# Bases de datos no relacionales

Luis Valencia Cabrera (lvalencia@us.es)

Research Group on Natural Computing
Departamento de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

26-11-2025, Bases de Datos

# Índice

# Índice

# Mundo relacional

Hemos trabajado con bases de datos relacionales...

# Mundo relacional

Hemos trabajado con bases de datos relacionales...

- Capturando requisitos funcionales

# Mundo relacional

Hemos trabajado con bases de datos relacionales...

- Capturando requisitos funcionales
- Abstrayendo a un diseño conceptual

# Mundo relacional

Hemos trabajado con bases de datos relacionales...

- Capturando requisitos funcionales
- Abstrayendo a un diseño conceptual
- Trasladando a diseño lógico relacional

# Mundo relacional

Hemos trabajado con bases de datos relacionales...

- Capturando requisitos funcionales
- Abstrayendo a un diseño conceptual
- Trasladando a diseño lógico relacional
- Normalizando el diseño anterior

# Mundo relacional

Hemos trabajado con bases de datos relacionales...

- Capturando requisitos funcionales
- Abstrayendo a un diseño conceptual
- Trasladando a diseño lógico relacional
- Normalizando el diseño anterior
- Comenzando a implementar en MySQL/Access

# Mundo relacional

Hemos trabajado con bases de datos relacionales...

- Capturando requisitos funcionales
- Abstrayendo a un diseño conceptual
- Trasladando a diseño lógico relacional
- Normalizando el diseño anterior
- Comenzando a implementar en MySQL/Access
- Planteando consultas sobre la BD con SQL

# Fortalezas

Podemos resaltar algunas bondades de este enfoque:

# Fortalezas

Podemos resaltar algunas bondades de este enfoque:

- Diseño sólido

# Fortalezas

Podemos resaltar algunas bondades de este enfoque:

- Diseño sólido
- Captura de estructuras lógicas del *minimundo*
- Estandarización del lenguaje de consultas

# Fortalezas

Podemos resaltar algunas bondades de este enfoque:

- Diseño sólido
- Captura de estructuras lógicas del *minimundo*
- Estandarización del lenguaje de consultas
- Separación de datos y aplicaciones

# Otros ámbitos

## ¿Es siempre el enfoque adecuado?

Veamos algunas situaciones que se pueden dar...

# Otros ámbitos

## ¿Es siempre el enfoque adecuado?

Veamos algunas situaciones que se pueden dar...

- *Necesidades distintas* según ámbito de aplicación

# Otros ámbitos

## ¿Es siempre el enfoque adecuado?

Veamos algunas situaciones que se pueden dar...

- *Necesidades distintas* según ámbito de aplicación
- **Estructura** de *minimundo* **demasiado dinámica**

# Otros ámbitos

## ¿Es siempre el enfoque adecuado?

Veamos algunas situaciones que se pueden dar...

- *Necesidades distintas* según ámbito de aplicación
- **Estructura** de *minimundo* **demasiado dinámica**
- **Imposible** dotar de **esquema estable**

# Otros ámbitos

## ¿Es siempre el enfoque adecuado?

Veamos algunas situaciones que se pueden dar...

- *Necesidades distintas* según ámbito de aplicación
- **Estructura** de *minimundo* **demasiado dinámica**
- **Imposible** dotar de **esquema estable**
- **Volúmenes de datos** que lo hagan **impracticable**

# Otros ámbitos

## ¿Es siempre el enfoque adecuado?

Veamos algunas situaciones que se pueden dar...

- *Necesidades distintas* según ámbito de aplicación
- **Estructura** de *minimundo* **demasiado dinámica**
- **Imposible** dotar de **esquema estable**
- **Volúmenes de datos** que lo hagan **impracticable**
- **Cantidad** de **consultas** y **actualizaciones concurrentes**

# Otros ámbitos

## ¿Es siempre el enfoque adecuado?

Veamos algunas situaciones que se pueden dar...

- *Necesidades distintas* según ámbito de aplicación
- **Estructura** de *minimundo* **demasiado dinámica**
- **Imposible** dotar de **esquema estable**
- **Volúmenes de datos** que lo hagan **impracticable**
- **Cantidad** de **consultas** y **actualizaciones concurrentes**
- **Problemas de escalabilidad**

# Otros ámbitos

## ¿Es siempre el enfoque adecuado?

Veamos algunas situaciones que se pueden dar...

- *Necesidades distintas* según ámbito de aplicación
- **Estructura** de *minimundo* **demasiado dinámica**
- **Imposible** dotar de **esquema estable**
- **Volúmenes de datos** que lo hagan **impracticable**
- **Cantidad** de **consultas** y **actualizaciones concurrentes**
- **Problemas de escalabilidad**

¿Qué hacemos?

# Otros ámbitos

## ¿Es siempre el enfoque adecuado?

Veamos algunas situaciones que se pueden dar...

- *Necesidades distintas* según ámbito de aplicación
- **Estructura** de *minimundo* **demasiado dinámica**
- **Imposible** dotar de **esquema estable**
- **Volúmenes de datos** que lo hagan **impracticable**
- **Cantidad** de **consultas** y **actualizaciones concurrentes**
- **Problemas de escalabilidad**

¿Qué hacemos?

- Buscar otras soluciones que escalen bien horizontalmente y nos ayuden a *afrontar **mejor** estos escenarios*

# Índice

# Nuevos retos

- El mundo en el que vivimos se encuentra en **constante evolución**, formamos parte de un *súper organismo* sujeto a cambio. Nuestro entorno cambia día a día, y más en el aspecto tecnológico.

# Nuevos retos

- El mundo en el que vivimos se encuentra en **constante evolución**, formamos parte de un *súper organismo* sujeto a cambio. Nuestro entorno cambia día a día, y más en el aspecto tecnológico.
- Google, Wikipedia, redes sociales como Instagram, X/Twitter, recursos como ChatGPT, etc. implican un **cambio de escala** en el empleo de las TIC, y un **reto** en cuanto a la **actualización concurrente y disponibilidad** de ***enormes conjuntos*** de información en red.

# Nuevos retos

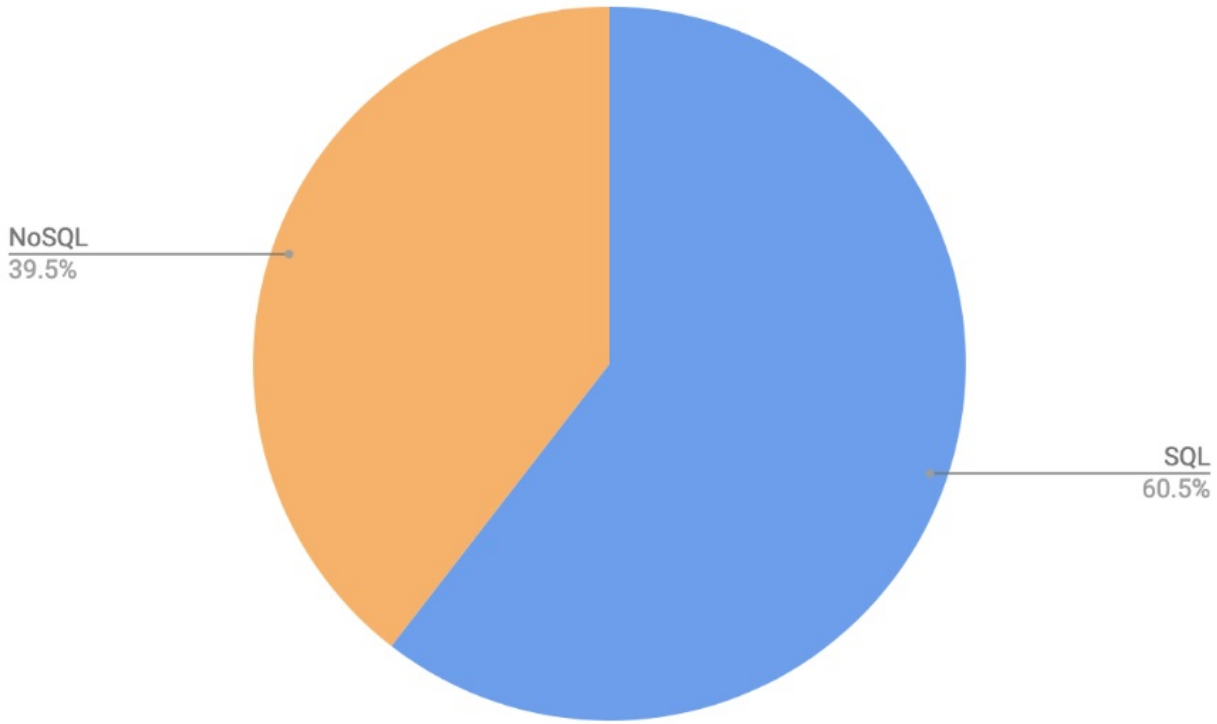
- El mundo en el que vivimos se encuentra en **constante evolución**, formamos parte de un *súper organismo* sujeto a cambio. Nuestro entorno cambia día a día, y más en el aspecto tecnológico.
- Google, Wikipedia, redes sociales como Instagram, X/Twitter, recursos como ChatGPT, etc. implican un **cambio de escala** en el empleo de las TIC, y un **reto** en cuanto a la **actualización concurrente y disponibilidad** de ***enormes conjuntos*** de información en red.
- Surge el mundo NoSQL (Not Only SQL)

# Nuevos retos

- El mundo en el que vivimos se encuentra en **constante evolución**, formamos parte de un *súper organismo* sujeto a cambio. Nuestro entorno cambia día a día, y más en el aspecto tecnológico.
- Google, Wikipedia, redes sociales como Instagram, X/Twitter, recursos como ChatGPT, etc. implican un **cambio de escala** en el empleo de las TIC, y un **reto** en cuanto a la **actualización concurrente y disponibilidad** de ***enormes conjuntos*** de información en red.
- Surge el mundo NoSQL (Not Only SQL)
- Además, el *Big Data* lleva el tamaño de los **volúmenes de datos** a **otra dimensión**.

# ¿Estado actual?

## Cuota relacional VS no relacional




[1]

[1] https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/

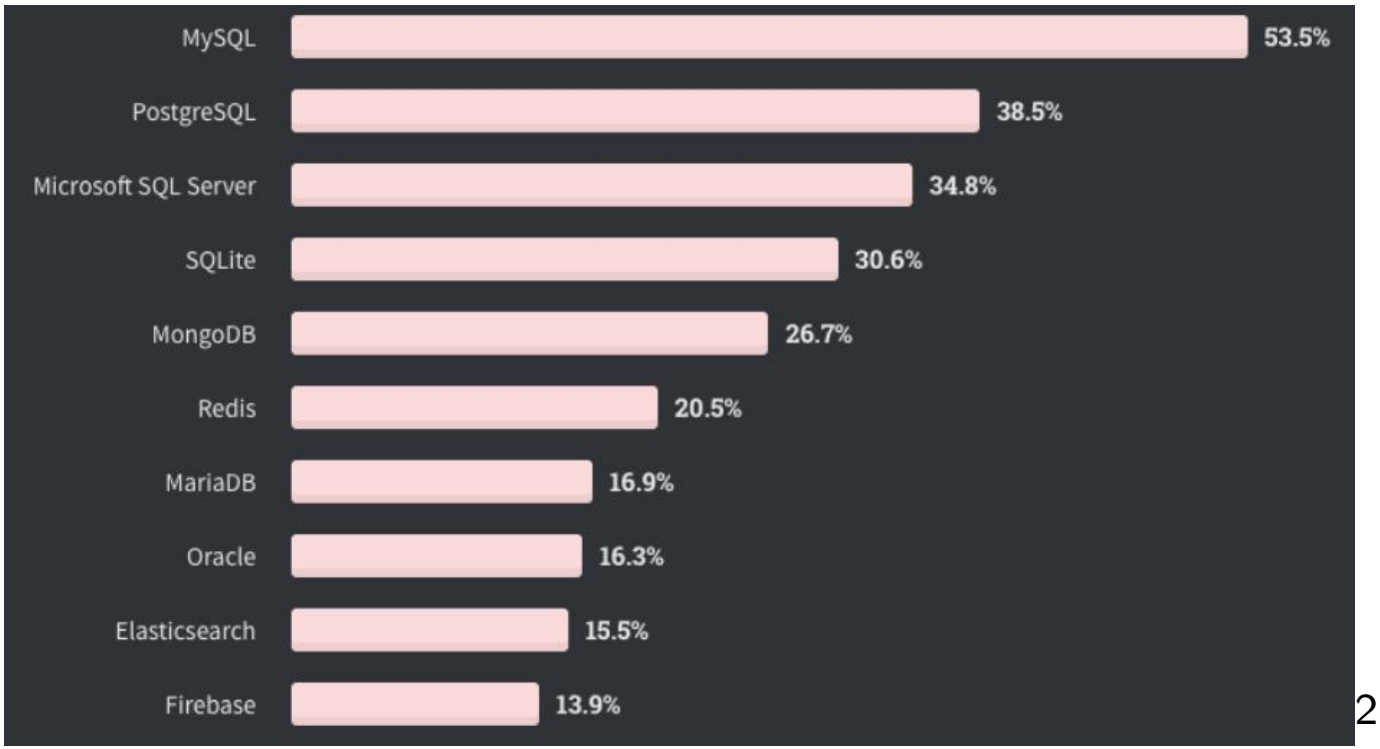
# ¿Estado actual?

## Ranking de popularidad a noviembre de 2025

| Rank Nov 2025 | Rank Oct 2025 | Rank Nov 2024 | DBMS | Database Model | Score Nov 2025 | Score Oct 2025 | Score Nov 2024 |
|---|---|---|---|---|---|---|---|
| 1. | 1. | 1. | Oracle | Relational, Multi-model | 1239.78 | +27.01 | -77.23 |
| 2. | 2. | 2. | MySQL | Relational, Multi-model | 865.82 | -13.84 | -151.98 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational, Multi-model | 718.87 | +3.82 | -80.94 |
| 4. | 4. | 4. | PostgreSQL | Relational, Multi-model | 651.36 | +8.16 | -2.97 |
| 5. | 5. | 5. | MongoDB | Multi-model | 371.68 | +3.66 | -29.25 |
| 6. | 6. | ↑ 7. | Snowflake | Relational | 197.84 | -0.81 | +55.35 |
| 7. | 7. | ↓ 6. | Redis | Key-value, Multi-model | 145.09 | +2.76 | -3.55 |
| 8. | 8. | ↑ 14. | Databricks | Multi-model | 131.50 | +2.70 | +45.04 |
| 9. | 9. | 9. | IBM Db2 | Relational, Multi-model | 119.28 | -3.10 | -2.47 |
| 10. | 10. | ↓ 8. | Elasticsearch | Multi-model | 113.97 | -2.69 | -17.67 |
| 11. | ↑ 12. | ↓ 10. | SQLite | Relational | 104.19 | -0.36 | +4.71 |
| 12. | ↓ 11. | ↓ 11. | Apache Cassandra | Wide column, Multi-model | 102.71 | -2.45 | +5.00 |
| 13. | 13. | ↑ 15. | MariaDB | Relational, Multi-model | 87.36 | -0.41 | +4.66 |
| 14. | 14. | ↓ 12. | Microsoft Access | Relational | 78.29 | -2.50 | -13.03 |
| 15. | ↑ 16. | ↑ 16. | Microsoft Azure SQL Database | Relational, Multi-model | 76.38 | +0.90 | -0.15 |
| 16. | ↑ 18. | ↓ 13. | Splunk | Search engine | 76.10 | +2.22 | -12.37 |
| 17. | ↓ 15. | 17. | Amazon DynamoDB | Multi-model | 75.78 | -0.13 | +3.38 |
| 18. | ↓ 17. | 18. | Apache Hive | Relational | 72.87 | -2.35 | +21.37 |
| 19. | 19. | 19. | Google BigQuery | Relational | 62.79 | -0.42 | +12.20 |
| 20. | 20. | ↑ 21. | Neo4j | Graph | 52.36 | -0.15 | +9.66 |

# Un poco de Historia

## ¿Estado en los últimos años?

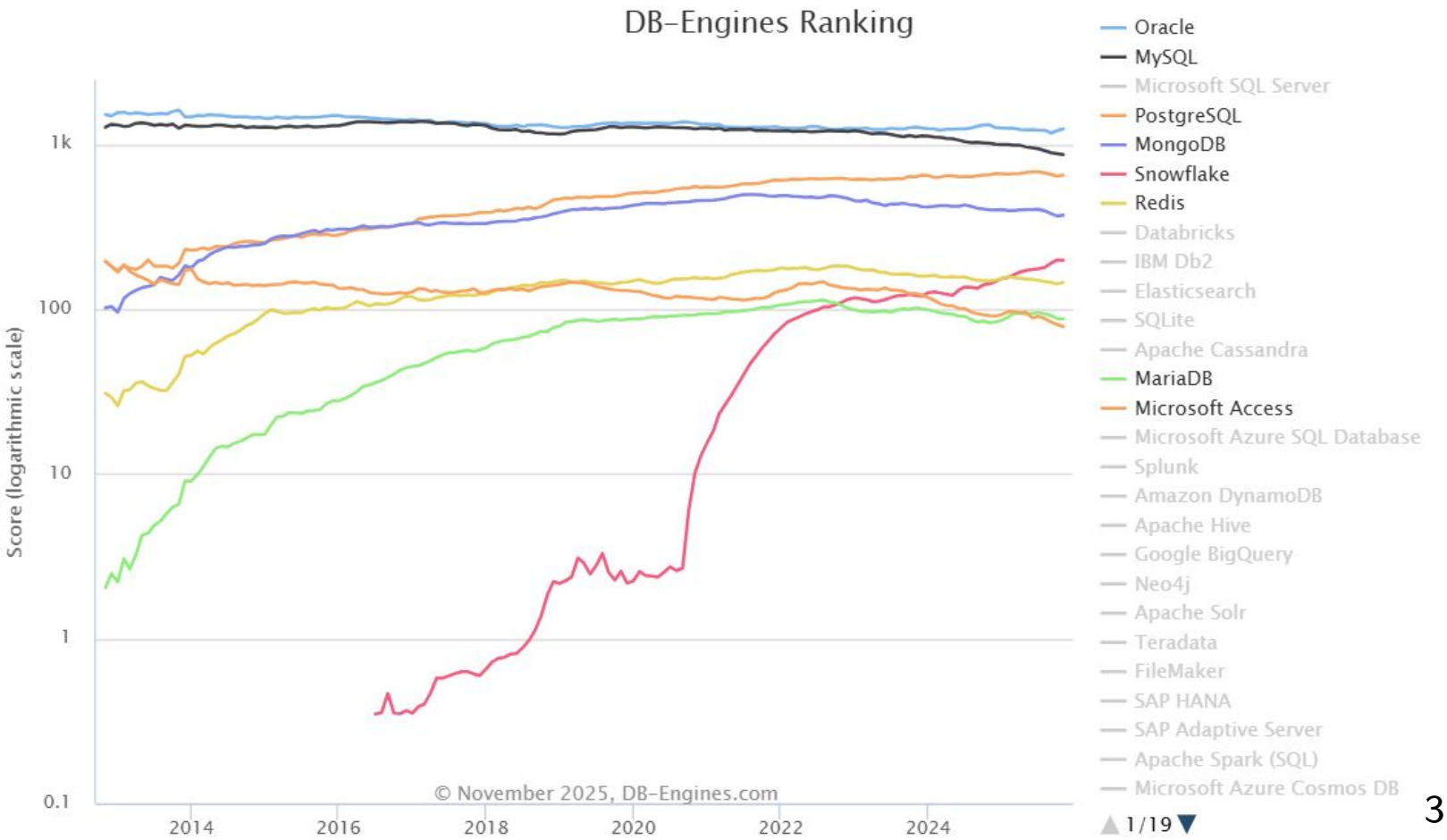


[2]

[2] https://www.eversql.com/most-popular-databases-in-2020/

# Un poco de Historia

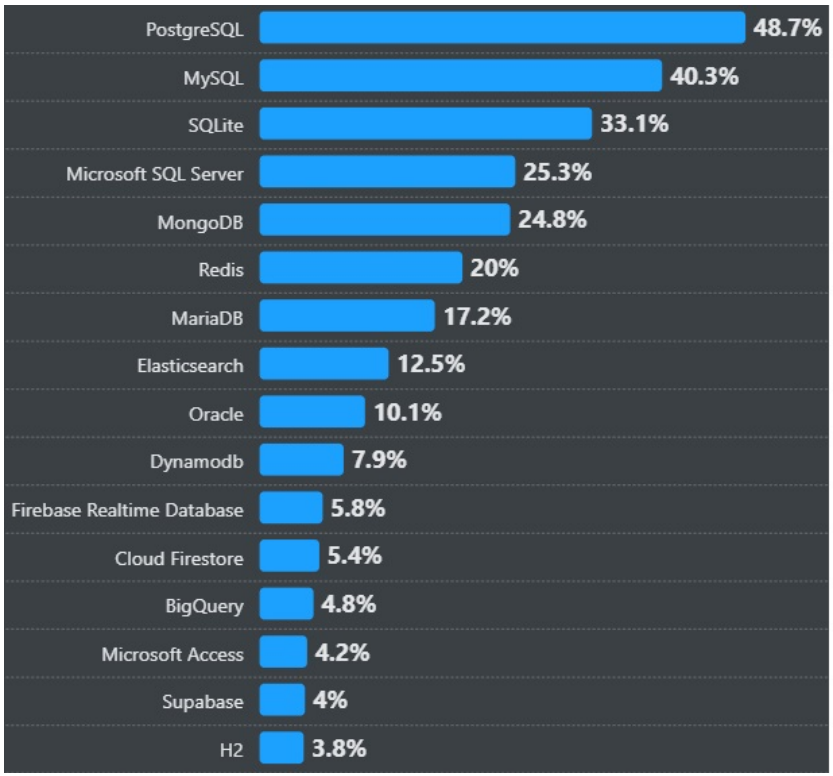
## ¿Estado actual? DB-Engines - Evolución MySQL VS MongoDB

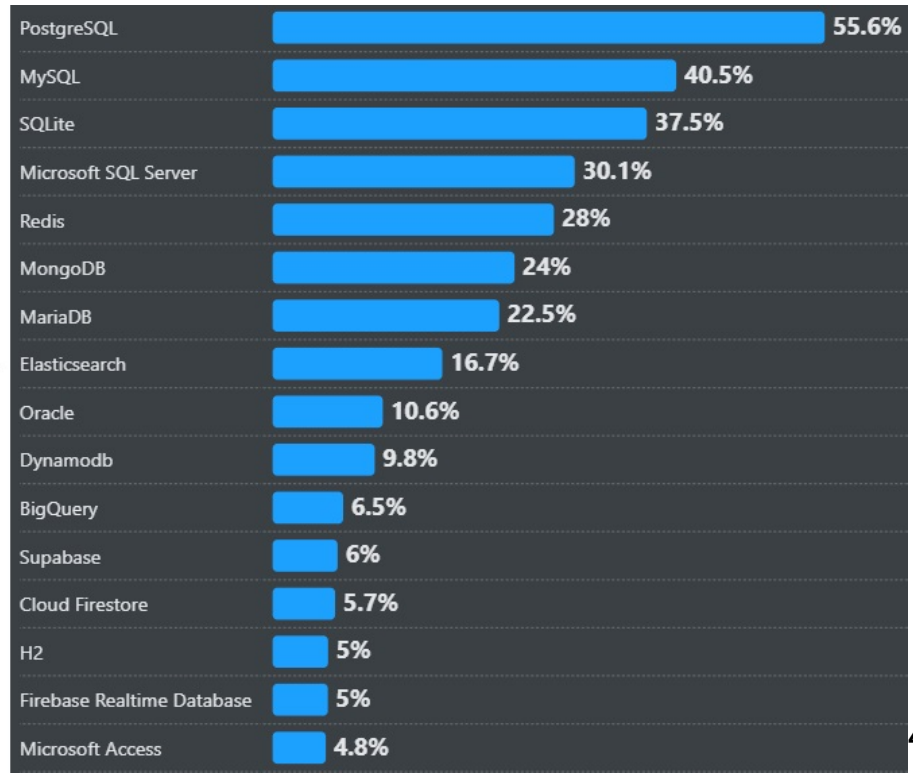



[3]

[3] https://db-engines.com/en/ranking_trend

# Stack Overflow Survey 2024 → 2025 - Todos

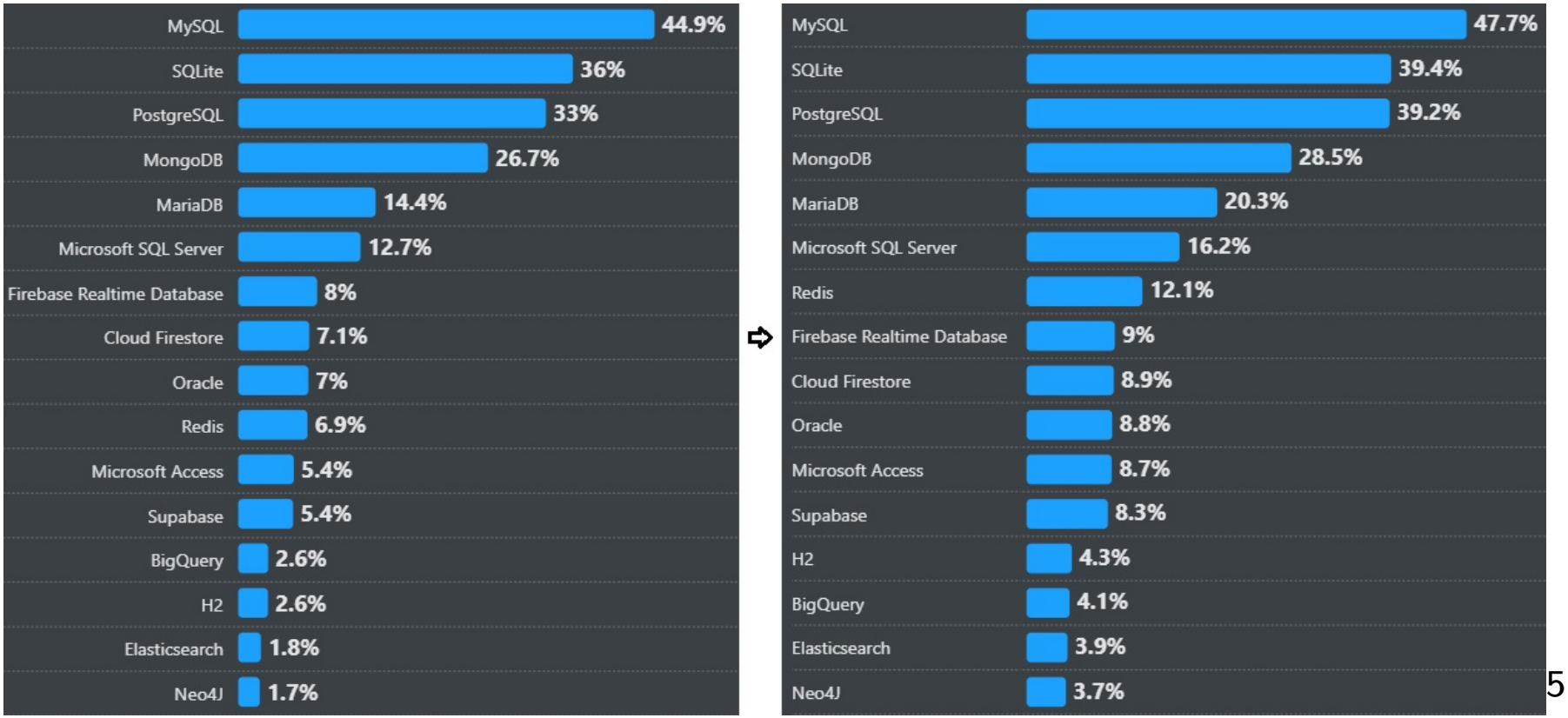



[4]

[4] https://survey.stackoverflow.co/2025/technology#1-databases

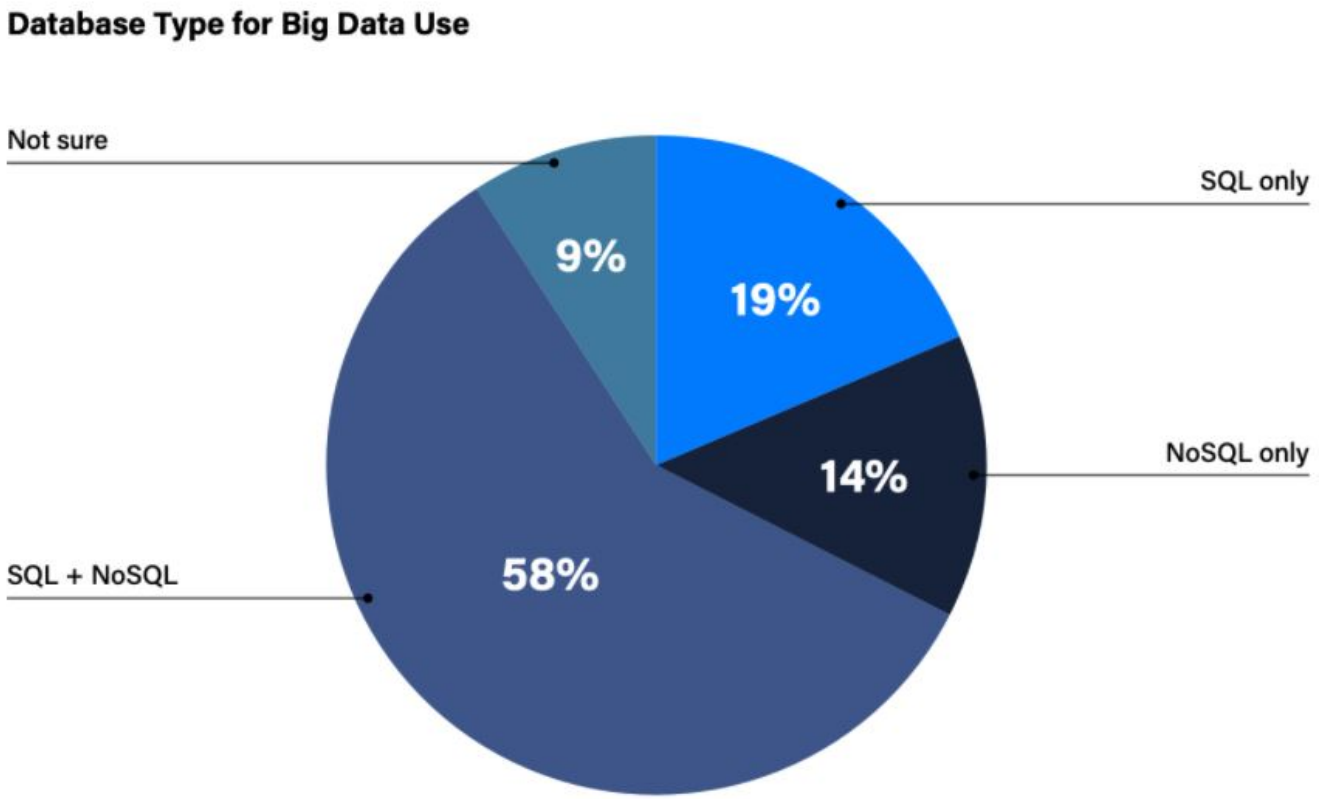
# Stack Overflow Survey 2024 → 2025 - Principiantes



[5]

[5] https://survey.stackoverflow.co/2025/technology#1-databases

# Situación actual: Coexistencia




6

[6] https://www.cockroachlabs.com/blog/dzone-sql-trend/

# Índice

# Very Large Databases (VLDB)

## Definición

Las siglas VLDB (*Very Large Databases*) denotan a bases de datos que contienen tablas cuyo número de tuplas resulta **extremadamente alto**.

- No se puede hablar de cifras concretas, esto va cambiando con el tiempo, evolucionando con la capacidad de almacenamiento de equipos personales, servidores, aplicaciones, etc.
- Por hacernos una idea, se puede entender como VLDB una base de datos de cientos de terabytes, con cientos de miles de millones de filas.
- Aparece incluso el concepto de XLDB (*Extremely Large Databases*).

# Big Data

Los términos anteriores dejaron paso a lo largo de aprox. la última década a la denominación de **Big Data**.

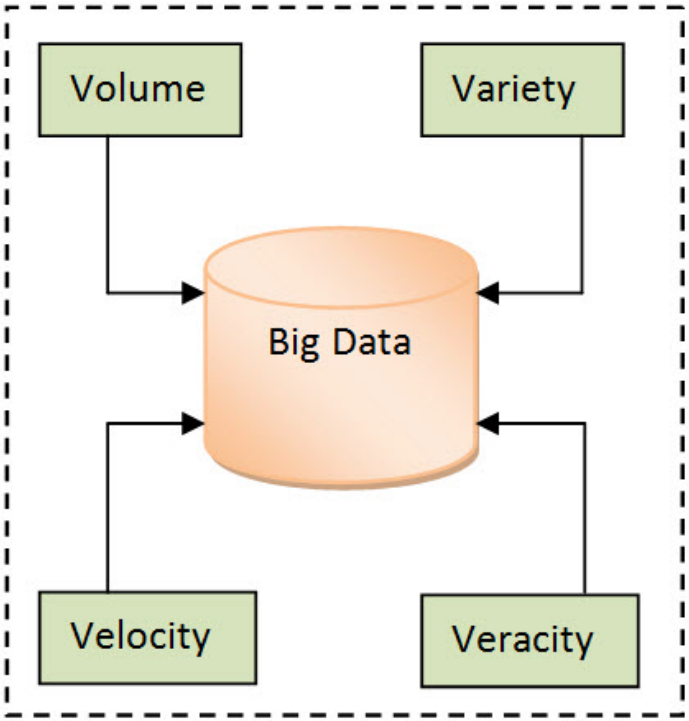
## Definición

Entendemos por *Big Data* a conjuntos de información enormes, que superan con mucho la capacidad del software de gestión de bases de datos convencional para procesar datos en tiempo razonable.

Los retos con este tipo de tecnologías se centran en la captura, almacenamiento, búsqueda, compartición, procesamiento, análisis y visualización de los datos.

# Big Data - 4 V's

En el marco de Big Data se viene imponiendo la terminología de las 4 V's para resumir las características fundamentales:




## 4 V's

| | |
|---|---|
| Volume | Grandes volúmenes de datos (petabytes, exabytes, etc.) |
| Variety | No limitado a información relacional, puede incluir datos no estructurados |
| Velocity | Los datos se deben procesar y analizar rápidamente |
| Veracity | Veracidad, adhesión a la verdad, precisión. |

# Big Data - 7 V's

## 7 V's

| | |
|---|---|
| Volume | Grandes volúmenes de datos (petabytes, exabytes, etc.) |
| Variety | No limitado a información relacional, puede incluir datos no estructurados |
| Velocity | Los datos se deben procesar y analizar rápidamente |
| Veracity | Veracidad, adhesión a la verdad, precisión. |
| Variability | Datos cuyo significado cambia constantemente |
| Visualisation | Presentar los datos de forma comprensible |
| Value | Extraer información para la toma de decisiones |

# Big Data

## ¿8 V's?


01
VOLUME
Can you find the information you are looking for?
02
VALUE
Can you find it when you most need it?
03
VERACITY
Are you dealing with information or disinformation?
04
VISUALISATION
Can you make sense at a glance? Does it trigger a decision?
05
VARIETY
Is a picture worth a thousand words in 70 languages? Is your information balanced?
06
VELOCITY
Information gains momentum and crises & opportunities evolve in real time. How is outlook for today?
07
VISCOSITY
Does it stick with you? Does it call for action?
08
VIRALITY
AHA to-go? Does it convey a message that can be pasted into a presentation or instgrammed?
BIG DATA
with 8 V's

# Índice

# Big Data y NoSQL

- **Enormes volúmenes de datos**, y condiciones impuestas por las V's (disponibilidad, frecuencia de actualización, etc.) $\rightarrow$ ***reto distinto*** a generaciones anteriores.

# Big Data y NoSQL

- **Enormes volúmenes de datos**, y condiciones impuestas por las V's (disponibilidad, frecuencia de actualización, etc.) $\rightarrow$ ***reto distinto*** a generaciones anteriores.
- BD relacionales proporcionan potencia y robustez, funcionalidad y estandarización

# Big Data y NoSQL

- **Enormes volúmenes de datos**, y condiciones impuestas por las V's (disponibilidad, frecuencia de actualización, etc.) $\rightarrow$ ***reto distinto*** a generaciones anteriores.
- BD relacionales proporcionan potencia y robustez, funcionalidad y estandarización, pero ***necesitamos nuevas soluciones***.

# Big Data y NoSQL

- **Enormes volúmenes de datos**, y condiciones impuestas por las V's (disponibilidad, frecuencia de actualización, etc.) $\rightarrow$ ***reto distinto*** a generaciones anteriores.
- BD relacionales proporcionan potencia y robustez, funcionalidad y estandarización, pero ***necesitamos nuevas soluciones***.
- Cambio de enfoque: mientras que en `BD relacionales` prima la **atomicidad** y **consistencia** de datos

# Big Data y NoSQL

- **Enormes volúmenes de datos**, y condiciones impuestas por las V's (disponibilidad, frecuencia de actualización, etc.) $\rightarrow$ ***reto distinto*** a generaciones anteriores.
- BD relacionales proporcionan potencia y robustez, funcionalidad y estandarización, pero ***necesitamos nuevas soluciones***.
- Cambio de enfoque: mientras que en `BD relacionales` prima la **atomicidad** y **consistencia** de datos, en `NoSQL` (*Not only SQL*) se apuesta por la **escalabilidad** y **disponibilidad**.

# Big Data y NoSQL

- **Enormes volúmenes de datos**, y condiciones impuestas por las V's (disponibilidad, frecuencia de actualización, etc.) $\rightarrow$ ***reto distinto*** a generaciones anteriores.
- BD relacionales proporcionan potencia y robustez, funcionalidad y estandarización, pero ***necesitamos nuevas soluciones***.
- Cambio de enfoque: mientras que en `BD relacionales` prima la **atomicidad** y **consistencia** de datos, en `NoSQL` (*Not only SQL*) se apuesta por la **escalabilidad** y **disponibilidad**.En general, se reemplazan los principios ACID por los BASE, y teniendo presente las limitaciones que establece el teorema CAP.

# Big Data y NoSQL

- **Enormes volúmenes de datos**, y condiciones impuestas por las V's (disponibilidad, frecuencia de actualización, etc.) $\rightarrow$ ***reto distinto*** a generaciones anteriores.
- BD relacionales proporcionan potencia y robustez, funcionalidad y estandarización, pero ***necesitamos nuevas soluciones***.
- Cambio de enfoque: mientras que en `BD relacionales` prima la **atomicidad** y **consistencia** de datos, en `NoSQL` (*Not only SQL*) se apuesta por la **escalabilidad** y **disponibilidad**.En general, se reemplazan los principios ACID por los BASE, y teniendo presente las limitaciones que establece el teorema CAP.
- El modelo relacional implica ***muchas interrelaciones*** entre entidades, dificultando distribución entre múltiples nodos servidores $\rightarrow$ **barrera para la escalabilidad**.

# Big Data y NoSQL

- **Enormes volúmenes de datos**, y condiciones impuestas por las V's (disponibilidad, frecuencia de actualización, etc.) → ***reto distinto*** a generaciones anteriores.
- BD relacionales proporcionan potencia y robustez, funcionalidad y estandarización, pero ***necesitamos nuevas soluciones***.
- Cambio de enfoque: mientras que en `BD relacionales` prima la **atomicidad** y **consistencia** de datos, en `NoSQL` (*Not only SQL*) se apuesta por la **escalabilidad** y **disponibilidad**.En general, se reemplazan los principios ACID por los BASE, y teniendo presente las limitaciones que establece el teorema CAP.
- El modelo relacional implica ***muchas interrelaciones*** entre entidades, dificultando distribución entre múltiples nodos servidores → **barrera para la escalabilidad**.
- La necesidad de ***realizar joins*** sobre tablas con ***miles de millones de registros*** → **pérdida de eficiencia**.

# ACID

- **Atomicity**: cada transacción de la BD debe ser **atómica**, realizando **todo** lo que se le pida **o** bien **nada**. Si *algo falla*, la transacción completa se deshace, quedando el sistema en el estado previo al intento de ejecutarla.

# ACID

- **Atomicity**: cada transacción de la BD debe ser **atómica**, realizando **todo** lo que se le pida **o** bien **nada**. Si *algo falla*, la transacción completa se deshace, quedando el sistema en el estado previo al intento de ejecutarla.
- **Consistency**: se asegura que la base de datos quedará **siempre** en un **estado consistente**, pasando siempre de un estado válido a otro, según las reglas de negocio subyacentes. Si un elemento de una *transacción perturba la consistencia* de la BD, toda la transacción se cancela.

# ACID

- **Atomicity**: cada transacción de la BD debe ser **atómica**, realizando **todo** lo que se le pida **o** bien **nada**. Si *algo falla*, la transacción completa se deshace, quedando el sistema en el estado previo al intento de ejecutarla.
- **Consistency**: se asegura que la base de datos quedará **siempre** en un **estado consistente**, pasando siempre de un estado válido a otro, según las reglas de negocio subyacentes. Si un elemento de una *transacción perturba la consistencia* de la BD, toda la transacción se cancela.
- **Isolation**: el SGBD garantiza el **aislamiento entre** distintas **transacciones concurrentes** o simultáneas. Cada transacción debe *partir* del estado resultante del final de otra transacción, no viendo *nunca un estado intermedio* producido por la anterior.

# ACID

- **Atomicity**: cada transacción de la BD debe ser **atómica**, realizando **todo** lo que se le pida **o** bien **nada**. Si *algo falla*, la transacción completa se deshace, quedando el sistema en el estado previo al intento de ejecutarla.
- **Consistency**: se asegura que la base de datos quedará **siempre** en un **estado consistente**, pasando siempre de un estado válido a otro, según las reglas de negocio subyacentes. Si un elemento de una *transacción perturba la consistencia* de la BD, toda la transacción se cancela.
- **Isolation**: el SGBD garantiza el **aislamiento entre** distintas **transacciones concurrentes** o simultáneas. Cada transacción debe *partir* del estado resultante del final de otra transacción, no viendo *nunca un estado intermedio* producido por la anterior.
- **Durability**: una vez concluida una transacción, el **estado** resultante queda **permanentemente registrado** en el sistema *hasta* que una *transacción subsiguiente lo actualice*. Un *potencial fallo posterior* no puede hacer desaparecer al estado anterior, que debe ser persistente.

# BASE

- **Basic Availability**: la **base de datos**, y la consiguiente posibilidad de realizar **lecturas** y **escrituras** sobre la misma, deben estar **tan disponibles como sea posible**, *incluso en caso de fallos* (mediante un enfoque distribuido y con replicación, y usando tantos nodos como sea necesario). Esto *puede conducir incluso a inconsistencias* (con escrituras que podrían no quedar correctamente persistidas, o lecturas no accediendo correctamente a la última versión de los datos, que podría estar en un nodo caído).

# BASE

- **Basic Availability**: la **base de datos**, y la consiguiente posibilidad de realizar **lecturas y escrituras** sobre la misma, deben estar **tan disponibles como sea posible**, *incluso en caso de fallos* (mediante un enfoque distribuido y con replicación, y usando tantos nodos como sea necesario). Esto *puede conducir incluso a inconsistencias* (con escrituras que podrían no quedar correctamente persistidas, o lecturas no accediendo correctamente a la última versión de los datos, que podría estar en un nodo caído).
- **Soft state**: se **deja de garantizar la consistencia total**, tenemos una ***alta probabilidad de estar accediendo al estado actual, válido***, pero *no puede garantizarse*, por lo indicado antes.

# BASE

- **Basic Availability**: la **base de datos**, y la consiguiente posibilidad de realizar **lecturas y escrituras** sobre la misma, deben estar **tan disponibles como sea posible**, *incluso en caso de fallos* (mediante un enfoque distribuido y con replicación, y usando tantos nodos como sea necesario). Esto *puede conducir incluso a inconsistencias* (con escrituras que podrían no quedar correctamente persistidas, o lecturas no accediendo correctamente a la última versión de los datos, que podría estar en un nodo caído).
- **Soft state**: se **deja de garantizar la consistencia total**, tenemos una ***alta probabilidad de estar accediendo al estado actual, válido***, pero *no puede garantizarse*, por lo indicado antes.
- **Eventual consistency**: aunque *no se garantice la consistencia total*, se llega al **compromiso** de que se llegará al **estado válido**, consistente, **en algún momento** en el futuro. Se contrapone a la garantía de los principios ACID, que mediante consistencia y aislamiento garantiza los estados válidos en todo momento.

# Teorema CAP

En un sistema distribuido, solamente se pueden garantizar simultáneamente dos de las tres siguientes propiedades:

- **Consistencia**: todas las máquinas ven exactamente la **misma información** al **mismo tiempo**.

# Teorema CAP

En un sistema distribuido, solamente se pueden garantizar simultáneamente dos de las tres siguientes propiedades:

- **Consistencia**: todas las máquinas ven exactamente la **misma información** al **mismo tiempo**.
- **Disponibilidad**: cualquier petición recibida por un nodo activo del sistema debe obtener una **respuesta (no errónea)** en un **tiempo razonable**, *aunque* haya *otros nodos* en el sistema que *fallen* o no estén disponibles (garantice o no que se está devolviendo el último estado válido fruto de la escritura más reciente).

# Teorema CAP

En un sistema distribuido, solamente se pueden garantizar simultáneamente dos de las tres siguientes propiedades:

- **Consistencia**: todas las máquinas ven exactamente la **misma información** al **mismo tiempo**.
- **Disponibilidad**: cualquier petición recibida por un nodo activo del sistema debe obtener una **respuesta (no errónea)** en un **tiempo razonable**, *aunque* haya *otros nodos* en el sistema que *fallen* o no estén disponibles (garantice o no que se está devolviendo el último estado válido fruto de la escritura más reciente).
- **Tolerancia al particionamiento**: el **sistema** debe seguir **funcionando** *aunque algunos nodos no* se encuentren *disponibles* (aunque haya mensajes que hayan fallado en ciertos nodos o se encuentren retrasados).

# NoSQL

¿**Cuándo usar** bases de datos **NoSQL**? En las siguientes situaciones:

# NoSQL

¿**Cuándo usar** bases de datos **NoSQL**? En las siguientes situaciones:

- Volúmenes de datos **muy grandes**, más allá de los equipos convencionales, requiriendo escalabilidad horizontal.

# NoSQL

¿**Cuándo usar** bases de datos **NoSQL**? En las siguientes situaciones:

- Volúmenes de datos **muy grandes**, más allá de los equipos convencionales, requiriendo escalabilidad horizontal.
- Aplicaciones que **crezcan rápido** y **necesidades cambiantes** en cuanto a la estructura de los datos.

# NoSQL

¿**Cuándo usar** bases de datos **NoSQL**? En las siguientes situaciones:

- Volúmenes de datos **muy grandes**, más allá de los equipos convencionales, requiriendo escalabilidad horizontal.
- Aplicaciones que **crezcan rápido** y **necesidades cambiantes** en cuanto a la estructura de los datos.
- Almacenamiento de datos en **forma jerárquica o de grafo** o red, no datos planos interrelacionados.

# NoSQL

¿**Cuándo usar** bases de datos **NoSQL**? En las siguientes situaciones:

- Volúmenes de datos **muy grandes**, más allá de los equipos convencionales, requiriendo escalabilidad horizontal.
- Aplicaciones que **crezcan rápido** y **necesidades cambiantes** en cuanto a la estructura de los datos.
- Almacenamiento de datos en **forma jerárquica o de grafo** o red, no datos planos interrelacionados.
- Necesidad de **alta disponibilidad** y **tolerancia a fallos**.

# NoSQL

¿**Cuándo usar** bases de datos **NoSQL**? En las siguientes situaciones:

- Volúmenes de datos **muy grandes**, más allá de los equipos convencionales, requiriendo escalabilidad horizontal.
- Aplicaciones que **crezcan rápido** y **necesidades cambiantes** en cuanto a la estructura de los datos.
- Almacenamiento de datos en **forma jerárquica o de grafo** o red, no datos planos interrelacionados.
- Necesidad de **alta disponibilidad** y **tolerancia a fallos**.
- Requisito de **datos distribuidos**, **ubicuos** y a un coste razonable.

# NoSQL

¿**Cuándo usar** bases de datos **NoSQL**? En las siguientes situaciones:

- Volúmenes de datos **muy grandes**, más allá de los equipos convencionales, requiriendo escalabilidad horizontal.
- Aplicaciones que **crezcan rápido** y **necesidades cambiantes** en cuanto a la estructura de los datos.
- Almacenamiento de datos en **forma jerárquica o de grafo** o red, no datos planos interrelacionados.
- Necesidad de **alta disponibilidad** y **tolerancia a fallos**.
- Requisito de **datos distribuidos**, **ubicuos** y a un coste razonable.
- **Alta velocidad** para consulta y actualización constante por **número enorme** de accesos **concurrentes**

# NoSQL

¿**Cuándo usar** bases de datos **NoSQL**? En las siguientes situaciones:

- Volúmenes de datos **muy grandes**, más allá de los equipos convencionales, requiriendo escalabilidad horizontal.
- Aplicaciones que **crezcan rápido** y **necesidades cambiantes** en cuanto a la estructura de los datos.
- Almacenamiento de datos en **forma jerárquica o de grafo** o red, no datos planos interrelacionados.
- Necesidad de **alta disponibilidad** y **tolerancia a fallos**.
- Requisito de **datos distribuidos**, **ubicuos** y a un coste razonable.
- **Alta velocidad** para consulta y actualización constante por **número enorme** de accesos **concurrentes**

¿**Cuándo no** usarlas? Cuando la ***robustez***, ***integridad*** y ***estructura consistente*** sean lo principal, o se quiera contar con un estándar de consulta flexible como SQL.

# NoSQL - Tipos

**Clave-valor** Forma mas típica, como **HashMap** con cada elemento identificado por una clave única. Muy eficientes para lecturas y escrituras, y escalan fácilmente particionando los valores según su clave (*e.g.*, clave entre 1 y 1000 a un servidor, 1001 a 2000 a otro, etc.) Ejemplos: ***Redis***, ***DynamoDB*** de **Amazon**, ***Berkeley DB*** **de Oracle**, ***Riak***.

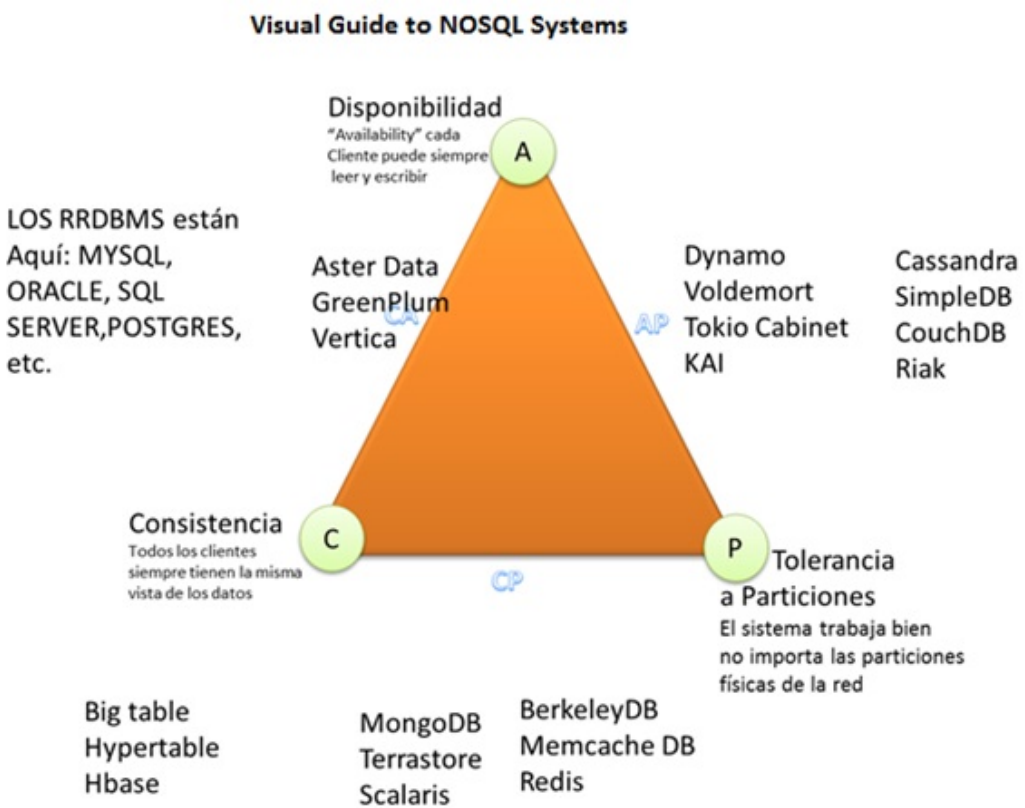
**Documentos** Almacenan información como un **documento** (similar a JSON o XML) y con una clave única. El fichero generado puede ser entendido, y el servidor puede hacer operaciones con ellos. Varias implementaciones permiten **consultas muy avanzadas**, y establecer relaciones sin joins. Ejemplos: ***MongoDB***, ***CouchDB***, ***Firestore***.

**Columnas** Guardan los valores en **columnas** en lugar de filas. Ganamos mucha velocidad en lecturas (al recuperar columnas concretas en lugar de toda la tabla), pero ineficiente para realizar escrituras. Se usa en aplicaciones con pocas escrituras en relación a las lecturas. Por ejemplo, ***Cassandra*** de **Facebook**, ***BigTable*** de **Google**, ***SimpleDB*** de **Amazon**, ***HBase***.

**Grafos** Almacenan información como **grafos**; las ***relaciones entre*** los ***nodos*** son lo más importante. Útiles para representar información de redes sociales. Más eficiente navegar entre estas relaciones, que pueden tener atributos, que en modelo relacional. Aprovechables si la información a modo de red. Ejemplo: ***Neo4j***, ***InfiniteGraph***.

# NoSQL y Teorema CAP

Ante la imposibilidad de cumplir todos los preceptos deseables a la vez, cada BD NoSQL opta por conjugar dos de los tres:



# Bibliografía I



Antonio Sarasa Cabezuelo
*Introducción a las bases de datos NoSQL usando Cassandra*.
Madrid : Ediciones Complutense (2019)
ISBN: 978-84-669-3638-5


Gaurav Vaish
*Getting started with NoSQL - your guide to the world and technology of NoSQL*.
Packt Publishing (2013)
ISBN: 978-1-4842-4890-4


Andreas Meier, Michael Kaufmann
*SQL and NoSQL Databases - Models, Languages, Consistency Options and Architectures for Big Data Management*.
Springer Fachmedien Wiesbaden (2019)
ISBN: 978-1-4842-4890-4

# Bibliografía II



Subhashini Chellappan, Dharanitharan Ganesan
*MongoDB Recipes - With Data Modeling and Query Building Strategies*.
Apress (2020)
ISBN: 978-1-4842-4890-4


Aitor Medrano
*Almacenamiento de datos. NoSQL*.
Curso de especialización *Inteligencia Artificial y Big Data* (2023)
IES Severo Ochoa de Elche