Práctica 16. Bases de datos no relacionales (NoSQL)

Apuntes de BD para DAW, DAM y ASIR

Curso 2026/2027

Índice general

1 Bases de datos NoSQL **1**

1.1 Características de las bases de datos NoSQL 2

1.2 El Teorema CAP 2

1.3 Tipos de bases de datos NoSQL 2

1.3.1 Bases de datos de tipo clave-valor 2

1.3.2 Bases de datos de tipo documentos JSON 4

1.3.3 Bases de datos de tipo grafo 6

1.3.4 Bases de datos orientadas a columnas (Wide-column stores) 8

1.4 ¿Qué base de datos NoSQL elegir? 9

1.5 Referencias 9

2 Licencia **11**

1 Bases de datos NoSQL

Las bases de datos **no relacionales** o **NoSQL** nos permiten almacenar información en situaciones donde las bases de datos relaciones pueden tener problemas de escalabilidad y rendimiento. Estas bases de datos están diseñadas para modelos de datos específicos y tienen esquemas flexibles.

Ejemplo:

Por ejemplo, suponga que queremos modelar el esquema de una base de datos sencilla para almacenar libros.

Si utilizásemos una **base de datos relacional** almacenaríamos la información en diferentes tablas, que estarían relacionadas entre sí mediante restricciones de claves primarias y ajenas. El modelo relacional está diseñado para permitir que exista integridad referencial entre tablas y para reducir la redundancia de la información mediante la normalización. En este ejemplo, tendríamos las siguientes tablas:

- **libros**(isbn, título, año_edición)
- **autores**(id_autor, nombre_autor)
- **autor_isbn**(id_autor, isbn)

En una **base de datos NoSQL**, el registro de un libro se puede almacenar en un único documento JSON. A diferencia del modelo relacional, podemos incluir información relacionada, como los autores, dentro del propio documento mediante el uso de listas u otros objetos anidados, evitando así la necesidad de realizar operaciones JOIN.

```
{
    "isbn": 9788448190330,
    "título": "Fundamentos de Bases de Datos",
    "año_edición": 2015,
    "autores": [
        {
            "id_autor": 11,
            "nombre_autor": "Abraham Silverschatz"
        },
        {
            "id_autor": 12,
            "nombre_autor": "Henry F. Korth"
        }
    ]
}
```

1.1. Características de las bases de datos NoSQL

- No cumplen con el esquema entidad–relación.
- No utilizan estructuras fijas de almacenamiento como las tablas.
- Estas bases de datos tienen esquemas de datos flexibles, lo que hace que sean ideales para aplicaciones que trabajan con datos semiestructurados y no estructurados.
- Normalmente no soportan operaciones de tipo `JOIN`.
- Tradicionalmente no garantizan las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) para favorecer la escalabilidad, aunque muchos sistemas modernos (como MongoDB o DynamoDB) ya soportan transacciones ACID multi-documento.

1.2. El Teorema CAP

Para entender por qué existen tantos tipos de bases de datos NoSQL, es fundamental conocer el **Teorema CAP**, también conocido como **Teorema de Brewer**. Este teorema afirma que en un sistema distribuido es imposible garantizar simultáneamente más de dos de las siguientes tres propiedades:

- **Consistencia (*Consistency*)**: Todos los nodos ven los mismos datos al mismo tiempo.
- **Disponibilidad (*Availability*)**: Cada petición recibe una respuesta de éxito o fallo, sin garantía de que contenga la versión más reciente.
- **Tolerancia al particionado (*Partition Tolerance*)**: El sistema continúa funcionando a pesar de la pérdida de mensajes o el fallo de parte del sistema.

Dado que en un sistema distribuido la **Tolerancia al particionado** es obligatoria para evitar caídas totales, las bases de datos deben elegir entre ser **CP** (Consistentes y Tolerantes al particionado) o **AP** (Disponibles y Tolerantes al particionado).

1.3. Tipos de bases de datos NoSQL

1.3.1. Bases de datos de tipo clave-valor

Las bases de datos clave-valor son altamente divisibles y permiten un escalado horizontal a escalas que otros tipos de bases de datos no pueden alcanzar.

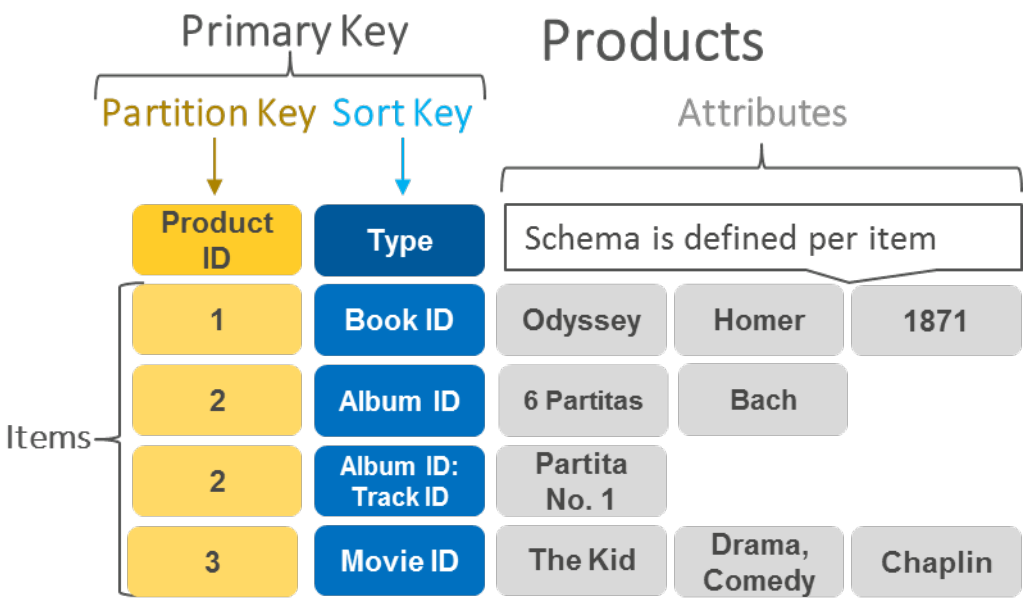

Imagen: Ejemplo de una base de datos clave-valor. Amazon Web Services.

Casos de uso

- Almacén de sesiones.
- Carrito de la compra.
- Juegos.
- IoT.

Ejemplos de bases de datos de tipo clave-valor

- Redis.
- Amazon DynamoDB.
- Riak.

Tutoriales

- Getting started with redis
- Learning Redis
- Quickstart: How to Use Redis on Java. 2019. DZone.
- A Docker/docker-compose setup with Redis and Node/Express. 2018.

Práctica con Redis

- Servidor:

Iniciamos un contenedor Docker con `redis` en el puerto 6379.

```
1  docker run --rm --name redis-server -p 6379:6379 redis
```

- Cliente:

Iniciamos un contenedor Docker con `redis-cli` que conecta con el servidor `redis` que hemos creado en el paso anterior. Utilizamos el parámetro `--raw` para que los caracteres especiales como acentos o signos de exclamación se muestren correctamente.

```
docker run -it --rm --link redis-server:redis-server redis redis-cli -h redis-
    server -p 6379 --raw
```

Para almacenar valores utilizamos `SET`.

```
redis-server:6379> SET miclave "¡Hola mundo!"
OK
```

Para recuperar valores utilizamos `GET`.

```
redis-server:6379> GET miclave
"\xc2\xa1Hola mundo!"
```

Casos de uso de Redis

- Top 5 Redis Use Cases. ByteByteGo.

1.3.2. Bases de datos de tipo documentos JSON

En algunas aplicaciones, los datos se representan como un objeto o un documento de tipo `JSON` porque es un modelo de datos intuitivo para los desarrolladores. Las bases de datos de `JSON` tienen una naturaleza flexible y facilitan a los desarrolladores el almacenamiento y la consulta de datos. Internamente, sistemas como **MongoDB** utilizan un formato binario llamado **BSON** (Binary JSON) que permite almacenar más tipos de datos (como fechas o datos binarios) y realizar búsquedas más eficientes.

Ejemplo

Ejemplo de un documento `JSON`que describe una película.

```
{
    "year" : 2019,
    "title" : "Avengers: Endgame",
    "info" : {
        "directors" : [ "Anthony Russo", "Joe Russo"],
        "release_date" : "2019-04-25T00:00:00Z",
        "rating" : 8.8,
        "genres" : ["Action", "Adventure", "Sci-Fi"],
        "image_url" : "https://your.server.com/images/avengers.jpg",
        "plot" : "After the devastating events of Vengadores: Infinity War
            (2018), the universe is in ruins. With the help of remaining allies,
             the Avengers assemble once more in order to undo Thanos' actions
            and restore order to the universe.",
        "actors" : ["Robert Downey Jr.", " Chris Evans", "Mark Ruffalo"]
    }
}
```

Casos de uso

- Administración de contenidos, como blogs y plataformas de video.

- Catálogos, como los productos de una aplicación de e-commerce.

Ejemplos de bases de datos comerciales de tipo documentos `JSON`

- MongoDB.
- Couchbase.
- Amazon DocumentDB.

Tutoriales

- MongoDB Hello World—How to connect, insert and find data. 2018. Medium.

Práctica con MongoDB

- Servidor:

Iniciamos un contenedor Docker con `mongo`.

```
docker run -d --rm --name mongo-server mongo
```

- Cliente:

Iniciamos un contenedor Docker con un cliente de `mongo` que conecta con el servidor `mongo` que hemos creado en el paso anterior.

```
docker run -it --rm --link mongo-server mongo mongosh --host mongo-server
```

Una vez que hemos conectado con el servidor mongo podemos ejecutar los siguientes comandos. En primer lugar, listamos todas las bases de datos que existen en el servidor.

```
show databases
```

Para seleccionar una base de datos existente o para crear una nueva base de datos utilizamos el comando `use`. Por ejemplo, el siguiente comando creará una base de datos llamada `tienda`.

```
use tienda
```

Para listar las colecciones de documentos que existen en la base de datos utilizamos el comando `show collections`.

```
show collections
```

Como acabamos de crear la base de datos, el comando anterior no devolverá ningún resultado porque todavía no existe ninguna colección.

Vamos a crear una colección llamada `libros`.

```
db.createCollection("libros")
```

Ahora insertamos el primer documento JSON en la colección `libros`.

```
db.libros.insertOne({titulo: "Bases de datos NoSQL", autor: "Pepe"})
```

También es posible insertar una lista de documentos JSON en una única operación.

```
db.libros.insertMany([{titulo: "Libro A", autor: "María"},{titulo: "Libro B",
    autor: "Juan"}])
```

Para obtener la lista de documentos de la base de datos utilizamos el comando `find`.

```
db.libros.find()
```

Podemos formatear la salida con el comando `pretty`, para mejorar la legibilidad de los documentos JSON (aunque en las versiones modernas de `mongosh` la salida ya aparece formateada por defecto).

```
db.libros.find().pretty()
```

Si quisiéramos realizar una búsqueda sobre algún autor en concreto podemos ejecutar el siguiente comando.

```
db.libros.find({autor: "Pepe"})
```

1.3.3. Bases de datos de tipo grafo

Las bases de datos no relacionales basadas en grafos utilizan **nodos** para almacenar las entidades de datos y **aristas** para almacenar las relaciones entre las entidades. Una arista siempre tiene un nodo de inicio, un nodo final, un tipo y una dirección, y puede describir relaciones principales y secundarias, las acciones, la propiedad y cosas similares. No hay límite para la cantidad y el tipo de relaciones que puede tener un nodo.

Ejemplo

Este ejemplo muestra cómo sería el grafo de una red social. Las personas serían los nodos y sus relaciones son las aristas. De este modo es posible conocer quiénes son los amigos de los amigos de una persona específica.

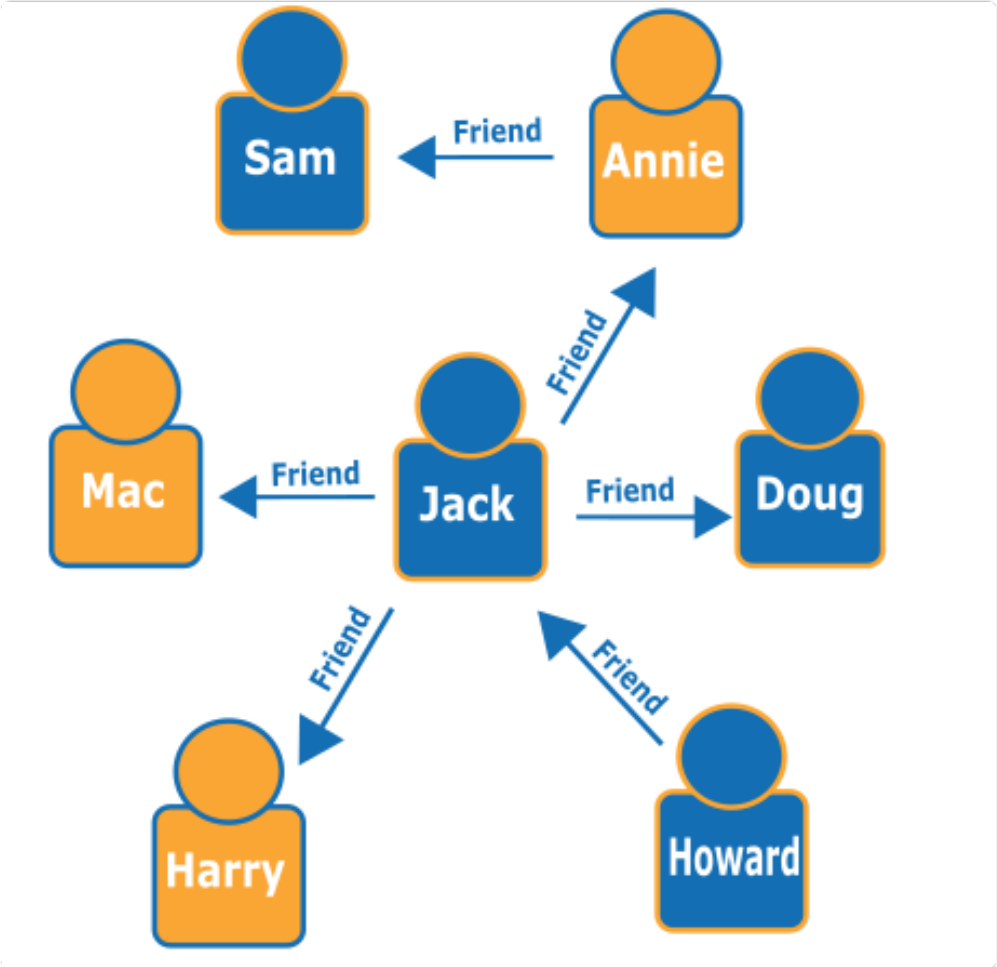

Imagen: Ejemplo de una base de datos basada en grafos. Amazon Web Services.

Casos de uso

- Detección de fraudes.
- Motores de recomendaciones.

Ejemplos de bases de datos de tipo grafo

- Neo4j.
- InfinteGraph.

Práctica con Neo4j

- Servidor:

Iniciamos un contenedor Docker con `neo4j`. El puerto `7474` es para la interfaz web y el `7687` para el protocolo Bolt, que es un protocolo binario de comunicación entre el cliente y el servidor.

```
docker run -d --rm --name neo4j-server -p 7474:7474 -p 7687:7687 -e NEO4J_AUTH=
    none neo4j
```

- Cliente:

Podemos usar la interfaz web en `http://localhost:7474` o conectarnos vía consola con `cypher-shell`.

```
docker exec -it neo4j-server cypher-shell
```

Una vez dentro (o en la web), utilizamos el lenguaje **Cypher** para crear nodos y relaciones.

```
// Crear dos nodos de tipo Persona
CREATE (p1:Persona {nombre: "Pepe", edad: 25})
CREATE (p2:Persona {nombre: "María", edad: 30})

// Crear una relación de amistad entre ellos
MATCH (a:Persona {nombre: "Pepe"}), (b:Persona {nombre: "María"})
CREATE (a)-[:AMIGO_DE]->(b)

// Consultar los amigos de Pepe
MATCH (p:Persona {nombre: "Pepe"})-[:AMIGO_DE]->(amigo)
RETURN amigo.nombre
```

1.3.4. Bases de datos orientadas a columnas (Wide-column stores)

Las bases de datos orientadas a columnas están optimizadas para obtener columnas de datos de una forma muy rápida y eficiente. Han sido diseñadas para gestionar grandes volúmenes de datos en clústeres distribuidos, lo que las hace ideales para el procesamiento de Big Data.

Ejemplos de bases de datos orientadas a columnas

- Apache Cassandra.
- HBase.

Referencias

- Sistema gestor de base de datos orientado a columnas.
- Base de datos columnar.

Práctica con Cassandra

- Servidor:

Iniciamos un contenedor Docker con `cassandra`.

```
docker run -d --rm --name cassandra-server cassandra
```

- Cliente:

Iniciamos un contenedor con el cliente `cqlsh` que se conecta al servidor.

Nota: Cassandra puede tardar entre 30 y 60 segundos en arrancar completamente. Si al ejecutar el siguiente comando recibes un error de tipo `Connection refused`, espera unos instantes y vuelve a intentarlo.

```
docker exec -it cassandra-server cqlsh
```

Una vez dentro, podemos crear un `KEYSPACE` y una tabla.

Un `KEYSPACE` es el equivalente a una base de datos o esquema en el mundo de las bases de datos relacionales. Es un espacio de nombres que agrupa tablas relacionadas entre sí.

```
CREATE KEYSPACE tienda WITH replication = {'class': 'SimpleStrategy', '
    replication_factor': 1};
```

El parámetro `replication` indica cómo se replicarán los datos entre los nodos del clúster. En este caso, `SimpleStrategy` es una estrategia simple que replica los datos en un único nodo (`replication_factor : 1`).

```
USE tienda;
CREATE TABLE usuarios (id int PRIMARY KEY, nombre text, email text);
INSERT INTO usuarios (id, nombre, email) VALUES (1, 'Pepe', 'pepe@gmail.com');
SELECT * FROM usuarios;
```

1.4. ¿Qué base de datos NoSQL elegir?

Tipo	Cuándo usarla	Ejemplos
Clave-valor	Almacenamiento simple, caché, sesiones, carritos de compra.	Redis, Riak, DynamoDB
Documentos	Datos semiestructurados, catálogos, perfiles de usuario, blogs.	MongoDB, CouchDB
Grafos	Relaciones complejas, redes sociales, recomendaciones, fraude.	Neo4j, JanusGraph
Wide-column	Escrituras masivas, series temporales, registros de actividad (logs).	Cassandra, HBase

1.5. Referencias

El contenido de esta web ha sido extraído de las siguientes referencias:

- NoSQL.
- Bases de datos NoSQL. Qué son y tipos que nos podemos encontrar.
- ¿Qué son las bases de datos NoSQL?. Amazon Web Services.

2 Licencia

Esta página forma parte del curso Bases de Datos de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.