WWW.CIIA.XYZ

CII.IA®

Data bases Introduction

Part 2

- Dra. MaryPaz Rico Fernández

CONTENIDO

WWW.CIIIA.XYZ

WWW.CIIIA.XYZ

NoSQL= ?

WWW.CIIIA.XYZ

NoSQL= ?

NoSQL (originally referring to "non SQL" or **"non relational"**) database provides a mechanism for storage and retrieval of data that is modeled in means **other than the tabular relations used in relational databases.**

HISTORICAL CONTEXT

HISTORICAL CONTEXT

Such databases have existed since the late **1960s**, but did not obtain the "NoSQL" moniker until a surge of popularity in the early **21st century**, triggered by the needs of Web 2.0 companies. NoSQL databases are increasingly used in **big data and real-time web applications**.

NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages, or sit alongside SQL database in a polyglot persistence architecture.

HISTORICAL CONTEXT

The term NoSQL was used by **Carlo Strozzi in 1998** to name his lightweight Strozzi NoSQL open-source relational database that did not expose the standard Structured Query Language (SQL) interface, but was still relational.

His NoSQL RDBMS is distinct from the circa-2009 general concept of NoSQL databases. Strozzi suggests that, because the current NoSQL movement **"departs from the relational model altogether, it should therefore have been called more appropriately 'NoREL'**, referring to **'No Relational'.**.

HISTORICAL CONTEXT

WWW.CIIA.XYZ

Johan Oskarsson, then a developer at Last.fm, reintroduced the term **NoSQL** in early **2009** when he organized an event to discuss **"open source distributed, non relational databases".** The name attempted to label the emergence of an increasing number of non-relational, distributed data stores, including open source clones of Google's Bigtable/MapReduce and Amazon's DynamoDB.[2]

WWW.CIIIA.XYZ

NOSQL

NOSQL

Frequently, websites are always performing the same queries and transactions; the queries are very simple once they are de-normalized. In most of the cases, **the websites work with a simple API which manages CRUD operations.** Working with key/value storage is simple fast and scalable.

NOSQL

Frequently, websites are always performing the same queries and transactions; the queries are very simple once they are de-normalized. In most of the cases, **the websites work with a simple API which manages CRUD operations.** Working with key/value storage is simple fast and scalable.

NOSQL

NoSQL satisfies today's needs of handling huge amounts of data.

In the past, the problems related to handle big amount of data were solved vertically; getting more powerful machines.

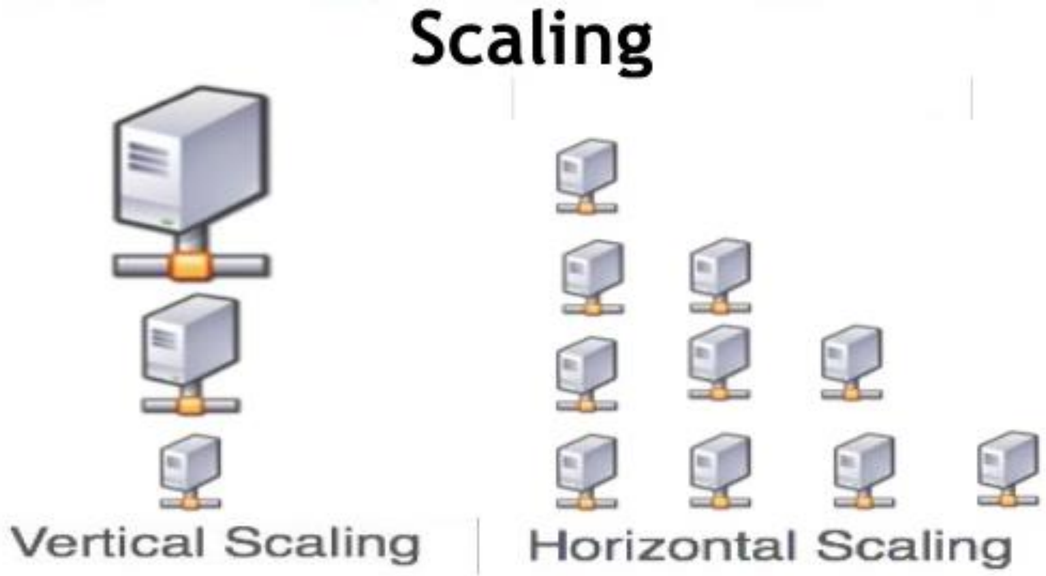
Nowadays, **we solve them** horizontally, with distributed systems; if our system is good enough, we add more machines (nodes) to our mesh.

NOSQL

NoSQL satisfies today's needs of handling huge amounts of data.

In the past, the problems related to handle big amount of data were solved vertically; getting more powerful machines.

Nowadays, **we solve them horizontally**, with distributed systems; if our system is good enough, we add more machines (nodes) to our mesh.



NOSQL -> CAP THEOREM

NOSQL -> CAP THEOREM

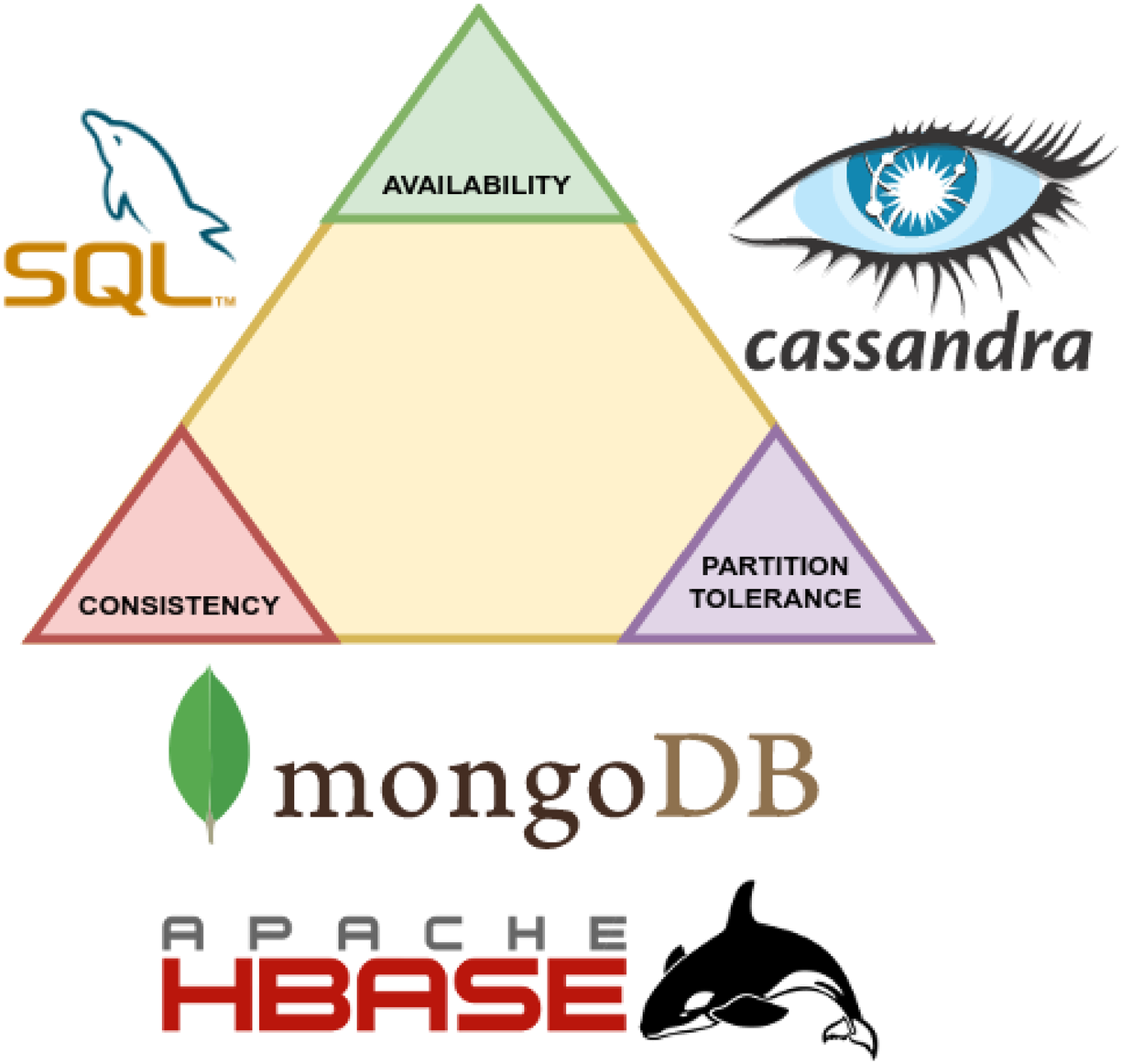
CAP stands for *Consistency, Availability and Partition Tolerance*. The CAP theorem says that we can only accomplish **2 out of these 3 requirements.**

Consistency is that when there's a **new value** (or it's updated) and it's immediately **available** (in the update case, we don't get the old value).

Availability means that the database is **always reliable, up and running**. This is achieved with redundancy.

Partition tolerance means that the database can be easily **split it up** and distributed across the cluster.

NOSQL -> CAP THEOREM



NOSQL -> CAP THEOREM

NOSQL -> CAP THEOREM

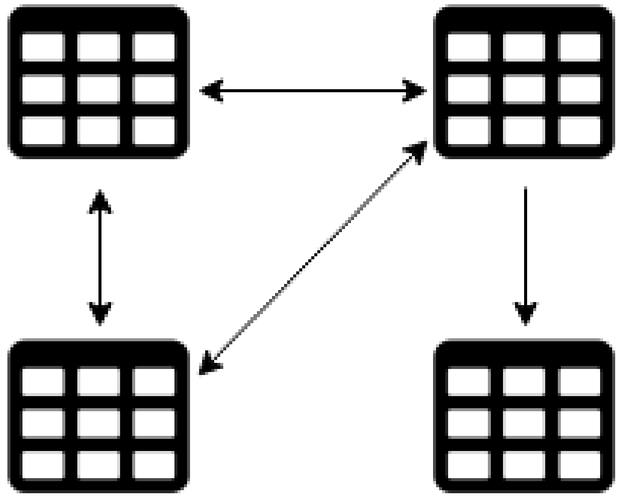
Probably, we all have experienced when we post something in a social network, we don't see our post, we publish it again and, few seconds later, we see both publications.

Now we can understand that social network is using a database system like Cassandra.

They have a distributed system (partition tolerance) and they prefer their system to be always running even if there are some data misbehaviour (availability over consistency).

RELATIONAL VS DOCUMENT DB

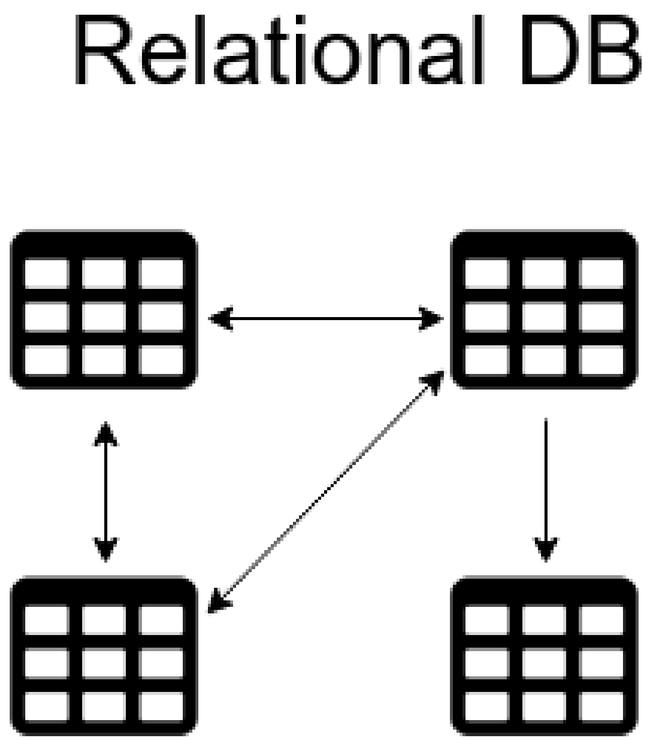
Relational DB



Document DB

RELATIONAL VS DOCUMENT DB

- To avoid redundancy (and therefore mistakes) the information it's split in normalized tables.
- The tables contain a fixed structure that is respected by all the rows of that table.
- The tables have relationships between them and each relation has cardinality.



RELATIONAL VS DOCUMENT DB

- No primary key is needed; although if we want to partition our database in shards, we need an index.
- Data it's stored in different documents. There's no need of join operations.
- Documents are independent from each other.
- Documents may have different schema.
- CRUD (Create Read Update Delete) operations are performed at document level.
- Pretty flexible; the only restriction is that you cannot move collections between different databases

WWW.CIIIA.XYZ

MONGODB

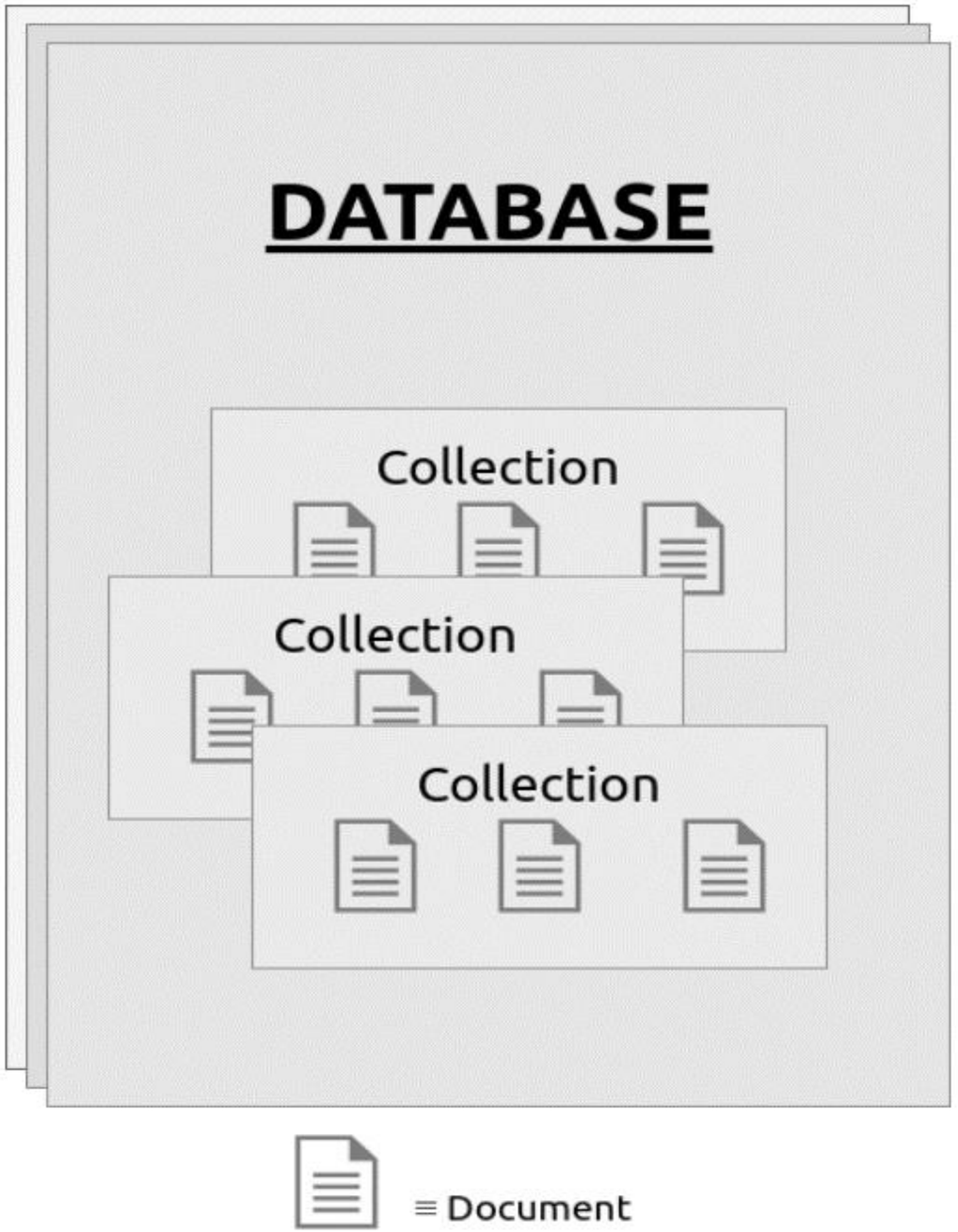
WWW.CIIIA.XYZ

MONGO DB= ?

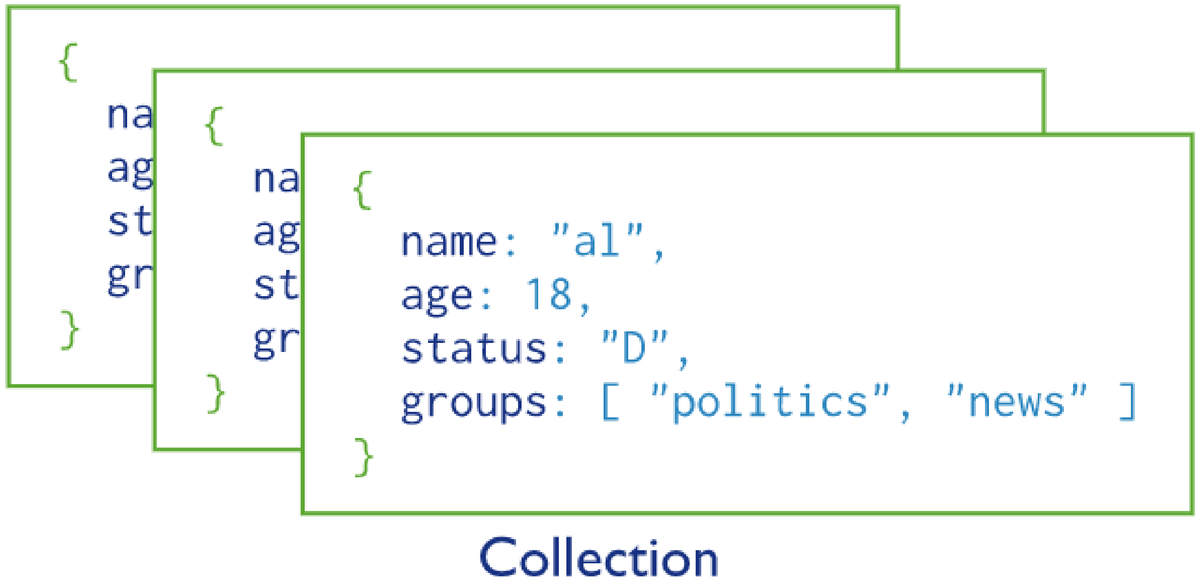
WWW.CIIIA.XYZ

Se recomienda si tu aplicación:

- Va a tener un crecimiento rapido
- Va tener servidores
- Rapidez en montar base de datos
- Estructura de los datos variables
- Información dinamica
- Gran demanda de usuarios accediendo al mismo tiempo.



Although it's not mandatory, usually, collections are groups of documents which share the same fields. I.e.: One collection for users, another collection for bills, etc...



MongoDB stores data records as BSON documents. BSON is a binary serialization format used to store documents and make remote procedure calls in MongoDB .

WWW.CIIIA.XYZ

CII.IA®

How to install?

Access to:
https://www.mongodb.com/download-center/community

Download the msi file

Install

Create a folder in C:// data and inside db
C://data/db

Execute in the folder of Mongo/ bin
First Mongod.exe and then Mongo.exe (server and program)
Done !!

If you have troubles you can also access to the mongo web shell
https://docs.mongodb.com/manual/tutorial/getting-started/

- Los documentos son una abstraccion/**descripcion de una entidad**/objeto de la vida real.
- Es un conjunto de pares clave-valor.
- Por ejemplo, definamos un documento para una PERSONA...

```
{
“nombre”: “Leonardo”,
“apellido”: “Kuffo”,
“id”: “0999999999”,
“edad”: 21
}
```

```
>use mitienda
>db
>show dbs
```

Create a collection implicit and insert a document

```
>db.usuarios.insert({
    “cedula”: “0926448614”
    “name”: “Leonardo Kuffo”,
    “clave”: “12345678”,
    “pais”: “Ecuador”
})

>show dbs
```

Create a collection implicit and insert a document

```
>db.createCollection("productos")

>show collections
```

Insert document

```
>db.productos.insert({
    “id”: “2”
    “nombre”: “Camiseta M”,
    “valor”: 15.0,
    “stock”: 5
})

>db.productos.find().pretty()
```

Insert document

```
>db.productos.insert({
    “id”: “2”
    “nombre”: “Camiseta M”,
    “valor”: 15.0,
    “stock”: 5
})
```

Insert document

```
>db.usuarios.insert({
    “cedula”: “0999999999”
    “name”: “Pepito”,
    “clave”: “12345678”,
    “pais”: “Chile”,
    “productos”: [1],
    “n_ordenes”: 1
})

>db.usuarios.find().pretty()
```

Update a document

```
db.productos.update({
  “id”: “1”
’
 {$set: {‘valor’: 20.45}}


db.productos.find().pretty()
```

Solo actualiza un document.

Si queremos actualizar todos los que cumplan con la condición agregamos un tercer parámetro

{multi:true}

Consultas

```
>db.productos.find({
    “valor”: 15.0
})

>db.productos.find({
    “valor”: {$lt: 16.0}
})
```

```
> db.productos2.find($or:[{"stock":{$gt:40}
},{"stock":{$lt:500}}])
```

Consultas

Operacion	Ejemplo
Igualdad	`{ "stock": 0 }`
Menor Que	`{ "valor": {$lt: 15.0} }`
Menor o Igual Que	`{ "valor": {$lte: 16.0} }`
Mayor Que	`{ "valor": {$gt: 18.0} }`
Mayor o Igual Que	`{ "valor": {$gte: 16.0} }`
No es Igual	`{ "valor": {$ne: 0} }`
AND	`{ {key1: value1, key2:value2} }`

Consultas

Operacion	Ejemplo
OR	`{ $or: [ {key1: value1}, {key2:value2} ] }`
AND + OR	`{ key1: value1, $or: [ { key2: {$lt: value2}, {key3: value3} } ] }`

Consultas

```
>db.usuarios.find().pretty()

>db.usuarios.findOne().pretty()
```

Consultas

```
>db.productos.find({
    "valor": 15.0
})

>db.productos.find({
    "valor": {$lt: 16.0}
})
```

Consultas

```
>db.productos.find().limit(1)

>db.productos.find().sort({valor:1})
```

Consultas

1: De menor a mayor
-1: De menor a mayor

Eliminar uno doc

```
>db.productos.deleteOne({
    “id”: “1”
})

>db.productos.find().pretty()
```

Delete collection

```
>db.productos.drop()
>show collections
```

Delete data base

```
›use mitienda
›db.dropDatabase()

›show dbs
```

WWW.CIIIA.XYZ

THANK YOU

#SHORTCUTTOTHEFUTURE

CII.IA®