Bases de datos no relacionales

PID_00251488

Jesús Alonso-Zárate
Jordi Casas Roma

Tiempo mínimo de dedicación recomendado: 2 horas

Universitat Oberta de Catalunya

Índice

Introducción

Como hemos visto en materiales anteriores, el almacenamiento y procesamiento de datos juega un papel fundamental en la industria 4.0.

En este material, vamos a conocer las bases de datos no relacionales; dadas las limitaciones de las bases de datos relacionales, entenderemos qué puntos fuertes ofrecen las bases de datos no relacionales.

Describiremos brevemente los diferentes tipos de bases de datos no relacionales existentes, con el objetivo de entender sus fortalezas y limitaciones.

Objetivos

Los objetivos de este material son:

1) Conocer la existencia de las bases de datos no relacionales.

2) Conocer las fortalezas de las bases de datos no relacionales.

3) Conocer las debilidades de las bases de datos no relacionales.

4) Conocer las principales tecnologías de bases de datos no relacionales.

5) Adquirir un pensamiento crítico para poder decidir, en cada caso, qué solución de base de datos puede ser la más adecuada para una necesidad concreta.

1. Introducción a las bases de datos no relacionales

Un elemento fundamental en la industria 4.0 será la gran cantidad de datos que van a estar disponibles para la toma de decisiones, ya sea de manera manual o automatizada. Como hemos visto anteriormente, la industria 4.0 se caracteriza, precisamente, por la automatización de una gran mayoría de los procesos.

1.1. Los antecedentes: las bases de datos relacionales

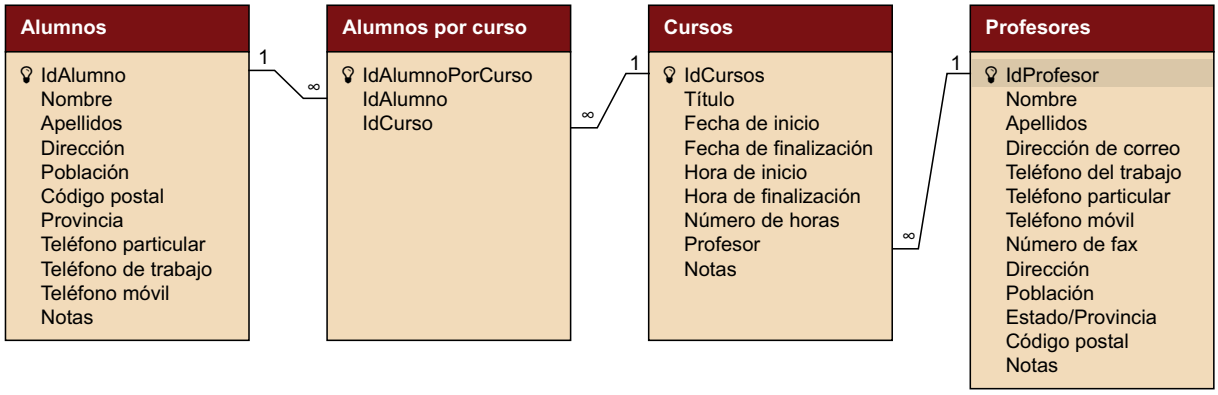
Los datos, además de capturarse para poderse procesar y permitir la toma de decisiones, deben ser **almacenados**. Los datos se almacenan en **bases de datos**.

Hasta hace relativamente poco tiempo, la mayoría de las **bases de datos** que se usaban para almacenar datos eran bases de datos de tipo **relacional**. Suelen referirse también por el acrónimo RDBMS (del inglés *relational data base management system*).

Las bases de datos **relacionales** usan tablas para almacenar los datos.

Las tablas de una base de datos relacional contienen registros, y cada registro tiene una serie de campos asociados. Por lo tanto, cada tabla suele tener definidas una serie de **columnas**, que son los **campos**, y un número variable de **filas**, que son los **registros**. En estos casos, la información entre tablas se puede «conectar» a través de **relaciones**, que se establecen en un ámbito de campo. Un claro ejemplo de base relacional la encontramos en *Microsoft Access*, que permitía crear estructuras de bases de datos como la que se muestra en la figura siguiente.

Figura 1. Ejemplo de base de datos relacional



En esta figura, vemos la arquitectura de una base de datos con 4 tablas de información:

1) Tabla «Alumnos».
2) Tabla «Alumnos por curso».
3) Tabla «Cursos».
4) Tabla «Profesores».

Cada una de estas tablas tiene definidos una serie de campos, que definen la información que tendrá cada registro. Estos campos pueden ser, por ejemplo, numéricos, alfanuméricos, booleanos (cierto/falso), fechas, divisas, campos autonuméricos (se incrementan en una unidad automáticamente para cada registro), etc.

A modo de ejemplo, una fila de la tabla «Alumnos» podría ser la formada por la siguiente información:

1) IdAlumno: 345
2) Nombre: Juan
3) Apellidos: García Gómez
4) Dirección: Calle Mayor, 45, 2-3
5) Población: Barcelona
6) Código postal: 08345
7) Provincia: Barcelona
8) Teléfono particular: 936342211
9) Teléfono trabajo: 936450000
10) Teléfono móvil: 660000000
11) Notas: alumno de Ingeniería de Telecomunicaciones.

Según se ha definido esta base de datos:

- Cada registro en la tabla «Alumnos por curso» estará asociado con el «IdAlumno» de una entrada en la tabla de «Alumnos».

- Cada registro de la tabla «Alumnos por curso» tendrá asociado un «IdCurso» de la tabla «Cursos».

- Cada entrada en la tabla «Cursos» tendrá asociado un «IdProfesor» de una entrada en la tabla de «Profesores».

A través de la definición de este tipo de «relaciones» entre los elementos de las tablas, se puede organizar la información de la base de datos para, posteriormente, poder consultarla y tratarla de una manera ordenada y automatizada.

El acceso a los datos de estas bases de datos suele hacerse con el lenguaje SQL, del inglés *structured query language*. Se trata de un lenguaje relativamente sencillo que

permite consultar, añadir, eliminar y editar los datos almacenados en bases de datos relacionales.

Por ejemplo, la siguiente consulta, efectuada sobre la base de datos del ejemplo anterior:

```
SELECT * FROM "ALUMNOS"
```

devolvería un listado con todos los registros de la tabla «Alumnos».

Las bases de datos relacionales, RDBMS, se han mostrado eficientes para muchas aplicaciones en las que **los datos están estructurados y pueden ser indexados**, y ofrecen buen rendimiento cuando:

1) Las lecturas o ediciones de datos son muy frecuentes, pero siempre siguiendo un mismo patrón de acción, es decir, cuando las consultas o ediciones se repiten muchas veces.

2) Para gestionar un gran número de consultas, pero con pocas peticiones de escritura de datos en las bases de datos.

En el siguiente apartado, describimos las principales limitaciones de las bases de datos relacionales.

1.2. Limitaciones de las bases de datos relacionales

Aunque durante muchos años se han usado esencialmente las bases de datos relacionales, en los últimos años se está imponiendo el uso de bases de datos no relacionales, que veremos más adelante. Sin embargo, como justificaremos con más detalle más adelante, las bases de datos no relacionales tampoco son la solución para todo; cuando la naturaleza de los datos es estructurada y puede ser indexada, las bases de datos relacionales son la mejor alternativa.

Algunas de las principales limitaciones de las bases de datos relacionales son:

1) **Coste de diseño y mantenimiento**: la definición de la estructura de la base de datos requiere un complejo proceso de diseño que, por lo general, implica la contratación de expertos en software y gestión de datos capaz de llevar a cabo el diseño de la misma. Una vez diseñada, introducir cambios no resulta trivial, ya que se deben tener en cuenta todas las relaciones establecidas entre tablas para asegurar la coherencia del diseño. De este modo, tanto el diseño como su mantenimiento evolutivo pueden resultar ciertamente costosos en tiempo y en dinero.

2) **Poca flexibilidad y compatibilidad**: las bases de datos relacionales son muy adecuadas cuando los datos que hay que tratar son homogéneos en su estructura y forma-

to. Sin embargo, cuando se trata de juntar datos de diferentes orígenes, con posibles diferentes estructuras y formatos, este tipo de bases de datos no ofrecen flexibilidad ni capacidad de compatibilizar la existencia de distintos tipos de datos en una misma base de datos.

3) **Poca escalabilidad para grandes volúmenes de datos**: por la naturaleza muy estructurada de las bases de datos relacionales, su rendimiento suele verse afectado a medida que la cantidad de datos que hay que tratar aumenta; de este modo, su aplicación para entornos de *big data* puede resultar un cuello de botella a la hora de agilizar procesos de toma de decisiones en tiempo real, por ejemplo.

4) **Limitación en la combinación de datos**: como hemos dicho antes, las bases de datos relacionales funcionan bien cuando los datos tienen características comunes; si este no es el caso, las bases de datos relacionales no se pueden usar. Un simple caso puede ser el hecho de tener un campo de información entre dos bases de datos con diferentes longitudes de campo; por ejemplo, una contraseña de 8 o 10 caracteres que no puede compararse entre dos sistemas de acceso. Además, en muchos casos, las relaciones son «obligatorias», por lo que en el caso de que un campo clave (para definir una relación) no exista para un origen de datos concreto, la base de datos podría no aceptar la inclusión de determinados datos.

5) **Falta de interoperabilidad**: el hecho de que la estructura de una base de datos relacional sea estática y muy rígida limita la posibilidad de compartir datos entre bases de datos. Esto puede darse incluso entre bases de datos internas de una misma empresa, que se han diseñado de manera independiente, e implica el diseño de interfaces de conexión que, de nuevo, deben ser hechas a medida y ofrecen muy poca flexibilidad.

6) **Limitaciones en tratamiento de información multimedia**: las bases de datos relacionales han mostrado poca eficiencia en determinadas aplicaciones que combinan datos en formato texto con información multimedia, tales como audio, imágenes o vídeo, y sujetas a una gran variedad de formatos y especificaciones.

1.3. ¿Por qué nacen las bases de datos no relacionales?

Podríamos decir que hay dos grandes motivaciones para la emergencia de las bases de datos no relacionales:

1) Superar las limitaciones de las bases de datos relacionales.
2) Ofrecer una solución más adaptada a la emergencia de la computación distribuida.

En los siguientes apartados, elaboramos un poco estos dos argumentos.

1.3.1. Superar las limitaciones de las bases de datos relacionales

En parte debido a los motivos descritos en el apartado anterior, en el que hemos repasado los principales inconvenientes de las bases de datos relacionales, han emergido con fuerza en los últimos años las bases de datos **no relacionales**.

Como el nombre indica, las bases de datos no relacionales no se basan en la definición de relaciones entre tablas de datos. Estas bases de datos suelen conocerse como bases de datos NoSQL (***not only SQL***), y no aplican el modelo relacional ni solo el lenguaje **SQL** para el acceso y tratamiento de los datos.

Las bases de datos NoSQL no requieren esquemas o arquitecturas de información fijas. Se dice que estas bases de datos escalan «horizontalmente», ya que cada registro de datos, cada fila de una base relacional, puede crecer indefinidamente en cantidad de datos.

Las bases de datos NoSQL pueden manejar un gran número de datos y gestionar volúmenes de lectura y escritura muy superiores a las bases de datos relacionales, ofreciendo, además, mucha más flexibilidad en el tipo de datos que se pueden almacenar, así como el tipo de consultas y escrituras que se pueden hacer en los datos.

Sin embargo, no todo son puntos fuertes para las bases de datos no relacionales; justamente, su falta de estructura hace de ellas que no puedan garantizar consistencia de los datos, así como el cumplimiento de las normas ACID, que sí cumplen las bases de datos relacionales. Estas normas ACID son:

1) **A**tomicidad.
2) **C**onsistencia; todos los usuarios acceden a los mismos datos al mismo tiempo.
3) **I**solation ('aislamiento').
4) **D**urabilidad.

Debido a esto, es posible usar técnicas que permiten trabajar con bases de datos no relacionales y, aun así, asegurar que las normas ACID se cumplen. Este es el ejemplo de los sistemas AppScale o CloudTPS, que no vamos a entrar a detallar en este material. Estas herramientas se conocen como el *middleware* de la base de datos.

Muchos sistemas NoSQL emplean una estructura de hardware distribuida; esto quiere decir que los datos se almacenan de manera redundante en varios servidores, posiblemente almacenados en la nube. Esto permite dos cosas:

1) Escalar muy bien cuando sean necesarios más recursos.

2) Tolerar fallos de un servidor, de modo que se evita tener toda la información en un único punto de fallo posible.

1.3.2. Adaptarse mejor a la emergencia de la computación distribuida

La **computación distribuida** (del inglés *cloud computing*) es un paradigma de computación emergente en el que los datos y servicios residen en centros de datos muy escalables que pueden ser accedidos desde cualquier dispositivo conectado a internet.

Por lo tanto, *cloud computing* hace referencia a la capacidad computacional y de almacenamiento virtualizada, expuesta mediante infraestructura agnóstica a la plataforma y accedida por internet. Esto quiere decir que los recursos de IT (tecnologías de información) se pueden compartir según las necesidades cambiantes, creando y destruyendo instancias de «ordenadores» a partir de los cambios en la demanda.

Las tecnologías de computación distribuida han madurado mucho en los últimos años; no hay más que ver aplicaciones como Google Drive o Dropbox, de gran aceptación en el ámbito personal, así como aplicaciones de *cloud* de uso profesional.

Este concepto de computación distribuida requiere un tipo de almacenamiento de datos, de bases de datos, por lo tanto, que se puedan flexibilizar y adaptar de manera más flexible al concepto de *cloud computing*. Las bases de datos no relacionales, como veremos ahora, ofrecen esta flexibilidad y capacidad de adaptación.

1.4. Puntos fuertes de las bases de datos no relacionales

Por lo tanto, podemos resumir los puntos fuertes de las bases de datos no relacionales como:

- Capacidad de gestionar muchos más datos que las bases de datos relacionales.
- Mejores tiempos de respuesta para lecturas y escrituras en los datos.
- Gran escalabilidad.
- En implementaciones distribuidas, se evitan posibles cuellos de botella.
- Mejor compatibilidad entre datos de diferentes bases de datos; esto permite mayor interacción entre datos de diferentes servicios, y habilita la colaboración entre distintas empresas, por ejemplo.
- Diseño sencillo y de bajo coste.

1.5. Puntos débiles de las bases de datos no relacionales

No todo son fortalezas en la bases de datos no relacionales. A continuación, listamos algunas de las principales limitaciones de las bases de datos no relacionales:

- La gran mayoría de las bases de datos no relacionales se basan en código abierto, en contraposición con las bases de datos relacionales (con proveedores como Oracle, IBM, o Microsoft, entre otros), y esto es un punto fuerte desde el punto de vista técnico. Desde el punto de vista operacional, no suelen contar con el apoyo posventa de las soluciones más tradicionales, operadas por grandes empresas.

- Falta de madurez; aunque cada vez se aplican en más entornos profesionales, su madurez todavía no es comparable con las bases de datos relacionales más tradicionales.

- Limitación para aplicar los productos existentes de inteligencia artificial o *business intelligence* (BI); aunque creciendo, la cantidad de herramientas que existen para la minería de datos y el tratamiento del *big data* es mucho menor en el caso de las bases de datos no relacionales, en comparación con el abanico de opciones para soluciones relacionales tradicionales.

- Complejidad; su instalación, el modelo de datos, y las consultas a este tipo de bases de datos son más complejos que en las bases de datos relacionales.

- De manera relacionada con el punto anterior, encontramos menor disponibilidad de expertos; no estamos diciendo aquí que no hay expertos, pero en comparación con la cantidad de perfiles técnicos con grandes conocimientos en bases de datos SQL, el número de expertos en bases de datos no relacionales es mucho menor. Esto puede verse como una barrera de entrada y de competición, pero también como un punto débil, ya que resulta más costoso encontrar y retener el talento con conocimiento en este tipo de tecnologías del almacenamiento de datos.

- Del mismo modo que la flexibilidad era un punto positivo para poder interconectar bases de datos de diferente naturaleza, también puede analizarse como un punto negativo, ya que puede dificultar la interconexión entre bases datos; la falta de estructura de datos es flexible, sí, pero no facilita la intercomunicación entre diferentes sistemas.

Por lo tanto, es importante transmitir el mensaje de que las bases de datos no relacionales no son la solución para todo. Este tipo de bases de datos son apropiadas cuando los datos no son relacionales por naturaleza; en el caso de datos relacionales y bien estructurados por naturaleza, la solución relacional puede ser la mejor opción.

1.6. ¿Relacional o no relacional? El teorema CAP

Según el teorema de Brewer, es imposible para un sistema computacional distribuido, como puede ser el caso de una base de datos, ofrecer simultáneamente las siguientes tres garantías:

1) **Consistencia**: todos los usuarios ven los mismos datos al mismo tiempo.

2) Disponibilidad (***availability***): garantiza que cada petición de acceso o escritura de datos recibe una respuesta en un tiempo limitado.

3) Tolerancia a la partición (***partitioning***): la base de datos continúa funcionando, aunque se pierdan parte de los datos.

El teorema CAP apunta a que hay que elegir dos de estas opciones, ya que no es posible contar con las tres cualidades de manera simultánea. Por lo tanto, se da lugar al siguiente cuadro de decisión:

1) Disponible y resistente a la partición: bases de datos no relacionales.
2) Disponible y consistente: bases de datos relacionales.
3) Consistente y resistente a la partición: bases de datos no relacionales.

2. Tipos de bases de datos no relacionales

2.1. Arquitectura de las bases de datos NoSQL

Las bases de datos no relacionales suelen ofrecer garantías de consistencia débiles al emplear una arquitectura distribuida.

Como hemos comentado antes, los datos pueden incluso almacenarse físicamente en servidores distintos, usando espacios de memoria disjuntos.

Por este motivo, suelen basarse en estructuras de datos sencillas, tales como *arrays* asociativos o almacenes de pares de clave-valor.

2.2. Una posible clasificación

Según la manera de organizar y almacenar los datos, las bases de datos no relacionales pueden ser de varios tipos:

1) Bases de datos como almacenes de clave-valor.
2) Bases de datos como familias de columnas.
3) Bases de datos como almacenes de documentos.
4) Bases de datos como almacenes de grafos.

A continuación, detallamos brevemente cada uno de estos tipos de bases de datos no relacionales.

2.3. Bases de datos clave y multivalor

El precursor de este tipo de bases de datos fue Amazon Dynamo, basado en DHT (*distributed hash tables*).

En este caso, los datos se almacenan como **pares clave-valor**.

La clave es un valor que identifica cada registro, y los valores son vistos como cajas negras por la base de datos.

En este tipo de bases de datos, las consultas se llevan a cabo por la clave.

Ejemplos de este tipo de bases de datos son Dynamite, Voldemort o Tokyo.

2.4. Bases de datos orientadas a columnas

El precursor de este tipo de bases de datos es Google BigTable.

Se trata de un conjunto de columnas con distinta información. Cada entrada en la «tabla» puede contener información de columnas diferentes. De algún modo, podemos decir que cada «clave» está asociada con varios «valores» o «atributos», que se corresponden a diferentes columnas en la estructura.

Son idóneas para:

1) Almacenar grandes cantidades de datos.
2) Permitir cargas masivas de datos.
3) Garantizar alta disponibilidad.

Algunos ejemplos de este tipo de based de datos son HBase, Hypertable, Cassandra, o Riak.

2.5. Bases de datos documentales

El precursor de este tipo de bases de datos fue Lotus Notes.

En este caso, los datos son colecciones de documentos que contienen colecciones de pares de clave-valor. En general, el concepto de «documento» es utilizado para encapsular y codificar datos o información siguiendo algún formato estándar, como por ejemplo XML, JSON o formatos binarios como PDF y documentos Microsoft Office (MS Word, Excel y otros). El formato JSON (*JavaScript object notation*) es un estándar muy popular hoy día; se trata de un estándar abierto, basado en texto diseñado para intercambio de datos legible por humanos, que permite representar estructuras de datos simples y listas asociativas.

Por este motivo, se suele decir que este tipo de bases de datos guardan datos semiestructurados.

Los documentos en una base de datos orientada a documentos son similares a registros, pero no requieren un esquema estándar con las mismas secciones, partes, claves, etc.

Los documentos suelen ser direccionables por una clave que los representa de manera única.

Además de la búsqueda por clave de documento, estas bases de datos suelen ofrecer un lenguaje de consultas que permite recuperar documentos a partir de sus contenidos.

Este tipo de bases de datos gozan de las ventajas siguientes:

1) Son un modelado de datos muy natural.
2) Son amigables para el programador.
3) Permiten desarrollo de soluciones muy rápidas.
4) Están muy orientadas a integración con la web.

Ejemplo de este tipo de bases de datos son CouchDB o la popular MongoDB.

2.6. Bases de datos basadas en grafos

Este tipo de bases de datos están inspiradas en la teoría de grafos de Euler.

Estas bases de datos organizan la información basándose en la estructura de un grafo, es decir, como un conjunto de nodos y las relaciones entre pares de nodos. Estas bases de datos permiten emplear la teoría de grafos para recorrer la base de datos y hacer consultas sobre los datos.

Ejemplos de este tipo de bases de datos son AllegroGraph, VertexBD y Neo4J.

3. Ejemplos

Por ejemplo, **Instagram** utiliza una base de datos de **clave-valor**, conocida como **Redis**, para guardar las sesiones de usuario y para almacenar a los usuarios que han subido cada foto. Redis es una base de datos apoyada por VMWare. Se trata de una base de datos tipo clave-valor que se puede imaginar como un bloque de memoria reservada para almacenar datos, datos encadenados, cadenas encriptadas de datos o listas, por ejemplo.

Dentro de las bases de datos NoSQL **orientadas a columnas**, destacan HBase y Cassandra.

HBase es un sistema gestor de bases de datos agregado en columnas que ofrece alta disponibilidad, se ejecuta sobre HDFS y está totalmente integrado con Hadoop.

Cassandra también ofrece alta disponibilidad, rápido acceso a grandes cantidades de datos y una alta tolerancia a fallos. Se trata de una solución multiplataforma escrita en Java, que goza de popularidad por haber sido inicialmente adoptada por Facebook. Las últimas versiones incluyen un propio lenguaje de consulta, el llamado CQL (*Cassandra query language*), que posee una sintaxis similar a SQL, aunque con menos funcionalidades. Los dos sistemas permiten el almacenamiento de tablas de gran tamaño, es decir, tablas de miles de millones de filas por millones de columnas, en un entorno distribuido.

Aplicaciones como Facebook o Twitter usan Cassandra como almacén de datos, aunque en el caso de Facebook, por ejemplo, su estructura de datos es más compleja, ya que combina diferentes soluciones de datos para distintas funcionalidades de la aplicación de red social y mensajería.

MongoDB es un sistema gestor de base de datos documental de código abierto. Pretende combinar lo mejor de los almacenes de clave-valor, las bases de datos documentales y las tradicionales bases de datos relacionales. MongoDB usa JSON y tiene un propio lenguaje para hacer consultas. El hecho de estar programado en C++ hace que tenga muy buena acogida por parte de los desarrolladores. Se trata de la base de datos usada por SourceForge, Bit.ly, Foursquare o GitHub, entre otros. MongoDB es un sistema de datos NoSQL orientado a documentos; guarda estructuras de datos en documentos tipos BSON (JSON binario), lo que hace que la integración de datos y el acceso y la edición de los mismos sean realmente muy ágiles y rápidos.

Información adicional

`http://www.mongodb.org`.

Neo4j es un sistema gestor de bases de datos en grafo de código abierto. Almacena los datos en grafos de propiedades. En estos tipos de grafos, la información se

representa mediante nodos, relaciones entre nodos, etiquetas y propiedades. Los nodos permiten representar datos concretos del mundo real; las relaciones hacen posible representar interrelaciones entre nodos; las etiquetas permiten asignar nombres a los nodos y relaciones; y las propiedades son parejas clave-valor que pueden asignarse tanto a los nodos como a las relaciones. A diferencia de las otras bases de datos comentadas anteriormente, la distribución de los datos en las bases de datos en grafo es problemática y compleja. Compañías como eBay o Walmart utilizan Neo4j en sus soluciones empresariales. Otras, como por ejemplo Google, han desarrollado una base de datos en grafo propia, llamada Pregel, para relacionar páginas web.

Conclusiones

Como hemos visto, existen ventajas e inconvenientes respecto a las bases de datos relacionales y no relacionales. No hay una solución para todo; en algunos casos, es necesario seleccionar entre una opción y otra, y adoptar soluciones de compromiso; en otros casos, es posible que diferentes servicios deban usar esquemas de datos diferentes, combinando distintas soluciones y aplicando bases de datos relacionales y no relacionales, combinadas para dar una solución completa.

En general, en las bases de datos NoSQL los datos se pueden recuperar más rápido que en las bases de datos RDBMS. Sin embargo, las consultas que se pueden hacer son más limitadas, de modo que se necesita trasladar la complejidad del tratamiento de datos a un ámbito de aplicación. Es decir, si se desea aplicar técnicas de *big data* o *machine learning*, por ejemplo, en el caso de las bases de datos no relacionales, estas técnicas deben aplicarse en un ámbito de aplicación, y no de base de datos. Las bases de datos relacionales, por su parte, permiten aplicar inteligencia en el método en el que se consultan sus datos.

Por lo tanto, el uso de bases de datos NoSQL no es recomendable para aplicaciones que generan informes que usan consultas complejas.

Cuando los datos son muy estructurados, por lo tanto, las bases de datos relacionales siguen siendo una alternativa muy interesante.

Como conclusión final, podemos decir que, en un ámbito de sistema, de aplicación extremo a extremo, la combinación de SQL y NoSQL podría ser la solución ideal.

Resumen

En este material, hemos visto las limitaciones de las bases de datos relacionales. La gran estructuración de los datos en las bases de datos relacionales impone limitaciones para muchas de las aplicaciones que serán posibles en el mundo hiperconectado al que nos acercamos, precursor de la industria 4.0.

Como solución, aparecen las bases de datos no relacionales, que ofrecen mucha más flexibilidad y escalabilidad, algo que, por lo tanto, las hace muy interesantes para uso en las aplicaciones de la industria 4.0. Su uso viene motivado también por la generalización de la computación en la nube y la computación distribuida.

Hemos visto que hay diferentes tipos de bases de datos no relacionales y hemos descrito, en cada caso, las principales ventajas y desventajas de cada tipo de base de datos:

1) Bases de datos clave y multivalor.
2) Bases de datos documentales.
3) Bases de datos basadas en grafos.
4) Bases de datos orientadas a objetos.

Finalmente, hemos descrito algunos ejemplos específicos reales de empresas y productos que se basan en el uso de bases de datos no relacionales, en contraposición con las bases de datos relacionales tradicionales.

A lo largo del material, hemos valorado pros y contras de las bases de datos relacionales y no relacionales. Como conclusión final podemos decir que, en un ámbito de sistema, de aplicación extremo a extremo, la combinación de SQL y NoSQL podría ser la solución ideal. Las dos opciones tienen puntos fuertes y debilidades; lo más aconsejable, por tanto, es combinar soluciones para obtener lo mejor de las dos alternativas.