa traductor sepa cómo lo debe tratar y almacenar de manera apropiada.

1) En los datos de tipo elemental, el tipo de dato determina el espacio que se usa en memoria.

2) En los datos de tipo estructurado, se presentan dos casos:

- **Estructura de datos dinámica**. El número de componentes que tiene puede ir creciendo. El espacio de memoria que necesita se asigna dinámicamente. Son ejemplos de este caso las listas y los árboles, que veremos mas adelante.
- **Estructura de datos estática**. El dato siempre ocupa el mismo espacio de memoria. Un ejemplo de este caso son las matrices.

Las **estructuras de datos** pueden ser:

- **Contiguas:** sus elementos se almacenan en un conjunto consecutivo de ubicaciones de memoria. Ejemplos: matriz, cadena de caracteres, registro, lista lineal.
- **No contiguas:** sus elementos se almacenan en ubicaciones de memoria no contiguas y utilizan punteros para conectarse. Ejemplos: árbol, lista vinculada.

Los elementos de datos se organizan de manera que se puedan recuperar en un orden particular que no es necesariamente el mismo en el que están almacenados. Para cada elemento de datos se almacenan dos partes: el dato del mismo elemento y un **puntero** con la dirección del elemento siguiente. Esto permite que se puedan insertar o eliminar elementos sin necesidad de mover los otros elementos para dejar espacio.

El **puntero** (apuntador o *locator*) contiene una variable auxiliar que especifica la **dirección de memoria** (es decir, la posición o ubicación en memoria en la que se almacena físicamente un elemento de datos determinado).

Los **tipos de datos estructurados** más utilizados son los siguientes:

1) Matrices

Una matriz (también llamada *formación* o *array*) consiste en una cantidad fija de elementos de datos del mismo tipo, cada uno de los cuales tiene asociado al menos un índice que determina de manera unívoca la posición del elemento en la matriz. Cada combinación de valores de índice (a veces llamados *subíndice*s) identifica un elemento de la matriz y sólo uno.

Una matriz se puede imaginar como una estructura de celdas en las que se almacenan valores. Es una estructura de datos estática, puesto que al definirla se especifica el número de elementos de los que se compone.

Es la estructura de datos más usual. El número de índices de la matriz se denomina *número de dimensiones*. Una matriz de una dimensión se denomina *vector* o *matriz lineal*. Las matrices de dos dimensiones son útiles para almacenar tablas.

2) Cadenas de caracteres

Una cadena de caracteres (también llamada *string*) está formada por una secuencia de caracteres. Permite almacenar una palabra, una frase, una temperatura, etc.

Ejemplo de cadena de caracteres

El nombre de una persona: "Miguel Martín".

3) Registros

Un registro (también llamado *file*) se compone de elementos, almacenados juntos, que contienen información relativa a una misma entidad. Estos elementos (llamados *campos*) pueden ser de tipos diferentes, aparecen en un orden determinado y se identifican por medio de un nombre. Para definir un registro hay que especificar el nombre y el tipo de cada campo.

Ejemplo de registro y matriz

El registro de un alumno individual puede constar del nombre del alumno (caracteres), el número de ausencias (número entero) y la media de sus calificaciones (número con coma flotante).

Por otro lado, el conjunto de registros de los alumnos de un aula forma una matriz de dos dimensiones de registros individuales (una tabla).

4) Listas

Una lista está formada por un número variable de datos (o elementos) de un mismo tipo, ordenados según una secuencia lineal (primero, segundo, tercero, etc.).

Formalmente, se puede definir como una estructura de datos formada por registros de al menos dos campos, en los que uno de ellos contiene información que permite localizar el registro siguiente en la lista según una secuencia determinada.

Ejemplos de lista

El conjunto de letras del abecedario, las poblaciones del trayecto de un medio de transporte interurbano, la lista de espera para una operación de menisco, etc.

A diferencia de la matriz lineal (o vector), es una estructura dinámica (ocupa en memoria el espacio necesario para albergar sus elementos) y no es direccionable, sino secuencial (sólo se puede procesar un elemento accediendo previamente a los que lo preceden).

Hay dos tipos principales de lista:

a) Lista lineal: es una lista simple y ordenada de elementos en la que cada elemento, excepto el primero, va a continuación de otro elemento, y cada elemento, excepto el último, precede a otro elemento.

b) Lista vinculada (o enlazada): es como una matriz, excepto por el hecho de que las posiciones de la memoria física en la que se almacenan los elementos no son necesariamente consecutivas. Sus elementos tienen enlaces (por medio de punteros) a los elementos que los siguen lógicamente (la posición del elemento siguiente se almacena junto con cada elemento).

5) Árboles

Un árbol es una estructura de datos similar a la estructura de lista vinculada, pero más compleja, puesto que cada elemento incorpora (almacena) las direcciones de dos o más elementos, en lugar de la dirección de sólo un elemento.

Se puede definir, recursivamente, como un conjunto de nodos formado por un nodo principal denominado *raíz* y una serie finita de subconjuntos de nodos (disjuntos) que también son árboles. El *nodo* es una localización en el árbol que puede tener enlaces a uno o más nodos por debajo de él. Cada elemento (nodo) tiene un único *padre*. Para representar un árbol, normalmente se utilizan tres punteros por elemento: uno para dirigirse al padre, otro para el primer hijo y el tercero para el hermano siguiente.

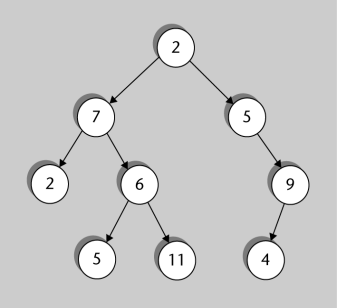


Figura 2. Ejemplo de estructura en árbol

El árbol es una estructura dinámica y una manera eficaz de almacenar elementos que deben ser buscados y recuperados rápidamente.

4. Problemas de la gestión de ficheros de datos

El análisis de los problemas que supone el *procesamiento de ficheros tradicionales* permite justificar la utilización de *bases de datos* como alternativa para eliminar (o, al menos, reducir) estos inconvenientes.

En un sistema de gestión de datos basado en ficheros, la información está dispersa en varios ficheros de datos y diferentes programas que los agrupan y los recuperan. A medida que crecen las necesidades de información, se crean programas de aplicación y ficheros de datos nuevos (como parte de estos programas de aplicación).

Los inconvenientes que presenta la gestión de ficheros de datos se pueden agrupar en dos bloques: los que afectan a los ficheros y los que afectan a los datos, como se describe a continuación:

4.1. Problemas respecto a los ficheros

1) Necesidad de controlar la **integridad semántica**

La integridad semántica es un conjunto de restricciones (también llamadas *reglas de validación*, *normas de introducción* o *limitantes de consistencia*) que controla el almacenamiento de determinados valores a la hora de introducirlos en el sistema.

Los diferentes programas que permiten la introducción de datos pueden tener exigencias diferentes respecto a la integridad semántica. Si cada fichero dispone de sus propias reglas de validación para la adición o modificación de datos, es difícil que los diferentes programas tengan en cuenta a la vez todas las reglas de validación, por lo que se produce **inconsistencia** de la información en todo el sistema, aunque los datos introducidos cumplan las reglas de validación creadas para los subsistemas o ficheros individuales.

Ejemplo de regla de validación

El valor del atributo o campo *NIF (número de identificación fiscal)* está formado por ocho dígitos numéricos y una letra la cual se obtiene mediante un algoritmo en que intervienen los ocho dígitos. Este algoritmo (que calcula si una letra determinada cumple la condición de ser correcta para un NIF) es una regla de validación.

2) Dificultad para gestionar el **control de autorizaciones** (seguridad)

Es habitual establecer mecanismos que controlen el acceso a los datos de los diferentes usuarios en función de su categoría o perfil para evitar que se produzcan accesos indebidos, por lo que se otorgan un identificador y una clave (o contraseña) a cada usuario. El identificador permite discriminar los usuarios reales del sistema de los que intentan acceder de manera no autorizada; la clave permite autenticar al usuario.

Para implementar este control de autorizaciones no suele haber ningún problema. Pero, por el hecho de que los elementos (ficheros y programas)

están distribuidos en el sistema sin organizar, no es posible establecer un método eficaz que controle el acceso a cada elemento. Esto provoca que un usuario pueda acceder a programas o datos determinados sobre los que no tiene permiso de acceso, puesto que, al modificar el programa para que tenga estos permisos o no, es difícil prever y crear los controles de seguridad adecuados. En este caso, puede suceder que, de manera no deseada, se permita a usuarios no autorizados acceder a datos sobre los que no deberían tener permiso de acceso.

(Des)control de autorizaciones a terceros

Un problema añadido se produce cuando un usuario que puede otorgar permisos para acceder a unos elementos determinados los ha otorgado a terceras personas y más tarde se le retiran. Puede quedar la duda de si se han eliminado o no los permisos para estas terceras personas.

3) Falta de **control de concurrencia**

La concurrencia es el acceso simultáneo de varios usuarios a los mismos datos. Cuando no hay un programa que controle el acceso de los diferentes usuarios que acceden a la vez al mismo fichero para modificar los datos que contiene, no se puede saber cuál de los datos modificados se guardará ni en qué orden se han producido estas modificaciones. Esto provoca que la información obtenida sea inconsistente respecto a las operaciones llevadas a cabo.

4.2. Problemas respecto a los datos

Los principales problemas causados por la estructura física de los datos (es decir, por el modo como éstos están almacenados en los diferentes ficheros) son los siguientes:

1) **Redundancia** (repetición de información)

Es la repetición innecesaria de información en varios ficheros. También se produce cuando hay datos que no aportan información adicional, puesto que se pueden obtener o calcular a partir de otros.

La redundancia se debe a la ausencia de integración de datos en ficheros procedentes de departamentos diferentes o al uso de lenguajes diferentes.

Integridad de los datos

Es la exactitud y consistencia interna de la información almacenada en un sistema de gestión de datos.

Problemas de la redundancia

Asociados a los problemas de redundancia están los derivados de la **inconsistencia**, la **falta de integridad** de los datos, el derroche del espacio de almacenamiento (**coste del espacio** ocupado inútilmente), la dificultad para **mantener actualizadas** los datos comunes y la **recuperación lenta** de los datos.

2) Inconsistencia (difícil actualización)

Como consecuencia de la existencia de información redundante, la actualización de los datos duplicados en varios ficheros se hace muy difícil, puesto que cuando se modifica un dato en algún fichero puede suceder que no se modifique en los otros en los que también se encuentra duplicado; esto hace que la información de los diferentes ficheros no concuerde. El resultado es que la base de datos se vuelve inconsistente y proporciona información incorrecta o contradictoria.

Ejemplo de inconsistencia

Si no se actualiza al mismo tiempo la información de un alumno repetida en diferentes ficheros, ésta puede ser diferente según el departamento donde se localiza, lo que provocaría, por ejemplo, que se mantuvieran dos cuentas corrientes diferentes, en lugar de una única cuenta actualizada.

3) Aislamiento (datos repartidos)

Es la fragmentación de la información que se produce cuando los datos referentes a un objeto se almacenan en ficheros diferentes y es difícil obtener a la vez toda la información relativa a un mismo objeto.

Los usuarios de departamentos o proyectos diferentes interesados en datos de la misma entidad (u objeto) del mundo real mantienen por separado ficheros independientes y programas diferentes para manipular los datos porque cada uno requiere algunos datos que no se encuentran en los ficheros de los demás. Esto, además, complica la escritura de programas.

Gestión de ficheros separados con programas diferentes

Por ejemplo, un usuario del departamento de gestión académica puede mantener un fichero de alumnos con sus calificaciones (para introducirlas en el fichero e imprimirlas se crea el programa pertinente). Un usuario del departamento de contabilidad lleva el control de tasas de los alumnos y su pago en otro fichero. A pesar de que los dos gestionan datos de los alumnos, mantienen ficheros separados (y programas diferentes) porque cada uno necesita otros datos que no puede obtener del fichero del otro.

4) Dificultad de acceso a los datos

Es un problema derivado del aislamiento de la información que consiste en la carencia de un catálogo actualizado que incluya la relación de ficheros de datos y sus contenidos y que permita obtener los datos deseados en un tiempo adecuado. La carencia de este catálogo implica un problema organizativo (es necesario que el administrador del sistema obtenga los datos solicitados) y otro temporal (la demora puede hacer que la información solicitada se vuelva obsoleta antes de ser recibida).

Por otro lado, las funciones de recuperación y actualización de datos de los que disponen los diferentes programas existentes son limitadas (sólo contemplan una serie de opciones específicas) y pueden obligar a reprogramar, con la consiguiente pérdida de tiempo.

5) Dependencia de los programas (y diferencia de formatos)

Los datos se almacenan en varios ficheros (en formatos diferentes o no) que están gestionados por aplicaciones diferentes.

La definición de datos forma parte de los propios programas de aplicación. Por lo tanto, estos programas sólo pueden trabajar con un fichero concreto, cuya estructura está codificada en el mismo programa (utilizando terminología de programación, se dice que la estructura está *declarada* mediante instrucciones no ejecutables).

Cualquier cambio en la estructura de los ficheros (es decir, la organización según el tipo de acceso para el que se definieron: secuencial, relativo, direccional, aleatorio, etc.) puede afectar a los programas de aplicación que los gestionan.

5. Objetivos y características de las bases de datos

El objetivo principal de una base de datos para superar los inconvenientes de la gestión de datos mediante ficheros es **centralizar la información**. Esto permite que los programas **integren los datos** (que antes se encontraban dispersos en ficheros de diferentes departamentos) y mantengan **la información actualizada**.

Los inconvenientes de la gestión de ficheros

Se detallan en el apartado "Problemas de la gestión de ficheros de datos" de este módulo.

Otro objetivo es **estandarizar un modo de gestión de datos** más eficaz, universal e independiente del hardware y del sistema operativo sobre el que se implemente, de manera que se genere un sistema de almacenamiento con una **interfaz de usuario fácil** de consultar y de modificar.

No menos importante es que permiten **optimizar el coste** del soporte de almacenamiento de los datos, que depende de la configuración del SGBD implementado (monousuario, multiusuario, centralizado, distribuido, etc.) y del volumen de la información almacenada.

Con independencia de cómo se organiza la información en una base de datos, para ser considerada como tal, debe cumplir una serie de requisitos o requerimientos que permiten definirla de la manera siguiente:

> Una base de datos está formada por un conjunto estructurado de datos gestionado por un software, de tal manera que se controla el almacenamiento de datos redundantes, los datos son independientes de los programas que los utilizan, las relaciones entre los datos se almacenan junto con éstos y es posible acceder a los datos de varias maneras.

Las **características de las bases de datos** se pueden enumerar resumidamente como sigue:

- **Integración de toda la información** de la organización (ausencia de redundancia y de inconsistencia).
- **Persistencia de los datos** (disponibilidad inmediata para almacenamiento en memoria secundaria).
- **Accesibilidad simultánea y concurrente** para diferentes usuarios (compartición de la información).
- **Descripción unificada** de los datos (almacenamiento conjunto de los datos y de la definición de las estructuras) **e independiente** de las aplicaciones.

- **Independencia** de las aplicaciones respecto a la representación **física de los datos**.

Abstracción de datos y separación entre programas y datos

La característica que permite la independencia de las aplicaciones y los datos es la **abstracción de datos**. Un SGBD ofrece al usuario una representación conceptual de los datos que no incluye muchos detalles sobre su *almacenamiento* ni sobre cómo *se implementan* las operaciones.

El **modelo de datos** es un tipo de abstracción que permite ocultar estos detalles que no interesan a la mayoría de los usuarios de la base de datos y que se sirve de conceptos lógicos (los objetos y sus propiedades e interrelaciones) que son más fáciles de comprender para el usuario.

Hay muchos modelos de datos que permiten ofrecer al usuario esta abstracción de los datos.

Los modelos de datos y los tipos de abstracción

Estos modelos utilizados para la representación de los datos se tratan en los apartados "Modelo, esquema y estado de la base de datos" y "Tipo de abstracción en el diseño de bases de datos".

- **Definición de vistas** parciales de los datos para diferentes usuarios.

- **Gestión de los datos** (manipulación de los datos y mantenimiento de la base de datos).

- **Mantenimiento de la integridad** (calidad y corrección) **y la seguridad** (accesos autorizados) de los datos.

- **Flexibilidad** (utilización de diferentes métodos de acceso con tiempos de respuesta pequeños).

Para cumplir estos objetivos y requisitos, los SGBD, independientemente del tipo al que pertenezcan y de su fabricante, disponen de componentes con funciones muy definidas y tienen una arquitectura estándar denominada *arquitectura de niveles*.

Arquitectura, funciones y componentes de los SGBD

Se presentan en los apartados "Arquitectura de los SGBD" y "Funciones y componentes del SGBD".

Las características definitorias de las bases de datos las diferencian de técnicas anteriores de gestión de datos (como el procesamiento de ficheros tradicionales) y aportan una serie de ventajas, si bien también presentan algún inconveniente. Todos ellos se tratan a continuación.

6. Ventajas e inconvenientes de las bases de datos

En este apartado se exponen las ventajas que presentan las bases de datos (respecto a los ficheros tradicionales), agrupados según si afectan a los datos, los usuarios o la organización. Ahora bien, su utilización puede suponer una serie de inconvenientes en cuanto a su inversión inicial, implantación, integración, normalización de sistemas o rentabilidad, así como también para sus usuarios. En casos muy específicos, es mejor utilizar un sistema de ficheros en lugar de una base de datos, como comentaremos.

6.1. Ventajas

Las ventajas que las bases de datos presentan frente a los sistemas de gestión de datos basados en ficheros son consecuencia de sus características.

La integración de todos los elementos del sistema de base de datos (datos, hardware, software y personal humano) y su control centralizado facilita la disponibilidad de los datos a los usuarios sin depender de los propietarios de porciones físicas de la aplicación y supone el aumento de la seguridad del sistema por los niveles de seguridad diferentes que se implementan en la base de datos.

Las ventajas se pueden referir a los datos, a los usuarios y a la organización:

6.1.1. Ventajas respecto a los datos

1) Disminución de la redundancia

Reducir la redundancia es uno de los objetivos principales de las bases de datos, siempre que no implique el aumento de su complejidad ni la disminución del tiempo de respuesta (o eficiencia).

Si se hace un buen diseño, la redundancia se ve notablemente disminuida, aunque no se evita totalmente. Aunque a menudo interesa un cierto nivel de redundancia controlada.

Redundancia controlada

Los motivos para repetir ciertos datos son básicamente dos:

- La **redundancia lógica** se utiliza para asegurar la integridad de los datos. Para representar las interdependencias existentes entre los objetos del mundo real (que forman parte del dominio del problema), se deben repetir datos que permitan establecer asociaciones entre ellos.

- La **redundancia física** se utiliza por motivos de eficiencia y rendimiento. La fragmentación y replicación de información utilizada en bases de datos distribuidas implica una redundancia deseada que permite mejorar la disponibilidad de los datos, el tiempo de respuesta y el coste de las comunicaciones.

En cualquier caso, la gestión de la información redundante se hace de manera interna en el mismo sistema de base de datos.

2) Prevención de la inconsistencia

La disminución de la redundancia evita la inconsistencia y permite más eficiencia en la validación y la introducción de datos en el sistema.

Propagación de actualizaciones

Si hay un cierto grado de redundancia, debemos constatar que las actualizaciones se graben en todos los lugares donde aparecen los datos. Este proceso es posible gracias a la propagación de actualizaciones que el sistema puede hacer automáticamente.

3) Representación de asociaciones entre los datos

Se pueden representar varios tipos de vínculos (más o menos complejos) existentes entre los datos, lo que permite obtener y actualizar con rapidez y eficiencia datos que están mutuamente interrelacionados.

Simplicidad del sistema

La naturaleza del problema que se va a tratar informáticamente es un factor de complejidad de partida que la base de datos ayuda a minimizar mediante representaciones lógicas sencillas que permiten la modificación de sus requisitos, de manera que la inclusión y/o la modificación de nuevos elementos de datos e interrelaciones no conporta una complejidad excesiva.

4) Mantenimiento de la integridad

La eliminación de la redundancia minimiza la falta de integridad de los datos. Aunque se compartan datos entre diferentes usuarios y ubicaciones o se modifiquen los objetos que forman la base de datos, el sistema permite definir restricciones de integridad y garantizar su cumplimiento.

Ejemplos de restricciones de integridad

Algunas restricciones de integridad que derivan de la semántica o el significado de los datos son, por ejemplo, especificar el tipo de dato para los atributos (campos), obligar a que los registros de un fichero estén relacionados con registros de ficheros determinados, imponer que los valores de los registros sean únicos, etc.

5) Suministro de copias de seguridad y recuperación

Si el hardware o el software fallan durante la ejecución de una actualización, el subsistema de copias de seguridad *y recuperación* del SGBD asegura la restauración de la base de datos al estado anterior a la ejecución o permite que se retome la ejecución del proceso en el punto donde fue interrumpido.

6) Almacenamiento de las especificaciones de la base de datos (naturaleza autodescriptiva del sistema).

El sistema no contiene únicamente la base de datos, sino también una definición o descripción completa de la estructura y los elementos de la base de datos mediante metadatos (datos que describen los datos) que incluyen, por ejemplo, tipos y formato de almacenamiento de cada elemento de datos, restricciones sobre los datos, etc. Esta definición se almacena en el diccionario de datos (o catálogo) del SGBD separada de los programas de acceso.

Dado que un SGBD de propósito general no se crea para una única base de datos, el software del SGBD (y también el usuario) puede acceder a cualquier base de datos mediante la extracción de sus definiciones del diccionario de datos, mientras que el software para el procesamiento de ficheros tradicionales sólo puede acceder al fichero de datos específico para el que se ha creado (y la definición de datos está declarada en el propio programa).

7) Independencia entre los programas y los datos

La independencia

Hay que destacar que este aspecto (no tan evidente, a pesar de estar implícito en otras ventajas) es un objetivo importante de las bases de datos.

Los programas de aplicación desarrollados para manipular los datos son independientes de la implementación elegida para las estructuras de la base de datos. Hay dos tipos de independencia:

- **Independencia lógica**, por la que la modificación de la representación lógica general del dominio del problema (los objetos de la base de datos y sus asociaciones) no afecta a los programas de aplicación que manipulan los datos.

- **Independencia física**, por la que la modificación de la estructura física y/o la distribución de los datos en las unidades de almacenamiento no afecta a los programas que manejan los datos ni a la estructura lógica general de la información.

Flexibilidad en la modificación de la organización de los datos

1) Por cambio del soporte físico

Al cambiar los objetos de un esquema de base de datos de un equipo a otro no hay que hacer cambios en los procedimientos de acceso a los datos por el cambio de equipo o de soporte físico. Un ejemplo de esto es el de usuarios que utilizan ordenadores portátiles o con movilidad dentro de una red.

2) Por afinación de la organización física

La modificación de la organización física de los datos para su evolución (por ejemplo, para mejorar el tiempo de respuesta) no afecta a las representaciones de los datos ni a los procedimientos que operan sobre ellos.

En ambos casos, una modificación del diseño (a nivel físico o conceptual) no implica modificaciones en las aplicaciones, y viceversa.

En cambio, en el procesamiento de ficheros tradicionales, la estructura de los ficheros de datos está integrada en los programas de acceso. Por lo tanto, la modificación de la estructura de un fichero puede requerir la modificación de los programas que acceden a él.

8) Compatibilidad entre formatos (importación y exportación)

Permite reconocer información organizada físicamente por otro software de manera diferente de la que utiliza la base de datos. Esto se refiere tanto a la información existente antes de la implantación de la base de datos como a posibles cambios posteriores.

6.1.2. Ventajas respecto a los usuarios

La tecnología de base de datos se ha desarrollado para dar respuesta a las exigencias crecientes de funcionalidad y eficiencia que plantea el usuario. Las ventajas descritas a continuación, si no se indica lo contrario, se consideran referidas a sistemas multiusuario.

1) Posibilidad de aplicación eficiente de restricciones de seguridad

El subsistema de seguridad y autorización del SGBD permite garantizar automáticamente el cumplimiento de restricciones de acceso establecidas por el administrador de la base de datos según el perfil de cada usuario. Sin ellas, la información de una base de datos (centralizada) está en más peligro que la de un sistema de ficheros tradicionales (dispersos).

Varias restricciones de acceso

Los controles de seguridad permiten restringir:

- El **acceso a los datos** (limitan los usuarios que pueden acceder a información delicada o confidencial).
- Las **operaciones de acceso** (permiten a algunos usuarios sólo la recuperación de datos determinados y a otros, su actualización).
- **El acceso al software** del SGBD (permiten al administrador utilizar determinado *software privilegiado* –como el que sirve para crear cuentas y perfiles de usuario– y al operador de consola sólo efectuar determinadas *transacciones programadas*).

2) Suministro de múltiples interfaces de usuario

El SGBD ofrece diferentes interfaces para usuarios de varios niveles de conocimiento técnico y con diferentes necesidades de utilización.

Las interfaces de usuario

Las interfaces de usuario que ofrecen un SGBD se pueden consultar en el subapartado "Interfaces del SGBD", dentro del apartado "Funciones y componentes del SGBD".

Ejemplos de interfaces de usuario

- Hay interfaces de lenguaje de programación para programadores de aplicaciones, lenguajes de consulta para usuarios ocasionales, interfaces de lenguaje natural para usuarios autónomos, etc. Como ***interfaces gráficas de usuario*** (GUI) debemos mencionar a las interfaces controladas por menús para usuarios autónomos, los formularios y las interfaces de transacciones programadas para usuarios paramétricos y las interfaces web para el acceso a través de Internet.

3) Soporte de múltiples vistas de datos

Una *vista* es una representación externa de la base de datos que sólo considera un subconjunto de los datos almacenados. A pesar de que la información que forma parte del dominio de un problema o sistema es única, un SGBD multiusuario proporciona mecanismos para definir múltiples vistas globales y parciales para los distintos programas de aplicación y grupos de usuarios sin perder el carácter integrador de la base de datos.

Una vista puede contener datos virtuales que no están explícitamente almacenados, sino que son derivados de datos almacenados (como, por ejemplo, el valor calculado de una media) o la edad de un alumno calculada a partir de la fecha de nacimiento. De hecho, la mayoría de los usuarios no necesita saber si utiliza datos almacenados o derivados.

Diferentes vistas de los datos

Hay usuarios que necesitan vistas particulares que les permitan actualizar determinados datos relevantes para ellos.

- Tendrán diferentes vistas el usuario que sólo gestiona los expedientes de los alumnos de un centro y el usuario que comprueba que los alumnos han cursado todos los prerrequisitos de cada curso en el que están matriculados.

Como ventaja adicional, la utilización de las vistas constituye en sí misma un mecanismo de seguridad importante, puesto que proporciona un control discrecional de autorizaciones.

4) Disponibilidad de información actualizada

El subsistema de control de concurrencia y de recuperación del SGBD hace posible que todos los usuarios puedan disponer, de manera inmediata y transparente, de los datos actualizados por otros usuarios.

5) Rapidez y sencillez de acceso a la información

La capacidad de procesamiento del sistema permite, en cada momento, una recuperación rápida y sencilla de la información requerida por el usuario. Esto se concreta en dos aspectos:

- **Eficiencia**

  La base de datos asegura un bajo tiempo de respuesta adecuado al usuario y permite el acceso simultáneo a los conjuntos de elementos de datos por el mismo procedimiento, la misma aplicación o el mismo usuario o por unos diferentes.

- **Capacidad de acceso**

  La base de datos responde, en un tiempo aceptable, a cualquier consulta sobre su información, sin restricciones importantes en cuanto a los elementos de datos, sus relaciones, la agrupación (o el formato solicitados).

6) Accesibilidad múltiple y simultánea

Varios usuarios pueden compartir la base de datos y acceder a ella simultáneamente. Esto se traduce en las ventajas siguientes:

- **Compartición de datos**

  Las diferentes partes de la base de datos se pueden compartir entre varios grupos de usuarios sin que se produzcan conflictos y manteniendo la integridad de los datos.

> **La independencia facilita la compartición**
>
> La independencia entre datos y aplicaciones facilita la compartición y la disponibilidad de los datos tanto para los usuarios como para las aplicaciones, sin necesidad de modificar la base de datos.

- **Procesamiento de transacciones multiusuario**

  El SGBD incluye un gestor de control de concurrencia que asegura que cuando varios usuarios intentan actualizar los mismos datos, lo hagan de manera controlada para que el resultado sea correcto. Para que las *transacciones concurrentes* operen adecuadamente, los accesos sucesivos se sitúan en cola, de manera que, hasta que un usuario no acabe de hacer modificaciones, el resto no puede hacer otras nuevas.

> **Procesamiento de transacciones en línea (*online transaction processing*, OLTP)**
>
> Un ejemplo de esto se presenta cuando varios encargados de reservas intentan asignar una localidad de un espectáculo (o medio de transporte) a un cliente. El SGBD garantiza que, en un momento determinado, sólo un usuario tenga acceso a la reserva de una localidad concreta para asignarla al espectador (o pasajero) correspondiente.

7) Flexibilidad para atender nuevos requerimientos de información

Cuando cambian los requerimientos de información, es posible modificar la estructura de la base de datos sin afectar a los datos almacenados ni a los programas de aplicación existentes gracias a la independencia entre la base de datos y las aplicaciones que los manipulan.

Adaptación a requerimientos nuevos

Este caso se da cuando es necesario añadir un fichero nuevo o ampliar los elementos de datos (campos) de un fichero existente para atender las necesidades de información de un nuevo grupo de usuarios.

8) Coherencia de los resultados

El hecho de que en todos los tratamientos se utilicen los mismos datos (y que se prevenga la inconsistencia) provoca que los resultados sean coherentes.

6.1.3. Ventajas respecto a la organización

Aparte de los aspectos mencionados anteriormente, hay otras implicaciones de la utilización de bases de datos (muchas de ellas tienen que ver con el coste) que pueden resultar beneficiosas a la organización. De hecho, las bases de datos son un instrumento de influencia decisiva en las organizaciones.

1) Integración de toda la información de la organización

La base de datos da servicio a toda la organización, o a una parte importante de ésta, y no únicamente a algunos departamentos o usuarios individuales, de manera que se evita la redundancia de datos y los problemas de inconsistencia.

2) Eficiencia del almacenamiento físico

La disminución de la redundancia y las técnicas de compactación suponen la reducción del espacio de almacenamiento y de los costes asociados al consumo de espacio físico.

3) Reducción del coste de almacenamiento y de mantenimiento

Esta ventaja es consecuencia de la eficiencia del almacenamiento físico y de la independencia lógica y física de los datos.

4) Reducción del coste de formación del personal

El hecho de que la mayoría de los SGBD del mercado dispongan de herramientas que ofrecen un uso fácil repercute favorablemente, a largo plazo, en el coste de formación del personal.

5) Economía de escala

La consolidación de los datos y las aplicaciones reduce el derroche producido por el solapamiento de actividades del personal de procesamiento de datos en diferentes departamentos. Esto implica la reducción del coste total de opera-

ción y gestión, puesto que la organización puede invertir en sistemas globales de procesamiento, almacenamiento y/o comunicación más potentes, en lugar de hacerlo en equipamiento de capacidad inferior para cada departamento.

6) Capacidad para establecer normas

El administrador de la base de datos puede garantizar el desempeño de normas para la representación de los datos. Él es responsable de definir e imponer a los usuarios las normas de la organización para facilitar la comunicación y la cooperación entre varios departamentos o proyectos.

Ejemplos de definición de normas

La normalización permite unificar, por ejemplo: los nombres y los formatos de los elementos de datos (y así facilita la escritura de programas, el intercambio de información y la migración de datos entre sistemas); la terminología (para documentar los datos); las estructuras de formularios e informes, etc.

7) Reducción del tiempo de creación de aplicaciones

Si bien se tarda más en diseñar e implementar una base de datos desde cero que en crear una aplicación de ficheros especializada, cuando la base de datos está operativa se requiere mucho menos tiempo (alrededor de una quinta parte) para la creación de una nueva aplicación (como la que permite obtener una información determinada en un informe) con los recursos del SGBD que con un sistema tradicional de ficheros.

6.2. Inconvenientes

A pesar de sus ventajas, un sistema de base de datos puede suponer una serie de inconvenientes que supongan barreras a su implantación. Antes de tomar esta decisión, hay que evaluar si la base de datos soluciona los problemas que plantea la información en la organización.

Los costes adicionales que genera la implantación de una base de datos se deben a los hechos siguientes:

1) Inversión inicial elevada

Las primeras fases de implantación de una base de datos implican los inconvenientes siguientes:

a) Instalación costosa de nuevos equipos y herramientas que se adecuen a las utilidades del sistema. Esto incluye la actualización o ampliación del **hardware** existente sobre el que funcione el nuevo SGBD, la conversión del **software** (sistemas operativos, compiladores, etc.) y la adquisición y el mantenimiento del propio SGBD.

b) Coste de capacitación de personal especializado (final e informático) generado en las primeras fases de diseño, puesta en marcha, administración y utilización.

2) Implantación larga y difícil

Las dificultades que van apareciendo a lo largo de la implantación de la base de datos pueden hacer que se superen ampliamente los plazos previstos inicialmente para estar plenamente operativa.

3) Rentabilidad a medio (o largo) plazo

A pesar de que los beneficios en cuanto a velocidad de proceso y obtención de resultados son inmediatos, la rentabilidad no se obtiene hasta que no se ha afinado el sistema. Esto significa que debemos considerar el sistema de base de datos globalmente como instrumento para la obtención de beneficios en otras áreas de la empresa.

4) Ausencia de normalización de los SGBD comerciales

Los SGBD comerciales difícilmente cumplen todas las especificaciones definidas por los estándares de bases de datos y además tienen características añadidas que los hacen incompatibles entre ellos.

Pero actualmente este inconveniente se puede obviar, puesto que las técnicas de compartición permiten utilizar los datos donde cualquier SGBD con independencia con que se almacenen.

5) Problemas de integración de bases de datos

Cuando se quieren obtener bases de datos distribuidas por integración de otras, creadas por diseñadores diferentes o no, se pueden producir problemas de redundancia o de obtención de objetos diferentes incoherentes porque no existe un diseño único para las bases de datos.

6) Riesgo de frustración de los usuarios

Existe un riesgo evidente de decepción por parte del usuario (especialmente directivo) que puede hacer pasar por alto el potencial real de un sistema de base de datos.

Las expectativas frustradas por el fracaso de proyectos de bases de datos se deben a varios factores: diseño inadecuado de la base de datos, implementación incorrecta de las aplicaciones del sistema, asunción de funcionalidades o prestaciones que sólo son estudios teóricos (de hecho, la divergencia con la implantación práctica de innovaciones en SGBD comerciales es considerable), etc.

7) Otros inconvenientes

Otros inconvenientes de los sistemas de base de datos se deben al propósito generalista que presenta el SGBD para la definición y el procesamiento de los datos, a los costes que implica la necesidad de ofrecer funciones de seguridad, control de concurrencia, recuperación e integridad, etc.

6.3. Casos en los que no se aconseja usar un sistema de base de datos

Conviene utilizar un sistema de ficheros tradicionales en lugar de un sistema de base de datos en las situaciones siguientes:

- El sistema y las aplicaciones existentes están muy definidos, son sencillos y, previsiblemente, no deben cambiar.

- No hace falta el acceso multiusuario y simultáneo a los datos.

- El tiempo de respuesta riguroso requerido por los programas es difícil de cumplir con un SGBD.

7. Elementos de un sistema de base de datos

Un sistema de base de datos está formado por cuatro recursos o elementos principales: el recurso principal es la propia base de datos y el secundario, el software relacionado con ella; los otros son el hardware y las personas que intervienen en el sistema.

Se puede añadir un quinto elemento: los procedimientos (que son las instrucciones y reglas que dirigen el diseño y la utilización de los componentes de software).

7.1. El contenido: los datos

El contenido de la base de datos lo forman los datos, que constituyen la parte esencial del sistema de base de datos y justifican su existencia. Los datos de una base de datos siguen una jerarquía establecida y deben presentar una serie de propiedades o características.

Propiedades de los datos de una base de datos

Los datos almacenados en una base de datos deben ser:

- **Perdurables**: deben permanecer invariables en el tiempo, no deben ser efímeros.
- **Estructurados**: deben tener una organización que facilite la compartición entre diferentes usuarios.
- **Operacionales y transaccionales**: deben ser manejables aplicando operadores para obtener los resultados deseados.
- **Significantes**: deben tener un significado o sentido semántico concreto.
- **Íntegros**: deben reflejar, sin contradicciones, una realidad existente.

Jerarquía de los datos

Los datos de una base de datos se almacenan siguiendo la jerarquía siguiente:

Bit → Carácter → Dato → Registro → Fichero → Base de datos

Esto se muestra en el esquema siguiente:

Concepto	Descripción	Ejemplo	Comentario
Bit	Dígito binario.	0	Puede ser 0 o 1.
Byte = carácter	Conjunto ordenado de 8 bits (octeto) que representa un carácter.	010001101	Representa al carácter "m".
Campo = dato	Conjunto ordenado de bytes que representa un único elemento (o ítem) de datos. También se utiliza el término ***atributo***.	Mercedes	Nombre del alumno.
Registro	Conjunto de campos relacionados que caracterizan un único objeto, persona, etc. También se utiliza el término ***entidad***.	Datos del alumno	Incluye nombre, apellidos, fecha de nacimiento, DNI, dirección, teléfono, etc.
Fichero	Colección de registros relacionados referidos a un conjunto de objetos de un mismo tipo.	Alumnos	Incluye todos los datos personales relativos a todos los alumnos de la universidad.
Base de datos	Conjunto de ficheros relacionados que representa el dominio de un sistema.	Datos de la universidad	Incluye todos los ficheros relativos a alumnos, profesores, estudios, asignaturas, aulas, etc.

Fuente: adaptado de Parker y Case (1993)

7.2. El equipo lógico: el software

Básicamente, hace referencia al programa de gestión de datos instalado, que permite a los usuarios manipular los datos sin necesidad de conocer cómo se han almacenado físicamente. Este programa recibe el nombre de *sistema gestor de bases de datos* (SGBD) y se puede considerar el entorno de la base de datos.

En sentido amplio, también engloba al sistema operativo, el software de comunicaciones, las utilidades, los programas de aplicación que procesan los datos, los compiladores de lenguajes de programación, etc.

7.3. El equipo físico: el hardware

Engloba el ordenador sobre el que se instala el SGBD y donde se almacenan físicamente los datos (y otros objetos) de la base de datos y los dispositivos que permiten la gestión. Es decir, incluye el disco duro, la CPU, las unidades de entrada/salida, etc.

En bases de datos distribuidas, el hardware también comprende los equipos conectados en red al equipo servidor de datos.

7.4. El personal humano: los usuarios

El personal humano que interacciona con un sistema de base de datos se ha considerado tradicionalmente un componente fundamental para el funcionamiento correcto y adecuado del SGBD.

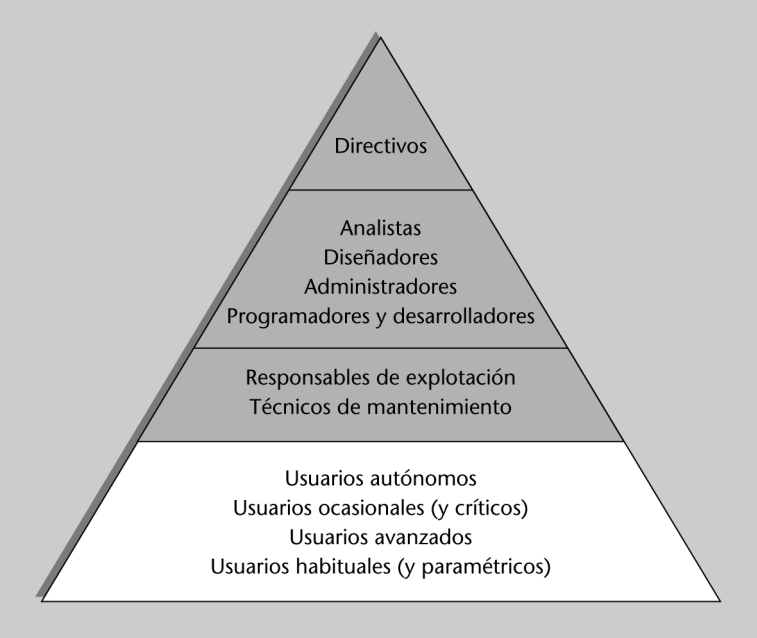
El número y el tipo de usuarios dependen de la magnitud de la base de datos

En la definición, la construcción y la manipulación de una base de datos personal pequeña (como una lista de direcciones y teléfonos) interviene una única persona. En cambio, en el diseño, la utilización y el mantenimiento de una base de datos grande o corporativa (como puede ser el conjunto de declaraciones de la renta que gestiona la Agencia Tributaria) participan muchas personas.

8. Usuarios de las bases de datos

En organizaciones grandes los usuarios de las bases de datos se pueden dividir en dos tipos principales: el personal informático y los usuarios finales.

Figura 3. Pirámide de usuarios de las bases de datos



8.1. Personal informático

El personal humano que interviene en el desarrollo y el mantenimiento de bases de datos se puede clasificar en las categorías siguientes:

1) Directivos

Sus tareas principales son la **organización y coordinación** del departamento de informática, la definición de las bases de datos que se deben desarrollar y la organización de cursos de formación.

2) Analistas de sistemas

Los analistas de sistemas planifican y controlan el desarrollo de las bases de datos aprobadas por la dirección y realizan los estudios de viabilidad necesarios. A partir del **análisis de necesidades** de tratamiento de información de los usuarios finales (especialmente los paramétricos) desarrollan especificaciones para transacciones programadas que satisfagan estos requerimientos y generen la documentación analítica necesaria para el trabajo de los programadores.

Los usuarios paramétricos

Los que usan transacciones programadas se describen en el subapartado "Usuarios finales" de este mismo apartado.

Los analistas tienen funcionalidad de administradores de bases de datos cuando éstas ya están creadas.

3) Diseñadores de bases de datos

Se encargan de tareas previas a la implementación de la base de datos, como por ejemplo identificar los datos (ABD) que se almacenarán en la base de datos y elegir las **estructuras** apropiadas para presentarlos y almacenarlos.

Los diseñadores se comunican con los futuros grupos de usuarios de la base de datos con objeto de comprender sus necesidades y de implementar un diseño que satisfaga sus requerimientos de datos y de procesamiento, así como desarrollar las vistas correspondientes.

A menudo forman parte del personal de administración de la base de datos y, una vez acabado su diseño, pueden asumir otras responsabilidades.

4) Programadores de aplicaciones

Mediante un lenguaje de programación, implementan las especificaciones determinadas por los analistas de sistemas en forma de **programas**, probando, depurando y manteniendo las transacciones programadas que los usuarios paramétricos utilizarán para interactuar con la base de datos.

Para llevar a cabo sus tareas, estos usuarios, también llamados ingenieros de software, deben conocer perfectamente todas las capacidades del SGBD (igual que los analistas de sistemas).

5) Administradores de bases de datos

Como principales usuarios informáticos de la base de datos, los administradores (o gestores) de bases de datos (ABD) son responsables de supervisar y gestionar los **recursos** del sistema (la propia base de datos y el software relacionado con ésta) y de adquirir los recursos de software y hardware necesarios. Se encargan de diseñar y modificar la estructura de almacenamiento de la base de datos, determinar las estrategias de acceso a los datos, coordinar y vigilar su utilización, comprobar el rendimiento del sistema y resolver los problemas de falta de eficiencia con objeto de garantizar la **calidad** y la **disponibilidad** de los datos.

Por lo tanto, deben gestionar la **seguridad** de la base de datos respecto a los accesos y la **integridad** de los mismos. Definen los esquemas externos de cada usuario, crean protecciones de acceso para evitar violaciones de la seguridad y aseguran una gestión correcta de las transacciones que evite la pérdida o corrupción de los datos. También son responsables de definir la estrategia de **recuperación** con la creación de las copias de seguridad necesarias.

Responsabilidades del administrador de bases de datos

Las tareas más importantes del ABD tienen que ver con los siguientes aspectos:

- **Recuperabilidad**: crear y probar copias de seguridad de la base de datos.
- **Integridad**: verificar o ayudar a verificar la integridad de los datos.
- **Seguridad**: implementar y gestionar controles de acceso a los datos.
- **Disponibilidad**: asegurar el acceso a los datos en todo momento (24 horas × 365 días).
- **Rendimiento**: asegurar el mínimo tiempo de respuesta.
- **Soporte a pruebas**: proporcionar datos de funcionamiento y rendimiento a programadores y diseñadores para mejorar la eficiencia de la base de datos.

6) Responsables de explotación

Se ocupan de **poner en marcha** las bases de datos y los sistemas que trabajan diariamente.

Hay otros usuarios informáticos que, a pesar de que son cruciales para que el usuario final pueda utilizar el sistema, no suelen tener un interés claro en el contenido de la base de datos. Se trata de personal relacionado con el diseño, la creación y el funcionamiento del software del SGBD y su entorno. Son los siguientes:

7) Desarrolladores del SGBD

Diseñan e implementan los **módulos** y las **interfaces** del SGBD en forma de paquetes de software.

8) Desarrolladores de herramientas

Diseñan e implementan **herramientas y utilidades** del SGBD que facilitan el diseño y la utilización del sistema y ayudan a mejorar su rendimiento.

9) Técnicos de mantenimiento

Son profesionales de administración del sistema y operadores responsables de dar solución a los problemas de hardware y de software producidos durante el **funcionamiento** del sistema de base de datos.

8.2. Usuarios finales

La base de datos existe principalmente para que el usuario final consiga los objetivos personales o empresariales propuestos. Su trabajo requiere acceder a la base de datos para realizar consultas y actualizaciones, generar informes, e interactuar con ella directamente mediante una interfaz gráfica. Los usuarios finales se pueden clasificar en las categorías siguientes:

Las aplicaciones verticales

Se describen en el punto sobre lenguajes y aplicaciones del subapartado "Lenguajes de base de datos" dentro del apartado "Funciones y componentes del SGBD".

1) Usuarios habituales

A pesar de no ser usuarios especializados, tienen ciertos conocimientos que les permiten usar sistemas guiados de comunicación con los datos (como asisten-

tes y herramientas visuales) o *aplicaciones verticales* para consultar o modificar datos. Cada grupo de usuarios dispone de vistas externas según sus permisos de acceso y de ejecución específicos.

Un tipo específico de usuario habitual es el usuario paramétrico.

Las transacciones programadas

Son procedimientos o conjuntos de operaciones (que pueden ser de inserción, de eliminación, de modificación y/o de recuperación) invocados por los usuarios paramétricos desde una interfaz de usuario (como un formulario) para la ejecución correcta de las operaciones de almacenamiento y recuperación de información.

- **Usuarios paramétricos**

Se trata de usuarios cotidianos dedicados casi exclusivamente a trabajar con la base de datos que no necesitan conocer demasiado sobre los recursos del SGBD. Basta con que aprendan a realizar un tipo de consultas y actualizaciones estándar, llamadas *transacciones programadas*, que se han diseñado, implementado y probado cuidadosamente para ellos. Su función principal puede consistir, por ejemplo, en introducir datos de manera masiva y constante.

Los lenguajes e interfaces

Los lenguajes e interfaces que permiten a los usuarios paramétricos usar transacciones programadas se tratan en el apartado "Funciones y componentes del SGBD".

Estos usuarios, también denominados *terminales*, suelen formar parte de la estructura informática de la empresa en el nivel de operadores de consola o administrativos y representan una parte importante de la totalidad de usuarios finales.

Ejemplos de usuarios paramétricos

Los usuarios paramétricos pueden ser de tipos muy variados:

- *Cajeros* de entidades bancarias, que revisan saldos y realizan depósitos y reintegros de dinero.
- *Encargados de reservas* de líneas aéreas, hoteles, compañías de alquiler de coches o espectáculos que revisan la disponibilidad y hacen reservas.
- *Personal de correos*, que introduce la identificación de los paquetes (mediante códigos de barras) e información descriptiva (a través de botones) con el objetivo de actualizar la base de datos centralizada de los paquetes recibidos y de los que están en camino.

2) Usuarios avanzados

Están suficientemente familiarizados con la mayor parte de los recursos del SGBD (utilidades, LMD, etc.) como para ejecutar procesos propios e implementar aplicaciones que cumplan sus requerimientos más o menos complejos. Necesitan una buena gestión de la información que es procesada por otro sistema (comercial o desarrollado por ellos) y, por lo tanto, utilizan el SGBD como submódulo o herramienta para el desarrollo de otros sistemas particulares.

Funcionalmente, los usuarios avanzados se corresponden con usuarios expertos que utilizan los datos para realizar análisis de soporte a la toma de decisiones en la organización.

Ejemplos de usuarios avanzados

Pueden ser ingenieros, científicos, analistas de negocio o, incluso, diseñadores de programas no relacionados con los procesos habituales que se llevan a cabo en la base de datos.

Un ejemplo de este grupo son los especialistas en desarrollo de *sistemas expertos*, que utilizan el SGBD para la gestión de la *base de hechos* y la *metabase de conocimientos* del sistema experto.

3) Usuarios ocasionales

Acceden a la base de datos esporádicamente y suelen requerir información diferente en cada ocasión. Para especificar sus peticiones utilizan interfaces con un lenguaje de consulta de alto nivel, pero sin emplear órdenes estructuradas. Con independencia de sus conocimientos en tecnología de bases de datos, para recuperar y manipular la información suelen utilizar pocos recursos del SGBD.

Un tipo de usuario ocasional muy concreto es el usuario crítico.

- **Usuarios críticos**

Normalmente se trata de **personal de gerencia** o perteneciente al *staff* de la organización, que requiere, en un tiempo mínimo, información de la base de datos en un formato, con un detalle y con unos requisitos no previstos con antelación.

Dado que la mayoría de los SGBD disponen de *lenguajes de cuarta generación* integrados (que permiten resultados eficaces en el desarrollo de procesos específicos como, por ejemplo, la generación de informes), el usuario no necesita la asistencia de personal técnico (programadores) para realizar consultas no predefinidas. Sin embargo, el tiempo de respuesta es alto y, a menudo, los resultados no son los deseados, lo que genera la **crítica generalizada** de este usuario que, por otro lado, suele tener poder decisorio sobre aspectos de inversión y contratación dentro de la organización.

Los lenguajes de cuarta generación (4GL, *4th generation languages*)

Son lenguajes de muy alto nivel que suelen combinar elementos procedimentales con elementos declarativos. Básicamente, facilitan el tratamiento de la base de datos, pero también la definición de menús, cuadros de diálogo, formularios de pantalla e informes.

4) Usuarios autónomos

Mantienen bases de datos personales utilizando paquetes de software específicos que tienen interfaces de uso fácil (de tipo gráfico o basadas en menús), para los que suelen adquirir una gran habilidad.

Un ejemplo de esto es el usuario de un paquete fiscal que almacena datos financieros personales.

9. Modelo, esquema y estado de la base de datos

Una característica fundamental de un sistema de base de datos es que proporciona un cierto nivel de abstracción de los datos porque oculta detalles de almacenamiento que la mayoría de los usuarios no necesita conocer.

La abstracción es la acción y el efecto de abstraer, es decir, el proceso mental innato del ser humano de esconder los detalles y centrarse en aquello que es esencial. Su objetivo es buscar las propiedades comunes de una realidad o de un conjunto de objetos del mundo real, reduciendo la complejidad y ayudando a comprenderla. Mediante este proceso, los elementos o las calidades de una realidad son discernidos de los otros para considerarlos aisladamente, o esta realidad es considerada en su propia esencia.

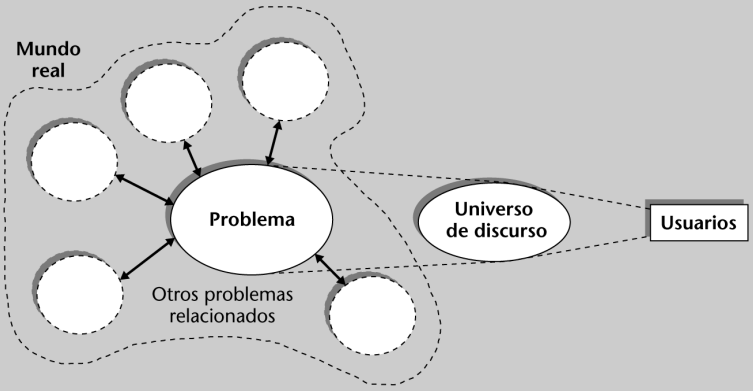
9.1. Modelo de datos

Un **modelo de datos** es un instrumento que proporciona los medios necesarios para conseguir el proceso de abstracción que conduce, de la parte del mundo real que interesa estudiar, denominada *universo del discurso*, al mundo de los datos.

Universo de discurso de un problema del mundo real

Un modelo de datos permite representar cualquier fenómeno real o abstracto, es decir, cualquier **problema** sobre el que se quiere obtener información para su conocimiento. Para la representación del conocimiento sobre un problema es necesaria su caracterización.

Figura 4. Universo de discurso y límites de un problema del mundo real



Definición y ejemplos de modelo

Un modelo es la simulación o representación (gráfica, matemática, conceptual, etc.) del comportamiento de un sistema material o inmaterial del mundo real. Es la representación simplificada de parte de la realidad que se toma como objeto de estudio con objeto de reproducirla.

Son ejemplos de ello un modelo matemático de la distribución de la materia al universo, un modelo meteorológico, un modelo gráfico de una molécula, un modelo numérico de una hoja de cálculo o sobre operaciones financieras.

El problema que debemos representar se puede considerar un sistema en el que intervienen una serie de propiedades que se pueden interrelacionar con el resto de propiedades del sistema. Un **sistema** es un conjunto de elementos cuyo comportamiento queda determinado por sus propiedades e interdependencias. Para el estudio de un sistema, es necesaria la simplificación (basada en la abstracción) del problema que representa, mediante la determinación de sus límites respecto al resto de problemas o fenómenos existentes y las propiedades de interés (que representan los datos que intervienen en el sistema).

El **universo de discurso** (a veces denominado *minimundo*) es la visión concreta que tienen del problema del mundo real los usuarios de la base de datos (en especial, el diseñador).

Un modelo de datos es una herramienta intelectual (un conjunto de conceptos, reglas y convenios definidos explícitamente) que permite describir la estructura de una base de datos en niveles de abstracción diferentes. El concepto de estructura de una base de datos hace referencia a elementos como los tipos de objetos, sus asociaciones (o interrelaciones) y las restricciones que deben cumplir.

El uso de *modelo* en el entorno de bases de datos

- No se debe confundir la **base de datos** considerada como *modelo de la realidad* con el concepto **modelo de base de datos** que aquí se trata.
- La denominación correcta es **modelo de base de datos**, a pesar de que se suele utilizar la expresión **modelo de datos**, puesto que permite modelizar los datos (a pesar de que algunos autores consideran que la expresión no está bien elegida porque *es menos un modelo en sí mismo que un marco para concebir modelos del mundo real*).

Los modelos de datos se basan en la abstracción para definir categorías o tipos de objetos, puesto que la utilización del ordenador requiere la tipificación de los datos tratados.

9.1.1. Ocurrencia y tipo

Determinados elementos de una base de datos pueden tener dos acepciones: tipo y ocurrencia.

- El **tipo** es una categoría generalizada que describe un conjunto de elementos más específicos que tienen las mismas propiedades. Los tipos equivalen a las categorías que se utilizan para organizar información del mundo real. Por ejemplo, los seres vivos se organizan en animales y vegetales.

  En el entorno de los modelos de datos, los conceptos categoría, clase o tipo son sinónimos. El término más utilizado es *tipo*, mientras que *clase* se suele utilizar en los modelos orientados al objeto.

- Una **ocurrencia** es un caso concreto de un tipo. Las ocurrencias comparten propiedades similares, pero cada elemento tiene su propio valor (o conjunto de valores) para cada propiedad. Por ejemplo, un geranio y un roble son ocurrencias de vegetales que, a pesar de que comparten determinadas propiedades, tienen características diferenciales.

  El término más utilizado es *ocurrencia*. Sin embargo, también se utiliza a menudo el término *instancia*, aunque su significado no responde al concepto expuesto. Otros sinónimos de ocurrencia más apropiados pero menos utilizados son *ejemplar*, *realización* o *extensión*.

Instanciación y clasificación

Se relacionan con ocurrencia y tipo tal como se describe en el apartado "Tipos de abstracción en el diseño de bases de datos".

Una ocurrencia es el resultado de una instanciación y un tipo es el resultado de una clasificación.

Las dos acepciones descritas se aplican a los conceptos:

- **Tipo de objeto**: es la conceptualización de un objeto genérico del mundo real como clase.

  Por ejemplo, no un estudiante concreto ni el conjunto de estudiantes, sino la abstracción ESTUDIANTE.

  Generalmente, los diferentes tipos de objetos del universo de discurso se indican con mayúsculas. Por ejemplo, COCHE, VEHÍCULO, DOCUMENTO, PROFESOR, ESTUDIANTE, PERSONA.

- **Ocurrencia de objeto**: es la conceptualización de un objeto concreto del mundo real como individuo. Por ejemplo, el estudiante Miguel Nebot Jover.

  Cada ocurrencia de objeto pertenece a un tipo de objeto genérico. Todas las ocurrencias de un mismo tipo se caracterizan por tener las mismas propiedades (o atributos). Por ejemplo, todos los estudiantes tienen nombre, fecha de nacimiento, número de matrícula, DNI, etc.

Utilización de los términos *objeto* y *ocurrencia*

Para simplificar, con frecuencia se utiliza simplemente el término *objeto* para referirse a un "tipo de objeto", y la expresión **ocurrencia de objeto** para referirse a un objeto concreto de este tipo.

Esta extralimitación en el uso del lenguaje no implica ningún error de interpretación, dado que habitualmente por el contexto se puede deducir a cuál de las dos acepciones se hace referencia (ocurrencia o tipo). Aunque si no queda suficientemente claro, debemos especificarlo.

9.2. Esquema de la base de datos

En cualquier modelo de datos es importante la distinción entre la descripción de la base de datos y la propia base de datos; es decir, entre el esquema y el estado de la base de datos.

El **esquema** es la descripción de la estructura de la base de datos, que se especifica durante el proceso de diseño de la base de datos y, en principio, es invariable en el tiempo.

Los **elementos del esquema** son cada uno de los objetos del esquema, sus asociaciones, las propiedades y las restricciones.

La mayor parte de los modelos de datos utilizan ciertas convenciones para representar los esquemas. La representación de un esquema se denomina **diagrama del esquema** y presenta la estructura de todos los tipos de objetos, pero

no las ocurrencias de objetos. Este diagrama visualiza únicamente algunos aspectos del esquema como, por ejemplo: los nombres de los tipos de objeto, los nombres de las propiedades y algunos tipos de restricciones. En cambio, no aparecen reflejados en él los tipos de datos de cada propiedad, o algunos tipos de restricciones que son muy difíciles de representar.

Ejemplo de diagrama del esquema

El diagrama siguiente representa el esquema de una base de datos que almacena información de las asignaturas, los estudiantes y las calificaciones de los diferentes estudios o facultades de una universidad.

ASIGNATURA

Código asignatura	Nombre asignatura	Número de créditos	Código estudios

ESTUDIOS

Código estudios	Nombre estudios

ESTUDIANTE

Código estudiante	Nombre y apellidos estudiante	DNI estudiante	Teléfono estudiante	Fecha nacimiento

PROFESOR

Código profesor	Nombre y apellidos profesor	DNI profesor	Teléfono profesor	Dirección profesor

CURSO

Código curso	Código asignatura	Código profesor	Semestre-año

CALIFICACIÓN

Código estudiante	**Código curso**	Calificación

Con mayúsculas se indican los nombres de los tipos de objeto y encuadrados, los nombres de sus propiedades (atributos). En negrita aparecen los conjuntos de atributos que forman la clave primaria.

Atributo y clave primaria

Son conceptos que se desarrollan con detalle en el módulo "El modelo relacional y el álgebra relacional".

9.3. Estado de la base de datos

Los datos almacenados en la base de datos pueden cambiar muy a menudo; por ejemplo, cuando se añade una nueva ocurrencia de un elemento de datos.

El conjunto de valores que toman los objetos de un esquema en un momento dado recibe el nombre de **estado de la base de datos**.

También se denomina *conjunto actual de ocurrencias* de la base de datos, puesto que en un estado concreto cada elemento del esquema tiene su propio conjunto de ocurrencias. Por ejemplo, el conjunto de estudiantes individuales (los registros) son las ocurrencias del elemento ESTUDIANTE. También se pueden utilizar las denominaciones *ocurrencia del esquema* o *extensión del esquema*.

A un esquema concreto le pueden corresponder muchos estados de la base de datos. Cada vez que se modifica el valor de un elemento de datos o que se añade o se elimina un registro, la base de datos pasa de un estado a otro.

Los estados en los que se puede encontrar una base de datos son:

- **Estado vacío** (o sin datos): cuando se acaba de definir una base de datos nueva, es decir, cuando se ha especificado su esquema.
- **Estado inicial**: después de cargar los datos por primera vez.
- **Estado actual**: en cualquier momento (en general, después de ejecutar cualquier operación de actualización).

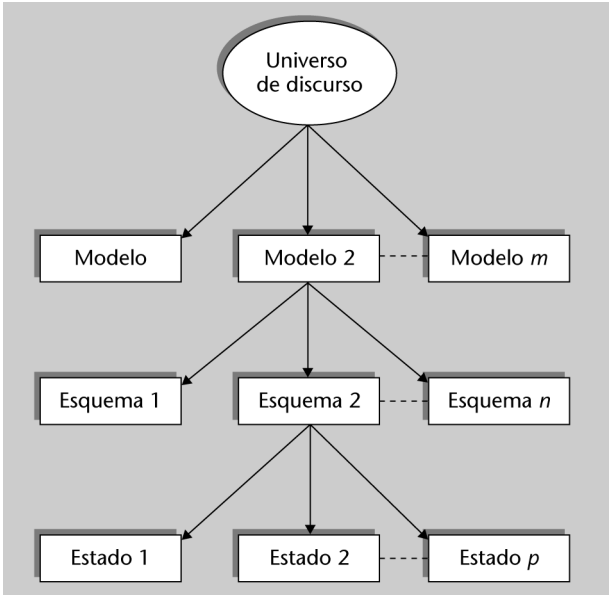
El SGBD debe asegurar que todos los estados de la base de datos sean *estados válidos*, es decir, que satisfagan la estructura y las restricciones especificadas en el esquema. El SGBD almacena las descripciones de estos elementos del esquema en su diccionario de datos para que, cuando sea necesario, sean consultados por el software del SGBD.

9.4. Relación entre modelo, esquema y estado

Como se ha visto, un **modelo** es el conjunto de reglas para estructurar los datos del universo de discurso; un **esquema** es la percepción de un universo de discurso determinado interpretado según un modelo concreto, y el **estado** es el valor que toma el esquema en un momento determinado.

La descripción del universo de discurso del mundo real mediante un modelo de datos da como resultado un esquema de la base de datos, que puede tener varios estados.

Figura 5. Modelo, esquema y estado



Los estados representados en la figura corresponden al esquema 2, que se ha obtenido aplicando el modelo de datos 2 al universo de discurso de un problema del mundo real.

Un **modelo de datos** es un conjunto de herramientas conceptuales que, fundamentándose en la abstracción, permite describir los objetos que intervienen en un sistema, así como también sus relaciones, los limitantes de integridad que los afectan y la terminología que se debe utilizar.

Un **esquema** es la representación abstracta de las propiedades estáticas y dinámicas de una parte del mundo real utilizando un modelo de datos y atendiendo sus aspectos fundamentales más significativos.

El **estado de una base de datos** es un conjunto de datos estructurados según un esquema determinado en un momento dado.

9.5. Evolución del esquema

En principio, el **esquema es invariable** en el tiempo, puesto que describe la estructura de la base de datos, mientras que el **estado cambia muy a menudo** porque depende de las ocurrencias de la base de datos en un momento dado.

A pesar de que se supone que el esquema no varía, muy esporádicamente puede ser necesaria su modificación si cambian los requerimientos del universo de discurso. Los SGBD más modernos incluyen operaciones para tener en cuenta la evolución del esquema, que se pueden aplicar mientras la base de datos está operativa, a pesar de que este proceso es más complicado que la ejecución de simples actualizaciones en la base de datos.

9.6. Intensión y extensión del esquema

El esquema de la base de datos se puede describir en términos de intensión y extensión. Así, se denomina *intensión* al mismo esquema y *extensión del esquema* al estado de la base de datos.

- La **intensión** es la representación genérica de las ocurrencias de la base de datos. Se corresponde con el **tipo**, que se obtiene por *clasificación* de un conjunto de objetos en un ente de nivel de abstracción superior. La intensión es la parte definitoria y estática (invariable en el tiempo) del esquema.

- La **extensión** (efecto de hacer extensivo o aplicable) es la concreción del **conjunto de ocurrencias** del tipo (se obtiene por *instanciación*) que, en un momento determinado, satisfacen el esquema y están almacenadas en la base de datos. La extensión varía en el transcurso del tiempo.

Intensión y extensión

Se relacionan con los tipos de abstracción, clasificación e instanciación, como se ve en el apartado siguiente, "Tipos de abstracción en el diseño de bases de datos".

10. Tipos de abstracción en el diseño de bases de datos

Los modelos de datos (especialmente los semánticos) ofrecen tipos de abstracción diferentes para facilitar la representación de los datos en el diseño de bases de datos. Los tipos de abstracción básicos son la **clasificación**, la **generalización**, la **agregación** y la **asociación**. Combinándolos, se obtienen mecanismos semánticos eficientes para estructurar los datos.

Las abstracciones permiten establecer vínculos entre los elementos de un modelo. En la *clasificación* se establece un vínculo entre una categoría (*tipo*) de objetos y cada objeto concreto (*ocurrencia*) de esta categoría, mientras que en la *generalización*, la *agregación* y la *asociación* el vínculo se establece entre categorías de objetos (y, por lo tanto, también entre sus ocurrencias correspondientes). Por otro lado, la clasificación, la generalización, la agregación y la asociación son síntesis conceptuales, mientras que los procesos inversos respectivos (la *instanciación*, la *especialización*, la *desagregación* y la *disociación*) son refinamientos conceptuales.

Ocurrencia y *tipo* se definen en el subapartado "Modelo de datos" del apartado "Modelo, esquema y estado de la base de datos"

Otro tipo de abstracción (que no se considera básica) es la *identificación*, que permite identificar de manera única a los tipos de objeto y sus ocurrencias mediante un identificador.

A continuación, se exponen estos tipos de abstracción.

10.1. Clasificación (e instanciación)

La **clasificación** es el proceso de abstraer las características comunes a un conjunto de ocurrencias para crear una categoría a la que pertenezcan estas ocurrencias.

Permite categorizar un conjunto de objetos (que comparten las mismas propiedades, asociaciones y restricciones) en una nueva categoría de nivel más alto, llamada **clase** o **tipo** de objeto. La clasificación corresponde a la función de **pertenencia** a un conjunto. Una **instancia** u **ocurrencia** está relacionada con el tipo mediante las relaciones “pertenece a”, “es miembro de” o “es una ocurrencia de” un tipo. Por ejemplo, *enero* es una ocurrencia del tipo MES.

Según la teoría de conjuntos, el tipo se puede considerar la **intensión** (parte definitoria) de todas las ocurrencias posibles de este tipo. El conjunto de estas ocurrencias en un momento dado constituye una **extensión** del tipo correspondiente.

La **instanciación** (o particularización) es el proceso inverso a la clasificación. Consiste en la generación y el examen de los diferentes objetos particulares (ocurrencias) a partir del tipo que los describe.

Figura 6. Clasificación e instanciación

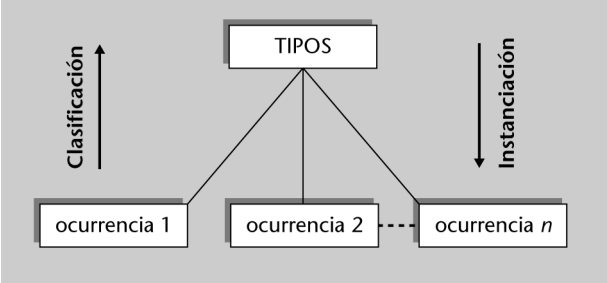
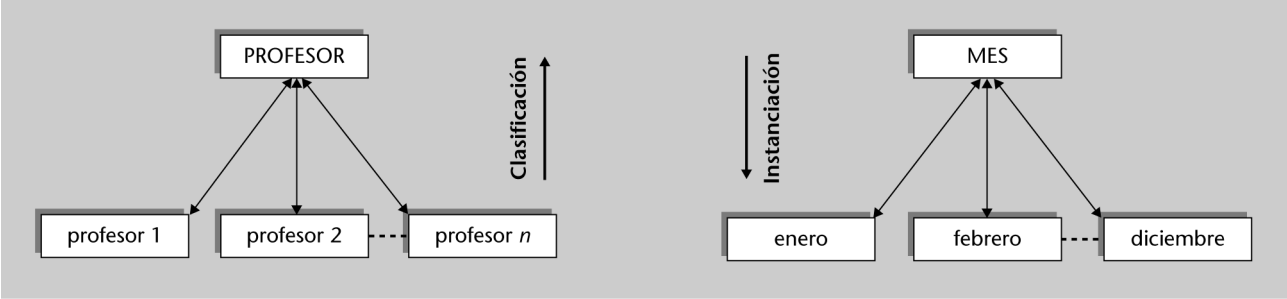


Figura 7. Ejemplos de clasificación/instanciación



Cada uno de los profesores de una universidad *se categorizan* en el tipo PROFESOR. ← Clasificación → Cada uno de los meses del año *se categorizan* en el tipo MES.

Cada uno de los profesores de una universidad *es una ocurrencia* del tipo PROFESOR. ← Instanciación → Cada uno de los meses del año *es una ocurrencia* del tipo MES.

La clasificación se puede considerar un tipo de **generalización**, puesto que permite agrupar cada una de las ocurrencias de objeto en un tipo de objeto. No obstante, debemos advertir que la generalización establece únicamente vínculos entre tipos de objeto.

10.2. Generalización (y especialización)

La **generalización** es el proceso de agrupar varios tipos de objeto para obtener un tipo más general (de nivel de abstracción más alto) que incluye los objetos de todos estos tipos.

Por agrupación de objetos más simples (los **subtipos**) se obtiene un nuevo elemento, el **supertipo**. Para relacionar un subtipo con un supertipo se utilizan las expresiones "es un subtipo de" o simplemente "es uno". Por ejemplo, un libro es un subtipo de DOCUMENTO.

La **especialización** es el proceso inverso por el que se categoriza un tipo de objeto en subtipos más especializados. En el mundo real, es habitual la descom-

posición (o especialización) de un tipo de objeto creando una jerarquía en la que se distingue un supertipo del que dependen varios subtipos.

Figura 8. Generalización y especialización

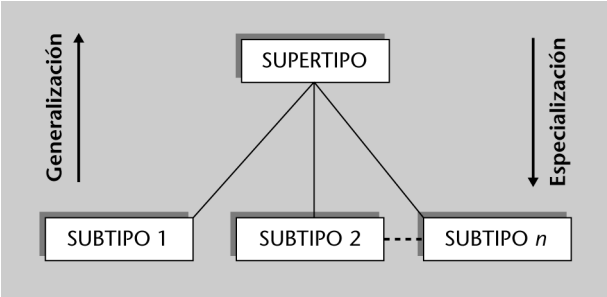
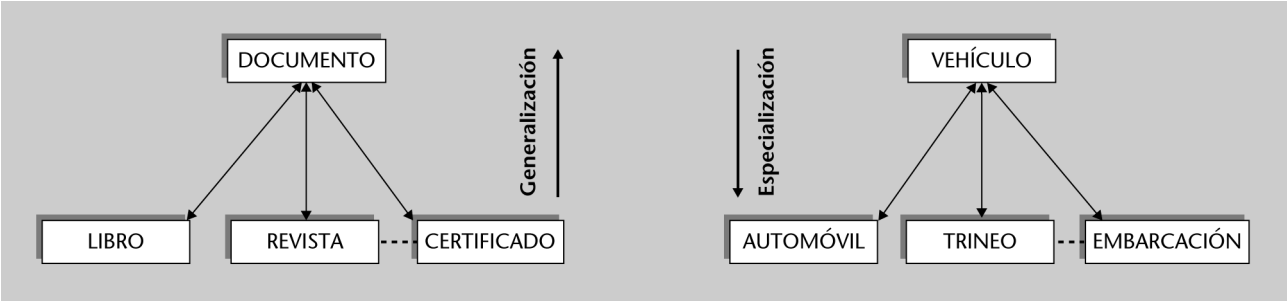


Figura 9. Ejemplos de generalización/especialización



Los tipos de objeto LIBRO, REVISTA y CERTIFICADO *se agrupan* en el nuevo tipo de objete DOCUMENTO, que es el supertipo de la jerarquía. ← Generalización → Los tipos de objeto AUTOMÓVIL, TRINEO y EMBARCACIÓN *se agrupan* en el nuevo tipo de objeto VEHÍCULO.

Los tipos de objeto LIBRO, REVISTA y CERTIFICADO *son subtipos* del supertipo DOCUMENTO. ← Especialización → Los tipos de objeto AUTOMÓVIL, TRINEO y EMBARCACIÓN *son subtipos* del supertipo VEHÍCULO.

La generalización no se utiliza en todos los modelos de datos, pero se está introduciendo en las extensiones de los diferentes modelos, como por ejemplo en el modelo entidad-interrelación E/R.

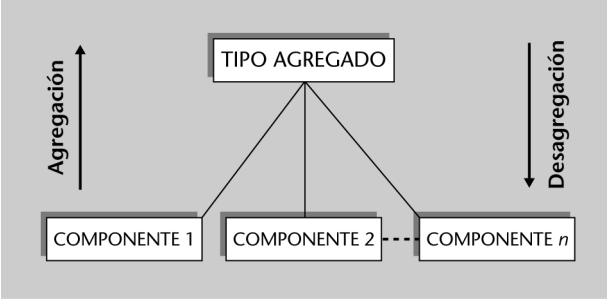
10.3. Agregación (y desagregación)

La **agregación** es el proceso de construir objetos compuestos a partir de sus objetos componentes.

Permite considerar un objeto según los elementos que lo constituyen. Básicamente, se pueden agregar tipos (para obtener un tipo compuesto, de nivel más alto), pero también propiedades (para obtener una propiedad compuesta o un tipo de objeto). Para relacionar los objetos componentes con el objeto agregado se utilizan las nociones "es parte de" o "es un componente de".

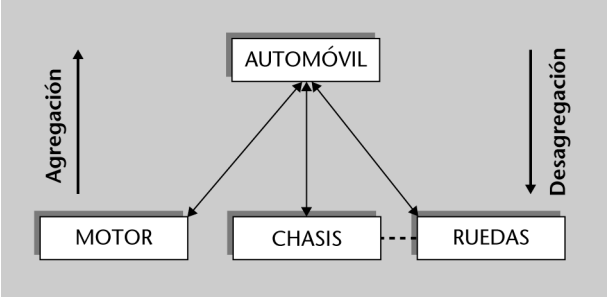
La **desagregación** (o descomposición) es el proceso inverso a la agregación. Consiste en pasar del elemento compuesto a sus componentes. Permite representar tanto los objetos componentes como las propiedades que caracterizan un tipo de objeto.

Figura 10. Agregación y desagregación



Los componentes también son tipos.

Figura 11. Ejemplo de agregación/desagregación



Los tipos de objeto MOTOR, CHASIS y RUEDAS *son componentes* del tipo de objeto AUTOMÓVIL.

Agregación: El tipo AUTOMÓVIL *es la agregación* de los tipos MOTOR, CHASIS y RUEDAS.

Desagregación: Los tipos de objeto MOTOR, CHASIS y RUEDAS *se obtienen por descomposición* del tipo AUTOMÓVIL.

10.4. Asociación (y disociación)

La **asociación** es el proceso de vincular dos objetos o más de varios tipos independientes (permite vincular tanto los tipos como sus ocurrencias).

Se obtiene un nuevo elemento, *la asociación*, con características distintivas propias que la distinguen de los participantes. Para relacionar un tipo con otro se utiliza la noción "está asociado a".

La **disociación** es el proceso inverso a la asociación. Consiste en la eliminación de la asociación.

Figura 12. Asociación y disociación

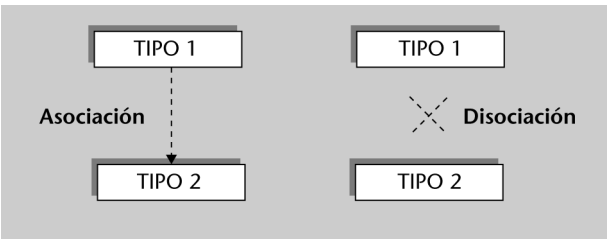
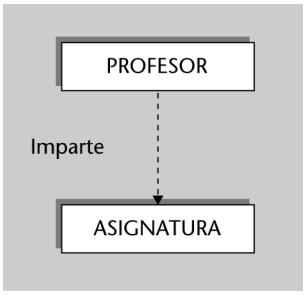
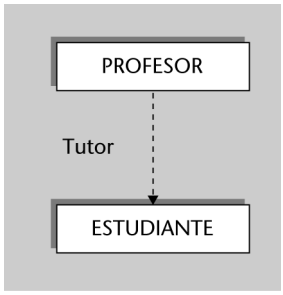


Figura 13. Ejemplos de asociación



El tipo PROFESOR *está asociado* al tipo ASIGNATURA mediante la asociación IMPARTE.



El tipo PROFESOR *está asociado* al tipo ESTUDIANTE mediante la asociación TUTOR.

La asociación es un tipo de agregación, a pesar de que en algunos modelos de datos no existe ningún concepto especial para representarla como abstracción.

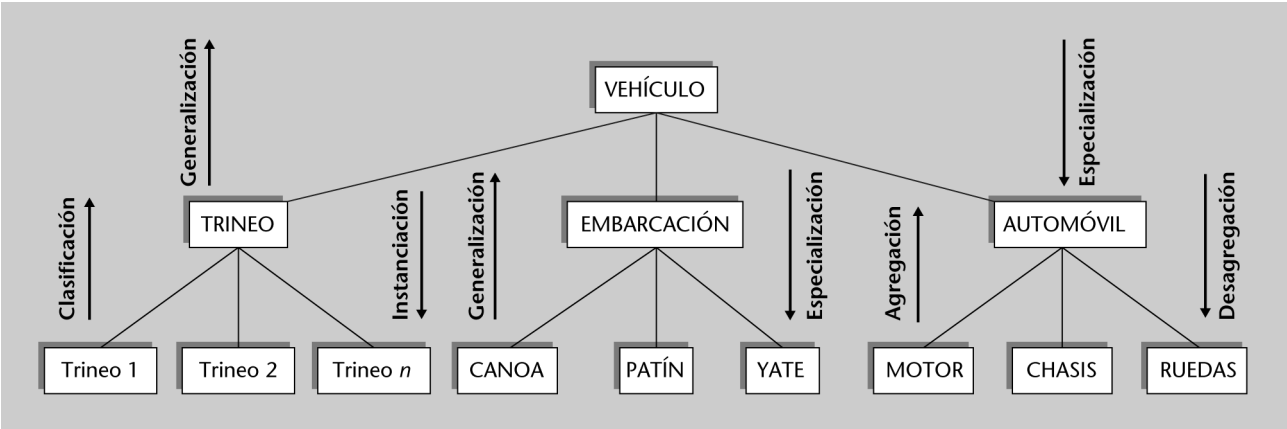
La asociación en los modelos de datos: relación e interrelación

En el *modelo relacional*, tanto los tipos como las asociaciones entre tipos se representan mediante **relaciones**, y no se hace distinción entre unos y otras. En cambio, la asociación es un concepto fundamental en el *modelo entidad-interrelación*, en el que el tipo nuevo se llama de **interrelación**.

Hay diferencias importantes entre agregación y asociación. La principal diferencia estructural es que si se elimina una ocurrencia de asociación, los objetos participantes pueden continuar existiendo, mientras que la eliminación del objeto agregado implica la eliminación de sus objetos componentes. Por ejemplo, si se elimina el objeto COCHE, se eliminarán sus componentes MOTOR, CHASIS y RUEDAS.

10.5. Utilización de la abstracción

Figura 14. Ejemplo de aplicación de varios tipos de abstracción



La especialización permite la abstracción del tipo de objeto vehículo a los subtipos trineo, embarcación y automóvil. El proceso inverso de abstracción de estos últimos a vehículo se realiza en la generalización.
El mismo proceso de especialización permite abstraer del tipo de objeto embarcación a sus tipologías específicas (canoa, patín, yate...).
Por otra parte, la instanciación es el tipo de abstracción que permite especificar varias ocurrencias del tipo de objeto trineo (que, por ejemplo, pueden pertenecer a propietarios o fabricantes diferentes). El proceso de abstracción inverso permite la clasificación de estas ocurrencias al tipo trineo.
La desagregación es el proceso de descomposición del tipo de objeto automóvil en sus componentes (motor, chasis, ruedas...). El proceso de abstracción inverso permite la agregación de estos componentes en el tipo de objeto automóvil.

11. Arquitectura de los SGBD

En este apartado se presenta la arquitectura de los SGBD según criterios diferentes: la distribución de su procesamiento, los niveles de abstracción de los datos y su composición interna y externa.

La arquitectura estándar de los SGBD actuales divide la base de datos en tres niveles de abstracción, según la perspectiva desde la que es vista. Se examinan las correspondencias entre estos niveles y las limitaciones que implica la ligadura para la independencia de los datos.

11.1. Las diferentes arquitecturas de los SGBD

La arquitectura de un SGBD se puede considerar desde enfoques o puntos de vista diferentes, los presentamos a continuación:

Arquitectura

Una arquitectura (como cualquier otra manera de esquematizar la realidad) es una herramienta sencilla y potente para abstraer y entender las características fundamentales de un sistema.

11.1.1. Arquitectura operacional

La arquitectura operacional especifica cómo se organiza la *distribución del procesamiento* del SGBD en el sistema informático.

1) Arquitectura centralizada (utilizada en los primeros sistemas)

Consta de un ordenador central único en el que se hace todo el procesamiento del SGBD y en el que residen los datos de la base de datos. Los usuarios acceden al sistema de manera remota (mediante **redes de comunicaciones**) a través de terminales sin poder de procesamiento, los cuales sólo ofrecen capacidad de visualización. El ordenador central envía a los terminales la información y los controles de pantalla.

2) Arquitectura cliente-servidor (predomina en los sistemas actuales)

El procesamiento del SGBD se reparte entre dos tipos de módulos: los clientes (la interfaz con el usuario, que se suele ejecutar sobre un ordenador personal) y el servidor o los servidores (en el que se ejecuta el corazón del SGBD, llamado *motor*).

La funcionalidad del sistema se suele distribuir de la manera siguiente:

a) El **módulo cliente** maneja la interacción con el usuario. Normalmente, proporciona los programas de aplicación y las interfaces de usuario (GUI basadas en formularios y menús) que acceden a la base de datos.

b) El **módulo servidor** se encarga de realizar las tareas propias de la base de datos. Normalmente, maneja el almacenamiento, el acceso y la búsqueda de datos.

Los dos módulos se pueden ejecutar en el mismo ordenador o, habitualmente, en ordenadores diferentes si éstos se interconectan a través de un **sistema de comunicaciones**.

Dado que la ejecución de la base de datos se puede efectuar sobre un único ordenador, se puede decir que el módulo servidor es, en sí, el SGBD y el cliente es el consumidor de recursos de la base de datos.

Lenguaje para la definición y consulta de bases de datos

Para la comunicación entre el servidor y la base de datos se utiliza un lenguaje de consulta y protocolos de red. SQL (siglas de *structured query language*, lenguaje de consulta estructurado, en inglés) es el lenguaje estándar para la definición y el acceso a bases de datos relacionales en sistemas cliente-servidor.

11.1.2. Arquitectura de referencia

La arquitectura de referencia determina los *niveles de abstracción de los datos* en el SGBD. Puede ser de dos o de tres niveles:

1) Arquitectura de dos niveles (utilizada en los sistemas clásicos, hoy en desuso).

a) El **nivel lógico** oculta los detalles físicos de almacenamiento y acceso a los datos. Incluye entidades, atributos, interrelaciones y reglas de integridad.

b) El **nivel físico** incluye elementos de almacenamiento físico como índices, espacio físico en el que se agrupan los registros, magnitud de las páginas o bloques, etc.

2) Arquitectura de tres niveles (predomina en los sistemas actuales, aunque no en todos).

a) El **nivel externo** gestiona la información desde el punto de vista individual de cada grupo de usuarios.

El enfoque de tres niveles

Es objeto de los contenidos del subapartado "Arquitectura de niveles de los SGBD" dentro de este mismo apartado.

b) El **nivel conceptual** describe la estructura de la base de datos para todos los usuarios con independencia del ordenador en el que se implemente.

c) El **nivel interno** describe la información en función del sistema en el que se implementará la base de datos.

11.1.3. Arquitectura externa

La arquitectura externa (o aparente) se refiere a las aplicaciones que se ejecutan en el software del SGBD y que el usuario ve como un conjunto de partes más o menos muy estructurado.

A pesar de que esta arquitectura suele estar determinada por criterios comerciales, no hay muchas diferencias externas entre los SGBD del mercado. Puede

cambiar el empaquetado, la presentación o el nombre de las diferentes utilidades por razones comerciales. Destacan las utilidades siguientes:

- Procesadores de consultas (o generadores de vistas).
- Generadores de formularios (o pantallas).
- Generadores de informes y gráficos.
- Generadores de menús.
- Procesadores de lenguaje natural.

11.1.4. Arquitectura interna

La arquitectura interna (o funcional) es el modo como el SGBD se estructura internamente en partes componentes con sus funciones.

Las funcionalidades que proporciona un SGBD se agrupan en bloques o módulos de software llamados **gestores**. La arquitectura funcional es una simplificación y, por lo tanto, no coincide necesariamente con la arquitectura real de los componentes que constituyen un SGBD específico.

Los componentes internos del SGBD

Son objeto de los contenidos del subapartado "El núcleo del SGBD" dentro del apartado "Funciones y componentes de los SGBD".

Cada SGBD estructura internamente el conjunto de su funcionalidad de manera diferente. Incluso, un mismo SGBD puede ir cambiando de estructura interna en versiones sucesivas para mejorar el rendimiento y facilitar la adición de funcionalidades nuevas.

Funcionalidad de los módulos del SGBD

El motor del SGBD tiene módulos para ejecutar funciones específicas como la interpretación de SQL, la optimización de consultas, la gestión de vistas, la gestión de memorias intermedias, la gestión del espacio de la memoria externa, la gestión de procesos y de transacciones, el control del acceso concurrente, la creación de copias de seguridad, el control de la privacidad, etc.

Existe un alto grado de **encabalgamiento** (ejecución simultánea concurrente de fases de instrucciones consecutivas) entre algunas de las funciones de un SGBD (las más físicas o básicas) y las del **sistema operativo** (SO). Para aspectos determinados como, por ejemplo, la gestión de memorias intermedias, la gestión del espacio del disco, los controles de seguridad o la gestión de procesos, los SGBD pueden aprovechar (o complementar) la funcionalidad del SO o resolverlos directamente. Algunas razones para no aprovechar el SO son que el SGBD lo puede hacer de manera más eficiente o para aumentar la confidencialidad.

11.2. Arquitectura de niveles de los SGBD

11.2.1. Niveles de abstracción y esquemas

Para que una base de datos pueda satisfacer los requisitos que se le exigen, es necesario que sus usuarios tengan una visión tan abstracta como sea posible de los datos almacenados.

Hay tres características inherentes al enfoque de bases de datos que es importante destacar. Se trata de la separación entre los programas y los datos (llamada **independencia de datos**), el soporte de múltiples **vistas de usuario** y la utilización de un **catálogo** para almacenar la descripción de la base de datos.

Las características de las bases de datos

Las características se enumeran en el apartado "Objetivos y características de las bases de datos" de este módulo.

La **independencia de datos** asegura que los programas de aplicación son independientes de los cambios introducidos en los datos que no se utilizan o en los detalles de implementación de los datos a los que se accede. Esta propiedad es una concreción del principio de **abstracción de datos** de los lenguajes de programación, entendido como la independencia entre la especificación de las estructuras de datos y su implementación.

Según quién acceda a la base de datos, ésta debe presentar una **vista** de los datos que el usuario sea capaz de reconocer, interpretar y manejar. Por ejemplo, es conveniente ocultar la complejidad de la base de datos al usuario final, puesto que no tiene la necesidad de conocer cómo se organizan físicamente los datos.

Para conseguir este objetivo, los SGBD permiten la definición de la base de datos en **niveles de abstracción** diferentes. La definición en cada uno de estos niveles se denomina *esquema*.

Los conceptos abstracción y esquema

Se tratan en el apartado "Modelo, esquema y estado de la base de datos" de este mismo módulo.

Los SGBD necesitan una descripción o definición de la base de datos, que recibe el nombre de *esquema*.

El esquema de la base de datos es un elemento fundamental de la arquitectura de un SGBD. Permite independizar el SGBD de la base de datos y permite que el SGBD sepa cómo es la estructura de la base de datos con la que debe trabajar. De este modo, se puede cambiar el diseño de la base de datos (su esquema) sin tener que realizar ningún cambio en el SGBD.

11.2.2. La arquitectura de tres niveles ANSI/SPARC

La arquitectura de dos niveles de los SGBD se amplía a tres

La distinción clásica entre dos niveles de abstracción (lógico y físico) en los SGBD es ampliada a tres niveles en el año 1975 por la arquitectura recomendada por la ANSI/SPARC (American National Standard Institute - Standards Planning and Requirements Committee). La idea consiste en descomponer el nivel lógico en dos: el nivel externo y el nivel conceptual. El nivel físico corresponde a lo que se denominará *nivel interno*.

El objetivo principal de esta arquitectura estandarizada por la ANSI/SPARC es establecer una división de la base de datos en tres niveles de esquemas según la perspectiva desde la que ésta es vista. Es decir, tres **niveles de abstracción** que se corresponden con los tres principales grupos de **usuarios** de la base de datos (usuarios finales, programadores y administradores). Estos niveles se introducen a continuación:

Los niveles de abstracción de la arquitectura ANSI/SPARC

Se describen con detalle más adelante en este mismo apartado.

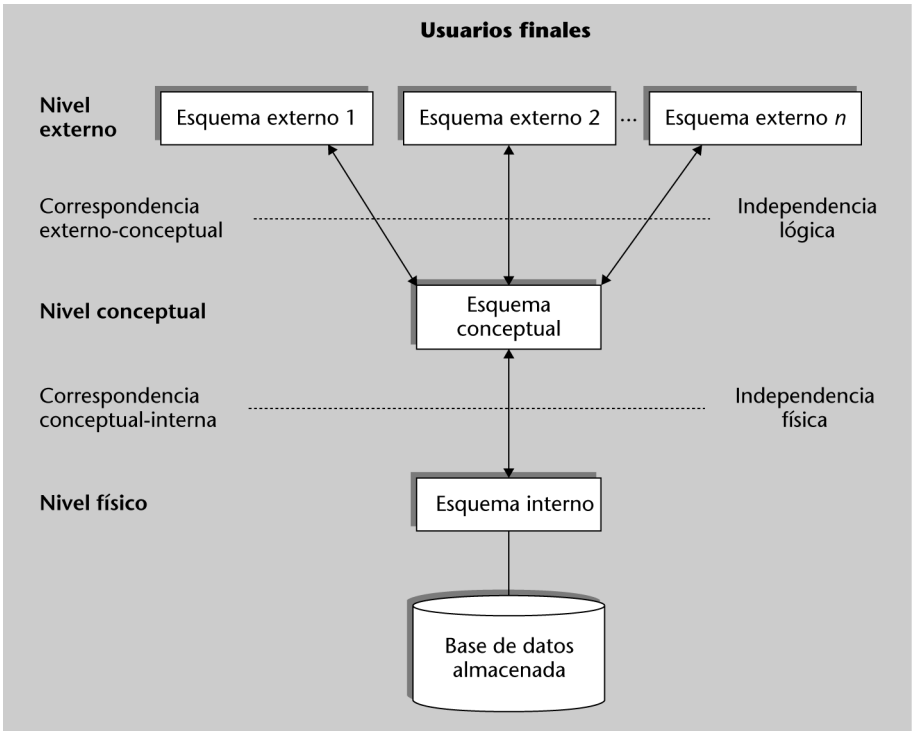
- El **nivel externo**, que es el nivel más abstracto, permite gestionar la información desde el punto de vista individual de cada usuario o grupo de usuarios. Se trata de visiones lógicas que los procesos usuarios (programas de aplicación y usuarios finales) tienen de las partes de la base de datos que utilizan. Estas visiones se denominan **esquemas externos**.

- El **nivel conceptual**, que se encuentra por debajo del anterior, representa a alto nivel toda la información de la base de datos independientemente del ordenador en el que se implemente. En este nivel intermedio hay una descripción lógica básica, única y global, denominada **esquema conceptual**, que describe la estructura de la base de datos completa para una comunidad de usuarios.

- El **nivel interno** es el nivel de abstracción más bajo, en el que se describe la información en función del sistema en el que se implementará la base de datos. Esta descripción física, que es única, se denomina **esquema interno**.

La figura siguiente de la arquitectura de tres niveles de un SGBD muestra los esquemas, los niveles, la correspondencia entre éstos y la independencia.

Figura 15. Arquitectura ANSI/SPARC de tres niveles de un SGBD



Cada nivel representa una visión diferente de los datos de la base de datos.

Niveles, esquemas y modelos de datos de los SGBD del mercado

La mayoría de los SGBD no separan claramente los tres niveles de descripción. Pero algunos, en cierto modo, soportan la arquitectura de tres esquemas:

- Algunos SGBD permiten incluir detalles del nivel físico en el esquema conceptual.

- Casi todos los SGBD que utilizan vistas de usuario especifican los esquemas externos en el mismo modelo de datos que describe la información de modo conceptual.
- Algunos SGBD permiten utilizar diferentes modelos de datos en los niveles conceptual y externo.

Por ejemplo, en el **modelo de datos relacional**, que se limita al nivel lógico y no hace referencia al nivel físico, los SGBD relacionales hacen referencia a un único esquema (denominado ***schema***), pero éste incluye descripciones de los tres niveles. En el ***schema*** se describen elementos como entidades, atributos y restricciones (que corresponden al esquema conceptual en la arquitectura ANSI/SPARC) y vistas (que equivalen al esquema externo). Además, los SGBD posibilitan incluir descripciones de estructuras y características físicas como índice, espacio de tabla (*tablespace)*, unidad de asignación de espacio de memoria (clúster), etc. (que corresponden al esquema interno)

Se definen **esquemas** en los tres niveles: el esquema interno (descripción de las características físicas), el esquema conceptual (visión centralizada) y los esquemas externos (visiones de los usuarios).

Una base de datos específica tiene un único esquema interno y un único esquema conceptual, pero puede tener varios esquemas externos, cada uno definido para un usuario o más de uno.

La arquitectura de tres esquemas es una herramienta adecuada para que el usuario visualice los niveles del **esquema de un sistema de base de datos**.

Para crear una base de datos de acuerdo con la arquitectura ANSI/SPARC de tres niveles, hay que definir los esquemas. Siguiendo la orden de ejecución de las fases del diseño de una base de datos, primeramente se definen los esquemas del nivel lógico (un esquema conceptual y un esquema externo, como mínimo) y, a continuación, se define un esquema interno. El proceso sería el siguiente:

1) Definición del esquema **conceptual**. Éste sirve de referencia para los otros dos tipos de esquemas y actúa de intermediario entre ellos. Los tipos de estructuras que se pueden utilizar dependen del modelo de datos subyacente al SGBD.

2) Definición de un esquema **externo**, como mínimo. Para cada grupo de usuarios se puede definir una vista parcial de la base de datos, que consiste en un conjunto de estructuras derivadas a partir de las estructuras del esquema conceptual.

3) Definición del esquema **interno**. A pesar de que, por cuestiones de eficiencia y rendimiento, puede interesar definir los detalles de organización física, si esto no se hace, el propio SGBD se encarga de decidir una implementación para cada una de las estructuras definidas en el esquema conceptual.

A continuación, se desarrollan en este mismo orden los tres niveles de descripción de la base de datos.

11.2.3. El nivel conceptual

La descripción de este nivel se realiza mediante un **esquema conceptual** (también conocido simplemente como *esquema de la base de datos*).

1) Se obtiene a partir de los requerimientos de los usuarios potenciales del sistema para implantar, y de las necesidades de la empresa, por lo que se crea de manera centralizada durante el llamado *diseño lógico* de la base de datos.

2) Oculta los detalles de las estructuras físicas de almacenamiento de los datos (forma y lugar).

3) Permite definir la organización de los datos y describe las interrelaciones que se establecen entre ellas, junto con otras características como la seguridad. Concretamente, se definen:

a) Todos los elementos de datos que intervienen en el sistema: los objetos (**entidades**), sus propiedades o características (**atributos**) y las dependencias que existen entre ellos (**interrelaciones**).

b) Las órdenes de control de integridad, las restricciones y reglas que rigen el funcionamiento de la empresa, las reglas de validación, los valores permitidos para los atributos, los tipos de datos, etc.

El esquema conceptual es la representación abstracta de un problema tal como éste se presenta en el mundo real, independientemente de cómo será tratada la información, de qué esquemas externos pueda tener y de cómo esta información pueda ser almacenada físicamente (esquema interno). El esquema conceptual de la base de datos no cambia si no cambia la naturaleza del problema.

En este nivel se puede usar un modelo de datos de alto nivel o uno de implementación.

Este nivel es el que utilizan los programadores y los diseñadores de la base de datos, que son responsables de considerar la información prevista por los analistas de sistemas y crear adecuadamente las estructuras lógicas de la base de datos.

Nota

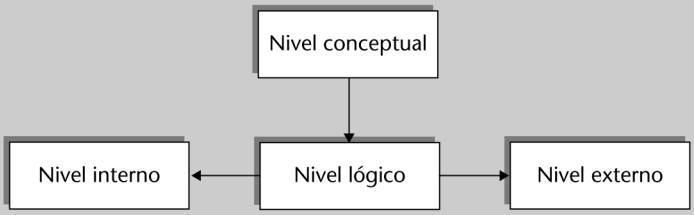
En este nivel, desde el punto de vista funcional del programador, se ejecutan los procesos siguientes:

- **Acceso al esquema de la base de datos** mediante el lenguaje de definición de datos (DDL, *data definition language*) y el lenguaje de control de datos (DCL, *data control language*).
- **Llamamientos al sistema** mediante el lenguaje de manipulación de datos (DML, *data management language*).
- **Utilización de herramientas de desarrollo visual** implementadas en el gestor de base de datos.

El nivel lógico (independencia del nivel conceptual)

Se puede considerar que hay un cuarto nivel de abstracción en la representación de la información en una base de datos: el nivel lógico. Para centrar el tema, debemos analizar el nivel de descripción conceptual del que deriva.

Figura 16. Relación del nivel lógico con los otros niveles



El nivel conceptual es lo más importante, puesto que en él se sostienen los otros niveles y permite garantizar la descripción correcta de los elementos de información que intervienen en el sistema o en aquella parte del sistema (el dominio del problema) que se quiere tratar (las entidades, sus propiedades y atributos, así como también las características de las relaciones existentes entre ellas).

Hay muchas maneras válidas de describir o representar, de manera abstracta, un sistema (un problema del mundo real):

- Se puede realizar **una descripción estricta** de un problema utilizando varias técnicas (diagramas de estructuras, diagramas de entidades y relaciones, árboles, tablas, redes, etc.). Pero **no todos los procedimientos son capaces de interpretar cualquier representación conceptual**.
- Una **representación abstracta** (conceptual) de un problema supone la aplicación de una serie de reglas que restringen el modo como éste es representado. Pero, en realidad, **el problema del mundo real es independiente del modo como sea representado** (por el ser humano).

De aquí se concluye que el esquema **conceptual** es independiente de los esquemas externo e interno, puesto que sólo depende del problema objeto de la representación. Si el problema no cambia, su esquema conceptual no cambia, aunque cambien los mecanismos para su representación.

Por el contrario, la **representación del nivel lógico** sí que depende de la manera, los mecanismos o los procedimientos por los que se manipula la información. El esquema **lógico** deriva del esquema **conceptual** según la aplicación de reglas y restricciones que tienen en consideración cómo la información representada puede ser tratada por los procedimientos que manipulan y definen la información según las representaciones.

Así, puede haber muchas representaciones lógicas o canónicas de una misma representación conceptual, del mismo modo que de una representación lógica se derivan muchas representaciones externas.

La inclusión de este nivel nuevo de representación del dominio del problema, que depende del software que manipula la información, permite garantizar la independencia de la información en una base de datos.

El nivel conceptual, en ocasiones, recibe el nombre de *nivel lógico global*.

11.2.4. El nivel externo

La visión externa oculta los detalles físicos

Del mismo modo que el nivel conceptual, el **nivel externo** oculta la complejidad interna de las bases de datos y ofrece al usuario una visión mucho más sencilla, puesto que éste no necesita conocer los detalles físicos sobre cómo se almacenan los datos o cómo se lleva a cabo su mantenimiento.

El nivel externo, también llamado *nivel de visión* o de usuario, define cómo la base de datos es percibida por el usuario final. Incluye un conjunto de esquemas externos, que algunos sistemas denominan *subesquemas*.

El usuario ve la base de datos mediante un **esquema externo** apropiado a sus necesidades. Cada **esquema externo** o **vista** individual describe la parte del esquema conceptual de la base de datos que interesa a un grupo de usuarios determinado y oculta a este grupo el resto de la base de datos (sin preocuparse por la correspondencia con el esquema conceptual).

Cada uno de los esquemas externos se puede considerar como una redefinición del esquema conceptual, con los elementos que convengan a las necesidades de cada usuario (o aplicación de usuario). Cada usuario puede tener su visión particular y parcial del esquema conceptual.

Por lo tanto, el usuario no percibe la base de datos en su totalidad. La seguridad del sistema permite que, según el nivel de acceso que se le haya concedido, tenga acceso a una parte de la base de datos (los datos que necesita, las relaciones que utiliza y las restricciones de uso que se le hayan definido).

Hay tantos esquemas externos como exigen las diferentes aplicaciones o los diferentes usuarios. El mismo esquema externo (un subesquema) puede ser utilizado por varias aplicaciones y compartido por varios usuarios (y un usuario tiene la posibilidad de acceder a diferentes subesquemas). De estas posibilidades, surge el concepto de **rol** o grupo de usuarios (conjunto de permisos y restricciones de acceso que pueden otorgarse a varios usuarios).

Usuarios del esquema externo

Se pueden distinguir dos tipos de usuarios en cuanto al esquema externo:

- Los que acceden a un esquema externo muy limitado a través de **sistemas de menús** y **transacciones programadas** o predefinidas. Habitualmente, son usuarios finales paramétricos que sólo hacen tareas de lectura o adición de datos (operadores de terminal o de estación de trabajo).
- Los que acceden a su **propio esquema externo**, desde el que realizan todas las consultas y acciones que desean. Son otros tipos de usuarios, como administradores, programadores o, incluso, usuarios finales más o menos experimentados (usuarios autónomos y avanzados).

Estos subesquemas o visiones particulares de los usuarios son proporcionados por los procedimientos o programas de aplicación que sólo manejan parte de la información de la base de datos.

Desde el punto de vista funcional del usuario, su nivel externo consiste en el conjunto de objetos a los que puede acceder. Para él, lo que tiene existencia real son las tablas y sus registros, las vistas, los formularios, los informes, etc. Dispone de un conjunto de herramientas visuales de generación de consultas, formularios e informes, y un lenguaje de consulta para la creación de programas que permiten acceder a los datos desde la base de datos.

Tareas que permite el esquema externo

Algunas posibilidades que facilita la definición de un esquema externo son las siguientes:

- Mostrar sólo los atributos y las entidades que interesan y omitir otros.
- Redefinir una entidad para que parezca que son dos.
- Definir combinaciones de entidades para que parezca que sean una sola (combinando atributos de entidades diferentes).

- Definir datos derivados a partir de atributos descritos en el esquema conceptual (campos calculados, etc.).
- Cambiar el nombre de entidades y atributos.
- Reordenar atributos.

En este nivel se puede usar un modelo de datos de alto nivel o uno de implementación.

El nivel externo es el conjunto de percepciones individuales que tienen los usuarios finales de la base de datos. Puede haber tantos esquemas externos diferentes como usuarios tiene la base de datos.

11.2.5. El nivel interno

Es el nivel más bajo y complejo de descripción de la base de datos, también llamado *nivel físico*.

Este nivel tiene asociado un **esquema interno** que describe los detalles de almacenamiento de los datos en los soportes físicos, así como también los caminos de acceso y las especificaciones de las estructuras físicas que representan el dominio del problema de una manera comprensible para el ordenador (dispositivos, volúmenes, archivos, tipos de datos, apuntadores, etc.). El esquema interno es la representación más cercana al almacenamiento físico de los datos.

Organización física de los datos

A la hora de especificar la organización física de los datos en los dispositivos de almacenamiento, debemos considerar los aspectos siguientes:

1) Estrategias de almacenamiento

Corresponden a la asignación de espacios para el conjunto de datos.

a) Gestión y reserva de espacio para almacenar los datos (volumen de la página, etc.).

b) Descripción de archivos de la base de datos que contienen la información (nombre, tipo de organización, unidad física en la que se almacenan, etc.).

c) Descripción de tablas de datos que contienen los registros.

d) Descripción de registros (longitud y tipo de datos de cada campo).

2) Estrategias de emplazamiento

Correspondientes a la designación de los lugares que deben ocupar los elementos de datos para optimizar los tiempos de respuesta y los espacios de la memoria secundaria.

a) Métodos o caminos de acceso a los registros (especificaciones de claves, índices, apuntadores, etc.).

b) Desfragmentación de datos (desplazamiento físico y concatenación de datos con objeto de reducir la cantidad de espacio utilizado y concentrar todo el espacio que no se utiliza en una única área indivisa). Tiene relación con la compresión o el empaquetado de datos.

c) Cifrado de datos (encriptación o codificación de los datos, mediante una clave adecuada, para que resulten crípticas o secretas).

d) Descripción del diccionario de datos (y el directorio de datos), si el sistema no realiza esta tarea automáticamente.

Los usuarios que intervienen en este nivel son los administradores de la base de datos.

El esquema interno responde a las cuestiones de **rendimiento** (espacio y tiempo de respuesta) planteadas al elaborar el diseño físico de la base de datos y el ajuste de ésta a las necesidades cambiantes.

El esquema interno utiliza un modelo de datos físico.

En el nivel físico, se describen los datos desde el punto de vista del ordenador que los almacena.

11.2.6. Usuarios y niveles

En cada nivel se describen los objetos de interés que pueden ser entendidos por los usuarios de aquel nivel. Esto se puede concretar de la manera siguiente:

- **El usuario final** sólo entiende de registros o formas de comunicación similares a los documentos externos manipulados en la organización (**nivel externo**).
- **El programador**, además del diseñador y el analista de sistemas, entiende de tipos de entidades o clases de objetos, relaciones existentes entre ellos y procedimientos ejecutados para la solución del problema que la organización desea tratar (**nivel conceptual**).
- **El administrador** de la base de datos se encarga de determinar la organización física que garantice el funcionamiento óptimo del sistema (**nivel físico**).

11.2.7. Correspondencia entre niveles

Debemos señalar que los tres esquemas no son más que *descripciones* de los datos. De hecho, los únicos datos que existen *realmente* se encuentran en el nivel interno. Pero, en un SGBD basado en la arquitectura de tres esquemas, cada grupo de usuarios hace referencia exclusivamente a su propio esquema externo.

Lógicamente, hay comunicación entre los tres niveles de esquemas, que el SGBD se encarga de llevar a cabo mediante dos procedimientos de transformación de los datos.

El SGBD debe transformar una solicitud expresada en términos de un esquema externo en una solicitud expresada en términos de un esquema conceptual y después en una solicitud en el esquema interno que, finalmente, se procesará sobre la base de datos almacenada.

El proceso de transformar solicitudes y resultados de un nivel a otro se denomina **correspondencia o transformación** (*mapping*). Este proceso se puede definir, de manera elemental, como la asignación o conversión de los registros de datos reales desde un soporte físico en un formato útil para ser presentados al usuario (y el proceso inverso).

La integración entre los tres niveles se establece de la manera siguiente:

- **Correspondencia externa-conceptual**: transforma los datos desde su representación externa a la conceptual.

  Cuando un usuario, con un nivel externo determinado, solicita mediante una consulta el acceso a unos datos determinados, el SGBD interpreta la solicitud, verifica el esquema externo definido para este usuario y transforma la solicitud desde el nivel externo al nivel conceptual.

- **Correspondencia conceptual-interna**: transforma los datos desde su representación conceptual a la interna.

  Una vez verificado el esquema conceptual, se pasa la solicitud desde el nivel conceptual al interno. En el nivel interno se selecciona la estructura de almacenamiento de datos sobre la que se ha realizado la petición y se llevan a cabo las operaciones necesarias para dar una respuesta a la solicitud del usuario.

  Cualquier modificación a nivel interno implicará un cambio en esta correspondencia sin que se modifique el nivel conceptual.

Este proceso se repite a la inversa para poder acceder a los registros objeto de la consulta. En el ejemplo expuesto más adelante se muestran en detalle las diferentes fases de un proceso entero de transformación. El conjunto de procedimientos que transforma un nivel en otro recibe el nombre de *reglas de correspondencia.*

Estas correspondencias pueden requerir bastante tiempo, por lo que algunos SGBD no soportan esquemas externos (tienen una arquitectura de dos niveles). De cualquier manera, siempre debemos realizar la transformación de solicitudes entre los niveles conceptual e interno.

En los SGBD de tres niveles es necesario ampliar el *diccionario de datos* o catálogo de manera que, además de contener las definiciones de los esquemas de la base de datos, incluya información sobre cómo se ha de establecer la correspondencia entre los tres niveles. El SGBD utiliza software adicional para establecer estas correspondencias haciendo referencia a la información de correspondencia que se encuentra en el diccionario, que también puede contener algún control de estadísticas de uso y de accesos.

Transformación de datos entre esquemas

Las fases de transformación de datos de un esquema a otro, en el caso de realizar una consulta en una base de datos relacional, son las siguientes:

1) Solicitud del usuario que quiere ver unos datos determinados de una tabla y creación de una consulta.

2) Verificación del esquema externo para este usuario.

3) Aceptación del esquema externo.

4) Transformación de la solicitud al nivel conceptual.

5) Verificación del esquema conceptual.

6) Aceptación del esquema conceptual. Se manipulan las tablas (entidades) que contienen los datos solicitados para la consulta.

7) Transformación de la solicitud al nivel interno. Se busca en los espacios de tabla (objetos lógicos formados por un archivo o más de uno) que contienen la información sobre el almacenamiento físico de los datos sobre los registros seleccionados en cada tabla.

8) Selección de la tabla objeto de la consulta.

9) Ejecución de la consulta.

10) Transformación del nivel interno al nivel conceptual.

11) Transformación del nivel conceptual al nivel externo. Debemos modificar el formato de la información extraída para que coincida con la vista externa del usuario.

12) Se muestran al usuario los registros correspondientes.

En la explicación no se hace referencia al diccionario de datos, si bien es el encargado de las transformaciones que se deben producir entre los diferentes esquemas.

El SGBD debe asegurar que los tres **niveles sean independientes entre sí**, es decir, que los cambios introducidos en cualquiera de ellos no afecten a los niveles superiores.

Con la arquitectura de niveles, los programas con los que el usuario accede a la base de datos a través de un esquema externo serán totalmente independientes de los cambios siguientes:

- Los cambios introducidos en el esquema **lógico** relativos a los datos no incluidos en su esquema **externo**.
- Los cambios introducidos en el esquema **físico** relativos a la implementación de las estructuras de datos del esquema **lógico**.

11.2.8. Independencia de los datos

La descripción de los datos en tres niveles de abstracción permite explicar la independencia de los datos, que es uno de los objetivos principales de las bases de datos.

El concepto independencia de datos se puede definir como la capacidad para modificar el esquema en un nivel del sistema de base de datos sin tener que modificar el esquema del nivel inmediatamente superior.

Se pueden definir dos tipos de independencia de datos: la independencia física y la independencia lógica de los datos.

1) Independencia física de los datos

Es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual (ni los esquemas externos).

Hay independencia física cuando los cambios en la organización física de la base de datos no afectan al esquema conceptual (ni a los programas de aplicación ni al usuario).

Puede ser necesario modificar el esquema interno para mejorar el rendimiento de las operaciones de recuperación y actualización de datos.

Modificación del esquema interno

La mayor parte de los cambios en el esquema interno suponen rehacer la base de datos real almacenada. Esto puede requerir acciones de **reorganización física** como las siguientes:

- Crear estructuras de acceso adicionales cambiando el método de acceso a unos registros determinados.
- Modificar el formato o la codificación de datos concretos.
- Reorganizar ficheros físicos determinados moviendo datos de un soporte a otro.

Si hay independencia física, lo único que varía al modificar el esquema interno son las *correspondencias* entre el esquema conceptual y el interno.

2) Independencia lógica de los datos

Es la capacidad de modificar el esquema conceptual sin tener que modificar los esquemas externos ni los programas de aplicación, siempre que no se eliminen del esquema conceptual objetos requeridos en el nivel externo.

Efectos de los cambios en los esquemas conceptual y externo

En un sentido más amplio, se considera que hay independencia lógica cuando los usuarios finales y los programas de aplicación no se ven afectados por los cambios en los niveles conceptual y externo (que corresponden al nivel lógico en la arquitectura clásica de dos niveles de bases de datos). Esto quiere decir lo siguiente:

1) Los **cambios en el esquema conceptual**...

a) pueden afectar a los **esquemas externos** que utilicen los elementos modificados (entidades o atributos);

b) no afectan a los esquemas externos que no hagan referencia o no utilicen los elementos modificados;

2) Los **cambios en un esquema externo**...

a) pueden afectar a los **usuarios** que utilizan los elementos modificados;

b) no afectan al resto de usuarios (ni al esquema conceptual ni, en consecuencia, al esquema interno).

Puede ser necesario modificar el esquema conceptual para ampliar o reducir la base de datos, añadiendo o eliminando entidades o atributos o, incluso, cambiando las características. Esto recibe el nombre de *reorganización lógica* de la base de datos.

Ejemplo

Si se elimina el atributo *apellido* en el esquema conceptual:

Se deberá modificar el esquema externo en el que se haya definido *nombre completo* como la concatenación de *nombre* y *apellido*.

No se ven afectados los esquemas externos (ni los usuarios) que no hagan referencia a este atributo.

Si el SGBD dispone de independencia lógica, sólo será necesario modificar la definición de los esquemas externos pertinentes y las **correspondencias**. Después de la reorganización lógica del esquema conceptual, los programas de aplicación que hagan referencia a elementos del esquema externo deberán funcionar como lo hacían antes. Además, las restricciones se podrán modificar sin que afecten a los esquemas externos ni a los programas de aplicación.

Los esquemas externos pueden cambiar debido a la incorporación al dominio del problema de nuevas necesidades funcionales u operativas para la organización. De hecho, los SGBD actuales dan bastante independencia lógica, pero no la suficiente, puesto que las exigencias de cambios constantes en los sistemas informáticos piden grados de flexibilidad muy elevados (está claro que los sistemas de ficheros tradicionales, en cambio, no presentan independencia lógica).

Como se ha dicho, la independencia de datos se consigue cuando, al modificar el esquema en algún nivel, el esquema del nivel inmediatamente superior permanece sin cambios; sólo se modifica la **correspondencia** entre los dos niveles. Por lo tanto, no es necesario modificar los programas de aplicación que hacen referencia al esquema del nivel superior.

Independencia de datos en los SGBD del mercado

La arquitectura de tres niveles puede facilitar la construcción de la verdadera independencia de datos, tanto física como lógica. No obstante, las **correspondencias** pueden requerir bastante tiempo, puesto que implican un gasto adicional durante la compilación y la ejecución de una consulta o de un programa, lo que reduce la eficiencia del SGBD. Por este motivo, son pocos los SGBD que han implementado completamente la arquitectura de tres niveles. Algunos, como los que gestionan bases de datos pequeñas, no soportan vistas externas.

Para que haya una independencia de los datos, los tres niveles de abstracción deben ser completamente independientes.

Este hecho, que no es del todo posible en la mayor parte de las bases de datos, se puede conseguir si:

- El esquema interno no es una traducción que depende del esquema conceptual. Un mismo esquema conceptual se puede representar físicamente de maneras diferentes. Los requisitos funcionales y de rendimiento determinarán esta representación.

- Aunque los esquemas externos dependen del esquema conceptual por el hecho de que los objetos del nivel externo que manipula el usuario (entidades, atributos, registros, etc.) deben estar formados por elementos de datos representados en el nivel conceptual, la estructura de estos objetos (número de elementos, disposición, etc.) debe ser independiente de cómo estos elementos se han representado en el nivel conceptual y de las relaciones que mantienen. Lo mismo se puede decir entre el nivel interno y el conceptual.

3) Independencia de datos y ligadura

Para garantizar la integridad de la base de datos, es necesario que, en algún momento, el proceso de manipulación de un nivel de esquema determinado tenga en cuenta cómo se representa la información en los otros niveles (o esquemas). En este momento, los diferentes esquemas se vinculan entre sí y, por lo tanto, se pierde la independencia entre ellos. Este proceso de vinculación o establecimiento de las correspondencias entre esquemas ejecutado por el SGBD se conoce con el nombre de *ligadura*.

La **ligadura** es la transformación de una operación descrita en términos de un esquema externo en otra operación descrita en términos del esquema interno.

Esta transformación es necesaria porque el SGBD necesita convertir un dato del esquema externo en un agregado o elemento de datos de un registro del

esquema interno. Dado que esta transformación pasa por el esquema conceptual, la ligadura se puede desglosar en dos:

- **Ligadura conceptual** (transformación entre el esquema externo y el conceptual).

- **Ligadura física** (transformación entre el esquema conceptual y el interno).

Cuando se produce la ligadura, se vinculan los diferentes esquemas. Esto provoca la pérdida de la independencia de datos, puesto que el esquema externo ha sido traducido en términos del esquema interno. Así, resulta que si la ligadura se ha producido al compilar el programa de aplicación, cuando se produzca un cambio en el esquema conceptual o interno habrá que volver a compilarlo. Por lo tanto, es conveniente retrasar tanto como se pueda el proceso de vinculación.

Compilar

Es la acción de traducir el código fuente de un programa (escrito en un lenguaje de programación de alto nivel) a lenguaje máquina (código objeto) ejecutable por el ordenador.

El proceso de vinculación o ligadura entre los diferentes esquemas se puede producir en cualquiera de las fases de un programa de aplicación (por orden de ejecución en el proceso):

– En la **compilación** o en un paso de precompilación.

– En el **montaje** para generar el módulo ejecutable del programa.

– En el **arranque** de un programa (esta fase corresponde al inicio de la **ejecución** del programa o, más concretamente, antes de que el programa empiece a solicitar accesos a los datos).

– En cada **acceso** a los datos.

Cuanto más tarde se produzca la ligadura, menos modificaciones habrá que hacer en los programas y, por lo tanto, más grande será la independencia de datos (a pesar de que la implementación del SGBD será más compleja).

Por otro lado, dado que la ligadura puede producir un gasto notable de tiempo, el funcionamiento de las aplicaciones que utilizan la base de datos será menos eficiente cuanto más frecuente sea la ligadura.

Ligaduras estática y dinámica

Los sistemas que realizan la ligadura en la fase **de acceso a los datos** mantienen la independencia hasta el último momento y sólo vinculan los datos a los que se puede acceder y que están implicados en cada uno de los accesos. En cambio, los sistemas que realizan la ligadura en la fase de **compilación** pierden la independencia de los datos en el momento en el que los programas de aplicación son traducidos a código objeto.

- **Ligadura estática**: si la ligadura se produce en una **fase temprana** (de compilación o de montaje), esto implica que los programas de aplicación deberán ser vueltos a compilar cada vez que se produzca una modificación de los esquemas conceptual y físico, a pesar de que, por otro lado, el rendimiento de éstos será alto porque el tiempo que implica la ligadura se consume una única vez (y no en el arranque del programa o en cada acceso a los datos).

- **Ligadura dinámica**: si la ligadura se produce en una **fase tardía** (de ejecución o en cada acceso a los datos) se podrán modificar los esquemas conceptual y físico sin que los programas de aplicación se deban traducir de nuevo a código máquina (compilar). A pesar de todo, el rendimiento de estos programas será menor porque necesitan un gasto de recursos adicional para llevar a cabo el proceso de ligadura (o bien en el arranque del programa o bien en cada acceso a los datos).

La solución de compromiso adoptada por muchos SGBD es que el proceso de ligadura se produzca en la fase de ejecución. Esto supone un gasto de recursos adicional en el arranque del programa de aplicación, pero evita una compilación reiterada y continua de las aplicaciones cada vez que se alteran las representaciones de los datos.

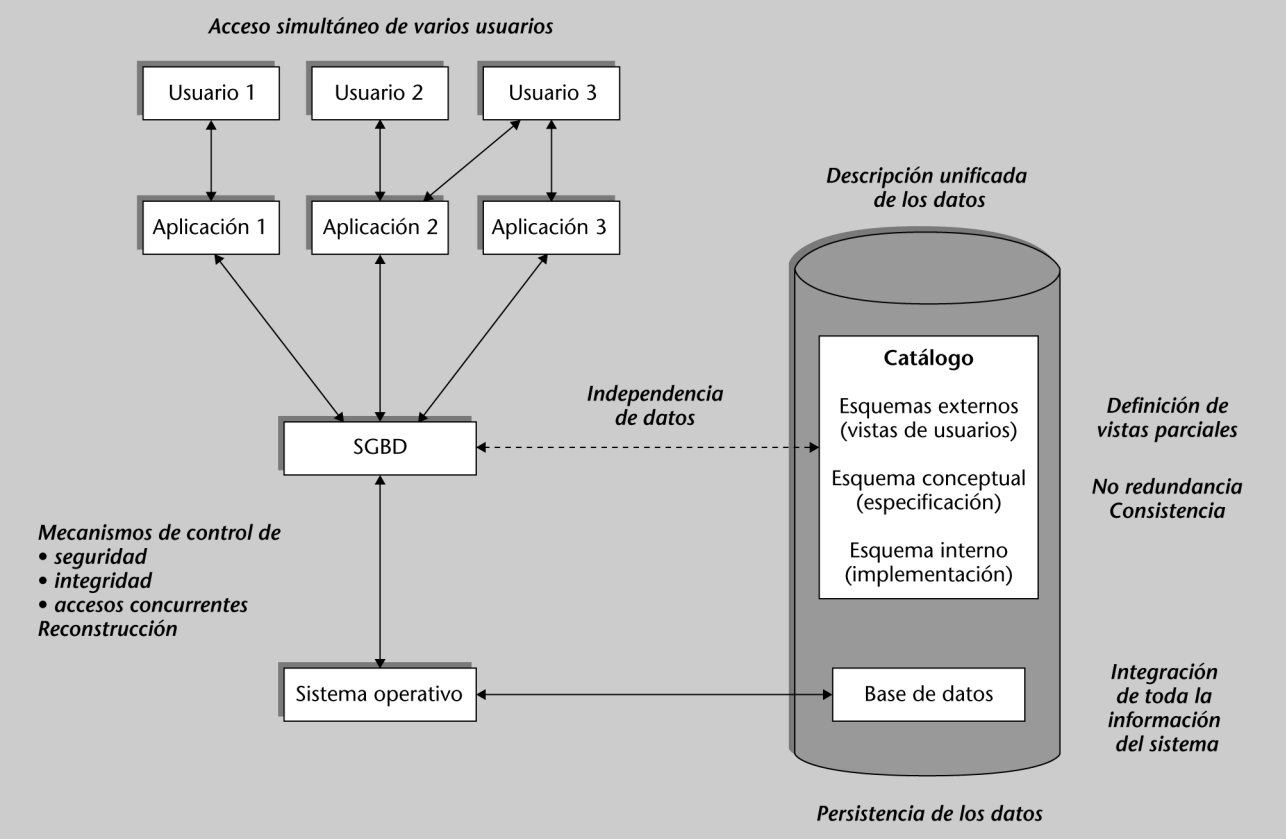
12. Estructura global de un sistema de base de datos

Sintetizando la materia expuesta previamente, se da una visión gráfica integrada de las características y los elementos de un sistema de base de datos entendido como el sistema que integra el SGBD y la base de datos en un sentido amplio: los **datos** (base de datos), sus **definiciones** (esquemas), el **software** de manipulación (SGBD y aplicaciones), los **usuarios**, etc.

Los elementos y las características

Los elementos y las características de un sistema de base de datos se exponen respectivamente en los apartados "Elementos de un sistema de base de datos" y "Objetivos y características de las bases de datos" de este módulo.

Figura 17. Sistema de base de datos



Elementos y características de un sistema de base de datos. En este mapa conceptual, los **elementos** del sistema se muestran encuadrados, y las *características*, en cursiva. Las flechas continuas representan órdenes y flujo de datos. La línea discontinua simboliza consultas del SGBD al catálogo en el que se encuentran las definiciones del sistema.

El SGBD, con la ayuda del sistema operativo, maneja las solicitudes del usuario para llevar a cabo las acciones de base de datos y permite cumplir los requisitos de control de seguridad, integridad, acceso concurrente a los datos, etc.

Se puede observar que el SGBD introduce un nivel de independencia nuevo entre el usuario y los datos y constituye una interfaz o capa sobre el sistema operativo subyacente (que no existía en los sistemas de gestión de datos basados en ficheros).

La separación entre programas y datos

Se encuentra en el contenido complementario referido a la independencia respecto a la representación física de los datos del subapartado "Objetivos y características de las bases de datos".

En el procesamiento de ficheros tradicionales, los programas de usuario accedían a los datos a través de los métodos de acceso a ficheros del sistema operativo, lo que provocaba que hubiera una estrecha dependencia entre los programas y los datos.

13. Almacenamiento de bases de datos

Este apartado trata sobre la organización de las bases de datos en el soporte de almacenamiento.

Las bases de datos suelen almacenar grandes volúmenes de datos que deben *perdurar* durante períodos de tiempo largos en los que se procesan los datos y se accede muy a menudo (a diferencia de las estructuras de datos *transitorios* que persisten un tiempo limitado durante la ejecución de un programa).

Por este motivo, los datos de una base de datos se deben almacenar en un medio de almacenamiento informático, de manera que el SGBD pueda recuperar, actualizar y procesar los datos cuando sea necesario.

Los medios de almacenamiento del ordenador pueden ser de dos tipos:

1) Almacenamiento primario: incluye la memoria principal y la memoria caché sobre las que el procesador o la unidad central de proceso (PCU) del ordenador puede operar directamente. Tienen un acceso rápido a los datos pero su capacidad de almacenamiento es limitada y son más caras.

2) Almacenamiento secundario: son dispositivos de almacenamiento externo con más capacidad y más baratos, pero con un acceso más lento a los datos. El procesador no puede procesar directamente los datos en almacenamiento secundario; se deben copiar en un medio de almacenamiento primario.

Las razones por las que la mayor parte de las bases de datos se almacenan de manera *persistente* en discos magnéticos (como el disco duro) u otros **medios de almacenamiento secundario** (y no en medios primarios) son las siguientes:

a) Por su elevado **volumen de datos**, las bases de datos grandes no caben enteras en la memoria principal.

b) Las **pérdidas permanentes** de datos se presentan menos a menudo que en el almacenamiento primario, que es volátil.

c) El **coste de almacenamiento** por unidad de datos es menor.

Las aplicaciones de bases de datos normalmente requieren localizar, cargar en memoria principal y procesar sólo una parte pequeña de la base de datos en un momento dado. Después, si ha habido alguna modificación, la escriben otra vez en el disco.

Las bases de datos se almacenan físicamente en el disco como **ficheros de registros**; así, los registros se pueden localizar de manera eficiente cuando se los necesita.

Cada **registro** consta de un conjunto de **valores** o elementos de datos relacionados, en los que cada valor está formado por al menos un byte y corresponde a un **campo** determinado del registro. En general, los registros suelen ser homogéneos (del mismo tipo) y describen entidades y sus atributos. Una colección de nombres de campos y sus tipos de datos correspondientes constituye una definición de **tipo de registro** o **formato de registro**. El **tipo de datos** asociado a cada campo especifica el tipo de valores que éste puede tomar.

Un fichero puede estar constituido por registros de los tipos siguientes:

- **Registros de longitud fija:** si todos los registros del fichero tienen exactamente la misma medida (en bytes).
- **Registros de longitud variable:** si algunos registros del fichero tienen medidas diferentes. Esto puede ser por el hecho de que haya campos de longitud variable, campos repetitivos (que se repiten un número variable de veces) o campos opcionales, o que se trate de un fichero mixto con registros de tipos diferentes.

  Hay varias maneras de especificar la longitud del registro:

  – El sistema reserva un **campo de control** al inicio de cada registro para indicar la longitud.

  – El sistema incluye un **carácter especial de separación** (o carácter delimitador de control) al final de cada registro. Se denominan **registros delimitados**.

  – El **programa de aplicación** se encarga de localizar el principio y el final de cada registro. Se denominan **registros indefinidos**.

Campos de longitud variable

Hay dos maneras de especificar la longitud de un campo:

- Al final del campo se incluye un **carácter separador especial** que no aparezca en ningún valor de campo (por ejemplo, los caracteres ?, % o $).
- Ante el valor del campo se guarda su **longitud en bytes**.

Hoy en día, se suelen utilizar **discos magnéticos** para almacenar las bases de datos de manera persistente. En un disco magnético, la memoria se divide en bloques de longitud fija, que se definen cuando se da formato al disco.

El **bloque** es la unidad de transferencia de datos entre el disco y la memoria principal. Es decir, el bloque es la cantidad de datos (en bytes) que el sistema escribe o lee de una vez en una única operación física de entrada o salida.

Concretamente, la transferencia la realiza el **administrador de entrada y salida** (E/S) del sistema operativo, que transfiere el bloque leído en el disco a la memoria intermedia de la memoria principal (áreas contiguas de memoria con capacidad para almacenar temporalmente varios bloques de datos, que en inglés se denominan *buffers*).

El **registro** es la unidad de transferencia entre la memoria intermedia y el programa de usuario.

Los registros se estructuran en un fichero siguiendo alguna de las organizaciones de ficheros disponibles en el sistema. Cada tipo de organización implica una distribución física particular de los registros en el disco y un tipo de acceso (secuencial o directo) a éstos. Estas organizaciones de ficheros son las que se utilizan para implementar las **estructuras de datos** de la base de datos, según como describe el **esquema físico**.

Las estructuras de datos

Las estructuras de datos utilizadas en los sistemas de información se tratan en el apartado "Estructuras de datos" de este mismo módulo.

Lectura y escritura de registros y bloques

En un sistema monousuario es frecuente que se ejecuten simultáneamente muchos procesos. Cada proceso puede trabajar con más de un fichero de datos y le puede convenir tener unos cuantos bloques en la memoria intermedia. Pero la medida de los bloques está condicionada por el espacio disponible para el conjunto de memorias intermedias en la memoria principal.

Un bloque puede tener varios registros y un registro puede ocupar varios bloques. Como la medida de un registro suele ser mucho menor que la de un bloque, los registros se agrupan en bloques. El bloque mínimo es un sector, pero se suele leer de golpe toda una serie de sectores.

En algunas ocasiones, al bloque se denomina **registro físico**, y **registro lógico** a lo que aquí se denomina *registro*. Por otro lado, en el entorno de bases de datos, habitualmente se usa el término *página* para referirse al bloque.

bloques de datos → ficheros de registros → estructuras de base de datos

Esta jerarquía de abstracción de datos (bloques de datos – ficheros de registros – estructuras de base de datos) existente en un sistema de base de datos permite explicar el **esquema de acceso del SGBD a los datos** para satisfacer un requisito de consulta que se trata en el apartado siguiente.

El acceso del SGBD a los datos

El acceso del SGBD a los datos para satisfacer una consulta se esquematiza en la figura del apartado siguiente, "Acceso del SGBD a los datos".

14. Acceso del SGBD a los datos

El SGBD es el conjunto de **módulos de software** para acceder a los datos que permite localizar un elemento de información específico (un registro) en la base de datos y presentarlo al usuario.

Para comprender su funcionamiento, a continuación se examinan los pasos implicados en el proceso de ejecución de una consulta de los datos de un registro hecha por un programa de usuario al SGBD, y que éste lleva a cabo con la ayuda del sistema operativo.

Lenguajes de consulta

La consulta también se puede hacer mediante un lenguaje de base de datos (por ejemplo, SQL) tal como se describe en el subapartado "Lenguajes de base de datos" dentro del apartado "Funciones y componentes del SGBD".

A pesar de que puede haber diferencias importantes en los detalles de funcionamiento (y terminología) entre los diferentes SGBD, todos ellos suelen seguir unas líneas generales bastante comunes que se pueden enunciar conceptualmente de la manera siguiente:

Los usuarios acceden a los datos a través de peticiones al SGBD que consisten en operaciones sobre las estructuras de datos de un esquema externo determinado.

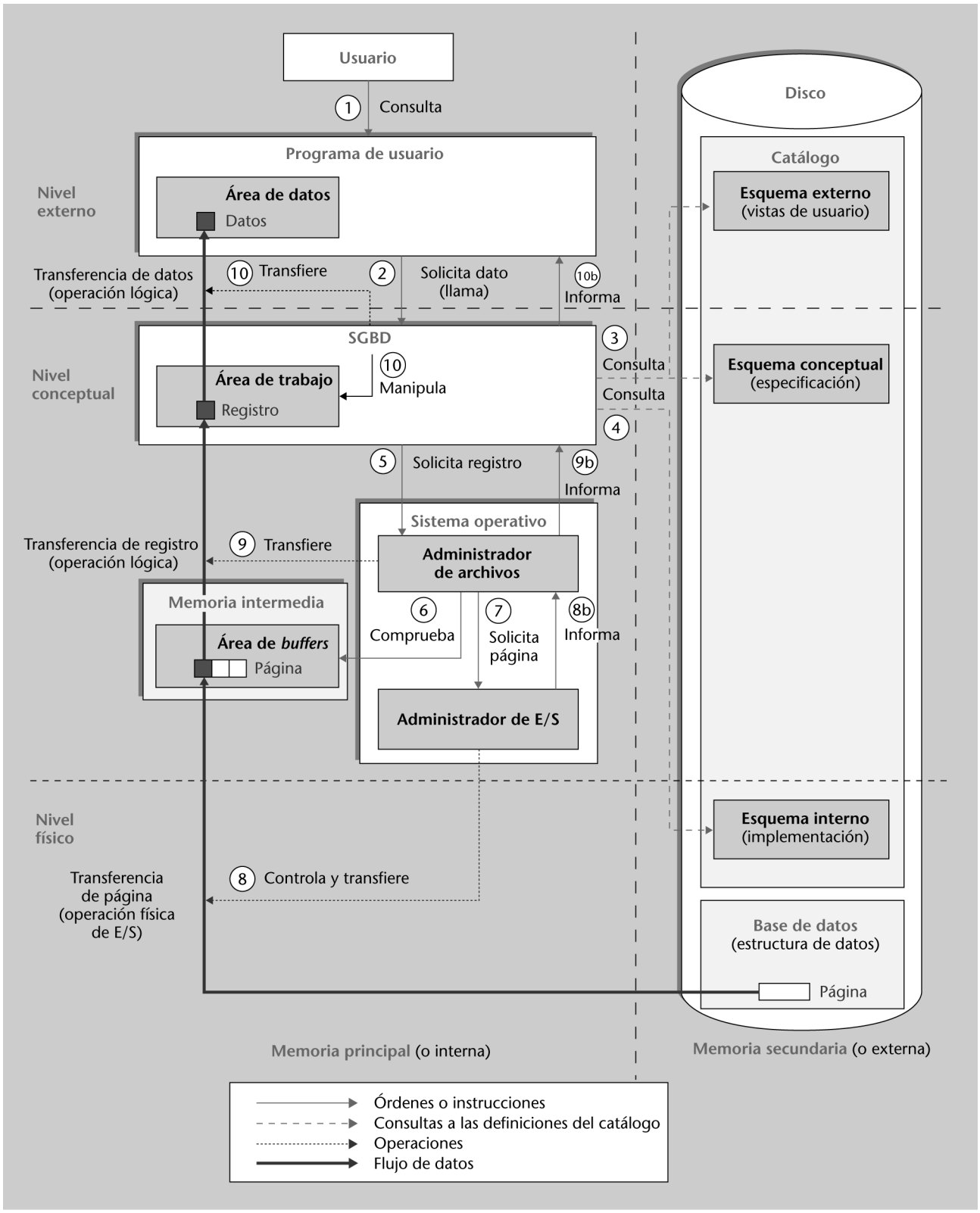
El SGBD, con el conocimiento que tiene de los esquemas externo, conceptual y físico, traduce estas peticiones en operaciones sobre los ficheros en los que se implementa la base de datos y encarga la ejecución al sistema operativo.

Éste lee páginas del soporte físico en el que se almacena la base de datos y las transfiere al área de memoria intermedia en la memoria principal.

Finalmente, el SGBD pasa registros desde la memoria intermedia al área de trabajo del programa de usuario. (Cada etapa puede requerir conversiones de tipos de datos u otros.)

La figura siguiente representa el esquema general del **flujo de datos y de control** que sigue el proceso entre el programa de usuario que hace la consulta, el SGBD, el sistema operativo subyacente y la base de datos.

Figura 18. Esquema de la gestión de acceso a los datos hecha por el SGBD



El flujo de datos se puede producir en sentido contrario al que se muestra en la figura. Éste es el caso de la ejecución de una operación de escritura de datos en el disco (en lugar de una lectura o consulta como se muestra en el esquema).

En la figura se muestran los **tres niveles** de abstracción de la arquitectura ANSI/SPARC en los que se desarrolla el proceso: **físico, conceptual y externo.** (Por ejemplo, el programa de usuario interviene a nivel lógico –más concretamente, a nivel externo– y las operaciones que ejecuta son lecturas de registros lógicos). Aun así, se aprecia que la base de datos y el catálogo se almacenan en

la memoria externa (disco) y los diferentes módulos de software se cargan en **la memoria principal**, en la que también residen los almacenes de memoria intermedia.

Los pasos indicados en la figura son los siguientes:

1) El usuario realiza **una consulta** utilizando un programa de usuario o aplicación.

2) Esto genera una **llamada** al SGBD en la que se envía la operación de consulta de datos.

3) El SGBD interpreta la solicitud recibida y la analiza (verifica si la sintaxis de la operación es correcta, si el usuario está autorizado a hacerla, etc.). Para hacerlo, se basa en el **esquema externo** con el que trabaja el programa de usuario y en el **esquema conceptual** (haciendo la correspondencia externa-conceptual asociada). Si la consulta no es válida, se pasa al paso 10b, que pone fin al proceso.

4) Si la consulta es válida, el SGBD, después de hacer la correspondencia conceptual-interna, examina el **esquema interno** y traduce la consulta del usuario en una **operación de lectura** sobre un fichero (y determina qué mecanismo físico se debe seguir).

5) El SGBD, una vez determinado qué **registro** se necesita, indica el fichero en el que se almacena y solicita al administrador de archivos que extraiga este registro.

El administrador de archivos

En algunos sistemas, el administrador de archivos es un componente del sistema operativo subyacente; en otros, se encuentra incluido en el SGBD.

Las operaciones que puede llevar a cabo el administrador de archivos son leer, modificar, añadir o eliminar registros de un archivo, y crear o destruir archivos. Estas operaciones primitivas permiten que el SGBD construya y manipule las estructuras de almacenamiento de datos.

En la práctica, la consulta del usuario (paso 1) puede exigir la recuperación de un **conjunto de registros**. En este caso, se repiten los pasos 5 a 9 para cada uno de los registros, que, una vez recuperados, son examinados por el SGBD en la memoria principal para encontrar el registro requerido. A continuación, se ejecuta el paso 10.

6) El administrador de archivos comprueba si la **página** en la que se encuentra el registro requerido es, casualmente en aquel momento, en el área de memoria intermedia de la memoria principal como resultado de una lectura anterior. En este caso, no hay que acceder al disco para leer la página de nuevo (y, por lo tanto, se pueden obviar los pasos 7 y 8 y pasar directamente al paso 9).

La memoria intermedia o área de memoria intermedia (*buffers*)

Esta área de memoria es muy importante para aumentar la velocidad de proceso del SGBD. Se guardan ahí temporalmente los datos a los que el SGBD ha accedido recientemente, puesto que existe una alta probabilidad de que deba volver a acceder a ellos. Esto permite ahorrar un gran número de operaciones físicas de entrada/salida que suponen más gasto de tiempo que los accesos a la memoria principal, en la que reside la memoria intermedia.

7) Si la **página** solicitada no está en el área de memoria intermedia, el administrador de archivos pide la lectura al administrador de E/S del disco del sistema operativo subyacente (este módulo, en algunos sistemas, recibe el nombre de componente de servicios básicos de E/S).

En definitiva, el administrador de archivos (y, finalmente, el SGBD) solicita operaciones de E/S de páginas y gestiona el espacio dedicado a la memoria intermedia.

8) El administrador de E/S determina la localización física en el disco de la página deseada, ejecuta la operación física de E/S necesaria (en este caso, su recuperación) y transfiere la página desde el disco al área de memoria intermedia.

9) El SGBD (con la ayuda del administrador de archivos) extrae el **registro** solicitado entre los diferentes registros que la página puede contener y lo deposita en el **área de trabajo**. El SGBD interpreta la codificación de registro según lo que indica el esquema interno.

10) El SGBD, comparando el esquema conceptual y el esquema externo, lleva a cabo las transformaciones eventuales que implica este último (inversas a los hechos en el paso 3) para determinar los **datos** requeridos, que transfiere al **área de datos** del programa de usuario que ha realizado la consulta original. Finalmente, el SGBD devuelve el control (10b) al programa y da por terminada la ejecución de la consulta.

Los pasos 8b y 9b, indicados en la figura, permiten que los módulos correspondientes informen de las acciones ejecutadas para devolver el control a los módulos que los han precedido en el proceso. Los pasos 3, 4 y 10 implican procesos internos de manipulación de registros por parte del SGBD que no quedan reflejados en la figura. Por otro lado, debemos remarcar que en la descripción se ha supuesto que la ligadura se produce en cada acceso a la base de datos.

Los lenguajes de bases de datos

Los lenguajes de bases de datos y la especificación de los procesos internos del SGBD se discuten con detalle en el próximo apartado, "Funciones y componentes del SGBD".

Compilación de las solicitudes de acceso a los datos

Obviamente, el esquema de acceso a los datos mostrado en la figura es una descripción simplificada y puede inducir a pensar que todo el proceso es interpretativo, puesto que sugiere que los procesos de analizar la solicitud, inspeccionar los distintos esquemas, etc. se hacen en el momento de la ejecución.

El concepto ligadura

Se describe al tratar la independencia de datos en el subapartado "Arquitectura de niveles" dentro de la apartado "Arquitectura de los SGBD".

La **interpretación** implica casi siempre un bajo rendimiento (debido al aumento del tiempo de ejecución). Pero en la práctica es posible **compilar** las solicitudes de acceso antes del momento de la ejecución (usando un lenguaje de base de datos hospedado).

En términos generales, se puede concluir lo siguiente:

- El **programa de usuario** percibe la base de datos como **un conjunto de datos**.
- El **SGBD** percibe la base de datos como **un conjunto de registros** almacenados.
- El **administrador de archivos** percibe la base de datos como **un conjunto de páginas** (bloques).
- El **administrador de E/S** percibe la base de datos tal como es físicamente **en realidad**.

15. Funciones y componentes del SGBD

Los SGBD se caracterizan por permitir la **descripción unificada de los datos** y la **definición de vistas parciales** de éstos para diferentes usuarios. Para satisfacer los objetivos de las bases de datos, se debe exigir al SGBD que asegure el mantenimiento de las **propiedades** de la base de datos: la **independencia**, la **integridad** y la **seguridad** de los datos.

A partir de los **objetivos** (y de las características) de las bases de datos, se pueden deducir las **funciones** de un SGBD. Basándose en el análisis de estas funciones, se puede saber cuáles deben ser sus **componentes** básicos.

Los objetivos y las características

Los objetivos y las características de las bases de datos se exponen en el apartado "Objetivos y características de las bases de datos" de este módulo.

15.1. Funciones del SGBD

El SGBD debe proporcionar los medios necesarios para llevar a cabo las tareas siguientes:

- **Definición** del esquema de la base de datos.
- Gestión de la **organización física** de los datos: ubicación, organización, agrupación y estructuras de recuperación de datos por fallos del sistema.
- Gestión de **acceso a los datos** desde los dispositivos de almacenamiento. Estas tareas de almacenamiento y recuperación de la información se pueden ejecutar desde el sistema operativo, desde las utilidades de la base de datos o mediante algún lenguaje de alto nivel (de cuarta generación, 4GL) como SQL.
- **Interrogación y respuesta a las peticiones del usuario**: el sistema interpreta, analiza y envía las peticiones del usuario para su ejecución; finalmente, presenta la información de respuesta recuperada.

Si se relacionan estas tareas con las operaciones que afectan a los archivos y sus registros, se puede deducir que, de acuerdo con su definición, el SGBD está formado por **componentes** que le permiten cumplir tres tipos de **funciones**: descripción, manipulación y control de la base de datos. No obstante, estas funciones no se corresponden de manera lineal con los diferentes niveles de abstracción.

15.1.1. Función de definición de datos

Permite especificar el esquema o diseño de una base de datos mediante la definición de los objetos que la constituyen, su estructura, las interrelaciones existentes entre ellos, las reglas de integridad semántica, así como también las características físicas.

Los tres niveles de abstracción intervienen en esta función de la manera siguiente:

- A nivel físico se definen las condiciones físicas de las estructuras de datos: espacio en disco, tamaño de las páginas (bloques), tamaño del espacio de tablas, etc. Esto se realiza por medio del lenguaje de definición de almacenamiento (LDE).

- A nivel conceptual se crean los objetos para los que se ha reservado el espacio y se han definido las características físicas. Esto se realiza mediante interfaces gráficas o con el lenguaje de definición de datos (LDD).

- A nivel externo, es posible crear y eliminar perfiles de usuario, definir papeles y asignar sus permisos.

Otras tareas incluidas en esta función son las relativas al catálogo (o diccionario de datos), que es un componente especial del SGBD en el que se almacenan los esquemas de definición de datos y que permite la transformación de las instrucciones que comunican unos niveles de abstracción con otros.

15.1.2. Función de manipulación de datos

Permite ejecutar acciones sobre los datos almacenados o añadir otras nuevas, trabajando sobre la totalidad de los registros de uno o más archivos o sobre cada registro de manera individual, de acuerdo con las normas de seguridad. Las acciones básicas son:

- La **recuperación** (o consulta), que implica la lectura de datos.

- La **actualización** (o mantenimiento), que supone la escritura de datos. Por ejemplo, la inserción de datos nuevos, la modificación de datos existentes o su eliminación.

Para ejecutar esta función, el administrador y el programador utilizan el lenguaje de manipulación de datos *(LMD)*, aunque, a nivel externo, se pueden utilizar menús, interfaces gráficas y transacciones previamente definidas que facilitan la utilización al usuario.

15.1.3. Función de control de datos

No es una función muy definida y a menudo se considera incluida en las dos funciones anteriores.

Si se acepta su identidad propia, se puede desdoblar en dos tipos de funciones:

- Funciones referentes a la **gestión de usuarios**: formadas por utilidades e **interfaces** que les permiten acceder, según sus permisos, a los diferentes elementos del sistema. Estas funciones de definición se incluyen aquí porque el administrador puede crear perfiles, modificar y eliminar permisos, etc.

- Funciones relativas a la **administración de la base de datos**: monitorización del **funcionamiento** (estadísticas, capacidades, espacios utilizados, fragmentación); control de la **seguridad** (gestión de accesos –privacidad–, gestión de conexión a redes, gestión de transacciones, gestión de copias de recuperación, etc.); mantenimiento de la **integridad** (respecto a los datos en sí mismos, sus valores y sus relaciones), etc.

Estas funciones se suelen llevar a cabo mediante la utilización de interfaces gráficas implementadas en los SGBD y el lenguaje de control de datos (LCD).

Objetivos de las bases de datos y funciones y componentes del SGBD

En la tabla siguiente se presenta un resumen de la correlación entre los objetivos de las bases de datos las funciones del SGBD y sus componentes asociados.

<table>
<tr><th>Objetivos de las BD</th><th>Funciones del SGBD</th><th>Componentes del SGBD</th></tr>
<tr><td>Describir los datos de manera unificada e independientemente de las aplicaciones</td><td rowspan="3">Definición de la base de datos en varios niveles de esquemas:
• Conceptual (definición de las estructuras de la base de datos)
• Interno (implementación de las estructuras del esquema conceptual)
• Externo (definición de estructuras derivadas)
Establecimiento de correspondencias entre esquemas</td><td rowspan="3">Lenguajes de definición de esquemas de la base de datos y traductores</td></tr>
<tr><td>Independizar las aplicaciones respecto a la representación física de los datos</td></tr>
<tr><td>Definir vistas parciales de los datos para diferentes usuarios</td></tr>
<tr><td rowspan="2">Gestionar los datos</td><td>Manipulación de los datos: consulta y actualización</td><td>Lenguajes de manipulación de los datos y traductores asociados</td></tr>
<tr><td>Administración de la base de datos</td><td>Herramientas para la administración:
• Reestructuración
• Simulación
• Obtención de estadísticas
• Impresión de datos</td></tr>
<tr><td>Mantener la integridad y la seguridad de los datos</td><td>Funciones de control de:
• Integridad semántica
• Seguridad
• Reconstrucción de la base de datos en caso de fallos
• Accesos concurrentes</td><td>Herramientas para el control:
• Control de la integridad
• Control de seguridad
• Reconstrucción
• Control de concurrencia</td></tr>
</table>

Las funciones de definición de esquemas y de establecimiento de correspondencias entre ellos se tratan con detalle en el apartado "Arquitectura de los SGBD".

15.2. Componentes del SGBD

El SGBD es un sistema muy complejo que se puede considerar formado por un conjunto de software dividido en dos grupos diferenciados: el motor de base de datos y las interfaces, utilidades y herramientas.

15.2.1. Motor de base de datos

Se encarga de las tareas de gestión de los datos: el proceso de consultas, la gestión de transacciones, el acceso a los datos y la forma de almacenamiento. Por lo tanto, acepta las consultas de los clientes, las procesa y devuelve los resultados. Está formado por los elementos siguientes:

- **Lenguajes de base de datos**. El usuario interpela el SGBD a través de un lenguaje de manejo de la base de datos (por ejemplo, SQL).

- **Diccionario de datos o catálogo**. Contiene información como los nombres de los ficheros y los elementos de datos, los detalles de almacenamiento de cada fichero, la información de correspondencia entre los esquemas y las restricciones, además de otros tipos de información que necesitan consultar los módulos del SGBD.

- **Núcleo del SGBD**. Es el conjunto complejo de módulos de software que se ejecutan como procesos transparentes para el usuario (que no se piden explícitamente). Se ocupa de las tareas básicas de la base de datos: control de almacenamiento, diálogo con el sistema operativo, manejo del diccionario de datos (o catálogo) y tareas necesarias para el trabajo con el lenguaje de base de datos.

15.2.2. Interfaces, utilidades y herramientas

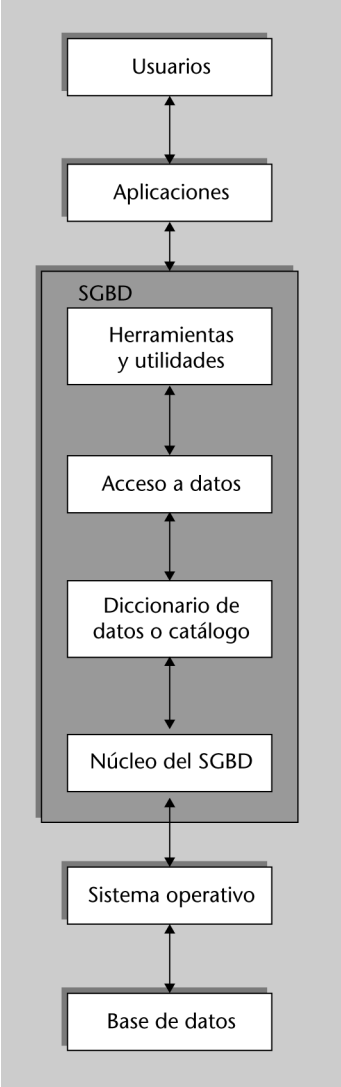
- **Interfaces**. El SGBD ofrece interfaces apropiadas a cada categoría de usuario (ocasional, paramétrico, administrador o programador de aplicaciones) como ayuda para que éste especifique sus solicitudes (ya sean, respectivamente, consultas interactivas, transacciones programadas, sentencias en LDD o instrucciones de un programa de aplicación). Las interfaces gráficas de usuario (GUI) de los actuales SGBD permiten omitir el uso del lenguaje de base de datos (la mayor parte del tiempo) por parte del usuario.

- **Utilidades y herramientas**. Un grupo importante de funcionalidades del SGBD es constituido por las herramientas y utilidades de valor añadido, que son aplicaciones de software integradas en el sistema para simplificar las tareas del administrador, el programador y el usuario final.

Si sólo se instala el núcleo, básicamente se puede prestar el mismo servicio de almacenamiento y recuperación de información que con el SGBD completo. La diferencia reside en el hecho de que los diferentes usuarios (administrador, programador y usuario final) deben utilizar herramientas más complejas y de nivel más bajo.

En la figura siguiente se muestran los elementos del SGBD integrado en un sistema de base de datos.

Figura 19. Elementos de un SGBD en un sistema de base de datos



El catálogo del SGBD y la base de datos se suelen almacenar en disco. Por otro lado, el sistema operativo suele controlar el acceso al disco planificando la entrada y salida (E/S) del disco.

El almacenamiento en disco

El almacenamiento en disco de la base de datos y el catálogo del SGBD se explican en los apartados "Almacenamiento de bases de datos" y "Acceso del SGBD a los datos" de este módulo.

15.3. Lenguajes de base de datos

Los lenguajes de base de datos permiten manejar la información contenida en la base de datos mediante conjuntos de órdenes o instrucciones con una sin-

taxis propia para cada uno de los lenguajes (igual que un lenguaje de programación habitual).

Se pueden analizar y clasificar según distintos criterios:

- Su **funcionalidad**.
- La modalidad de uso o modo de trabajo.
- Las **aplicaciones** que soporten.
- El **modelo de datos** en el que se fundamenta la base de datos.
- Los tipos de **usuarios** que acceden a la base de datos.

A continuación, se tratan estos aspectos con detalle.

15.3.1. Lenguajes y función

Los tipos de lenguaje se clasifican en tres grupos (o sublenguajes) en cuanto a su función: lenguajes de definición, lenguajes de manipulación y lenguajes de control. Estos lenguajes se pueden subdividir en partes más pequeñas, para varias funciones especializadas. Ahora bien, lo más frecuente es que el mismo lenguaje disponga de construcciones para los tres tipos de funciones. Generalmente, es un lenguaje simple basado en una gramática sencilla que dispone de un conjunto limitado de morfemas (unidades mínimas de significación).

1) Definición de esquemas

Después de completar el diseño de una base de datos y de elegir un SGBD para su implementación, el diseñador y el administrador (o el usuario que ejecuta tareas de administrador con los datos de su propiedad) deben especificar los esquemas conceptual e interno de la base de datos y la correspondencia existente entre ellos.

En un SGBD con arquitectura de tres niveles: Hay un **lenguaje de definición de datos** (LDD) que puede tener dos subcomponentes:

a) El LDE, **lenguaje de definición de almacenamiento** (esquema interno).
b) El LDV, **lenguaje de definición de vistas** (esquemas externos).

Seguidamente, se exponen los que tienen usos en los SGBD:

a) En los SGBD que no mantienen una separación estricta de niveles, se utiliza el **lenguaje de definición de datos** (**LDD**) para definir, de manera no ambigua, los esquemas conceptual e interno.

Esta definición debe ser compilada (por el **compilador de LDD** que tiene el SGBD) para dar lugar a una representación orientada al ordenador que es el que uti-

liza el SGBD en tiempo de procesamiento. La función del compilador es procesar las sentencias de este lenguaje para identificar las descripciones de los elementos de los esquemas. La representación de los datos obtenida en este proceso de compilación se almacena en el diccionario de datos o catálogo del SGBD.

Las órdenes principales que debemos destacar permiten crear o eliminar objetos de la base de datos, definir restricciones y reglas de integridad semántica.

Ejemplos de instrucciones de definición

En el lenguaje SQL, las órdenes para crear objetos de la bases de datos, eliminarlos, definir restricciones y definir reglas de integridad semántica corresponden, respectivamente, a CREATE, DROP, CHECK y CONSTRAIN.

b) En los SGBD que mantienen una separación clara entre los niveles conceptual e interno, el LDD sirve para especificar únicamente el esquema conceptual. Para especificar el esquema interno se utiliza el **lenguaje de definición de almacenamiento** (**LDE**) que a menudo se incorpora en el mismo LDD, de manera que se lo considera un subcomponente de éste. La correspondencia entre los dos esquemas se puede especificar en cualquiera de los dos lenguajes.

c) En una verdadera arquitectura de tres niveles sería necesario un tercer lenguaje, el **lenguaje de definición de vistas** (**LDV**) del usuario, para especificar el esquema externo y sus correspondencias con el esquema conceptual.

Ahora bien, la realidad es que en la mayor parte de los SGBD el LDD se utiliza tanto para describir el esquema conceptual (crear, modificar y eliminar tablas, índices, etc.) como para definir y modificar el esquema externo (vistas) y los permisos de acceso para los usuarios.

2) Control de datos

El **lenguaje de control de datos** (**LCD**) asegura un buen uso de la base de datos: permite el control del acceso a la información almacenada en el diccionario de datos (definición de privilegios y tipos de acceso), así como también el control de la seguridad de los datos. A menudo, también se considera un subcomponente del LDD.

Ejemplos de instrucciones de control

Órdenes SQL del lenguaje de control son definir o conceder permisos (GRANT) y denegar o revocar permisos (REVOKE).

3) Manipulación de datos

Una vez compilados los esquemas de la base de datos e introducidos los datos en ésta, en la fase de explotación de la base de datos el usuario requiere algún mecanismo para manipularla. El SGBD ofrece un **lenguaje de manipulación de datos** (**LMD**) para esta finalidad.

Las cuatro instrucciones básicas corresponden a las cuatro funciones básicas de manipulación de datos (recuperación, inserción, eliminación y modificación).

Ejemplos de instrucciones de manipulación

Las principales órdenes en SQL que intervienen en la función de manipulación de datos (recuperación, inserción, eliminación y modificación) son, respectivamente: SELECT, INSERT, DELETE y UPDATE.

El **lenguaje de consulta** (*query language, QL*) es un subconjunto del LMD, especialmente la parte relacionada con la recuperación y visualización de los datos. Ocasionalmente, se utiliza para referirse al conjunto completo del LMD.

Para recuperar los datos, exige que se definan criterios de selección y búsqueda. A la hora de modificar uno o más registros de la base de datos, se utiliza para especificar los criterios de actualización (por ejemplo, para buscar los productos de un proveedor concreto para incrementar el valor de su precio en un porcentaje determinado).

En los SGBD actuales, los tipos de lenguaje mencionados no se consideran lenguajes diferentes, sino diferentes funciones integradas en un mismo lenguaje más amplio, llamado **lenguaje de datos** o de base de datos (que dispone de elementos para definir esquemas conceptuales, definir vistas, manipular datos y definir su almacenamiento). La definición del almacenamiento, normalmente, se mantiene separada, dado que se utiliza para definir las estructuras físicas de almacenamiento para armonizar la ejecución del sistema de base de datos y normalmente es utilizada por el administrador de la base de datos.

Ejemplo de lenguaje de base de datos

El lenguaje de base de datos relacionales **SQL** es un ejemplo representativo que combina LDD, LDV y LMD, y también sentencias para la especificación de restricciones y de evolución del esquema. El LDE, que fue un componente en las primeras versiones de SQL, se ha retirado para mantener este lenguaje únicamente a nivel conceptual y externo.

Funciones y lenguajes asociados

Como resumen de lo que se ha expuesto, en el cuadro siguiente se especifica la función que realiza cada lenguaje.

Funciones	Componentes asociados
Definición de los datos	**Lenguajes de definición** de esquemas de la base de datos
Manipulación de los datos	**Lenguajes de manipulación** de los datos
Control de los datos	**Lenguajes de control** de la base de datos

Los principales lenguajes de base de datos se pueden definir de la manera siguiente:

El lenguaje de definición de datos (LDD) es un lenguaje especializado en la descripción (escritura de esquemas) de la base de datos (es decir, en la creación y el mantenimiento de la estructura). Hay lenguajes específicos para esquemas conceptuales, para esquemas internos y para esquemas externos.

El lenguaje de manipulación de datos (LMD) es un lenguaje especializado en la utilización de la base de datos (consultas y mantenimiento).

15.3.2. Modalidades de uso del lenguaje

Básicamente, hay dos maneras de trabajar con el lenguaje de base de datos, las definimos a continuación.

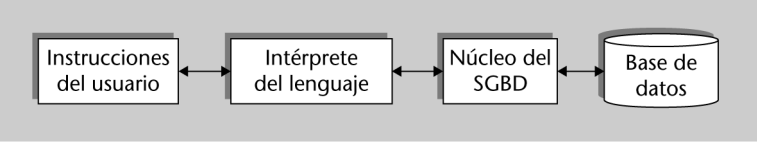
1) Modo interactivo

En el modo interactivo (o directo) se establece una conversación entre el usuario y el software del SGBD similar al modo como se mantiene un diálogo con

el sistema operativo. Cualquier instrucción del lenguaje de base de datos (LDD, LMD, etc.) se introduce sin restricciones directamente desde el teclado de un terminal. El SGBD dispone de un intérprete que analiza, verifica y ejecuta las instrucciones.

Esta modalidad de uso, no muy habitual, responde al esquema descrito en la figura siguiente.

Figura 20. Ejecución del lenguaje desde un terminal



2) Modo programado

En el modo programado (desde un programa de aplicación), el usuario ejecuta una aplicación sobre el sistema operativo.

Dado que, en general, los lenguajes de programación convencionales no tienen instrucciones propias para bases de datos, debemos usar el lenguaje de base de datos desde dentro de un programa. En este caso, se pueden presentar dos posibilidades:

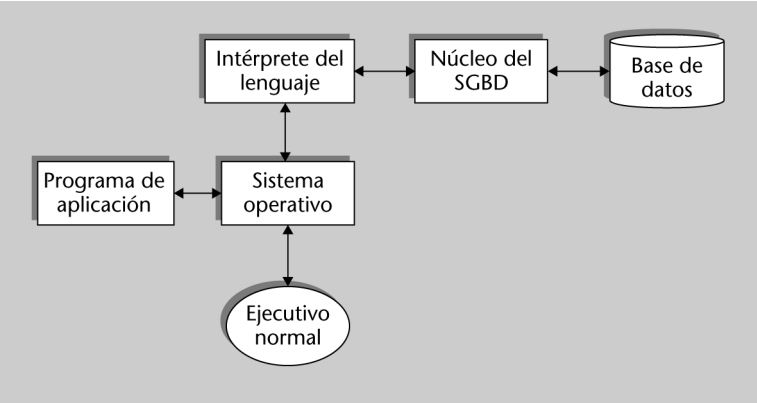
a) Un **programa escrito íntegramente en el lenguaje de base de datos** (SQL). Hay extensiones del SQL que lo dotan de las estructuras de programación imperativa usuales (bucles, selección). Un ejemplo de esto es el lenguaje PL/SQL del entorno Oracle y PostgreSQL. Se trata de un lenguaje procedimental que permite desarrollar programas imperativos que finalmente acceden a la base de datos a través del SQL. Estos programas son como guiones que el *intérprete* del SQL va siguiendo.

Dentro del programa escrito en el lenguaje habitual se incluyen **llamadas a funciones** (*call level interface, CLI*), que son librerías de funciones especializadas en solicitar al servidor de la base de datos la ejecución de instrucciones del lenguaje de base de datos (por ejemplo, librerías ODBC o JDBC). Estas funciones se encargan de enviar las instrucciones en el lenguaje de base de datos al SGBD en tiempo de ejecución.

b) Lenguaje hospedado o inmerso (*embedded*). Consiste en incorporar las instrucciones del lenguaje de base de datos (casi siempre SQL) directamente al programa escrito en un lenguaje de programación de propósito general (Cobol, PL/Y, C, Pascal, Java). Esto implica disponer de un *precompilador* especializado que acepte, dentro del lenguaje de programación, las instrucciones del lenguaje de base de datos. El lenguaje de programación que contiene las sentencias de base de datos se denomina **lenguaje anfitrión** (*host*).

Las instrucciones del lenguaje de programación se ejecutan por el procedimiento usual y las de SQL se transfieren a un módulo especial de ejecución del SGBD. En la figura siguiente se muestra un esquema de esta modalidad de uso:

Figura 21. Ejecución del lenguaje desde un programa



Una implementación del SQL hospedado establece las relaciones que deben mantener los objetos de la base de datos con los objetos del programa anfitrión, así como también ciertas restricciones de funcionamiento.

La modalidad de lenguaje hospedado admite dos variantes:

- **Lenguaje estático**: el programador escribe explícitamente las instrucciones que quiere que se ejecuten en el programa, que no admite cambios durante la ejecución. Tiene la ventaja de ahorrar tiempo de ejecución, puesto que la verificación y la traducción se llevan a cabo en tiempo de compilación. Éste es el método usado en la mayor parte de las aplicaciones.

- **Lenguaje dinámico**: el programador escribe parcialmente las instrucciones que se deben ejecutar y deja algunas para que sean completadas en tiempos de ejecución del programa. Es necesario usarlo si durante la ejecución varía algún parámetro o parte del programa. Permite escribir programas genéricos (como las herramientas de acceso visual a bases de datos) que dialogan con el usuario final para determinar qué deben hacer. Al escribir el programa, no se sabe exactamente qué instrucciones se deberán enviar al SGBD (que se concretarán en tiempo de ejecución), puesto que esto depende de lo que el usuario pida. Esta variante es menos eficiente que un programa en lenguaje estático y más difícil para el programador, puesto que utiliza técnicas dinámicas de manejo de variables.

  La modalidad CLI del lenguaje es dinámica, puesto que el programa pasa las instrucciones del lenguaje de base de datos como el valor de un parámetro y, por lo tanto, las puede construir en tiempos de ejecución.

15.3.3. Lenguajes y aplicaciones

Los tipos de lenguaje, en cuanto a las aplicaciones que soportan, se clasifican en dos grupos (principalmente, referidos a LMD):

1) Lenguajes de alto nivel o no procedimentales

Sólo requieren que en las sentencias se especifique qué datos se quieren manipular y qué se quiere obtener, pero no cómo se debe hacer (el propio lenguaje

es el encargado de determinar los procedimientos más efectivos para hacerlo). Por ello, estos lenguajes también se denominan **declarativos o implícitos**.

Se pueden utilizar de manera independiente para especificar operaciones complejas de base de datos de manera concisa. En muchos SGBD, es posible introducir instrucciones de LMD de alto nivel de dos maneras diferentes:

a) Interactivamente desde un terminal.

b) Hospedadas en un lenguaje de programación de propósito general. En este caso, debemos identificar las sentencias del lenguaje de datos dentro del programa para que el **precompilador** las pueda extraer y el SGBD las pueda procesar.

Permiten extraer información agrupada especificando y recuperando muchos registros con una sola instrucción, y por este motivo se denominan **lenguajes orientados a conjunto**.

También se denominan **lenguajes orientados a la consulta**, dado que permiten que el usuario haga uso de las herramientas del sistema para acceder a los datos mediante instrucciones aisladas (utilizando las herramientas de diseño visual como generadores de formularios, informes, etc.) o mediante grupos de instrucciones.

Dado que proporcionan un cierto nivel de abstracción del código máquina subyacente (cada sentencia se traduce en más de una instrucción de lenguaje máquina), son fáciles de entender y manejar por parte de usuarios no expertos.

Los LDD son declarativos

Por su propia naturaleza, los lenguajes de definición de datos (LDD) son declarativos o no procedimentales.

Cuando los requisitos de tiempos de respuesta del SGBD en los procesos de manipulación de los datos son importantes, es necesaria la modificación del código que generan los lenguajes declarativos para asegurar un rendimiento adecuado del sistema. Esto se consigue con los lenguajes procedimentales.

2) Lenguajes de bajo nivel o procedimentales (o explícitos)

Requieren que en las sentencias se especifique qué datos se quieren manipular, qué se quiere obtener y con qué procedimientos (se deben conocer más cuestiones del funcionamiento del SGBD para detallar paso a paso cómo se deben ejecutar las operaciones). Por ello se denominan **lenguajes orientados a procedimientos**. (Cada instrucción escrita se suele corresponder con una instrucción de máquina; es, por lo tanto, un lenguaje que depende del ordenador que lo utiliza, comprensible pues para el lenguaje máquina.) También se llaman explícitos o no declarativos.

Deben estar incorporados u hospedados en un lenguaje de programación de propósito general.

Se utilizan para crear programas, módulos o procedimientos (secuencia, con nombre, de instrucciones) que se ejecutan desde un **gestor del lenguaje** o se incrustan en **un lenguaje anfitrión**. Su utilidad consiste en permitir el desarrollo de aplicaciones verticales, interactivas o por lotes para su uso destinadas al usuario final con pocos conocimientos informáticos, mantener para este usuario un esquema externo tan limitado como sea posible y ocultarle la complejidad de la base de datos, que utiliza básicamente para almacenar y recuperar información sin ningún tratamiento más complejo.

Las aplicaciones verticales

Son programas de aplicación diseñados para satisfacer las necesidades de un sector empresarial muy específico (por ejemplo, restaurantes, comercios al por menor, consultas médicas, etc.). Suelen proporcionar funciones completas de administración, como por ejemplo planificación, facturación, control de inventarios y compras.

En contraposición, las aplicaciones horizontales apoyan a organizaciones y áreas de negocio muy variadas. Son programas de aplicación transversales como por ejemplo herramientas ofimáticas (procesadores de texto u hojas de cálculo), sistemas CRM, etc.

En general, este tipo de lenguaje recupera registros (u objetos) individuales de la base de datos y los procesa por separado. Por lo tanto, necesita utilizar elementos de lenguajes de programación, como los bucles (subrutinas), para recuperar y procesar cada registro individual de un conjunto de registros. Por esta razón, se conocen como **lenguajes orientados a registro**.

Los lenguajes procedimentales se suelen aprender y utilizar con más dificultad que los declarativos (por este motivo, sólo son utilizados por usuarios informáticos).

SGBD prerelacionales

Los lenguajes utilizados en los SGBD prerrelacionales eran procedimentales

El SQL relacional es básicamente declarativo, pero tiene algunos detalles procedimentales

Resumint:

> Los **lenguajes declarativos** sólo requieren que se especifique qué datos se quieren manipular y qué se quiere obtener, pero no cómo. En cambio, los **lenguajes procedimentales** requieren que, además, se especifique con qué procedimientos se debe hacer.

Los lenguajes procedimentales permiten formular procesos más eficientes que los declarativos.

Por otro lado, los LMD de alto nivel utilizados de manera interactiva e independiente reciben el nombre de **lenguajes de consulta.** En general, tanto las instrucciones de recuperación como las de actualización de datos de un LMD de alto nivel se pueden utilizar de manera interactiva; por lo tanto, se consideran parte del lenguaje de consulta. (A pesar de que, de acuerdo con el significado de la palabra *consulta*, realmente sólo se debería utilizar para describir la recuperación de datos, no la actualización).

15.3.4. Lenguajes y modelos de datos

Según el modelo de datos los lenguajes de bases de datos se pueden clasificar de cinco maneras diferentes. Las presentamos a continuación:

1) Lenguajes relacionales

a) Basados en el álgebra relacional

Son lenguajes procedimentales en los que el usuario o el programador deben conocer la sintaxis del lenguaje que utilizan.

- **SQL**. Es el **lenguaje** más utilizado en bases de datos relacionales.

  Aparece en 1979 de la mano de Oracle, que se avanza por poco tiempo a IBM (1982). Es la evolución de **SEQUEL** (*structured english query language*), lenguaje propuesto en el año 1975 por Boyce y que procede de **SQUARE** (*specifying queries as relational expressions*). Tiene una sintaxis en inglés casi natural. Dispone de instrucciones de tres tipos diferentes:

  – **De manipulación**: por ejemplo, SELECT para hacer consultas e INSERT, UPDATE y DELETE para el mantenimiento de los datos.

  – **De definición**: por ejemplo, CREATE TABLE para definir las tablas, las columnas y las restricciones.

  – **De control del entorno**: por ejemplo, COMMIT y ROLLBACK para delimitar transacciones.

Algunas instrucciones de SQL se comentan en el subapartado "Lenguajes y funciones" de este mismo apartado.

b) Basados en el cálculo relacional

Son lenguajes no procedimentales en los que el usuario o el programador deben conocer el tipo de resultado que esperan conseguir, los datos originales a partir de los que deben obtener el resultado y los criterios de selección o filtrado impuestos a los datos de entrada. Destacan los siguientes:

- **QUEL** (*query language*, creado en 1974 por IBM). Es el DML del SGBD Ingres (que se ejecutaba en máquinas UNIX) y por ello muchas de sus características proceden de este lenguaje. Se basa en cálculo relacional de tuplas. Su sintaxis es en inglés y se asemeja a la de SQL.

- **QBE** (*query by example*, creado en 1977 también por IBM). Mediante una interfaz gráfica interactiva, similar a los filtros implementados en las hojas de cálculo, muestra tablas y permite generar expresiones sobre sus campos utilizando un ejemplo de resultado. Se basa en cálculo relacional de dominios. Access y dBase incorporan SQL y QBE.

2) Lenguajes de sistemas en red

A partir del modelo en red Codasyl aparecen lenguajes para los SGBD siguientes:

a) TOTAL. Su lenguaje se debe incrustar en un lenguaje anfitrión para su ejecución.

b) IDMS (*integrated database management system*). Dispone de DML, DDL y DCL, que cumplen parcialmente el estándar Codasyl.

3) Lenguajes de sistemas jerárquicos

El lenguaje principal para acceder a bases de datos jerárquicas es:

a) DL/1 (*data language one*) de IBM. Es el lenguaje de datos del SGBD jerárquico DMI (*information management system*) de IBM aparecido en los años sesenta.

Se trata de un lenguaje huésped y procedimental que se debe incrustar en otros lenguajes. A pesar de que ha sido superado por otras alternativas tecnológicas, ha sobrevivido, puesto que soporta aplicaciones en Java, JDBC, XML y servicios web.

4) Lenguajes orientados en la programación

Un lenguaje de base de datos definido expresamente para la programación es el siguiente:

a) NATURAL. Es el lenguaje de cuarta generación del SGBD ADABAS (de Software AG). Procesa la información de bases de datos que almacenan los datos en ficheros secuenciales o secuenciales indexados. Se puede ejecutar como módulos de software y puede efectuar llamamientos a otros lenguajes.

5) Lenguajes orientados al objeto

Los lenguajes de consulta de SGBD orientados al objeto más destacados son los siguientes:

a) OSQL. Es el lenguaje de consulta del SGBD orientado al objeto IRIS, que es una extensión dialectal OO del SQL.

b) OQL. Es el lenguaje de consulta del SGBD orientado al objeto O2 (actualmente, Ardent).

15.3.5. Lenguajes y usuarios

Para comunicarse con el SGBD, cada usuario (o programa de aplicación) usa un lenguaje. Según el tipo de usuario al que van destinados, los lenguajes se pueden clasificar en tres grupos:

Comunicación entre el usuario y el SGBD

El usuario necesita comunicarse con el SGBD para formular (mediante un lenguaje) acciones como, por ejemplo:

- Describir la base de datos (diseñada por el diseñador).
- Actualizar la base de datos (con los datos facilitados).
- Solicitar la recuperación de información (consultas).

1) Lenguajes para usuarios informáticos expertos

Administradores y programadores requieren **lenguajes potentes y flexibles** para escribir procesos complejos que les permitan definir, administrar, extraer y manipular datos o, incluso, crear y ajustar las aplicaciones verticales que acceden a ellos.

a) En muchos casos, utilizan DDL y DML en su **forma hospedada en un lenguaje anfitrión**, por lo que éste deberá permitir **llamadas** al SGBD **desde un programa** de aplicación.

b) En otros casos, el lenguaje utilizado sirve para parametrizar la base de datos antes del inicio de los trabajos del usuario final.

2) Lenguajes para usuarios ocasionales

Estos usuarios, como simples gestores de las aplicaciones que utilizan, normalmente sólo suelen hacer consultas. Para especificarlas, necesitan un **lenguaje de consulta de alto nivel** que sea bastante sencillo (sin utilizar órdenes estructuradas), aunque dé un rendimiento bajo en tiempo de respuesta.

3) Lenguajes para usuarios simples y paramétricos

Los usuarios paramétricos (dedicados, por ejemplo, a introducir datos masivamente) necesitan **lenguajes muy eficientes y compactos**, quizá especializados en tipos concretos de tareas (transacciones programadas), aunque no sean fáciles de aprender.

Los usuarios paramétricos

Los usuarios paramétricos y las transacciones programadas se tratan en el subapartado "Usuarios finales" dentro del apartado "Usuarios de las bases de datos".

Para los usuarios paramétricos, los ocasionales y otros que no quieren aprender los detalles de un lenguaje de consulta de alto nivel, el SGBD proporciona **interfaces amigables para el usuario** que permiten interactuar con la base de datos. Estas interfaces se analizan a continuación.

15.4. Interfaces del SGBD

El SGBD puede ofrecer diferentes tipos de interfaces según el método utilizado para realizar las solicitudes y la categoría de usuario a la que van dirigidas.

15.4.1. Interfaces y método

Según el método utilizado para hacer las solicitudes al sistema (tecleando instrucciones, eligiendo opciones de menú, haciendo clic en los botones con el ratón, etc.), las interfaces pueden ser de cuatro tipos:

1) Interfaces de lenguaje natural

Aceptan solicitudes escritas (en una línea de órdenes o de comando) utilizando un subconjunto de palabras de algún idioma, habitualmente el inglés. La interfaz consulta las palabras estándar de su propio esquema (el *intérprete de órdenes*) para interpretar la solicitud. Si la interpretación tiene éxito, la interfaz **genera una consulta** de alto nivel (que corresponde a la solicitud realizada por el usuario en lenguaje natural) y la envía al SGBD para su procesamiento. En caso contrario, se inicia un diálogo con el usuario para aclarar la solicitud. También se puede denominar *interfaz de órdenes*.

2) Interfaces de navegación basadas en menús

Son interfaces que presentan listas de opciones denominadas *menús*, que guían al usuario para formular una solicitud.

Menú

Un **menú** es una lista que se presenta en pantalla y que muestra varias opciones junto con un mecanismo que recoge la opción elegida y, seguidamente, lo ejecuta. Un menú hace innecesario memorizar las instrucciones y la sintaxis específica de un lenguaje de consulta, puesto que permite elaborar la solicitud paso a paso.

Actualmente, los **menús desplegables** (muy utilizados en interfaces basadas en ventanas) permiten que el usuario examine los contenidos de la base de datos utilizando una forma de exploración no estructurada.

Éste ha sido un tipo de interfaz muy popular y cómodo hasta la llegada al mercado de los SGBD basados en ventanas (*windows*). Aun así, la existencia de programas de aplicación que no utilizan interfaces gráficas de usuario (GUI) provoca que se siga utilizando.

3) Interfaces basadas en formularios

Este tipo de interfaces presentan a cada usuario un conjunto de formularios que permite atacar los datos almacenados (básicamente, modificar o añadir) de acuerdo con las especificaciones de seguridad y acceso definidas previamente.

Formulario

Un **formulario**, que también recibe los nombres de *ventana de usuario, formato de pantalla* o, simplemente, *pantalla*, forma parte del esquema externo del usuario. Es la manera más cómoda y natural para comunicar al usuario con la base de datos y ofrece una buena presentación y organización de los datos. Internamente, hace lo mismo que se puede hacer trabajando con mandos directos del lenguaje de consulta.

Un formulario es una ventana o cuadro estructurado de presentación en pantalla con áreas predefinidas de bloques de datos (contenedores lógicos de cuadros de texto, listas, etc.) y otros elementos de control (casillas de verificación, botones, etc.) que permiten hacer tareas que se presentan a continuación:

1) Establecimiento de un orden predeterminado para la gestión de los datos.

2) Visualización de los datos **en un formato** determinado, según el tipo de dato almacenado. Muy esporádicamente, un formulario se usa para mostrar datos, ya que para ello es mejor elaborar un informe.

3) Recuperación de datos de registros que coinciden con valores especificados. La consulta de datos tampoco es el uso más habitual de los formularios.

4) Filtrado de la información. Esta tarea se puede llevar a cabo en dos niveles:

a) Filtrado en la introducción mediante una serie de controles.
Por ejemplo, impidiendo la introducción de letras en un campo numérico o limitando las opciones de una lista desplegable en función de un dato introducido previamente.

b) Filtrado visual en la recuperación de los datos.
Por ejemplo, mostrando un registro único o varios registros a la vez, y ver algunos o todos los atributos de los datos subyacentes.

5) Gestión de gran parte de la coherencia de la información para obtener resultados de calidad durante cualquier acción sobre los datos.

Por ejemplo, en una base de datos bibliográfica, impidiendo que se borre el registro correspondiente a un autor si antes no se han eliminado todas sus obras.

6) Modificación y eliminación de datos existentes.

7) Introducción de datos nuevos.

Se trata de aplicaciones construidas bajo el concepto *cliente-servidor*, ya que están a la espera de una acción por parte del usuario para responder de manera programada. Es decir, permanecen inactivas mientras el usuario lee datos, escribe texto o no interactúa, pero, a una petición de éste (por ejemplo, un clic de ratón sobre un botón), responden, como servidor, con la ejecución de la acción o transacción solicitada. Por este motivo, también se denominan **interfaces de transacciones programadas**.

Suelen estar destinadas a usuarios paramétricos que habitualmente realizan un tipo de transacciones estándar (consultas y actualizaciones).

4) Interfaces gráficas de usuario

Las interfaces gráficas de usuario (GUI, *graphic user interface*, en inglés) suelen presentar los esquemas de la base de datos en forma de diagrama y permiten que el usuario especifique una consulta manipulando el diagrama. Suelen utilizar un *dispositivo apuntador*, como el ratón, para elegir las partes del diagrama del esquema mostrado. Habitualmente, los elementos gráficos (ventanas, iconos, botones, etc.) también se combinan con menús y formularios, como es el caso del sistema *Windows* de *Microsoft*.

15.4.2. Interfaces y usuarios

Según los tipos de usuarios a los que van destinadas, destacan las siguientes interfaces de usuario:

1) Interfaces para usuarios paramétricos

Disponen de un conjunto restringido de **operaciones abreviadas** que este tipo de usuarios, como por ejemplo cajeros de un banco, deben hacer repetidamente. El objetivo es que para realizar cada tarea o solicitud concreta, sólo haya que introducir una cantidad mínima de parámetros o efectuar un número reducido de pulsaciones de teclado. Por ejemplo, se pueden programar las teclas de función para que se inicien instrucciones determinadas.

2) Interfaces para el administrador de la base de datos

La mayoría de los SGBD contienen **instrucciones privilegiadas** que sólo puede utilizar el administrador.

Son ejemplos de ello las instrucciones para establecer los parámetros del sistema, crear perfiles de usuario, otorgar roles o autorizaciones a los perfiles, crear conexiones a bases de datos remotas, modificar los esquemas y reorganizar la estructura de almacenamiento de una base de datos.

15.5. El núcleo del SGBD

El núcleo es el componente encargado de garantizar que el almacenamiento de los datos y el acceso a éstos sean correctos, seguros, íntegros y eficientes. Los mecanismos de almacenamiento de los datos, transparentes para el usuario, son tan importantes como los dedicados a proteger la base de datos contra posibles riesgos y peligros (intencionados o accidentales).

Por ello, un SGBD multiusuario debe ofrecer un conjunto de controles de seguridad de acceso y de seguridad interna frente a pérdidas de integridad; controles de concurrencia y recuperación frente a fallos del sistema, etc. Otro aspecto importante para obtener un buen rendimiento es la optimización de consultas.

Recuperación y concurrencia en sistemas monousuario

Los SGBD monousuario, normalmente, no ofrecen sistemas de recuperación automática por si se presenta un fallo. Se considera que el usuario es responsable de preparar copias de seguridad de la base de datos y recuperar sus datos manualmente.

Por otro lado, la concurrencia no es pertinente en sistemas monousuario.

Corresponde al núcleo, como interfaz con el sistema operativo, transformar las peticiones de datos que recibe del usuario en instrucciones que el sistema operativo pueda entender y gestionar. El núcleo proporciona una interfaz entre los datos (a nivel de almacenamiento físico) y los programas de aplicación diseñados para su manipulación. Se puede considerar que actúa como intérprete entre el usuario y los datos. Cada operación que se debe hacer "contra" la base de datos debe ser permitida previamente por el núcleo, que, una vez interpretada y validada la instrucción, o bien ejecuta la operación devolviendo el resultado de ésta al programa (o procedimiento) que lo ha solicitado o bien la rechaza.

Interfaz con el sistema operativo

En la figura 19 del subapartado "Componentes del SGB" de este mismo apartado se ve cómo el núcleo actúa como interfaz entre el SGB y el S.O.

El motor del SGBD (que siempre se ejecuta en el servidor) suele tener módulos que se concentran en el núcleo del sistema, para ejecutar, como mínimo, las funciones siguientes:

- Control de la seguridad y la privacidad.
- Gestión de vistas.
- Interpretación del lenguaje de base de datos (SQL).
- Optimización de consultas.
- Control del acceso concurrente (por ejemplo, la gestión de bloqueos).
- Gestión de memorias intermedias (*buffers*).
- Gestión del espacio de la memoria externa.
- Método de acceso (por ejemplo, los índices).
- Gestión de procesos (multitarea, paralelismo).
- Gestión de transacciones, disparadores y procedimientos almacenados.
- Mantenimiento de diarios de actualizaciones (*log*).

Arquitectura funcional y procesamiento de una sentencia

Para describir y comprender la arquitectura funcional de un SGBD se puede seguir el procesamiento de una sentencia LMD del lenguaje de base de datos (habitualmente SQL).

La estructura interna de un SGBD moderno incluye una gran cantidad de componentes que son módulos de software. Desde un punto de vista de alto nivel, el núcleo del SGBD se puede considerar constituido, únicamente, por tres bloques llamados gestores: el gestor de sentencias, el gestor de concurrencia y el gestor de datos.

Los componentes del núcleo del SGBD

Los componentes del núcleo del SGBD intervienen en la ejecución de una consulta hecha por un programa de usuario como se muestra en el esquema de la gestión de acceso a los datos realizada por el SGBD del apartado "Acceso del SGBD a los datos".

15.5.1. Gestor de sentencias

Es la parte del SGBD que interactúa directamente con el usuario. Por este motivo, recibe el nombre de parte **de delante** (*front-end*) o sección frontal del SGBD. Se corresponde con el procesador **cliente** que incluye la interfaz de usuario y los procesos o servicios en primer plano. Engloba cuatro componentes con funcionalidades muy diferenciadas:

1) Componente de control de seguridad de acceso

Antes de procesar una sentencia, debemos comprobar si el usuario que pide la ejecución está autorizado para ejecutar el conjunto de operaciones que implica la sentencia (para determinarlo se usa un conjunto de técnicas de seguridad relativas a la privacidad de los datos). Si la sentencia no supera estos filtros de seguridad, es rechazada por el sistema y su ejecución no sigue adelante.

2) Componente de transformación de esquemas

Si la sentencia es autorizada, el esquema externo se transforma en el esquema conceptual. Este proceso, llamado **procesamiento de vistas**, permite eliminar las referencias a vistas y obtener una sentencia de operaciones más homogé-

nea referida únicamente a entidades u objetos del esquema conceptual (tablas, en el caso de bases de datos relacionales).

3) Componente de control de integridad

Debemos comprobar que los datos almacenados satisfagan los **requerimientos de integridad** definidos en el esquema conceptual y, así, asegurar la consistencia (claves primarias y foráneas, y otras restricciones predefinidas).

Una vez superados todos estos controles, la sentencia se debe ejecutar. Pero si el lenguaje de la sentencia es declarativo, se debe **transformar** (traducir + descomponer) en un conjunto de acciones expresadas en un lenguaje procedimental que pueda ser entendido por el sistema operativo.

4) Componente de procesamiento de sentencias

Primero se debe traducir la sentencia a operaciones, que posteriormente se deben descomponer en operaciones más elementales sobre los elementos físicos de la base de datos. (En una base de datos relacional, una sentencia SQL se traduce a operaciones del álgebra relacional.)

Las sentencias de los usuarios de la base de datos

Los usuarios ocasionales ejecutan consultas interactivas, los usuarios paramétricos ejecutan transacciones programadas, los programadores de aplicaciones crean programas de aplicación que ejecutan sentencias de manipulación de datos (LMD), y los administradores de la base de datos ejecutan sentencias de definición de datos (LDD) e instrucciones privilegiadas.

Las solicitudes de los clientes (usuarios y programas de aplicación) se transforman en código de acceso a la base de datos para ser ejecutadas por el procesador de base de datos en tiempo de ejecución.

El componente de procesamiento de sentencias incluye los módulos siguientes:

- **Compilador de consultas**. Maneja las consultas de alto nivel que el usuario introduce de manera interactiva. Analiza su sintaxis, las compila o interpreta y crea el código de acceso a la base de datos (para ejecutar este código, genera llamadas al **procesador en tiempo de ejecución**).

- **Precompilador**. Extrae las instrucciones escritas en LMD hospedado en un programa de aplicación escrito en un lenguaje de programación anfitrión (y las agrupa en **un módulo de solicitudes a la base de datos**). Estas instrucciones se envían al **compilador de LMD**. El resto del programa se envía al **compilador del lenguaje anfitrión**.

- **Compilador de LMD**. Convierte las instrucciones recibidas (de parte del precompilador) en código objeto para el acceso a los datos.

- **Ligador**. Enlaza (monta, *linkedit*) el código objeto de las instrucciones en LMD y el del resto del programa y forma una **transacción programada** (que recibe el nombre de **plan de acceso**) cuyo código ejecutable incluye llamadas al procesador de la base de datos durante el tiempo de ejecución.

- **Optimizador**. Es uno de los subcomponentes principales del ligador. En cada una de las dos transformaciones de la sentencia (traducción + descomposición), se encarga de elegir la mejor combinación posible de operaciones procedimentales que satisfaga la ejecución de la sentencia LMD procesada por el ligador.

Por el hecho de que las sentencias se pueden ejecutar de maneras diferentes (no todas las operaciones tienen el mismo coste en tiempo de respuesta), se debe establecer el método de acceso más idóneo para aumentar la rapidez sin que afecte negativamente al resto del sistema. Es decir, se deben reducir al mínimo los accesos al disco para la obtención de los datos, la utilización del procesador (CPU), etc.

Los SGBD relacionales en la optimización de sentencias

Un SGBD se considera eficiente cuando la respuesta a una sentencia se lleva a cabo en el menor tiempo posible, para lo que debe disponer de un método que optimice esta respuesta. Esta optimización sólo es aplicable en SGBD relacionales, que, aplicando el álgebra relacional, deben encontrar la manera más corta de ejecutar una sentencia.

El proceso de optimización acaba reflejando este resultado en un procedimiento llamado **plan de acceso o de ejecución**, que es la estrategia de ejecución de una sentencia que tiene asociado un consumo mínimo de recursos (básicamente, tiempos de respuesta).

Procesamiento (y optimización) de consultas

Una consulta expresada en un lenguaje de alto nivel (como SQL) debe pasar primero por un análisis léxico, un análisis sintáctico y una validación. El proceso en una base de datos relacional incluye:

- **Analizador léxico.** Identifica los símbolos del lenguaje (como las palabras clave del lenguaje SQL, los nombres de los campos y los nombres de las tablas) en el texto de la consulta.

- **Analizador sintáctico**. Revisa la sintaxis de la consulta para determinar si está formulada de acuerdo con las reglas sintácticas del lenguaje de consulta.

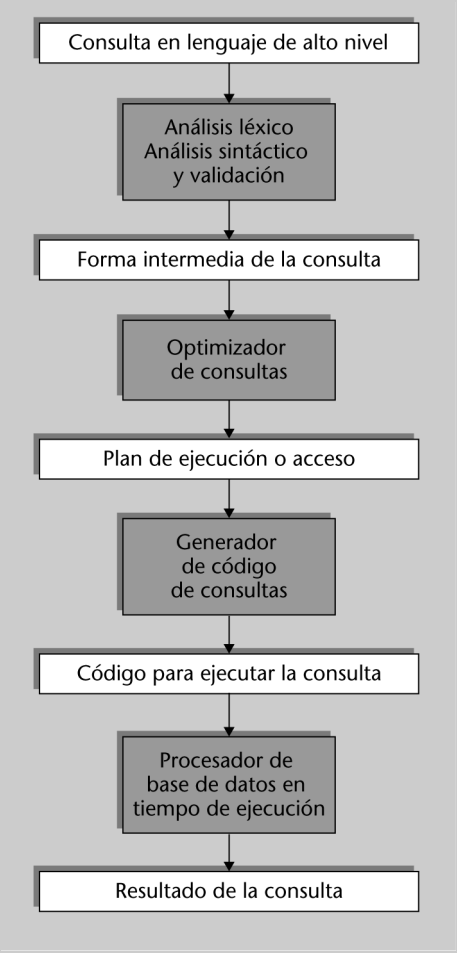
- **Validación de la consulta**. Se comprueba que todos los nombres de campos y de tablas sean válidos y tengan sentido desde el punto de vista semántico en el esquema de la base de datos.

A continuación, se crea una **representación interna** de la consulta, normalmente en forma de estructura de datos de árbol (llamada *estructura de árbol de consulta*).

Seguidamente, el SGBD debe crear la estrategia de ejecución o plan **de ejecución** para obtener el resultado de la consulta a partir de los ficheros implicados de la base de datos. El proceso de elegir la estrategia de ejecución más adecuada para procesar una consulta se denomina **optimización de consultas**.

En la figura se muestran los diferentes pasos para procesar una consulta de alto nivel.

Figura 22. Procesamiento de una consulta de alto nivel



- El módulo **optimizador de consultas** es responsable de elegir el método de acceso o plan de ejecución más eficiente para llevar a cabo la consulta.
- El **generador de código** genera el código necesario para ejecutar la consulta. El código se puede ejecutar directamente (modo interpretado) o almacenar y ejecutar cuando sea necesario (modo compilado).
- El **procesador de base de datos en tiempo de ejecución** se encarga de ejecutar el código de la consulta para producir el resultado de la misma. Si se presenta un error durante la ejecución, este procesador genera un mensaje de error.

El SGBD lleva a cabo tres tipos de optimización: sintáctica, física y semántica. Normalmente, la optimización semántica no se encuentra implementada en los SGBD comerciales.

Cuando la sentencia LMD se ha transformado en el **plan de acceso**, cada una de las operaciones elementales que lo forman sigue otros tratamientos a través del resto del SGBD.

En primer lugar, pasa por el **gestor de concurrencia** y, finalmente, por el **gestor de datos**.

15.5.2. Gestor de concurrencia

Este gestor (también llamado **componente de servicios de bloqueo**) suministra los controles necesarios para gestionar el acceso concurrente a bases de datos multiusuario evitando la pérdida de consistencia o integridad de los datos. Es decir, controla la ejecución concurrente de múltiples operaciones elementales de actualización correspondientes a diferentes sentencias que afectan al mismo dato y comprueba que la acción que intenta hacer cada operación elemental no entre en conflicto con la acción de otra operación que también se esté ejecutando.

15.5.3. Gestor de datos

Finalmente, si no hay ningún problema de concurrencia, cada operación elemental se ejecuta "contra" la base de datos. De esto se encarga el gestor de datos, que representa la parte del SGBD más alejada del usuario. Por ello, se suele denominar **parte trasera** (*back-end*) o sección posterior del SGBD. Corresponde al servidor donde se producen los procesos en segundo plano (a veces, se denomina **componente de servicios de base de datos**).

El gestor de datos es responsable de la **interacción con el sistema operativo** (en particular, con su administrador de archivos). La estructura utilizada para el almacenamiento de los archivos varía en función del sistema operativo implantado en el ordenador.

El gestor de datos es uno de los componentes de más complejidad del SGBD. Depende del propio SGBD, de las características del sistema operativo y del hardware en el que se implante.

Factores que afectan a las funciones del gestor de datos

La correcta ejecución de las funciones asignadas al gestor de la base de datos depende de factores muy variados, de entre los que destacan los factores siguientes:

- El volumen de la base de datos.
- Las estructuras físicas definidas para el almacenamiento de los datos.
- Los procedimientos desarrollados para la manipulación de los datos.
- Las características del hardware.
- La calidad del propio gestor de datos.

El gestor de datos se divide en varios subcomponentes principales, cuyas funciones se detallan a continuación:

1) Procesador de base de datos en tiempo de ejecución

Este componente se encarga de los accesos a la base de datos durante la ejecución. Recibe operaciones de recuperación o de actualización y las ejecuta sobre la base de datos según las instrucciones del **plan de acceso** (que supervisa el acceso al disco, mediante el gestor de datos almacenados).

2) Gestor de recuperación

El gestor de recuperación se encarga de tomar nota de las operaciones elementales (lectura o escritura de datos) antes de que no se produzcan en **un diario de operaciones** que permitirá recuperarlas en caso de fallos posteriores del sistema.

3) Gestor de datos almacenados

Este gestor controla el acceso a la información almacenada en el disco (tanto de la **base de datos** como del **catálogo**).

Puede utilizar servicios básicos del sistema operativo para transferir físicamente los datos entre el disco y la memoria principal. Concretamente, es responsable de las funciones atribuidas al **administrador de archivos**. Ahora bien, durante la ejecución de su tarea básica, controla otros aspectos de la transferencia de datos invocando a diferentes componentes cuando es necesario llevar a cabo funciones de detalle (como el bloqueo, la conexión de usuarios, el manejo de memorias intermedias, etc.).

El administrador de archivos

Las funciones del administrador de archivos que permiten manipular las estructuras de almacenamiento de datos se comentan en el paso 5 del esquema de la gestión de acceso a los datos hecha por el SGBD, del apartado "Acceso del SGBD a los datos".

4) Gestor de memoria intermedia

El gestor de memoria intermedia resuelve todas las operaciones elementales que puede en la memoria intermedia y sólo accede al disco cuando es necesario. Cuando los datos son en esta área de memoria intermedia, pueden ser procesados por otros módulos del SGBD o por programas de aplicación.

Otros módulos del núcleo

El núcleo del SGBD incluye otros componentes:

- Procesadores de interfaces.
- Módulo para implementar el catálogo.
- Módulos de comunicación con otros programas como el sistema operativo o los compiladores de lenguajes de programación.

Interacción con el sistema operativo

Cuando se requiere acceso al disco (a la base de datos o al catálogo), el SGBD interactúa con el sistema operativo por medio del módulo gestor de datos.

Cuando se ejecutan programas anfitrión que acceden a los datos desde el LMD hospedado en éstos, por cada programa se crea una unidad de ejecución en la que hay un área de trabajo de usuario, con sus áreas de entrada, salida y de comunicación con el gestor de datos. El área de comunicación es la responsable de recibir los mensajes y la información de control procedentes del gestor.

La ejecución de las **órdenes del LMD** sigue los pasos siguientes:

- Una **unidad de ejecución** llama al gestor de datos. Esta llamada tiene información sobre el esquema externo del cliente (usuario final) que quieren acceder a la base de datos.
- El gestor de datos completa la información del esquema externo con la estructura conceptual e interna. La información de estas estructuras está almacenada en forma de esquemas conceptual e interno en el **catálogo**.
- El gestor de datos traduce la petición, la convierte en órdenes para los **métodos de acceso** del sistema operativo y éstos acceden al disco en el que está almacenada la base de datos.
- Se recuperan los datos solicitados y se guardan en un **área de memoria intermedia**, desde la que son transferidas al **área de trabajo** (de E/S) del usuario.

- El gestor de datos, cuando ha completado la manipulación de los mismos, pasa al **área de comunicación** los indicadores de estado en los que se indica si la operación ha finalizado correctamente. Si al revisar el estado de los indicadores no hay ninguno que haya tenido problemas, los datos son utilizados por el programa de aplicación.

Resumen

La definición de una serie de **conceptos básicos** al inicio del módulo nos ha ayudado a entender los contenidos sobre sistemas de base de datos desarrollados a lo largo de este módulo. Hemos definido *base de datos* como el conjunto estructurado de ficheros de datos entre los que se establecen relaciones, y sistema gestor de base de datos (SGBD), como el conjunto de software que permite la gestión automática de una base de datos.

Hemos hecho un repaso de la **evolución de la gestión de datos** empezando por los sistemas manuales de fichas de cartulina, pasando por la gestión informatizada con ficheros de datos, hasta los sistemas de base de datos. Hemos visto que los ficheros de datos están orientados al proceso (ponen el énfasis en el tratamiento de los datos, que están dispersos en ficheros diseñados para una aplicación determinada), mientras que las bases de datos están orientadas a los datos (que se organizan y se mantienen en un conjunto estructurado no diseñado para una aplicación concreta).

Para especificar cómo se organizan los datos, hemos descrito las **estructuras de datos** más utilizadas en los sistemas de información: matrices, cadenas de caracteres, registros, árboles, listas y listas vinculadas.

Para justificar el uso de las bases de datos, hemos analizado los **problemas** que supone la utilización de sistemas **de gestión de ficheros** de datos, que hemos agrupado en:

- los que afectan a los ficheros: necesidad de controlar la integridad semántica, dificultad para gestionar el control de autorizaciones y falta de control de concurrencia, y

- los que afectan a los datos: redundancia, inconsistencia, aislamiento, dificultad de acceso a los datos y dependencia de los programas.

Hemos nombrado los principales **objetivos de las bases de datos**:

- Centralizar la información e integrar los datos y los programas.

- Estandarizar una gestión de datos eficaz, universal e independiente del hardware y del sistema operativo.

- Proporcionar un sistema de almacenamiento con una interfaz de usuario fácil de consultar y de modificar.

- Optimizar el coste del soporte de almacenamiento de los datos.

Las **características** que deben cumplir nos permiten manifestar que "las bases de datos permiten la integración de la información del sistema para evitar redundancia, sin que por ello se pierdan las diferentes perspectivas que tienen los usuarios (vistas), y las herramientas de software (SGBD) usadas para gestionarlas aseguran la independencia (de los programas respecto a la representación física de los datos), la integridad y la seguridad de los datos".

Hemos agrupado las **ventajas de las bases de datos** en tres bloques:

1) Las ventajas referidas a los datos

- Disminución de la redundancia.
- Prevención de la inconsistencia.
- Representación de asociaciones entre los datos.
- Mantenimiento de la integridad.
- Suministro de copias de seguridad y recuperación.
- Almacenamiento de las especificaciones de la base de datos.
- Independencia entre los programas y los datos.
- Compatibilidad entre formatos.

2) Las ventajas referidas a los usuarios (en sistemas multiusuario)

- Aplicación eficiente de restricciones de seguridad.
- Suministro de múltiples interfaces de usuario.
- Soporte de múltiples vistas de datos.
- Disponibilidad de información actualizada.
- Acceso rápido y sencillo a la información.
- Accesibilidad múltiple y simultánea.
- Flexibilidad para atender nuevos requerimientos.
- Coherencia de los resultados.

3) Las ventajas referidas a la organización

- Integración de toda la información de la organización.
- Eficiencia del almacenamiento físico.
- Reducción del coste de almacenamiento y de mantenimiento.
- Reducción del coste de formación del personal.
- Economía de escala.
- Capacidad para establecer normas.
- Reducción del tiempo de creación de aplicaciones.

Hemos visto que la implantación de una base de datos puede implicar ciertos **inconvenientes**: inversión inicial elevada, implantación larga y difícil, rentabilidad a largo plazo, ausencia de normalización de los SGBD comerciales, problemas de integración de bases de datos, riesgo de frustración de los usuarios, etc.

Asimismo, hemos comentado que hay **situaciones en las que es mejor utilizar un sistema de ficheros** en lugar de un sistema de base de datos: si el sistema existente está muy definido, es sencillo y no debe variar; si no es necesario el acceso multiusuario y simultáneo a los datos, o si el tiempo de respuesta requerido por las aplicaciones no se puede cumplir con el SGBD.

Hemos aprendido que los principales **elementos de un sistema de bases de datos** son el contenido (los datos y su descripción), el equipo lógico (el SGBD, el sistema operativo, el software de comunicaciones, etc.), el equipo físico (el ordenador o los ordenadores en los que se instala el SGBD y en los que se almacena la base de datos, y los dispositivos que permiten la gestión) y el personal humano (los usuarios que interactúan con la base de datos).

Hemos clasificado los **usuarios de una base de datos** en dos grandes categorías:

1) El personal informático que interviene en el desarrollo y mantenimiento de la base de datos: directivos, analistas, diseñadores de bases de datos, programadores, administradores, responsables de explotación. Aquí también hemos incluido usuarios sin interés en el contenido de la base de datos relacionados con el diseño, la creación y el funcionamiento del software del SGBD: desarrolladores del SGBD y de herramientas y técnicos de mantenimiento.

2) Los usuarios finales, entre los que hemos distinguido los tipos siguientes:

a) Usuarios habituales, que utilizan asistentes y sistemas visuales guiados. Un tipo específico, los *usuarios paramétricos*, que introducen datos de manera masiva mediante transacciones programadas, son operadores de consola como cajeros de banco o encargados de reservas.

b) Usuarios avanzados o expertos, familiarizados con la mayor parte de los recursos del SGBD, que lo utilizan para implementar sistemas que ejecutan procesos propios y cumplen sus requerimientos específicos. Suelen ser ingenieros, científicos y analistas de negocio que utilizan los datos como soporte a la toma de decisiones.

c) Usuarios ocasionales o esporádicos que requieren información diferente en cada ocasión y utilizan pocos recursos del SGBD. Un tipo concreto de este grupo es el *usuario crítico*, por el elevado tiempo de respuesta y la dificultad de obtener resultados en el formato y el nivel de detalle requeridos; suele ser personal de gerencia o *staff* con poder decisorio en la organización.

d) Usuarios autónomos que mantienen bases de datos personales y tienen gran habilidad en el uso de paquetes de software específicos con interfaces gráficas muy amigables.

Más adelane hemos definido los conceptos de **modelo**, **esquema** y **estado de la base de datos** de este modo:

- Un modelo de datos es una herramienta intelectual que permite conseguir el proceso de abstracción que conduce de la parte del mundo real que interesa estudiar (denominada universo del discurso) al mundo de los datos.

- El esquema es la descripción de la estructura de la base de datos.

- El estado de la base de datos es el conjunto de valores que toman los objetos de un esquema en un momento dado y que puede estar vacío (sin datos), inicial o actual.

Para señalar la relación **entre estos conceptos** hemos apuntado que "la descripción del universo del discurso del mundo real mediante un modelo de datos da como resultado un esquema, que puede tomar varios estados". Además, hemos definido conceptos importantes para el diseño de bases de datos, como ocurrencia, tipo, intensión y extensión.

Los modelos de datos ofrecen diferentes **tipos de abstracción** para facilitar la representación de los datos en el diseño de bases de datos y establecer vínculos entre los elementos del modelo. Hemos visto que la clasificación establece un vínculo entre un tipo de objeto y cada ocurrencia, mientras que la generalización, la agregación y la asociación establecen el vínculo entre tipos de objetos (y, por lo tanto, también entre sus ocurrencias). Además, hemos señalado que estas abstracciones son síntesis conceptuales y los respectivos procesos inversos (la instanciación, la especialización, la desagregación y la disociación) son refinamientos conceptuales.

Para abstraer, esquematizar y entender las características fundamentales de los SGBD, hemos analizado la **arquitectura de los SGBD** desde diferentes enfoques:

1) *Arquitectura operacional*: según cómo se distribuye el procesamiento del SGBD en el sistema informático, puede ser:

a) *Arquitectura centralizada*: un ordenador central realiza todo el procesamiento y almacena los datos a los que acceden los usuarios a través de terminales remotos.

b) *Arquitectura cliente-servidor*: el procesamiento se reparte entre el cliente, que proporciona las aplicaciones e interfaces de usuario en un ordenador personal, y el *servidor*, que efectúa las tareas propias de la base de datos (almacenamiento y acceso a los datos).

2) *Arquitectura de referencia*: según los niveles de abstracción de los datos en el SGBD, puede ser de dos o de tres niveles.

3) *Arquitectura externa o aparente*: muestra las utilidades del SGBD que el usuario ve como un conjunto estructurado (procesadores de consultas, generadores de formularios, de informes, de menús, etc.).

4) *Arquitectura interna o funcional*: muestra cómo el SGBD estructura internamente su funcionalidad en módulos de software para ejecutar funciones específicas (interpretación de SQL, optimización de consultas, gestión de vistas, gestión de transacciones, control de concurrencia, etc.)

Hemos visto que los SGBD actuales deben respetar la **arquitectura de tres niveles** (propuesta por ANSI/SPARC), que aporta una gran flexibilidad para los cambios y establece una división de la base de datos según la perspectiva desde la que ésta es vista. Hemos descrito estos **niveles de abstracción** del SGBD, que se corresponden con sus tres principales grupos de usuarios (finales, programadores y administradores), de la manera siguiente:

- El *nivel externo* gestiona la información desde el punto de vista individual de cada grupo de usuarios.

- El *nivel conceptual* describe la estructura de la base de datos para todos los usuarios independientemente del ordenador en el que ésta se implemente.

- El *nivel interno* describe la información en función del sistema en el que se implementará la base de datos.

Además, hemos estudiado las **correspondencias** entre estos niveles y hemos señalado las limitaciones que supone la ligadura para la independencia de los datos.

Como síntesis de la materia expuesta, hemos presentado una visión integrada de los elementos y las características de un sistema de base de datos entendido como el sistema que integra el SGBD y la base de datos en un sentido amplio, que hemos denominado **estructura global del sistema de base de datos**.

Después, hemos visto cómo se organizan los datos en soportes de **almacenamiento** secundario, que permiten guardar de manera persistente grandes volúmenes de datos que pueden ser recuperados, actualizados y procesados por el SGBD cuando sea necesario. Normalmente, se utilizan discos magnéticos en los que la memoria se divide en bloques. Hemos definido registro físico (bloque o página) como la unidad de transferencia de datos entre el disco y la memoria principal, y registro lógico como la unidad de transferencia entre la memoria intermedia y el programa de usuario.

La jerarquía de abstracción de datos existente en un sistema de base de datos (bloques de datos – ficheros de registros – estructures de base de datos) nos ha permitido explicar el **acceso del SGBD a los datos** para satisfacer una

consulta. Para ello, hemos mostrado el esquema general del flujo de datos y de control que sigue el proceso de ejecución de una consulta hecha por un programa de usuario al SGBD. Hemos visto que el programa de usuario percibe la base de datos como un conjunto de datos, el SGBD la ve como un conjunto de registros almacenados, el administrador de archivos la ve como un conjunto de bloques y el administrador de E/S la percibe como es físicamente en realidad.

A partir de los objetivos de las bases de datos, hemos deducido que las **funciones principales del SGBD** son la definición, la manipulación y el control de los datos. El análisis de estas funciones nos ha permitido determinar los **componentes** de software básicos del SGBD, que hemos dividido en dos grupos:

- El motor, que se encarga de las tareas de gestión de los datos (acepta las consultas de los clientes, las procesa y devuelve los resultados), está formado por: lenguajes de base de datos, diccionario de datos y núcleo del SGBD.

- Las interfaces, que permiten omitir el uso del lenguaje de base de datos, y las utilidades y herramientas, que simplifican las tareas de los usuarios principales.

Como síntesis, hemos presentado la correlación entre los objetivos de las bases de datos, las funciones del SGBD y sus componentes asociados.

Hemos clasificado los **lenguajes de base de datos** que permiten manejar la base de datos según varios criterios:

- Según la función, tenemos lenguaje de definición de datos (LDD), especializado en la creación de la estructura de la base de datos, y un lenguaje de manipulación de datos (LMD), especializado en la consulta y el mantenimiento de los datos.

- Según la *modalidad de uso* del lenguaje, distinguimos el modo interactivo (en el que el usuario introduce por medio del teclado las instrucciones que son analizadas, verificadas y ejecutadas por el software) y el modo programado (en el que el usuario ejecuta un programa de aplicación sobre el sistema operativo).

- Según las *aplicaciones que soportan*, pueden ser lenguajes de alto nivel (declarativos no procedimentales) o lenguajes de bajo nivel (procedimentales).

- Según *el modelo de datos* en el que se fundamenta la base de datos. Hemos destacado SQL, lenguaje relacional de consulta basado en el álgebra relacional.

- Según el tipo de usuario a quien van dirigidos, se pueden clasificar en lenguajes para usuarios informáticos expertos, para usuarios ocasionales y para usuarios paramétricos.

Después hemos presentado los tipos de **interfaces del SGBD** según el método utilizado para realizar las solicitudes (lenguaje natural, menús, formularios o gráfico) y según la categoría principal de usuario a quien se dirigen (administrador o usuario paramétrico).

Finalmente, hemos visto que el **núcleo del SGBD** proporciona una interfaz entre los datos y los programas de aplicación diseñados para su manipulación, y que realiza una serie de tareas que permiten garantizar que el almacenamiento de los datos y el acceso a éstos sean correctos, seguros, íntegros y eficientes. También hemos analizado los módulos de software que incorpora al núcleo:

- Gestor de sentencias, con componentes para control de seguridad, la transformación de esquemas, el control de integridad y el procesamiento de sentencias (este último incluye compilador de consultas, precompilador, compilador del LMD, ligador y optimizador).

- Gestor de concurrencia. Gestiona el acceso concurrente a bases de datos multiusuario y evita la pérdida de consistencia o integridad de los datos.

- Gestor de datos. Ejecuta cada operación elemental contra la base de datos e interactúa con el administrador de archivos del sistema operativo. Incluye procesador en tiempo de ejecución, gestor de recuperación, gestor de datos almacenados y gestor de memoria intermedia.

Bibliografía

Connolly, T. M.; Begg, C. E. (2005). *Sistemas de bases de datos* (4.ª ed.). Madrid: Addison Wesley (Pearson Educación).

Date, C. J. (2001). *Introducción a los sistemas de bases de datos* (7.ª ed.). Madrid: Prentice Hall (Pearson Educación).

Elmasri, R.; Navathe, S. B. (2007). *Fundamentos de sistemas de bases de datos* (5.ª ed.). Madrid: Addison-Wesley (Pearson Educación).

García-Molina, H.; Ullman, J. D.; Widom, J. (2009). *Database Systems: The Complete Book* (2.ª ed.). Prentice Hall.

Ramakrishnan, R.; Gehrke, J. (2002). *Database Management Systems* (3.ª ed.). Boston, Massachusetts: McGraw-Hill.

Silberschatz, A.; Korth, H. F.; Sudarshan, S. (2006). *Fundamentos de bases de datos* (5.ª ed.). Madrid: McGraw-Hill.

Smith, P. D.; Barnes, G. M. (1987). *Files and Databases: An Introduction*. Reading, Massachusetts: Addison-Wesley.

Ullman, J. D.; Widom, J. (1999). *Introducción a los sistemas de bases de datos* (1.ª ed.). Prentice Hall.