# INTRODUCCIÓN A NoSQL

## Introducción.

Amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de bases de datos relacionales *RDBMS* (*Relational DataBase Management System*) en aspectos importantes, siendo el más destacado que no usan *SQL* como lenguaje principal de consultas.

Los datos almacenados no requieren estructuras fijas como tablas.

Normalmente no soportan operaciones JOIN, ni garantizan completamente *ACID* (*Atomicity, Consistency, Isolation, Durability*).

Normalmente escalan bien horizontalmente.

Los sistemas *NoSQL* a veces se denominan “*no sólo SQL*”, ya que también pueden soportar lenguajes de consulta tipo *SQL*.

*NoSQL* y *RDBMS* son tecnologías complementarias.

## Características.

### Consistencia Eventual.

En la mayoría de sistemas *NoSQL* no se implementan mecanismos rígidos de consistencia que garanticen que cualquier cambio llevado a cambio en el sistema distribuido sea visto, al mismo tiempo, por todos los nodos y asegurando, también, la no violación de posibles restricciones de integridad de los datos u otras reglas definidas.

En lugar de estos mecanismos, para obtener un mayor rendimiento, se ofrece el concepto de “consistencia eventual”, en el que los cambios realizados “con el tiempo” serán propagados a todos los nodos por lo que, una consulta podría no devolver los últimos datos disponibles o proporcionar datos inexactos, problema conocido como “lecturas sucias” o “lecturas obsoletas”.

En algunos sistemas *NoSQL* se pueden presentar pérdidas de datos en escritura. Esto se conoce como *BASE* (*Basically Available Soft-state Eventual Consistency*), en contraposición a *ACID*.

### Flexibilidad en el esquema.

En la mayoría de bases de datos *NoSQL*, los esquemas de datos son flexibles; es decir, a diferencia de las bases de datos relacionales en las que, la escritura de los datos, debe adaptarse a unas estructuras (tablas) y tipos de datos predefinidos, en los sistemas *NoSQL*, cada registro (o documento) puede contener una información con diferente forma cada vez, pudiendo así almacenar solo los atributos que interesen en cada uno de ellos, facilitando el polimorfismo de datos bajo una misma colección de información.

También se pueden almacenar estructuras complejas de datos en un solo documento.

### Escalabilidad horizontal.

Posibilidad de incrementar el rendimiento del sistema añadiendo más nodos (servidores) e indicando al sistema cuáles son los nodos disponibles.

### Estructura distribuida.

Generalmente los datos se distribuyen, entre los diferentes nodos que componen el sistema.

#### Particionado (o Sharding).

Distribuye los datos entre múltiples servidores de forma que, cada servidor, actúe como única fuente de un subconjunto de datos.

Normalmente se utilizan mecanismos de tablas hash distribuidas (*DHT*) para realizar esta distribución.

#### Réplica.

Copia los datos entre múltiples servidores, de forma que cada bit de datos pueda ser encontrado en múltiples lugares.

##### Réplica maestro-esclavo.

Un servidor gestiona la escritura de la copia autorizada, mientras que los esclavos se sincronizan con este servidor maestro y solo gestionan las lecturas.

##### Réplica peer-to-peer.

Se permiten escrituras a cualquier nodo y ellos se coordinan entre sí para sincronizar sus copias de los datos.

## Tipos de bases de datos NoSQL.

No todas las tecnologías *NoSQL* usan el mismo modelo de datos.

La idoneidad particular de una base de datos *NoSQL* dependerá del problema a resolver.

Los modelos de datos usados por *NoSQL* se pueden agrupar en cuatro grandes categorías.

### Base de datos de documentos.

Bases de datos *NoSQL* más versátiles.

Almacena la información como un documento.

Habitualmente se usa una estructura simple como *JSON* (*JavaScript Object Notification*), *BSON* (*Binary JSON*), *XML* (*eXtensive Markup Language*).

Se utiliza una clave única para cada registro.

Este tipo de implementación permite, además de realizar búsquedas por clave-valor, realizar consultas más avanzadas sobre el contenido del documento.

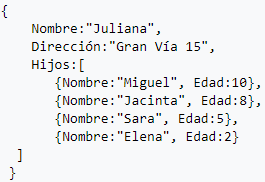
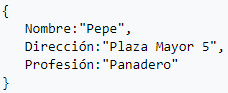


Figura 1. Registros JSON en una base de datos documental.

### Almacenamiento clave-valor.

Bases de datos NoSQL más populares y sencillas en cuanto a su funcionalidad.

Cada elemento está identificado por una clave única, lo que permite la recuperación de la información de forma muy rápida, la cual suele almacenarse como un objeto binario.

Son muy eficientes tanto para lecturas como para escrituras.

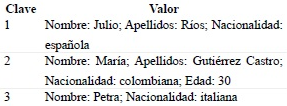


Figura 2. Ejemplo de base de datos que utiliza almacenamiento clave-valor.

### Bases de datos de grafos.

Los datos se almacenan en estructuras grafo con nodos (entidades) y bordes (conexiones entre las entidades).

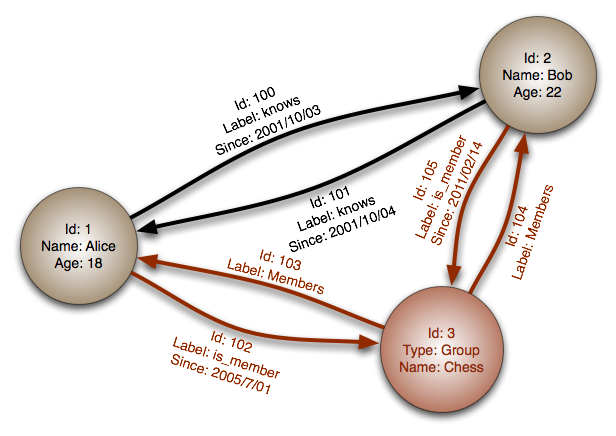


Figura 3. Ejemplo de base de datos orientada a grafos, que emplea nodos, propiedades y bordes.

### Base de datos columnar (o Columna ancha).

En vez de tablas tradicionales, se utilizan familias de columnas que, son los contenedores de las filas.

A diferencia de los *RDBMS*, no necesita conocer de antemano todas las columnas, cada fila no tiene por qué tener el mismo número de columnas.

Este tipo de bases de datos se adecúan mejor a operaciones analíticas sobre grandes conjuntos de datos.

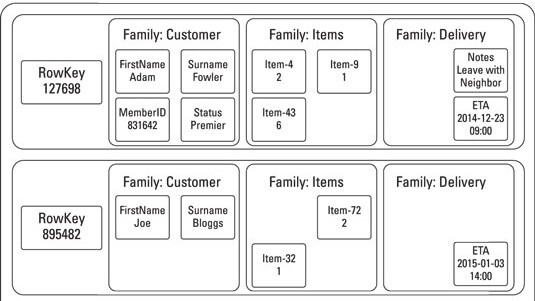


Figura 4. Ejemplo de base de datos columnar.

## SQL

### Ventajas.

* Está más adaptado su uso y los perfiles que los conocen son mayoritarios y más baratos.
* Debido al largo tiempo que llevan en el mercado, estas herramientas tienen un mayor soporte y mejores suites de productos y add-ons para gestionar estas bases de datos.
* La atomicidad de las operaciones en la base de datos. Esto es, que en estas bases de datos o se hace la operación entera o no se hace utilizando la famosa técnica del rollback.
* Los datos deben cumplir requisitos de integridad tanto en tipo de dato como en compatibilidad.

### Desventajas.

* La atomicidad de las operaciones juega un papel crucial en el rendimiento de las bases de datos.
* Escalabilidad: Aunque probada en muchos entornos productivos suele, por norma, ser inferior a las bases de datos NoSQL.

## NoSQL

### Ventajas.

* La escalabilidad y su carácter descentralizado. Soportan estructuras distribuidas.
* Suelen ser bases de datos mucho más abiertas y flexibles. Permiten adaptarse a necesidades de proyectos con mayor facilidad que los modelos de Entidad-Relación.
* Se pueden hacer cambios de los esquemas sin tener que parar bases de datos.
* Escalabilidad horizontal: Son capaces de crecer en número de máquinas, en lugar de tener que residir en grandes máquinas.
* Se pueden ejecutar en máquinas con pocos recursos.
* Optimización de consultas en base de datos para grandes cantidades de datos.

### Desventajas.

* No todas las bases de datos NoSQL contemplan la atomicidad de las instrucciones y la integridad de los datos. Soportan lo que se llama consistencia eventual.
* Problemas de compatibilidad entre instrucciones SQL: Las nuevas bases de datos utilizan sus propias características en el lenguaje de consulta y no son 100% compatibles con el SQL de las bases de datos relacionales.
* El soporte a problemas con las queries de trabajo en una base de datos NoSQL es más complicado.
* Falta de estandarización: Hay muchas bases de datos NoSQL y aún no hay un estándar como si lo hay en las bases de datos relacionales. Se presume un futuro incierto en estas bases de datos.
* Soporte multiplataforma: Aún quedan muchas mejoras en algunos sistemas para que soporten sistemas operativos que no sean Linux.
* Suelen tener herramientas de administración no muy usables o se accede por consola.

## SQL vs NoSQL: Cuándo utilizar cada tipo de bases de datos.

* Cuando los datos deben ser consistentes sin dar posibilidad al error utilizar una base de datos relacional: SQL.
* Cuando nuestro presupuesto no se puede permitir grandes máquinas y debe destinarse a máquinas de menor rendimiento: NoSQL.
* Cuando las estructuras de datos que manejamos son variables: NoSQL.
* Análisis de grandes cantidades de datos en modo lectura: NoSQL.
* Captura y procesado de eventos: NoSQL.
* Tiendas online con motores de inteligencia complejos: NoSQL.