Base de Dados

El Problema De Los Datos

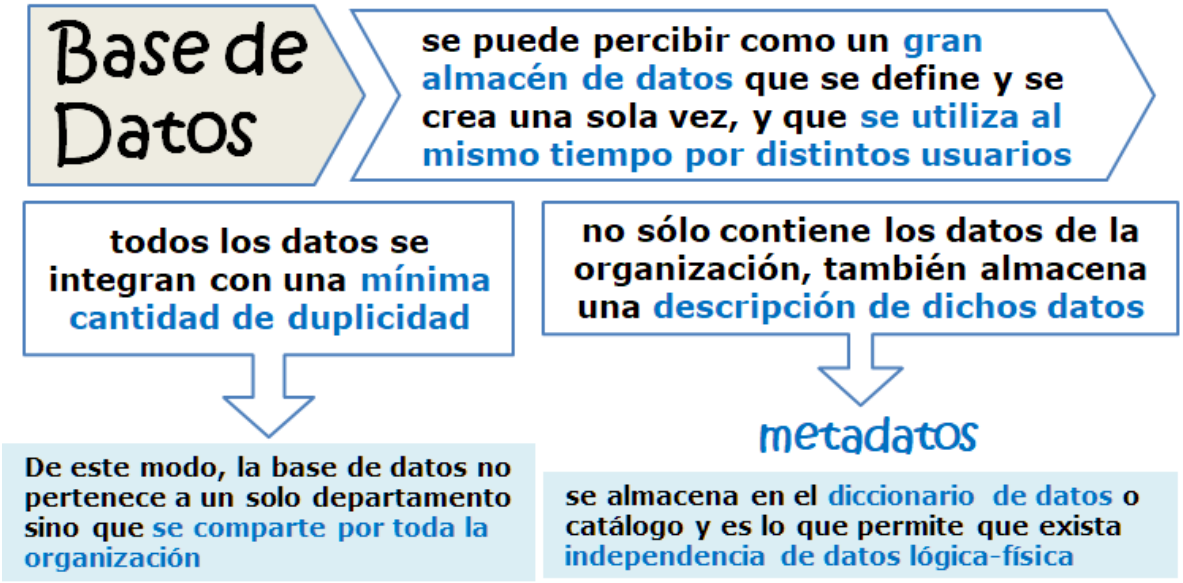
En informática se conoce como **DATO** a cualquier elemento informativo que tenga relevancia para el sistema. Desde el inicio de la informática se ha reconocido al dato como al elemento fundamental de trabajo en un ordenador. Por ello se han realizado numerosos estudios y aplicaciones para mejorar la gestión que desde las computadoras se realiza de los datos.

Inicialmente los datos que se necesitaba almacenar y gestionar eran pocos, pero poco a poco han ido creciendo. En la actualidad las numerosas aplicaciones de Internet han producido enormes sistemas de información que incluso para poder gestionarles requieren decenas de máquinas haciendo la información accesible desde cualquier parte del planeta y en un tiempo rápido. Eso ha requerido que la **ciencia de las bases de datos** esté en continua renovación para hacer frente a esas enormes necesidades.

Una **base de datos** es un **conjunto de datos almacenados** en memoria externa que están organizados mediante una **estructura de datos**. Cada base de datos ha sido **diseñada para satisfacer los requisitos de información** de una empresa u otro tipo de organización, como por ejemplo, una universidad o un hospital.

Antes de existir las bases de datos se trabajaba con sistemas de ficheros. Los sistemas de ficheros surgieron al informatizar el manejo de los archivadores manuales para proporcionar un acceso más eficiente a los datos almacenados en los mismos. Un sistema de ficheros sigue un modelo descentralizado, en el que cada departamento de la empresa almacena y gestiona sus propios datos mediante una serie de programas de aplicación escritos especialmente para él. Estos programas son totalmente independientes entre un departamento y otro, y se utilizan para introducir datos, mantener los ficheros y generar los informes que cada departamento necesita. Es importante destacar que en los sistemas de ficheros, tanto la estructura física de los ficheros de datos como la de sus registros, están definidas dentro de los programas de aplicación. Cuando en una empresa se trabaja con un sistema de ficheros, los departamentos no comparten información ni aplicaciones, por lo que los datos comunes deben estar duplicados en cada uno de ellos. Esto puede originar inconsistencias en los datos. Se produce una inconsistencia cuando copias de los mismos datos no coinciden: dos copias del domicilio de un cliente pueden no coincidir si sólo uno de los departamentos que lo almacenan ha sido informado de que el domicilio ha cambiado.

Otro inconveniente que plantean los sistemas de ficheros es que cuando los datos se separan en distintos ficheros, es más complicado acceder a ellos, ya que el programador de aplicaciones debe sincronizar el procesamiento de los distintos ficheros implicados para garantizar que se extraen los datos correctos. Además, ya que la estructura física de los datos se encuentra especificada en los programas de aplicación, cualquier cambio en dicha estructura es difícil de realizar. El programador debe identificar todos los programas afectados por el cambio, modificarlos y volverlos a probar, lo que cuesta mucho tiempo y está sujeto a que se produzcan errores. A este problema, tan característico de los sistemas de ficheros, se le denomina también falta de independencia de datos lógica-física.



Para poder almacenar datos y cada vez más datos, el ser humano ideó nuevas herramientas archivos, cajones, carpetas y fichas en las que se almacenaban los datos. Antes de la aparición del ordenador, el tiempo requerido para manipular estos datos era enorme. Sin embargo el proceso de aprendizaje era relativamente sencillo ya que se usaban elementos que el usuario reconocía perfectamente. Por esa razón, la informática adaptó sus herramientas para que los elementos que el usuario maneja en el ordenador se parezcan a los que utilizaba manualmente. Así en informática se sigue hablado de ficheros, formularios, carpetas, directorios,....

SISTEMA GESTOR DE BASES DE DATOS

Un sistema gestor de bases de datos o **SGBD** (aunque se suele utilizar más a menudo las siglas **DBMS** procedentes del inglés, *Data Base Management System*) es el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos.

Un Sistema De Bases De Datos sirve para integrar los datos. Lo componen los siguientes elementos:

- **Hardware.** Máquinas en las que se almacenan las bases de datos. Incorporan unidades de almacenamiento masivo para este fin.
- **Software.** Es el sistema gestor de bases de datos. El encargado de administrar las bases de datos.
- **Datos.** Incluyen los datos que se necesitan almacenar y los **metadatos** que son datos que sirven para describir lo que se almacena en la BASE DE DATOS.
- **Usuarios.** Personas que manipulan los datos del sistema. Hay tres categorías:
  - ***Usuarios finales.*** Aquellos que utilizan datos de la base de datos para su trabajo cotidiano que no tiene por qué tener que ver con la informática. Normalmente no utilizan la base de datos directamente, si no que utilizan aplicaciones creadas para ellos a fin de facilitar la manipulación de los datos. Estos usuarios sólo acceden a ciertos datos.
  - ***Desarrolladores.*** Analistas y programadores encargados de generar aplicaciones para los usuarios finales.
  - ***Administradores.*** También llamados DBA (*Data Base Administrator*), se encargan de gestionar las bases de datos.

Hay que tener en cuenta que las necesidades de los usuarios son muy diferentes en función del tipo de usuario que sean: a los finales les interesa la facilidad de uso, a los desarrolladores la potencia y flexibilidad de los lenguajes incorporados del sistema de bases de datos, a los administradores herramientas de gestión avanzada para la base de datos.

Personas en el entorno de las bases de datos

Hay **cuatro grupos de personas** que intervienen en el entorno de una base de datos: el *administrador de la base de datos*, los *diseñadores de la base de datos*, los *programadores de aplicaciones* y los *usuarios*.

El **administrador de la base de datos** se encarga de la implementación física de la base de datos: escoge los tipos de los ficheros de datos y de los índices que deben crearse, determina dónde deben ubicarse ficheros e índices y, en general, toma las decisiones relativas al almacenamiento físico en función de las posibilidades que le ofrezca el SGBD con el que trabaje. Además, el administrador de la base de datos se encarga de establecer la política de seguridad y del acceso concurrente. También se debe preocupar de que el sistema se encuentre siempre operativo y procurar que los usuarios y las aplicaciones obtengan buenas prestaciones. El administrador debe conocer muy bien el SGBD con el que trabaja, así como el equipo informático sobre el que esté funcionando.

Los **diseñadores de la base de datos** realizan el diseño de la base de datos, debiendo identificar los datos, las relaciones entre ellos y las restricciones sobre los datos y sobre sus relaciones. El diseñador de la base de datos debe tener un profundo conocimiento de los datos de la empresa y también debe conocer sus reglas de negocio. Las reglas de negocio describen las características principales sobre el comportamiento de los datos tal y como las ve la empresa. Para obtener un buen resultado, el diseñador de la base de datos debe implicar en el proceso a todos los usuarios de la base de datos, tan pronto como sea posible.

Una vez se ha diseñado e implementado la base de datos, los **programadores de aplicaciones** se encargan de implementar los programas de aplicación que servirán a los usuarios finales. Estos programas de aplicación son los que permiten consultar datos, insertarlos, actualizarlos y eliminarlos. Estos programas se escriben mediante lenguajes de tercera generación o de cuarta generación.

Los **usuarios finales** son los clientes de la base de datos: la base de datos ha sido diseñada e implementada, y está siendo mantenida, para satisfacer sus requisitos en la gestión de su información.

Estructura de la Base de Datos

El **modelo** seguido con los *Sistemas De Bases De Datos* es muy similar al modelo que se sigue en la actualidad para el desarrollo de programas con lenguajes orientados a objetos, *en donde se da una implementación interna de un objeto y una especificación externa separada*. *Los usuarios del objeto sólo ven la especificación externa y no se deben preocupar de cómo se implementa internamente el objeto. Una ventaja de este modelo, conocido como abstracción de datos, es que se puede cambiar la implementación interna de un objeto sin afectar a sus usuarios ya que la especificación externa no se ve alterada*. Del mismo modo, los Sistemas De Bases De Datos separan la *definición de la estructura física de los datos de su estructura lógica*, y *almacenan esta definición en la base de datos*. Todo esto es gracias a la existencia del SGBD, que se sitúa **entre la base de datos y los programas de aplicación.**

Las bases de datos están compuestas (como ya se han comentado), de **datos** y de **metadatos**. Los metadatos son datos (valga la redundancia) que **sirven para *especificar la estructura de la base de datos***; por ejemplo qué tipo de datos se almacenan (si son texto o números o fechas ...), qué nombre se le da a cada dato (nombre, apellidos,...), cómo están agrupados, cómo se relacionan,....

De este modo se producen **dos visiones** de la base de datos:

- **Estructura LÓGICA.** Indica **la composición y distribución teórica** de la base de datos. Sirve para que las aplicaciones puedan utilizar los elementos de la base de datos sin saber realmente cómo se están almacenando. Es una estructura que permite idealizar a la base de datos. Sus elementos son objetos, entidades, nodos, relaciones, enlaces,... que realmente no tienen presencia real en la física del sistema. Por ello para acceder a los datos tiene que haber una posibilidad de traducir la estructura lógica en la estructura física.
- **Estructura FÍSICA**. Es la estructura de **los datos tan cual se almacenan** en las unidades de disco. La correspondencia entre la estructura lógica y la física se almacena en la base de datos (en los metadatos).

El **éxito del DBMS** reside en mantener la seguridad e integridad de los datos.

Lógicamente tiene que proporcionar herramientas a los distintos usuarios. Entre las herramientas que proporciona están:

- Herramientas para la ***creación*** y ***especificación de los datos***. Así como la estructura de la base de datos.

- Herramientas para ***administrar*** y ***crear la estructura física*** requerida en las unidades de almacenamiento.
- Herramientas para la ***manipulación de los datos*** de las bases de datos, para añadir, modificar, suprimir o consultar datos.
- Herramientas de ***recuperación*** en caso de desastre
- Herramientas para la creación de ***copias de seguridad***
- Herramientas para la ***gestión de la comunicación*** de la base de datos

Funciones de un DBMS

1. **Función de DESCRIPCIÓN**. Sirve para describir los datos, sus relaciones y sus condiciones de acceso e integridad. Además del control de vistas de usuarios y de la especificación de las características físicas de la base de datos. Para poder realizar todas estas operaciones se utiliza un **lenguaje de definición de datos** o **DDL**.
2. **Función de MANIPULACIÓN**. Permite buscar, añadir, suprimir y modificar datos de la base de datos. El DBMS proporciona una **lenguaje de manipulación de datos** (**DML**) para realizar esta función.
3. **Función de CONTROL**. Incorpora las funciones que permiten una buena comunicación con la base de datos. Además proporciona al DBA los procedimientos necesarios para realizar su labor.

Generalmente, un SGBD proporciona los servicios que se citan a continuación:

✓ El SGBD permite la definición de la base de datos mediante un **lenguaje de definición de datos.** Este lenguaje permite especificar la estructura y el tipo de los datos, así como las restricciones sobre los datos.

✓ El SGBD permite la inserción, actualización, eliminación y consulta de datos mediante un **lenguaje de manejo de datos**. El hecho de disponer de un lenguaje para realizar consultas reduce el problema de los sistemas de ficheros, en los que el usuario tiene que trabajar con un conjunto fijo de consultas, o bien, dispone de un gran número de programas de aplicación costosos de gestionar. Hay dos tipos de lenguajes de manejo de datos: los procedurales y los no procedurales. Estos dos tipos se distinguen por el modo en que acceden a los datos.

Los lenguajes procedurales manipulan la base de datos registro a registro, mientras que los no procedurales operan sobre conjuntos de registros. En los lenguajes procedurales se especifica qué operaciones se debe realizar para obtener los datos resultado, mientras que en los lenguajes no procedurales se especifica qué datos deben obtenerse sin decir cómo hacerlo. El lenguaje no procedural más utilizado es el SQL (Structured Query Language) que, de hecho, es un estándar y es el lenguaje de los SGBD relacionales.

✓ El SGBD proporciona un acceso controlado a la base de datos mediante:

- Un **sistema de seguridad**, de modo que los usuarios no autorizados no puedan acceder a la base de datos.
- Un **sistema de integridad** que mantiene la integridad y la consistencia de los datos.
- Un **sistema de control de concurrencia** que permite el acceso compartido a la base de datos.

- Un **sistema de control de recuperación** que restablece la base de datos después de que se produzca un fallo del hardware o del software.
- Un **diccionario de datos** o catálogo, accesible por el usuario, que contiene la descripción de los datos de la base de datos.

✓ A diferencia de los sistemas de ficheros, en los que los programas de aplicación trabajan directamente sobre los ficheros de datos, el SGBD se ocupa de la estructura física de los datos y de su almacenamiento.

Con esta funcionalidad, el SGBD se convierte en una herramienta de gran utilidad. Sin embargo, desde el punto de vista del usuario, se podría discutir que los SGBD han hecho las cosas más complicadas, ya que ahora los usuarios ven más datos de los que realmente quieren o necesitan, puesto que ven la base de datos completa. Conscientes de este problema, los SGBD proporcionan un **mecanismo de vistas** que permite que cada usuario tenga su propia vista o visión de la base de datos. El lenguaje de definición de datos permite definir vistas como subconjuntos de la base de datos.

Todos los SGBD no presentan la misma funcionalidad, depende de cada producto.

En general, los grandes SGBD multiusuario ofrecen todas las funciones que se acaban de citar e incluso más. Los sistemas modernos son conjuntos de programas extremadamente complejos y sofisticados, con millones de líneas de código y con una documentación consistente en varios volúmenes. Lo que se pretende es proporcionar un sistema que permita *gestionar cualquier tipo de requisitos y que tenga un 100% de fiabilidad ante cualquier tipo de fallo*.

Los SGBD están en continua evolución, tratando de satisfacer los requisitos de todo tipo de usuarios. Por ejemplo, muchas aplicaciones de hoy en día necesitan almacenar imágenes, vídeo, sonido, etc. Para satisfacer a este mercado, los SGBD deben evolucionar. Conforme vaya pasando el tiempo, irán surgiendo nuevos requisitos, por lo que los SGBD nunca permanecerán estáticos.

Los sistemas de bases de datos presentan numerosas **ventajas** gracias, fundamentalmente, a la integración de datos y a la interfaz común que proporciona el SGBD. Estas ventajas se describen a continuación.

✓ Control sobre la redundancia de datos. Los sistemas de ficheros almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además de provocar faltas de consistencia de datos (copias que no coinciden). En los sistemas de bases de datos todos estos ficheros están integrados, por lo que no se almacenan varias copias de los mismos datos. Sin embargo, en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos, o bien es necesaria para mejorar las prestaciones.

✓ Control sobre la consistencia de datos. Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantengan consistentes. Desgraciadamente, no todos los SGBD de hoy en día se encargan de mantener automáticamente la consistencia.

- ✓ Compartición de datos. En los sistemas de ficheros, los ficheros pertenecen a los departamentos que los utilizan, pero en los sistemas de bases de datos, la base de datos pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados. Además, las nuevas aplicaciones que se vayan creando pueden utilizar los datos de la base de datos existente.
- ✓ Mantenimiento de estándares. Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio; pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.
- ✓ Mejora en la integridad de datos. La integridad de la base de datos se refiere a la validez de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se encargará de mantenerlas.
- ✓ Mejora en la seguridad. La seguridad de la base de datos consiste la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros. Sin embargo, los SGBD permiten mantener la seguridad mediante el establecimiento de claves para identificar al personal autorizado a utilizar la base de datos. Las autorizaciones se pueden realizar a nivel de operaciones, de modo que un usuario puede estar autorizado a consultar ciertos datos pero no a actualizarlos, por ejemplo.
- ✓ Mejora en la accesibilidad a los datos. Muchos SGBD proporcionan lenguajes de consulta o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.
- ✓ Mejora en la productividad. El SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros. A nivel básico, el SGBD proporciona todas las rutinas de manejo de ficheros típicas de los programas de aplicación. El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel. Muchos SGBD también proporcionan un entorno de cuarta generación consistente en un conjunto de herramientas que simplifican, en gran medida, el desarrollo de las aplicaciones que acceden a la base de datos. Gracias a estas herramientas, el programador puede ofrecer una mayor productividad en un tiempo menor.
- ✓ Mejora en el mantenimiento gracias a la independencia de datos. En los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de aplicación que los manejan. Esto hace que los programas sean dependientes de los datos, de modo que un cambio en su estructura, o un cambio en el modo en que se almacena en disco, requiere cambios importantes en los programas cuyos datos se ven afectados. Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos, gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.
- ✓ Aumento de la concurrencia. En algunos sistemas de ficheros, si hay varios usuarios que pueden acceder simultáneamente a un mismo fichero, es posible que el acceso interfiera entre ellos de modo que se pierda información o, incluso, que se pierda la integridad. La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y pueden garantizar que no ocurran problemas de este tipo.
- ✓ Mejora en los servicios de copias de seguridad y de recuperación ante fallos. Muchos sistemas de ficheros dejan que sea el usuario quien proporcione las medidas necesarias para proteger los datos ante fallos en el sistema o en las aplicaciones. Los usuarios tienen que hacer copias de seguridad cada día, y si se produce algún fallo, utilizar estas copias para restaurarlos.

En este caso, todo el trabajo realizado sobre los datos desde que se hizo la última copia de seguridad se pierde y se tiene que volver a realizar. Sin embargo, los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo perdido cuando se produce un fallo.

La integración de los datos y la existencia del SGBD también plantean ciertos **inconvenientes**, como los que se citan a continuación.

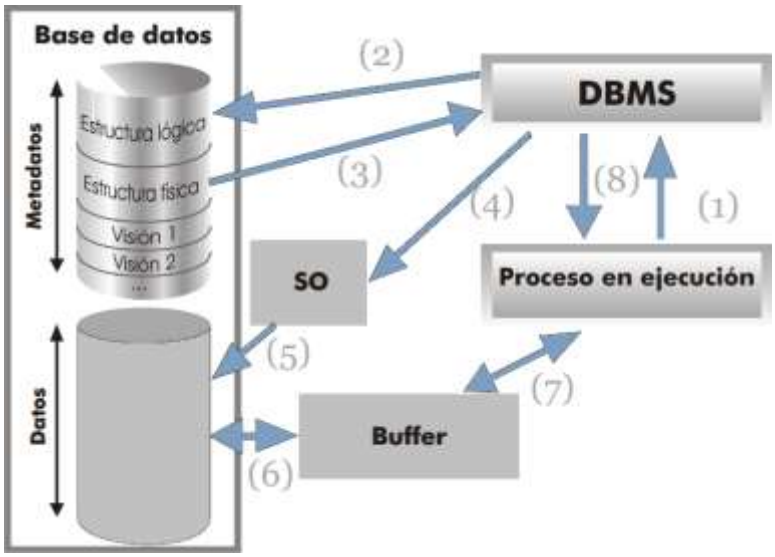
- ✓ Alta complejidad. Los SGBD son conjuntos de programas muy complejos con una gran funcionalidad. Es preciso comprender muy bien esta funcionalidad para poder sacar un buen partido de ellos.
- ✓ Gran tamaño. Los SGBD son programas complejos y muy extensos que requieren una gran cantidad de espacio en disco y de memoria para trabajar de forma eficiente.
- ✓ Coste económico del SGBD. El coste de un SGBD varía dependiendo del entorno y de la funcionalidad que ofrece. Por ejemplo, un SGBD para un ordenador personal puede costar 500 e, mientras que un SGBD para un sistema multiusuario que dé servicio a cientos de usuarios puede costar entre 10000 y 100000 e. Además, hay que pagar una cuota anual de mantenimiento que suele ser un porcentaje del precio del SGBD. En los últimos años han surgido SGBD libres (open source) que ofrecen una gran funcionalidad y muy buenas prestaciones.
- ✓ Coste del equipamiento adicional. Tanto el SGBD, como la propia base de datos, pueden hacer que sea necesario adquirir más espacio de alma cenamiento. Además, para alcanzar las prestaciones deseadas, es posible que sea necesario adquirir una máquina más grande o una máquina que se dedique solamente al SGBD. Todo esto hará que la implantación de un sistema de bases de datos sea más cara.
- ✓ Coste de la conversión. En algunas ocasiones, el coste del SGBD y el coste del equipo informático que sea necesario adquirir para su buen funcionamiento es insignificante comparado al coste de convertir la aplicación actual en un sistema de bases de datos. Este coste incluye el coste de enseñar a la plantilla a utilizar estos sistemas y, probablemente, el coste del personal especializado para ayudar a realizar la conversión y poner en marcha el sistema. Este coste es una de las razones principales por las que algunas empresas y organizaciones se resisten a cambiar su sistema actual de ficheros por un sistema de bases de datos.
- ✓ Prestaciones. Un sistema de ficheros está escrito para una aplicación es pecífica, por lo que sus prestaciones suelen ser muy buenas. Sin embargo, los SGBD están escritos para ser más generales y ser útiles en muchas aplicaciones, lo que puede hacer que algunas de ellas no sean tan rápidas como antes.
- ✓ Vulnerable a los fallos. El hecho de que todo esté centralizado en el SGBD hace que el sistema sea más vulnerable ante los fallos que puedan producirse.

Funcionamiento de los DBMS

Los datos son responsabilidad del DBMS, por lo que cualquier acceso debe ser realizado por éste. Lógicamente el DBMS va a acabar comunicándose con el Sistema Operativo ya que el acceso a los ficheros de datos implica utilizar funciones del sistema operativo.

Interacción completa entre un proceso de usuario y un sistema gestor de bases de datos:

1. El proceso lanzado por el usuario llama al DBMS indicando la porción de la base de datos que se desea tratar
2. El DBMS traduce la llamada a términos del esquema lógico de la base de datos. Accede al esquema lógico comprobando derechos de acceso y la traducción física
3. El DBMS obtiene el esquema físico
4. El DBMS traduce la llamada a los métodos de acceso del Sistema Operativo que permiten acceder a los datos requeridos
5. El Sistema Operativo accede a los datos tras traducir las órdenes dadas por el DBMS
6. Los datos pasan del disco a una memoria intermedia o buffer. En ese buffer se almacenarán los datos según se vayan recibiendo
7. Los datos pasan del buffer al área de trabajo del usuario (ATU) del proceso del usuario.
8. El DBMS devuelve indicadores en los que manifiesta si ha habido errores o advertencias a tener en cuenta. Esto se indica al área de comunicaciones del proceso de usuario. Si las indicaciones son satisfactorias, los datos de la ATU serán utilizables por el proceso de usuario.



Esquema completo de la comunicación entre procesos de usuario, DBMS y Sistema Operativo

El proceso que realiza un SGBD está en realidad formado por varias capas que actúan como interfaces entre el usuario y los datos.

Para llegar a los datos hay que pasar una serie de capas que desde la parte más externa poco a poco van entrando más en la realidad física de la base de datos

Modelo de referencia de las facilidades de usuario

- ✓ **Facilidades de usuario.** Son las herramientas que proporciona el SGBD a los usuarios para permitir un acceso más sencillo a los datos. Actúan de interfaz entre el usuario y la base de datos, y son el único elemento que maneja el usuario. Son, en definitiva, las páginas web y las aplicaciones con las que los usuarios manejan la base de datos. Permite abstraer la realidad de la base de datos a las usuarias y usuarios, mostrando la información de una forma más humana.
- ✓ **Capa de acceso a datos.** La capa de acceso a datos es la que permite comunicar a las aplicaciones de usuario con el diccionario de datos. Es un software (un driver o controlador en realidad) que se encarga traducir las peticiones del usuario para que lleguen de forma correcta a la base de datos y ésta pueda responder de forma adecuada.
- ✓ **Diccionario de datos.** Se trata del elemento que posee todos los metadatos. Gracias a esta capa las solicitudes de los clientes (que son conceptuales antes de llegar aquí) se traducen en instrucciones que hacen referencia al esquema interno de la base de datos.
- ✓ **Núcleo.** El núcleo de la base de datos es la encargada de traducir todas las instrucciones requeridas y prepararlas para su correcta interpretación por parte del sistema. Realiza la traducción física de las peticiones.
- ✓ **Sistema operativo.** Es una capa externa al software SGBD pero es la única capa que realmente accede a los datos en sí. En realidad los SGBD no acceden directamente al disco, sino que piden al Sistema Operativo que lo haga

Independencia lógico/física

El esquema conceptual debe ser absolutamente independiente del físico. Esto significa:

- ***Independencia física de los datos***. Aunque el esquema físico cambie, el esquema 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 esquema conceptual permanece invariable.
- ***Independencia lógica de los datos***. Significa que aunque se modifique el esquema conceptual, la vista que poseen las aplicaciones (los esquemas externos) no serán afectados.

ARQUITECTURA DE LOS SGBD. ESTÁNDARES

Desde la aparición de los primeros gestores de base de datos se intentó llegar a un acuerdo para que hubiera una estructura común para todos ellos, a fin de que el aprendizaje y manejo de este software fuera más provechoso y eficiente.

El acuerdo nunca se ha conseguido del todo, no hay estándares aceptados del todo. Aunque sí hay unas cuentas propuestas de estándares que sí funcionan como tales.

✓ **Modelo ANSI/X3/SPARC**

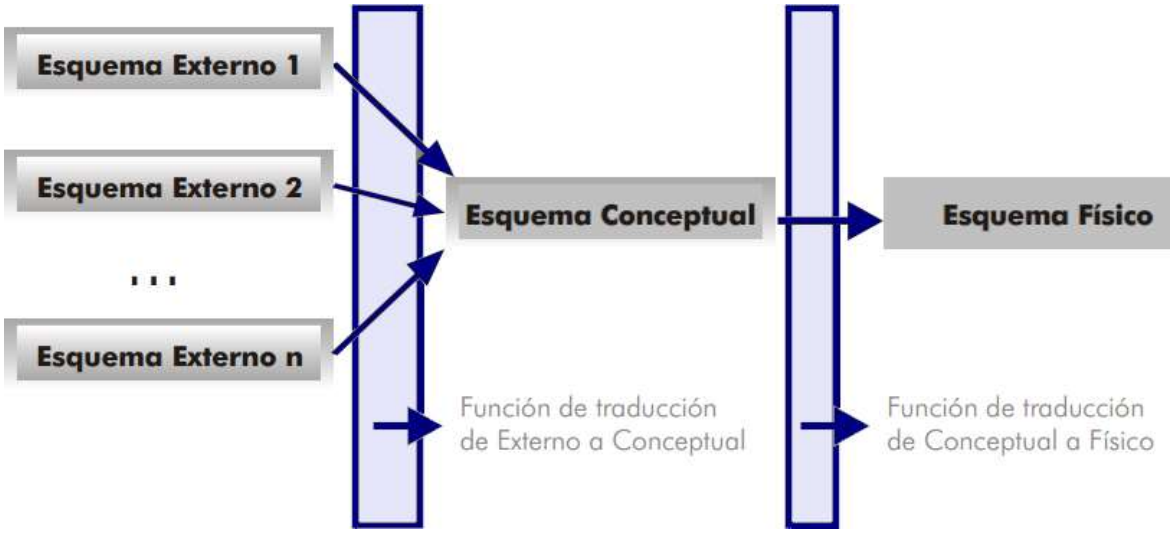
El organismo ANSI ha marcado la referencia para la construcción de SGBD. El modelo definido por el grupo de trabajo SPARC se basa en estudios anteriores en los que se definían tres niveles de abstracción necesarios para gestionar una base de datos. ANSI profundiza más en esta idea y define cómo debe ser el proceso de creación y utilización de estos niveles. En definitiva el modelo ANSI es una propuesta teórica sobre cómo debe de funcionar un sistema gestor de bases de datos (sin duda, la propuesta más importante).

Una base de datos se puede ver de diferentes formas. Cada programa que accede a la base de datos manipula sólo ciertos datos y estructuras. Así cada programa posee una visión de la base de datos.

En el modelo ANSI se indica que hay tres modelos: externo, conceptual e interno. Se entiende por modelo, el conjunto de normas que permiten crear esquemas (diseños de la base de datos). Los esquemas externos reflejan la información preparada para el usuario final, el esquema conceptual refleja la unión de los datos y sus relaciones de la base de datos y el esquema interno representa el almacenamiento de los datos y sus formas de acceso. Propusieron tres niveles de abstracción en las bases de datos, de acuerdo con el siguiente esquema.

- **ESQUEMA EXTERNO**. Visión de la base de datos que ofrece cada aplicación. Lógicamente es distinta en cada aplicación. Representan vistas concretas de la base de datos.
- **ESQUEMA CONCEPTUAL**. Representación teórica de los datos y de sus relaciones. Representa la lógica de la base de datos. contiene la información lógica de la base de datos. Su estructuración y las relaciones que hay entre los datos

- **ESQUEMA FÍSICO**. Representa los datos según son almacenados en el medio físico (en los discos). contiene información sobre cómo están almacenados los datos en disco. Es el esquema más cercano a la organización real de los datos



Niveles ANSI/SPARC

El paso de un esquema a otro se realiza utilizando una interfaz o función de traducción. En su modelo, la ANSI no indica cómo se debe realizar esta función, sólo que debe existir.

NIVEL EXTERNO

Nivel de usuarios individuales donde cada usuario tendrá ciertos lenguajes a su disposición

- **Usuario Final**
  - Lenguaje de consulta (cuarta generación): SQL
  - Lenguaje de la aplicación (menús, comandos, sistemas ventanas, etc.)
- **Programador de aplicaciones**
  - Lenguaje de consulta (cuarta generación): SQL
  - Lenguaje de programación de alto nivel (C, PL/I, COBOL, PASCAL...)
- **DBA**
  - Todos los lenguajes utilizados por los anteriores usuarios.

NIVEL CONCEPTUAL

VISTA CONCEPTUAL: Representación de **TODO** el contenido de la Base de Datos.

- Representación abstracta de los datos.
- NO coincide con la representación física.
- NO coincide con vista externa.
- Vista formada por el conjunto de REGISTROS CONCEPTUALES que definen entidades y relaciones entre entidades.
  Registros conceptuales no han de coincidir con registros externos o registros internos.
- Vista definida mediante ESQUEMA CONCEPTUAL.

ESQUEMA CONCEPTUAL: Definición de cada tipo de registro conceptual.

- Definido mediante DDL CONCEPTUAL (*Data Definition Language*).
- Definición del contenido de información.
- No se han de incluir **relaciones con el nivel interno** (estructura de almacenamiento, técnicas de acceso, campos y registros físicos).

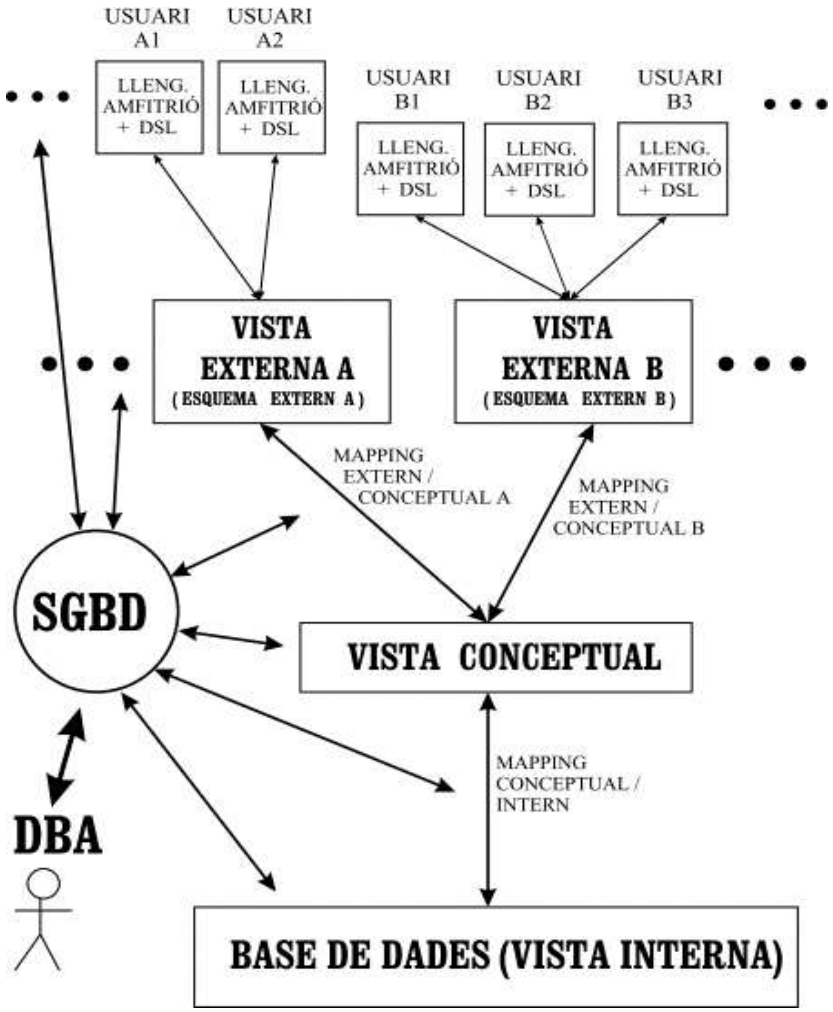
NIVEL INTERNO

VISTA INTERNA: Representación de bajo nivel de la Base de Datos.

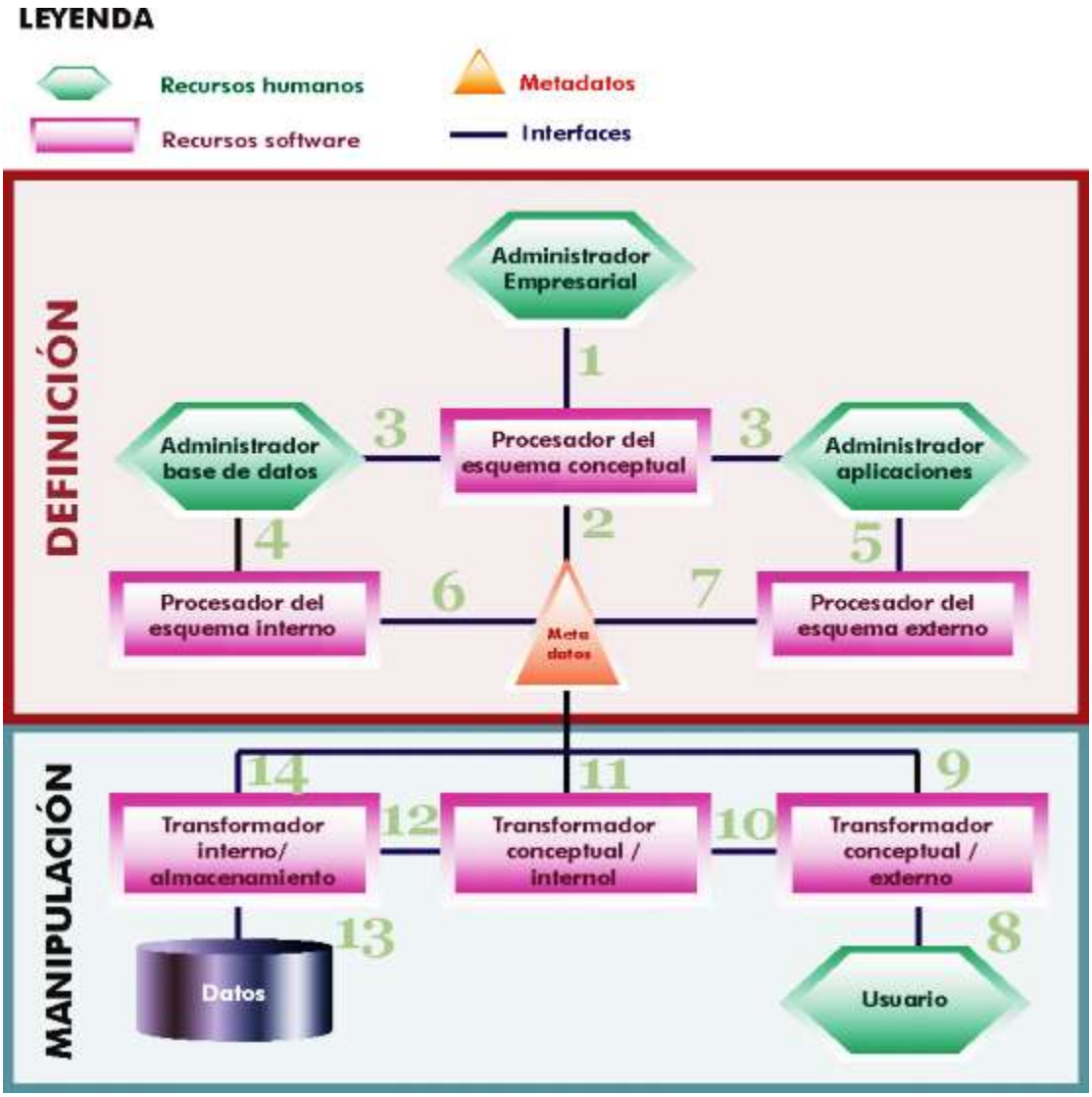
- Vista más cercana a la máquina, a un paso del nivel físico. No manipula registros físicos.
- Supone espacio lineal ∞ en direccionamiento. No se especifica correspondencia entre este espacio y los dispositivos lógicos.
- Vista compuesta de ocurrencias de diferentes tipos de REGISTROS INTERNOS.
- Vista definida mediante ESQUEMA INTERNO.

ESQUEMA INTERNO: Definiciones de los registros internos, índices y campos.

- Definido mediante DDL INTERNO.



ESQUEMA DE LA ARQUITECTURA ANSI/SPARC



Arquitectura ANSI, explicación clásica del funcionamiento de un Sistema Gestor de Bases de Datos incluyendo los recursos humanos implicados

La arquitectura completa está dividida en dos secciones, la zona de definición de datos y la de manipulación. Esa arquitectura muestra las funciones realizadas por humanos y las realizadas por programas.

En la fase de definición, una serie de interfaces permiten la creación de los metadatos que se convierten en el eje de esta arquitectura. La creación de la base de datos comienza con la elaboración del esquema conceptual realizándola el administrador de la empresa (actualmente es el diseñador, pero ANSI no lo llamó así).

Ese esquema se procesa utilizando un procesador del esquema conceptual (normalmente una herramienta CASE, interfaz 1 del dibujo anterior) que lo convierte en los metadatos (interfaz 2).

La interfaz 3 permite mostrar los datos del esquema conceptual a los otros dos administradores: el administrador de la base de datos y el de aplicaciones (el desarrollador). Mediante esta información construyen los esquemas internos y externos mediante las interfaces 4 y 5 respectivamente, los procesadores de estos esquemas almacenan la información correspondiente a estos esquemas en los metadatos (interfaces 6 y 7).

En la fase de manipulación el usuario puede realizar operaciones sobre la base de datos usando la interfaz 8 (normalmente una aplicación) esta petición es transformada por el transformador externo/conceptual que obtiene el esquema correspondiente ayudándose también de los metadatos (interfaz 9). El resultado lo convierte otro transformador en el esquema interno (interfaz 10) usando también la información de los metadatos (interfaz 11). Finalmente del esquema interno se pasa a los datos usando el último transformador (interfaz 12) que también accede a los metadatos (interfaz 13) y de ahí se accede a los datos (interfaz 14). Para que los datos se devuelvan al usuario en formato adecuado para él se tiene que hacer el proceso contrario (observar dibujo)

Diseño de bases de datos

La teoría siempre es algo tedioso, aunque intentaremos que resulte amena, ya que es necesaria. Aconsejo leer con atención estos capítulos. En ellos aprenderemos técnicas como el "modelado" y la "normalización" que nos ayudarán a diseñar bases de datos, mediante la aplicación de ciertas reglas muy sencillas. Aprenderemos a identificar claves y a elegir claves principales, a establecer interrelaciones, a seleccionar los tipos de datos adecuados y a crear índices.

Como siempre que emprendamos un nuevo proyecto, grande o pequeño, antes de lanzarnos a escribir código, crear tablas o bases de datos, hay que analizar el problema sobre el papel. En el caso de las bases de datos, pensaremos sobre qué tipo de información necesitamos guardar, o lo que es más importante: qué tipo de información necesitaremos obtener de la base de datos. En esto consiste el modelado de bases de datos.

El proceso de trasladar un problema del mundo real a un ordenador, usando bases de datos, se denomina *modelado*.

Para el modelado de bases de datos es necesario seguir un procedimiento determinado. Pero, cuando el problema a modelar es sencillo, con frecuencia estaremos tentados de pasar por alto, algunoS de los pasos, y crear directamente bases de datos y tablas. En el caso de las bases de datos, como en cualquier otra solución informática, esto es un gran error. Siempre será mejor seguir todos los pasos del diseño, esto nos ahorrará (con toda seguridad) mucho tiempo más adelante. Sobre todo si alguna vez tenemos que modificar la base de datos para corregir errores o para implementar alguna característica nueva, algo que sucede con mucha frecuencia.

Además, seguir todo el proceso nos facilitará una documentación necesaria para revisar o mantener la aplicación, ya sea por nosotros mismos o por otros administradores o programadores.

La primera fase del diseño de una aplicación (la base de datos, generalmente, es parte de una aplicación), consiste en hablar con el cliente para saber qué quiere, y qué necesita realmente.

Esto es una tarea ardua y difícil. Generalmente, los clientes no saben demasiado sobre programación y sobre bases de datos, de modo que normalmente, no saben qué pueden pedir. De hecho, lo más habitual es que ni siquiera sepan qué es lo que necesitan.

Los modelos conceptuales ayudan en esta fase del proyecto, ya que facilitan una forma clara de ver el proceso en su totalidad, puesto que se trata de una representación gráfica. Además, los modelos conceptuales no están orientados a ningún sistema físico concreto: tipo de ordenador, sistema operativo, SGBD, etc. Ni siquiera tienen una orientación informática clara, podrían servir igualmente para explicar a un operario cómo funciona el proceso de forma manual. Esto facilita que sean comprensibles para personas sin conocimientos de programación.

Además de consultar con el cliente, una buena técnica consiste en observar el funcionamiento del proceso que se quiere informatizar o modelar. Generalmente esos procesos ya se realizan, bien de una forma manual, con ayuda de libros o ficheros; o bien con un pequeño apoyo ofimático.

Con las bases de datos lo más importante es observar qué tipo de información se necesita, y que parte de ella se necesita con mayor frecuencia. Por supuesto, modelar ciertos procesos puede proporcionarnos ayudas extra sobre el proceso manual, pero no debemos intentar que nuestra aplicación lo haga absolutamente todo, sino principalmente, aquello que es realmente necesario.

Cuando los programas se crean sin un cliente concreto, ya sea porque se pretende crear un producto para uso masivo o porque sólo lo vamos a usar nosotros, el papel del cliente lo jugaremos nosotros mismos, pero la experiencia nos enseñará que esto no siempre es una ventaja. Es algo parecido a los que pasa con los abogados o los médicos. Se suele decir que *"el abogado que se defiende a si mismo tiene un necio por cliente"*. En el caso de los programadores esto no es tan exagerado; pero lo cierto es que, demasiadas veces, los programadores somos nuestros peores clientes.

Toda esta información recogida del cliente debe formar parte de la documentación. Nuestra experiencia como programadores debe servir, además, para ayudar y guiar al cliente. De este modo podemos hacerle ver posibles "cuellos de botella", excepciones, mejoras en el proceso, etc. Así mismo, hay que explicar al cliente qué es exactamente lo que va a obtener. Cuando un cliente recibe un producto que no esperaba, generalmente no se siente muy inclinado a pagar por él.

Una vez recogidos los datos, el siguiente paso es crear un modelo conceptual. El modelo más usado en bases de datos es el *modelo Entidad-Relación*, que es el que vamos a explicar en este capítulo.

Muy probablemente, esta es la parte más difícil de la resolución del problema. Es la parte más "intelectual" del proceso, en el sentido de que es la que más requerirá pensar. Durante esta fase, seguramente, deberemos tomar ciertas decisiones, que en cierto modo limitarán en parte el modelo.

Cuando esto suceda, no estará de más consultar con el cliente para que estas decisiones sean, al menos, aceptadas por él, y si es posible, que sea el propio cliente el que las plantee.

La siguiente fase es convertir el modelo conceptual en un modelo lógico. Existen varios modelos lógicos, pero el más usado es el *modelo Relacional*. La conversión entre el modelo conceptual y el lógico es algo bastante mecánico, aunque no por ello será siempre sencillo. En el caso del modelo lógico relacional, existe un proceso que sirve para verificar que hemos aplicado bien el modelo, y en caso contrario, corregirlo para que sea así. Este proceso se llama *normalización*, y también es bastante mecánico. El último paso consiste en codificar el modelo lógico en un modelo físico. Este proceso está ligado al DBMS elegido, y es, seguramente, la parte más sencilla de aplicar, se requiere el conocimiento del lenguaje de consulta SQL

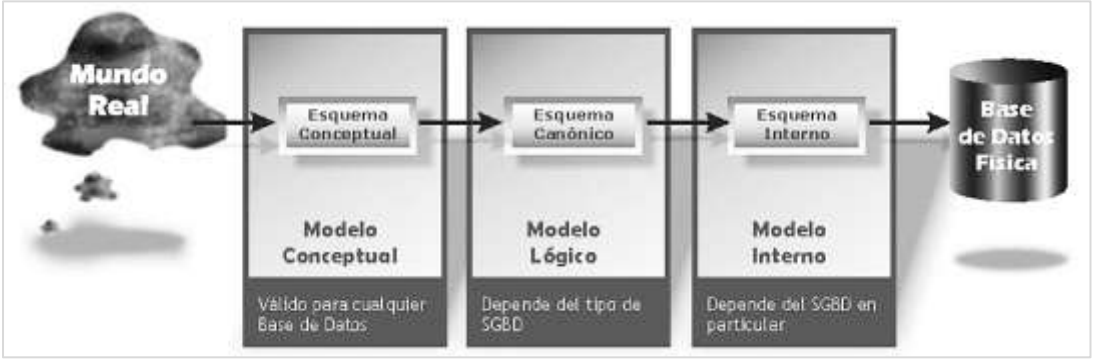
MODELOS DE DATOS

Los modelos se utilizan en todo tipo de ciencias. Su finalidad es la de simbolizar una parte del mundo real de forma que sea más fácilmente manipulable. En definitiva es un esquema mental (conceptual) en el que se intentan reproducir las características de una realidad específica. En el caso de los modelos de datos, lo que intentan reproducir es una información real que deseamos almacenar en un sistema informático.

Una de las características fundamentales de los sistemas de bases de datos es que proporcionan cierto *nivel de abstracción de datos*, al ocultar las características sobre el almacenamiento físico que la mayoría de usuarios no necesita conocer. Los **modelos de datos** son el instrumento principal para ofrecer dicha abstracción a través de su jerarquía de niveles.

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos, es decir, los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos. Los modelos de datos contienen también un conjunto de operaciones básicas para la realización de consultas (lecturas) y actualizaciones de datos. Además, los modelos de datos más modernos incluyen mecanismos para especificar acciones compensatorias o adicionales que se deben llevar a cabo ante las acciones habituales que se realizan sobre la base de datos.

Se denomina **esquema** a una descripción específica en términos de un modelo de datos. El conjunto de datos representados por el esquema forma la base de datos.



Clasificación de los modelos de datos

En la ilustración anterior aparecen los distintos esquemas que llevan desde el mundo real a la base de datos física. Como se ve aparecen varios esquemas intermedios. Los que están más a la izquierda se alejan más de las características físicas. Los elementos de ese esquema son:

- **Mundo real**. Contiene la información tal cual la percibimos como seres humanos. Es el punto de partida
- **Esquema conceptual**. Representa el modelo de datos de forma independiente del DBMS que se utilizará.
- **Esquema canónico (o de base de datos)**. Representa los datos en un formato más cercano al del ordenador
- **Esquema interno**. Representa los datos según el modelo concreto de un sistema gestor de bases de datos (por ejemplo Oracle)
- **Base de datos física**. Los datos tal cual son almacenados en disco.

Para conseguir estos esquemas se utilizan modelos de datos. El paso entre cada esquema se sigue con unas directrices concretas. Estas directrices permiten adaptar un esquema hacia otro.

Los modelos de datos se pueden clasificar dependiendo de los tipos de conceptos que ofrecen para describir la estructura de la base de datos, formando una jerarquía de niveles. Los modelos de datos de alto nivel, o modelos conceptuales, disponen de conceptos muy cercanos al modo en que la mayoría de los usuarios percibe los datos, mientras que los modelos de datos de bajo nivel, o modelos físicos, proporcionan conceptos que describen los detalles de cómo se almacenan los datos en el ordenador.

Los conceptos de los modelos físicos están dirigidos al personal informático, no a los usuarios finales. Entre estos dos extremos se encuentran los modelos lógicos, cuyos conceptos pueden ser entendidos por los usuarios finales, aunque no están demasiado alejados de la forma en que los datos se organizan físicamente. Los modelos lógicos ocultan algunos detalles de cómo se almacenan los datos, pero pueden implementarse de manera directa en un SGBD.

Los modelos conceptuales utilizan conceptos como entidades, atributos y relaciones. Una entidad representa un objeto o concepto del mundo real como, por ejemplo, un cliente de una empresa o una de sus facturas. Un atributo representa alguna propiedad de interés de una entidad como, por ejemplo, el nombre o el domicilio del cliente. Una relación describe una interacción entre dos o más entidades, por ejemplo, la relación que hay entre un cliente y las facturas que se le han realizado.

Cada SGBD soporta un modelo lógico, siendo los más comunes el relacional, el de red y el jerárquico. Estos modelos representan los datos valiéndose de estructuras de registros, por lo que también se denominan modelos orientados a registros. Hay una familia más moderna de modelos lógicos, son los modelos orientados a objetos, que están más próximos a los modelos conceptuales. En el modelo relacional los datos se describen como un conjunto de tablas con referencias lógicas entre ellas, mientras que en los modelos jerárquico y de red, los datos se describen como conjuntos de registros con referencias físicas entre ellos (punteros).

Los modelos físicos describen cómo se almacenan los datos en el ordenador: el formato de los registros, la estructura de los ficheros (desordenados, ordenados, agrupados) y los métodos de acceso utilizados (índices, tablas de dispersión).

A la descripción de una base de datos mediante un modelo de datos se le denomina esquema de la base de datos. Este esquema se especifica durante el diseño, y no es de esperar que se modifique a menudo. Sin embargo, los datos que se almacenan en la base de datos pueden cambiar con mucha frecuencia: se insertan datos, se actualizan, se borran, etc. Los datos que la base de datos contiene en un determinado momento conforman el estado de la base de datos, o como también se denomina: una ocurrencia de la base de datos.

La distinción entre el esquema y el estado de la base de datos es muy importante. Cuando definimos una nueva base de datos, sólo especificamos su esquema al SGBD. En ese momento, el estado de la base de datos es el estado vacío, sin datos. Cuando se cargan datos por primera vez, la base datos pasa al estado inicial. De ahí en adelante, siempre que se realice una operación de actualización de la base de datos, se tendrá un nuevo estado. El SGBD se encarga, en parte, de garantizar que todos los estados de la base de datos sean estados válidos que satisfagan la estructura y las restricciones especificadas en el esquema. Por lo tanto, es muy importante que el esquema que se especifique al SGBD sea correcto y se debe tener gran cuidado al diseñarlo. El SGBD almacena el esquema en su catálogo o diccionario de datos, de modo que se pueda consultar siempre que sea necesario.

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 es más cercano al ordenador

Es más cercano al usuario el modelo conceptual, el lógico forma el paso entre el informático y el sistema.

Algunos ejemplos de modelos conceptuales son:

- Modelo E/R
- Modelo RM/T
- Modelos semántico

Ejemplos de modelos lógicos son:

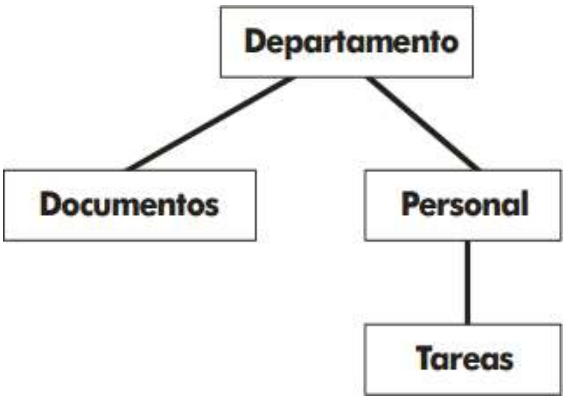
- Modelo relacional
- Codasyl
- Jerárquico

✓ **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 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).

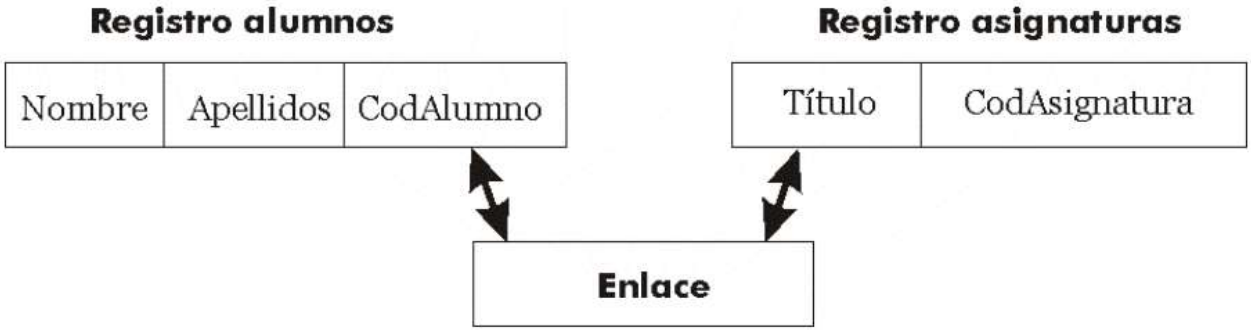
Los datos de este modelo se almacenan en estructuras lógicas llamadas segmentos. Los segmentos se relacionan entre sí utilizando arcos. La forma visual de este modelo es de árbol invertido, en la parte superior están los padres y en la inferior los hijos.



Ejemplo de esquema jerárquico

Este esquema está en absoluto desuso ya que no es válido para modelar la mayoría de problemas de bases de datos.

- ✓ **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 Codasyl a principios de los 70 que se ha convertido en el modelo en red más utilizado. 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.



Ejemplo de diagrama de estructura de datos Codasyl

- ✓ **Modelo Relacional.** En 1970, el modo en que se veían las bases de datos cambió por completo cuando E. F. Codd introdujo el modelo relacional. El modelo relacional representa la

segunda generación de los SGBD. En él, todos los datos están estructurados a nivel lógico como tablas formadas por filas y columnas, aunque a nivel físico pueden tener una estructura completamente distinta. Un punto fuerte del modelo relacional es la sencillez de su estructura lógica. Pero detrás de esa simple estructura hay un fundamento teórico importante del que carecen los SGBD de la primera generación, lo que constituye otro punto a su favor. La teoría de conjuntos es la verdadera base del modelo relacional. Los datos se agrupan en relaciones (actualmente llamadas tablas) que es un concepto que se refiere a la estructura que aglutina datos referidos a una misma entidad de forma independiente respecto a su almacenamiento físico. Los datos en una base de datos están almacenados en tablas, las tablas relacionales están definidas por sus columnas las cuales tienen un nombre. Los datos se almacenan en filas dentro de la tabla. Las tablas pueden relacionarse unas con otras y la base de datos puede ser usada para obligar a cumplir con estas relaciones.

- ✓ **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 en UML y el lógico actualmente en ODMG (Object Data Management Group, grupo de administración de objetos de datos, organismo que intenta crear estándares para este modelo).
- ✓ **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 SQL 99. 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, Informix, ...) son objeto relacionales.
- ✓ **Bases de datos NoSQL.** Bajo este nombre se agrupan las bases de datos (con arquitecturas muy diversas) pensadas para grabar los datos de manera veloz para así poder atender a miles y miles de peticiones. Es decir, es el modelo de las bases de datos que se utilizan en los grandes servicios de Internet (como twitter, Facebook, Amazon,…). La idea es que los datos apenas necesitan validarse y relacionarse y lo importante es la disponibilidad de la propia base de datos. El nombre NoSQL, hace referencia a que este modelo de bases de datos rompe con el lenguaje SQL (el lenguaje de las bases de datos relacionales, las bases de datos dominantes hasta la actualidad) para poder manipular los datos con lenguajes de otro tipo.