Base de Datos

Unidad 1 – Los Sistemas de Almacenamiento de la Información

Índice

1. Introducción

La información forma parte de nuestra vida diaria y está presente en casi todas nuestras actividades: usar un GPS, hacer una llamada, pagar con tarjeta,... Esto ha llevado a que sea esencial su almacenamiento y tratamiento eficiente.

Inicialmente, las empresas usaban archivos en papel, pero con el aumento del volumen y complejidad de los datos, fue necesario el paso a sistemas informatizados. Así nació el concepto de fichero digital, donde los datos son organizados en estructuras similares a las que existían en papel.

Surge la necesidad de desarrollar métodos eficientes de almacenamiento y gestión de la información. Las bases de datos nacen para dar respuesta a estas necesidades, permitiendo el acceso concurrente, la integridad y eliminación de redundancias.

2. Almacenamiento de la información

El almacenamiento de datos se basa en dispositivos llamados **memorias**, que permiten **guardar y recuperar información** cuando el sistema lo requiere. Estas memorias están construidas con distintas tecnologías, cada una con características, ventajas e inconvenientes, por lo que **la elección adecuada de las memorias es fundamental en el diseño de un equipo de procesamiento de datos**.

Características comunes de los dispositivos de almacenamiento:

1. **Unidad de almacenamiento:** es la **mínima cantidad de información** que puede leerse o escribirse en el dispositivo.
   - Puede variar desde **1 byte** hasta **miles de bytes**, dependiendo de la tecnología y el tipo de memoria.
   - Ejemplo comparativo: si imaginamos la memoria como un archivador, la unidad de almacenamiento sería **el contenido de un solo cajón**.

2. **Tipo de acceso:** determina cómo se accede a la información almacenada:
   - **Acceso aleatorio:** se puede acceder directamente a una unidad de almacenamiento si se conoce su dirección (como sucede con la RAM).
   - **Acceso secuencial:** se accede a los datos **en orden**, uno tras otro, como ocurre con las cintas magnéticas.

Niveles de almacenamiento:

La memoria de un sistema se organiza en **dos niveles principales**:

- **Memoria primaria:** directamente accesible por el procesador (ej. RAM, ROM, caché). Es rápida, pero de menor capacidad.
- **Memoria secundaria:** no accesible directamente por el procesador (ej. discos duros, SSD, USB). Es más lenta, pero de gran capacidad.

2.1. Almacenamiento primario

El almacenamiento primario está compuesto por los dispositivos de memoria que son **directamente accesibles por el procesador**. Su función principal es **guardar temporalmente los datos y las instrucciones** que necesita el procesador para ejecutar tareas.

Componentes del almacenamiento primario:

- **RAM (Memoria de Acceso Aleatorio):** volátil y de alta velocidad. Almacena datos e instrucciones en uso.
- **ROM (Memoria de Solo Lectura) / Flash:** no es volátil. Contiene instrucciones permanentes, como el firmware del sistema.
- **Memoria caché:** una forma más rápida y pequeña de RAM que almacena datos de uso frecuente para acelerar el acceso.

Características:

- Utilizan **acceso aleatorio**, lo que permite leer o escribir datos directamente en cualquier ubicación sin necesidad de seguir un orden secuencial.
- Son **rápidas pero de menor capacidad** en comparación con el almacenamiento secundario.
- Toda la información que el procesador necesita debe estar previamente cargada en este tipo de memoria. Si los datos están en otro tipo de almacenamiento (como el disco duro), deben transferirse a la memoria primaria antes de poder ser utilizados.
- El tamaño de unidad de acceso se denomina **palabra**, y varía **entre 8 y 64 bits**, dependiendo de la arquitectura del procesador.

2.2. Almacenamiento secundario

El almacenamiento secundario incluye todos los dispositivos no accesibles directamente por el procesador. Su función es guardar grandes cantidades de datos de forma permanente o a largo plazo.

Características principales:

- **Mayor capacidad** que el almacenamiento primario.
- **Menor velocidad** de acceso.
- **Unidad básica de almacenamiento:** es un bloque o página, no palabras. Que suele tener un tamaño típico de entre 128 bytes y 4 KB.
- **Tipo de acceso:** suele ser aleatorio en discos duros, SSD, discos ópticos,...y secuencial e cintas magnéticas.

Tipos de dispositivos:

- Discos duros (HDD) – magnéticos.
- Discos ópticos – CD, DVD, Blu-ray.
- Cintas magnéticas – usadas en copias de seguridad masivas.
- Discos de estado sólido (SSD) – más rápidos que los HDD, sin partes móviles.

3. Ficheros

Los **ficheros** son el **elemento básico de almacenamiento permanente** en los dispositivos de almacenamiento secundario.

Estructura de un fichero:

- Está compuesto por **registros lógicos**, cada uno relacionado con un objeto o entidad.
- Cada registro se divide en **campos**, que contienen datos individuales.

Propiedades de un campo:

- **Nombre:** es el que identifica el campo de forma única dentro del registro.
- **Tipo:** define el tipo de dato que contiene como texto, número, fecha,...
- **Tamaño:** puede ser **fijo**, si ocupa siempre el mismo espacio o **variable** si depende del contenido, aunque con un límite máximo.

Registros físicos y bloques:

- Debido al gran tamaño de los ficheros, **solo una parte puede cargarse en memoria principal** para su procesamiento.
- La **unidad mínima de transferencia** entre el almacenamiento secundario y la memoria es el **registro físico o bloque**.
- Un **bloque** puede contener **varios registros lógicos**.
- El **factor de bloqueo** indica cuántos registros lógicos caben en un bloque.
- Este proceso se denomina **bloqueo de registros**.

4. Tipos de ficheros

4.1. Según su función

Ficheros Permanentes: son los que contienen información esencial y duradera para el funcionamiento de una aplicación. **No se generan fácilmente a partir de otros ficheros.**

- **Ficheros Maestros**
  - Reflejan el **estado actual** de los datos.
  - Se **actualizan periódicamente**, pero su estructura no cambia.
  - Ejemplo: datos de usuarios de una plataforma educativa.
- **Ficheros Constantes**
  - Contienen datos **fijos** o que cambian muy poco.
  - Son usados como **consulta**.
  - Ejemplo: fichero de códigos postales.
- **Ficheros Históricos**
  - Guardan datos que fueron actuales en el **pasado**.
  - Sirven para la **reconstrucción** de situaciones anteriores.
  - Ejemplo: usuarios dados de baja en una plataforma.

Ficheros Temporales: se crean para procesos específicos. **Tienen vida corta** y se usan para obtener resultados o actualizar ficheros permanentes.

- **Ficheros Intermedios**
  - Guardan resultados de una aplicación que serán usados por otra.
  - Ejemplo: selección de usuarios para modificar su rol.
- **Ficheros de Maniobras**
  - Evitan la pérdida de información cuando **falta espacio** en memoria principal.
  - Funcionan como una **extensión temporal** de la memoria.

- **Ficheros de Resultados**
  - Contienen los **resultados finales** de un proceso que se enviarán a un **dispositivo de salida**.
  - Ejemplo: fichero de impresión.

4.2. Según la longitud de registros

1. **Ficheros de Registros de Longitud Fija:**
   - **Todos los registros tienen la misma estructura y tamaño.**
   - Cada campo tiene una **longitud fija**.
   - **Ventajas:**
     - Acceso directo y rápido: para encontrar un registro basta con multiplicar su posición por el tamaño del registro.
     - Simples de manipular y gestionar.
   - **Desventajas:**
     - **Poco flexibles.**
     - **Desperdicio de espacio** si los campos no se usan completamente.
   - **Ejemplo:**
     - Si cada registro mide 108 bytes, el registro 1 empieza en el byte 1, el 2 en el 109, el 3 en el 217, etc.
     - Aunque algunos campos (como nombre o apellidos) estén parcialmente vacíos, ocupan el mismo espacio.

2. **Ficheros de Registros de Longitud Variable:**
   - Los registros pueden tener:
     - **Campos de longitud variable** o estructuras distintas.
     - **El tamaño de cada registro se guarda junto al contenido.**
   - **Ventajas:**
     - **Mayor eficiencia en el uso del espacio.**
     - Solo se almacena lo necesario por cada registro.

- **Desventajas:**
  - Más **difíciles de gestionar**: requiere lectura secuencial o uso de índices para localizar datos.
  - Insertar, modificar o borrar es **más complejo**.
  - Puede ser necesario actualizar longitudes manualmente.
- **Ejemplo:**
  - Si el nombre y apellidos varían en longitud, se guarda primero la longitud real de esos campos antes del dato.
  - Ahorro de espacio notable (ej.: 93 bytes en registros variables frente a 216 bytes en registros fijos para 2 personas), pero con **mayor coste de acceso**.

5. Organización de ficheros

5.1. Operaciones básicas

- **Consulta:** accede a los datos de un registro.
- **Inserción:** añade un nuevo registro al fichero.
- **Modificación:** cambia el contenido de uno o varios campos de un registro existente.
- **Borrado:** elimina un registro del fichero.

La organización del fichero se indica cómo están de preparados el soporte de los registros en el almacenamiento. Existen dos tipos de accesos:

- **Secuencial:** recorren los registros uno a uno hasta encontrar el buscado.
- **Directo:** se accede directamente al registro mediante una clave.

5.2. Tipos de organización

1. **Secuencial:**
   - **Almacenamiento**: registros uno tras otro, según el orden de inserción.
   - **Inserción**: simple, siempre al final.
   - **Consulta**: lenta; se recorre desde el principio (lectura secuencial).
   - **Borrado**: crea "huecos"; se soluciona con **compactación**.
   - **Modificación**: simple si longitud fija y compleja si longitud variable → se borra y se vuelve a insertar.
   - **Usos**: ficheros que se procesan completos, con pocas modificaciones.

2. **Secuencial encadenada:**
   - Cada registro contiene un **puntero** al siguiente.
   - **Ventajas:** los registros en cualquier dirección física y las inserciones ordenadas según clave sin necesidad de mover registros.
   - **Inserción:** añadir registro y actualizar puntero del anterior.
   - **Borrado:** actualizar puntero del registro anterior para saltar el eliminado.
   - **Consulta:** secuencial pero optimizada mediante punteros.
   - **Modificación:** si cambia tamaño/clave, insertar nuevo → borrar viejo.

3. **Secuencial indexada:** se añade una zona de índices que permite buscar más rápido en bloques.
   - Dos zonas:
     1. **Zona de registros:** dividida en bloques ordenados por clave.
     2. **Zona de índices:** contiene claves representativas y la dirección del primer registro de cada bloque.
   - **Consulta:** búsqueda primero en el índice → luego secuencial en el bloque.
   - **Ventaja:** la más rápida que la secuencial pura.
   - **Requiere:** Soporte de almacenamiento direccionable (como disco duro).

4. **Directa o aleatoria (hash):** la posición del registro se calcula con una función hash. Muy eficiente, pero puede haber colisiones.
   - La **posición física** del registro se determina por una función (hash, directa, o indexada).
   - **Ventajas**: Acceso **muy rápido** a registros concretos.
   - **Problemas**: Pueden ocurrir **colisiones** si dos claves dan la misma dirección.

6. Sistemas de bases de datos

Antiguamente los programas trabajaban directamente con **ficheros físicos**. Cada uno de los departamentos o equipos tenían sus propios ficheros. Por lo que con el paso del tiempo **aparecieron problemas**.

Los **principales problemas** de usos de los ficheros:

- **Separación y aislamiento de datos:** cada fichero contiene datos distintos, lo que dificulta el acceso global y la coherencia entre ellos.
- **Duplicación de datos:** los mismos datos en distintos ficheros son inconsistencias, errores y pérdida de espacio.
- **Dependencia de los datos respecto a las aplicaciones:** cualquier cambio en la estructura de los datos requiere modificar todos los programas relacionados.
- **Formatos incompatibles:** los ficheros dependen del lenguaje de programación usado, dificultando su integración.
- **Consultas rígidas:** solo se pueden hacer consultas predefinidas en los programas; no hay flexibilidad.
- **Falta de control de concurrencia:** el acceso simultáneo puede causar errores y datos inconsistentes.
- **Ausencia de catálogo general:** sin un esquema común, es difícil saber dónde está cada dato. Cada área almacena por separado.

6.1. Arquitectura (ANSI, 1975)

El modelo de **arquitectura de tres capas** propuesto por ANSI busca **separar los niveles de abstracción** en una base de datos, desde la forma en que los datos se almacenan físicamente hasta cómo los ven los usuarios finales.

- **Nivel interno (físico):** es el que especifica **cómo se almacenan físicamente los datos** en el sistema. Los detalles de **ficheros, discos, índices, directorios, métodos de acceso,...** Se define como el **esquema interno**.
- **Nivel conceptual (medio):** representa la **estructura lógica completa** de la base de datos para toda la organización. Oculta los detalles físicos del nivel interno. Trata de hablar sobre **entidades, relaciones, atributos y restricciones**. Se define como el **esquema conceptual**.
- **Nivel externo (vista del usuario):** muestra solo la parte de la base de datos que **interesa a cada usuario o grupo de usuarios**. Utiliza **esquemas externos** o **vistas** personalizadas. El usuario interactúa con este nivel a través de **aplicaciones**.

6.2. Modelos de datos

Dado que la **arquitectura de bases de datos** tiene tres niveles (externo, conceptual e interno), también existen **tres tipos de modelos** asociados a cada nivel.

6.3. Tipos de modelos

6.3.1. Modelos conceptuales

Representa los datos de forma cercana a cómo los entienden las personas, pero **formalizada**. Se utiliza para describir **datos existentes, sus características y cómo se relacionan entre sí**, sin preocuparte de cómo se almacenan o procesan.

Se caracterizan por ser muy flexibles, incluir datos, relaciones y restricciones. Además de **no depender** de ningún sistema de gestión ni de tecnología correcta. El modelo más utilizado es el **Modelo Entidad-Relación (MER)**.

6.3.2. Modelos lógicos

Se deriva del modelo conceptual, pero **adaptado a un sistema de gestión específico**. Se definen las **tablas, claves primarias y foráneas**, tipos de datos,... como por ejemplo el diagrama de tablas en un SGBD relacional.

- **Modelo jerárquico:** tiene una **estructura de árbol**, relaciones solo de uno a muchos. Es **rápido y fácil de navegar** en estructuras simples. Sin embargo es **muy rígido y no permite relaciones complejas**. Además se necesita **duplicar la información** para representar las relaciones más flexibles en problemas de consistencia.
- **Modelo de red:** tiene una **estructura de grafo**, relaciones de muchos a muchos. Es **más flexible que el modelo jerárquico y tiene mejor consistencia** en estructuras simples. Sin embargo es **difícil de administrar y mantener**.
- **Modelo relacional:** tiene una de **tablas bidimensionales**, se definen mediante claves primarias y foráneas. Es **muy flexible y fácil de usar con SQL**. Sin embargo puede **volverse ineficiente con datos complejos si no se gestiona bien**.
- **Modelo orientado a objetos:** basado en **clases y objetos**. Permite **almacenar el comportamiento** junto con los datos, soportando la encapsulación, herencia,... Sin embargo desde menor estandarización que en el relacional y la **mayor complejidad de implementación y mantenimiento**.

6.3.3. Modelos físicos

Representa **cómo se almacenan realmente los datos** en el sistema. Detallando la estructura de **ficheros, índices, particiones, bloques,...** además depende directamente del motor de base de datos.

7. Sistemas Gestores de Bases de Datos (SGBD o DBMS)

Un **SGBD** es un **conjunto de programas** que permite **crear, definir, mantener y acceder** a bases de datos. Actúa como **intermediario entre los usuarios (o aplicaciones) y la base de datos física**.

Usuarios:

- Son quienes consumen o producen información.
- No siempre son personas: pueden ser **otros sistemas automáticos** como sensores, cámaras,...

Aplicaciones:

- Proveen una **interfaz amigable** para el usuario como formularios, informes, ventanas,...
- **Ocultan la complejidad** del acceso a los datos.

Comunicación con el SGBD:

- Las aplicaciones se conectan al **SGBD** para poder consultar datos tanto como insertar, modificar o eliminar registros.

Base de datos física:

- El **SGBD gestiona directamente** el almacenamiento físico.
- Se encarga de guardar, organizar y recuperar los datos cuando se necesiten.

7.1. Funciones

- **Gestión del diccionario de datos (catálogo):** almacena **metadatos**, datos sobre los datos. Contiene nombre, tipo y tamaño de cada dato, relaciones entre los datos, reglas de integridad, permisos de usuario sobre los datos y estadísticas de uso y acceso.
- **Garantiza la integridad transaccional:** una transacción es un conjunto de **operaciones que deben de completarse** en completa totalidad o no hacerse. Si algo falla, **se deshacen todos los cambios**. Se usa el **registro de logs** para llevar a cabo un control.
- **Gestión del acceso concurrente:** controla el **acceso simultáneo de varios usuarios** a los mismos datos. De esta forma, evita inconsistencias mediante **técnicas de bloqueo de locks**.
- **Recuperación ante fallos:** permite **recuperar la base de datos a un estado consistente** tras un error. Al igual, se intenta **minimizar la pérdida de datos**.
- **Proporcionar interfaces de uso:** facilita la conexión de aplicaciones o usuarios a través de acceso por red, acceso local, memoria compartida...
- **Gestión de restricciones sobre los datos:** control de que los datos cumplan las **reglas de integridad**. Se impide interrumpan las modificaciones.
- **Herramientas de administración:** proporciona herramientas para la gestión de usuarios, monitorización, copias de seguridad,... Algunas de ellas sin del fabricante y otras de terceros.

7.2. Componentes

1. **Lenguaje de datos**
   Tres tipos principales, según su función:
   - **DDL (Data Definition Language):** define la **estructura** de la base de datos (tablas, relaciones, restricciones...).
   - **DCL (Data Control Language):** controla el **acceso y seguridad** de los datos (usuarios, privilegios...).
   - **DML (Data Manipulation Language):** permite **consultar, insertar, modificar y eliminar** datos.

2. **Diccionario de datos**
   - Base de datos interna con **metadatos**: información sobre estructuras, relaciones y permisos.
   - Gestionado automáticamente por el SGBD.

3. **Objetos de base de datos**
   - Varían según el SGBD, pero suelen incluir: tablas y vistas, consultas, tipos de datos y dominios, restricciones y aserciones, funciones y procedimientos almacenado y disparadores (triggers).

4. **Herramientas administrativas**
   - Permiten **gestionar**: la seguridad, integridad, acceso concurrente, migraciones,...

5. **Optimizador de consultas**
   - Convierte las **consultas DML en operaciones físicas** eficientes (lectura, indexado...). Y mejora el rendimiento automáticamente.

6. **Gestor de transacciones**
   - Administra las transacciones y el acceso concurrente.
   - Garantiza la consistencia y aislamiento de las operaciones.

7. **Planificador**
   - Ejecuta tareas automáticas programadas o en respuesta a eventos (como copias o limpieza de datos).

8. **Gestión de replicación**
   Facilita crear copias de seguridad o réplicas de la base de datos en otros servidores o sistemas.

7.3. Usuarios

Los tipos de usuarios en una base de datos son los siguientes:

- **Administradores:**
  - **Responsables del mantenimiento total** del sistema.
  - Tareas principales:
    - Instalación y configuración.
    - Gestión de copias de seguridad y recuperación.
    - Asignación y supervisión del almacenamiento.
    - Seguridad del control de accesos y permisos.
  - **Tienen el mayor nivel de control y privilegios.**

- **Diseñadores de la base de datos:**
  - Diseñan el **modelo lógico** y las **estructuras de datos**.
  - Se encargan de **definir**:
    - Entidades, atributos y relaciones.
    - Reglas de integridad y restricciones.
  - Deben conocer bien los **procesos de la organización** (reglas de negocio).
  - **Involucran a los usuarios** desde el inicio para crear un diseño útil y coherente.

- **Programadores de aplicaciones:**
  - Desarrollan **software que interactúa con la base de datos**.

  - Permiten a los **usuarios finales**:
    - Consultar datos.
    - Insertar, actualizar y eliminar registros.
  - Trabajan con **interfaz de usuario** (formularios, menús, reportes, etc.).

- **Usuarios finales:**
  - Utilizan las **vistas** del nivel externo.
  - Interactúan con la base de datos a través de **aplicaciones**.
  - No necesitan **conocimientos técnicos** sobre:
    - Estructura interna.
    - Lenguajes de datos.
    - Ubicación de la información.
  - Son los **consumidores principales** de la información.

7.4. Tipos de SGBD

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

- **Número de usuarios**
  - Monousuario
  - Multiusuario
- **Modelo lógico**
  - Jerárquico
  - En red
  - Relacional
  - Objeto-relacional
  - Orientado a objetos
- **Número de sitios**
  - Centralizados: en un solo servidor o equipo.
  - Distribuidos: en varios equipos que pueden ser homogéneos y heterogéneos.

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

7.4.1. SGBD Comerciales

SGBD	Características
ORACLE	Muy potente, seguro y confiable. Multiplataforma. Alto coste. Usado en grandes empresas. Tiene versión gratuita limitada (Oracle XE).
MySQL	Muy rápido y extendido. Licencia libre o comercial. Ideal para web. Multiplataforma, multihilo, multiusuario.
DB2	Relacional y multiplataforma. Soporta XML nativamente. Útil para búsquedas jerárquicas y relacionales.

SQL Server	Relacional, cliente/servidor. Solo para Windows. Alternativa a Oracle o MySQL. De Microsoft.
Sybase	Relacional, escalable y de alto rendimiento. Soporta grandes volúmenes de datos. Bajo coste. Tres versiones según necesidad.

7.4.2. SGBD Libres

SGBD	Características
MySQL	Relacional, multihilo y multiusuario. Muy usado en aplicaciones web. Multiplataforma. Accesible desde muchos lenguajes. Licencia libre y comercial.
PostgreSQL	Relacional orientado a objetos. Muy avanzado y estable. Código abierto, comunidad activa. Multiplataforma y multilenguaje.
Firebird	Relacional, multiplataforma, bajo consumo de recursos. Alta concurrencia, buen rendimiento y soporte para varios lenguajes.
Apache Derby	Ligero y escrito en Java. Multiplataforma, portátil. Funciona como embebido o cliente/servidor. Soporta varios lenguajes.
SQLite	Relacional y muy ligero. Biblioteca en C. Muy rápido y eficiente. Multiplataforma. Ideal para apps móviles y embebidas.
MariaDB	Fork de MySQL con mejoras. Muy popular en aplicaciones web. Totalmente libre y compatible con MySQL.