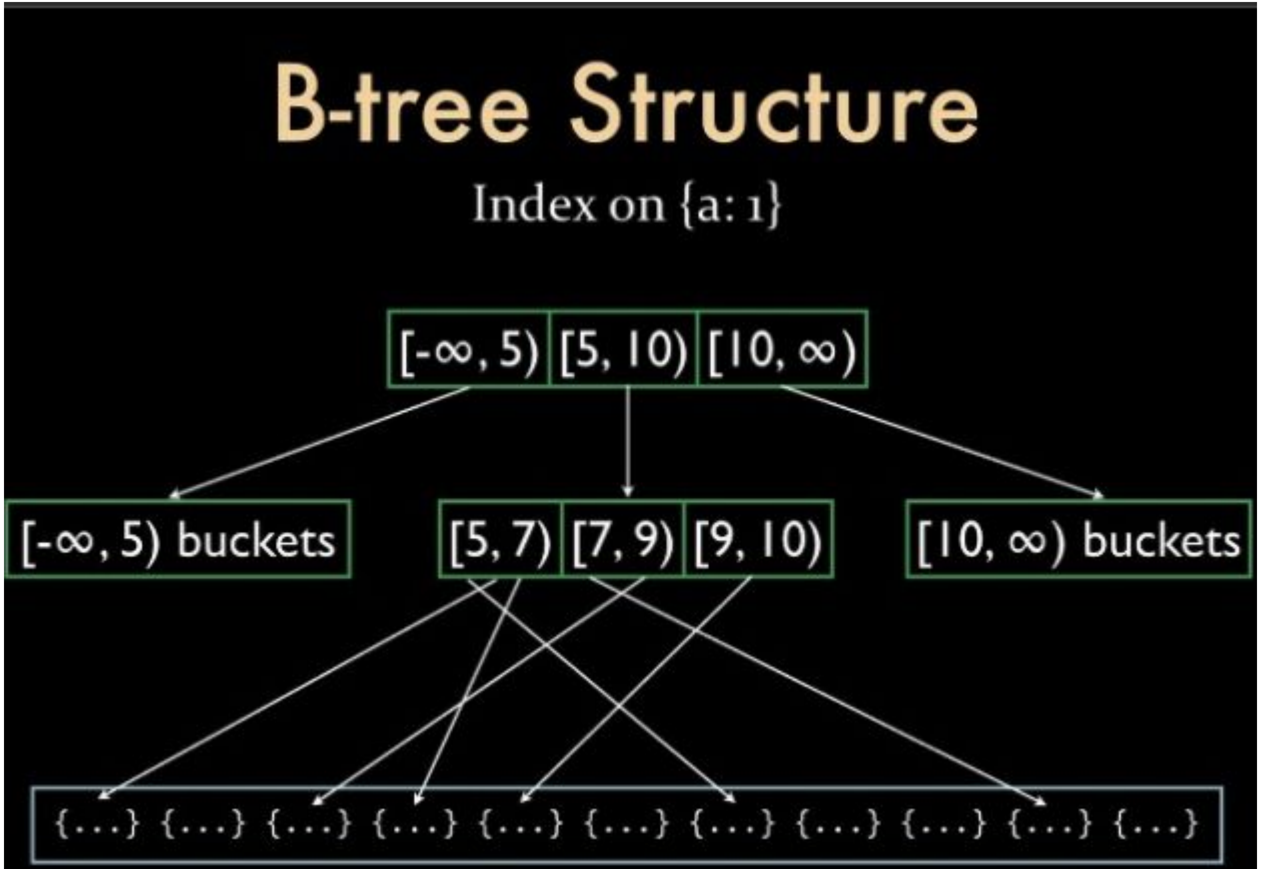
Dev-MongoDB Best Practices About Index

Kuririn Wade

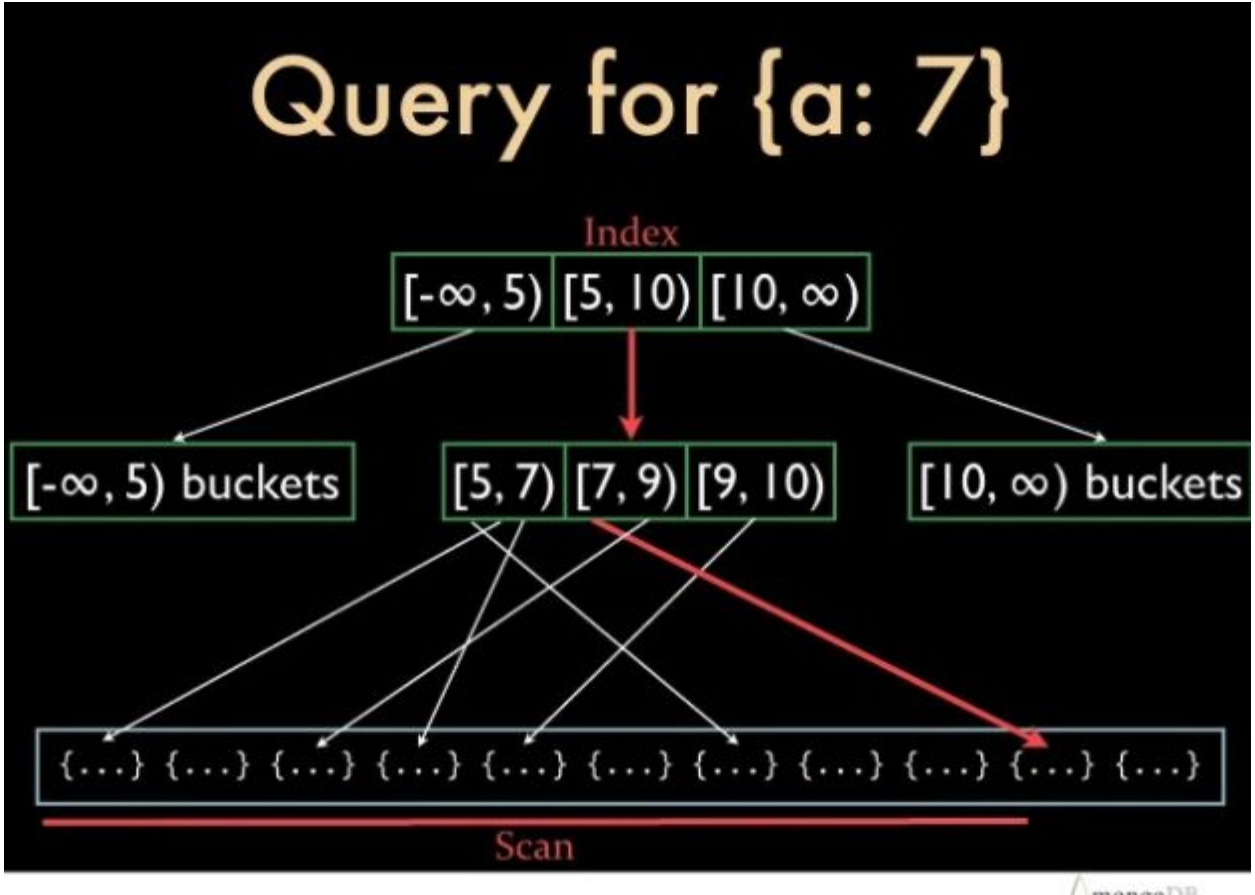
索引是



B-tree Structure

Index on {a: 1}

索引是



Query for {a: 7}

Index

[-∞, 5) [5, 10) [10, ∞)

[-∞, 5) buckets

[5, 7) [7, 9) [9, 10)

[10, ∞) buckets

{...} {...} {...} {...} {...} {...} {...} {...} {...} {...} {...}

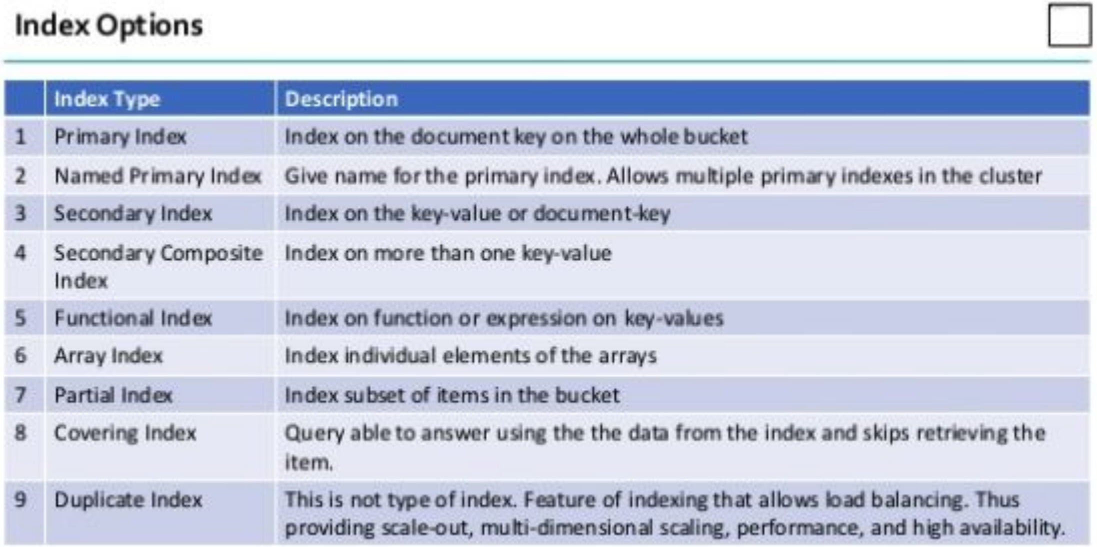
Scan

类别

- Index
- Unique Index
- Sub-document index
- Compound Index
- ...

类别

- Index
- Unique Index
- Sub-document index
- Compound Index
- ...



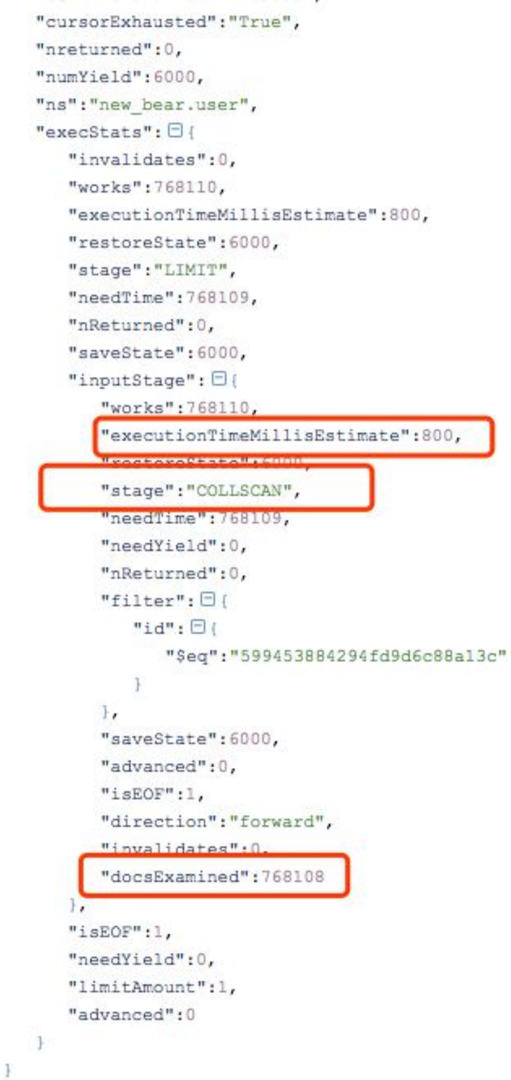
Index Options

	Index Type	Description
1	Primary Index	Index on the document key on the whole bucket
2	Named Primary Index	Give name for the primary index. Allows multiple primary indexes in the cluster
3	Secondary Index	Index on the key-value or document-key
4	Secondary Composite Index	Index on more than one key-value
5	Functional Index	Index on function or expression on key-values
6	Array Index	Index individual elements of the arrays
7	Partial Index	Index subset of items in the bucket
8	Covering Index	Query able to answer using the the data from the index and skips retrieving the item.
9	Duplicate Index	This is not type of index. Feature of indexing that allows load balancing. Thus providing scale-out, multi-dimensional scaling, performance, and high availability.

类别：Compound Index

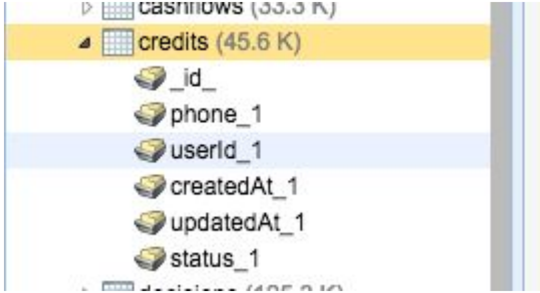
- Difference on Compound index order {userId:1,date:1} vs {date:1,userId:1}
- Difference on Sort order {userId:-1} vs {userId:1}
- Should I create two separate indexes or compound index or both? -- It depends

典型的慢查询



```
  "cursorExhausted":"True",
  "nreturned":0,
  "numYield":6000,
  "ns":"new_bear.user",
  "execStats":⊟{
    "invalidates":0,
    "works":768110,
    "executionTimeMillisEstimate":800,
    "restoreState":6000,
    "stage":"LIMIT",
    "needTime":768109,
    "nReturned":0,
    "saveState":6000,
    "inputStage":⊟{
      "works":768110,
      "executionTimeMillisEstimate":800,
      "restoreState":6000,
      "stage":"COLLSCAN",
      "needTime":768109,
      "needYield":0,
      "nReturned":0,
      "filter":⊟{
        "id":⊟{
          "$eq":"599453884294fd9d6c88a13c"
        }
      },
      "saveState":6000,
      "advanced":0,
      "isEOF":1,
      "direction":"forward",
      "invalidates":0,
      "docsExamined":768108
    },
    "isEOF":1,
    "needYield":0,
    "limitAmount":1,
    "advanced":0
  }
}
```

典型的慢查询

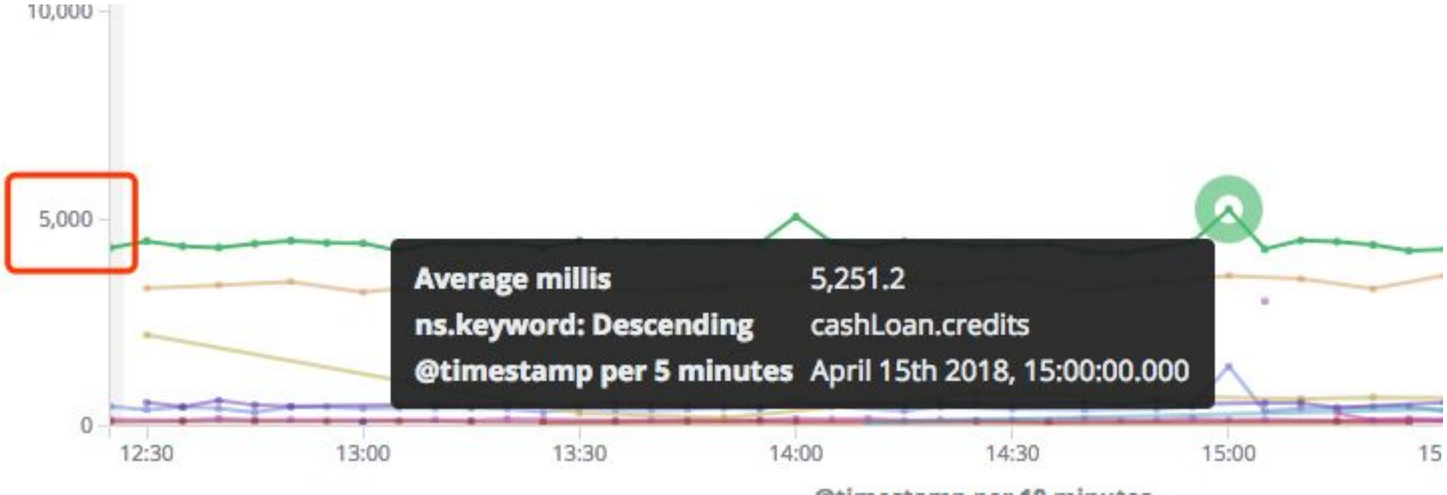


cashflows (33.3 K)

credits (45.6 K)

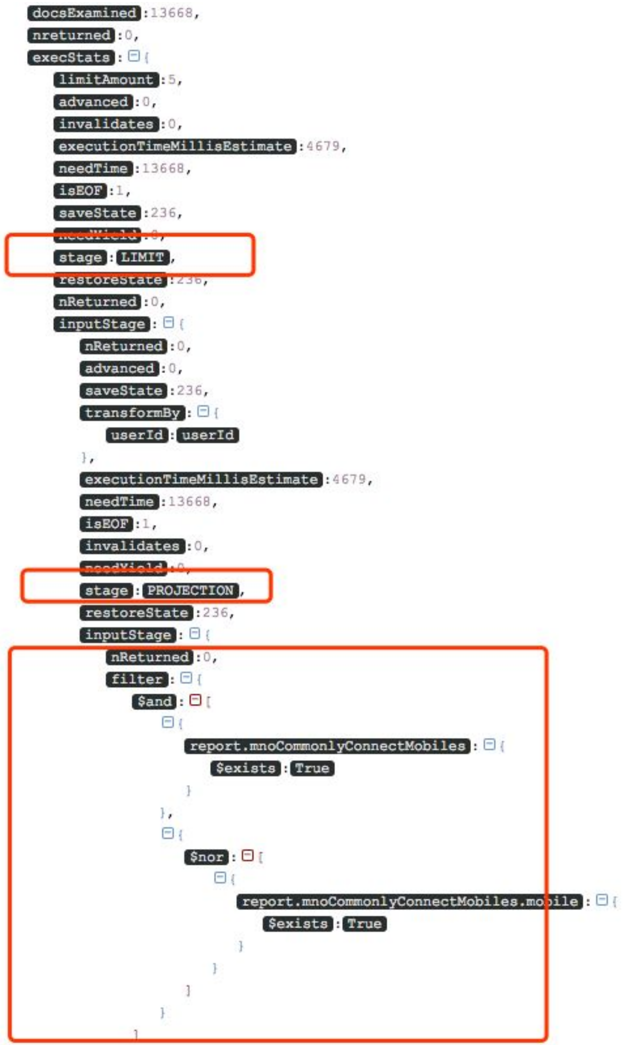
- _id_
- phone_1
- userId_1
- createdAt_1
- updatedAt_1
- status_1

decisions (125.2 K)

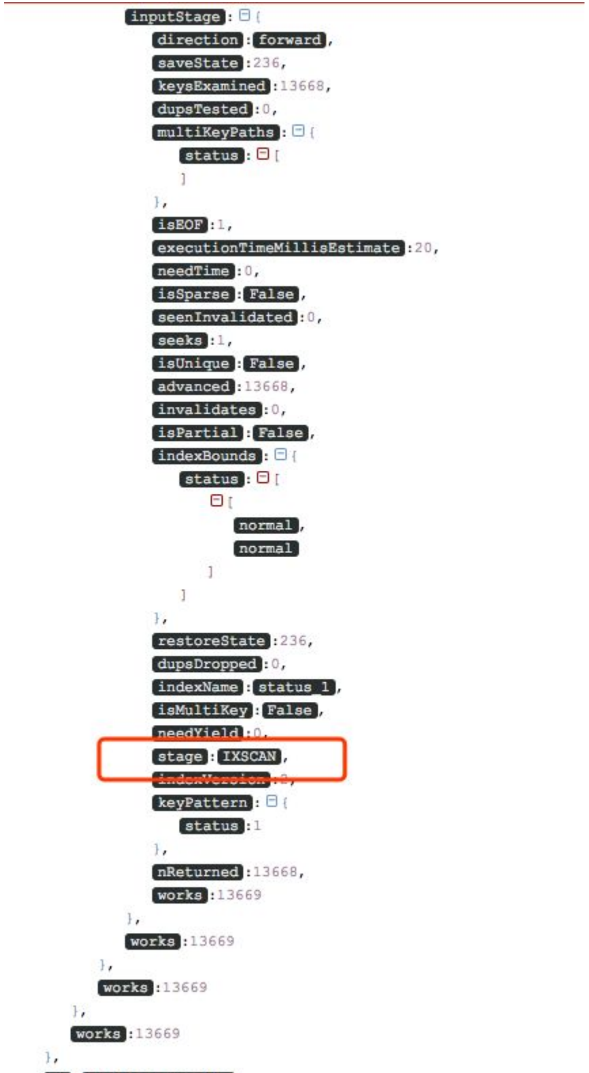


Average millis	5,251.2
ns.keyword: Descending	cashLoan.credits
@timestamp per 5 minutes	April 15th 2018, 15:00:00.000

典型的慢查询



```
docsExamined:13668,
nreturned:0,
execStats: ⊟{
    limitAmount:5,
    advanced:0,
    invalidates:0,
    executionTimeMillisEstimate:4679,
    needTime:13668,
    isEOF:1,
    saveState:236,
    needYield:0,
    stage: LIMIT,
    restoreState:236,
    nReturned:0,
    inputStage: ⊟{
        nReturned:0,
        advanced:0,
        saveState:236,
        transformBy: ⊟{
            userId: userId
        },
        executionTimeMillisEstimate:4679,
        needTime:13668,
        isEOF:1,
        invalidates:0,
        needYield:0,
        stage: PROJECTION,
        restoreState:236,
        inputStage: ⊟{
            nReturned:0,
            filter: ⊟{
                $and: ⊟[
                    ⊟{
                        report.mnoCommonlyConnectMobiles: ⊟{
                            $exists: True
                        }
                    },
                    ⊟{
                        $nor: ⊟[
                            ⊟{
                                report.mnoCommonlyConnectMobiles.mobile: ⊟{
                                    $exists: True
                                }
                            }
                        ]
                    }
                ]
```



```
        inputStage: ⊟{
            direction: forward,
            saveState:236,
            keysExamined:13668,
            dupsTested:0,
            multiKeyPaths: ⊟{
                status: ⊟[
                ]
            },
            isEOF:1,
            executionTimeMillisEstimate:20,
            needTime:0,
            isSparse: False,
            seenInvalidated:0,
            seeks:1,
            isUnique: False,
            advanced:13668,
            invalidates:0,
            isPartial: False,
            indexBounds: ⊟{
                status: ⊟[
                    ⊟[
                        normal,
                        normal
                    ]
                ]
            },
            restoreState:236,
            dupsDropped:0,
            indexName: status_1,
            isMultiKey: False,
            needYield:0,
            stage: IXSCAN,
            indexVersion:2,
            keyPattern: ⊟{
                status:1
            },
            nReturned:13668,
            works:13669
        },
        works:13669
    },
    works:13669
},
works:13669
},
```

典型的慢查询

Usually building index on status can actually HARM on performance

理想的查询

- IXSCAN
- Limit
- Projection
- Docs Size less is better
- Docs Examined less is better
- nReturned=totalDocsExamined=totalKeysExamined
- SORT in index
- Limit Array elements

理想的查询

对于普通查询，我们最希望看到的组合有这些：

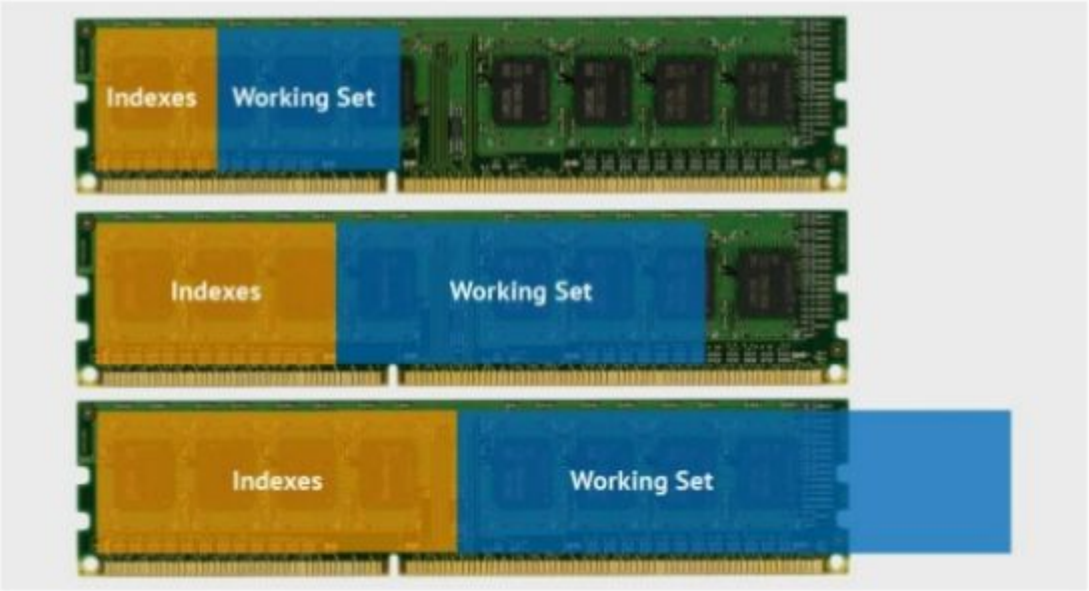
- Fetch+IDHACK
- Fetch+ixscan
- Limit+（Fetch+ixscan）
- PROJECTION+ixscan
- SHARDING_FILTER+ixscan
- Sort + Index

理想的查询

- SSD + Large RAM
- Consider sharding when data size > RAM

理想的查询

Ensure indexes fit in RAM



理想的查询

不希望看到包含如下的stage：

- COLLSCAN（全表扫），SORT（使用sort但是无index），不合理的SKIP，SUBPLA（未用到index的$or）

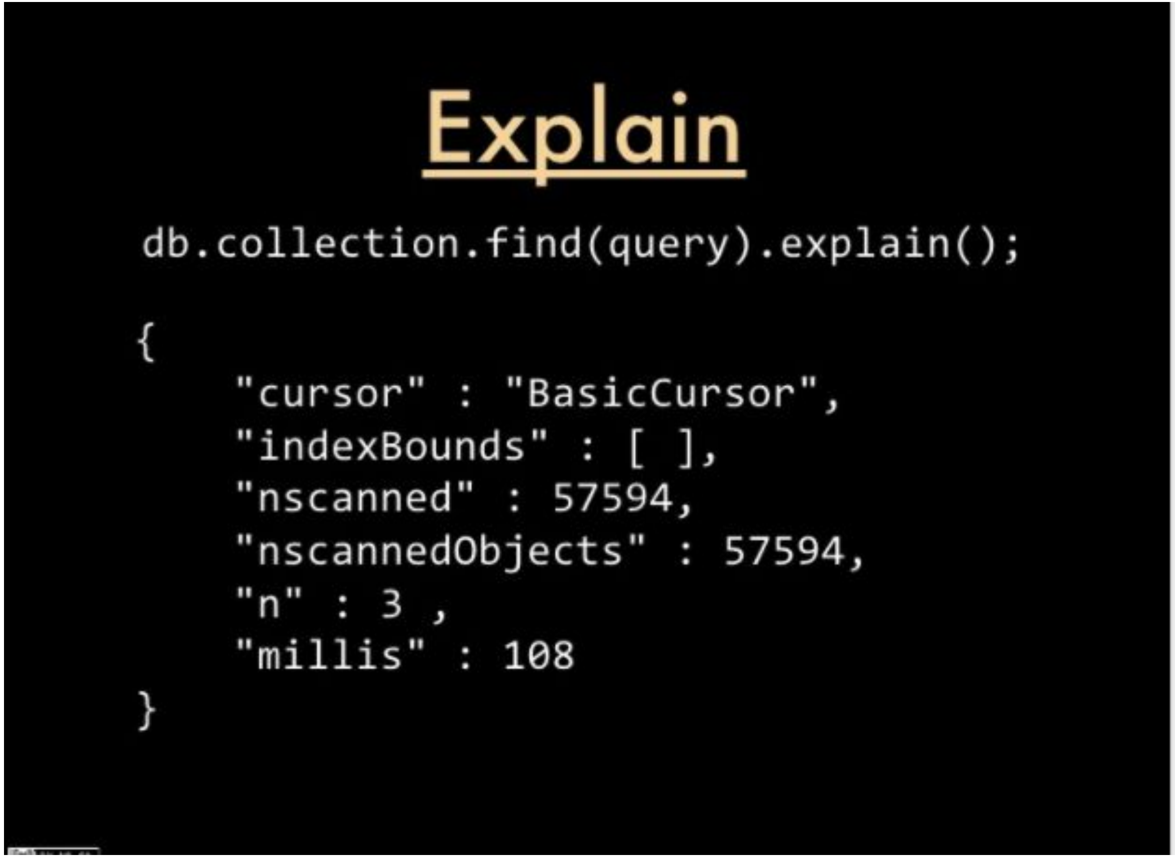
对于count查询，希望看到的有：

- COUNT_SCAN

不希望看到的有:

- COUNTSCAN

Explain() 指标

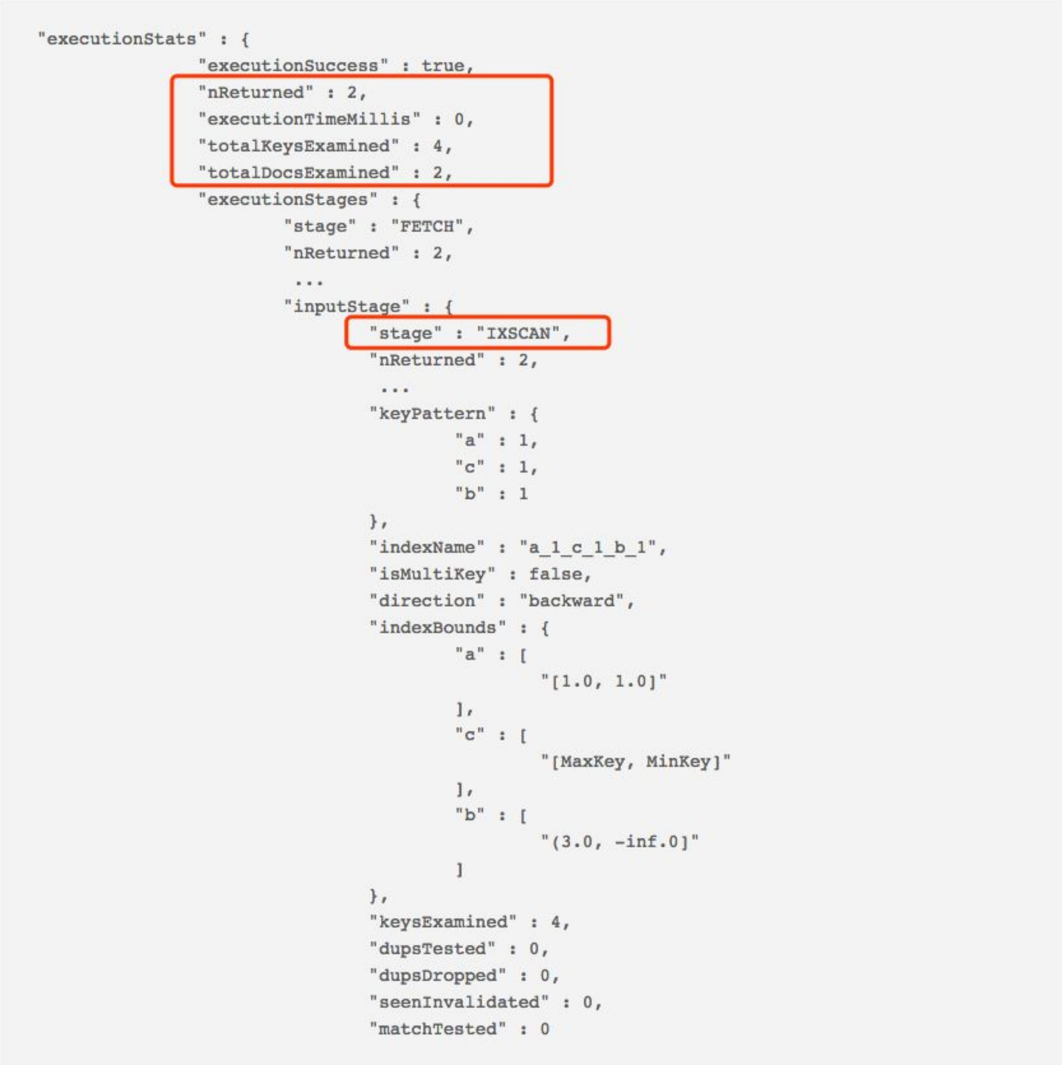


Explain

```
db.collection.find(query).explain();

{
    "cursor" : "BasicCursor",
    "indexBounds" : [ ],
    "nscanned" : 57594,
    "nscannedObjects" : 57594,
    "n" : 3 ,
    "millis" : 108
}
```

Explain() 指标



```
"executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 2,
                "executionTimeMillis" : 0,
                "totalKeysExamined" : 4,
                "totalDocsExamined" : 2,
                "executionStages" : {
                        "stage" : "FETCH",
                        "nReturned" : 2,
                         ...
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "nReturned" : 2,
                                 ...
                                "keyPattern" : {
                                        "a" : 1,
                                        "c" : 1,
                                        "b" : 1
                                },
                                "indexName" : "a_1_c_1_b_1",
                                "isMultiKey" : false,
                                "direction" : "backward",
                                "indexBounds" : {
                                        "a" : [
                                                "[1.0, 1.0]"
                                        ],
                                        "c" : [
                                                "[MaxKey, MinKey]"
                                        ],
                                        "b" : [
                                                "(3.0, -inf.0]"
                                        ]
                                },
                                "keysExamined" : 4,
                                "dupsTested" : 0,
                                "dupsDropped" : 0,
                                "seenInvalidated" : 0,
                                "matchTested" : 0
```

Explain() 指标

more explain

指标

More in cloud manager index

So, 越多索引越好？

- According to Use frequency
- Index Consume space
- Update index after new data insert
- Consume time during restore data

坑

- Foreground Build Index (default) will block operation
- Upsert won’t duplicate ONLY when in unique index
- Avoid array length too large (performance & 16M Documents size & Agg Limit)

What If

- Q：我觉得我的DB优化到头了，降不下来query 时间？

What If

Answer:

- 我觉得
- Cache
- Hardware(Mem,Sharding,...,Enterprise Version)
- 历史数据Snapshot

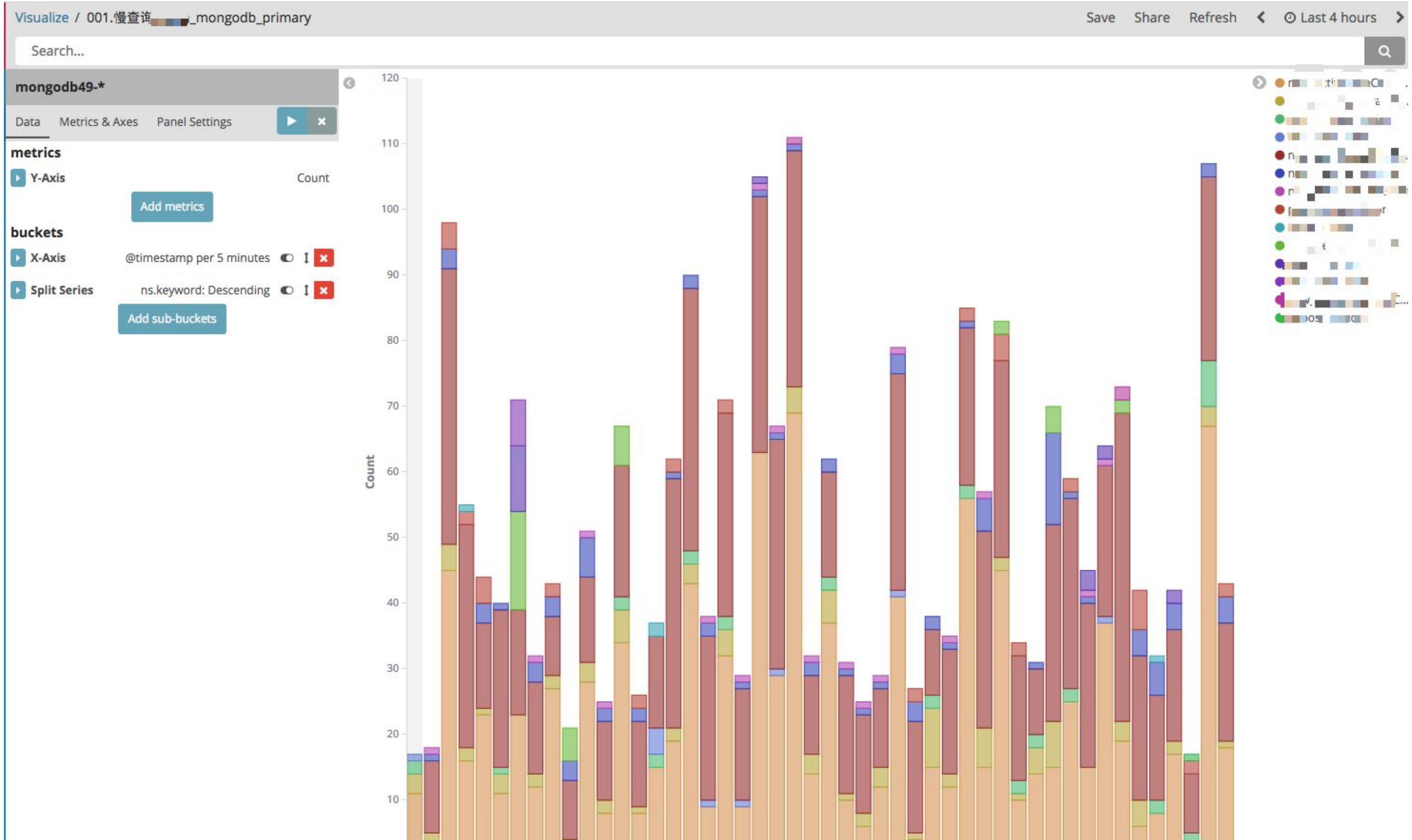
经验

- count() > 10K must create index
- Query should < 10ms
- Index should < 100 ms

无数据不优化

```
db.setProfilingLevel(1,100)
```

无数据不优化



Visualize / 001.慢查询[redacted]_mongodb_primary

mongodb49-*

metrics

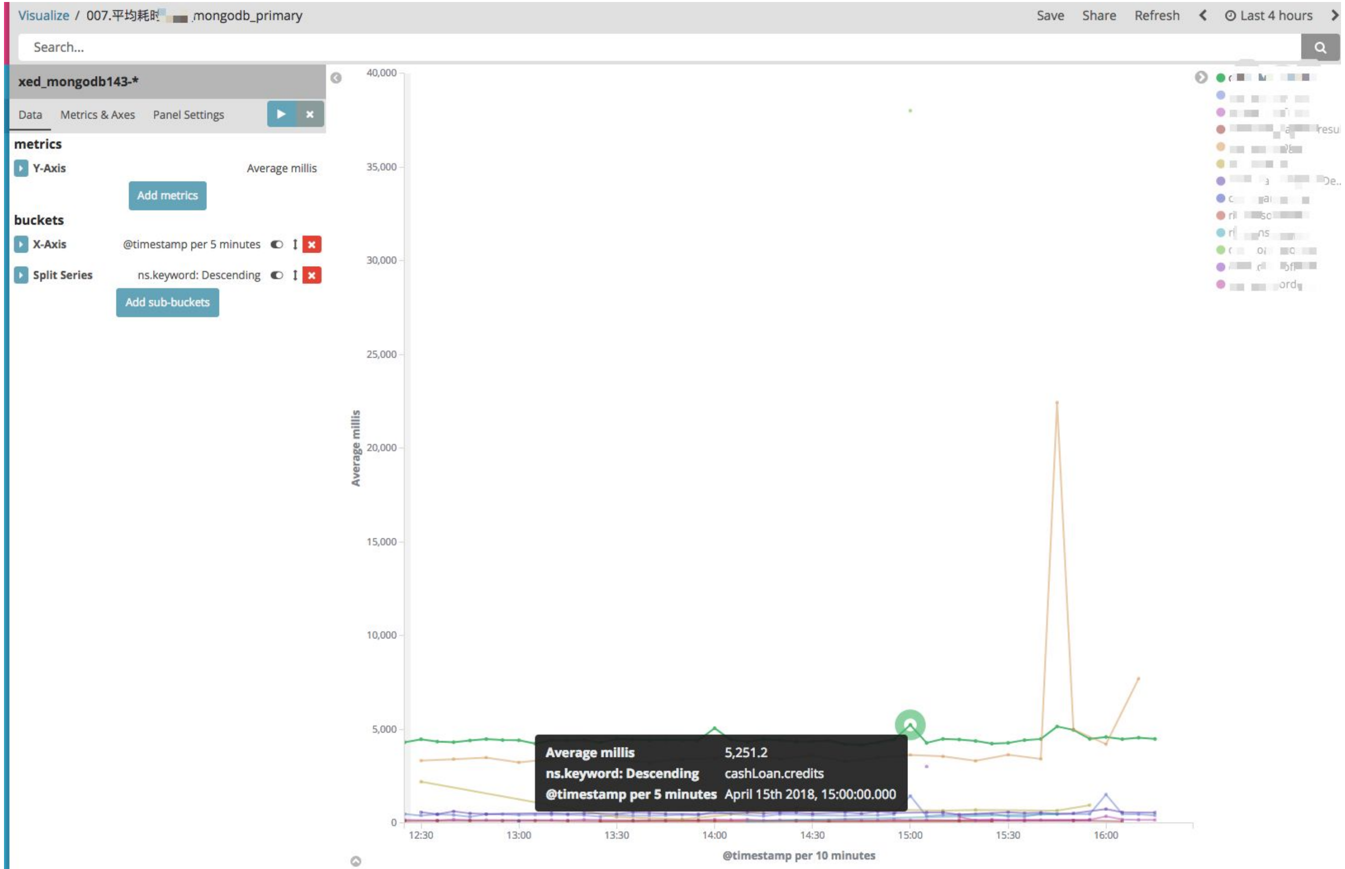
Y-Axis Count

buckets

X-Axis @timestamp per 5 minutes

Split Series ns.keyword: Descending

无数据不优化



Visualize / 007.平均耗时[redacted]mongodb_primary

Save Share Refresh ‹ Last 4 hours ›

Search...

xed_mongodb143-*

Data Metrics & Axes Panel Settings

metrics

Y-Axis Average millis

Add metrics

buckets

X-Axis @timestamp per 5 minutes

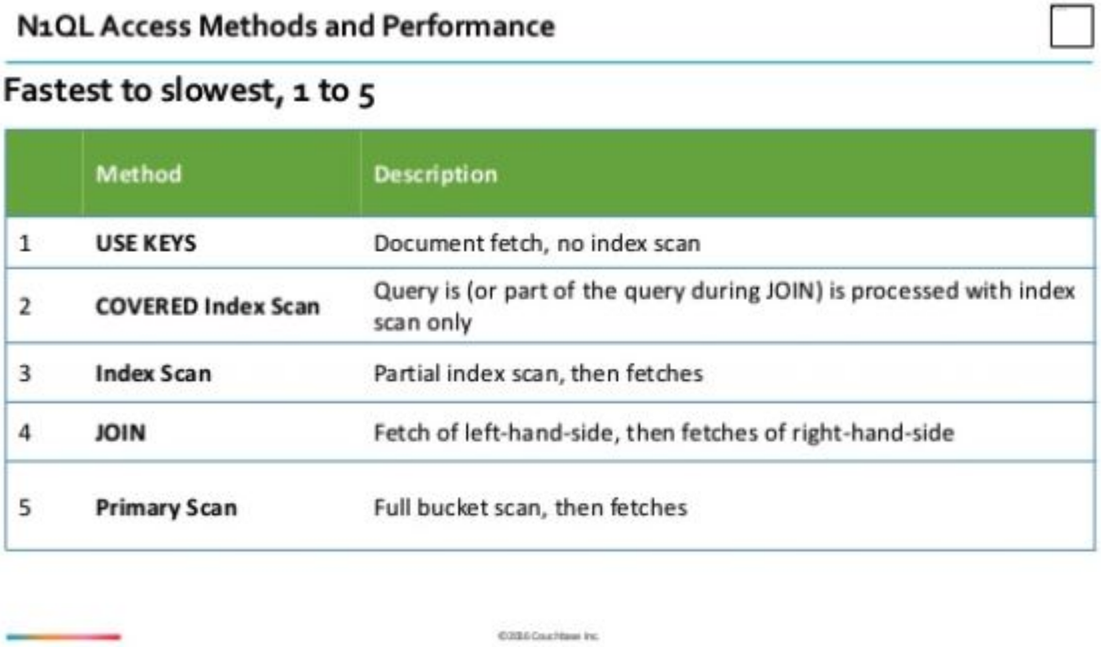
Split Series ns.keyword: Descending

Add sub-buckets

Average millis 5,251.2
ns.keyword: Descending cashLoan.credits
@timestamp per 5 minutes April 15th 2018, 15:00:00.000

[redacted]

More Tuning(not only MongoDB)



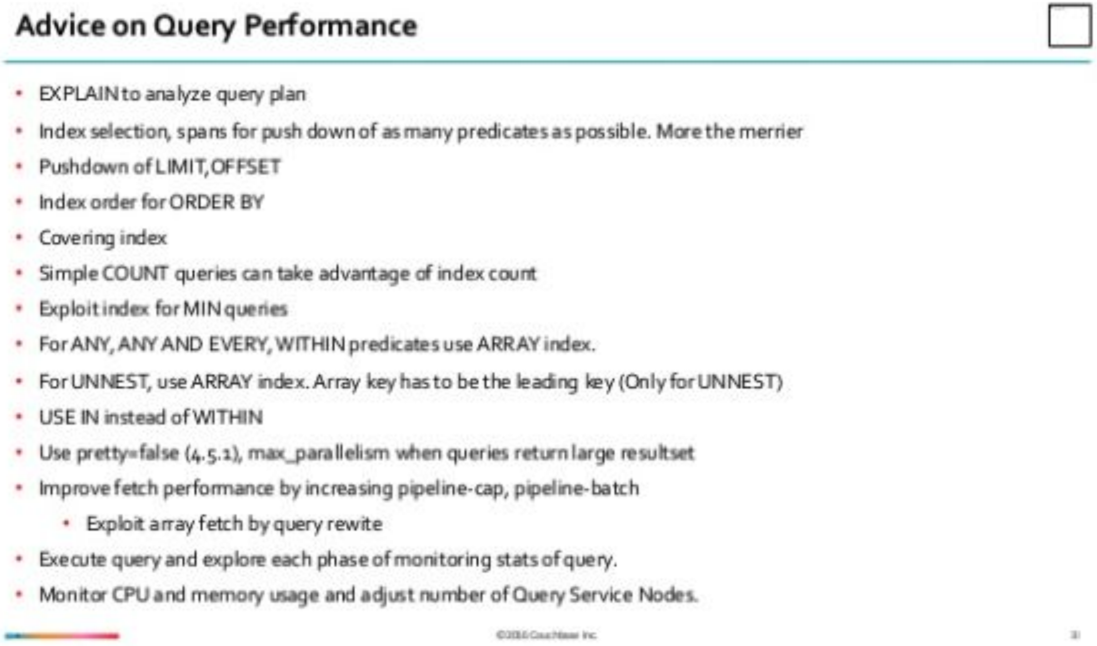
N1QL Access Methods and Performance

Fastest to slowest, 1 to 5

	Method	Description
1	**USE KEYS**	Document fetch, no index scan
2	**COVERED Index Scan**	Query is (or part of the query during JOIN) is processed with index scan only
3	**Index Scan**	Partial index scan, then fetches
4	**JOIN**	Fetch of left-hand-side, then fetches of right-hand-side
5	**Primary Scan**	Full bucket scan, then fetches

©2016 Couchbase Inc.

More Tuning(not only MongoDB)



Advice on Query Performance

- EXPLAIN to analyze query plan
- Index selection, spans for push down of as many predicates as possible. More the merrier
- Pushdown of LIMIT, OFFSET
- Index order for ORDER BY
- Covering index
- Simple COUNT queries can take advantage of index count
- Exploit index for MIN queries
- For ANY, ANY AND EVERY, WITHIN predicates use ARRAY index.
- For UNNEST, use ARRAY index. Array key has to be the leading key (Only for UNNEST)
- USE IN instead of WITHIN
- Use pretty=false (4.5.1), max_parallelism when queries return large resultset
- Improve fetch performance by increasing pipeline-cap, pipeline-batch
  - Exploit array fetch by query rewite
- Execute query and explore each phase of monitoring stats of query.
- Monitor CPU and memory usage and adjust number of Query Service Nodes.

©2016 Couchbase Inc.

More Tuning

https://www.slideshare.net/mongodb/fast-querying-indexing-for-performance-4 #54

https://www.slideshare.net/journalofinformix/tuning-for-performance-indexes-queries

https://www.slideshare.net/mongodb/indexing-with-mongodb

https://www.slideshare.net/mongodb/indexing-and-query-optimizer-richard-kreuter

https://www.slideshare.net/mdirolf/indexing