

Sistemas gestores de bases de datos. Funciones. Componentes. Arquitecturas de referencia y operacionales. Tipos de sistemas.

TEMA 34 (36 SAI)

ABACUS NT

Oposiciones 2021

Índice

- 1. Introducción**
- 2. Sistemas Gestores de Bases de Datos (SGBD)**
 - 2.1. Concepto**
 - 2.2. Historia de los SGDB**
- 3. Funciones del SGBD**
 - 3.1. Tareas del SGBD**
 - 3.2. Requisitos de un SGBD**
 - 3.3. Abstracción de la información**
- 4. Componentes de un SGBD**
 - 4.1. Lenguajes para la definición y manipulación de datos**
 - 4.1.1. Lenguajes para la definición de datos**
 - 4.1.2. Lenguajes para la manipulación de datos**
 - 4.2. Procesador de consultas.**
 - 4.3. Gestor de la base de datos.**
 - 4.4. Utilidades**
 - 4.5. Componentes físicos**
 - 4.6. Usuarios de la base de datos**
- 5. Tipos de bases de datos**
 - 5.1. Bases de datos jerárquicas**
 - 5.2. Bases de datos en red**
 - 5.3. Bases de datos relacionales**
 - 5.1. Sistemas NoSQL.**
 - 5.2. Sistemas en la Nube.**
- 6. Arquitecturas de referencia**
 - 6.1. Niveles de abstracción**
 - 6.2. Nivel interno (estructura física).**
 - 6.3. Nivel conceptual (estructura lógica global)**
 - 6.4. Nivel externo (estructura lógica de usuario)**
 - 6.5. Interfaces**
- 7. Arquitecturas operacionales.**
 - 7.1. Estructura cliente-servidor.**
 - 7.2. Sistemas distribuidos.**
- 8. Conclusión**
 - 8.1. Relación del tema con el sistema educativo actual**
- 9. Bibliografía**

1. Introducción

Las bases de datos surgen con la necesidad del almacenamiento de grandes volúmenes de datos. Los pioneros en Ciencias de la Computación de finales de los 50 y principios de los 60 desarrollaron muchas técnicas nuevas para manipular los ficheros. Los ficheros se pueden manipular mediante lenguajes de tercera generación convencionales empleando los servicios del sistema operativo.

La complejidad del trabajo de programación y la rigidez de las restricciones que imponía la tecnología de los ordenadores en aspectos como la inserción de datos o la gestión de casos especiales, hacía arduo el trabajo.

En una aplicación convencional con ficheros, éstos se diseñan de acuerdo con los programas. Esto es, una vez planteado el programa, se decide si debe haber ficheros, cuántos deben ser, qué organización tendrán, qué información contendrá cada uno, qué programas actuarán sobre ellos y cómo lo harán.

Esto tiene la ventaja, en principio, de que los programas son bastante eficientes, ya que la estructura de un fichero está pensada para el programa que lo va a usar; sin embargo, conlleva graves inconvenientes. En una aplicación convencional con ficheros aparecen, pues, los siguientes problemas:

- **Dificultad de mantenimiento.** Si hay ficheros con información duplicada, realizar las actualizaciones necesarias puede ser complejo y costoso. Incluso puede ser necesario utilizar archivos con diferentes organizaciones. Si la actualización no se realiza correctamente en todos los lugares se producirán inconsistencias en la base de datos.
- **Redundancia.** Este problema consiste en tener datos que no aportan información, ya que se pueden deducir de otros datos. El caso típico de redundancia es mantener el mismo dato almacenado en varios sitios.
- **Rigidez de búsqueda.** A cada fichero se le dará una determinada organización, de acuerdo con los tipos de acceso que se creían más frecuentes, pero puede ocurrir que se necesiten otros modos de acceso y sean difíciles de llevar a cabo sobre la organización ya existente.
- **Dependencia con los programas.** Las relaciones entre los datos almacenados en los ficheros no están presentes en éstos. Es el programa que los utiliza el que se encarga de ello. Esto implica que cualquier modificación de la estructura de un fichero obliga a modificar todos los programas que lo usen.
- **Confidencialidad y seguridad.** En general, es necesario impedir la consulta de determinados datos a determinados usuarios (confidencialidad) e impedir modificaciones efectuadas por usuarios no autorizados (seguridad). Si trabajamos convencionalmente con ficheros, el control deberá realizarlo el propio programa. En cualquier caso, no se podrá impedir que alguien construya un programa para modificar o ver el contenido de un fichero, si el sistema operativo le permite el acceso.

Además, en un sistema tradicional de ficheros, cada una de las aplicaciones deberá realizar la descripción de los registros, así como determinar la organización de ficheros, los tipos de acceso,

etc. En un entorno de base de datos dichas especificaciones las realiza el SGBD, y es el administrador de la base de datos el encargado de realizar éstas y otras funciones sobre el sistema.

2. Sistemas Gestores de Bases de Datos (SGBD)

2.1. Concepto

Según la Real Academia de la Lengua, una BD es la *representación de una información de manera adecuada para su tratamiento por un ordenador*.

Además, es necesario distinguir entre la base de datos BD y el sistema que la gestiona (SGBD ó DBMS – Data Base Management System–)

Así, un SGBD es una colección de programas de aplicación que proporcionan al usuario de la base de datos los medios necesarios para realizar las siguientes tareas:

1. Definición de los datos a los distintos niveles de abstracción (físico, lógico y externo).
2. Manipulación de los datos en la base de datos. Es decir, la inserción, modificación y borrado y acceso o consulta a los mismos.
3. Mantenimiento de la integridad de la BD. Integridad en cuanto a los datos en sí, sus valores y las relaciones entre ellos.
4. Control de la privacidad y seguridad de los datos en la BD.

Se denomina **sistema de gestión de base de datos** al conjunto de software destinado a la creación, control y manipulación de la información sobre una base de datos.

2.2. Historia de los SGDB

Primera generación de los SGBD:

Los sistemas jerárquicos y de red constituyen la primera generación de los SGBD. El modelo jerárquico estructura los datos de manera escalonada, existiendo una relación del tipo padre-hijo entre sus registros. La representación de la información es similar a una estructura en árbol. El modelo en red, elimina la restricción del modelo jerárquico, en el sentido de que un registro puede tener más de un parente, por lo que resulta más flexible, dando lugar a una estructura de grafo. Pero estos sistemas presentan algunos inconvenientes:

Es necesario escribir complejos programas de aplicación para responder a cualquier tipo de consulta de datos, por simple que ésta sea.

La independencia de datos es mínima. No tienen un fundamento teórico.

Segunda generación de los SGBD:

En 1970 Codd, de los laboratorios de investigación de IBM, escribió un artículo presentando el modelo relacional. En este artículo, presentaba también los inconvenientes de los sistemas previos, el jerárquico y el de red. Entonces, se comenzaron a desarrollar muchos sistemas relationales,

apareciendo los primeros a finales de los setenta y principios de los ochenta. Esto condujo a dos grandes desarrollos: El desarrollo de un lenguaje de consultas estructurado denominado SQL, que se ha convertido en el lenguaje estándar de los sistemas relacionales.

La producción de varios SGBD relacionales durante los años ochenta, como DB2 y SLQ/DS de IBM, y ORACLE de ORACLE Corporation.

Hoy en día, existen cientos de SGBD relacionales, tanto para microordenadores como para sistemas multiusuario, aunque muchos no son completamente fieles al modelo relacional. Los SGBD relacionales constituyen la segunda generación de los SGBD.

Tercera generación de los SGBD:

Como respuesta a la creciente complejidad de las aplicaciones que requieren bases de datos, han surgido nuevos modelos de datos y SGBD que los soportan: el modelo de datos objeto-relacional y el modelo relacional extendido. Sin embargo, a diferencia de los modelos que los preceden, la composición de estos modelos no está clara.

Cuarta generación de los SGBD:

aparecen los sistemas no relacionales como son las bases de datos NoSQL, las bases de datos documentales y los sistemas en la nube (cloud).

3. Funciones del SGBD

3.1. Tareas del SGBD

Concretamente, un SGBD debe permitir la realización de las siguientes tareas:

- **Definición del esquema de la base de datos.** Una vez diseñado el esquema de la base de datos, hemos de describirlo mediante un conjunto de instrucciones. Esto se realiza mediante un lenguaje específico, denominado **lenguaje de definición de datos (DDL)**.
- **Acceso a los datos desde un lenguaje de alto nivel.** Esto se realiza también mediante un lenguaje específico, denominado **lenguaje de manipulación de datos (DML)**. El DML puede ser utilizado de dos formas diferentes. Por una parte, se incluyen sentencias DML en programas escritos en lenguaje de alto nivel. Esto hace necesario incluir en el SGBD un precompilador que traduzca las instrucciones DML en instrucciones reconocibles por el compilador del lenguaje de alto nivel (lenguaje anfitrión). La otra forma de utilización del DML es mediante programas que contengan exclusivamente sentencias propias de este lenguaje.
- **Acceso a la información en modo conversacional.** El SGBD debe incorporar una interfaz de usuario a través de la cual introducir sentencias directamente desde un terminal, para obtener información interactiva.
- **Gestión de ficheros.** Función realizada por un módulo gestor de ficheros que se encarga de la comunicación con el sistema operativo. Además, realiza otras funciones, tales como

control de usuarios, recuperar la información tras fallos del sistema, organización física de la base de datos, control de seguridad y privacidad, y gestión de accesos concurrentes.

3.2. Requisitos de un SGBD

Los requisitos que debe cumplir un buen sistema de base de datos son:

- **Acceso múltiple.** Varios usuarios pueden acceder simultáneamente a la base de datos, sin que se produzcan conflictos ni vistas incoherentes.
- **Utilización múltiple.** Cada usuario podrá tener una imagen o visión particular de la estructura de la base de datos.
- **Flexibilidad.** Se podrán utilizar distintos métodos de acceso.
- **Confidencialidad y seguridad.** Se controlará el acceso a los datos, impidiéndoselo a los usuarios no autorizados.
- **Protección contra fallos.** Deben existir mecanismos concretos de recuperación en caso de fallo del ordenador.
- **Independencia física.** Se puede cambiar el soporte físico de la base de datos sin que repercuta en la base de datos ni en los programas que la usan.
- **Independencia lógica.** Se pueden modificar los datos contenidos en la base, las relaciones existentes entre ellos o incluir nuevos datos, sin que afecte a los programas que la usan.
- **Redundancia controlada.** Los datos se almacenan una sola vez, excepto en casos excepcionales.
- **Interfaz de alto nivel.** Existe una forma sencilla y cómoda de utilizar la base de datos desde un lenguaje de programación de alto nivel.
- **Interrogación directa (query).** Existe una utilidad que permite el acceso a los datos de forma conversacional.

3.3. Abstracción de la información

El SGBD debe proporcionar información a usuarios y desarrolladores a distintos niveles, representando cada uno de ellos una abstracción de datos:

- **Nivel de visión.** A este nivel, cada grupo de usuarios posee conocimiento únicamente de aquella parte de la base de datos que le afecta. El usuario sabe de la existencia de los datos y su significado, pero ignora los detalles sobre su formato, tipo, estructura, y, en general, cualquier aspecto físico.
- **Nivel conceptual.** La unión de todas las vistas da lugar a este nivel. En él se conoce la descripción de todos los datos y las relaciones existentes entre ellos.
- **Nivel físico.** En este nivel se describe cómo se encuentran los datos almacenados físicamente en memoria secundaria. Es el nivel más cercano al hardware y se encuentra íntimamente ligado a él.

4. Componentes de un SGBD

Un sistema gestor de bases de datos se divide en módulos que se encargan de las responsabilidades del sistema. Algunas de estas funciones las proporciona el sistema operativo.

Los componentes funcionales principales de un sistema gestor de base de datos son el procesador de consultas y el gestor de la base de datos. Además, serán necesario los soportes y estructuras para su almacenamiento físico.

La arquitectura completa está dividida en dos fases:

Definición de datos. La creación de la base de datos comienza con la elaboración del esquema conceptual, que se procesa utilizando una herramienta CASE que lo convierte en los metadatos. Utilizando esta información, se construyen los esquemas interno y externo.

Manipulación de datos. El usuario puede realizar operaciones sobre la base de datos. Esta petición es transformada por el transformador externo/conceptual que obtiene el esquema correspondiente ayudándose también de los metadatos. El resultado lo convierte otro transformador en el esquema interno usando también la información de los metadatos. Finalmente, del esquema interno se pasa a los datos usando el transformador que también accede a los metadatos y de ahí se accede a los datos. Para que los datos se devuelvan al usuario en formato adecuado para él se tiene que hacer el proceso contrario.

4.1. Lenguajes para la definición y manipulación de datos

Según el autor **C.J Date**, en su libro *Introduction to Database Systems* (2003), “una base de datos es un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada” (entendiéndose como empresa una organización).

Para trabajar con dicho conjunto de datos, la arquitectura ANSI/X3/SPARC diferencia dos tipos de lenguajes:

- **Lenguaje de definición de datos:**
- **Lenguaje de manipulación de datos**

A continuación, vamos a ver en qué consiste cada uno de ellos

4.1.1. Lenguajes para la definición de datos

El lenguaje de descripción o definición de datos (DDL, Data Definition Language), propio de cada SGBD, permite al desarrollador de la base de datos especificar los elementos de datos que integran su estructura, y las relaciones que existen entre ellos, las reglas de integridad semántica, las características de tipo físico y las vistas lógicas de los usuarios.

Generalmente, los DDL de los diferentes SGBD son lenguajes muy simples basados en una gramática muy sencilla.

La representación de los datos compilación es almacenada en denominado **Diccionario de Datos**.

el DDL cuenta con un **sublenguaje** encargado del control y seguridad de los datos, el cual se denomina **Lenguaje de Control de Datos (DCL)** y permite el control del acceso de a la información almacenada en el diccionario de datos (definición de privilegios y tipos de acceso), así como el control de seguridad de los datos.

4.1.2. Lenguajes para la manipulación de datos

Una vez descrita la estructura de la base de datos, los usuarios utilizarán un lenguaje de manipulación de datos (DML, Data Manipulation Language) para realizar las siguientes operaciones sobre dicha estructura:

- **Recuperar información:** Consultar la base de datos.
- **Actualizar la información:** La actualización supondrá tres tipos de operaciones
 - Inserción
 - Borrado
 - Modificación de los datos.

Al igual que el DDL, el DML está basado en un modelo de datos y, por tanto, los SGBD basados en distintos modelos de datos tienen diferente DML. Se trata también de un lenguaje basado en una gramática completa, sencilla y, generalmente, fácil de entender por usuarios no expertos.

Los SGBD también son capaces de procesar peticiones de DML que se han formulado desde programas escritos en otros lenguajes de programación

Dentro de los **DML**, podemos destacar los siguientes ejemplos comerciales:

- **SQL (Structured Query Language):** Lo estudiaremos en el siguiente apartado.
- **QBE (Query By Example):** Creado a principios de los años 70. Hay varias **implementaciones** de este lenguaje, como el original de IBM (Sistema QBE), o QBE de Microsoft (en Access). Se fundamenta en el cálculo relacional orientado a dominios.
- **QUEL (Query Language):** Basado en el cálculo relacional orientado a tuplas.

4.2. Procesador de consultas.

Es el componente principal, transforma las consultas en un conjunto de instrucciones de bajo nivel que se dirigen al gestor de la base de datos:

- **Compilador de DML** (Data Manipulation Language). Traduce las instrucciones de DML a instrucciones de bajo nivel que entiende el motor de evaluación de consultas. Además, intenta transformar las peticiones del usuario en otras equivalentes, pero más eficientes, encontrando así una buena estrategia para ejecutar la consulta.

- **Precompilador de DML embebido.** Convierte las instrucciones de DML embebidas en un programa de aplicación en llamadas a procedimientos normales en el lenguaje anfitrión. Interactúa con el compilador de DML para generar el código apropiado.
- **Intérprete de DDL (Data Definition Language).** Interpreta las instrucciones de DDL y las registra en un conjunto de tablas que contienen metadatos (catálogo).
- **Motor de evaluación de consultas.** Ejecuta las instrucciones de bajo nivel generadas por el compilador de DML.

4.3. Gestor de la base de datos.

Proporciona la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación. Acepta consultas y examina los esquemas externo y conceptual para determinar qué registros se requieren para satisfacer la petición.

Entonces realiza una llamada al gestor de ficheros para ejecutar la petición:

- **Gestor de autorización e integridad.** Comprueba que se satisfagan las restricciones de integridad y la autorización de los usuarios para acceder a los datos.
- **Gestor de transacciones.** Asegura que la base de datos quede en un estado consistente a pesar de los fallos del sistema, y que las ejecuciones de transacciones concurrentes ocurran sin conflictos.
- **Gestor de ficheros.** Maneja los ficheros en disco en donde se almacena la base de datos utilizando los métodos de acceso del sistema operativo que se encargan de leer o escribir los datos en el buffer del sistema. Este gestor establece y mantiene la lista de estructuras e índices definidos en el esquema interno.
- **Gestor de buffers.** Es responsable de traer los datos del disco de almacenamiento a memoria principal, y decidir qué datos tratar en la memoria caché.
- **Gestor del diccionario de datos.** Controla los accesos al diccionario de datos y se encarga de mantenerlo.
- **Gestor de recuperación.** Este módulo garantiza que la base de datos permanece en un estado consistente en caso de que se produzca algún fallo.
- **Planificador.** Este módulo es el responsable de asegurar que las operaciones que se realizan concurrentemente sobre la base de datos tienen lugar sin conflictos.
- **Optimizador de consultas.** Este módulo determina la estrategia óptima para la ejecución de las consultas.

4.4. Utilidades

Son aplicaciones que facilitan el trabajo a los usuarios y programadores. Tienen la característica común de tener un interfaz fácil de entender. Se basan en menús que guían al usuario para conseguir el objetivo final.

La mayoría de los SGBD incluyen:

- **Generador de menús:** Diseña la interfaz de usuario de una aplicación.

- **Generador de Informes:** Presenta datos en pantalla o impresora con un formato predefinido o fácil de definir sin conocer lenguajes de BD o programación.
- **Generador de Formularios:** Genera pantalla de diálogos que permiten la introducción sencilla de información; bien por teclado, bien por botones.

4.5. Componentes físicos

Los componentes físicos de un sistema gestor de base de datos son los siguientes:

- **Ficheros de datos.** Almacenan la base de datos en sí.
- **Diccionario de datos.** Almacena metadatos acerca de la estructura de la base de datos.
- **Índices.** Proporcionan acceso rápido a elementos de datos que tienen valores particulares.
- **Datos estadísticos.** Almacenan información estadística sobre los datos en la base de datos. El procesador de consultas usa esta información para ejecutar una consulta.

4.6. Usuarios de la base de datos

Se consideran tres clases de usuarios que manipulan un base de datos:

- **Programador de aplicaciones.** Encargado de escribir programas de aplicación que utilicen las bases de datos. Estos programas se escriben en algún lenguaje de programación de alto nivel, y están diseñados para apoyar a un usuario final.
- **Usuario final.** Accede a la base de datos desde un terminal empleando un lenguaje de consulta (DML) o a través de un programa de aplicación.
- **Administrador de la base de datos.** Es la persona (o grupo de personas) encargada del control general del sistema de base de datos. Entre sus responsabilidades se incluyen las siguientes:
 - Decidir el contenido de la información de la base de datos.
 - Decidir la estructura de almacenamiento y la estrategia de acceso.
 - Vincularse con el resto de usuarios de la base de datos.
 - Definir los controles de autorización y procedimientos de validación.
 - Definir una estrategia de respaldo y recuperación tras posibles fallos del sistema.
 - Controlar el rendimiento y utilización de la base de datos.
 - Responder a los cambios de requerimientos.

5. Tipos de bases de datos

Tradicionalmente, las bases de datos se clasifican en tres grupos: **jerárquicas, en red y relacionales**. Las bases de datos jerárquicas fueron las primeras en aparecer. La información se representa en forma de árbol. El problema con este tipo de estructura es que no todas las bases de datos se adaptaban a la **estructura en árbol**. En un intento de eliminar la rigidez de las bases de datos jerárquicas, se desarrolló la estructura en red, en la cual se permitían todo tipo de relaciones. Cualquier conjunto de información es representable mediante una base de datos en red. Por último,

aparecieron las bases de datos relacionales, que pretendían obtener una mayor flexibilidad y rigor en el tratamiento de datos. Nacieron en los años 70 de la mano de E. F. Codd, quién planteó una alternativa a las bases de datos jerárquicas y en red existentes hasta el momento.

5.1. Bases de datos jerárquicas

Sólo se pueden crear estructuras jerárquicas en forma de árbol. Están formadas por una colección de registros unidos a través de relaciones que pueden ser de uno a uno o de uno a muchos, no siendo posible definir relaciones de muchos a muchos. Las relaciones se implementan físicamente utilizando punteros (igual que una estructura en árbol).

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

5.2. Bases de datos en red

Son muy parecidas a las jerárquicas. Se permite cualquier tipo de relaciones, aunque, en principio, se distingue entre base de datos en red simple (no permite relaciones muchos a muchos y es la más usual) y base de datos en red compleja. Cualquier sistema es representable en una base de datos en red.

Tomemos, por ejemplo, la relación Profesor-Alumno, que es una relación de muchos a muchos. Dicha relación se puede incluir en una base de datos en red compleja, pero, en principio, no en una base de datos en red simple. Para lograrlo, en el caso simple, se debe expresar la relación muchos a muchos mediante dos relaciones uno a muchos. Para ello, se introduce un nuevo registro, del que existirá una ocurrencia por cada par Profesor-Alumno que estén relacionados. A estos registros que introducimos se les llama enlaces.

5.3. Bases de datos relacionales

Una base de datos relacional está formada por **tablas**. Una tabla es una estructura bidimensional formada por una sucesión de registros del mismo tipo. Si se imponen ciertas restricciones a las tablas, se pueden tratar como relaciones matemáticas. De ahí el nombre de este tipo de base de datos y el hecho de que a las tablas de una base de datos relacional se les llame relaciones.

El modelo relacional es uno de los más utilizados en el diseño y gestión de bases de datos, debido fundamentalmente a dos razones: por una parte, se basa en un reducido número de conceptos, que hacen de este sistema uno de los más fáciles de entender, diseñar, cambiar, administrar y acceder, y, por otra parte, posee un lenguaje de definición y manipulación de datos muy potente y flexible.

5.1. Sistemas NoSQL.

Típicamente las bases de datos relacionales modernas han mostrado poca eficiencia en determinadas aplicaciones que usan los datos de forma intensiva, incluyendo el indexado de un gran número de documentos, la presentación de páginas en sitios que tienen gran tráfico, y en sitios de streaming audiovisual. Las implementaciones típicas de SGBDR se han afinado o bien para una

cantidad pequeña pero frecuente de lecturas y escrituras o para un gran conjunto de transacciones que tiene pocos accesos de escritura. Por otro lado, NoSQL puede servir gran cantidad de carga de lecturas y escrituras.

5.2. Sistemas en la Nube.

Una base de datos en la nube es una colección de contenido, estructurado o no estructurado, que reside en una plataforma de infraestructura de computación en la nube privada, pública o híbrida.

Existen dos modelos de entorno de base de datos en nube: tradicional y base de datos como servicio (DBaaS).

6. Arquitecturas de referencia

6.1. Niveles de abstracción

Un **SGBD** es un conjunto de procedimientos, ayudas de documentación, lenguajes y programas de software que administran los ficheros de la base de datos.

Uno de los *objetivos* de un SGBD es proporcionar a los usuarios una visión abstracta de la información, es decir, ocultar ciertos detalles referentes a la forma en que los datos se almacenan y mantienen, pero siempre permitiendo una recuperación eficaz de la información.

La independencia de datos es la capacidad para modificar el esquema de un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior. Esto significa:

Independencia física de los datos. Aunque el nivel físico cambie, el nivel conceptual no debe verse afectado. En la práctica esto significa que, aunque se añadan o cambien discos u otro hardware, o se modifique el sistema operativo u otros cambios relacionados con la física de la base de datos, el nivel conceptual permanece invariable.

Independencia lógica de los datos. Significa que, aunque se modifique el esquema conceptual, los esquemas externos y los programas de aplicación no serán afectados.

En 1975 el comité ANSI/SPARC propuso una arquitectura cuyo objetivo es el de separar los programas de aplicación de la base de datos física, que en su división X3 establece: “la arquitectura de una base de datos debe poseer tres niveles: **Interno, Conceptual y Externo**”. Cada uno de ellos pertenece a un tipo de vista diferente: **almacenamiento físico, del usuario, y del programador**.

6.2. Nivel interno (estructura física).

Es el nivel más bajo de abstracción de la información. Es la representación inferior de una base de datos, por ello, es el más cercano al almacenamiento físico. Permite describir los datos tal como están almacenados en el ordenador. Por ejemplo:

- Los ficheros que contienen (nombre, abstracción, ubicación)

- Los registros de estos ficheros (longitud, campos, ...)
- Las rutas de acceso a esos registros (índices, encadenamientos, ...)

El nivel interno es descrito por el SGBD por medio del **esquema interno** o **vista interna**.

Hay que recordar que para un programador un registro interno almacenado difiere de un registro lógico. Un registro físico puede estar formado por uno o más registros lógicos junto a elementos descriptivos del sistema: longitud y tipo de registro, banderas y punteros, y está almacenado en un dispositivo electrónico codificado como una combinación de ceros y unos.

La operación de transformar registros lógicos en físicos se denomina *transformación de datos o mapeo*.

6.3. Nivel conceptual (estructura lógica global)

Es el siguiente nivel más alto de abstracción, el administrador de la base de datos del SGBD define el nivel conceptual por medio de un **esquema** o **vista conceptual**, al decidir qué información se guarda en la base de datos.

Este nivel corresponde a la estructura organizacional de los datos de la base, obtenida al reunir los requerimientos de todos los usuarios de una empresa, sin preocuparse de su organización física ni de las vías de acceso.

El esquema conceptual podría contener:

- Los datos elementales que definen los campos, **atributos** de los elementos de una empresa (NIF, nombre, concepto, ...).
- Los datos compuestos que permiten reagrupar los **campos** para describir los registros, las entidades del mundo real (persona, artículo, ...).
- Los datos compuestos que permiten reagrupar los campos para describir las **asociaciones** del mundo real (compras de artículos, propietarios de viviendas, ...).

6.4. Nivel externo (estructura lógica de usuario)

Es el nivel más alto de abstracción, y, por ello, el más cercano a los usuarios. El nivel externo representa la percepción individual de cada usuario o programador de la base de datos.

El SGBD es el encargado de extraer los datos requeridos por los registros lógicos externos de uno o más registros físicos de la base de datos, cada vez que se ejecuta una operación de entrada/salida en un programa específico.

Por cada programa es necesario especificar un **esquema externo**, **subesquema** o **vista externa** para poder acceder de forma selectiva a un subconjunto de datos de la base integrada. Habrá usuarios que puedan acceder a más de un esquema externo, y un esquema externo puede ser compartido por varios usuarios; se protege, de esta manera, el acceso no autorizado a datos y el de usuarios mal intencionados.

6.5. Interfaces

Entre estos niveles existen unos interfaces que realizan funciones de traducción:

Función de traducción externo/conceptual. Define la correspondencia entre cada una de las vistas externas y la única vista conceptual (diferentes tipos de datos, diferentes nombres de campos, múltiples registros conceptuales fundidos en un único registro externo).

Función de traducción conceptual/interno. Establece cómo se almacena a nivel interno los registros y campos conceptuales. Si se modifica el almacenamiento de los datos, sólo es necesario modificar la aplicación de correspondencia, y no la vista conceptual.

7. Arquitecturas operacionales.

7.1. Estructura cliente-servidor.

Estructura clásica, la base de datos y su SGBD están en un servidor al cual acceden los clientes. El cliente posee software que permite al usuario enviar instrucciones al SGBD en el servidor y recibir los resultados de estas instrucciones. Para ello el software cliente y el servidor deben utilizar software de comunicaciones en red.

En las arquitecturas actuales, se utiliza con frecuencia la de cliente-servidor-web: El cliente se conecta a un servidor mediante un navegador web y desde las páginas de éste ejecuta las consultas. El servidor web traduce esta consulta al servidor (o servidores) de datos.

7.2. Sistemas distribuidos.

Ocurre cuando los clientes acceden a datos situados en más de un servidor. También se conoce esta estructura como base de datos distribuida. El cliente no sabe si los datos están en uno o más servidores, ya que el resultado es el mismo independientemente de dónde se almacenan los datos. En esta estructura hay un servidor de aplicaciones que es el que recibe las peticiones, y el encargado de traducirlas a los distintos servidores de datos para obtener los resultados.

Los sistemas homogéneos utilizan el mismo SGBD en múltiples sitios; los heterogéneos dotan de cierta autonomía local a los SGBD participantes.

8. Conclusión

Desde que en los años 70 del siglo pasado se popularizaron las bases de datos como soluciones comunes a la manipulación de grandes volúmenes de datos por parte de aplicaciones diversas, se han desarrollado varias arquitecturas, siendo actualmente dos, las relacionales (SQL) y las NoSQL (No relacionales) las que son principalmente utilizadas hoy en día.

Ejemplos de bases de datos relacionales son: **Oracle, Microsoft SQL Server, MariaDB, MySQL.**

Como ejemplo de bases de datos NoSQL, tenemos: **MongoDB, Redis, Cassandra**.

Aunque actualmente (año 2020) miles de usuarios implementan soluciones como MariaDB o MySQL, los servicios en la nube distribuidos NoSQL, comienzan a ser tendencia en las grandes bases de datos, y así empresas de la talla de **Facebook, Twitter, Instagram, Spotify o Netflix** utilizan el SGDB Cassandra.

8.1. Relación del tema con el sistema educativo actual

Este tema es aplicado en el aula en los módulos profesionales siguientes, con las atribuciones docentes indicadas (PES/SAI):

Formación profesional básica

- Operaciones auxiliares para la configuración y la explotación(TPB en Informática de Oficina/ TPB en informática y Comunicaciones) (PES/SAI)
- Ofimática y archivo de documentos (TPB en Informática de Oficina) (PES/SAI)

Grado Medio

- Aplicaciones ofimáticas (GM de SMR) (PES/SAI)

Grado Superior

- Gestión de bases de datos (ASIR) (PES)
- Bases de Datos (DAW/DAM) (PES)

Bachillerato:

- 4º ESO – Tecnología de la Información y la comunicación (PES)
- Bachillerato – Tecnologías de la Información y la Comunicación (PES)

9. Bibliografía

- C.J. Date: **Introducción a los sistemas de bases de datos** Pearson, 2001.
- Elmasri, R.A. y Navathe S.B: "**Fundamentos de Sistemas de Bases de Datos**". Addison-Wesley, 3^a Edic, 2002.
- Olga Pons, J M Medina, M.A. Vila. **Introducción a los Sistemas de Bases de Datos** Edt Paraninfo (2005)
- Korth, H.F. y Silberschatz: "**Fundamentos de Bases de Datos**". McGraw -Hill, 4^a Edic., 2002.
- Garcia-Molina, H.; Ullman, J.D.; Widom, J. **Database systems: the complete book** - Pearson Education Limited, 2013.
- Abraham Silberschatz, Henry F. Korth, y S. Sudarshan, **Fundamentos de bases de datos** Edt. Mc Graw-Hill (2014)

