

Maestría en Ciencia de Datos

Facultad de Ingeniería, Diseño y Ciencias



INFRAESTRUCTURA Y ARQUITECTURA DE TI

Ángela Villota Gómez
apvillota@icesi.edu.co

Mónica Rojas
mmrojas@icesi.edu.co



UNIDAD 2 - MODELOS NO SQL



1. Explicar de forma general las características de los modelos NoSQL tales como orientados a documentos, a columnas, y a grafos entre otros.
2. Escribir consultas que permitan hacer tareas de ETL sobre los modelos NoSQL

OBJETIVO ESPECÍFICOS



1

INTRODUCCIÓN

Características generales
BD NoSQL

3

PRÁCTICA MONGODB

Instrucciones básicas:
Inserción, modificación y
consultas

2

BD ORIENTADAS A DOCUMENTOS

Algunas generalidades.
MongoDB

4

CIERRE

Actividad

INTRODUCCIÓN BD NOSQL



BD RELACIONALES

- Surgieron en 1970.

- Almacenamiento en tablas.
- Lenguaje SQL
- Joins, índices
- Integridad de los datos
- Propiedades ACID
- Modelo centralizado

The Oracle logo, featuring the word "ORACLE" in a bold, red, sans-serif font.

Si te interesa conocer la historia detallada de las bases de datos: <https://www.youtube.com/watch?v=EATv303tyck>

INTRODUCCIÓN

Desde los años 80's la mayoría de los sistemas organizacionales han utilizado gestores relacionales:

- Aplicaciones de bancos, bibliotecas, ... Sistemas operacionales que manejan datos transaccionales.
- CRM (Customer Relationship Management)
- ERP (Enterprise Resource Planning)
- Sistemas académicos
- Etc.

CONNECTING TO A DATABASE

Documentation about connection [\[link\]](#)

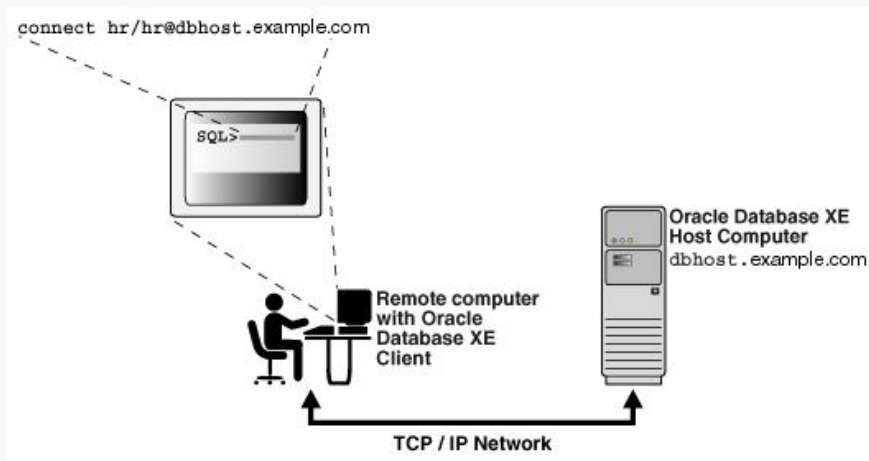
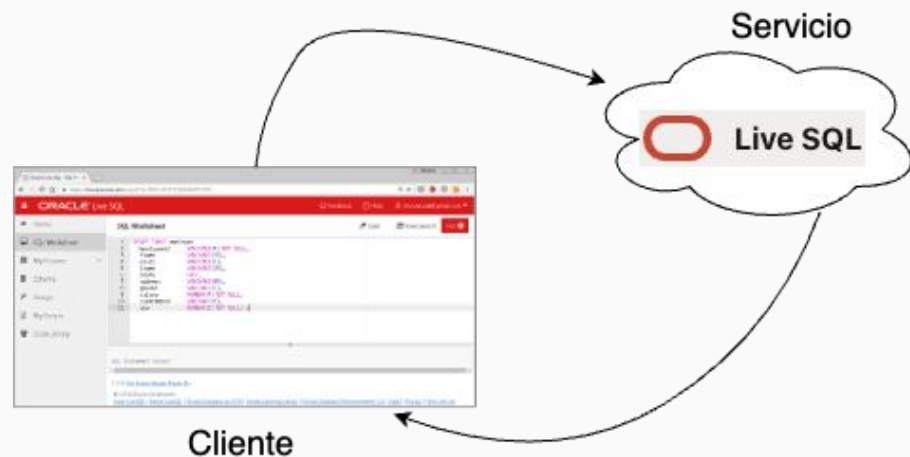
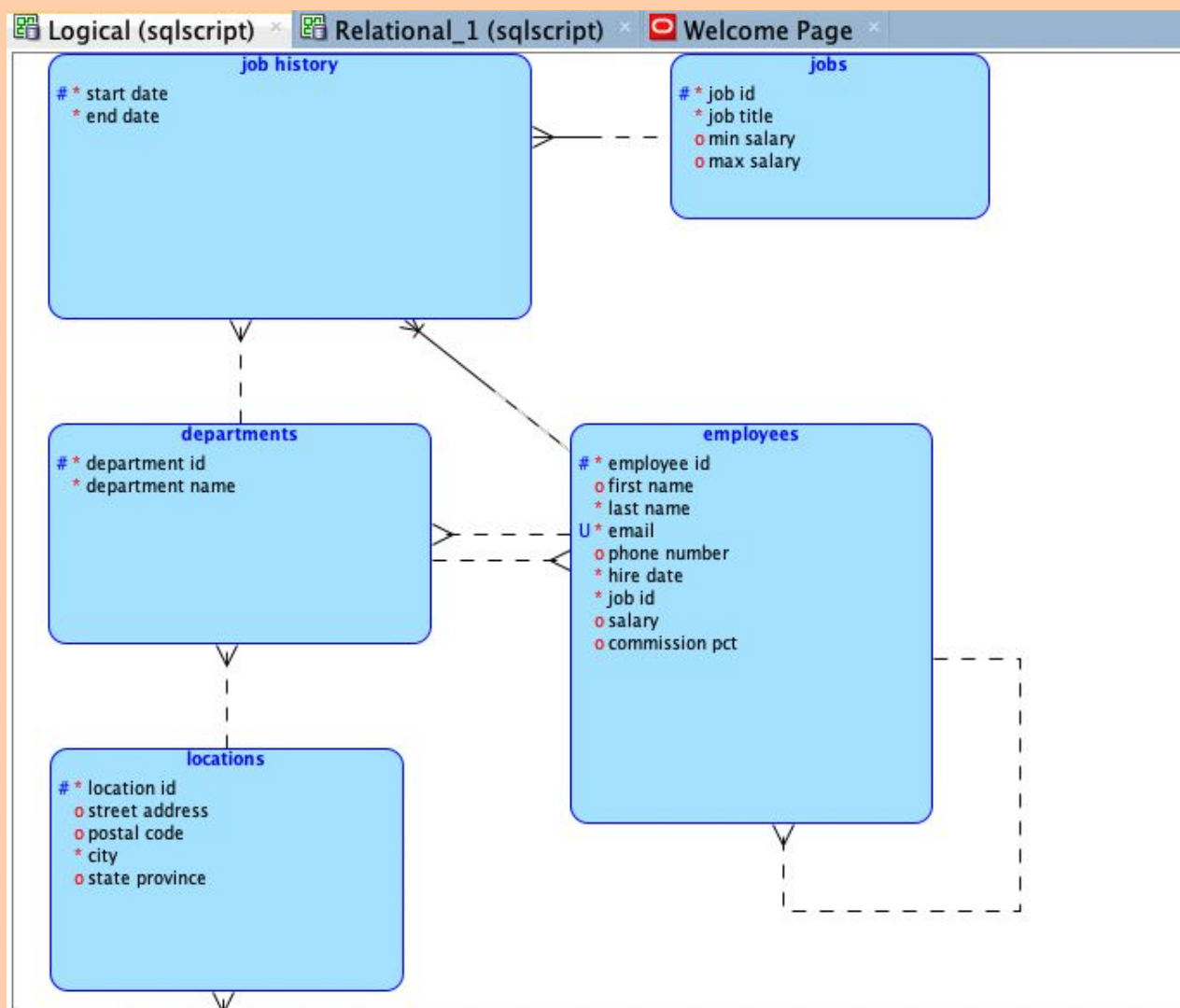


Image taken from [\[here\]](#)



MODELO DE DATOS



Employees

```
1 select * from employees;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	-	-	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	-	100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	-	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	-	102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000	-	103	60
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-05	IT_PROG	4800	-	103	60

```
1 select * from Departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500

¿QUÉ ES NOSQL?

Es una categoría general de sistemas de gestión de bases de datos que difiere de los RDBMS en diferentes modos:

No tienen esquemas, no permiten JOINS, no intentan garantizar ACID y escalan horizontalmente

Tanto las bases de datos NoSQL como las relacionales son tipos de Almacenamiento Estructurado.

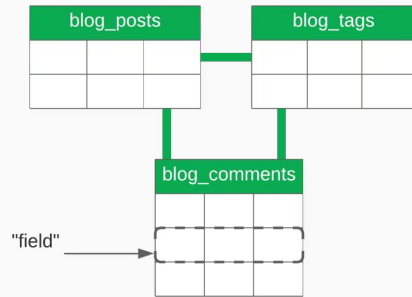
- El término surge en 1998 por Carlo Strozzi. NoSQL - A relational database management system. Carlo Strozzi
- Resucitado en 2009 por Eric Evans

¿POR QUÉ SURGEN?

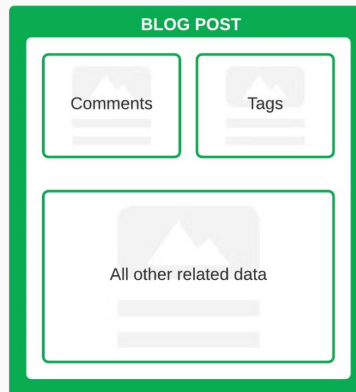
- Necesidad de manipulación de información no estructurada y semi-estructurada.
- Manejo de grandes volúmenes de datos.
- Tradicionalmente los sistemas relacionales asumen modelo computacional centralizado.



Relational



Non-Relational



INTRODUCCIÓN A NOSQL

- Aplicaciones como Facebook, Amazon, Google necesitaban dar servicio a miles de usuarios concurrentes y responder millones de preguntas diarias y la tecnología relacional no ofrecía ni el nivel de escalabilidad ni el rendimiento adecuado.

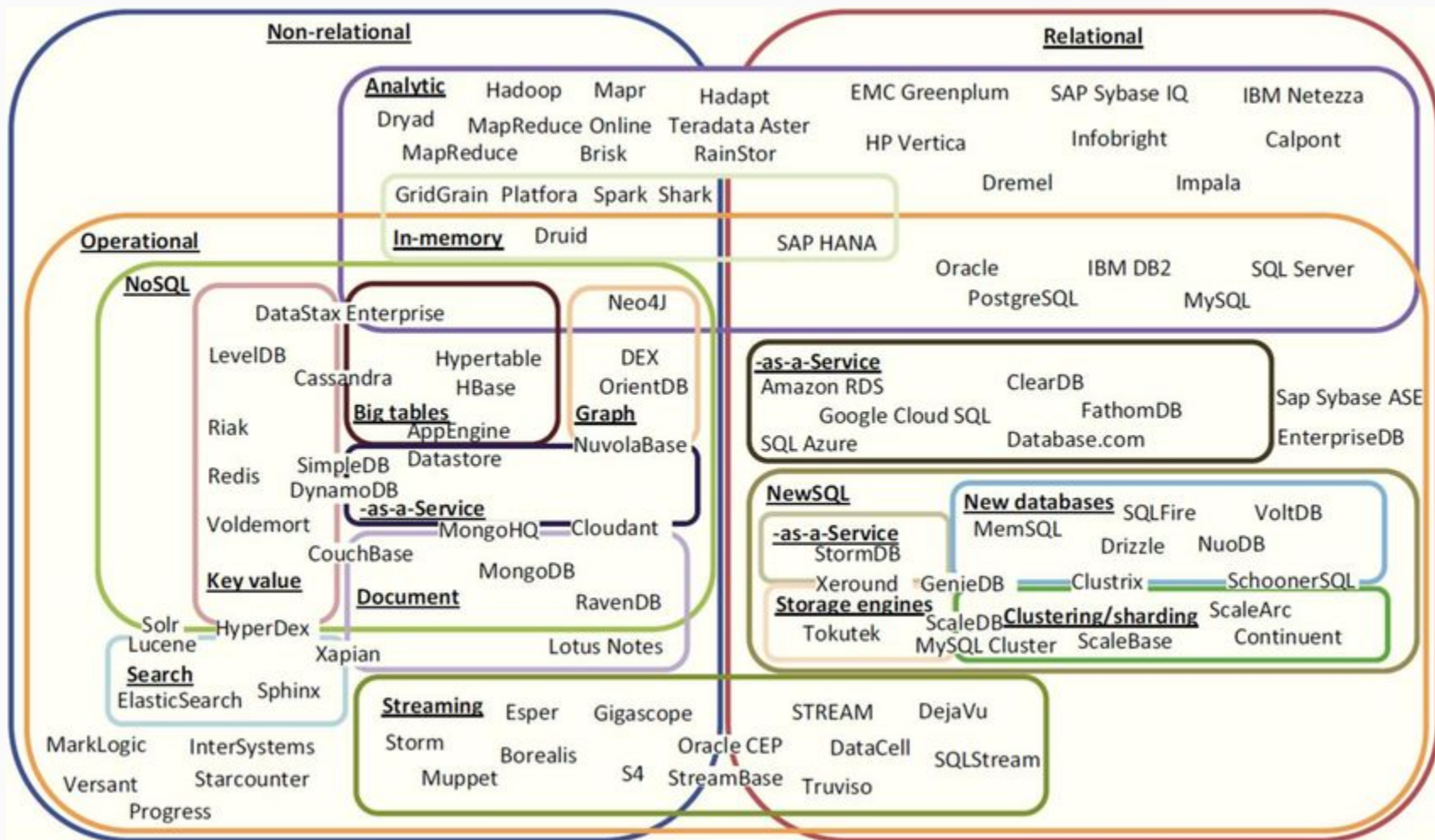
Los RDBMS presentan varios cuellos de botella:

- Gestión de Logs.
- Control de concurrencia.
- Protocolos de transacciones distribuidas.
- Administración de buffers.
- Interfaces CLI (JDBC, ODBC, etc.).

SOLUCIÓN:

- Implementar modelos alternativos que se ajusten a lo que realmente se necesita.
- Google, Facebook, Amazon diseñan su propia solución.





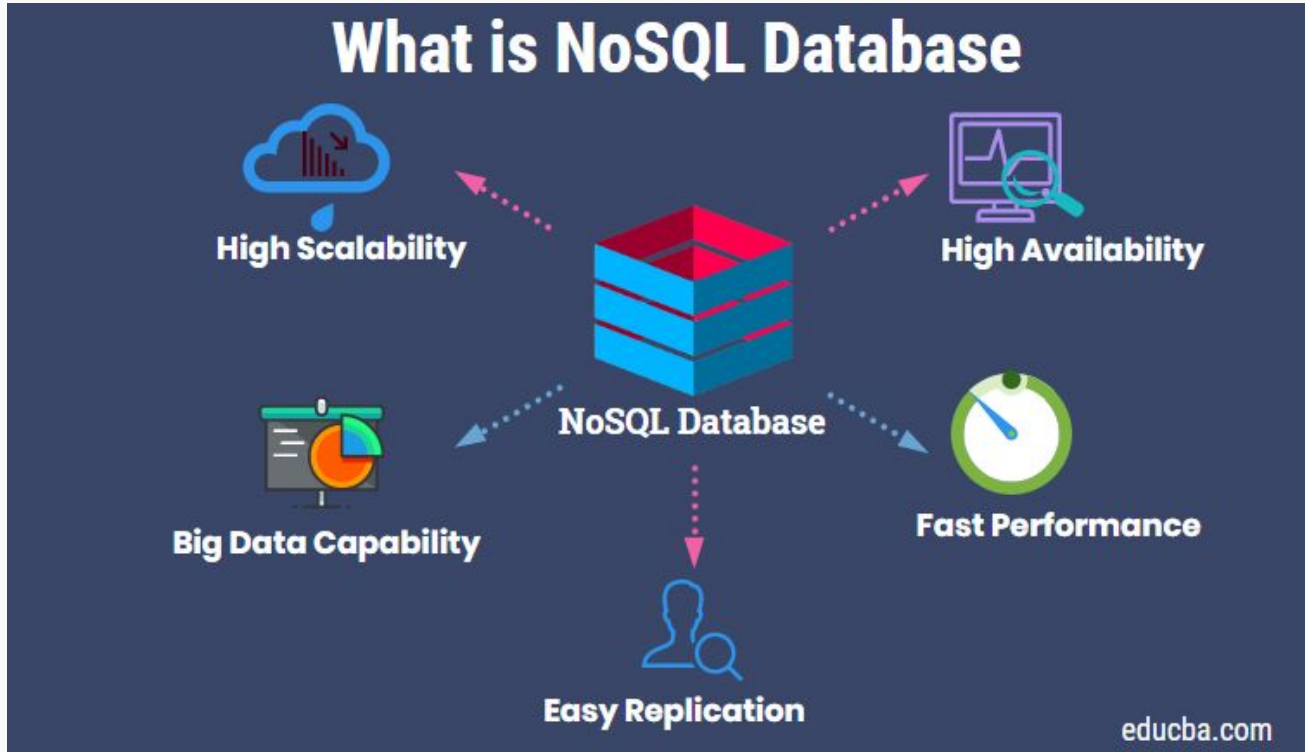
DB ENGINE RANKING

414 systems in ranking, April 2023

Rank			DBMS	Database Model	Score		
Apr 2023	Mar 2023	Apr 2022			Apr 2023	Mar 2023	Apr 2022
1.	1.	1.	Oracle	Relational, Multi-model	1228.28	-33.01	-26.54
2.	2.	2.	MySQL	Relational, Multi-model	1157.78	-25.00	-46.38
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	918.52	-3.49	-19.94
4.	4.	4.	PostgreSQL	Relational, Multi-model	608.41	-5.41	-6.05
5.	5.	5.	MongoDB	Document, Multi-model	441.90	-16.89	-41.48
6.	6.	6.	Redis	Key-value, Multi-model	173.55	+1.10	-4.05
7.	7.	8.	IBM Db2	Relational, Multi-model	145.49	+2.57	-14.97
8.	8.	7.	Elasticsearch	Search engine, Multi-model	141.08	+2.01	-19.76
9.	9.	10.	SQLite	Relational	134.54	+0.72	+1.75
10.	10.	9.	Microsoft Access	Relational	131.37	-0.69	-11.41
11.	12.	11.	Cassandra	Wide column	111.81	-1.98	-10.19
12.	11.	14.	Snowflake	Relational	111.12	-3.27	+21.68
13.	13.	12.	MariaDB	Relational, Multi-model	95.93	-0.90	-14.38
14.	14.	13.	Splunk	Search engine	85.44	-2.54	-9.81
15.	16.	15.	Microsoft Azure SQL Database	Relational, Multi-model	79.06	+1.62	-6.72
16.	15.	16.	Amazon DynamoDB	Multi-model	77.45	-3.32	-5.46
17.	17.	17.	Hive	Relational	71.65	+0.74	-9.77
18.	18.	18.	Teradata	Relational, Multi-model	61.59	-2.14	-5.98
19.	19.		Databricks	Multi-model	60.97	+0.11	
20.	21.	24.	Google BigQuery	Relational	53.32	-0.12	+5.34
21.	20.	19.	Neo4j	Graph	51.60	-1.91	-7.92

<https://db-engines.com/en/ranking>

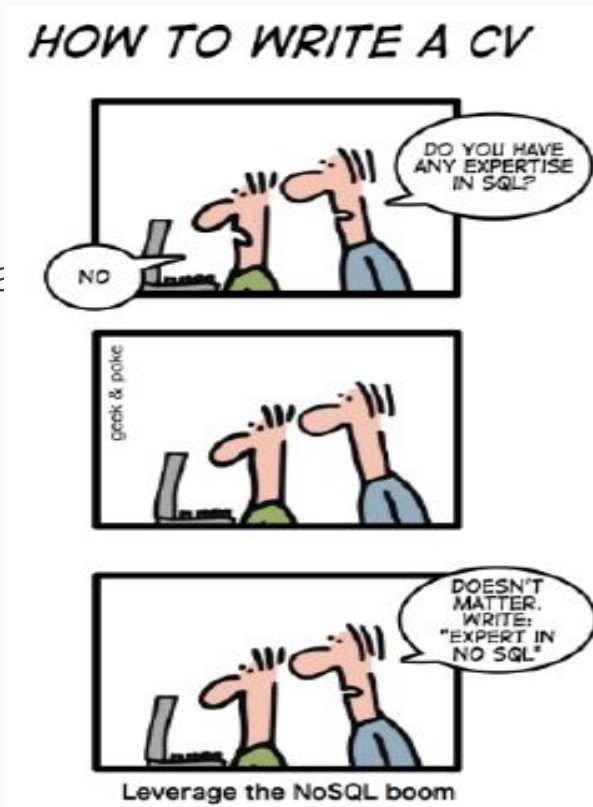
CARACTERÍSTICAS



Source:
<https://www.educba.com/what-is-nosql-database/>

DEFINICIÓN

- *NoSQL* is a set of concepts that allows the rapid and efficient processing of data sets with a focus on performance, reliability, and agility. Dan McCreary - Ann Kelly



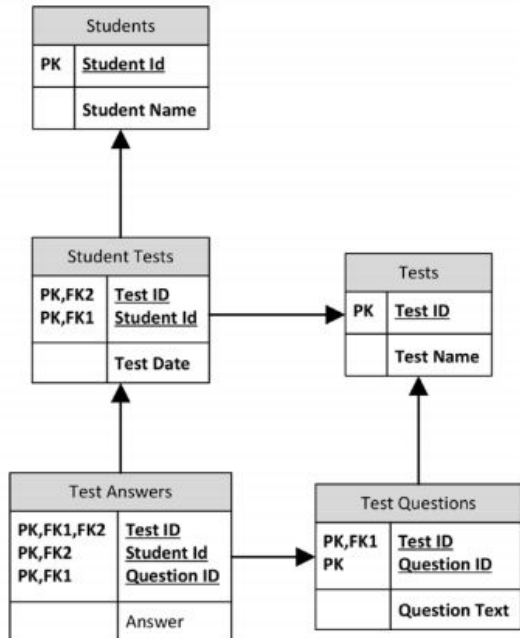
PRINCIPIOS DE NOSQL

- Tanto las bases de datos NoSQL como las relacionales son tipos de Almacenamiento Estructurado.
- La principal diferencia radica en cómo guardan los datos.

Un-normalized data

Test scores	
Student Name	
Test Name	
Test Date	
Answer 1	
Answer 2	
Answer 3	
Answer 4	
Answer 5	
Answer 6	
Answer N	

Normalized data



PRINCIPIOS DE NOSQL

NoSQL se basa en 4 principios:

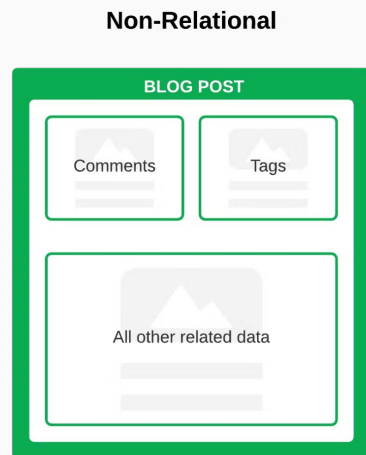
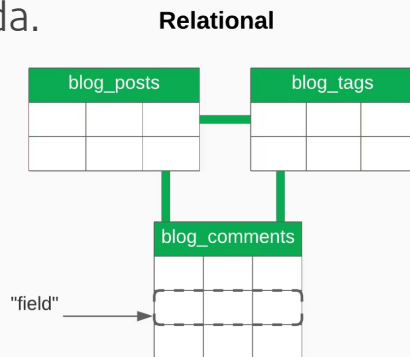
- El control transaccional ACID no es importante.
- Los JOINS tampoco lo son. En especial los complejos y distribuidos. Se persigue la desnormalización.
- Algunos elementos relacionales son necesarios y aconsejables: claves (keys).
- Gran capacidad de escalabilidad y de replicación en múltiples servidores.

CARACTERÍSTICAS PRINCIPALES

- Se ejecutan en máquinas con pocos recursos (clusters)
- No tiene esquemas fijos
- Fáciles de usar en clústers de balanceo de carga convencionales □ facilitan **escalabilidad horizontal**
- No genera cuellos de botella
- Tienen propiedades ACID en un nodo del clúster y son “**eventualmente consistentes**” en el clúster.
- Las consultas se realizan principalmente por key o índice (otro tipo de consultas son muy costosas).
- Las consultas complejas se realizan mediante una infraestructura de procesamiento externo tal como MapReduce.

DIFERENCIAS CON LAS SQL

- Modelo de datos relacional vs no relacionales
- SQL vs lenguajes dinámicos
- Almacenamiento en tablas vs estructuras flexibles.
- Info no estructurada (Json) vs transacciones con varias filas, JOINS.
- Arquitectura distribuida vs centralizada.
- Escalabilidad horizontal vs vertical



COMPUTACIÓN DISTRIBUIDA

Clustering: un clúster o racimo de computadoras consiste en un grupo de computadoras de relativo bajo costo conectadas, unidas entre sí, normalmente por una red de alta velocidad y que se comportan como si fuesen una única computadora (balanceo de carga).



TEOREMA CAP

Teorema de Brewer: “es imposible para un sistema computacional distribuido ofrecer simultáneamente las siguientes tres garantías”:

Consistencia (*Consistency*) – todos los nodos ven los mismos datos al mismo tiempo.

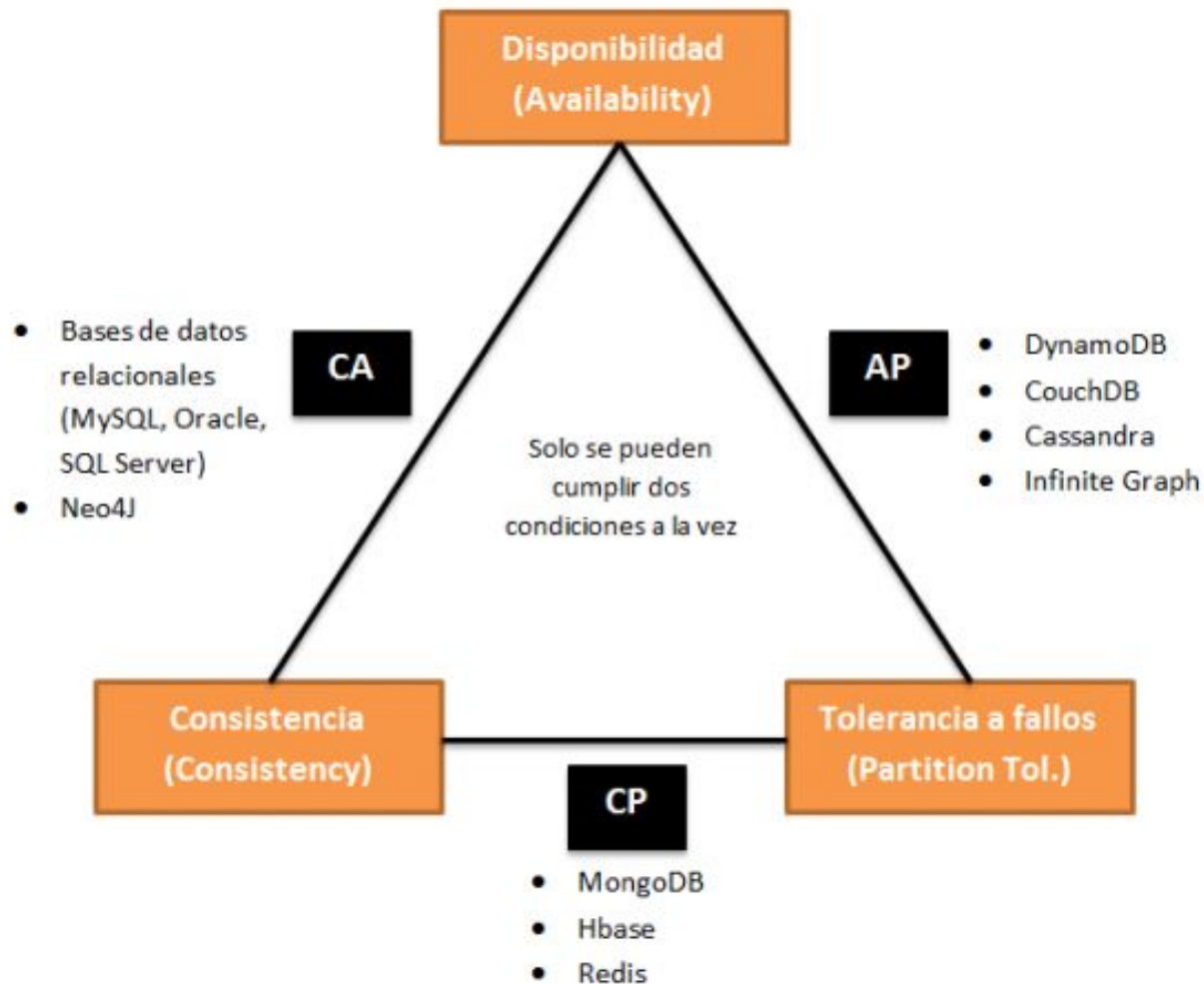
Disponibilidad (*Availability*) – garantiza que cada petición recibe una respuesta.

Tolerancia a la partición (*Partition*) – el sistema continua funcionando a pesar de que fallen algunos nodos.

Equivalente a:

“You can have it good, you can have it fast, you can have it cheap: pick two.”

DOS DE TRES



EJEMPLO

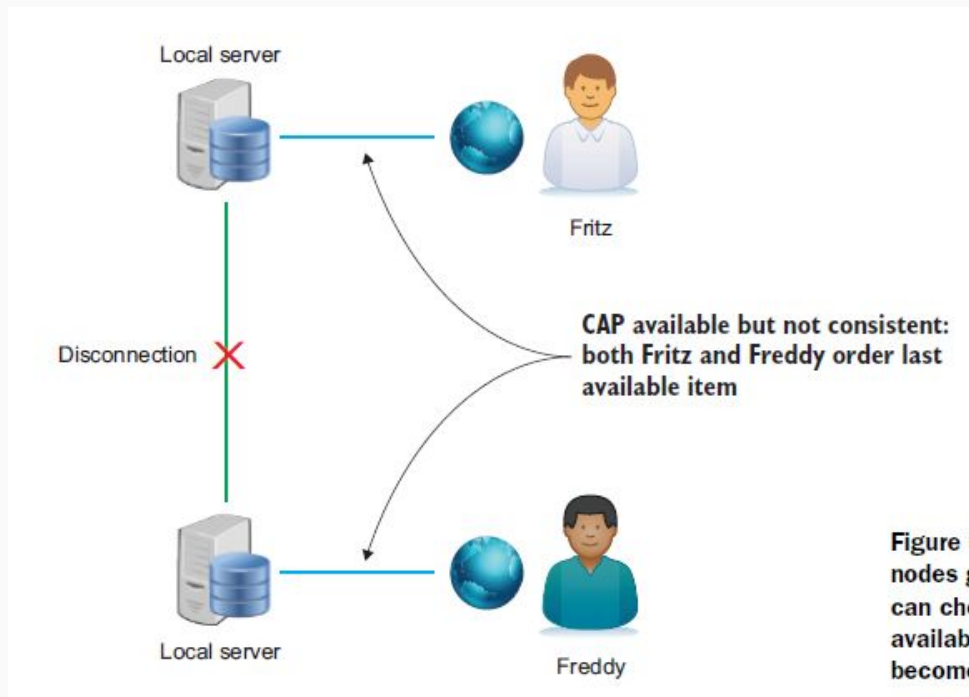
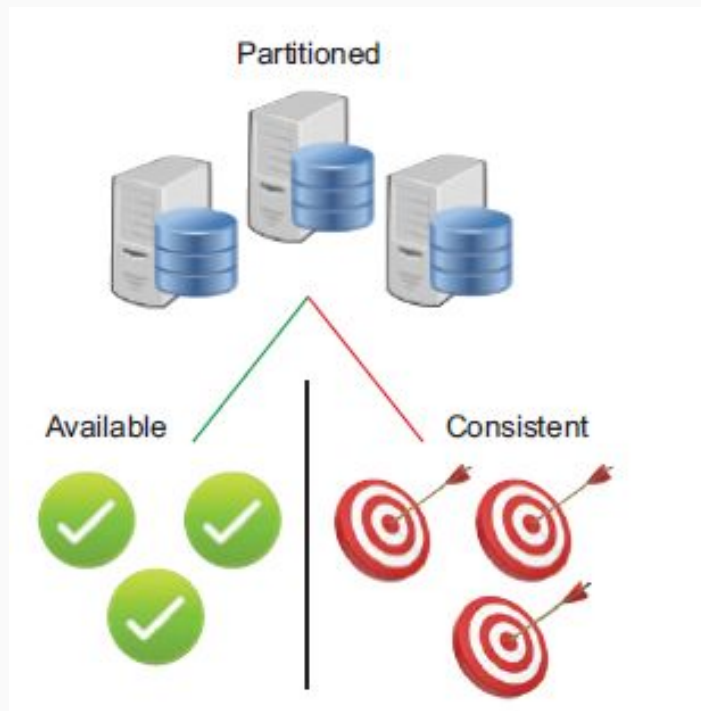


Figure 6
nodes g
can cho
availabl
become

ACID VS BASE

En las BD relacionales, estamos familiarizados con las transacciones ACID, que garantizar la consistencia y estabilidad de las operaciones pero requieren bloqueos sofisticados:

ACID = **A**tomicidad, **C**onsistencia, (**I**solation) aislamiento y **D**urabilidad

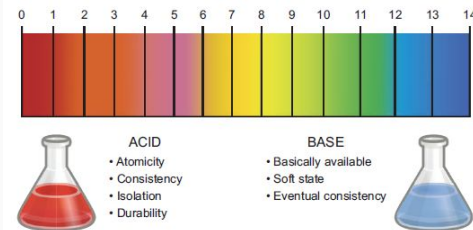
Las BBDD NoSQL son repositorios de almacenamiento más optimistas , siguen el **modelo BASE**:

Basic availability - funciona la mayoría del tiempo incluso ante fallos gracias al almacenamiento distribuido y replicado

Soft-sate - no tienen porque ser consistentes sus réplicas en todo momento.

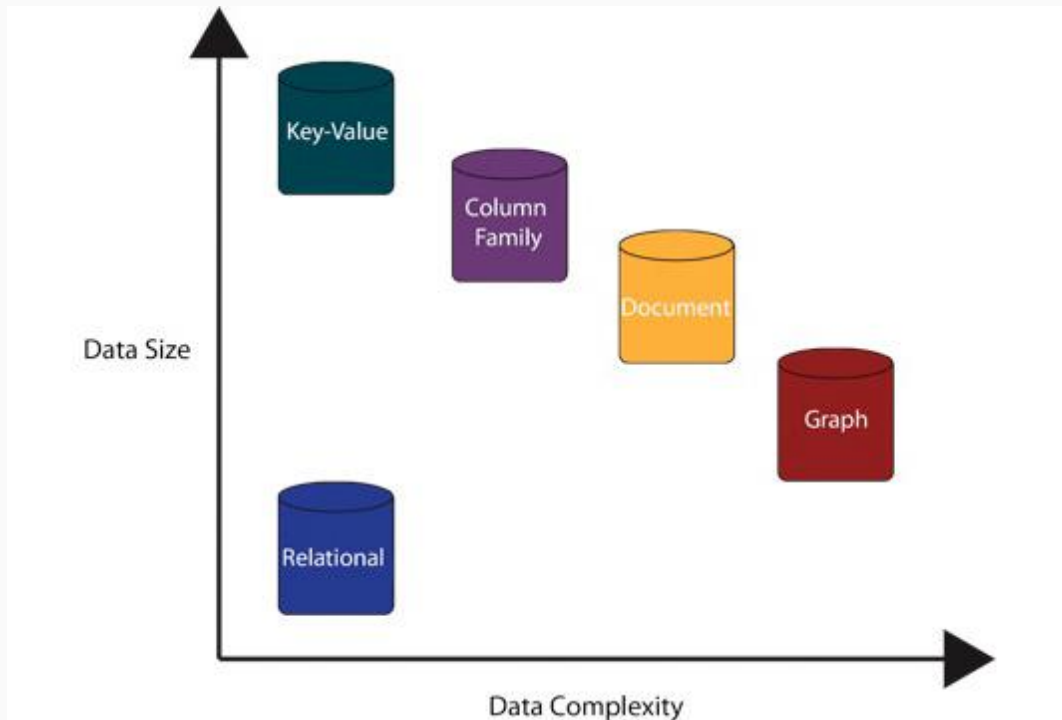
Eventual consistency - la consistencia se da pasado cierto tiempo

BASE es una alternativa flexible a ACID para aquellas bases de datos que no requieren un manejo estricto de las transacciones.



TIPOS DE BD NOSQL

- Clave-valor
- Orientadas a documentos
- Orientadas a columnas
- Orientadas a grafos



EJEMPLO RELACIONAL

Name	Birthday	Person ID
Jos The Boss	11-12-1985	1
Fritz von Braun	27-1-1978	2
Freddy Stark		3
Delphine Thewiseone	16-9-1986	4

Person info table: represents person-specific information

Person ID	Hobby ID
1	2
1	2
2	3
2	4
3	5
3	6
3	1

Person-Hobby linking table: necessary because of the many-to-many relationship between hobbies and persons

Hobby ID	Hobby Name	Hobby Description
1	Archery	Shooting arrows from a bow
2	Conquering the world	Looking for trouble with your neighboring countries
3	Building things	Also known as construction
4	Surfing	Catching waves on a plank
5	Swordplay	Fencing with swords
6	Lollygagging	Hanging around doing nothing

Hobby info table: represents hobby-specific information

ORIENTADA A COLUMNAS

Name	Row ID
Jos The Boss	1
Fritz von Braun	2
Freddy Stark	3
Delphine Thewiseone	4

Birthday	Row ID
11-12-1985	1
27-1-1978	2
16-9-1986	4

Hobbies	Row ID
Archery	1, 3
Conquering the world	1
Building things	2
Surfing	2
Swordplay	3
Lollygagging	3

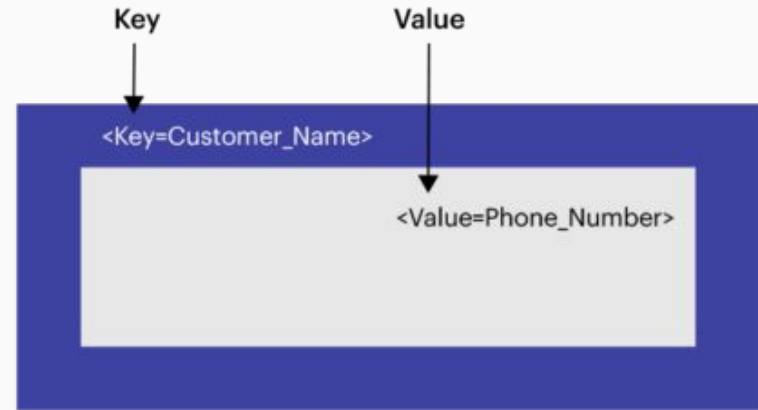
KEY-VALUE

Key	Value
Name	Jos The Boss
Birthday	11-12-1985
Hobbies	Archery, conquering the world

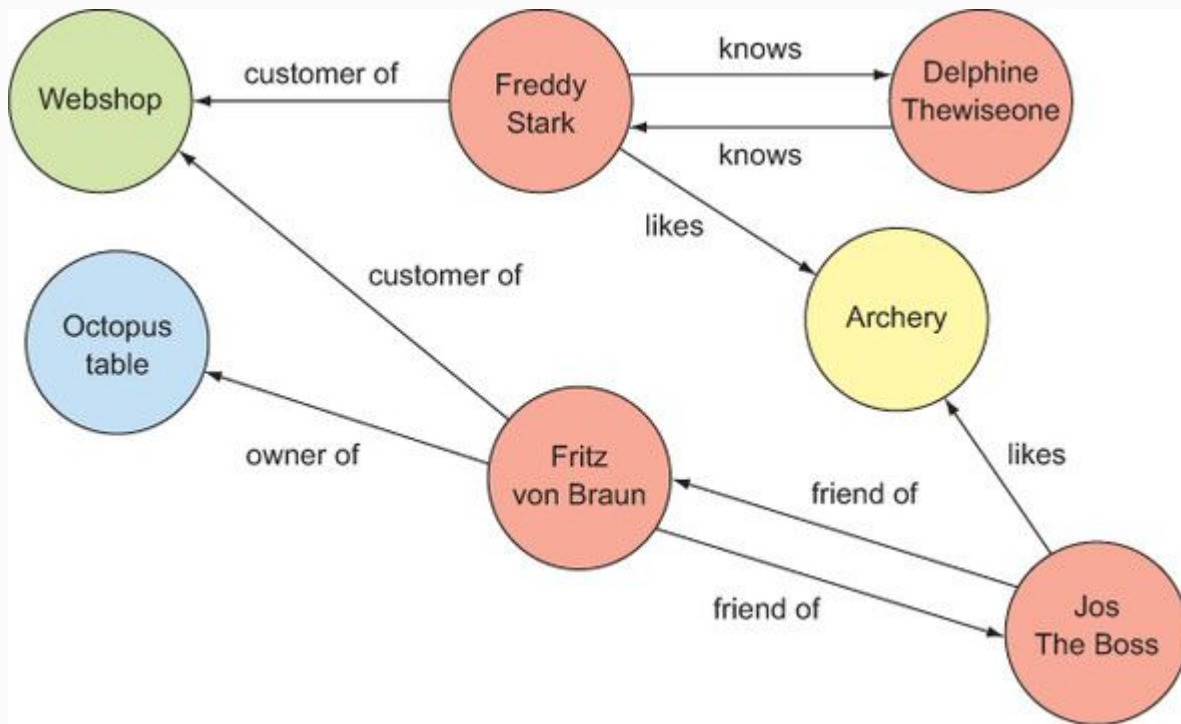
```
{
  "internal data": [
    {
      "entities": [
        {
          "customer": [
            {
              "id": 1,
              "name": "Freddy"
            },
            {
              "id": 2,
              "name": "Fritz"
            }
          ]
        },
        {
          "legal entities": [
            {
              "id": 1,
              "company": "Maiton"
            }
          ]
        }
      ]
    },
    {
      "Products": [
        {
          "furniture": [
            {
              "id": 1,
              "name": "Octopus Table",
              "stock": 1
            }
          ]
        }
      ]
    }
  ]
}
```


Phone directory

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



GRAFOS



Otro ejemplo



DynomiteDB



Voldemort



HYPERTABLE INC

APACHE
HBASE



Cassandra

riak

Key-Value

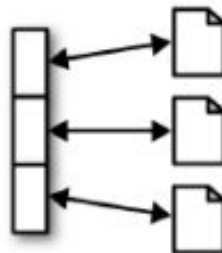
Column Family

					1	
1					1	
	1			1		
	1	1				
					1	
	1				1	
					1	
		1			1	
					1	

Graph DB



Document



ArangoDB



AllegroGraph



Neo4j

the graph database



mongoDB



CouchDB
relax

CONCLUSIONES

- Se recuperan los datos más rápidamente que en RDBMS, consultas más limitadas. La complejidad se traslada a la aplicación.
- Si se requiere manejo transaccional y garantía de ACID es mejor usar RDBMS.
- NoSQL no es adecuado para consultas complejas (... a excepción de las Orientadas a grafos, soportan consultas muy complejas).
- La tendencia es a que convivan diferentes tipos de BD. Persistencia políglota.

BD ORIENTADAS A DOCUMENTOS



BASES DE DATOS ORIENTADAS A DOCUMENTOS

Las bases de datos orientadas a documentos son un tipo de base de datos NoSQL que almacenan, recuperan y gestionan datos en forma de documentos.

Cada documento es una entidad autocontenida, típicamente almacenada en formato XML, JSON o BSON (Binary JSON)

Las bases de datos orientadas a documentos ofrecen flexibilidad de esquema, permitiendo que cada documento tenga su propia estructura.

Algunos ejemplos de este tipo son:

MongoDB o **CouchDB**.



CONCEPTOS CLAVE

Documentos

- Unidad autocontenida de datos, análogos a las filas en las tablas
- Consiste en pares clave-valor, donde los valores pueden ser tipos de datos simples, como cadenas, números o valores booleanos, o tipos de datos complejos, como matrices o documentos anidados.

Colecciones

- Agrupamiento lógico para los documentos que comparten un propósito similar.
- Las colecciones no imponen un esquema

Bases de datos

Conjunto de una o más colecciones



<https://www.mongodb.com/es>

MONGODB



MONGODB

MongoDB (de la palabra en ingles “humongous” que significa enorme) es un sistema de base de datos NoSQL orientado a documentos.

MongoDB **guarda estructuras de datos en documentos** tipo **BSON** (*Binary JSON* - JSON Binario) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.



MODELO DE DATOS - RDBMS VS MONGODB

RDBMS	MongoDB
Base de datos	Base de datos
Tabla	Colección
Índice	Índice
Fila	Documento JSON
Columna	Campo del documento
Join	Documentos embebidos y búsqueda

PRÁCTICA MONGODB INSTRUCCIONES BÁSICAS



CREAR UN BASE DE DATOS

Relacional SQL

```
CREATE DATABASE  
base_datos;
```

```
> db //muestra la BD actual
```

MongoDB

base_datos



**Nombre de la
base de datos**

INSERTAR UN DOCUMENTO EN UNA COLECCIÓN

Poco flexible,
Ligado a la estructura!

Relacional SQL

TABLA + INSERT

```
CREATE TABLE users (  
  id int primary key,  
  nombre varchar(255) not null,  
  apellido varchar(255) not null,  
  edad int not null,  
  estatura float not null  
);  
  
INSERT INTO users(id,nombre, apellido, edad, estatura)  
VALUES (1,"María", "Fernandez", 38, 1.72)
```

MongoDB

```
usuario = { nombre : "Monica", apellido : "Rojas",  
edad : 43, estatura : 1.60 };
```

Colección

```
db.users.insertOne(usuario);
```

BD actual

Documento

SELECCIONAR = BUSCAR

Relacional SQL

```
SELECT * FROM users;
```

MongoDB

Colección



```
db.users.find ();
```



BD actual



Operador

BUSCANDO

Relacional SQL

```
SELECT top 1  
FROM users;
```

Identificador único autogenerado

**¿Con base en qué se
genera el Identificador
Único?**


MongoDB

```
var usuario = db.users.findOne();
```

BD actual

Colección

Operador

 `{ "_id" : ObjectId("59c3331d45953192e2928d50"),
nombre: "Maria", apellido: "Fernandez", edad: 38,
estatura: 1.72
}`

SELECT top 1 apellido FROM **users** WHERE nombre = "Maria"

MongoDB

Colección ↓
Case sensitive
Condicional ↓
db.users.findOne({ nombre : "Maria" }, { apellido : true });
↑ **BD actual** ↑ **Proyección**

```
> db.usuarios.findOne({nombre:"Maria"}, {apellido:true})
{ "_id" : ObjectId("5f6a5f44a5f0c5bd29483fa8"), "apellido" : "Fernandez" }
>
```


MONGODB COMPASS

base_datos.users

DOCUMENTS 10 STORAGE SIZE 20.5KB AVG. SIZE 94B INDEXES 1

Documents Aggregations Schema Explain Plan Indexes Validation




FILTER { nombre : "Maria" } **OPTIONS**

PROJECT { apellido : true }

SORT { field: -1 } or [['field', -1]] **MAX TIME MS** 60000

COLLATION { locale: 'simple' } **SKIP** 0 **LIMIT** 0

FIND

VIEW   

Displaying documents 1 - 1 of 1

```
_id: ObjectId("6234b094efe50c37f9da527b")
apellido: "Andrade"
```

MONGODB COMPASS

FILTER

{nombre:"Maria", apellido: "Andrade"}

OPTIONS

PROJECT

{ field: 0 }

SORT

{ field: -1 } or [['field', -1]]

MAX TIME MS

60000

COLLATION

{ locale: 'simple' }

SKIP

0

LIMIT

0

ADD DATA

VIEW

{}

Displaying documents 1 - 1 of 1

```
_id: ObjectId("6234b094efe50c37f9da527b")
nombre: "Maria"
apellido: "Andrade"
edad: 18
estatura: "1.50"
```

OPERADORES

```
SELECT * FROM USERS WHERE nombre <> "Maria";
```

MongoDB

Colección

Operador
desigualdad

```
var usuario = db.users.find({nombre: {$ne : "Maria"}});
```

BD actual

Operador

```
> db.usuarios.find({nombre: {$ne : "Maria"}});
{ "_id" : ObjectId("5f6a5f05a5f0c5bd29483fa7"), "nombre" : "Monica", "apellido" : "Rojas", "edad" : 43, "estatura" : 1.6 }
>
```

base_datos.users

DOCUMENTS 10

STORAGE SIZE
20.5KB

AVG. SIZE
94B

INDEXES

Documents

Aggregations

Schema

Explain Plan

Indexes

Validation

FILTER {nombre: {\$ne : "Maria"}}

▼ OPTIONS

FIND

PROJECT { field: 0 }

SORT { field: -1 } or [['field', -1]]

MAX TIME MS 60000

COLLATION { locale: 'simple' }

SKIP 0

LIMIT 0

ADD DATA ▼



VIEW



Displaying documents 1 - 9 of 9

```
_id: ObjectId("6234b094efe50c37f9da527a")
nombre: "Monica"
apellido: "Rojas"
genero: "femenino"
edad: 43
estatura: "1.60"
```

```
_id: ObjectId("6234b094efe50c37f9da527c")
nombre: "Luis"
apellido: "Ruiz"
edad: 48
estatura: "1.70"
```

OPERADORES

Operador	Expresión
Igual	\$eq
Diferente	\$ne
Mayor que	\$gt
Mayor o igual que	\$gte
Menor que	\$lt
Menor o igual que	\$lte
Existencia en array	\$in
Inexistencia en array	\$nin

CONSULTA OPERADOR AND

```
SELECT * FROM users  
WHERE nombre = 'Maria' AND apellido = 'Andrade';
```

MongoDB

Colección


db.users.find({nombre:"Maria", apellido: "Andrade"});



BD actual

CONSULTA OPERADOR OR

```
SELECT * FROM users  
WHERE nombre = 'Maria' OR apellido = 'Cardenas'
```

MongoDB

Colección



```
db.users.find({$or:[{nombre:"Maria"},{apellido:"Cardenas"}]});
```



BD actual

FILTER

{ \$or: [{ nombre: "Maria"}, { apellido: "Cardenas" }] }

PROJECT

{ field: 0 }

SORT

{ field: -1 } or [['field', -1]]

MAX TIME MS

60000

COLLATION

{ locale: 'simple' }

SKIP

0

LIMIT

0

ADD DATA

VIEW

{}

Displaying documents 1 - 2 of 2

_id: ObjectId("6234b094efe50c37f9da527b")
nombre: "Maria"
apellido: "Andrade"
edad: 18
estatura: "1.50"

_id: ObjectId("6234b094efe50c37f9da5281")
nombre: "Lucia"
apellido: "Cardenas"
edad: 24
estatura: "1.68"

```
_id: ObjectId("6234b094efe50c37f9da5281")
nombre: "Lucia"
apellido: "Cardenas"
edad: 24
estatura: "1.68"
```


CLÁUSULA ORDER BY

```
SELECT * FROM USERS ORDER BY nombre, edad ASC;
```

```
SELECT * FROM USERS ORDER BY nombre, edad DESC;
```

MongoDB

Colección



```
db.users.find().sort({"nombre":1, "edad":-1})
```



BD actual

1 para ordenar Ascendente
-1 para ordenar Descendente

MONGODB COMPASS

FILTER { field: 'value' }

▼ OPTIONS

FIND

RESET



PROJECT { field: 0 }

SORT [{"nombre":1, "edad":-1}]

MAX TIME MS 60000

COLLATION { locale: 'simple' }

SKIP 0

LIMIT 0

ADD DATA ▼



VIEW



Displaying documents 1 - 10 of 10



REFRESH

```
_id: ObjectId("6234b094efe50c37f9da5280")
nombre: "Alejandro"
apellido: "Sanchez"
edad: 10
estatura: "1.40"
```

```
_id: ObjectId("6234b094efe50c37f9da527f")
nombre: "Camila"
apellido: "Jurado"
genero: "femenino"
edad: 12
estatura: "1.50"
```

```
_id: ObjectId("6234b094efe50c37f9da5283")
nombre: "Diego"
```

LIKE: CONTIENE

SELECT * FROM TABLE users WHERE nombre LIKE '%Leon%'

db.users.find({nombre: /Leon/});

```
Atlas atlas-mrr86e-shard-0 [primary] users_infrati> db.users.find({nombre: /Leon/});
[
  {
    _id: ObjectId("6346dd3dc060d51f1b9c73e1"),
    nombre: 'Leonor',
    apellido: 'Restrepo',
    edad: 78
  },
  {
    _id: ObjectId("6346dd3dc060d51f1b9c73e2"),
    nombre: 'Leonardo',
    apellido: 'Jurado',
    edad: 44,
    estatura: '1.80'
  }
]
```

ⓘ FILTER

{nombre: /Leon/}

ⓘ PROJECT

{ field: 0 }

ⓘ SORT

{ field: -1 } or [['field', -1]]

ⓘ COLLATION

{ locale: 'simple' }

ⓘ MAX TIME MS

60000

ⓘ SKIP

0

ⓘ LIMIT

0

▼ OPTIONS

FIND

59

LIKE: COMIENZA

SELECT * FROM TABLE users WHERE nombre LIKE 'L%'

db.users.find({nombre: /^L/});

```
Atlas atlas-mrr86e-shard-0 [primary] users_infrati> db.users.find({nombre: /^L/});
[
  {
    _id: ObjectId("6346dd3dc060d51f1b9c73e0"),
    nombre: 'Luis',
    apellido: 'Ruiz',
    edad: 48,
    estatura: '1.70'
  },
  {
    _id: ObjectId("6346dd3dc060d51f1b9c73e1"),
    nombre: 'Leonor',
    apellido: 'Restrepo',
    edad: 78
  },
  {
    _id: ObjectId("6346dd3dc060d51f1b9c73e2"),
    nombre: 'Leonardo',
    apellido: 'Jurado',
    edad: 44,
    estatura: '1.80'
  },
  {
    _id: ObjectId("6346dd3dc060d51f1b9c73e5"),
    nombre: 'Lucia',
    apellido: 'Cardenas',
    edad: 24,
    estatura: '1.60'
  }
]
```

FILTER	{nombre: /^L/}	OPTIONS	FIND
PROJECT	{ field: 0 }		
SORT	{ field: -1 } or [['field', -1]]	MAX TIME MS	60000
COLLATION	{ locale: 'simple' }	SKIP	0
		LIMIT	0

LIKE, CONTINUACIÓN

```
SELECT * FROM TABLE users WHERE nombre LIKE '%o'
```

```
db.users.find({nombre: /o$/});
```

FILTER	{nombre: /o\$/}	OPTIONS	FIND
PROJECT	{ field: 0 }		
SORT	{ field: -1 } or [['field', -1]]	MAX TIME MS 60000	
COLLATION	{ locale: 'simple' }	SKIP 0 LIMIT 0	

BETWEEN

```
SELECT * FROM TABLE users WHERE edad BETWEEN 20 AND 38;  
db.users.find({edad: {$gte:20,$lte:38}});
```

```
SELECT * FROM TABLE users WHERE edad>=20 AND edad <38;  
db.users.find({edad: {$gte:20,$lt:38}});
```

FILTER	<code>{edad: {\$gte:20,\$lte:38}}</code>	OPTIONS
PROJECT	<code>{ field: 0 }</code>	
SORT	<code>{ field: -1 } or [['field', -1]]</code>	MAX TIME MS 60000
COLLATION	<code>{ locale: 'simple' }</code>	SKIP 0 LIMIT 0

DISTINCT

```
SELECT DISTINCT genero FROM TABLE users;
```

```
db.users.distinct("genero");
```

The screenshot shows a MongoDB query editor interface. At the top, there's a dropdown menu set to '\$group' with a green toggle switch to its right. Below this, the query editor contains a JSON document with a '\$group' stage. The output panel on the right shows two sample documents resulting from the query.

```
1 ▾ /**  
2   * _id: The id of the group.  
3   * fieldN: The first field name.  
4   */  
5 ▾ {  
6   id: "$genero",  
7   |  
8 }
```

Output after `$group` stage (Sample of 2 documents)

`_id: "femenino"`

`_id: null`

TALLER

1. Consulte todos los documentos de la colección users
2. Muestre la cantidad de documentos que hay en la colección users
3. Muestre los nombres de todos los usuarios
4. Muestre los nombres y apellidos de los users mayores de 18 años
5. Muestre los nombres y apellidos de las usuarias menores de 40 años

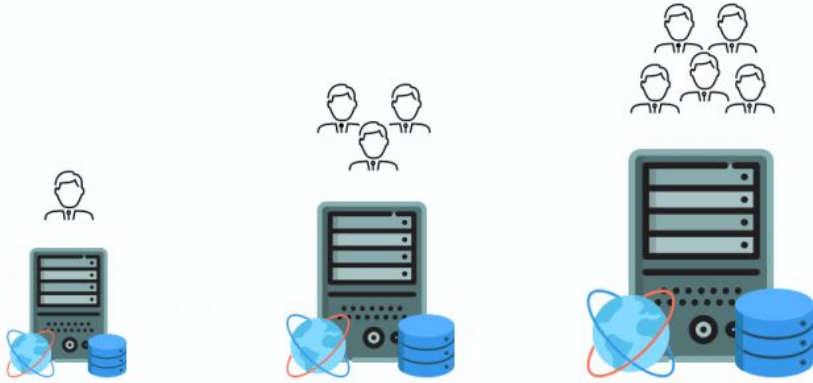
REFERENCIAS

Cielen, D., Meysman, A., & Ali, M. (2016). Introducing Data Science: Big Data. Machine Learning and More, Using Python Tools. Manning, Shelter Island, US, 322.

MOOC BigData: Sistemas gestores de bases de datos orientados a documentos III
<https://www.youtube.com/watch?v=eYiebokW2hg>

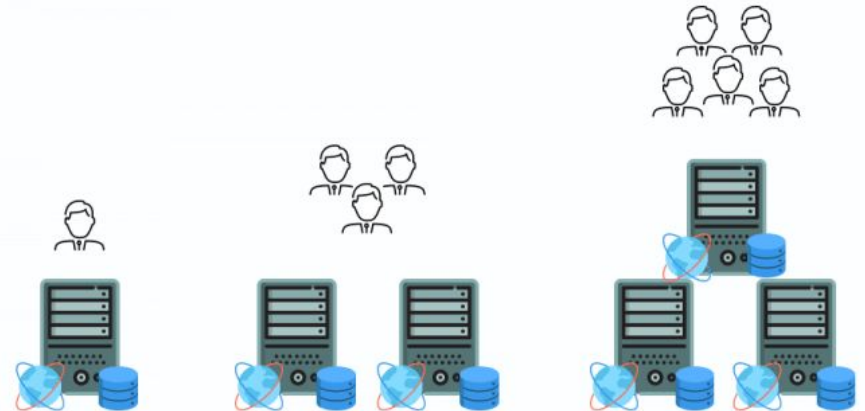
Manual MongoDB
<https://docs.mongodb.com/manual/replication/>

Vertical Scaling



TIPOS DE ESCALABILIDAD

Horizontal Scaling



MAP REDUCE

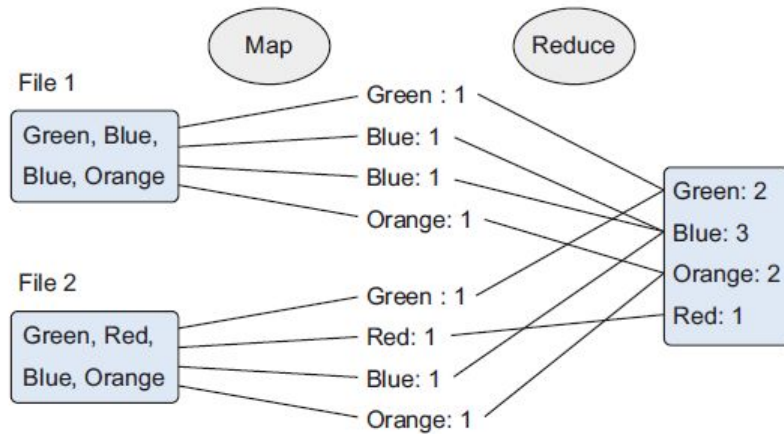
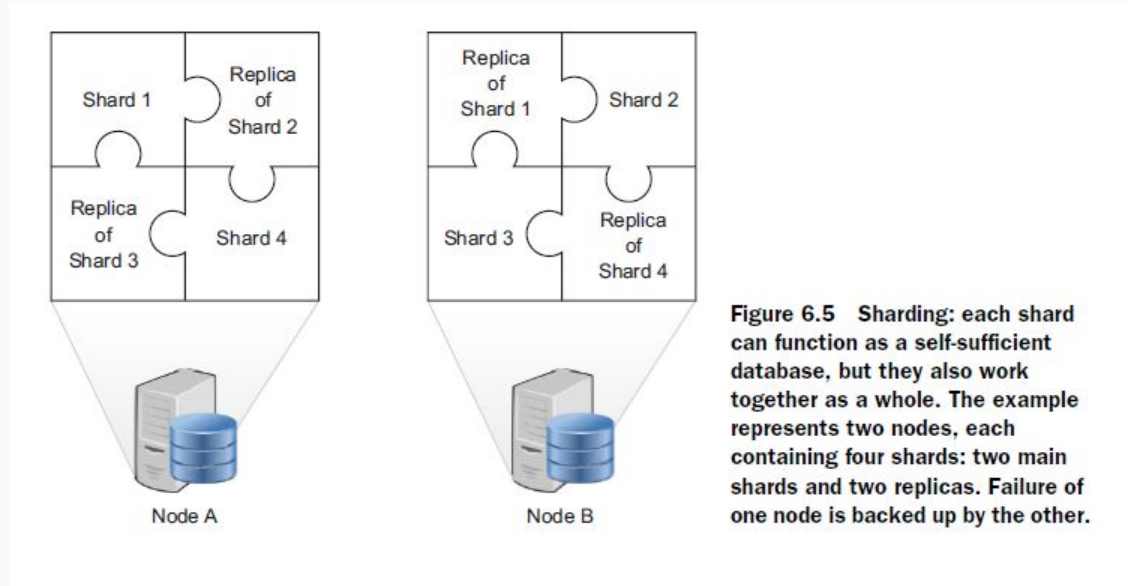


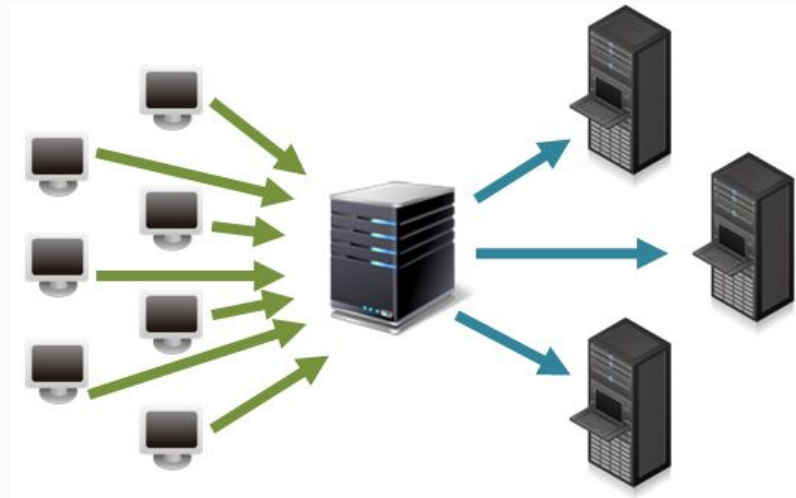
Figure 5.3 A simplified example of a MapReduce flow for counting the colors in input texts

CONSISTENCIA EVENTUAL



BALANCEO DE CARGA

- Centralizado
- Semi-distribuido
- Completamente distribuido



PROPIEDADES ACID - TRANSACCIONES

- **A – Atomicity** (Atomicidad)

Que todas las operaciones se realicen adecuadamente o ninguna de ellas.

- **C – Consistency** (Consistencia)

Antes de ejecutarse la transacción la BD es consistente, al finalizar debe seguir consistente.

- **I – Isolation** (Aislamiento)

Aunque se ejecuten varias TX concurrentemente, esta propiedad asegura que una operación no puede afectar a otras.

- **D – Durability** (Durabilidad)

Al finalizar con éxito una Tx, los cambios en la BD deben permanecer, incluso si hay fallos.

Transacción (Tx):

Es una **Unidad** de la ejecución de un programa que accede y posiblemente actualiza varios elementos de los datos.

ARQUITECTURA DE LAS NOSQL

- A menudo ofrecen sólo **garantías de consistencia débiles**, como por ejemplo *eventual consistency*, o transacciones restringidas a elementos de datos simples
- Emplean una **arquitectura distribuida**, donde los datos se guardan de modo redundante en distintos servidores, a menudo usando tablas hash distribuidas.
- Suelen ofrecer **estructuras de datos sencillas** como arrays asociativos o almacenes de pares clave-valor.

- Su uso es adecuado en:
 - Manejo de grandes volúmenes
 - Frecuencia alta de accesos de lectura y escritura
 - Cambios frecuentes en los esquemas de datos
 - Y que no requieran propiedades ACID

APLICACIONES