BASES DE DATOS

Bases de datos NoSQL

BASES DE DATOS

Bases de datos NoSQL

- Introducción
- Tipos
- MongoDB
  - Formato JSON
  - Comandos básicos
  - DDL
  - DML

Introducción

Las bases de datos NoSQL (*Not only SQL*) son bases de datos no relacionales.

- Utilizan estructuras alternativas para almacenar los datos.
- Se popularizaron desde el año 2010.
- No son un sustito del modelo relacional, si no una alternativa.

Introducción

Empresas como Google, Amazon, Twitter, Facebook necesitan tratar con enormes cantidades de datos.

Las bases de datos relacionales no ofrecen el rendimiento que necesitan.

- Priorizan la consistencia e integridad de los datos sobre la velocidad.
- Su escalabilidad no es la esperada.

Introducción

Empresas como Google, Amazon, Twitter, Facebook necesitan tratar con enormes cantidades de datos.

Comienzan a hacer uso o incluso desarrollan sus propias bases de datos:

Twitter, Facebook

Uber, FourSquare, SourceForge

Flickr, Instagram, Github

eBay, Infojobs

Yahoo, Adobe

Introducción

Características

Las bases de datos NoSQL (*Not only SQL*) son bases de datos no relacionales.

- No organizan los datos en relaciones (tablas).
- Priorizan la velocidad sobre la consistencia de los datos.
- Priorizan la operaciones de lectura y búsqueda sobre las de actualización o escritura.
- Centrados en ofrecer una mayor escalabilidad y flexibilidad de esquema.

Tipos

Clave-Valor

- Se basa en la estructura de datos del tipo diccionario.
- Cada clave tiene un valor asociado.
- La recuperación de datos es muy rápida.

Phone directory

Key	Value
Paul	(091) 9786453778
Greg	(091) 9686154559
Marco	(091) 9868564334

MAC table

Key	Value
10.94.214.172	3c:22:fb:86:c1:b1
10.94.214.173	00:0a:95:9d:68:16
10.94.214.174	3c:1b:fb:45:c4:b1

Tipos

Columnar

- Organiza los datos por columnas en vez de filas con el objeto de realizar menos accesos al disco.

Relacional

Número	Apellido	Nombre	Clave
1.	Skywalker	Luke	3FN-Z768
2	Kenobi	Obi-wan	7TR-K345
3	Organa	Leia	8NN-R266

```
1, Skywalker, Luke, 3FN-Z768; 2, Kenobi, Obi-wan, 7TR-K345; 3, Organa, Leia, 8NN-R266
```

Columnar

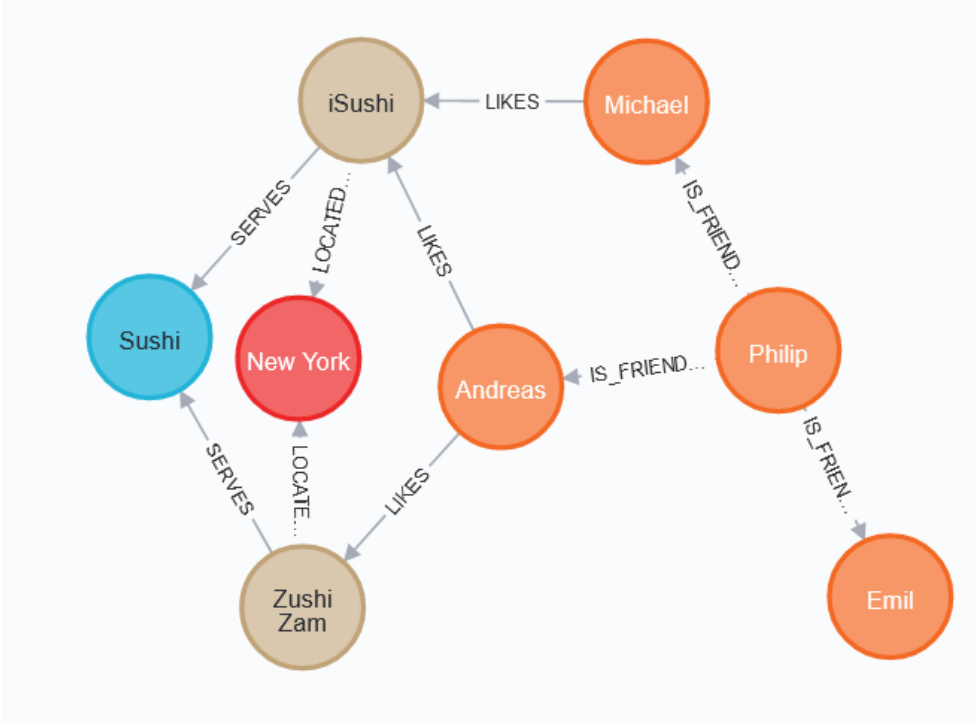
Número	1.	2.	3.
Apellido	Skywalker	Kenobi	Organa
Nombre	Luke	Obi-wan	Leia
Clave	3FN-Z768	7TR-K345	8NN-R266

```
1, 2, 3; Skywalker, Kenobi, Organa; Luke, Obi-wan, Leia; 3FN-Z768, 7TR-K345, 8NN-R266
```

Tipos

Grafo

- Los datos se representan como nodos de un grafo y sus relaciones como las aristas entre los nodos.



Tipos

Documental

- Los datos se almacenan en documentos utilizando formatos como JSON o XML.

```
{
    “nombre”: “Juan”,
    “edad”: 25
    “dirección”:
         {
              “ciudad”: “Barcelona”
         },
    “aficiones”:[
         {“nombre”: “Fútbol” },
         {“nombre”: “Esquí” }
    ]
}
```

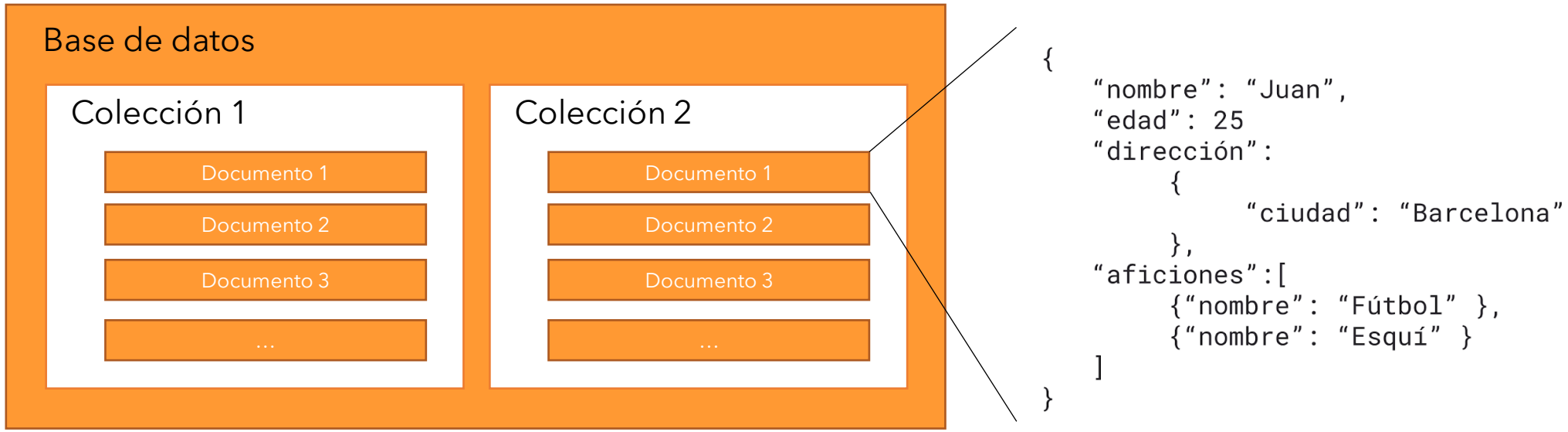
- Permite realizar búsquedas por clave-valor.
- También se pueden hacer búsquedas más avanzadas por el contenido del documento.

Relacional VS NoSQL

Relacional		NoSQL	
Ventajas	Inconvenientes	Ventajas	Inconvenientes
Madurez de la tecnología.	Menor flexibilidad a la hora de modificar el esquema.	Mayor flexibilidad en el esquema.	Consistencia eventual: No asegura que los datos leídos estén actualizados.
Aseguran consistencia e integridad.	Prioriza la consistencia e integridad sobre la velocidad.	Mayor escalabilidad.	Dificultad para realizar consultas complejas.
Estándares bien definidos (SQL).	Requieren mayores recursos hardware.	Priorizan la velocidad sobre la integridad y consistencia de datos.	Aplicaciones cliente más complejas.
Sencillez del modelo de datos.		Requieren menos recursos hardware.	
Aplicaciones cliente más sencillas.		Uso de formatos fácilmente integrables en aplicaciones (JSON, XML).	

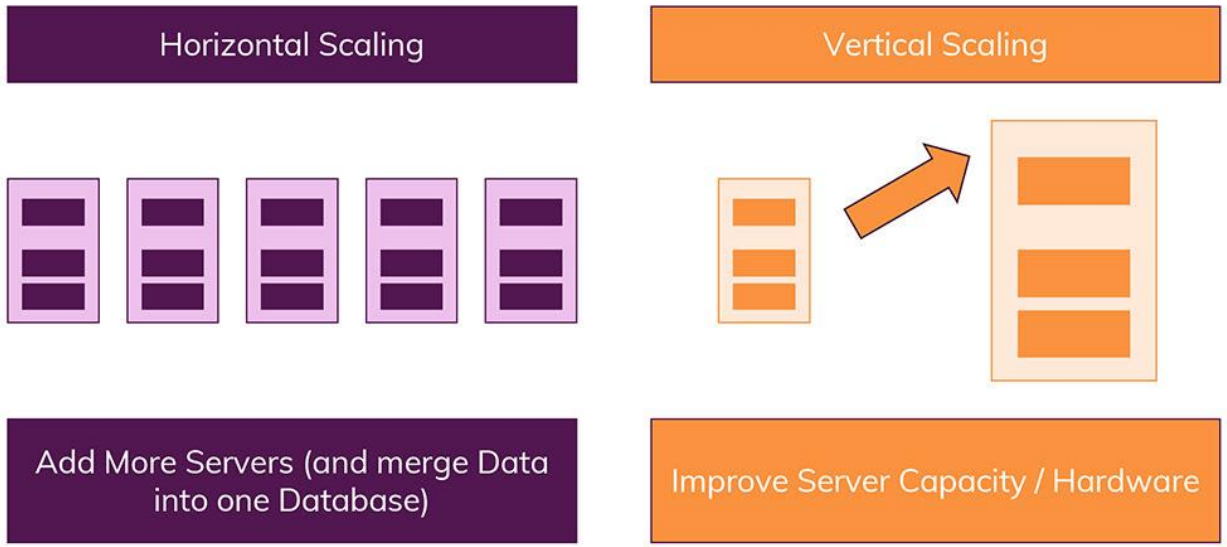
MongoDB

- Sistema Gestor de Base de Datos NoSQL.
- Orientado a documentos (JSON).
- Los documentos se organizan en colecciones.



MongoDB

- Esquema flexible: no hay que definir la estructura de las colecciones.
- Base de datos distribuida:
  - Escalabilidad horizontal.
  - Alta disponibilidad.



MongoDB

JSON

- La creación de un objeto JSON se indica mediante el uso de llaves {}.
- Un objeto se compone de duplas *identificador:valor*
- El identificador siempre va entre comillas.
- El valor no lleva comillas si es de tipo numérico, *booleano* o *null*.

```
{
   "nombre": "Pedro"
}
```

MongoDB

JSON

Si el objeto tiene más de una dupla se separan por comas.

```
{
   "latitude": 40.416875,
   "longitude": -3.703308,
   "city": "Madrid",
   "description": "Puerta del Sol"
}
```

```
{
   "id": 1,
   "nombre": "Galaxy S21“,
   "precio": 650.32
}
```

MongoDB

JSON

El valor de una dupla puede ser otro objeto:

```
{
   "nombre": "Alicante",
   "habitantes": 331577,
   "provincia" : {
      "nombre": "Alicante",
      "habitantes": 1863000,
      "superficie": 5816
   }
}
```

MongoDB

JSON

El valor de una dupla puede ser otro objeto:

```
{
   "nombre": "Alicante",
   "habitantes": 331577,
   "provincia": {
      "nombre": "Alicante",
      "habitantes": 1863000,
      "superficie": 5816,
      "comunidad": {
         "nombre": "Comunidad Valenciana"
      }
   }
}
```

MongoDB

JSON

```
{
   "nombre": "Alicante",
   "habitantes": 331577,
   "provincia": {
      "nombre": "Alicante",
      "habitantes": 1863000,
      "superficie": 5816,
      "comunidad": {
         "nombre": "Comunidad Valenciana"
      }
   },
   "alcalde": {
      "nombre": "Luis José Barcala",
      "partido": "PP"
   }
}
```

MongoDB

JSON

Podemos utilizar *arrays* para almacenar datos del mismo tipo:

```
{
   "tienda": "Leroy Merlin",
   "articulos": [
      "Ventilador de techo",
      "Puerta Lucerna",
      "Barbacoa para carbón"
   ]
}
```

MongoDB

JSON

También podemos utilizar tipos *booleanos* (*true*, *false*):

```
{
    "tienda": "Leroy Merlin",
    "articulos": [
        {
         "nombre": "Ventilador de techo",
         "precio": 199.0,
         "disponible": true
        },
        {
         "nombre": "Puerta Lucerna",
         "precio": 159.0
        }
    ]
}
```

MongoDB

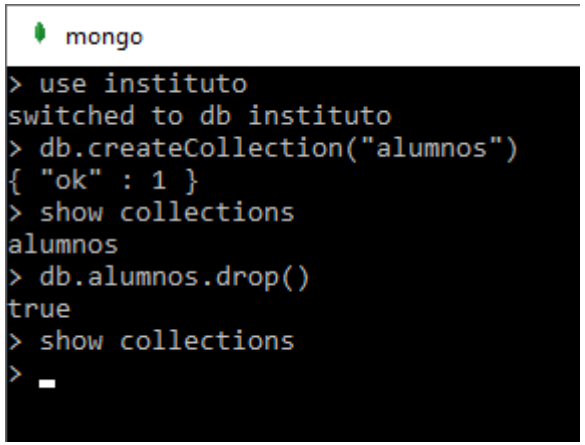
Comandos

Comando	Significado
`show dbs`	Muestra las bases de datos.
`use basededatos`	Para acceder a la base de datos llamada *basededatos*.
`db`	Muestra la base de datos actual.

MongoDB

DDL

Comando	Significado
`db.createCollection("nombreColeccion")`	Crea una colección llamada *nombreColección*
`db.nombreColeccion.drop()`	Elimina la colección *nombreColeccion*
`show collections`	Mostrar las colecciones



MongoDB

DML: Inserciones

Inserción de un único documento:

`db.`*`nombreColeccion`*`.insertOne(`*`documento`*`)`

Donde:

- *nombreColeccion* es el nombre de la colección
- *documento* el documento JSON que se quiere insertar.

```
> db.alumnos.insertOne({"dni": "20000000A", "nombre": "Juan", "apellido": "Sánchez"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("609999668ca7b4ef66b1f025")
}
```

MongoDB

DML: Inserciones

Inserción de múltiples documentos:

`db.nombreColeccion.insertMany([documento1, documento2, …])`

Donde:

- *nombreColeccion* es el nombre de la colección
- *documento1, documento2, …* los documento JSON que se quieren insertar.

```
> db.alumnos.insertMany([{"dni": "30000000B", "nombre": "Sonia", "apellido": "Herrera"},
... {"dni": "40000000C", "nombre": "Marta", "apellido": "Mendez"}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("60999d31be7c62e058ffba06"),
                ObjectId("60999d31be7c62e058ffba07")
        ]
}
```

MongoDB

DML: Consultas simples

Consulta de todos los documentos de una colección:

`db.nombreColeccion.find()`

```
> db.alumnos.find()
{ "_id" : ObjectId("609999668ca7b4ef66b1f025"), "dni" : "20000000A", "nombre" : "Juan", "apellido" : "Sánchez" }
{ "_id" : ObjectId("60999d31be7c62e058ffba06"), "dni" : "30000000B", "nombre" : "Sonia", "apellido" : "Herrera" }
{ "_id" : ObjectId("60999d31be7c62e058ffba07"), "dni" : "40000000C", "nombre" : "Marta", "apellido" : "Mendez" }
```

MongoDB

DML: Modificaciones

Modificar múltiples documentos que cumplan una condición:

db.*nombreColeccion*.updateMany(*filtro*, *acción*)

```
> db.alumnos.updateMany({"dni": "20000000A"}, {$set:{"nombre": "Pedro"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.alumnos.find()
{ "_id" : ObjectId("609999668ca7b4ef66b1f025"), "dni" : "20000000A", "nombre" : "Pedro", "apellido" : "Sánchez" }
{ "_id" : ObjectId("60999d31be7c62e058ffba06"), "dni" : "30000000B", "nombre" : "Sonia", "apellido" : "Herrera" }
{ "_id" : ObjectId("60999d31be7c62e058ffba07"), "dni" : "40000000C", "nombre" : "Marta", "apellido" : "Mendez" }
```

En el ejemplo anterior:

- El filtro es todos los alumnos con DNI 20000000A.
- La acción utilizada es **$set** que permite indicar qué valores se sustituirán.

MongoDB

DML: Modificaciones

Modificar todos los documentos de la colección:

`db.nombreColeccion.updateMany({}, acción)`

```
> db.alumnos.updateMany({}, {$set:{"nota_media": 5.0}})
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
> db.alumnos.find()
{ "_id" : ObjectId("609999668ca7b4ef66b1f025"), "dni" : "20000000A", "nombre" : "Pedro", "apellido" : "Sánchez",
 "nota_media" : 5 }
{ "_id" : ObjectId("60999d31be7c62e058ffba06"), "dni" : "30000000B", "nombre" : "Sonia", "apellido" : "Herrera",
 "nota_media" : 5 }
{ "_id" : ObjectId("60999d31be7c62e058ffba07"), "dni" : "40000000C", "nombre" : "Marta", "apellido" : "Mendez",
"nota_media" : 5 }
```

MongoDB

DML: Modificaciones

Ejemplo: Modifica el nombre del alumno con DNI 20000000A para que pase a ser Rodrigo. Modifica también su nota media para que pase a ser 6.2

```
> db.alumnos.updateMany({"dni": "20000000A"}, {$set: {"nombre": "Rodrigo", "nota_media": 6.2}})
```

```
> db.alumnos.find()
{ "_id" : ObjectId("609999668ca7b4ef66b1f025"), "dni" : "20000000A",
"nombre" : "Rodrigo", "apellido" : "Sánchez", "nota_media" : 6.2 }
{ "_id" : ObjectId("60999d31be7c62e058ffba06"), "dni" : "30000000B",
"nombre" : "Sonia", "apellido" : "Herrera", "nota_media" : 5 }
{ "_id" : ObjectId("60999d31be7c62e058ffba07"), "dni" : "40000000C",
"nombre" : "Marta", "apellido" : "Mendez", "nota_media" : 5 }
```

MongoDB

DML: Borrado

Inserción de un único documento:

`db.`*`nombreColeccion`*`.deleteMany(`*`filtro`*`)`

Ejemplo: Borrado de todos los alumnos llamados Rodrigo.

```
> db.alumnos.deleteMany({"nombre": "Rodrigo"})
```

```
> db.alumnos.find()
{ "_id" : ObjectId("60999d31be7c62e058ffba06"), "dni" : "30000000B",
"nombre" : "Sonia", "apellido" : "Herrera", "nota_media" : 5 }
{ "_id" : ObjectId("60999d31be7c62e058ffba07"), "dni" : "40000000C",
"nombre" : "Marta", "apellido" : "Mendez", "nota_media" : 5 }
```