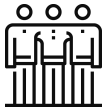
MongoDB

Marco Robles Pulido

Academy Code of Conduct

Seamos respetuosos, no existen preguntas malas



Seamos pacientes

Cuidemos nuestro lenguaje

Objetivos de la sesión

Al final de la sesión seremos capaces de:

- Identificar posibles índices y aplicarlos
- Utilizar Hashes
- Realizar replicas

Tabla de contenido

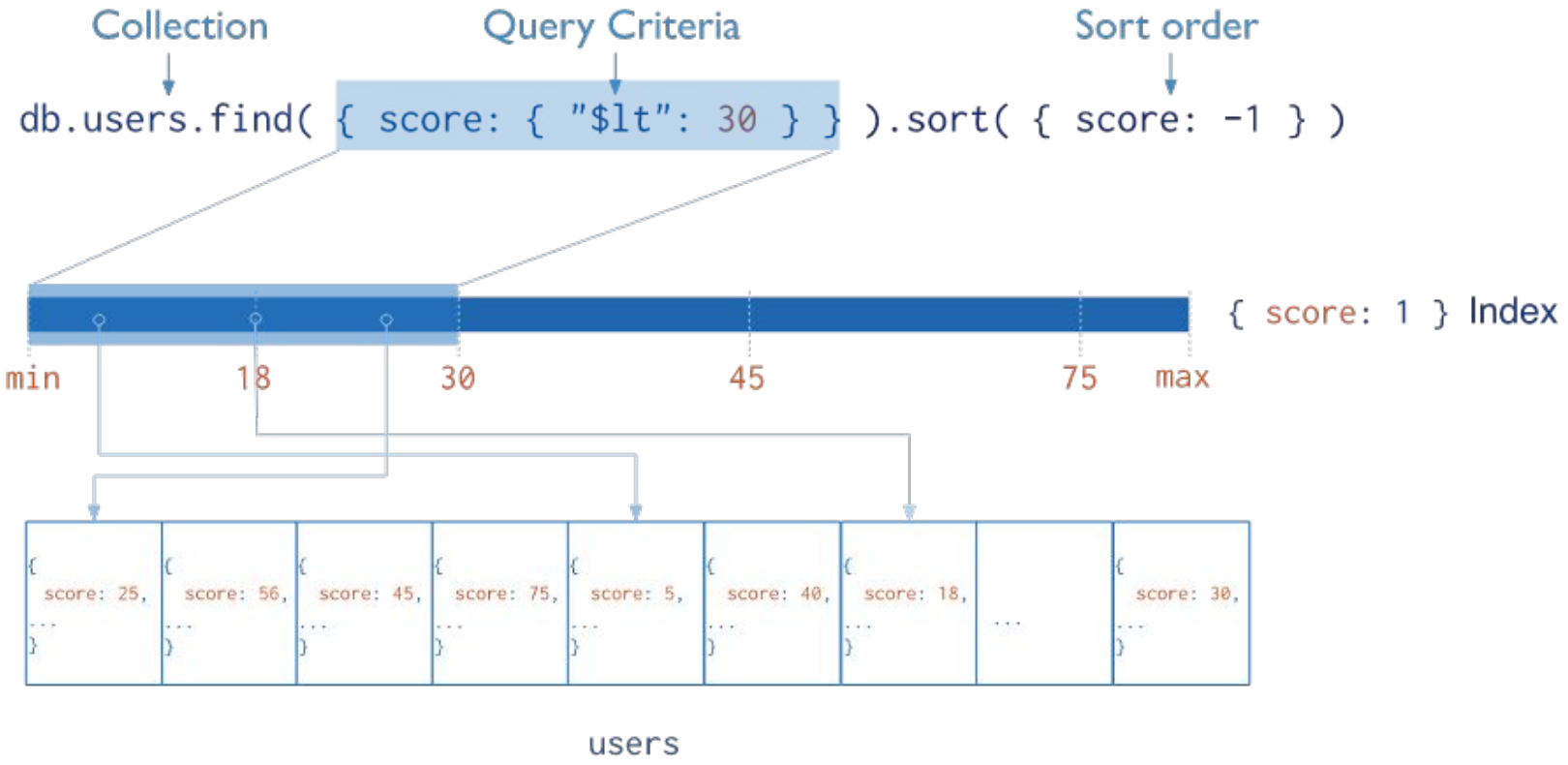
Indices

Los índices apoyan la **ejecución eficiente de las consultas**. Sin índices, MongoDB debe **escanear cada documento de una colección** para seleccionar aquellos que coinciden con la consulta.

Si existe un índice apropiado para una consulta, MongoDB puede utilizar el índice para **limitar el número de documentos** que debe inspeccionar.

El índice almacena el valor de un campo específico o un conjunto de campos, ordenado por el valor del campo.

Indices



Crear indices

MongoDB crea un índice único en el campo *_id* durante la creación de una colección. Esto evita que se inserten dos documentos con el mismo valor para el campo *_id*. No se puede eliminar este índice en el campo *_id*.

Para crear un índice se puede utilizar el siguiente comando:

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

Ejemplo de índice

Consideremos una aplicación que consulta con frecuencia la colección de productos para rellenar los datos del inventario existente. El siguiente método *createIndex()* crea un índice sobre el artículo (ascendente) y la cantidad (descendente) con nombre de consulta para el inventario:

```
db.products.createIndex(
  { item: 1, quantity: -1 } ,
  { name: "query for inventory" }
)
```

Tipos de índices

- Índice de campo único
- Índice compuesto
- Índice multi-llave
- Índice geoespacial
- Índice de texto
- Índice Hash
- Índice agrupado

Tabla de contenido

dbHash

Devuelve los valores hash de las colecciones de una base de datos y un valor MD5 para estas colecciones. *dbHash* es útil para comparar bases de datos entre mongod como, por ejemplo, entre miembros de conjuntos de réplicas.

El comando tiene la siguiente sintaxis:

```
db.runCommand(
   {
     dbHash: 1,
     collections: [ <collection1>, ... ]
   }
)
```

dbHash

El comando devuelve un documento con los siguientes campos:

- *collections*
- *capped*
- *uuids*
- *md5*
- *timeMillis*
- *$clusterTime*

dbHash

Ejemplo:


```
{
   "host" : "myHostName.local:27017",
   "collections" : {
      "foo" : "d27b769230edc551d869060ec3fb68bd",
      "inventory" : "ec3d821581ea1bd3aa8196c94b946874",
      "log" : "d41d8cd98f00b204e9800998ecf8427e",
      "orders" : "0242c0a128c284ea9576a34db2306c12",
      "restaurants" : "5dc9b88091c36f0d529567b5b6e3fc92",
      "zipcodes" : "31ede812bf397509a87359c65bf2a08c"
   },
   "capped" : [
      "log"
   ],
   "uuids" : {
      "foo" : UUID("469592fe-3bfe-425e-975f-cedbe0c4741d"),
      "inventory" : UUID("0830e0ad-cc24-4fc7-80d0-8e22fe45e382"),
      "log" : UUID("4be024ff-711b-4ab8-836b-dee662e090f1"),
      "orders" : UUID("755be489-745f-400c-ac3b-f27525ad0108"),
      "restaurants" : UUID("520b56ec-3276-4904-b6e5-286bc9bfa648"),
      "zipcodes" : UUID("12e97b70-c174-40af-a178-5d83a241fe20")
   },
   "md5" : "0cb7417ae9d9eb865000b4debdc671da",
   "timeMillis" : 53,
   "ok" : 1,
   "operationTime" : Timestamp(1529208582, 4),
   "$clusterTime" : {
      "clusterTime" : Timestamp(1529208582, 4),
      "signature" : {
         "hash" : BinData(0,"X3MmevDqUgCVvN1AhnT+fiOL/Lc="),
         "keyId" : NumberLong("6567898567824900097")
      }
   }
}
```

Tabla de contenido

Mongod

El núcleo de MongoDB encargado de:

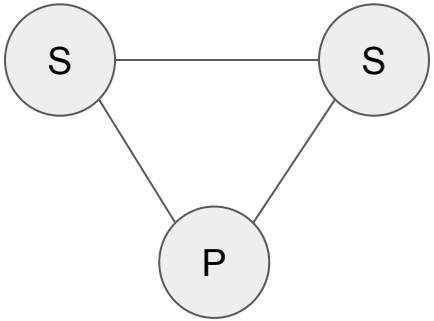
- Gestionar las conexiones
- Solicitudes y persistencia de datos

Es el proceso demonio (daemon), indicando que no interactuamos directamente con este, pero sí a través de los drives encargados de gestionar las comunicaciones de dicho proceso.

Replica set

Un *replica set* en MongoDB es un grupo de *mongod* que mantienen el mismo conjunto de datos.

Los conjuntos de réplica proporcionan redundancia y alta disponibilidad, y son la base de todos los despliegues de producción.



Redundancia y disponibilidad de datos

La replicación proporciona un **nivel de tolerancia a los fallos** frente a la pérdida de un único servidor de bases de datos.

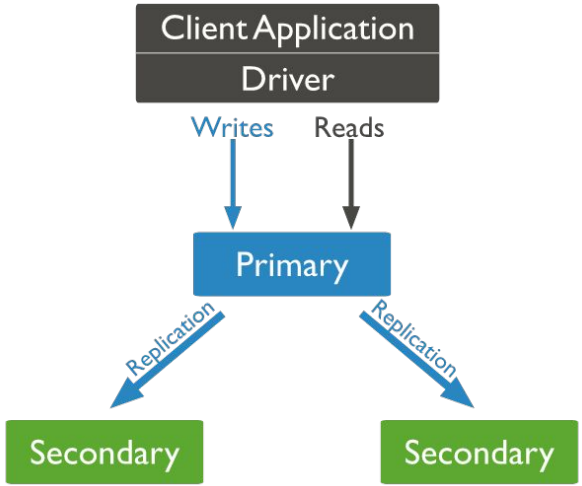
En algunos casos, la replicación puede proporcionar una **mayor capacidad de lectura**, ya que los clientes pueden enviar operaciones de lectura a diferentes servidores.

Mantener copias de los datos en diferentes centros de datos puede aumentar la localidad de los datos y la disponibilidad para las aplicaciones distribuidas.

Replicación

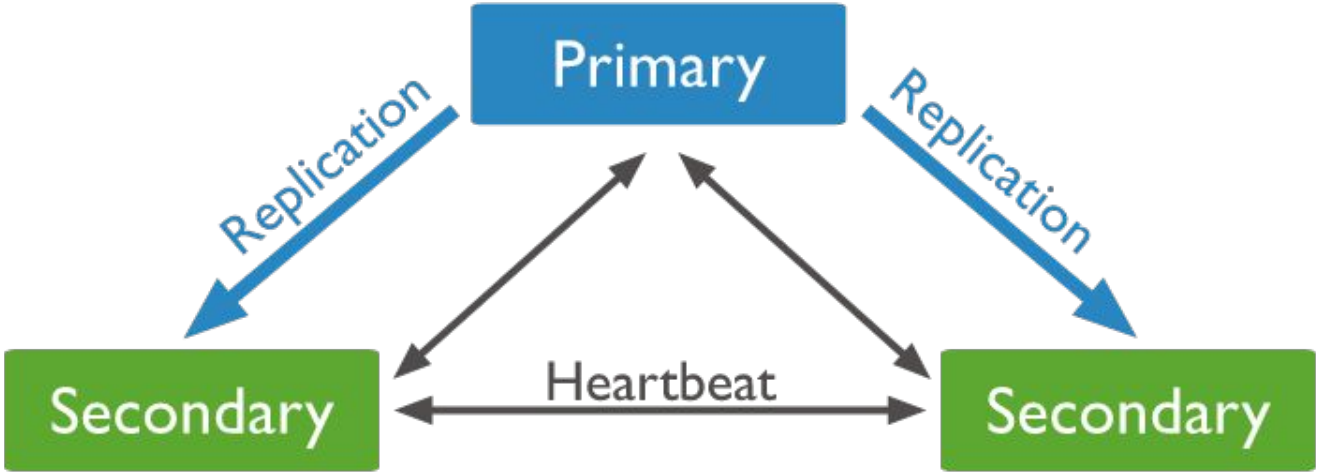
De los nodos portadores de datos, uno y sólo uno se considera el nodo primario, los otros nodos se consideran nodos secundarios.

El nodo primario recibe todas las operaciones de escritura.



Replicación

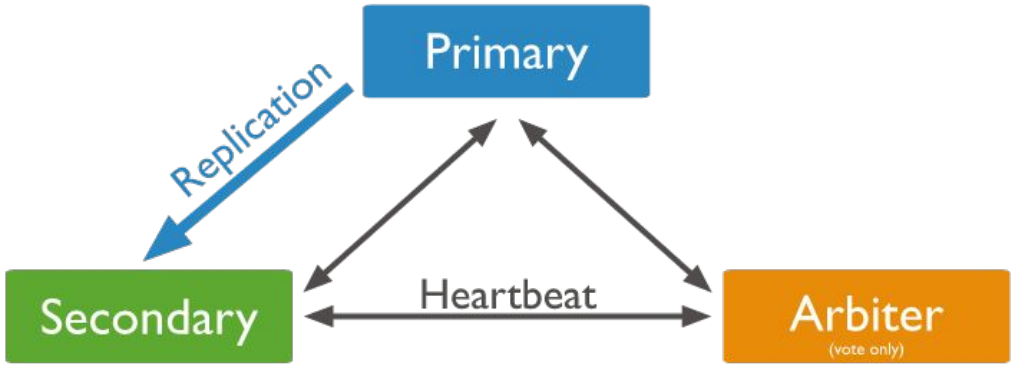
Los secundarios replican el *oplog* del primario y aplican las operaciones a sus conjuntos de datos. Si el primario no está disponible, un secundario elegible celebrará una elección para elegir él mismo el nuevo primario.



Replicación

En algunas circunstancias (como cuando se tiene un primario y un secundario pero las restricciones de costes prohíben añadir otro secundario), se puede optar por añadir una instancia de mongod a un conjunto de réplicas como árbitro.

Un árbitro participa en las elecciones, pero no tiene datos, y siempre será un árbitro.

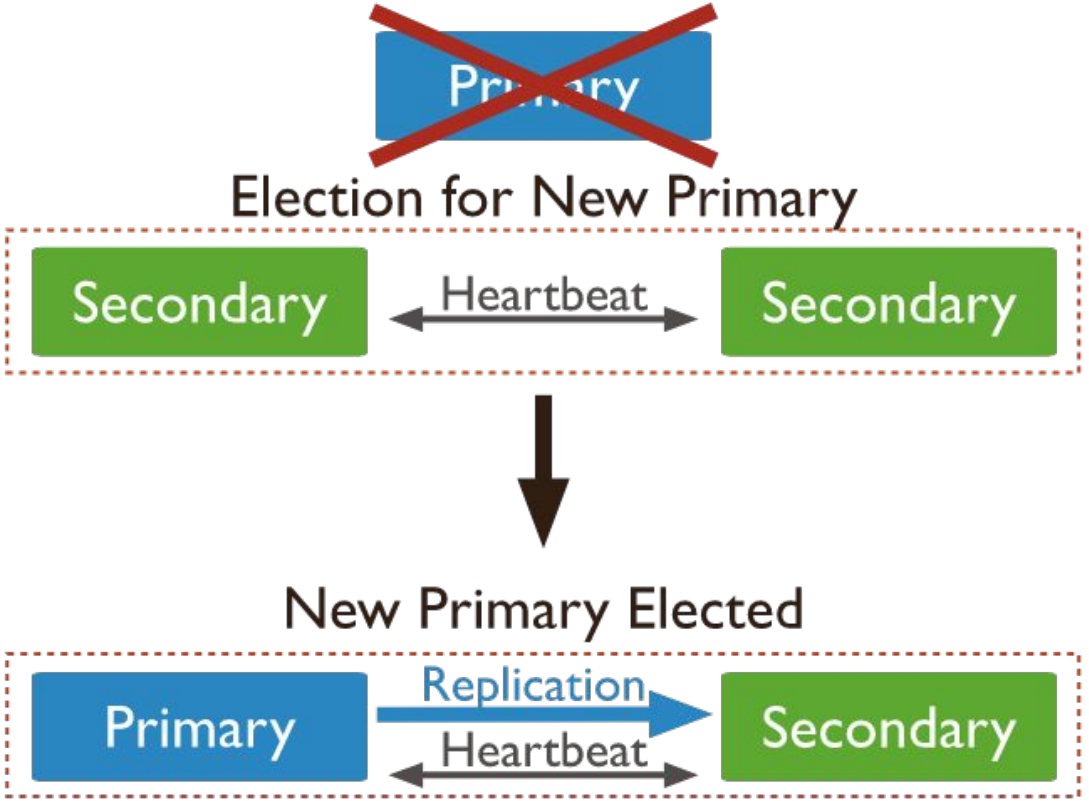


Proceso de replicación

El cluster intenta completar la elección de un nuevo primario y reanudar las operaciones normales.

El *replica set* no puede procesar operaciones de escritura hasta que la elección se complete con éxito.

Si puede seguir sirviendo consultas de lectura si están configuradas para ejecutarse en los secundarios.



Preferencia de lectura

Los clientes pueden especificar una preferencia de lectura para enviar las operaciones de lectura a los secundarios.

- La *replicación asíncrona* a los secundarios significa que las lecturas de los secundarios pueden devolver datos que no reflejan el estado de los datos en el primario.
- Las *transacciones de varios documentos* que contienen operaciones de lectura deben utilizar la preferencia de lectura primario.

Recapitulando

- ¿SQL vs NOSQL?
  - Escalabilidad
  - Estructura
  - Propiedades
- ¿Documento?
- ¿Colección?
- ¿Modelos?
- ¿Esquema?
- ¿Indexes?
- ¿Hashing?
- ¿Replicación?

Feedback Form

Let us know your feedback!

https://forms.gle/WKtc8wZeSxWnjGo8A