

Bases de datos NOSQL

Breve historia

Carlo Strozzi usó el término NoSQL en 1998 para referirse a su base de datos (open-source, ligera, sin interface SQL, pero que sí seguía el modelo relacional).

Eric Evans, un empleado de Rackspace, reintrodujo en 2009 el término NoSQL cuando Johan Oskarsson de [Last.fm](https://last.fm) quiso organizar un evento para discutir bases de datos distribuidas de código abierto. El nombre intentaba recoger el número creciente de bases de datos no relacionales y distribuidas que no garantizaban ACID, atributo clave en las SGBDR clásicas.

Bases de datos NOSQL

Clasificación

- Clave-valor



redis



- Orientadas a grafos



- Orientadas a documentos



mongoDB

- Familias de columnas



Bases de datos NOSQL

Clasificación

All in the NoSQL Family

NoSQL databases are geared toward managing large sets of varied and frequently updated data, often in distributed systems or the cloud. They avoid the rigid schemas associated with relational databases. But the architectures themselves vary and are separated into four primary classifications, although types are blending over time.



Document databases

Store data elements in document-like structures that encode information in formats such as JSON.



Common uses include content management and monitoring Web and mobile applications.



EXAMPLES:

Couchbase Server, CouchDB, MarkLogic, MongoDB



Graph databases

Emphasize connections between data elements, storing related "nodes" in graphs to accelerate querying.



Common uses include recommendation engines and geospatial applications.



EXAMPLES:

Allegrograph, IBM Graph, Neo4j



Key-value databases

Use a simple data model that pairs a unique key and its associated value in storing data elements.



Common uses include storing clickstream data and application logs.



EXAMPLES:

Aerospike, DynamoDB, Redis, Riak



Wide column stores

Also called table-style databases—store data across tables that can have very large numbers of columns.



Common uses include Internet search and other large-scale Web applications.



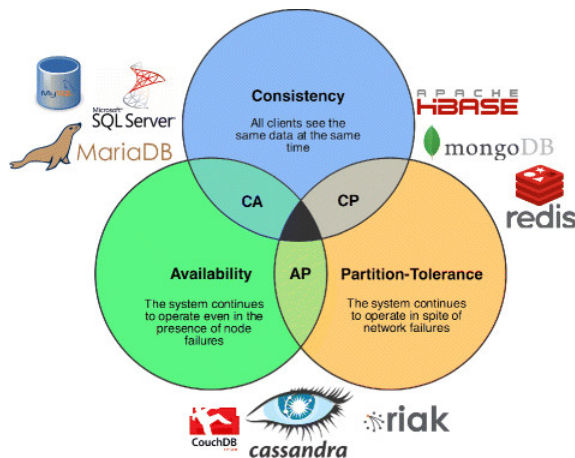
EXAMPLES:

Accumulo, Cassandra, HBase, Hypertable, SimpleDB

Bases de datos NOSQL

Conjetura o Teorema de Brewer (CAP)

- “If you cannot limit the number of faults and requests directed to any server and you insist in serving every request you receive then you cannot possibly be consistent”.



Bases de datos *NOSQL*

Conjetura o Teorema de Brewer - Consistency

- Todas las réplicas tienen la misma versión de los datos.
- El cliente siempre ve la misma información sin importar el nodo.
- Ante un evento de falla (partición), si se desea asegurar la consistencia entonces el sistema devolverá un error si no puede asegurar la misma información en cualquier nodo (en otras palabras no hay disponibilidad).

Bases de datos NOSQL

Conjetura o Teorema de Brewer – Availability

- El sistema permanece activo aún con nodos fallando.
- Todos los clientes tienen la posibilidad de escribir y de leer.
- En este caso, ante un evento de falla se elige la disponibilidad por sobre la consistencia (no es importante que todos los clientes reciban exactamente la misma información de todos los nodos).

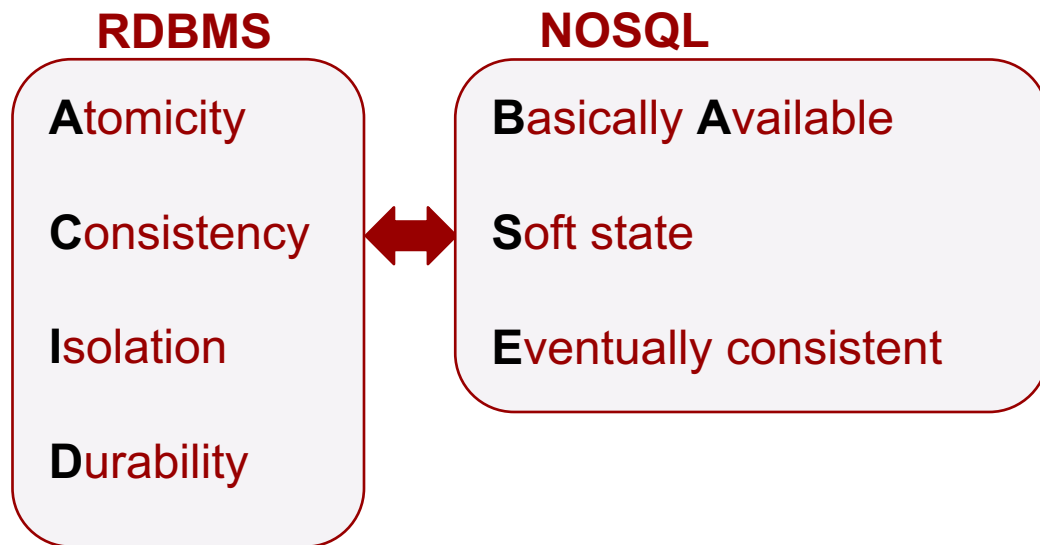
Bases de datos NOSQL

Conjetura o Teorema de Brewer – Partition Tolerance

- El sistema presenta múltiples puntos de entrada debido a fallas en la red que conecta los nodos.
- Dos alternativas posibles:
 - Si se prefiere la consistencia, entonces se pierde la disponibilidad.
 - Si se requiere disponibilidad, entonces se cede la consistencia.

Bases de datos NOSQL

Diferencias entre el modelo transaccional de bases de datos relacionales y NOSQL



MongoDB

Principales características

- Base de datos NOSQL orientada a documentos.
- Basada en hashing.
- No maneja la noción de esquemas.
- Carece de un lenguaje DDL.
- El formato de almacenamiento es BSON (Binary JSON).
- Ofrece drivers para la mayoría de los lenguajes.

MongoDB

BSON

- Es un formato de codificación para documentos tipo JSON (aunque contiene extensiones propias).
- Fue diseñado con 3 objetivos en mente:
 - Liviano: trata de mantener el espacio ocupado al mínimo.
 - Recorrible: debe ser posible recorrerlo fácilmente. Esto es fundamental ya que el formato primario de representación para MongoDB.
 - Eficiente: la codificación/decodificación puede ser realizada muy rápidamente ya que se basa en tipos de datos de C.

MongoDB

Un ejemplo simple de un documento

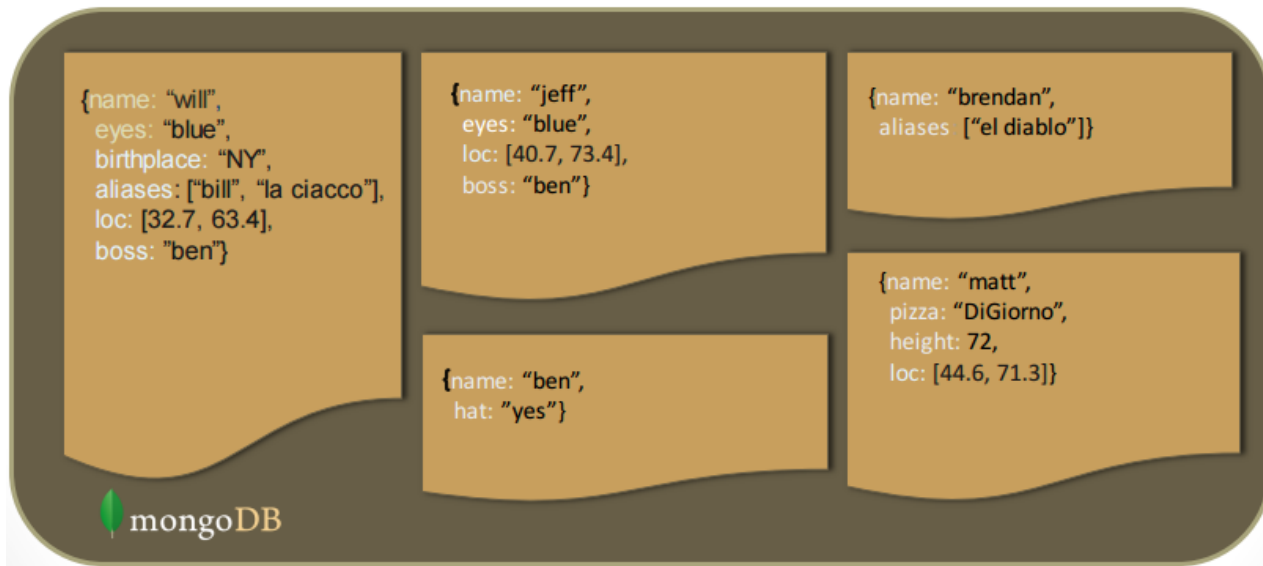
- Formato JSON

```
{  
  "day": [ 2010, 01, 23 ],  
  "products": {  
    "apple": { "price": 10 "quantity": 6 },  
    "kiwi": { "price": 20 "quantity": 2 }  
  },  
  "checkout": 100  
}
```

MongoDB

Carencia de esquemas

- Cada documento puede tener su propia “estructura”



MongoDB

Analogías entre bases de datos relacionales y MongoDB

- Principales conceptos

RDBMS

Base de datos

Tabla

Fila

Columna

Índice

Join

Clave foránea

MongoDB

Base de datos

Colección

Documento

Campo

Índice

Documento embebido

Referencia

MongoDB

Organización conceptual

- Una instancia de MongoDB tiene 0 o más bases de datos.
- Una base de datos tiene 0 o más colecciones.
- Una colección tiene 0 o más documentos.
- Un documento tiene 0 o más campos.



MongoDB

MongoDB soporta las habituales operaciones C.R.U.D.

- Create: inserta nuevos documentos
 - Read: permite recuperar documentos de una colección
 - Update: modifica un documento existente
 - Delete: elimina un documento
-
- Estas operaciones se pueden realizar a través de la consola, de librerías o de herramientas como Robo3T.



MongoDB

CREATE

- Inserta nuevos documentos.
- Si la colección no existe, entonces la crea previamente.

- `db.collection.insertOne()`
- `db.collection.insertMany()`

```
db.users.insertOne(  ← collection
{
  name: "sue",        ← field: value
  age: 26,             ← field: value
  status: "pending"   ← field: value
}
)
```


MongoDB

READ

- Permite recuperar documentos.

- `db.collection.find()`

```
db.users.find(  
  { age: { $gt: 18 } },  
  { name: 1, address: 1 }  
) .limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

MongoDB

UPDATE

- Permite modificar documentos.
- Las modificaciones apuntan a una única colección.
- Cada documento se modifica atómicamente.

- `db.collection.updateOne()`
- `db.collection.updateMany()`
- `db.collection.replaceOne()`

```
db.users.updateMany(  
  { age: { $lt: 18 } },  
  { $set: { status: "reject" } }  
)
```

← collection
← update filter
← update action

MongoDB

DELETE

- Permite eliminar documentos.
- Las eliminaciones afectan a una única colección.
- Cada documento se modifica atómicamente.

- `db.collection.deleteOne()`

- `db.collection.deleteMany()`

```
db.users.deleteMany(  
  { status: "reject" }  
)
```

← collection
← delete filter

MongoDB

Índices

- Los índices se definen a nivel de las “collections”.
- Se soportan índices por cualquier campo o subcampo del documento.
- MongoDB crea un índice sobre el campo `_id` durante la creación de una colección. Este índice controla la unicidad del valor del `_id`. No se puede eliminar este índice.
- Especificación
 - `db.collection.createIndex(<key and index type specification>, <options>)`
- Ejemplo
 - `db.collection.createIndex({ name: -1 })`

MongoDB

Tipos de índices

- Single field
- Compound index
- Multikey index
- Geospatial index
- Text index
- Hashed index

MongoDB

Tipos de índices

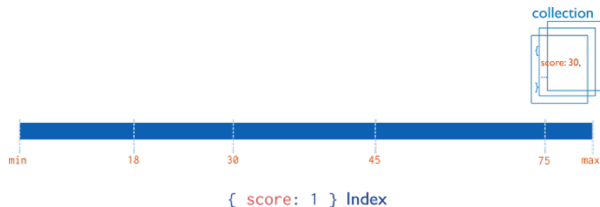
- Single field

- `db.collection.createIndex(<key and index type specification>, <options>)`

- `db.records.createIndex({ score: 1 })`

- `db.records.createIndex({ "location.state": 1 })`

```
{
  "_id": ObjectId("570c04a4ad233577f97dc459"),
  "score": 1034,
  "location": { state: "NY", city: "New York" }
}
```



MongoDB

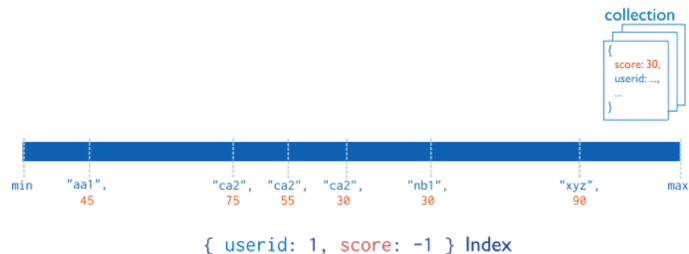
Tipos de índices

- Compound index

- `db.collection.createIndex({ <field1>: <type>, <field2>: <type2>, ... })`

- `db.products.createIndex({ "item": 1, "stock": 1 })`

```
{
  "_id": ObjectId(...),
  "item": "Banana",
  "category": ["food", "produce", "grocery"],
  "location": "4th Street Store",
  "stock": 4,
  "type": "cases"
}
```



MongoDB

Tipos de índices

- Multikey index
 - `db.coll.createIndex({ <field>: < 1 or -1 > })`



`{ "addr.zip": 1 } Index`

MongoDB

Tipos de índices

- GeoSpatial
- Utilizan datos tanto en formato de par de coordenadas como GeoJSON.
 - `db.collection.createIndex({ <location field> : "2dsphere" })`
 - `db.places.createIndex({ loc : "2dsphere" })`
 - `db.places.createIndex({ loc : "2dsphere" , category : -1, name: 1 })`

```
db.places.insert(  
  {  
    loc : { type: "Point", coordinates: [ -73.88, 40.78 ] },  
    name: "La Guardia Airport",  
    category : "Airport"  
  }  
)
```

MongoDB

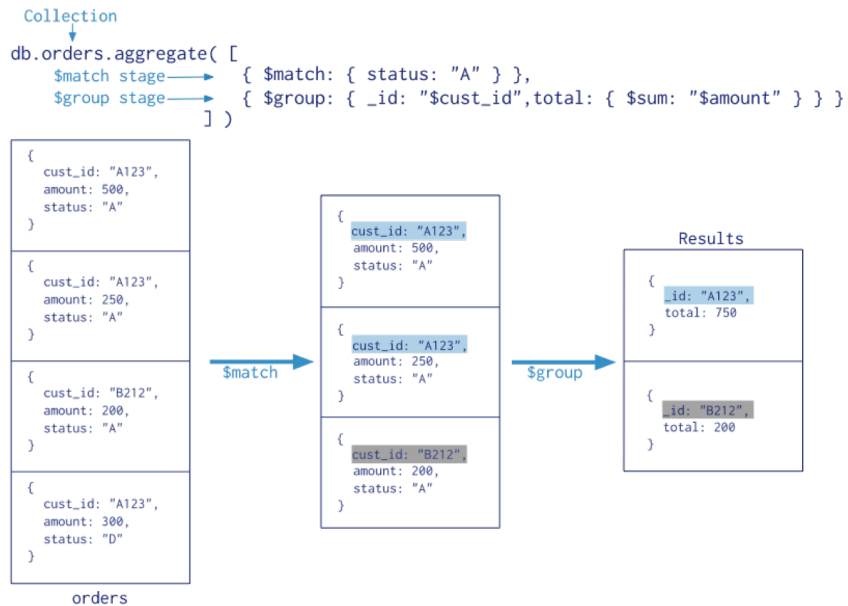
Tipos de índices

- Text
 - Una colección puede tener a lo sumo un índice de texto.
 - `db.reviews.createIndex({ comments: "text" })`
 - `db.reviews.createIndex({ subject: "text", comments: "text" })`
 - `db.collection.createIndex({ "$**": "text" })`
 - `db.quotes.createIndex({ content : "text" }, { default_language: "spanish" })`

MongoDB

Operaciones de agregación

- Procesan datos y devuelven resultados



MongoDB

Operaciones MapReduce

- En general consta de dos fases
 - Una fase “map” en la cual se procesan los datos y se “emite” un resultado parcial
 - Una fase “reduce” que combina el resultado en un solo resultado final

```
Collection
↓
db.orders.mapReduce(
  map   → function() { emit( this.cust_id, this.amount ); },
  reduce → function(key, values) { return Array.sum( values ) },
  query → { query: { status: "A" },
  output → { out: "order_totals" }
)
```

