MongoDB

BASE DE DATOS NOSQL

Contenidos

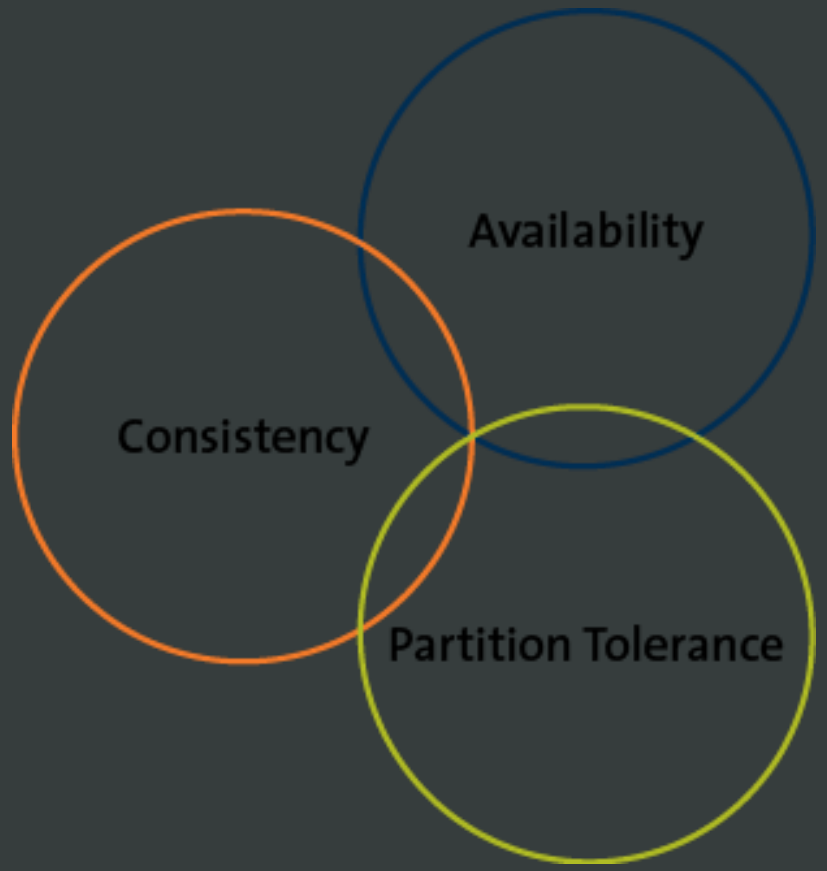
- Bases de Datos NoSQL
- MongoDB

Qué es NoSQL

- NoSQL no es un estándar
- NoSQL no significa “No SQL” sino “Not Only SQL”
- No es un sustituto de los RDBMS (SGBDR)
- Relacionado con el teorema CAP
- BASE vs ACID

NoSQL

- Teorema CAP – Eric Brewer
  - **Consistency –** Consistencia
  - **Availability –** Disponibilidad
  - **Partition Tolerance –** Particionado
- Solo podemos garantizar dos de tres



NoSQL

Consistencia – C	Disponibilidad – A	Particionado – P
Lecturas ➔ leen la última escritura Escrituras ➔ escriben o error	Cada petición recibe una respuesta (sin error) No garantiza que contenga la última escritura	El sistema funciona entre diferentes nodos No sabemos donde estamos leyendo o escribiendo Pueden haber diferentes valores en cada nodo

Los SGBDR garantizan CA gracias a las transacciones **ACID** (Atomicity, Consistenty, Isolation, Durability)

Los sistemas NoSQL podrán garantizar AP o CP

NoSQL

NoSQL	SQL
– Sin esquema o esquema ligero	– Esquema rígido
– Escalado horizontal dinámico	– Estático o sin escalado horizontal
– Bueno grandes cantidades de datos	– Bueno para datos relacionados
– Gran flexibilidad	– Flexibilidad media
– No es necesario ACID completo	– ACID completo
– BASE • **B**asically **A**vailable • **S**oft State • **E**ventually Consistently	
– Objetos: colecciones, documentos, …	– Objetos: tablas, filas, columnas

Ranking Bases de datos

- Ranking

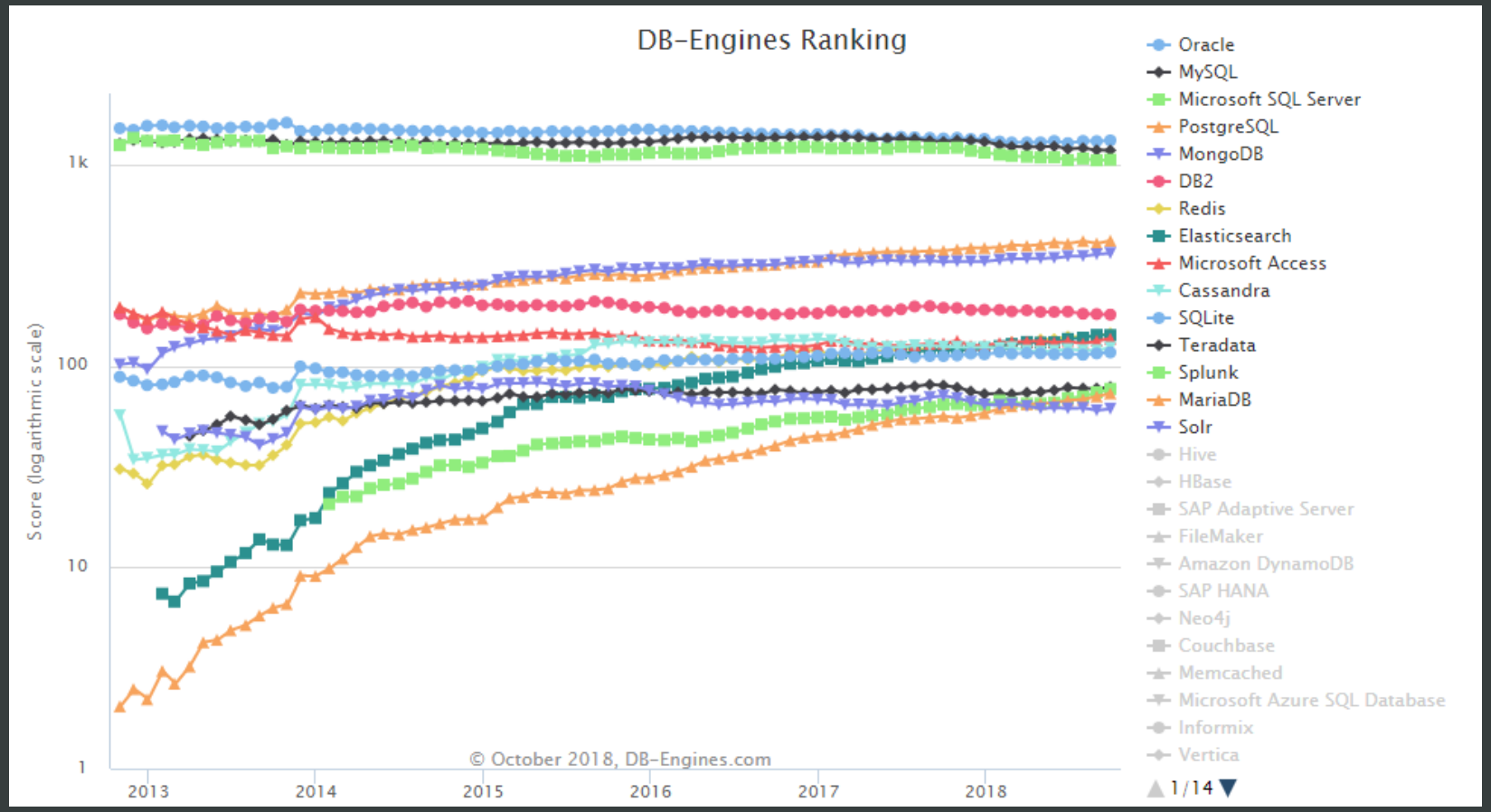
https://db-engines.com/en/ranking

Rank Oct 2018	Rank Sep 2018	Rank Oct 2017	DBMS	Database Model	Score Oct 2018	Score Sep 2018	Score Oct 2017
1.	1.	1.	Oracle	Relational DBMS	1319.27	+10.15	-29.54
2.	2.	2.	MySQL	Relational DBMS	1178.12	-2.36	-120.71
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1058.33	+7.05	-151.99
4.	4.	4.	PostgreSQL	Relational DBMS	419.39	+12.97	+46.12
5.	5.	5.	MongoDB	Document store	363.19	+4.39	+33.79
6.	6.	6.	DB2	Relational DBMS	179.69	-1.38	-14.90
7.	↑ 8.	↑ 9.	Redis	Key-value store	145.29	+4.35	+23.24
8.	↓ 7.	↑ 10.	Elasticsearch	Search engine	142.33	-0.28	+22.09
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	136.80	+3.41	+7.35
10.	10.	↓ 8.	Cassandra	Wide column store	123.39	+3.83	-1.40

Ranking bases de datos

- Ranking

https://db-engines.com/en/ranking_trend



Sistemas NoSQL

Tipos de sistemas NoSQL

Documentales	De grafo	Clave / Valor
Multi-valor	Orientadas a Objetos	Series temporales
Motores de búsqueda	Columna ancha	RDF
Contenido	Eventos	Navegación

Bases de datos documentales

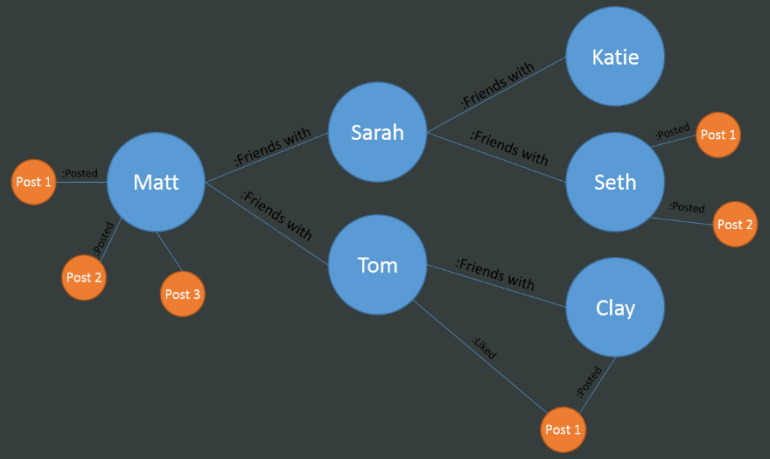
- Características
  - **Documento** = unidad mínima de información
  - En diferentes formatos: XML, YAML, JSON, BSON
  - Identificados por una **clave única**
  - Búsquedas basadas en clave
  - Lenguaje de consultas permite buscar en el propio documento a través de una API
  - Organización de documentos mediante:
    - Colecciones
    - Etiquetas
    - Metadatos
    - Jerarquías

Principales BD

1. **MongoDB**
2. **Amazon DynamoDB**
3. **Couchbase**

Bases de datos de grafos

- Características
  - Almacena tres elementos principales:
    - **Nodos**
      - Representa entidades
    - **Extremos**
      - Representa relaciones
    - **Propiedades**
      - Representa información del nodo o de la relación



Principales BD

1. **Neo4j**
2. **Titan**
3. **Giraph**

Bases de datos clave-valor

- Características
  - Almacena un **diccionario** o hash con dos componentes:
    - **Clave**: identificador del objeto
    - **Valor**: detalles del objeto
  - Pueden ser consistentes en diferentes niveles
    - **Eventualmente consistentes**
    - Ordenados
    - RAM – optimizado en flash
    - SSD – optimizado en discos de estado sólido

Principales BD

1. **Redis**
2. **Memcached**
3. **Hazelcast**

Bases de datos multi-valor

- Características
  - Las columnas de las tablas pueden almacenar diferentes valores
  - Reduce el de tablas relacionadas
  - PostgreSQL permite crear columnas de tipo array de valores

ID	Nombre	E-mail	Teléfono
1111	Antonio	antonio@correo.es	666555444
			999888777
1112	Juan	juan@correo.es	678876678

Principales BD
1. **Adabas**
2. **UniData, UniVerse (U2)**
3. **jBASE**

Bases de datos orientadas a objetos

- Características
  - Combina las características de:
    - Bases de datos
    - Programación Orientada a Objetos (OOP)
  - Al utilizar un lenguaje de programación orientada a objetos:
    - Mantenemos un único modelo de datos
    - La gestión de estados de los objetos es más eficiente
    - Las capacidades de sincronización están optimizadas

Principales BD

1. **ObjectStore**
2. **Versant Object Database**
3. **Matisse**

Bases de datos de series temporales

- Características
  - Optimizada para el almacenamiento y la gestión de:
    - Datos en series de tiempos
    - Arrays o números indexados por tiempo
  - Series de tiempos conocidas como:
    - Perfiles
    - Curvas
    - Trazas
  - Facilita:
    - La consulta de datos filtrados por tiempo

Principales BD
1. **InfluxDB**
2. **RRDtool**
3. **Graphite**

Bases de datos de tipo motores de búsqueda

- Características
  - Motores de búsqueda que funciona como almacén de objetos en una base de datos
- Categorías:
  - Búsqueda Web
  - Búsqueda de datos estructurados
  - Búsqueda empresarial

Principales BD

1. **Elasticsearch**
2. **Solr**
3. **Splunk**

Bases de datos de columna ancha (wide-column)

- Características
  - Es un tipo de base de datos clave-valor
  - Almacena datos en:
    - Tablas
    - Filas
    - Columnas
  - Los nombres y los formatos de las columnas pueden variar en cada fila

Principales BD

1. **Cassandra**
2. **HBase**
3. **Accumulo**

Bases de datos de RDF

- Características
  - Base de datos para el almacenamiento de elementos triples
  - RDF = Resource Description Framework
  - RDF
    - Entidad de datos compuesta por:
      - Sujeto
      - Predicado
      - Objeto
    - Ejemplo:
      - Juan compra Producto

Principales BD

1. **Jena**
2. **Algebraix**
3. **4store**

Bases de datos almacenes de contenido

- Características
  - También llamados repositorios
  - Almacenan contenido digital:
    - Textos
    - Imágenes
    - Vídeos
    - Metadatos
  - Soportan búsquedas
    - Full-Text
    - Versionado
    - Jerarquías

Principales BD
1. **Jackrabbit**
2. **ModeShape**

Bases de datos de eventos

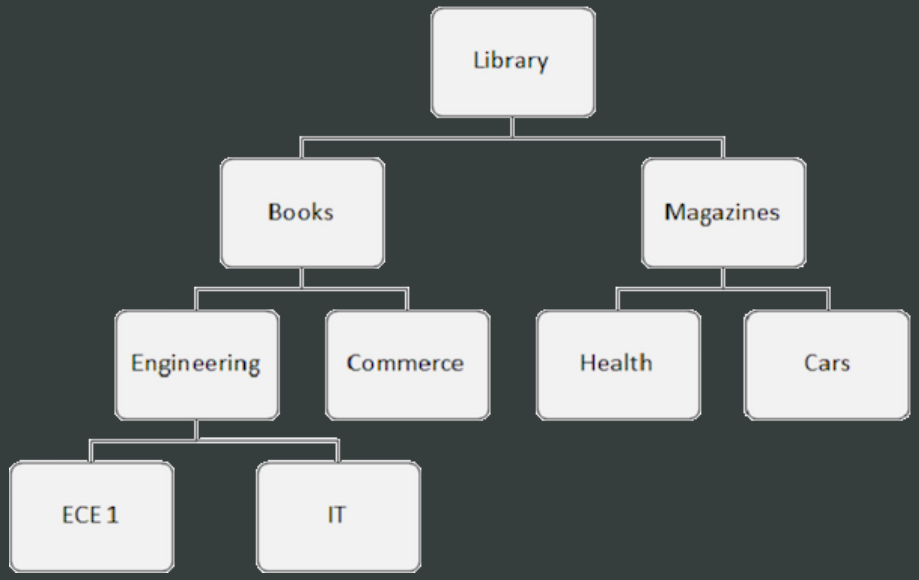
- Características
  - Almacenan estados de objetos y los eventos de cambio (histórico)
  - Soporta operaciones:
    - Modificación
    - Consulta sobre líneas de tiempo
  - Ofrece una buena gestión de snapshots:
    - Estado de objetos en un punto concreto de tiempo

Principales BD

1. **Event Store**
2. **NEventStore**

Bases de datos de navegación

- Características
  - Permite el acceso a conjuntos de datos usando registros enlazados
  - Dos tipos de implementaciones principales:
    - Jerárquicas
    - Topología de red



Principales BD
1. **IMS**
2. **IDMS**

MongoDB

Visión general

Ranking Bases de datos

- Ranking

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

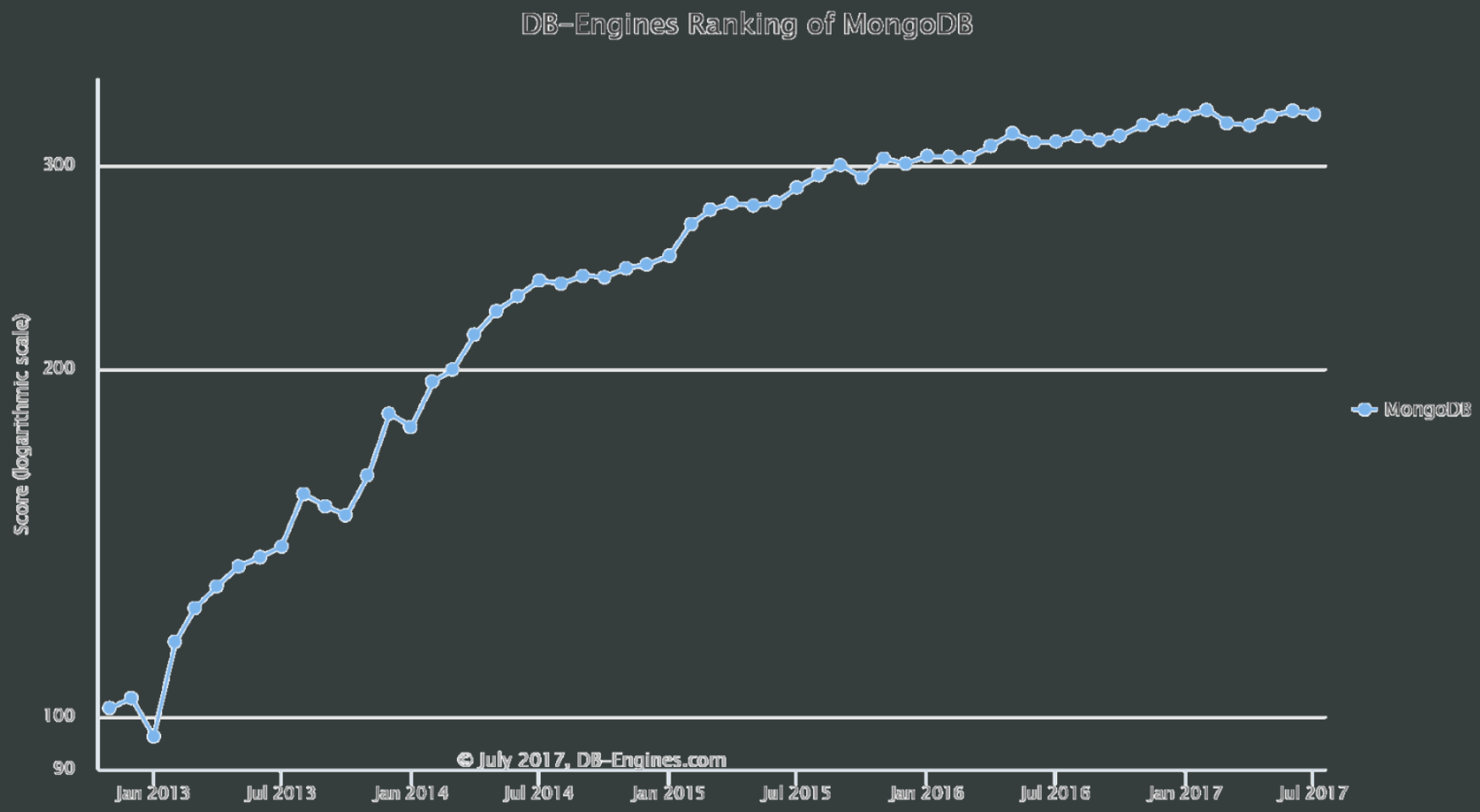
328 systems in ranking, July 2017

Rank Jul 2017	Rank Jun 2017	Rank Jul 2016	DBMS	Database Model	Score Jul 2017	Score Jun 2017	Score Jul 2016
1.	1.	1.	Oracle	Relational DBMS	1374.88	+23.11	-66.65
2.	2.	2.	MySQL	Relational DBMS	1349.11	+3.80	-14.18
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1226.00	+27.03	+33.11
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	369.44	+0.89	+58.28
5.	5.	↓ 4.	MongoDB	Document store	332.77	-2.23	+17.77
6.	6.	6.	DB2	Relational DBMS	191.25	+3.74	+6.17
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	126.13	-0.42	+1.23
8.	8.	↓ 7.	Cassandra	Wide column store	124.12	-0.00	-6.58
9.	9.	↑ 10.	Redis	Key-value store	121.51	+2.63	+13.48
10.	↑ 11.	↑ 11.	Elasticsearch	Search engine	115.98	+4.42	+27.36

Ranking bases de datos

- Ranking

https://db-engines.com/en/ranking_trend/system/MongoDB



Tipos de sistemas NoSQL

Documentales	**De grafo**	**Clave / Valor**
Multi-valor	**Orientadas a Objetos**	**Series temporales**
Motores de búsqueda	**Columna ancha**	**RDF**
Contenido	**Eventos**	**Navegación**

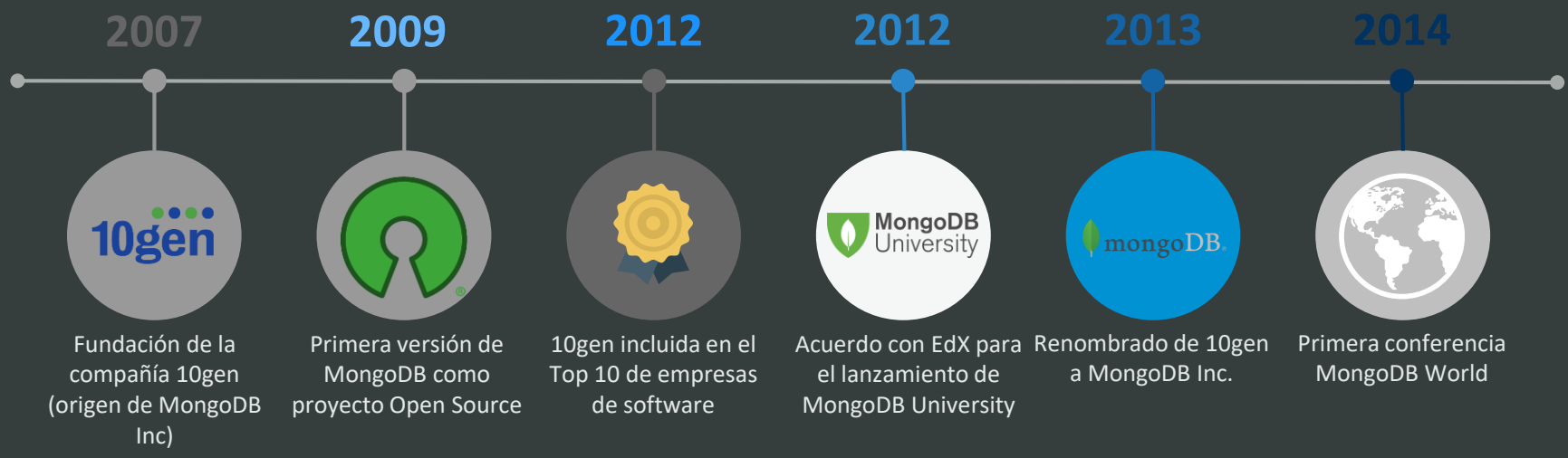
Bases de datos documentales

- Características
  - **Documento** = unidad mínima de información
  - En diferentes formatos: XML, YAML, JSON, BSON
  - Identificados por una **clave única**
  - Búsquedas basadas en clave
  - Lenguaje de consultas permite buscar en el propio documento a través de una API
  - Organización de documentos mediante:
    - Colecciones
    - Etiquetas
    - Metadatos
    - Jerarquías

Principales BD
1. **MongoDB**
2. **Amazon DynamoDB**
3. **Couchbase**

MongoDB

- Orígenes
  - Open Source
  - Gestionado por **MongoDB Inc.**
  - Versión actual 4.0.3 (a 22 de julio de 2017)



Arquitectura de MongoDB

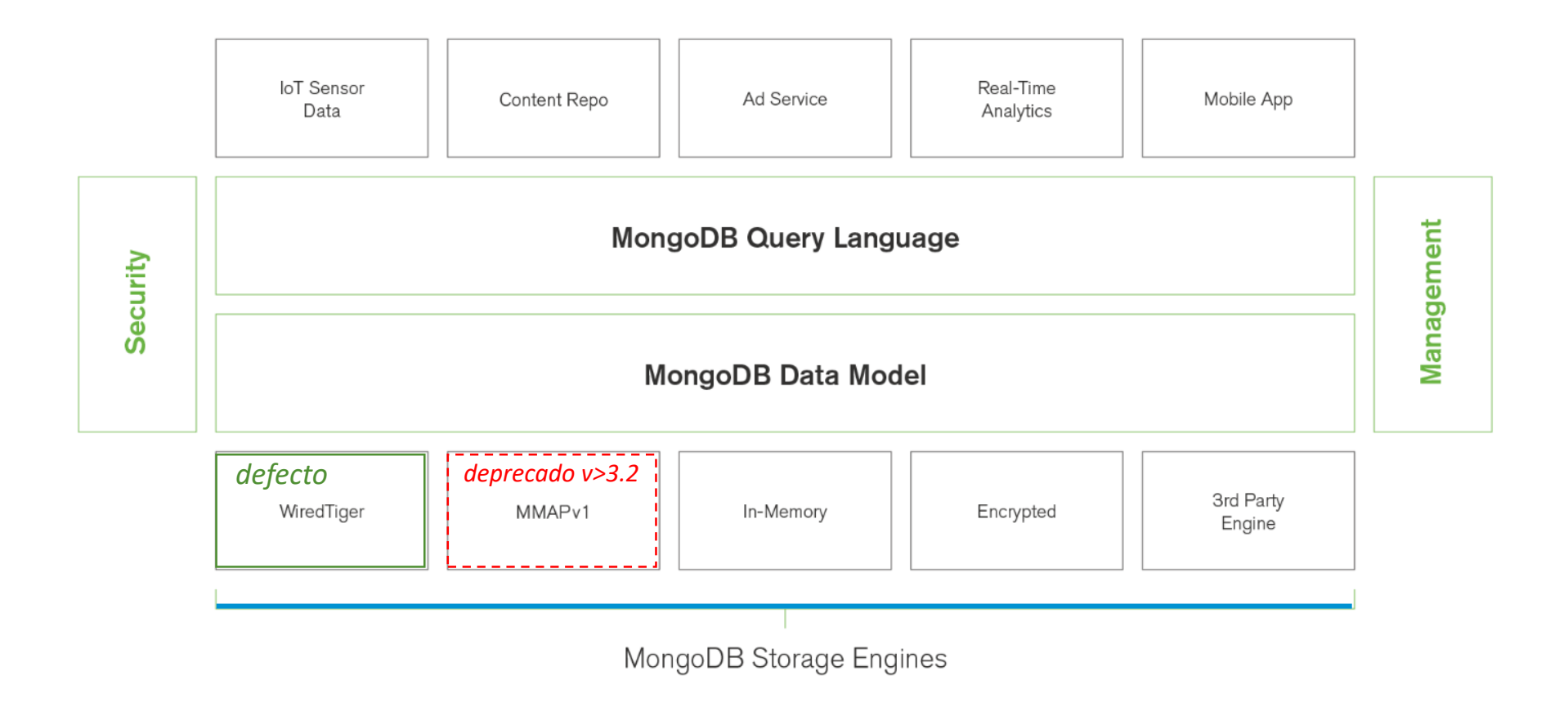
- Características
  - Base de datos NoSQL documental
  - Open Source y alto rendimiento
  - Consultas a partir de documentos ➔ **Fácil lectura**
  - Soporte completo de índices ➔ **Alto rendimiento**
  - Replicación y failover ➔ **Alta disponibilidad**
  - Auto Sharding ➔ **Escalabilidad sencilla**
  - Map / Reduce ➔ **Agregación**
  - Disponible en www.mongodb.com

MongoDB

- SQL almacena **datos** ← → MongoDB almacena **documentos (o objetos)**
- Las aplicaciones trabajan con **objetos**
- Necesitamos almacenar **objetos** → Almacenemos objetos
- Tenemos los detalles completos del objeto → Reducimos la necesidad de JOINS
- Transacciones
  - **SIN** joins
  - **SIN** multi-documentos

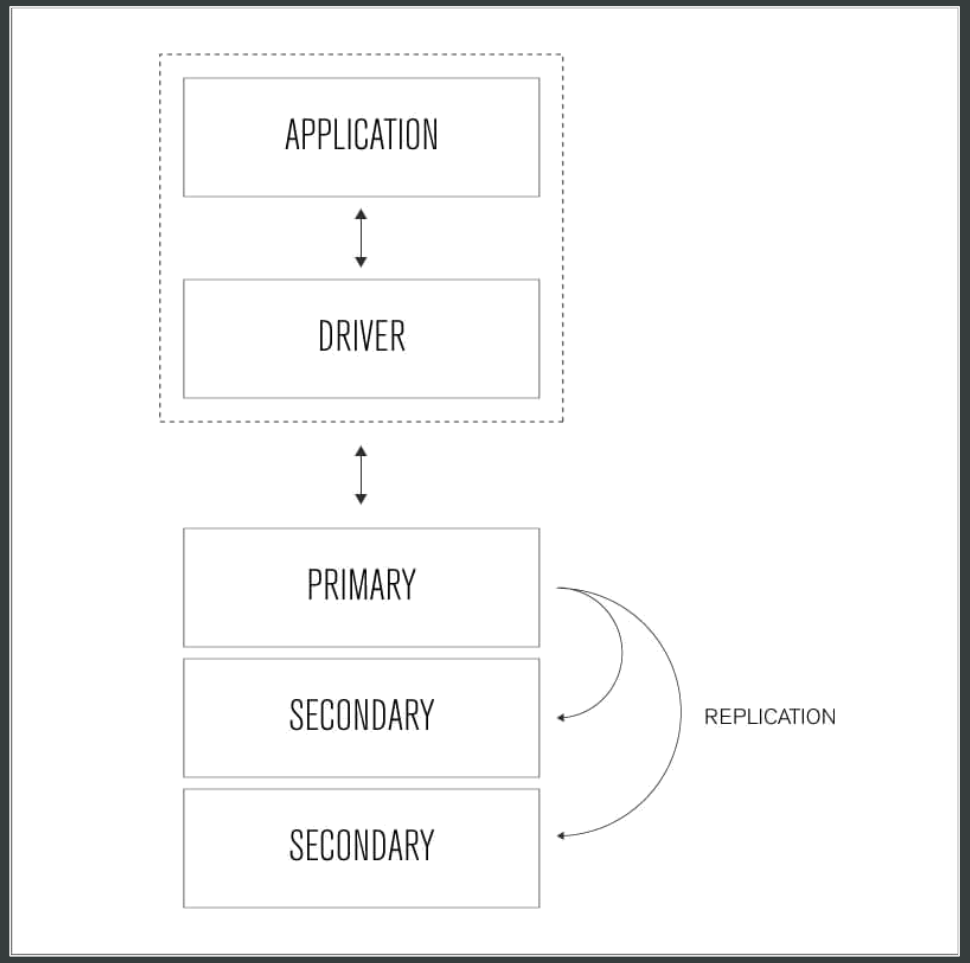
- Buen funcionamiento
  - Como reemplazo de SGBDR para apps web
  - Gestión de contenidos **semi-estructurados**
  - Analítica en **tiempo real**
  - Logging de **alta velocidad**
  - Cacheado y **alta disponibilidad**
- Mal funcionamiento
  - Aplicaciones altamente **transaccionales**
  - Necesidades alineadas con consultas SQL

Arquitectura de almacenamiento

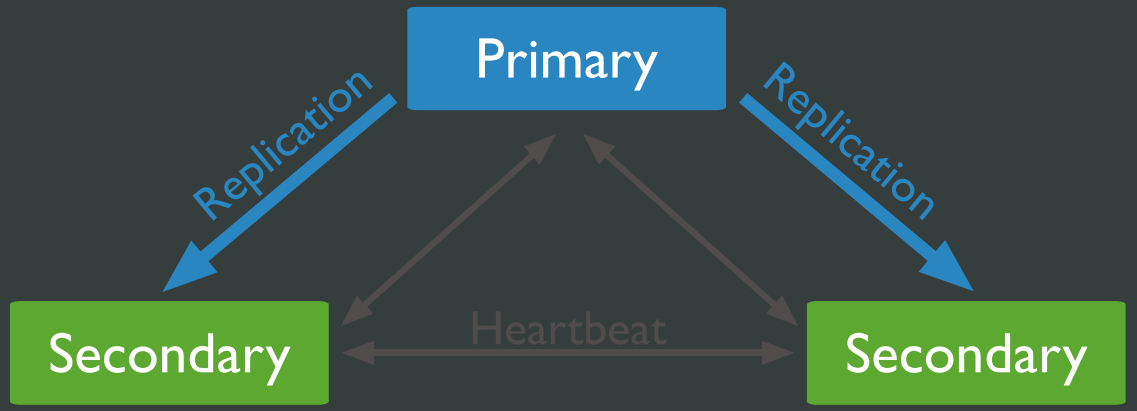


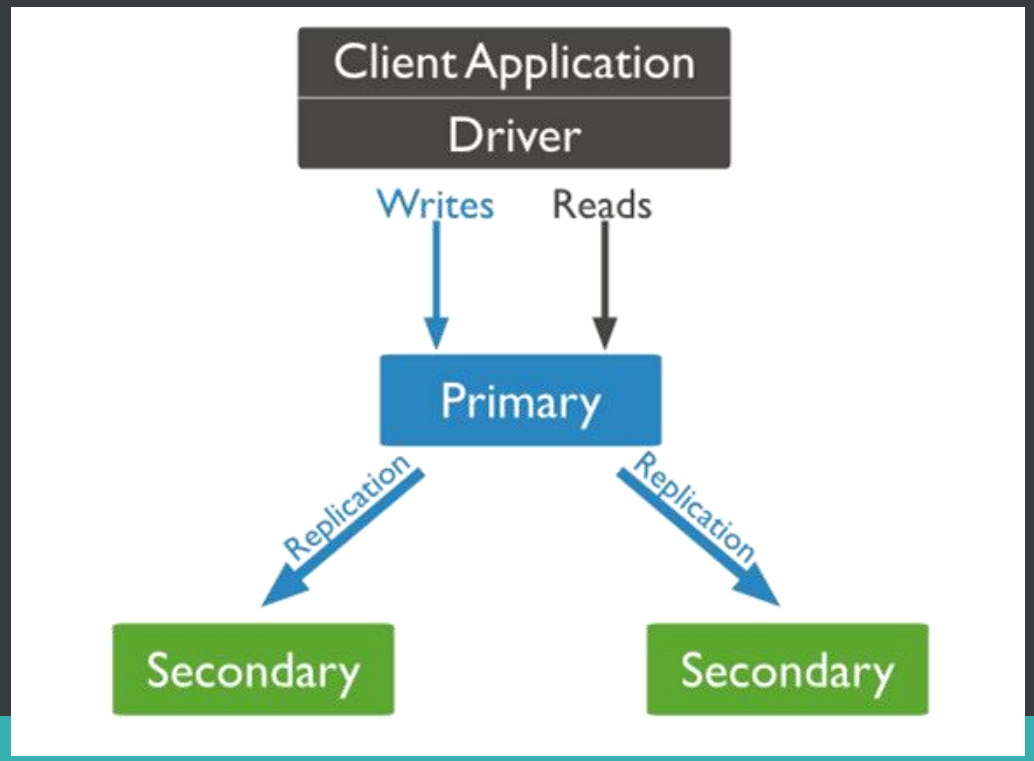
Conjuntos de réplica

- Da soporte a redundancia y alta disponibilidad:
  - **Conjunto de réplica**
    - Formado por instancias
      - Primario
      - Secundarios
  - **Primario**
    - Acepta todas las operaciones
    - Si falla se elige nuevo primario
  - **Secundarios**
    - Reflejos del primario
    - Actualización asíncrona
    - Puede aceptar lecturas
    - No acepta escrituras

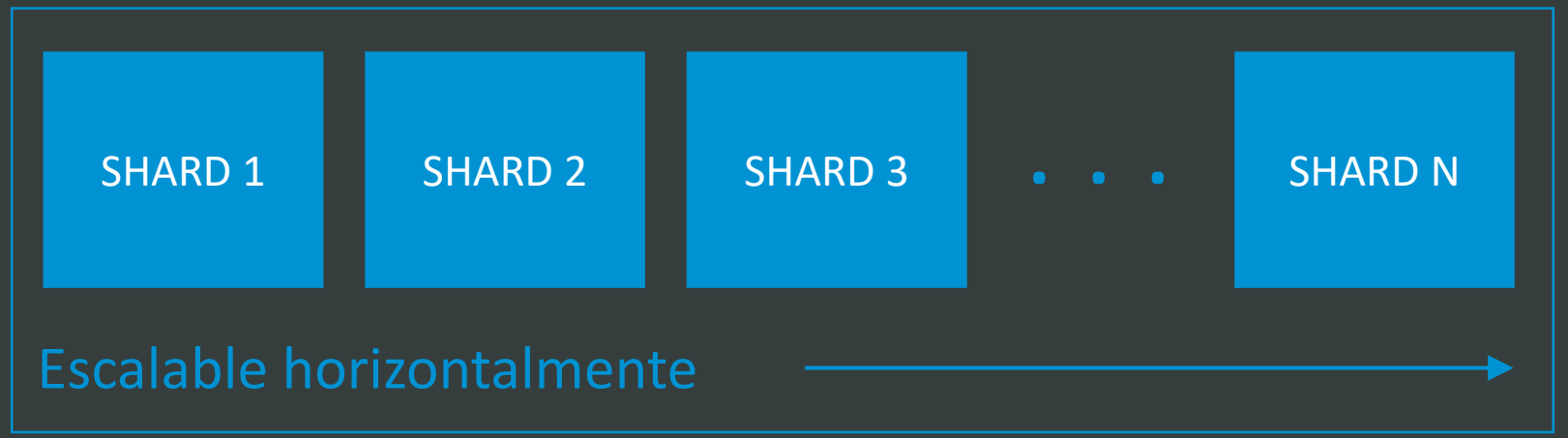


Replicación



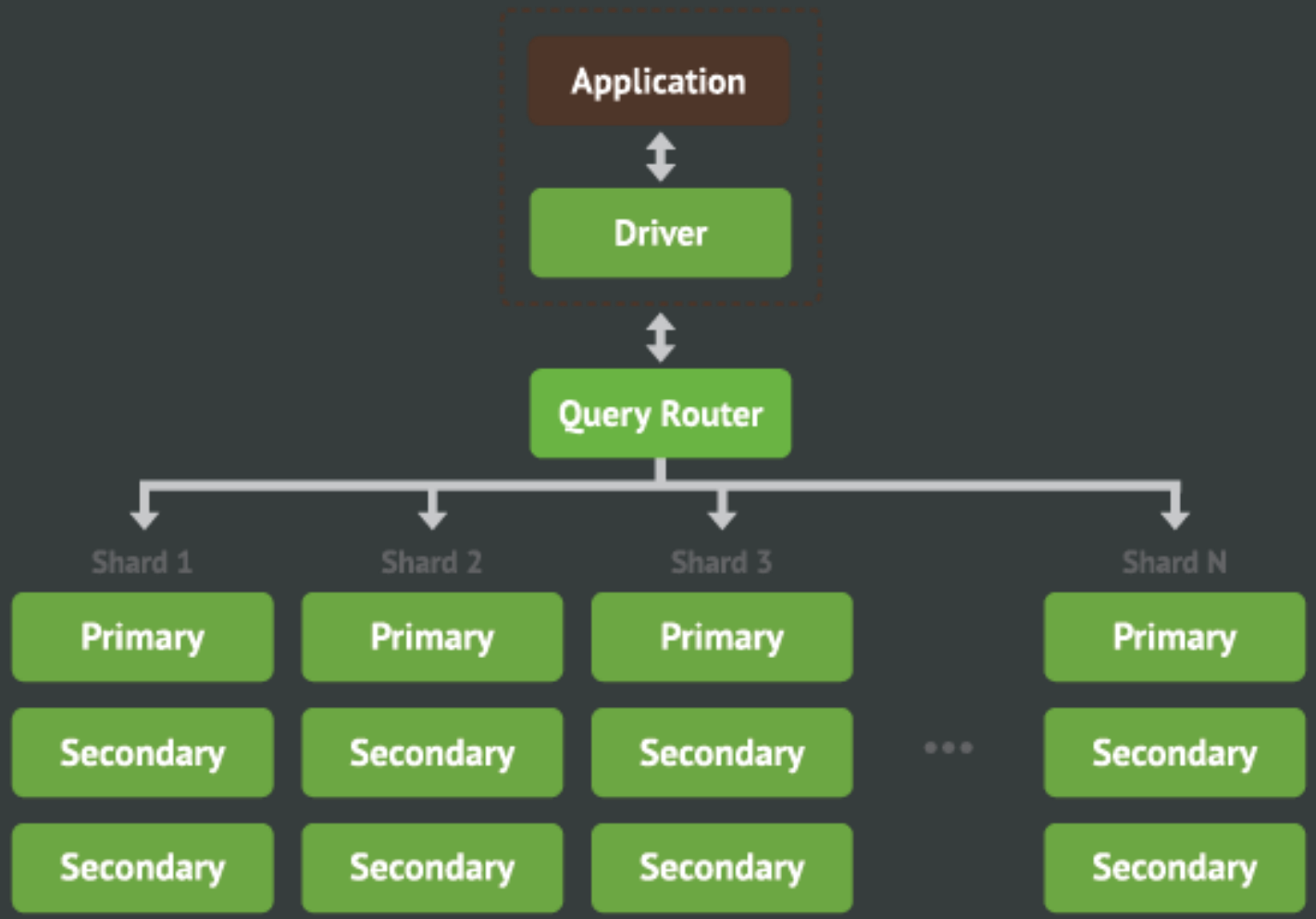


- Sharding
  - Particionado horizontal de los datos



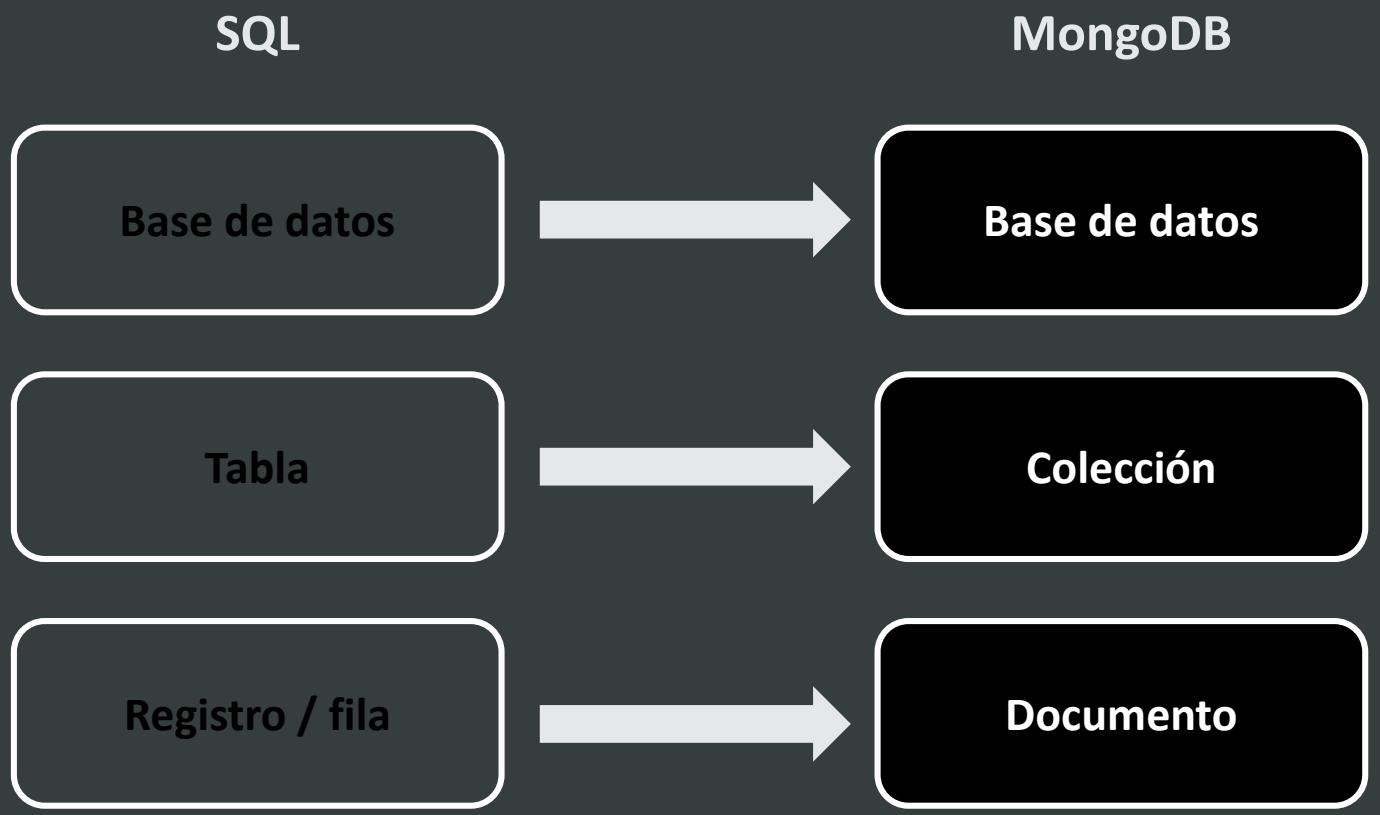
  - Tipos:
    - **Range Sharding**
    - **Hash Sharding**
    - **Zone Sharding**

Sharding

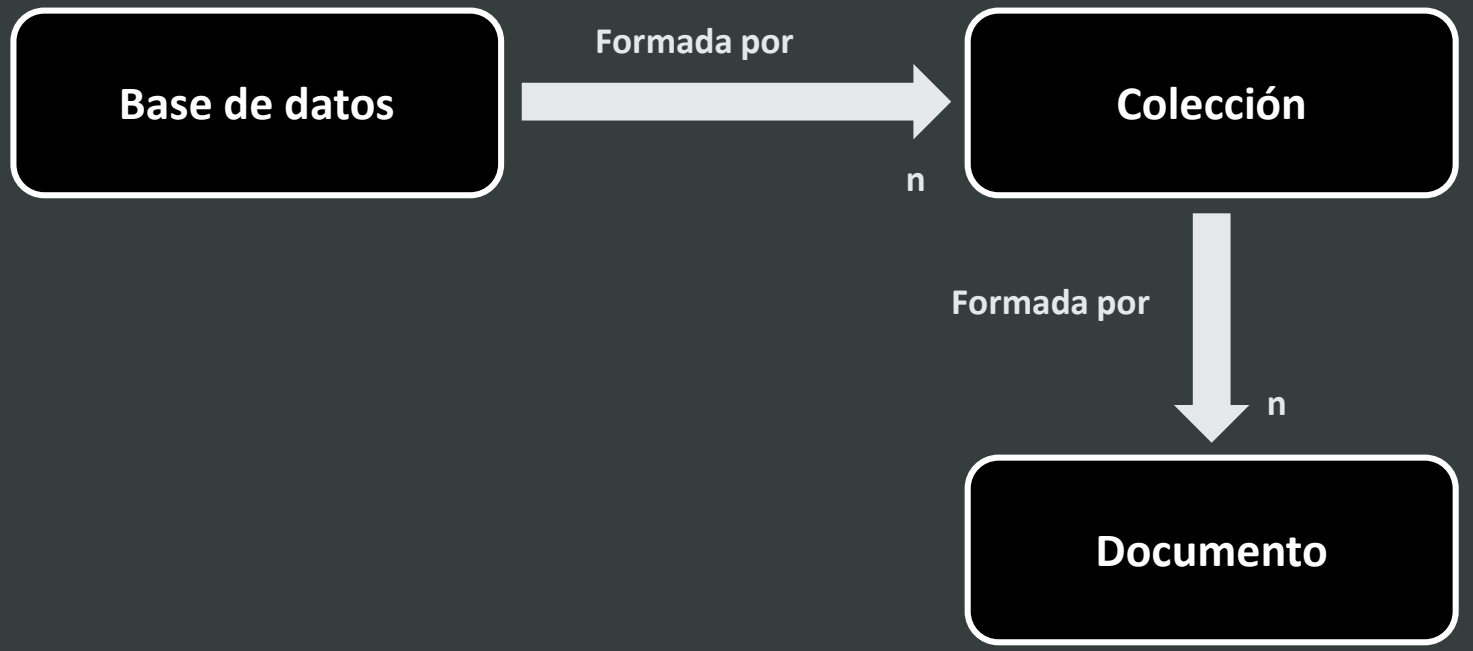

Application
Driver
Query Router
Shard 1
Shard 2
Shard 3
Shard N
Primary
Primary
Primary
Primary
Secondary
Secondary
Secondary
Secondary
Secondary
Secondary
Secondary
Secondary

Modelos de datos documentales

MongoDB


SQL
MongoDB
Base de datos
Base de datos
Tabla
Colección
Registro / fila
Documento

MongoDB


Base de datos
Formada por
n
Colección
Formada por
n
Documento

MongoDB: Estructura

Base de datos

Se crea al referenciarla por primera vez
Contenedor físico de colecciones
MongoDB == múltiples bbdd
Cada bd conjunto de ficheros

Colección

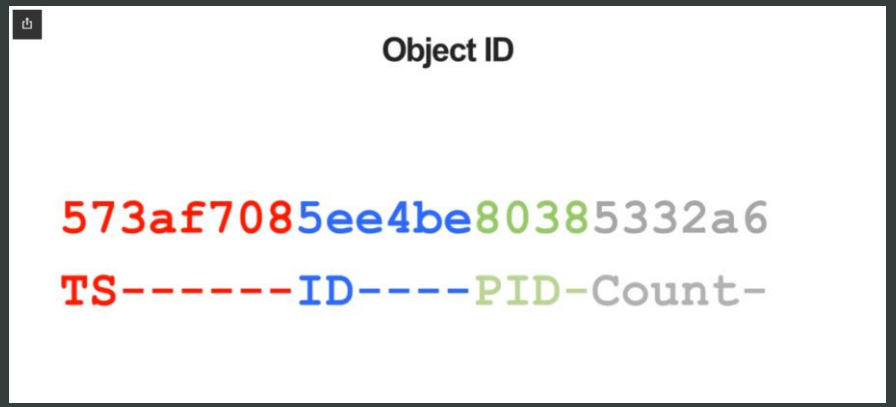
Sin esquema
Indexable por una o más claves
Se crea al referenciarla por primera vez
Limitadas ➔ registro antiguos se eliminan

Documento

Almacenados en colecciones
Pueden tener clave **_id** == claves primarias
Relaciones soportadas == **incrustadas** o **referencias**
Almacenamiento en formato **BSON (Binary JSON)**

MongoDB – Ejemplo de Documento

- _id: clave del documento de 12 bytes, sino se informa lo crea MongoDB automáticamente

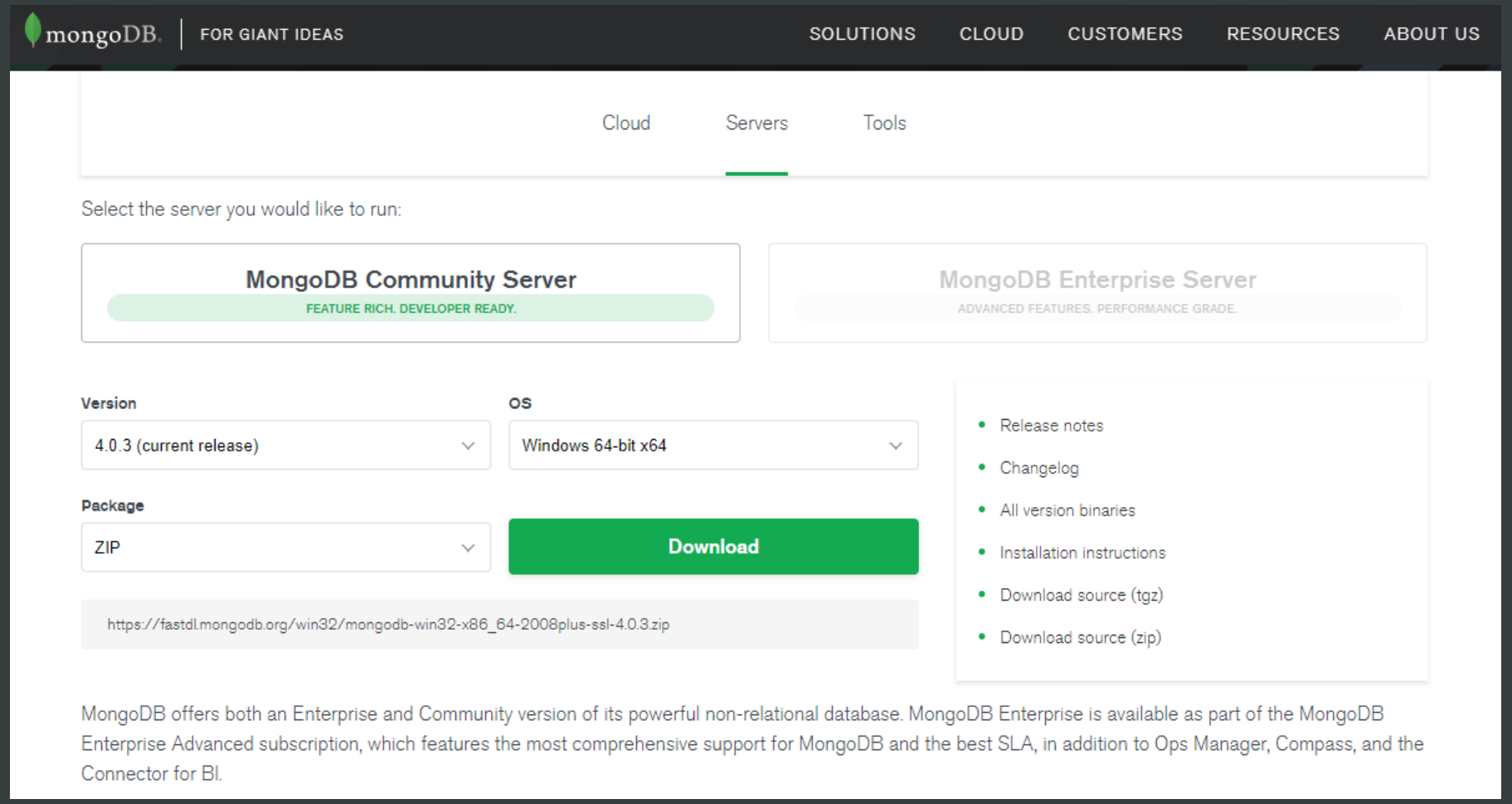



```
{
   _id: ObjectId(7df78ad8902c)
   title: ‘Mean Stack 3 ',
   description: 'MongoDB no es SQL',
   by: ‘Desarrollo Mean',
   url: 'http://www.DesarrolloMean.com',
   tags: ['mongodb', 'database', 'NoSQL'],
   likes: 100,
   comments: [
      {
         user:'user1',
         message: ‘Mi primer comentario',
         dateCreated: new Date(2018,1,20,2,15),
         like: 0
      },
      {
         user:'user2',
         message: 'My second comments',
         dateCreated: new Date(2018,1,25,7,45),
         like: 5
      }
   ]
}
```

Instalación MongoDB

MongoDB – Instalación

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



MongoDB – Instalación

- Extraer fichero y renombrarlo a mongodb
- Crear el fichero de datos: “c:\data\db”
- Arrancar el servidor desde el termial de comandos Windows con la instrucción:

  [ruta a mongodb]\mongod
- El terminal de usuario se arranca desde otro terminal con la instrucción:

  [ruta a mongodb]\mongo
- Crear un documento:

```
> db.test.save({a:1})
WriteResult({ "nInserted" : 1 })
> db.test.find({a:1})
{ "_id" : ObjectId("5bcc3314ea5446628d4b292e"), "a" : 1 }
```

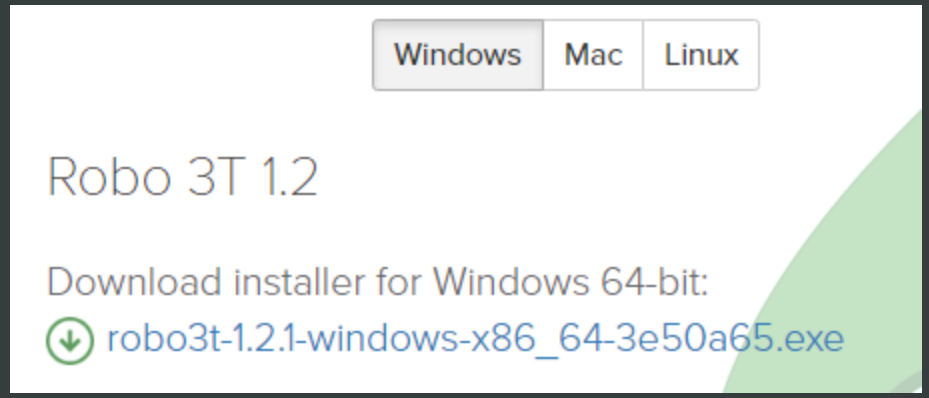
MongoSQL: Comandos generales

```
db.help()                    help on db methods
db.mycoll.help()             help on collection methods
sh.help()                    sharding helpers
rs.help()                    replica set helpers
help admin                   administrative help
help connect                 connecting to a db help
help keys                    key shortcuts
help misc                    misc things to know
help mr                      mapreduce

show dbs                     show database names
show collections             show collections in current database
show users                   show users in current database
show profile                 show most recent system.profile entries with time >= 1ms
show logs                    show the accessible logger names
show log [name]              prints out the last segment of log in memory, 'global' is default
use <db_name>                set current database
db.foo.find()                list objects in collection foo
db.foo.find( { a : 1 } )     list objects in foo where a == 1
it                           result of the last line evaluated; use to further iterate
DBQuery.shellBatchSize = x   set default number of items to display on shell
exit                         quit the mongo shell
```

Robo3T (RoboMongo) - Instalación

- GUI para trabajar con MongoDB
- https://robomongo.org/download



MongoDB – Instalación DBaaS

- Servicio DBaaS MongoDB Atlas https://www.mongodb.com/cloud/atlas
- Servicio DBaaS MLab https://mlab.com/signup/
  - Pulsar sobre Sign Up
  - Indicar y confirmar nuestros datos
  - Elegir clúster gratuito – 500 MB de almacenamiento

EJERCICIO 1

- Crea una colección de documentos (facturas) en formato JSON con al menos los siguientes campos:
  - Código_Factura
  - Cliente
  - Importe
  - Estado (Pagado, Pendiente, Vencido, …)
- STEPS:
  1. Abre Robo3T
  2. Crea una conexión (File->Connect)
  3. Crea una colección (Collection->Create Collection) con el nombre de Factura
  4. Inserta un documento (puedes usar el ejemplo de GitHub): https://github.com/rglepe/MeanStack/tree/master/Ejercicio1

Data Modelling

MongoDB – Modelado de datos

- Esquema flexible de documentos en la misma colección:
  - Pueden tener distinto grupo de campos
  - Un mismo campo puede tener distinto tipo de dato

- A la hora de diseñar:
  - Diseña según los requerimientos de usuario
  - Combina objetos en un mismo documento si se van a utilizar juntos. Si no, sepáralos pero asegúrate de que no necesitarás JOINS
  - Duplica datos. Disco + barato que Memoria
  - Realiza las JOINS al escribir y no al leer
  - Optimiza el esquema para los casos de uso más frecuentes
  - Realiza agregaciones complejas en el esquema

1ª Forma Normal (1FN)

id	name	phone_number	zip_code
1	Rick	555-111-1234	30062
2	Mike	555-222-2345	30062
3	Jenny	555-333-3456	01209

id	name	phone_number	zip_code
1	Rick	555-111-1234	30062
2	Mike	555-222-2345;555-212-2322	30062
3	Jenny	555-333-3456;555-334-3411	01209

- Documento embebido (embedded Document)

```
// Contact document:
{
"_id": 3,
"name": "Jenny",
"zip_code": "01209"
}
// Number documents:
{ "contact_id": 3, "number": "555-333-3456" }
{ "contact_id": 3, "number": "555-334-3411" }
```

MongoDB – Documentos embebidos

- Localización => para búsquedas en discos duros rotatorios (spinning disks)
  - MongoDB almacena todos los documentos de forma contigua en el disco
  - MongoDB funcionaría aún peor en búsquedas por referencia:

```
contact_info = db.contacts.find_one({'_id': 3})
number_info = list(db.numbers.find({'contact_id': 3})
```

- Atomicidad y Aislamiento (A+I de ACID)
  - Los documentos embebidos no necesitan control de las transacciones:

```
BEGIN TRANSACTION;
DELETE FROM contacts WHERE contact_id=3;
DELETE FROM numbers WHERE contact_id=3;
COMMIT;
```

```
db.contacts.remove({'_id': 3})
db.numbers.remove({'contact_id': 3})
```

MongoDB – Referencias

- Flexibilidad => Normalizar

```
{ "_id": "First Post",
"comments": [
{ "author": "Stuart", "text": "Nice post!"
},
{ "author": "Mark", "text": "Dislike!" } ] },
{ "_id": "Second Post",
"comments": [
{ "author": "Danielle", "text": "I am
intrigued" },
{ "author": "Stuart", "text": "I would like
to subscribe" } ] }
```

A una búsqueda por autor de comentarios se ajusta mejor un esquema parcialmente normalizado

```
// db.posts schema
{
"_id": "First Post",
"author": "Rick",
"text": "This is my first post"
}
// db.comments schema
{
"_id": ObjectId(...),
"post_id": "First Post",
"author": "Stuart",
"text": "Nice post!"
}
```

MongoDB – Referencias

- Flexibilidad => Normalizar

```
{ "_id": "First Post",
"comments": [
{ "author": "Stuart", "text": "Nice post!"
},
{ "author": "Mark", "text": "Dislike!" } ] },
{ "_id": "Second Post",
"comments": [
{ "author": "Danielle", "text": "I am
intrigued" },
{ "author": "Stuart", "text": "I would like
to subscribe" } ] }
```

A una búsqueda por autor de comentarios se ajusta mejor un esquema parcialmente normalizado

```
// db.posts schema
{
"_id": "First Post",
"author": "Rick",
"text": "This is my first post"
}
// db.comments schema
{
"_id": ObjectId(...),
"post_id": "First Post",
"author": "Stuart",
"text": "Nice post!"
}
```

MongoDB – Referencias

- Relaciones de alto grado (aridad) => Normalizar

  Ej.: blog con muchos comentarios (~100-1000)

  - Cuanto más grande un documento más RAM
  - Documentos muy grandes se desplazan a otros espacios de disco
  - Documentos límite de 16 MB
- Relaciones M:N => Compromiso

```
// db.product schema
{ "_id": "My Product", ... }
// db.category schema
{ "_id": "My Category", ... }
// db.product_category schema
{ "_id": ObjectId(...),
"product_id": "My Product",
"category_id": "My Category" }
```

```
// db.product schema
{ "_id": "My Product",
"category_ids": [ "My Category", ... ] }
// db.category schema
{ "_id": "My Category" }
```

MongoDB – Documentos Embebidos

- Esquemas polimórficos (POO) Ej.: Wiki comentarios y fotos
- Almacenamiento ineficiente de BSON => Cabeceras repetidas por documento
- Soporte a Esquemas semi-estructurados Productos con atributos pero no todos los productos tienen los mismos atributos Ej. Discos duros con distintas características => Índices sobre propiedades embebidas

Casos de Uso

- Ecommerce:
  - CATÁLOGO DE PRODUCTOS: Una única colección para todos los productos
  - CATEGORÍAS DE PRODUCTOS
  - MANTENIMIENTO DE INVENTARIO (CARRITO DE LA COMPRA)

- Red Social:
  - Publicaciones
  - Seguidores
  - Mensajes
  - Notificaciones
  - Likes