



# Tema 7. Introducción a las bases de datos noSQL

El funcionamiento de casi todas las aplicaciones web se basa en el uso de una **base de datos**.

Hasta hace poco las bases de datos que se utilizaban eran bases de datos relacionales y se trabajaba con SQL (MySQL, Oracle, SQL Server, ...)

Hace poco han aparecido las otro tipo de bases de datos que reciben el nombre de bases de datos NoSQL(Not Only SQL).

El término NoSQL (aunque ya se había utilizado antes) toma impulso con la aparición de la web 2.0, con la llegada de Facebook, Twitter o YouTube, donde cualquier usuario podía subir contenido, provocando así un crecimiento exponencial de los datos.



# 90%

de los datos  
almacenados en el  
mundo se han  
creado en los  
últimos

# 2 años

ScienceDaily, 22 May 2013

Por un lado, el volumen de datos ha crecido tremendamente y, además, esta información es información no estructurada, información que no encaja en una tabla.

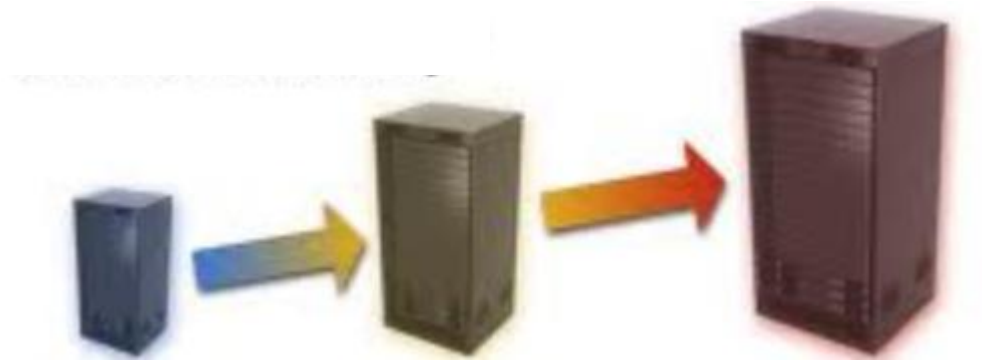
Por otro lado, los usuarios quieren respuestas inmediatas.

Las bases de datos relacionales no sirven para manejar este volumen de datos.

Los sistemas tienen que **escalar** para poder ser más grandes y más rápidos.

Existen 2 maneras de escalar:

Escalado vertical



Escalado horizontal

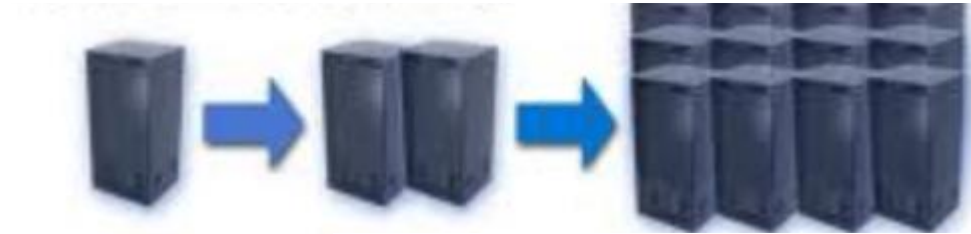


## Escalado vertical



- Es caro
- tiene un límite,
- y las regulaciones de ciertos países no te permiten guardar los datos fuera del país, por lo que a veces no puedes hacerlo.

## Escalado horizontal



Hay que distribuir los datos

### Replicación

(todos los nodos deben guardar la misma información)

Distribuir datos en BD relacionales es complicado, hay que mantener las propiedades


#### **ACID:**

hay que hacer un commit de 2 fases, hay que esperar a que todos los nodos de la red contesten. Unas transacciones se quedan bloqueadas esperando a que otras terminen

### Sharding

(los nodos tienen datos distintos)

- También es difícil llevarlo a la práctica en las bases de datos relacionales

Hacemos un  en el tema NoSQL para explicar una característica fundamental de las BD Relacionales

Las BD Relacionales cumplen las propiedades **ACID**.



# CONCEPTOS BÁSICOS

## TRANSACCIÓN

Es un conjunto de órdenes que se ejecutan sobre una base de datos constituyendo una **unidad lógica de trabajo**.

Si la transacción termina bien, todas las modificaciones de los datos realizadas durante la transacción se confirman y se convierten en una parte permanente de la base de datos.

Si en una transacción se producen errores, debe cancelarse completamente todas las acciones llevadas a cabo por la transacción.

# TRANSACCIÓN. Ejemplo clásico

CUENTAS

num	saldo	...
...	...	...
1245	5700	
...	...	...
3567	1200	
...	...	...



1000 euros

UPDATE cuentas SET saldo= saldo-1000 WHERE num=1245;

UPDATE cuentas SET saldo= saldo+1000 WHERE num=3567;

# TRANSACCIÓN

CUENTAS		
num	saldo	...
...	...	...
1245	5700	
...	...	...
3567	1200	
...	...	...

UPDATE cuentas SET saldo= saldo-1000 WHERE num=1245;



CUENTAS		
num	saldo	...
...	...	...
1245	4700	
...	...	...
3567	1200	
...	...	...

# TRANSACCIÓN

CUENTAS		
num	saldo	...
...	...	...
1245	5700	
...	...	...
3567	1200	
...	...	...



**UPDATE cuentas SET saldo= saldo-1000 WHERE num=1245;**  
**UPDATE cuentas SET saldo= saldo+1000 WHERE num=3567;**

Dos posibilidades

Posibilidad 1

num	saldo	...
...	...	...
1245	4700	
...	...	...
3567	2200	
...	...	...

Posibilidad 2

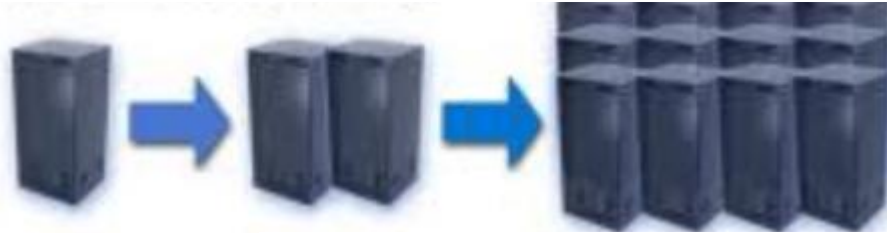
num	saldo	...
...	...	...
1245	5700	
...	...	...
3567	1200	
...	...	...

# CONCEPTOS BÁSICOS

## Propiedades de las transacciones

- A** **Atomicidad:** una transacción es indivisible. O se efectúan todas las operaciones o debe quedar la BD como si no se hubiese efectuado ninguna.
- C** **Consistencia:** después de ejecutarse una transacción, la BD debe quedar en un estado correcto (sin necesidad de que sea correcto entre operaciones de una transacción).
- I** **Aislamiento (Isolation):** el comportamiento de una transacción no se ve afectado porque otras transacciones se estén ejecutando a la vez.
- D** **Durabilidad:** los efectos de una transacción son permanentes tras su grabación.

## Escalado horizontal



Hay que distribuir los datos

Retomamos el tema

### Replicación

(todos los nodos deben guardar la misma información)

Distribuir datos en BD relacionales es complicado, hay que mantener las propiedades

#### **ACID:**

hay que hacer un commit de 2 fases, hay que esperar a que todos los nodos de la red contesten. Unas transacciones se quedan bloqueadas esperando a que otras terminen

### Sharding

(los nodos tienen datos distintos)

- También es difícil llevarlo a la práctica en las bases de datos relacionales

En este contexto Google y Amazon se dan cuenta que las bases de datos relacionales no responden a sus necesidades ante el volumen de información que mueven, y deciden crear sus propias bases de datos (BigTable y DynamoDB), cada uno la suya para dar respuesta a sus necesidades.

El éxito de estos sistemas hizo que se iniciara el desarrollo de otros sistemas de bases de datos de código abierto y características similares (Cassandra, MongoDB, DynamoDB, HBase, Redis, ...)

Estas bases de datos tienen varias características en común, que las diferencian de las bases de datos relacionales:

- **arquitectura distribuida:** pensadas para trabajar en múltiples servidores que se comunican entre sí.
- **Pueden manejar gran cantidad de datos.**
- **No genera cuellos de botella:** los sistemas relacionales necesitan transcribir cada sentencia para poder ser ejecutada, y la ejecución de muchas sentencias complejas puede ralentizar el sistema.
- **Flexibilidad de esquema:** a diferencia de las relacionales donde el esquema de la base de datos es fijo, estático, en estas bases de datos existe el esquema no es fijo, es un esquema dinámico.
- **No ACID → eventualmente consistentes (BASE)**



## **BASE** (Basically Available Soft-state Eventual Consistency)

NoSQL no garantiza las propiedades ACID, permite la **Consistencia Eventual**.

En un sistema relacional el commit de una transacción EXIGE que este cambio se replique de forma inmediata en todos los nodos (sistemas transaccionales). Esto no es así en los sistemas NoSQL, esto se conoce también como BASE (coherencia eventual flexible básicamente disponible).

El principio **BASE** facilita enormemente el escalado horizontal.

El principio **BASE** permite que las operaciones sean mucho más rápidas que los sistemas que garantizan las propiedades ACID

¿Voy a usar una base de datos NoSQL si mis datos pueden no ser consistentes?

Depende ...

***Si es un sistema bancario***

¿le gustaría a un cliente no ver reflejada (aunque sea momentáneamente) en su saldo un ingreso que le han realizado?

***Si es una red social***

¿tan importante es para un usuario ver un like más o menos en este momento?

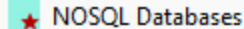
En 2009 se celebraba en San Francisco una conferencia sobre: Sistemas distribuidos, de código abierto y no relacionales (académicamente no existía un término para referirse a ese conjunto de características).


En twitter se refirieron al evento con el hashtag #nosql (el término nosql ya se había utilizado antes en 1998)

El término fue ampliamente aceptado y se comenzó a usar para referirse a bases de datos con esas características.


Se decidió que podía referirse a “not only” SQL como una idea de que no es una batalla, pueden coexistir ambos tipos de bases de datos.


# ¿Cuántos sistemas de bases de datos NoSQL existen actualmente?


 NOSQL Databases



[nosql-database.org](https://nosql-database.org)







Your Ultimate Guide to the  
Non-Relational Universe!

[including a **historic** [Archive](#) 2009-2011]  
**News Feed** covering some changes [here](#) !

---

**NOSQL DEFINITION:**Next Generation Databases mostly addressing some of the points: being **non-relational, distributed, open-source** and **horizontally scalable**.


The original intention has been **modern web-scale databases**. The movement began early 2009 and is growing rapidly. Often more characteristics apply such as: **schema-free, easy replication support, simple API, eventually consistent / BASE** (not ACID), a **huge amount of data** and more. So the misleading term "*nosql*" (the community now translates it mostly with "**not only sql**") should be seen as an alias to something like the definition above. [based on 7 sources, 15 constructive feedback emails (thanks!) and 1 disliking comment. Agree / Disagree? [Tell](#) me so! By the way: this is a strong definition and it is out there here since 2009!]


**NoSQL RELATED EVENTS:**

- June 26-27 2018 **MongoDB World** [»](#)

Register your event 4free: [»](#)

**NoSQL ARCHIVE**



  
the **multi-model** NoSQL DB

---

**LIST OF NOSQL DATABASES** [currently >225]

---

Core NOSQL Systems: [Mostly originated out of a Web 2.0 need]

# Tipos de Bases de Datos noSQL

- **Clave-Valor**
- **Columnas**
- **Documentales**
- **Grafos**

# Bases de Datos NoSQL **clave-valor**

Almacena cualquier tipo de objeto (número, cadena de caracteres, array, json,...)**Valor** y accede a estos objetos por un identificador. **Clave**

Clave	Valor
23567	346
23568	{depart1:"Lenguajes y Sistemas, depart2:"Física Aplicada",depart3:"Ciencias"}
23590	45.22
23591	"El Quijote"
23596	"Vaya usted a saber"

- Muy rápida la búsqueda por clave ya que se crea un índice de tipo hash para la clave.
- La búsqueda por valores de la base de datos no la puede realizar. No hay una patrón en este contenido. Hay que ir recuperando todos los contenidos y entonces ir analizándolos.

*Ejemplo: DynamoDB (Amazon)*

# Bases de Datos NoSQL **de columna**

Es como si pusiéramos lo que serían las filas en columnas

ID	nombre	fecha_nac	num_hijos
1	Juan Robles	22/05/1985	2
2	María Alarcón	23/10/1980	3
3	Pedro Vigo	12/08/2000	0
4	Rosa Blas	04/06/1977	2

Tabla CLIENTE  
en M. Relacional

- Se almacena por filas.
- Cada lectura recupera una fila completa

¿Cuál es la idea sobre cómo se almacena en una BD Relacional?

1	Juan Robles	22/05/1985	2	2	María Alarcón	23/10/1980	3	3	Pedro Vigo	12/08/2000	0	4	Rosa Blas	04/06/1977	2
---	-------------	------------	---	---	---------------	------------	---	---	------------	------------	---	---	-----------	------------	---

# Bases de Datos NoSQL de columna

ID	nombre	fecha_nac	num_hijos
1	Juan Robles	22/05/1985	2
2	María Alarcón	23/10/1980	3
3	Pedro Vigo	12/08/2000	0
4	Rosa Blas	04/06/1977	2

¿Cuál es la idea sobre cómo se almacena en una BD NoSQL de

1	2	3	4	Juan Robles	María Alarcón	Pedro Vigo	Rosa Blas	22/05/1985	23/10/1980	12/08/2000	04/06/1977	2	3	0	2
---	---	---	---	-------------	---------------	------------	-----------	------------	------------	------------	------------	---	---	---	---

- ¿Dónde viene bien esta estructura?

En BD analíticas

¿Cuál es el número medio de hijos de mis clientes?

- Desventajas: se ralentizan las operaciones de borrado y modificación

*Ejemplo de BD columna: BigTable (Google)*



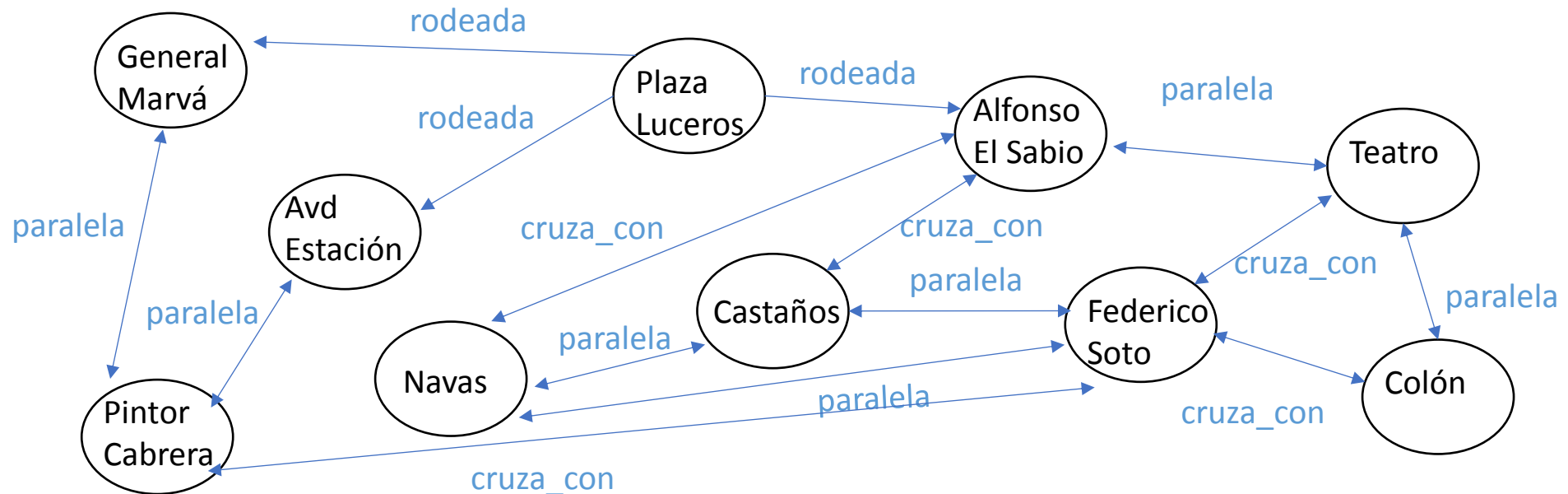
# Bases de Datos NoSQL de grafo

La información se representa como nodos de un grafo y sus relaciones son las aristas del mismo.

Este tipo de bases de datos es muy útil para información muy interconectada entre sí y donde esta conexión es tan importante como la información en sí.

Son muy muy eficientes para ir pasando de un nodo a otro

Se usan en casos muy específicos: redes sociales, tráfico aéreo, distribución eléctrica...



*Ejemplo de BD grafo: Neo4j*

# Bases de Datos NoSQL **documentales**

NO se refiere a una BD que guarda pdf's o documentos Word, ...

Se refiere a una BD que guarda estructuras que a su vez pueden tener muchos niveles de información.

Para cada entrada no tenemos las típicas columnas de estructura fija, cada entrada tendrá su propia estructura.

Cada una de estas estructuras que contiene información desestructurada se llaman **documento**.

Muchas bases de datos documentales representan estas estructuras como objetos JSON.

# Bases de Datos NoSQL **documentales**

```
{
Nombre:      "Departamento de Lenguajes y Sistemas Informáticos",

Empleados:   70,

Asignaturas: [ "Diseño de Bases de Datos", "Fundamentos de Bases de
               Datos", "Programación", "Gestión de la Información" ],

Dirección:   {      calle:      "Carretera de San Vicente",
               número:      2,
               codigopostal:  03690
             },

Teléfono:    {      centralUA:  {      número:      965903400,
                                   extensión:      3972
                                 }
               directo:      [965903466, 965943211]
             }
}
```

# Bases de Datos NoSQL **documentales**

```
{
  Nombre:      "Departamento de Lenguajes y Sistemas Informáticos",
  Empleados:   70,
  Asignaturas: [ "Diseño de Bases de Datos", "Fundamentos de Bases de
                  Datos", "Programación", "Gestión de la Información" ],
  Dirección:   {
                  calle:      "Carretera de San Vicente",
                  número:     2,
                  codigopostal: 03690
                },
  Teléfono:    {
                  centralUA:   {
                                número:      965903400,
                                extensión:    3972
                              },
                  directo:     [965903466, 965943211]
                }
}
```

**campos**

**Valor numérico**

**Cadena caracteres**

**Array**

**Subdocumentos**

# Bases de Datos NoSQL documentales

```
{
  Nombre: "Departamento de Lenguajes y Sistemas
  Informáticos",
  Empleados: 70,
  Asignaturas: [ "Diseño de Bases de Datos",
  "Fundamentos de Bases de Datos", "Programación",
  "Gestión de la Información" ],
  Dirección: { calle: "Carretera de San Vicente",
  número: 2,
  codigopostal: 03690
  },
  Teléfono: { centralUA: { número: 965903400,
  extensión: 3972
  },
  directo: [965903466, 965943211]
  }
}
```

```
{
  Nombre: "Departamento Física Aplicada"
  Empleados: 25,
  Asignaturas: [ "Física1 ", Física 2"],
  Dirección: { calle: "Carretera de San Vicente",
  número: 2,
  codigopostal: 03690
  },
  Teléfono: { centralUA: { número: 965903400,
  extensión: 3977
  },
  directo: [965903477, 965943220]
  }
}
```

```
{
  Nombre: "Departamento Biología"
  Empleados: 40,
  Asignaturas: [ "Biología 1 ", Biología 2"],
  Dirección: { calle: "Carretera de San Vicente",
  número: 2,
  codigopostal: 03690
  },
  Teléfono: { centralUA: { número: 965903400,
  extensión: 3988
  },
  directo: 965907766
  }
}
```

Los **documentos** se agrupan formando **colecciones**.

# Bases de Datos NoSQL **documentales**

Permiten hacer indexación

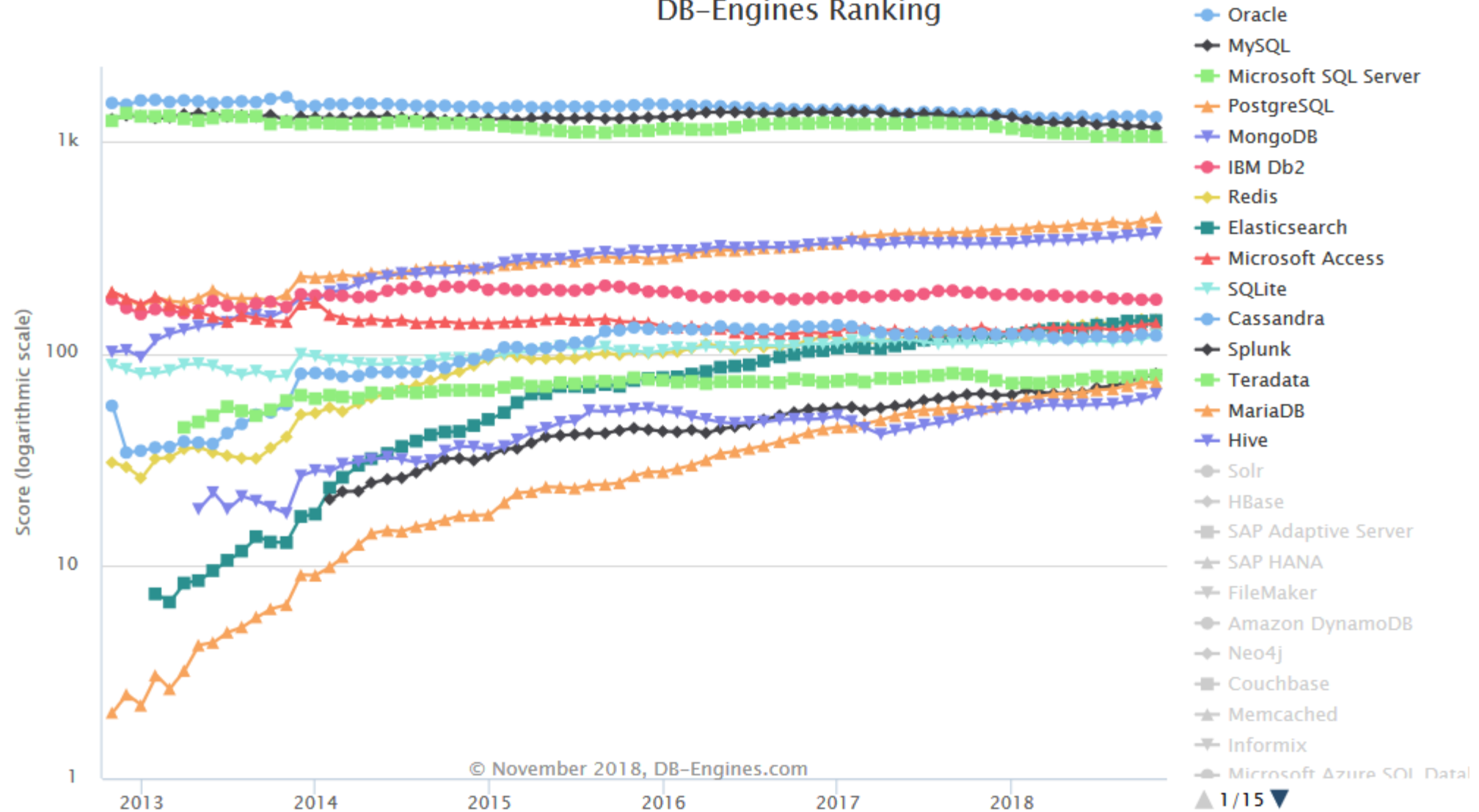
Permiten hacer búsquedas por valor a cualquier nivel.

Permiten agrupamientos

*Ejemplo: MongoDB, CouchDB*

Conociendo ya algunas características de las bases de datos NoSQL,  
¿se usan tanto como las relacionales?

## DB-Engines Ranking



[https://db\\_engines.com/en/ranking](https://db_engines.com/en/ranking)



350 systems in ranking, December 2019

Rank			DBMS	Database Model	Score		
Dec 2019	Nov 2019	Dec 2018			Dec 2019	Nov 2019	Dec 2018
1.	1.	1.	Oracle +	Relational, Multi-model i	1346.39	+10.33	+63.17
2.	2.	2.	MySQL +	Relational, Multi-model i	1275.67	+9.38	+114.42
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model i	1096.20	+14.29	+55.86
4.	4.	4.	PostgreSQL +	Relational, Multi-model i	503.37	+12.30	+42.74
5.	5.	5.	MongoDB +	Document, Multi-model i	421.12	+7.94	+42.50
6.	6.	6.	IBM Db2 +	Relational, Multi-model i	171.35	-1.25	-9.40
7.	7.	↑ 8.	Elasticsearch +	Search engine, Multi-model i	150.25	+1.85	+5.55
8.	8.	↓ 7.	Redis +	Key-value, Multi-model i	146.23	+1.00	-0.59
9.	9.	9.	Microsoft Access	Relational	129.47	-0.60	-10.04
10.	10.	↑ 11.	Cassandra +	Wide column	120.71	-2.52	-1.10
11.	11.	↓ 10.	SQLite +	Relational	120.36	-0.66	-2.65
12.	12.	12.	Splunk	Search engine	90.53	+1.46	+8.34
13.	13.	↑ 14.	MariaDB +	Relational, Multi-model i	86.79	+1.22	+9.53
14.	14.	↑ 15.	Hive +	Relational	86.05	+1.83	+18.67
15.	15.	↓ 13.	Teradata +	Relational, Multi-model i	78.49	-1.86	-0.67

[https://db\\_engines.com/en/ranking](https://db_engines.com/en/ranking)

Ya que de los sistemas de bases de datos **noSQL**, el **más utilizado** es **MongoDB**, será el que veamos un poco más en profundidad.



# Tema 7. Introducción a las bases de datos noSQL