JSON 형태의 데이터가 저장됨

문서의 Field의 개수가 다르면 서로 다른 문서가 됨

{"greeting" : "Hello, World"}

{"greeting" : "Hello, World", "foo" : 3}

동일한 Field를 가지더라도 순서가 다르다면, 서로 다른 문서가 됨

문서의 key는 문자열로 하고, value는 어느 갓이든 쓸 수 있다.

Key는 예약어를 제외하고 모두 사용할 수 있다.

Key는 null문자를 포함할 수 없다

Key는 .과 $ 문자는 사용할 수 없다.

\_로 시작하는 문자는 사용하지 않는다

대소문자 및 데이터형을 정확히 구분하며, 다른 경우 서로 다른 문서로 인식한다.

인덱스생성및사용실스

모든 MongoDB의 컬렉션은 기본적으로 \_id 필드에 인덱스가 존재합니다.

mongod드라이버가 자동으로 \_id 필드 값을 ObjectId로 설정해줍니다.

MongoDB 드라이버가 지정하는 \_id 인덱스 외에도, 사용자가 지정 할 수 있는 단일 필드 인덱스

두개 이상의 필드를 사용하는 복합 인덱스

필드 타입이 배열인 필드에 인덱스를 적용 할 때는 Multikey 인덱스

지도의 좌표와 같은 데이터를 효율적으로 쿼리하기 위해서 사용되는 Geospatial(공간적) Index

텍스트 관련 데이터를 효율적으로 쿼리하기 위한 Text 인덱스

Hash 자료구조의 해쉬 (hashed) 인덱스

db.COLLECTION.createIndex( { KEY: 1 } )

db.report.createIndex( { score: 1 } )

db.report.find( { score: 57 } )

db.report.find( { score: { $gt: 60 } } )

db.report.createIndex( { age: 1, score: -1} )

partial 속성은 document의 조건을 정하여 일부 document에만 인덱스를 적용 할 때 사용

db.userinfo.createIndex( { email: 1 }, { unique: true } )

db.userinfo.createIndex( { firstName: 1, lastName: 1 }, { unique: true } )

db.store.createIndex(

{ name: 1 },

{ partialFilterExpression: { visitors: { $gt: 1000 } } }

)

Date 타입, 혹은 Date 배열 타입의 필드에 적용 할 수 있는 속성을 사용하여 document를 expire(만료) 시킬 수 있습니다.

db.notifications.createIndex( { "notifiedDate": 1 }, { expireAfterSeconds: 3600 } )

db.COLLECTION.getIndexes()

db.COLLECTION.dropIndex( { KEY: 1 } )

**Shard** 구성

https://docs.mongodb.com 과

https://c5ecbb38d638.gitbooks.io/mongodb-install-manual/content/replica.html 참고

Sharding의 가장 큰 목적은 파티셔닝을 통한 데이터 분산 처리와 성능 향상을 위한 load balancing입니다.

빅 데이터의 효율적 관리와 백업 및 복구 전략 수립을 위한 솔루션

3대 이상 샤드 구축 권장

싱글 노드를 운영할 때 요구되는 메모리 영역보다 최소 20~30% 이상의 추가 메모리 영역이 요구됩니다.

config 서버는 최소 3대 이상 활성화할 것을 권장

config 서버는 샤드 시스템 구축과 관련된 메타 데이터를 저장 관리하며 빅 데이터의 빠른 검색을 위한 인덱스 정보를 저장, 관리하기 때문에 샤드 서버와는 별도의 서버에 구축

>> 실습

-- 폴더를 미리 만듬.

c:\mongodb\shard1, c:\mongodb\shard2, c:\mongodb\shard3

c:\mongodb\config1, c:\mongodb\config2, c:\mongodb\config3

>> Shard서버 포트

mongod --shardsvr --replSet shardset --dbpath d:\mongodb\shard1 --port 40001

mongod --shardsvr --replSet shardset --dbpath d:\mongodb\shard2 --port 40002

mongod --shardsvr --replSet shardset --dbpath d:\mongodb\shard3 --port 40003

mongod --shardsvr --replSet shardset --dbpath /Users/kosta/data/shard1 --port 40001

mongod --shardsvr --replSet shardset --dbpath /Users/kosta/data/shard2 --port 40002

mongod --shardsvr --replSet shardset --dbpath /Users/kosta/data/shard3 --port 40003

mongo localhost:40001/admin

rs.initiate(

{

\_id :"shardset",

members: [

{ \_id : 0, host : "localhost:40001" },

{ \_id : 1, host : "localhost:40002" },

{ \_id : 2, host : "localhost:40003" }

]

}

)

>> Config서버 포트

config1 50001

config2 50002

config3 50003

mongod --configsvr --replSet configset --dbpath d:\mongodb\config1 --port 50001

mongod --configsvr --replSet configset --dbpath d:\mongodb\config2 --port 50002

mongod --configsvr --replSet configset --dbpath d:\mongodb\config3 --port 50003

mongod --configsvr --replSet configset --dbpath /Users/kosta/data/config1 --port 50001 &

mongod --configsvr --replSet configset --dbpath /Users/kosta/data/config2 --port 50002 &

mongod --configsvr --replSet configset --dbpath /Users/kosta/data/config3 --port 50003 &

mongo localhost:50001/admin

rs.initiate(

{

\_id: "configset",

configsvr: true,

members: [

{ \_id : 0, host : "localhost:50001" },

{ \_id : 1, host : "localhost:50002" },

{ \_id : 2, host : "localhost:50003" }

]

}

)

>> Route Server (mongos)

mongos --configdb configset/localhost:50001,localhost:50002,localhost:50003 --port 50000

>> mongos 접속 후 샤드 노드 등록

mongo localhost:50000/admin

db.runCommand({addshard :"shardset/localhost:40001"})

db.runCommand({addshard :" shardset/localhost:40002"})

db.runCommand({addshard :" shardset/localhost:40003"})

>> 해당 디비의 샤드 활성화

db.runCommand({enablesharding: "testDB"})

>> collection을 생성하고 샤드키 등록, 샤드키 설정시 반드시 인덱스를 생성해야 함.

db.runCommand( {shardcollection:"testDB.things", key:{empno:1, time:1} })

>> shard 노드 조회

db.runCommand({listshards:1})

>> testDB의 things 컬렉션에 데이터를 넣으면 샤드노드 서버에 데이터가 분산되서 들어감.

mongos 프로세스의 주요 특징

빅 데이터를 샤드 서버로 분배해 주는 프로세스 입니다.

하나 이상의 프로세스가 활성화 됩니다

Application Server에서 실행 가능 합니다

Config 서버로부터 Meta Data를 캐시합니다.

**Replica Set** 구성

https://docs.mongodb.com 과

https://c5ecbb38d638.gitbooks.io/mongodb-install-manual/content/replica.html 참고

기본적으로 클라이언트가 Primary를 이용해서 데이터의 읽기(Read)와 쓰기(Write)를 한다. 그리고 Primary는 데이터를 Secondary에 Replication(복제)한다.

Primary서버에 문제가 생겨서 데이터 통신이 되지 않는다면, 10 초안에 새로운 Primary 서버를 결정 한다.

mongod --dbpath D:\mongodb\data\replicaset-27030 --replSet replicaset --port 27030 --oplogSize 10

mongod --dbpath D:\mongodb\data\replicaset-27020 --replSet replicaset --port 27020 --oplogSize 10

mongod --dbpath D:\mongodb\data\replicaset-27017 --replSet replicaset --port 27017 --oplogSize 10

mongod --dbpath D:\mongodb\data\arbit --replSet replicaset --port 27050 --oplogSize 10

mongod --dbpath /Users/kosta/data/replicaset-27030 --replSet replicaset --port 27030 --oplogSize 10

mongod --dbpath /Users/kosta/data/replicaset-27020 --replSet replicaset --port 27020 --oplogSize 10

mongod --dbpath /Users/kosta/data/replicaset-27017 --replSet replicaset --port 27017 --oplogSize 10

mongod --dbpath /Users/kosta/data/arbit --replSet replicaset --port 27050 --oplogSize 10

Replica set 설정

Primary Server로 사용될 mongod에 접속

rptmongo - Mongod instance 생성시 정의된 replica set 이름

members.\_id:1 이 Primary

members.\_id:2~ 은 Secondary

Secondary중 arbiterOnly : true는 arbiter 설정

mongo localhost:27017/admin

db.runCommand({ "replSetInitiate" :

{

"\_id" : "replicaset",

"members":

[

{"\_id": 1, "host" : "localhost:27017"},

{"\_id": 2, "host" : "localhost:27020"},

{"\_id": 3, "host" : "localhost:27030"},

{"\_id": 4, "host" : "localhost:27050", arbiterOnly: true}

]

}

}

)

Replica 테스트

저장 후 복제 확인 테스트

Primary에 데이터 저장

mongo localhost:27017

use test

for (var n = 10 ; n < 30 ; n++) db.things.save( { replno:n , replName: "repl test"} )

Secondary 복제 확인

mongo localhost:27020

use test

db.setSlaveOk()

db.things.find()

FailOver 테스트

Primary server shutdown

mongo localhost:27017

db.shutdownServer()

Secondary server가 Primary server로 변경되었는지 확인

프롬프트로 확인 가능함

mongo localhost:27020

mongo localhost:27030

MongoDB Transaction : Two Task Commit;