Week 5: Document Databases and Column-family Stores

DIANE WOODBRIDGE, PH.D

MongoDB Interview Questions

MongoDB's type

MongoDB's characteristics

Alternative databases

Supported programming languages

Index

Aggregation Operations(aggregation pipeline)

Sharding

Replication

GridFS

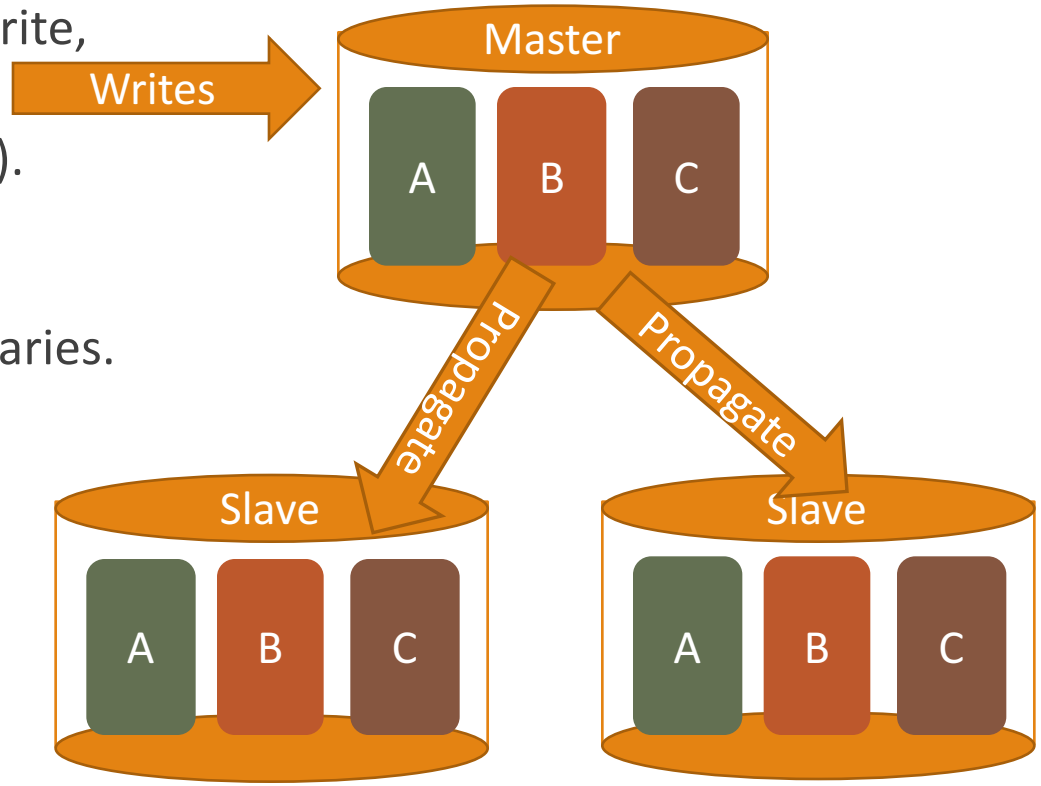
ObjectId

Consistency

MongoDB

Replication

- Clients can send a master(Primary) read, write, command, index, etc. requests.
- Clients cannot write to slaves (Secondaries).
- By default, clients cannot read from secondaries. However by explicitly setting "setSlaveOk()" client can read from secondaries.

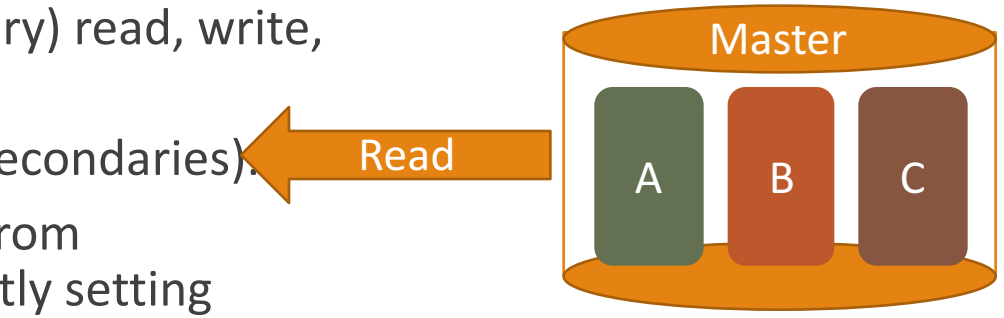


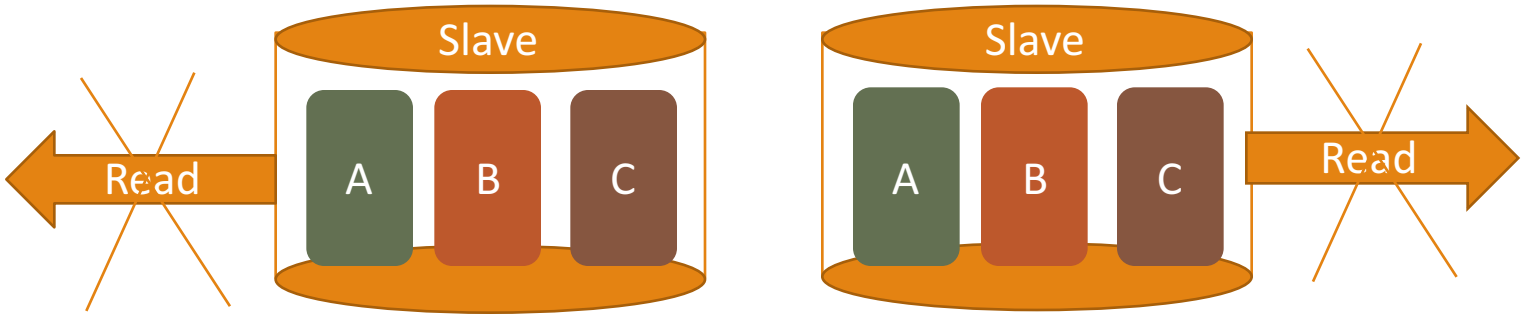
https://docs.mongodb.com/manual/replication/

MongoDB

Replication

- Clients can send a master(Primary) read, write, command, index, etc. requests.
- Clients cannot write to slaves (Secondaries).
- By default, clients cannot read from secondaries. However by explicitly setting "setSlaveOk()" client can read from secondaries.





https://docs.mongodb.com/manual/replication/

MongoDB

Replication

- Replica sets should always have an odd number of members.
  This ensures that election using quorum will proceed smoothly.
- Create 3 folders.
  - rs_1, rs_2, rs_3
  - start all 3 mongod processes with
    - --dbpath definite_path_of_each_folder
    - --port port_number
    - --replSet replica_set_name

MongoDB

Replication

- Replica sets should always have an odd number of members.
  This ensures that election using quorum will proceed smoothly.
- Create 3 folders.
  - start all 3 mongod processes.

```
ML-ITS-603436:~ dwoodbridge$ mongod --port 20003 --dbpath ~/Class/MSAN697/Week5/mongodb/replica_sets/rs_1 --replSet test
2016-11-14T21:02:14.233-0800 I CONTROL  [initandlisten] MongoDB starting : pid=2191 port=20003 dbpath=/Users/dwoodbridge/Class/MSAN697/Week5/
mongodb/replica_sets/rs_1 64-bit host=ML-ITS-603436
2016-11-14T21:02:14.233-0800 I CONTROL  [initandlisten] db version v3.2.10
2016-11-14T21:02:14.233-0800 I CONTROL  [initandlisten] git version: 79d9b3ab5ce20f51c272b4411202710a082d0317
2016-11-14T21:02:14.233-0800 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2j  26 Sep 2016
2016-11-14T21:02:14.233-0800 I CONTROL  [initandlisten] allocator: system
2016-11-14T21:02:14.233-0800 I CONTROL  [initandlisten] modules: none
2016-11-14T21:02:14.233-0800 I CONTROL  [initandlisten] build environment:
2016-11-14T21:02:14.233-0800 I CONTROL  [initandlisten]     distarch: x86_64
2016-11-14T21:02:14.233-0800 I CONTROL  [initandlisten]     target_arch: x86_64
2016-11-14T21:02:14.233-0800 I CONTROL  [initandlisten] options: { net: { port: 20003 }, replication: { replSet: "test" }, storage: { dbPath:
 "/Users/dwoodbridge/Class/MSAN697/Week5/mongodb/replica_sets/rs_1" } }
2016-11-14T21:02:14.234-0800 I -        [initandlisten] Detected data files in /Users/dwoodbridge/Class/MSAN697/Week5/mongodb/replica_sets/rs
_1 created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2016-11-14T21:02:14.234-0800 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=9G,session_max=20000,eviction=(threads_max=
4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),c
heckpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2016-11-14T21:02:14.705-0800 I STORAGE  [initandlisten] Starting WiredTigerRecordStoreThread local.oplog.rs
2016-11-14T21:02:14.705-0800 I STORAGE  [initandlisten] The size storer reports that the oplog contains 4 records totaling to 368 bytes
2016-11-14T21:02:14.705-0800 I STORAGE  [initandlisten] Scanning the oplog to determine where to place markers for truncation
```

MongoDB

Replication

- Start the mongo shell with one of the node.

```
ML-ITS-603436:~ dwoodbridge$ mongo --port 20003
MongoDB shell version: 3.2.10
connecting to: 127.0.0.1:20003/test
Server has startup warnings:
2016-11-14T17:20:10.905-0800 I CONTROL  [initandlisten]
2016-11-14T17:20:10.905-0800 I CONTROL  [initandlisten] ** WARNING: soft rlimits
 too low. Number of files is 256, should be at least 1000
>
```

MongoDB

Replication

- Initiate the replica set using the rs.initiate() method and a configuration document
  - _id : replSet Name.
  - members : An array of each replica set information.

```
> config_test={_id: "test", members:[{_id:0,host:"localhost:20003"},{_id:1,host:"localhost:20004"},{_id:2,host:"localhost:20005"}]}
{
        "_id" : "test",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "localhost:20003"
                },
                {
                        "_id" : 1,
                        "host" : "localhost:20004"
                },
                {
                        "_id" : 2,
                        "host" : "localhost:20005"
                }
        ]
}
> rs.initiate(config_test)
{ "ok" : 1 }
```

https://docs.mongodb.com/v3.2/tutorial/deploy-shard-cluster/

MongoDB

Replication

- Check the status.
  - rs.status()

```
> rs.status()
{
        "set" : "test",
        "date" : ISODate("2016-11-15T01:59:09.529Z"),
        "myState" : 1,
        "term" : NumberLong(1),
        "heartbeatIntervalMillis" : NumberLong(2000),
        "members" : [
                {
                        "_id" : 0,
                        "name" : "localhost:20003",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 1474,
                        "optime" : {
                                "ts" : Timestamp(1479175094, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2016-11-15T01:58:14Z"),
                        "lastHeartbeat" : ISODate("2016-11-15T01:59:07Z"),
                        "lastHeartbeatRecv" : ISODate("2016-11-15T01:59:08Z"),
                        "pingMs" : NumberLong(0),
                        "electionTime" : Timestamp(1479174993, 2),
                        "electionDate" : ISODate("2016-11-15T01:56:33Z"),
                        "configVersion" : 1
                },
                {
                        "_id" : 1,
                        "name" : "localhost:20004",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 65,
                        "optime" : {
                                "ts" : Timestamp(1479175094, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2016-11-15T01:58:14Z"),
                        "lastHeartbeat" : ISODate("2016-11-15T01:59:07Z"),
                        "lastHeartbeatRecv" : ISODate("2016-11-15T01:59:08Z"),
                        "pingMs" : NumberLong(0),
                        "syncingTo" : "localhost:20003",
                        "configVersion" : 1
                },
                {
                        "_id" : 2,
                        "name" : "localhost:20005",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 65,
                        "optime" : {
                                "ts" : Timestamp(1479175094, 1),
```

https://docs.mongodb.com/v3.2/tutorial/deploy-shard-cluster/

MongoDB

Replication

- Connect to the PRIMARY and its database.
- Check whether it is a master.

```
> conn1 = new Mongo("localhost:20003")
connection to localhost:20003
> primaryDB = conn1.getDB("test")
test
> primaryDB.isMaster()
{
        "hosts" : [
                "localhost:20003",
                "localhost:20004",
                "localhost:20005"
        ],
        "setName" : "test",
        "setVersion" : 1,
        "ismaster" : true,
        "secondary" : false,
        "primary" : "localhost:20003",
        "me" : "localhost:20003",
```

Example

Insert data into the friend collection by primaryDB.friend.insert()

Connect to a secondary node and read data from the friend collection.

Try to use connection_name.setSlaveOk(), before writing to the secondary.

Write data into the secondary.

```
WriteResult({ "writeError" : { "code" : 10107, "errmsg" : "not master" } })
```

MongoDB

Replication

- Automatic failover
  - If the primary database goes down, MongoDB will automatically choose a new primary.

```
test:PRIMARY> primaryDB.adminCommand({"shutdown":1})
2016-11-14T22:03:47.013-0800 E QUERY    [thread1] Error: error doing query: failed: network error while attempting to r
ost 'localhost:20003' ;
DB.prototype.runCommand@src/mongo/shell/db.js:135:1
DB.prototype.adminCommand@src/mongo/shell/db.js:153:16
@(shell):1:1

2016-11-14T22:03:47.019-0800 I NETWORK  [thread1] trying reconnect to localhost:20003 (127.0.0.1) failed
2016-11-14T22:03:48.317-0800 I NETWORK  [thread1] Socket recv() errno:54 Connection reset by peer 127.0.0.1:20003
2016-11-14T22:03:48.317-0800 I NETWORK  [thread1] SocketException: remote: (NONE):0 error: 9001 socket exception [RECV_
0003]
2016-11-14T22:03:48.317-0800 I NETWORK  [thread1] reconnect localhost:20003 (127.0.0.1) failed failed
```

MongoDB

Replication

- Automatic failover
  - If the primary database goes down, MongoDB will automatically choose a new primary.

```
> secondaryDB.isMaster()
{
        "hosts" : [
                "localhost:20003",
                "localhost:20004",
                "localhost:20005"
        ],
        "setName" : "test",
        "setVersion" : 1,
        "ismaster" : false,
        "secondary" : true,
        "primary" : "localhost:20005",
        "me" : "localhost:20004",
        "maxBsonObjectSize" : 16777216,
        "maxMessageSizeBytes" : 48000000,
        "maxWriteBatchSize" : 1000,
        "localTime" : ISODate("2016-11-15T06:05:08.514Z"),
        "maxWireVersion" : 4,
        "minWireVersion" : 0,
        "ok" : 1
}
```

MongoDB

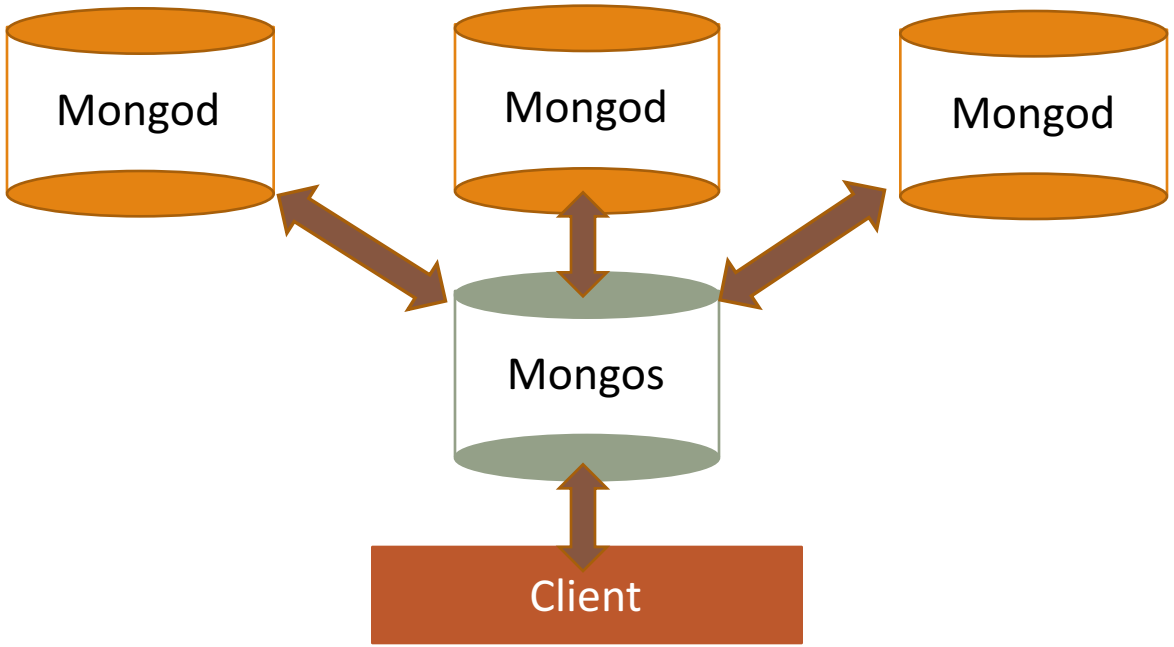
Replication

- Clients can send a master (primary) all operations including read, write, command, index builds, etc.
- Clients cannot write to slaves (secondaries).
- By default, clients cannot read from secondaries.

MongoDB

Sharding

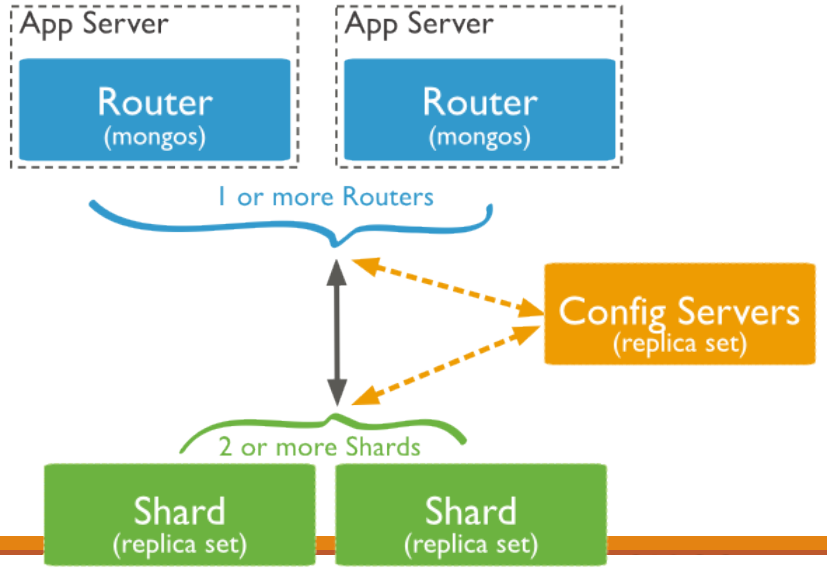
- MongoDB supports autosharding.
  - Create a cluster of many machines (shards).
  - Mongos : Routing process to manage storing different data on different servers and query against the right server.



MongoDB

Sharding

- Config servers store the metadata for a sharded cluster.
  - The metadata stores
    - state and organization for all data and components within the sharded cluster.
    - the list of data chunks on every shard and the ranges that define the chunks.
  - In MongoDB 3.2+, you can deploy config servers as a replica set.



https://docs.mongodb.com/v3.2/tutorial/convert-replica-set-to-replicated-shard-cluster/

MongoDB

Sharding

- Create 4 folders and start mongods.
  - Three for sharding servers.
    - Start each mongod with --dbpath, --port port_number and --shardsvr.
      - --shardsvr : Enabling sharding on this specific instance.
  - One for configuration server.
    - Start mongod with --dbpath, --port and --configsvr.
      - --configsvr : Indicating that this instance will be used for configuration.

MongoDB

Sharding

- Create 4 folders and start mongods.
  - Three for sharding servers.
    - Start each mongod with --dbpath, --port port_number and --shardsvr.

```
ML-ITS-603436:~ dwoodbridge$ mongod --dbpath ~/Class/MSAN697/Week5/mongodb/sharding_sets/sh_2 --port 20007 --shardsvr
2016-11-15T11:23:00.580-0800 I CONTROL  [initandlisten] MongoDB starting : pid=6759 port=20007 dbpath=/Users/dwoodbridge/Class/MSAN697/Week5/mongodb/shardi
_sets/sh_2 64-bit host=ML-ITS-603436
2016-11-15T11:23:00.581-0800 I CONTROL  [initandlisten] db version v3.2.10
2016-11-15T11:23:00.581-0800 I CONTROL  [initandlisten] git version: 79d9b3ab5ce20f51c272b4411202710a082d0317
2016-11-15T11:23:00.581-0800 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2j  26 Sep 2016
2016-11-15T11:23:00.581-0800 I CONTROL  [initandlisten] allocator: system
2016-11-15T11:23:00.581-0800 I CONTROL  [initandlisten] modules: none
2016-11-15T11:23:00.581-0800 I CONTROL  [initandlisten] build environment:
2016-11-15T11:23:00.581-0800 I CONTROL  [initandlisten]     distarch: x86_64
2016-11-15T11:23:00.581-0800 I CONTROL  [initandlisten]     target_arch: x86_64
2016-11-15T11:23:00.581-0800 I CONTROL  [initandlisten] options: { net: { port: 20007 }, sharding: { clusterRole: "shardsvr" }, storage: { dbPath: "/Users/
oodbridge/Class/MSAN697/Week5/mongodb/sharding_sets/sh_2" } }
2016-11-15T11:23:00.582-0800 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=9G,session_max=20000,eviction=(threads_max=4),config_base
alse,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2G
,statistics_log=(wait=0),
2016-11-15T11:23:00.967-0800 I CONTROL  [initandlisten]
2016-11-15T11:23:00.968-0800 I CONTROL  [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should be at least 1000
2016-11-15T11:23:00.968-0800 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory '/Users/dwoodbridge/Class/MSAN697/Wee
/mongodb/sharding_sets/sh_2/diagnostic.data'
```

MongoDB

Sharding

- Create 4 folders and start mongods.
  - Three for sharding servers.
    - Start each mongod with --dbpath, --port port_number and --shardsvr.
      - --shardsvr : Enabling sharding on this specific instance.
  - One for configuration server.
    - Start mongod with --dbpath, --port and --configsvr.

```
ML-ITS-603436:~ dwoodbridge$ mongod --dbpath ~/Class/MSAN697/Week5/mongodb/sharding_sets/config_1 --port 20009 --configsvr
2016-11-15T11:23:17.322-0800 I CONTROL  [initandlisten] MongoDB starting : pid=6781 port=20009 dbpath=/Users/dwoodbridge/Class/MSAN697/Week5/mongodb/sharding_sets/config_1 master=1 64-bit host=ML-ITS-603436
2016-11-15T11:23:17.323-0800 I CONTROL  [initandlisten] db version v3.2.10
2016-11-15T11:23:17.323-0800 I CONTROL  [initandlisten] git version: 79d9b3ab5ce20f51c272b4411202710a082d0317
2016-11-15T11:23:17.323-0800 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.2j  26 Sep 2016
2016-11-15T11:23:17.323-0800 I CONTROL  [initandlisten] allocator: system
2016-11-15T11:23:17.323-0800 I CONTROL  [initandlisten] modules: none
2016-11-15T11:23:17.323-0800 I CONTROL  [initandlisten] build environment:
2016-11-15T11:23:17.323-0800 I CONTROL  [initandlisten]     distarch: x86_64
2016-11-15T11:23:17.324-0800 I CONTROL  [initandlisten]     target_arch: x86_64
2016-11-15T11:23:17.324-0800 I CONTROL  [initandlisten] options: { net: { port: 20009 }, sharding: { clusterRole: "configsvr" }, storage: { dbPath: "/Users/dwoodbridge/Class/MSAN697/Week5/mongodb/sharding_sets/config_1" } }
2016-11-15T11:23:17.324-0800 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=9G,session_max=20000,eviction=(threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2016-11-15T11:23:17.715-0800 I CONTROL  [initandlisten]
2016-11-15T11:23:17.715-0800 I CONTROL  [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should be at least 1000
```

MongoDB

Sharding

- Run mongos with --configdb and --chunksize (optional).
  - --configdb : Tells the routing process which Configuration Server to connect to for it’s configuration.

```
ML-ITS-603436:~ dwoodbridge$ mongos --configdb localhost:20009 --chunkSize 1
2016-11-15T11:23:25.613-0800 W SHARDING [main] Running a sharded cluster with fewer than 3 config servers should only be
nd is not recommended for production.
2016-11-15T11:23:25.624-0800 I SHARDING [mongosMain] MongoS version 3.2.10 starting: pid=6792 port=27017 64-bit host=ML-I
)
2016-11-15T11:23:25.624-0800 I CONTROL  [mongosMain] db version v3.2.10
2016-11-15T11:23:25.624-0800 I CONTROL  [mongosMain] git version: 79d9b3ab5ce20f51c272b4411202710a082d0317
2016-11-15T11:23:25.624-0800 I CONTROL  [mongosMain] OpenSSL version: OpenSSL 1.0.2j  26 Sep 2016
2016-11-15T11:23:25.624-0800 I CONTROL  [mongosMain] allocator: system
2016-11-15T11:23:25.624-0800 I CONTROL  [mongosMain] modules: none
2016-11-15T11:23:25.624-0800 I CONTROL  [mongosMain] build environment:
2016-11-15T11:23:25.624-0800 I CONTROL  [mongosMain]     distarch: x86_64
2016-11-15T11:23:25.624-0800 I CONTROL  [mongosMain]     target_arch: x86_64
2016-11-15T11:23:25.624-0800 I CONTROL  [mongosMain] options: { sharding: { chunkSize: 1, configDB: "localhost:20009" } }
2016-11-15T11:23:25.624-0800 I SHARDING [mongosMain] Updating config server connection string to: localhost:20009
2016-11-15T11:23:25.639-0800 I SHARDING [LockPinger] creating distributed lock ping thread for localhost:20009 and proces
37805:-556389220 (sleeping for 30000ms)
2016-11-15T11:23:25.683-0800 I SHARDING [LockPinger] cluster localhost:20009 pinged successfully at 2016-11-15T11:23:25.6
 pinger 'localhost:20009/ML-ITS-603436:27017:1479237805:-556389220', sleeping for 30000ms
2016-11-15T11:23:25.860-0800 I SHARDING [mongosMain] distributed lock 'configUpgrade/ML-ITS-603436:27017:1479237805:-5563
lizing config database to new format v6', ts : 582b60adcb5727eb4bf62fa0
2016-11-15T11:23:25.861-0800 I SHARDING [mongosMain] initializing config server version to 6
2016-11-15T11:23:25.861-0800 I SHARDING [mongosMain] writing initial config version at v6
2016-11-15T11:23:25.938-0800 I SHARDING [mongosMain] initialization of config server to v6 successful
```

MongoDB

Sharding

- Run mongo and add shards.
  - use admin
  - sh.addShard("localhost:20006")
  - sh.addShard("localhost:20007")
  - sh.addShard("localhost:20008")

```
Last login: Tue Nov 15 11:23:19 on ttys004
ML-ITS-603436:~ dwoodbridge$ mongo
MongoDB shell version: 3.2.10
connecting to: test
mongos> use admin
switched to db admin
mongos> sh.addShard("localhost:20006")
{ "shardAdded" : "shard0000", "ok" : 1 }
mongos> sh.addShard("localhost:20007")
{ "shardAdded" : "shard0001", "ok" : 1 }
mongos> sh.addShard("localhost:20008")
{ "shardAdded" : "shard0002", "ok" : 1 }
mongos>
```

MongoDB

Sharding

- Run mongo and add shards.
  - use admin
  - sh.addShard("localhost:20006")
  - sh.addShard("localhost:20007")
  - sh.addShard("localhost:20008")

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("582b60adcb5727eb4bf62fa1")
}
  shards:
        {  "_id" : "shard0000",  "host" : "localhost:20006" }
        {  "_id" : "shard0001",  "host" : "localhost:20007" }
        {  "_id" : "shard0002",  "host" : "localhost:20008" }
  active mongoses:
        "3.2.10" : 1
  balancer:
        Currently enabled:  yes
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
```

MongoDB

Sharding

- Enable sharding and indicate collection and sharding key/method.
  - sh.enableSharding(db_name)
  - sh.shardCollection(db_name.collection_name, {key:”hashed"})

```
mongos> sh.enableSharding("mydb")
{ "ok" : 1 }
mongos> sh.shardCollection("mydb.friends",{"name":"hashed"})
{ "collectionsharded" : "mydb.friends", "ok" : 1 }
```

MongoDB

Sharding

- Enable sharding and indicate collection and sharding key/method.
  - sh.enableSharding(db_name)
  - sh.shardCollection(db_name.collection_name, {key:”hashed"})

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("582b60adcb5727eb4bf62fa1")
  }
  mydb.friends
          shard key: { "name" : "hashed" }
          unique: false
          balancing: true
          chunks:
                  shard0000       2
                  shard0001       2
                  shard0002       2
          { "name" : { "$minKey" : 1 } } -->> { "name" : NumberLong("-6148914691236517204") } on : shard0000 Timestamp(3, 2)
          { "name" : NumberLong("-6148914691236517204") } -->> { "name" : NumberLong("-3074457345618258602") } on : shard0000 Timestamp(3, 3)
          { "name" : NumberLong("-3074457345618258602") } -->> { "name" : NumberLong(0) } on : shard0001 Timestamp(3, 4)
          { "name" : NumberLong(0) } -->> { "name" : NumberLong("3074457345618258602") } on : shard0001 Timestamp(3, 5)
          { "name" : NumberLong("3074457345618258602") } -->> { "name" : NumberLong("6148914691236517204") } on : shard0002 Timestamp(3, 6)
          { "name" : NumberLong("6148914691236517204") } -->> { "name" : { "$maxKey" : 1 } } on : shard0002 Timestamp(3, 7)
  databases:
```

Example

Using the sharding enabled db and collection, insert data with the key.

Check how data are distributed using, db. collection_name.getShardDistribution().

```
mongos> db.friends.getShardDistribution()

Shard shard0000 at localhost:20006
 data : 98B docs : 2 chunks : 2
 estimated data per chunk : 49B
 estimated docs per chunk : 1

Shard shard0001 at localhost:20007
 data : 37B docs : 1 chunks : 2
 estimated data per chunk : 18B
 estimated docs per chunk : 0

Shard shard0002 at localhost:20008
 data : 89B docs : 2 chunks : 2
 estimated data per chunk : 44B
 estimated docs per chunk : 1

Totals
 data : 224B docs : 5 chunks : 6
 Shard shard0000 contains 43.75% data, 40% docs in cluster, avg obj size on shard : 49B
 Shard shard0001 contains 16.51% data, 20% docs in cluster, avg obj size on shard : 37B
 Shard shard0002 contains 39.73% data, 40% docs in cluster, avg obj size on shard : 44B
```

MongoDB

Consider to apply sharding and replication together.

Further Readings

Convert replica set to replicated shard cluster:

https://docs.mongodb.com/v3.2/tutorial/convert-replica-set-to-replicated-shard-cluster/

Convert sharded cluster to replica set:

https://docs.mongodb.com/manual/tutorial/convert-sharded-cluster-to-replica-set/

MongoDB

Features

- Consistency
  - Use replica sets and wait for the writes to be replicated to all/some slaves.
  - Read consistency
    - Secondaries will refuse read requests by default to prevent from reading from stale data.
    - Setting up setSlaveOk() allows read operations to run on secondaries .
  - Write consistency
    - For a replica set, the default writeConcern() requests acknowledgement only from the primary.
    - Overriding the write concern allows to confirm write operations on a specified number of the replica set members.
      - db.friend.insert({ name: ”Jonathan"}, { writeConcern: {w: “majority”} })

Make sure to propagate to the primary and at least one secondary.

https://docs.mongodb.com/manual/reference/write-concern/

Cassandra Interview Questions

Cassandra data model

Tunable consistency (Quorum)

Difference between RDBMS and Cassandra

CAP Theorem

Cassandra query language

cqlsh

Compaction

Super column

Column family

Memtable

SSTable

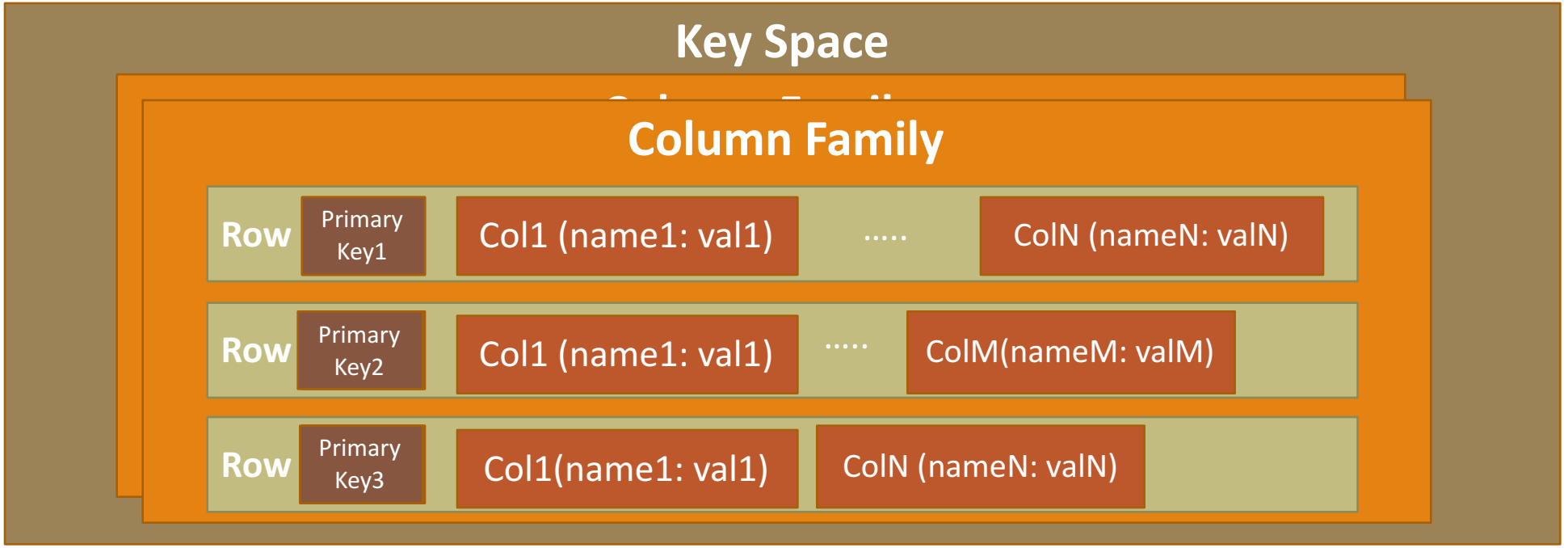
Keyspace

Column-Family Stores

Column-family Stores: also called as wide column stores.

Store data in column families where rows that have many columns associated with a primary key (row key).

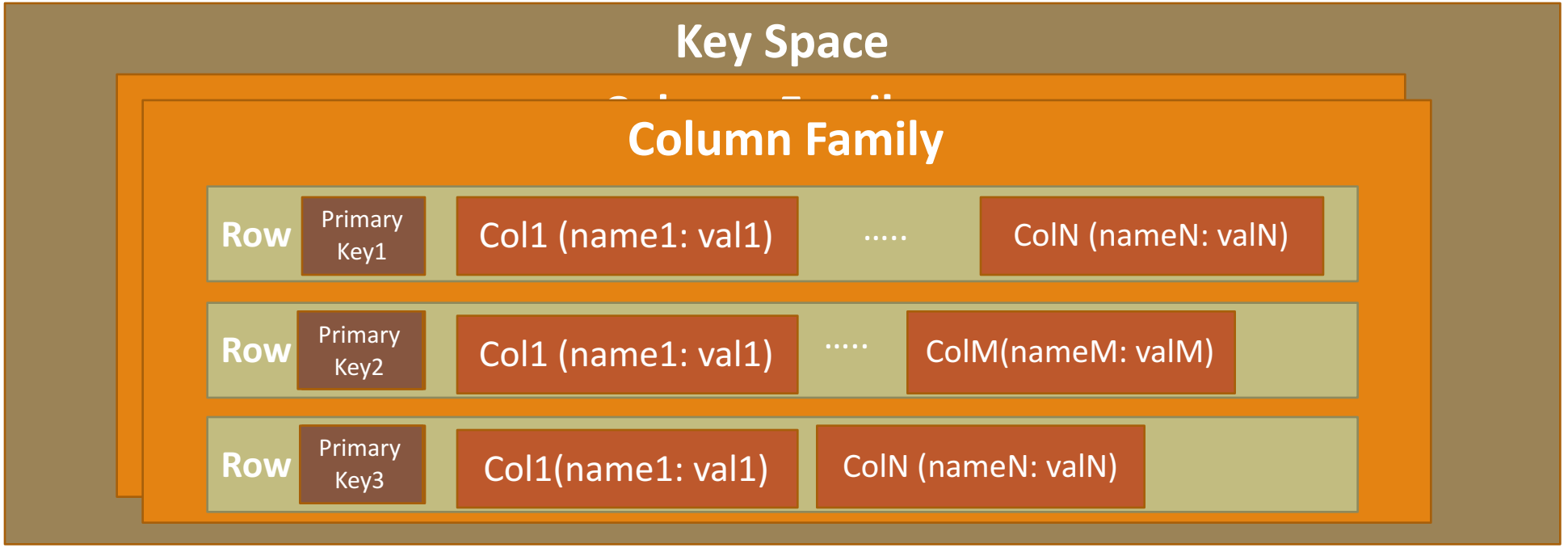
Column families are groups of related data that is often accessed together.



Column-Family Stores

Various rows do not have to have the same columns.

Columns can be added to any row at any time without having to add it to other rows.



Column-Family Stores

8 systems in ranking, November 2016

Rank			DBMS	Database Model	Score		
Nov 2016	Oct 2016	Nov 2015			Nov 2016	Oct 2016	Nov 2015
1.	1.	1.	Cassandra	Wide column store	133.97	-1.09	+1.05
2.	2.	2.	HBase	Wide column store	58.74	+0.54	+2.28
3.	3.	3.	Accumulo	Wide column store	3.47	+0.04	-0.47
4.	4.	4.	Hypertable	Wide column store	0.52	-0.03	-0.18
5.	5.		Google Cloud Bigtable	Wide column store	0.31	-0.04	
6.	6.	↓ 5.	Sqrrl	Multi-model	0.28	+0.02	-0.14
7.	7.		ScyllaDB	Wide column store	0.21	+0.06	
8.			MapR-DB	Multi-model	0.15		

Cassandra Installation

1. Install Cassandra 3.9 using Homebrew

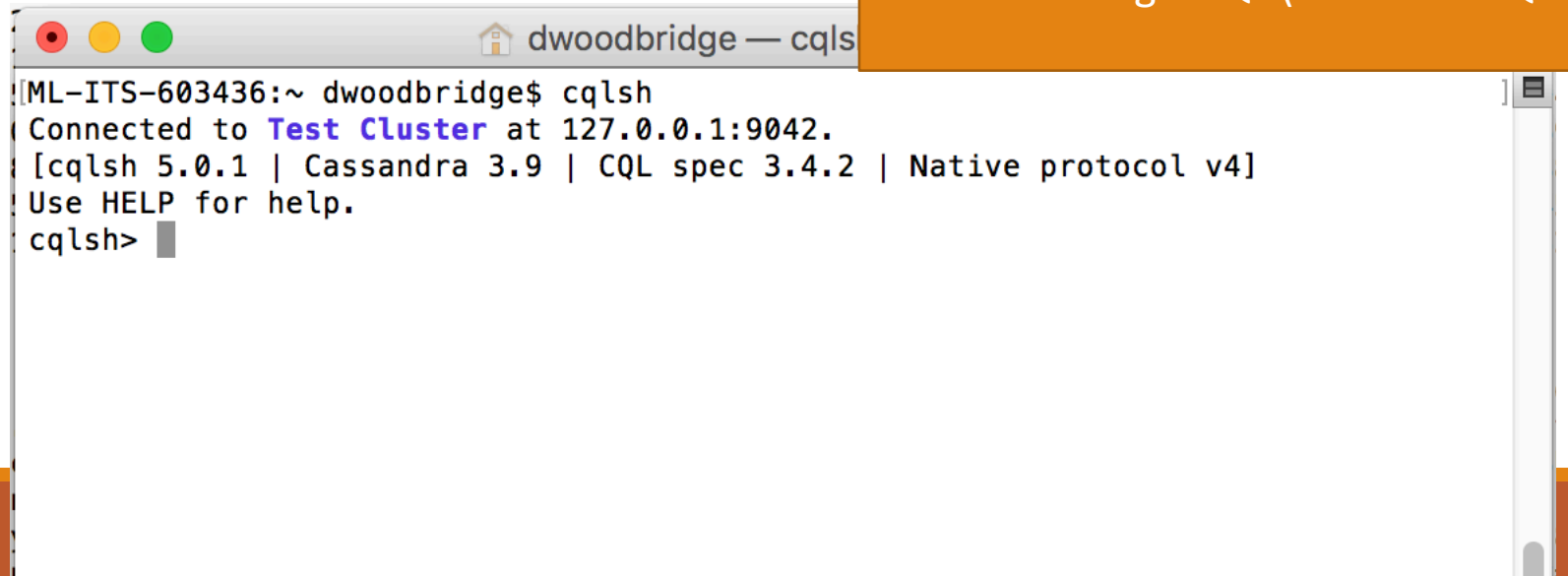
```
brew install cassandra
```

2. on terminal, type cassandra -f to start the server

3. on a new terminal, type cqlsh

4. If you see a screen as below, you're good to go.

cqlsh : command line shell to interact with Cassandra through CQL (Cassandra Query Language)



Cassandra

Why Cassandra?

- Open source
- Distributed and decentralized
- Highly scalable
- High availability and fault tolerant
- Tunable consistent
- High performance (Writes *)

Cassandra

Terminology

- Keyspace
- Column family

SQL(RDBMS)	Cassandra
	Cluster
Database	Keyspace
Table	Column Family/Table
Row	Row
Column	Column (can be different per row)

http://cassandra.apache.org/doc/latest/

Cassandra

Structure

- Column : Basic/atomic unit of storage.
  - Name(Key)-value pair.
  - Stored with a timestamp.
    - Used to expire data, resolve conflicts, deal with stale data, etc.
  - Super column : a map of columns.
    - Pros : Good to keep related data together.
    - Cons: When some of the columns are not needed most of the time, the columns are still fetched and deserialized.
- Row : collection of columns attached to a key.
  - Data is organized around rows.
  - Doesn't need to have all the same columns. (Flexible schema)
  - Wide row (Partition)
- Column family : collection of similar rows.

http://cassandra.apache.org/doc/latest/

Cassandra

Column Data Type

CQL Type	Description
ascii	US-ASCII character string
bigint	64-bit signed long
blob	Arbitrary bytes (no validation), expressed as hexadecimal
boolean	true or false
counter	Distributed counter value (64-bit long)
decimal	Variable-precision decimal
double	64-bit IEEE-754 floating point
float	32-bit IEEE-754 floating point
inet	IP address string in IPv4 or IPv6 format*
int	32-bit signed integer
list	A collection of one or more ordered elements
map	A JSON-style array of literals: { literal : literal, literal : literal ... }
set	A collection of one or more elements
text	UTF-8 encoded string
timestamp	Date plus time, encoded as 8 bytes since epoch
uuid	A UUID in standard UUID format
timeuuid	Type 1 UUID only (CQL 3)
varchar	UTF-8 encoded string
varint	Arbitrary-precision integer

https://docs.datastax.com/en/cql/3.0/cql/cql_reference/cql_data_types_c.html

Cassandra

Column Data Type

- uuid : type 4 universal unique identifier. Fixed length random number (128 bit value) .
- timeuuid : type1 universal unique identifier. Combinatino of MAC address, system time and sequence numbers.
- Set : unordered collection. {}
- List : ordered collection. []
- Map : collection of key-value pairs.
- User defined type (UDT). : considered as a set.

```
CREATE TYPE name { name type, name type, …};
```

```
cqlsh:mydb> CREATE TYPE address (street text, city text, state text, zip int);
```

https://docs.datastax.com/en/cql/3.0/cql/cql_reference/cql_data_types_c.html

Cassandra

Start Cassandra

```
cassandra –f
```

- -f : all the server logs will print to standard out.

Stop Casssandra

```
ps auwx | grep cassandra
sudo kill pid
```

Cassandra

CQL – Cassandra Query Language.

```
cqlsh> help

Documented shell commands:
===========================
CAPTURE  CLS          COPY  DESCRIBE  EXPAND  LOGIN   SERIAL  SOURCE   UNICODE
CLEAR    CONSISTENCY  DESC  EXIT      HELP    PAGING  SHOW    TRACING

CQL help topics:
================
AGGREGATES               CREATE_KEYSPACE           DROP_TRIGGER      TEXT
ALTER_KEYSPACE           CREATE_MATERIALIZED_VIEW  DROP_TYPE         TIME
ALTER_MATERIALIZED_VIEW  CREATE_ROLE               DROP_USER         TIMESTAMP
ALTER_TABLE              CREATE_TABLE              FUNCTIONS         TRUNCATE
ALTER_TYPE               CREATE_TRIGGER            GRANT             TYPES
ALTER_USER               CREATE_TYPE               INSERT            UPDATE
APPLY                    CREATE_USER               INSERT_JSON       USE
ASCII                    DATE                      INT               UUID
BATCH                    DELETE                    JSON
BEGIN                    DROP_AGGREGATE            KEYWORDS
BLOB                     DROP_COLUMNFAMILY         LIST_PERMISSIONS
BOOLEAN                  DROP_FUNCTION             LIST_ROLES
COUNTER                  DROP_INDEX                LIST_USERS
CREATE_AGGREGATE         DROP_KEYSPACE             PERMISSIONS
CREATE_COLUMNFAMILY      DROP_MATERIALIZED_VIEW    REVOKE
CREATE_FUNCTION          DROP_ROLE                 SELECT
CREATE_INDEX             DROP_TABLE                SELECT_JSON
```

Cassandra

Create and use a keyspace

```
CREATE KEYSPACE key_space_name WITH replication = {’class’:
strategy_name , ‘replication_factor’:number} AND durable_writes = true;
```

USE key_space_name;

```
CREATE KEYSPACE mydb WITH replication = {'class':'SimpleStrategy', 'replication_factor':'1'} AND durable_writes=true;
```

```
cqlsh> USE mydb;
cqlsh:mydb>
```

https://docs.datastax.com/en/cql/3.1/cql/cql_reference/create_keyspace_r.html

Cassandra

Create a table.

- CREATE TABLE table_name (name type, name type, PRIMARY KEY(name));

```
cqlsh:mydb> CREATE TABLE friend(name text, PRIMARY KEY (name));
```

```
cqlsh:mydb> DESCRIBE TABLE friend;

CREATE TABLE mydb.friend (
    name text PRIMARY KEY
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

Data access requires primary key!
Primary key cannot be modified.

Cassandra

Insert and select data.

```
INSERT INTO table_name (column_names) VALUES (values);

SELECT column_names FROM table_name;
```

```
cqlsh:mydb> INSERT INTO friend (name) VALUES ('Diane MK Woodbridge');
cqlsh:mydb> SELECT * From friend
        ... ;

 name
----------------------
 Diane MK Woodbridge
```

Cassandra

Add additional columns.

```
ALTER TABLE table_name ADD column_name type;
```

```
cqlsh:mydb> ALTER TABLE friend ADD id uuid;
```

Cassandra

Update data.

```
UPDATE table_name SET column_name = value WEHRE constraint;
```

```
cqlsh:mydb> UPDATE friend SET id = uuid() WHERE name = 'Diane MK Woodbridge';
cqlsh:mydb> SELECT * From friend ;

 name                 | id
----------------------+--------------------------------------
 Diane MK Woodbridge | b784eba6-f125-4a92-ab2d-bf173f5c38ae


cqlsh:mydb> SELECT name, id, writetime(id) From friend ;

 name                 | id                                   | writetime(id)
----------------------+--------------------------------------+------------------
 Diane MK Woodbridge | b784eba6-f125-4a92-ab2d-bf173f5c38ae | 1479438120964274

(1 rows)
```

writetime(column_name) : timstamp of data write.
Not working on the primary key.

Cassandra

Update data.

```
UPDATE table_name SET column_name = value WEHRE constraint;
```

set

```
lsh:mydb> ALTER TABLE friend ADD emails set<text>;
cqlsh:mydb> UPDATE friend SET emails = emails + {'dwoodbridge@gmail.com'} WHERE name = 'Diane MK Woodbridge';
cqlsh:mydb> select * from friend;

 name                | emails                       | id
---------------------+------------------------------+--------------------------------------
 Diane MK Woodbridge | {'dwoodbridge@gmail.com'} | b784eba6-f125-4a92-ab2d-bf173f5c38ae

(1 rows)
cqlsh:mydb> UPDATE friend SET emails = emails + {'dwoodbridge@usfca.edu'} WHERE name = 'Diane MK Woodbridge';
cqlsh:mydb> select * from friend;

 name                | emails                                                  | id
---------------------+---------------------------------------------------------+--------------------------------------
 Diane MK Woodbridge | {'dwoodbridge@gmail.com', 'dwoodbridge@usfca.edu'} | b784eba6-f125-4a92-ab2d-bf173f5c38ae

(1 rows)
```

Cassandra

Update data.

```
UPDATE table_name SET column_name = value WEHRE constraint;
```

list

```
ydb> ALTER TABLE friend ADD phone_numbers list<text>;
cqlsh:mydb> UPDATE friend SET phone_numbers = phone_numbers + {'123-456-7890'} WHERE name = 'Diane MK Woodbridge';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Invalid set literal for phone_numbers of type list<text>"
cqlsh:mydb> UPDATE friend SET phone_numbers = phone_numbers + ['123-456-7890'] WHERE name = 'Diane MK Woodbridge';
cqlsh:mydb> select * from friend;

 name                | emails                                                | id                                   | phone_numbers
---------------------+-------------------------------------------------------+--------------------------------------+-----------------
 Diane MK Woodbridge | {'dwoodbridge@gmail.com', 'dwoodbridge@usfca.edu'} | b784eba6-f125-4a92-ab2d-bf173f5c38ae | ['123-456-7890']

(1 rows)
cqlsh:mydb> UPDATE friend SET phone_numbers = phone_numbers + ['111-222-3333'] WHERE name = 'Diane MK Woodbridge';
cqlsh:mydb> select * from friend;

 name                | emails                                                | id                                   | phone_numbers
---------------------+-------------------------------------------------------+--------------------------------------+-----------------------------------
 Diane MK Woodbridge | {'dwoodbridge@gmail.com', 'dwoodbridge@usfca.edu'} | b784eba6-f125-4a92-ab2d-bf173f5c38ae | ['123-456-7890', '111-222-3333']

(1 rows)
```

Cassandra

Update data.

```
UPDATE table_name SET column_name = value WEHRE constraint;
```

map

```
cqlsh:mydb> ALTER TABLE friend ADD kids map<text,int>;
cqlsh:mydb> UPDATE friend SET kids = {'Abigail':6, 'Jackson':2} WHERE name = 'Diane MK Woodbridge';
cqlsh:mydb> SELECT * FROM friend;

 name                | emails                                                | id                                   | kids                          | phone_number
---------------------+-------------------------------------------------------+--------------------------------------+-------------------------------+--------------
 Diane MK Woodbridge | {'dwoodbridge@gmail.com', 'dwoodbridge@usfca.edu'} | b784eba6-f125-4a92-ab2d-bf173f5c38ae | {'Abigail': 6, 'Jackson': 2} | ['123-456-78
```

Cassandra

Update data.

```
UPDATE table_name SET column_name = value WEHRE constraint;
```

User Defined Type

```
cqlsh:mydb> UPDATE friend SET address = {street:'101 Howard St', city : 'San Francisco', state : 'CA'} WHERE name = 'Diane MK Woodbridge';
cqlsh:mydb> select * from friend ;

 name                | address                                                                | emails                                                | id
       | kids                         | phone_numbers
---------------------+------------------------------------------------------------------------+-------------------------------------------------------+----------------------------------
-------+------------------------------+----------------------------------
 Diane MK Woodbridge | {street: '101 Howard St', city: 'San Francisco', state: 'CA', zip: null} | {'dwoodbridge@gmail.com', 'dwoodbridge@usfca.edu'} | b784eba6-f125-4a92-ab2d-bf173f
5c38ae | {'Abigail': 6, 'Jackson': 2} | ['123-456-7890', '111-222-3333']

(1 rows)
```

Casandra

Selecting the data which is not a primary key.

- Query is not allowed.
- Assign a secondary index on a column that is not part of the primary key.
  - Each node maintains its own copy of a secondary index based on the data.
  - Secondary indexes are not recommended.
  - ➔ Denormalized tables or materialized views are preferred.

Queries involving a secondary index typically involve more nodes, making them more expensive.

```
cqlsh:mydb> select * from friend WHERE emails CONTAINS 'dwoodbridge@usfca.edu';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may
u want to execute this query despite the performance unpredictability, use ALLOW FILTERING"
cqlsh:mydb> select * from friend WHERE emails CONTAINS 'dwoodbridge@usfca.edu' ALLOW FILTERING;

 name                | address                                                                                | emails
      | kids                          | phone_numbers
---------------------+----------------------------------------------------------------------------------------+-----------------------------------------
-------+-------------------------------+-----------------------------------
 Diane MK Woodbridge | {street: '101 Howard St', city: 'San Francisco', state: 'CA', zip: null} | {'dwoodbridge@gmail.com', 'dwoodbridge@usfca
5c38ae | {'Abigail': 6, 'Jackson': 2} | ['123-456-7890', '111-222-3333']

(1 rows)
```

Casandra

Selecting the data which is not a primary key.

- Query is not allowed.
- Assign a secondary index on a column that is not part of the primary key.
  - Each node maintains its own copy of a secondary index based on the data.
  - Secondary indexes are not recommended.
  - ➔ Denormalized tables or materialized views are preferred.

```
cqlsh:mydb> CREATE INDEX ON friend (emails);
cqlsh:mydb> select * from friend WHERE emails CONTAINS 'dwoodbridge@usfca.edu';

 name                | address                                                                                | emails
      | kids                          | phone_numbers
---------------------+----------------------------------------------------------------------------------------+----------------------
------+-------------------------------+--------------------------------------
 Diane MK Woodbridge | {street: '101 Howard St', city: 'San Francisco', state: 'CA', zip: null} | {'dwoodbridge@gmail.com
5c38ae | {'Abigail': 6, 'Jackson': 2} | ['123-456-7890', '111-222-3333']

(1 rows)
```

Additional Readings : Using a SSTable Attached Secondary Index (SASI)
https://docs.datastax.com/en/cql/3.3/cql/cql_using/useSASIIndex.html

Cassandra

RDBMS VS. Cassandra

- No Joins : Have to work on the client side or create denormailzed second table/materilalized view (preferred).
  - Materialized view :

```
CREATE MATERIALIZED VIEW view_name
AS SELECT  column_names FROM table_name WHERE constraints
PRIMARY KEY (primary_keys_in_table_name_and_additional);
```

- No referential integrity across tables : no primary-foreign key relationships.
- Query-first design : Model the queries first and let the data be organized around them.
  - Logical design and then physical design (data type, etc.)

Materialized Views

http://www.datastax.com/dev/blog/new-in-cassandra-3-0-materialized-views

https://docs.datastax.com/en/cql/3.3/cql/cql_using/useCreateMV.html

References

Chodorow, Kristina. *MongoDB: the definitive guide*. O'Reilly Media, Inc., 2013.

MongoDB. *MongoDB Documentation*, https://docs.mongodb.com/, 2016.

Sadalage, Pramod J., and Martin Fowler. *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education, 2012.

Apache Cassandra Documentation v4.0, http://cassandra.apache.org/doc/latest/, 2016.

Carpenter, Jeff, and Eben Hewitt. *Cassandra: The Definitive Guide*. " O'Reilly Media, Inc.", 2016.