MongoDB 优化器 & 执行器介绍

樊智辉(一拴)

主题

- 慢查询的产生 & 发现
- 索引的工作原理
- 优化器的工作原理 & 人为干预
- MultiKey 索引的影响 & 注意事项

慢查询的产生

1	{name: “abc123.com”, users: 100 , location: “江苏” }
2	{ name: “b107.com”, users: 120 , location: “江苏” }
3	{ name: “c.com”, users: 83 , location: “上海” }
	…
100,001	{ name: “a.com”, users: 100 , location: “浙江” }
100,002	{ name: “abc1250.com”, users: 100 , location: “浙江” }
100,003	{ name: “abc126.com”, users: 100 , location: “浙江” }
100,004	{ name: “abc127.com”, users: 100 , location: “浙江” }
	…
100,000,000	{ name: ”thebillion.com”, users: 100 , location: “浙江” }


```
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 1,
        "executionTimeMillis" : 3388,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 1000000,
        "executionStages" : {
                "stage" : "COLLSCAN",
                "filter" : {
                        "name" : {
                                "$eq" : "abc127.com"
                        }
                },
                "nReturned" : 1,
                "executionTimeMillisEstimate" : 3019,
                "works" : 1000002,
                "advanced" : 1,
                "needTime" : 1000000,
                "needYield" : 0,
                "saveState" : 7898,
                "restoreState" : 7898,
                "isEOF" : 1,
                "invalidates" : 0,
                "direction" : "forward",
                "docsExamined" : 1000000
        },
        "allPlansExecution" : [ ]
},
```


- db.sites.find({name:“abc127.com"}).**explain(true)**

慢查询通常是因为读了过多的文档数产生的

慢查询的设置

- db.setProfileLevel(level，options)

level	0 - 不会记录任何查询到 system.profile (默认值) 1 - 只记录慢查询 2 - 记录所有的查询 注： 任何时候慢查询都会记录在mongod.log
options	slowms - 定义慢查询的临界值，默认100ms sampleRate - sample 的百分比，默认为1.

https://docs.mongodb.com/manual/reference/method/db.setProfilingLevel/

慢查询的查看

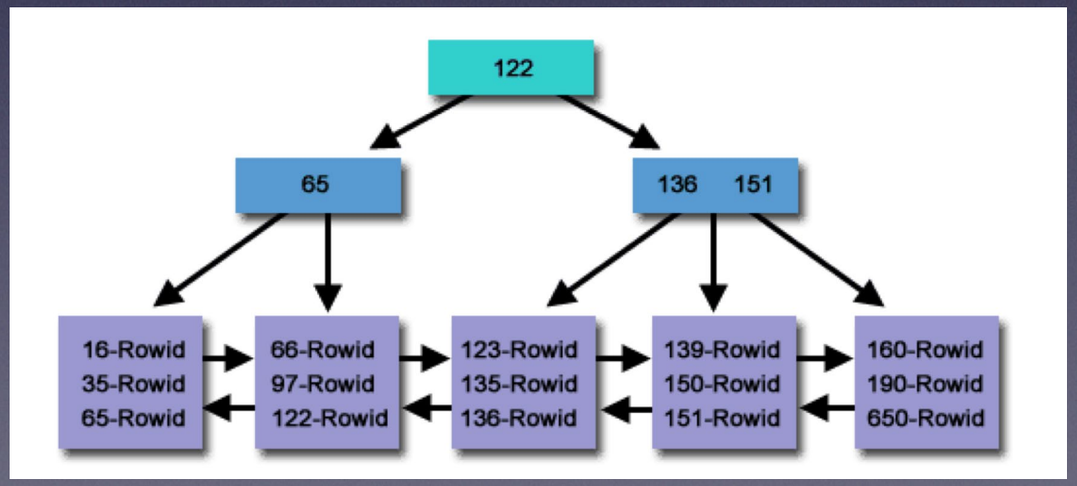
```
2019-04-15T16:06:48.394+0800 I COMMAND  [conn5]
command: find
{ find: "tab",
  batchSize: 10.0,
  filter: { a: { $gt: 10.0, $lt: 100000.0 } },
  sort: { _id: -1.0 }..}
planSummary: IXSCAN { _id: 1 }
keysExamined:900010
docsExamined:900010
numYields:7218
nreturned:10
reslen:457
locks:{ .. }
protocol:op_msg 9724ms
```

```
{
        "op" : "query",
        "ns" : "test.tab",
        "command" : {
                "find" : "tab",
                "batchSize" : 10,
                "filter" : {
                        "a" : {
                                "$gt" : 10,
                                "$lt" : 100000
                        }
                },
                "sort" : {
                        "_id" : -1
                },
                "$db" : "test"
        },
        "cursorid" : 30238491490,
        "keysExamined" : 900010,
        "docsExamined" : 900010,
        "numYield" : 7218,
        "nreturned" : 10,
        "locks" : {
                "Global" : {
                        "acquireCount" : {
                                "r" : NumberLong(7219)
                        }
                },
                "Database" : {
                        "acquireCount" : {
```

索引的工作原理

- db.sites.createIndex({name:1})

a.com	100,001
abc123.com	1
abc1250.com	100,002
abc126.com	100,003
abc127.com	100,004

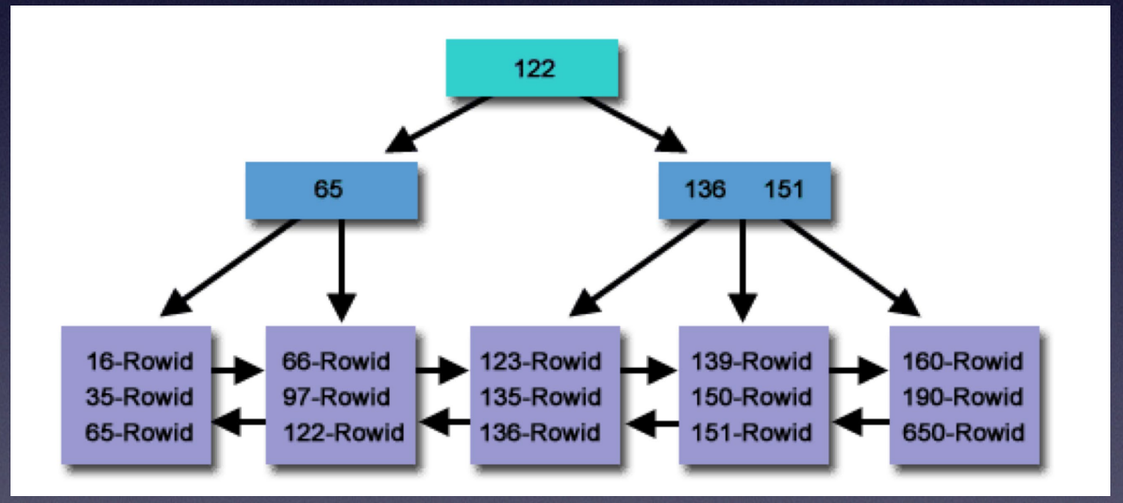


- db.sites.find({name:“abc127.com"})

```
"executionSuccess" : true,
"nReturned" : 1,
"executionTimeMillis" : 3,
"totalKeysExamined" : 1,
"totalDocsExamined" : 1,
"executionStages" : {
        "stage" : "FETCH",
        "nReturned" : 1,
        "executionTimeMillisEstimate" : 0,
        "works" : 2,
        "advanced" : 1,
        "needTime" : 0,
        "needYield" : 0,
        "saveState" : 0,
        "restoreState" : 0,
        "isEOF" : 1,
        "invalidates" : 0,
        "docsExamined" : 1,
        "alreadyHasObj" : 0,
        "inputStage" : {
                "stage" : "IXSCAN",
                "nReturned" : 1,
                "executionTimeMillisEstimate" : 0,
                "works" : 2,
                "advanced" : 1,
                "needTime" : 0,
                "needYield" : 0,
```

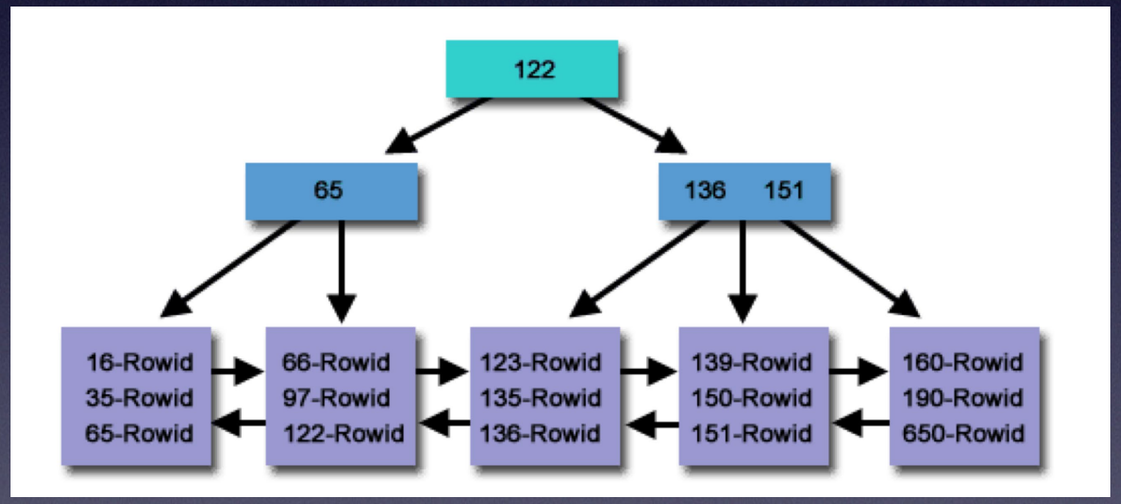
索引的特点

- 快速检索一个key 或者一段key
- 因为索引有序，可以避免排列

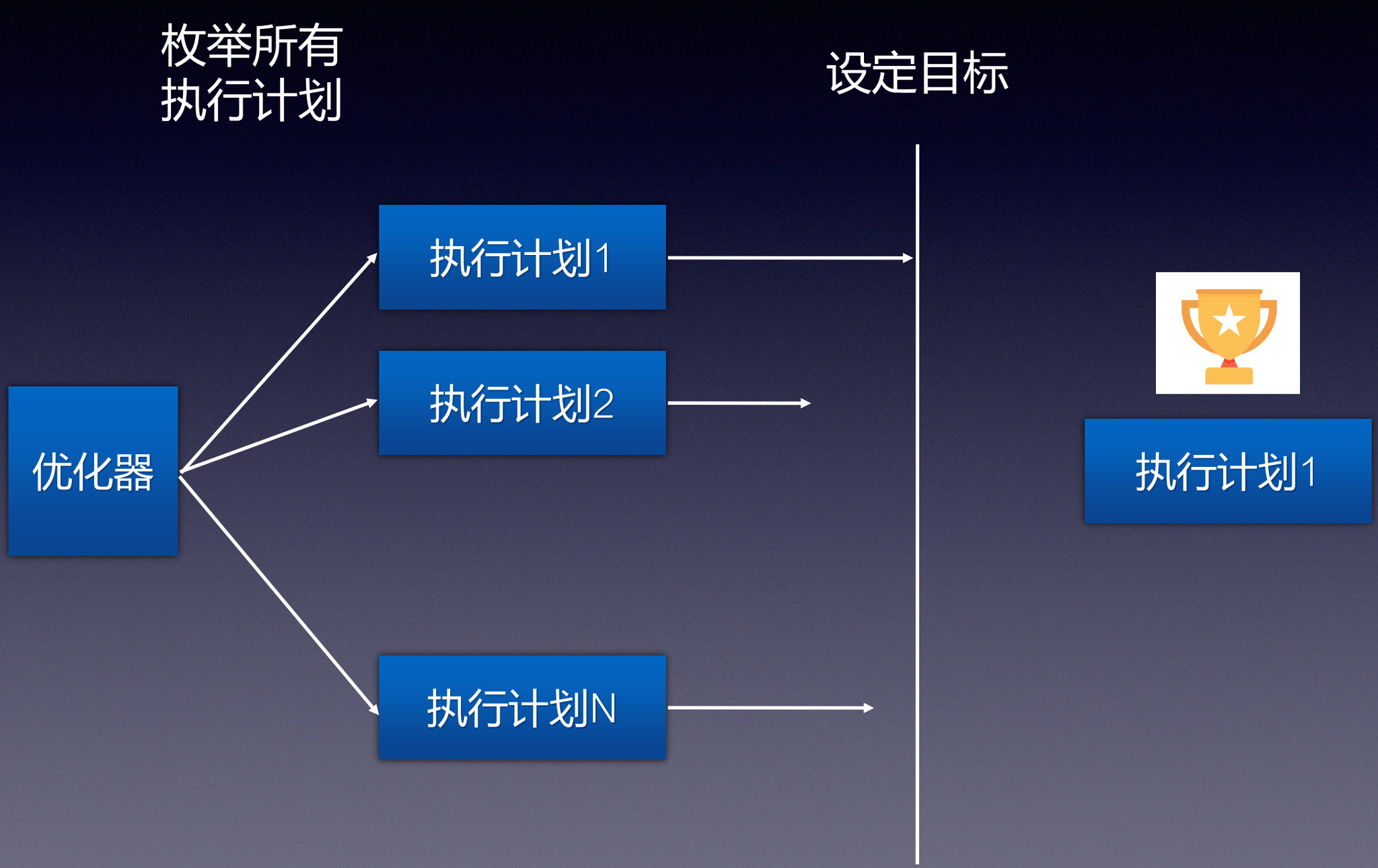


讨论

- 2 个索引： {a : 1}, {b: 1}
- 查询:
  - filter: {a: {$gt: 1, $lt: 3}}
  - sort: {b: 1}
  - limit: 1
- 有多少种方法来取到这一条数据？ 哪一种是最好的?

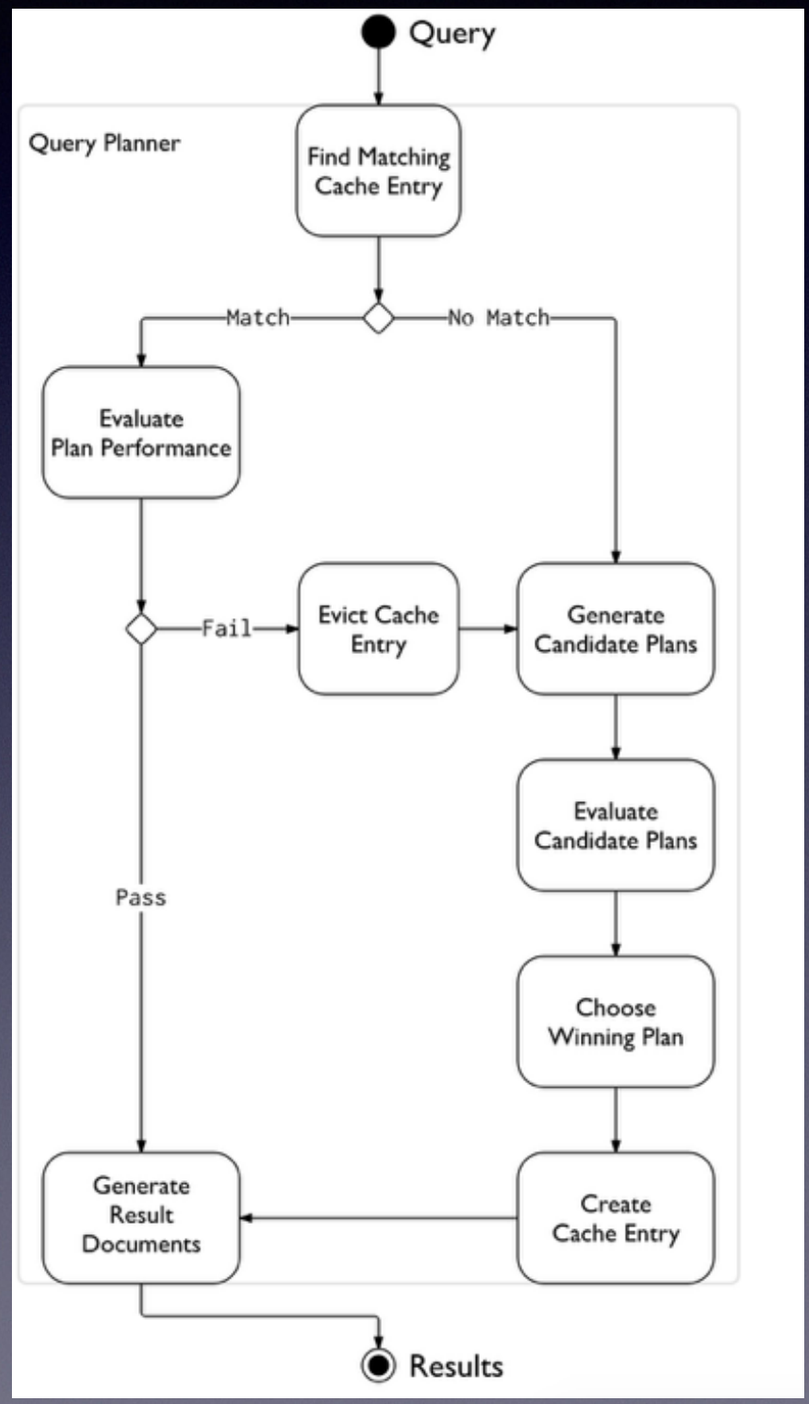


优化器的工作原理



执行计划的重用

- MongoDB 缓存执行计划
- MongoDB 会在特定场景使得 cache 失效
  - 创建/删除索引
  - 数据变化太多
  - 人为flush 等等



思考

什么时候产生的执行计划是不合理的?

影响优化器做决定

客户端干预

```
db.tab.find({a: {…}})
.sort({b: 1})
.hint({a: 1})
```

服务端干预

```
db.runCommand(
    {
        planCacheSetFilter: “tab”,
        query: {a: {$gt: 10, $lt: 100}},
        sort: {b: 1},
        indexes:  [ { a: 1} ]
})
```

MongoDB 数组运算

- db.employee.insert(
  {
  …
  hobbies： [“music”, “reading”, “sport”]
  })

- db.employee.find({**hobbies**: “music”})

- db.employee.find({**hobbies**: “music”, hobbies: “reading”})

MongoDB 数组操作

- {…. , scores: [30, 40] },
  {…. , scores: [10, 60] }

- db.tab.find({ scores: {$gt: 20, $lt: 50} })

- 2条结果都返回

- db.tab.find({scores: {**$elemMatch**: {$gt: 20, $lt: 50}})

MongoDB 数组操作

- {…. , tags: [{name: “A”, value: “B”},
  {name: “C”, value: “D”}]
  }

- db.tab.find({ tags.name: “A”, tags.value: “D"} })

- 上述记录满足条件

- db.tab.find({ tags: {**$elemMatch**: {name: “A”, value: “D”}}})

MongoDB Multi Key 索引

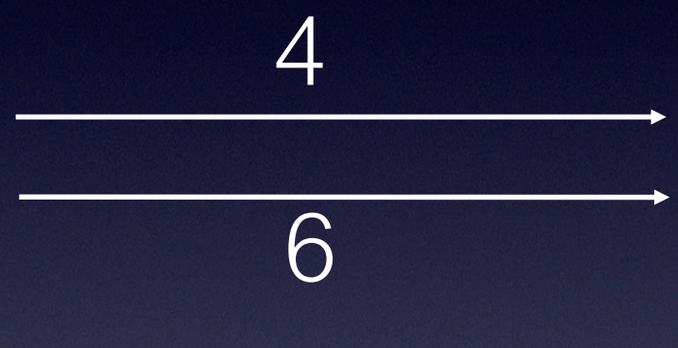
- db.employee.createIndex({**hobbies: 1}**)

- db.tab1.createIndex({scores: 1})

- db.tab2.createIndex({tags.name: 1})

Multi Key Index 结构

```
db.tab2.insert({
   scores: [4, 6]
})
```



Key	行号
0	10
3	6
5	7

```
db.tab2.find({
   scores: {$gt: 3, $lt: 4}
})
```

Key	行号
0	10
3	6
4	**20**
5	7
6	**20**

Multi Key Index 结构

```
db.tab2.insert({
   scores: [4, 6]
})
```

4

6

Key	行号
1	10
3	6
5	7

```
db.tab2.find({
   scores: {$gt: 2, $lt: 4}
})
```

Key	行号
1	10
3	6
4	20
5	7
6	20

讨论

```
{
    "appId" : "Gf93VvCfOdttrxSOemt_03ff",
    "deviceId" : "bbc-lmc-03991933",
    "nodeType" : "FACTORY",
    "creationTime" : ISODate("2019-03-01T10:11:39.852Z"),
    "lastModifiedTime" : ISODate("2019-03-03T10:45:40.249Z"),
    "tags" : [
        {
            "tagName" : "pipeline",
            "tagValue" : "multi",
            "tagType" : 1
        }
    ],
    ...
}
```

```
//过滤条件
{
  appId: "Gf93VvCfOdttrxSOemt_03ff",
  tags.tagName: "pipeline",
  tags.tagValue: "multi",
  _id: { $gt: ObjectId('000000000000000000000000')
}

//排序
sort: { _id: 1 }

//限制条数
limit: 10
```

```
{ appId: 1, tags.tagName: 1, tags.tagValue: 1, _id: 1 }
```

http://mongoing.com/archives/24814

Multi Key Index

一定要考虑 $elemMatch!

Multi Key Index 内存占用

db.tab2.find({scores: {$gt： 3}})

如何保证行20只被返回一次?

Key	行号
0	10
3	6
4	**20**
5	7
6	**20**

```
set<RecordId>  returned;   // RecordId 大小为8 字节
…
kv = _indexCursor->next();
if (isMutliKeyIndex) {
   if (!returned.insert(kv).second) {
      break;
   }
}
```

Recall

- 慢查询的产生 & 发现
- 索引的工作原理
- 优化器的工作原理 & 人为干预
- MultiKey 索引的影响 & 注意事项

QA

找出应用中的慢查询，尝试优化它们吧

😄😄

谢谢！