UD1. Introducción a las bases de datos

1 Datos y archivos

1.1 La necesidad de gestionar la información

En el mundo actual existe una cada vez mayor demanda de gestión de la información. No es una demanda nueva, todas las sociedades a lo largo de la historia han tenido esta necesidad. Desde el principio de los tiempos se ha necesitado disponer de herramientas que facilitaran la gestión de los datos ya que, como herramienta, el ser humano al principio sólo disponía de su memoria y cálculo inmediato y, como mucho, de la ayuda de sus dedos.

La escritura fue la herramienta que permitió al ser humano almacenar información y realizar cálculos sobre ella. Además de permitir compartir esa información entre diferentes personas, también posibilitó que los datos se guardaran de manera continua e incluso estuvieran disponibles para las siguientes generaciones. Los problemas actuales 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.

Cuanto más se han desarrollado las sociedades, mejor han gestionado la información, creando nuevas herramientas: archivos, cajones, fichas, ... Antes de la aparición de las computadoras, el tiempo requerido para manipular estos datos era enorme. Sin embargo el procesamiento era sencillo.

Por esa razón, la informática se adaptó para que la terminología en el propio ordenador se pareciera a los términos de organización de datos clásicos. Así, en informática se sigue hablado de ficheros, formularios, carpetas, directorios,....

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.

Desde la aparición de las primeras computadoras, se intentó automatizar la gestión de los datos. El propio nombre **Informática** hace referencia al hecho de ser una ciencia que trabaja con información. Por ello las bases de datos son una de las aplicaciones más antiguas de la informática.

1.2 Datos e información

Tradicionalmente siempre se ha separado el concepto de **dato** sobre el de **información**.

Un dato es una propiedad en *crudo*, es decir, sin contextualizar. Por ejemplo *Sánchez* y *32* son datos. Desde el punto de vista de la computación, ambos datos se pueden almacenar. Sin embargo, *no podemos considerarlos información hasta que no les demos significado*. Si decimos que *Sánchez* es mi primer apellido y que *32* son los grados centígrados de temperatura que hace ahora en la calle, esos datos pasan a ser información.

La información tiene una importancia humana, es interpretable, reconocible, presentable,.. El dato es simplemente el paso previo. Resumiendo: todo dato debe de ser procesado para obtener información, y es la información lo que nos interesa a los seres humanos.

Como veremos más adelante, los datos se convierten en información gracias a los **metadatos**: datos que sirven para describir otros datos.

1.3 Sistemas de información

Un sistema se define como "*un conjunto de cosas que ordenadamente relacionadas entre sí contribuyen a un determinado objeto*".

El principal cliente de la informática es la empreasa (privada o pública); de ahí que hablemos casi siempre de *sistemas empresariales*.

1.3.1 Sistemas de información

Los sistemas que se dedican a la gestión y *explotación* de la información se les conoce como *sistemas de información*. Un sistema de información genérico está compuesto por los siguientes elementos:

- *Recursos físicos*: carpetas, documentos, equipamiento, discos, ...
- *Recursos humanos*: personal que maneja la información.
- *Protocolo*: normas que se deben cumplir al manejar la información (pensemos por ejemplo en una plantilla o modelo para un documento).

1.3.2 Sistema de información electrónico

Se trata de un sistema de información en el que el manejo de la información se realiza a través de dispositivos electrónicos (o informáticos). Sus componentes son:

- *Datos*. Se trata de la información relevante que almacena y gestiona el sistema de información. Ejemplos de datos son: Sánchez, 12764569F, Calle Mayo 5, Azul...
- *Hardware*. Equipamiento físico que se utiliza para gestionar los datos. cada uno de los dispositivos electrónicos que permiten el funcionamiento del sistema de información: servidores, máquinas de los clientes, routers, switches, impresoras,...
- *Software*. Aplicaciones informáticas que se encargan de la gestión de la base de datos y de las herramientas que facilitan su uso.
- *Recursos humanos*. Personal que maneja el sistema de información.

2 Tipos de gestión de información mediante ordenador

2.1 Sistemas de gestión de ficheros

2.1.1 ¿Qué es un fichero?

En la década de los 60s/70s, los procesos básicos que se realizaban en una empresa eran, eminentemente, la contabilidad y facturación. Para gestionar la información sobre éstos, se solían (aún se hace) utilizar archivos de papel agrupados y ordenados.

Imagen: Archivadores tipo A-Z que siguen utilizándose a día de hoy.

Al informatizar dichos procesos, se pasó de tener esta información en papel a tenerla en el ordenador. En esta adaptación, se mantuvieron conceptos como: archivo, fichero, directorio, caperta, ...

> **Fichero o archivo**: conjunto de información relacionada, tratada como un todo y organizada de forma estructurada. Es una secuencia de dígitos binarios que organiza información relacionada con un mismo aspecto.

Normalmente, los ficheros están formados por *registros lógicos* que contienen los datos relativos a un mismo elemento u objeto (por ejemplo, los datos de un alumno matriculado en un colegio). A su vez, los registros están divididos en campos que contenien los datos elementales que conforman el registro: nombre, dirección, teléfono, email,...

2.1.2 Tipos de ficheros

Según la forma en la que se accede a los ficheros, podemos encontrar algunos tipos, de entre los que destacamos los siguientes:

- **Secuenciales**: se caracteriza porque sus registros están almacenados de forma contigua, por lo que la única forma de acceder a él es leyendo un registro tras otro desde el principio hasta el final. Suelen ser habituales en dispositivos de almacenamiento de acceso secuencial, como las cintas magnéticas; también se utiliza en los CD/DVD, ya que audio, imágenes o vídeo se almacenan en ellos como una espiral continua.

- **Ficheros de acceso directo**: en este tipo se puede acceder a un registro indicando la posición del mismo, o a través de una clave que forma parte del registro como un campo más. Lo más habitual es utilizar como índica la posición física del disco donde está el inicio del registro. Así se puede localizar fácilmente sin recorrer el fichero secuencialmente.
- **Ficheros indexados**: se basa en el uso de índices, que permite el acceso totalmente libre. Estos índices son similares a los de un libro; si queremos leer un capítulo concreto, podemos consultar el índice y nos dice la página en la que comienza. Por tanto, dentro del fichero tendremos una **zona de registros** y una **zona de índices**. Usualmente, los CD/DVD funcionan de esta forma, teniendo al inicio una *TOC* (*Table of content*), de forma que podemos navegar rápidamente entre las pistas de audio.

Otros tipos de ficheros serían: secuenciales indexados (también conocidos como parcialmente indexados), ficheros de acceso calculado o hash, ...

2.1.3 Sistemas de gestión de ficheros

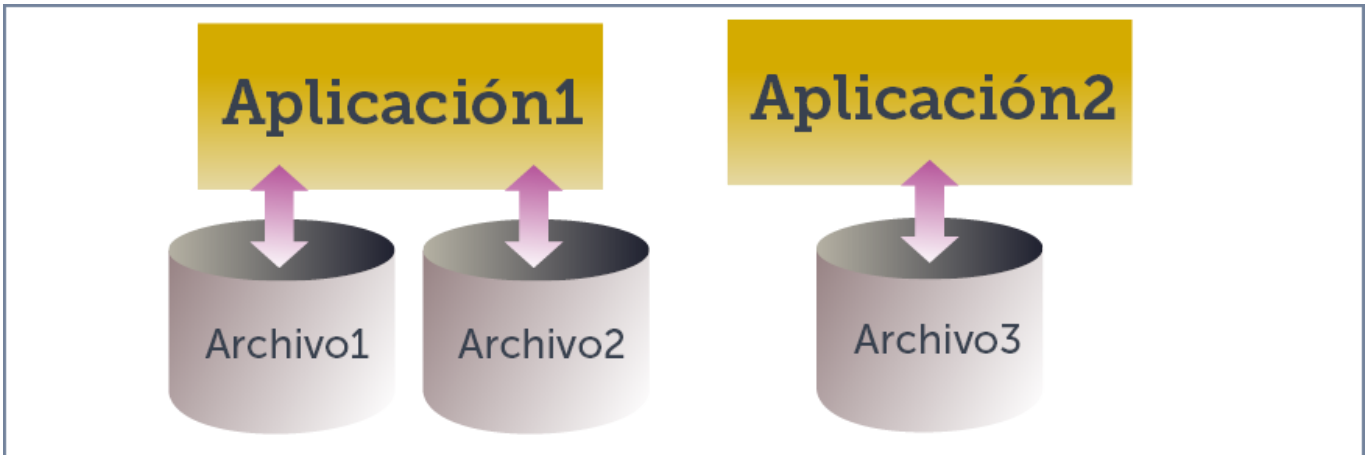


Imagen: diagrama de ejemplo de un sistema basado en ficheros.

Este tipo de sistemas hace referencia a la forma que inicialmente se desarrolló en la informática para gestionar ficheros (y que aún se usa).

La idea es que los datos se almacenan en ficheros y se crean aplicaciones (cuyo código posee la empresa que crea dichas aplicaciones) para acceder a los ficheros. Cada aplicación organiza los datos en los ficheros como le parece mejor y si incorporamos aplicaciones nuevas, estas usarán sus propios ficheros.

Cada aplicación almacena y utiliza sus propios datos de forma un tanto caótica. La ventaja de este sistema (la única ventaja), es que los procesos son independientes por lo que la modificación de uno no afecta al resto. Pero tiene grandes inconvenientes:

- *Programación de aplicaciones compleja*. Ya que los programadores se deben de encargar de lo que tiene que hacer la aplicación y además de estructurar los datos en disco.
- *Datos redundantes*. Ya que se repiten continuamente. Podría, por ejemplo, ocurrir que una segunda aplicación utilice datos de personales, que resulta que ya estaban almacenados en los ficheros de una primera aplicación, pero como ambas son independientes, los datos se repetirán.
- *Datos inconsistentes*. En relación con el problema anterior, ya que un proceso cambia sus datos y no los del resto. Por lo que la misma información puede tener distintos valores según qué aplicación acceda a él.
- *Difícil acceso a los datos*. Cada vez que se requiera una consulta no prevista inicialmente, hay que modificar el código de las aplicaciones o incluso crear una nueva aplicación. Esto hace imposible pensar

en nuevas consultas e instantáneamente obtener sus resultados; inviable para aplicaciones que requieren grandes capacidades de consultas y análisis de datos.
- *Coste de almacenamiento elevado*. Al almacenarse varias veces el mismo dato, se requiere más espacio en los discos. Además, las aplicaciones también ocupan mucho al tener que pensar en todas las posibles consultas sobre los datos que la organización precisa.
- *Dependencia de los datos a nivel físico*. Para poder saber cómo se almacenan los datos, es decir qué estructura se utiliza de los mismos, necesitamos ver el código de la aplicación; es decir el código y los datos no son independientes.
- *Dificultad para el acceso simultáneo a los datos*. El acceso simultáneo requiere que varios usuarios al puedan acceder a la misma información. Con este tipo de sistemas es extremadamente difícil conseguir esta capacidad.
- *Dificultad para administrar la seguridad del sistema*. Ya que cada aplicación se crea independientemente. Es, por tanto, muy difícil establecer criterios de seguridad uniformes. Es decir, los permisos que cada usuario tiene sobre los datos, se establecen de forma muy confusa (y nada uniforme ya que cada aplicación puede variar la seguridad).

2.2 Sistemas de Bases de Datos

A finales de los 70s la aparición de los sistemas de bases de datos supuso un gran cambio, permitiendo solventar muchas de las desventajas de los sistemas basados en ficheros. Pero, ¿qué es una base de datos?

> Una **base de datos** es una colección de datos relacionados lógicamente entre sí, con una definición y descripción comunes y que están estructurados de una determinada manera. Es un conjunto estructurado de datos que representa entidades y sus interrelaciones, almacenados con la mínima redundancia y posibilitando el acceso a ellos eficientemente por parte de varias aplicaciones y usuarios. La base de datos no sólo contiene los datos de la organización, también almacena una descripción de dichos datos. Esta descripción es lo que se denomina metadatos, se almacena en el diccionario de datos o catálogo y es lo que permite que exista independencia de datos lógica–física.

Esto serán los que utilicemos a lo largo de todo el curso.

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.

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.

Imagen: diagrama de ejemplo de un sistema basado en bases de datos.

Ventajas

- *Independencia de los datos y los programas*. 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 más 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*. Nuevamente el SGBD tiene 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

- *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).

3 ¿Cómo funciona un Sistema Gestor de Bases de Datos?

3.1 Funciones de un SGBD

Podemos estudiar el funcionamiento de un SGBD a tres niveles.

3.1.1 A nivel de definición o descripción

Permite al diseñador de la base de datos crear las estructuras apropiadas para integrar adecuadamente los datos. Realmente la función de definición lo que hace es **gestionar los metadatos**. Los metadatos son la estructura de la dispone el sistema de base de datos para documentar cada dato. Los metadatos también son datos que se almacenan en la propia base de datos; pero su finalidad es describir los datos.

Un metadato nos permite para saber a qué información real se refiere cada dato. Por ejemplo: *Sánchez*, *Rodríguez* y *Crespo* son datos. Pero *Primer Apellido* es un metadato que, si está correctamente creado, nos permite determinar que *Sánchez*, *Rodríguez* y *Crespo* son primeros apellidos.

Dicho de otra forma, sin los metadatos, no podríamos manejar los datos como información relevante. Por ello son fundamentales. Son, de hecho, la base de la creación de las bases de datos.

Los metadatos pueden indicar cuestiones complejas. Por ejemplo, que de los Alumnos almacenamos su dni el cual lo forman 9 caracteres. Incluso podremos indicar que en el dni los 8 primeros son números y el noveno un carácter en mayúsculas que además cumple una regla concreta y sirve para identificar al alumno.

Por lo tanto, en realidad, la función de definición sirve para crear, eliminar o modificar metadatos.

Un lenguaje conocido como **lenguaje de descripción de datos** o **DDL**, es el que permite realizar la función de definición en las bases de datos.

3.1.2 A nivel de manipulación

Permite cambiar y consultar los **datos** de la base de datos. Se realiza mediante un **lenguaje de modificación de datos** o **DML**. Mediante este lenguaje se puede:

- Añadir datos
- Eliminar datos
- Modificar datos
- Consultar datos

Actualmente se suele diferenciar la **función de consulta de datos**, diferenciándola del resto de operaciones de manipulación de datos. Se habla de que la función de consulta se realiza con un **lenguaje de consulta de datos** o **DQL** (*Data Query Language*).

3.1.3 A nivel de control

Mediante esta función los administradores poseen mecanismos para proteger los datos. De manera que se permite a cada usuario ver ciertos datos y otros no, o bien usar ciertos recursos concretos de la base de datos y prohibir otros. Es decir, es la función encargada de establecer los permisos de acceso a los elementos que forman parte de la base de datos.

El lenguaje que implementa esta función es el **lenguaje de control de datos** o **DCL**.

3.2 Utilidades de un SGBD

> En este punto sería bueno diferenciar los conceptos de **base de datos** y **sistema gestor de bases de datos**. Una *base de datos* sería el conjunto de los datos y metadatos, mientras que el *sistema gestor de base de datos* sería el conjunto de herramientas que nos van a permitir *definir*, *manipular* y *controlar* nuestra base de datos.

Los SGBD proporcionan un conjunto coordinado de programas, procedimientos y lenguajes que permiten a los distintos usuarios realizar sus tareas habituales con los datos, garantizando además la seguridad de los mismos.

Además de las tres funciones principales comentadas anteriormente, hoy en día los SGBD son capaces de realizar numerosas operaciones. Para ello proporcionan numerosas herramientas, muchas de ellas permiten trabajar de una forma más cómoda con las . Las más destacadas son:

- *Herramientas para la creación y especificación del diccionario de datos*. El diccionario de datos es la estructura de la base de datos que almacena los metadatos. Es decir el diccionario de datos contiene la descripción de todos los datos de la base de datos.
- *Herramientas para administrar y crear la estructura física de la base de datos*. El SGBD proporciona herramientas para especificar la forma en la que se almacenarán los datos en la computadora (o computadoras) que alojen la base de datos. Estas herramientas nos permitirán diseñar una forma de almacenamiento centrada en optimizar el acceso a los datos.
- *Herramientas para la manipulación de los datos*. Nos permitirán añadir, modificar, suprimir o consultar datos (función de manipulación) de la forma más sencilla posible.
- *Herramientas de recuperación en caso de desastre*. Si ocurre un mal funcionamiento del sistema, un fallo en la alimentación del sistema, errores de red, etc. En ese caso los buenos SGBD poseen y proporcionan mecanismos para que se recupere la máxima información posible y se asegure su integridad.
- *Herramientas para la creación y restablecimiento de copias de seguridad*. Es una de las tareas fundamentales, ya que permite recuperar la información en caso de pérdida de datos.
- *Herramientas para la gestión de la comunicación de la base de datos*. Encargadas de configurar el hardware y software de conexión a la red. Así como los mecanismos necesarios para configurar adecuadamente el software que se encarga de recibir y comunicar las peticiones de los clientes.
- *Herramientas para la creación de aplicaciones de usuario*. Es decir, herramientas para los programadores de aplicaciones, los cuales crean el software con el que los usuarios accederán de forma cómoda a la base de datos.
- *Herramientas de instalación y configuración* de la base de datos.
- *Herramientas para la exportación e importación* de datos a o desde otros sistemas.
- *Herramientas para gestionar la seguridad*. Permiten establecer privilegios y permisos diferentes para los usuarios, así como impedir el acceso no deseado (función de control).

3.3 Niveles de abstracción de una base de datos

En cualquier software siempre hay dos puntos de vista: la que tienen los *usuarios finales*, llamada visión o **nivel externo**; y la que tienen los *creadores* del software, llamada **nivel interno**.

Podemos asemejarlo a la instalación eléctrica de una casa:

- El *usuario final* aprecia el **nivel externo**, que serían los *interruptores*, *conmutadores*, *enchufes*, *bombillas*, ...
- El *creador*, en este caso un electricista, sería capaz de apreciar las diferentes conexiones, el cableado de fase, neutro, toma de tierra, las cajas repartidoras, etc...

En el caso de una base de datos hay más niveles, más formas diferentes de observar la base de datos. Cada nivel se corresponde con los diferentes *usuarios* que pueden manejar la base de datos. A cada nivel se le llama **nivel de abstracción**, que nos permite *abstraernos* de detalles concretos para entender la base de datos sin más complejidad de la necesaria. Por ejemplo, los usuarios finales podrán conocer la base de datos sin saber de los *entresijos* técnicos; y los administradores de la base de datos conocerán los detalles técnicos sin tener por qué saber cómo introducen los datos los usuarios finales.

Los niveles habituales son:

- *Nivel físico*. Nos permite saber la forma en la que está almacenada la base de datos. Por ejemplo en qué discos duros, qué archivos utiliza, de qué tipo son los archivos, bajo qué sistema operativo,... Este nivel es el que está más cercano a la visión de la base de datos que posee la computadora, por lo que es absolutamente dependiente del hardware y el software (especialmente del Sistema Operativo).
- *Nivel interno*. Un poco más cercano a la visión que tenemos las personas. Permite observar la base de datos como un conjunto de estructuras que relacionan la información humana con la información digital. A este nivel no se depende del hardware concreto que tengamos; es decir, no se habla de discos, servidores, archivos,... sino de las estructuras que disponemos en nuestro SGBD en particular para organizar los datos.
- *Nivel conceptual*. Es el nivel de mayor abstracción y el más importante. Se trata de una visión organizativa de los datos independiente tanto del hardware como del software que tengamos. Es el plano o modelo general de la base de datos y a este nivel es al que trabajan las o los analistas y diseñadores cuando crean el primer esquema de la base de datos. En ningún momento queda influido por el SGBD en particular que usemos.
- *Nivel externo*. Se trata de la visión de los datos que poseen los usuarios y usuarias finales de la base de datos. Esa visión es la que obtienen a través de las aplicaciones. Las aplicaciones creadas por los desarrolladores abstraen la realidad conceptual de modo que el usuario no conoce las relaciones entre los datos, como tampoco conoce dónde realmente se están almacenando los datos. Es la forma en la que cualquier persona desea manejar una base de datos a través de formularios, informes, listas,...

> La idea de estos niveles procede de la normalización hecha en el modelo ANSI/X3/SPARC y sigue estando muy presente en la gestión actual de las bases de datos.

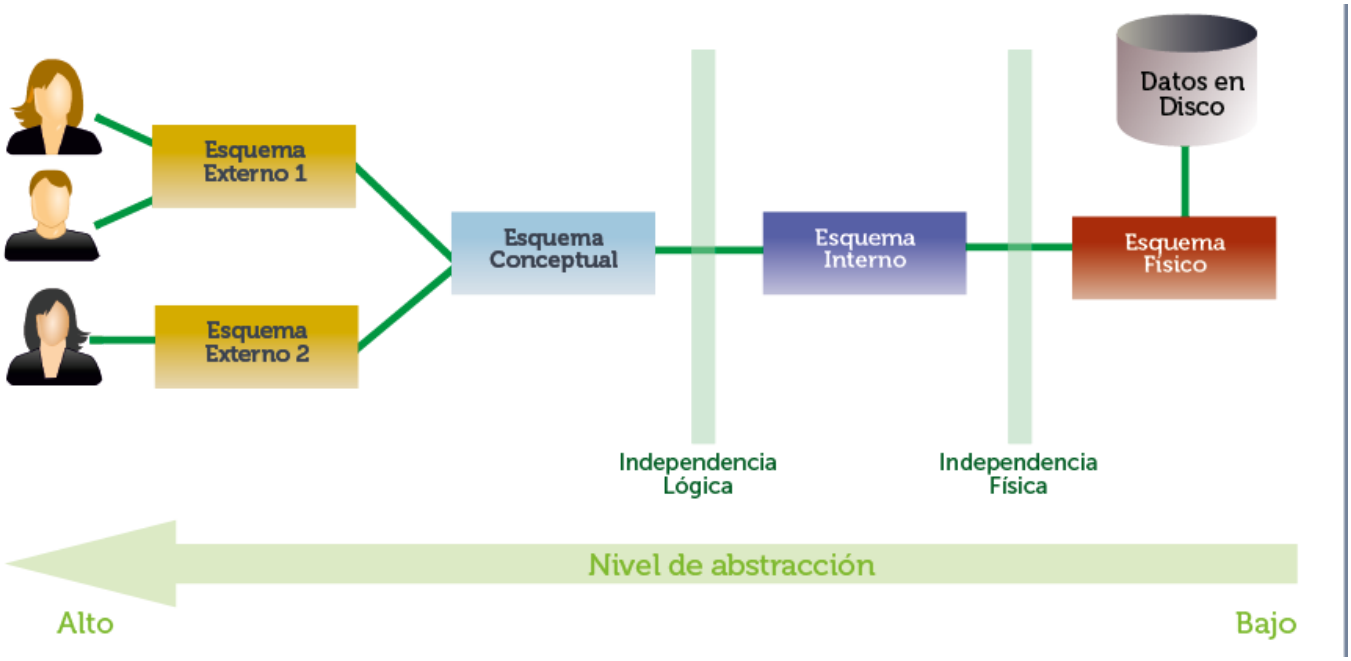


Imagen: diferentes esquemas y visiones de una bases de datos.

En la ilustración anterior se observa la distancia que poseen los usuarios de la base de datos respecto a la realidad física de la base de datos (representada con el cilindro). La física son los datos en crudo, es decir en formato binario dentro del disco o discos que los contienen. El esquema físico es el que se realiza pensando más en esa realidad y los esquemas externos los que se crean pensando en la visión de los usuarios.

Las dos columnas que aparecen en la imagen reflejan dos fronteras a tener en cuenta:

- **Independencia Lógica**. Los esquemas de los niveles conceptual y externo son independientes del software concreto de base de datos que usemos; no dependen en absoluto de él. Por ello esos esquemas nos valdrían para cualquier SGBD que utilicemos.
- **Independencia Física**. La da la barrera entre el esquema físico y el interno e indica que el esquema interno es independiente del hardware concreto que usemos. El esquema físico se diseña en base a un hardware concreto, pero él interno no. Eso permite concentrarse en detalles más conceptuales.

3.4 Diferentes tipos de usuarios de una base de datos

Intervienen (como ya se ha comentado) muchas personas en el desarrollo y manipulación de una base de datos. Se describen, a continuación, los actores más importantes.

Los **informáticos**, lógicamente, son los profesionales que definen y preparan la base de datos. Pueden ser:

- *Directivos/as*. Organizadores y coordinadores del proyecto a desarrollar y máximos responsables del mismo..
- *Analistas*. Son los encargados de controlar el desarrollo de la base de datos aprobada por la dirección. Dirigen a los desarrolladores y operadores. Normalmente son, además, los diseñadores de la base de datos: es decir, crean el esquema conceptual de la misma.
- *Administradores/as de las bases de datos*. Encargados de crear el esquema interno de la base de datos. También gestionan el correcto funcionamiento del SGBD. Sus tareas incluyen la planificación de copia de seguridad, gestión de usuarios y permisos, optimización del rendimiento, monitorización de problemas y creación de los objetos de la base de datos.
- *Desarrolladores/as o programadores/as*. Encargados de la realización de las aplicaciones de usuario para que estos accedan a la base de datos.

- *Equipo de mantenimiento*. También se les llama operadores. Encargados de dar soporte a los usuarios en el trabajo diario (suelen incorporar además tareas administrativas como la creación de copias de seguridad por ejemplo o el arreglo de problemas de red por ejemplo).

Por otro lado, estarían los **usuarios**:

- *Expertos/as*. Realizan operaciones avanzadas sobre la base de datos. Normalmente conocen el lenguaje de manipulación de datos (DML) para acceder a la base de datos. Son usuarios, por lo tanto, con conocimientos informáticos que se encargan en las empresas de los clientes de algunas acciones más complejas sobre la base de datos que las que realizan los usuarios habituales.
- *Habituales*. Utilizan las aplicaciones creadas por los desarrolladores para consultar y actualizar los datos. Son los que trabajan en la empresa a diario con estas herramientas y el objetivo fundamental de todo el desarrollo de la base de datos.
- *Ocasionales*. Son usuarios que utilizan un acceso mínimo a la base de datos a través de una aplicación que permite consultar ciertos datos. Serían por ejemplo los usuarios que consultan el horario de trenes a través de Internet. Aunque se les llama ocasionales son el núcleo del trabajo con la base de datos ya que son los que más la utilizan (ya que son sus usuarios más numerosos) y son, por ejemplo, los que visitan la base de datos para realizar compras o para informarse del negocio representado en la base de datos.

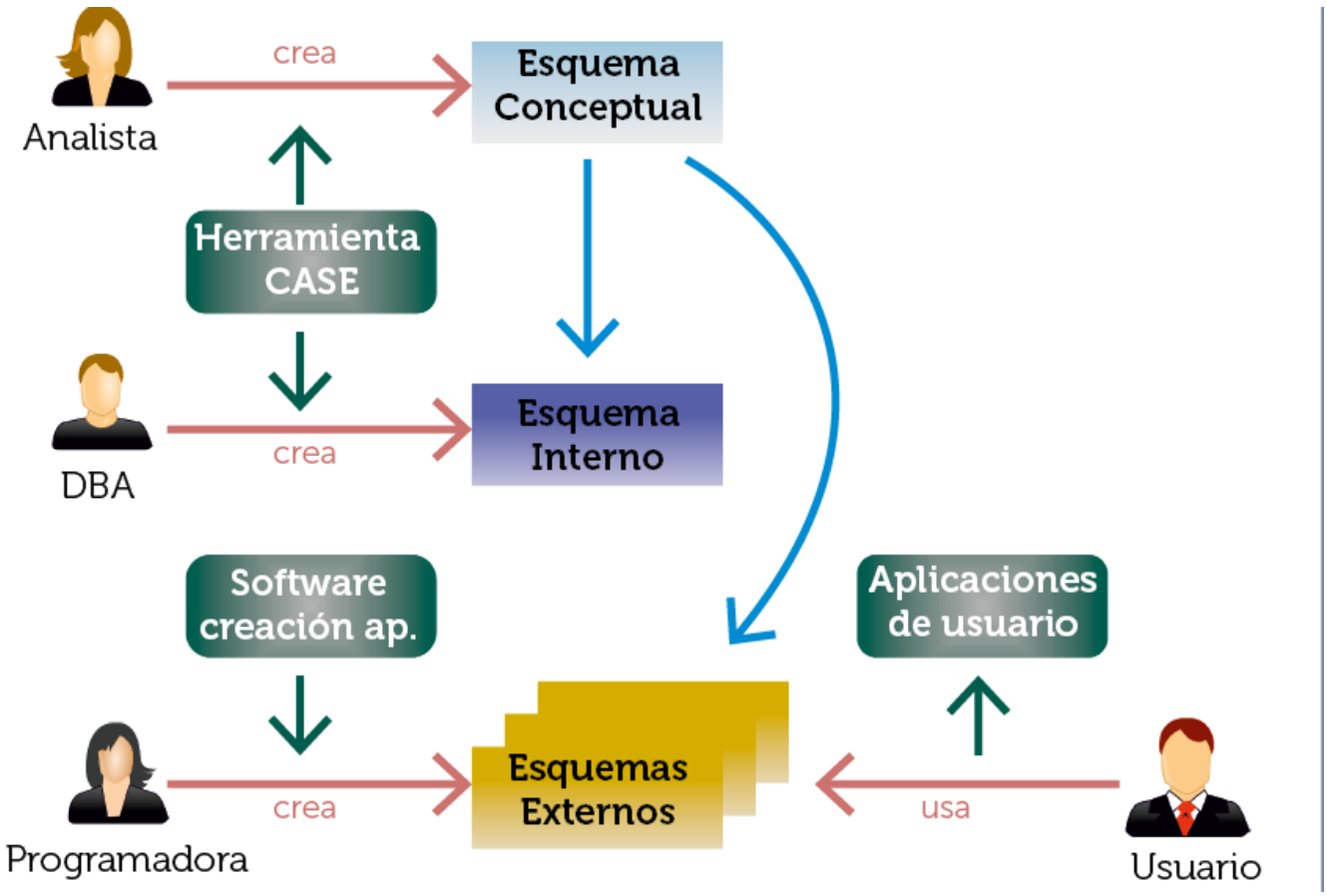


Imagen: Tipos de usuarios y esquema de base de datos al que acceden.

4 Tipos de modelos lógicos

Para entendernos, podemos decir que el modelo lógico es el que se utiliza para representar los datos en la visión conceptual. Esto determinará el tipo de SGBD que vamos a utilizar. Los modelos lógicos más conocidos (sobre todo en pasado) y extendidos (en la actualidad) son los siguientes:

4.1 Modelo jerárquico

Fue uno de los primeros modelos usados por los SGBD primigenios (finales de los 60-inicio de los 70 del siglo XX).

La información se organiza con un 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).

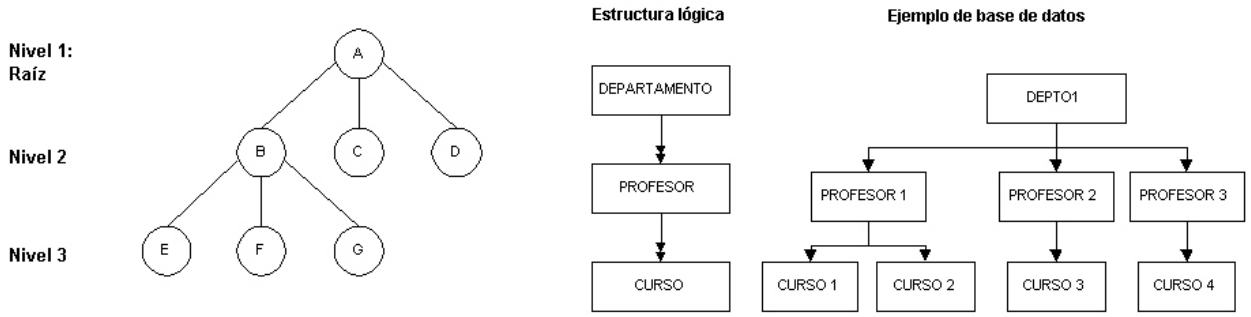


Imagen: estructura lógica y ejemplo de una base de datos jerárquica.

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.

> Si deseas completar tus conocimientos acerca de este modelo, te proponemos los siguientes enlaces: http://es.wikipedia.org/wiki/Modelo_jer%C3%A1rquico http://ddd.uab.cat/pub/elies/elies_a2000v9/4-2-1.htm

4.2 Modelo *Codasyl* o en red

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

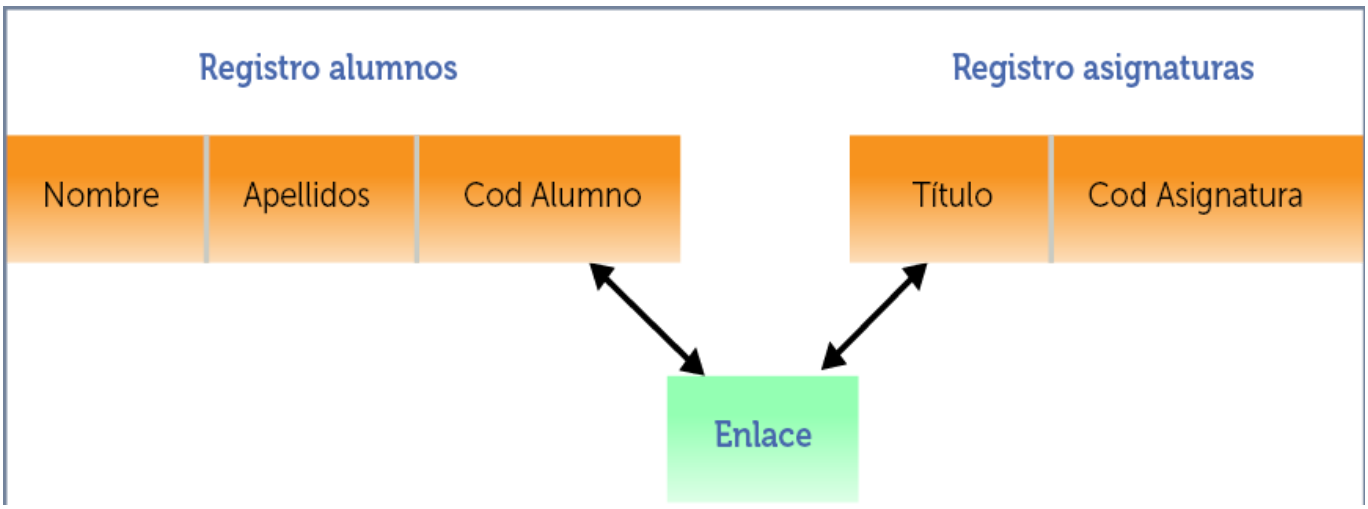


Imagen: ejemplo de estructura de modelo Codasyl, con un enlace entre registros.

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. Al no tener que duplicar la información se ahorra espacio de almacenamiento.

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.

> Si deseas completar tus conocimientos acerca de este modelo, te proponemos los siguientes enlaces: http://es.wikipedia.org/wiki/Modelo_de_red http://ddd.uab.cat/pub/elies/elies_a2000v9/4-2-2.htm

4.3 Modelo relacional

Este modelo es posterior a los dos anteriores y fue desarrollado por Codd en 1970. Hoy en día las bases de datos relacionales son las más utilizadas.

En el modelo relacional la base de datos es percibida por el usuario como un conjunto de tablas. Esta percepción es sólo a nivel lógico, ya que a nivel físico puede estar implementada mediante distintas estructuras de almacenamiento. Se basa en la teoría de conjuntos y consigue una gran separación entre lo conceptual y lo físico, consiguiendo su total independencia. 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.

El modelo relacional utiliza tablas bidimensionales (relaciones) para la representación lógica de los datos y las relaciones entre ellos. Cada relación (tabla) posee un nombre que es único y contiene un conjunto de columnas.

Se llamará registro, entidad o tupla a cada fila de la tabla y campo o atributo a cada columna de la tabla.

A los conjuntos de valores que puede tomar un determinado atributo, se le denomina dominio.

Una clave será un atributo o conjunto de atributos que identifique de forma única a una tupla.

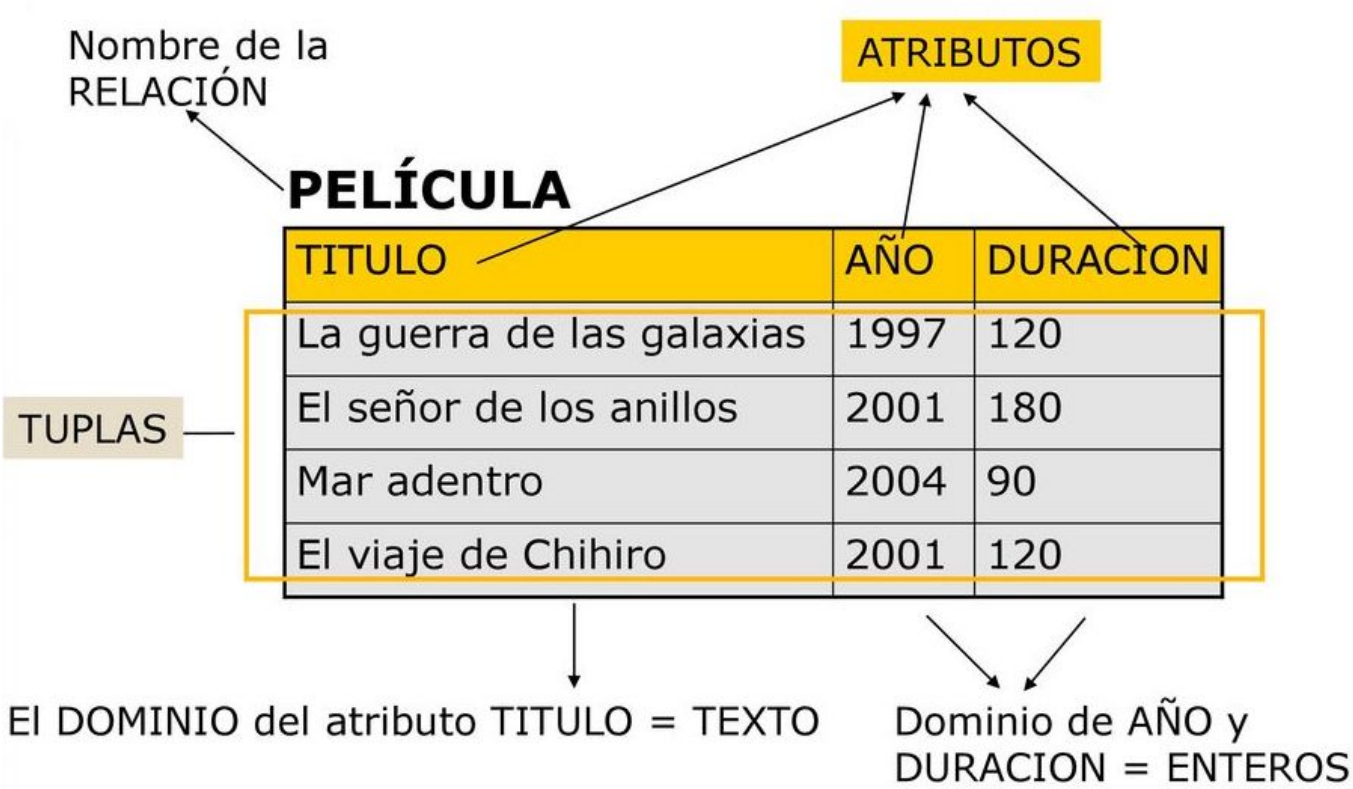


Imagen: ilustración sobre el modelo relacional, resaltando algunas de sus características.

Las tablas deben cumplir una serie de requisitos:

- Todos los registros son del mismo tipo.
- La tabla sólo puede tener un tipo de registro.
- No existen campos o atributos repetidos.
- No existen registros duplicados.
- No existe orden en el almacenamiento de los registros.
- Cada registro o tupla es identificada por una clave que puede estar formada por uno o varios campos o atributos.

El lenguaje habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se conoce como normalización de una base de datos.

> Si deseas completar tus conocimientos acerca de este modelo, te proponemos el siguiente enlace: http://es.wikipedia.org/wiki/Modelo_relacional

4.4 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.

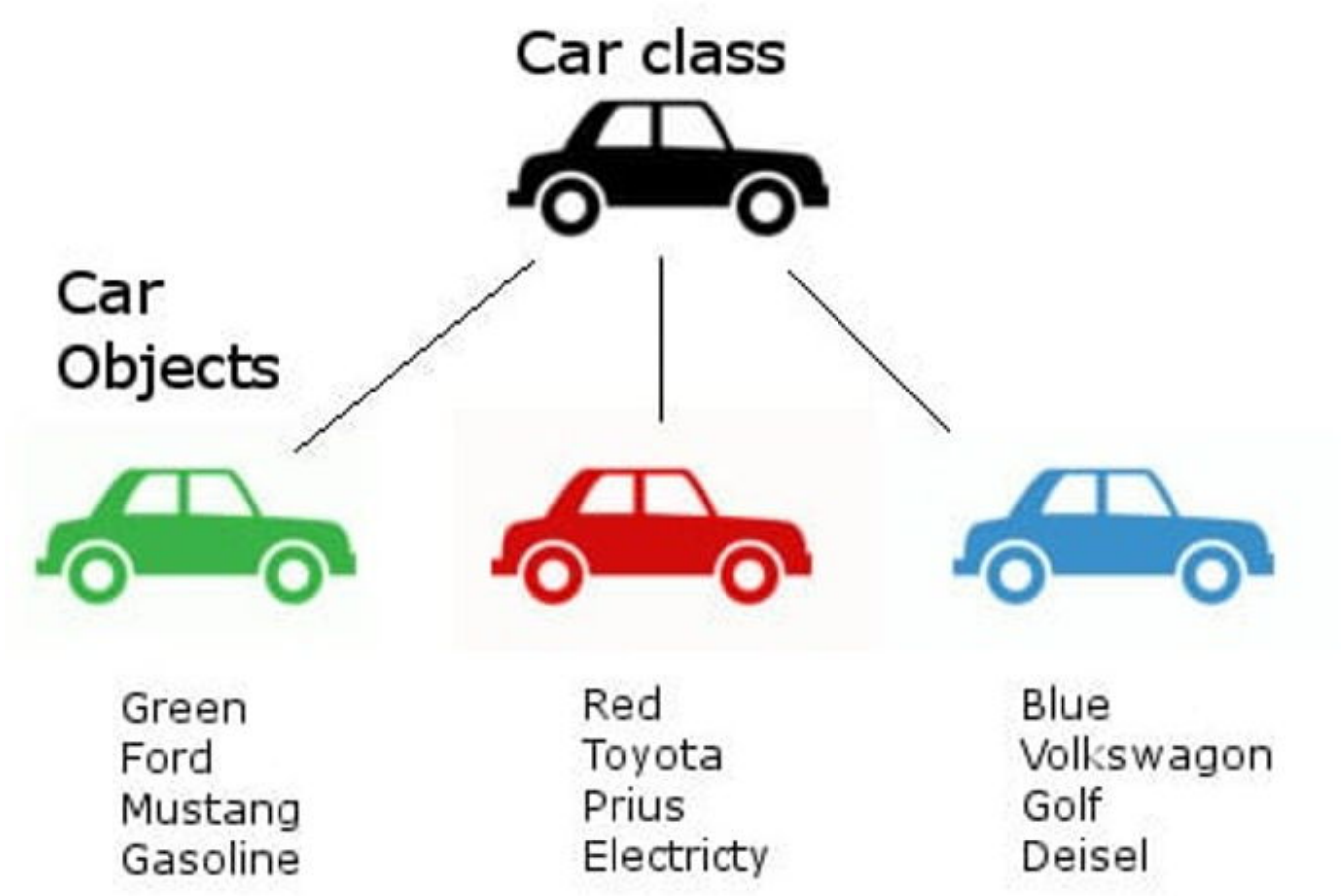


Imagen: representación conceptual de clases y objetos.

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.

> Si deseas completar tus conocimientos acerca de este modelo, te proponemos el siguiente enlace: http://es.wikipedia.org/wiki/Base_de_datos_orientada_a_objetos

4.5 Modelo objeto-relacional

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.

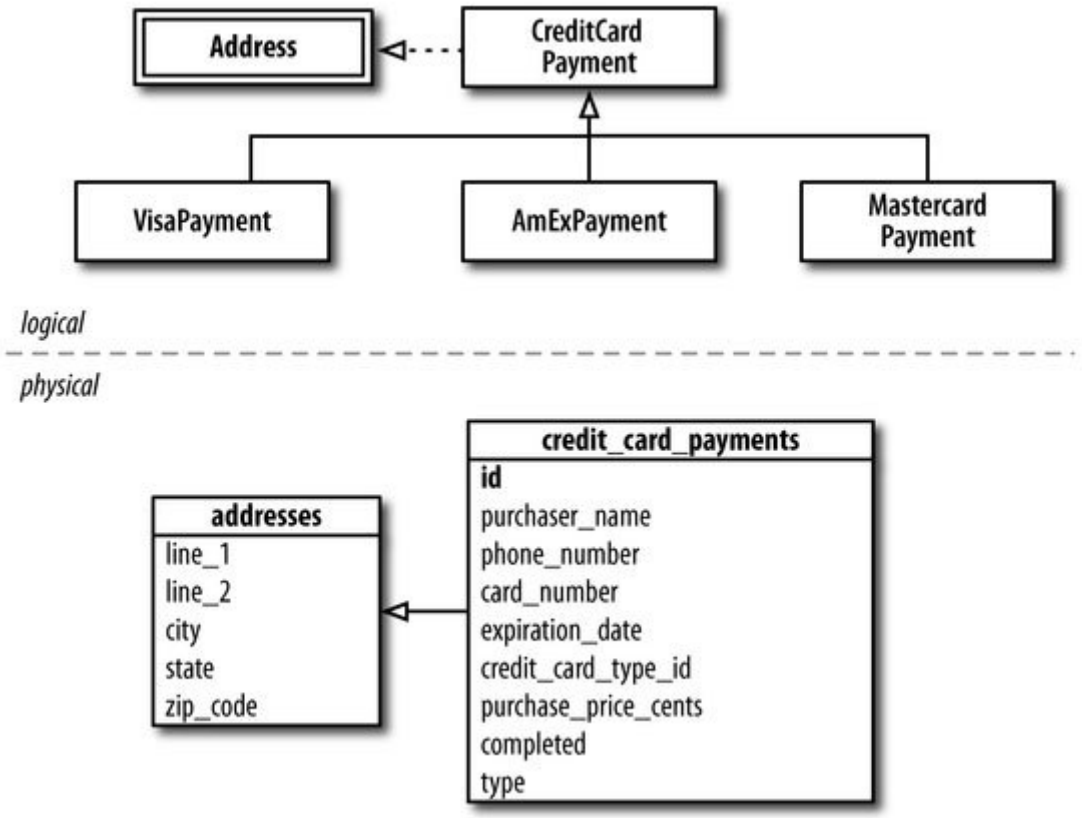


Imagen: ejemplo de uso de herencia en una base de datos objeto-relacional.

Por otro lado, es muy habitual utilizar, sobre una base de datos relacional, una serie de capas de software intermedias, conocidas como ORM (*Object relational mapping*) que se encargan de hacer la traducción del mundo orientado a objetos al mundo relacional. Unos de loos softwares más famosos es Hibernate.

> Si deseas completar tus conocimientos acerca de este modelo, te proponemos el siguiente enlace: https://es.wikipedia.org/wiki/Base_de_datos_objeto-relacional

4.6 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.

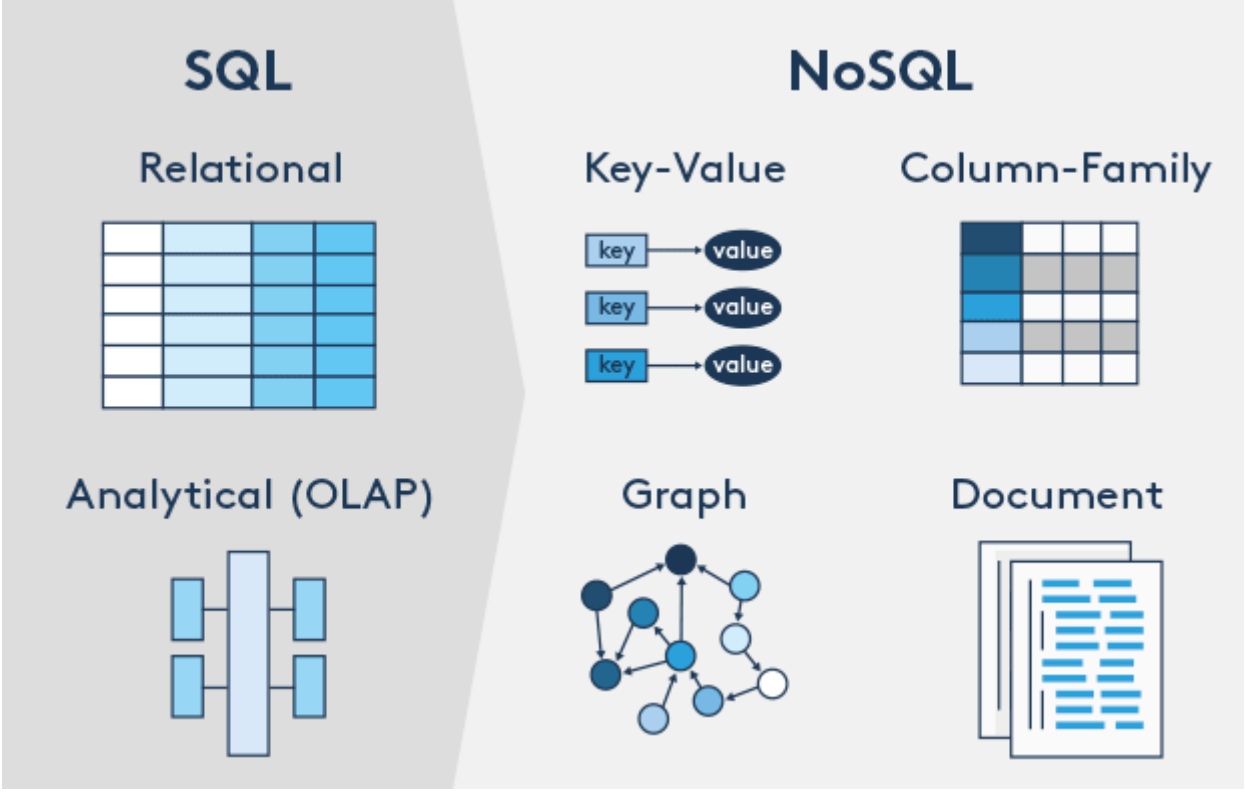


Imagen: comparativa entre modelo relacional (SQL) y otros modelos de datos más modernos (NoSQL).

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.

AMPLIACIÓN

5. Tipos de Bases de Datos

Como hemos visto, por cada modelo de datos se establecen sustanciales diferencias entre unas bases de datos y otras, pero, ¿Esta es la única clasificación de las bases de datos existente? No, vamos a ver a continuación una detallada descripción de los tipos de bases de datos teniendo en cuenta varios criterios.

5.1 Bases de datos según la localización de la información

- **Bases de datos centralizadas**: Se trata de bases de datos ubicadas en un único lugar, un único computador. Pueden ser bases de datos monousuario que se ejecutan en ordenadores personales o sistemas de bases de datos de alto rendimiento que se ejecutan en grandes sistemas. Este tipo de organización facilita las labores de mantenimiento, sin embargo, hace que la información contenida en dicha base, sea más vulnerable a posibles fallos y limita su acceso. Este tipo de bases de datos puede ofrecer dentro de la arquitectura Cliente/Servidor dos configuraciones:
    - *Basada en anfitrión*: ocurre cuando la máquina cliente y la máquina servidor son la misma. Los usuarios se conectarán directamente a la máquina donde se encuentra la base de datos.
    - *Basada en Cliente/Servidor*: ocurrirá cuando la base de datos reside en una máquina servidor y los resultados acceden a la base de datos desde su máquina cliente a través de una red
- **Bases de datos distribuidas**: Según la naturaleza de la organización es probable que los datos no se almacenen en un único punto, sino que se sitúen en un lugar o lugares diferentes a donde se encuentran los usuarios. Una base de datos distribuida es la unión de las bases de datos mediante redes. Los usuarios se vinculan a los servicios de bases de datos distantes mediante una amplia variedad de redes de comunicación. Puede imaginarse una compañía con diferentes oficinas regionales, donde se encuentra distribuida la base de datos. Si embargo, los ejecutivos pueden tener acceso a la información de todas las oficinas reginales.

Abundaremos sobre ellas en un apartado posterior.

5.2 Bases de datos según su contenido

- **Bases de datos con información actual**: Contienen información muy concreta y actualizada, normalmente, de tipo numérico: estadísticas, series históricas, resultados de encuestas, convocatorias de becas o subvenciones, convocatorias de eventos, ofertas de empleo, ...
- **Directorios**: recogen datos sobre personas o instituciones especializadas en una actividad o materia concreta. Hay directorios de profesionales, de investigadores, de centros de investigación, de bibliotecas, de revistas científicas, de empresas, de editoriales, ...
- **Bases de datos documentales**: En éste último grupo, cada registro se corresponde con un documento, sea éste de cualquier tipo: una publicación impresa, un documento audiovisual, gráfico. Dependiendo de si incluyen o no el contenido completo de los documentos que describen, podremos tener:
    - *Bases de datos de texto completo*: constituidas por los propios documentos en formato electrónico, con un volcado completo de su texto.
    - *Archivos electrónicos de imágenes*: Constituidos por referencias que permiten un enlace directo con la imagen del documento original, sea éste un documento iconográfico (fotografías, imágenes de televisión, ...) o un documento impreso digitalizado en formato de imagen.
    - *Bases de datos referenciales*: Sus registros no contienen el texto original sino tan sólo la información fundamental para describir y permitir la localización de documentos impresos, sonoros, iconográficos, audiovisuales o electrónicos. En estos sistemas de información sólo se puede obtener referencias sobre documentos que habrá que localizar posteriormente en otro servisio (archivo, biblioteca, fonoteca,...) o solicitar a un servicio de suministro de documentos.

5.3 Bases de datos según su uso

- **Base de datos individual**: Es una base de datos utilizada básicamente por una persona. El sistema administrador de la base de datos y los datos son controlados por el mismo usuario. Puede estar

almacenada en la unidad de disco duro del usuario o en el servidor de archivos de una red de área local. Por ejemplo, un gerente de ventas podría contar con una base de datos para el control de sus vendedores y su desmpeño.

- **Base de datos compartida**: Son bases de datos con múltiples usuarios y que muy probablemente pertenezcan a la misma organización, como la base de datos de una compañía. Se encuentra almacenada en una computadora potente y bajo el cuidado de un profesional en el área, el administrador de la base de datos. Los usuarios tienen acceso a la base de datos mediante una red de área local o una red de área extensa.
- **Bases de datos de acceso público**: Son bases de datos accesibles por cualquier persona. Puede no ser necesario pagar un canon para hacer uso de los datos contenidos en ellas.
- **Bases de datos propietarias o bancos de datos**: Se trata en general de bases de datos de gran tamaño, desarrolladas por una organización y que contienen temas especializados o de carácter particular. El público general puede tener acceso a estas bases a veces de forma gratuita y otras mediante el pago de una cuota. Pueden ofrecer información que va desde negocios, economía, inversión, técnica y científica hasta servicios de entretenimiento.Permiten encontrar en minutos lo que tardaría horas ojeando revistas.

5.4 Bases de datos según la variabilidad de la información

- **Bases de datos estáticas**: Son bases de datos de sólo lectura. Se utilizan para el almacenamiento de datos históricos que pueden ser analizados y utilizados para el estudio del comportamiento de un conjunto de datos a través del tiempo. Permiten realizar proyecciones y toma de decisiones.
- **Bases de datos dinámicas**: Son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como

6. Bases de datos distribuidas

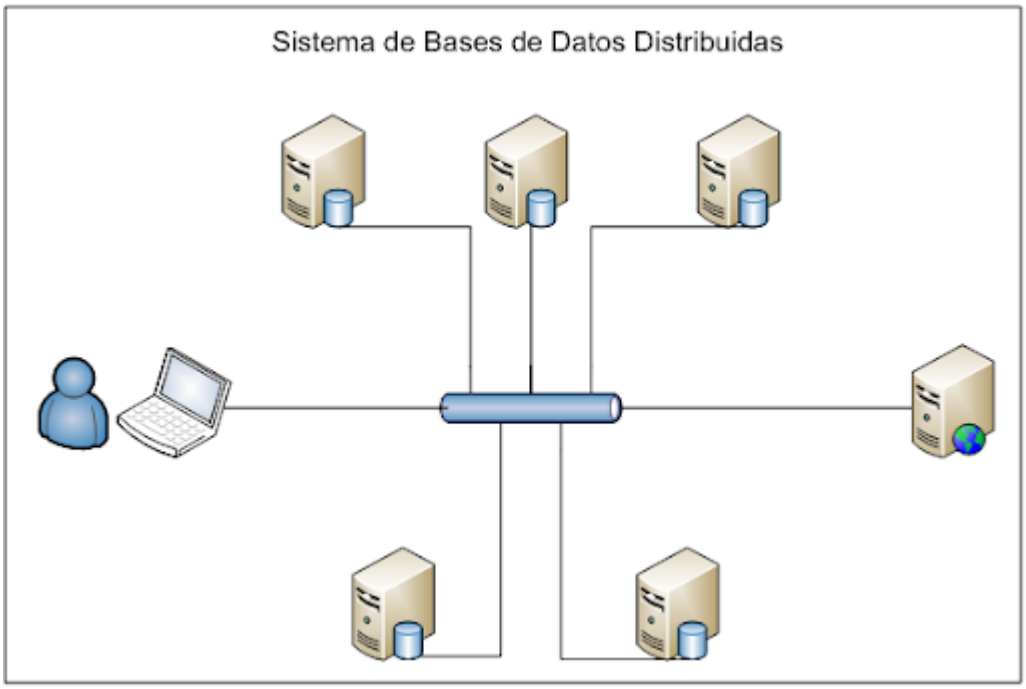
La necesidad de integrar información de varias fuentes y la evolución de las tecnologías de comunicaciones, han producido cambios muy importantes en los sistemas de bases de datos. La respuesta a estas nuevas necesidades y evoluciones se materializa en los sistemas de bases de datos distribuidas.

- **Base de datos distribuida (BDD)**: es un conjunto de múltiples bases de datos lógicamente relacionadas las cuales se encuentran distribuidas entre diferentes nodos interconectados por una red de comunicaciones.
- **Sistema de bases de datos distribuida (SBDD)**: es un sistema en el cual múltiples sitios de bases de datos están ligados por un sistema de comunicaciones, de tal forma que, un usuario en cualquier sitio puede acceder los datos en cualquier parte de la red exactamente como si los datos estuvieran almacenados en su sitio propio.
- **Sistema gestor de bases de datos distribuida (SGBDD)**: es aquel que se encarga del manejo de la BDD y proporciona un mecanismo de acceso que hace que la distribución sea transparente a los usuarios. El término transparente significa que la aplicación trabajaría, desde un punto de vista lógico, como si un solo SGBD ejecutado en una sola máquina, administrara esos datos.

Un SGBDD desarrollará su trabajo a través de un conjunto de sitios o nodos, que poseen un sistema de procesamiento de datos completo con una base de datos local, un sistema de gestor de bases de datos e interconcetados entre sí. Si estos nodos están dispersos geográficamente se interconetarán a través de una red de área amplia o WAN, pero si se encuentran en edificios relativamente cercanos, pueden estar interconectados por una red de área local o LAN. Este tipo de sistemas es utilizado en: organizaciones con

estructura descentralizada, industrias de manufactura con múltiples sedes (automoción), aplicaciones militares, lineas aéreas, cadenas hoteleras, servicios bancarios, etc.

A día de hoy, en esta distribución de nodos se utilizan tecnologías como el *Cloud computing*, de forma que éstos se encuentran directamente en la nube, y posiblemente en ubicaciones físicas dispersas.



Ventajas	Desventajas
El acceso y procesamiento de los datos es más rápido ya que varios nodos comparten carga de trabajo.	La probabilidad de violaciones de seguridad es creciente si no se toman las precauciones debidas.
Desde una ubicación puede accederse a información alojada en diferentes lugares.	Existe una complejidad añadida que es necesaria para garantizar la coordinación apropiada entre los nodos.
Los costes son inferiores a los de las bases de datos centralidas	La inversión inicial es menor, pero el mantenimiento y el control puede resultar costoso.
Existe cierta tolerancia a fallos. Mediante la replicación, si un nodo deja de funcionar el sistema completo no deja de funcionar.	Dado que los datos pueden estar replicados, el control de concurrencia y los mecanismos de recuperación son mucho más complejos que en un sistema centralizado.
El enfoque distribuido de las bases de datos se adapta más naturalmente a la estructura de las organizaciones. Permiten la incorporación de nodos de forma flexible y fácil.	El intercambio de mensajes y el cómputo adicional necesario para conseguir la coordinación entre los distintos nodos constituyen una forma de sobrecarga que no surge en los sistemas centralizados.
Aunque los nodos están interconectados, tienen independencia local.	Dada la complejidad del procesamiento entre nodos es difícil asegurar la corrección de los algoritmos, el funcionamiento correcto durante un fallo o la recuperación.

6.1 Fragmentación

Sabemos que en los sistemas de bases de datos distribuidas la información se encuentra repartida en varios lugares. La forma de extraer los datos consultados puede realizarse mediante la fragmentación de distintas tablas pertenecientes a distintas bases de datos que se encuentran en diferentes servidores. El problema de fragmentación se refiere al particionamiento de la información para distribuir cada parte a los diferentes sitios de la red.

Pero hay que tener en cuenta el grado de fragmentación que se aplicará, ya que éste es un factor determinamente a la hora de la ejecución de consultas. Si no existe fragmentación, se tomarán las relaciones o tablas como la unidad de fragmentación. Pero también puede fragmentarse a nivel de tupla (fila o registro) o a nivel de atributo (columna o campo) de una tabla. No será adecuado un grado de fragmentación nulo, ni tampoco un grado de fragmentación demasiado alto. El grado de fragmentación deberá estar equilibrado y dependerá de las particularidades de las aplicaciones que utilicen dicha base de datos. Concretando, el objetivo de la fragmentación es encontrar un nivel de particionamiento adecuado en el rango que va desde tuplas o atributos hasta relaciones completas.

Cuando se lleva a cabo una fragmentación, existen tres reglas fundamentales a cumplir:

- **Completitud**. Si una relación R se descompone en fragmentos R1, R2, ..., Rn, cada elemento de datos que pueda encontrarse en R deberá poder encontrarse en uno o varios fragmentos Ri.
- **Reconstrucción**. Si una relación R se descompone en una serie de fragmentos R1, R2, ..., Rn, la reconstrucción de la relación a partir de sus fragmentos asegura que se preservan las restricciones definidas sobre los datos.
- **Disyunción**. Si una relación R se descompone verticalmente, sus atributos primarios clave normalmente se repiten en todos sus fragmentos.

Existen tres tipos de fragmentación:

- **Fragmentación horizontal**: La fragmentación horizontal se realiza sobre las tuplas de la relación, dividiendo la relación en subrelaciones que contienen un subconjunto de las tuplas que alberga la primera.
- **Fragmentación vertical**: La fragmentación vertical, en cambio, se basa en los atributos de la relación para efectuar la división. Una relación R produce fragmentos R1, R2, ..., Rr, cada uno de los cuales contiene un subconjunto de los atributos de R así como la llave primaria de R. El objetivo de la fragmentación vertical es particionar una relación en un conjunto de relaciones más pequeñas de manera que varias de las aplicaciones de usuario se ejecutarán sobre un fragmento. En este contexto, una fragmentación óptima es aquella que produce un esquema de fragmentación que minimiza el tiempo de ejecución de las consultas de usuario. La fragmentación vertical es más complicada que la horizontal, ya que existe un gran número de alternativas para realizarla.
- **Fragmentación Híbrida o mixta**: Podemos combinar ambas, utilizando por ello la denominada fragmentación mixta. Si tras una fragmentación vertical se lleva a cabo otra horizontal, se habla de la fragmentación mixta (HV). Para el caso contrario, estaremos ante una fragmentación (VH). Para representar los dos tipos de fragmentación, se utilizan los árboles.

> Si quieres saber más sobre bases de datos distribuídas, puedes acceder al siguiente enlace https://es.wikipedia.org/wiki/Base_de_datos_distribuida

Bibliografía

1. Jorge Sánchez. Manual de Gestión de Bases de Datos. (1) Sistemas Gestores de Bases de Datos. https://jorgesanchez.net/manuales/gbd/sgbd.html
2. José Luis Comesaña. Apuntes Desarrollo de Aplicaciones Web. Bases de Datos. Tema 1. https://github.com/statickidz/TemarioDAW/blob/master/BBDD/BD01.pdf