NoSQL

Historia y surgimiento de NoSQL

Hasta los años 90, las bases de datos relacionales (SQL) dominaban el mercado porque eran ideales para sistemas bancarios y administrativos. Con la expansión de Internet surgieron nuevas necesidades: manejar grandes volúmenes de datos, escalar rápidamente y mantener alta disponibilidad.

- **1998:** Carlo Strozzi utilizó por primera vez el término *NoSQL* para una base de datos que no seguía el modelo relacional tradicional.
- **Principios de los 2000-2005:** empresas como Google, Amazon y Facebook comenzaron a generar cantidades masivas de datos que los sistemas SQL tradicionales tenían dificultades para manejar.

  Google BigTable introdujo el concepto de almacenamiento distribuido entre múltiples servidores.

- **2008-2009:** Eric Evans y Johan Oskarsson popularizaron el término **NoSQL (Not Only SQL)** para referirse a las nuevas bases de datos no relacionales, destacando que SQL y NoSQL pueden coexistir para resolver problemas distintos.

¿Qué es NoSQL y cómo funciona?

Las bases de datos **NoSQL** son sistemas no relacionales diseñados para almacenar grandes volúmenes de información con:

- Alta escalabilidad.
- Flexibilidad en la estructura de datos.
- Distribución de información en múltiples servidores.
- Mayor disponibilidad y rendimiento.

A diferencia de SQL, no requieren tablas ni relaciones estrictas.

Diferencias entre SQL y NoSQL

Característica	SQL	NoSQL
Modelo	Tablas con filas y columnas	Documentos, clave-valor, grafos, columnas
Esquema	Rígido y predefinido	Flexible y dinámico
Escalabilidad	Principalmente vertical	Principalmente horizontal
Relaciones	Mediante claves foráneas y JOIN	Datos embebidos o relaciones específicas
Consultas	Lenguaje SQL estándar	Dependen de cada sistema
Uso ideal	Finanzas, ERP, sistemas transaccionales	Big Data, redes sociales, IoT, aplicaciones web

Propiedades SQL: ACID

Garantizan transacciones seguras y consistentes:

- **Atomicidad:** todo o nada.
- **Consistencia:** datos válidos.
- **Aislamiento:** transacciones independientes.
- **Durabilidad:** cambios permanentes.

Propiedades NoSQL: CAP

En sistemas distribuidos se priorizan dos de tres propiedades:

- **Consistencia**
- **Disponibilidad**
- **Tolerancia a particiones**

Análisis de las diferencias entre SQL y NoSQL

- Aunque ambos tipos de bases de datos tienen el mismo objetivo principal, almacenar y gestionar información, fueron diseñados para resolver problemas diferentes.
- Las bases de datos SQL se centran en mantener la integridad y consistencia de los datos mediante estructuras bien definidas y relaciones entre tablas. Esto las convierte en una excelente opción para sistemas donde los errores o inconsistencias pueden generar consecuencias importantes, como aplicaciones bancarias, financieras o de gestión empresarial.
- Por otro lado, las bases de datos NoSQL fueron creadas para responder a los desafíos de las aplicaciones modernas, donde los datos crecen rápidamente y pueden cambiar constantemente. Su estructura flexible permite incorporar nuevos campos o tipos de información sin necesidad de modificar toda la base de datos, facilitando el desarrollo ágil y la adaptación a nuevos requerimientos.
- Otra diferencia importante es la forma en que escalan. Mientras que SQL suele requerir servidores cada vez más potentes para soportar una mayor carga de trabajo, NoSQL distribuye los datos entre múltiples servidores, permitiendo aumentar la capacidad del sistema de forma más sencilla y económica.
- En términos generales, SQL prioriza la consistencia y la organización de la información, mientras que NoSQL prioriza la flexibilidad, el rendimiento y la capacidad de manejar grandes volúmenes de datos distribuidos.

Ventajas y desventajas de NoSQL

Ventajas

- Esquema flexible.
- Escalabilidad horizontal.
- Alto rendimiento.
- Menor costo de crecimiento.
- Replicación y alta disponibilidad.
- Ideal para datos semiestructurados y no estructurados.

Desventajas

- Menor estandarización.
- Algunas soluciones no garantizan ACID completo.
- Relaciones complejas más difíciles de implementar.

- Consultas avanzadas pueden ser más complicadas.
- Puede ofrecer menos funcionalidades empresariales que algunas bases SQL.

Tipos de bases de datos NoSQL

1. Clave-Valor

Almacenan pares **clave → valor**.

Ventajas:

- Muy rápidas.
- Simples de implementar.

Usos:

- Caché.
- Sesiones de usuarios.
- Carritos de compra.
- Videojuegos.

Ejemplos: Redis, Memcached, RocksDB.

---

2. Documentales

Almacenan documentos JSON, BSON o XML.

Ventajas:

- Gran flexibilidad.
- Fácil adaptación a cambios.

Usos:

- Catálogos de productos.
- Perfiles de usuarios.
- Aplicaciones web y móviles.

Ejemplos: MongoDB, CouchDB, Firebase Firestore.

---

3. Grafos

Representan información mediante nodos y relaciones.

Ventajas:

- Excelente manejo de relaciones complejas.

Usos:

- Redes sociales.
- Sistemas de recomendación.
- Detección de fraude.
- Navegación y rutas.

Ejemplos: Neo4j, Amazon Neptune, JanusGraph.

---

4. En Memoria

Almacenan datos directamente en la RAM.

Ventajas:

- Máxima velocidad.

Desventajas:

- Mayor costo.
- Riesgo de pérdida de datos ante fallos.

Usos:

- Caché.
- Videojuegos online.
- Sistemas de mensajería.

---

5. De Búsqueda

Especializadas en indexar y buscar información rápidamente.

Usos:

- Motores de búsqueda.
- Tiendas online.
- Análisis de logs.

Ejemplos: Elasticsearch, Apache Solr, MeiliSearch.

---

6. De Columna Ancha

Almacenan datos por columnas en lugar de filas.

Ventajas:

- Muy eficientes para Big Data.
- Escalabilidad horizontal.

Usos:

- IoT.
- Grandes volúmenes de datos.

Ejemplos: Cassandra, HBase, ScyllaDB.

Futuro de NoSQL

Las tendencias apuntan a:

- **Modelos híbridos (SQL + NoSQL)** para aprovechar las ventajas de ambos enfoques.
- Mayor integración con servicios en la nube.
- Sistemas multi modelo que soportan varios tipos de datos.
- Más compatibilidad con transacciones ACID.
- Uso creciente en **Big Data**, **IoT** e **Inteligencia Artificial**.

Conclusión

SQL prioriza la consistencia y las relaciones complejas. NoSQL prioriza la escalabilidad, flexibilidad y rendimiento para grandes volúmenes de datos distribuidos.