

Introducción a las Bases de Datos



INDICE

0. Bases de Datos	p.3
1. Sistemas de Ficheros vs. SGBD	p.4
1.1 Sistemas de Ficheros	p.4
1.2 Sistemas Gestores de Bases de Datos	p.7
2. Objetivos o Funcionalidades de los SGBD	p.10
3. Áreas de aplicación de los SGBD	p.13
4. Evolución histórica de los SGBD	p.14
4.1 Sistemas de Navegación	p.15
4.2 Modelo Relacional	p.16
4.3 Modelos Post-relacionales	p.16
5. Clasificación de los SGBD en función de su modelo lógico	p.17
5.1 Modelo Jerárquico	p.17
5.2 Modelo en Red	p.18
5.3 Modelo Relacional	p.20
5.4 Modelo Objeto-Relacional	p.20
5.5 BD Orientadas a Objeto	p.20
5.6 XML Nativo	p.20
5.7 'Content store'	p.20
5.8 NoSQL	p.22
5.9 NewSQL	p.22
6. Otros tipos de clasificaciones de los SGBD	p.24
6.1 En función del número de usuarios	p.24
6.2 En función del coste	p.24
6.3 En función del propósito o funcionalidad	p.24
7. Componentes de los SGBD	p.25
7.1 Lenguajes del SGBD (DML, DDL, DCL)	p.25
7.2 Diccionario de Datos	p.26
8. Arquitectura ANSI/X3/SPARC	p.29
9. Bases de Datos DISTRIBUIDAS	p.32
9.1 BDD, Bases de Datos Distribuidas	p.32
9.2 Clasificación según el tipo de distribución de los datos	p.33
10. Modelo ER: "Entity-Relationship"	p.35
11. Bibliografía	p.36

Introducción a las Bases de Datos

0. BASES DE DATOS

¿Qué es una BASE DE DATOS?

Miremos varias definiciones que podemos encontrar:

Una **Base De Datos** es un conjunto de datos almacenados entre los que existen relaciones lógicas y ha sido diseñada para satisfacer los requerimientos de información de una empresa u organización.

Una **Base de Datos** consiste en un conjunto de ficheros de datos interrelacionados

Una **base de datos** o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

A **database** is an organized collection of inter-related data that models some aspect of the real-world.

A **database** is an organized collection of data.

Database: a collection of pieces of information that is organized and used on a computer (Diccionario Merriam-Webster)

Database, also called electronic database, any collection of data, or information, that is specially organized for rapid search and retrieval by a computer. Databases are structured to facilitate the storage, retrieval, modification, and deletion of data in conjunction with various data-processing operations. (Enciclopedia Britannica)

Curiosidad: El término Base de Datos:

La introducción del término 'Base de Datos' coincide con la disponibilidad de dispositivos de almacenamiento de acceso directo ('discos duros') a partir de mediados de los 1960s. El término representó un contraste respecto los sistemas del pasado basados en cintas magnéticas (de acceso secuencial), permitiendo un uso más interactivo en lugar del típico proceso diario por lotes ('batch'). El Oxford English Dictionary cita un informe del System Development Corporation of California en 1962, como **la primera vez que se usó el término 'Base de Datos'** (database) con un sentido técnico específico.

1. SISTEMAS DE FICHEROS vs SGBD (Sistemas Gestores de Bases de Datos)

1.1 SISTEMAS DE FICHEROS (de datos):

Sistemas de ficheros (SF): conjunto de programas informáticos que permiten al usuario almacenar, consultar y modificar datos. Dichos datos se almacenan en ficheros diseñados para una determinada aplicación. Cada programa define y maneja sus propios datos.

Los **Sistemas de Ficheros** surgieron al tratar de informatizar el manejo de los archivadores manuales con objeto de proporcionar un acceso más eficiente a los datos.

En lugar de establecer un sistema centralizado en donde almacenar todos los datos de la organización o empresa, se escogió un modelo descentralizado en el que cada sección o departamento almacena y gestiona sus propios datos.

Para ayudar a gestionar dicha información descentralizada, se desarrollaban (a medida) un conjunto de programas que prestaban servicio a los usuarios finales. Cada programa define y maneja sus propios datos.

Debido a esto, los sistemas de ficheros presentan **una serie de inconvenientes:**

- **Separación y aislamiento de los datos.** 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 asegurar que se extraen los datos correctos.
- **Duplicación de datos. (o datos Redundantes)** La redundancia (o repetición) de datos existente en los sistemas de ficheros hace que se desperdicie espacio de almacenamiento innecesariamente y lo que es más importante: puede llevar a que se pierda la consistencia de los datos.
- **Datos Inconsistentes:** Se produce una inconsistencia cuando copias de los mismos datos no coinciden.

Por ejemplo, si un cliente cambia su dirección, y no lo actualizamos en todos los sitios donde aparecen sus datos, tendremos inconsistencia de datos.

- **Dependencia de los datos a nivel físico.** Ya que la estructura física de los datos (la definición de los ficheros y de los registros) se encuentra codificada en los programas de aplicación, cualquier cambio en dicha estructura es difícil de realizar. El programador debe identificar todos los programas afectados por este 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*.

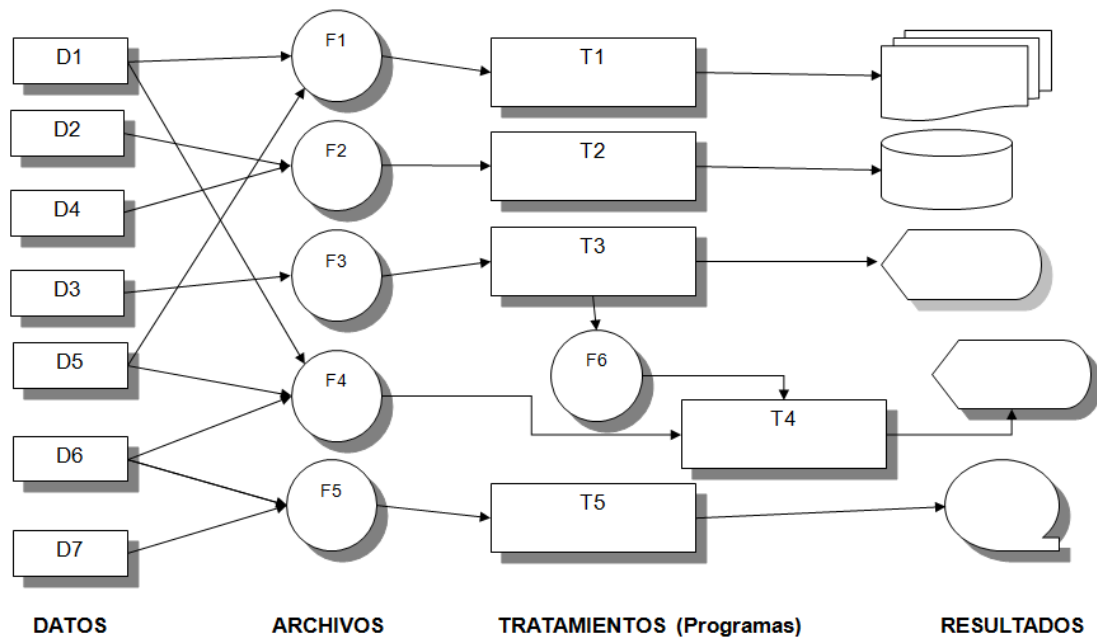
- **Dificultad para gestionar el acceso simultáneo a los datos (o Concurrencia)** Es casi imposible de conseguir ya que se usan archivos que no admiten esa posibilidad.
- **Formatos de ficheros incompatibles.** Ya que la estructura de los ficheros se define en los programas de aplicación, es completamente dependiente del lenguaje de programación. La incompatibilidad entre ficheros generados por distintos lenguajes hace que los ficheros sean difíciles de procesar de modo conjunto.
- **Dificultad en el acceso a los datos. Consultas fijas y proliferación de programas de aplicación.** Desde el punto de vista de los usuarios finales, los sistemas de ficheros fueron un gran avance comparados a los sistemas manuales. A consecuencia de esto, creció la necesidad de realizar distintos tipos de consultas de datos. Sin embargo, los sistemas de ficheros son muy dependientes del programador de aplicaciones: cualquier consulta o informe que se quiera realizar debe ser programado por él, lo cual dificultaba el acceso a los datos. En algunas organizaciones se conformaron con fijar el tipo de consultas e informes, siendo imposible realizar otro tipo de consultas que no se hubieran tenido en cuenta a la hora de escribir los programas de aplicación.

En otras organizaciones hubo una proliferación de programas de aplicación (o modificación de los existentes) para resolver todo tipo de consultas, hasta el punto de desbordar al departamento de proceso de datos, que no daba abasto para validar, mantener y documentar dichos programas.

- **Dificultad para administrar la Seguridad del Sistema.** Ya que cada aplicación se crea independientemente (y los ficheros están desperdigados en diferentes ubicaciones) es muy difícil establecer criterios de seguridad uniformes.
- **Problemas de integridad.** Los valores de los datos almacenados en la BD deben satisfacer ciertas restricciones. Los desarrolladores hacen cumplir estas restricciones en el sistema añadiendo código apropiado en las diversas aplicaciones. Sin embargo, cuando se añaden nuevas restricciones es difícil cambiar los programas para hacer que se cumplan. Esto se complica cuando las restricciones implican diferentes elementos de datos de diferentes archivos.
- **Problemas de atomicidad (o gestión de las Transacciones).** En muchas aplicaciones es crucial asegurar que, cuando ocurra un fallo y sea detectado, se restauren los datos a un estado de consistencia que existía antes del fallo. Es difícil asegurar esta propiedad en un sistema de archivos tradicional.

Nota: Algunos de estos conceptos se profundizan y se explican con más detalle en el apartado de **Objetivos o Funcionalidades de los SGBD**

Esquema de acceso a los datos en los Sistemas de Ficheros



Los Sistemas de Ficheros se consideran **los predecesores** de los sistemas gestores de bases de datos (SGBD)

1.2 SGBD: Sistemas Gestores de Bases de Datos

¿Qué es un SGBD?

Aquí tenemos varias definiciones:

El Sistema Gestor de Base de Datos (SGBD) es una aplicación (=software) que permite a los usuarios definir, crear y mantener la base de datos, y proporciona acceso controlado a la misma.

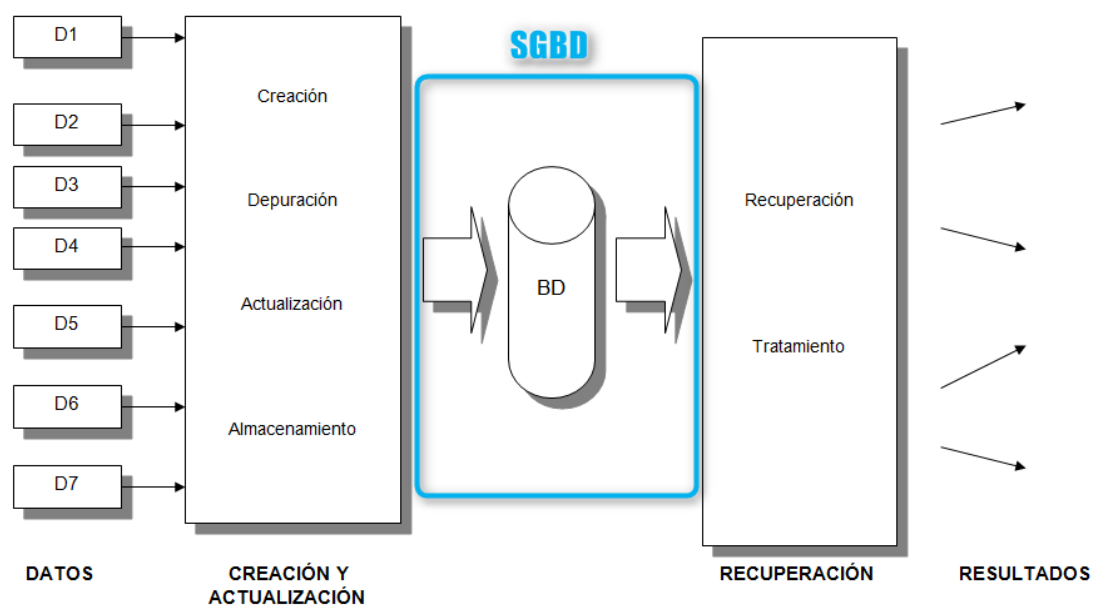
El SGBD es la aplicación que interacciona con los usuarios de los programas de aplicación y la base de datos.

Un SGBD es una aplicación informática (=software) especializado en gestionar y administrar bases de datos

Un SGBD es el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos. En estos Sistemas se proporciona 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.

DBMS: DataBase Management System =Siglas de SGBD en Inglés

Esquema de acceso a los datos con SGBD



Formalmente, una Base de Datos se refiere a un conjunto de datos, y la forma en la que se organizan. El acceso a estos datos usualmente se realiza mediante un SGBD, que consiste en un conjunto integrado de software que permite a los usuarios interactuar con una o más bases de datos y proporciona acceso a todos los datos contenidos en la base de datos (aunque puedan existir restricciones que impidan el acceso a determinados usuarios a ciertos datos).

El SGBD proporciona varias funciones que permiten la entrada, almacenamiento y recogida de grandes cantidades de información, así como maneras de gestionar y organizar dicha información.

Debido a la estrecha relación entre BD y SGBD, a veces nos tomamos la libertad de decir Base de Datos, y referirnos indistintamente tanto a una BD como al SGBD que se usa para manipularla.

Ahora, con los SGBD, los datos ya no estarán ubicados en ficheros desconectados con información redundante (como ocurría en los Sistemas de Ficheros), sino que están **centralizados** y gestionados por el SGBD,

Además, 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.

Independencia Lógica-Física: El modelo seguido con los sistemas de bases de datos, en donde se separa la definición de los datos de los programas de aplicación, es muy similar al modelo que se sigue en la actualidad para el desarrollo de programas, en donde se da una definición interna de un objeto y una definición externa separada. Los usuarios del objeto sólo ven la definición externa y no se deben preocupar de cómo se define internamente el objeto y cómo funciona. Una ventaja de este modelo, conocido como abstracción de datos, es que se puede cambiar la definición interna de un objeto sin afectar a sus usuarios ya que la definición externa no se ve alterada. Del mismo modo, los sistemas de bases de datos separan la definición de la estructura de los datos, de los programas de aplicación y almacenan esta definición en la base de datos. Si se añaden nuevas estructuras de datos o se modifican las ya existentes, los programas de aplicación no se ven afectados ya que no dependen directamente de aquello que se ha modificado.

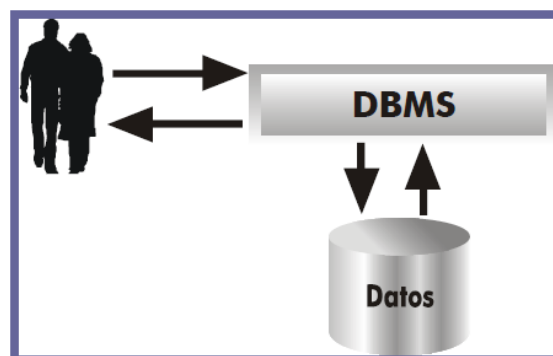


Tabla resumen, comparativa de ventajas y desventajas de ambos Sistemas

SGBD	Sistemas de FICHEROS
VENTAJAS: Datos centralizados Mínima Redundancia (o más controlable) Consistencia de los datos (consecuencia de las otras dos) Fácil obtención de cualquier conjunto de datos (debido a independencia lógico-física) Facilita el control de la Integridad de los datos (mediante restricciones) Atomicidad o Control de Transacciones sencillo Acceso Concurrente fácil de gestionar Seguridad en el acceso a los datos Fácil realizar Backups (incorporado) Expandible, flexible, escalable	DESVENTAJAS: Datos dispersos Existen más repeticiones de datos Se producen más Inconsistencia de datos Acceso complicado a datos, requieren de programas costosos y a medida. Integridad difícil de mantener. Los programas deben implementar todas las restricciones en el código fuente. Mantenimiento complicado. Difícil de garantizar e implementar la Atomicidad o Control de Transacciones (en los programas) No permite gestionar el acceso simultáneo Muy difícil de garantizar la Seguridad de los datos. Difícil realizar copias de Seguridad. Cuando crece, es inviable!
DESVENTAJAS: Caros: instalación costosa. Requiere personal cualificado Más complejo. Se justifica si gestiona gran volumen de datos.	VENTAJAS: Baratos (en principio), aunque cuando el sistema crece acaba siendo más caro. Sencillo (al principio). Si el volumen de datos es pequeño, puede ser una solución.

2. Objetivos o Funcionalidades de los SGBD

1. Un SGBD debe proporcionar a los usuarios la **capacidad de almacenar datos** en la base de datos, **acceder a ellos y actualizarlos**.

Esta es la función fundamental de un SGBD y por supuesto, el SGBD debe ocultar al usuario la estructura física interna (la organización de los ficheros y las estructuras de almacenamiento).

2. Asegurar la **Consistencia de los Datos:** Datos verdaderos, correctos, coherentes, que no se contradigan (que no hayan 'mentiras' ni falsedades)

Sería una inconsistencia que un campo derivado **TOTAL Facturado** de un cliente no coincida con la suma real de todas sus facturas

El SGBD debe proporcionar un mecanismo capaz de recuperar la base de datos en caso de que ocurra algún suceso que la dañe (como puede ser un fallo de hardware o software), y llevarla a un estado consistente.

3. **Asegurar la Integridad de los Datos:** Integridad = **Consistencia + Calidad de los datos**

La **integridad se ocupa de la calidad de los datos**. Normalmente se expresa mediante restricciones, que son una serie de reglas que la base de datos no puede violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

Violar dichas reglas, puede empeorar la calidad de los datos (aunque podrían seguir siendo consistentes: ej. diferentes formatos de fecha), pero por lo general, además también afecta a la consistencia de los datos (es decir, que haya incoherencias o 'mentiras')

Por tanto integridad es un término más general que consistencia.

La integridad se puede considerar como otro modo de proteger la base de datos, para asegurar que los datos son correctos y de calidad

Ejemplos de Reglas de Integridad:

- 1) que un campo derivado **TOTAL Facturado** de un cliente debe coincidir (o calcularse) con la suma real de todas sus facturas. Si no se aplica dicha regla, la base de datos quedaría inconsistente (errónea)
- 2) se puede establecer la restricción de que cada empleado no puede tener asignados más de diez inmuebles.
- 3) Que el campo Sexo sólo admita 2 valores: H, M

- 4) Que el campo altura (de una persona), no admita valores negativos, ni mayores a 3 metros (se podría considerar un error)
- 5) En una biblioteca donde se realizan préstamos de libros, debería cumplirse siempre que la fecha de préstamo debe ser anterior a la fecha de devolución.
- 6) En los amarres de un puerto: si tenemos los datos de la anchura de los barcos (y su calado), y de la anchura de los amarres (y su profundidad) , una regla de integridad podría ser que no podemos nunca asignarle un amarre a un barco que sea más ancho que el amarre (o su calado sea mayor que la profundidad del amarre menos un 30%)

Ejemplos de Falta de Integridad:

Tener en campo fecha de nacimiento valores del tipo: (22-03-98), (8-22-1983), (5 de Enero de 1974), (2003 marzo 4), (2 Sept. 1966), (tres del cuatro de mil novecientos cuarenta y dos)

Tener un campo Sexo, con los valores: Hombre, Mujer, H, M, Varón, Femenino, Masculino, ...

En general cualquier restricción que el modelo de negocio imponga sobre sus datos y que no se cumpla.

4. Gestion de Transacciones:

Una **transacción** es un conjunto de acciones que cambian el contenido de la base de datos, y que deben considerarse como un TODO, de forma que si ocurre algún fallo en medio del proceso, todos los cambios realizados hasta ese momento deben deshacerse, y dejar la base de datos en un **estado Consistente**

Un SGBD debe proporcionar un mecanismo que garantice que todas las actualizaciones correspondientes a una determinada transacción se realicen, o que no se realice ninguna. Si la transacción falla durante su realización, por ejemplo porque falla el hardware, la base de datos quedará en un estado inconsistente. Algunos de los cambios se habrán hecho y otros no, por lo tanto, los cambios realizados deberán ser deshechos para devolver la base de datos a un estado consistente.

Ejemplos de Transacciones:

Una transacción en el sistema informático de la empresa inmobiliaria sería dar de alta a un empleado o eliminar un inmueble.

Una transacción un poco más complicada sería eliminar un empleado y reasignar sus inmuebles a otro empleado. En este caso hay que realizar varios cambios sobre la base de datos.

Una transferencia bancaria

In [computer science](#), **ACID** ([atomicity](#), [consistency](#), [isolation](#), [durability](#)) is a set of properties of [database transactions](#) intended to guarantee data validity despite errors, power failures, and other mishaps. In the context of [databases](#), a sequence of database operations that satisfies the ACID properties (which can be perceived as a single logical operation on the data) is called a *transaction*. For example, a transfer of funds from one bank account to another, even involving multiple changes such as debiting one account and crediting another, is a single transaction.

5. Un SGBD debe proporcionar un **catálogo (o diccionario de datos)**, en el que se almacenen las descripciones de los datos de la base de datos y que sea accesible por los usuarios. Este catálogo contiene información que describe **los datos de la base de datos (metadatos)**

6. Gestión de Concurrencia:

Concurrencia: acceso simultáneo a los datos por parte de varios usuarios a la vez.

Un SGBD debe proporcionar un mecanismo que asegure que la base de datos se actualice correctamente cuando varios usuarios la están actualizando concurrentemente (a la vez). Uno de los principales objetivos de los SGBD es el permitir que varios usuarios tengan acceso concurrente a los datos que comparten. El acceso concurrente es relativamente fácil de gestionar si todos los usuarios se dedican a leer datos, ya que no pueden interferir unos con otros. Sin embargo, cuando dos o más usuarios están accediendo a la base de datos y al menos uno de ellos está actualizando datos, pueden interferir de modo que se produzcan inconsistencias en la base de datos. El SGBD se debe encargar de que estas interferencias no se produzcan en el acceso simultáneo.

7. Gestión de la Seguridad: Un SGBD debe proporcionar un mecanismo que garantice que **sólo los usuarios autorizados** pueden acceder a la base de datos, o a una parte de ella. La protección debe ser contra accesos no autorizados, tanto intencionados como accidentales.

8. Herramientas de Administración: Un SGBD debe proporcionar una serie de herramientas que permitan administrar la base de datos de modo efectivo. Algunas herramientas trabajan a nivel externo, por lo que habrán sido producidas por el administrador de la base de datos. Las herramientas que trabajan a nivel interno deben ser proporcionadas por el distribuidor del SGBD. Algunas de ellas son:

- Herramientas para importar y exportar datos.
- Herramientas de Copias de Seguridad y Restauración
- Herramientas para monitorizar el uso y el funcionamiento de la base de datos.
- Programas de análisis estadístico para examinar las prestaciones o las estadísticas de utilización.
- Herramientas para reorganización de índices.

El **DBA** (DataBase Administrator) es la persona encargada de Administrar la Base de Datos

3. Áreas de aplicación de los SGBD

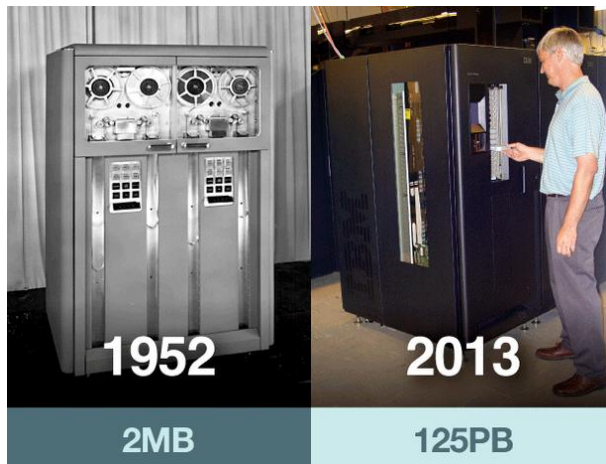
Se hace difícil (si no imposible) pensar en un área o negocio que no requieran de un SGBD. Aquí se ofrece una lista general de áreas de aplicación que requieren de SGBD.

1. **Banking:** For customer information, accounts, and loans, and banking transactions.
2. **Airlines:** For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner — terminals situated around the world accessed the central database system through phone lines and other data networks.
3. **Universities:** For student information, course registrations, and grades.
4. **Credit card transactions:** For purchases on credit cards and generation of monthly statements.
5. **Telecommunication:** For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
6. **Finance:** For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
7. **Sales:** For customer, product, and purchase information.
8. **Manufacturing:** For management of supply chain and for tracking production of items in factories, inventories of items in warehouses / stores, and orders for items.
9. **Human resources:** For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.

4. Evolución Histórica de los SGBD

Para poder entender mejor por qué los SGBD son actualmente como los conocemos, debemos repasar brevemente su historia.

Siguiendo el progreso tecnológico de las computadoras en las áreas de los procesadores (CPU), memoria, capacidad de almacenamiento, y la evolución de las Redes (su interconexión), podemos decir que las Bases de Datos y sus correspondientes SGBD han crecido órdenes de magnitud en términos de tamaño, capacidad y rendimiento. Podemos decir que la evolución de los SGBD ha estado ligada a la evolución del hardware.



El desarrollo de la tecnología de los SGBD puede ser dividido en 3 generaciones basándonos en su modelo de datos o estructura:

1ª generación: **Sistemas de Navegación** ('Navigational'): (1960s)

Modelo Jerárquico ('Hierarchical'):

Modelo en Red ('Network'), también llamado CODASYL

2ª generación: **Modelo RELACIONAL** ('Relational') (1970s)

3ª generación: **Modelos Post-Relacionales** (1990s)

Objeto-Relacional

Orientadas a Objeto ('Object Oriented')

XML Nativo

'Content store'

NoSQL ('Not only SQL' or 'non SQL' or 'non relational') (2000s)

Surgieron principalmente para solucionar problemas del 'Big Data':

'Document store'

'Key-value store'

'Multivalue'

'Wide Columnar store'

'Multimode'

'Graph'

'RDF'

NewSQL

4.1. Sistemas de Navegación ('Navigational') <= 1ª generación

Se dice que los sistemas de bases de datos tienen sus raíces en el proyecto estadounidense Apolo de mandar al hombre a la luna, en los años sesenta. En aquella época, no había ningún sistema que permitiera gestionar la inmensa cantidad de información que requería el proyecto. La primera empresa encargada del proyecto, NAA (North American Aviation), desarrolló un software denominado GUAM (General Update Access Method) que estaba basado en el concepto de que varias piezas pequeñas se unen para formar una pieza más grande, y así sucesivamente hasta que el producto final está ensamblado. Esta estructura, que tiene la forma de un árbol, es lo que se denomina una **estructura Jerárquica**. A mediados de los sesenta, IBM se unió a NAA para desarrollar GUAM en lo que ahora se conoce como **IMS** (Information Management System). El motivo por el cual IBM restringió IMS al manejo de jerarquías de registros fue el de permitir el uso de dispositivos de almacenamiento serie, más exactamente las cintas magnéticas, ya que era un requisito del mercado por aquella época.

A mitad de los sesenta, se desarrolló IDS (Integrated Data Store), de General Electric. Este trabajo fue dirigido por uno de los pioneros en los sistemas de bases de datos, Charles Bachmann. IDS era un nuevo tipo de sistema de bases de datos conocido como **Sistema de Red**, que produjo un gran efecto sobre los sistemas de información de aquella generación. El sistema de red se desarrolló, en parte, para satisfacer la necesidad de representar relaciones entre datos más complejas que las que se podían modelar con los sistemas jerárquicos, y, en parte, para imponer un estándar de bases de datos. Para ayudar a establecer dicho estándar, CODASYL (Conference on Data Systems Languages), formado por representantes del gobierno de EEUU y representantes del mundo empresarial, formaron un grupo denominado DBTG (Data Base Task Group), cuyo objetivo era definir unas especificaciones estándar que permitieran la creación de bases de datos y el manejo de los datos. El DBTG presentó su informe final en 1971 y aunque éste no fue formalmente aceptado por ANSI (American National Standards Institute), muchos sistemas se desarrollaron siguiendo la propuesta del DBTG. Estos sistemas son los que se conocen como sistemas de red, o sistemas CODASYL o DBTG.

Los sistemas jerárquico y de red constituyen la primera generación de los SGBD. Pero estos sistemas presentan algunos inconvenientes:

- Es necesario escribir complejos programas de aplicación para responder a cualquier tipo de consulta de datos, por simple que ésta sea.
- La independencia de datos es mínima.
- No tienen un fundamento teórico.

1960s

First Computerized
Database Models

Hierarchical
Model (IMS)



Network Model
(CODASYL)

4.2. Modelo RELACIONAL <= 2ª generación

En 1970 Codd, de los laboratorios de investigación de IBM, escribió un artículo presentando el **Modelo Relacional**. En este artículo, presentaba también los inconvenientes de los sistemas previos, el jerárquico y el de red. Entonces, se comenzaron a desarrollar muchos sistemas relacionales, apareciendo los primeros a finales de los setenta y principios de los ochenta. Uno de los primeros es System R, de IBM, que se desarrolló para probar la funcionalidad del modelo relacional, proporcionando una implementación de sus estructuras de datos y sus operaciones. Esto condujo a dos grandes desarrollos:

- El desarrollo de un lenguaje de consultas estructurado denominado **SQL**, que se ha convertido en el lenguaje estándar de los sistemas relacionales.
- La producción de varios SGBD relacionales durante los años ochenta, como DB2 y SLQ/DS de IBM, y **ORACLE** de ORACLE Corporation.

Otros sistemas relacionales multiusuario son INGRES de Computer Associates, Informix de Informix Software Inc. y Sybase de Sybase Inc. Ejemplos de sistemas relacionales de microordenadores son Paradox y dBase IV de Borland, Access de Microsoft, FoxPro y R:base de Microrim.

Los SGBD relacionales constituyen la segunda generación de los SGBD.

4.3. Modelos Post-Relacionales <= 3ª generación

El modelo relacional también tiene sus carencias, siendo uno de ellos su limitada capacidad al modelar algunos tipos de datos. Se ha hecho mucha investigación desde entonces tratando de resolver este problema.

En 1979, Codd intentó subsanar algunas de las deficiencias de su modelo relacional con una versión extendida denominada RM/T (1979) y más recientemente RM/V2 (1990).

Como respuesta a la creciente complejidad de las aplicaciones que requieren bases de datos, **surgieron nuevos modelos**: el modelo de datos **orientado a objetos**, el modelo **objeto-relacional**, modelos **XML Nativo**, **'Content store'**, ...

Además, las necesidades de aplicaciones de 'Big Data' dieron lugar a modelos

NoSQL :

'Document store', 'Key-value store', 'Multivalue', 'Wide Columnar store', 'Multimode', 'Graph', 'RDF', ...

Una "nueva generación" de bases de datos está surgiendo para hacer la competencia a las NoSQL. Son las llamadas **NewSQL** que intentan combinar las ventajas del modelo relacional y el alto rendimiento de las NoSQL (respecto a los SGBD relacionales disponibles).

Sin embargo, a diferencia de los modelos que los preceden, la composición de estos nuevos modelos no está tan clara. A veces un modelo tiene características de varios tipos

Esta evolución representa la tercera generación de los SGBD.

5. Clasificación de los SGBD en función de su modelo lógico

Un modelo de base de datos, es un modelo de tipo de datos que determina la estructura lógica de la base de datos y determina fundamentalmente de qué manera los datos pueden ser almacenados, organizados y manipulados.

5.1 Modelo JERARQUICO

Las BD jerárquicas se desarrollaron a principios de los años sesenta, y aún hoy se utilizan debido a su buen rendimiento y mejor estabilidad que proporcionan con grandes volúmenes de información.

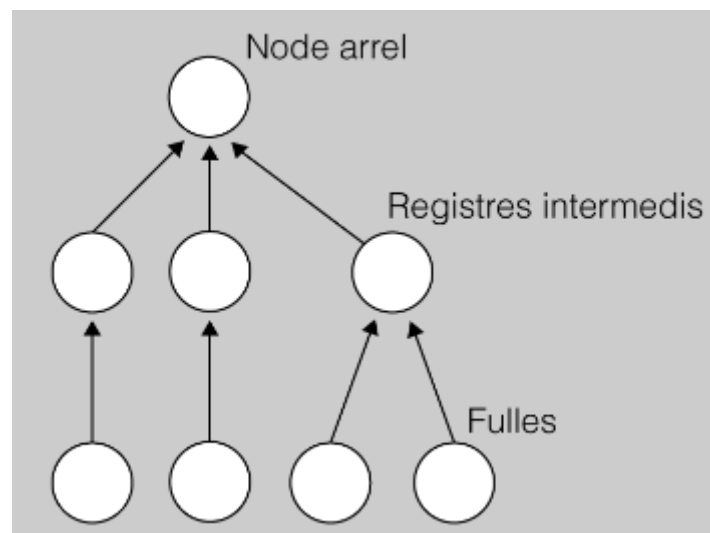
Utiliza una estructura en árbol para organizar los datos.

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 registros. Los registros se relacionan entre sí utilizando enlaces (o punteros).

Cualquier registro contiene la dirección física de su registro padre en el medio de almacenamiento utilizado (ej. sectores y pistas en el caso de discos duros). Esta circunstancia proporciona un rendimiento muy bueno: el acceso desde un registro a otro es prácticamente inmediato.

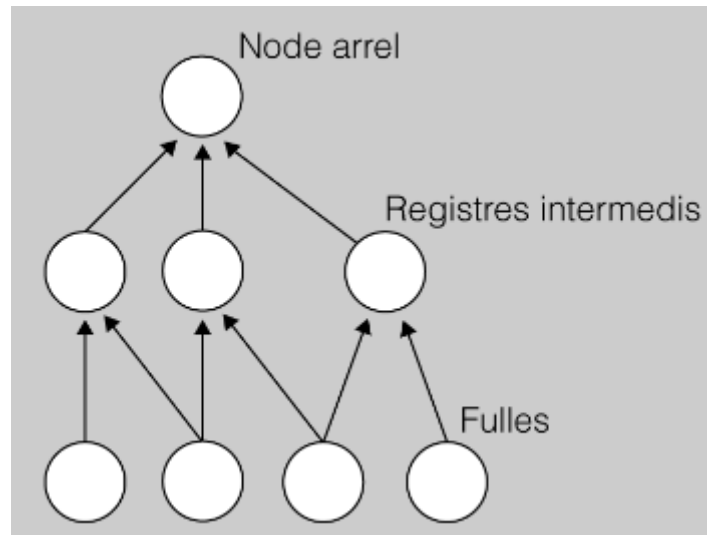
La forma visual de este modelo es de árbol invertido, en la parte superior están los padres y en la inferior los hijos.



IMS ([Information Management System](#)) de IBM : IMS se sigue utilizando hoy en día (en su origen fue un desarrollo de software escrito para el programa Apollo)

5.2 Modelo en RED

A mediados de los años sesenta, en el mercado fueron surgiendo BD que seguían un modelo en Red, parecido al jerárquico, con registros interrelacionados mediante una estructura en forma de árbol invertido, pero más flexible, ya que permitía que los nodos tuvieran más de un padre.



IDS (Integrated Data Store): usado ampliamente en la industria, conocido por su alto rendimiento con grandes volúmenes de datos.

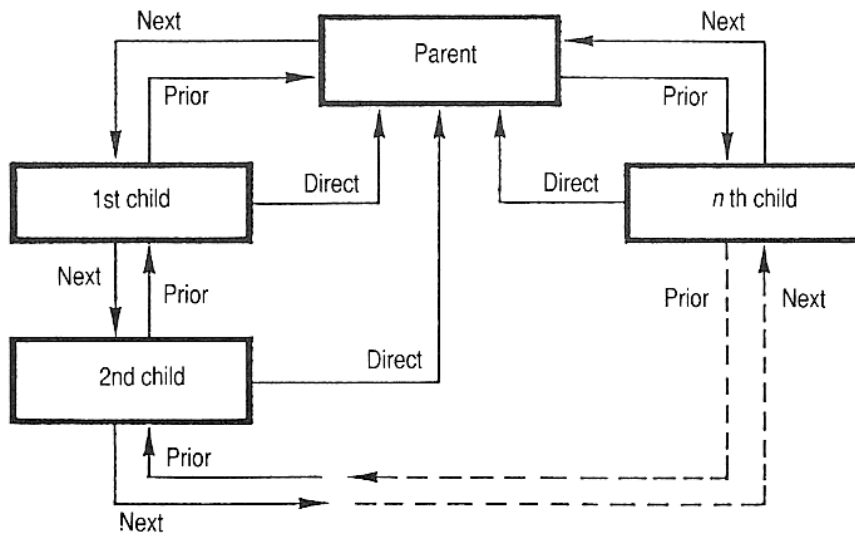
IDMS: (Integrated Database Management System), es un SGBD para 'Mainframes'

No era sencillo usar o implementar aplicaciones con estos sistemas porque estaban diseñados para maximizar el rendimiento usando el hardware disponible en ese tiempo. Eran aplicaciones muy dependientes de la implementación física, y se volvían muy complejas, requiriendo personal altamente cualificado.

Charles Bachmann: desarrolló IDS en 1963. Se considera el pionero de los Sistemas en Red



Estructura básica del modelo de Navegación CODASYL o en Red



A closed chain of records in a navigational database model (e.g. CODASYL), with **next pointers**, **prior pointers** and **direct pointers** provided by keys in the various records.

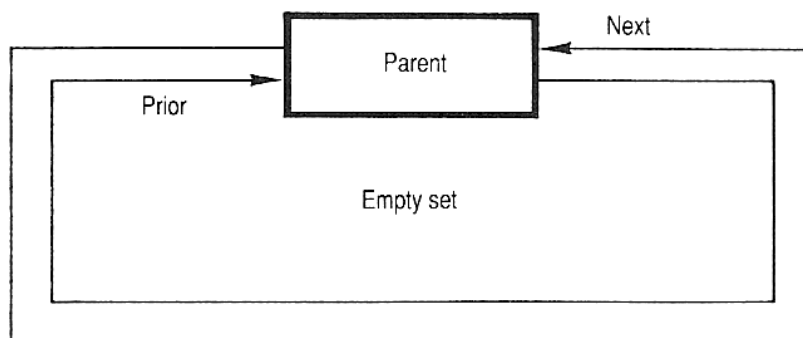


Illustration of an **empty set**

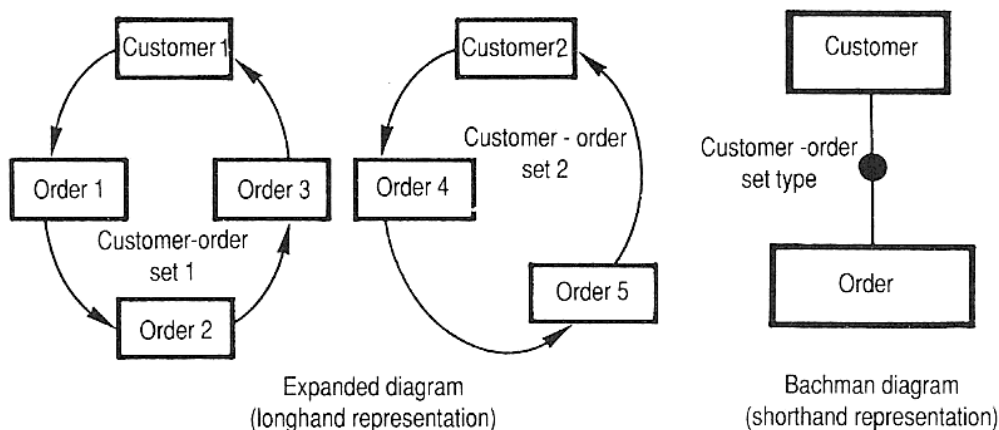


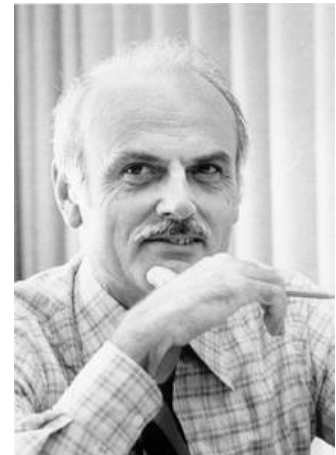
Illustration of a set type using a **Bachman diagram**

The record set, basic structure of navigational (e.g. CODASYL) database model. A set consists of one parent record (also called "the owner"), and n child records (also called members records)

5.3 Modelo RELACIONAL

El modelo Relacional fue propuesto por **Edgar F. Codd** en 1970

Los datos se estructuran en representaciones tabulares, llamadas **tablas** (también llamadas Relaciones), que representan entidades tipo del mundo conceptual, y que están formadas por filas y columnas. Las columnas forman los campos, que implementan los atributos, es decir, las características que nos interesan de las entidades. Y las filas son los registros, que implementan las entidades instancia, constituidas por los conjuntos de los valores que presentan los campos correspondientes a cada instancia.



Sólo a mediados de la década de 1980, el hardware de computación se convirtió en lo suficientemente potente como para permitir el amplio despliegue de sistemas relacionales. A principios de la década de 1990, los sistemas relacionales dominaban en todas las aplicaciones de procesamiento de datos a gran escala, y actualmente (2015) siguen siendo dominantes: **Oracle, MySQL, Microsoft SQL Server, PostgreSQL, IBM DB2** son los principales SGBD.

El lenguaje de base de datos de los SGBD Relacionales es el **SQL**

El modelo Relacional es el más extendido y utilizado en todo el mundo.

Top-10 DBMS

Rank			DBMS	Database Model	Score		
Sep 2020	Aug 2020	Sep 2019			Sep 2020	Aug 2020	Sep 2019
1.	1.	1.	Oracle +	Relational, Multi-model	1369.36	+14.21	+22.71
2.	2.	2.	MySQL +	Relational, Multi-model	1264.25	+2.67	-14.83
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model	1062.76	-13.12	-22.30
4.	4.	4.	PostgreSQL +	Relational, Multi-model	542.29	+5.52	+60.04
5.	5.	5.	MongoDB +	Document, Multi-model	446.48	+2.92	+36.42
6.	6.	6.	IBM Db2 +	Relational, Multi-model	161.24	-1.21	-10.32
7.	7.	↑ 8.	Redis +	Key-value, Multi-model	151.86	-1.02	+9.95
8.	8.	↓ 7.	Elasticsearch +	Search engine, Multi-model	150.50	-1.82	+1.23
9.	9.	↑ 11.	SQLite +	Relational	126.68	-0.14	+3.31
10.	↑ 11.	10.	Cassandra +	Wide column	119.18	-0.66	-4.22
11.	↓ 10.	↓ 9.	Microsoft Access	Relational	118.45	-1.41	-14.26
12.	12.	↑ 13.	MariaDB +	Relational, Multi-model	91.61	+0.69	+5.54
13.	13.	↓ 12.	Splunk	Search engine	87.90	-2.01	+0.89
14.	14.	↑ 15.	Teradata +	Relational, Multi-model	76.39	-0.39	-0.57
15.	15.	↓ 14.	Hive	Relational	71.17	-4.12	-11.93
16.	16.	↑ 18.	Amazon DynamoDB +	Multi-model	66.18	+1.43	+8.36
17.	17.	↑ 25.	Microsoft Azure SQL Database	Relational, Multi-model	60.45	+3.60	+32.91
18.	18.	↑ 19.	SAP Adaptive Server	Relational	54.01	+0.05	-2.09
19.	19.	↑ 21.	SAP HANA +	Relational, Multi-model	52.86	-0.26	-2.53
20.	20.	↓ 16.	Solr	Search engine	51.62	-0.08	-7.35

<http://db-engines.com/en/ranking>

5.4 Modelo Objeto-Relacional

Una base de datos Objeto-Relacional es un sistema de gestión de base de datos similar a una base de datos relacional, pero con un modelo de base de datos orientada a objetos: objetos, clases y herencia son directamente soportadas en esquemas de bases de datos y en el lenguaje de consulta

El objetivo básico de la base de datos objeto-relacional es cerrar la brecha entre las bases de datos relacionales y las técnicas de modelado orientados a objetos utilizados en los lenguajes de programación como Java, C ++, Visual Basic .NET o C #.

5.5 BD Orientadas a Objeto ('Object Oriented'= OO)

Una base de datos Orientada a Objeto es un sistema de gestión de base de datos en la que la información se representa en la forma de objetos tal como se utiliza en la programación orientada a objetos.

Nunca alcanzaron el nivel de popularidad de los sistemas Relacionales.

Ej. *Objectivity/DB*, *VelocityDB*, y *Versant*.

5.6 XML Nativo

Las bases de datos XML son un tipo de base de datos documental estructurado que permite la consulta basada en atributos del documento XML.

Bases de datos XML se utilizan sobre todo en la gestión de la base de datos de la empresa, donde se utiliza XML como estándar de interoperabilidad de datos de máquina a máquina

- XQuery es un lenguaje de consulta XML estándar implementado por los sistemas de bases de datos XML

5.7 'Content store'

Un repositorio de Contenidos ('Content store') es una base de datos de contenidos digitales con un conjunto asociado de gestión de datos, búsqueda y métodos de acceso que permiten a la aplicación acceso independiente a los contenidos, más bien como una biblioteca digital, pero con la capacidad de almacenar y modificar el contenido, además de búsqueda y recuperación. El repositorio de contenido actúa como el motor de almacenamiento para una aplicación más grande, como un gestor de contenidos o de un sistema de gestión de documentos, lo que añade una interfaz de usuario en la parte superior de la interfaz de programación de aplicaciones del repositorio.

5.8 NoSQL

'Not only SQL' or 'non SQL' or 'non relational'

Surgieron principalmente para solucionar problemas del 'Big Data':

Tipos:

'Document store': MongoDB, CouchDB

'Key-value store'

'Multivalue'

'Wide Columnar store'

'Multimode'

'Graph' :

'RDF'

...

Las Bases de datos NoSQL son normalmente muy rápidas, no requieren esquemas de tablas fijas, evitan las operaciones de JOIN mediante el almacenamiento de datos no normalizados, y están diseñados para escalar horizontalmente. Los sistemas más populares NoSQL son: MongoDB, Couchbase, Riak, Memcached, Redis, CouchDB, Hazelcast, Apache Cassandra HBase, que son todos productos de software de código abierto.

5.9 NewSQL

NewSQL es una clase de bases de datos relacionales modernas que tiene como objetivo ofrecer el mismo rendimiento escalable de sistemas NoSQL para el procesamiento de transacciones online (lectura y escritura) sin dejar de utilizar SQL y el manteniendo las características ACID de un sistema de base de datos tradicional.

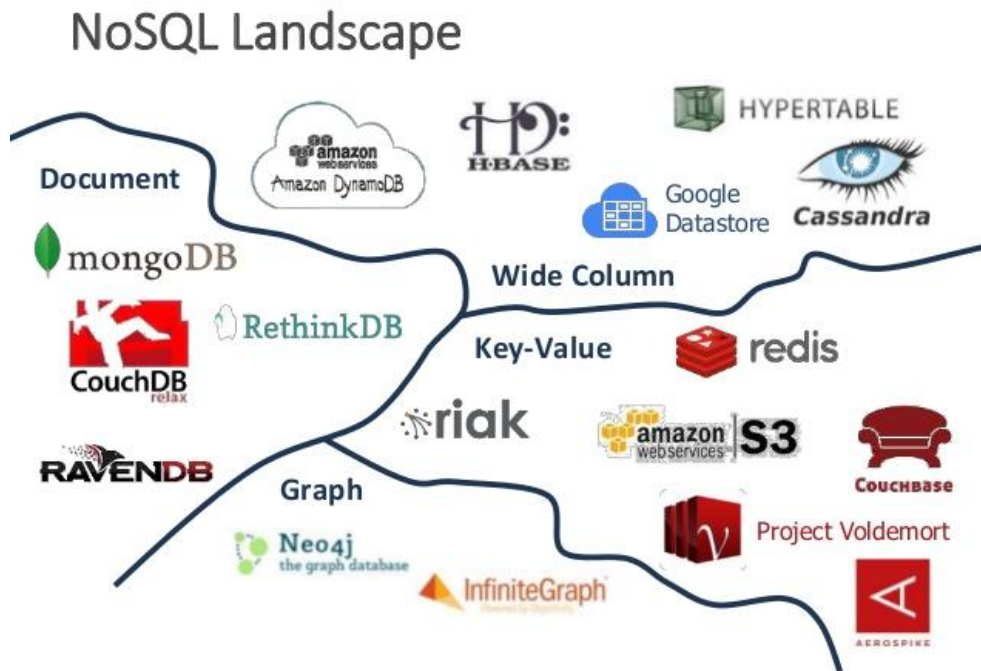
Tales bases de datos incluyen ScaleBase, Clustrix, EnterpriseDB, MemSQL, NuoDB y VoltDB.

Logotipos de algunos SGBD Relacionales

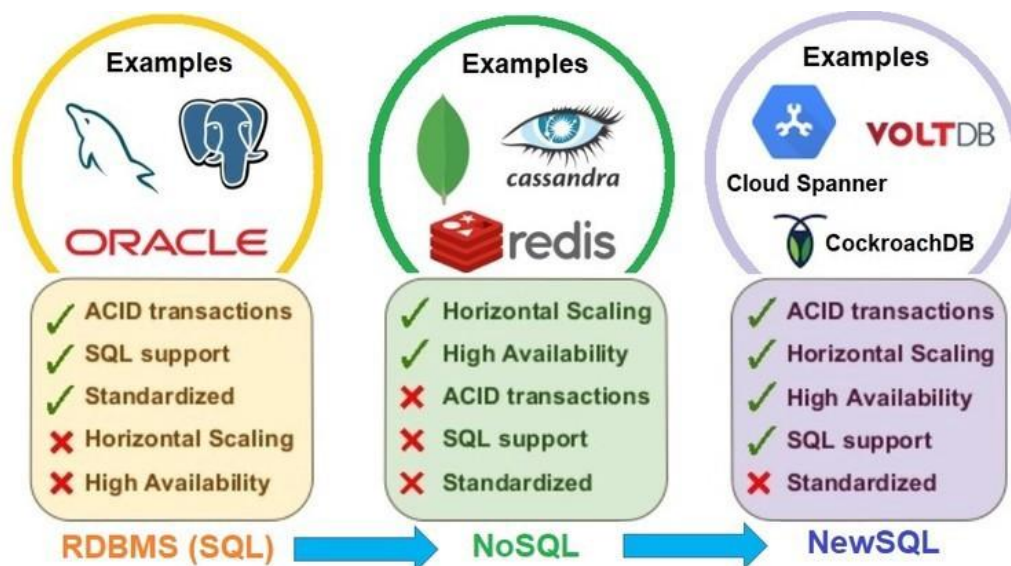


Traditional relational databases weren't invented with mobile, social, and big data types -- or extreme scale -- in mind

Principales SGBD tipo NoSQL



Principales características de los SGBD Relacionales, NoSQL y NewSQL



6. Otros tipos de Clasificaciones de los SGBD

6.1 En función del número de usuarios

Un **segundo criterio** para clasificar los SGBD es el número de usuarios a los que da servicio el sistema. Los sistemas *monousuario* sólo atienden a un usuario a la vez, y su principal uso se da en los ordenadores personales. Los sistemas *multiusuario*, entre los que se encuentran la mayor parte de los SGBD, atienden a varios usuarios al mismo tiempo.

6.2 En función del coste

Un **cuarto criterio** es el coste del SGBD. La mayor parte de los paquetes de SGBD cuestan entre 10.000 y 100.000 euros. Los sistemas monousuario más económicos para microcomputadores cuestan entre 100 y 3.000 euros. En el otro extremo, los paquetes más completos cuestan más de 100.000 euros.

Ejercicio: buscar Precios de implementación de diferentes SGBD del mercado (Oracle, SQL Server, etc.), viendo parámetros de los que depende (CPUs, usuarios?,...)

6.3 En función del propósito o funcionalidad

Los SGBD pueden ser de propósito general o de propósito específico. Cuando el rendimiento es fundamental, se puede diseñar y construir un SGBD de propósito especial para una aplicación específica, y este sistema no sirve para otras aplicaciones. Muchos sistemas de reservas de líneas aéreas son SGBD de propósito especial y pertenecen a la categoría de *sistemas de procesamiento de transacciones en línea* (OLTP), que deben atender un gran número de transacciones concurrentes sin imponer excesivos retrasos.

7. COMPONENTES de los SGBD

7.1 LENGUAJES del SGBD

LENGUAJES DDL , DML y DCL

- Permite la definición de la base de datos mediante el lenguaje de definición de datos (DDL: Data Definition Language) Este lenguaje permite especificar la estructura y el tipo de los datos, así como las restricciones sobre los datos. Todo esto se almacenará en la base de datos.
- Permite la inserción, actualización, eliminación y consulta de datos mediante el lenguaje de manejo de datos (DML: Data Manipulation Language) 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.
- Permite gestionar permisos para controlar el acceso a la base de datos mediante el lenguaje de control de datos (DCL: Data Control Language)

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 deben 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 **lenguaje SQL** contiene instrucciones cuyas funcionalidades se pueden clasificar en los 3 tipos mencionados

- DDL: CREATE, ALTER, DROP, ...
- DML: SELECT, INSERT, UPDATE, DELETE, ...
- DCL: GRANT, REVOKE, ...

7.2 DICCIONARIO DE DATOS

En una base de datos, además de los datos, también se almacena su descripción.

El Diccionario de Datos, es un catálogo en el que se almacenan las descripciones de los datos de la base de datos y que es accesible por los usuarios y contiene información que describe los datos de la base de datos (metadatos)

Es una Base de Datos que guarda información sobre sí misma!

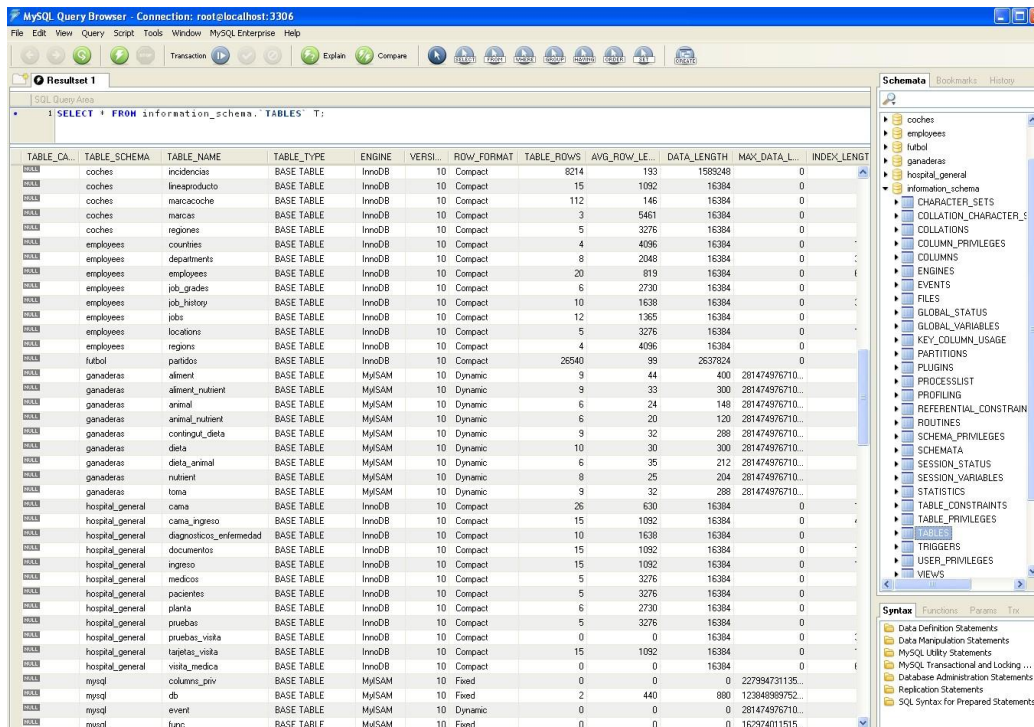
Normalmente, un diccionario de datos almacena:

- Nombre, tipo y tamaño de los datos.
- Nombre de las relaciones entre los datos.
- Restricciones de integridad sobre los datos.
- Nombre de los usuarios autorizados a acceder a la base de datos.
- Esquemas externos, conceptual e interno, y correspondencia entre los esquemas.
- Estadísticas de utilización, tales como la frecuencia de las transacciones y el número de accesos realizados a los objetos de la base de datos.

Algunos de los beneficios que reporta el diccionario de datos son los siguientes:

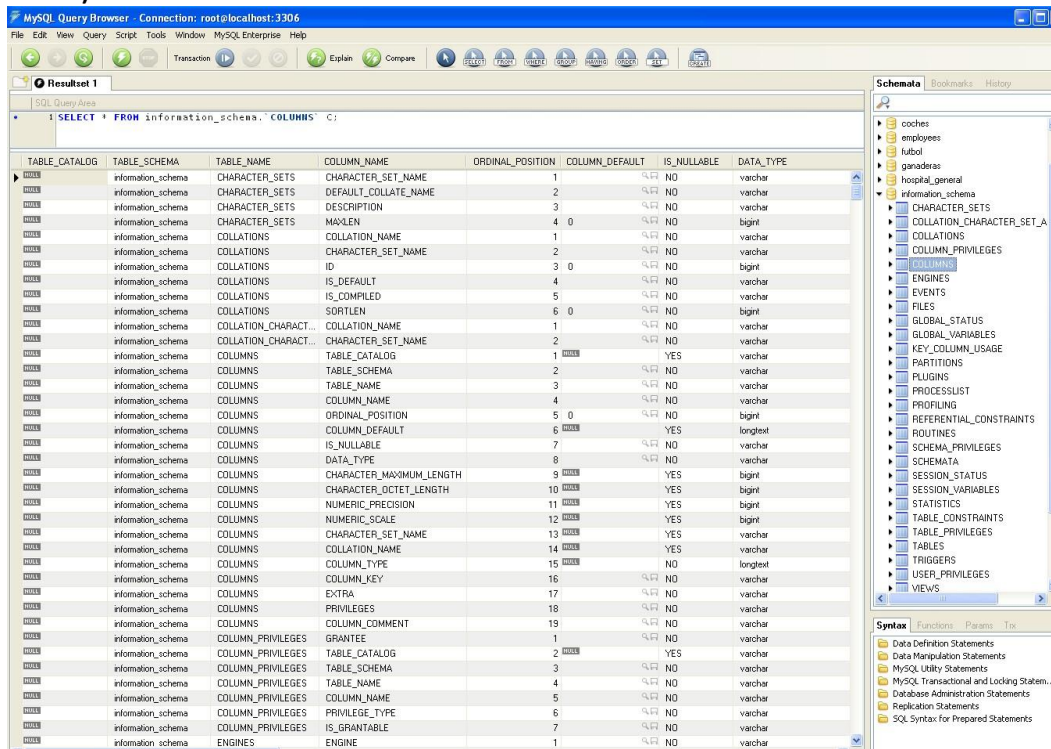
- La información sobre los datos se puede almacenar de un modo centralizado. Esto ayuda a mantener el control sobre los datos, como un recurso que son.
- El significado de los datos se puede definir, lo que ayudará a los usuarios a entender el propósito de los mismos.
- La comunicación se simplifica ya que se almacena el significado exacto. El diccionario de datos también puede identificar al usuario o usuarios que poseen los datos o que los acceden.
- Las redundancias y las inconsistencias se pueden identificar más fácilmente ya que los datos están centralizados.
- Se puede tener un historial de los cambios realizados sobre la base de datos.
- El impacto que puede producir un cambio se puede determinar antes de que sea implementado, ya que el diccionario de datos mantiene información sobre cada tipo de dato, todas sus relaciones y todos sus usuarios.
- Se puede hacer respetar la seguridad.
- Se puede garantizar la integridad.

Ejemplo del contenido del Diccionario de Datos de SGBD MySQL. Se encuentra en el Schema llamado information schema. La captura de pantalla nos muestra el contenido de la tabla llamada TABLES, que contiene la información acerca de todas las tablas de todos los esquemas de Bases de Datos que almacenamos!



TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE	ENGINE	VERSION	ROW_FORMAT	TABLE_ROWS	AVG_ROW_LENGTH	DATA_LENGTH	MAX_DATA_LENGTH	INDEX_LENGTH
mysql	coches	incidencias	BASE TABLE	InnoDB	10	Compact	8214	133	1593248	0	
mysql	coches	lineasproducto	BASE TABLE	InnoDB	10	Compact	15	1092	16384	0	
mysql	coches	marcacoches	BASE TABLE	InnoDB	10	Compact	112	146	16384	0	
mysql	coches	marcas	BASE TABLE	InnoDB	10	Compact	3	5461	16384	0	
mysql	coches	regiones	BASE TABLE	InnoDB	10	Compact	5	3276	16384	0	
mysql	employees	countries	BASE TABLE	InnoDB	10	Compact	4	4096	16384	0	
mysql	employees	departments	BASE TABLE	InnoDB	10	Compact	8	2048	16384	0	
mysql	employees	employees	BASE TABLE	InnoDB	10	Compact	20	819	16384	0	
mysql	employees	job_grades	BASE TABLE	InnoDB	10	Compact	6	2730	16384	0	
mysql	employees	job_history	BASE TABLE	InnoDB	10	Compact	10	1638	16384	0	
mysql	employees	jobs	BASE TABLE	InnoDB	10	Compact	12	1365	16384	0	
mysql	employees	locations	BASE TABLE	InnoDB	10	Compact	5	3276	16384	0	
mysql	employees	regions	BASE TABLE	InnoDB	10	Compact	4	4096	16384	0	
mysql	employees	parisios	BASE TABLE	InnoDB	10	Compact	25940	99	2637824	0	
mysql	ganaderias	aliment	BASE TABLE	MyISAM	10	Dynamic	9	44	400	201474976710...	
mysql	ganaderias	aliment_nutrient	BASE TABLE	MyISAM	10	Dynamic	9	33	300	201474976710...	
mysql	ganaderias	animal	BASE TABLE	MyISAM	10	Dynamic	6	24	148	201474976710...	
mysql	ganaderias	animal_nutrient	BASE TABLE	MyISAM	10	Dynamic	6	20	120	201474976710...	
mysql	ganaderias	contingut_dieta	BASE TABLE	MyISAM	10	Dynamic	9	32	288	201474976710...	
mysql	ganaderias	dieta	BASE TABLE	MyISAM	10	Dynamic	10	30	300	201474976710...	
mysql	ganaderias	dieta_animal	BASE TABLE	MyISAM	10	Dynamic	6	35	212	201474976710...	
mysql	ganaderias	puerbas_visita	BASE TABLE	MyISAM	10	Dynamic	8	25	204	201474976710...	
mysql	ganaderias	tona	BASE TABLE	MyISAM	10	Dynamic	9	32	288	201474976710...	
mysql	hospital_general	cama	BASE TABLE	InnoDB	10	Compact	26	630	16384	0	
mysql	hospital_general	cama_ingreso	BASE TABLE	InnoDB	10	Compact	15	1092	16384	0	
mysql	hospital_general	diagnosticos_enfermedad	BASE TABLE	InnoDB	10	Compact	10	1638	16384	0	
mysql	hospital_general	documentos	BASE TABLE	InnoDB	10	Compact	15	1092	16384	0	
mysql	hospital_general	ingreso	BASE TABLE	InnoDB	10	Compact	15	1092	16384	0	
mysql	hospital_general	medicos	BASE TABLE	InnoDB	10	Compact	5	3276	16384	0	
mysql	hospital_general	pacientes	BASE TABLE	InnoDB	10	Compact	5	3276	16384	0	
mysql	hospital_general	planta	BASE TABLE	InnoDB	10	Compact	6	2730	16384	0	
mysql	hospital_general	pruebas	BASE TABLE	InnoDB	10	Compact	5	3276	16384	0	
mysql	hospital_general	pruebas_visita	BASE TABLE	InnoDB	10	Compact	0	0	16384	0	
mysql	hospital_general	registros_visita	BASE TABLE	InnoDB	10	Compact	15	1092	16384	0	
mysql	hospital_general	visita_medica	BASE TABLE	InnoDB	10	Compact	0	0	16384	0	
mysql	mysql	column_priv	BASE TABLE	MyISAM	10	Fixed	0	0	0	227994731135...	
mysql	mysql	db	BASE TABLE	MyISAM	10	Fixed	2	440	880	12384989752...	
mysql	mysql	event	BASE TABLE	MyISAM	10	Dynamic	0	0	0	201474976710...	
mysql	mysql	func	BASE TABLE	MyISAM	10	Fixed	0	0	0	162974011515...	

La captura de pantalla nos muestra el contenido de la tabla llamada COLUMNS, que contiene la información acerca de todas las columnas de todas las tablas de todos los esquemas de Bases de Datos que almacenamos! En particular, la tabla COLUMNS también tiene columnas, y su propia descripción se encuentra en dicha tabla! Se está describiendo a sí misma (además de a todas las demás)



TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE
mysql	information_schema	CHARACTER_SETS	CHARACTER_SET_NAME	1		NO	varchar
mysql	information_schema	CHARACTER_SETS	DEFAULT_COLLATE_NAME	2		NO	varchar
mysql	information_schema	CHARACTER_SETS	DESCRIPTION	3		NO	varchar
mysql	information_schema	CHARACTER_SETS	MAXLEN	4	0	NO	bigint
mysql	information_schema	COLLATIONS	COLLATION_NAME	1		NO	varchar
mysql	information_schema	COLLATIONS	CHARACTER_SET_NAME	2		NO	varchar
mysql	information_schema	COLLATIONS	ID	3	0	NO	bigint
mysql	information_schema	COLLATIONS	IS_DEFAULT	4		NO	varchar
mysql	information_schema	COLLATIONS	IS_COMPILED	5		NO	varchar
mysql	information_schema	COLLATIONS	SORTLEN	6	0	NO	bigint
mysql	information_schema	COLLATION_CHARACTER_SET_APPLICABILITY	COLLATION_NAME	1		NO	varchar
mysql	information_schema	COLLATION_CHARACTER_SET_APPLICABILITY	CHARACTER_SET_NAME	2		NO	varchar
mysql	information_schema	COLUMNS	TABLE_CATALOG	1		YES	varchar
mysql	information_schema	COLUMNS	TABLE_SCHEMA	2		NO	varchar
mysql	information_schema	COLUMNS	TABLE_NAME	3		NO	varchar
mysql	information_schema	COLUMNS	COLUMN_NAME	4		NO	varchar
mysql	information_schema	COLUMNS	ORDINAL_POSITION	5	0	NO	bigint
mysql	information_schema	COLUMNS	COLUMN_DEFAULT	6		YES	longtext
mysql	information_schema	COLUMNS	IS_NULLABLE	7		NO	varchar
mysql	information_schema	COLUMNS	DATA_TYPE	8		NO	varchar
mysql	information_schema	COLUMNS	CHARACTER_MAXIMUM_LENGTH	9		YES	bigint
mysql	information_schema	COLUMNS	CHARACTER_OCTET_LENGTH	10		YES	bigint
mysql	information_schema	COLUMNS	NUMERIC_PRECISION	11		YES	bigint
mysql	information_schema	COLUMNS	NUMERIC_SCALE	12		YES	bigint
mysql	information_schema	COLUMNS	CHARACTER_SET_NAME	13		YES	varchar
mysql	information_schema	COLUMNS	COLLATION_NAME	14		YES	varchar
mysql	information_schema	COLUMNS	COLUMN_TYPE	15		NO	longtext
mysql	information_schema	COLUMNS	COLUMN_KEY	16		NO	varchar
mysql	information_schema	COLUMNS	EXTRA	17		NO	varchar
mysql	information_schema	COLUMNS	PRIVILEGES	18		NO	varchar
mysql	information_schema	COLUMNS	COLUMN_COMMENT	19		NO	varchar
mysql	information_schema	COLUMN_PRIVILEGES	GRANTEE	1		NO	varchar
mysql	information_schema	COLUMN_PRIVILEGES	TABLE_CATALOG	2		YES	varchar
mysql	information_schema	COLUMN_PRIVILEGES	TABLE_SCHEMA	3		NO	varchar
mysql	information_schema	COLUMN_PRIVILEGES	TABLE_NAME	4		NO	varchar
mysql	information_schema	COLUMN_PRIVILEGES	COLUMN_NAME	5		NO	varchar
mysql	information_schema	COLUMN_PRIVILEGES	PRIVILEGE_TYPE	6		NO	varchar
mysql	information_schema	COLUMN_PRIVILEGES	IS_GRANTABLE	7		NO	varchar
mysql	information_schema	ENGINES	ENGINE	1		NO	varchar

La misma tabla COLUMNS, pero la parte que nos muestra la información de las columnas del Schema employees:

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	CHARACTER_SET_NAME
mysql	employees	departments	manager_id	3	NULL	YES	int		utf8
mysql	employees	departments	location_id	4	NULL	YES	int		utf8
mysql	employees	employees	employee_id	1	NULL	NO	int		utf8
mysql	employees	employees	first_name	2	NULL	YES	varchar	20	utf8
mysql	employees	employees	last_name	3	NULL	NO	varchar	25	utf8
mysql	employees	employees	email	4	NULL	NO	varchar	25	utf8
mysql	employees	employees	phone_number	5	NULL	YES	varchar	20	utf8
mysql	employees	employees	hire_date	6	NULL	NO	datetime		utf8
mysql	employees	employees	job_id	7	NULL	NO	varchar	10	utf8
mysql	employees	employees	salary	8	NULL	YES	decimal		utf8
mysql	employees	employees	commission_pct	9	NULL	YES	decimal		utf8
mysql	employees	employees	manager_id	10	NULL	YES	int		utf8
mysql	employees	employees	department_id	11	NULL	YES	int		utf8
mysql	employees	job_grades	grade_level	1	NULL	NO	varchar	3	utf8
mysql	employees	job_grades	lowest_sal	2	NULL	YES	decimal		utf8
mysql	employees	job_grades	highest_sal	3	NULL	YES	decimal		utf8
mysql	employees	job_history	employee_id	1	NULL	NO	int		utf8
mysql	employees	job_history	start_date	2	NULL	NO	datetime		utf8
mysql	employees	job_history	end_date	3	NULL	NO	datetime		utf8
mysql	employees	job_history	job_id	4	NULL	NO	varchar	10	utf8
mysql	employees	job_history	department_id	5	NULL	YES	int		utf8
mysql	employees	jobs	job_id	1	NULL	NO	varchar	10	utf8
mysql	employees	jobs	job_title	2	NULL	NO	varchar	35	utf8
mysql	employees	jobs	min_salary	3	NULL	YES	int		utf8
mysql	employees	jobs	max_salary	4	NULL	YES	int		utf8
mysql	employees	locations	location_id	1	NULL	NO	int		utf8
mysql	employees	locations	street_address	2	NULL	YES	varchar	40	utf8
mysql	employees	locations	postal_code	3	NULL	YES	varchar	12	utf8
mysql	employees	locations	city	4	NULL	NO	varchar	30	utf8
mysql	employees	locations	state_province	5	NULL	YES	varchar	25	utf8
mysql	employees	locations	country_id	6	NULL	YES	char	2	utf8
mysql	employees	regions	region_id	1	NULL	NO	decimal		utf8
mysql	employees	regions	region_name	2	NULL	YES	varchar	25	utf8
mysql	employees	partidos	compart	1	NULL	NO	bigint		utf8
mysql	employees	partidos	temporada	2	NULL	YES	varchar	16	utf8
mysql	employees	partidos	division	3	NULL	YES	varchar	20	utf8
mysql	employees	partidos	jornada	4	NULL	YES	int		utf8
mysql	employees	partidos	plocal	5	NULL	YES	varchar	40	utf8
mysql	employees	partidos	visitante	6	NULL	YES	varchar	40	utf8

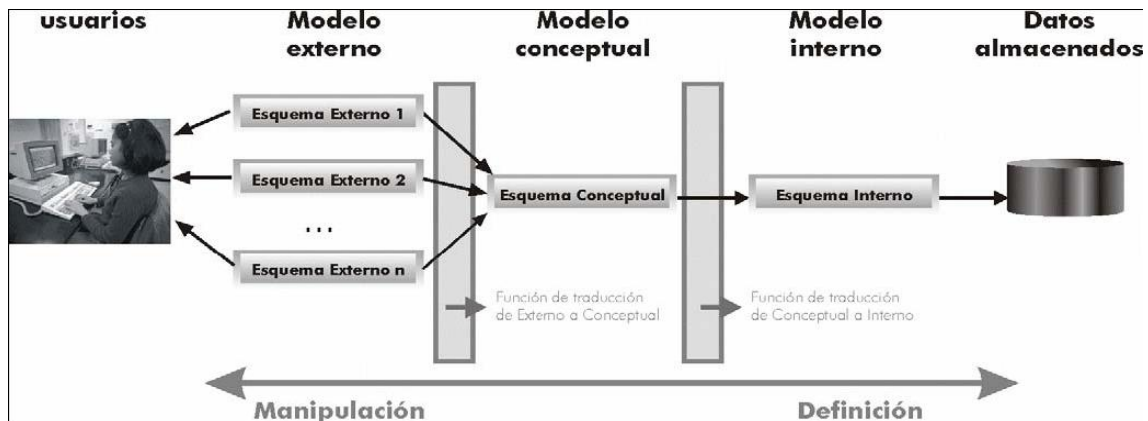
Esta captura NO forma parte del Diccionario de Datos. Es el contenido de la tabla employees, del Schema employees. Son sus DATOS. Pero la ponemos aquí para comprobar que la primera captura anterior (del diccionario de datos) guardaba información sobre esta tabla. Por ejemplo, allí se decía que tenía 20 Registros (columna TABLE_ROWS), y efectivamente, aquí están:

employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id
100	Steven	King	SKING	515.123.4567	1987-06-17 00:00:00	AD_PRES	24000.00	NULL	NULL
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-09-21 00:00:00	AD_VP	17000.00	NULL	100
102	Lex	De Haan	LDEHAAN	515.123.4569	1993-01-13 00:00:00	AD_VP	17000.00	NULL	100
103	Alexander	Hunold	AHUNOLD	590.423.4567	1990-01-03 00:00:00	IT_PROG	9000.00	NULL	100
104	Bruce	Ernst	BERNST	590.423.4568	1991-05-21 00:00:00	IT_PROG	6000.00	NULL	100
107	Diana	Lorentz	DLORENTZ	590.423.5567	1999-02-07 00:00:00	IT_PROG	4200.00	NULL	100
124	Kevin	Mouergos	KMOURGOS	650.123.5234	1999-11-16 00:00:00	ST_MAN	5800.00	NULL	100
141	Trenna	Rajs	TRAJS	650.121.8009	1995-10-17 00:00:00	ST_CLERK	3500.00	NULL	100
142	Curtis	Davies	CDAVIES	650.121.2994	1997-01-29 00:00:00	ST_CLERK	3100.00	NULL	100
143	Randall	Matos	RMATOS	650.121.2874	1998-03-15 00:00:00	ST_CLERK	2600.00	NULL	100
144	Peter	Vargas	PVARGAS	650.121.2004	1998-07-09 00:00:00	ST_CLERK	2500.00	NULL	100
149	Eleri	Zlotkey	EZLOTKEY	011.44.1344.429018	2000-01-29 00:00:00	SA_MAN	10500.00	0.20	100
174	Ellen	Abel	EABEL	011.44.1644.429267	1996-05-11 00:00:00	SA_REP	11000.00	0.30	100
176	Jonathan	Taylor	JTAYLOR	011.44.1644.429265	1998-03-24 00:00:00	SA_REP	8600.00	0.20	100
178	Kimberely	Grant	KGRANT	011.44.1644.429263	1999-05-24 00:00:00	SA_REP	7000.00	0.15	100
200	Jennifer	Whalen	JWHALEN	515.123.4444	1987-09-17 00:00:00	AD_ASST	4400.00	NULL	100
201	Michael	Hartstein	MHARTSTE	515.123.5555	1996-02-17 00:00:00	MK_MAN	13000.00	NULL	100
202	Pat	Fay	PFAY	603.123.6666	1997-08-17 00:00:00	MK_REP	6000.00	NULL	100
205	Shelley	Higgins	SHIGGINS	515.123.8080	1994-06-07 00:00:00	AC_MGR	12000.00	NULL	100
206	William	Gietz	WGIEZT	515.123.8181	1994-06-07 00:00:00	AC_ACCOUNT	8300.00	NULL	100

Además, la tercera captura nos mostraba la información de sus columnas, y podemos comprobar las características que allí nos decía.

8. Arquitectura de los SGBD: ANSI/X3/SPARC

Existen en el mercado varios paquetes de SGBD con diferentes arquitecturas. La más estandarizada es la que cumple los requerimientos de la normativa ANSI/X3/SPARC surgida en 1977, esta establece que la arquitectura de una BD debe poseer tres niveles de abstracción.



En el **modelo ANSI/X3/SPARC** se indica que hay tres modelos (**externo, conceptual e interno**) entendiendo por modelo las 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 los datos y relaciones de la base de datos y el **esquema interno** la preparación de los datos para ser almacenados.

Los **esquemas externos** describe solamente una parte de la BD, se define para cada usuario que lo necesite un subesquema de la BD.

El **esquema conceptual** contiene la información lógica de la base de datos. Su estructuración y las relaciones que hay entre los datos. Se trata de la propuesta teórica de los datos (es quizá la más importante).

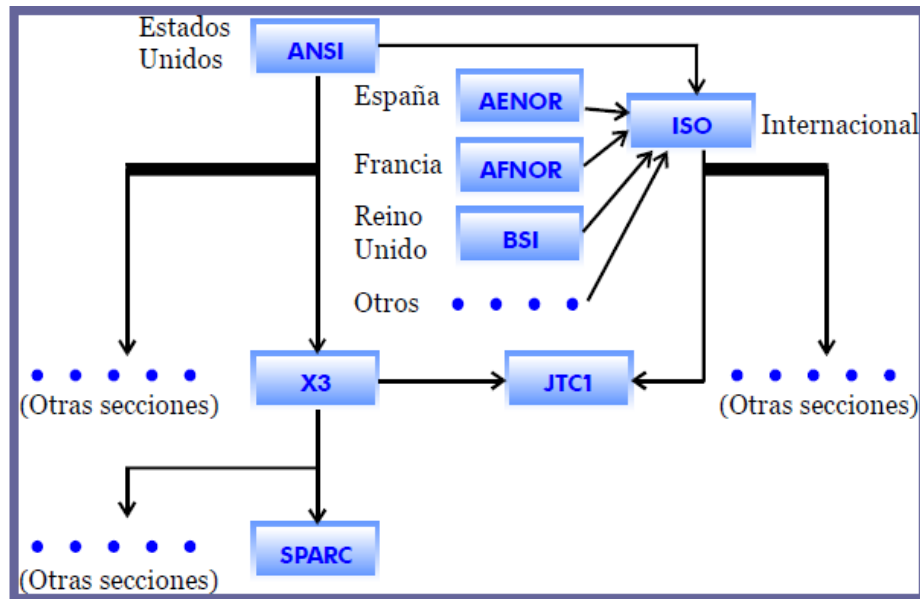
El **esquema interno** contiene información sobre cómo están almacenados los datos en disco. Es el esquema más cercano a la organización real de los datos.

Observar que puede haber varios esquemas externos, pero sólo puede haber uno conceptual y uno interno

El SGBD debe garantizar la transferencia de datos entre estos niveles.

ANSI (**American National Science Institute**) es un organismo científico de Estados Unidos que ha definido diversos estándares en el campo de las bases de datos. **X3** es la parte de ANSI encargada de los estándares en el mundo de la electrónica. Finalmente **SPARC**, **System Planning and Reparatments Committee**, comité de planificación de sistemas y reparaciones es una subsección de X3 encargada de los estándares en Sistemas Informáticos en especial del campo de las bases de datos. Su logro fundamental ha sido definir un modelo de referencia para las bases de datos

Relación entre los organismos de Estandarización



Independencia de los datos, es la capacidad para modificar un esquema de definición sin afectar a los programas de aplicación.

Existen dos niveles de independencia:

- **Física**, cuando es posible modificar el esquema físico sin que afecte a las aplicaciones.
- **Lógica**, cuando es posible modificar el esquema conceptual sin obligar a escribir de nuevo las aplicaciones.

A diferencia de los sistemas de ficheros, el SGBD gestiona la estructura física de los datos y 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.

Las vistas, además de reducir la complejidad permitiendo que cada usuario vea sólo la parte de la base de datos que necesita, **tienen otras ventajas**:

- Las vistas proporcionan un nivel de seguridad, ya que permiten excluir datos para que ciertos usuarios no los vean.
- Las vistas proporcionan un mecanismo para que los usuarios vean los datos en el formato que deseen.
- Una vista representa una imagen consistente y permanente de la base de datos, incluso si la base de datos cambia su estructura.

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 y muchas 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 fallo *hardware* o *software*. Los SGBD están en continua evolución, tratando de satisfacer los requerimientos 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 cambiar. Conforme vaya pasando el tiempo irán surgiendo nuevos requisitos, por lo que los SGBD nunca permanecerán estáticos.

9. Bases de Datos DISTRIBUIDAS

Originalmente se almacenaba la información de manera **centralizada**: los datos se almacenan en un solo computador. Los SGBD centralizados pueden atender a varios usuarios, pero el SGBD y la base de datos en sí residen por completo en una sola máquina.

Pero con el paso del tiempo las necesidades aumentaron y esto produjo ciertos inconvenientes que no era posible solucionarlos o volverlos eficientes de la forma centralizada. Estos problemas impulsaron la creación de **almacenamiento distribuido**, los cuales hoy en día proveen características indispensables en el manejo de información; es decir, la combinación de las redes de comunicación y las bases de datos

9.1 BDD: Base de Datos Distribuida

Una **base de datos distribuida (BDD)** es un conjunto de múltiples bases de datos lógicamente relacionadas las cuales se encuentran distribuidas en diferentes espacios lógicos y geográficos (pej. un servidor corriendo 2 máquinas virtuales) e interconectados por una red de comunicaciones.

Un sistema distribuido de bases de datos se almacena en varias computadoras.

Los principales factores que distinguen un SBDD de un sistema centralizado son los siguientes:

- Hay múltiples computadores, llamados sitios o **nodos**.
- Estos nodos deben de estar comunicados por medio de algún tipo de red de comunicaciones para transmitir datos y órdenes entre los sitios.

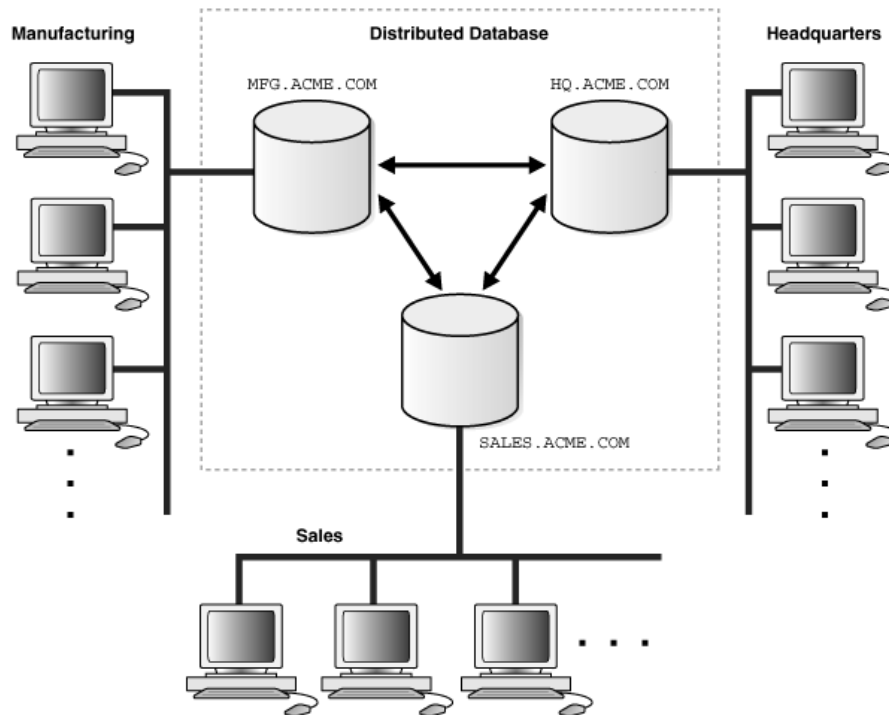
Los SGBD *distribuidos* **Homogéneos** utilizan el mismo SGBD en múltiples sitios (por ejemplo: Oracle en todos los nodos)

Los SGBD *distribuidos* **Heterogéneos** utilizan diferentes SGBD en varios nodos (por ejemplo: algunos nodos pueden tener Oracle, otros SQLServer, otros PostgreSQL, y otros MySQL. Y todos juntos forman la Base de Datos Distribuida)

Un objetivo básico de SGBD distribuido es que el usuario lo perciba como si de una base de datos centralizada se tratase.

La principal **ventaja** SGBD *distribuido* es la capacidad de compartir y acceder a la información de manera eficiente.

Como **desventajas** presenta mayor complejidad y coste de software y mayor posibilidad de errores: se vuelve más difícil la gestión de la concurrencia, las transacciones, la seguridad, la integridad.



9.2 Clasificación según el tipo de distribución de los datos.

Replicadas: consiste en que cada nodo debe tener su copia completa de la base de datos

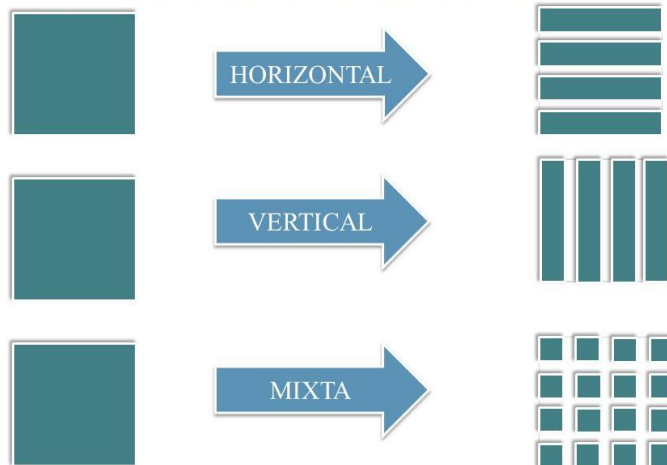
este esquema tiene un alto costo en el almacenamiento de la información. Debido a que la actualización de los datos debe ser realizada en todas las copias, también tiene un alto costo de escritura, pero todo esto vale la pena si tenemos un sistema en el que se va a escribir pocas veces y leer muchas, y dónde la disponibilidad y fiabilidad de los datos sea de máxima importancia

Particionadas (o Fragmentada): consiste en que sólo hay una copia de cada elemento, pero los datos están repartidos a través de los nodos. Esta fragmentación se puede realizar de tres formas:

- Horizontal
- Vertical
- Mixta

Híbridas: es la combinación del esquema de partición y replicación: se particionan los datos y a la vez los fragmentos pueden encontrarse replicados en varios nodos.

TIPOS DE FRAGMENTACIÓN:



Fragmentación **Horizontal** de una tabla:

S

SNUM	NOMBRE	CD
S1	X	L
S2	Y	L
S3	Z	P
S4	A	P
S5	B	L

Una posible fragmentación Horizontal de esta tabla sería la sig:

FH1 = S WHERE CD='L'
FH2 = S WHERE CD='P'

SNUM	NOMBRE	CD
S1	X	L
S2	Y	L
S5	B	L

SNUM	NOMBRE	CD
S3	Z	P
S4	A	P

Fragmentación **Vertical** de una tabla:

Ejemplo:

E

EMP#	NOMBRE	SALARIO	IMPTO.	#JEFE	DEPT#
e1	X	1000	100	J1	D1
e2	Y	1500	300	J1	D1
e3	Z	500	20	J2	D2
e4	A	4000	1000	J3	D3
e5	B	2000	350	J2	D2

EMP#	NOMBRE	#JEFE	DEPT#
e1	X	J1	D1
e2	Y	J1	D1
e3	Z	J2	D2
e4	A	J3	D3
e5	B	J2	D2

EMP#	SALARIO	IMPTO
e1	1000	100
e2	1500	300
e3	500	20
e4	4000	1000
e5	2000	350

FV1 = E[EMP#, NOMBRE, #JEFE, DEPT#] **FV2** = E[EMP#, SALARIO, IMPTO]

10. El Modelo ER: "Entity-Relationship"

En 1976: P. Chen propone el **Modelo ER** ("Entity-Relationship" o Entidad Interrelación) para el **DISEÑO DE Bases de Datos**.

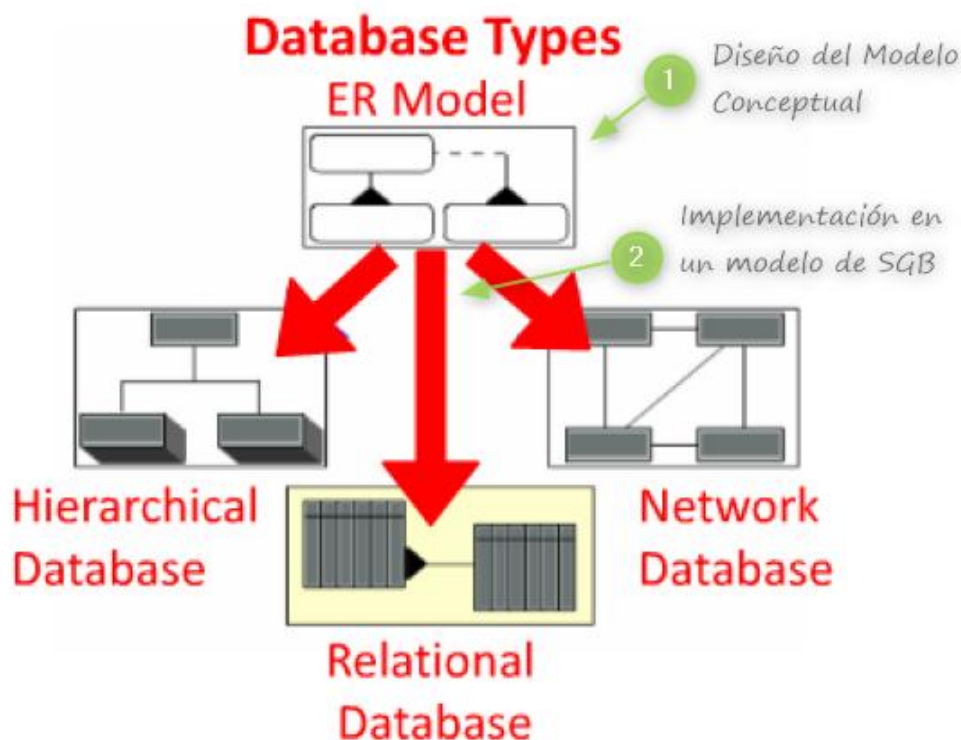
(Recuerda que el Modelo Relacional apareció en 1972, propuesto por E.F.Codd)

El **modelo ER** es una especie de "**lenguaje gráfico**" que nos ayuda a **Diseñar Bases de Datos**, independientemente del Modelo de SGBD en el que la vayamos a implementar.

El modelo ER nos servirá para crear un diagrama conceptual o lógico de nuestra Base de Datos.

Observación Importante: El **Modelo ER NO ES un Modelo de SGBD** (es decir, no está al mismo nivel que el Relacional, Jerárquico, en Red, ...).

Un buen modelo conceptual seguirá siendo el mismo independientemente del tipo de base de datos (es decir modelo de Sistema Gestor) en el que vayamos a implementarla. El diseño ER debe ser un paso previo a la implementación física en un SGBD.



11. Bibliografía

<https://en.wikipedia.org/wiki/Database>

<https://en.wikipedia.org/wiki/NoSQL>

<https://uvfdatabases.wordpress.com/2009/02/02/sistemas-de-bases-de-datos-vs-sistemas-de-archivos/>

IOC

https://ioc.xtec.cat/materials/FP/Materials/2252_DAM/DAM_2252_M02/web/html/media/fp_dam_m02_u1_pdfindex.pdf

<https://en.wikipedia.org/wiki/IDMS>

https://en.wikipedia.org/wiki/Integrated_Data_Store

http://docs.oracle.com/cd/B28359_01/server.111/b28310/ds_concepts001.htm

<http://www.jorgesanchez.net/bd/gbd2012.pdf>

<https://medium.com/omarelgabrys-blog/database-database-options-part-10-380c6e4467d0>