Tema 1: Sistemas de Almacenamiento

[1.2] Bases de datos

[1.2.1]la necesidad de gestionar información

Debido a que cada vez había una mayor demanda de gestión de la información y la complejidad de gestionar dicha información mediante el uso de ficheros, surge en la década de los 60 una nueva estructura y forma de almacenamiento y gestión de los datos, las Bases de Datos.

Además de permitir almacenar la información también era necesario compartir esa información entre diferentes personas, posibilitar que los datos se guardaran de manera continua e incluso estuvieran disponibles para las siguientes generaciones. Los problemas con la privacidad ya aparecieron con la propia escritura y así el cifrado de datos es una técnica tan antigua como la propia escritura para conseguir uno de los todavía requisitos fundamentales de la gestión de datos, la seguridad.

En estos últimos años, la demanda ha crecido a niveles espectaculares debido al acceso multitudinario a Internet y a los enormes flujos de información que generan los usuarios. Cada año la necesidad de almacenar información crece exponencialmente en un frenesí que, por ahora, no parece tener fin.

[1.2.2]Sistemas de Bases de Datos

En este tipo de sistemas, los datos se centralizan en una **base de datos** común a todas las aplicaciones. Un software llamado **Sistema Gestor de Bases de Datos (SGBD)** es el que realmente accede a los datos y se encarga de gestionarlos. Las aplicaciones que creen los programadores no acceden directamente a los datos, de modo que la base de datos es común para todas las aplicaciones. **Este software de gestión de Bases de Datos, lo estudiaremos en más detalle en la unidad 4 de este módulo.**

De esta forma, hay, al menos, dos capas a la hora de acceder a los datos. Las aplicaciones se abstraen sobre la forma de acceder a los datos, dejando ese problema al SGBD. Así se pueden concentrar exclusivamente en la tarea de conseguir una interfaz de acceso a los datos para los usuarios.

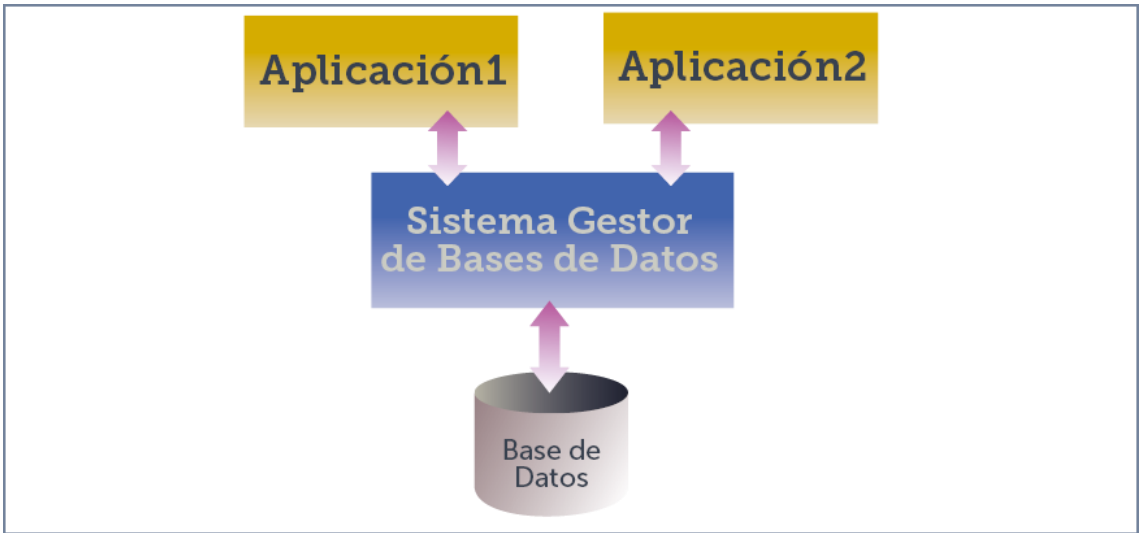

Ilustración 2. Sistemas de información orientados a datos

Cuando una aplicación modifica un dato, la modificación será visible inmediatamente para el resto de las aplicaciones; ya que todas utilizarán la misma base de datos.

Características que aportan las bases de datos

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

Ventajas del uso de Bases de Datos

- **Independencia de los datos y los programas (independencia lógica).** Esto permite modificar los datos sin modificar el código de las aplicaciones y viceversa.
- **Menor redundancia**. Este modelo no requiere que los datos se repitan para cada aplicación que los requiera., en su lugar se diseñan los datos de forma independiente a las aplicaciones. Los programadores de aplicaciones deberán conocer la estructura creada para los datos y la forma en la que deben acceder a ellos.
- **Integridad de los datos**. Al estar centralizados, es más difícil que haya datos incoherentes. Es decir, que una aplicación muestre información distinta al resto de aplicaciones, ya que los datos son los mismos para todas.
- **Mayor seguridad en los datos.** El SGBD es el encargado de la seguridad y se puede centrar en ella de forma independiente a las aplicaciones. Como las aplicaciones deben atravesar la capa del SGBD para llegar a los datos, no se podrán saltar la seguridad.

- **Visiones distintas según el usuario.** Nuevamente, centralizar los datos facilita crear políticas que permitan que los usuarios vean la información de la base de datos de forma distinta.
- **Datos más documentados.** Las bases de datos tienen mucho mejor gestionados los **metadatos,** que permiten describir la información de la base de datos y que pueden ser consultados por las aplicaciones.
- **Acceso a los datos más eficiente.** Esta forma de organizar los datos produce un resultado óptimo en rendimiento ya que los sistemas gestores centralizan el acceso pudiendo ejecutar políticas diferentes en función de la demanda.
- **Menor espacio de almacenamiento.** Puesto que hay muy poca redundancia.
- **Acceso simultáneo a los datos (concurrencia).** Nuevamente las bases de datos tienen más capacidad de conseguir esto. Cuando hay varias aplicaciones que intentan acceder a los datos en los sistemas orientados a los ficheros, compiten por los datos y es fácil el bloqueo mutuo. En el caso de los sistemas orientados a bases de datos, toda petición pasa la capa del SGBD y esto permite evitar los bloqueos.

Desventajas del uso de Bases de Datos

- **Instalación costosa.** El control y administración de bases de datos requiere de un software y hardware poderoso.
- **Requiere personal cualificado.** Debido a la dificultad de manejo de este tipo de sistemas.
- **Implantación larga y difícil.** En relación a los puntos anteriores. La adaptación del personal y del equipamiento es mucho más complicada y lleva bastante tiempo.
- **Ausencia de estándares totales.** Lo cual significa una excesiva dependencia hacia los sistemas comerciales del mercado. Aunque, hoy en día, hay un funcionamiento base y un lenguaje de gestión (**SQL**) que desde hace tiempo se considera estándar (al menos en las bases de datos relacionales).

[1.2.3] Tipos de Bases de datos.

Las bases de datos pueden clasificarse de varias maneras, en base al contexto que se esté manejando, la utilidad de estas o las necesidades que deban cubrir:

Según la variabilidad de la base de datos

- **Estáticas:** su contenido varia muy poco. De consulta, datos históricos usados en tomas de decisiones.
- **Dinámicas:** datos modificados periódicamente. Su volumen varía.

Según el contenido almacenado

- **General:** bases de datos que almacenan contenidos de propóstio general (web, gestión empresarial, e-commerce, …)
- **Bibliográficas:** Bd que almacenan datos de autor, fecha de publicación, editorial, titulo, etc.

- **Texto completo:** permiten buscar términos específicos, palabras claves y todas las opciones de las bibliográficas**.**
- **Directorios:** Son bases de datos con las que tratamos a diario (agenda, móvil)

Según el número de usuarios

- **Mono usuario:** sólo hay un usuario que hace uso de ella (apps móviles)
- **Multi usuario:** múltiples usuarios conectados concurrentemente

Según la localización

- **Centralizadas:** están ubicada en una localización física.
- **Distribuidas:** están fragmentadas o replicadas almacenando copias en varias ubicaciones físicas. Los fragmentos o replicas están en uso por los usuarios.

Según el modelo lógico que se utiliza

- **Jerárquicas:** almacena la información en una **estructura jerárquica** o con un orden de importancia. En este modelo los **datos** están organizados en una figura que nos hacer recordar a árbol puesto al revés.
- **En red:** gran parecido a las jerárquicas; su diferencia principal en la composición del **nodo**. En este modelo los nodos pueden tener diversos padres.
- **Relacionales:** Las bases de datos relacionales han acaparado durante un gran tiempo el terreno de las BD. El centro de este modelo de BD es el uso de las "relaciones" entre datos. Permiten establecer **relaciones** entre datos existentes de forma sencilla y cruzar rápidamente para emitir los análisis necesarios
- **Multidimensionales:** pensadas para funciones específicas. No existe mayor diferencia entre las **bases de datos multidimensionales** y las relacionales. El punto que las separa es apreciable sólo a nivel conceptual. Ya que en estas, los campos o atributos de una tabla pueden ser de dos tipos. **CUBOS OLAP**
- **Orientadas a objetos:** son de las más modernas con las que contamos. Además no hay que dejar de lado su gran capacidad y potencia. En estas, no se almacena información detallada sobre el objeto, se almacena por completo al objeto.
- **Objeto-relacionales:** comparten las características de las bases de datos de modelo relacional (consultas) y del modelo orientado a objetos (almacenamiento)
- **Deductivas:** permite la posibilidad de hacer deducciones a través de una inferencia. Su funcionalidad depende de las condiciones y hechos que se almacenan en la base de datos. También son conocidas como **bases de datos lógicas** ya que sus principios están fundamentados en la lógica matemática. **Inteligencia Artificial**
- **NoSQL** utilizan un modelo básicamente jerárquico**.**

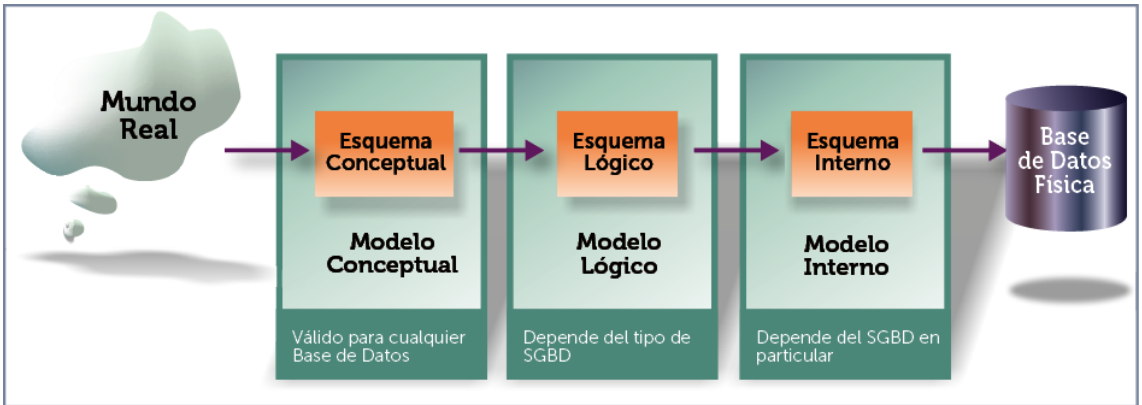

Ilustración 10. Modelos de datos utilizados en el desarrollo de una BD

El punto de partida es el uso en el mundo real que tendrá la base de datos. Ese punto es en el que están los usuarios y es crucial tenerle muy claro. El punto final es el almacenamiento físico de la base de datos.

En este esquema aparece el llamado **Esquema lógico**, que permite pasar de forma más gradual del esquema conceptual al esquema interno.

No obstante, existen modelos lógicos comunes, ya que hay SGBD de diferentes tipos. En la realidad el modelo conceptual clásico se modifica para que existan dos modelos internos: el modelo lógico (referido a cualquier SGBD de ese tipo) y el modelo conceptual propiamente interno (aplicable sólo a un SGBD en particular). De hecho, en la práctica, al definir las bases de datos desde el mundo real hasta llegar a los datos físicos se pasa por todos los esquemas señalados en la Ilustración 10.

Por lo tanto, la diferencia entre los distintos SGBD está en que proporcionan diferentes modelos lógicos.

diferencias entre el modelo lógico y el conceptual

- El modelo conceptual es independiente del DBMS que se vaya a utilizar. El lógico depende de un **tipo** de SGBD en particular
- El modelo lógico está más cerca del modelo físico, el que utiliza internamente el ordenador
- El modelo conceptual es el más cercano al usuario, el lógico es el encargado de establecer el paso entre el modelo conceptual y el modelo físico del sistema.

Algunos ejemplos de modelos conceptuales son:

- **Modelo Entidad Relación (Unidad 2 de este módulo)**
- **Modelo RM/T**
- **Modelo UML**

Ejemplos de modelos lógicos son:

- **Modelo relacional (Unidad 3 de este módulo)**
- **Modelo Codasyl**
- **Modelo Jerárquico**

A continuación, se comentarán los modelos lógicos más importantes.

[1.2.4]modelo jerárquico

Era utilizado por los primeros SGBD, desde que IBM lo definió para su IMS (*Information Management System*, Sistema Administrador de Información) en 1970. Se le llama también modelo en árbol debido a que utiliza una estructura en árbol para organizar los datos.

La información se organiza con una jerarquía en la que la relación entre las entidades de este modelo siempre es del tipo **padre / hijo**. De esta forma hay una serie de nodos que contendrán atributos y que se relacionarán con nodos hijos de forma que puede haber más de un hijo para el mismo padre (pero un hijo sólo tiene un padre).

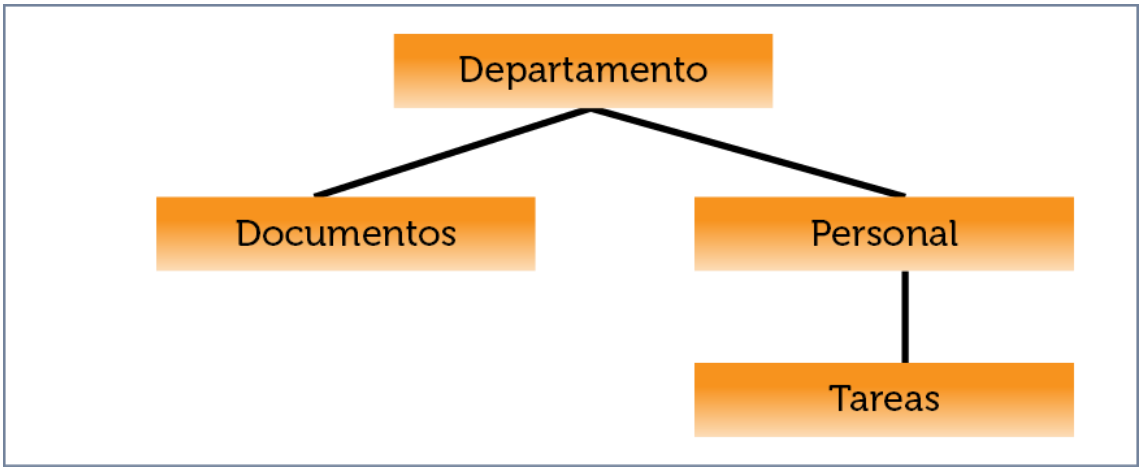

Ilustración 11. Ejemplo de esquema jerárquico

Los datos de este modelo se almacenan en estructuras lógicas llamadas **segmentos**. Los segmentos se relacionan entre sí utilizando **arcos.**

La forma visual de este modelo es de árbol invertido, en la parte superior están los padres y en la inferior los hijos.

Este esquema está en absoluto desuso ya que no es válido para modelar la mayoría de problemas de bases de datos. Su virtud era la facilidad de manejo ya que sólo existe un tipo de relación (padre/hijo) entre los datos; su principal desventaja es que no basta para representar la mayoría de relaciones. Además no mantenía la independencia con la física de la base de datos.

[1.2.5]modelo en red (Codasyl)

Es un modelo que ha tenido una gran aceptación (aunque apenas se utiliza actualmente). En especial se hizo popular la forma definida por el estándar Codasyl a principios de los 70 que se convirtió en el modelo en red más utilizado.

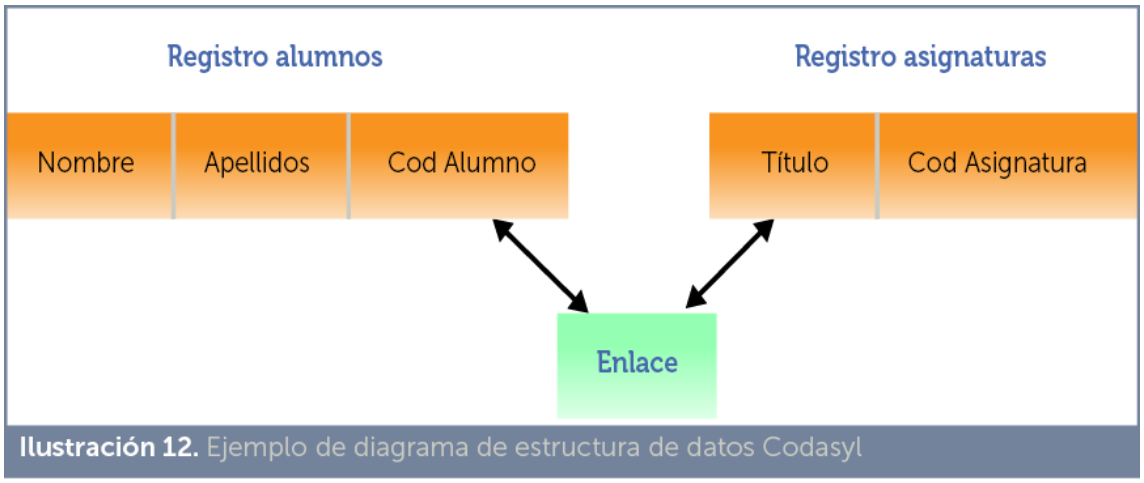

Ilustración 12. Ejemplo de diagrama de estructura de datos Codasyl

El modelo en red organiza la información en **registros** (también llamados **nodos**) y **enlaces**. En los registros se almacenan los datos, mientras que los enlaces permiten relacionar estos datos. Las bases de datos en red son parecidas a las jerárquicas sólo que en ellas puede haber más de un padre.

En este modelo se pueden representar perfectamente cualquier tipo de relación entre los datos (aunque el Codasyl restringía un poco las relaciones posibles), pero hace muy complicado su manejo.

Poseía un lenguaje poderoso de trabajo con la base de datos. El problema era la complejidad para trabajar con este modelo tanto para manipular los datos como programar aplicaciones de acceso a la base de datos. Tampoco mantenía una buena independencia con la física de la base de datos.

[1.2.6]modelo relacional

Es el modelo más popular. Los datos se organizan en tablas y estas en columnas y filas de datos. Las tablas se relacionan entre sí para ligar todos los datos.

Se basa en la teoría de conjuntos y consigue una gran separación entre lo conceptual y lo físico, consiguiendo su total independencia. Tiene un lenguaje considerado estándar, el SQL y una enorme red de usuarios y documentación que facilita su aprendizaje. Además dota de una gran facilidad para establecer reglas complejas a los datos.

El problema es que la simplicidad de manejo y la independencia que consigue se logra a base de un software muy complejo que requiere también un hardware poderoso.

[1.2.7]modelo de bases de datos orientadas a objetos

Desde la aparición de la programación orientada a objetos (**POO** u **OOP**) se empezó a pensar en bases de datos adaptadas a estos lenguajes. La programación orientada a objetos permite cohesionar datos y procedimientos, haciendo que se diseñen estructuras que poseen datos (**atributos**) en las que se definen los procedimientos (**operaciones**) que pueden realizar con los datos. En las bases orientadas a objetos se utiliza esta misma idea.

A través de este concepto se intenta que estas bases de datos consigan arreglar las limitaciones de las relacionales. Por ejemplo el problema de la herencia (el hecho de que no se puedan realizar relaciones de herencia entre las tablas), tipos definidos por el usuario, disparadores (triggers) almacenables en la base de datos, soporte multimedia...

Se supone que son las bases de datos de tercera generación (la primera fue las bases de datos en red y la segunda las relacionales), lo que significa que el futuro parece estar a favor de estas bases de datos. Pero siguen sin reemplazar a las relacionales, aunque son el tipo de base de datos que más está creciendo en los últimos años.

Su modelo conceptual se suele diseñar usando la notación **UML** y el lógico usando **ODMG** (*Object Data Management Group*, grupo de administración de objetos de datos), organismo que intenta crear estándares para este modelo.

Sus ventajas están en el hecho de usar la misma notación que la de los programas (lo que facilita la tarea de su aprendizaje a los analistas y desarrolladores) y que el significado de los datos es más completo. Lo malo es que no posee un lenguaje tan poderoso como el modelo relacional para manipular datos y metadatos, que tiene más dificultades para establecer reglas a los datos y que al final es más complejo para manejar los datos.

[1.2.8]bases de datos objeto-relacionales

Tratan de ser un híbrido entre el modelo relacional y el orientado a objetos. El problema de las bases de datos orientadas a objetos es que requieren reinvertir capital y esfuerzos de nuevo para convertir las bases de datos relacionales en bases de datos orientadas a objetos. En las bases de datos objeto relacionales se intenta conseguir una compatibilidad relacional dando la posibilidad de integrar mejoras de la orientación a objetos.

Estas bases de datos se basan en el estándar **ISO SQL 2000** y los siguientes. En ese estándar se añade a las bases relacionales la posibilidad de almacenar procedimientos de usuario, triggers, tipos definidos por el usuario, consultas recursivas, bases de datos OLAP, tipos LOB,...

Las últimas versiones de la mayoría de las clásicas grandes bases de datos relacionales (**Oracle**, **SQL Server**, **DB2**, ...) son objeto relacionales.

[1.2.9]bases de datos NoSQL

En los últimos años ha aparecido todo un género de bases de datos (de varios tipos) que intentan paliar deficiencias detectadas en el modelo relacional.

El dominio de este modelo parecía demostrar, durante décadas, que era el tipo ideal de base de datos. El cambio de perspectiva se ha producido por la altísima demanda de servicios que requiere Internet. En especial si lo que se requiere es escribir o modificar datos, ya que actualmente todos los usuarios de Internet crean muchísimos datos cada día que requieren ser almacenados inmediatamente (el caso más claro es el de las redes sociales).

Con este panorama han aparecido nuevos tipos de bases de datos y se han modificado y actualizado tipos antiguos que ahora parecen útiles. Lo que aportan la mayoría de estos

tipos de bases de datos, es el uso de otro tipo de esquemas conceptuales e internos más apropiados para este tipo de demandas de usuario.

En resumen las bases de datos NoSQL renuncian al modelo relacional para paliar las carencias del modelo relacional en estos aspectos:

- Aceptar un enorme cantidad peticiones de consulta y especialmente de modificación de datos por minuto
- Gestionar datos muy heterogéneos (irregulares, con tipos de datos cambiantes)
- Gestionar datos que se relacionan de manera muy compleja
- Usar otros lenguajes (diferentes a SQL), más aptos para otras tareas

Esto no significa que cada base de datos NoSQL sea capaz de mejorar en todos los aspectos anteriores, cada tipo de base de datos NoSQL está pensado para algunos de los puntos anteriores.