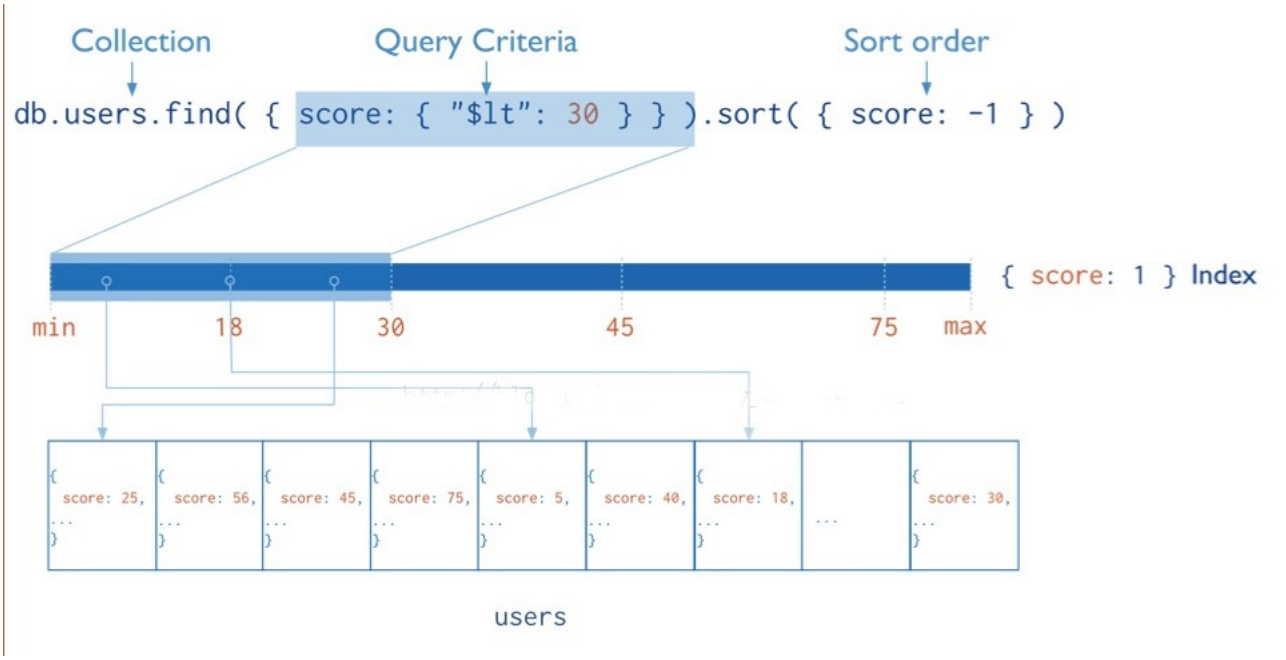
link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=56 sentences=142, words=522

1. 索引

- 索引通常能够极大的提高查询的效率，如果没有索引，MongoDB在读取数据时必须扫描集合中的每个文件并选取那些符合查询条件的记录。
- 这种扫描全集合的查询效率是非常低的，特别在处理大量的数据时，查询可以要花费几十秒甚至几分钟，这对网站的性能是非常致命的。
- 索引是特殊的数据结构，索引存储在一个易于遍历读取的数据集合中，索引是对数据库表中一列或多列的值进行排序的一种结构



- 特殊的数据结构，按顺序保存文档中的一个或多个字段
- 使用索引，方便范围查询和匹配查询。

2 建立索引

2.1 插入数据

```
var students = [];
for(var i=1;i300000;i++){
   students.push({name:'zfpx'+i,age:i,random:i});
}
 db.students.insert(students);
db.students.find({age:299999}).explain(true);
```

2.1.2 创建匿名索引

```
db.students.ensureIndex({age:1});
db.students.find({age:299999}).explain(true);
```

2.1.3 创建命名索引

```
db.students.ensureIndex({name:1},{name:'namedIndex'});
db.students.getIndexes()
```

2.1.4 分析索引的执行过程

MongoDB 提供了一个 explain 命令让我们获知系统如何处理查询请求。利用 explain 命令，我们可以很好地观察系统如何使用索引来加快检索，同时可以针对性优化索引。

- cursor: 返回游标类型
    - BasicCursor而没有使用索引的查询并不是胡乱查询，而是使用了基本游标：，同理，
    - 使用索引的查询就是BtreeCursor
- nscanned: 查找过的索引条目的数量
- n: 返回的文档数量
- nscannedObjects：数据库按照索引去磁盘上查找实际文档的次数
- millis: 执行本次查询所花费的时间，以毫秒计算，这也是判断查询效率的一个重点
- indexBounds: 描述索引的使用情况
- isMultiKey:是否使用了多键索引
- scanAndOrder: 是否在内存中对结果进行了排序
- indexOnly:是否仅仅使用索引完成了本次查询

```
db.students.find({name:'zfpx150000'}).explain();
```

2.1.5 指定使用的索引

```
db.students.find({name:'zfpx299999',age:299999}).hint({name:1}).explain(true);
```

2.1.6 创建唯一索引并删除重复记录

```
db.person.ensureIndex({ "name" : -1 },{ "name" : "indexname", "unique" : true,dropDups:true })
```

2.1.7 删除索引

```
db.students.dropIndex('namedIndex');
db.students.dropIndex('*');
db.runCommand({dropIndexes:"students",index:"namedIndex"});
```

2.1.8 在后台创建索引

```
db.students.ensureIndex({name:1},{name:'nameIndex',unique:true,background:true});
```

2.1.9 建立多键索引

mongodb可以自动对数组进行索引

```
db.students.insert({hobby:['basketball','football','pingpang']});
db.students.ensureIndex({hobby:1});
db.students.find({hobby:'football'},{hobby:1,_id:0}).explain(true);
```

2.1.10 复合索引

查询的条件不止一个，需要用复合索引

```
db.students.ensureIndex({name:1,age:1});
db.students.find({name:1,age:2},{name:1,age:1,_id:0}).explain(true);
```

2.1.11 过期索引

在一定的时间后会过期，过期后相应数据数据被删除,比如 session、日志、缓存和临时文件

```
db.stus.insert({time:new Date()});
db.stus.ensureIndex({time:1},{expireAfterSeconds:10});
db.stus.find();
```

- 1.索引字段的值必须Date对象，不能是其它类型比如时间戳
- 2.删除时间不精确，每60秒跑一次。删除也要时间，所以有误差。

2.1.12 全文索引

大篇幅的文章中搜索关键词,MongoDB为我们提供了全文索引

```
db.article.insert({content:'I am a gir'});
db.article.insert({content:'I am a boy'});
```

- $text:表示要在全文索引中查东西
- $search:后边跟查找的内容, 默认全部匹配

```
db.article.find({$text:{$search:'boy'}});
db.article.find({$text:{$search:'girl'}});
db.article.find({$text:{$search:'boy girl'}});
db.article.find({$text:{$search:"a b"}});
db.article.find({$text:{$search:"boy -girl"}});
db.article.find({$text:{$search:"a \"coco cola\" b "}});
```

2.2 二维索引

mongodb提供强大的空间索引可以查询出一定落地的地理坐标

```
[{ gis : { x : 50 , y : 30 } ];
```

2.2.1 创建2D索引

```
db.maps.ensureIndex({gis:'2d'},{min:1,max:3});
```

默认会创建一个2D索引

2.2.2 查询出距离点()最近的3个点

```
db.maps.find({gis:{$near:[1,1]}}).limit(3);
```

2.2.3 查询以点(50,50)和点(190,190)为对角线的正方形中的所有的点

```
db.map.find({gis:{$within:{$box:[[1,1],[2,2]]}}},{_id:0,gis:1});
```

2.2.4 查出以圆心为为半径为规则下圆心面积中的点

```
db.maps.find({gis:{$within:{$center:[[2,2],1]}}},{_id:0,gis:1});
```

2.2.5 索引使用的注意事项

1. 1为正序 -1为倒序
2. 索引虽然可以提升查询性能，但会降低插件性能，对于插入多查询少不要创索引
3. 数据量不大时不需要使用索引。性能的提升并不明显，反而大大增加了内存和硬盘的消耗。
4. 查询数据超过表数据量30%时，不要使用索引字段查询
5. 排序工作的时候可以建立索引以提高排序速度
6. 数字索引，要比字符串索引快的多

附录

queryPlanner分析

- queryPlanner: queryPlanner的返回
    - queryPlanner.namespace:该值返回的是该query所查询的表
    - queryPlanner.indexFilterSet:针对该query是否有indexfilter
    - queryPlanner.winningPlan:查询优化器针对该query所返回的最优执行计划的详细内容。
    - queryPlanner.winningPlan.stage:最优执行计划的stage，这里返回是FETCH，可以理解为通过返回的index位置去检索具体的文档（stage有数个模式，将在后文中进行详解）。
    - queryPlanner.winningPlan.inputStage:用来描述子stage，并且为其父stage提供文档和索引关键字。
    - queryPlanner.winningPlan.stage的child stage，此处是IXSCAN，表示进行的是index scanning。
    - queryPlanner.winningPlan.keyPattern:所扫描的index内容，此处是did:1,status:1,modify_time: -1与scid : 1
    - queryPlanner.winningPlan.indexName：winning plan所选用的index。
    - queryPlanner.winningPlan.isMultiKey是否是Multikey，此处返回是false，如果索引建立在array上，此处将是true。
    - queryPlanner.winningPlan.direction：此query的查询顺序，此处是forward，如果用了.sort({modify_time:-1})将显示backward。
    - queryPlanner.winningPlan.indexBounds:winningplan所扫描的索引范围,如果没有制定范围就是[MaxKey, MinKey]，这主要是直接定位到mongodb的chunck中去查找数据，加快数据读取。
    - queryPlanner.rejectedPlans：其他执行计划（非最优而被查询优化器reject的）的详细返回，其中具体信息与winningPlan的返回中意义相同，故不在此赘述。

对executionStats返回逐层分析

- 第一层，executionTimeMillis
- 最为直观explain返回值是executionTimeMillis值，指的是我们这条语句的执行时间，这个值当然是希望越少越好。
- 其中有3个executionTimeMillis，分别是：
    - executionStats.executionTimeMillis 该query的整体查询时间。
    - executionStats.executionStages.executionTimeMillisEstimate
    - 该查询根据index去检索document获得2001条数据的时间。
    - executionStats.executionStages.inputStage.executionTimeMillisEstimate
    - 该查询扫描2001行index所用时间。
- 第二层，index与document扫描数与查询返回条目数 这个主要讨论3个返回项，nReturned、totalKeysExamined、totalDocsExamined，分别代表该条查询返回的条目、索引扫描条目、文档扫描条目。 这些都是直观地影响到executionTimeMillis，我们需要扫描的越少速度越快。 对于一个查询，我们最理想的状态是：nReturned=totalKeysExamined=totalDocsExamined
- 第三层，stage状态分析 那么又是什么影响到了totalKeysExamined和totalDocsExamined？是stage的类型。类型列举如下：
- COLLSCAN：全表扫描
- IXSCAN：索引扫描
- 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$or查询的stage返回
- TEXT：使用全文索引进行查询时候的stage返回
- PROJECTION：限定返回字段时候stage的返回
- 对于普通查询，我希望看到stage的组合(查询的时候尽可能用上索引)：
    - Fetch+IDHACK
    - Fetch+ixscan
    - Limit+（Fetch+ixscan）
    - PROJECTION+ixscan
    - SHARDING_FITER+ixscan
    - COUNT_SCAN
- 不希望看到包含如下的stage：
    - COLLSCAN(全表扫描),SORT(使用sort但是无index),不合理的SKIP,SUBPLA(未用到index的$or),COUNTSCAN(不使用index进行count)