CURSO:
BASES DE DATOS
Unidad I: Modelamiento de datos.

Clase 3: NoSql.

Profesor: Diego Miranda

Data Scientist

INTRODUCCIÓN

Aunque las bases de datos relacionales son la herramienta más usada para manejar los datos hoy en día, esto no quiere decir que son la única solución disponible. Un objetivo de las bases de datos relacionales es permitir la modelación de cualquier caso de uso a través de tablas, lo que las hace muy general. Por otro lado, existen muchos contextos más específicos donde uno no necesita todo lo que ofrece el modelo relacional. Además, en un contexto muy especializado, utilizar una base de datos relacional puede ser menos eficiente precisamente por la generalidad del modelo relacional.

Para abordar este problema, en la primera década del siglo 21 nace el paradigma NoSQL (de non-SQL, o Not only SQL, por su sigla en inglés), que aborda varios modelos de datos que no son relacionales y que permiten manejar escenarios específicos con más eficiencia que una base de datos relacional, sin perder todas las ventajas que nos ofrece manejo estructurado de los datos. Además, las bases de datos NoSQL son generalmente muy fáciles de paralelizar, y se usan mucho para el manejo de grandes volúmenes de datos

OTROS MODELOS DE DATOS

Volviendo al ejemplo de WebShop de las clases anteriores, revisemos cómo uno implementaba un carro de compras en la página web del WebShop. Para esto, era necesario revisar la relación Productos (Código, Nombre, Precio, Stock), buscar el producto indicado por el usuario a través de su código, revisar si hay stock disponible, y en este caso, agregar el producto a carro de compra actualizando el monto total.

Pensando en esta funcionalidad, el uso de una tabla relacional no es esencial aquí. En realidad, lo que nos sirve sería un diccionario que use el código como la llave de búsqueda, y como el valor tiene (una representación de) la tupla (Nombre, Precio, Stock) que corresponde a este código. Formalmente un diccionario es simplemente una función que mapea una llave a un valor. El ejemplo de abajo nos muestra cómo ver una instancia de la tabla Productos como un diccionario.

TABLA:

Código	Nombre	Precio	Stock
123	Sony AC2020	$150,000	2
127	Panasonic KS12	$250,000	5

DICCIONARIO:

Key (llave)	Value (valor)
123	"Nombre": Sony AC2020, "precio":150,000, "stock": 2
127	"Nombre": Panasonic KS12, "precio": 250.000, "stock:" 5

Aunque usar tablas ofrece varias ventajas, si uno solamente necesita buscar un producto por su código y no necesita otras operaciones, usar un diccionario basta. Otra ventaja que ofrece un diccionario es que es fácil dividirlo en varios archivos o varias bases de datos. Esto quiere decir, si quiero pasar de un sistema centralizado donde guardo el diccionario con la información sobre los productos a un sistema distribuido, la simplicidad de un diccionario facilita mucho este proceso. El diccionario es en realidad como las empresas grandes manejan sus carros de compra en una página web. Por ejemplo, Amazon utiliza diccionarios para manejar los carros de compras de sus millones de clientes. Obviamente el diccionario no basta en ciertas aplicaciones y por lo tanto existe una secuencia de distintos modelos de datos que permiten manejar escenarios de distinta complejidad. Los modelos que no son relacionales se conocen por su nombre común: NoSQL, y en general permiten resolver ciertas limitaciones de las bases de datos relacionales.

LIMITACIONES DEL MODELO RELACIONAL

Muchas funcionalidades que ofrecen bases de datos relacionales como la posibilidad de representar y analizar datos de alta complejidad, la posibilidad de estructurar los datos según un esquema, o las garantías ACID, imponen ciertas limitaciones a su uso óptimo en ciertos escenarios. Aquí mostraremos tres limitaciones importantes de las bases de datos relacionales:

GENERALIDAD

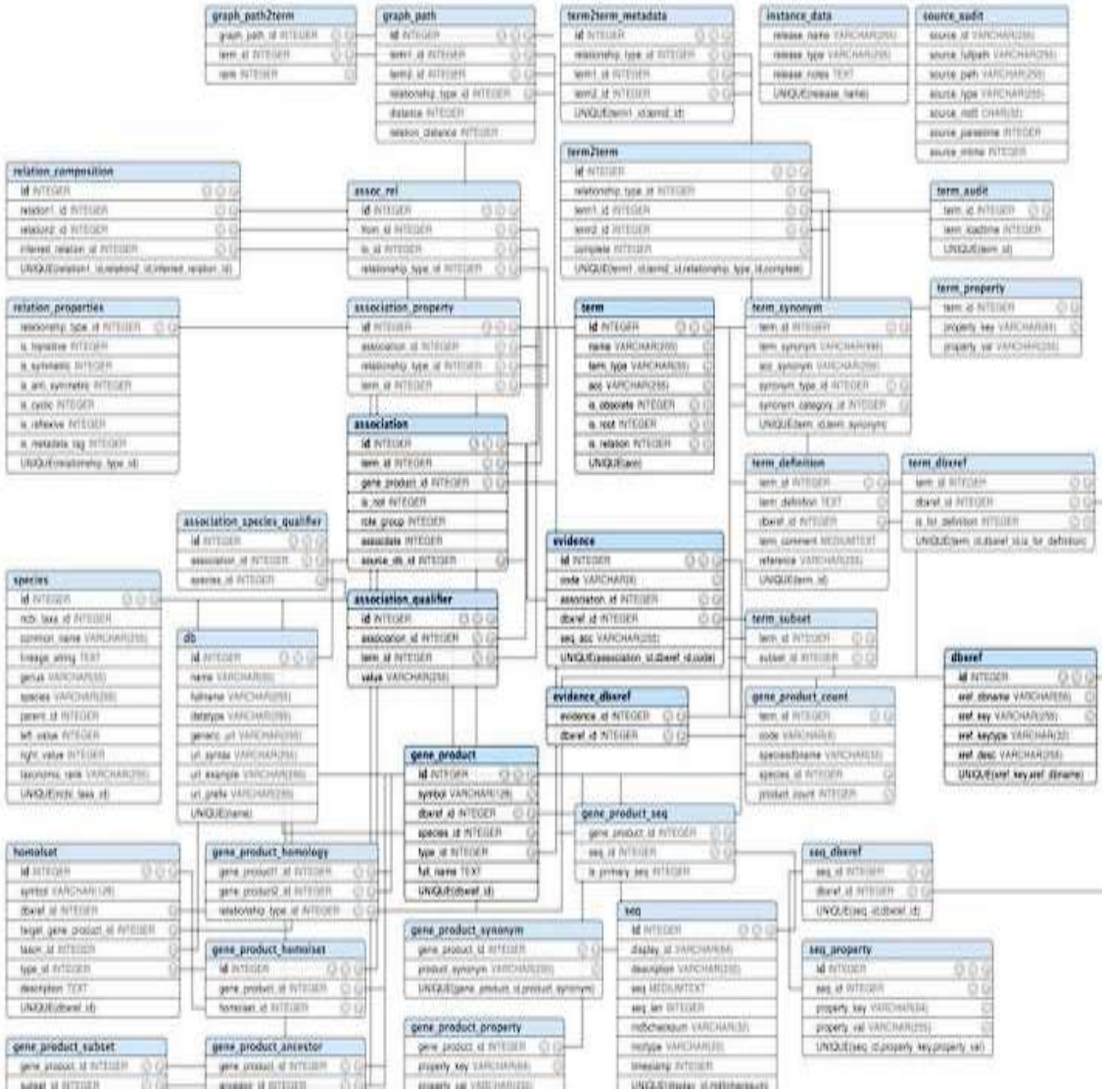
El hecho de que el modelo relacional permita modelar cualquier caso de uso y guardar datos de diferentes tipos, lo hace muy general, pero también menos especializado. Esto quiere decir que existen contextos donde los datos se pueden guardar de una maneja más simple y eficiente que usando tablas, como en nuestro ejemplo de diccionario. Similarmente, el lenguaje de consultas SQL ofrece una interfaz muy amplia para explorar los datos, pero en muchos contextos no son necesarias todas las funcionalidades que ofrece SQL. La generalidad del lenguaje de consultas SQL también lo hace menos eficiente en ciertos escenarios especializados.

SQL

no SQL

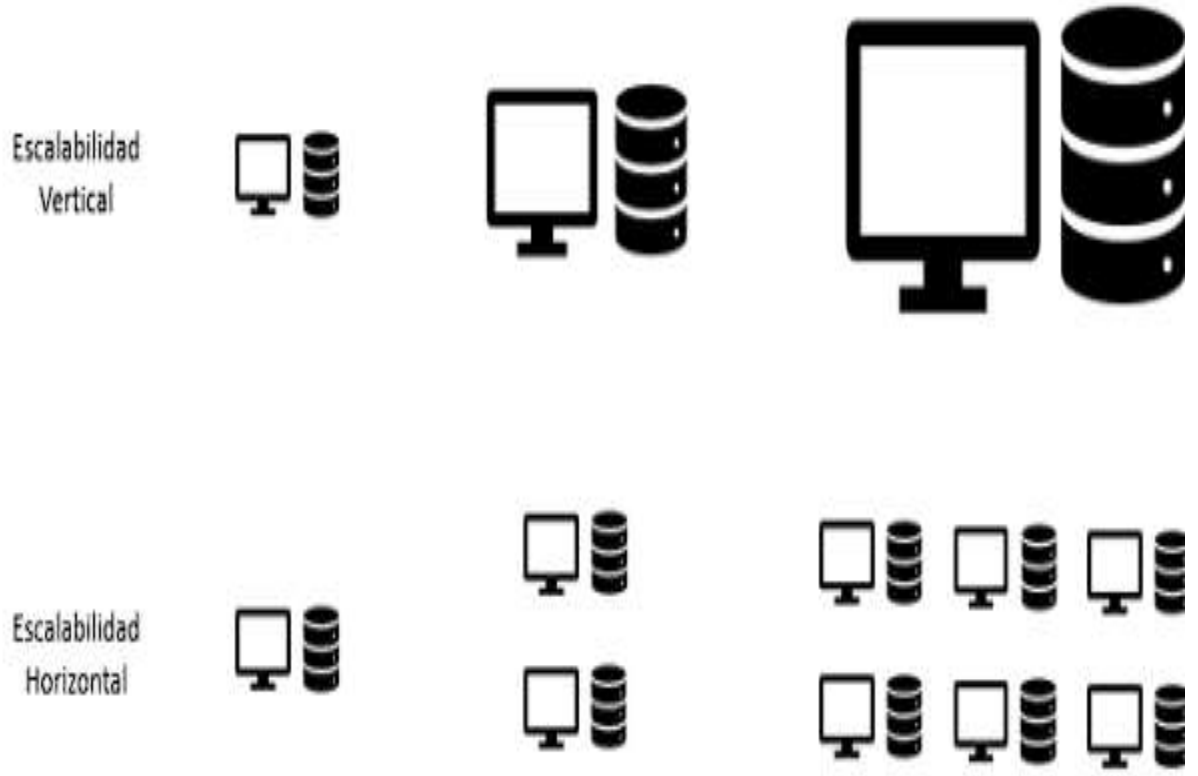
ESTRUCTURA FIJA

Las bases de datos relacionales son definidas por su esquema. El esquema nos dice precisamente qué tipo de tablas vamos a tener, qué atributos tienen estas tablas, qué tipo de valor se guarda en cada atributo, la relación entre los valores de distintas tablas, entre otros. Por un lado, tener una estructura así es muy bueno porque permite hacer ciertas cosas más eficiente. ¿Pero qué pasa cuando uno necesita cambiar la estructura de los datos, permitir atributos con distintos tipos de valores, o hacer una reorganización de las tablas? En el estricto rigor, las bases de datos relacionales permiten este tipo de cambios, pero para realizarlos se necesita un esfuerzo muy grande. Generalmente, cambiar el esquema de una base de datos relacional es un problema no trivial.



ESCALABILIDAD

Finalmente, es muy importante destacar que las bases de datos relacionales estaban pensadas principalmente como sistemas centralizados. Esto quiere decir que cuando uno necesitaba crecer en una base de datos relacional, lo que normalmente se hacía era comprar un servidor más potente (lo que se llama escalabilidad vertical). Por el otro lado, cuando se hace distribución de los datos uno quiere lograr escalabilidad horizontal, es decir, debería ser capaz de integrar más maquinas al sistema, como explicamos en la clase anterior. Aunque existe mucha investigación en hacer a los sistemas relacionales escalar horizontalmente, muchas veces esto es una tarea difícil.



Para superar a estas limitaciones, en el siglo 21 surgieron varios modelos de datos nuevos, que trataban de resolver casos más específicos, y que muchas veces pierden la generalidad de bases de datos relacionales y del lenguaje de consulta SQL. Estas tecnologías se conocen por el nombre **NoSQL**

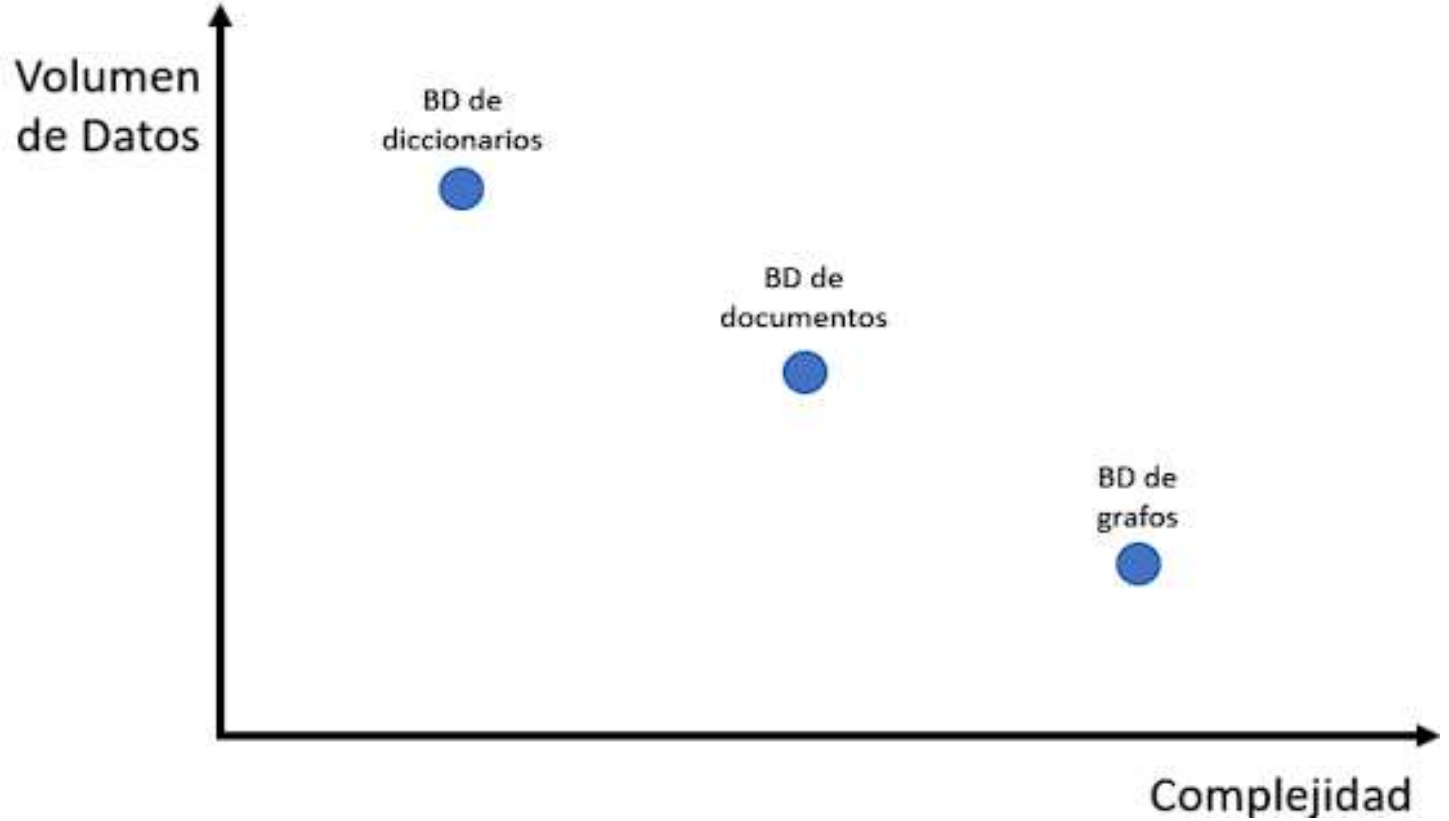
PARADIGMA NoSQL

NoSQL se refiere a cualquier modelo de datos que no es igual al modelo relacional, o que soporta un lenguaje de consultas distinto a SQL. Es importante destacar que no existe solo un modelo de datos NoSQL, sino que son varios. En particular, cada modelo de datos NoSQL está destinado a un caso de uso muy específico y funciona de manera muy eficiente en este contexto, pero quizás no es aplicable en otros contextos. La especificidad de modelos NoSQL es precisamente lo opuesto a la generalidad de bases de datos relacionales, y permite anular las limitaciones que nos impone esta generalidad. Similarmente, los lenguajes de consultas ocupados por una base de datos NoSQL son en general mucho más sencillos que SQL, que los puede hacer muy eficiente en ciertos escenarios.

En general, los modelos de datos NoSQL son mucho más flexibles que el modelo relacional, ya que nos permiten cambiar su estructura de manera más fácil. Por ejemplo, a un diccionario uno puede agregar cualquier tipo de dato, sin preocuparse de cumplir con un esquema relacional. En términos de distribución, las bases de datos NoSQL son construidas para ser escalables horizontalmente, y hacen muy fácil particionar los datos en distintos servidores, agregar nuevas máquinas al sistema distribuido, o tareas similares. El costo de esta flexibilidad es que ya no soportan las garantías ACID, sino que se basan en el paradigma BASE, garantizando consistencia eventual, que es lo suficientemente bueno en muchas aplicaciones.

Los modelos NoSQL son varios, y generalmente intentan balancear la complejidad de operaciones que soportan con el tamaño de datos que pueden manejar de manera eficiente. Por lo tanto, los modelos más sencillos soportan una cantidad más grande de los datos, pero no permiten análisis muy complejo. Por otro lado, modelos más complejos permiten hacer análisis incluso más general que en una base de datos relacional, pero no pueden procesar muchos datos.

En lo que sigue vamos a exponer tres modelos de datos NoSQL muy populares hoy en día: las bases de datos basadas en diccionarios, las bases de datos de documentos y las bases de datos de grafos. Estos modelos van desde el más sencillo hasta el más complejo, y la cantidad de datos que pueden procesar en general va en la dirección opuesta.



KEY-VALUE STORE

El modelo más simple de bases de datos NoSQL es el key-value store (base de datos de llave-valor, o base de datos de diccionario). Como ya hemos explicado antes, un diccionario es simplemente una función que vincula un elemento de su dominio (i.e. a una llave) con un valor específico. Por ejemplo, la lista de contactos que guardamos en nuestros teléfonos es un ejemplo de diccionario:

Nombre	Número
Johanna	+56 9 123 4567
Cristian	+ 56 9 111 2222

Pero un diccionario en realidad puede modelar cualquier dato complejo que se puede buscar por una llave única. Para dar un ejemplo más realista, consideremos de nuevo el caso de un servicio Web que maneja compras (como por ejemplo Amazon, o WebShop de las clases anteriores). Una aplicación fundamental para este servicio es manejar carros de compra, y esto es muy fácil de hacer a través de diccionarios. Quiere decir que cada carro de compra está identificado por un código único (CartID), y sus contenidos se pueden representar como un diccionario. Abajo tenemos un ejemplo:

CartID	Contenidos
126	usuario: Juan, contenidos: [Sony XYZ, Pan Integral]
127	usuario: Juan, contenidos: [Pan Integral, Arroz, Leche]
199	usuario: Cristina, contenidos: [Martillo, Neumáticos GGHT]

Aquí estamos asumiendo que cada carro de compra tiene un código único que sirve como su llave, y el valor de esta llave es el contenido del carro, junto con alguna información adicional (como el usuario dueño del carro de compras). Nótense que un usuario puede generar varios carros de compra.

EJEMPLO KEY-VALUE STORE

Aunque una empresa grande, como por ejemplo Amazon, necesita varias herramientas para manejar y analizar sus datos, esto no significa que debería utilizar la misma herramienta para todas las aplicaciones que se ocupan en la empresa. Por ejemplo, para manejar los carros de compras de sus millones de clientes, Amazon no necesita todo lo que ofrece el lenguaje SQL, ni tampoco las garantías del paradigma ACID. En particular, para este caso de uso Amazon aplica su propia base de datos key-value store llamada **Amazon Dynamo**.

BASES DE DATOS DE DOCUMENTOS

El segundo modelo NoSQL que exploraremos son bases de datos de documentos. Una base de datos de documentos (document database en inglés) se parece mucho a un key-value store, con la diferencia que la base de datos conoce la estructura de los documentos. Típicamente, una base de datos de documentos guarda una colección de documentos con una estructura común, y que describen ciertas entidades. Los documentos son muchas veces representados por un estándar de serialización como por ejemplo JSON. Intuitivamente, un documento JSON es simplemente un diccionario anidado; quiere decir, que valor de cada llave en nuestro diccionario puede ser un diccionario igualmente. Por ejemplo, si nuestros documentos guardan información sobre personas, se pueden ver de siguiente manera:

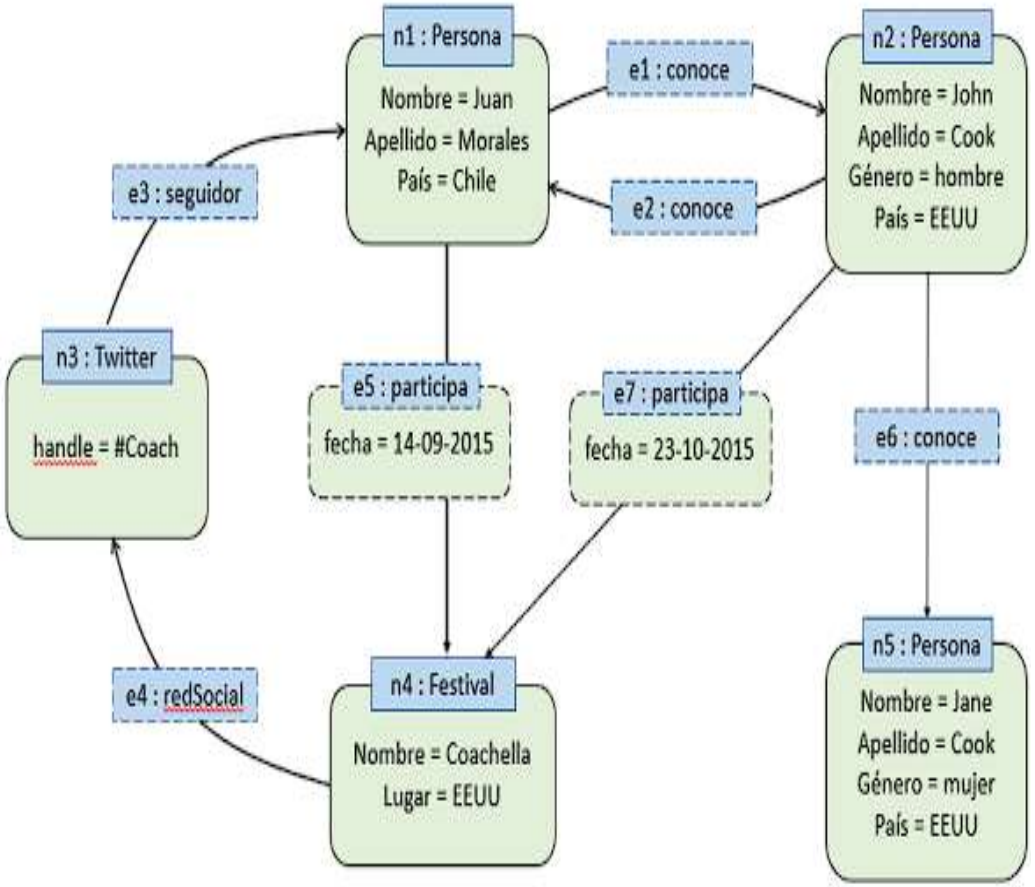
```
{
    "ID": 0xff82,
    "nombre": "Pedro",
    "apellidos": "Sanchez",
    "direccion": {
        "calle": "Av. Vicuna Mackenna 4860",
        "region": RM,
        "comuna": "Macul",
    }
}
```

EJEMPLO BD DE DOCUMENTOS

La base de datos MongoDB es una base de datos de documentos guardados en formato BSON, o Binary JSON. El formato BSON es casi idéntico a JSON, solo que permite varios tipos de datos extra, como fechas (en un formato estándar), arreglos de bytes, entre otros. MongoDB es una base de datos diseñada con la distribución en mente, y lo logra usando el proceso de fragmentación de los datos llamado sharding. En simples términos, sharding divide los datos en pedazos más pequeños, llamados shards. Cada shard se guarda en una o más máquinas separadas, distribuyendo la carga del sistema. En MongoDB el usuario puede decidir cómo se hace el sharding usando las llaves de sharding. Una llave de sharding (shard key en inglés), es simplemente un criterio sobre cómo dividir la colección de documentos en partes.

BASES DE DATOS DE GRAFOS

El último modelo NoSQL que exploraremos son las bases de datos de grafos. La idea aquí es muy simple: lo que queremos explorar no son solo los datos crudos sobre una entidad (por ejemplo, el precio de los ítems en nuestro carro de compras), sino también las conexiones entre distintas entidades en nuestra base de datos. El ejemplo canónico de una base de datos de grafos es una red social como Facebook, Twitter, o Instagram. En una red así a uno no le interesan solo los datos sobre una persona, sino que también los datos sobre sus amigos (conexiones en la red social), los grupos en las que participa, los productos que les gustan, etcétera. La gran ventaja que ofrecen las bases de datos de grafos es que son muy intuitivas. Esto quiere decir que son de cierto modo auto explicativas. Cuando uno mira una base de datos de grafos de manera inmediata puede reconocer qué tipo de entidades la base de datos representa, o que tipo de conexiones trata de modelar, sin tener que revisar un esquema como en el caso de bases de datos relacionales.



EJEMPLO BD DE GRAFOS

El sistema Neo4j soporta el modelo de grafos property graph, y permite explorar a los datos utilizando su lenguaje de consultas llamado Cypher. Cypher es un lenguaje que permite explorar las conexiones que forman distintos nodos en el grafo, y buscar a los patrones en el mismo.

```
MATCH (x :Persona) -[:conoce]-> (y :Persona)
WHERE x.nombre == "John"
RETURN y.nombre
```

CONCLUSIONES DE LA CLASE

En esta clase mostramos tecnologías para manejar los datos que nos son relacionales. En particular, hemos explicado que son bases de datos NoSQL, y mostramos tres distintos modelos de datos NoSQL: key-value stores, bases de datos de documentos, y bases de datos de grafos. En contraste con bases de datos relacionales, que ofrecen las garantías ACID, para soportar distribución de mejor manera, muchas bases de datos NoSQL adoptan el paradigma BASE, optando por accesibilidad en vez de consistencia, y usan varias técnicas para lograr consistencia eventual (eventual consistency). Por esta razón, es generalmente mucho más fácil escalar bases de datos NoSQL horizontalmente, cuando una base de datos relacional escala mejor verticalmente. Por sus características, las bases de datos NoSQL son muy útiles en algunos escenarios específicos, donde uno no necesita todas las funcionalidades de SQL. Por otro lado, bases de datos relacionales son necesarias cuando uno necesita estrictas garantías de consistencia, como, por ejemplo, en el mundo financiero.

BIBLIOGRAFÍA

- Ramakrishnan, R., Gehrke, J., Database Management Systems, 3rd edition, McGraw-Hill, 2002.