MongoDB 索引优化与管理

刘诚杰（天痕）

2017.7

1. 索引概念
2. 索引优化
3. 执行计划
4. 索引管理

1. 索引概念

索引概念

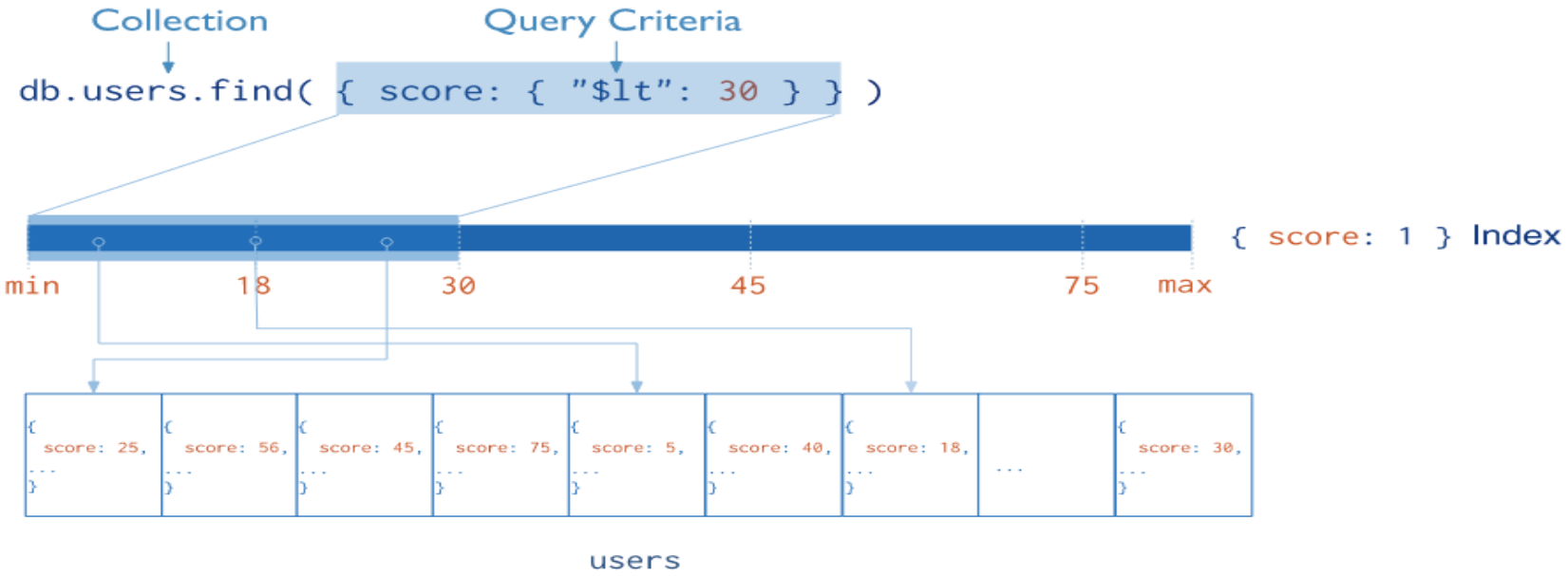
值与位置

汉语拼音音节索引

1. 每一音节后举一字做例，可按例字读音去查同音的字。
2. 数字指本字典正文页码。

好用 废纸

索引概念


Collection
Query Criteria
db.users.find({ score: { "$lt": 30 } })
{ score: 1 } Index
min
18
30
45
75
max
{ score: 25, ... }
{ score: 56, ... }
{ score: 45, ... }
{ score: 75, ... }
{ score: 5, ... }
{ score: 40, ... }
{ score: 18, ... }
...
{ score: 30, ... }
users

索引概念

文档内容	物理位置
user_id:1,score:25	0x0020
user_id:2,score:56	0x0017
user_id:3,score:45	0x0007
user_id:4,score:75	0x0016

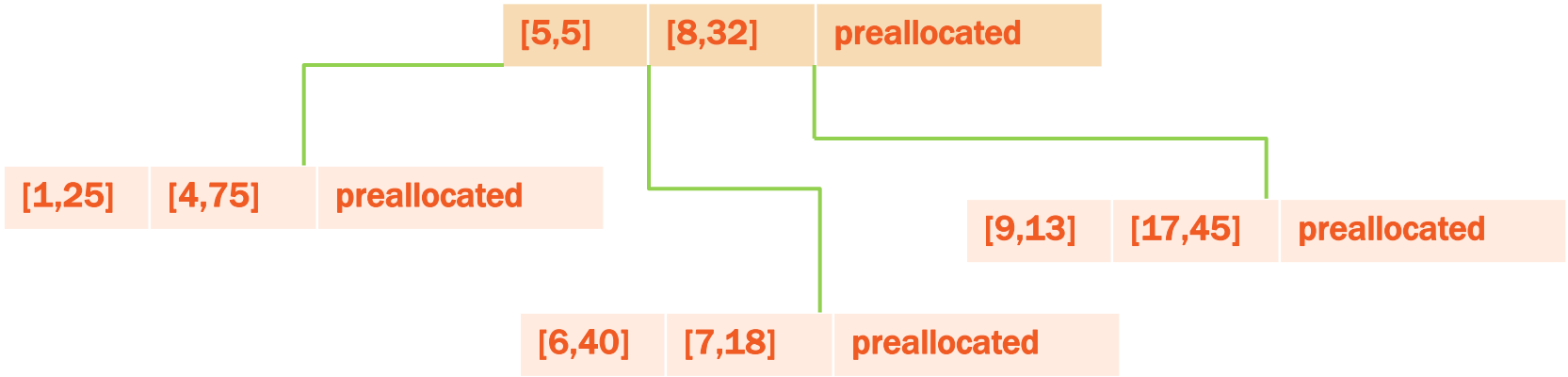
集合信息

user_id和score联合索引

SCORE索引	物理位置
25	0x0020
45	0x0007
56	0x0017
75	0x0016

score单键索引

索引概念



B-Tree

记住结构即可

2.索引优化

索引优化

选择性——重复率低

A 5%数据重复
B 10%数据重复
C 20%数据重复
D 50%数据重复

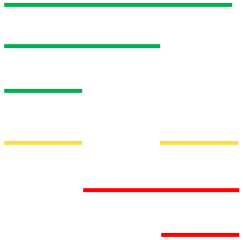
重复率越低越适合做索引

状态、性别等高重复率不适合

联合索引，索引前缀由低到高

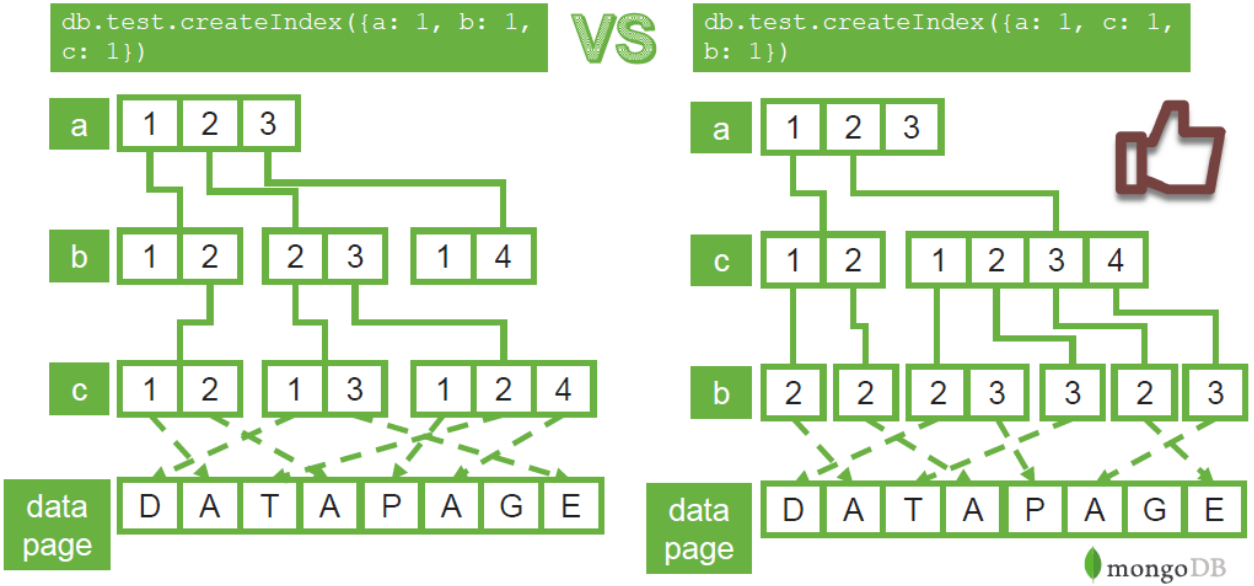
索引前缀

db.test.createIndex({a:1,b:1,c:1})



索引顺序（等值与范围）

db.test.find({a:2,b:{$gte:2,$lte:3},c:1})

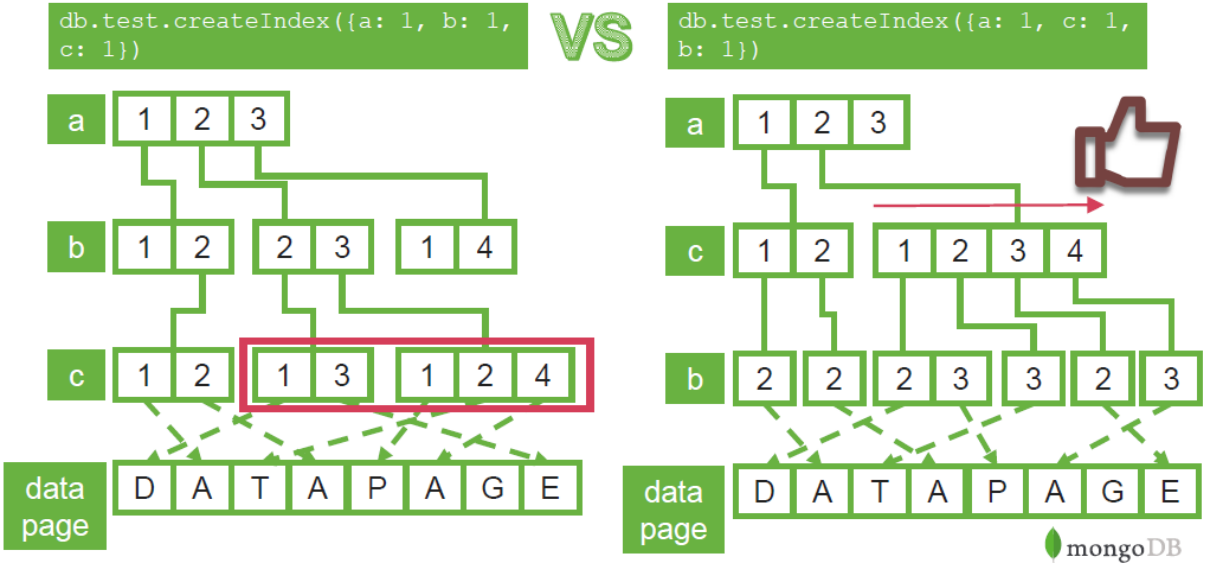


索引优化

Tips:
大表的sort字段要有索引

索引顺序（等值、范围与排序）

```
db.test.find({a:2,b:{$gte:2,$lte:3}}).sort({c:1})
```



覆盖索引

db.test.createIndex({a:1,b:1,c:1})

db.test.find({a:3},{b:1,c:1,_id:0})
select b,c from test where a = 3;

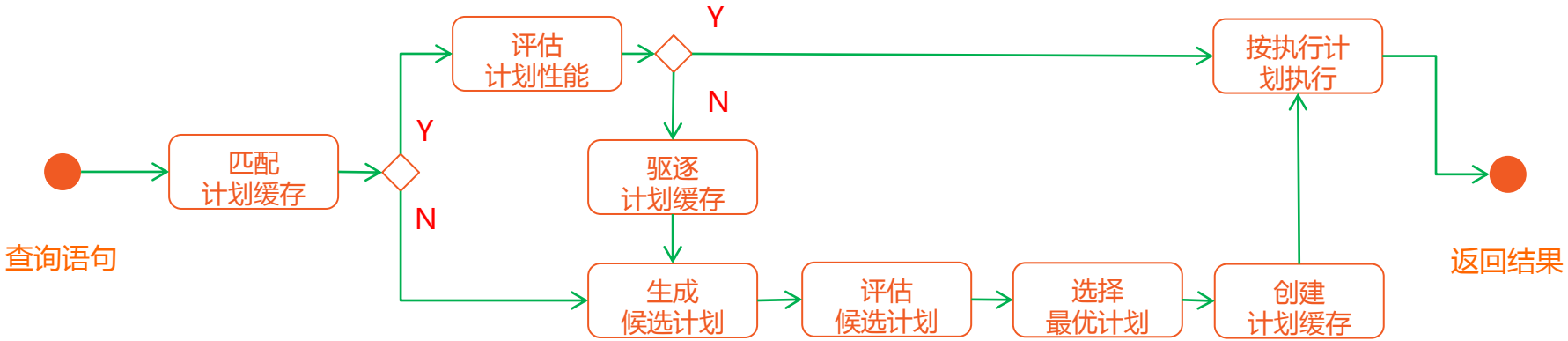
能用到索引，且获取的值在索引中

结论

- 索引字段重复率要低
- 联合索引，索引前缀顺序为，等值-排序-范围
- 有条件尽量匹配覆盖索引

3.执行计划

计划器流程图


查询语句
匹配
计划缓存
Y
N
评估
计划性能
Y
N
驱逐
计划缓存
生成
候选计划
评估
候选计划
选择
最优计划
创建
计划缓存
按执行计
划执行
返回结果

执行计划

简易查询计划说明

db.test.find({条件}).explain(“executionStats”)

executionStages.stage尽量为IXSCAN(索引扫描)
避免COLLSCAN（全表扫描）、SORT（内存排序）

totalKey(Docs)Examined值接近nReturned
executionTimeMilles越快越好

4.索引管理

索引管理

查看磁盘存储大小
db.userlevel.stats().indexSizes
{ "_id_" : 2535424, "user_id_1" : 24576}
单位byte

查看各索引具体内存占用大小
db.userlevel.stats({indexDetails:true}).indexDetails
"bytes currently in the cache" : 4587

索引管理

- 索引需要适配内存，放在内存里才是最快的
- 存储索引很大，没有压缩
- 内存中会放最近用的索引，或者说B-Tree右端近来单调递增的索引

索引管理

$indexStats 索引状态

```
db.foo.aggregate( [ { $indexStats: { } } ] )
{
        "name" : "_id_",....
}
{
        "name" : "vip_level_1",                 索引名
        "key" : {                               索引键值
                "vip_level" : 1
        },
        "host" : "mongo001:27017",              目前主机名与端口
        "accesses" : {
                "ops" : NumberLong(0),          调用次数
                "since" : ISODate("2017-05-20T14:49:27.073Z")
        }
}                                               开始收集时间
```

索引管理

- Mongod服务重启或者重建索引后会重置该统计
- 不想重启，可找两个时间点进行抽样

- 基于索引名称:
  db. foo.aggregate([{$indexStats: {}},
  {$match: {"name": " vip_level_1 "}}])
- 基于索引字段:
  db. foo.aggregate([{$indexStats: {}},
  {$match: { “key": " vip_level "}}])

稀疏索引

```
db.addresses.createIndex( { "xmpp_id": 1 }, { sparse: true } )
```

不存储Null信息的索引，常和唯一索引连用

局部索引（稀疏索引进化版）

一种在指定赛道上，油耗更低的索引

```
db.userlevel.createIndex({vip_level:1},
{partialFilterExpression:{vip_level:{$gte:2}}})
```

也可以跨赛道

```
db.userlevel.createIndex({user_id:1},
{partialFilterExpression:{vip_level:{$gte:2}}})
```

局部索引（稀疏索引进化版）

partialFilterExpression支持以下过滤条件：

$eq	等值
$exists: true	存在
$gt, $gte, $lt, $lte	大小等于
$type	类型
$and	支持最外层有and条件

局部索引

注意事项：

- 与业务需求密切相关，条件外的会用不到索引
- 3.2以上才有
- 不能当做分片的片键
- _id不能创建局部索引
- 同一个索引不能和sparse同时使用
- 一个键上不能有多个不同的局部索引

后台创建索引

```
db.people.createIndex( { zipcode: 1}, {background: true} )
```

生产环境切记要加上background参数，虽然会比默认的慢，但是不会锁表

指定从节点创建索引

1、该从节点priority设为0（不会变成主节点）

2、关闭该从节点，以单机模式启动（配置文件加注释）

3、添加索引

4、关闭该从节点，以副本集模式启动（配置文件还原）

适用于BI、报表的查询，需要大量索引，不影响主节点写入，不会驱逐掉常规缓存

Q & A