.

Wednesday, June 5, 2013

# MongoDB course note 4

## Indexes

Which optimization will typically have the greatest impact on the performance of a database.
Adding appropriate indexes on large collections so that only a small percentage of queries need to scan the collection.

## Creating Indexes

Please provide the mongo shell command to add an index to a collection named*students*, having the index key be *class*, *student_name*.
db.students.ensureIndex({class:1, student_name:1})

## Multikey Indexes

Suppose we have a collection *foo* that has an index created as follows:

```
db.foo.ensureIndex({a:1, b:1})
```

Which of the following inserts are valid to this collection?

- ☑ db.foo.insert({a:["apples","oranges"], b:"grapes"})
- ☑ db.foo.insert({a:"grapes", b:"oranges"})
- ☑ db.foo.insert({a:"grapes", b:[8,9,10]})
- □ db.foo.insert({a:[1,2,3], b:[5,6,7]})

Two parallel arrays index are not allowed.

## Index Creation option, Unique

Please provide the mongo shell command to add a unique index to the collection*students* on the keys *student_id*, *class_id*.
db.students.ensureIndex({student_id:1, class_id:1}, {unique: true})

## Index Creation, Removing Dups

If you choose the dropDups option when creating a unique index, what will the MongoDB do to documents that conflict with an existing index entry?
Delete them for ever and ever, Amen.

## Index Creation, Sparse

Suppose you had the following documents in a collection called *people* with the following docs:

```
> db.people.find()
{ "_id" : ObjectId("50a464fb0a9dfcc4f19d6271"), "name" : "Andrew", "title" : "Jester" }
{ "_id" : ObjectId("50a4650c0a9dfcc4f19d6272"), "name" : "Dwight", "title" : "CEO" }
{ "_id" : ObjectId("50a465280a9dfcc4f19d6273"), "name" : "John" }
```

**Search This Blog**

Search

**Labels**

- Java (48)
- Web Service (14)
- Ruby on Rails (13)
- Hadoop (11)
- SCM (10)
- MVNE / MVNO / Telecom (9)
- Mongodb (9)
- Git (5)
- Rest (5)
- Unit test (5)
- clean code (5)
- Software architecture (4)
- Ruby (3)
- Heroku (2)
- Lunix (2)
- Requirement engineering (2)
- C# (1)
- CSS (1)
- Python (1)

**About Me**

.

Copenhagen, Denmark

I'm a software developer, clean code practitioner and requirement analyst. I'm working on improving software quality from software engineering perspectives (Requirement engineering, software architecture, design, SCM, clean code, and testing). Currently, I work as a Telecom technical consultant in CPH.

View my complete profile

And there is an index defined as follows:

```
db.people.ensureIndex({title:1}, {sparse:1})
```

If you perform the following query, what do you get back, and why?

```
db.people.find({title:null})
```

No documents, because the query uses the index and there are no documents with title:null in the index.

### Index Creation, Background

Which things are true about creating an index in the background in MongoDB. Check all that apply.
Although the database server will continue to take requests, a background index creation still blocks the mongo shell that you are using to create the index.
Creating an index in the background takes longer than creating it in the foreground

### Using Explain

Given the following output from explain, what is the best description of what happened during the query?

```
{
 "cursor" : "BasicCursor",
 "isMultiKey" : false,
 "n" : 100000,
 "nscannedObjects" : 10000000,
 "nscanned" : 10000000,
 "nscannedObjectsAllPlans" : 10000000,
 "nscannedAllPlans" : 10000000,
 "scanAndOrder" : false,
 "indexOnly" : false,
 "nYields" : 7,
 "nChunkSkips" : 0,
 "millis" : 5151,
 "indexBounds" : {

 },
 "server" : "Andrews-iMac.local:27017"
}
```

The query scanned 10,000,000 documents, returning 100,000 in 5.2 seconds.

### When is an index used?

Given collection foo with the following index:

```
db.foo.ensureIndex({a:1, b:1, c:1})
```

Which of the following queries will use the index?

☐ db.foo.find({b:3, c:4})
☑ db.foo.find({a:3})
☑ db.foo.find({c:1}).sort({a:1, b:1})
☐ db.foo.find({c:1}).sort({a:-1, b:1})

### How large is your index?

Is it more important that your index or your data fit into memory?
Index

## Index Cardinality

Let's say you update a document with a key called tags and that update causes the document to need to get moved on disk. If the document has 100 tags in it, and if the tags array is indexed with a multikey index, how many index points need to be updated in the index to accomodate the move?
100

## Index Selectivity

Given the following attributes of automobiles: color, weight, manufacturer, odometer mileage, which index is likely be the most selective, provided you can provide all four attributes on a search:

○ Color

○ Weight

○ Manufacturer

● Odometer Mileage

## Hinting an Index

Given the following data in a collection:

```
> db.people.find()
{ "_id" : ObjectId("50a464fb0a9dfcc4f19d6271"), "name" : "Andrew", "title" : "Jester" }
{ "_id" : ObjectId("50a4650c0a9dfcc4f19d6272"), "name" : "Dwight", "title" : "CEO" }
{ "_id" : ObjectId("50a465280a9dfcc4f19d6273"), "name" : "John" }
```

and the following indexex:

```
> db.people.getIndexes()
[
 {
  "v" : 1,
  "key" : {
   "_id" : 1
  },
  "ns" : "test.people",
  "name" : "_id_"
 },
 {
  "v" : 1,
  "key" : {
   "title" : 1
  },
  "ns" : "test.people",
  "name" : "title_1",
  "sparse" : 1
 }
]
```

Which query below will return the most documents.

● db.people.find().sort({'title':1}).hint({$natural:1})

○ db.people.find().sort({'title':1})

○ db.people.find({name:{$ne:"Kevin"}}).sort({'title':1})

○ db.people.find({'title':{$ne:null}}).hint({'title':1})

hint natural to use BasicCursor returns all docs.

## Geospatial Indexes

Suppose you have a 2D geospatial index defined on the key *location* in the collection *places*. Write a query that will find the closest three places (the closest three documents) to the location 74, 140.

db.places.find({location: {$near: [74,140]}}).limit(3)

## Profiling

Write the query to look in the system profile collection for all queries that took longer than one second, ordered by timestamp descending.

db.system.profile.find({millis: {$gt: 1000}}).sort({ts: -1})

profile level 0: log off.

level 1: only record slow logs.

level 2: record all logs

## HW 4.1

Suppose you have a collection with the following indexes:

```
> db.products.getIndexes()
[
 {
  "v" : 1,
  "key" : {
   "_id" : 1
  },
  "ns" : "store.products",
  "name" : "_id_"
 },
 {
  "v" : 1,
  "key" : {
   "sku" : 1,
   "unique" : true
  },
  "ns" : "store.products",
  "name" : "sku_1_unique_true"
 },
 {
  "v" : 1,
  "key" : {
   "price" : -1
  },
  "ns" : "store.products",
  "name" : "price_-1"
 },
 {
  "v" : 1,
  "key" : {
   "description" : 1
  },
  "ns" : "store.products",
  "name" : "description_1"
 },
 {
  "v" : 1,
  "key" : {
   "category" : 1,
   "brand" : 1
  },
  "ns" : "store.products",
  "name" : "category_1_brand_1"
 },
 {
  "v" : 1,
  "key" : {
   "reviews.author" : 1
  },
  "ns" : "store.products",
  "name" : "reviews.author_1"
 }
```

Which of the following queries can utilize an index. Check all that apply.

- □ db.products.find({'brand':"GE"})
- ☑ db.products.find({'brand':"GE"}).sort({price:1})

- ☑ db.products.find({$and:[{price:{$gt:30}},{price:{$lt:50}}]}).sort({brand:1})
- □ db.products.find({brand:'GE'}).sort({category:1, brand:-1}).explain()

## HW 4.2

Suppose you have a collection called *tweets* whose documents contain information about the *created_at* time of the tweet and the user's *followers_count* at the time they issued the tweet. What can you infer from the following *explain* output?

```
db.tweets.find({"user.followers_count":{$gt:1000}}).sort({"created_at" : 1 }).limit(10).skip(5000).explain()
{
        "cursor" : "BtreeCursor created_at_-1 reverse",
        "isMultiKey" : false,
        "n" : 10,
        "nscannedObjects" : 46462,
        "nscanned" : 46462,
        "nscannedObjectsAllPlans" : 49763,
        "nscannedAllPlans" : 49763,
        "scanAndOrder" : false,
        "indexOnly" : false,
        "nYields" : 0,
        "nChunkSkips" : 0,
        "millis" : 205,
        "indexBounds" : {
                "created_at" : [
                        [
                                {
                                        "$minElement" : 1
                                },
                                {
                                        "$maxElement" : 1
                                }
                        ]
                ]
        },
        "server" : "localhost.localdomain:27017"
}
```

- ☑ This query performs a collection scan.
- ☑ The query uses an index to determine the order in which to return result documents.
- □ The query uses an index to determine which documents match.
- □ The query returns 46462 documents.
- ☑ The query visits 46462 documents.
- □ The query is a "covered index query".

Posted by . at 6:00 AM

+1 Recommend this on Google

Labels: Mongodb

## 1 comment:

**Toulon Fouryou** September 25, 2013 at 6:44 AM

Thank you

Reply

Enter your comment...

**Comment as:** Google Accou ▼

Publish Preview

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

**Blog Archive**

- ▼ 2013 (85)
  - ► November (1)
  - ► September (18)
  - ► August (6)
  - ► July (13)
  - ▼ June (11)
    - Better Rest API design 4
    - Better Rest API design 3
    - Better Rest API design 2
    - Better Rest API design 1
    - Liferay (Tomcat container) setup connections pool
    - MongoDB course note 7 (final)
    - ssh and scp
    - Aws, git, heroku env setup
    - MongoDB course note 6
    - MongoDB course note 5
    - MongoDB course note 4
  - ► May (14)
  - ► March (18)
  - ► February (4)
- ► 2012 (8)
- ► 2011 (37)
- ► 2010 (19)

@Hao. Simple template. Powered by Blogger.