



Facultad de ingeniería

# Bases de datos documentales

Materia: Laboratorio 2

Alumnos: Gabriel Pérez, Daira Orlandini, Enoc García, Favio Ontón

Profesor: Juan Lopreiato

Ciudad Autónoma de Buenos Aires, Argentina – 2022

## Tabla de contenido

NoSQL .....	2
Bases De Datos Documentales .....	4
Ventajas .....	5
Desventajas: .....	5
Casos de uso .....	6
Administración de contenido .....	6
Catálogos .....	6
Perfiles de usuario.....	7
Big data en tiempo real .....	7
Diferencias con otras DDBB .....	7
Diferencias con bases de datos relacionales .....	7
Diferencias con otras bases NoSQL .....	8
Motores más utilizados .....	8
Escalabilidad.....	9
Bibliografía .....	10

## NoSQL

El término "NoSQL" se refiere a tipos de bases de datos no relacionales que almacenan datos en un formato distinto a las tablas relacionales. Sin embargo, las bases de datos

NoSQL se pueden consultar utilizando API de lenguaje natural, lenguajes de consulta estructurados declarativos y lenguajes de consulta mediante ejemplo, por lo que también se les llama bases de datos "no solo SQL".

Las bases de datos NoSQL se utilizan de forma generalizada en aplicaciones web en tiempo real y big data, ya que sus principales ventajas son los elevados niveles de escalabilidad y disponibilidad.

Las bases de datos NoSQL también son la opción preferida entre los desarrolladores, ya que propician por su propia naturaleza un desarrollo ágil dada su rápida adaptación a los requisitos en constante cambio. Las bases de datos NoSQL permiten almacenar los datos de forma más intuitiva y fácil de entender, o más cercanas a la forma en que las aplicaciones utilizan los datos: no necesitan realizar tantas transformaciones cuando almacenan o recuperan datos utilizando interfaces API de NoSQL. Además, las bases de datos NoSQL pueden aprovechar al máximo la nube para evitar por completo el tiempo de inactividad.

Si se comparan con las bases de datos relacionales, las bases de datos NoSQL son más escalables y ofrecen un mayor rendimiento; además, su modelo de datos aborda varias cuestiones que el modelo relacional pasa por alto tales como:

- Grandes volúmenes de datos estructurados, semiestructurados y no estructurados en constante cambio
- Sprints de desarrollo ágiles, iteración rápida de los esquemas y generación frecuente de código
- Programación orientada a objetos flexible y fácil de usar
- Arquitectura de escalado horizontal distribuida geográficamente, en lugar de una arquitectura monolítica

# Bases De Datos Documentales

Las bases de datos documentales, en inglés, documents stores, se utilizan para la administración de datos semiestructurados. Se trata de datos que no siguen una estructura fija, sino que llevan la estructura casi en sí misma. Sin embargo, con ayuda de marcadores dentro de estos datos, la información puede ordenarse. Debido a la falta de una estructura clara, estos datos no son adecuados para bases de datos relacionales porque su información no puede clasificarse en tablas.

Este modelo es un tipo de base de datos no relacional que ha sido diseñada para almacenar y consultar datos como documentos de tipo JSON. Las bases de datos de documentos facilitan a los desarrolladores el almacenamiento y la consulta de datos en una base de datos mediante el mismo formato de modelo de documentos que emplean en el código de aplicación. La naturaleza flexible, semi-estructurada y jerárquica de los documentos y las bases de datos de documentos permite que evolucionen según las necesidades de las aplicaciones.

Estas crean un par simple: un documento específico se asigna a una clave. En este documento, que puede tener formato XML, JSON o YAML, por ejemplo, se puede encontrar la información propiamente dicha. Como la base de datos no requiere un esquema específico, en un almacén de documentos pueden integrarse diferentes clases de documentos. Los cambios en los documentos no afectan a la base de datos.

Este es un ejemplo en formato JSON

```
{
  '_id' : 1,
  'artistName' : { 'Iron Maiden' },
  'albums' : [
    {
      'albumname' : 'The Book of Souls',
      'datereleased' : 2015,
      'genre' : 'Hard Rock'
    }, {
      'albumname' : 'Killers',
      'datereleased' : 1981,
      'genre' : 'Hard Rock'
    }, {
      'albumname' : 'Powerslave',
      'datereleased' : 1984,
      'genre' : 'Hard Rock'
    }, {
      'albumname' : 'Somewhere in Time',
      'datereleased' : 1986,
      'genre' : 'Hard Rock'
    }
  ]
}
```

# Ventajas

- **Creación más rápida de documentos y mantenimiento:** es muy simple crear un documento y aparte de esto, no requiere de mucho mantenimiento.
- **Sin esquema:** estos son muy buenos para retener datos existentes en volúmenes masivos porque no hay absolutamente ninguna restricción en el formato y la estructura del almacenamiento de los datos.
- **Formatos abiertos:** tiene un proceso de compilación muy simple que usa XML, JSON y sus otras formas.
- **Control de versiones incorporados:** tiene un control de versiones incorporado, lo que significa que a medida que los documentos aumentan de tamaño, existe la posibilidad que aumentan en complejidad. El control de versiones reduce los conflictos.
- **Facilidad de integrar datos nuevos:** mientras que, en una base de datos relacional, el nuevo punto de información se debe insertar en todos los registros de datos, en una base de datos orientada a documentos basta con integrar la información nueva en solo unos pocos registros. El contenido adicional se puede añadir a otros documentos, pero esto no es necesario.
- Tiene un modelo que facilita la tarea de actualización de datos.
- De igual forma, mantienen un gran formato para contener muchos datos e información.
- Da la posibilidad de consultar y mover datos semiestructurados, aunque no tenga estructura definida.
- Es de las mejores cuando se trata de guardar grandes cantidades de información.
- Cuentan con buenos motores de búsqueda y buenas propiedades de indexación.

# Desventajas:

- **Redundancia:** elementos relacionales no encajan en el modelo documental, lo que implica que los datos se insertan de manera redundante en los documentos al no poder referenciar otro documento para extraer la información. Hacer referencias entre documentos puede hacer crecer de manera exponencial el volumen de datos, por lo que es preferible en esos casos usar modelos relacionales.
- **Atomicidad débil:** carece de soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) de múltiples documentos. Un cambio en el modelo de datos del documento que involucre dos colecciones requerirá que ejecutemos dos consultas separadas, es decir, una para cada colección. Aquí es donde rompe los requisitos de atomicidad.

- **Limitaciones de la verificación de consistencia:** se pueden buscar las colecciones y los documentos que no están conectados a una colección de autor, pero hacerlo podría crear un problema en el rendimiento de la base de datos.
- **Seguridad:** hoy en día, muchas aplicaciones web carecen de seguridad, lo que a su vez da como resultado la fuga de datos confidenciales. Entonces se convierte en un punto de preocupación, uno debe prestar atención a las vulnerabilidades de las aplicaciones web.

## Casos de uso

### Administración de contenido

Para administrar eficazmente el contenido, debe poder recopilar y agrupar contenido de una variedad de orígenes y enviárselo al cliente. Debido a su esquema flexible, las bases de datos documentales son perfectas para recopilar y almacenar cualquier tipo de datos. Puede utilizarlas para crear e incorporar nuevos tipos de contenido, incluido el contenido generado por el usuario, como imágenes, comentarios, y vídeos

Una base de datos de documentos es una excelente opción para aplicaciones de administración de contenido, como blogs y plataformas de vídeo. Con una base de datos documental, cada entidad que rastrea la aplicación se puede almacenar como un único documento. La base de datos de documentos es más intuitiva para que un desarrollador actualiza una aplicación a medida que evolucionan los requisitos. Además, si el modelo de datos necesita cambiar, solo se deben actualizar los documentos afectados. No se requiere actualización del esquema y no es necesario tiempo de inactividad de la base de datos para realizar los cambios.

### Catálogos

Las bases de datos de documentos son eficientes y efectivas para almacenar información de catálogo. Por ejemplo, en una aplicación de e-commerce, los diferentes productos generalmente tienen diferentes números de atributos. La administración de miles de atributos en bases de datos relacionales no es eficiente y afecta al rendimiento de lectura. Al utilizar una base de datos de documentos, los atributos de cada producto se pueden describir en un solo documento para que la administración sea fácil y la velocidad de lectura sea más rápida. Cambiar los atributos de un producto no afectará a otros.

## Perfiles de usuario

Como las bases de datos documentales tienen un esquema flexible, pueden almacenar documentos que tengan atributos y valores de datos diferentes. Las bases de datos documentales son una solución práctica para los perfiles online en los que diferentes usuarios proporcionan diferentes tipos de información. Mediante una base de datos documental, puede almacenar cada perfil de usuario de forma eficaz almacenando solo los atributos que son específicos de cada usuario.

Suponga que un usuario decide añadir o eliminar la información de su perfil. En este caso, su documento podría reemplazarse fácilmente por una versión actualizada que contuviera los atributos y datos recién añadidos u omitir todos los atributos y datos recién omitidos. Las bases de datos documentales administran fácilmente este nivel de detalle y fluidez.

## Big data en tiempo real

Históricamente, la capacidad de extraer información de datos operativos se ha visto obstaculizada por el hecho de que las bases de datos operativas y las bases de datos de análisis se mantenían en diferentes entornos: operativos y de negocio, respectivamente. Ser capaces de extraer información operativa en tiempo real es fundamental en un entorno empresarial altamente competitivo. Mediante el uso de bases de datos documentales, una empresa puede almacenar y administrar datos operativos de cualquier origen e incluir los datos de forma simultánea en el motor de BI elegido para su análisis. No es necesario tener dos entornos.

## Diferencias con otras DDBB

### Diferencias con bases de datos relacionales

Las bases de datos relacionales están diseñadas para aplicaciones de procesamiento de transacciones online (OLTP) altamente coherentes y transaccionales, y son buenas para el procesamiento analítico online (OLAP). Por el contrario, las bases de datos NoSQL están diseñadas para varios patrones de acceso a datos que incluyen aplicaciones de baja latencia. Las bases de datos de búsqueda NoSQL están diseñadas para hacer análisis sobre datos semiestructurados.

El modelo relacional normaliza los datos en tablas conformadas por filas y columnas. Un esquema define estrictamente las tablas, las filas, las columnas, los índices, las relaciones entre las tablas y otros elementos de las bases de datos. La base de datos impone la integridad referencial en las relaciones entre tablas. Mientras que las bases de datos

NoSQL proporcionan una variedad de modelos de datos, como clave-valor, documentos y gráficos, que están optimizados para el rendimiento y la escala.

## Diferencias con otras bases NoSQL

Las bases de datos documentales se caracterizan por admitir estructuras de datos complejas y flexibles contra otras bases de datos NoSQL como la de clave-valor ya que los documentos admiten muchos pares de clave-valor además de que pares clave-matriz o documentos anidados. Por otra parte, las bases de datos documentales al tener estructuras más complejas son menos performantes en algunos contextos en comparación con bases de datos NoSQL de clave-valor o de columnas. Además, las bases de datos documentales no están pensadas para crear relaciones entre los elementos de la base (en este caso, documentos), cosa que es posible en bases de datos de grafos.

## Motores más utilizados

Las bases de datos documentales son de gran importancia, en particular para el desarrollo de aplicaciones web. Debido a la demanda resultante del desarrollo web, hay ahora numerosos sistemas de gestión de bases de datos (SGBD) en el mercado. La siguiente selección enumera los más conocidos:

- BaseX: este proyecto open source utiliza Java y XML. BaseX viene con una interfaz de usuario.
- CouchDB: la Apache Software Foundation lanzó el software de código abierto CouchDB. El sistema de gestión de bases de datos está escrito en Erlang, utiliza JavaScript y se utiliza en aplicaciones de Ubuntu y Facebook, entre otras.
- Elasticsearch: este motor de búsqueda funciona con una base de datos documental. Para esto, emplea documentos JSON.
- eXist: el SGBD eXist, de código abierto, se ejecuta en una máquina virtual Java y, por lo tanto, puede utilizarse con independencia del sistema operativo. Utiliza principalmente documentos XML.
- MongoDB: MongoDB es la base de datos NoSQL más extendida del mundo. El software está escrito en C++ y emplea documentos similares a JSON.
- SimpleDB: con SimpleDB (escrito en Erlang), Amazon ha desarrollado su propio SGBD para los servicios en la nube de la compañía. El proveedor cobra una tarifa por su uso.
- DynamoDB. Azure Cosmos. RavenDB.

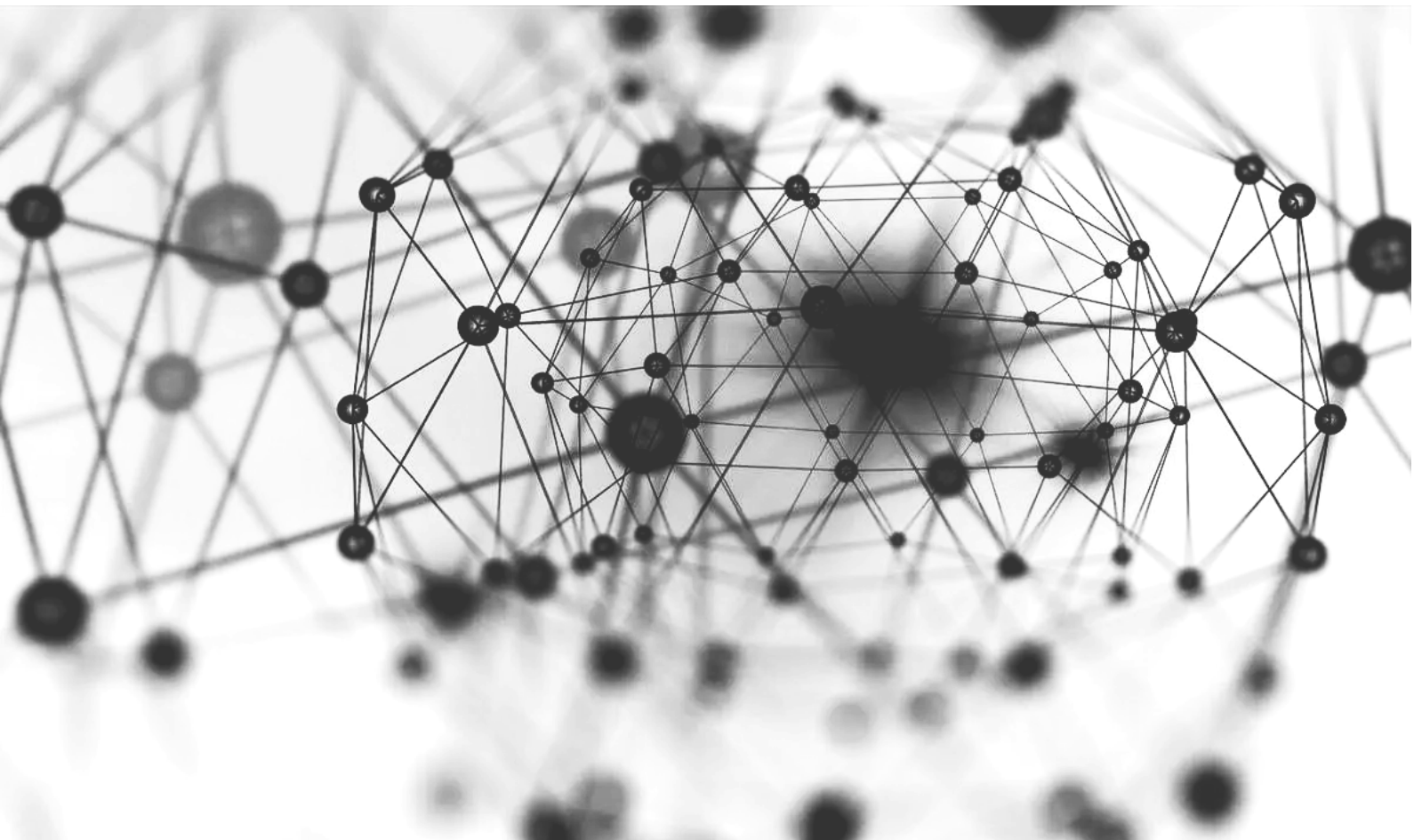


# Escalabilidad

Las bases de datos NoSQL normalmente se pueden particionar porque los patrones de acceso son escalables mediante el uso de arquitectura distribuida para aumentar el rendimiento que proporciona un rendimiento constante a una escala casi ilimitada.

## Bibliografía

- <https://www.oracle.com/ar/database/nosql/what-is-nosql/>
- <https://www.mongodb.com/es/nosql-explained>
- <https://aws.amazon.com/es/nosql/>
- <https://www.geeksforgeeks.org/document-databases-in-nosql/>
- <https://www.diegocalvo.es/caracteristicas-y-comparativa-de-las-bases-de-datos-nosql/>



LABORATORIO II - Trabajo Práctico 1 Bases de Datos NoSQL

# Base de datos orientada a **Grafos**

Profesor: Juan Lopreiato

Integrantes:

- Bautista Bullejos
- Julián Martínez Fontañá
- Carla Montani
- Axel Speroni

## Índice

o1. Introducción .....	3
o2. Definición conceptual - Noción general del tema .....	3
o3. Ventajas - Desventajas .....	5
o4. Para que se utiliza .....	6
o5. Ejemplos de uso reales .....	7
o6. Motores de ejemplo .....	9
o7. Diferencias con otras DDBB .....	11
o8. Escalabilidad .....	12
Bibliografía .....	14

## 01. Introducción.

En los últimos años la necesidad de procesar grandes volúmenes de información en poco tiempo para el uso cotidiano en las redes sociales y dispositivos móviles llevo el desarrollo de nuevas soluciones tecnológicas, que sin desafiar la importancia de las bases de datos relacionales, buscan estructurar una forma diferente de almacenar datos. Las Bases de Datos NoSQL, nombre en el cual podemos incluir las bases de datos clave-valor, las documentales, las basadas en columnas y las basadas en grafos, surgieron como una solución para los problemas no resueltos eficientemente por la bases de datos relacionales, que no logran procesar satisfactoriamente conjuntos de datos grandes y complejos, las NoSQL (Not only SQL) abarcan diferentes tipos de bases de datos, cada una con sus características intrínsecas que le confieren un comportamiento más o menos apropiados para diferentes escenarios de aplicación.

En este caso vamos a profundizar en el estudio de las características, usos, ventajas y desventajas de las Bases de datos orientadas a Grafos.

## 02. Definición conceptual - Noción general del tema.

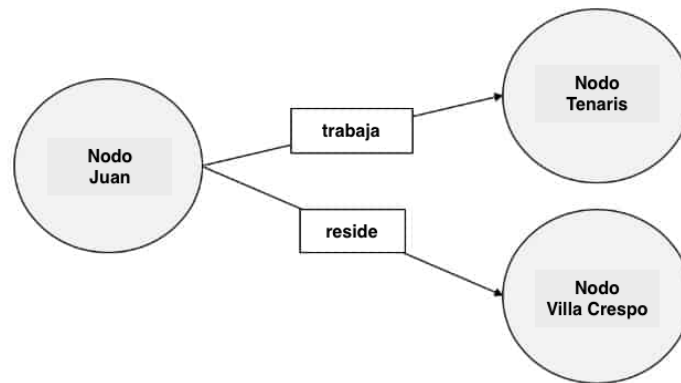
Las **Bases de datos orientadas a Grafos** (BDOG) son bases de datos que se basan en la teoría de grafos, una rama de la matemática y las ciencias de la computación que estudia las propiedades de los grafos, un **grafo** es un conjunto de objetos llamados **vértices** o **nodos** unidos por **enlaces** llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto. Los grafos, dentro del contexto de software, representan las entidades como nodos y las aristas por la forma como estos nodos se pueden encontrar relacionados, ya sean entre sí o no.

En este tipo de base de datos, los datos se muestran como entidades o nodos, los nodos pueden tener propiedades y se conectan con otros nodos por medio de relaciones o enlaces (edges) la conexión entre dos nodos son propiedades.

Los grafos son un conjunto de objetos (vértices y aristas) que permite representar datos interconectados, así como las relaciones entre ellas, de forma comprensible y como un único y más amplio conjunto de datos. Los grafos están formados por nodos o vértices, que son propiedades de datos u objetos claramente señalizados e identificables, y **aristas** o **arcos**, que representan las relaciones entre los objetos. Gráficamente, estos dos componentes tienen forma de puntos y líneas, respectivamente. Las aristas tienen un extremo inicial y uno final, mientras que cada nodo siempre contiene un número concreto de relaciones a otros nodos, ya sean de entrada o de salida.

Los nodos representan una entidad en la que almacenaremos piezas de datos o atributos de tipo clave-valor, mientras que las **aristas** representan cómo los nodos se conectan y asocian entre sí. Las relaciones también tienen nombre y dirección, de tal modo que cada relación siempre tiene un nodo de inicio y un nodo de fin.

Por ejemplo: Juan(nodo) trabaja en(enlace) Tenaris(nodo) y vive en (enlace) Villa Crespo



-

Grafos: relaciones nodos y aristas

Las relaciones también tienen nombres en función de los nodos que conectan. Esta variación en cómo funcionan las relaciones también causa que haya dos subtipos de bases de datos orientadas a grafos.

### **Labeled-Property Graph** (grafo de propiedades etiquetadas).

Se asignan propiedades (properties) concretas tanto a los nodos como a las aristas, cada nodo contiene todas sus propiedades contenidas dentro de sí mismo y las relaciones solo sirven para conectar distintos tipos de entidades. Un ejemplo simple de esto es un Nodo Empleado que tiene los atributos de un empleado como su legajo, nombre y dirección conectado al nodo Departamento, que tiene la descripción y dirección del departamento, donde la relación se llamaría “Trabaja En” y partiría desde el nodo empleado hasta el nodo Departamento, de tal modo que se lea Empleado X trabaja en Departamento Y.

### **Resource Description Framework** (grafo de descripción de recursos, RDF)

La estructura del grafo se regula mediante tripels y quads: los tripels se componen de tres elementos, siguiendo el esquema nodo-borde-nodo. Los quads complementan a los tripels con información de contexto adicional, facilitando así su separación en grupos. Los nodos optan un modelo más dinámico, donde tienen 1 o unos pocos atributos dentro de sí y cualquier otro atributo que se quiera añadir deberá ser agregado como un nodo con una relación específica para el tipo de atributo que es. Por ejemplo, si decimos que existe el nodo entidad Empleado que solo tiene el legajo del empleado, si después queremos añadir su nombre tendremos que crear el nodo de atributo Nombre y su relación sería Es Nombre, donde el nodo empleado apunta al nodo nombre.

### 03. Ventajas y Desventajas

Ventajas	Desventajas
<b>Rendimiento</b> pueden manejarse grandes cantidades de datos relacionados de forma rápida y efectiva. Las relaciones están guardadas en la propia base de datos, por lo que no son calculadas a partir de cada solicitud de búsqueda. Gracias a ello, la base de datos opera a gran velocidad incluso en búsquedas complicadas.	<b>Inexistencia de lenguaje estandarizado</b> cada plataforma de BDOG tiene un lenguaje de peticiones diferente por lo que no hay un lenguaje estandarizado para realizar peticiones a diferencia de las bases de datos relacionales con el SQL ( Structured Query Language ) Los más utilizados son PGQL, Gremlin, SPARQL, AQL, etc.
<b>Prevención de fraudes</b> gracias al análisis de las relaciones en grafos se pueden detectar de forma efectiva patrones de blanqueo y otros fraudes.	<b>Baja comunidad</b> al ser un tipo de BBDD bastante reciente, la comunidad aún no es muy grande y, por lo tanto, cuesta encontrar soporte a los distintos problemas que se pueden dar.
<b>Flexibilidad</b> La teoría de grafos en que se basan las BDOG permite solucionar múltiples problemas encontrando la solución más óptima. Facilidad para representar relaciones, dinamismo y flexibilidad para agregar nuevos datos.	<b>Ineficiencia para peticiones transaccionales</b> las bases de datos de grafos no funcionan bien para peticiones transaccionales a diferencia de las BBDD relacionales.
<b>Escalabilidad</b> las bases de datos orientadas a grafos permiten una buena escalabilidad gracias a que se pueden añadir nuevos nodos y nuevas relaciones entre ellos. Estructuras flexibles y ágiles	<b>Escalabilidad</b> puesto que están diseñadas especialmente para arquitecturas de un solo servidor, el crecimiento supone un desafío (matemático).
<b>Velocidad</b> la velocidad de búsqueda depende únicamente del número de relaciones concretas, no del conjunto de datos. Resultados en tiempo real.	<b>Altos volúmenes de datos</b> Ineficiente al manejar altos volúmenes de datos, para estos casos se debería usar con una combinación de DB Relaciones.
<b>Presentación</b> intuitiva y resumida de las relaciones.	<b>Búsqueda</b> La velocidad de búsqueda de nodos físicamente distribuidos es bastante lenta.

#### **04. Para que se utiliza**

El uso de una BDOG es una respuesta a la dificultad de representar los diversos sistemas complejos que caracterizan al mundo actual, y a la necesidad de tener un alto rendimiento de un sistema que se encuentre involucrado en contextos con altos volúmenes y concurrencia de datos.

##### **Redes sociales**

Las personas o grupos corresponden a nodos en una base de datos orientada a grafos, y las formas cómo interactúan dichos individuos generan las distintas relaciones de los mismos permitiendo así predecir sus comportamientos.

##### **Ecommerce**

Las redes neuronales de grafos, pueden ayudar también a nivel de las propuestas de ecommerce. Como ya comentamos, las recomendaciones suman un valor importante al tener contexto sobre el usuario, pero, con las redes neuronales se pueden generar predicciones en tiempo real tomando en cuenta los hábitos de compra del usuario para así poder impulsar diferentes tipo de acciones de conversión. Estos modelos de recomendación aprovechan el aprendizaje automatizado para proveer soluciones de valor. Diversos retails del mundo aprovechan estos modelos de soluciones basados en grafos para proponer de forma eficiente, nuevos productos a los usuarios.

##### **Recomendaciones**

Los algoritmos de recomendación establecen las relaciones entre individuos y los servicios a los que pueden estar sometidas las personas. Ya sea al momento de realizar una lectura de interés, la visualización de algún vídeo, las compras que realice una persona o las diversas variedades de consumo de los individuos tienden a establecer un interés en algún tema en particular generando una conducta de la cual se pueden abstraer y almacenar múltiples relaciones para su posterior recomendación.

##### **Geolocalización**

Las distintas operaciones geoespaciales dependen de estructuras de datos específicos, las cuales se pueden representar de una forma jerárquica; dicha representación facilita los cálculos de rutas o cualquier obtención de información entre las ubicaciones en alguna red específica tales como la de carreteras, ferroviaria o espacio aéreo. Las aplicaciones geoespaciales de las bases de datos orientadas a grafos son especialmente relevantes en las áreas de telecomunicaciones, logística, viajes, horarios y planificación de rutas. Las empresas que gestionan envíos les interesa calcular la ruta más rápida y eficiente



para aumentar el número de entregas por hora. Calcular la ruta más rápida puede hacerse usando el algoritmo de Dijkstra el cual se aplica en grafos.

### Controles de acceso

La autorización del uso de recursos o aplicaciones por parte de los diferentes tipos de usuarios basados en sus roles en un sistema permite que dicho flujo de información pueda ser representada mediante la utilización de grafos. La tecnología de grafos nos permite capturar en tiempo real, conexiones entre entidades de datos. También ayudan a detectar anomalías y comportamientos erróneos en los conjuntos de datos que se conectan en una red de TI para evitar de forma eficiente amenazas que pueden perjudicar la integridad de los datos. Numerosas empresas en la actualidad soportan algunos procesos de seguridad en este tipo de bases de datos, para contar con detección en tiempo real y visualización de datos para fortalecer la seguridad de sus datos.

### Política de privacidad de datos

La hiper conectividad de los últimos años, ha representado también un problema en cuanto a la privacidad de los datos. Algunas empresas deben limitar el acceso a datos claves, ya que no siempre dentro de una estructura, todos los usuarios tienen la capacidad o necesidad de acceder a todos los archivos. Los derechos de acceso suelen ser complejos de gestionar, por lo que una solución en grafos representa una estructura más fluida, escalable y confiable para lograr resultados efectivos. Un grafo puede ayudarnos a construir estructuras jerárquicas que sean dinámicas y que los diferentes usuarios puedan consultar dentro del grafo la información a la que tienen acceso, sin perder tiempo en diferentes autorizaciones.

## 05. Ejemplos de uso reales

### AirBnb

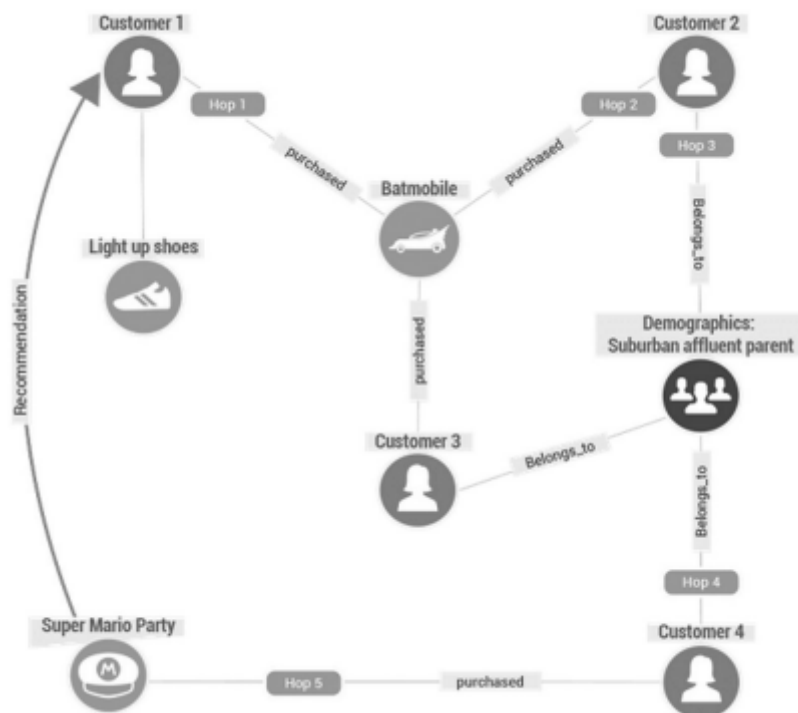
Gracias a la innovación, flexibilidad y prestación eficiente de servicios, Airbnb creció de forma muy rápida, los recursos de datos internos y externos también se incrementaron de forma exponencial y se volvieron un aspecto restrictivo e inmanejable, se determinó que los datos de Airbnb estaban siendo almacenados en ubicaciones restringidas y que carecían de un contexto adecuado para su análisis. Los datos no estaban democratizados y se hacía casi imposible contar con resultados totalmente confiables y fiables. Para eso, nació el proyecto de creación de **dataportal**.

Dataportal fue pensado para ser un motor de búsqueda de recursos de datos, donde las interacciones rápidas, detalladas y precisas contribuyen a fomentar la

exploración. Neo4j, ayuda en este caso a buscar relaciones y conexiones en cuestiones de segundos dentro de los millones de datos que integran las bases de datos de Airbnb.

### Wish

La plataforma tuvo un alto crecimiento en ecommerce a nivel mundial y quería mostrar a sus usuarios recomendaciones personalizadas, realizando consultas a la base de datos en tiempo real mostrando las correctas opciones que se adapten a sus usuarios. Mediante **TigerGraph** lograron mejorar la experiencia del usuario con recomendaciones personalizadas en menos tiempo y mejorar las ventas.



- Wish database TigerGraph

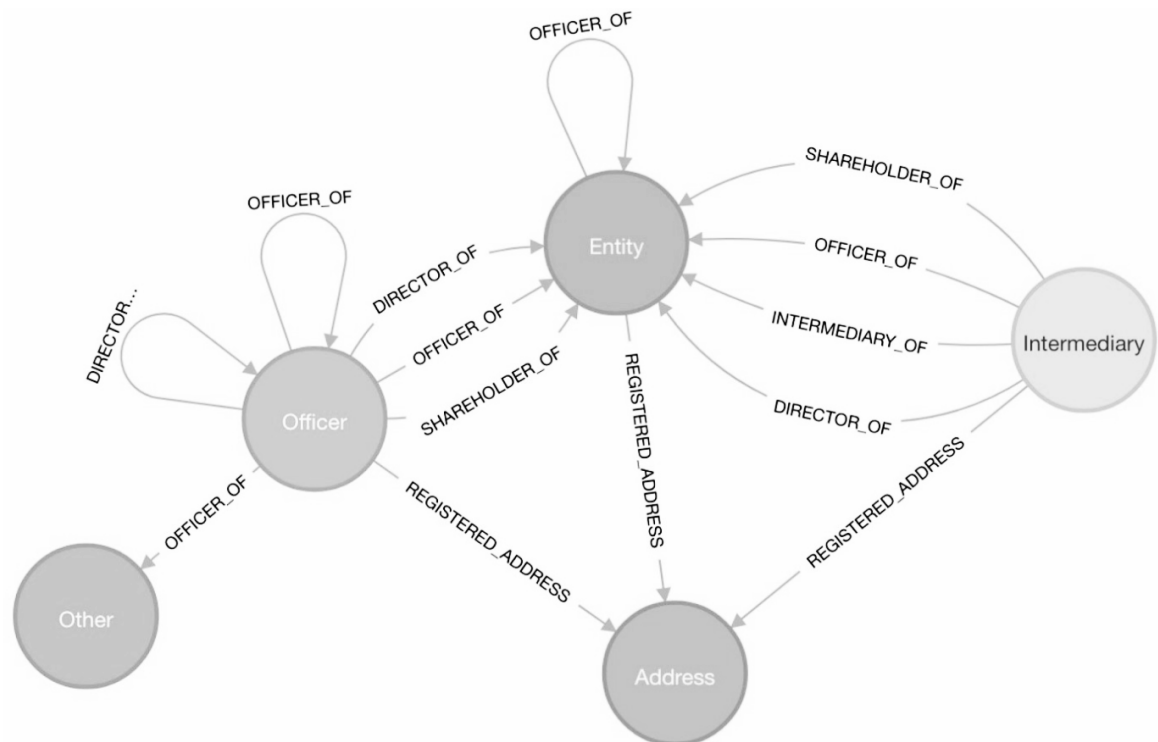
### Amazon

Amazon Neptune es un servicio de base de datos orientada a grafos completamente administrado que permite fácilmente integrar aplicaciones que funcionan con conjuntos de datos altamente conectados. Amazon Neptune es compatible con los lenguajes de consultas Apache TinkerPop Gremlin y SPARQL, los cuales permiten al usuario crear consultas para navegar por conjuntos de datos. Amazon Neptune es la solución ideal para casos de uso tales como crear motores de recomendaciones, ya que permite almacenar las relaciones que existen entre clientes y su historial de compras y así poder hacer consultas que

permitan emitir recomendaciones personalizadas. Con Amazon Neptune también se pueden crear consultas que permitan identificar patrones de fraude en transacciones financieras que contengan elementos en común con transacciones fraudulentas conocidas como correo electrónico, nombre de la persona o dirección IP. Otros casos de uso de Amazon Neptune incluyen su aplicación en redes sociales, operación de redes/T.I. y ciencias de la salud.

### Panama Papers

Un ejemplo (Detección de fraude) lo tenemos en los “Papeles de Panamá”, en el que tenemos más de 2.6 Terabytes de información, siendo complejo y costoso analizar toda esta información. Con un modelo basado en grafos, nos facilita el trabajo pudiendo seguir la pista, sobre todo si hay una representación visual de los datos, ejecutando consultas y búsquedas de los elementos, llegando a establecer las relaciones por donde haya podido circular el dinero.



-

Panama Papers database Neo4j

### o6. Motores de ejemplo (open source y/o pagos)

Existen varias plataformas, tanto de pago como gratuitas que permiten incorporar este tipo de base de datos.

### Neo4j

Esta BBDD orientada a grafos es muy popular en el mundo Big Data utilizada por empresas como **Ebay, Walmart** o **IBM**. Está pensada para manipular datos conectados de forma rápida y sencilla ayudando al análisis de la información. Tiene una gran comunidad por lo que es ideal para empezar a usar este tipo de bases de datos y está concebida como modelo de código abierto.

### ArangoDB

ArangoDB es una de las BDOG multi modelo **open source** más usadas del momento. Combina el modelo de grafos con el de clave-valor además de tener un completo motor de búsqueda de texto dando al usuario la posibilidad de combinar diferentes modelos de datos y realizar consultas a través de un lenguaje declarativo parecido a SQL.

### Amazon Neptune

Amazon Neptune es una base de datos de grafos de alto rendimiento con una baja latencia de realización de consultas. Permite almacenar información con millones de relaciones además de poder realizar predicciones de datos usando el módulo de aprendizaje autónomo Amazon Neptune ML. Puede usarse a través de la nube pública de Amazon Web Services y se abrió al público en 2018 como base de datos de alto rendimiento.

### FlockDB

FlockDB es una base de datos **open-source** orientada a grafos pensada para escalar de forma horizontal y no tanto de forma vertical. Al tener una menor capacidad que otro tipo de BBDD es usada en proyectos que requieren operaciones con una alta rapidez y baja latencia. Fue usada en sus inicios por Twitter para establecer relaciones entre los usuarios de su plataforma.

### SAP Hana Graph

SAP ha creado con SAP Hana una plataforma basada en un sistema de gestión de bases de datos relacional que se completa con el modelo integrado SAP Hana Graph, orientado a grafos.

### OrientDB

Esta graph database está considerada como uno de los modelos más rápidos disponibles actualmente.

### TigerGraph

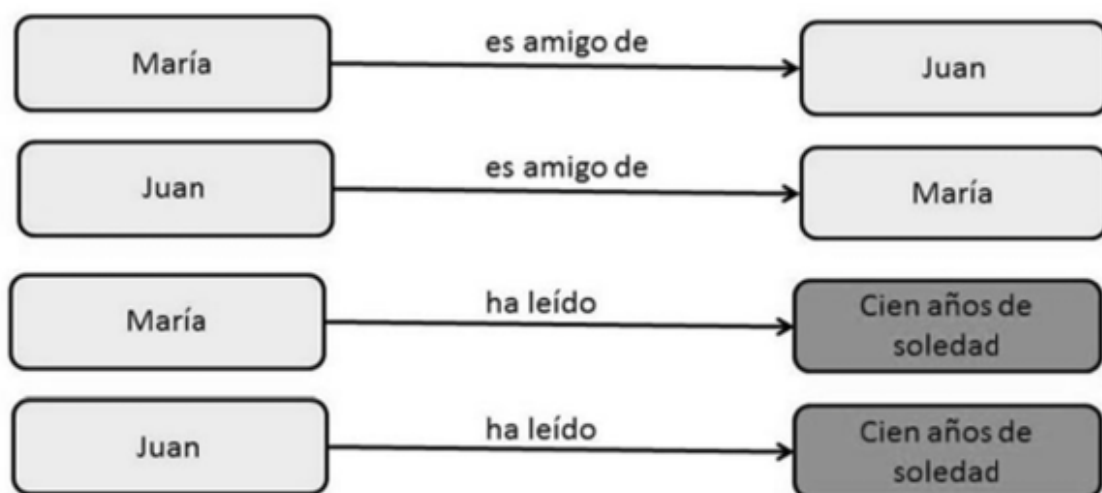
TigerGraph ofrece una amplia plataforma de gestión de bases de datos orientadas a grafos. Especialmente pensada para ser un acompañante en las soluciones

empresariales, este motor de grafos nos permite realizar análisis de conexiones entre datos a profundidad en tiempo real. Esto permite establecer mejores y más fiables modelos de toma de decisiones basadas en datos.

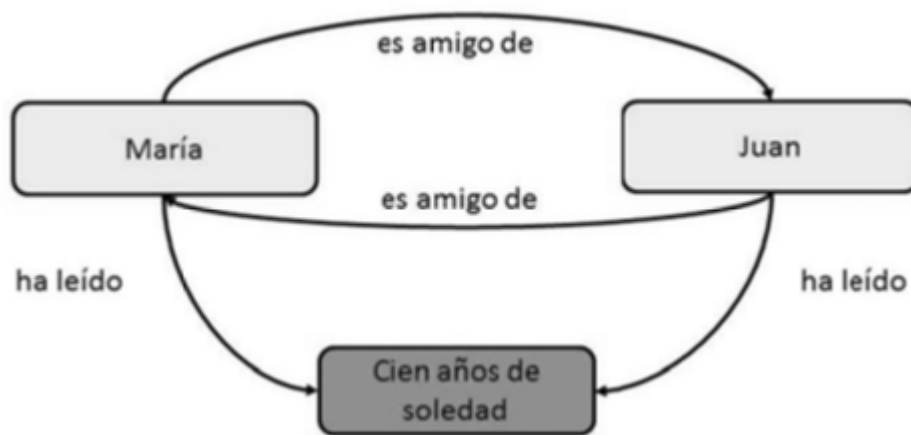
## 07. Diferencias con otras DDBB

Las bases de datos relacionales han sido las bases de datos estándar desde su nacimiento en 1970. A diferencia de las de grafos, las bases de datos relacionales tienen una estructura más compleja y orden más rígido, se basan en tablas que gestionan las relaciones de los registros de datos en cada línea. Las columnas, por su parte, contienen propiedades con diferentes valores de atributo. Además de por su estructura, las bases de datos relacionales también se diferencian esencialmente de las de grafos por su modo de funcionamiento. Para poder presentar y guardar las relaciones entre datos con interconexiones complejas, es necesario hacer cálculos con varias tablas superpuestas, lo cual suele suponer mucho tiempo y esfuerzo si se trata de un conjunto grande.

Las consultas en bases de datos relacionales normalizadas utilizando SQL (lenguaje de consulta estructurado) se realizan empleando la sentencia JOIN entre tablas, la cual une las diferentes tablas normalizadas para extraer los datos que se requieren. La dificultad se presenta si se desea obtener información cuando las entidades son altamente relacionadas con otras, pues implica construir consultas complejas difíciles de mantener y entender, que resultan muchas veces en la sobreutilización de recursos en los motores de bases de datos relacionales cuando se ejecutan dichas consultas. Para solucionar esto, los diseñadores de bases de datos deben encargarse de optimizar sus modelos y se ven obligados a adaptarlos a esta limitante, por eso los modelos de bases de datos muchas veces no representan claramente y de manera natural la lógica de negocio.



Mientras que las bases de datos basadas en tablas solo usan el lenguaje de consulta SQL (Structured Query Language), las bases de datos modernas del tipo NoSQL cada vez se alejan más de este lenguaje y del concepto relacional que va unido a él. En este enfoque también se incluyen las bases de datos de grafos como parte de la familia NoSQL. Además de las graph databases, muchos otros modelos también pertenecen a este grupo: las bases de datos key-value, las bases de datos orientadas a columnas y las bases de datos orientadas a documentos, por ejemplo. Se trata de repositorios de datos que gestionan y almacenan sobre todo conjuntos de datos estructurados, si bien menos interconectados.



-

Relaciones con Modelo de datos mediante grafos

En las BDOG los datos no se adhieren estrictamente a un tipo de dato, es decir nodos y relaciones similares y que pueden ser identificados como un tipo de datos por que comparten un conjunto de atributos, también pueden tener atributos propios y únicos respecto a otros nodos similares. La libertad en los esquemas permite de asociar la información en las diferentes formas de representarlos en nodos, propiedades o relaciones, permitiendo realizar cambios durante la evolución del modelo de la lógica de negocio, por ejemplo, una entidad asociado con un dato que puede constituir una propiedad en una entidad, también se puede modelar como una entidad completamente independiente.

## o8. Escalabilidad

La escalabilidad en bases de datos es un factor clave para quienes se desempeñan en el campo tecnológico. Para las bases de datos es uno de los procesos decisivos que inclina a

las organizaciones a elegir modelos, gestores e inclusive, sirve para determinar la cantidad de inversión que se debe realizar en equipamiento.

El concepto de escalabilidad se refiere a la capacidad que posee un software para adaptarse a las necesidades o las demandas de rendimiento a medida que el número de usuarios y operaciones crece. Generalmente al incrementarse ampliamente las transacciones en un sistema, el funcionamiento de las bases de datos empieza a verse afectado.

Sin embargo, cuando nos adentramos en el ámbito de las bases de datos orientadas a grafos, podemos encontrar un problema en la escalabilidad. Y es que su principal problema para escalar reside en su propio diseño: al estar diseñadas específicamente para un solo servidor, el crecimiento del volumen de datos acarrea un gran desafío matemático tanto para la base de datos como para quien la implementa.

### **Preguntas**

- **¿Qué es un nodo para las Bases de datos orientadas a Grafos (BDOG) ?**

1. Representan una entidad en la se almacenan piezas de datos.
2. Son una serie de puntos que podemos encontrar en una línea.
3. Son los vértices o elementos del Árbol.
4. Permiten conectar los diferentes equipos instalados en campo.

- **¿En qué caso aplicarías BDOG?**

1. Ecommerce
2. No code
3. Base de datos de un banco
4. Microemprendimiento

- **¿Cuáles son los beneficios de usar BDOG?**

1. Rendimiento manejan grandes cantidades de datos forma rápida
2. Gran soporte de de la comunidad para resolver problemas
3. Utilizan el lenguaje estandarizado SQL
4. Eficiencia para las peticiones transaccionales

- **¿Cuál empresa - institución usa BDOG?**

1. Wish
2. McDonald's
3. Anses
4. Mercado Libre

## **Bibliografía**

- <https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/graph-database/>
- <https://icc.fcen.uba.ar/bases-de-datos-en-grafos-cuando-las-apariencias-no-enganan/>
- Bases de datos orientadas a grafos. Pinilla, C.; Bello, M.; Peña, C. (2017).
- <https://abdatum.com/informatica/base-datos-orientada-grafos>
- <https://www.grapheverywhere.com/comparativa-bases-de-datos-de-grafos-cual-es-mejor-cual-elegir/>
- <https://www.grapheverywhere.com/10-casos-de-uso-reales-basados-en-tecnologia-de-grafos/>



# Laboratorio 2

**Tema:** Hibernate

**Profesor:** JUAN LOPREIATO

**Alumnos:**

Victoria Gabay  
Sebastian Javier Roca  
Tomás Stambulsky  
Gian Franco Stigliano  
Gastón Cuervo

**Cuatrimestre y año:** 2do 2022



<b>Introducción</b>	<b>2</b>
<b>¿Qué ventaja y desventaja tiene?</b>	<b>3</b>
<b>Algunos ejemplos de uso reales</b>	<b>4</b>
<b>Sintaxis y forma de uso</b>	<b>4</b>
<b>¿Qué tan escalable es hibernate?</b>	<b>7</b>

## Introducción

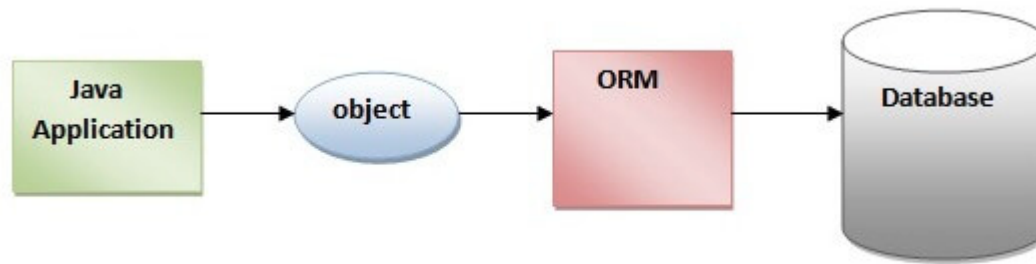


Al momento de crear una aplicación, existen varios paradigmas que se pueden utilizar, uno de ellos, es la programación orientada a objetos. Bajo este paradigma, tenemos la necesidad de crear objetos, a esto, podemos sumarle herencias e interfaces con distintos tipos de patrones si queremos obtener una arquitectura más robusta y escalable en el tiempo.

Una de las necesidades que nace al momento de construir la aplicación, es la de persistir los datos en una base, para esto, una de las herramientas que podemos utilizar en Java es JDBC. Es una API que viene incluida en la JDK de Java y nos permite realizar conexiones de bajo nivel a la base de datos. Para poder utilizarla, es necesario conocer el lenguaje SQL, ya que JDBC es meramente el nexo que permite la comunicación de nuestra aplicación y la base de datos. No nos provee de herramientas de alto nivel para realizar operaciones sobre ella, como puede ser un método para realizar una consulta a una tabla según un parámetro de entrada u otras operaciones similares.

**Hibernate ORM** (Object Relational Mapping) nace en la búsqueda de facilitar este mapeo y consultas hacia la base de datos. Es un framework que está construido sobre JDBC, es decir, lo implementa y nos abstrae de él, lo que nos permite realizar operaciones de alto nivel con facilidad. Gracias a esto, podemos realizar operaciones de manera sencilla y rápida a la base de datos.

Hibernate es una herramienta de mapeo objeto-relacional (ORM) bajo licencia GNU LGPL para Java, que facilita el mapeo de atributos en una base de datos tradicional, y el modelo de objetos de una aplicación mediante archivos declarativos o anotaciones en los beans de las entidades que permiten establecer estas relaciones. Es una herramienta con gran escalabilidad y muy eficiente que busca optimizar nuestro flujo de trabajo, agilizando la relación entre la aplicación y nuestra base de datos



Como información previa para entender qué es Hibernate, explicamos el concepto de mapeador objeto-relacional (ORM).

La programación orientada a objetos y bases de datos relacionales son dos paradigmas diferentes. El modelo relacional trata con relaciones y conjuntos. Sin embargo, el paradigma orientado a objetos trata con objetos, sus atributos y asociaciones de unos a otros.

Un **ORM** es un sistema que permite almacenar objetos de aplicaciones Java en tablas de sistemas de bases de datos relacionales usando metadatos que describen la relación entre los objetos y la base de datos, y lo hace de una manera transparente y autónoma.

Hibernate parte de una filosofía de mapear objetos Java, también conocidos como “POJOs” (Plain Old Java Objects). Con Hibernate no es necesario escribir código específico en nuestros objetos ni hacer que hereden de clases determinadas. En vez de eso trabajamos con ficheros XML y objetos que proporciona la librería. Una de las principales características de Hibernate es su flexibilidad, envolviéndolo todo bajo un marco de trabajo común.

Existen otros tipos de ORM en el mercado, uno de ellos es **Apache Cayenne** que cuenta con algunas diferencias, una de ellas es que en vez de utilizar POJOs, utiliza OO que significa Object-oriented clases. Otra de las diferencias es que Cayenne realiza una inicialización lazy de todas las relaciones por default. Cuando una de estas relaciones es necesitada y no hay una sesión a la base de datos activa, crea una nueva y realiza la consulta.

Una de las desventajas que podría verse sobre Cayenne, es que no cuenta con las especificaciones JPA (java persistence API). Hibernate hace uso de estas especificaciones, esto quiere decir que podemos cambiar la implementación que utilizamos como acceso a la base por otra que también la implemente.

## ¿Qué ventaja y desventaja tiene?

Entre las principales ventajas técnicas que aporta Hibernate están las siguientes:

1. Permite trabajar con la base de datos por medios de entidades en vez de queries.
2. Nos ofrece un paradigma orientado 100% a objetos.
3. Elimina errores en el tiempo de ejecución.

4. Mejora el mantenimiento del software.
5. Genera gran parte del SQL en tiempo de inicialización del sistema en vez de en tiempo de ejecución.

Luego la desventaja principal sería:

1. Hibernate solo impone una condición para poder utilizarse. Las claves deben tener una clave primaria.
2. No es una solución óptima para proyectos de migración de datos.
3. No ofrece toda la funcionalidad que ofrecería tirar consultas nativas.
4. Puede representar una curva de aprendizaje más grande.
5. Gran cantidad de ficheros de configuración.

## Algunos ejemplos de uso reales

Hibernate puede ser utilizado en cualquier aplicación Java que requiere mapear un conjunto de tablas SQL a un conjunto de objetos (instancias de una clase).

Hibernate debería ser el ORM elegido para desarrollar principalmente cuando se requiere de llevar un código rápidamente a producción, a parte de considerar algunos puntos extras como

- La mayor parte de las entidades del sistema requieren de un CRUD (Create, Read, Update, Delete)
- Situaciones en las que no sabemos si el día de mañana la estructura de datos va a ser modificada.
- La aplicación tiene formularios de búsqueda complejos.
- Se deben manejar grandes volúmenes de datos en caché.

En un principio, Hibernate debía ser configurado mediante un archivo xml, que especifique la tabla y la clase a mapear, ahora es posible configurarlo mediante anotaciones dentro de la misma clase.

Las anotaciones, definidas por un @ y la palabra clave pueden ir tanto sobre la clase en sí como sobre las propiedades y son acumulables. Puedo definir una clase como @Entity y asignarle la tabla a la que corresponde con la anotación @Table(name = "MiTabla").

Dentro de Hibernate están implementadas las funcionalidades de JPA (Java Persistence API), lo que permite definir las entidades, relaciones y comportamientos de las mismas dentro de nuestra aplicación. Esa implementación es la que facilita el uso de las anotaciones.

## Sintaxis y forma de uso

- **@Entity**: Le indica a JPA que esa clase puede almacenarse en una tabla.
- **@Table**: le indica a JPA a que tabla hace referencia esa entidad.
- **@Id** : Marca a ese atributo de la clase como Primary Key
- **@GeneratedValue(strategy)**: Especifica la estrategia de generación del atributo marcado como @Id.
  - AUTO: decide que generador usar según como lo haga el motor de base de datos utilizado.
  - IDENTITY: la base de datos es la encargada de determinar y asignar las PKs
  - SEQUENCE: Un objeto que puede ser utilizado como una fuente de PKs
  - TABLE: indica que se usa una tabla separada para mantener las PKs.
- **@Column**: Indica a qué columna de la tabla pertenece ese atributo.
- **@ManyToMany**: indica que ese campo va a estar dentro de una relación de muchos a muchos. Se le debe indicar la clase target de esa relación .
- **@JoinTable**: Es la anotación que indica cual es la tabla asociativa que contiene esa relación muchos a muchos.

```
@Entity
@Table(name = "characters")

public class Characters {

    @Id //SET ID AS PRIMARY KEY
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "nombre")
    private String name;

    @Column(name = "imagen");//SPECIFY COLUMN NAME
    private String imageUrl;

    @Column(name = "edad")
    private int age;
```

```
@Column(name = "peso")
private float weight;

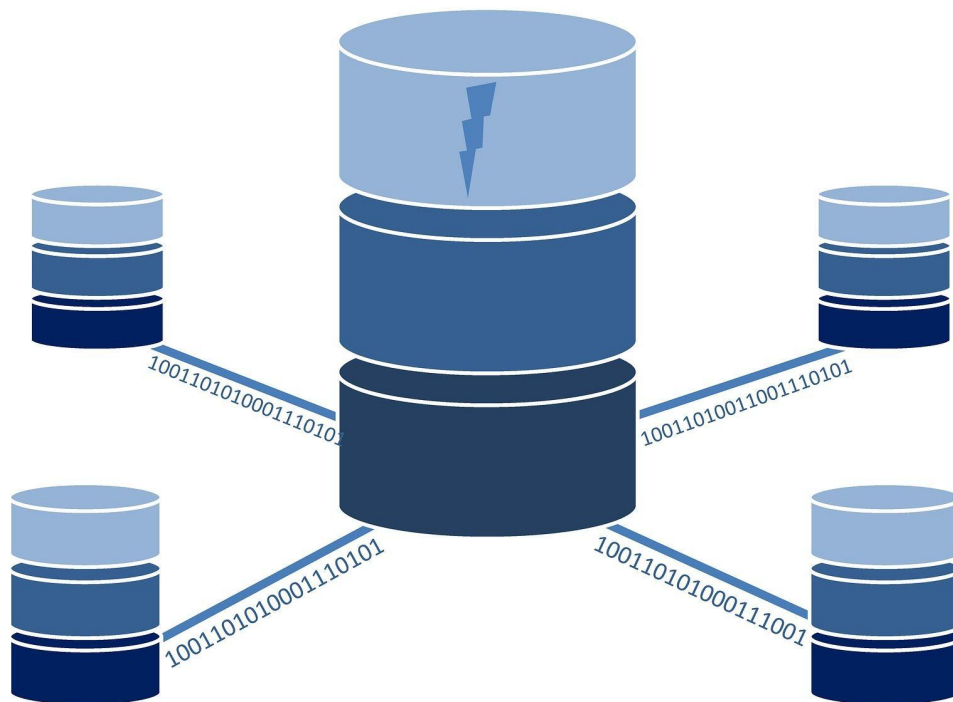
@Column(name = "historia")
private String history;
```

Si tuviera que realizar una aplicación para una compañía médica que debe permitir cosas como dar de alta usuarios (médicos, pacientes, administradores), solicitar, cancelar y modificar turnos, entre otras cosas utiliza Hibernate como ORM, además de agregar Spring JPA que implementa el patrón Repository y permite una gran agilización (a nivel de desarrollo de código) para consultas más complejas.

```
// Ejemplo de un repositorio que implementa spring JPA y que
// permite realizar una búsqueda de personajes por nombre
// simplemente poniendo el nombre del método como findByName y
// enviando nombre como parámetro.
public interface CharactersRepository extends
JpaRepository<Characters,Integer> {

    public Optional<Characters> findByName(String name);
    public List<Characters> findByAge(int age);
}
```

## ¿Qué tan escalable es hibernate?



Existe una gran variedad de bases de datos NoSQL, muchas de las cuales han surgido de las necesidades específicas de distintas compañías. Una de las características más importantes para dichas compañías suele ser lograr una buena escalabilidad. Aquí es donde entra Hibernate, dado que tiene una gran escalabilidad y resulta ser muy eficiente. Esto se debe a su arquitectura de doble capa y al hecho de que podría ser usado en un cluster.

Además, Hibernate le da mucha importancia a la concurrencia. Porque asegura la identidad sólo a nivel de unidades de trabajo en un hilo simple no requiere un bloqueo costoso ni otros medios de sincronización. Esto se termina traduciendo en una mayor facilidad para la escalabilidad.

Usualmente usar el caché ayuda mucho a aumentar el rendimiento de la base de datos, dado que al guardar las estructuras de objetos en memoria, permite un acceso más rápido a ellas. Pero todos tienen problemas que se remarcen cuanto mayor sea la concurrencia de la base de datos. Estos problemas pueden llevar a una caída del rendimiento sustancial. Hibernate le da mucha importancia a optimizar y arreglar estos temas mediante su arquitectura.





En el primer nivel del cache se guardan los objetos a recuperar de la base de datos. Además de otras ventajas más técnicas. La segunda capa del caché permite una gran configuración y seleccionar estrategias a medida para la situación.

También Hibernate da la opción de habilitar individualmente la caché que será para cada una de las entidades. Al hacer esto se debe establecer una estrategia de concurrencia, que será utilizada al sincronizar la caché de primer nivel con la segunda y esta con la base de datos. Dándole más versatilidad al poder elegir la estrategia.

## Fuentes

1. <https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/97>
2. <https://prezi.com/vwadwvpqgmak/hibernate-es-una-herramienta-de-mapeo-objeto-relacional-orm/>
3. [https://docs.jboss.org/hibernate/orm/3.5/reference/es-ES/html\\_single/](https://docs.jboss.org/hibernate/orm/3.5/reference/es-ES/html_single/)
4. <https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/97>
5. <https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/180>
6. <http://jcesarperez.blogspot.com/2008/09/cuando-usar-no-usar-hibernatede.html>
- 7.

	<p>UNIVERSIDAD DE PALERMO Laboratorio II</p>	
---	--	---

## Trabajo Práctico NoSQL

### Informe sobre Bases de Datos Clave-Valor

#### Año / Cuatrimestre

2022 - 2C

#### Curso / Código / Modalidad

Laboratorio II – 020093 – **Presencial**

#### Docentes

Juan Lopreiato

#### Grupo N.º

2

#### Alumnos

Andrés Isaac Biso - Legajo 0125044

Matias Hernan Coria - Legajo 0127111

Fidel Honorato Schinelli - Legajo 0115710

Angel Matias Bouin - Legajo 103352

Felipe Mattiazzi

Fecha de Presentación: 07/11/2022

Calificación: \_\_\_\_\_

## Índice

Índice	2
<b>Consignas</b>	<b>3</b>
<b>Introducción</b>	<b>4</b>
¿Cómo funcionan las bases de datos de clave-valor?	5
¿Cuáles son las características de una base de datos clave-valor?	6
¿Cuándo usar una base de datos de clave-valor?	7
<b>Ventajas</b>	<b>7</b>
<b>Desventajas</b>	<b>8</b>
<b>Ejemplos de uso reales</b>	<b>8</b>
<b>Usos de este tipo de Base de Datos</b>	<b>10</b>
<b>Motores de ejemplo (open source y/o pagos)</b>	<b>10</b>
<b>Diferencias con otras DDBB</b>	<b>11</b>
<b>Escalabilidad</b>	<b>12</b>
<b>Links</b>	<b>13</b>

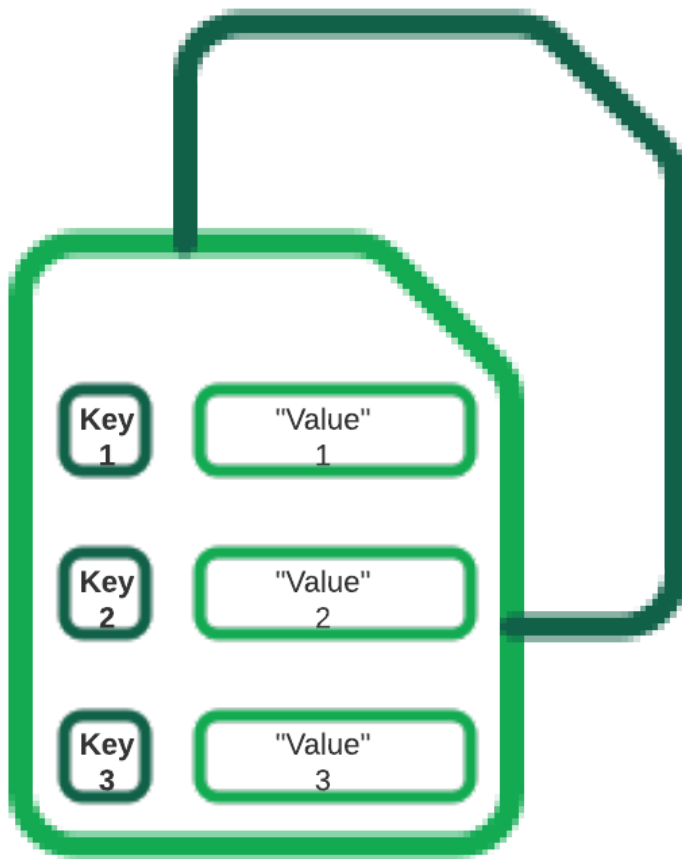
## Consignas

- Carátula
- Noción general del tema
- Ventajas
- Desventajas
- Para que se utiliza
- Ejemplos de uso reales
- Motores de ejemplo (open source y/o pagos)
- Diferencias con otras DDBB
- Escalabilidad
- Cualquier otro contenido con consideren apropiado

## Introducción

Las bases de datos de claves-valor, “key-value” - en inglés - o “store” clave-valor, son tipos de bases de datos en los que los datos se almacenan en un formato de “clave-valor” y se optimizan para leer y escribir esos datos. Los datos se obtienen mediante una o varias claves únicas para recuperar el o los valores asociados con cada clave; análogo a la estructura de datos conocida como diccionario o hash-table. Los valores almacenados pueden ser tipos de datos simples como cadenas y números u objetos complejos; y se leen y escriben en función de la clave para almacenar/recuperar su valor.

Las store de par clave-valor no son un concepto nuevo y han estado con nosotros por décadas. Uno de los stores conocidos es el antiguo Registro de Windows que permite que el sistema o las aplicaciones almacenen datos en una estructura de “clave-valor”, donde una clave se puede representar como un identificador único o una ruta única al valor.



### ¿Cómo funcionan las bases de datos de clave-valor?

Una base de datos de clave-valor asocia cada registro almacenado con una clave única que se utiliza para realizar un seguimiento del objeto. Las Bases de Datos Clave-Valor proporcionan a los desarrolladores una gran escalabilidad y flexibilidad para almacenar valores, con distintos tipos y tamaños.

En su forma más simple, un almacén de clave-valor es como un objeto de diccionario/array/map tal como existe en la mayoría de los paradigmas de programación, conformado de registros compuestos de un par clave-valor, la primera sirviendo como un identificador único para el registro, y la segunda siendo una

secuencia de bits continuos sin formato de longitud arbitraria. La diferencia siendo que las BBDD clave-valor se almacenan de forma persistente y son administradas por un Sistema de gestión de base de datos (DBMS).

Las bases de datos de clave-valor utilizan estructuras de índice compactas y eficientes para poder ubicar un valor de manera rápida y confiable por su clave, lo que las hace ideales para sistemas que necesitan poder encontrar y recuperar datos en tiempo constante. Redis, por ejemplo, es una base de datos clave-valor que está optimizada para rastrear estructuras de datos relativamente simples (tipos primitivos, listas, montones y mapas) en una base de datos persistente. Al admitir sólo una cantidad limitada de tipos de valores, Redis puede exponer una interfaz extremadamente simple para consultarlos y manipularlos, y cuando se configura de manera óptima, es capaz de un alto rendimiento.

### ¿Cuáles son las características de una base de datos clave-valor?

Una base de datos de clave-valor se define por el hecho de que permite que los programas o los usuarios de los programas recuperen datos de los registros mediante las claves que los identifican. Debido a que las bases de datos de valores clave tienen una definición muy general; no hay una lista global de características, pero hay algunas comunes:

- Recuperar un valor (si lo hay) almacenado y asociado con una clave dada
- Eliminar el valor (si lo hay) almacenado y asociado con una clave dada
- Establecer, actualizar y reemplazar el valor (si lo hay) asociado con una clave dada

Las aplicaciones modernas probablemente requerirán más que lo anterior, pero esto es lo mínimo para un almacén de clave-valor.



### ¿Cuándo usar una base de datos de clave-valor?

Hay varios casos en los que elegir un enfoque de almacenamiento de valor clave es una solución óptima:

- En caso de necesitar acceso aleatorio a datos en tiempo real, por ejemplo, atributos de sesión de usuario en una aplicación en línea, como juegos o finanzas.
- Mecanismo de almacenamiento en caché para datos de acceso frecuente o configuración basada en claves.
- Una aplicación basada en consultas simples basadas en claves.

### Ventajas

- **Esquemas flexibles:** Se pierde la estructura en tabla, pudiendo establecer diferentes niveles de jerarquía.
- **Sin esquemas predefinidos:** no es necesario predefinir el esquema de la base de datos, sino que se pueden añadir nuevos campos a posteriori.
- **Escalabilidad horizontal:** Permiten el funcionamiento en clusters.
- **Replicabilidad y alta disponibilidad:** Al funcionar en clusters es posible tener varias copias en un mismo documento en diferentes equipos de modo que si uno de los equipos falla automáticamente los otros toman el mando sin pérdida de datos ni rendimiento.
- **Particionado:** La posibilidad de particionar se basa en que un documento puede distribuirse en varios equipos, eliminando la necesidad de que un solo disco tenga la capacidad de guardar todo un conjunto de datos.

- **Velocidad de consultas:** Diferentes estructuras de datos y arquitecturas en clusters reducen los tiempos de consultas cuando la cantidad de datos que maneja es elevada.
- **Velocidad de procesamiento:** Cuando se quieren hacer operaciones sobre toda la base de datos las BD NoSQL llevan incorporados motores de procesamiento en paralelo que reducen los tiempos de lectura y escritura sobre grandes tablas.
- **Alto rendimiento y baja exigencia:** Se pueden desplegar y utilizar en equipos con recursos de HW modestos.

## Desventajas

- **Funciones de fiabilidad:** No existen las Claves Foráneas. No existe restricción referencial de que exista un registro en una tabla relacionada.  
Por esta razón, se debe implementar/solventar en el código del programa lo que agrega mayor complejidad a los sistemas y una debilidad en los esquemas de seguridad.
- **Aplicabilidad:** Relacionado con el punto anterior (funciones de fiabilidad). Limita la aplicabilidad a funciones delicadas como por ejemplo el sector bancario.
- **Incompatibilidad de consultas SQL:** En la mayoría de los casos las BBDD NoSQL son incompatibles a consultas SQL. Se debe incluir una consulta manual que puede ocasionar procesos lentos y complejos.

## Ejemplos de uso reales

- ❖ Listado de elementos más recientes.
- ❖ Sistemas de chat y mensajería.
- ❖ Cestas de la compra e-commerce.

- ❖ Almacenamiento de datos de sesión.
- ❖ Memoria cache de paginas web.
- ❖ Etcd en Kubernetes.

Para dar ejemplos concretos podemos mencionar los siguientes:

1. BioCatch es una empresa israelí de identidad digital que utiliza un seguimiento biométrico innovador para adelantarse a los estafadores en espacios de retail, gaming y finanzas digitales.

BioCatch utiliza módulos, características y varias estructuras de datos de Redis para crear una única base de datos de fuente confiable que brinda información de misión crítica en toda la organización. BioCatch captura datos de comportamiento, metadatos y API durante las sesiones de usuario activas. También crea subconjuntos de perfiles de comportamiento del usuario y perfiles predefinidos de comportamiento fraudulento.

2. El minorista de ropa Gap Inc. deseaba brindar a sus clientes de comercio electrónico información de envío en tiempo real para cada artículo que los compradores agregaron a sus carritos. La empresa enfrentó problemas con retrasos e información de inventario inexacta.

Este problema creó una mala experiencia del cliente que infló los costos y erosionó la lealtad a la marca.

Los desarrolladores de aplicaciones de Gap Inc. encontraron que la escalabilidad lineal y el rendimiento por debajo de los milisegundos de Redis Enterprise a una escala masiva fueron de gran ayuda, especialmente para los picos estacionales del Black Friday. En entornos de microservicios, los modelos de datos rápidos y flexibles protegen contra el aprovisionamiento excesivo de partes de la infraestructura que no se utilizan durante períodos más lentos.

## Usos de este tipo de Base de Datos

Se utilizan cuando se requiere una alta velocidad con un gran volumen de datos.

Son muy efectivas en la consulta y fáciles de escalar.

Estas bases de datos deben crear continuamente nuevas entradas y eliminar las antiguas.

En el ejemplo de uso real, etcd en Kubernetes, donde etcd es una base de datos de clave-valor distribuida desarrollada por el equipo de CoreOS, asegura que este replique la información a los nodos sucesores.

La coordinación del líder con los nodos del clúster a través de una base de datos etcd es especialmente conveniente en el caso de las aplicaciones distribuidas.

Kubernetes en sí es un sistema distribuido que se ejecuta en un clúster desde varios dispositivos. En consecuencia, se beneficia enormemente de una base de datos distribuida como etcd que almacena de forma segura los datos críticos

## Motores de ejemplo (open source y/o pagos)

**Amazon DynamoDB:** es de Amazon Web Services (AWS).

**Berkeley DB:** desarrollado por Oracle, ofrece interfaces para diferentes lenguajes de programación.

**Redis (Remote Dictionary Server):** Proyecto de código abierto. Constituye uno de los SGBD más utilizados y en una fase temprana fue empleado por Instagram y GitHub. Estructura master/slave.

Permite guardar información directamente en la memoria RAM. Esto acelera enormemente la respuesta por parte del servidor.

**Riak:** Riak existe en una variante de código libre, como solución empresarial y en forma de almacenamiento en la nube.

**Voldemort:** extendido SGBD, utilizado e impulsado por LinkedIn, entre otros.

### Diferencias con otras DDBB

#### Redis vs DDBB

Redis	RDBMS
Almacena los datos en la memoria primaria.	Almacena todo en la memoria secundaria.
Las operaciones de lectura y escritura son extremadamente rápidas debido al almacenamiento de datos en la memoria primaria.	Las operaciones de lectura y escritura son lentas debido al almacenamiento de datos en la memoria secundaria.
No puede almacenar archivos grandes o binarios, debido al menor tamaño de la memoria primaria aparte de ser muy costosa.	Puede almacenar archivos grandes, debido a que la memoria secundaria es de un tamaño abundante y mucho más barata por lo cual puede manejar este tipo de archivo fácilmente.
Se utiliza para almacenar una pequeña información textual a la que se debe acceder, modificar e insertar a una velocidad muy rápida, pero si intenta escribir datos masivos que superen la memoria primaria disponible comenzará a recibir errores.	Se utiliza para almacenar datos muchos más grandes que tienen un uso menos frecuente y no requieren ser rápidos.

## Escalabilidad

La escalabilidad indica la habilidad de un sistema para reaccionar y adaptarse sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida y estar preparado para hacerse más grande sin perder calidad y confiabilidad en los servicios ofrecidos.

Las bases de datos NoSQL están diseñadas para escalar usando clústeres de hardware en lugar de añadir costosos servidores y sólidos, lo cuál es muy conveniente para aplicaciones que tendrán un gran y quizás impredecible volumen de lectura y escritura en sus aplicaciones.

Tomemos por ejemplo el caso de AWS Dynamo DB, siguiendo la premisa de la escalabilidad en bases de datos NoSQL.

Dynamo fue pensado con la idea de tener una base de datos clave-valor altamente confiable y ultra-escalable.

Este es un servicio fully managed por AWS lo que significa que está construido para que el desarrollador no deba preocuparse por manejar la infraestructura de la base, y pueda concentrarse en el desarrollo de la aplicación. Esto también significa que el aprovisionamiento de los servidores, clusters y escalado será provisto de manera automática por AWS, librando de responsabilidades al desarrollador.

Este servicio está diseñado para poder escalar los recursos dedicados al mismo a cientos o incluso miles de servidores distribuidos en Availability Zones (distintas locaciones dentro de una misma región de de AWS) para satisfacer las necesidades de almacenamiento y rendimiento de la aplicación en cuestión. No hay límites predefinidos sobre la cantidad de data que cada tabla puede guardar. Los desarrolladores pueden guardar y recuperar cualquier cantidad de datos en DynamoDB y esta automáticamente se va a distribuir entre más servidores a medida que la data aumente en cantidad.

## Links

<https://www.mongodb.com/databases/key-value-database>

<https://redis.com/blog/5-industry-use-cases-for-redis-developers/>

<https://www.slideshare.net/arnabmitra/introduction-to-redis-71446732>

[http://www.thinkmind.org/download.php?articleid=immm\\_2012\\_4\\_10\\_20050](http://www.thinkmind.org/download.php?articleid=immm_2012_4_10_20050)

[https://faculty.washington.edu/wlloyd/courses/tcss562/papers/Spring2017/team7\\_NOSQL\\_DB/Survey%20on%20NoSQL%20Database.pdf](https://faculty.washington.edu/wlloyd/courses/tcss562/papers/Spring2017/team7_NOSQL_DB/Survey%20on%20NoSQL%20Database.pdf)



## Trabajo Práctico

# Laboratorio

### **Integrantes**

- Ivan Donato Ngua
- Daniel Seoane
- Lucas Bonanni
- 

### **Docente**

Juan Lopreiato

### **Fecha de entrega**

27/10/2022



# Guía de contenido

<b>Introducción</b>	<b>3</b>
<b>Ventajas</b>	<b>4</b>
<b>Desventajas</b>	<b>4</b>
<b>Para qué se utiliza</b>	<b>4</b>
<b>Ejemplos de usos reales</b>	<b>4</b>
<b>Motores de ejemplo</b>	<b>4</b>
<b>Diferencias con otras DDBB</b>	<b>5</b>
<b>Escalabilidad</b>	<b>5</b>

# Introducción

Mientras una base de datos relacional está optimizada para almacenar filas de datos, normalmente para aplicaciones transaccionales, una base de datos en columnas está optimizada para lograr una recuperación rápida de columnas de datos, normalmente en aplicaciones analíticas.

Las bases de datos columnares almacenan datos en registros de manera que puedan contener un gran número de columnas dinámicas. A diferencia de base de datos relacionales que están optimizadas para almacenar filas de datos.

Las bases de datos columnares se introdujeron por primera vez en 1970 en productos como Model 204 y ABABAS, desde 2004 han tenido una evolución constante para implementaciones comerciales. Hoy en día su desarrollo y aplicación ha resultado en una gran competencia y variedad de opciones en bases de datos columnares.

Por último, hay que destacar que desde su diseño las bases columnares están pensadas para reducir la escala de clústeres distribuidos en hardware de bajo costo. Para aumentar el desempeño. Lo que los hace una de las primeras opciones en cuanto a procesamiento de Big Data y para almacenamiento de datos.

## Ventajas

Las bases de datos basadas en columnas son creadas para la velocidad, trabajan de una forma que permite omitir los datos irrelevantes para el análisis y leer de inmediato lo que se busca. De esta manera las consultas de agregación se vuelven especialmente rápidas. Las principales ventajas son:

### Escalabilidad

La mayor ventaja que tiene este tipo de bases de datos es la escalabilidad, principalmente para almacenamiento de big data. Con la habilidad para esparcirse en múltiples nodos dependiendo de la escala de la base de datos, soporta procesamiento en paralelo, lo que significa que soporta varios procesadores para que trabajen en el mismo set de datos.

### Compresión

Generalmente, estos motores también son buenos guardando y comprimiendo estos datos para ahorrar espacio de almacenamiento. Los almacenes de columnas son muy eficientes en la compresión y/o partición de datos. La compresión permite que las operaciones en columna, como MIN, MAX, SUM, COUNT y AVG, se realicen muy rápidamente. Debido a su estructura, las bases de datos en columnas funcionan particularmente bien con las consultas de agregación (como SUM, COUNT, AVG, etc.).

### Rapidez

El tiempo de carga es mínimo, las consultas realizadas se ejecutan rápidamente, ya que están diseñadas para contener enormes cantidades de datos y que sean prácticas para el análisis de datos.

## Desventajas

### Diseño de índices para el esquema

Es muy difícil hacer el diseño efectivo de un esquema.

### Mecanismos de seguridad

No tiene mecanismos de seguridad como si tiene las bases de datos relacionales donde se puede restringir por esquemas, vistas, etc.

## Procesamiento transaccional en línea

Las bases de datos columnares no son eficientes en el procesamiento transaccional en línea, tanto como para el procesamiento de datos analíticos. Lo que quiere decir que no son muy buenos actualizando transacciones, pero están diseñados para analizarlas.

## Carga incremental de datos

Mientras las cargas incrementales no son imposibles, las bases de datos columnares no son las más eficientes para ello, porque para eso primero se tendría que escanear la columna para identificar las filas y se tiene que escanear nuevamente para buscar datos modificados o que necesitan ser muy escritas.

## Consulta específica de filas

Para poder consultar una fila específica es necesario agregar un paso extra que es identificar las filas y después leer los datos. Y lleva más tiempo obtener un dato individual dispersado en múltiples columnas comparado con acceder a registros agrupados en una columna.

## Para qué se utiliza

La arquitectura de base de datos columnar ha sido llamada el futuro de la inteligencia de negocios por Business Insider porque permite el manejo de queries instantaneos de carácter analítico que casi todos los emprendimientos dependen crucialmente a la hora de tomar decisiones de negocios. Estas bases proveen acceso sencillo a los elementos más relevantes, lo cual justamente incrementa la velocidad de un query incluso en una base conteniendo millones de registros. Se cuenta con un análisis mucho más sencillo de la data en general.

Basada en líneas y se utiliza sobre todo cuando hay que realizar muchas transacciones rápidamente, en muchos campos de aplicación (por ejemplo, pero no exclusivamente, en la investigación) los datos pasan por evaluaciones continuas. Esto es mucho más rápido con sistemas basados en columnas: la razón de esto es que se requiere acceder menos al disco duro. Los datos de una categoría se almacenan muy próximos entre sí. Si se desea leer y evaluar un registro de datos, basta con cargar un bloque; no es necesario leer la base de datos completa.

# Ejemplos de usos reales

## Netflix

Netflix es uno de los principales valedores de las bases de datos NoSQL ya que utilizan Cassandra para su capa de persistencia. Desarrollada en primera instancia por Facebook y luego continuada por la Fundación Apache, Cassandra se caracteriza por ser distribuida, escalable y ofrecer un gran rendimiento... características esenciales para aguantar las tremendas cifras de Netflix: más de 1 millón de peticiones de escritura por segundo.

## Apple

Hbase – (Maps, Siri, iAd, iCloud, and more, often related to Hadoop deployments)

Cassandra – (Maps, iAd, iCloud, iTunes, and more)

## Black rock

Para ayudar a impulsar nuestra plataforma de gestión de inversiones Aladdin. En esta charla daré una visión general de nuestro uso de Cassandra, con énfasis en cómo gestionamos la multi-tenencia en nuestra infraestructura de Cassandra.

## Spotify

Spotify utilizan Kafka para la recopilación de registros, Storm para el procesamiento de eventos en tiempo real, Crunch para la ejecución de trabajos map-reduce por lotes en Hadoop y Cassandra para almacenar los atributos del perfil del usuario y los metadatos sobre entidades como listas de reproducción, artistas, etc.

# Motores de ejemplo

- Apache Cassandra
- Redshift
- MaríaDB ColumnStore
- Apache Hadoop Hbase

## Bases de datos columnares en AWS

Amazon Web Services (AWS) proporciona una variedad de opciones de base de datos columnares para los desarrolladores. Puede operar su propio almacén de datos no relacional en columnas en la nube en Amazon EC2 y Amazon EBS, trabajar con proveedores de soluciones de AWS, o aprovechar los servicios de base de datos columnares totalmente gestionados. Amazon Redshift es un almacén de datos orientado a columnas, rápido y totalmente administrado a escala de petabytes que permite analizar todos los datos de forma sencilla y rentable utilizando las herramientas de inteligencia empresarial existentes.

Amazon Redshift consigue un almacenamiento eficiente y un rendimiento óptimo a través de una combinación de procesamiento paralelo de forma masiva, almacenamiento de datos en columnas y esquemas de codificación de compresión de datos muy específicos y eficientes.

## Bases de datos columnares en Amazon EC2 o Amazon EMR

Los desarrolladores pueden instalar las bases de datos orientadas a columnas que elijan en Amazon EC2 y Amazon EMR, lo que significa que evitan la fricción del aprovisionamiento de la infraestructura, al tiempo que les permite acceder a diferentes motores de bases de datos columnares estándar.

## Apache Cassandra

Cassandra es una base de datos orientada a columnas de código abierto diseñada para gestionar grandes cantidades de datos en muchos servidores comerciales. A diferencia de una tabla en una base de datos relacional, las diferentes filas en la misma tabla (familia de columna) no tienen que compartir el mismo conjunto de columnas.

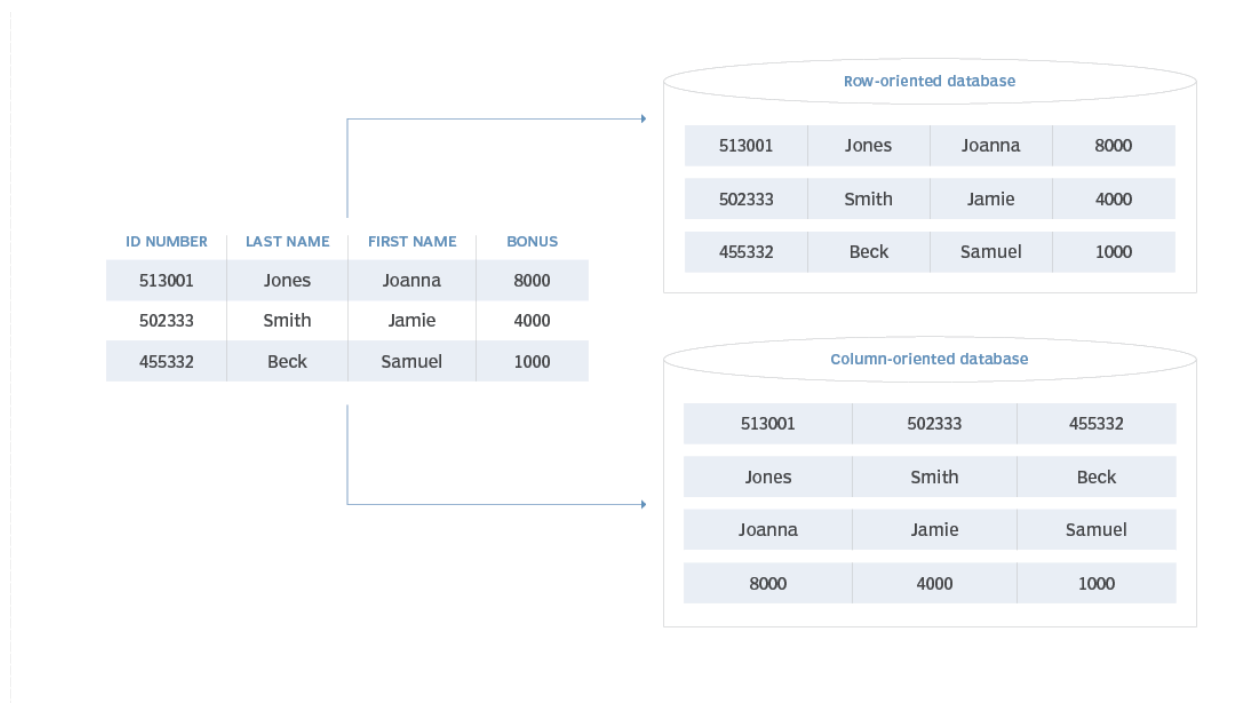
La variedad de aplicaciones para las que puede emplearse Cassandra base de datos la configuran como una opción a tener en cuenta por las organizaciones que se interesan por el internet de las cosas y las aplicaciones relacionadas, el rastreo y la monitorización de la actividad de los usuarios en su interacción con los productos o servicios, la analítica social media y los motores de recomendación. Debido a sus posibilidades, resulta muy recomendable para empresas del sector retail que cuentan con e-commerce o quieren utilizarla para sus apps o catálogos, contribuyendo a mejorar el nivel de soporte.

# Diferencias con otras DDBB

La diferencia entre una base de datos columnares y una base de datos relacional es que la base de datos relacional está optimizada para almacenar filas de datos, generalmente realizando operaciones transaccionales. En cambio, una base de datos en columnas está optimizada para lograr una recuperación rápida de columna de datos.

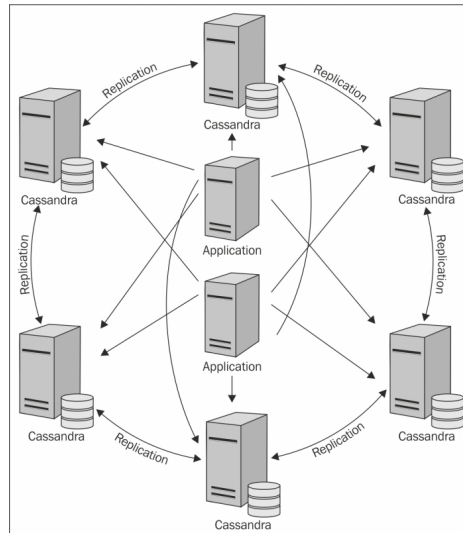
Las bases de datos tradicionales almacenan data secuencialmente de manera orientada a filas. En este caso, data similar no está junta o una al lado de la otra, incrementando el tiempo para acceder a información y a unidades de almacenamiento.

Diferentemente en las bases de datos columnares, los nombres de todos los empleados se encuentran en serie uno al lado del otro, todos los nombres en la columna "NOMBRE" y los nombres en la columna "DEPARTMENT" están almacenados uno detrás del otro. Esto simplifica el proceso de extracción de información que sea similar ya que la data guardada en la columna entera es agrupada y almacenada al mismo tiempo.



# Escalabilidad

Las bases de datos columnares están diseñadas para reducir la escala utilizando clúster distribuido de hardware de bajo costo para aumentar el desempeño. En otras palabras, están orientadas a escalar principalmente de manera horizontal que vertical.



Las siguientes imágenes son usando Casandra con múltiples volúmenes Amazon Elastic Block Store (Amazon EBS). Estos volúmenes se utilizan con instancias Amazon EC2 y ofrecen una baja latencia para trabajar con grandes volúmenes de datos.

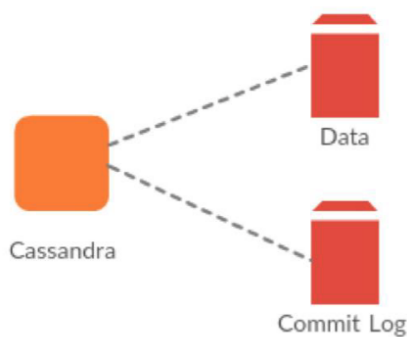


Figure 3: Single EBS Volume

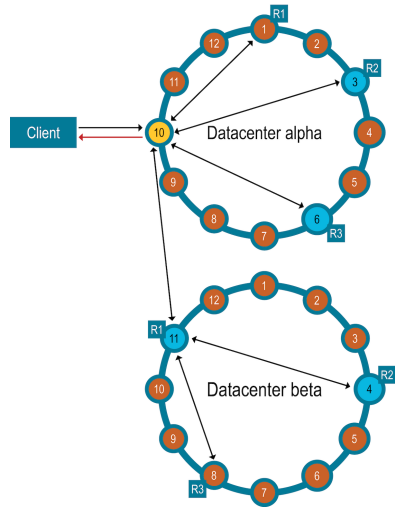


Figure 4: Multiple EBS Volumes for Data

En una estructura distribuida de Cassandra todos los nodos juegan el mismo rol ( no hay un nodo principal o máster).



Cassandra tiene nodos semillas (seed nodes) que son consultados por un nuevo nodo que quiere unirse a la red de Cassandra (una nueva instancia).  
Cada nodo se encarga de una partición de datos.  
Luego la estructura se puede expandir múltiples regiones.



# Bibliografía

Bekker, Alex. 2018. "Cassandra Performance: The Most Comprehensive Overview You'll Ever See." ScienceSoft. <https://www.scnsoft.com/blog/cassandra-performance>.

Cockcroft, Adrian, and Denis Sheahan. n.d. "Benchmarking Cassandra Scalability on AWS — Over a million writes per second | by Netflix Technology Blog." Netflix TechBlog. Accessed November 8, 2022.

<https://netflixtechblog.com/benchmarking-cassandra-scalability-on-aws-over-a-million-writes-per-second-39f45f066c9e>.

Mishra, Kinshuk, and Matt Brown. 2015. "Personalization at Spotify using Cassandra - Spotify Engineering." Spotify Engineering.

<https://engineering.atspotify.com/2015/01/personalization-at-spotify-using-cassandra/>.

"¿Qué es una base de datos columnar? – AWS." n.d. AWS. Accessed November 8, 2022.

<https://aws.amazon.com/es/nosql/columnar/>.