NoSQL

Gestión y Arquitectura de Datos, Universidad de San Andrés

Si encuentran algún error en el documento o hay alguna duda, mandenmé un mail a rodriguezf@udesa.edu.ar y lo revisamos.

1. ¿Por qué surge NoSQL?

Las bases de datos NoSQL (Not Only SQL) surgieron como respuesta a limitaciones específicas de las bases de datos relacionales:

- **Escalabilidad**: Las bases SQL tradicionales escalan verticalmente (más recursos en un servidor), mientras que NoSQL está diseñado para escalar horizontalmente (más servidores)
- **Flexibilidad**: Los esquemas rígidos de SQL dificultan cambios en la estructura de datos
- **Big Data**: Necesidad de manejar grandes volúmenes de datos no estructurados
- **Velocidad**: Requerimientos de alta velocidad en escritura y lectura
- **Disponibilidad**: Necesidad de sistemas distribuidos con alta disponibilidad

2. Tipos de Bases de Datos NoSQL

Vamos a ver cuatro tipos de bases de datos NoSQL: Documentales, Clave-Valor, Columnares y de Grafos. Cada una tiene sus ventajas y desventajas, y se usan en diferentes casos de uso. El objetivo es entender cuándo usar cada una.

2.1. Bases de Datos Documentales

Almacenan datos en documentos similares a JSON, o un diccionario. Las ventajas de las bases de datos documentas son la flexibilidad, la capacidad de almacenar datos anidados y no estructurados. Tampoco necesitamos definir

un esquema de antemano, lo que nos permite cambiar la estructura de los datos fácilmente. Además, son ligeras y fáciles de usar. Por ejemplo:

```
{
  "id": "1",
  "cliente": {
    "nombre": "Juan Garcia",
    "email": ["juan@email.com", "juan.g@email.com"],
    "direcciones": [
      {
        "tipo": "casa",
        "calle": "Av. Libertador 1234"
      }
    ]
  },
  "pedidos": [
    {
      "fecha": "2024-01-01",
      "total": 1500
    }
  ]
}
```

2.1.1. MongoDB

MongoDB es una base de datos documental que almacena datos en documentos JSON. Fue creada en 2009 por MongoDB Inc. Su objetivo principal es ofrecer flexibilidad y escalabilidad.

2.2. Bases de Datos Clave-Valor

Las bases de datos clave-valor almacenan datos en pares de clave-valor, similar a un diccionario. Las ventajas de las bases de datos clave-valor son la velocidad, la simplicidad y la escalabilidad horizontal. Además, son ideales para cachés. Por ejemplo:

```
SET usuario:1 "Juan Garcia"
SET usuario:1:email "juan@email.com"
SET carrito:1 "['producto1', 'producto2']"
```

2.2.1. Redis

Redis es una base de datos clave-valor que almacena datos en memoria. Fue creada en 2009 por Salvatore Sanfilippo. Su objetivo principal es ofrecer alta velocidad y escalabilidad.

2.3. Bases de Datos Columnares

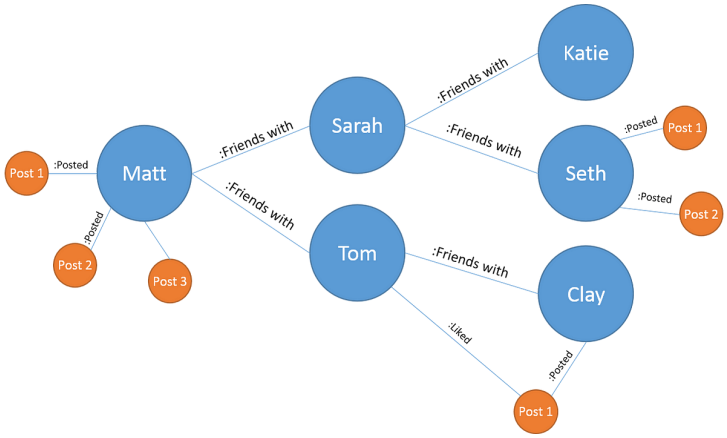
Las bases de datos columnares almacenan datos en columnas, similar a una tabla de una base de datos relacional. Las ventajas de las bases de datos columnares son la eficiencia, la capacidad de analizar datos y la escalabilidad horizontal. Además, son ideales para análisis de datos. En las bases de datos columnares, cada columna se guarda por separado y se distribuye independientemente en diferentes nodos. Esto permite una mejor compresión y consultas rápidas.

2.3.1. Cassandra

Cassandra es una base de datos columnar que almacena datos en columnas. Fue creada en 2008 por Apache Software Foundation. Su objetivo principal es ofrecer escalabilidad y alta disponibilidad.

2.4. Bases de Datos de Grafos

Las bases de datos de grafos almacenan datos en grafos, que son estructuras de datos que representan relaciones entre entidades. Las ventajas de las bases de datos de grafos son la capacidad de representar relaciones complejas y la eficiencia en consultas. Además, son ideales para redes sociales y sistemas de recomendación. Un ejemplo de grafo sería algo como esto:



2.4.1. Neo4j

Neo4j es una base de datos de grafos que almacena datos en grafos. Fue creada en 2007 por Neo Technology. Su objetivo principal es ofrecer alta velocidad y escalabilidad.

3. Teorema CAP

El teorema CAP establece que un sistema distribuido no puede garantizar simultáneamente:

- **Consistency** (Consistencia): Todos los nodos ven los mismos datos al mismo tiempo
- **Availability** (Disponibilidad): Cada petición recibe una respuesta
- **Partition Tolerance** (Tolerancia a particiones): El sistema continúa funcionando a pesar de fallos en la red

Base de Datos	Prioriza	Sacrifica
MongoDB	CP	Availability
Cassandra	AP	Consistency
Redis	CP	Availability
Neo4j	CA	Partition Tolerance

Nota: MongoDB es configurable para priorizar Consistency o Availability (modo CP o modo AP, por defecto es CP). Neo4J no tolera particiones.

4. Cuándo Usar NoSQL

- **Datos no estructurados**: Cuando los datos no tienen un esquema fijo
- **Escalabilidad horizontal**: Necesidad de escalar añadiendo servidores
- **Alta velocidad**: Requerimientos de lectura/escritura muy rápidos
- **Agilidad**: Desarrollo rápido con esquemas flexibles
- **Grandes volúmenes**: Manejo de Big Data

5. Cuándo NO Usar NoSQL

- **Datos altamente relacionales**: Muchas relaciones complejas entre entidades
- **Transacciones ACID**: Necesidad de garantías ACID estrictas
- **Reportes complejos**: Consultas complejas con múltiples JOINs
- **Equipos sin experiencia**: Curva de aprendizaje significativa