ACCESO A DATOS

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Introducción a las bases de datos No-SQL
Concepto Big Data

Características de las bases de datos No-SQL

Tradicionalmente, la industria del software como ya bien sabemos, usa las bases de datos relacionales para almacenar y manejar datos de forma persistente. **No solo bases de datos SQL y No-SQL** son las que han emergido en este pasado reciente.

No-SQL se refiere a **todas** las bases de datos y almacenes de datos que **no están basadas** en el sistema tradicional de bases de **datos relacionales** en sí, o no están basadas en sus **principios**.

Suele relacionarse con **grandes conjuntos de datos** a los que se accede y se manipulan a **escala web**. El concepto No-SQL no representa un producto único o una única tecnología, representa un **grupo de productos** y un conjunto relacionado de **conceptos** para el almacenamiento y su administración. Fue también un **hashtag** escogido para una **gran reunión técnica** para discutir las **nuevas bases de datos** (un hashtag es un conjunto de caracteres precedidos por una almohadilla (#) que sirve para identificar o etiquetar un mensaje en las webs de microblogs, p. e. el hashtag #elecciones en Twitter).

Las bases de datos No-SQL tienen una serie de **características comunes** como:

1. Modelo de datos **no relacional**.
2. Funcionan bien en **clusters** (cluster podría traducirse al español como aglomerado, grupo, racimo o conjunto. Dentro de las Tecnologías de la Información (TI), cluster significa integrar **dos o más computadoras** para que trabajen **simultáneamente** en el **procesamiento de una determinada tarea**).
3. Suelen ser en su **mayoría** de **código abierto**.
4. Su construcción está orientada para las **aplicaciones web de nueva generación**.
5. Son bases de datos con **ausencia de esquema** (schemaless).

Dentro de las diferentes características mencionadas aclararemos algunas de ellas que puedan dar lugar a dudas:

Funciona bien en modo cluster: Podemos definir básicamente este concepto como un conjunto de máquinas implementadas como servidores de procesamiento paralelo. Funcionan entre sí como un único recurso. Y como bien sabemos a cada elemento o máquina lo llamamos nodo.

Cluster de servidores: es un grupo de servidores vinculados que trabajan en estrecha **colaboración** y **se implementan** para mejorar el **rendimiento** y/o la **disponibilidad**, en comparación a los recursos ofrecidos por un solo servidor.

Bases de datos con ausencia de esquema o schemaless: Es una característica **muy flexible**; se puede almacenar **información no uniforme** y se facilita así la **evolución**.

Fundamentos de las bases de datos No-SQL

Con la explosión del **social-media** y el **contenido manejado** por el usuario, han **incrementado** el volumen y el tipo de datos que se **producen, se manejan, analizan y se archivan**, por lo que han provocado la rápida eclosión de las bases de datos No-SQL. Además, a esta cantidad ingente de datos, hay que sumarle las aportaciones de las nuevas fuentes de información como:

- **sensores**,
- sistemas globales de posicionamiento o **GPS**,
- **rastreadores**,
- otros tipos de **sistemas que monitorean grandes cantidades** de información de forma regular.

Estos grandes paquetes de información, han introducido nuevos retos y oportunidades para el almacenamiento de información. Además, los datos son cada vez más **semiestructurados y escasos**. Esto quiere decir que las **bases de datos relacionales son examinadas** y requieren una **definición de esquema inicial** y **referencias** relacionales.

Para resolver el problema relacionado con los grandes volúmenes de información y los datos semi-estructurados, surge **nuevas clases de bases de datos** dentro de la familia **No-SQL**. Este tipo de base de datos consisten en el **almacenamiento basado en:**

- **columnas**,
- **clave/valor**,
- **documentos**.

A continuación, veremos las **diferencias más notables** entre las bases de datos relacionales y las No-SQL:

Diferencias BBDD relacionales y NoSQL

Bases de datos relacionales	Bases de datos NoSQL
• La información está almacenada en un modelo relacional con filas y columnas. • Una fila contiene información sobre un elemento mientras que las columnas contienen información específica. • Sigue un esquema fijo: Las columnas son **definidas y establecidas** antes de la entrada de datos. Además, **cada fila** contiene información de **cada columna**. • A favor del escalado vertical. • Atomicidad, consistencia, aislamiento y durabilidad. (ACID)	• La información está almacenada en un cliente o en una base de datos diferente con modelos de datos distintos. • Sigue un esquema dinámico, puedes añadir columnas en cualquier momento. • Facilita el escalado horizontal. Se puede escalar a través de **servidores múltiples**. Los servidores múltiples tienen la **ventaja** del **precio** en relación al Hardware que se va añadiendo comparado con el escalado vertical. • No está a favor de los principios ACID.

Beneficios de las bases de datos No-SQL

A continuación, en las próximas diapositivas, veremos en profundidad los diferentes beneficios técnicos de una solución No-SQL a nivel empresarial:

a) Capacidad de fuente de datos primaria y de análisis

El primer criterio de una clase empresarial NoSQL es que debe servir como **fuente principal** de datos que **recibe** información de **distintas aplicaciones** de negocio. También debe actuar como **segunda fuente de información o análisis** las aplicaciones de "**business intelligence**". Desde el punto de vista de negocio, este tipo de bases de datos deben de ser capaces de **integrarse de una forma rápida** a todos los tipos de **datos estructurados, semiestructurados o sin estructura**. Además, deben ser capaces de ejecutar consultas de alto rendimiento.

b) Capacidad Big Data

Las bases de datos NoSQL no se limitan a trabajar con Big Data. Una base de datos de este tipo de clase empresarial, puede escalar para **administrar grandes volúmenes de datos** desde Terabytes **hasta Petabytes (10^6 GB, o 1 millón de GigaBytes)**. Además de almacenar grandes volúmenes de datos, ofrece un **alto rendimiento** para la **velocidad, variedad y complejidad** de los datos.

c) Disponibilidad continua

Para que una base de datos sea considerada de clase empresarial, debe ofrecer disponibilidad continua, sin un solo punto de fallo.

Además, en lugar de proporcionar la función de disponibilidad continua fuera del software, la solución NoSQL ofrece una disponibilidad continúa integrada, que debe incluir las siguientes características clave:

1. Todos los nodos de un clúster deben poder **atender solicitudes** de lectura incluso **si algunas máquinas no funcionan**.

2. Debe ser capaz de **replicar y segregar datos fácilmente** entre diferentes partes físicas en un centro de datos. Esto **evitará cortes** de hardware.

3. Debe poder admitir **diseños de distribución de datos** que sean **centros de datos múltiples** ya sea en instalaciones físicas o en la nube.

d) Capacidad de tener **múltiples centros de datos:**

Por lo general, en un entorno profesional, las empresas poseen bases de datos altamente distribuidas que se encuentran en varios centros de datos y ubicaciones geográficas distintas.

La replicación de datos es una característica que ofrecen todas las bases de datos relacionales. Sin embargo, ninguna puede ofrecer un **modelo simple de distribución de datos entre varios centros** de datos **sin causar problemas** de rendimiento.

Una buena solución empresarial NoSQL debe admitir la **implementación de varios centros de datos** y debe proporcionar una **opción configurable** para mantener un **equilibrio entre el rendimiento y la coherencia**.

e) **Fácil replicación** *independientemente* de la ubicación:

Para evitar que la pérdida de datos afecte a una aplicación, una buena solución NoSQL proporciona una **gran capacidad de replicación**. Estos incluyen una capacidad de lectura y escritura **en cualquier lugar** con **compatibilidad** total y *independencia* de ubicación.

Esto significa que se pueden escribir datos en **cualquier nodo de un clúster**, hacer que se repliquen en otros nodos, y ponerlos a disposición de todos los usuarios **independientemente de su ubicación**.

Además, la capacidad de escritura en cualquier nodo debe garantizar que los d**atos estén seguros en caso de un corte** de suministro eléctrico o cualquier otro tipo de incidente.

f) **Sin capa de almacenamiento en caché separada**

Una buena solución NoSQL es capaz de **utilizar múltiples nodos** y **distribuir datos** entre **todos los nodos** adjuntos.

Una vez aclarado esto, podemos decir que **no requiere** una capa de almacenamiento en **caché específica** para almacenar datos. Las **memorias caché de todos los nodos** pueden almacenar datos para una **entrada y salida rápida** o para un **acceso entrada/salida**. La base de datos NoSQL **elimina el problema de sincronizar** los datos de la **caché con la base de datos** persistente. Por lo tanto, admite una escalabilidad simple con **menos problemas de administración**.

g) **Lista para la nube**

Dado que la adaptación de la **infraestructura de la nube** aumenta día a día, una solución NoSQL de nivel empresarial debe estar preparada para la nube.

Un clúster de base de datos NoSQL debe poder funcionar en una configuración de la nube, como **Amazon EC2**, y también debe poder **extender y reducir un clúster** cuando sea necesario. También debe admitir una **solución híbrida** en la que parte de la base de datos se aloje dentro de las **instalaciones de la empresa** y otra parte se aloje en la **nube**.

h) **Alto rendimiento con escalabilidad lineal**

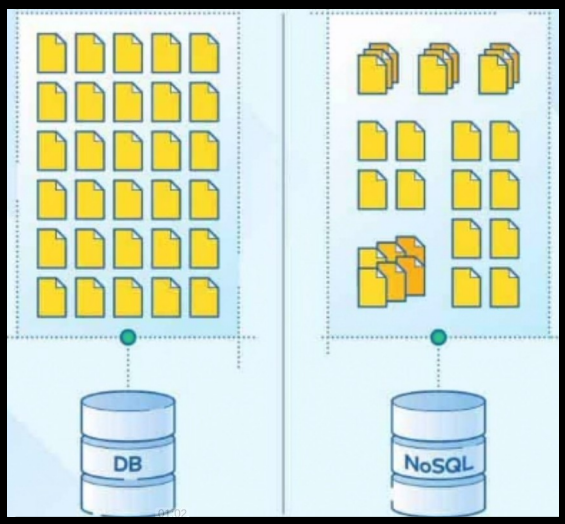
Una base de datos de este tipo puede mejorar el rendimiento **agregando nodos** a un clúster. Normalmente, el rendimiento de otros sistemas de bases de datos puede disminuir cuando se agregan los nodos adicionales, sin embargo, una buena solución NoSQL **aumenta el rendimiento** de las operaciones de lectura y escritura cuando se agregan **nodos adicionales**. Estas **ganancias** de rendimiento son **de naturaleza lineal**.

Finalmente, algunos beneficios más de las bases de datos No-SQL serían:

- Soporte de **esquema flexible**.
- Admite **lenguajes y plataformas clave** para desarrolladores.
- **Fácil de implementar, mantener y extender**.
- **Comunidad de código abierto**.

Desventajas NoSQL

Podemos ver aquí esquemáticamente un resumen de lo que son la base de datos relacionales y la base de datos no SQL.



Referente a la base de datos NoSQL, ya hemos dicho que son bases de datos que incluyen diferentes tipos de almacenes, como almacenes de tipo columna, de documento, de clave valor, de gráficos, etc., cosas que hacen fácil y rápido el acceso; por ejemplo, en las de clave valor, cuando se asigna un hash directamente a una clave para obtener el valor, este tipo de acceso y de entrega de documentos es muy eficiente.

Los beneficios ya los hemos visto, pero lo que realmente no hemos visto aún son algunas **desventajas**, que también hay que decirlas:

1. Hay que decir que la mayoría de bases de datos no SQL no admiten funciones de fiabilidad y que son soportadas por sistemas de bases de datos relacionales, estas de las que hablamos. Estas características las podemos resumir como los principios de:
    - atomicidad,
    - consistencia,
    - aislamiento,
    - durabilidad.
2. Con el fin de apoyar estas características de fiabilidad y coherencia, los propios desarrolladores de bases de datos no SQL deben implementar su propio código, lo que esto le da un plus de **complejidad** al sistema. Al tener el propio tipo de sistema desarrollado por ellos mismos, ellos son los propios que tienen que desarrollar el código. Podría ser otra desventaja. Esto podría limitar el número de aplicaciones en las que podemos confiar para relacionar, por ejemplo, transacciones seguras y confiables.

En este tipo de **aplicaciones bancarias y de sistemas de datos** no SQL estas acciones **no se pueden** hacer.

3. La mayoría de bases de datos no SQL no tienen funciones de fiabilidad y coherencia. Esto, por ejemplo, puede ser que con el fin de apoyar estas características de fiabilidad y coherencia no se pueda hacer.
4. Otras formas de complejidad encontradas en la mayoría de las bases de datos no SQL incluyen que no tienen compatibilidad con las famosas consultas SQL. Esto significa que tienen que tener su propio lenguaje de consulta de datos manual y hace el proceso un poco más lento y complejo.

Tipos de bases de datos NoSQL

Existen **cuatro tipos** de bases de datos NoSql en el mercado:

1. Bases de datos clave/valor:

   Velocidad brutal.

   La base de datos clave valor, es realmente una **tabla hash de claves y valores**.

   Algunos de los ejemplos son:

   - **Riak**,
   - **Tokyo Cabinet**,
   - servidor **Redis**,
   - Memcached,
   - Scalaris.

2. Bases de datos basadas en documentos:

   Este tipo de bases de datos basadas en documentos almacena documentos compuestos por **elementos etiquetados**.

   Algunos ejemplos son:

   - **MongoDB**,
   - **CouchDB**,
   - **OrientDB,**
   - **RavenDB**.

3. Bases de datos basadas en columnas:

   Cada bloque de almacenamiento contiene **datos de una sola columna**.

   Ejemplos:

   - **BibTable**,
   - **Cassandra**,
   - **Hbase**,
   - **Hypertable**.

4. Bases de datos basadas en gráficos:

   Una base de datos basada en gráficos es una base de datos de red que **utiliza nodos para representar y almacenar datos**.

   Algunos ejemplos son:

   - **Neo4J**,
   - **InfoGrid**,

- **Infinite Graph**,
- **FlockDB**.

La **disponibilidad de opciones** en las bases de datos NoSQL tiene sus propias

- ventajas (permite elegir un diseño **de acuerdo a los requisitos** de su sistema),
- e inconvenientes (aun ajustando el producto a los recursos del sistema **no siempre funcionará** correctamente).

Ejemplo REDIS NoSQL Database

Para realizar una **conexión directa** con una base de datos **No-SQL Redis**, imaginemos que nuestra aplicación está desarrollada en **lenguaje Java** y tenemos que implementar la parte de la **conexión a base de datos**.

Vamos a realizar la **construcción de la capa de acceso a datos** de la aplicación.

Los pasos a seguir serán los siguientes:

1. Agregación de **dependencias** a nuestro fichero pom.xml del proyecto, para cargar la **librería** necesaria y así **acceder a la base de datos Redis**.

Dependencia Redis

```
<dependency>
	<groupId>redis.clients</groupId>
	<artifactId>jedis</artifactId>
	<version>2.9.0</version>
</dependency>
```

Como vemos en el código anterior de esta forma **agregaremos la versión** específica que nos realizará la **conexión con la base de datos**.

2. Implementación de la nueva capa usando las **librerías previamente importadas** en **Maven**.

Integración Redis

```
import redis.clients.jedis.Jedis;

public class RedisConnector {
	public static void main(String[] args) {
		Jedis cliente = new Jedis("localhost");
		cliente.set("clave", "valor");
		String value = cliente.get("clave");
		System.out.println(value);
		cliente.close();
	}
}
```

De esta forma podremos **instanciar tanto el cliente**, como **establecer valores en la base de datos** al igual que **obtenerlos** con los diferentes métodos que observamos en el código anterior.

Introducción a Big Data

DATA: Realmente son las **cantidades, los caracteres o símbolos** con los que se realizan **operaciones** mediante una **máquina**. Pueden **almacenarse y transmitirse** en forma de señales eléctricas y registrarse en soportes de almacenamiento **magnéticos, ópticos o mecánicos**.

Una vez conocido el concepto de datos veremos Big Data.

Big Data: Son realmente **datos**, pero con la diferencia que su **tamaño** es **muy grande**. Con Big Data definimos en realidad una **colección de datos** que es **gigante** y, sin embargo, **crece** cada vez más con el tiempo. Los datos que manejamos son **tan grandes y tan tediosos de administrar** que no se podía manejarlos con las herramientas que había hasta la fecha.

¿**Por qué** las bases de datos Big Data **son sistemas diferentes**?

Para poder trabajar con Big Data necesitaríamos **en principio el mismo conocimiento** como si fuésemos a trabajar con **datos de menor volumen**.

Pero algunas características como la **escalabilidad** a gran volumen, la **rapidez** de mover información y de procesarla, y las **propiedades** de algunos de los datos que se tratarán en el proceso, presentan nuevos desafíos importantes a la hora de diseñar soluciones. El objetivo de la mayoría de los sistemas de Big Data es sacar a la luz **conocimientos y conexiones de grandes volúmenes de datos heterogéneos** que no serían posibles con métodos convencionales.

Aplicaciones Big Data

Vamos a ver a qué están orientadas las bases de datos Big Data y en qué tipo de aplicaciones se usan.

Las bases de datos Big Data, sobre todo se usan para la **web y redes sociales**, comprenden toda la información que podemos tener de los usuarios gracias a sus interacciones en distintas redes sociales, por ejemplo, Facebook, Twitter, Instagram y LinkedIn, entre otras. ¿Qué significa esto? Que hay **gran cantidad de datos, gran nivel de interacciones**.

Si queremos una **buena eficiencia** en la base de datos, con la cual podamos obtener y generar **gran cantidad de datos al mismo tiempo**, encaja perfectamente este modelo.

Otras opciones son para trabajar con **grandes transacciones**. No son datos tan sencillos de obtener y de segmentar como los anteriores, que hablábamos de las redes sociales, pero son aquellos que provienen de **redes sociales**. Estos datos son los **registros de facturación** que tenemos almacenados, así como las **llamadas**. Estos datos van a ser estructurados y no estructurados.

Otro tipo de datos que tenemos son los datos **M2M**, de las siglas **Machine to Machine**. Estos datos se obtienen a través de las tecnologías que **se conectan a otros dispositivos**. Por ejemplo, cuando un usuario accede a nuestro Wi-Fi, se conecta a nuestro dispositivo mediante Bluetooth. También se puede acceder a otros dispositivos mediante redes inalámbricas o híbrida. Estos datos son los M2M Machine to Machine.

También se almacenan en este tipo de bases de datos **datos biométricos**. Por ejemplo, el escaneo de las retinas, las huellas digitales o las tecnologías de reconocimiento facial, permiten recoger este tipo de datos biométricos. Son datos realmente interesantes. Para algunas áreas como por ejemplo la de seguridad e inteligencia. Este tipo de datos suelen almacenarse también de esta forma.

Tipos de Big Data

Podemos encontrar Big Data de **tres formas** diferentes:

a) Datos Estructurados

Definimos datos estructurados como toda aquella información que pueda ser **almacenada**, se pueda **acceder** y se accede, **de forma fija**.

Poco a poco conforme ha ido pasando el tiempo, los sistemas de almacenamiento han sido mejorados con distintas **técnicas innovadoras** que **mejoran el trabajo** con este tipo de datos. Pero en la actualidad existe un **problema real**, los datos crecen en tal medida que alcanzan tamaños como el **zettabyte**:

> **10^{21} bytes** forman 1 zettabyte o lo que es lo mismo **un billón de terabytes** forman 1 zettabyte.

Viendo estas cifras podemos entender fácilmente por qué se le da el nombre de Big Data.

b) Datos No estructurados

Cualquier dato cuya **forma o estructura sea desconocida** se clasificará como datos no estructurados. Además de que el tamaño es enorme, los datos no estructurados plantean múltiples desafíos si nos referimos al procesamiento de los mismos. Hoy en día **muchas compañías** disponen de sistemas no estructurados donde **mezclan** colecciones que contienen **imágenes de texto, números, direcciones, ficheros media, etc**. Esto puede suponer un **problema real** si estos datos **no son procesados** y si no se sabe sacar partido de ellos.

c) Datos Semi-estructurados

Aparentemente los datos semi-estructurados tienen la similitud de ser en parte, datos estructurados en cuyo interior poseen datos no estructurados. La realidad es que **no están planteados para ser plenamente estructurados como una tabla**.

Un **ejemplo** sería los datos **representados** como en el **formato XML**.

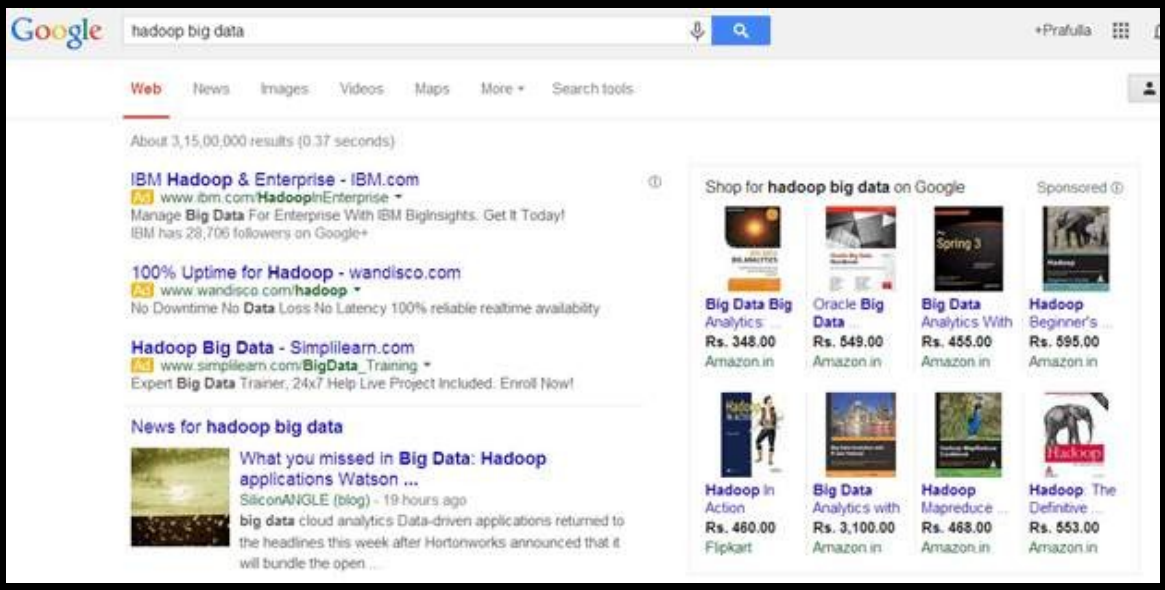
Ejemplos de diferentes tipos estructuras Big Data

Vamos a **clasificar** entre los distintos tipos de **estructuras existentes de Big Data**, los siguientes tipos:

Employee_ID	Employee_Name	Gender	Departament	Salary_In_lacs
2365	Rajesh Kulkarni	Male	Finance	650000
3398	Pratibha Joshi	Female	Admin	650000
7465	Shushil Roy	Male	Admin	500000
7500	Shubhojit Das	Male	Finance	500000
7699	Priya Sane	Female	Finance	550000

```
<rec>
        <name>Prasnant Kao</name>
        <sex>Male</sex>
        <age>35</age>
</rec>
<rec>
        <name>Seema R.</name>
        <sex>Female</sex>
        <age>41</age>
</rec>
<rec>
        <name>Satish Mane</name>
        <sex>Male</sex>
        <age>29</age>
</rec>
<rec>
```

```
        <name>Subrato Roy</name>
        <sex>Male</sex>
        <age>26</age>
</rec>
<rec>
        <name>Jeremiah J.</name>
        <sex>Male</sex>
        <age>35</age>
</rec>
```



Vamos a identificar la **tipología de Big Data** en función a los parámetros estudiados previamente.

Si nos fijamos en la tabla podemos ver que es un tipo de **datos estructurados**, es el ejemplo de una tabla de empleado con sus columnas bien definidas.

En el código XML podemos ver que tenemos un ejemplo claro de **datos semi estructurados**, un buen ejemplo de datos semi-estructurados son los ficheros XML y su estructura.

En la imagen del navegador con el buscador de Google, podemos observar un ejemplo de **datos no estructurados** en este caso es la respuesta de una búsqueda en Google Chrome. Podemos ver que tenemos una respuesta de **texto, etiquetas, información multimedia, etc**.

Top 5 NoSQL

Las cinco **soluciones NoSQL más relevantes y más conocidas** en el mercado, son:

• MongoDB: Estamos ante la base de datos NoSQL **más conocida** a nivel de solución empresarial. Es un importante gestor de datos que almacena documentos en un **formato muy parecido a JSON**.

• Apache Casandra: Base de datos **basada en columnas**, diseñada para almacenar cantidades muy grandes de datos y realizar **balanceo a distintos nodos**. A modo curiosidad es una de las herramientas que se usan en **Facebook**.

• CouchDB: Nació con la idea de ser la principal base de datos NoSQL de la red, pero en 2008 el proyecto pasó a formar parte del proyecto Apache Incoubator. Su intención principal es la de **adaptar y compatibilizar la web** con distintos tipos de **dispositivos**. Almacena los datos en formato **Json**.

• Redis: Es otro importante sistema de base de datos NoSQL en este caso **clave-valor**. Es de **código abierto** y patrocinada por RedisLabs. Está basado en el almacenamiento de **tablas tipo hash**, aunque no se cierra solo a esta tipología, no obstante, es **su fuerte**.

• Neo4J: Fue desarrollada en **software libre**, es totalmente distinta a las 4 anteriores que hemos presentado ya que es una base de datos **orientada a grafos**, desarrollada en **lenguaje Java**.

Instalación y conexión a base de datos Cassandra

Instalación Cassandra

1. Instalación de Cassandra. Descarga e instala Cassandra en tu máquina desde el sitio oficial.

   https://cassandra.apache.org/_/download.html

2. Iniciar el Servidor de Cassandra. Inicia el servidor de Cassandra ejecutando el comando correspondiente a tu sistema operativo.

3. Creación de una Keyspace

   a. Con CLI

```
CREATE KEYSPACE demo
With placement_strategy = 'org.apache.cassandra.locator.SimpleStrategy'
and strategy_options = { replication_factor :1 };
```

   b. Con CQL

```
CREATE KEYSPACE demodb
 WITH REPLICATION = {'class' : 'SimpleStrategy', 'replication_factor': 1};
```

4. Crear tabla

```
CREATE TABLE MiTabla (id UUID PRIMARY KEY, nombre TEXT, edad INT);
```

Conexión con Java

Para conectar Cassandra con una aplicación Java existen varios Drivers de conectores. Uno de los conectores populares es **DataStax** Java Driver para Cassandra. Para usarlo lo añadimos a las **dependencias** Maven del proyecto:

```
<dependency>
        <groupId>com.datastax.oss</groupId>
        <artifactId>java-driver-core</artifactId>
        <version>4.13.0</version><!-- Verifica la última versión en el
        repositorio de Maven -->
</dependency>
```

Código Java de ejemplo:

```
import com.datastax.oss.driver.api.core.CqlSession;
import com.datastax.oss.driver.api.core.cql.Row;
import java.util.UUID;

public class CassandraConnector {

	public static void main(String[] args) {
		CqlSession session = null;
		try {
			// Conectar a Cassandra
			session = CqlSession.builder().build();
			// Ejemplo de inserción
			UUID id = UUID.randomUUID();
			String nombre = "John Doe";
			int edad = 30;
			session.execute(
				"INSERT INTO MiTabla (id, nombre, edad) VALUES (?, ?, ?)",
				id,
				nombre,
				edad
			);
			// Ejemplo de consulta
			com.datastax.oss.driver.api.core.CqlResult result = session.execute(
				"SELECT *
				FROM MiTabla"
				);
				for (Row row : result) {
				UUID resultId = row.getUuid("id");
				String resultNombre = row.getString("nombre");
				int resultEdad = row.getInt("edad");
				System.out.printf(
					"ID: %s, Nombre: %s, Edad: %d%n",
					resultId,
					resultNombre,
```

```
                                        resultEdad
                                );
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                } finally {
                        if (session != null) {
                                session.close();
                        }
                }
        }
}
```

Sectores donde el Big Data está más extendido y sus usos

1. Banca
    - Plataformas de análisis de riesgos financieros.
    - Sistemas de detección de fraudes.
    - Herramientas de personalización para servicios financieros.
2. Salud
    - Análisis de datos clínicos para diagnósticos precisos.
    - Gestión de registros médicos electrónicos.
    - Investigación biomédica y farmacéutica.
3. Comercio Electrónico
    - Recomendaciones personalizadas para compradores.
    - Análisis de patrones de compra y comportamiento del cliente.
    - Optimización de la cadena de suministro.
4. Telecomunicaciones
    - Análisis de datos de uso para mejorar la calidad del servicio.
    - Predicción de la demanda de servicios.
    - Gestión de redes y prevención de fallas.
5. Manufactura
    - Mantenimiento predictivo de maquinaria.
    - Optimización de la cadena de producción.
    - Control de calidad basado en datos.
6. Educación
    - Análisis de datos para mejorar la retención estudiantil.
    - Personalización del aprendizaje.
    - Evaluación del desempeño de los estudiantes.
7. Energía
    - Monitoreo y gestión de la red eléctrica.
    - Optimización de la distribución de energía.
    - Análisis de datos para eficiencia energética.
8. Medios y Entretenimiento
    - Recomendaciones de contenido personalizado.
    - Análisis de audiencia y comportamiento de visualización.

- Producción de contenido basada en datos.

El uso de Big Data se extiende a través de diversos sectores, desde la banca y la salud hasta la manufactura y la educación. Las aplicaciones varían, desde análisis de riesgos financieros y diagnósticos médicos hasta recomendaciones personalizadas y optimización de la cadena de suministro. La capacidad de extraer conocimientos significativos de grandes conjuntos de datos está transformando la forma en que las organizaciones toman decisiones y ofrecen servicios.