MongoDB

BASE DE DATOS NOSQL

Contenidos

- Replicación
- Sharding
- Administración y Optimización

REPLICACIÓN

- Proceso por el que se sincronizan los datos almacenados en la base de datos con las copias de la misma localizadas en distintos servidores

- Ventajas:

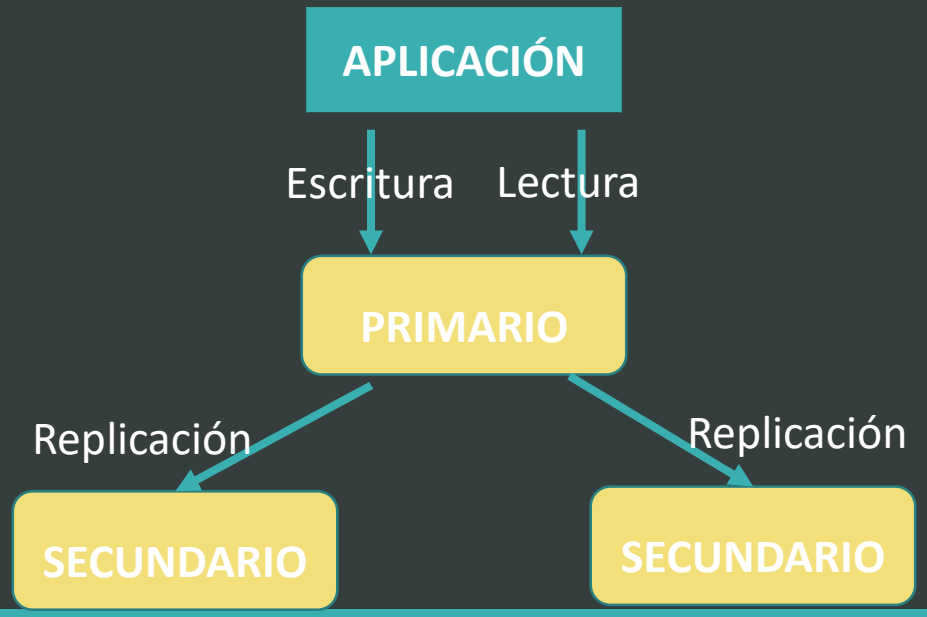
- Incrementa la disponibilidad de los datos al haber redundancia de datos

- Protección frente a pérdidas o fallos del servidor al haber copias en diferentes nodos de red

- Incrementa el rendimiento en los procesos de lectura al reducir tiempos de latencia; los clientes realizan peticiones a diferentes servidores en paralelo y se selecciona el más cercano

GRUPO DE REPLICACIÓN (REPLICATION SET)

- Grupo de instancias de mongod que almacenan el mismo conjunto de datos
- Existe una instancia PRIMARIA que recibe las operaciones de escritura y n instancias SECUNDARIAS que realizan las operaciones indicadas por la primaria para tener el mismo conjunto de datos
- Cada REPLICA SET SÓLO un nodo primario y varios secundarios (normalmente impares >3) con consistencia estricta de lectura al nodo primario.
- Existen nodos secundarios sin datos que actúan como árbitros y sólo participan en procesos de fallo de servidor

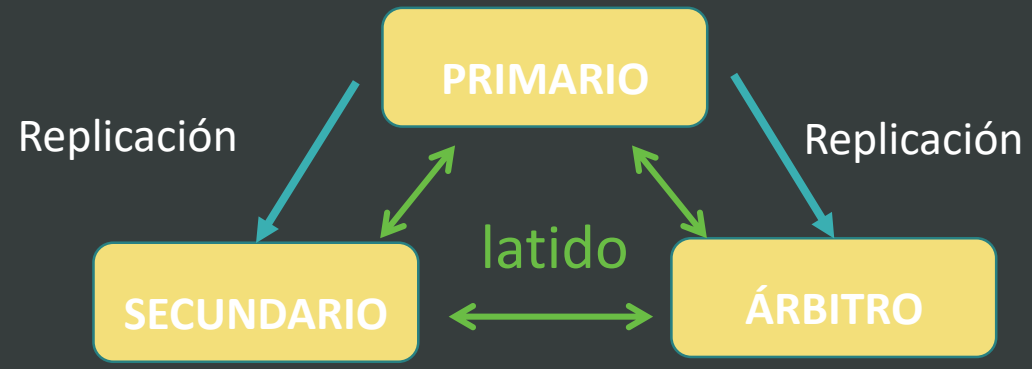


GRUPO DE REPLICACIÓN (REPLICATION SET)

- Bajar de gitHub el fichero con los pasos para montar un replication Set: https://github.com/rglepe/MeanStack/blob/master/Replicacion/Ejemplo_replicacion.txt

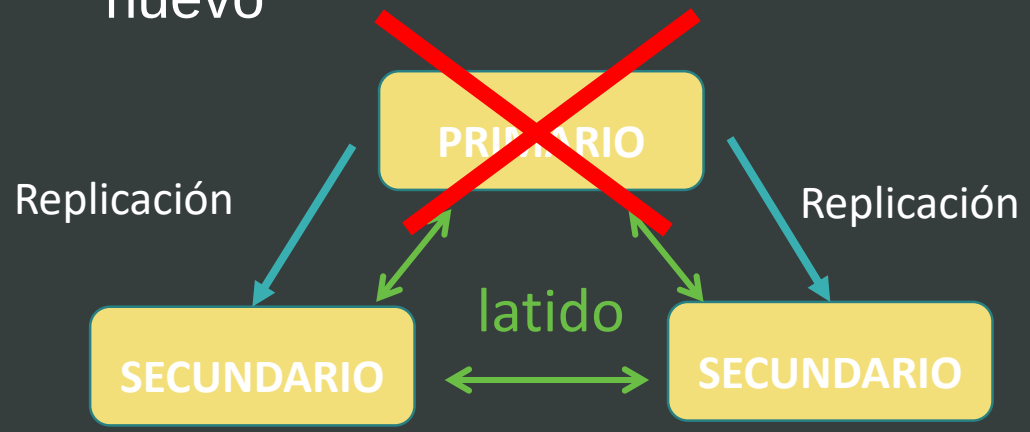
GRUPO DE REPLICACIÓN: Proceso de replicación

- El nodo primario mantiene un log: **oplogs** con los cambios
- Cada nodo secundario replica el oplog del nodo primario y aplica las operaciones pendientes contenidas en el mismo a sus propios datos de manera asíncrona
- Los nodos secundarios realizan un ping (heartbeat) al resto de los nodos cada 2 segundos.



GRUPO DE REPLICACIÓN: Recuperación tras fallo

- Cuando el nodo primario no se comunica con el resto de los nodos secundarios durante más de 10 segundos, se inicia el proceso para elegir un nuevo primario. No se permiten escrituras hasta que no se elija uno nuevo



- Para elegir un servidor secundario como nuevo primario debe cumplir:
  - Tiene la mayor prioridad. Los nodos con prioridad 0 no son elegibles.
  - Puede ver a una mayoría de servidores que tengan voto.
  - Sus datos están sincronizados en el momento del fallo del primario.

GRUPO DE REPLICACIÓN: Recuperación tras fallo

- Un grupo de réplica puede tener hasta 50 nodos pero sólo 7 tienen derecho a voto

- Los cambios producidos durante la caída del servidor no son conocidos por el nuevo nodo primario

- El servidor caído al reconectar, debe descartar los cambios no propagados y ponerse al día con el primario. Los cambios descartados se pueden ver con bsondump y restaurarse manualmente con mongorestore.

GRUPO DE REPLICACIÓN: Configuración de replicación

- Configuración de nodos secundarios (rs.config()):

```
{"_id" : 0,
"host" : "localhost:27017",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 1,
"tags" : {},
    "slaveDelay" : NumberLong(0),
"votes" : 1          }
```

GRUPO DE REPLICACIÓN: Proceso de replicación

- *priority*: parámetro para poder ser elegido primario en caso de fallo. “0” no puede ser elegido.
- *hidden*: parámetro de visibilidad. Si se fija a **true**, no será visible desde las aplicaciones aunque mantiene la información replicada.
- *slaveDelay*: parámetro para configurar un “nodo retardado” - nodos que se replican con un retardo de tiempo - Requieren también hidden true y priority 0.
- *votes*: permiten añadir nuevos miembros al grupo con posibilidad de voto (+ de 7). Para configurar como votes:0 requiere priority 0
- *arbiterOnly*: convierte un nodo secundario en árbitro. Requiere eliminar los datos del nodo.

GRUPO DE REPLICACIÓN: Otros comandos

instrucción	descripción
rs.conf ()	Muestra la configuración del replica set
rs.status ()	Comprueba el estado del replica set
1. **mongod --port 27020 --dbpath /data/n4 --replSet rs0** 2. **rs.add ("hostname: puerto")**	Añadir una réplica al replica Set: 1. Se Crea una instancia nueva en el path de datos indicado 2. Se añade al set desde la consola conectada al primario
rs.add ("hostname: puerto", true)	Añade un árbitro al Set
ctrl+c del servidor **rs.remove("hostname: puerto")**	Quitar un servidor de la réplica.
cfg = rs.conf(); **cfg.members[2].priority = 0;** **cfg.members[2].hidden =** **true;** **cfg.members[2].slaveDelay =** **3600 ;** **rs.reconfig(cfg);**	Configurar un nodo retardado que se replica con una hora de retardo.
ctrl+c del servidor **rs.remove("hostname: puerto")** **eliminación fichero bd (archivos del s.o.)** **Aplicar los pasos de añadir un nuevo árbitro**	Configurar un nodo secundario como árbitro sobre el mismo puerto: Desconectar el nodo del Set Eliminar el secundario

GRUPO DE REPLICACIÓN: Conclusiones

- ✓ La replicación Mongo da soporte a la alta disponibilidad; viene de fábrica y; es sencilla de configurar e implementar
- Problema de disponibilidad en caso de fallo del primario,
  - El **failover** (conmutación debido a error) está automatizado en MongoDB, pero tarda un tiempo para que ocurra.
- La replicación no ayuda mucho con la escalabilidad.
  - Las escrituras siempre se hacen en el primario, los secundarios no pueden compartir esta carga
  - La carga de lectura se puede compartir con los secundarios, pero con pérdida de consistencia
- No hay mucha configuración para la replicación
  - Se hace en el nivel de la instancia – no se puede sintonizar por colección
  - No se puede ajustar la consistencia si se comparten las lecturas.

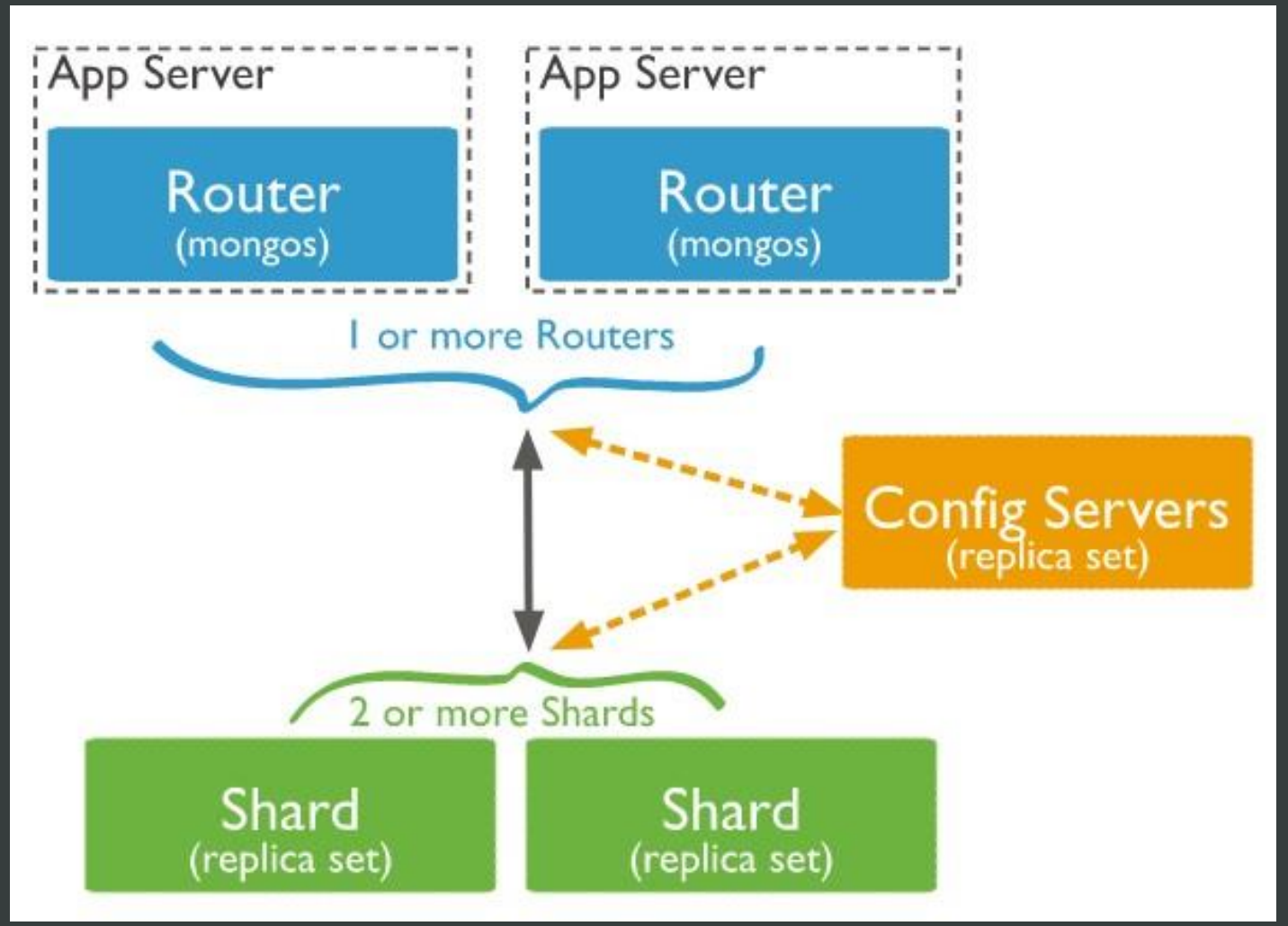
SHARDING

- Mecanismo que implementa MongoDB para realizar un procesamiento distribuido de los datos.
- A medida que se manejan volúmenes ingentes de datos aparecen problemas:
- Limitaciones de la CPU en cuanto a las consultas que se pueden procesar
- Falta de espacio en memoria para el almacenamiento de índices
- Soluciones:
  - Escalado vertical: mejores máquinas: caro y limitado
  - Escalado horizontal: más máquinas: baratas e “ilimitadas”

SHARDING

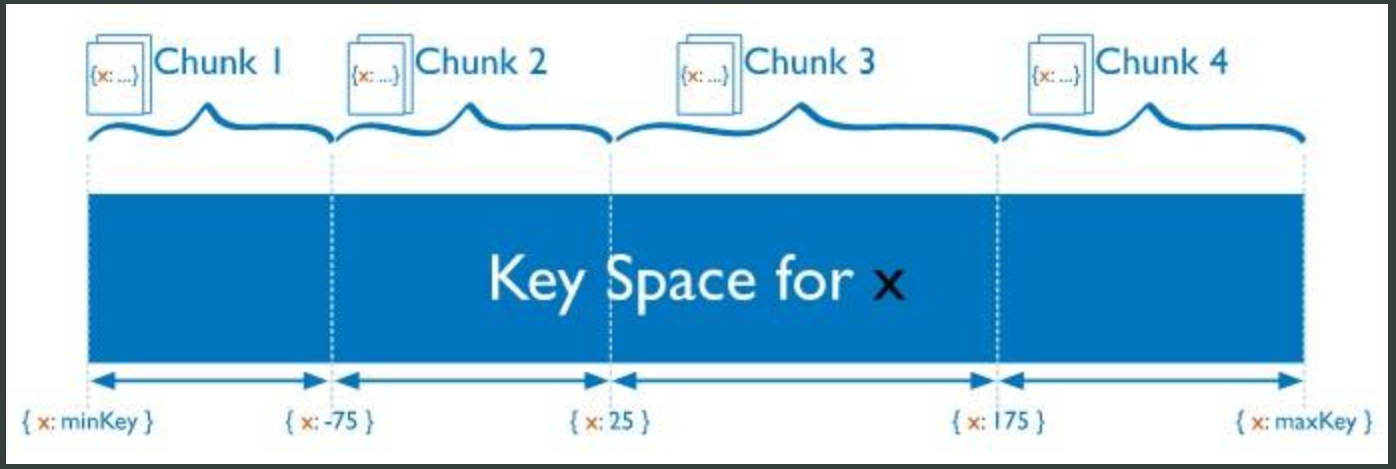
- Consiste en particionar los datos de una base de datos horizontalmente agrupándolos de algún modo que tenga sentido y que permita un direccionamiento (enrutado) más rápido.
- Ej: “Los usuarios del 1 al millón están en bd1”; o por nombre “los usuarios cuyo nombre va de la A a la E” están en otra base de datos”.
- Componentes:
  - **Servidores de configuración**: los servidores de configuración almacenan metadatos y configuraciones para el clúster. A partir de MongoDB 3.4, los servidores de configuración deben implementarse como un conjunto de réplicas.
  - **Shard** o partición: contiene un subconjunto de los datos fragmentados. Cada fragmento se puede implementar como un conjunto de replicación.
  - **mongos**: enrutador de queries o consultas, proporcionando una interfaz entre las aplicaciones cliente y el clúster de shards.

SHARDING


App Server
Router
(mongos)
App Server
Router
(mongos)
1 or more Routers
Config Servers
(replica set)
2 or more Shards
Shard
(replica set)
Shard
(replica set)

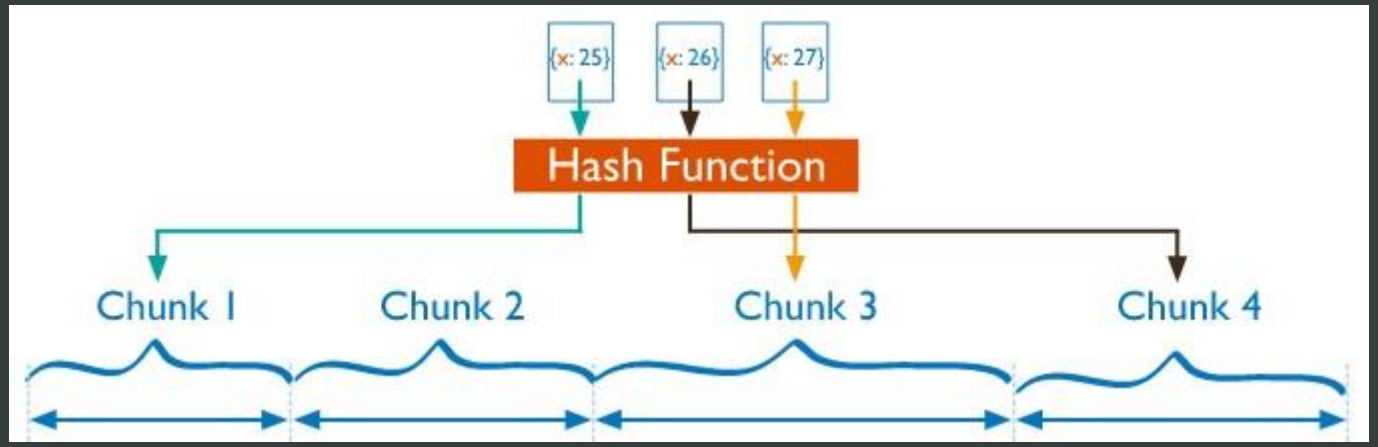
SHARDING: Estrategias de sharding

- **Ranged Sharding**
  - Se divide los datos en rangos basados en los valores clave del shard.
  - A cada fragmento se le asigna un rango basado en los valores clave del fragmento.
  - Existe más probabilidad de que una gama de claves de shard cuyos valores sean “cercanos” residan en el mismo shard
  - La eficacia del particionado depende de la clave de shard elegida. Claves de fragmentos mal escogidas pueden dar lugar a una distribución desigual de los datos



SHARDING: Estrategias de sharding

- **Hashed Sharding**
  - Se calcula un hash del valor del campo clave del shard.
  - A cada shard se le asigna un rango basado en los valores de la clave hasheada del shard.
  - Facilita una distribución de datos más homogénea entre elementos del cluster
  - Dificulta las consultas basadas en rangos



EJERCICIO 1

1. Monta un cluster de sharding con un router, un servidor de configuración y dos shards. Servidores de configuración y shards formarán parte de grupos de réplica.

   Seguir el fichero: github

OPTIMIZACIÓN

- Cualquier db se enfrenta a problemas de rendimiento cuando crece el tamaño de los datos. Reescribir queries problemáticas u optimizer el esquema de la db puede llevar a una mejora sustancial del rendimiento
- De forma similar a las bases de datos relacionales, en MongoDB dispone de diversas herramientas para optimizar el rendimiento de una consulta: Explain plans, Query profiler, Mongostat, Mongotop, logs,...
- Veremos:
  - **Query Profiler**
  - Almacena información estadística y detalles de la ejecución de consultas que se ejecutan con “lentitud”
  - **Explain Plans**
  - Retorna información sobre el plan de ejecución de las operaciones: aggregate (), count (), find (), group (), remove () y update ()

OPTIMIZACIÓN

- El profiler almacena todos los datos en la colección **system.profile**
- Existen 3 niveles de perfil. El nivel 0 es el nivel por defecto:

Level 0	No almacena ningún dato
Level 1	Sólo información de operaciones lentas por encima de un umbral
Level 2	Todas las operaciones

- Para activarlo: db.setProfilingLevel(level, slowms)
  - Ej: db.setProfilingLevel(1,40)

OPTIMIZACIÓN

Información	Significado
system.profile.ts	Cuándo se ejecutó la operación.
system.profile.op	El tipo de operación: insert, query, update,remove, getmore, command.
system.profile.ns	Nombre de la operación en el formato: base_datos.colección.
system.profile.query	Documento de consulta usado.
system.profile.command	Operación de comando.
system.profile.updateobj	El documento de actualización usado durante la operación.
system.profile.cursorid	El id del cursor accedido durante una operación de tipo getmore.
system.profile.ntoreturn	Número de documentos que se retornan como resultado de la operación.
system.profile.ntoskip	Número de documentos especificados en el método skip().
system.profile.nscanned	Número de documentos escaneados en el índice para llevar a cabo la operación.
system.profile.scanAndOrder	Si toma el valor de «true», indica si el sistema no puede usar el orden de los documentos en el índice para retornar los resultados ordenados.
system.profile.client	IP de la conexión cliente que origina la operación.

OPTIMIZACIÓN

Información	Significado
system.profile.moved	Aparece con el valor de true en caso de que la operación de actualización haya movido uno o más documentos a una nueva localización en disco.
system.profile.nmoved	Número de documentos que la operación movió en el disco
system.profile.nupdated	Número de documentos actualizados en la operación.
system.profile.keyUpdates	Número de claves del índice que la actualización cambió en la operación.
system.profile.numYield	El número de veces que la operación cedió su turno a otras operaciones para que se completaran antes que ella.
system.profile.lockStats	Tiempo en microsegundos que la operación espera a adquirir y mantener un bloqueo. Puede retornar: R(bloqueo global de lectura), W(Bloqueo global de escritura), r(bloqueo de lectura de una base de datos específica) y w(bloqueo de escritura de una base de datos específica)
system.profile.lockStats.timeLockedMicros	Tiempo en microsegundos que la operación mantiene un bloqueo específico.
system.profile.lockStats.timeAcquiringMicros	Tiempo en microsegundos que la operación espera para adquirir un bloqueo específico.
system.profile.nreturned	Número de documentos retornados por la operación.
system.profile.responseLength	Longitud en bytes del documento resultante de la operación.
system.profile.millis	Tiempo en milisegundos desde el comienzo de la operación hasta la finalización.
system.profile.user	Usuario autenticado que ejecuta la operación.

OPTIMIZACIÓN: Ejemplos

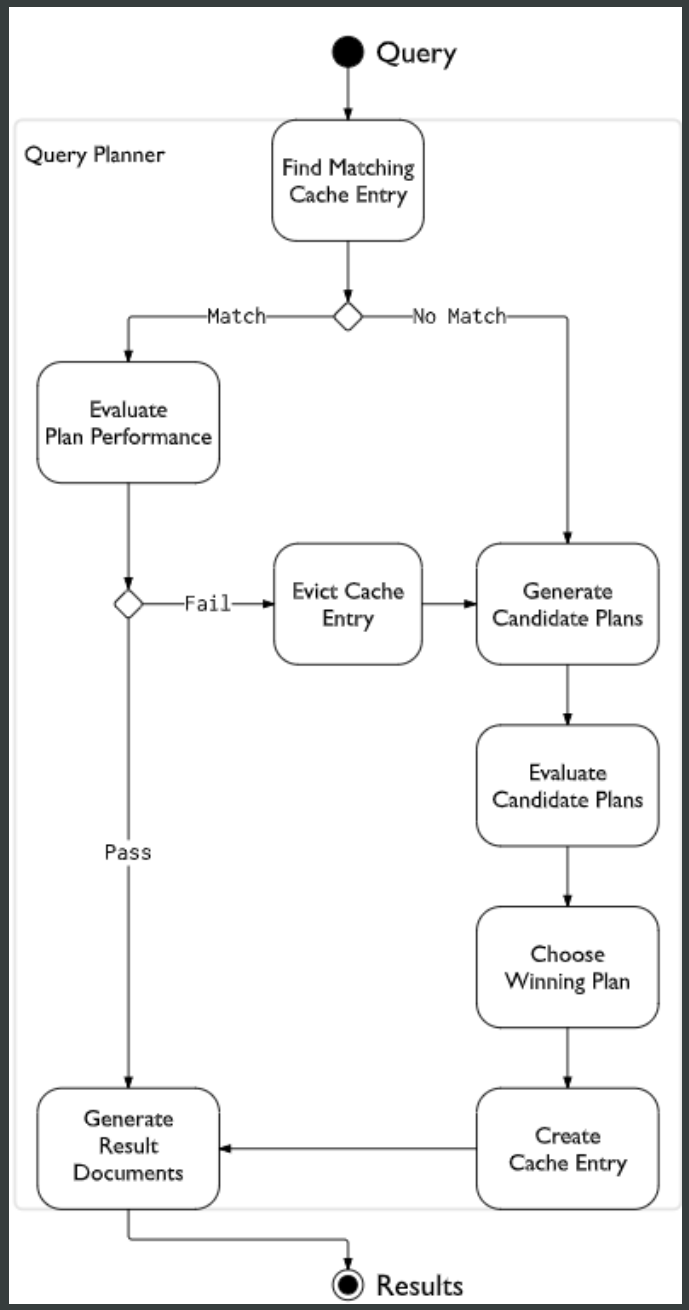
- Retornar los 10 logs más recientes:
  - db.system.profile.find ().limit (10).sort ({ts: -1}).pretty ()
- Retornar todas las operaciones excepto las operaciones command:
  - db.system.profile.find ({op: {$ne: “command”}}).pretty ()
- Retornar las operaciones para una colección particular:
  - db.system.profile.find ({ns: “mydb.test”}). pretty ()
- Retornar operaciones más lentas que 5 milisegundos:
  - db.system.profile.find ({millis :{ $gt: 5}}).pretty ()
- Retornar información de un cierto rango de tiempo:
- db.system.profile.find (
  - { ts: { $gt: new ISODate (“2018-11-05T09:00:00Z”), $lt: new ISODate (“2018-11-05T020:00:00Z”)}}).pretty ()

EJERCICIO 1

1. Ordenar por millis todos los documents para obtener las 10 queries más lentes
2. Obtener todas las queries que tardan más de 30 millisegundos en ejecutarse
3. Obtener las 10 queries de agregación/commandos más lentas
4. Obtener todas las operaciones que implicaron movimiento de documentos
5. Obtener las queries que realizan operaciónes de scanning demasiado largas (Más de 10000 documentos)
6. Tiempo máximo y medio de las operaciones de agregación
7. Tº máx y medio de operaciones de agregación por bd.

Query Plans

- El optimizador de consultas de MongoDB procesa las queries y elige el plan más eficiente según los índices disponibles. El sistema utiliza este plan cada vez que se ejecuta la consulta.
- Para ver estadísticas sobre el query plan de una consulta se utilizan:
  - db.collection.explain() o
  - cursor.explain()



Query Plans: db.collection.explain()

- El método tiene un parámetro opcional denominado «verbosity » que determina la cantidad de información que se retorna.. Puede tomar los valores:

1. queryPlanner. Es el valor por defecto.
   - Se ejecuta el optimizador de consultas para elegir el plan de ejecución.
   - Se retorna el plan de ejecución ganador.
2. executionStats
   - Se ejecuta el optimizador de consultas para elegir el plan de ejecución, ejecuta el plan de ejecución ganador, y retorna las estadísticas de la ejecución y el plan de ejecución ganador.
   - En operaciones de escritura, retorna información sobre las operaciones de actualización o borrado que serían realizadas, pero no las aplica a la base de datos.
   - No proporciona información sobre los planes rechazados.
3. allPlansExecution
   - Ejecuta el optimizador de consultas para elegir el plan ganador, y a continuación lo ejecuta.
   - Retorna estadísticas tanto sobre el plan de ejecución ganador como del resto de los planes considerados por el optimizador (información parcial), así como el plan de ejecución ganador.
   - En operaciones de escritura, retorna información sobre las operaciones de actualización o borrado que serían realizadas, pero no las aplica a la base de datos.