索引扫描（ixscan）/集合or表扫描（collscan）

1.索引扫描:只要需要扫描非常小的数据，索引页。
2.集合or表扫描：对所有的数据集扫描，一页一页翻开。

selectivity--过滤性

在一个有10000条记录的集合中， 1.满足gender=f的记录有4000条。 2.满足city=LA的记录有100条。 3.满足ln="parker"的记录有10条。 总结：ln能过滤掉最多的数据，city其次，gender最弱，所以ln的过滤性（selectivity）大于city大于gender 如果：需要建立一个索引，那么给哪个字段建立索引更好？ln字段，只需要load 10条数据在内存中 问题： 如果要查询同时满足，genderf&&citySZ&&ln=="parker"的记录，只允许为gender/city/ln中的一个建立索引，应该吧索引放在哪里，肯定用过滤性最强的，这样扫描的文档只有10条，内存中只有10条数据

hash索引

1.hash索引理论上的查询时间复杂度是0（1）
但是这么高效为什么不使用hash索引?
1.hash计算是随机的，如果在磁盘上放置数据，查询会非常慢
2.无法对范围查询进行优化
3.无法利用前缀索引
4.排序也无法优化
5.必须回行，查询的数据必须去数据页面中查询

索引分类

1._id索引: _id 索引是绝大多数集合默认建立的索引。对于每一个插入的数据。MongoDB会自己主动生成一条唯一的 _id 字段。
2.单键索引：单键索引是最普通的索引，与_id 索引不同。单键索引不会自己主动创建
3.多键索引：多键索引和单键索引创建形式同样，差别在于字段的值。单键索引：值为一个单一的值，比如字符串，数字或者日期。多键索引：值具有多个记录，比如数组。
4.复合索引（重点）：当我们的查询条件不仅仅有一个时。就须要建立复合索引。
5.过期索引：过期索引是在一段时间后会过期的索引。在索引过期后，对应的数据会被删除。这适合存储一些在一段时间之后会失效的数据。比方用户的登陆信息、存储的日志等
使用限制
    1. 存储在过期索引字段的值必须是指定的时间类型。（必须是ISODate或者ISODate数组，不能使用时间戳。否则不能被自己主动删除）
    2. 假设指定了ISODate数组。则依照最小的时间进行删除。
    3. 过期索引不能是复合索引。
    4. 删除时间不是精确的。（删除过程是由后台程序没60s跑一次，并且删除也须要一些时间，所以存在误差）
6.全文索引：$meta操作符：{score：{$meta:"textScore"}}，返回结果的类似度，与 sort 一起使用能够达到非常好的有用效果。
7.地理位置索引：将一些点的位置存储在 MongoDB 中。创建索引后，能够依照位置来查找其它点。
8.hash索引

index prefix（前缀索引，按照顺序匹配）

```
db.human.createIndex({firstName:1,lastName:1,gender:1,age:1})
以上索引的全部前缀包括
    1.{firstName:1}
    2.{firstName:1,lastName:1}
    3.{firstName:1,lastName:1,gender:1}
    所有索引的前缀都可以被该索引覆盖，没有必要针对这些索引再建额外的索引了。
```

索引执行计划

展示流程图

explain（）

```
1.db.W00000000109_mk_qw_user.find({"name":"朱启铭-研
发"}).projection({"name":1}).explain(true)
 "executionStats" : {
    "nReturned" : 1,（重点关注：返回数量），返回值和执行时间的比列还是很重要
的
    "executionTimeMillis" : 4,(重点关注：执行时间)
    "totalKeysExamined" : 1,（重点关注：扫描多少个key）
    "totalDocsExamined" : 0,（重点关注：扫描多少个文档）
    "executionStages" : {
        "stage" : "SINGLE_SHARD",
        "nReturned" : 1,
        "executionTimeMillis" : 4,
        "totalKeysExamined" : 1,
        "totalDocsExamined" : 0,
        "totalChildMillis" : 0,
        "shards" : [
            {
                "shardName": "rs0",
                "executionSuccess": true,
                "executionStages": {
                    "stage": "COLLSCAN",（重点关注）
                    "filter": {
                        "createTime": {
                            "$eq": "2020-09-10 21:16:34"
                        }
                    },
最理想情况是totalKeysExamined/nReturned=1
```

复合/联合索引（重点）

索引满足ESR原则（按照这个顺序建立索引）

Equal 等值查询的字段放最前面
Sort 条件放中间，也就是**sort**字段放中间，（原理：利用索引来排序，减少数据取出来之后在内存中排序）
Range 匹配的字段放后面,这里的匹配指的是=,>,<等这类匹配
案例分析

```
db.qw_user.find({"name":"启铭","age":
{$gte:18}}).sort("createTime":1)
```

ES和ER原则同ESR原则类似

流程图展示

未加索引的排序字段(进行了全表扫描，没有使用到index，在内存中sort，很显然，和都是不可取的)

```
https://blog.csdn.net/harleylau/article/details/78327334
    "executionStages" : {
        "stage" : "SINGLE_SHARD",
        "nReturned" : 2,
        "executionTimeMillis" : 0,
        "totalKeysExamined" : 0,  (未使用索引)
        "totalDocsExamined" : 2,
        "totalChildMillis" : 0,
        "shards" : [
            {
                "executionSuccess" : true,
                "executionStages" : {
                    "stage" : "SORT", (未使用index的sort)
                    "nReturned" : 2,
                    "executionTimeMillisEstimate" : 0,
                    "sortPattern" : {
                        "createTime" : 1
                    },
                    "memUsage" : 456, (占用的内存)
                    "memLimit" : 33554432, (内存限制，超出后报错)
                        "inputStage" : {
                        "stage" : "SORT_KEY_GENERATOR",
                        "nReturned" : 2,
                        "inputStage" : {
                            "stage" : "COLLSCAN", (全表扫描)
```

覆盖索引（covered query）

1.流程图展示

2.所有的查询字段是索引的一部分
3.所有的查询返回字段在同一个索引中
mongodb不需要从数据页中加载数据，这就是查询覆盖，同MySQL，这种覆盖索引非常高效，怎么确定一个索引是否是覆盖索引？如果 explain() totalDocsExamined 为0，重点，就算你用了projection设置了，一定要过滤掉"_id"，否则这个totalDocsExamined不会为0的，你们可以试试。
example：customerId是索引
db.W00000000018_mk_qw_chatroom_member_intention_statistics.find({"customerId":"wmDolHEAAA8IplP_0by03I9iuPz9TVBA"},{"customerId":1,"_id":0}).explain("allPlansExecution")，执行这条语句可以看到执行计划里面totalDocsExamined为0

部分索引

MongoDB部分索引只为那些在一个集合中，满足指定的筛选条件的文档创建索引。由于部分索引是一个集合文档的一个子集，因此部分索引具有较低的存储需求，并降低了索引创建和维护的性能成本。部分索引通过指定过滤条件来创建，可以为MongoDB支持的所有索引类型使用部分索引。
1.按照创建时间排序：
db.T00000000011_mk_qw_user.find({}).sort({"createTime":-1})
2.创建部分索引：db.W00000000018_mk_qw_user.createIndex({"a":1},{
    partialFilterExpression:{"createTime": {$gt:"2020-09-10 21:16:34"}}
})
3.查看执行计划：db.T00000000011_mk_qw_user.find({"createTime":"2020-10-20 21:43:52"}).explain(true)
可以用于isDelete=false的情况，软删除，或者线索，等等，很多可能都使用
4.用于软删除

针对存在字段建立部分索引

```
db.W00000000018_mk_qw_user.createIndex({"a":1},{
    partialFilterExpression:{"createTime": {$exists:true}}
})
```

针对部分索引的优化文档

1.https://segmentfault.com/q/1010000016620953

项目实战（mk-material-svc）

现场

explain的使用

db.collection.explain().<method(...)>

1.**explain**可以设置参数
    >1.queryPlanner 查询计划
    >2.executionStats 执行状态
    >3.allPlansExecution 所有的计划，这个包含了上面两种
2.db.W00000000018_mk_qw_chatroom_member_intention_statistics.find({
"customerId":"wmDolHEAAA8IplP_0by03I9iuPz9TVBA"},
{"customerId":1,"_id":0}).**explain**("allPlansExecution")
    ``` **json**
    {
        "queryPlanner" : {
            "mongosPlannerVersion" : 1,
            "winningPlan" : {
                "stage" : "SINGLE_SHARD",
                "shards" : [
                    {
                        "shardName" : "rs0",
                        "connectionString" :
"rs0/192.168.0.124:7210,192.168.0.127:7210",
                        "serverInfo" : {
                            "host" : "iZbp1bkar79cnzgeq7dtcrZ",
                            "port" : 7210,
                            "version" : "4.0.10",
                            "gitVersion" :
"c389e7f69f637f7a1ac3cc9fae843b635f20b766"
                        },
                        "plannerVersion" : 1,
                        "namespace" :
"mk_data_20191209.W00000000018_mk_qw_chatroom_member_intention_stat
istics",
                        "indexFilterSet" : **false**,
    ```

```
                        "indexFilterSet" : false,
                        "parsedQuery" : {
                            "customerId" : {
                                "$eq" :
"wmDolHEAAA8IplP_0by03I9iuPz9TVBA"
                            }
                        },
                        "winningPlan" : {
                            "stage" : "PROJECTION",
                            "transformBy" : {
                                "customerId" : 1,
                                "_id" : 0
                            },
                            "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                    "customerId" : 1
                                },
                                "indexName" : "customerId_1",
                                "isMultiKey" : false,
                                "multiKeyPaths" : {
                                    "customerId" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                    "customerId" : [
                                        "
[\"wmDolHEAAA8IplP_0by03I9iuPz9TVBA\",
\"wmDolHEAAA8IplP_0by03I9iuPz9TVBA\"]"
                                    ]
                                }
                            }
                        },
                        "rejectedPlans" : [ ]
                    }
                ]
            }
        },
        "executionStats" : {
            "nReturned" : 29,
            "executionTimeMillis" : 2,
            "totalKeysExamined" : 29,
            "totalDocsExamined" : 0,
            "executionStages" : {
                "stage" : "SINGLE_SHARD",
```

```
                "nReturned" : 29,
                "executionTimeMillis" : 2,
                "totalKeysExamined" : 29,
                "totalDocsExamined" : 0,
                "totalChildMillis" : 1,
                "shards" : [
                    {
                        "shardName" : "rs0",
                        "executionSuccess" : true,
                        "executionStages" : {
                            "stage" : "PROJECTION",
                            "nReturned" : 29,
                            "executionTimeMillisEstimate" : 0,
                            "works" : 30,
                            "advanced" : 29,
                            "needTime" : 0,
                            "needYield" : 0,
                            "saveState" : 0,
                            "restoreState" : 0,
                            "isEOF" : 1,
                            "invalidates" : 0,
                            "transformBy" : {
                                "customerId" : 1,
                                "_id" : 0
                            },
                            "inputStage" : {
                                "stage" : "IXSCAN",
                                "nReturned" : 29,
                                "executionTimeMillisEstimate" : 0,
                                "works" : 30,
                                "advanced" : 29,
                                "needTime" : 0,
                                "needYield" : 0,
                                "saveState" : 0,
                                "restoreState" : 0,
                                "isEOF" : 1,
                                "invalidates" : 0,
                                "keyPattern" : {
                                    "customerId" : 1
                                },
                                "indexName" : "customerId_1",
                                "isMultiKey" : false,
                                "multiKeyPaths" : {
                                    "customerId" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
```

```
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                    "customerId" : [
                                        "
[\"wmDolHEAAA8IplP_0by03I9iuPz9TVBA\",
\"wmDolHEAAA8IplP_0by03I9iuPz9TVBA\"]"
                                    ]
                                },
                                "keysExamined" : 29,
                                "seeks" : 1,
                                "dupsTested" : 0,
                                "dupsDropped" : 0,
                                "seenInvalidated" : 0
                            }
                        }
                    }
                ]
            },
            "allPlansExecution" : [
                {
                    "shardName" : "rs0",
                    "allPlans" : [ ]
                }
            ]
        },
        "ok" : 1,
        "operationTime" : Timestamp(1596881496, 29),
        "$clusterTime" : {
            "clusterTime" : Timestamp(1596881496, 29),
            "signature" : {
                "hash" : BinData(0,"w2gsk5V8bwBZoQ7IO+Qutdp+56Q="),
                "keyId" : 6817369591434444806
            }
        }
    }

    ```

3.上面执行计划有几个重要返回值
    1.>executionStats.executionTimeMillis: query的整体查询时间。
    2.>executionStats.nReturned: 查询返回的条目。
    2.>executionStats.totalKeysExamined: 索引扫描条目。
    3.>executionStats.totalDocsExamined: 文档扫描条目。
        "executionTimeMillis" : 2,执行时间
        "nReturned" : 29,返回的条目数量
        "totalKeysExamined" : 29,如果为0表示没有用到索引
        "totalDocsExamined" : 0,全文扫描，这里用到了索引覆盖，所以不存在到
```
```

```
数据页扫描数据
    4.>理想情况（用到索引覆盖）

    nReturned=totalKeysExamined & totalDocsExamined=0

4.Stage状态分析
    1.>属性
        COLLSCAN 全表扫描
        IXSCAN  扫描索引
        FETCH   根据索引去检索指定document
        SHARD_MERGE 将各个分片返回数据进行merge
        SORT    表明在内存中进行了排序
        LIMIT   使用limit限制返回数
        SKIP    使用skip进行跳过
        IDHACK  针对_id进行查询
        SHARDING_FILTER 通过mongos对分片数据进行查询
        COUNT   利用db.coll.explain().count()之类进行count运算
        COUNTSCAN   count不使用Index进行count时的stage返回
        COUNT_SCAN  count使用了Index进行count时的stage返回
        SUBPLA  未使用到索引的$or查询的stage返回
        TEXT    使用全文索引进行查询时候的stage返回
        PROJECTION  限定返回字段时候stage的返回
    2>.对于普通查询，我希望看到stage的组合(查询的时候尽可能用上索引)：
        Fetch+IDHACK
        Fetch+ixscan
        Limit+（Fetch+ixscan)
        PROJECTION+ixscan
        SHARDING_FITER+ixscan
        COUNT_SCAN
    3>.不希望看到包含如下的stage：
        COLLSCAN(全表扫描)
        SORT(使用sort但是无index),不合理的SKIP,SUBPLA(未用到index的$or)
        COUNTSCAN(不使用index进行count)
```

最后索引创建

```
db.W00000000097_mall_order_log.createIndex({"action":1,"createTime"
:-1,"orderNumber":1},{background:true})
db.W00000000097_mall_order_log.find({"action":2,"orderNumber":
{$lt:3}}).sort({"createTime":-1}).explain(true)
db.W00000000097_mall_order_log.getIndexes()
db.W00000000097_mall_order_log.dropIndexes()
```