Optimización y persistencia para el almacenamiento de datos

Tema 4. Tecnologías de Persistencia de Datos NoSQL

Departamento de Sistemas Informáticos

E.T.S.I. de Sistemas Informáticos - UPM

¿Qué es una base de datos NoSQL?

`NoSQL` *es un acrónimo de* ***"Not Only SQL"****, se refiere a un tipo de base de datos que* ***no utiliza el modelo relacional*** *(tablas y filas) para almacenar datos. En su lugar, utiliza* ***estructuras de datos*** *más flexibles como* ***documentos****,* ***claves-valores****,* ***grafos****,* ***columnas****, entre otros. Estos tipos de bases de datos se caracterizan por ser escalables y manejar grandes cantidades de datos y usuarios simultáneos.*

Casos de uso para una base de datos `NoSQL`

Las bases de datos `NoSQL` se utilizan en diferentes ámbitos y casos de uso:

- **Aplicaciones en tiempo real**: Se requiere una alta velocidad de lectura y escritura, como chatbots, juegos en línea, entre otros.
- **Big Data**: Ideales para el almacenamiento y análisis de grandes volúmenes de datos no estructurados, como datos de redes sociales, sensores, entre otros.
- **Microservicios**: Arquitecturas de microservicios para almacenar los datos de forma independiente.
- **Almacenamiento de objetos**: Almacenar objetos con relaciones complejas.
- **Aplicaciones móviles**: Almacenar y recuperar información de forma rápida y sin necesidad de conexión a un servidor central.

SQL vs. NoSQL - Modelo de datos

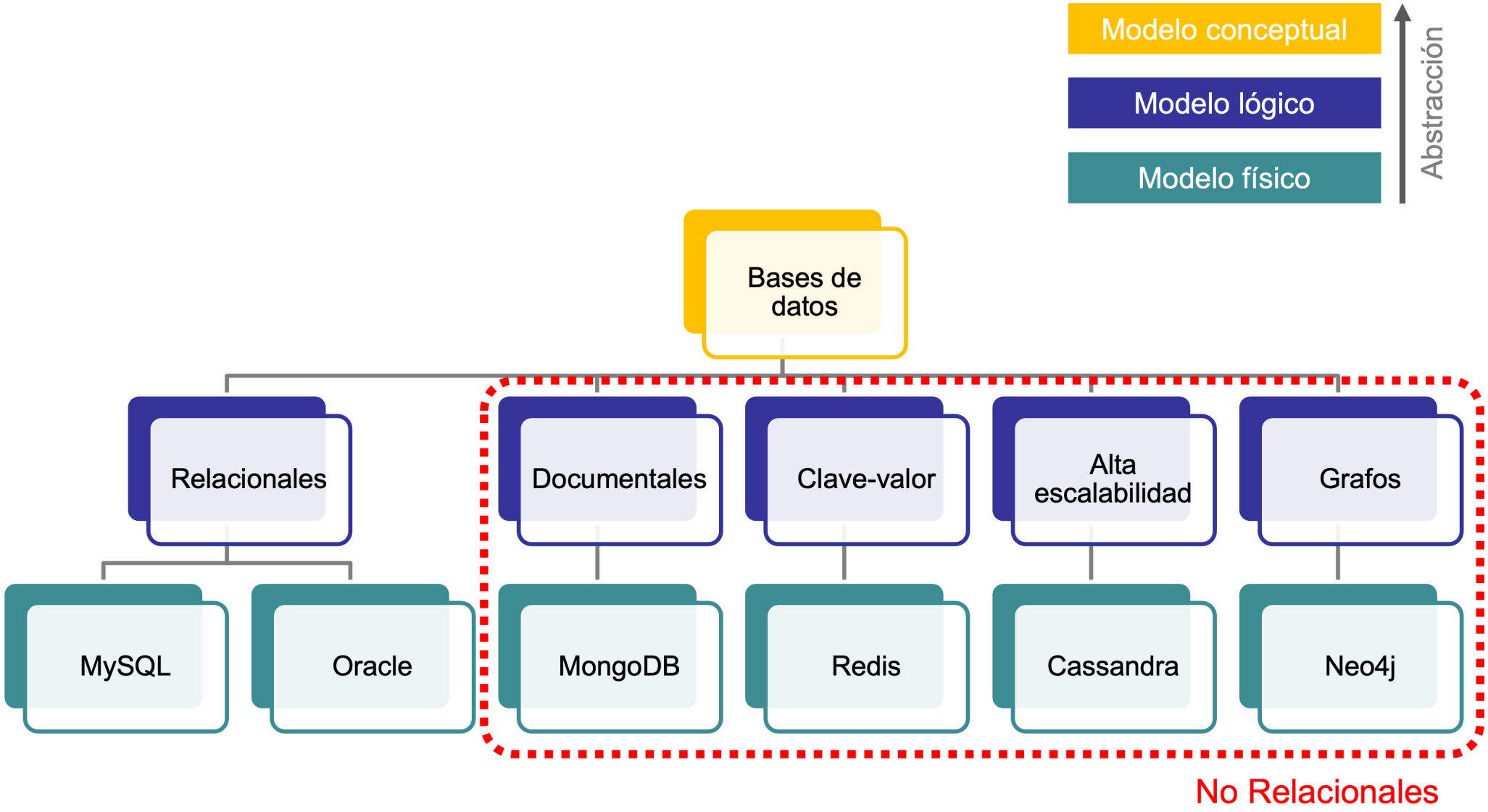
Permite describir las propiedades de la información almacenada en una base de datos:

- Estructuras de datos
- Restricciones
- Dependencias
- Dominios

Los modelos de datos son fundamentales para introducir la abstracción en una base de datos.

SQL vs. NoSQL - Modelo de datos

- **SQL** utilizan tablas con columnas (atributos) y filas (registros) fijas.
- **NoSQL** no se ciñen a un formato rígido donde aparecen diferentes categorías:
  - **Basadas en documentos**: almacenan y codifican los datos en documentos en formatos (JSON, XML, YAML y BSON).
  - **Basadas en grados**: estructuran los datos como nodos y relaciones para mostrar las conexiones entre los distintos elementos de datos.
  - **Basadas en columnas**: almacenan los datos en celdas agrupadas en un número ilimitado de columnas en lugar de filas.
  - **Basadas en clave-valor**: almacenan los datos como pares clave-valor, donde cada clave es un identificador único que corresponde a un valor asociado.


Modelo conceptual
Modelo lógico
Modelo físico
Abstracción
Bases de datos
Relacionales
Documentales
Clave-valor
Alta escalabilidad
Grafos
MySQL
Oracle
MongoDB
Redis
Cassandra
Neo4j
No Relacionales

SQL vs. NoSQL - Esquemas o arquitecturas

Las bases de datos `SQL` requiere un esquema *rígido, predefinido, estático o fijo*. Organiza los datos de forma tabular y relacional. Por lo tanto, es necesario *estructurar y organizar los datos* antes de crear una base de datos `SQL`.

Las bases de datos `NoSQL` tienen esquemas *flexibles y dinámicos* para datos que *no están estructurados*. Por lo tanto, no hay mucha necesidad de estructurar u organizar los datos antes de colocarlos en una base de datos `NoSQL`.

SQL vs. NoSQL - Escalabilidad

Tanto SQL como NoSQL son escalables, aunque la naturaleza de su escalabilidad es diferente.

- **SQL** pueden escalar "*verticalmente*" si se supera la capacidad actual del servidor lo que significa que se puede aumentar la potencia de procesamiento del hardware actual migrando a un servidor más grande.
- **NoSQL** pueden escalar fácilmente de forma "*horizontal*" añadiendo más servidores para gestionar un mayor tráfico según sea necesario.

> **¡Atención!** - Aunque las bases de datos `SQL` se pueden escalar horizontalmente, no están bien soportadas.

SQL vs. NoSQL - Velocidad *(parte 1)*

En general, ni `SQL` ni `NoSQL` son más rápidos que el otro. Su velocidad depende más bien del contexto en el que se utilicen.

- Las bases de datos `SQL`:
  - Se diseñaron cuando el almacenamiento de datos era caro y la duplicación de datos podía hacer perder mucho dinero.
  - Estan preparadas para ser más rápidas para consultas, uniones, actualizaciones, etc.

SQL vs. NoSQL - Velocidad *(parte 2)*

- Las bases de datos `NoSQL`:
  - Se diseñaron para datos no estructurados, es decir, pueden ser orientadas a columnas, grafos, documentos o tuplas clave-valor.
  - Los datos se almacenan juntos, es decir, es más rápido realizar operaciones de lectura o escritura en una entidad de datos.

> ***Resumen***: `SQL` es excelente para proteger la validez de los datos, mientras que `NoSQL` es ideal para cuando se necesita una rápida disponibilidad de big data.

Pros y contras *SQL*

- **Fiabilidad**: robustas y confiables, los datos están seguros y no se pierden fácilmente.
- **Modelado de datos estructurado**: con datos estructurados y relacionales.
- **Integridad de los datos**: se garantizan y mantiene la consistencia de los datos.
- **Lenguaje de consulta estándar**: `SQL` es un lenguaje estándar que es ampliamente utilizado y conocido.

Pros y contras *SQL*

- **Complejidad**: pueden ser más complejas de implementar y mantener que otras.
- **Costo**: a menudo se requieren licencias y hardware.
- **Rendimiento**: pueden tener un rendimiento limitado.
- **Dificultad de escalabilidad horizontal**: pueden ser más difíciles de escalar horizontalmente que las bases de datos `NoSQL`.

Pros y contras *NoSQL*

- **Flexibilidad de datos**: mayor flexibilidad en la estructura y modelado de datos.
- **Escalabilidad horizontal**: escalan horizontalmente para manejar grandes cantidades de datos y usuarios.
- **Rendimiento**: tienen un mejor rendimiento en situaciones de alta concurrencia o grandes cantidades de datos.
- **Coste**: no requieren licencias o hardware especializado.

Pros y contras NoSQL

- **Integridad de los datos**: no mantienenl la integridad de datos como SQL, lo que puede comprometer la exactitud y consistencia de los datos.
- **Lenguaje de consulta no estándar**: tienen lenguajes de consulta no estándar.
- **Dificultad en la realización de consultas complejas**: limitaciones en la realización de consultas complejas.
- **Falta de soporte y recursos**: son relativamente nuevas, puede haber menos soporte y recursos disponibles.

Curiosidad - ¿Qué usa Google?

Google es un gran ejemplo de empresa que entiende sus objetivos y puede elegir la mejor opción para sus necesidades entre una base de datos `SQL` y una `NoSQL`.

Dado que trabaja con conjuntos de **datos masivos**, ha optado por trabajar con una base de datos `NoSQL`. La empresa utiliza `Bigtable`, que es una base de datos de creación propia.

`Bigtable` es una tabla poco poblada que puede escalar hasta miles de millones de filas y miles de columnas, lo que te permite almacenar petabytes de datos. Se indexa solo un valor de cada fila; este es conocido como la clave de fila. Admite una capacidad de procesamiento de lectura y escritura alta con baja latencia, y es una fuente de datos ideal para las operaciones de **MapReduce**.

Cuando usamos *SQL*

Las bases de datos `SQL` son ideales cuando:

- Se necesita un alto nivel de seguridad e integridad de los datos.
- Tiene datos muy estructurados que no cambian con regularidad.
- Necesita realizar solicitudes ad hoc u otras consultas complejas
- No necesita escalar horizontalmente.
- Soporta sistemas transaccionales, como aplicaciones financieras o contables.

Cuando usamos *NoSQL*

Es mejor utilizar bases de datos `NoSQL` cuando:

- No requieres un alto nivel de seguridad e integridad de los datos
- Tiene muchos datos no estructurados o semiestructurados.
- Tiene datos que cambian con frecuencia y necesita la flexibilidad de un esquema dinámico.
- Quiere agilizar el desarrollo y ahorrar dinero utilizando un enfoque estructurado.
- Necesita escalar horizontalmente.

Ejemplos de Bases de datos

Bases de datos *relacionales*

Cumplen con el modelo relacional:

- Normalización

Es el tipo de base de datos más utilizado.

Utilizan el lenguaje `SQL` (*Structured Query Languaje*) para consultar y manipular datos.

Los datos son almacenados en tablas:

- Es posible "unir" diferentes tablas para recuperar información

Bases de datos documentales

1. **Modelo de documento:** Información almacenada en documentos
2. **Datos no estructurados:** Información semi-estructurada (Sin esquema fijo)
3. **Escalabilidad:** Horizontal y vertical
4. **Acceso flexible:** Muy eficientes para la manipular datos
5. **Replicación y distribución:** Replican datos y distribuyen sus nodos geográficamente

Aconsejan duplicar información:

- Mejora el rendimiento de las consultas
- Consultas muy limitadas.

Bases de datos clave-valor

Almacena toda la información en pares `<clave, valor>`.

- La clave es única, el valor puede ser cualquier objeto.
  - Clave: `a013`, Valor: `name = "Juana"; surname = "Roi"`

1. **Escalabilidad**: Principalmente horizontal
2. **Simplicidad**: La estructura de clave-valor es sencilla.
3. **Flexibilidad**: No requieren una estructura fija de datos
4. **Alta velocidad**: Lectura y escritura **muy** rápidas
5. **Altamente divisibles** (suelen almacenarse en memoria)

Falta de capacidad de relacionar datos y dificultad en consultas complejas.

Bases de datos columnares

1. Optimizadas para la completa recuperación de columnas de datos (*analítica de datos*)
2. **Escalabilidad horizontal**: Orientadas a ser distribuidas
3. **Compresión de datos**: Reduce el tamaño de los datos y aumenta la eficiencia.
4. **Eficiencia en la recuperación de datos**: *Muy* eficientes para recuperar de datos dadas columnas específicas.
5. **No limitadas a un esquema fijo**, con capacidad de adaptación a los cambios de datos
6. Tienden a ser **simples de gesionar**

Pensadas para entornos con pocas escrituras

Bases de datos orientadas a grafos

1. **Modelado de relaciones**: Información ⟶ grafo
    - Los nodos son entidades, las aristas son relaciones
2. **Completamente normalizadas**: No duplican información
3. Pensadas para **analizar y visualizar** redes de relaciones
4. Amplia variedad de consultas para acceder a los datos de manera eficiente (lenguaje de consultas es complejo)

Limitaciones:

- Falta de capacidad para realizar cálculos matemáticos complejos
- Dificultad para manejar grandes cantidades de nodos y relaciones en tiempo real