INDICE

1. Definición de Bases de Datos y SGBD

En primer lugar, es importante diferenciar entre el concepto de base de datos y el de sistema gestor, ya que es habitual confundirlos y sin embargo son cosas muy distintas.

- Una **base de datos (BD)** *es un conjunto de datos relacionados y organizados con cierta estructura*. Según dicha organización distinguimos entre diferentes modelos de bases de datos como el relacional, jerárquico o en red.

El modelo de bases de datos más extendido es el **relacional**. Para su manipulación y gestión surgieron los sistemas gestores de bases de datos (SGBD).

- El **sistema de gestión de la base de datos (SGBD)** *es una aplicación que permite a los usuarios definir, crear y mantener bases de datos, proporcionando acceso controlado a las mismas*. Es una herramienta que sirve de interfaz entre el usuario y las bases de datos.

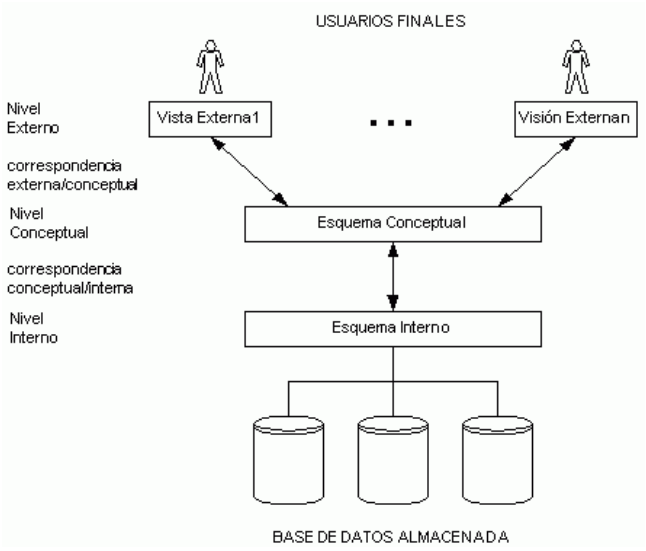
Es decir, por un lado tenemos los datos organizados según ciertos criterios y, por otro, un software que nos permite o facilita su gestión con distintas herramientas y funcionalidades que describimos a continuación.

2. Estructura de un SGBD

2.1. Sistemas Gestores de Bases de Datos

Un **Sistema Gestor de Base de Datos** *es el software que permite gestionar bases de datos, ocultando la física de la misma y permitiendo manejarla desde un nivel más conceptual*. Dicho software permite separar las aplicaciones (los programas) de los datos; de modo que los programas negocian con el SGBD el acceso a los datos.

En definitiva se trata de un software complejo, pero de gran importancia por lo delicado de la rama de la información a la que se dedica. Los SGBD han crecido de manera exponencial estos últimos años por el éxito de Internet, que ha provocado el acceso a miles y miles de bases de datos por parte de millones de usuarios cada día.



2.2. Niveles conceptuales. Modelo ANSI[1]/X3/SPARC

El grupo de trabajo **SPARC** de la sección X3 de ANSI, diseño un modelo en el que indicaba cómo debía funcionar un SGBD para asegurar la independencia entre datos y aplicaciones y así especificó **tres niveles**:

1. **Nivel externo**. Define el nivel en el que los usuarios utilizan la base de datos. La forma de ver la misma que no tiene nada que ver con la estructura real de la base de datos. Los usuarios manejan los esquemas externos de este nivel, pero dichos esquemas los realizan los desarrolladores/as de aplicaciones.

[1] http://es.wikipedia.org/wiki/Instituto_Nacional_Estadounidense_de_Est%C3%A1ndares

2. **Nivel físico**. Se refiere a la forma en la que realmente se almacena la información de la base de datos. Los administradores de la base de datos (DBA) son los encargados de manejar este nivel.
3. **Nivel conceptual**. Define la base de datos haciendo referencia a la forma en la que se relaciona la información.

Hoy en día se definen más niveles de modo que se habla de cinco niveles en realidad. Empezando desde el más cercano al usuario:

1. **Nivel externo**. Tal cual se explicó antes. En realidad los esquemas de este nivel son los últimos que se crean y lo hacen programadoras/es y analistas bajo la dirección de las/os analistas.
2. **Nivel conceptual**. Con la misma idea indicada anteriormente. En realidad el esquema (los planos) conceptual de la base de datos es lo primero que se diseña por los o las analistas o diseñadores de la misma utilizando un modelo para realizar los esquemas. El **MER** sigue siendo el modelo más popular.
3. **Nivel lógico**. Acerca más el esquema anterior a la física de la base de datos. En este nivel se hace referencia a estructuras lógicas del tipo de SGBD a manejar (tablas, filas, columnas por ejemplo en el modelo relacional de bases de datos, el modelo lógico más utilizado). Este nivel sigue siendo manejado por los analistas. En muchos casos (aunque ciertamente es peligroso) los diseñadores/as de la base de datos empiezan por este nivel saltándose el anterior. En la actualidad el **Modelo Relacional** sigue siendo el modelo más habitual para crear esquemas a nivel lógico.
4. **Nivel interno**. Es el primero en el proceso de modelado de la base de datos que se realiza sobre el software gestor de la base de datos (teniendo en cuenta que las aplicaciones, se crean más tarde). **Usa el lenguaje de la base de datos para crear las estructuras de datos definidas en el nivel lógico**. Este nivel lo maneja el administrador de la base de datos (o DBA).
5. **Nivel físico**. Se refiere a como se organizarán los datos en el disco, en que ordenadores se crea la base de datos, si es distribuida o no, sistema operativo necesario, estructura de directorios y archivos, configuración de servidores y sistema operativo, política de copia de seguridad,…

2.3. Funciones del SGBD

El manejo de la SGBD implica que nos permita realizar estas funciones:

- **Catálogo**

  Se denomina ***diccionario de datos*** y contiene información que describe los datos de la base de datos (metadatos) y debe ser accesible por los usuarios de la base de datos. Normalmente, un diccionario de datos describe entre otras cosas:
  - Nombre, tipo y tamaño de los datos.
  - Relaciones entre los datos.
  - Restricciones de integridad.
  - Usuarios autorizados a acceder a los objetos de la base de datos.
  - Estadística de utilización, tales como la frecuencia de las transacciones y el número de accesos realizados a los objetos de la base de datos.

- **Garantizar la integridad,** disponer de un mecanismo que garantice que todas las actualizaciones correspondientes a una determinada transacción se realicen, o que no se realice ninguna.
- **Permitir actualizaciones,** asegurar que la base de datos se actualice correctamente cuando varios usuarios la están actualizando concurrentemente. Uno de los principales objetivos de los SGBD es el permitir que varios usuarios tengan acceso concurrente a los datos que comparten.
- **Recuperación de datos,** permitir recuperar la base de datos en caso de que ocurra algún suceso que la dañe. Debe proporcionar un mecanismo capaz de recuperar la base de datos llevándola a un estado consistente.
- **Integración,** ser capaz de integrarse con algún software de comunicación. Muchos usuarios acceden a la base de datos desde terminales.
- **Cumplir restricciones,** proporcionar los medios necesarios para garantizar que tanto los datos de la base de datos, como los cambios que se realizan sobre estos datos, sigan ciertas reglas.
- **Herramientas de administración,** proporcionar herramientas que permitan administrar la base de datos de modo efectivo, lo que implica un diseño óptimo de las mismas, garantizar la disponibilidad e integridad de los datos, controlar el acceso a al servidor y a los datos, monitorizar el funcionamiento del servidor y optimizar su funcionamiento. Muchas de ellas van integradas en el sistema gestor, otras son creadas por terceros o por el propio administrador según sus requerimientos.

2.4. Componentes de un SGBD

Son los elementos que deben proporcionar los servicios comentados en la sección anterior. No se puede generalizar ya que varían mucho según la tecnología. Sin embargo, es muy útil conocer sus componentes y cómo se relacionan cuando se trata de comprender lo que es un sistema de bases de datos.
El SGBD es la aplicación que interacciona con los usuarios de los programas de aplicación y la base de datos.

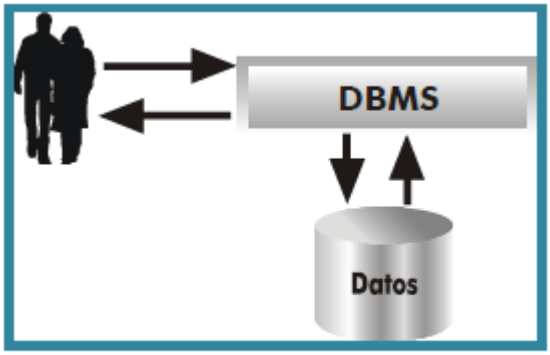


Ilustración 7, Esquema del funcionamiento y utilidad de un sistema gestor de bases de datos

2.4.1. El lenguaje SQL

La principal herramienta de un SGBD es la interfaz de programación con el usuario. Este interfaz consiste en un lenguaje muy sencillo mediante el cual el usuario realiza preguntas al servidor, contestando este a las demandas del usuario. Este lenguaje comúnmente se denomina SQL, (Structured Query Language, ***Lenguaje de Consulta Estructurado***), está estandarizado por la ISO[2], es decir, todas las bases de datos que soporten

[2] **ISO**, es el acrónimo de International Organization for Standardization, Organización Internacional de Normalización.

SQL deben tener la misma sintaxis a la hora de aplicar el lenguaje. Se divide en 4 sublenguajes, la totalidad de todos ellos permite al SGBD cumplir con las funcionalidades requeridas por CODD:

- **Función de descripción o definición**. Permite al diseñador de la base de datos crear las estructuras apropiadas para integrar adecuadamente los datos. Esta función se realiza mediante el **Lenguaje de Definición de Datos** o **DDL (Data Definition Language**). Mediante ese lenguaje:
  - ✓ Se definen las estructuras de datos (los metadatos).
  - ✓ Se definen las relaciones entre ellas.
  - ✓ Se definen las reglas que han de cumplir.

  COMANDOS DDL: CREATE, ALTER, DROP

- **Función de manipulación**. Permite modificar y utilizar los datos de la base de datos. Se realiza mediante el **Lenguaje de Manipulación de Datos** o **DML (Data Manipulation Language).** Mediante ese lenguaje se puede:
  - ✓ Añadir datos.
  - ✓ Eliminar datos.
  - ✓ Modificar datos.
  - ✓ Buscar datos.

  COMANDOS DML: INSERT, UPDATE, DELETE

  - ✓ **Lenguaje de Consulta de Datos o DQL (Data Query Language).** Actualmente se suele distinguir la función de buscar datos en la base de datos (**función de consulta**).

    COMANDO DQL: SELECT

- **Lenguaje de Control de Transacciones o DTL (Data TransactionLanguage)**. El propósito de este lenguaje es permitir ejecutar varios comandos de forma simultánea como si fuese un comando atómico o indivisible.

  COMANDOS DTL: START TRANSACTION, SAVEPOINT, COMMIT, ROLLBACK

- **Función de control**. Mediante esta función los administradores poseen mecanismos para proteger las visiones de los datos permitidas a cada usuario, además de proporcionar elementos de creación y modificación de esos usuarios. El lenguaje que implementa esta función es el **lenguaje de control de datos** o **DCL.**

  COMANDOS DCL: GRANT, REVOKE

2.5. Funciones avanzadas de un SGBD

Para poder garantizar que una SGBD cumple las funciones anteriores de la mejor forma posible y además seguir garantizando la independencia entre los tres esquemas fundamentales (externo, conceptual y físico) y además facilitar su manejo, los SGBD tienen que cumplir de forma estricta una serie de reglas.

Edgar F. Codd (que teorizó el modelo relacional de la base de datos) escribió 12 reglas que habían de cumplir todos los SGBD. Hoy en día casi todas las grandes bases de datos son relacionales y los SGBD se afanan en

cumplir esas reglas, aunque con matices, puesto que la realidad actual está haciendo quedar a este modelo un tanto obsoleto en algunos aspectos.

Hoy en día las **funciones que se esperan de un buen SGBD** son:

- **Lenguaje que permita crear todos los elementos de la base de datos y gestionar el diccionario de datos.** Normalmente este lenguaje será SQL (aunque cada SGBD impone variantes al SQL estándar).
- **Herramientas gráficas** que faciliten muchas tareas habituales tanto de gestión como de administración del sistema.
- Posibilidad de **establecer reglas de integridad avanzada** e incluirlas como parte de la base de datos. Especialmente complicado suele ser poder establecer las reglas de integridad referencial (**foreign key**), por lo que algunos SGBD más simples no lo incluyen. Dentro de estas reglas están las restricciones estándar como: **UNIQUE** (unicidad, prohibir repetición de valores), **CHECK** (cumplimiento de condiciones simples), **NOT NULL** (obligatoriedad), **PRIMARY KEY** (establecimiento de las claves de las tablas) o la propia **FOREIGN KEY** (clave foránea); pero también restricciones más complejas como las que establecen los **Triggers** de los lenguajes procedimientales (como **PL/SQL**) presentes en la mayoría de sistemas.
- Gestión de **copias de seguridad**. Una de las funciones críticas de la base de datos ya que permite la recuperación de información en caso de problemas.
- Aplicaciones de **exportación e importación de datos**. Para poder utilizar datos de otros SGBD u otro software.
- Posibilidad de **recuperación en caso de desastre**. Para evitar perder información en caso de problemas serios con el software (errores de hardware, apagones prolongados,…)
- **Archivos LOG**. Desde los que podemos examinar las incidencias y monitorizar el funcionamiento de la base de datos.
- **Herramientas para programar aplicaciones**. Que permitan crear las aplicaciones (o facilidades) de usuario.
- **Gestión de la comunicación** con los clientes de la base de datos. Permiten establecer conexión con la base de datos desde máquinas remotas.
- **Optimización de consultas**. Busca el mínimo tiempo de respuesta para las operaciones sobre los datos.
- **Herramientas para automatizar tareas**. Permiten realizar programaciones sobre operaciones habituales sobre la base de datos.
- Posibilidad de **distribuir la base de datos** entre diferentes máquinas, y así mejorar su alta disponibilidad.
- **Gestión de transacciones**, **ACID**. Es una norma obligatoria que deben de cumplir las bases de datos para que una transacción se pueda considerar como tal.

EJERCICIO: Explica brevemente el significado de cada término que forma el acrónimo ACID en el contexto de los sistemas gestores de bases de datos.

2.6. Usuarios de un SGBD

Generalmente distinguimos cuatro grupos de usuarios de sistemas gestores de bases de datos: los usuarios administradores, los diseñadores de la base de datos, los programadores y los usuarios de aplicaciones que interactúan con las bases de datos.

- **Administrador de la base de datos**

Se encarga del diseño físico de la base de datos y de su implementación, realiza el control de la seguridad y de la concurrencia, mantiene el sistema para que siempre se encuentre operativo y se encarga de que los usuarios y las aplicaciones obtengan buenas prestaciones. El administrador debe conocer muy bien el SGBD que se esté utilizando, así como el equipo informático sobre el que esté funcionando.

- **Diseñadores de la base de datos**

Realizan el diseño lógico de la base de datos, debiendo identificar los datos, las relaciones entre datos y las restricciones sobre los datos y 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 de los datos tal y como los ve la empresa. Para obtener un buen resultado, el diseñador de la base de datos debe implicar en el desarrollo del modelo de datos a todos los usuarios de la base de datos, tan pronto como sea posible.
El diseño lógico de la base de datos es independiente del SGBD concreto que se vaya a utilizar, es independiente de los programas de aplicación, de los lenguajes de programación y de cualquier otra consideración física.

- **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.

- **Usuarios finales**

Clientes de la base de datos que hacen uso de ella sin conocer en absoluto su funcionamiento y organización. Son personas con pocos o nulos conocimientos de informática.

2.7. Tareas del DBA

Un administrador de bases de datos (DBA), tiene una serie de tareas asignadas. Todas ellas son de vital importancia para la base de datos; algunas son especialmente críticas. Las tareas más comúnmente aceptadas como parte de la profesión del DBA son:

(1) **Configurar e instalar el hardware necesario**. Para el correcto funcionamiento del SGBD. Eso incluye ampliar memoria, discos duros,… además de configurarles para su óptimo rendimiento. También la gestión mínima del sistema operativo para que la base de datos funcione correcta y rápidamente.

(2) **Instalación y mantenimiento del SGBD**. Seleccionando la más adecuada forma de instalación y configurando lo necesario para su óptimo rendimiento acorde con las necesidades, así como realizar las actualizaciones del sistema que sean necesarias.

(3) **Crear las estructuras de almacenamiento de la base de datos**. Es quizá la tarea que de forma más habitual se relaciona con los DBA. Consiste en crear y configurar las estructuras físicas y los elementos lógicos que permitan un rendimiento optimizado de la base de datos.

(4) **Crear y configurar la base de datos.** Creación de la estructura interna de la base de datos (tablas, usuarios, permisos, vistas,…). Es otra de las tareas más habitualmente relacionadas con el DBA y la primera fase (y las más crítica) en a administración de una base de datos.

(5) **Control de los usuarios y los permisos.** En definitiva establecer las políticas de seguridad tan imprescindibles en toda base de datos.

(6) **Monitorizar y optimizar el rendimiento de la base de datos**. Un DBA debe detectar los ***cuellos de botella*** del sistema y actuar en consecuencia. Esto incluye optimizar las instrucciones SQL por lo que implica asistir a los desarrolladores para que utilicen las instrucciones más eficientes sobre las bases de datos.

(7) **Realizar tareas de copia de seguridad y recuperación**. Quizá la tarea más crítica. Consiste en realizar acciones para en caso de catástrofe poder recuperar todos los datos.

3. Opciones de funcionamiento de un SGBD

3.1. SGBD monocapa

Se trata de SGBD instalados en una máquina desde la que se conectan los propios usuarios y administradores. Es un modelo que solo se utiliza con bases de datos pequeñas y poca cantidad de conexiones.

3.2. SGBD de dos capas

Es el modelo tipo **cliente/servidor**. La base de datos y el sistema gestor se alojan en un servidor al cual se conectan los usuarios desde máquinas clientes. Un software de comunicaciones se encarga de permitir el acceso a través de la red. Los clientes deben instalar el software cliente de acceso según las instrucciones de configuración del administrador.

Hay dos posibilidades:

- Arquitectura **cliente/servidor único**. Un solo servidor gestiona la base de datos, todos los clientes se conectan a él pare realizar las peticiones a la base de datos.
- Arquitectura **cliente/multiservidor**. La base de datos se distribuye entre varios servidores. El cliente no sabe realmente a que servidor se conecta, el software de control de comunicaciones se encargará de dirigirle al servidor adecuado. De forma lógica, es como si se tratara de un solo servidor aunque físicamente sean muchos.

3.3. SGBD de tres o más capas

En este caso entre el cliente y el servidor hay al menos una capa intermedia (puede haber varias). Esa capa (o capas) se encargan de recoger las peticiones de los clientes y luego de comunicarse con el servidor (o servidores) para recibir la respuesta y enviarla al cliente.

El caso típico es que la capa intermedia sea un servidor web, que recibe las peticiones a través de páginas web; de este modo para conectarse a la base de datos, el usuario solo requiere un navegador web, que es un software muy habitual en cualquier máquina.
Este modelo es el que más se está potenciando en la actualidad por motivos de seguridad y portabilidad de la base de datos.

4. Sistemas Gestores de Bases de Datos Comerciales

4.1. Licencias de software

El gurú del software libre, **Richard Stallman** considera al **software propietario** (software cuyo uso y explotación se rige por un contrato propio de la empresa), software privativo, puesto que dicho software no permite examinar el código fuente con el que se creó y, por lo tanto, impide modificar el mismo y adaptarlo a nuevas funcionalidades.
Por otro lado, él mismo define al software que sí permite este proceso, **software libre**. En cualquier caso ambos tipos de software no tienen por qué ser gratuitos, es decir la diferencia no es la gratuidad (aunque sí ocurre a menudo que el software de código abierto además suele ser gratuito) sino la libertad de utilizar el código fuente del software.
Una definición quizá menos tendenciosa es la que diferencia al software en: **software de código abierto** y **software de código cerrado u oculto**. Está diferencia de software se debe a dos formas diferentes de hacer negocio con él; los defensores del código cerrado argumentan que es lógico protegerle para evitar copiar su tecnología por parte de la competencia e incluso por razones de seguridad del mismo al no poder asegurar su correcto funcionamiento ante modificaciones de terceros.
Los defensores del segundo modelo están a favor de la versatilidad del código abierto que permite poder modificar el código por parte de miles de programadores en todo el mundo que pueden compartir dichas mejoras y así mejorar enormemente y de manera dinámica el producto.

4.2. SGBD Relacionales de código cerrado

Normalmente las licencias de uso de Sistemas Gestores de Bases de Datos con código cerrado usan licencias tipo **CLUF** o **EULA** (en inglés), acrónimo **contrato de licencia de usuario final**.
En estas licencias, el usuario firma unas condiciones de uso por el software, entre las que siempre figuran el hecho de no poder distribuir libremente el mismo y que está restringido a unas condiciones de trabajo concretas (por ejemplo el hecho de que normalmente sólo se pueda utilizar cada licencia en una sola máquina o por parte de un solo usuario). **Ejemplos** de SGBD de este tipo son:

- **Oracle**. Propiedad de Oracle Corporation. Es el SGBD más veterano y más influyente ya que la mayoría de mejoras al SQL original se desarrollaron para este SGBD. Sigue siendo uno de los SGBD comerciales más utilizados y además posee una gran relación con el lenguaje Java, acrecentada por la compra de la empresa creadora del mismo, **Sun Microsystems**.

  Presume de su gran estabilidad y escalabilidad, un control avanzado de transacciones y de sus lenguajes internos de manejo, especialmente famoso es su lenguaje procedimental PL/SQL. Es un SGBD multiplataforma, que tiene certificación para instalarse en Linux (aunque sólo con los compatibles con **Red Hat** y con condiciones muy concretas de instalación).
- **DB2**. Propiedad de IBM, es una de las bases de datos comerciales más populares. Desarrollada para Windows, UNIX y Linux. Implementa XML de manera nativa y dispone de amplias facilidades de migración de datos (especialmente desde Oracle) así como uso de transacciones avanzadas.

- **SQL Server (de Microsoft)**. Originalmente basado en el código del SGBD **SyBase** que Microsoft compró al propietario de **SyBase**, ahora es un SGBD distinto distribuido sólo para Windows y que compite con los dos anteriores. Dispone de una gran escalabilidad, estabilidad, uso de transacciones, entorno gráfico avanzado y de éxito entre los programadores de la plataforma .NET (también de Microsoft por su compatibilidad con esta).

Las tres son de las bases de datos más utilizadas en la actualidad por su contrastada potencia. Ninguna de las tres cumple completamente los estándares y aportan sus propios lenguajes y forma de trabajo. Además las tres disponen de versiones gratuitas para uso personal con base de datos más pequeñas.

4.3. SGBD relacionales de código abierto

Algunas utilizan la licencia **GNU GPL (*GNU General Public License*)**, que utilizan por ejemplo los sistemas Linux y que permite modificar el código, redistribuirlo; pero manteniendo la licencia.

Otra licencia muy utilizada de código abierto es la que utiliza el sistema operativo **BSD**, que incluso permite redistribuir el software cerrando el código.

Los SGBD más conocidos de código abierto son:

- **MySQL**. Inicialmente creada por la empresa MySQL AB, posteriormente comprada por Sun Microsystems que, a su vez, fue comprada por Oracle. Ha sido considerada como la principal SGBD de la comunidad de programadores de código abierto y de hecho en Internet sigue siendo la principal base de datos asociada a una aplicación web. Mantiene su licencia de tipo GPL, pero posee una segunda licencia cerrada para opciones de trabajo más avanzadas.

  Es muy popular por su histórica asociación con PHP, por su buena estabilidad, gran escalabilidad, e incluso uso de transacciones y lenguaje procedimental; además de ser un producto con infinidad de plataformas posibles para su instalación.

- **PostgreSQL**. Versión de código abierto basada en el producto **Ingres** de la **Universidad de Berkeley**. Usa licencia de tipo **MIT** (del **Instituto Tecnológico de Machassuchets**) que es una de las más libres, permite su modificación, redistribución incluso hacia otro tipo de licencias del tipo que sean, sin usar en ningún momento copyright.

  Está considerado como el SGBD de código abierto más potente y, sobre todo, más fidedigno con los estándares. Posee uso de transacciones avanzadas, lenguaje procedimental, gran estabilidad y escalabilidad.
  Hoy en día está considerada como la más potente de las bases de datos de código abierto y a partir de su núcleo se han creado otros productos libres de software base de datos.
- **Firebird**. Se trata de un SGBD liberado del producto comercial **Interbase** que era propiedad de **Borland** y que fue liberado con una variante de la licencia que usa Mozilla (**MPL**) que, a su vez, se basa en la licencia BSD. Tiene soporte transaccional avanzado, buena estabilidad pero no es muy escalable.
- **Apache Derby**. Anteriormente, IBM **Cloudscape**, usa licencia Apache que es poco restrictiva con las redistribuciones. Está programa en Java y pensada para ser utilizada en ese mismo lenguaje.

4.4. Bases de Datos NoSQL

4.4.1. Introducción

Las bases de datos relacionales han sido el modelo más popular desde finales de los años 70 por su solidez y gran facilidad para diseñar sistemas complejos. Sin embargo en estos últimos años empiezan a estar desbordadas ante el uso de bases de datos que tienen que dar servicio veloz y concurrente a miles de usuarios que son capaces de generar ingentes cantidades de información en poco tiempo.

Esta información en una base de datos relacional habría que validarla con las reglas e integridad que se imponen en esas bases de datos, indexarla y asegurar su uso en transacciones… y todo eso significa que un sistema con miles de entradas por minuto (como ocurre con las redes sociales) se bloquearía. Por ello se han diseñado bases de datos que se saltan el modelo relacional y en especial el lenguaje SQL y de ahí el nombre de sistemas **NoSQL**.

Aunque se utiliza para designar a las bases de datos documentales, gráficas y otros esquemas de bases de datos; actualmente se utiliza especialmente para designar a las bases de datos que requieren tantas transacciones por segundo, que el esquema relacional tradicional no daría abasto para ello.
La base teórica de este modelo se basa en el **teorema de CAP**, que indica que en un sistema distribuido no se pueden asegurar simultáneamente estas tres reglas:

- **Consistencia (C).** Que hace que la información sea la misma en todos los nodos que almacenan los datos.
- **Disponibilidad (A** de ***Availability)*** que hace que cada petición sobre los datos reciba la confirmación de si ha sido satisfactoria o no.
- **Tolerancia a fallos (P** de ***Partition tolerance*****)**, que permite que el sistema siga funcionando aun cuando ocurran errores o caídas en algunos nodos.

Un sistema relacional de base de datos podría asegurar la C y la P, pero no la disponibilidad en caso de una gran demanda de peticiones. Las bases de datos NoSQL varían las dos reglas a cumplir para dar más importancia a la disponibilidad (en general eliminan la primera la regla).

4.4.2. Diferencias con bases de datos SQL

Por esto utilizan un modelo diferente en el que los datos se almacenan de forma menos estricta, en especial no siguen estas reglas:

- **Transacciones-ACID**, como sí hacen los SGBD potentes relacionales (como **Oracle**, **DB2**, **SQLServer** o **PostgreSQL**). ACID hace referencia a las propiedades atomicidad, consistencia, aislamiento y durabilidad de las bases de datos. Y permite que las operaciones de manipulación sobre la base de datos sean revocables y así poder arreglar problemas de consistencia. Cumplir esas propiedades de forma estricta no permite que sitios como web puedan atender con garantías a todas sus peticiones por el tiempo necesario para actualizar los índices, por lo que necesitan otro esquema más rápido (aunque sea menos consistente).

- **No usar operaciones de tipo JOIN**. Los datos se almacenan sin validar su relación con otras tablas para hacerlo de forma más rápida.
- **No usan SQL como lenguaje de consulta**. En su lugar utilizan lenguajes de programación como **Java** o **C++** para acceder a los datos y otros como **XML** o **JSON** para definir los metadatos.
- **Datos no relacionales**. No se da importancia a las relaciones avanzadas con los datos. Por lo que no sirven para almacenar bases de datos complejas, sino más bien simples, con escasa relación y reglas. Por ejemplo los tweets de twitter, estadísticas a tiempo real, streaming de vídeo, conexiones a servidores remotos,… Es decir datos simples (de los que apenas se usan muy pocos valores clave) pero que llegan a gran velocidad.

La **clave** de este tipo de bases de datos es su capacidad para ser sistemas muy distribuidos y ultrarrápidos al almacenar datos: asegurando en todo momento su altísima disponibilidad y tolerancia a fallos.

Su **desventaja** es la ya comentada: al no poder establecer relaciones y reglas complejas sobre los datos, no sirven para almacenar bases de datos complejas o donde se desee realizar tareas como ordenar los datos en base a distintas claves o realizar búsquedas avanzadas en base a múltiples criterios. Triunfan donde ese tipo de tareas de gestión no son críticos (como las redes sociales, donde las cosas sólo se suelen ordenar en base a la fecha y hora).

5. Arquitectura de un SGBD

5.1. ¿Qué es la arquitectura?

La **arquitectura de un SGBD** hace referencia al modelo interno de funcionamiento del sistema. Es decir a las estructuras internas/físicas que proporciona para el almacenamiento y su relación con las estructuras lógicas/conceptuales. Como es lógico, cada SGBD propone diferentes arquitecturas.

5.2. Estructuras lógicas de la base de datos

En todas las bases de datos relacionales disponemos de estas estructuras lógicas para organizar la información:

- **Tablas**. Compuestas de filas y columnas en las que se almacenan los datos relevantes de cada base de datos. La mayoría de SGBD usan distintos tipos de tablas, pero en general cuando se habla de tablas se habla del elemento lógico encargado de almacenar los datos.
- **Restricciones**. Se definen al crear las tablas, pero se almacenan aparte. Están disponibles en el diccionario de datos y marcan las reglas que han de cumplir los datos para que se consideren válidos.
- **Índices**. Se trata de una lista ordenada de claves que permite acceder a los valores de una o más columnas de una tabla de forma veloz.
- **Vistas**. Son consultas almacenadas que nos permiten mostrar de forma personalizada los datos de una o varias tablas.
- **Procedimientos y funciones**. Código del lenguaje procedimental de la base de datos utilizado para ejecutar acciones sobre las tablas (incluidos los triggers).

5.3. Estructuras físicas e internas de la base de datos

Al final todos los elementos lógicos se deben almacenar en archivos cuyo tamaño, dirección,... etc. debe de ser controlado por el DBA. En los distintos tipos de SGBD hay variaciones sobre las estructuras lógicas, en el caso de las físicas su diferencia puede ser total, lo que obliga a conocer muy bien la parte interna del sistema concreto que estemos utilizando.

Las **estructuras internas** permiten analizar un nivel intermedio entre estructuras lógicas (como las tablas) y las físicas (como los archivos). Por ejemplo, Oracle proporciona espacios de tabla o tablespaces para aglutinar distintos elementos lógicos con distintos elementos físicos a fin de optimizar el rendimiento de la base de datos.

5.4. Instancias de bases de datos

Los usuarios que deseen conectarse a una base de datos, se conectan a lo que se conoce como la instancia de la base de datos (del inglés **instance**).
En el modo más sencillo de trabajo, el usuario dispone de un software en su máquina local, por lo que se encuentra en el lado del cliente, capaz de conectar con el SGBD. En ese momento se lanza un **proceso de usuario**. Ese proceso deberá comunicarse (a través de las redes apropiadas) con el **proceso de servidor**, un programa lanzado en el lado del servidor que está permanentemente en ejecución.
El proceso de servidor comunica a su vez con la **instancia de la base de datos**, otro proceso en ejecución a través del cual se accede a la base de datos.

Ilustración 1, Proceso de trabajo con la instancia de una base de datos