

Sistemas de bases de datos distribuidos

TEMA 42

ABACUS NT

Oposiciones 2021

Índice

- 1. Introducción**
- 2. Bases de datos distribuidas**
- 3. Conceptos en un BD Distribuida**
 - 3.1. Sección frontal y sección posterior.**
 - 3.2. El administrador de comunicaciones de datos**
 - 3.3. Procedimiento distribuido de consultas**
- 4. Diseño de BD Distribuidas**
 - 4.1. Aspectos a considerar**
 - 4.2. Características**
 - 4.2.1. Autonomía local**
 - 4.2.2. No dependencia de un nodo central**
 - 4.2.3. Operación continua**
 - 4.2.4. Transparencia de localización**
 - 4.2.5. Independencia respecto a la fragmentación**
 - 4.2.6. Independencia de réplica**
 - 4.2.7. Procesamiento distribuido de consultas**
 - 4.2.8. Manejo distribuido de transacciones**
 - 4.2.9. Independencia respecto al equipo**
 - 4.3. Construcción de una base de datos distribuida**
 - 4.4. Transparencia de la red y autonomía local**
 - 4.5. Ventajas de las bases de datos distribuidas**
 - 4.6. Desventajas**
- 5. Bases de datos NoSQL**
- 6. Conclusión**
 - 6.1. Relación del tema con el sistema educativo actual**
- 7. Bibliografía**

1. Introducción

Una Base de Datos (BBDD) es un conjunto almacenado de información para ser procesada. Los Sistemas Gestores de Bases de Datos son las aplicaciones de software que permiten almacenar y acceder a la información.

Los principales tipos de BBDD son:

–Relacionales: la información que almacena la BBDD está relacionada entre sí. Los datos relacionados (registros o filas) son almacenados en tablas que constan de varios campos (columnas).

–No relacionales: los datos no tienen por qué estar relacionados entre sí y por lo tanto no tienen que almacenarse en estructuras fijas como las tablas del modelo de base de datos relacional.

Tradicionalmente todas las bases de datos distribuidas han sido construidas bajo el paradigma relacional. Actualmente se empiezan a implementar grandes bases de datos distribuidas no relacionales, aunque hay conceptos de diseño que son aplicables a ambas.

2. Bases de datos distribuidas

Hay situaciones en las que el empleo de un único ordenador central no es una buena opción. Puede ser debido a situaciones geográficas, razones de fiabilidad, cuestiones funcionales, etc.

- Un sistema de base de datos distribuida está compuesto por un conjunto de **nodos** (también llamados **localidades**) separados físicamente y conectados entre si mediante un enlace de red tal que:
- Cada nodo es un sistema de base de datos en sí mismo.

Los nodos pueden intercambiar datos y trabajar conjuntamente si es necesario, con el fin de que un usuario de cualquier sitio pueda obtener acceso a los nodos de la red como si los datos estuvieran almacenados en su propio nodo.

Inconsecuencia la base de datos distribuida es En realidad Una base de datos virtual Huyas partes componentes se almacenan físicamente en varias bases de datos Convencionales. Esto es una base datos distribuida es la unidad lógica de todas esas bases de datos En una base datos distribuidas cada nodo es un sistema de base datos que tiene su propio DBMS, sus usuarios, sus programas de aplicaciones y su propio administrador local.

Un componente de software con el que debe contar cada nodo deberá encargarse de realizar las funciones de conexión y transferencia necesarias con los demás nodos de la red Este nuevo software con el DBMS ya existente es lo que se denomina Distributed Database Management System o **DDBMS**.

El DDBMS tiene además como función ser capaz de trabajar de forma **transparente** con datos

Las BDD pueden ser:

- **Homogéneas:** El SGBD utiliza un sólo lenguaje para describir los esquemas.
- **Heterogéneas:** El SGBD utiliza más de 1 lenguaje para describir los esquemas.

En un sistema de BDD la información se encuentra dispersa por varios lugares. Desde cada una de ellos se pueden ejecutar los datos de la propia localidad o los datos situados en varias localidades diferentes, dando lugar a dos **tipos de transacciones**:

- **Transacciones locales:** Procesan información situada en el mismo nodo.
- **Transacciones globales:** Procesan información situada en nodos diferentes. En este tipo los ordenadores deben estar interconectados entre sí.

Según la situación geográfica, la **conexión entre los nodos** se realiza mediante:

- **Redes de larga distancia.**
- **Redes de área local.**

3. Conceptos en un BD Distribuida

Antes de abordar el diseño de un BD, es necesario aclarar una serie de conceptos fundamentales.

3.1. Sección frontal y sección posterior.

Una base de datos distribuida posee una estructura sencilla compuesta por dos secciones:

La **sección posterior** es el DBMS en sí.

Las **secciones frontales** son las diversas aplicaciones ejecutadas dentro del DBMS, las cuales se pueden diferenciar en:

- Aplicaciones escritas por los propios usuarios.
- Aplicaciones suministradas por los proveedores, a menudo llamadas herramientas, tales como los procesadores de DDL, generadores de informes, subsistema de gráficas, hojas de cálculo, paquetes estadísticos, copias de seguridad, herramientas de desarrollo y herramientas CASE.

Esta división tan clara en secciones permite que sean ejecutadas en máquinas diferentes, es decir, de forma distribuida.

3.2. El administrador de comunicaciones de datos

En una base de datos distribuida, todas las solicitudes del usuario contra la BD y sus respectivas respuestas, se transmiten en forma de mensajes por la red.

Al estar los nodos de la red distribuidos por diversos lugares es imprescindible un **Gestor de Comunicaciones** para garantizar que los mensajes enviados a través de la red, llegan a su destino sin sufrir ningún tipo de interferencias o duplicidad de la información.

El encargado de controlar estas comunicaciones es el administrador DC (Data Communication) el cual no es un componente más del DBMS, sino un sistema independiente, por lo que al dúo de programas DBMS+DC se le denomina DB/DC.

Entre los componentes propiamente dichos del DB/DC se encuentran:

En los puntos de acceso

- Los **catálogos** con la información que se refiere a los esquemas.
- El **procesador de transacciones fuente**, comprueba que sólo accedan a la red los usuarios autorizados y obtengan la información requerida, así como en qué puntos se encuentran los datos pedidos. A cada transacción la descompone en subtransacciones para reducir al mínimo el uso de las líneas de comunicación.

En los puntos de almacenamiento:

- El respectivo DBMS.
- El **gestor de transacciones** recibe las transacciones, y termina de compilar si es preciso, y las ejecuta. Cuando finaliza su ejecución, lo comunica al gestor de transacciones de la localidad que lo requirió y manda los resultados a los nodos reseñados en el plan de ejecución.

3.3. Procedimiento distribuido de consultas

El procesamiento de consultas es de suma importancia en bases de datos centralizadas. Sin embargo, en BDD éste adquiere una relevancia mayor. El objetivo es convertir transacciones de usuario en instrucciones para manipulación de datos. No obstante, el orden en que se realizan las transacciones afecta grandemente la velocidad de respuesta del sistema. Así, el procesamiento de consultas presenta un problema de optimización en el cual se determina el orden en el cual se hace la menor cantidad de operaciones. En BDD se tiene que considerar el procesamiento local de una consulta junto con el costo de transmisión de información al lugar en donde se solicitó la consulta.

Recuperación

En los entornos distribuidos de datos podemos encontrar lo siguiente:

Fallo de los nodos.

Cuando un nodo falla, el sistema deberá continuar trabajando con los nodos que aún funcionan.

Si el nodo a recuperar es una base de datos local, se deberán separar los datos entre los nodos restantes antes de volver a unir de nuevo el sistema.

Copias múltiples de fragmentos de datos.

El subsistema encargado del control de concurrencia es el responsable de mantener la consistencia en todas las copias que se realicen y el subsistema que realiza la recuperación es el responsable de hacer copias consistentes de los datos de los nodos que han fallado y que después se recuperarán.

Transacción distribuida correcta.

Se pueden producir fallos durante la ejecución de una transacción correcta si se plantea el caso de que al acceder a alguno de los nodos que intervienen en la transacción, dicho nodo falla.

Fallo de las conexiones de comunicaciones.

El sistema debe ser capaz de tratar los posibles fallos que se produzcan en las comunicaciones entre nodos. El caso más extremo es el que se produce cuando se divide la red. Esto puede producir la separación de dos o más particiones donde las particiones de cada nodo pueden comunicarse entre sí pero no con particiones de otros nodos.

Para implementar las soluciones a estos problemas, supondremos que los datos se encuentran almacenados en un único nodo sin repetición. De esta manera sólo existirá un único catálogo y un único DM (Data Manager) encargados del control y acceso a las distintas partes de los datos.

Para mantener la consistencia de los datos en el entorno distribuido contaremos con los siguientes elementos:

- Catálogo: Programa o conjunto de programas encargados de controlar la ejecución concurrente de las transacciones.
- CM (Cache Manager). Subsistema que se encarga de mover los datos entre las memorias volátiles y no volátiles, en respuesta a las peticiones de los niveles más altos del sistema de bases de datos. Sus operaciones son Fetch(x) y Flush(x).
- RM (Recovery Manager). Subsistema que asegura que la base de datos contenga los efectos de la ejecución de transacciones correctas y ninguno de incorrectas. Sus operaciones son Start, Commit, Abort, Read, Write, que utilizan a su vez los servicios del CM.
- DM (Data Manager). Unifica las llamadas a los servicios del CM y el RM.
- TM (Transaction Manager). Subsistema encargado de determinar que nodo deberá realizar cada operación a lo largo de una transacción.

Las operaciones de transacción que soporta una base de datos son: Start, Commit y Abort. Para comenzar una nueva transacción se utiliza la operación Start. Si aparece una operación commit, el sistema de gestión da por terminada la transacción con normalidad y sus efectos permanecen en la base de datos. Si, por el contrario, aparece una operación abort, el sistema de gestión asume que la transacción no termina de forma normal y todas las modificaciones realizadas en la base de datos por la transacción deben de ser deshechas.

4. Diseño de BD Distribuidas

4.1. Aspectos a considerar

Aparte de las consideraciones propias de cualquier Base de Datos, cuando se trata de diseñar una BDD es necesario tener en cuenta otros aspectos como son:

- La **repetición**: Consiste en mantener varias copias de ficheros en nodos diferentes, como se ha visto en el punto anterior.

- La **fragmentación**: De la información contenida en las relaciones. Cada fragmento se almacena en nodos diferentes. Un fragmento contiene información suficiente para poder restituir el fichero.

Las diferentes *formas en que se puede fragmentar una relación* son:

- a) **Horizontal**: Cada fragmento está formado por una serie de tuplas (filas).
- b) **Vertical**: Cada fragmento está formado por una serie de columnas de la tabla global. La fragmentación vertical se lleva a cabo por medio de un campo especial *denominado identificación de tupla* (un puntero (ID) con la dirección física o lógica del resto de las columnas en los diferentes fragmentos).
- c) **Mixta**: es un compendio de las dos anteriores.

La repetición y la fragmentación comprenden a las anteriores.

4.2. Características

Desde el punto de vista del usuario, un sistema distribuido deberá ser idéntico a un sistema no distribuido. Además, existen una serie de reglas que todo sistema distribuido debe cumplir:

- Autonomía local
- No dependencia de un nodo central
- Operación continua
- Independencia de localización
- Independencia respecto a la fragmentación
- Independencia de réplica
- Procesamiento distribuido de consultas
- Manejo distribuido de transacciones
- Independencia respecto al equipo
- Independencia respecto al Sistema Operativo
- Independencia respecto al tipo de red
- Independencia respecto al DBMS.

4.2.1. Autonomía local

Los nodos de un sistema distribuido deben ser autónomos. Esto implica que ningún nodo X debe depender de ningún otro nodo Y para su buen funcionamiento. Si un nodo cae, el sistema debe seguir funcionando. Si un nodo queda aislado temporalmente, este nodo debe también seguir funcionando correctamente.

Además, la autonomía local implica la existencia de un administrador y una administración local en la BD.

Esta característica **es imposible de cumplir** por completo. Así el objetivo realmente debe ser **garantizar siempre la máxima autonomía local** posible.

4.2.2. No dependencia de un nodo central

La autonomía local implica también la no dependencia de un nodo central. En este sentido hay que añadir que en un sistema distribuido puede haber hardware más potente en unos nodos que otros, sin embargo, la arquitectura de la BD debe ser la misma en todos los nodos. No debe haber ningún nodo “maestro”.

Esto evita que el sistema sea vulnerable a la caída de ese supuesto nodo central y cuellos de botella innecesarios.

4.2.3. Operación continua

Apagar un servidor no es una tarea habitual. Idealmente nunca debería haber necesidad de apagar a propósito el sistema, es decir, no debe existir ninguna función en absoluto que necesite un apagado o reinicio del sistema en ningún momento.

4.2.4. Transparencia de localización

La idea básica es que no debe ser necesario que los usuarios sepan dónde están almacenados físicamente los datos, sino que más bien deben poder comportarse -al menos desde un punto de vista lógico- como si todos los datos estuvieran almacenados en su propio nodo local.

4.2.5. Independencia respecto a la fragmentación

Un sistema distribuido maneja fragmentación de datos si es posible dividir una relación en partes o fragmentos para propósitos de almacenamiento físico.

La fragmentación es deseable por razones de productividad: los datos pueden almacenarse en la localidad donde se utilizan con mayor frecuencia, de forma que la mayor parte de las operaciones sean locales y se reduzca el tráfico en la red.

Existen al menos dos clases de fragmentación: horizontal y vertical correspondientes a las operaciones de selección y proyección, respectivamente.

La reconstrucción de la relación original a partir de los fragmentos, se hace mediante operaciones de reunión (fragmentos verticales) y unión (fragmentos horizontales).

Un sistema que ofrece fragmentación de los datos debe ofrecer también transparencia respecto a la fragmentación, es decir, los usuarios tendrán una vista como si los datos no estuvieran fragmentados en realidad.

4.2.6. Independencia de réplica

Un sistema maneja réplica si se puede representar en el nivel físico mediante varias copias almacenadas en distintos nodos.

La réplica tiene varias ventajas:

Las aplicaciones pueden trabajar con copias locales en lugar de tener que comunicarse continuamente con nodos remotos.

Aumenta la disponibilidad de los datos, incluso con propósitos de recuperación.

La desventaja principal de las réplicas es la propagación de actualizaciones.

4.2.7. Procesamiento distribuido de consultas

La optimización es todavía más importante en un sistema distribuido que en uno centralizado. Lo esencial es que en una consulta que implique solicitudes a nodos remotos habrá múltiples formas de trasladar los datos en la red para satisfacer la solicitud y es crucial una estrategia eficiente.

4.2.8. Manejo distribuido de transacciones

El manejo de transacciones tiene dos aspectos principales, el control de la recuperación y el control de concurrencia.

En un sistema distribuido, una sola transacción puede implicar la ejecución de código en varios nodos. Se dice que esta transacción está compuesta por varios agentes, donde un **agente** es el código ejecutado en nombre de una transacción dada en un determinado nodo.

El sistema deberá garantizar que todos los agentes correspondientes a esa transacción se comprometan al unísono o bien que retrocedan al unísono.

4.2.9. Independencia respecto al equipo

La base de datos debe ser independiente del hardware y del software, y **presentar al usuario una sola imagen del sistema**. Esto implica:

- Independencia del hardware en que se ejecute
- Independencia del sistema operativo concreto
- Independencia respecto a la red
- Independencia respecto al DBMS

Realmente esto es imposible de cumplir, por lo que se buscará siempre un alto grado de independencia.

4.3. Construcción de una base de datos distribuida

El punto de partida de las BDD es el **enfoque relacional**. La construcción se refiere a la distribución de ficheros relacionales por los distintos nodos de la red.

1. **Base de Datos Multiplicada y Distribuida.** Todos los ficheros están duplicados en cada nodo con el fin de reducir los costes de comunicación y aumentar el desempeño, ya que cada operación de recuperación se puede hacer en cada nodo. Presenta el inconveniente de la cantidad de espacio necesario para su almacenamiento, redundancia externa y los

problemas en la actualización de los datos redundantes en cada nodo, por lo que este tipo de distribución se utiliza raramente.

2. **Base de Datos Distribuida Particionada.** Es la forma más simple de distribuir los datos en una red. Ficheros específicos se distribuyen en cada nodo sin duplicidad de datos. Presenta la ventaja de tener que actualizar sólo un fichero cada vez que se produce una transacción. Pero cuando se desea hacer una consulta que implica la necesidad de acceder a datos situados en otras localidades diferentes lleva consigo un aumento del coste de tiempo para su realización. Un fallo en cualquier nodo puede dejar a toda la red sin servicio.
3. **Bases de Datos Distribuidas en donde los ficheros se ubican en un nodo específico.** Y, además, algunos ficheros se almacenan en otros nodos. La elección del nodo para acoger a todos los ficheros, así como de los ficheros que se guardan en cada nodo, debe realizarse con el fin de reducir al mínimo los costes de comunicación en las transacciones globales, es decir, escoger la localidad de referencia. Se han reducido los costes de comunicación, pero aún quedan por resolver los problemas de actualización. Si la localidad que acoge a todos los ficheros sufre un fallo, toda la red puede quedar inutilizada.
4. **Bases de Datos Distribuidas con ficheros duplicados seleccionados.** Es la más flexible de todas las construcciones. Considera que un fichero que se utiliza prácticamente en una localidad se almacena sólo en ella y que un fichero que se consulta en varias localidades está ubicado en todas ellas. Este sistema, a pesar de tener ventajas, también presenta inconvenientes: la actualización y hallar datos que se encuentran en varias localidades diferentes.

4.4. Transparencia de la red y autonomía local

Se denomina así, a la forma en que el sistema oculta los detalles de la distribución de los fragmentos por los distintos nodos de la red; en definitiva, el grado en que los usuarios pueden ignorar cómo está diseñada la BDD.

La transparencia está íntimamente ligada a la *autonomía local*, es el grado en que un diseñador de un nodo puede ser independiente del resto del sistema.

Estos dos conceptos se deben tener en cuenta en:

- Los **nombres de datos**, que deben ser únicos en la red global. Para que ningún diseñador de distintas localidades utilice el mismo nombre para designar a datos diferentes, a los nombres de datos se les incorpora habitualmente información acerca de la localidad que los genera y sobre los fragmentos que la componen.
- La **actualización**, que debe realizarse en todas las copias y en todos los fragmentos afectados.

El acceso a datos distribuidos es útil y pasará a ser crítico cuando se confirme la tendencia hacia la computación distribuida. Los principales vendedores de SGBD están comprometidos en la entrega de gestión de BDD, y algunos de ellos ya ofrecen capacidades limitadas de BDD. Los expertos en la

industria y otros han escrito extensamente acerca de las características que debería proporcionar un SGBD distribuido, y existe un acuerdo general sobre estas *características* "ideales":

- **Transparencia de ubicación.** El usuario no debería tener que preocuparse acerca de dónde están localizados físicamente los datos. El SGBD debería presentar todos los datos como si fueran locales y debería ser responsable de mantener esa ilusión.
- **Sistemas heterogéneos.** El SGBD debería soportar datos almacenados en sistemas diferentes, con diferentes arquitecturas y niveles de rendimiento, incluyendo PCs, estaciones de trabajo, servidores de LAN, miniordenadores y macroordenadores.
- **Transparencia de red.** Excepto por las diferencias en rendimiento, el SGBD debería trabajar de la misma manera sobre diferentes redes, desde las LANs de alta velocidad a los enlaces telefónicos de baja velocidad.
- **Consultas distribuidas.** El usuario debería ser capaz de componer datos procedentes de cualquiera de las tablas de la BDD, incluso si las tablas están localizadas en sistemas físicos diferentes.
- **Actualizaciones distribuidas.** El usuario debería ser capaz de actualizar datos en cualquier tabla para que la que tenga privilegios necesarios tanto si la tabla está en un sistema local como si está en un sistema remoto.
- **Transacciones distribuidas.** El SGBD debe soportar transacciones (Utilizando COMMIT y ROLLBACK) a través de fronteras de sistema, manteniendo la integridad de la BDD incluso en presencia de fallos de red y fallos de sistemas individuales.
- **Seguridad.** El SGBD debe proporcionar un esquema de seguridad adecuado para proteger la BDD completa frente a formas no autorizadas de acceso.
- **Acceso universal.** El SGBD debería proporcionar acceso uniforme y universal a todos los datos de la organización.

Ningún producto de SGBD distribuido actual ni siquiera se acerca a satisfacer este ideal. De hecho, existen formidables obstáculos que hacen difícil proporcionar incluso las formas más sencillas de gestión de una BDD. Estos obstáculos incluyen:

- Rendimiento.
- SQL estático.
- Optimización.
- Compatibilidad de Datos.
- Catálogos de sistema.
- Entorno de vendedores mixtos.
- Interbloqueos distribuidos.
- Recuperación.

A causa de estos obstáculos, los principales vendedores de SGBD han adoptado un planteamiento de paso a paso para la gestión de datos distribuida. Al principio el SGBD puede permitir que un usuario en un sistema consulte una Base de Datos localizada en algún otro sistema. En versiones posteriores las capacidades de SGBD distribuido pueden crecer, acercándose cada vez más al objetivo de proporcionar acceso de datos transparente y universal.

4.5. Ventajas de las bases de datos distribuidas

Entre las **ventajas** de los sistemas que soportan BDD están:

1. Reducir el coste de comunicación de datos, ya que la mayoría de las consultas se pueden realizar sin necesidad de acceder a los archivos situados en localidades diferentes.
2. Autonomía local.
3. Confiabilidad. Cuando el sistema de una determinada localidad sufre un fallo y se “cae”, toda la red no tiene por qué dejar de funcionar, debido a que cada localidad cuenta con su propio ordenador y fragmentos de datos.

Una vez solucionado el fallo, la localidad afectada deberá reintegrarse a la red con el mínimo de complicaciones.

4.6. Desventajas

Las BDD por su mayor complejidad presentan una serie de desventajas entre las que se encuentran:

1. Coste del software por la propia estructuración de los datos.
2. Actualización de datos duplicados.
3. Mayor posibilidad de errores como consecuencia de una mayor concurrencia desde distintas localidades a un mismo dato, lo que requiere complicados protocolos de control.
4. Transparencia de ubicación. Gran parte de los sistemas distribuidos no logran la transparencia de ubicación y deben ser los propios usuarios los que transfieran los archivos para el procesamiento local.
5. Mayor tiempo extra en el procesamiento, sobre todo cuando se trata de transacciones globales.

5. Bases de datos NoSQL

El progresivo aumento de popularidad de las bases de datos NoSQL (“Not Only SQL”) pertenecen en los últimos años, ha hecho que se replanteen los sistemas de BD Distribuidos relacionales.

El Big Data, y el escalado vertical (scale-up) que ofrecen las bases de datos relacionales, en principio, no parece una buena solución. El problema de estos sistemas es que están diseñados para trabajar con grandes volúmenes de datos en disco, mientras que el escalado horizontal (scale-out) de los entornos distribuidos de las principales bases de datos NoSQL mantienen los datos en memoria principal, incrementando notablemente la disponibilidad de la información.

Las principales características y ventajas de este tipo son:

- SQL no es el lenguaje de consulta/modificación de datos principal, aunque sí lo soportan, de ahí el nombre No Sólo SQL.
- Los datos no tienen que almacenarse en tablas.
- Generalmente, su arquitectura es distribuida.

- Son más eficientes en el procesamiento de los datos que las BBDD relacionales, por eso son la elección para aplicaciones que hacen un uso intensivo de estos (“streaming”, etc.).
- Utilizan lo que se conoce como consistencia eventual que consiste en que los cambios realizados en los datos serán replicados a todos los nodos del sistema, lo cual aumenta el rendimiento de estos sistemas en contraposición a las propiedades ACID de las BBDD relacionales (“Atomicity, Consistency, Isolation and Durability” - Atomicidad, Consistencia/Integridad, Aislamiento y Durabilidad).
- Los Sistemas de Gestión de Bases de Datos NoSQL no contemplan por definición la atomicidad de las instrucciones, es decir, cuando una operación sobre los datos consta de varios pasos, no se tienen que ejecutar todos, cosa que sí sucede en los modelos relacionales (transacciones completas). Hay algunas BBDD NoSQL que contemplan la atomicidad.
- Los gestores NoSQL no contemplan obligatoriamente la consistencia o integridad de la BBDD, esto quiere decir que no se comprueba que la operación a ejecutar sobre los datos se pueda completar desde un estado de la Base de Datos válido a otro válido (por ejemplo, no violación de ninguna restricción de tipos de datos o reglas).
- Al utilizar el mecanismo de consistencia eventual se puede dar el caso de que la misma consulta a diferentes máquinas del sistema produzca resultados diferentes porque las modificaciones de la BBDD aún no han sido replicadas a todos los nodos. Algunas BBDD de este tipo contemplan la propiedad de consistencia.
- Estas BBDD utilizan sus propios lenguajes de consulta de datos y APIs, por lo que no tienen una gran interoperabilidad (por ejemplo, dificultad de migraciones de una BBDD a otra, integración con aplicaciones, consultas heredadas en SQL, etc.).
- No hay estandarización para este tipo de BBDD, algo que sí es un punto fuerte de las relacionales.
- Las Bases de Datos NoSQL funcionan ampliamente en máquinas Linux, pero no existe en general soporte a otros Sistemas Operativos.
- Las interfaces de gestión de estas BBDD no son intuitivas ni sencillas y en algunos casos carecen de ellas gestionándose directamente desde consola de comandos.

¿Cuándo es recomendable utilizar una BBDD NoSQL?

1-Cuando se necesita una BBDD para una aplicación que hace una consulta/lectura intensiva de grandes cantidades de datos.

2-Cuando no hay la necesidad de que los datos sean consistentes.

3-Si los datos a almacenar no tienen una estructura fija.

Una misma aplicación puede usar una BBDD relacional y una BBDD NoSQL y guardar cosas diferentes en cada una de ellas.

6. Conclusión

Desde que en los años 70 del siglo pasado se popularizaran las bases de datos como soluciones comunes a la manipulación de grandes volúmenes de datos por parte de aplicaciones diversas, se han desarrollado varias arquitecturas, siendo actualmente dos, las relacionales (SQL) y las NoSQL (No relacionales) las que son principalmente utilizadas hoy en día.

Con el desarrollo de Internet se abre un nuevo camino a las bases de datos distribuidas, mediante técnicas como el streaming y la manipulación del Big Data.

Aunque actualmente (año 2020) miles de usuarios implementan soluciones relacionales como **Oracle, Microsoft SQL Server, MariaDB o MySQL**, los servicios en la nube distribuidos NoSQL, comienzan a ser tendencia en las grandes bases de datos, y así empresas de la talla de **Facebook, Twitter, Instagram, Spotify o Netflix** utilizan el SGDB **Cassandra**. Otros sistemas NoSQL populares son **Redis** y **MongoDB**.

6.1. Relación del tema con el sistema educativo actual

Este tema es aplicado en el aula en los módulos profesionales siguientes, con las atribuciones docentes indicadas (PES/SAI):

Formación profesional básica

- Operaciones auxiliares para la configuración y la explotación(TPB en Informática de Oficina/ TPB en informática y Comunicaciones) (PES/SAI)
- Ofimática y archivo de documentos (TPB en Informática de Oficina) (PES/SAI)

Grado Medio

- Aplicaciones ofimáticas (GM de SMR) (PES/SAI)

Grado Superior

- Gestión de bases de datos (ASIR) (PES)
- Bases de Datos (DAW/DAM) (PES)

Bachillerato:

- 4º ESO – Tecnología de la Información y la comunicación (PES)
- Bachillerato – Tecnologías de la Información y la Comunicación (PES)

7. Bibliografía

- C.J. Date: **Introducción a los sistemas de bases de datos** Pearson, 2001.
- Elmasri, R.A. y Navathe S.B: "**Fundamentos de Sistemas de Bases de Datos**". Addison-Wesley, 3^a Edic, 2002.
- Olga Pons, J M Medina, M.A. Vila. **Introducción a los Sistemas de Bases de Datos** Edt Paraninfo (2005)
- Korth, H.F. y Silberschatz: "**Fundamentos de Bases de Datos**". McGraw -Hill, 4^a Edic., 2002.
- Garcia-Molina, H.; Ullman, J.D.; Widom, J. **Database systems: the complete book** - Pearson Education Limited, 2013.
- Abraham Silberschatz, Henry F. Korth, y S. Sudarshan, **Fundamentos de bases de datos** Edt. Mc Graw-Hill (2014)
- <https://elbauldelprogramador.com/> (2020)
- www.Unir.net (2020)

