

Blog

Home (<https://Mindmajix.Com/>) / MongoDB (<https://Mindmajix.Com/Mongodb/>)
/ MongoDB Interview Questions

MongoDB Interview Questions

★★★★☆ (4.0) | 19294 Ratings

Bookmark

Email

19314



If you're looking for MongoDB Interview Questions for Experienced or Freshers, you are at right place. There are lot of opportunities from many reputed companies in the world. According to research MongoDB has a market share of about 4.5%. So, You still have opportunity to move ahead in your career in MongoDB Development. Mindmajix offers Advanced MongoDB Interview Questions 2019 that helps you in cracking your interview & acquire dream career as MongoDB Developer.

MongoDB Vs Elasticsearch

MongoDB Vs ElasticSearch		
Name	MongoDB	Elastic Search
Database Models	Database and Key-value Storage	Search Analytics
Latest Version	3.6.3	6.2.3
Developed By	MongoDB Inc	ElasticSearch

Call us on

Drop us a Query

Implementation Language Used	C++	Java
APIs and Access Methods	JSON Prorocol	Java API
Extra features	MapReduce and inbuilt memory capability	Triggers and map reducing
Explore more about differences @ MongoDB Vs ElasticSearch (https://mindmajix.com/mongodb-vs-elasticsearch)		

Q1) What's a good way to get a list of all unique tags for a collection of documents millions of items large when we doesn't have access to mongodb's new "distinct" command ?

The normal way of doing tagging seems to be indexing multikeys. Even if your MongoDB driver doesn't implement distinct, we can implement.

In JavaScript you can write something like this:

```
result = db.$cmd.findOne({"distinct" : "collection_name", "key" : "tags"})
```

You do a findOne on the "\$cmd" collection of whatever database you're using. Pass it the collection name and the key you want to run distinct on.

If you ever need a command your driver doesn't provide a helper for, you can look at the below link for complete list of database commands.

<https://docs.mongodb.com/manual/reference/command/>

Q2) How to get the names of all the keys in a MongoDB collection ?

For example, from this:

```
db.things.insert( { type : ['dog', 'cat'] } );
```

```
db.things.insert( { egg : ['cat'] } );
```

```
db.things.insert( { type : [] } );
```

```
db.things.insert( { hello : [] } );
```

How to get the unique keys: type, egg, hello

We could do this with MapReduce:

Subscribe to our youtube channel to get new updates..!



Mindmajix Trainingz

YouTube

```
mr = db.runCommand({
  "mapreduce" : "my_collection",
  "map" : function() {
    for (var key in this) { emit(key, null); }
  },
  "reduce" : function(key, stuff) { return null; },
```

📞 Call us on

✉ Drop us a Query

```
"out": "my_collection" + "_keys"
})
```

Then run distinct on the resulting collection so as to find all the keys:

```
db[my.result].distinct("_id")
["foo", "bar", "baz", "_id", ...]
```

Q3) How to browse or query live MongoDB data?

Below are some utilities available in the market.

1. MongoHub is moved to a native mac version, please check <https://github.com/bububa/MongoHub-Mac>.

2. <https://github.com/Imaginea/mViewer>

This is awesome with tree and document views.

3. genghisapp

It is a web-based GUI that is clean, light-weight, straight-forward, offers keyboard shortcuts, and works awesomely. It also supports GridFS.

Best of all, it's a single script.

To install it
\$ gem install genghisapp bson_ext
(bson_ext is optional but will greatly improve the performance of the gui)

To run it (this will automatically open your web browser and navigate to the app as well)
genghisapp

To stop it
genghisapp -kill

Q4) How to use map/reduce to handle more than 10000 unique keys for grouping in MongoDB?

Use Version 2.2. The db.collection.group() method's returned array can contain at most 20,000 elements; i.e. at most 20,000 unique groupings. For group by operations that results in more than 20,000 unique groupings, use mapReduce. Previous versions had a limit of 10,000 elements.

Q5) Can anyone share some insight on the index/RAM relationship and what happens when both an individual index and all of my indexes exceed the size of available RAM?

MongoDB keeps what it can of the indexes in RAM. They'll be swapped out on an LRU basis. You'll often see documentation that suggests you should keep your "working set" in memory: if the portions of index you're actually accessing fit in memory, you'll be fine.

It is the working set size plus MongoDB's indexes which should ideally reside in RAM at all times i.e. the amount of available RAM should ideally be at least the working set size plus the size of indexes plus what the rest of the OS (Operating System) and other software running on the same machine needs. If the available RAM is less than that, LRUing is what happens and we might therefore get significant slowdown. One thing to keep in mind is that in an index btree buckets are cached, not individual index

Call us on

Drop us a Query

keys i.e. if we had a uniform distribution of keys in an index including for historical data, we might need more of the index in RAM compared to when we have a compound index on time plus something else. With the latter, keys in the same btree bucket are usually from the same time era, so this caveat does not happen. Also, we should keep in mind that our field names in BSON are stored in the records (but not the index) so if we are under memory pressure they should be kept short.

Q10) What does an appropriate `db.ClockTime.update()` statement look like to convert these text based values to a date datatype?

This code should do it:

```
> var cursor = db.ClockTime.find()
> while (cursor.hasNext()) {
... var doc = cursor.next();
... db.ClockTime.update({_id : doc._id}, {$set : {ClockInTime : new Date(doc.ClockInTime)}})
... }
```

I have exactly the same situation as Jeff Fritz.

In my case I have succeed with the following simpler solution:

```
db.ClockTime.find().forEach(function(doc) {
doc.ClockInTime=new Date(doc.ClockInTime);
db.ClockTime.save(doc);
})
```

Q11) Updating a specific key/value inside of an array field with MongoDB

As a preface, I've been working with MongoDB for about a week now, so this may turn out to be a pretty simple answer.

I have data already stored in my collection, we will call this collection content, as it contains articles, news, etc. Each of these articles contains another array called author which has all of the author's information (Address, Phone, Title, etc).

The Goal – I am trying to create a query that will update the author's address on every article that the specific author exists in, and only the specified author block (not others that exist within the array). Sort of a "Global Update" to a specific article that affects his/her information on every piece of content that exists.

Here is an example of what the content with the author looks like.

```
{
  "_id" : ObjectId("4c1a5a948ead0e4d09010000"),
  "authors" : [
    {
      "user_id" : null,
      "slug" : "joe-somebody",
      "display_name" : "Joe Somebody",
      "display_title" : "Contributing Writer",
      "callus_on" : "Call us on"
    }
  ]
}
```



Drop us a Query

```
"display_company_name" : null,
"email" : null,
"phone" : null,
"fax" : null,
"address" : null,
"address2" : null,
"city" : null,
"state" : null,
"zip" : null,
"country" : null,
"image" : null,
"url" : null,
"blurb" : null
},
{
"user_id" : null,
"slug" : "jane-somebody",
"display_name" : "Jane Somebody",
"display_title" : "Editor",
"display_company_name" : null,
"email" : null,
"phone" : null,
"fax" : null,
"address" : null,
"address2" : null,
"city" : null,
"state" : null,
"zip" : null,
"country" : null,
"image" : null,
"url" : null,
"blurb" : null
},
],
"tags" : [
"tag1",
"tag2",
"tag3"
```



```
],  
"title" : "Title of the Article"  
}
```

I can find every article that this author has created by running the following command:

```
db.content.find({authors: {$elemMatch: {slug: 'joe-somebody'}}});
```

So theoretically I should be able to update the authors record for the slug joe-somebody but not jane-somebody (the 2nd author), I am just unsure exactly how you reach in and update every record for that author.

I thought I was on the right track, and here's what I've tried.

```
b.content.update(  
  {authors:  
    {$elemMatch:  
      {slug: 'joe-somebody'}  
    }  
  },  
  {$set:  
    {address: '1234 Avenue Rd.'}  
  },  
  false,  
  true  
);
```

I just believe there's something I am missing in the \$set statement to specify the correct author and point inside of the correct array. Any ideas?

****Update****

I've also tried this now:

Skip code block

```
b.content.update(  
  {authors:  
    {$elemMatch:  
      {slug: 'joe-somebody'}  
    }  
  },  
  {$set:  
    {'authors.$.address': '1234 Avenue Rd.'}  
  },  
  false,  
  true  
);
```

📞 Call us on

✉ Drop us a Query



Solution:

This is what finally worked for me!

```
db.content.update({'authors.slug':'joe-somebody'},{$set:{'authors.$address':'Address That I
wanted'}}},false,true);
```

It updates all the records properly, thanks!

Maybe you can use the \$ operator (positional-operator)?

Explore MongoDB Sample Resumes! Download & Edit, Get Noticed by Top Employers!**Download Now!**
(<https://mindmajix.com/mongo-db-sample-resumes>)

Q12) MongoDB – simulate join or subquery

I'm trying to figure out the best way to structure my data in Mongo to simulate what would be a simple join or subquery in SQL.

Say I have the classic Users and Posts example, with Users in one collection and Posts in another. I want to find all posts by users who's city is "london".

I've simplified things in this question, in my real world scenario storing Posts as an array in the User document won't work as I have 1,000's of "posts" per user constantly inserting.

Q13) Can Mongos \$in operator help here? Can \$in handle an array of 10,000,000 entries?

Honestly, if you can't fit "Posts" into "Users", then you have two options.

1. Denormalize some User data inside of posts. Then you can search through just the one collection.
2. Do two queries. (one to find users the other find posts)

Based on your question, you're trying to do #2.

Theoretically, you could build a list of User IDs (or refs) and then find all Posts belonging to a User \$in that array. But obviously that approach is limited.

Q14) Can \$in handle an array of 10,000,000 entries?

Look, if you're planning to "query" your posts for all users in a set of 10,000,000 Users you are well past the stage of "query". You say yourself that each User has 1,000s of posts so you're talking about a query for "Users with Posts who live in London" returning 100Ms of records.

100M records isn't a query, that's a dataset!

If you're worried about breaking the \$in command, then I highly suggest that you use map/reduce. The Mongo Map/Reduce will create a new collection for you. You can then trim down or summarize this dataset as you see fit.

\$in can handle 100,000 entries. I've never tried 10,000,000 entries but the query (a query is also a document) has to be smaller than 4mb (like every document) so 10,0000,0000 entries isn't possible.

Why don't you include the user and its town in the Posts collection? You can index this town because you can index properties of embedded entities. You no longer have to simulate a join because you can query the Posts on the towns of its embedded users. ^

☎ Call us on

✉ Drop us a Query

This means that you have to update the Posts when the town of a user changes but that doesn't happen very often. This update will be fast if you index the UserId in the Posts collection.

Q15) Mongo complex sorting?

I know how to sort queries in MongoDB by multiple fields, e.g., `db.coll.find().sort({a:1,b:-1})`.

Q16) Can I sort with a user-defined function; e.g., supposing a and b are integers, by the difference between a and b (a-b)?

I don't think this is possible directly; the sort documentation certainly doesn't mention any way to provide a custom compare function.

You're probably best off doing the sort in the client, but if you're really determined to do it on the server you might be able to use `db.eval()` to arrange to run the sort on the server (if your client supports it).

Server-side sort:

```
db.eval(function() {
return db.scratch.find().toArray().sort(function(doc1, doc2) {
return doc1.a - doc2.a
})
});
```

Versus the equivalent client-side sort:

```
db.scratch.find().toArray().sort(function(doc1, doc2) {
return doc1.a - doc2.b
});
```

Note that it's also possible to sort via an aggregation pipeline and by the `$orderby` operator (i.e. in addition to `.sort()`) however neither of these ways lets you provide a custom sort function either.

Why don't create the field with this operation and sort on it ?

Related Page: MongoDB Vs PostgreSQL Comparison (<https://mindmajix.com/mongodb-vs-postgresql>)

Q17) How to set a primary key in MongoDB?

I want to set one of **my fields as primary key**. I am using MongoDB as my NoSQL.

`_id` field is reserved for primary key in mongodb, and that should be an unique value. If you don't set anything to `_id` it will automatically fill it with "MongoDB Id Object". But you can put any unique info into that field.

Additional info: <https://www.mongodb.org/display/DOCS/BSON>

Hope it helps.

Q18) Delete everything in a MongoDB database

I'm doing development on MongoDB. For totally non-evil purposes, I sometimes want to blow away everything in a databasethat is, to delete every single collection, and whatever else might be lying around, and start from scratch. Is there a single line of code that will let me do this? Bonus points for

giving both a MongoDB console method and a MongoDB Ruby driver method.  **Drop us a Query**


```
use [database];
```

```
db.dropDatabase();
```

Ruby code should be pretty similar.

Also, from the command line:

```
mongo [database] -eval "db.dropDatabase();"
```

Use

```
[databaseName]
```

```
db.Drop+databaseName();
```

```
drop collection
```

```
use databaseName
```

```
db.collectionName.drop();
```

Q19) How to \$set sub-sub-array items in MongoDB

I'm developing a webapp which has a portal-ish component to it (think like multiple panels that can be drug around from column to column and added or removed). I'm using MongoDB to store this info with a format like so...

Skip code block

```
{
  _id: ObjectId(...),
  title: 'My Layout',
  columns: [
    {
      order: 1,
      width: 30,
      panels: [
        { title: 'Panel Title', top: 100, content: '...' },
        { title: 'Panel Title', top: 250, content: '...' },
      ]
    },
    {
      ... multiple columns ...
    }
  ]
}
```

I'm attempting to use atomic/modifier operations with update() and this is getting confusing. If I wanted to just update one property of a specific panel, how do I reference that?

```
update(
  { _id: ObjectId(...) },
  { $set: { columns.[???].panels.[???].top: 500 } }
```



Call us on

Drop us a Query

If you know the index in the array you can access the array element directly using dot notation.

```
update(
{ _id: ObjectId(xxxx) },
{ $set: { 'columns.0.panels.0.top' : 125}}
)
```

Make sure you encase the dot notated path in quotes as a string.

Edit:

To give more detail on how this could work dynamically, I'll give an example in PHP:

```
$action = array("columns.$colNum.panels.$panelNum" => $newValue);
```

Yes there is the positional operator, but it does not appear to be advanced enough to change arrays within arrays, this may change in MongoDB 1.7.0

There is an alternative you can do instead of trying to stuff this information into a nested document. Try to flatten it out. You can create a collection that has panel & column objects:

column object:

```
{
_id: // Mongoid
type: 'column',
user: 'username',
order: 1,
width: 30,
}
```

panel object:

```
{
_id: //Mongoid
type: 'panel',
user: 'username',
parentColumn: //the columns _id string
top: 125,
left: 100
}
```

Then you can find all columns that belong to a user by doing:

```
find({ type: 'column', user:'username' });
```

You can find all panels for a specific column by doing:

```
find({type: 'panel', columnOwner:'ownerID'});
```

Since each column and panel will have a unique ID given by MongoDB to it, you can easily query and atomically set options.

```
update({'_id': ObjectId('idstring')}, {$set : { 'top' : 125}});
```

Q20) Some beginner's questions about MongoDB



I'm a beginner with MongoDB and I've some questions:

1. When I'm connected to Mongo, and i executeshow dbs I see 2 databases: admin and local. What's their role? Then if I execute an insert command likedb.foo.insert({"value":"mongo"}), the test database appears. Why? How can i specify a custom name for a database?
2. Withshow dbs I get the databases (somehow like show databases in sql), how can I then list the collections inside a database (I would use show tables in sql)?
3. When executing a command, the MongoDB tutorial always uses thedb object. Is it the main object (a sort of "connection" object) that has to used for executing commands or it's something else?

Thanks!

1. Adminand local contain various settings local to the server, like users who are authenticated to connect. Under beginner usage, you shouldn't need to worry about them at all. By default you connect to a database named test. To connect to a new database, just use databasename from the mongo command line, or mongo databasename from your OS shell.
2. Use [database_name]and then show collections
3. Thedb object is your root handle to the currently-selected database on the mongo commmand line. The command line is really just a Javascript command line, and there are various mongodb-specific objects and functions exposed that let you do stuff. Try help() for a full listing.

Related Page: MongoDB GUI - Top 7 MongoDB GUI Tools (<https://mindmajix.com/mongodb-gui-tools>)

Q21) How to update based on existing data in mongo

Sort of a mongo noob, and I have a feeling I am tackling this problem from the wrong direction.

I have about a 7 million document collection. Each document has two fields that I want to modify(not replace), basically they are big strings that have , and I want to replace those with n.

I spent about an hour trying to find a way to "back reference" the object returned by the query, which totally doesn't exist. What is the best approach for something like this?

You'll have to query for all the documents and update them one by one. If you do it in JavaScript, it would be something like:

```
mydb = db.getSisterDB("whateverDBYoureUsing");
var cursor = mydb.foo.find();
while (cursor.hasNext()) {
  var x = cursor.next();
  /* replace with n in x's strings ... */
  db.foo.update({_id : x._id}, x);
}
```

You can copy this into a .js file (say, replace.js), change the db and collection names, and run it as a script from the shell:

```
mongo replace.js
```

Q22) How to print out more than 20 items (documents) in MongoDB's shell?



```
db.foo.find().limit(300)
```

won't do it... it still prints out 20

```
db.foo.find().toArray()
```

```
db.foo.find().forEach(printjson)
```

will both print out very expanded view of each document instead of the 1-line version for find():

```
DBQuery.shellBatchSize = 300
```

will do.

MongoDB Docs - Getting Started with the mongo Shell - Executing Queries

You can use it inside of the shell to iterate over the next 20 results. Just type it if you see “has more” and you will see the next 20 items.

from the shell if you want to show all results you could do `db.collection.find().toArray()` to get all results without it

Could always do:

```
db.foo.find().forEach(function(f){print(tojson(f, ", true"))});
```

To get that compact view.

Also, I find it very useful to limit the fields returned by the find so:

```
db.foo.find({}, {name:1}).forEach(function(f){print(tojson(f, ", true"))});
```

which would return only the `_id` and `name` field from `foo`.

Q23) How to store timestamps? Are created and updated fields available automatically?

What's the best way to store timestamps in MongoDB?

Which format is best:

```
# "created": { "d" : "2010-03-29", "t" : "20:15:34" }
```

```
# "created": "12343545234" # seconds since epoc
```

```
# "created": "2010-03-14T21:20:14+0000"
```

Is there a way to have MongoDB automatically set created and updated fields?

Which format is best

Best for what?

Is there a way to have MongoDB automatically set created and updated fields?

Created time is in the ObjectId but, as far as I know, you will have to update a updated field manually.

Example:

```
ObjectId("538141a9615760fd04ffef5f").getTimestamp()
```

1. The format you need to process it with best performance in your application should be preferred. Note that as default every document in MongoDB gets a created timestamp (<https://www.mongodb.org/display/DOCS/Object+IDs#ObjectIDs-DocumentTimestamps>)

2. See 1) + I think you need to manually set the “update” field.

If you do following on mongo shell it shows you time stamp that represents when that documents inserted using mongold. For ex. `ObjectId("51f3dee5ee49f9b91e0db133").getTimestamp()`, then it returns `ISODate`.



Q24) MongoDB: Updating documents using data from the same document

✉ Drop us a Query

I have a list of documents, each with lat and lon properties (among others).

```
{ 'lat': 1, 'lon': 2, someotherdata [...] }
{ 'lat': 4, 'lon': 1, someotherdata [...] }
[...]
```

I want to modify it so that it looks like this:

```
{ 'coords': { 'lat': 1, 'lon': 2 }, someotherdata [...] }
{ 'coords': { 'lat': 4, 'lon': 1 }, someotherdata [...] }
[...]
```

So far I've got this:

```
db.events.update({}, {$set : {'coords': {'lat': db.events.lat, 'lon': db.events.lon}}}, false, true)
```

But it treats the db.events.lat and db.events.lon as strings. How can I reference the document's properties?

Cheers.

The \$rename operator (introduced a month after this question was posted) makes it really easy to do these kinds of things where you don't need to modify the values.

Insert some test documents

```
db.events.insert({ 'lat': 1, 'lon': 2, someotherdata: [] })
```

```
db.events.insert({ 'lat': 4, 'lon': 1, someotherdata: [] })
```

use the \$rename operator

```
db.events.update({}, {$rename: {'lat': 'coords.lat', 'lon': 'coords.lon'}}, false, true)
```

Results

Skip code block

```
db.events.find()
```

```
{
  "_id" : ObjectId("5113c82dd28c4e8b79971add"),
  "coords" : {
    "lat" : 1,
    "lon" : 2
  },
  "someotherdata" : [ ]
}
{
  "_id" : ObjectId("5113c82ed28c4e8b79971ade"),
  "coords" : {
    "lat" : 4,
    "lon" : 1
  },
  "someotherdata" : [ ]
}
```

Update If all you have to do is change the structure of a document without changing the values, see [MongoDB Interview Questions](#)

gipset's answer for a nice solution.

According to a (now unavailable) comment on the Update documentation page, you cannot reference the current document's properties from within an update().

You'll have to iterate through all the documents and update them like this:

Skip code block

```
db.events.find().snapshot().forEach(
function (e) {
// update document, using its own properties
e.coords = { lat: e.lat, lon: e.lon };
// remove old properties
delete e.lat;
delete e.lon;
// save the updated document
db.events.save(e);
}
)
```

Such a function can also be used in a map-reduce job or a server-side db.eval() job, depending on your needs.

Q25) In MongoDB how do you use \$set to update a nested value?

In MongoDB how do you use \$set to update a nested value?

Using the dot notation:

```
db.people.update({ }, { $set: { "address.street": "Main Street" } })
```

Q26) Mongorestore of a db causing me trouble

I'm new to MongoDB and I have hard time to backup my local DB and restore it on my server. I found the link on Mongo's website: <https://www.mongodb.org/display/DOCS/Import+Export+Tools> but I still have problems with the restore.

When I do my backup I call

```
mongodump -db Gen
```

Then I see that all the collections are dump in /bin/dump/Gen folder

I copy-paste from local to the server in the same folder the call

```
mongorestore -db Gen -drop -dbpath dump/Gen
```

But I get the following : Error : root directory must be a dump of a single database when specifying a db name with -db

What am I doing wrong?

Thanks for the help!

Ok I find out what I'm doing wrong :

I was doing

```
mongorestore -db Gen -drop -dbpath dump/Gen
```

📞 Call us on



✉ Drop us a Query

But without the `-dbpath` it works just fine!

`mongorestore -db Gen -drop dump/Gen`

Thanks everyone!

Q27) How to create user accounts in MongoDB?

I wonder what's 'correct' way to create user accounts in MongoDB and actions like register/login. Do I have to create a specific collection for users (Username,Email,Password) or MongoDB has something built in already for users?

If I have to create users collection manually, how to deal with password encryption? Thanks.

You'll probably have to create and manage the collection of users manually.

As for encrypting passwords, the common approach is to hash the passwords using a suitable hash-function before you store them in the db. Later, when a user tries to login you use the same hash-function on the string they entered in the password field and compare that result to the password entry in your db. That way you never store the actual passwords and if someone hacks you they won't get your users passwords, just their hashes.

For extra security (against dictionary attacks) you should also salt the hashed passwords. Read about it [here](#)

Q28) How update the `_id` of one MongoDB Document?

I want update an `_id` MongoDB of one document. I know it's not a really good practice. But with some technical reason, I need update it. But If I try to update it I have :

```
> db.clients.update({'_id':ObjectId("4cc45467c55f4d2d2a000002")}, {'$set': {'_id':ObjectId("4c8a331bda76c559ef000004")}});
```

Mod on `_id` not allowed

And the update is not made. How I can really update it ?

You cannot update it. You'll have to save the document using a new `_id`, and then remove the old document.

Skip code block

```
// store the document in a variable
```

```
doc = db.clients.findOne({'_id': ObjectId("4cc45467c55f4d2d2a000002")})
```

```
// set a new _id on the document
```

```
doc._id = ObjectId("4c8a331bda76c559ef000004")
```

```
// insert the document, using the new _id
```

```
db.clients.insert(doc)
```

```
// remove the document with the old _id
```

```
db.clients.remove({'_id': ObjectId("4cc45467c55f4d2d2a000002")})
```

To do it for your whole collection you can also use a loop (based on Niels example):

```
db.status.find().forEach(function(doc){ var id=doc._id; doc._id=doc.UserId; db.status.insert(doc); db.status.remove({'_id:id}); })
```

In this case `UserId` was the new ID I wanted to use

📞 Call us on

✉ Drop us a Query

Related Page: MongoDB Query & Examples (<https://mindmajix.com/mongodb-query-and-examples>)

Q29) Get MongoDB Databases in a Javascript Array?

I know that in the MongoDB terminal, I can run “show dbs” to see the available databases. I want to get them programmatically so that I can iterate over them and delete some based upon a regular expression. I have tried `db.runCommand(“show dbs”)` but that doesn’t work.

Thanks in advance.

```
> db.getMongo().getDBNames()
[
  “test”,
  “admin”,
  “local”
]
> db.getMongo().getDBNames
function () {
  return this.getDBs().databases.map(function (z) {return z.name;});
}
```

Based upon this answer https://groups.google.com/group/mongodb-user/browse_thread/thread/9b3568f3a3cf4271, I was able to code up a solution.

use admin

```
dbs = db.runCommand({listDatabases: 1})
dbNames = []
for (var i in dbs.databases) { dbNames.push(dbs.databases[i].name) }
```

Hopefully this will help someone else.

The below will create an array of the names of the database:

```
var connection = new Mongo();
var dbNames = connection.getDBNames();
```

Q31) MongoDB and “joins”

I’m sure MongoDB doesn’t officially support “joins”. What does this mean?

Does this mean “We cannot connect two collections (tables) together.”?

I think if we put the value for `_id` in collection A to the `other_id` in collection B, can we simply connect two collections?

If my understanding is correct, MongoDB can connect two tables together, say, when we run a query. This is done by “Reference” written in <https://www.mongodb.org/display/DOCS/Schema+Design>.

Then what does “joins” really mean?

I’d love to know the answer because this is essential to learn MongoDB schema design. <https://www.mongodb.org/display/DOCS/Schema+Design>

It’s no join since the relationship will only be evaluated when needed. A join (in a SQL database) on the other hand will resolve relationships and return them as if they were a single table (you “join two tables

to tables” on

✉ Drop us a Query

You can read more about DBRef here: <https://docs.mongodb.org/manual/applications/database-references/>

There are two possible solutions for resolving references. One is to do it manually, as you have almost described. Just save a document's `_id` in another document's `other_id`, then write your own function to resolve the relationship. The other solution is to use DBRefs as described on the manual page above, which will make MongoDB resolve the relationship client-side on demand. Which solution you choose does not matter so much because both methods will resolve the relationship client-side (note that a SQL database resolves joins on the server-side).

The database does not do joins — or automatic “linking” between documents. However you can do it yourself client side. If you need to do 2, that is ok, but if you had to do 2000, the number of client/server turnarounds would make the operation slow.

In MongoDB a common pattern is embedding. In relational when normalizing things get broken into parts. Often in mongo these pieces end up being a single document, so no join is needed anyway. But when one is needed, one does it client-side.

Consider the classic ORDER, ORDER-LINEITEM example. One order and 8 line items are 9 rows in relational; in MongoDB we would typically just model this as a single BSON document which is an order with an array of embedded line items. So in that case, the join issue does not arise. However the order would have a CUSTOMER which probably is a separate collection – the client could read the `cust_id` from the order document, and then go fetch it as needed separately.

There are some videos and slides for schema design talks on the [mongodb.org](https://www.mongodb.org) web site I believe.

The first example you link to shows how MongoDB references behave much like lazy loading not like a join. There isn't a query there that's happening on both collections, rather you query one and then you lookup items from another collection by reference.

Mongo interface

Q31) What are some GUIs to use with Mongo, and what features do they offer? I'm looking for facts here, not opinions on which interface is best.

Official List from MongoDB

<https://www.mongodb.org/display/DOCS/Admin+UIs>

Web Based

For PHP, I'd recommend Rock Mongo. Solid, lots of great features, easy setup.

<https://rockmongo.com/>

If you don't want to install anything ... you can use MongoHQ's web interface (even if you your MongoDB isn't on MongoHQ.)

<https://mongohq.com/home>

Mac OS X

While MongoHub had been a decent option for a while it's bugs make it virtually unusable at this point ...

There is a more up-to-date (and less buggy) fork of the MongoHub project available: <https://github.com/fotonauts/MongoHub-Mac> you can download a binary here. ^

Windows

✉ Drop us a Query

By far, the best UI (for Windows) currently out there is MongoVUE.

<https://blog.mongovue.com/>

Looks great, lots of features, and if you are new it will really help you get going ...

<https://blog.mongovue.com/features/>

Here's a Q&A with the author too if you are interested ...

<https://learnmongo.com/posts/qa-ishann-kumar-creator-of-mongovue/>

On Mac there is MongoHub. On Windows you could try MongoVUE.

Also see Do any visual tools exist for MongoDB (for Windows)?

Screenshot of MongoHub:

Here's the official page of Admin UIs.

I have not really used any of them. But it looks like there is quite a bit of coverage there.

Web

At the shop where I work we use the Prudence platform for some stuff, and also MongoDB, so we of course use MongoVision a lot. Browser based, tabbed collection views, pretty-printed document editor, and three themes OOB. Open source.

<https://code.google.com/p/mongo-vision/>

OS X

MongoHub was as reliable as MongoVision.

<https://mongohub.todayclose.com/>

Q32) Why not mongodb?

I recently used MongoDB for the first time and found it exceptionally easy to use and high-performing. Which leads to my question – why not MongoDB?

Lets say I am implementing a Q & A app. My approach would be to implement the User data in a MySQL database and then use MongoDB for the question and answer storage – one collection storing a question and all responses.

Q33) Is there anything wrong with this approach?

MongoDB sounds like a fine application for your problem, but there are plenty of reasons why you would not use it.

MongoDB would not be well suited for applications that need:

1. Multi-Object Transactions: MongoDB only supports ACID transactions for a single document.
2. SQL: SQL is well-known and a lot of people know how to write very complex queries to do lots of things. This knowledge is transferrable across a lot of implementations where MongoDB's queries language are specific to it.
3. Strong ACID guarantees: MongoDB allows for things like inconsistent reads which is fine in some applications, but not in all.



4. Traditional BI: A lot of very powerful tools exist that allow for OLAP and other strong BI applications and those run against traditional SQL database.

Possible downsides:

1. You work in an organization that has only used SQL relational databases. You have no approval or support for using a NoSQL database yet.
2. You've never administered a MongoDB cluster; there's a learning curve, as with all technologies.
3. Your data is really relational (e.g., one User has many Questions; a Question has many Answers), and you've overlooked the possibility.

MongoDB is a fine solution, a good alternative for those situations where it applies. If you can use it, why not?

MongoDB is a brilliant database and I enjoy using it. That said, it has a few gotchas if you come from the world of SQL.

Apart from ACID and other things that are well documented (and in other answers too), these things have caught us by surprise:

1. MongoDB expects you to have memory. **Lots of memory.** If you can't fit your working set in memory, you can forget about it. This is different from most relational DBs which use memory only as cache! **To be more specific:** MongoDB uses RAM as primary storage and "swaps" the unneeded parts out to disk (Mongo leaves the decision over which parts get "swapped" to kernel). Traditional RDBMS work the other way around – they use disk as primary storage and use RAM as caching mechanism. So in general MongoDB uses more RAM. This is not a bad thing by itself, but as a consequence "real" RAM consumption is difficult to predict, which can lead to serious and unexpected degradation of performance once the working set grows over the (hard to predict) limit.

2. **Storage does not auto-shrink** when you remove records. The space that is allocated per collection stays allocated until you either repair DB or drop the collection. And it is allocated in huge chunks on a DB level (data files), which are then allocated to collections when needed (extents). That said, inside the collection's allocated space the documents that are removed DO release their space for other documents in the same collection. This is a good explanation of concepts: <https://www.10gen.com/presentations/storage-engine-internals>

3. As a contrast to SQL which is parsed server-side, in Mongo you pass the data structures to query and CRUD functions. The consequence is that **each driver provides a different syntax**, which is a bit annoying. For instance, PyMongo uses a list of tuples instead of a dictionary (probably because dict in Python does not preserve order of keys) to specify which fields will be returned by `find()`: (to be fair, that was probably the only sane way to do it – but it is a consequence of not using string-based language such as SQL)

a. MongoDB shell: `db.test.find({}, {a:1})`



b. PyMongo: `db.find({}, fields=[(a,1)])`

This should not be viewed as a criticism of MongoDB – I enjoy using it and it has proven to be a reliable and performant tool. But to use it properly you need to learn about its space management.

Q34) Track MongoDB performance?

Is there a way to track 'query' performance in MongoDB? Specially testing indexes or subdocuments?

In sql you can run queries, see execution time and other analytic metrics.

I have a huge mongoDB collection and want to try different variations and indexes, not sure how to test it, would be nice to see how long did it take to find a record.. (I am new in MongoDB). Thanks

There are two things here that you'll likely be familiar with.

1. Explain plans

2. Slow Logs

Explain Plans

Here are some basic docs on explain. Running explain is as simple as `db.foo.find(query).explain()`. (note that this actually runs the query, so if your query is slow this will be too)

To understand the output, you'll want to check some of the docs on the slow logs below. You're basically given details about "how much index was scanned", "how many are found", etc. As is the case with such performance details, interpretation is really up to you. Read the docs above and below to point you in the right direction.

Slow Logs

By default, slow logs are active with a threshold of 100ms. Here's a link to the full documentation on profiling. A couple of key points to get you started:

Get/Set profiling:

```
db.setProfilingLevel(2); // 0 => none, 1 => slow, 2 => all
```

```
db.getProfilingLevel();
```

See slow queries:

```
db.system.profile.find()
```

Q35) Mongodb can't start

today I updated my Mongo.. mongodb-stable (from 10gen repo)

but my service has down. the following command not working

```
$ sudo service mongodb start
```

```
$ start: Unknown job: mongodb
```

even this command not working

```
$ sudo /etc/init.d/mongodb start
```

```
$ Rather than invoking init scripts through /etc/init.d, use the service(8)
```

```
utility, e.g. service mongodb start
```

Since the script you are attempting to invoke has been converted to an

Upstart job, you may also use the start(8) utility, e.g. start mongodb

```
start: Unknown job: mongodb
```

📞 Call us on



✉ Drop us a Query

there is no mongo process running

```
$ ps -ef|grep mongo
```

```
$ user 9689 8121 0 13:01 pts/1 00:00:00 grep -color=auto mongo
```

log is here

Skip code block

```
tail /var/log/mongodb/mongodb.log
```

```
Fri Dec 10 11:24:35 [conn4] end connection 127.0.0.1:54217
```

```
Fri Dec 10 11:25:35 [initandlisten] connection accepted from 127.0.0.1:54229 #5
```

```
Fri Dec 10 11:26:25 [initandlisten] connection accepted from 127.0.0.1:54243 #6
```

```
Fri Dec 10 11:26:30 [conn6] end connection 127.0.0.1:54243
```

```
Fri Dec 10 11:30:13 got kill or ctrl c or hup signal 15 (Terminated), will terminate after current cmd ends
```

```
Fri Dec 10 11:30:13 [interruptThread] now exiting
```

```
Fri Dec 10 11:30:13 dbexit:
```

```
Fri Dec 10 11:30:13 [interruptThread] shutdown: going to close listening sockets...
```

```
Fri Dec 10 11:30:13 [interruptThread] closing listening socket: 5
```

```
Fri Dec 10 11:30:13 [interruptThread] closing listening socket: 6
```

```
Fri Dec 10 11:30:13 [interruptThread] closing listening socket: 7
```

```
Fri Dec 10 11:30:13 [interruptThread] closing listening socket: 8
```

```
Fri Dec 10 11:30:13 [interruptThread] shutdown: going to flush oplog...
```

```
Fri Dec 10 11:30:13 [interruptThread] shutdown: going to close sockets...
```

```
Fri Dec 10 11:30:13 [interruptThread] shutdown: waiting for fs preallocator...
```

```
Fri Dec 10 11:30:13 [interruptThread] shutdown: closing all files...
```

```
Fri Dec 10 11:30:13 closeAllFiles() finished
```

```
Fri Dec 10 11:30:13 [interruptThread] shutdown: removing fs lock...
```

```
Fri Dec 10 11:30:13 dbexit: really exiting now
```

for now, I'm running Mongo through this command just for a while, creating process manually

```
$ sudo mongod -f /etc/mongodb.conf
```

any idea? or has anyone updated Mongo-stable via Update manager?

Edits:

This mongodb version was v1.6.5 and it seems mongo team released it as stable with a bug. And they fixed it immediately at v1.7.4. You can see the major priority issue.

Bug has been reported and fixed.

<https://jira.mongodb.org/browse/SERVER-2200>

```
$ sudo apt-get purge mongodb-stable
```

```
$ sudo apt-get install mongodb-stable
```

(remove the lock file if present in /var/lib/mongodb)

```
$ sudo init 6
```

Then edit /etc/init/mongodb.conf removing the line "limit nofile 20000"

```
$ sudo vi /etc/init/mongodb.conf
```

```
$ sudo service mongodb start
```

📞 Call us on

✉ Drop us a Query

mongodb start/running, process 2351

Worked.

Running **Ubuntu 11.10** confirm you have the latest version of MongoDB:

```
$ mongod -version
```

```
db version v2.2.0
```

(If you don't have the latest version of MongoDB, follow the **Ubuntu Installation instructions** at:

<https://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/>)

First confirm that the **mongodb** user/group has permission to write to the data directory:

```
$ sudo chown -R mongodb:mongodb /var/lib/mongodb/.
```

Start up MongoDB as a Daemon (background process) using the following command:

```
$ mongod -fork -dbpath /var/lib/mongodb/ -smallfiles -logpath /var/log/mongodb.log -logappend
```

To Shut Down MongoDB enter the Mongo CLI, access the admin and issue the shutdown command:

```
$ ./mongo
```

```
> use admin
```

```
> db.shutdownServer()
```

See: <https://www.mongodb.org/display/DOCS/Starting+and+Stopping+Mongo>

Q36) MongoDB: Unique Key in Embedded Document

Is it possible to set a unique key for a key in an embedded document?

I have a Users collection with the following sample documents:

Skip code block

```
{
  Name: "Bob",
  Items: [
    {
      Name: "Milk"
    },
    {
      Name: "Bread"
    }
  ]
},
{
  Name: "Jim"
```

Is there a way to create an index on the property Items.Name?

I got the following error when I tried to create an index:

```
> db.Users.ensureIndex({"Items.Name": 1}, {unique:true});
```

```
E11000 duplicate key error index: GroceryGuruApp.Users.$Items.Name_1 dup key: {
```

```
: null }
```

📞 Call us on

✉ Drop us a Query

Any suggestions? Thank you!

Unique indexes exist only across collection. To enforce uniqueness and other constraints across document you must do it in client code. (Probably virtual collections would allow that, you could vote for it.)

What are you trying to do in your case is to create index on key Items.Name which doesn't exist in any of the documents (it doesn't refer to embedded documents inside array Items), thus it's null and violates unique constraint across collection.

The index will be across all Users and since you asked it for 'unique', no user will be able to have two of the same named item AND no two users will be able to have the same named Item.

Is that what you want?

Furthermore, it appears that it's objecting to two Users having a 'null' value for Items.Name, clearly Jim does, is there another record like that?

It would be unusual to require uniqueness on an indexed collection like this.

MongoDB does allow unique indexes where it indexes only the first of each value, see <https://www.mongodb.org/display/DOCS/Indexes#Indexes-DuplicateValues>, but I suspect the real solution is to not require uniqueness in this case.

If you want to ensure uniqueness only within the Items for a single user you might want to try the \$addToSet option. See <https://www.mongodb.org/display/DOCS/Updating#Updating-%24addToSet>

You can create a **unique compound sparse index** to accomplish something like what you are hoping for. It may not be the best option (client side still might be better), but it can do what you're asking depending on specific requirements.

To do it, you'll need to create another field on the same level as Name: Bob that is unique to each top-level record (could do FirstName + LastName + Address, we'll call this key Identifier).

Then create an index like this:

```
ensureIndex({'Identifier':1, 'Items.name':1},{ 'unique':1, 'sparse':1})
```

A sparse index will ignore items that don't have the field, so that should get around your NULL key issue. Combining your unique Identifier and Items.name as a compound unique index should ensure that you can't have the same item name twice per person.

Although I should add that I've only been working with Mongo for a couple of months and my science could be off. This is not based on empirical evidence but rather observed behavior.

More on MongoDB Indexes

1. Compound Keys Indexes
2. Sparse Indexes

Related Page: MongoDB Vs Elasticsearch Comparison (<https://mindmajix.com/mongodb-vs-elasticsearch>)

Q37) Creating a database in Mongo: can't connect, get "connect failed"

I want to create a new database in Mongo. However, I'm having trouble connecting:

~\$ mongo

MongoDB shell version: 1.6.5

connecting to: test
Call us on

✉ Drop us a Query

Tue Dec 21 18:16:25 Error: couldn't connect to server 127.0.0.1 (anon):1154

exception: connect failed

How can I connect to mongo in order to create a new database? Alternatively, can I create a new database from the command line?

Slightly surprisingly, the Mongo docs don't seem to cover how to create a database.

Thanks.

In order to open Mongo JavaScript shell, a Listener should be initialized first.

So, first run mongod.exe before running mongo.exe. Both are in the same location(/bin).

There is no separate commands to create a db in mongodb. Just type "use dbname;" in console. Now you have created a db of the name 'dbname'. Now, if you type 'show databases' you cannot see the db name you just created. Because, mongo will not create any db, until you create collection and insert a document into that collection.

Hope this is useful to you!

1. cd /var/lib/mongodb/
2. Remove mongod.lock file from this folder
3. Sudo start mongod (in console)
4. Mongo (in console)

And it runs fine.

First you'll need to run mongod on one terminal. Then fire up another terminal and type mongo. This shall open the mongo shell. You also need to create /data/db/ where mongo will store your databases.

Q38) How can I use MongoDB to find all documents which have a field, regardless of the value of that field?

For example, I have collection with documents, where documents can have field "url" (but most of them doesn't). How can I find all documents, which have field "url" (regardless of value of this field)?

To find if a key/field exists in your document use the \$exists operator.

Via the MongoDB shell ...

```
> db.things.find( { url : { $exists : true } } );
```

Q39) Mongodb Query To select records having a given key

let the records in database are

```
{ "_id": "1", "fn": "sagar", "ln": "Varpe" }
{ "_id": "1", "fn": "sag", "score": "10" }
{ "_id": "1", "ln": "ln1", "score": "10" }
{ "_id": "1", "ln": "ln2" }
```

I need to design a MongoDB query to find all records who has a given key

like if i pass "ln" as a parameter to query it should return all records in which "ln" is a Key, the results are

```
{ "_id": "1", "fn": "sagar", "ln": "Varpe" }
{ "_id": "1", "ln": "ln1", "score": "10" }
{ "_id": "1", "ln": "ln2" }
```



Call us on

✉ Drop us a Query

To find if a key/field exists in your document use the \$exists operator.

Via the MongoDB shell ...

```
db.things.find( { ln : { $exists : true } } );
```

Possible duplicate of this question: MongoDB queries

Q40) “Field name duplication not allowed with modifiers” on update

I get a “Field name duplication not allowed with modifiers” error while trying to update a field(s) in Mongo. An example:

```
> db.test.insert({test: “test1”, array: [0]});
> var testFetch = db.test.findOne({test: “test1”});
> db.test.update(testFetch,
{$push: {array: 1}, //push element to end of key “array”
$pop: {array: -1} //pop element from the start of key “array”
});
```

Field name duplication not allowed with modifiers

Is there no way to perform this atomic operation? I don’t want to be doing two separate updates for this.

There’s an outstanding issue for this on Mongo’s ticket system:

<https://jira.mongodb.org/browse/SERVER-1050>

Looks like it’s scheduled for this year. Your scenario is definitely a sensible scenario, but it’s also tied to a bunch of edge cases. What if you \$push and \$pop on an empty array? What’s expected? What do you want if you \$push and \$pull?

I don’t want to be doing two separate updates for this.

I know that doing this really has “code smell”, but is it a complete blocker for using this solution? Is the “double-update” going to completely destroy server performance?

Q41) In mongoDb, how do you remove an array element by its index

Skip code block

```
{
  “_id” : ObjectId(“4d1cb5de451600000000497a”),
  “name” : “dannie”,
  “interests” : [
    “guitar”,
    “programming”,
    “gadgets”,
    “reading”
  ]
}
```

In the example above, assume the above document is in the **db.people** collection. How to remove the 3rd element of the **interests** array by its **index**? ^

Edit:

This is my current solution:

```
var interests = db.people.findOne({"name":"dannie"}).interests;
interests.splice(2,1)
db.people.update({"name":"dannie"}, {"$set" : {"interests" : interests}});
```

Is there a more direct way?

There is no straight way of pulling/removing by array index. In fact, this is an opened issue <https://jira.mongodb.org/browse/SERVER-1014> , you may vote for it.

The workaround is using \$unset and then \$pull:

```
db.lists.update({}, {$unset : {"interests.3" : 1 }})
db.lists.update({}, {$pull : {"interests" : null}})
```

Update: as mentioned in some of the comments this approach is not atomic and can cause some race conditions if other clients read and/or write between the two operations. If we need the operation to be atomic, we could:

1. Read the document from the database
2. Update the document and remove the item in the array
3. Replace the document in the database. To ensure the document has not changed since we read it, we can use the update if current pattern described in the mongo docs.

You can use \$pull modifier of update operation for removing a particular element in an array. In case you provided a query will look like this:

```
db.people.update({"name":"dannie"}, {'$pull': {"interests": "guitar"}})
```

Also, you may consider using \$pullAll for removing all occurrences. More about this on the official documentation page – <https://www.mongodb.org/display/DOCS/Updating#Updating-%24pull>

This doesn't use index as a criteria for removing an element, but still might help in cases similar to yours. IMO, using indexes for addressing elements inside an array is not very reliable since mongodb isn't consistent on an elements order as far as I know.

Q42) MongoDB custom and unique IDs


I'm using MongoDB, and I would like to generate unique and cryptical IDs for blog posts (that will be used in restful URLs) such as s52ruf6wst or xR2ru286zjl.



What do you think is best and the more scalable way to generate these IDs ?

I was thinking of following architecture :

1. A periodic (daily?) batch running to generate a lot of random and unique IDs and insert them in a dedicated MongoDB collection with InsertIfNotPresent
2. And each time I want to generate a new blog post, I take an ID from this collection and mark it as "taken" with UpdateIfCurrent atomic operation

WDYT ?

This is exactly why the developers of MongoDB constructed their ObjectId's (the _id) the way they did ... to scale across nodes, etc. 

A BSON ObjectId is a 12-byte value consisting of a 4-byte timestamp (seconds since epoch), a 3-byte  Call us on  Drop us a Query

machine id, a 2-byte process id, and a 3-byte counter. Note that the timestamp and counter fields must be stored big endian unlike the rest of BSON. This is because they are compared byte-by-byte and we want to ensure a mostly increasing order. Here's the schema:

```
0123 456 78 91011
```

```
time machine pid inc
```

Traditional databases often use monotonically increasing sequence numbers for primary keys. In MongoDB, the preferred approach is to use Object IDs instead. Object IDs are more synergistic with sharding and distribution.

<https://www.mongodb.org/display/DOCS/Object+IDs>

So I'd say just use the ObjectID's

They are not that bad when converted to a string (these were inserted right after each other) ...

For example:

```
4d128b6ea794fc13a8000001
```

```
4d128e88a794fc13a8000002
```

They look at first glance to be "guessable" but they really aren't that easy to guess ...

```
4d128 b6e a794fc13a8000001
```

```
4d128 e88 a794fc13a8000002
```

And for a blog, I don't think it's that big of a deal ... we use it production all over the place.

What about using UUIDs?

<https://www.famkruihof.net/uuid/uuidgen> as an example.

Q43) MongoDB shell run() function

```
run("/test.js")
```

in mongo shell. i always get error

```
run("/test.js") shell: started program /test.js sh2516| Unable to start program /test.js errno:13 Permission denied 255
```

for test i add 777 permission on this file i doesn't have any passwords on DB

test.js is empty

What am I doing wrong?

try using load() rather than run(). load is for javascript while run is for system binaries.

Q44) "system." In a Collection Name in MongoDB

I just discovered a bizarre behavior exhibited by MongoDB.

Apparently, any collection name with the string "system." anywhere in it will just not function correctly.

To make matters worse, it won't even tell you anything is wrong!

It's really more a matter of curiosity, but does anybody have any idea why this would happen? Is it documented somewhere?

My assumption is that it uses "system.*" collections to store things internally (like indexes) and doesn't want you messing with them, but this doesn't seem like the correct behavior to me. ^

You are correct "system.*" is a reserved collection namespace used by MongoDB in each DB.

It is used to store indexes and users, etc.

 Drop us a Query

SQL Server has many such tables too, and I don't believe they warn you not to use them either :)

But you could always put in a request for such functionality: <https://jira.mongodb.org/>

You can see them by running ...

```
> show collections
```

and you'll see something like ...

```
system.indexes
```

```
system.users
```

So, you can see your indexes for example:

```
> db.system.indexes.find()
```

From the MongoDB docs:

The **.system.*** namespaces in MongoDB are special and contain database system information. System collections include:

1. **system.namespaces** lists all namespaces.
2. **system.indexes** lists all indexes.
3. Additional namespace / index metadata exists in the database.ns files, and is opaque.
4. **system.profile** stores database profiling information.
5. **system.users** lists users who may access the database.
6. **local.sources** stores replica slave configuration data and state.
7. Information on the structure of a stored object is stored within the object itself. See BSON .

There are several restrictions on manipulation of objects in the system collections. Inserting in system.indexes adds an index, but otherwise that table is immutable (the special drop index command updates it for you). **system.users** is modifiable. **system.profile** is droppable.

<https://docs.mongodb.org/manual/reference/system-collections/>

Q45) MongoDB Shell – access collection with period in name?

I have found a collection in one of our MongoDB databases with the name my.collection.

Is there a way to access this collection from the MongoDB shell, despite it having a point in the name?

```
> db.my.collection.findOne();
```

```
null
```

I'm pretty sure that that is not correct.

try this instead:

```
db["my.collection"].findOne();
```

you run into the same issue with hyphens or any other name that does not match on [a-zA-Z_\$][0-9a-zA-Z_\$]

This limitation comes from valid named for javascript object properties.

if collection name is "my.collection"

```
db.my.collection.findOne(); // OK
```

```
null
```

if collection name is "my.1.collection"

```
db.my.1.collection.findOne(); // Not OK
```

☎ Call us on



✉ Drop us a Query

SyntaxError: missing ; before statement

Fix:

```
db["my.1.collection"].findOne(); // Now is OK
```

null

Q46) How do I insert a record from one mongo database into another?

As far as I see all commands operate on the same database in mongodb. I want to do something like this:

```
db.mySourceCollection.find().forEach( function(x){ db.theDestinationCollection.save(x)} );
```

where mySourceCollection is on liveDatabase and theDestinationCollection is on testDatabase.

Use use :-)

```
> var documents = db.mySourceCollection.find()
```

```
> use testDatabase
```

switched to db testDatabase

```
> documents.forEach(function(x){ db.theDestinationCollection.insert(x) })
```

db is used to refer to the currently connected database, however you can switch databases on the fly using the use command, as I've shown above.

Check out the help command in the shell — it mentions this command and much more!

use dbname doesn't work in scripted mode (i.e. when scripting the shell with javascript), so you should use the db.getSiblingDB() method instead to reassign the 'db' variable, e.g.:

```
db = db.getSiblingDB("otherdb")
```

More info here: <https://www.mongodb.org/display/DOCS/Scripting+the+shell>

Related Page: MongoDB Vs. CouchDB (<https://mindmajix.com/mongodb-vs-couchdb>)

Q47) find inside a hash mongodb

I have this struct in my collection:

```
{foo : 1, bar : 4, baz : {a : 1, b : 2 ,c : "fafofu"}}
```

How do I find "a" and "b" inside baz ? It does not works db.my_collection.find({baz : {a : 1, b : 2}});

I don't care about if "c" is "fafofu" or "cacocu" does not matters.

You can use . to reach into the baz object.

```
db.my_collection.find({"baz.a" : 1, "baz.b" : 2});
```

Q48) Query IDE for MongoDB?

I'm wondering if there is an IDE for MongoDB that allows you to run queries and see the results? This would behave like query analyzer in SQL Server Management Studio. The issue I'm having right now is that I have to do queries, such as "db.MyTable.find()" from command prompt, which isn't a good solution.

If the answer is no, is there a more mature "no sql" solution like MongoDB that does have an IDE?

Web Based

For PHP, I'd recommend Rock Mongo. Solid, lots of great features, easy setup.

https://code.google.com/p/rock-php/wiki/rock_mongo

✉ Drop us a Query

If you don't want to install anything ... you can use MongoHQ's web interface (even if you your MongoDB isn't on MongoHQ.)

<https://mongohq.com/home>

Windows

By far, the best UI (for Windows) currently out there is MongoVUE.

<https://blog.mongovue.com/>

Looks great, lots of features, and if you are new it will really help you get going ...

<https://blog.mongovue.com/features/>

Here's a Q&A with the author too if you are interested ...

<https://learnmongo.com/posts/qa-ishann-kumar-creator-of-mongovue/>

There is an official list of admin tools here: <https://www.mongodb.org/display/DOCS/Admin+UIs>

Another contender : <https://www.robomongo.org/> Robomongo give you a shell like interface but outputs your results in the gui. Its available for windows, mac(dmg, zip) and linux (deb, rpm, tar.gz) as a desktop application. Currently its free!

Robomongo prints the results in a treeView or Json text representation and supports the generation of UUID (.NET-,Python-,Java-Encoding). It has autocomplete, shows multiple results at once and has a query history.

Q49) MongoDB: Getting "Client Cursor::yield can't unlock b/c of recursive lock" warning when use findAndModify in two process instances

I'm using: MongoDB 1.6.4, Python 2.6.6, PyMongo 1.9, Ubuntu 10.10

I'm getting "Client Cursor::yield can't unlock b/c of recursive lock" warning in my logs very often when use findAndModify in two process instances. When i use only one process warning doesn't appear.

How can i fix this?

*Update 8 March 2013 *

Is there a fix to this problem as of now?

this is usually means you are missing indexes on fields used in query.

I don't know tech details of this warning but from my experience adding index on the query field helps. check you have index on fields that used in query part of findAndModify. also run `db.collection.find().explain()` to check if it uses the index.

Thanks to the pingw33n who help solve this question.

Q50) how to query child objects in mongodb

I'm new to mongodb and am trying to query child objects. I have a collection of States, and each State has child Cities. One of the Cities has a Name property that is null, which is causing errors in my app. How would I query the State collections to find child Cities that have a name == null?

If it is exactly null (as opposed to not set):

```
db.states.find({"cities.name": null})
```

(but as javierfp points out, it also matches documents that have no cities array at all, I'm assuming that they do).

If it's the case that the property is not set:

 Drop us a Query

```
db.states.find({"cities.name": {"$exists": false}})
```

I've tested the above with a collection created with these two inserts:

```
db.states.insert({"cities": [{name: "New York"}, {name: null}]})
```

```
db.states.insert({"cities": [{name: "Austin"}, {color: "blue"}]})
```

The first query finds the first state, the second query finds the second. If you want to find them both with one query you can make an \$or query:

```
db.states.find({"$or": [
{"cities.name": null},
{"cities.name": {"$exists": false}}
]})
```

Assuming your "states" collection is like:

```
{"name": "Spain", "cities": [ { "name": "Madrid" }, { "name": null } ] }
{"name": "France" }
```

The query to find states with null cities would be:

```
db.states.find({"cities.name": {"$in": [null], "$exists": true}});
```

It is a common **mistake** to query for nulls as:

```
db.states.find({"cities.name": null});
```

because this query will return all documents lacking the key (in our example it will return Spain and France). So, unless you are sure the key is always present you must check that the key exists as in the first query.

Q51) MongoDB 1.6.5: how to rename field in collection

\$rename function is available only in development version 1.7.2. How to rename field in 1.6.5?

The simplest way to perform such an operation is to loop through the data set re-mapping the name of the field. The easiest way to do this is to write a function that performs the re-write and then use the .find().forEach() syntax in the shell.

Here's a sample from the shell:

Skip code block

```
db.foo.save({ a : 1, b : 2, c : 3});
db.foo.save({ a : 4, b : 5, c : 6});
db.foo.save({ a : 7, b : 8 });
db.foo.find();
remap = function (x) {
if (x.c){
db.foo.update({_id:x._id}, {$set:{d:x.c}, $unset:{c:1}});
}
}
db.foo.find().forEach(remap);
db.foo.find();
```

In the case above I'm doing an \$unset and a \$set in the same action. MongoDB does not support

☎ Call us on

✉ Drop us a Query

transactions across collections, but the above is a single document. So you're guaranteed that the set and unset will be atomic (i.e. they both succeed or they both fail).

The only limitation here is that you'll need to manage outside writers to keep the data consistent. My normal preference for this is simply to turn off writes while this updates. If this option is not available, then you'll have to figure out what level of consistency you want for the data. (I can provide some ideas here, but it's really going to be specific to your data and system)

```
db.collection_name.update({}, {$rename: {"oldname": "newname"}}, false, true);
```

This will rename the column for each row in the collection.

Also, I discovered that if your column (the one you're renaming) appears within the index catalog (db.collection_name.getIndexes()), then you're going to have to drop and recreate the index (using the new column name) also.

Q52) Is it normal size for MongoDB?

I just import my **Mysql** database (size 2,8 Mio) in my new **Mongo** database with very simple php script i build, import was ok without error but when i look my **Mongo** database (with RockMongo) i can see this : Data Size 8.01m, Storage Size 13.7m.

MongoDB is bigger than **Mysql** for the same amount of data, is this normal ?

Thanks for your help and sorry for my english.

Yes, it's normal that the "same" data will take up more space in mongodb. There's a few things you need to take into account:

- 1) the document `_id` that's stored for each document (unless you are specifying your own value for that) is 12 bytes per doc
- 2) you're storing the key for each key-value pair in each document, whereas in MySQL the column name is not stored for every single row so you have that extra overhead in your mongodb documents too. One way to reduce this is to use shortened key names ("column names") in your docs
- 3) mongodb automatically adds padding to allow documents to grow

In similar tests, loading data from SQL Server to MongoDB, with shortened 2 character document key names instead of the full names as per SQL Server, I see about 25-30% extra space being used in MongoDB.

Q53) Maximum number of databases supported by MongoDB

I would like to create a database for each customer. But before, I would like to know how many databases can be created in a single instance of MongoDB ?

There's no explicit limit, but there are probably some implicit limits due to max number of open file handles / files in a directory on the host OS/filesystem.

see: https://groups.google.com/group/mongodb-user/browse_thread/thread/01727e1af681985a?fwc=2

Q54) Mongo \$in operator performance

Is it slow / poor form to use the \$in operator in MongoDB with a large array of possibilities?

```
posts.find({
```

author: {



✉ Drop us a Query


```
$in : ['friend1','friend2','friend3'....'friend40']
}
})
```

App Engine, for example, won't let you use more than 30 because they translate directly to one query per item in the IN array, and so instead force you into using their method for handling fan out. While that's probably the most efficient method in Mongo too, the code for it is significantly more complex so I'd prefer to just use this generic method.

Will Mongo execute these \$in queries efficiently for reasonable-sized datasets?

It can be fairly efficient with small lists (hard to say what small is, but at least into the tens/hundreds) for \$in. It does not work like app-engine since mongodb has actual btree indexes and isn't a column store like bigtable.

With \$in it will skip around in the index to find the matching documents, or walk through the whole collection if there isn't an index to use.

If you build an index (ensureIndex) on the list element, it should be pretty quick.

Have you tried using explain()? It's a good, built-in way to profile your queries:
<https://www.mongodb.org/display/DOCS/Indexing+Advice+and+FAQ#IndexingAdviceandFAQ-Use%7B%7Bexplain%7D%7D>.

Q55) Mongo Query question \$gt,\$lt

I have a query below. I want get items between 4 and 6 so only a:1 should match because it has the value 5 in b.

```
> db.test.find({ b : { $gt : 4 }, b: {$lt : 6}});
{ "_id" : ObjectId("4d54cff54364000000004331"), "a" : 1, "b" : [ 2, 3, 4, 5 ] }
{ "_id" : ObjectId("4d54d0074364000000004332"), "a" : 2, "b" : [ 2, 4, 6, 8 ] }
>
```

Can someone tell me why a:2 is matching this query? I can't really see why it is being returned.

I also tried what was specified in the tutorial but it did not seem to work:

```
> db.test.find({ b : { $gt : 4, $lt : 6}});
{ "_id" : ObjectId("4d54cff54364000000004331"), "a" : 1, "b" : [ 2, 3, 4, 5 ] }
{ "_id" : ObjectId("4d54d0074364000000004332"), "a" : 2, "b" : [ 2, 4, 6, 8 ] }
>
```

And this one to avoid any confusion regarding GT/GTE

```
> db.test.find({b: {$gt: 4.5, $lt: 5.5}});
{ "_id" : ObjectId("4d54cff54364000000004331"), "a" : 1, "b" : [ 2, 3, 4, 5 ] }
{ "_id" : ObjectId("4d54d0074364000000004332"), "a" : 2, "b" : [ 2, 4, 6, 8 ] }
>
```

only a:1 should be returned.

As suggested, I gave \$elemMatch a try but it did not appear to work either (objectIds are different because I am on a different machine)

```
> db.test.find();
📞 Call us on
```

✉ Drop us a Query

```
{ "_id" : ObjectId("4d5a24a5e82e00000000433f"), "a" : 1, "b" : [ 2, 3, 4, 5 ] }
{ "_id" : ObjectId("4d5a24bbe82e000000004340"), "a" : 2, "b" : [ 2, 4, 6, 8 ] }
> db.test.find({b: {$elemMatch: {$gt : 4, $lt: 6 }}});
>
```

No documents were returned.

This is a really confusing topic. I work at 10gen and I had to spend a while wrapping my head around it. Let's walk through how the query engine processes this query.

Here's the query again:

```
> db.test.find({ b : { $gt : 4, $lt : 6 }});
```

When it gets to the record that seems like it shouldn't match...

```
{ "_id" : ObjectId("4d54cff54364000000004331"), "a" : 1, "b" : [ 2, 4, 6, 8 ] }
```

The match is not performed against each element of the array, but rather against the array as a whole.

The comparison is performed in three steps:

Step 1: Find all documents where b has a value greater than 4

b: [2,4,6,8] matches because 6 & 8 are greater than 4

Step 2: Find all documents where b has a value less than 6

b: [2,4,6,8] matches because 2 & 4 are less than 6

Step 3: Find the set of documents that matched in both step 1 & 2.

The document with b: [2,4,6,8] matched both steps 1 & 2 so it is returned as a match. Note that results are also de-duplicated in this step, so the same document won't be returned twice.

If you want your query to apply to the individual elements of the array, rather than the array as a whole, you can use the \$elemMatch operator. For example

```
> db.temp.find({b: {$elemMatch: {$gt: 4, $lt: 5}}})
> db.temp.find({b: {$elemMatch: {$gte: 4, $lt: 5}}})
{ "_id" : ObjectId("4d558b6f4f0b1e2141b66660"), "b" : [ 2, 3, 4, 5, 6 ] }
```

Related Page: MongoDB Show Collections (<https://mindmajix.com/mongodb-show-collections>)

Q56) MongoDB: How to change the type of a field?

There is a question already in Stackoverflow, very similar with my question. The thing is that the answer for that questions was for a Java Driver, I am trying to do it in the shell.

I am doing this...

```
db.meta.update({'fields.properties.default': { $type : 1 }}, {'fields.properties.default': { $type : 2 }})
```

This is not working!

The only way to change the \$type of the data is to perform an update on the data where the data has the correct type.

In this case, it looks like you're trying to change the \$type from 1 (double) to 2 (string).

<https://www.mongodb.org/display/DOCS/Advanced+Queries#AdvancedQueries-%24type>

So simply load the document from the DB, perform the cast (new String(x)) and then save the document again. ^

If you need to do this programmatically and entirely from the shell, you can use [Call us on](#) [Drop us a Query](#)

theFind(...).forEach(function(x) {})) syntax.

In response to the second comment below. Change the field bad from a number to a string in collection foo.

```
db.foo.find( { 'bad' : { $type : 1 } } ).forEach( function (x) {
x.bad = new String(x.bad); // convert field to string
db.foo.save(x);
});
// String to Integer
db.db-name.find({field-name : {$exists : true}}).forEach( function(obj) { obj.field-name = new
NumberInt( obj.field-name ); db.db-name.save(obj); } );
// Integer to String
db.db-name.find({field-name : {$exists : true}}).forEach( function(obj) { obj.field-name = ""+obj.field-
name; db.db-name.save(obj); } );
This is what I used for string to int conversion.
db.my_collection.find().forEach( function(obj) {
obj.my_value= parseInt(obj.my_value);
db.my_collection.save(obj);
});
db.coll.find().forEach(function(data) {
db.coll.update({_id:data._id},{ $set:{myfield:parseInt(data.myfield)}});
})
```

Q57) Many to many update in MongoDB without transactions

I have two collections with a many-to-many relationship. I want to store an array of linked ObjectIds in both documents so that I can take Document A and retrieve all linked Document B's quickly, and vice versa.

Creating this link is a two step process

1. Add Document A's ObjectId to Document B
2. Add Document B's ObjectId to Document A

After watching a MongoDB video I found this to be the recommended way of storing a many-to-many relationship between two collections

I need to be sure that both updates are made. What is the recommended way of robustly dealing with this crucial two step process without a transaction?

I could condense this relationship into a single link collection, the advantage being a single update with no chance of Document B missing the link to Document A. The disadvantage being that I'm not really using MongoDB as intended. But, because there is only a single update, it seems more robust to have a link collection that defines the many-to-many relationship.

Should I use safe mode and manually check the data went in afterwards and try again on failure? Or should I represent the many-to-many relationship in just one of the collections and rely on an index[↗] to make sure I can still quickly get the linked documents?

☎ Call us on

✉ Drop us a Query

Any recommendations? Thanks

@**Gareth**, you have multiple legitimate ways to do this. So the key concern is how you plan to query for the data, (i.e.: what queries need to be fast)

Here are a couple of methods.

Method #1: the “links” collection

You could build a collection that simply contains mappings between the collections.

Pros:

1. Supports atomic updates so that data is not lost

Cons:

2. Extra query when trying to move between collections

Method #2: store copies of smaller mappings in larger collection

For example: you have millions of Products, but only a hundred Categories. Then you would store the Categories as an array inside each Product.

Pros:

1. Smallest footprint
2. Only need one update

Cons:

1. Extra query if you go the “wrong way”

Method #3: store copies of all mappings in both collections

(what you’re suggesting)

Pros:

1. Single query access to move between either collection

Cons:

1. Potentially large indexes
2. Needs transactions (?)

Let’s talk about “needs transactions”. There are several ways to do transactions and it really depends on what type of safety you require.

Should I use safe mode and manually check the data went in afterwards and try again on failure?

You can definitely do this. You’ll have to ask yourself, what’s the worst that happens if only one of the saves fails?

Method #4: queue the change

I don’t know if you’ve ever worked with queues, but if you have some leeway you can build a simple queue and have different jobs that update their respective collections.

This is a much more advanced solution. I would tend to go with #2 or #3.

Q58) MongoDB log file growth

Currently my log file sits at 32 meg. Did I miss an option that would split the log file as it grows?

Rotate the logs yourself

<https://www.mongodb.org/display/DOCS/Logging>

or use ‘logrotate’ with an appropriate configuration.

☎ Call us on

✉ Drop us a Query



You can use logrotate to do this job for you.

Put this in /etc/logrotate.d/mongod (assuming you use Linux and have logrotated installed):

Skip code block

```
/var/log/mongo/*.log {
daily
rotate 30
compress
dateext
missingok
notifempty
sharedscripts
copytruncate
postrotate
/bin/kill -SIGUSR1 `cat /var/lib/mongo/mongod.lock 2> /dev/null` 2> /dev/null || true
endscript
}
```

If you think that 32 megs is too large for a log file, you may also want to look inside to what it contains.

If the logs seem mostly harmless (“open connection”, “close connection”), then you may want to start mongod with the -quiet switch. This will reduce some of the more verbose logging.

Q59) MongoDB field order and document position change after update

I am learning MongoDB and I noticed that whenever I do an update on a document the field being updated is pushed to the end of the order, so if I had something like:

```
db.collection.save({field1: value1, field2: value2, ..., field 10: value10});
```

```
db.collection.update({field1: value1}, {$set: {field2: new_value}});
```

then if you do:

```
db.collection.find();
```

it will display:

```
{ "field1?:"value1", ..., "field10?:"value10", "field2?:"new_value"}
```

You can see how the field order changes where the updated field is being pushed to the end of the document. In addition, the document itself is being pushed to the end of the collection. I know that it's a “schema-less” DB and it may not be a huge problem, but it just doesn't look “pretty” :). Is there a way to do an in-place update without changing the order?

MongoDB allocates space for a new document based on a certain padding factor. If your update increases the size of the document beyond the size originally allocated the document will be moved to the end of the collection. The same concept applies to fields in a document.

Both document structure and collection structure in MongoDB based on JSON principles. JSON is a **set** of key/value pairs (in particular fieldName/fieldValue for document and index/document for collection).

From this point of view it doesn't seem that you can rely on order at all. ^

In the case of the documents if the field size changes, it writes out a new document with the fields

☎ Call us on

✉ Drop us a Query

sorted by field name. This behavior can be seen with the following statements

Case 1: No change in size of field, so no change in field order

```
db.testcol.find()      db.testcol.save({a:1,c:3,b:2})      db.testcol.find()      {      "_id"      :
ObjectId("4d5efc3bec5855af36834f5a"), "a" : 1, "c" : 3, "b" : 2 } db.testcol.update({a:1},{set:{c:22}})
db.testcol.find() { "_id" : ObjectId("4d5efc3bec5855af36834f5a"), "a" : 1, "c" : 22, "b" : 2 }
```

Case 2: Field size changes and the fields are reordered

```
db.testcol.find()      db.testcol.save({a:1,c:"foo",b:2,d:4})      db.testcol.find()      {      "_id"      :
ObjectId("4d5efdceec5855af36834f5e"), "a" : 1, "c" : "foo", "b" : 2, "d" : 4 } db.testcol.update({a:1},{set:
{c:"foobar"}}) db.testcol.find() { "_id" : ObjectId("4d5efdceec5855af36834f5e"), "a" : 1, "b" : 2, "c" :
"foobar", "d" : 4 }
```

Is there a particular reason why you do not want the fields reordered? The above was using 1.8.0_rc0 on OS X

FYI, in MongoDB 2.6 updates will preserve field order, with the following exceptions:

1. The `_id` field is always the first field in the document.
2. Updates that include renaming of field names may result in the reordering of fields in the document.

Related Page: MongoDB Find Queries (<https://mindmajix.com/mongodb-find-queries>)

Q60) Limit MongoDB database size?

We're interested in deploying MongoDB, but we need to know if we can limit database/table sizes?

For example:

```
db.user1.find() db.user2.find()
```

As you can see from the above, each user will have their own database. We want to limit each user's database so we don't have any one user eating up all our hard drive space.

Is this possible with MongoDB?

Thanks.

You can create a database per user and enable the `-quota` option. This will allow you keep any user from using too much space.

<https://www.mongodb.org/display/DOCS/Command+Line+Parameters>

<https://www.mongodb.org/display/DOCS/Excessive+Disk+Space>

Q61) While Creating New User in MongoDB, it has to be created under admin?

I want to create a new user in MongoDB, So i have to do that by login to admin by (use admin) and then i have to use the command add user to create new user is it?

Thanks,

If no users created you can create new user without any authentication, but if you have created admin user for specific database you should authenticate, and then perform any operation.

Documentation:

If no users are configured in `admin.system.users`, one may access the database from the localhost interface without authenticating. Thus, from the server running the database (and thus on localhost), run the database shell and configure an administrative user:

Call us on

✉ Drop us a Query

```
$ ./mongo
```

```
> use admin
```

```
> db.addUser("theadmin", "anadminpassword")
```

We now have a user created for database admin. Note that if we have not previously authenticated, we now must if we wish to perform further operations, as there is a user in admin.system.users.

```
> db.auth("theadmin", "anadminpassword")
```

We can view existing users for the database with the command:

```
> db.system.users.find()
```

Now, let's configure a "regular" user for another database.

```
> use projectx
```

```
> db.addUser("joe", "passwordForJoe")
```

Q62) When to embed documents in Mongo DB

I'm trying to figure out how to best design Mongo DB schemas. The Mongo DB documentation recommends relying heavily on embedded documents for improved querying, but I'm wondering if my use case actually justifies referenced documents.

A very basic version of my current schema is basically: (Apologies for the psuedo-format, I'm not sure how to express Mongo schemas)

Skip code block

```
users {
  email (string)
}
games {
  user (reference user document)
  date_started (timestamp)
  date_finished (timestamp)
  mode (string)
  score: {
    total_points (integer)
    time_elapsed (integer)
  }
}
```

Games are short (about 60 seconds long) and I expect a lot of concurrent writes to be taking place.

At some point, I'm going to want to calculate a high score list, and possibly in a segregated fashion (e.g., high score list for a particular game.mode or date)

Is embedded documents the best approach here? Or is this truly a problem that relations solves better?

How would these use cases best be solved in Mongo DB?

... is this truly a problem that relations solves better?

The key here is less about "is this a relation?" and more about "how am I going to access this?"

MongoDB is not "anti-reference". MongoDB does **not** have the benefits of joins, but it **does** have the

☎ Call us on

✉ Drop us a Query



benefit of embedded documents.

As long as you understand these trade-offs then it's perfectly fair to use references in MongoDB. It's really about how you plan to query these objects.

Is embedded documents the best approach here?

Maybe. Some things to consider.

1. Dogames have value outside of the context of the user?
2. How many games will a single user have?
3. Is games transactional in nature?
4. How are you going to access games? Do you always need all of a user's games?

If you're planning to build leaderboards and a user can generate hundreds of game documents, then it's probably fair to have games in their own collection. Storing ten thousand instances of "game" inside of each users isn't particularly useful.

But depending on your answers to the above, you could really go either way. As the litmus test, I would try running some Map / Reduce jobs (i.e. build a simple leaderboard) to see how you feel about the structure of your data.

Q63) Why Is MongoDB So Fast

I was showing my co-worker performance benchmarks of MongoDB vs SQL 2008 and while he believes MongoDB is faster, he doesn't understand how its possible. His logic, was that SQL has been around for decades, and has some of the smartest people working on it, and how can MongoDB; a relatively new kid on the block be so superior in performance? I wasn't able to really provide a solid and technical answer, and I was hoping you guys could assist.

MongoDB isn't like a traditional relational database. It's noSQL or document based, it provides weak consistency guarantees, and it doesn't have to guarantee consistency like SQL.

MongoDB is fast because its web scale!

Its a fun video and well worth everyone watching, but it does answer your question - that most of the noSQL engines like MongoDB are not robust and not resilient to crashes and other outages. This security is what they sacrifice to gain speed.

SQL has to do quite a lot, Mongo just has to drop bits onto disk (almost)

Q64) How to remove an element from a doubly-nested array in a MongoDB document

I have a document structure something along the lines of the following:

```
{
  "_id" : "777",
  "someKey" : "someValue",
  "someArray" : [
    {
      "name" : "name1",
      "someNestedArray" : [
        {
          "name" : "someValue"
```



Call us on

✉ Drop us a Query


```

    },
    {
      "name" : "delete me"
    }
  ]
}
]
}
}

```

I want to delete the nested array element with the value "delete me".

I know I can find documents which match this description using nested \$elemMatch expressions. What is the query syntax for removing the element in question?

To delete the item in question you're actually going to use an update. More specifically you're going to do an update with the \$pull command which will remove the item from the array.

```
db.temp.update({ _id : "777" }, { $pull : { "someArray.0.someNestedArray" : { "name": "delete me" } } } )
```

There's a little bit of "magic" happening here. Using .0 indicates that we know that we are modifying the 0th item of someArray. Using { "name": "delete me" } indicates that we know the exact data that we plan to remove.

This process works just fine if you load the data into a client and then perform the update. This process works less well if you want to do "generic" queries that perform these operations.

I think it's easiest to simply recognize that updating arrays of sub-documents generally requires that you have the original in memory at some point.

In response to the first comment below, you can probably help your situation by changing the data structure a little

Skip code block

```

"someObjects" : {
  "name1": {
    "someNestedArray" : [
      {
        "name" : "value"
      },
      {
        "name" : "delete me"
      }
    ]
  }
}
}

```

Now you can do { \$pull : { "someObjects.name1.someNestedArray" : ...

Here's the problem with your structure. MongoDB does not have very good support for manipulating "sub-arrays". Your structure has an array of objects and those objects contain arrays of more objects. ^

If you have the following structure, you are going to have a difficult time using things like \$pull:

Call us on

Drop us a Query

```
array [
{ subarray : array [] },
{ subarray : array [] },
]
```

If your structure looks like that and you want to update subarray you have two options: 1. Change your structure so that you can leverage \$pull. 2. Don't use \$pull. Load the entire object into a client and use findAndModify.

Related Page: MongoDB Port (<https://mindmajix.com/mongodb-port>)

Q65) Can MongoDB work when size of database larger then RAM? or when index larger then RAM?

Yes it can work. To what level it will perform is more of an "It Depends"

The key thing is to ensure your **working set** can fit in RAM. So if you have 16GB of RAM and 20GB database (inc. indexes) for example, if you need to only access half of all the data as the other half is older/never actually queried then you'll be fine as only half of your database needs to be in RAM (10GB). Working set is key here. For example, if you have a logging application outputting to MongoDB, it may be that your working set is the amount of data (and indexes) from the past 3 months and that all data before that you don't access.

When your working set exceeds the amount of RAM, then it will carry on working but with noticeably degraded performance as things will then be constantly having to go to disk which is far less performant. If you're in this situation of exceeding RAM constraints on a machine, then this is where sharding comes into play – so you can balance the data out over a number of machines therefore increasing the amount of data that can be kept in RAM.

Q66) Multiple \$inc updates in MongoDB

Is it possible to update a single document by passing two \$inc operators in a single update document?

For example, I am trying to increment two different fields in a given document using the following update document:

```
{
"$inc" : { "ViewAggregates.4d75b891842f2d3930cf7674" : 1 },
"$inc" : { "ViewAggregates.Total" : 1 }
}
```

No errors are thrown and the document is updated but only one of the fields has been incremented. It is as if the server disregarded the first \$inc operator and only the second was actually applied.

Is this the intended/correct behavior or is there something I am missing?

This is an interesting side-effect of dictionary keys being unique — the second \$inc overwrites the first. However, it's still possible to increment more than one field:

```
{
"$inc": {
"ViewAggregates.4d75b891842f2d3930cf7674" : 1,
"ViewAggregates.Total" : 1
}
```

📞 Call us on

✉ Drop us a Query

}

}

This works for many other operators too :-)

Q67) MongoDB : Indexes order and query order must match?

This question concern the internal method to manage indexes and serching Bson Documents.

When you create a multiple indexes like "index1", "index2", "index3?...the index are stored to be used during queries, but what about the order of queries and the performance resulting.

sample

index1,index2,index3--> query in the same order index1,index2,index3 (best case) index1,index2,index3

--> query in another order index2,index1,index3 (the order altered)

Many times you use nested queries including these 3 index and others items or more indexes. The order of the queries would implicate some time lost?. Must passing the queries respecting the indexes order defined or the internal architecture take care about this order search? I searching to know if i do take care about this or can make my queries in freedom manier.

Thanks.

The order of the conditions in your query does not affect whether it can use an index or no.

e.g. typical document structure:

```
{
  "FieldA" : "A",
  "FieldB" : "B"
}
```

If you have an compound index on A and B :

```
db.MyCollection.ensureIndex({FieldA : 1, FieldB : 1})
```

Then both of the following queries will be able to use that index:

```
db.MyCollection.find({FieldA : "A", FieldB : "B"})
```

```
db.MyCollection.find({FieldB : "B", FieldA : "A"})
```

So the ordering of the conditions in the query do not prevent the index being used - which I think is the question you are asking.

You can easily test this out by trying the 2 queries in the shell and adding .explain() after the find. I just did this to confirm, and they both showed that the compound index was used.

however, if you run the following query, this will NOT use the index as FieldA is not being queried on:

```
db.MyCollection.find({FieldB : "B"})
```

So it's the ordering of the fields in the index that defines whether it can be used by a query and not the ordering of the fields in the query itself (this was what Lucas was referring to).

From <https://www.mongodb.org/display/DOCS/Indexes>:

If you have a compound index on multiple fields, you can use it to query on the beginning subset of fields. So if you have an index on

a,b,c

you can use it query on

☎ Call us on

✉ Drop us a Query

a
a,b
a,b,c

So yes, order matters. You should clarify your question a bit if you need a more precise answer.

Q68) MongoDB :: are Mongoids unique across collections?

I was wondering: can Mongo IDs have the same value in different collections in the same database?

Thank you,

The uniqueness constraint for `_id` is per collection, so yes – one and the same ID can occur once per Collection.

It's however very unlikely, if not impossible, for the same ID to be generated twice. So in order for this to happen you would have to manually insert duplicate IDs.

Q69) like query in mongoDB

I am working on mongodb . In which i Want to use like query. my collection structure is as follows.

```
{ "name" : "xy" , "age" : 34 , "location" : "sss" }
{ "name" : "xyx" , "age" : 45 , "location" : "sshs" }
{ "name" : "x" , "age" : 33 , "location" : "shhss" }
{ "name" : "pq" , "age" : 23 , "location" : "hhh" }
{ "name" : "pqr" , "age" : 12 , "location" : "sss" }
```

i want to find records matching to "name" : "x".

so query will return all three records matching xy ,xyz,x.

Is it possible in mongo.

if any one knows plz reply.

Thanks

You can use regular expressions to do this:

```
db.customers.find( { name : /^x/i } );
```

You will probably want to have some indexes on the name field.

Read more at the MongoDB Documetation site.

Q70) How to check if an array field contains a unique value or another array in MongoDB?

I am using mongodb now.

I have blogpost collection, and blogpost has a tags filed which is an array, e.g.

```
blogpost1.tags = ['tag1', 'tag2', 'tag3', 'tag4', 'tag5']
```

```
blogpost2.tags = ['tag2', 'tag3']
```

```
blogpost3.tags = ['tag2', 'tag3', 'tag4', 'tag5']
```

```
blogpost4.tags = ['tag1', 'tag4', 'tag5']
```

How can I do these searches

contains 'tag1'

contains ['tag1','tag2'],

contains any of ['tag3', 'tag4']



✉ Drop us a Query

Try this out:

```
db.blogpost.find({ 'tags' : 'tag1' }); //1
```

```
db.blogpost.find({ 'tags' : { $all : [ 'tag1', 'tag2' ] } }); //2
```

```
db.blogpost.find({ 'tags' : { $in : [ 'tag3', 'tag4' ] } }); //3
```

My experience is that for (2) the following solution is much faster than the one with "\$all":

```
db.blogpost.find({ $and: [ {tags: 'tag1'} ,{tags: 'tag2'} ] });
```

but to be honest, I do not not why. I would be interested in, if anyone knows.

Q71) create secure database in mongodb

I want to create the database in mongodb and thats secure.

here ..secure mean .application has to pass username/password to connect my database in mongodb.

javaamtho

From Mongo Java Tutorial

MongoDB can be run in a secure mode where access to databases is controlled through name and password authentication. When run in this mode, any client application must provide a name and password before doing any operations. In the Java driver, you simply do the following with the connected mongo object :

```
boolean auth = db.authenticate(myUserName, myPassword);
```

If the name and password are valid for the database, auth will be true. Otherwise, it will be false. You should look at the MongoDB log for further information if available.

Most users run MongoDB without authentication in a trusted environment.

Configuring Authentication and Security

Authentication is stored in each database's system.users collection. For example, on a database projectx, projectx.system.users will contain user information.

We should first configure an administrator user for the entire db server process. This user is stored under the special admin database.

If no users are configured in admin.system.users, one may access the database from the localhost interface without authenticating. Thus, from the server running the database (and thus on localhost), run the database shell and configure an administrative user:

```
$ ./mongo
```

```
> use admin
```

```
> db.addUser("theadmin", "anadminpassword")
```

We now have a user created for database admin. Note that if we have not previously authenticated, we now must if we wish to perform further operations, as there is a user in admin.system.users.

```
> db.auth("theadmin", "anadminpassword")
```

We can view existing users for the database with the command:

```
> db.system.users.find()
```

Now, let's configure a "regular" user for another database.

```
> use projectx
```

```
> db.addUser("joe", "passwordForJoe")
```

🔗 Call us on

✉ Drop us a Query

Finally, let's add a readonly user. (only supported in 1.3.2+)

```
> use projectx
```

```
> db.addUser("guest", "passwordForGuest", true)
```

Related Page: MongoDB Create Collection (<https://mindmajix.com/mongodb-create-collection>)

Q72) How do I copy a database from one MongoDB server to another?

I have two mongodbs in different server, both start with -auth. Now I want to copy a db from one server to another.

```
> mongo
```

```
> use admin
```

```
> db.copyDatabase("mydb","mydb","another_server")
```

It shows:

```
{ "errmsg" : "", "ok" : 0 }
```

And:

```
> db.getLastError()
```

```
null
```

Seems no error, but the copy is not successful. What's the correct command to use?

If you are using **-auth**, you'll need to include your username/password in there...

Also you must be on the "destination" server when you run the command.

```
db.copyDatabase(, , , );
```

If all that doesn't work, you might want to try something like creating a slave of the database you want to copy ...

Q73) Case-insensitive query on mongodb

Is there a way to query for a case-insensitive value on mongo without using map/reduce?

Suppose you have document that contains tag field and you want search on it

Tags

```
{
```

```
  tag,
```

```
  ...
```

```
}
```

First option is use regex(but it work slow as @RestRisiko said):

```
db.tags.find( { "tag" : { "$regex" : "C#", "$options" : "-i" } })
```

Second option is create another, lower case field(and in mongodb it best way):

Tags

```
{
```

```
  tag,
```

```
  tagLower,
```

```
  ..
```

```
}
```

📞 Call us on

And use find as usual:

✉ Drop us a Query



```
db.tags.find( { "tagLower" : "c#" })
```

It will work faster, because above code can use index for search.

You have to normalize the data to be queried. Using a regular expression for case-insensitive search might work as well it won't use indexes. So your only option is to normalize. If you need to preserve the original state then you need to denormalize the data and store the normalized values in a dedicated column of the document.

Q74) MongoDB load balancing

I was looking at best load balancing option for concurrent users with Mongo DB. I have looked at Master Slave replication but don't think this will load balance. Are there any open source DB load balancers for Mongo DB?

I have looked at Sequoia but looks like that project is no longer actively supported.

Please note: The data is not very huge & also not use case for sharding.

both Master Slave and Replica Sets will load balance in MongoDB, if you set slaveOK in your driver.

When slaveOK is enabled MongoDB drivers direct all reads to secondaries/slaves.

This provides relatively effective read balancing; for write balancing your only option would be sharding.

Q75) MongoDB: unconditional updates?

This seems like a silly question but I haven't yet found the answer. If I simply wanted to add the same field->value to EVERY record in a MongoDB collection, what would be the appropriate shell command to do so? I tried doing a multi update with a blank query ({}) but that resulted in this error:

multi update only works with \$ operators

I'm a bit puzzled about how to get around this. Any suggestions?

The error says it all: You can only modify multiple documents using the \$ modifier operators. You probably had something like this:

```
> db.coll.update( { }, { a: 'b' }, false, true);
```

Which would normally replace the first object in the collection with { a: 'b' } if multi was false. You wouldn't want to replace all the objects in your collection with the same document!

Use the \$set operator instead:

```
> db.coll.update( { }, { '$set': { a: 'b' } }, false, true);
```

This will set the a property of every document (creating it as necessary) to 'b'.

Q76) Mongo: find items that don't have a certain field

Is it possible in Mongo to search for documents in a collection that are missing a certain field?

Yeah, it's possible using \$exists:

```
db.things.find( { a : { $exists : false } } ); // return if a is missing
```

If you don't care if the field is missing or null (or if it's never null) then you can use the slightly shorter and safer:

```
db.things.find( { a : null } ); // return if a is missing or null
```

It's safer because \$exists will return true even if the field is null, which often is not the desired result and

can lead to an NPE.

✉ Drop us a Query

Q77) Is there a sample MongoDB Database along the lines of world for MySql?

As someone new to Mongo, I am looking for a sample MongoDB database that I can import and play with. Something along the lines of world for mysql or Northwind for MSSQL.

Is there one? (I couldn't find any reference to one at <https://www.mongodb.org> nor did my googling help)

For Windows users: Please follow the following steps to import from the json file if you are using windows 7:

1. Download the above mentionedJSON file and place it inside a folder (say d:sample)
2. Open a command prompt, start the mongo server by going in to the bin directory and typing in mongoD
3. Now take another command prompt and go to the bin directory again and write following command C:\mongodb\bin>mongoimport -db test -collection zips -file d:\sample\zips.json
4. The import would start working immediatly and at the end it would show some thing like this Thu Dec 19 17:11:22 imported 29470 objects

That's it!

I found this you can import the json data with:

```
mongoimport -db scratch -collection zips -file zips.json
```

I guess that you can import any json data that you find, it also supports csv and tsv. Hope this helps.

This doesn't have everything, but it is a nice step towards getting Northwind on MongoDB:

<https://github.com/tmcnab/northwind-mongo>

Q78) MongoDB query help: \$elemMatch in nested objects

```
> db.test.insert({"a" : { "b" : { "c" : { "d1" : [ "e1" ],
"d2" : [ "e2" ],
"d3" : [ "e3", "e4" ],
"d4" : [ "e5", "e6" ] } } } })
> db.test.find({'a.b.c' : {$exists : true}})
{ "_id" : ObjectId("4daf2ccd697ebaacb10976ec"), "a" : { "b" : { "c" : { "d1" : [ "e1" ], "d2" : [ "e2" ], "d3" : [ "e3", "e4" ], "d4" : [ "e5", "e6" ] } } } }
```

But none of these work:

```
> db.test.find({'a.b': "c"})
> db.test.find({'a.b': {$elemMatch : {"c" : {$exists: true}}})
> db.test.find({'a.b': {$elemMatch : {$elemMatch : {$all : ["e1"]} }}})
```

Suppose I don't know what the values of c and d1...d4 are. Is there a generic way to search the nested-objects' structure for particular values?

I thought that was what \$elemMatch was for.

 Gankyo

 Drop us a Query

I thought that was what \$elemMatch was for...

From the docs: Using the \$elemMatch query operator, you can match an entire document within an array.

This does not sounds like what you're looking for.

Is there a generic way to search the nested-objects' structure for particular values?

It sounds like you want to search "everything in object 'c' for an instance of 'e1'".

MongoDB supports two related features, but the features not quite what you're looking for.

1. Reach into objects, dot notation: `db.test.find({'a.b.c.d1' : 'e1'})`

2. Read through arrays: `db.test.find({'a.b.c.d4' : 'e5'})`

It sounds like you're looking for the ability to do both at the same time. You want to "reach into objects" and "read through arrays" in the same query.

Unfortunately, I do not know of such a feature. You may want to file a feature request for this.

Related Page: MongoDB Create Index (<https://mindmajix.com/mongodb-create-index>)

Q79) MongoDB Cursor Timeouts while doing a lot of writes

We have a cluster of 2 replica Sets, with 3 servers per set. With a single collection being sharded. We also have a quite a few more(8+) collections that we use on a daily basis. With the majority of the data being in the sharded collection with close to 100 Million records in it.

Recently we have added the requirement to obtain 100x the data that we had been getting previously, and we need to write this to mongodb. A daemon has been set in place to perform the writes necessary to keep the database up to date. The script performs at over 200 writes a second, with the majority going to the all separate collections.

With this quantity of writes, we have been unable to perform large reads for analytical purposes. Receiving a combination of Cursor Timeouts client-side and server-side("Cursor Not Found").

We have attempted to do limit/skip schemes on the reads, but the problem persists. What is the best course of action to remedy this as we require both a large amount of writes, with few, but large reads?

Typically, in a case like this you want to start looking at the queries causing the time. Then you want to look at the hardware to see what's being stressed.

1. Are these queries correctly indexed?
2. How big are the indexes? Do they fit in RAM?
3. Can you provide some details on where the bottlenecks are?
4. Are you locked on on IO?
5. Are your processors running at full speed?

Also, is there anything unusual in the logs?

Basically we need to ensure that you have: 1. Correctly built the system to handle the query 2. Correctly provisioned the system to handle the data volumes

 Call us on

 Drop us a Query

Q80) Is moving documents between collections a good way to represent state changes in MongoDB?

I have two collections, one (**A**) containing items to be processed (relatively small) and one (**B**) with those already processed (fairly large, with extra result fields).

Items are read from **A**, get processed and save()'d to **B**, then remove()'d from **A**.

The rationale is that indices can be different across these, and that the “incoming” collection can be kept very small and fast this way.

I've run into two issues with this:

1. if either remove() or save() time out or otherwise fail under load, I lose the item completely, or process it twice
2. if both fail, the side effects happen but there is no record of that

I can sidestep the double-failure case with findAndModify locks (not needed otherwise, we have a process-level lock) but then we have stale lock issues and partial failures can still happen. There's no way to atomically remove+save to different collections, as far as I can tell (maybe by design?)

Is there a Best Practice for this situation?

There's no way to atomically remove+save to different collections, as far as I can tell (maybe by design?)

Yes this is by design. MongoDB explicitly does not provides joins or transactions. Remove + Save is a form of transaction.

Is there a Best Practice for this situation?

You really have two low-complexity options here, both involve findAndModify.

Option #1: a single collection

Based on your description, you are basically building a queue with some extra features. If you leverage a single collection then you use findAndModify to update the status of each item as it is processing.

Unfortunately, that means you will lose this: ...that the “incoming” collection can be kept very small and fast this way.

Option #2: two collections

The other option is basically a two phase commit, leveraging findAndModify.

Take a look at the docs for this here.

Once an item is processed in A you set a field to flag it for deletion. You then copy that item over to B. Once copied to B you can then remove the item from A.

Q81) For MongoDB is it better to reference an object or use a natural String key?

I am building a corpus of indexed sentences in different languages. I have a collection of Languages which have both an ObjectId and the ISO code as a key. Is it better to use a reference to the Language collection or store a key like “en” or “fr”?

I suppose it's a compromise between:

1. ease of referencing the Language
2. object in that collection



4. the size of the data on disk

Any best practices that I should know of?

In the end, it really comes down to personal choice and what will work best for your application.

The only requirement that MongoDB imposes upon `_id` is that it be unique. It can be an ObjectId (which is provided by default), a string, even an embedded document (As I recall it cannot be an Array though). In this case, you can likely guarantee ISO Code is a unique value and it may be an ideal value. You have a 'known' primary key which is also useful in itself by being identifiable, so using that instead of a generated ID is probably a more sensible bet. It also means anywhere you 'reference' this information in another collection you can save the ISO Code instead of an Object ID; those browsing your raw data can immediately identify what information that reference points at.

As an aside:

The two big benefit of ObjectId is that they can be generated uniquely across multiple machines, processes and threads without needing any kind of central sequence tracking by the MongoDB server. They also are stored as a special type in MongoDB that only uses 12 bytes (as opposed to the 24 byte representation of the string version of an ObjectId)

Q82) Does the MongoDB stats() function return bits or bytes?

When using MongoDB's `.stats()` function to determine document size, are the values returned in bits or bytes?

Bytes of course. Unless you pass in a scale as optional argument.

Running the `collStats` command – `db.collection.stats()` – returns all sizes in bytes, e.g.

```
> db.foo.stats()
{
  "size" : 715578011834, // total size (bytes)
  "avgObjSize" : 2862,  // average size (bytes)
}
```

However, if you want the results in another unit then you can also pass in a scale argument.

For example, to get the results in KB:

```
> db.foo.stats(1024)
{
  "size" : 698806652, // total size (KB)
  "avgObjSize" : 2,   // average size (KB)
}
```

Or for MB:

```
> db.foo.stats(1024 * 1024)
{
  "size" : 682428, // total size (MB)
  "avgObjSize" : 0, // average size (MB)
}
```

📞 Call us on

✉ Drop us a Query



Q83) Storing Friend Relationships in MongoDB?

I was wondering what the best way of storing friend relationship data using MongoDB is?

Coming from mysql I had a separate table with friend relationships that had two foreign keys, each pointing to a friend in the “friendship” however, with MongoDB its possible to have arrays of references or even embedded documents..so whats the best way to store these relationships

Immediate first reaction was that I’d store an array of friend object ids in each user, however, that troubles me, because then, in order to remove a “friendship” I would have to do deletes on both friend’s documents whereas if I stored the relationship in a separate collection (a la SQL) i could remove or add a relationship by just modifying one collection.

Thanks!

Keeping a list of friend_ids in a user, is what I’ll recommend. Few reasons,

1.You query a user, and you have list of all friends upfront available.

2.The requests (pending, accepted) can be handled well, by seeing that a respective ids should be present in both the user’s friends list. And I can get list of actual and accepted friends by querying

```
my_id, my_friends = user._id, user.friends
```

```
db.users.find({'_id':{'$in': my_friends}, 'friends': my_id})
```

Yes, while removing a friendship , you have to \$pull from both the friend’s list of both users, but frequency of that would be much less. But you query less in getting the friend-list, which would be used frequently.

Q84) MongoDB – file size is huge and growing

I have an application that use mongo for storing short living data. All data older than 45 minutes is removed by script something like:

```
oldSearches = [list of old searches]
```

```
connection = Connection()
```

```
db = connection.searchDB
```

```
res = db.results.remove({'search_id':{'$in':oldSearches}})
```

I’ve checked current status –

```
>db.results.stats()
```

```
{
  "ns" : "searchDB.results",
  "count" : 2865,
  "size" : 1003859656,
  "storageSize" : 29315124464,
  "nindexes" : 1,
  "ok" : 1
}
```

So, according to this 1gb of data occupies 29GB of storage. Data folder looks like this(You may see that many files are very old – last accessed in the middle of may):

 Skip code block

 Drop us a Query

```
ls -l /var/lib/mongodb/
total 31506556
-rwxr-xr-x 1 mongodb nogroup      6 2011-06-05 18:28 mongod.lock
-rw---- 1 mongodb nogroup 67108864 2011-05-13 17:45 searchDB.0
-rw---- 1 mongodb nogroup 134217728 2011-05-13 14:45 searchDB.1
-rw---- 1 mongodb nogroup 2146435072 2011-05-20 20:45 searchDB.10
-rw---- 1 mongodb nogroup 2146435072 2011-05-28 00:00 searchDB.11
-rw---- 1 mongodb nogroup 2146435072 2011-05-27 13:45 searchDB.12
-rw---- 1 mongodb nogroup 2146435072 2011-05-29 16:45 searchDB.13
-rw---- 1 mongodb nogroup 2146435072 2011-06-07 13:50 searchDB.14
-rw---- 1 mongodb nogroup 2146435072 2011-06-06 01:45 searchDB.15
-rw---- 1 mongodb nogroup 2146435072 2011-06-07 13:50 searchDB.16
-rw---- 1 mongodb nogroup 2146435072 2011-06-07 13:50 searchDB.17
-rw---- 1 mongodb nogroup 2146435072 2011-06-06 09:07 searchDB.18
-rw---- 1 mongodb nogroup 268435456 2011-05-13 14:45 searchDB.2
-rw---- 1 mongodb nogroup 536870912 2011-05-11 00:45 searchDB.3
-rw---- 1 mongodb nogroup 1073741824 2011-05-29 23:37 searchDB.4
-rw---- 1 mongodb nogroup 2146435072 2011-05-13 17:45 searchDB.5
-rw---- 1 mongodb nogroup 2146435072 2011-05-18 17:45 searchDB.6
-rw---- 1 mongodb nogroup 2146435072 2011-05-16 01:45 searchDB.7
-rw---- 1 mongodb nogroup 2146435072 2011-05-17 13:45 searchDB.8
-rw---- 1 mongodb nogroup 2146435072 2011-05-23 16:45 searchDB.9
-rw---- 1 mongodb nogroup 16777216 2011-06-07 13:50 searchDB.ns
-rw---- 1 mongodb nogroup 67108864 2011-04-23 18:51 test.0
-rw---- 1 mongodb nogroup 16777216 2011-04-23 18:51 test.ns
```

According to “top” mongod uses 29G of virtual memory (and 780Mb of RSS)

Why do i have such abnormal values? Do i need to run something additional to .remove() function to clean up database from old values?

Virtual memory size and resident size will appear to be very large for the mongod process. This is benign: virtual memory space will be just larger than the size of the datafiles open and mapped; resident size will vary depending on the amount of memory not used by other processes on the machine.

<https://www.mongodb.org/display/DOCS/Caching>

When you remove an object from MongoDB collection, the space it occupied is not automatically garbage collected and new records are only appended to the end of data files, making them grow bigger and bigger. This explains it all:

<https://www.mongodb.org/display/DOCS/Excessive+Disk+Space>

For starters, simply use:

```
db.repairDatabase()
```



Q85) Mongo does not have a max() function, how do I work around this?

📞 Call us on

✉ Drop us a Query

I have a MongoDB collection and need to find the max() value of a certain field across all docs. This value is the timestamp and I need to find the latest doc by finding the largest timestamp. Sorting it and getting the first one gets inefficient really fast. Shall I just maintain a 'maxval' separately and update it whenever a doc arrives with a larger value for that field? Any better suggestions? Thanks much.

For sure if it will be big collection and if you need always display max timestamp you may need create separate collection and store statistic data there instead of order big collection each time.

statistic

```
{
  _id = 1,
  id_from_time_stamp_collection = 'xxx',
  max_timestamp: value
}
```

And whenever new doc come just update statistic collection with id = 1 (with \$gt condition in query, so if new timestamp will be greater than max_timestamp then max_timestamp will be updated, otherwise - no).

Also probably you can store and update other statistic data within statistic collection.

if you have an index on the timestamp field, finding the highest value is efficient! something like

```
db.things.find().sort({ts:-1}).limit(1)
```

but if having an index is too much overhead storing the max in a separate collection might be good.

Q86) Mongodb drop every database

I would like to know if there're a command to drop every databases from my mongodb ?

I know if I want to drop only one datatable, I just need to type the name of the database like the code below but I dont want to have to specify it.

```
mongo DB_NAME -eval 'db.dropDatabase();'
```

Any help would be appreciated.

Thanks.

John

you can create a javascript loop that do the job and then execute it in the mongoconsole.

```
var dbs = db.getMongo().getDBNames()
for(var i in dbs){
  db = db.getMongo().getDB( dbs[i] );
  print( "dropping db " + db.getName() );
  db.dropDatabase();
}
```

save it to dropall.js and then execute:

```
mongo dropall.js
```

Try this command:

```
mongo -quiet -eval 'db.getMongo().getDBNames().forEach(function(i)
{db.getSiblingDB(i).dropDatabase()})'
```

📞 Call us on

✉ Drop us a Query

You can do it easy through c# official driver:

```
var _mongoServer = MongoServer.Create("mongodb://localhost:27020");
var names = _mongoServer.GetDatabaseNames();
foreach (var name in names)
{
    _mongoServer.DropDatabase(name);
}
```

Q87) mongodb find by comparing field values

Is it possible to express the following SQL query in mongodb:

```
SELECT * FROM table AS t WHERE t.field1 > t.field2;
```

edit: To summarize:.

using a third field storing "field1 – field2" is almost perfect, but requires a little extra maintenance.

\$where will load and eval in JavaScript and won't use any indexes. No good for large data.

map/reduce has the same problem and will go through all records even if we need only one

You can do this using \$where:

```
db.coll.find( { $where: "this.field1 > this.field2" } );
```

But:

Javascript executes more slowly than the native operators, but it is very flexible

If performance is an issue better to go with way suggested by @yi_H.

You could store in your document field1 – field2 as field3, then search for { field3: { \$gt: 0 } }

It also possible to get matching documents with mapreduce.

Q88) insert to specific index for mongo array

Mongo supports arrays of documents inside documents. For example, something like

```
{_id: 10, "coll": [1, 2, 3] }
```

Now, imagine I wanted to insert an arbitrary value at an arbitrary index

```
{_id: 10, "coll": [1, {name: 'new val'}, 2, 3] }
```

I know you can update values in place with \$ and \$set, but nothing for insertion. it kind of sucks to have to replace the entire array just for inserting at a specific index.

Now you can achieve this with a new upcoming version 2.6. To do this you have to use \$positionoperator. For your particular example:

```
db.students.update(
{ _id: 10},
{ $push: {
coll: {
$each: [ {name: 'new val'} ],
$position: 1
}
}}
```

📞 Call us on

✉ Drop us a Query

Q89) Where does MongoDB store its documents?

I have inserted and fetched data using MongoDB, in PHP. Is there an actual copy of this data in a document somewhere?

MongoDB stores its data in the data directory specified by `-dbpath`. It uses a database format so it's not actual documents, but there are multiple documents in each file and you cannot easily extract the data from this format yourself.

To read and/or update a document you need to use a MongoDB client, in the same way that you send SQL queries to MySQL through a MySQL client. You probably want to do it programmatically by using one of the client libraries for your programming language, but there is also a command-line client if you need to do manual updates.

By default Mongo stores its data in the directory `/data/db`.

You can specify a different directory using the `-dbpath` option.

If you're running Mongo on Windows it will be `C:\data\db` (the driver letter will be the same as the working directory). To avoid confusion, I'd recommend that you always specify a data directory using `-dbpath`.

Q90) What does it mean to fit "working set" into RAM for MongoDB?

MongoDB is fast, but only when your working set or index can fit into RAM. So if my server has 16G of RAM, does that mean the sizes of all my collections need to be less than or equal to 16G? How does one say "ok this is my working set, the rest can be "archived?"

"Working set" is basically the amount of data AND indexes that will be active/in use by your system.

So for example, suppose you have 1 year's worth of data. For simplicity, each month relates to 1GB of data giving 12GB in total, and to cover each month's worth of data you have 1GB worth of indexes again totalling 12GB for the year.

If you are always accessing the last 12 month's worth of data, then your working set is: 12GB (data) + 12GB (indexes) = 24GB.

However, if you actually only access the last 3 month's worth of data, then your working set is: 3GB (data) + 3GB (indexes) = 6GB. In this scenario, if you had 8GB RAM and then you started regularly accessing the past 6 month's worth of data, then your working set would start to exceed past your available RAM and have a performance impact.

But generally, if you have enough RAM to cover the amount of data/indexes you expect to be frequently accessing then you will be fine.

Edit: Response to question in comments

I'm not sure I quite follow, but I'll have a go at answering. Firstly, the calculation for working set is a "ball park figure". Secondly, if you have a (e.g.) 1GB index on `user_id`, then only the portion of that index that is commonly accessed needs to be in RAM (e.g. suppose 50% of users are inactive, then 0.5GB of the index will be more frequently required/needed in RAM). In general, the more RAM you have, the better especially as working set is likely to grow over time due to increased usage. This is where sharding comes in - split the data over multiple nodes and you can cost effectively scale out. Your working set is then divided over multiple machines, meaning the more can be kept in RAM. Need more RAM? Add another machine to shard on to.

Call us on

Drop us a Query

The working set is basically the stuff you are using most (frequently). If you use index A for collection B to search for a subset of documents then you could consider that your working set. As long as the most commonly used parts of those structures can fit in memory then things will be exceedingly fast. As parts no longer fit in your working set, like many of the documents then that can slow down. Generally things will become much slower if your indexes exceed your memory.

Yes, you can have lots of data, where most of it is “archived” and rarely used without affecting the performance of our application or impacting your working set (which doesn’t include that archived data).

Q91) MongoDB: Updating a document in an array

I have a collection with documents of this schema:

```
{
  _id: something,
  recipients: [{id:1, name:"Andrey", isread:false}, {id:2, name:"John", isread:false}]
}
```

Now, I want to update “isread” for John (id = 2) using findAndModify(), because I also need to get the original document.

I’m trying this command:

```
db.messages.findAndModify({query:{'recipients.id':2}, update:{'recipients.$.isread':true}})
```

but what it does, it just replaces the whole “recipients” field with ‘recipients.\$.isread’, so the document now looks like:

```
{
  _id: someid,
  'recipients.$.isread':true
}
```

What am I doing wrong?

Try to use \$set like this:

```
db.messages.findAndModify({query:{'recipients.id':2}, update:{$set:{'recipients.$.isread':true}}})
```

Q92) How to limit number of updating documents in mongodb

How to implement somethings similar to db.collection.find().limit(10) but while updating documents?

Now I’m using something really crappy like getting documents with db.collection.find().limit() and then updating them.

In general I wanna to return given number of records and change one field in each of them.

Thanks.

Unfortunately the workaround you have is the only way to do it AFAIK. There is a boolean flag multi which will either update all the matches (when true) or update the 1st match (when false).

You can use:

```
db.collection.find().limit(NUMBER_OF_ITEMS_YOU_WANT_TO_UPDATE).forEach(
function (e) {
```

e.fieldToChange = “blah”;

✉ Drop us a Query

....

```
db.collection.save(e);
}
);
```

(Credits for forEach code: MongoDB: Updating documents using data from the same document)

What this will do is only change the number of entries you specify. So if you want to add a field called “newField” with value 1 to only half of your entries inside “collection”, for example, you can put in

```
db.collection.find().limit(db.collection.count() / 2).forEach(
function (e) {
e.newField = 1;
db.collection.save(e);
}
);
```

If you then want to make the other half also have “newField” but with value 2, you can do an update with the condition that newField doesn’t exist:

```
db.collection.update( { newField : { $exists : false } }, { $set : { newField : 2 } }, {multi : true} );
```

Q93) Compound indices for OR+Sort query in mongodb

For this OR query:

```
db.messages.find({ $or: [ { to: { $ne: null }, from: “xyz” }, { to: “xyz” } ] }).sort({_id: -1}).limit(50)
```

with following indices:

```
{to:1, from: 1, _id:-1} and {from:1, to:1, _id:-1}
```

mongo is always doing a full scan.

I was hoping that mongo could use these two indices and merge the results.

Do I need to split this into two queries (one for each OR clause) and merge myself? Or is there some other index that can help?

This is a known issue, <https://jira.mongodb.org/browse/SERVER-1205>, which you can vote for if it is very important for your use case.

Q94) MongoDB: Unique index on array element’s property

I have a structure similar to this:

```
class Cat {
int id;
List kittens;
}
class Kitten {
int id;
}
```

I’d like to prevent users from creating a cat with more than one kitten with the same id. I’ve tried creating an index as follows:

```
db.cats.ensureIndex({'id': 1, 'kittens.id': 1}, {unique:true})
```

✉ Drop us a Query

But when I attempt to insert a badly-formatted cat, Mongo accepts it.

Am I missing something? can this even be done?

As far as I know, unique indexes only enforce uniqueness across different documents, so this would throw a duplicate key error:

```
db.cats.insert( { id: 123, kittens: [ { id: 456 } ] } )
```

```
db.cats.insert( { id: 123, kittens: [ { id: 456 } ] } )
```

But this is allowed:

```
db.cats.insert( { id: 123, kittens: [ { id: 456 }, { id: 456 } ] } )
```

I'm not sure if there's any way enforce the constraint you need at the Mongo level, maybe it's something you could check in the application logic when you insert or update?

Q95) Updating embedded document property in MongoDB

I have a document that looks like this:

Skip code block

```
{
  "_id": 3,
  "Slug": "slug",
  "Title": "title",
  "Authors": [
    {
      "Slug": "slug",
      "Name": "name"
    }
  ]
}
```

I want to update all Authors.Name based on Authors.Slug. I tried this but it didn't work:

```
.update({"Authors.Slug":"slug"}, {$set: {"Authors.Name":"zzz"}});
```

What am I doing wrong here?

```
.update(Authors:{$elemMatch:{Slug:"slug"}}, {$set: {'Authors.$.Name':'zzz'}});
```

Q96) What is the best way to store dates in MongoDB?

I am just starting to learn about MongoDB and hoping to slowly migrate from MySQL.

In MySQL, there are two different data types - DATE ('0000-00-00') and DATETIME ('0000-00-00 00:00:00'). In my MySQL, I use the DATE type, but I am not sure how to transfer them into MongoDB. In MongoDB, there is a Date object, which is comparable to DATETIME. It seems it would be most appropriate to use Date objects, but that would be wasting space, since hours, min, sec are not utilized. On the other hand, storing dates as strings seems wrong.

Is there a golden standard on storing dates ('0000-00-00') in MongoDB?

BSON (the storage data format used by mongo natively) has a dedicated date type UTC datetime which is a 64 bit (so, 8 byte) signed integer denoting milliseconds since Unix time epoch. There are very few valid reasons why you would use any other type for storing dates and timestamps. [Drop us a Query](#)

If you're desperate to save a few bytes per date (again, with mongo's padding and minimum block size and everything this is only worth the trouble in very rare cases) you can store dates as a 3 byte binary blob by storing it as an unsigned integer in YYYYMMDD format, or a 2 byte binary blob denoting "days since January 1st of year X" where X must be chosen appropriately since that only supports a date range spanning 179 years.

EDIT: As the discussion below demonstrates this is only a viable approach in very rare circumstances.

Basically; use mongo's native date type

I'm actually in the process of converting a MongoDB database where dates are stored as proper Date() types to instead store them as strings in the form yyyy-mm-dd. Why, considering that every other answerer says that this is a horrible idea? Simply put, because of the neverending pain I've been suffering trying to work with dates in JavaScript, which has no (real) concept of timezones. I had been storing UTC dates in MongoDB, i.e. a Date() object with my desired date and the time set as midnight UTC, but it's unexpectedly complicated and error-prone to get a user-submitted date correctly converted to that from whatever timezone they happen to be in. I've been struggling to get my JavaScript "whatever local timezone to UTC" code to work (and yes, I'm aware of Sugar.js and Moment.js) and I've decided that simple strings like the good old MySQL standard yyyy-mm-dd is the way to go, and I'll parse into Date() objects as needed at runtime on the client side.

Incidentally, I'm also trying to sync this MongoDB database with a FileMaker database, which also has no concept of timezones. For me the simplicity of simply not storing time data, especially when it's meaningless like UTC midnight, helps ensure less-buggy code even if I have to parse to and from the string dates now and then.

Q97) MongoDB : Avoid excessive disk space

I am having a database which has been allocated a space of 85GB. I got this size using the show dbs command. But when I use the db.stats(), i get the storage size as 63GB. After going through the docs I found that mongo db allocates a size to the db that is created and then the actual data is filled.

Why does mongo does this preallocation and is there a way I can avoid this preallocation ?

If Yes, will it be a good idea to do it or it is going to affect from performance point of view ?

Thanks!!!

Details can be found here: <https://www.mongodb.org/display/DOCS/Excessive+Disk+Space>

Now to your case. I think that the current size of your datafiles cannot be caused by preallocation alone. The maximum size of allocated data files is 2G. Maybe you once stored much more data than today? The link above also shows how to reclaim this data if you're sure you will not need it soon anyway. This will require a lot of disk space during the process, so you must give this some planning.

Disabling preallocation will indeed have performance impact. When you do many inserts, the process would regularly have to wait to create a new file.

Q98) Get n-th element of an array in mongo

As part of my document in mongo I'm storing an array of objects. How can I query it for only the 4th element of the array for example? So I don't want to get the entire array out, just the 4th element.

Use \$elemMatch

☒ Drop us a Query

```
db.foo.find({ bar : "xyz" }, { my_array : { $slice : [n , 1] } } )
```

will retrieve the nth element of the array "my_array" of all documents in the foo collection where bar = "xyz".

Some other examples from the MongoDB documentation:

```
db.posts.find({}, {comments:{$slice: 5}}) // first 5 comments
```

```
db.posts.find({}, {comments:{$slice: -5}}) // last 5 comments
```

```
db.posts.find({}, {comments:{$slice: [20, 10]}}) // skip 20, limit 10
```

```
db.posts.find({}, {comments:{$slice: [-20, 10]}}) // 20 from end, limit 10
```

Which you can read here: <https://www.mongodb.org/display/DOCS/Retrieving+a+Subset+of+Fields>

Another way to do this is to use the update array syntax. Here, contribs.1 sets the second element in the contribs array to have value ALGOL 58 (Taken from the manual page on update syntax)

```
db.bios.update(
{ _id: 1 },
{ $set: { 'contribs.1': 'ALGOL 58' } }
)
```

Q99) How to delete n-th element of array in mongodb

For example i have a document

```
db.test.save({_id: 1, list:[
{key: "a"},
{key: "b"},
{key: "c"},
{key: "d"},
{key: "e"}
]})
```

and i need remove second element from the list. For now I do that in a two steps. First of all I unset second list element but unset operator do not remove element, it is going to be null, after that I pull any nullable value from the list field

```
db.test.update({_id: 1}, {$unset: {"list.2": 1}})
db.test.update({_id: 1}, {$pull: {list: null}})
```

I want to ask whether there is solution do that in a one operation?

No, unfortunately what you are doing is currently the best option. Have a look at this question: In mongoDb, how do you remove an array element by it's index which links to a Jira for this very issue.

Q100) Slow pagination over tons of records in mongo

I have over 300k records in one collection in Mongo.

When I run this very simple query:

```
db.myCollection.find().limit(5);
```

It took only few miliseconds. When I use skip in the query:

```
db.myCollection.find().skip(200000).limit(5)
```

📞 Call us on

✉ Drop us a Query

It won't return anything... it is running minutes and return nothing.

How to make it better?

From MongoDB documentation:

Paging Costs

Unfortunately skip can be (very) costly and requires the server to walk from the beginning of the collection, or index, to get to the offset/skip position before it can start returning the page of data (limit). As the page number increases skip will become slower and more cpu intensive, and possibly IO bound, with larger collections.

Range based paging provides better use of indexes but does not allow you to easily jump to a specific page.

You have to ask yourself a question: how often do you need 40000th page? Also see this article;

One approach to this problem, if you have large quantities of documents and you are displaying them in sorted order (I'm not sure how useful skip is if you're not) would be to use the key you're sorting on to select the next page of results.

So if you start with

```
db.myCollection.find().limit(100).sort(created_date:true);
```

and then extract the created date of the last document returned by the cursor into a variable `max_created_date_from_last_result`, you can get the next page with the far more efficient (presuming you have an index on `created_date`) query

```
db.myCollection.find({created_date      :      {      $gt      :      max_created_date_from_last_result}
}).limit(100).sort(created_date:true);
```

Q101) How to use variables in MongoDB Map-reduce map function

Given a document

```
{_id:110000, groupings:{A:'AV',B:'BV',C:'CV',D:'DV'},coin:{old:10,new:12}}
```

My specs call for the specification of attributes for mapping and aggregation at run time, as the groupings the interested in are not known up front, but specified by the user.

For example, one user would specify [A,B] which will cause mapping emissions of
`emit({A:this.groupings.A,B:this.groupings.B},this.coin)`

while another would want to specify [A,C] which will cause mapping emissions of
`emit({A:this.groupings.A,C:this.groupings.C},this.coin)`

B/c the mapper and reducer functions execute server side, and don't have access to client variables, I haven't been able to come up with a way to use a variable map key in the mapper function.

If I could reference a list of things to group by from the scope of the execution of the map function, this is all very strait forward. However, b/c the mapping function end up getting from from a different scope, I don't know how to do this, or if it's even possible.

Before I start trying to dynamically build java script to execute through the driver, does anyone have a better suggestion? Maybe a 'group' function will handle this scenario better?

You can pass global, read-only data into map-reduce functions using the "scope" parameter on the `map-reduce` command. It's not very well documented, I'm afraid.

✉ Drop us a Query

As pointed out by @Dave Griffith, you can use the scope parameter of the mapReduce function.

I struggled a bit to figure out how to properly pass it to the function because, as pointed out by others, the documentation is not very detailed. Finally, I realised that mapReduce is expecting 3 params:

1. map function
2. reduce function
3. object with one or more of the params defined in the doc

Eventually, I arrived at the following code in Javascript:

```
// I define a variable external to my map and to my reduce functions
var KEYS = {STATS: "stats"};

function m() {
  // I use my global variable inside the map function
  emit(KEYS.STATS, 1);
}

function r(key, values) {
  // I use a helper function
  return sumValues(values);
}

// Helper function in the global scope
function sumValues(values) {
  var result = 0;
  values.forEach(function(value) {
    result += value;
  });
  return result;
}

db.something.mapReduce(
  m,
  r,
  {
    out: {inline: 1},
    // I use the scope param to pass in my variables and functions
    scope: {
      KEYS: KEYS,
      sumValues: sumValues // of course, you can pass function objects too
    }
  }
);
```

Q102) mongo dbname -eval 'db.collection.find()' does not work



Why does this work:

```
# mongo dbname
```

MongoDB shell version: 1.8.3

connecting to: nextmuni_staging

```
> db.collection.find()
```

```
{ "foo" : "bar" }
```

```
> bye
```

While this does not work:

```
# mongo localhost/dbname -eval 'db.collection.find()'
```

MongoDB shell version: 1.8.3

connecting to: localhost/dbname

DBQuery: dbname.collection -> undefined

It should be exactly the same, no?

Thanks!

The return val of `db.collection.find()` is a cursor type. Executing this command from within the shell will create a cursor and show you the first page of data. You can start going through the rest by repeating the 'it' command.

I think the scope of variables used during the execution of an eval'd script is only for the lifetime of the script (data can be persisted into collections of course) so once the script terminates those cursor variables no longer exist and so you would be able to send another eval script to page the data. So the behaviour you get during a shell session wouldn't really work from an eval script.

To get close to the behaviour you could run something like this:

```
mongo dbname -eval "db.collection.find().forEach(function(x){printjson(x);})"
```

That shows you that the command does execute and produce a cursor which you can then iterate over sending the output to stdout.

Edit: I think the point I was trying to make was that the command you are issuing is working its just the output is not what you expect.

Q103) Compact command not freeing up space in MongoDB 2.0

I just installed MongoDB 2.0 and tried to run the compact command instead of the repair command in earlier versions. My database is empty at the moment, meaning there is only one collection with 0 entries and the two system collections (indices, users). Currently the db takes about 4 GB of space on the harddisk. The db is used as a temp queue with all items being removes after they have been processed.

I tried to run the following in the mongo shell.

```
use mydb
```

```
db.theOnlyCollection.runCommand("compact")
```

It returns with

```
ok: 1
```

But still the same space is taken on the harddisk. I tried to compact the system collections as well, but

☎ Call us on

✉ Drop us a Query



this did not work.

When I run the normal repair command

```
db.repairDatabase()
```

the database is compacted and only takes 400 MB.

Anyone has an idea why the compact command is not working?

Thanks a lot for your help.

Best

Alex

Collection compaction is not supposed to decrease the size of data files. Main point is to defragment collection and index data – combine unused space gaps into continuous space allowing new data to be stored there. Moreover it may actually increase the size of data files:

Compaction may increase the total size of your data files by up to 2GB. Even in this case, total collection storage space will decrease.

<https://www.mongodb.org/display/DOCS/compact+Command#compactCommand-Effectsofcompact>

Q104) MongoDB: order by two fields sum

```
SELECT (a+b) as c FROM my_table ORDER BY c ASC;
```

How can I order by two columns sum in Mongo?

You can't do it easy without an extra action.

To sort on any computed value you need to store it in a document first or in other words you need to create extra field 'c', and store a + b in it with each update/insert and only then sort on 'c' as usual.

Q105) Is it practical to use Mongo's capped collection as a memory cache?

Is it practical to use Mongo's capped collections as poor man's memcache assuming it will be kept in memory most of the time?

You can definitely use capped collections for this purpose. But it's important to know the basic limitations.

1. Data will expire based on insert order not time-to-expire
2. Data that is frequently accessed may still be pushed out of memory
3. `_id` columns are not defined on Capped Collections by default, you will need to ensure those indexes.
4. The objects cannot be modified in a way that changes the size of the object. For example, you could increment an existing integer, but you cannot add a field or modify a string value on the entry.
5. The capped collection cannot be sharded.

Because of #1 & #4 & #5, you are definitely losing some the core Memcache functionality.

There is a long-outstanding JIRA ticket for TTL-based capped collections which is probably exactly what you want.

Of course, the big question in this whole debate is "where is the extra RAM". Many people who use MongoDB as their primary store simply drop Memcache. If you have a bunch of extra RAM sitting around, why not just use it store the actual data instead of copies of that data?

📞 Call us on

✉ Drop us a Query

Q106) MongoDB Path Change?

My server went down from an electrical failure and for a few horrifying seconds, I thought I'd lost all MongoDB data. I then realized that when the server restarted, mongo automatically restarted without the `-dbpath` option.

What I can't figure out is why, even though my `mongodb.conf` has the `dbpath` set to `/var/lib/mongodb`, mongo looked for the db files in `/data/db` on startup. Can anyone shed some light on this?

Thanks!

`/data/db` is the default path that `mongod` will look for data files in if it is started with no options. Does your startup script instruct `mongod` to load the correct config file? If not, that would explain this behavior.

When was the last time you updated Mongod on your system? and how did you update it?

Depending how you installed / updated Mongod it could happen that either the `/etc/init.d/mongo` script or the `/etc/mongo.conf` files could be overwritten.

Or somebody edited those files a long time ago, and this is the first time Mongo got restarted, and now it looks like there's suddenly a change.

I haven't seen either of those two things happen in a long time (but if you're using Gentoo, you would probably see this happen)

What OS do you use?

It's a good idea to keep backups of those files...

Q107) In MongoDB, how do I find documents where array size is greater than 1?

I have a MongoDB collection (accommodations) that has documents in the following format:

Skip code block

```
{
  "_id" : ObjectId("4e8ae86d08101908e1000001"),
  "name" : ["Some Name"],
  "zipcode" : ["2223"]
}
{
  "_id" : ObjectId("4e8ae86d08101908e1000002"),
  "name" : ["Another ", "Name"],
  "zipcode" : ["2224"]
}
{
  "_id" : ObjectId("4e8ae86d08101908e1000003"),
  "name" : ["Yet", " Another ", "Name"],
  "zipcode" : ["2225"]
}
```

My question is: How can I select all documents that have a name array size of greater than 1?

Can get documents that match an array size:

 Drop us a Query

```
db.accommodations.find({"name":{"$size:2 }})
```

This will correctly return the documents where there are 2 elements in the name array. However I can't do a \$gt command to return all documents where the name field has an array size of greater than 2:

```
db.accommodations.find({"name":{"$size: {$gt:1} }})
```

Any ideas how I can do this on existing data... preferably without modifying the data?

Many thanks

Update:

For mongodb versions 2.2+ more efficient way to do this described by @JohnnyHK in another answer.

1.Using \$where

```
db.accommodations.find( { $where: "this.name.length > 1" } );
```

But...

Javascript executes more slowly than the native operators listed on this page, but is very flexible. See the server-side processing page for more information.

2.Create **extra** field NamesArrayLength, update it with names array length and then use in queries:

```
db.accommodations.find({"NamesArrayLength": {$gt: 1} });
```

It will be better solution, and will work much faster (you can create index on it).

There's a more efficient way to do this in MongoDB 2.2+ now that you can use numeric array indexes in query object keys.

```
// Find all docs that have at least a second name array element.
```

```
db.accommodations.find({'name.1': {$exists: true}})
```

I believe this is the fastest query that answers your question, because it doesn't use an interpreted \$where clause:

```
{ $nor: [
  { name: { $exists: false } },
  { name: { $size: 0 } },
  { name: { $size: 1 } }
]}
```

It means "all documents except those without a name (either non-existent or empty array) or with just one name."

Test:

Skip code block

```
> db.test.save({})
> db.test.save({name: []})
> db.test.save({name: ['George']})
> db.test.save({name: ['George', 'Raymond']})
> db.test.save({name: ['George', 'Raymond', 'Richard']})
> db.test.save({name: ['George', 'Raymond', 'Richard', 'Martin']})
> db.test.find({ $nor: [{name: { $exists: false }}, {name: { $size: 0 }}, {name: { $size: 1 }}]})
```

```
{ "_id" : ObjectId("511907e3fb13145a3d2e225b"), "name" : [ "George", "Raymond" ] }
```

📞 Call us on

✉ Drop us a Query

```
{ "_id" : ObjectId("511907e3fb13145a3d2e225c"), "name" : [ "George", "Raymond", "Richard" ] }
{ "_id" : ObjectId("511907e3fb13145a3d2e225d"), "name" : [ "George", "Raymond", "Richard", "Martin" ] }
>
```

Q108) heterogeneous bulk update in mongodb

I know that we can bulk update documents in mongodb with

```
db.collection.update( criteria, objNew, upsert, multi )
```

in one db call, but it's homogeneous, i.e. all those documents impacted are following one kind of criteria.

But what I'd like to do is something like

```
db.collection.update([ {criteria1, objNew1}, {criteria2, objNew2}, ...]
```

, to send multiple update request which would update maybe absolutely different documents or class of documents in single db call.

What I want to do in my app is to insert/update a bunch of objects with compound primary key, if the key is already existing, update it; insert it otherwise.

Can I do all these in one combine in mongodb?

That's two separate questions. To the first one; there is no MongoDB native mechanism to bulk send criteria/update pairs although technically doing that in a loop yourself is bound to be about as efficient as any native bulk support.

Checking for the existence of a document based on an embedded document (what you refer to as compound key, but in the interest of correct terminology to avoid confusion it's better to use the mongo name in this case) and insert/update depending on that existence check can be done with upsert :

document A :

```
{
  _id: ObjectId(...),
  key: {
    name: "Will",
    age: 20
  }
}
```

```
db.users.update({name:"Will", age:20}, {$set:{age: 21}}, true, false)
```

This upsert (update with insert if no document matches the criteria) will do one of two things depending on the existence of document A :

1. Exists : Performs update "\$set:{age:21}" on the existing document
2. Doesn't exist : Create a new document with fields "name" and field "age" with values "Will" and "20" respectively (basically the criteria are copied into the new doc) and then the update is applied (\$set: {age:21}). End result is a document with "name"="Will" and "age"=21.

Hope that helps

Q109) Is it possible to mongodump the last "x" records from a collection?



Can you use mongodump to dump the latest “x” documents from a collection? For example, in the mongo shell you can execute:

```
db.stats.find().sort({$natural:-1}).limit(10);
```

Is this same capability available to mongodump?

I guess the workaround would be to dump the above documents into a new temporary collection and mongodump the entire temp collection, but would be great to just be able to do this via mongodump.

Thanks in advance,

Michael

mongodump does not fully expose the cursor interfaces. But you can work around it, using the -query parameter. First get the total number of documents of the collection

```
db.collection.count()
```

Let's say there are 10000 documents and you want the last 1000. To do so get the id of first document you want to dump.

```
db.collection.find().sort({_id:1}).skip(10000 - 1000).limit(1)
```

In this example the id was “50ad7bce1a3e927d690385ec”. Now you can feed mongodump with this information, to dump all documents a with higher or equal id.

```
$ mongodump -d 'your_database' -c 'your_collection' -q '{_id: {$lte: ObjectId("50ad7bce1a3e927d690385ec")}}'
```

UPDATE the new parameters '-limit' and '-skip' were added to mongoexport will be probably available in the next version of the tool: <https://github.com/mongodb/mongo/pull/307>

mongodump supports the -query operator. If you can specify your query as a json query, you should be able to do just that.

If not, then your trick of running a query to dump the records into a temporary collection and then dumping that will work just fine. In this case, you could automate the dump using a shell script that calls a mongo with a javascript command to do what you want and then calling mongodump.

Q110) How do you get the size of an individual index in MongoDB?

I know I can use db.collection.totalIndexSize() to get the total index size, but I'm interested in seeing the size of an individual index.

Is this supported?

Certainly can. db.collection.stats().indexSizes is an embedded document where each index name is a key and the value is the total index size in bytes :

```
> db.test.stats()
```

```
{
```

```
  "ns" : "test.test",
```

```
  "indexSizes" : {
```

```
    "_id_" : 137904592,
```

```
    "a_1" : 106925728
```



```
},  
"ok" : 1  
}
```

Q111) mongodb Mongod complains that there is no /data/db folder

I am using my new mac for the first time today. I am following the get started guide on the mongodb.org up until the step where one creates the /data/db directory. btw, I used the homebrew route.

So I open a terminal, and I think I am at what you called the Home Directory, for when I do "ls", I see folders of Desktop Application Movies Music Pictures Documents and Library.

So I did a

```
mkdir -p /data/db
```

first, it says permission denied. I kept trying different things for half an hour and finally :

```
mkdir -p data/db
```

worked. and when I "ls", a directory of data and nested in it a db folder do exist.

then I fire up mongod and it complains about not finding data/db

Have I done something wrong?

Now I have done the

```
sudo mkdir -p /data/db
```

and when I do a "ls" I do see the data dir and the db dir. inside the db dir though, there is absolutely nothing in it and when I now run mongod

Skip code block

```
Sun Oct 30 19:35:19 [initandlisten] exception in initAndListen: 10309 Unable to create/open lock file:  
/data/db/mongod.lock errno:13 Permission denied Is a mongod instance already running?, terminating
```

```
Sun Oct 30 19:35:19 dbexit:
```

```
Sun Oct 30 19:35:19 [initandlisten] shutdown: going to close listening sockets...
```

```
Sun Oct 30 19:35:19 [initandlisten] shutdown: going to flush diaglog...
```

```
Sun Oct 30 19:35:19 [initandlisten] shutdown: going to close sockets...
```

```
Sun Oct 30 19:35:19 [initandlisten] shutdown: waiting for fs preallocator...
```

```
Sun Oct 30 19:35:19 [initandlisten] shutdown: lock for final commit...
```

```
Sun Oct 30 19:35:19 [initandlisten] shutdown: final commit...
```

```
Sun Oct 30 19:35:19 [initandlisten] shutdown: closing all files...
```

```
Sun Oct 30 19:35:19 [initandlisten] closeAllFiles() finished
```

```
Sun Oct 30 19:35:19 [initandlisten] shutdown: removing fs lock...
```

```
Sun Oct 30 19:35:19 [initandlisten] couldn't remove fs lock errno:9 Bad file descriptor
```

```
Sun Oct 30 19:35:19 dbexit: really exiting now
```

EDIT Getting error message for

```
sudo chown mongod:mongod /data/db
```

chown: mongod: Invalid argument

Thanks, everyone!



You created the directory in the wrong place

📞 Call us on

✉ Drop us a Query

/data/db means that it's directly under the '/' root directory, whereas you created 'data/db' (without the leading /) probably just inside another directory, such as the '/root' homedirectory.

You need to create this directory as root

Either you need to use sudo , e.g. `sudo mkdir -p /data/db`

Or you need to do su - to become superuser, and then create the directory with `mkdir -p /data/db`

Note:

MongoDB also has an option where you can create the data directory in another location, but that's generally not a good idea, because it just slightly complicates things such as DB recovery, because you always have to specify the db-path manually. I wouldn't recommend doing that.

Edit:

the error message you're getting is **"Unable to create/open lock file: /data/db/mongod.lock errno:13 Permission denied"**. The directory you created doesn't seem to have the correct permissions and ownership — it needs to be writable by the user who runs the MongoDB process.

To see the permissions and ownership of the '/data/db/' directory, do this: (this is what the permissions and ownership should look like)

```
$ ls -ld /data/db/
```

```
drwxr-xr-x 4 mongod mongod 4096 Oct 26 10:31 /data/db/
```

The left side 'drwxr-xr-x' shows the permissions for the User, Group, and Others. 'mongod mongod' shows who owns the directory, and which group that directory belongs to. Both are called 'mongod' in this case.

If your '/data/db' directory doesn't have the permissions and ownership above, do this:

sudo chmod 0755 /data/db

`sudo chown mongod:mongod /data/db`

that should make it work..

Check here to better understand the meaning of the directory permissions:

<https://www.dartmouth.edu/~rc/help/faq/permissions.html>

Maybe also check out one of the tutorials you can find via Google: "UNIX for beginners"

After getting the same error as Nik

chown: id -u: Invalid argument

I found out this apparently occurred from using the wrong type of quotation marks (should have been **backquotes**) Ubuntu Forums

Instead I just used

```
sudo chown $USER /data/db
```

as an alternative and now mongod has the permissions it needs.

Q112) mongodb replicaset host name change error

I have a mongodb replicaset on ubuntu.. In replica set, hosts are defined as localhost. You can see ;
Skip code block

```
{
  "_id" : "myrep",
  "Call us on
```



✉ Drop us a Query

```

"version" : 4,
"members" : [
{
  "_id" : 0,
  "host" : "localhost:27017"
},
{
  "_id" : 2,
  "host" : "localhost:27018"
},
{
  "_id" : 1,
  "host" : "localhost:27019",
  "priority" : 0
}
]
}

```

I want to change host addresses with real ip of server. But when i run rs.reconfig, I get error :

```

{
  "assertion" : "hosts cannot switch between localhost and hostname",
  "assertionCode" : 13645,
  "errmsg" : "db assertion failure",
  "ok" : 0
}

```

How can i solve it ? Thank you.

There is a cleaner way to do this:

use local

```
cfg = db.system.replset.findOne({_id:"replicaSetName"})
```

```
cfg.members[0].host="newHost:27017"
```

```
db.system.replset.update({_id:"replicaSetName"},cfg)
```

then restart mongo

Q113) Mongo compound indexes, using less-than-all in a query

I understand that with MongoDB, for a query to make use of a compound index it must use ALL of the keys in the index, or at least some of the keys starting from the left. For example

```
db.products.find({ "a":"foo", "b":"bar" })
```

Will happily make use of an index made up of {a, b, c}.

However, if I want to query:

```
db.products.find( { "a":"foo", "c":"thing" })
```

I believe this can't use the index. Could this be solved by adding a trivial condition on "b", e.g.

☎ Call us on

✉ Drop us a Query




```
db.products.find( {"a":"foo", "b":{" $ne : "" }, "c":"thing" })
```

Even when I don't actually care about the value of b. The reason for this is that we currently have 45m objects, and it's going to continue growing so we're looking to consolidate our indexes to save on resources.

Many thanks.

In general, a query on a multi-column index that does not sufficiently limit matches for one of the columns will restrict the usefulness of the multi-column index. In your example, using query criteria of {"a":"foo", "b":{" \$ne : "" }, "c":"thing"} will limit the usefulness of an {a,b,c} index to matching only on a. If your query criteria will be executed often, create an {a,c,b} index (or {a,c} if b will not be used in query criteria).

Use the explain function on your queries to see if an index is being used to its full potential. If explain tells you indexOnly is true, then your query is only using the index to find matching documents; otherwise, MongoDB needs to look at each document to find matches.

For further information, see:

1. Optimization
2. Indexes
3. Query Optimizer

Q114) New to MongoDB Can not run command mongo

I was trying to run MongoDB:

Skip code block

E:mongobin>mongod

mongod -help for help and startup options

Sun Nov 06 18:48:37

Sun Nov 06 18:48:37 warning: 32-bit servers don't have journaling enabled by default. Please use -journal if you want durability.

Sun Nov 06 18:48:37

Sun Nov 06 18:48:37 [initandlisten] MongoDB starting : pid=7108 port=27017 dbpath=/data/db 32-bit host=pykhmer-PC

Sun Nov 06 18:48:37 [initandlisten]

Sun Nov 06 18:48:37 [initandlisten] ** NOTE: when using MongoDB 32 bit, you are limited to about 2 gigabytes of data

Sun Nov 06 18:48:37 [initandlisten] ** see <https://blog.mongodb.org/post/137788967/32-bit-limitations>

Sun Nov 06 18:48:37 [initandlisten] ** with -journal, the limit is lower

Sun Nov 06 18:48:37 [initandlisten]

Sun Nov 06 18:48:37 [initandlisten] db version v2.0.1, pdf file version 4.5

Sun Nov 06 18:48:37 [initandlisten] git version: 3a5cf0e2134a830d38d2d1aae7e88cac31bdd684

Sun Nov 06 18:48:37 [initandlisten] build info: windows (5, 1, 2600, 2, 'Service Pack 3')

BOOST_LIB_VERSION=1_42

📞 Call us on

✉ Drop us a Query

Sun Nov 06 18:48:37 [initandlisten] options: {}

Sun Nov 06 18:48:37 [initandlisten] exception in initAndListen: 10296 dbpath (/data/db) does not exist, terminating

Sun Nov 06 18:48:37 dbexit:

Sun Nov 06 18:48:37 [initandlisten] shutdown: going to close listening sockets...

Sun Nov 06 18:48:37 [initandlisten] shutdown: going to flush diaglog...

Sun Nov 06 18:48:37 [initandlisten] shutdown: going to close sockets...

Sun Nov 06 18:48:37 [initandlisten] shutdown: waiting for fs preallocator...

Sun Nov 06 18:48:37 [initandlisten] shutdown: closing all files...

Sun Nov 06 18:48:37 [initandlisten] closeAllFiles() finished

Sun Nov 06 18:48:37 dbexit: really exiting now

E:mongobin>mongo

MongoDB shell version: 2.0.1

connecting to: test

Sun Nov 06 18:48:42 Error: couldn't connect to server 127.0.0.1 shell/mongo.js:84

exception: connect failed

E:mongo>ls

GNU-AGPL-3.0 README THIRD-PARTY-NOTICES bin data

I was looking at <https://www.mongodb.org/display/DOCS/Quickstart+Windows> and following the instructions. Could anyone tell me what is the problem with running MongoDB (I am using Windows 7)?

I think your log output states it clearly;

exception in initAndListen: 10296 dbpath (/data/db) does not exist, terminating

You may simply create this directory or better to define it as a configuration value within your configuration file then use it as mongod -f C:path-to-your-mongodb.conf.

After you've installed MongoDB you should firstly create a data directory for your db.

By default MongoDB will store data in /data/db,

but it won't automatically create that directory. To create it, do:

```
$ sudo mkdir -p /data/db/
```

```
$ sudo chown `id -u` /data/db
```

You can also tell MongoDB to use a different data directory,

with the -dbpath option.

For more detailed information go to MongoDB wiki page.

Specify the database path explicitly like so, and see if that resolves the issue.

```
mongod -dbpath data/db
```

```
mongod -dbpath "c://data/db"
```

run the above code, this will start the server.

Check that path to database data files exists :

Sun Nov 06 18:48:37 [initandlisten] **exception** in initAndListen: 10296 **dbpath (/data/db) does not exist**, terminating ^

 MongoDB: Mapreduce: reduce->multiple not supported yet

 Drop us a Query

I have a MongoDB collection (named “catalog”) containing about 5 astronomical catalogs. Several of these catalogs reference each other, so one of the the documents might look something like this:

```
{ “_id” : ObjectId(“4ec574a68e4e7a519166015f”), “bii” : 20.9519, “class” : 2480, “cpdname” : “CPD -21 6109”, “decdeg” : -21.8417, “decpm” : 0.004, “dmname” : “-21 4299”, “hdname” : “HD 145612”, “lii” : 352.8556, “name” : “PPM 265262”, “ppmname” : “PPM 265262”, “radeg” : 243.2005, “rapm” : 0.0012, “vmag” : 9.6, “xref” : [ ] }
```

What I want to do is use mapreduce to move the fields such as “hdname”, “ppmname”, etc into the xref array (then unset them).

So I try to do this one at a time, starting with the hdname field. Here are the map and reduce functions:

Skip code block

```
map = function() {
  for (var hdname in this.hdname) {
    emit(this._id, this.hdname);
  }
}

reduce = function(key, values) {
  var result = [];
  for (var hdname in values) {
    result.push(hdname);
  }
  return result;
}
```

I try running the following command in the mongo shell:

```
db.catalog.mapReduce(map, reduce, “catalog2?”);
```

Unfortunately, I get the following error:

```
Thu Nov 17 15:52:17 uncaught exception: map reduce failed:{
  “assertion” : “reduce -> multiple not supported yet”,
  “assertionCode” : 10075,
  “errmsg” : “db assertion failure”,
  “ok” : 0
}
```

Obviously I’m a newbie... can anyone help?

Jason

The documentation says “Currently, the return value from a reduce function cannot be an array (it’s typically an object or a number).”

So create an object instead and wrap your array in that. Make sure also that the output of reduce is the same as the input type, so you’ll need to emit a similar value in the map operation.

BUT ... why use Map-Reduce to do this? If you emit the _id value there’s nothing to reduce as each key will be unique. Why not just iterate over the collection copying the values and updating each record ^{one}

by one?
Call us on

✉ Drop us a Query

Q116) MongoDB sorting

I want implement a “bump” feature for topics. Once a topic is bumped, it will have a new “bump_date” field. I want to sort it so that when there is a “bump_date” field, it will be sorted as if it was the “created” field. Here’s an example of my db.topics:

Skip code block

```
{
  "text" : "test 1",
  "created" : "Sun Nov 20 2011 02:03:28 GMT-0800 (PST)"
},
{
  "text" : "test 2",
  "created" : "Sun Nov 18 2011 02:03:28 GMT-0800 (PST)"
},
{
  "text" : "test 3",
  "created" : "Sun Nov 17 2011 02:03:28 GMT-0800 (PST)",
  "bump_date" : "Sun Nov 19 2011 02:03:28 GMT-0800 (PST)"
}
```

I want the sort to return in the order of “test 1”, “test 3”, “test 2”

Sorting in MongoDB is done like so:

```
db.collection.find({ ... spec ... }).sort({ key: 1 })
```

where 1 is ascending and -1 is descending.

In your specific example: `db.topics.find().sort({ bump_date: 1 })`, although it might be better to call it something like “updated_at”.

You’ll also definitely want to put an index on your “bump_date” field.

1. sorting: <https://www.mongodb.org/display/DOCS/Sorting+and+Natural+Order>

2. indexes: <https://www.mongodb.org/display/DOCS/Indexes>

As Brian Hicks suggested, creating an additional updated_at field is the way to go. This way, when a document is created you can have created_at and updated_at initially be the same.

```
{
  "created_at": xxx,
  "updated_at": xxx
}
```

If you then “bump” the updated_at field by setting it to the current time when there is a a bump event you can sort on the updated_at field to achieve the ordering you desire.

Q117) MongoDB inserts float when trying to insert integer

```
> db.data.update({'name': 'zero'}, {'$set': {'value': 0}})
```

```
> db.data.findOne({'name': 'zero'})
```

```
{ 'name': 'zero', 'value': 0.0 }
```

Call us on

✉ Drop us a Query

How do I get Mongo to insert an integer?

Thank you

```
db.data.update({'name': 'zero'}, {'$set': {'value': NumberInt(0)}})
```

You can also use NumberLong.

Q118) Mongodb : \$in operator vs lot of single queries

I know mongo is able to handle a lot of requests/sec, but let's say I have to query a lot of documents of a collection given their _id; what sounds better, making a \$in on the _id attribute with all the ids I want to get, or loop over findOne queries?

Thanks by advance :)

I would definitely go with using the \$in query and providing a array of _ids instead of looping unless the documents are large and returning them all in a single query would go over the size limit of 4mb (16mb on more recent versions of MongoDB)

Example:

```
db.collection.find({
  "key": { "$in": [
    ObjectId("xxx"),
    ObjectId("yyy"),
    ObjectId("zzz")
  ]}}
)
```

Why?

1. If you loop, there is a certain amount of setup and teardown for each query creating and exhausting cursors which would create overhead.
2. If you are not doing this on a local machine it also creates tcp/ip overhead for every request. Locally you could use domain sockets.
3. There is a index on "_id" created by default and collecting a group of documents to return in a batch request should be extremely fast so there is no need to break this up into smaller queries.

There's some additional documentation here if you want to check it out.

Q119) Mongodb - How to find string in multiple fields?

Using Pymongo for this scenario.

I have User that has email, first_name, last_name.

I am using this Pymongo snippet:

```
user_found = users.find({'$or':[
  {'email':{'$regex':searchString, '$options':'i'}},
  {'first_name':{'$regex':searchString, '$options':'i'}},
  {'last_name':{'$regex':searchString, '$options':'i'}}]})
```

this example works, if I want to find searchString in:

1. email, or

2. first_name, or

✉ Drop us a Query

3. last_name

now I need to also find searchString in first_name + last_name combined.

how can I do that? Is there a way in mongo, through the query, to combine the two into a “fullname” then search the fullname?

Easiest way is to add an array field and populate it with all of the variants that you want to search on. Index that array field.

That way you only need one index and your search across all fields is simple and doesn't change when you want to search on some new search variant. You can also normalize the text you put into the search array, for example, lower casing it, removing punctuation etc.

See <https://stackoverflow.com/q/8206188/224370>

Edit: MongoDB's documentation now covers keyword search and the new full-text search feature.

Q120) Get BinData UUID from Mongo as string

I currently have some ids stored in Mongo as UUIDs (necessary for processing). They get returned like this:

```
“_id” : new BinData(3, “JliB6gIMRuSphAD2KmhzgQ==”)
```

What would be an easy way to turn this value into a string for debugging?

Just to be clear – the application can handle the data fine. I just need a way to get the actual UUID from Mongo quickly.

The answer to your question is more complicated that you would expect! The main reason it's complicated is that for historical reasons (unfortunately) different drivers have written UUIDs to the database using different byte orders. You don't mention which driver you are using, but I'll use the C# driver as an example.

Suppose I use the following code to insert a document:

```
var guid = new Guid(“00112233-4455-6677-8899-aabbccddeeff”);
collection.Insert(new BsonDocument {
{ “_id”, guid },
{ “x”, 1 }
});
```

If I then examine the document using the Mongo shell, it looks like this:

```
> db.test.findOne()
{ “_id” : BinData(3,“MyIRAFVEd2almaq7zN3u/w==”), “x” : 1 }
>
```

The Mongo shell has a built-in function called hex that you can use to display the binary value as a hex string:

```
> var doc = db.test.findOne()
> doc._id.hex()
33221100554477668899aabbccddeeff
>
```

Look carefully: the byte order of the hex string doesn't match the original UUID value used in the C#

☎ Call us on

✉ Drop us a Query

program. That's because the C# driver uses the byte order returned by Microsoft's ToByteArray method of the Guid class (which sadly returns the bytes in a bizarre order, which fact was not discovered for many months). Other drivers have their own idiosyncracies.

To help out with this we have some helper functions written in Javascript that can be loaded into the Mongo shell. They are defined in this file:

<https://github.com/mongodb/mongo-csharp-driver/blob/master/uuidhelpers.js>

The Mongo shell can be told to process a file as it starts up by providing the name of the file on the command line (along with the `-shell` argument). Having loaded this file we have access to a number of helper functions to create and display BinData values that are UUIDs. For example:

```
C:\mongodb\mongodb-win32-x86_64-2.0.1\bin>mongo -shell uuidhelpers.js
```

```
MongoDB shell version: 2.0.1
```

```
connecting to: test
```

```
type "help" for help
```

```
> var doc = db.test.findOne()
```

```
> doc._id.toCSUUID()
```

```
CSUUID("00112233-4455-6677-8899-aabbccddeeff")
```

```
> db.test.find({_id : CSUUID("00112233-4455-6677-8899-aabbccddeeff")})
```

```
{ "_id" : BinData(3,"MyIRAFVEd2almaq7zN3u/w=="), "x" : 1 }
```

```
>
```

In this example the `toCSUUID` function is used to display a BinData value as a CSUUID and the CSUUID function is used to create a BinData value for a UUID using the C# driver's byte ordering conventions so that we can query on a UUID. There are similar functions for the other drivers (`toJUUUID`, `toPYUUID`, `JUUUID`, `PYUUID`).

Some day in the future all drivers will standardize on a new binary subtype 4 with a standard byte order. In the meantime you have to use the appropriate helper function that matches whatever driver you are using.

Q121) All columns in MongoDB

I was wondering how you would find all of the column names in a table in MongoDB, like how you use `SHOW COLUMNS FROM foo;` in mysql.

MongoDB is schemaless and does not have tables. In MongoDB, each collection can have different types of items. You could store two very different items in the same collection:

```
db.test.insert( { "SomeString" : "How much wood would the woodchop chop ..." } );
```

```
db.test.insert( { "Amount": 2040.20, "Due": new ISODate("2012-11-10"), "UserId" : new ObjectId("...")} );
```

usually the objects are somehow related or have a common base type, but it's not required.

You can, however, take a look at individual records using

```
db.collectionName.findOne()
```

```
or
```



```
db.collectionName.find().pretty()
```

However, there's no guarantee from MongoDB that any two records look alike or have the same fields: **there's no schema.**

Q122) Can't map file memory in MongoDB

The windows server 2003 box hosting our MongoDB instance ran out of disk space, and Mongo started generating the error:

Can't map file memory.

After adding additional disk space to the server and restarting the MongoDB windows service, any operation against the database still generates the "Can't map file memory" error. I tried doing a repair even and it gives the same error:

```
> db.repairDatabase(); { "assertion" : "can't map file memory", "assertionCode" : 10085, "errmsg" : "db assertion failure", "ok" : 0 }
```

Any idea what I can do to get my database operational again??

Stopping the service, deleting the lock file, and then doing a mongod -repair worked, even though db.repairDatabase did not.

The answer from Justin worked for me.

Here are some more detailed instructions for Ubuntu:

Stop Mongo Service: `sudo service mongod stop`

Delete Lock File: `sudo rm /var/lib/mongodb/mongod.lock`

Repair the DB: `sudo mongod -repair -dbpath=/var/lib/mongodb`

Restart the Mongo Service: `sudo service mongod start`

Hope that helps someone.

(edited - note that mongod is the service name, but mongod is the correct command for repairing)

Q123) Why does MongoDB have collections

MongoDB being document-oriented, the structure of collections seems to be a special case of documents. By that I mean one can define a document to contain other documents. So a collection is just a document containing other documents.

So why do we need collections after all?

Logically yes, you could design a database system like that, but practically speaking no.

1. A collection has indexes on the documents in it.
2. A collection requires the documents in it to have unique ids.
3. A document is limited in size.

Q124) How to update a field in MongoDB using existing value

I've been looking for a way to create an update statement that will take an existing numeric field and modify it using an expression. For example, if I have a field called Price, is it possible to do an update that sets Price to 50% off the existing value ?

So, given { Price : 19.99 }

Can I do `db.collection.update({tag : "refurb"}, {$set {Price : Price * 0.50 }}, false, true)` or is a Query

Can this be done or do I have to read the value back to the client, modify, then update ? I guess the question then is can expressions be used in update, and can they reference the document being updated. Thanks.

You can run server-side code with `db.eval()`.

```
db.eval(function() {
  db.collection.find({tag : "refurb"}).forEach(function(e) {
    e.Price = e.Price * 0.5;
    db.collection.save(e);
  });
});
```

Note this will block the DB, so it's better to do find-update operation pair.

See <https://www.mongodb.org/display/DOCS/Server-side+Code+Execution>

As of the release notes from upcoming Mongo 2.6 you would be able to use the new `$mul` operator. It would multiply the value of the field by the number with the following syntax.

```
{
  field: { $mul: }
}
```

So in your case you will need to do the following:

```
db.collection.update(
  { tag : "refurb"},
  { $mul: { Price : 0.5 } }
);
```

Q125) MongoDB \$where queries and tailable cursors — WAS: date math best practices

My problem: give me a list of documents older than X amount of time.

If I have a document created with:

```
db.dates.insert({date: new Date()});
```

And now I want to find it only when the "date" has become 30 minutes old:

```
db.dates.find({ $where: "this.date.getTime() + 30 * 60000 <= new Date()"});
```

This works, but in the Mongo documentation states quite clearly that there is a significant performance penalty to `$where` queries.

Thus the question, is there a better way?

=====UPDATE 1=====

I should have added that I am hoping to have this query function "dynamically" be creating the query one time and using it to obtain a tailable cursor on a capped collection... and I am not sure any longer that it is actually possible.

I will test and repost.

=====UPDATE 2=====

So, looks like my "delayed" queue is going to have to be handled in code, either with polling or some "check, then sleep" algorithm, because that appears to be what mongo's delayed replication is doing

☎ Call us on

✉ Drop us a Query

(from db.cpp):

Skip code block

```
if ( replSettings.slavedelay && ( unsigned( time( 0 ) ) < nextOpTime.getSecs() + replSettings.slavedelay)
) {
    assert( justOne );
    oplogReader.putBack( op );
    _sleepAdviceTime = nextOpTime.getSecs() + replSettings.slavedelay + 1;
    dblock lk;
    if ( n > 0 ) {
        syncedTo = last;
        save();
    }
    log() << "repl: applied " << n << " operations" << endl;
    log() << "repl: syncedTo: " << syncedTo.toStringLong() << endl;
    log() << "waiting until: " << _sleepAdviceTime << " to continue" << endl;
    return okResultCode;
}
```

The \$lte operator (and other range queries) will work and utilize indexes, but it cannot evaluate expressions. You have to query against a query-time constant (which would be 'now - 30 min'):

```
var threshold = new Date();
threshold.setMinutes(-30);
// now, query against a constant:
db.dates.find({"date" : {$lte : threshold}});
```

Of course, you can do the same with any driver for any programming language, e.g. C#

```
var c = db.Collection.Find(Query.LTE("date", DateTime.UtcNow.AddMinutes(-30)));
```

Q126) How to implement post tags in Mongo?

I'm toying with Mongo to make a SO-like pet project, and I want to implement post tags. Each tag has a name and a slug (string to be used as an id in the URL), and a post has multiple tags. I'd like to be able to create queries like "find posts, which have tag A, don't have tag B", and I'm wondering what's the mongo-way to do this.

One way is to store an array of tag ids with each post - this will make the said query easy, but will require an extra one for each post to get tag name and a slug. Another way is to store an array of [tag name, tag slug] with each post, but I'm not sure I'll be able to use that info in a find.

Is there some other method, which will work better for mongo? I'm new to NoSQL, so I'd appreciate any advise on how this can be accomplished. Also, I'm using PHP binding, but that shouldn't matter probably.

If the tags you use and their respective slugs are unlikely to change, I think your second approach is the better one. However I would suggest a small change - rather than storing an array of [name, slug], make the fields explicit by creating a tag subdocument as in this example post document:

📞 Call us on

✉ Drop us a Query

Skip code block

```
{
  "_id" : ObjectId("4ee33229d8854784468cda7e"),
  "title" : "My Post",
  "content" : "This is a post with some tags",
  "tags" : [
    {
      "name" : "meta",
      "slug" : "34589734"
    },
    {
      "name" : "post",
      "slug" : "34asd97x"
    }
  ]
}
```

You can then query for posts with a particular tag using dot notation like this:

```
db.test.find({ "tags.name" : "meta" })
```

Because tags is an array, mongo is clever enough to match the query against any element of the array rather than the array as a whole, and dot-notation allows you to match against a particular field.

To query for posts not containing a specific tag, use \$ne:

```
db.test.find({ "tags.name" : { $ne : "fish" } })
```

And to query for posts containing one tag but not the other, use \$and:

```
db.test.find({ $and : [{ "tags.name" : { $ne : "fish" } }, { "tags.name" : "meta" } ] })
```

Hope this helps!

Q127) What operations are cheap/expensive in mongodb?

I'm reading up on MongoDB, and trying to get a sense of where it's best used. One question that I don't see a clear answer to is which operations are cheap or expensive, and under what conditions.

Can you help clarify?

Thanks.

It is often claimed that mongodb has insanely fast writes. While they are not slow indeed, this is quite an overstatement. Write throughput in mongodb is limited by global write lock. Yes, you heard me right, there can be only **ONE** write operation happening on the server at any given moment.

Also I suggest you take advantage of schemaless nature of mongodb and store your data denormalized. Often it is possible to do just one disk seek to fetch all required data (because it is all in the same document). Less disk seeks – faster queries.

If data sits in RAM – no disk seeks are required at all, data is served right from memory. So, make sure you have enough RAM. ^

Map/Reduce, group, \$where queries are slow.

☎ Call us on

✉ Drop us a Query

It is not fast to keep writing to one big document (using \$push, for example). The document will outgrow its disk boundaries and will have to be copied to another place, which involves more disk operations.

And I agree with @AurelienB, some basic principles are universal across all databases.

Q128) MongoDB: Find a document by non-existence of a field?

Is there a way to specify a condition of “where document doesn’t contain field” ?

For example, I want to only find the first of these 2 because it doesn’t have the “price” field.

```
{“fruit”:”apple”, “color”:”red”}
```

```
{“fruit”:”banana”, “color”:”yellow”, “price”:”2.00?”}
```

```
db.mycollection.find( { “price” : { “$exists” : false } } )
```

nostalgic URL : <https://www.mongodb.org/display/DOCS/Advanced+Queries#AdvancedQueries-%24exists>

updated URL: https://docs.mongodb.org/manual/reference/operator/query/exists/#op._S_exists

Q129) How to remove deprecated fields in Mongo?

I have removed some fields from document definition. I want to remove this field across all documents of collection. How can I do it?

Try:

```
db.collection.update(
{ “: { ‘$exists’: true } }, // Query
{ ‘$unset’: { “: true } }, // Update
false, true                // Upsert, Multi
)
```

where field is your deprecated field and collection is the collection it was removed from.

The general update command is of the form `db.collection.update(criteria, objNew, upsert, multi)`. The false and true trailing arguments disable upsert mode and enable multi update so that the query updates all of the documents in the collection (not just the first match).

Update for MongoDB 2.2+:

You can now provide a JSON object instead of positional arguments for upsert and multi.

```
db.collection.update(
{ “: { ‘$exists’: true } }, // Query
{ ‘$unset’: { “: true } }, // Update
{ ‘multi’: true }          // Options
)
```

just do something like this

```
db.people.find().forEach(function(x) {
delete x.badField;
db.people.save(x);
})
```

👉 Call us on 000 on the \$unset answer someone gave using update() here is pretty awesome too. ☒ Drop us a Query

Q130) Combine two \$or statements

I am trying to perform a query which is composed of two \$or's:

```
|-----
| Date1 | Date2 |
|-----
| NULL  | NULL   | *
| NULL  | TODAY  | *
| NULL  | TOMRW   |
| TODAY | TODAY  | *
| TODAY | NULL   | *
| TOMRW | NULL   |
|-----
```

(I've marked the rows that would match with an asterisk)

`(Date1 == null || Date1 <= today) && (Date2 == null || Date2 <= today)`

I am not sure how to express this query in MongoDB.

It can be broken down into two individual queries that do exactly what they should:

Skip code block

```
{
  "$or": [{
    "Date1": {
      "$exists": false
    }
  },
  {
    "Date1": {
      "$exists": true,
      "$lte": new Date("2012-01-07T04:45:52.057Z")
    }
  }
]
```

and

Skip code block

```
{
  "$or": [{
    "Date2": {
      "$exists": false
    }
  },
  {
```

Call us on

✉ Drop us a Query

```

"$exists": true,
"$lte": new Date("2012-01-07T04:45:52.057Z")
}
}]
}

```

Both of these select the correct set of documents – I just don't know how to execute them as a single query.

My initial thought was to do a query like this:

```

{
  $and: [orQuery1, orQuery2]
}

```

Using an \$and query returns 0 results. It was explained why here in this thread: \$and query returns no result

Also in that thread, a suggestion was made to do a query like this:

```

{
  Key: {valToMatch1: 1, valToMatch2: 2}
}

```

But I don't think an \$or can be executed this way.

So, the question is: How do I construct my query such that I can combine the two \$or's into a single query?

(It's getting very late so I hope this question makes sense.)

use test

```

db.test.insert({a:1})
db.test.insert({a:2, Date2:new Date("01/07/2012")})
db.test.insert({a:3, Date2:new Date("01/08/2012")})
db.test.insert({a:4, Date1:new Date("01/07/2012"), Date2:new Date("01/07/2012")})
db.test.insert({a:5, Date1:new Date("01/07/2012")})
db.test.insert({a:6, Date1:new Date("01/08/2012")})

```

first subquery db.test.distinct('a', {...});

[1, 2, 3]

second subquery db.test.distinct('a', {...});

[1, 5, 6]

(Date1 == null || Date1 <= today) && (Date2 == null || Date2 <= today)

unwind

Date1 == null && Date2 == null ||

Date1 == null && Date2 <= today ||

Date1 <= today && Date2 == null ||

Date1 <= today && Date2 <= today ||

query

Skip code block
 Call us on

 Drop us a Query



```

db.test.find(
{
$or :
[
{$and: [
{"Date1": {"$exists": false}},
{"Date2": {"$exists": false}}
]},
{$and: [
{"Date1": {"$exists": false}},
{"Date2": {
"$exists": true,
"$lte": new Date("2012-01-07T04:45:52.057Z")}
]},
{$and: [
{"Date2": {"$exists": false}},
{"Date1": {
"$exists": true,
"$lte": new Date("2012-01-07T04:45:52.057Z")}
]},
{$and: [
{"Date2": {
"$exists": true,
"$lte": new Date("2012-01-07T04:45:52.057Z")}
},
{"Date1": {
"$exists": true,
"$lte": new Date("2012-01-07T04:45:52.057Z")}
}
]}
]}
]>[ 1 ]

```

this should work too (assume that 'not exist' and 'null' is the same)

Skip code block

```

db.test.find(
{

```

 Call us on

 Drop us a Query

```
[
{$or: [
{"Date1": null},
{"Date1": { "$lte": new Date("2012-01-07T04:45:52.057Z")} }
]},
{$or: [
{"Date2": null},
{"Date2": { "$lte": new Date("2012-01-07T04:45:52.057Z")} }
]}
]
}
)
```

Q131) mongodb: how to do backup of mongodb

I think someone has already suggested:

1. stop the mongod
2. backup the data directory

Is it reliable, I mean, ensure 100% success to restore? And I can't find which directory stores the data... any command can help me to find it?

If mongod process exits cleanly (that is, no crashes or kill -9 stuff), then it is safe to copy data files somewhere.

If your current installation breaks (for example, data corruption due to unclean shutdown), you can delete its files, copy that backup over and start mongod again.

Default data directory is /data/db, but it may be set to another value in your config file. For example, I set it to /var/lib/mongodb.

You can also use mongodump to do a backup from a live server (this may impact performance). Use mongorestore to restore backups made by mongodump.

At IGN we do hot backups via mongodump running as an hourly cron job, and take filer snapshots (NetApp storage) on a half-hourly basis. If your system is not too write heavy and you can afford write blocks, try using fsync and lock to flush the writes to the disk and prevent further writes. This can be followed by mongodump and upon completion you can unlock the db. Please note that you've to be admin to do so.

```
db.runCommand({fsync:1,lock:1})
```

Q132) How to update date field in mongo console?

For example I want to update all records to '2012-01-01' ("time" : ISODate("2011-12-31T13:52:40Z")).

```
db.test.update( { time : '2012-01-01' }, false, true )
```

return error:

Assert failed : need an object

Error("Printing Stack Trace")@:0

Call to @shen/Utils.js:35



✉ Drop us a Query


```
("assert failed : need an object")@shell/utils.js:46
```

```
(false,"need an object")@shell/utils.js:54
```

```
([object Object],false,true)@shell/collection.js:189
```

```
@(shell):1
```

Wed Jan 11 17:52:35 uncaught exception: assert failed : need an object

You need to create a new ISODate object like this:

```
db.test.insert({"Time" : new ISODate("2012-01-10") });
```

This is true both for updates and for queries. Note that your query syntax is incorrect, it should be

```
db.test.update({ criteria }, { newObj }, upsert, multi);
```

For example, to update all objects, consider

```
db.test.update( {}, { $set : { "time" : new ISODate("2012-01-11T03:34:54Z") } }, true, true);
```

Also note that this is very different from

```
db.test.update( {}, { "time" : new ISODate("2012-01-11T03:34:54Z") }, true, false);
```

because the latter will **replace** the object, rather than add a new field to the existing document or updating the existing field. In this example, I changed the last parameter to false, because multi updates only work with \$ operators.

Q133) return query based on date

I have a data like this in mongodb

```
{ "latitude" : "", "longitude" : "", "course" : "", "battery" : "O", "imei" : "O", "altitude" : "F:3.82V", "mcc" : "07", "mnc" : "007B", "lac" : "2A83", "_id" : ObjectId("4f0eb2c406ab6a9d4d000003"), "createdAt" : ISODate("2012-01-12T20:15:31Z") }
```

How to query db.gpsdatas.find({'createdAt': ??what here??}) so that it return me the above result from db?

You probably want to make a range query, for example, all items created after a given date:

```
db.gpsdatas.find({"createdAt" : { $gte : new ISODate("2012-01-12T20:15:31Z") } });
```

I'm using \$gte (greater than or equals), because this is often used for date-only queries, where the time component is 00:00:00.

If you really want to find a date that equals another date, the syntax would be

```
db.gpsdatas.find({"createdAt" : new ISODate("2012-01-12T20:15:31Z") } );
```

Q134) MongoDB query comparing 2 fields in same collection without \$where

Does MongoDB supports comparing two fields in same collection by using native operators (not \$where and JavaScript)? I already looked at similar questions and all answers used \$where / JavaScript.

MongoDB documentation clearly states that:

JavaScript executes more slowly than the native operators listed on this page, but is very flexible.

My primary concern is speed and I would like to use indexes if possible. So is comparing two fields in MongoDB possible without using JavaScript?

This is not currently possible, but it will be possible through the new aggregation framework currently under development (2.1+). This aggregation framework is native and does not rely on relatively slow

📞 Call us on

✉ Drop us a Query

JavaScript execution paths.

For more details check <https://www.mongodb.org/display/DOCS/Aggregation+Framework> and the progress at <https://jira.mongodb.org/browse/SERVER-447>

Q135) Upserts in mongodb when using custom _id values

I need to insert a document if it doesn't exist. I know that the "upsert" option can do that, but I have some particular needs.

First I need to create the document with its _id field only, but only if it doesn't exist already. My _id field is a number generated by me (not an ObjectId). If I use the "upsert" option then I get "Mod on _id not allowed"

```
db.mycollection.update({ _id: id }, { _id: id }, { upsert: true });
```

I know that we can't use the _id in a \$set.

So, my question is: Is there any way to a "create if doesn't exist" atomically in mongodb?

EDIT: As proposed by @Barrie this works (using nodejs and mongoose):

```
var newUser = new User({ _id: id });
newUser.save(function (err) {
  if (err && err.code === 11000) {
    console.log('If duplicate key the user already exists', newTwitterUser);
    return;
  }
  console.log('New user or err', newTwitterUser);
});
```

But I still wonder if it is the best way to do it.

You can just use insert(). If the document with the _id you specify already exists, the insert() will fail, nothing will be modified – so "create if it doesn't exist" is what it's already doing by default when you use insert() with a user-created _id.

I had the same problem, but found a better solution for my needs. You can use that same query style if you simply remove the _id attribute from the update object. So if at first you get an error with this:

```
db.mycollection.update({ _id: id }, {$set: { _id: id, name: 'name' }}, { upsert: true });
```

instead use this:

```
db.mycollection.update({ _id: id }, {$set: { name: 'name' }}, { upsert: true });
```

This is better because it works for both insert and update.

Q136) How to list all collections in the mongo shell?

In the MongoDB shell, how do I list all collections for the current database that I'm using? I can't seem to find it anywhere in the docs.

You can do:

```
db.getCollectionNames()
```

> show collections

will list all the collections in the currently selected DB, as stated in the command line help (help).

👉 [Call us on](#)

📧 [Drop us a Query](#)

It gives the same result as Cameron's answer.

Apart from the options suggested by other people:

```
show collections //output every collection
```

```
show tables
```

```
db.getCollectionNames() //shows all collections as a list
```

There is also another way which can be really handy if you want to know how each of the collections was created (for example it is a capped collection with a particular size)

```
db.system.namespaces.find()
```

To list all databases

```
show dbs
```

enters or uses given database

```
use databasename
```

To list all collections:

```
show collections
```

Output:

```
collection1
```

```
collection2
```

```
system.indexes
```

(or)

```
show tables
```

```
//output
```

```
collection1
```

```
collection2
```

```
system.indexes
```

(or)

```
db.getCollectionNames()
```

```
//output
```

```
[ "collection1", "collection2", "system.indexes" ]
```

To enter or use given collection

```
use collectionname
```

Q137) How to remove one document by Id using the official CSharp driver for Mongo

can someone please show me if there is a better way to remove one document from a MongoDB using the standard 10gen driver for C# than what I have below:

```
var query = Query.EQ("_id", a.Id);
```

```
database.GetCollection("Animal").Remove(query);
```

The above works, but seems dirty and like too much work to me. The "Save" command for instance takes an instance and updates it. Perhaps I'm too used to things like EF/LINQToSQL/NHib where I'd say something like Remove(item). ^

Am I missing something?

📞 Call us on

✉ Drop us a Query

(also, I'm trying to use the official driver rather than NoRM which seems to be dead or Samus which seems equally out of date.)

That's the way you do it. I'm sure you know this, but if you want to put it on one line you could combine it so you don't need to define a query variable:

```
collection.Remove(Query.EQ("_id", a.Id));
```

If the [id] is string, you must use ObjectId instance explicitly.

```
var query = Query.EQ("_id", ObjectId.Parse(id));
```

Q138) Clone a collection in MongoDB

I want to clone a MongoDB collection and save it on the same server with a different name. So for example right now I have the following collections: demo1.categories, demo1.users and demo2.users.

I want to have a "demo2.categories" which is identical to "demo1.categories". (It just has a different name.)

Yet again the MongoDB documentation comes to the rescue

assuming that the collection actually is named "demo1.categories":

```
db.demo1.categories.find().forEach( function(x){db.demo2.categories.insert(x)} );
```

The most simple & efficient way is by using copyTo(), so you can use:

```
db.source.copyTo("target");
```

& if "target" doesn't exist, it will be created

— Update —

According to CopyTo Documentation, Because copyTo() uses eval internally, the copy operations will block all other operations on the mongod instance. So it shouldn't be used on production environment.

Q139) File write operations in mongo script?

Is this possible to write the query result to the file from mongo js script. I have searched a lot, but I didn't find any solution.

ex:-

```
cursor = db.users.find();
```

```
while(cursor.hasNext()) {
```

```
  cursor.next();
```

```
// writing the cursor output to file ????
```

```
}
```

You could use print and then redirect output:

script.js:

```
cursor = db.users.find();
```

```
while(cursor.hasNext()){
```

```
  printjson(cursor.next());
```

```
}
```

then run the script and redirect output to a file:

```
mongo -quiet script.js > result.txt
```

↳ Call us on

<https://www.mongodb.org/display/DOCS/Scripting+the+shell> paragraph "Differences between Scripted

Drop us a Query

and interactive/ Printing”.

```
./mongo server.com/mydb -quiet -eval "db.users.find().forEach(printjson);" > 1.txt
```

you can skip the while loop using forEach :

```
db.users.find().forEach( function(user){ printjson(user) });
```

P.S: I would have added this answer as a comment to @milan's answer but don't have enough reputation yet to comment :p

Q140) Do arrays stored in MongoDB keep their order?

Simple question, do arrays keep their order when stored in MongoDB?

yep MongoDB keeps the order of the array.. just like Javascript engines..

Yes, in fact from a quick google search on the subject, it seems that it's rather difficult to re-order them:

https://groups.google.com/group/mongodb-user/browse_thread/thread/1df1654889e664c1

Q141) MongoDB: update every document on one field

I have a collection named foo hypothetically.

Each instance of foo has a field called lastLookedAt which is a UNIX timestamp since epoch. I'd like to be able to go through the MongoDB client and set that timestamp for all existing documents (about 20,000 of them) to the current timestamp.

What's the best way of handling this?

In the Mongo shell, or with any Mongodb client, you can do something like that:

For Mongodb < 2.2:

```
db.foo.update({}, {$set: {lastLookedAt: Date.now() / 1000}}, false, true)
```

For Mongodb >= 2.2:

```
db.foo.update({}, {$set: {lastLookedAt: Date.now() / 1000}}, { multi: true })
```

See <https://www.mongodb.org/display/DOCS/Updating>

1. {} is the condition (the empty condition matches any document)
2. {\$set: {lastLookedAt: Date.now()}} is what you want to do
3. false is for the "upsert" parameter (insert if not present, or else update - not what you want)
4. true is for the "multi" parameter (update multiple records)

Q142) How to check if Mongodb is properly installed

I installed MongoDB yesterday on a Mac Snow Leopard and got the following error message

Mongo::ConnectionFailure: Failed to connect to a master node at localhost:27017

when trying to run some tests in Rails that used a mongodb.

Another SO question mongo - ruby connection problem about the same error message had an answer that recommended removing the lock file

```
sudo rm /var/lib/mongodb/mongod.lock
```

but when I run that command i'm getting

No such file or directory

Any ideas how I can figure out how to get it working or see if it's properly installed?

Can use

Drop us a Query

The easiest way to run mongodb on Mac OS is:

Download binary package from <https://www.mongodb.org/downloads>, for me, I am using latest 64 bit version (https://fastdl.mongodb.org/osx/mongodb-osx-x86_64-2.0.2.tgz)

1. `mkdir -p $HOME/opt`
2. `cd $HOME/opt`
3. `wget https://fastdl.mongodb.org/osx/mongodb-osx-x86_64-2.0.2.tgz` to download the latest (2.0.2 for now) 64 bit binary package for Mac OS
4. `tar xf mongodb-osx-x86_64-2.0.2.tgz -C $HOME/opt` to unpack the package, and it will be unpacked to `$HOME/opt/mongodb-osx-x86_64-2.0.2`
5. `mkdir -p $HOME/opt/mongoddata` to create the data directory for mongodb
6. `$HOME/opt/mongodb-osx-x86_64-2.0.2/bin/mongod -dbpath=$HOME/opt/mongoddata -logpath=$HOME/opt/mongod.log` to start the mongodb daemon
7. Then you can run `$HOME/opt/mongodb-osx-x86_64-2.0.2/bin/mongo` to connect to your local mongodb service

You can also have <https://www.mongodb.org/display/DOCS/Quickstart+OS+X> as additional reference

Q143) In MongoDB, how to filter by location, sort by another field and paginate the results correctly?

When i don't use pagination, everything works fine (i have only 3 records in this collection, so all of them are listed here):

```
db.suppliers.find({location: {$near: [-23.5968323, -46.6782386]}},{name:1,badge:1}).sort({badge:-1})
{ "_id" : ObjectId("4f33ff549112b9b84f000070"), "badge" : 3, "name" : "Dedetizadora Alvorada" }
{ "_id" : ObjectId("4f33ff019112b9b84f00005b"), "badge" : 2, "name" : "Sampex Desentupidora e
Dedetizadora" }
{ "_id" : ObjectId("4f33feae9112b9b84f000046"), "badge" : 1, "name" : "Higitec Desentupimento e
Dedetizao" }
```

But when i try to paginate from the first to the second page, one record doesn't show up and one is repeated:

```
db.suppliers.find({location: {$near: [-23.5968323, -46.6782386]}},{name:1,badge:1}).sort({badge:-1}).skip(0).limit(2)
{ "_id" : ObjectId("4f33ff549112b9b84f000070"), "badge" : 3, "name" : "Dedetizadora Alvorada" }
{ "_id" : ObjectId("4f33feae9112b9b84f000046"), "badge" : 1, "name" : "Higitec Desentupimento e
Dedetizao" }

db.suppliers.find({location: {$near: [-23.5968323, -46.6782386]}},{name:1,badge:1}).sort({badge:-1}).skip(2).limit(2)
{ "_id" : ObjectId("4f33feae9112b9b84f000046"), "badge" : 1, "name" : "Higitec Desentupimento e
Dedetizao" }
```

Am i doing something wrong or is this some kind of bug?

edit:



Here is a workaround for this. Basically you shouldn't mix \$near queries with sorting; use \$within instead. There is an open issue regarding the same problem. Please have a look & vote Geospatial result paging fails when sorting with additional keys

Q144) Add a new field to a collection with value of an existing field

I'd like to add a new field to a collection, with the value of the new field set to the value of an existing field.

Specifically, I'd like to go from this:

```
# db.foo.findOne()
{
  "_id" : ObjectId("4f25c828eb60261eab000000"),
  "created" : ISODate("2012-01-29T16:28:56.232Z"),
  "..." : ...
}
```

to this:

```
# db.foo.findOne()
{
  "_id" : ObjectId("4f25c828eb60261eab000000"),
  "created" : ISODate("2012-01-29T16:28:56.232Z"),
  "event_ts" : ISODate("2012-01-29T16:28:56.232Z"), #same as created
  "..." : ...
}
```

(New documents in this collection won't all have this peculiar redundancy, but I want to do this for my existing documents)

```
function addEventTsField(){
  db.foo.find().forEach(function(doc){
    db.foo.update({_id:doc._id}, {$set:{"event_ts":doc.created}});
  });
}
```

Run from console:

```
addEventTsField();
```

Q145) Mongoddb update the specific element from subarray

I have a collection with a following schema:

Skip code block

```
{
  "_id" : 28,
  "n" : [{
    "a" : ObjectId("4ef8466e46b3b8140e000000"),
    "c" : 28,
    "p" : ObjectId("4f00640646b3b88005000003"),
    "q" : ObjectId("4f00640146b3b88009000002")
  }
}
```

Call us on 12-Propose a Query

```

ObjectId("4f00637d46b3b8cc0e000001"),
ObjectId("4f00638246b3b8cc0e000003"),
ObjectId("4f00631846b3b85002000002")],
"u" : 26
}, {
"a" : ObjectId("4ef8466e46b3b8140e000000"),
"c" : 10,
"p" : [ObjectId("4f00640146b3b88005000002"), ObjectId("4f0063fd46b3b88005000001")],
"u" : 26
}, {
"a" : ObjectId("4ef8467846b3b8780d000001"),
"u" : 26,
"p" : [ObjectId("4f00637b46b3b8cc0e000000")],
"c" : 28
}, {
"a" : ObjectId("4ef85a3e46b3b84408000000"),
"u" : 26,
"p" : [ObjectId("4f00631046b3b85002000000")],
"c" : 28
}]
}

```

I need to update one of the elements in the array in the document with `_id = 28` but only if the `a =` to some value and `c =` some value

Skip code block

```

db.coll.update({
  '_id' : 28,
  'n.a' : new ObjectId('4ef85a3e46b3b84408000000'),
  'n.c' : 28
},
{
  $push : {
    'n.$p' : ObjectId("4b97e62bf1d8c7152c9ccb74")
  },
  $set : {
    'n.$t' : ISODate("2013-05-13T14:22:46.777Z")
  }
})

```

So basically I want to update specific element from array: and as far as one can see, this is the fourth element. The problem is that when the query is executing, it most likely updates the first element. ^

How can I fix it?

Call us on

✉ Drop us a Query

The problem in your code is dot-notation because When you specify the dot notation you assume that the filter criterias specified must match the single array element that satisfies all the criteria. But it doesnt. Dot notation on arrays may pickup any array element if any single criteria matches. Thats why you are getting the unexpected update.

You have to use \$elemMatch to match all the filters in the array element.

Skip code block

```
db.coll.update({
  '_id' : 28,
  n: {
    $elemMatch:{
      a : new ObjectId('4ef85a3e46b3b84408000000'),
      c : 28 }
    },
  {
    $push : {
      'n.$p' : ObjectId("4b97e62bf1d8c7152c9ccb74")
    },
    $set : {
      'n.$t' : ISODate("2013-05-13T14:22:46.777Z")
    }
  })
```

and the output is

```
{
  "a" : ObjectId("4ef85a3e46b3b84408000000"),
  "c" : 28,
  "p" : [
    ObjectId("4f00631046b3b85002000000"),
    ObjectId("4b97e62bf1d8c7152c9ccb74")
  ],
  "t" : ISODate("2013-05-13T14:22:46.777Z"),
  "u" : 26
}
```

Q146) How can I rename a field for all documents in MongoDB?

Assuming I have a collection in MongoDB with 5000 records, each containing something similar to:

```
{
  "occupation":"Doctor",
  "name": {
    "first":"Jimmy",
    Call us on
```



✉ Drop us a Query

```
"additional": "Smith"
```

```
}
```

Is there an easy way to rename the field "additional" to "last" in all documents? I saw the \$rename operator in the documentation but I'm not really clear on how to specify a subfield.

Not tested, but you can try:

```
db.foo.update({}, {$rename: {"name.additional": "name.last"}}, false, true);
```

If this does not work, maybe you have to use the former way:

```
remap = function (x) {
```

```
  if (x.additional){
```

```
    db.foo.update({_id:x._id}, {$set: {"name.last": x.name.additional}, $unset: {"name.additional": 1}});
```

```
  }
```

```
}
```

```
db.foo.find().forEach(remap);
```

please try `db.collectionName.update({}, { $rename : { 'name.additional' : 'name.last' } }, { multi: true })`

and read this :) https://docs.mongodb.org/manual/reference/operator/rename/#_S_rename

Q147) Mongoddb field not null delete

How do I remove all documents in a collection where a field's value isn't null? Basically the MySQL query version would be like this:

```
// MySQL query
```

```
DELETE FROM companies WHERE createdBy != NULL
```

```
// What I tried but did not work.
```

```
$this->mongo->companies->remove(array('createdBy' => true));
```

I don't even know if it is possible, if anyone could help me with this I would appreciate it

Thanks :)

You can do it easy via not equal operator:

```
db.companies.find( { createdBy : { $ne : null } } );
```

Q148) How can I see what ports mongo is listening on from mongo shell?

If I have a mongo instance running, how can I check what port numbers it is listening on from the shell? I thought that `db.serverStatus()` would do it but I don't see it. I see this

```
"connections" : {
```

```
  "current" : 3,
```

```
  "available" : 816
```

Which is close... but no. Suggestions? I've read the docs and can't seem to find any command that will do this.

From the system shell you can use `lsof` (see Derick's answer below) or `netstat -an` to view what a process is actually doing. However, assuming you only have access to the mongo shell (which your question title implies), then you can run the `serverCmdLineOpts()` command. That output will give you all the arguments passed on the command line (`argv`) and the ones from the config file (parsed) and you can infer the ports mongod is listening based on that information. Here's an example: [Drop us a Query](#)

Skip code block

```
db.serverCmdLineOpts()
{
  "argv" : [
    "./mongod",
    "-replSet",
    "test",
    "-rest",
    "-dbpath",
    "/data/test/r1",
    "-port",
    "30001"
  ],
  "parsed" : {
    "dbpath" : "/data/test/r1",
    "port" : 30001,
    "replSet" : "test",
    "rest" : true
  },
  "ok" : 1
}
```

If you have not passed specific port options like the ones above, then the mongod will be listening on 27017 and 28017 (http console) by default. Note: there are a couple of other arguments that can alter ports without being explicit, see here:

<https://docs.mongodb.org/manual/administration/security/#interfaces-and-port-numbers>

You can do this from the Operating System shell by running:

```
sudo lsof -iTCP -sTCP:LISTEN | grep mongo
```

MongoDB only listens on one port by default (27017). If the `-rest` interface is active, port 28017 (27017+1000) will also be open handling web requests for details.

MongoDB supports a `getParameter` command, but that only works if you're already connected to the Database (at which point you already know the port).

Q149) mongodb indexing embedded fields (dot notation)

Let's assume this is a document representing a customer.

```
{
  company_name: 'corporate ltd.',
  pocs: [
    {name: 'Paul', email: 'paul@corporate.com'},
    {name: 'Jessica', email: 'jessica@corporate.com'}
  ]
}
```

📞 Call us on

✉ Drop us a Query

```
}
```

I wanted to define a unique index for pocs.email So I issued the following command:

```
db.things.ensureIndex({"pocs.email": 1}, {unique: true})
```

The strange thing is that when trying to add another company with a poc having an email already exists in another company, mongo rejects that, respecting the unique index constraint.

that is, the following cannot exists:

Skip code block

```
{
  company_name: 'corporate ltd.',
  pocs: [
    {name: 'Paul', email: 'paul@corporate.com'},
    {name: 'Jessica', email: 'jessica@corporate.com'}
  ]
},
{
  company_name: 'contoso llc',
  pocs: [
    {name: 'Paul', email: 'paul@corporate.com'},
  ]
}
```

Which is fine. However, having duplicate poc within the same doc is possible, e.g.

```
{
  company_name: 'corporate ltd.',
  pocs: [
    {name: 'Paul', email: 'paul@corporate.com'},
    {name: 'Paul', email: 'paul@corporate.com'},
    {name: 'Jessica', email: 'jessica@corporate.com'}
  ]
},
```

see my cli commands sequence below:

Skip code block

```
> version()
version: 2.0.2
>
> use test
switched to db test
> db.test.ensureIndex({"poc.email": 1}, {unique: true})
>
> db.test.insert({company: "contoso", poc: [{email: 'me@comapny.com'}]})
> db.test.insert({company: "contoso", poc: [{email: 'me@comapny.com'}]})
```

Call us on

Drop us a Query

```
E11000 duplicate key error index: test.test.$poc.email_1 dup key: { : "me@comapny.com" }
> ({company: "contoso", poc: [{email: 'me.too@comapny.com'}, {email: 'me.too@company.com'}]})
>
>
> db.test.find()
{ "_id" : ObjectId("4f44949685926af0ecf9295d"), "company" : "contoso", "poc" : [ { "email" :
"me@comapny.com" } ] }
{ "_id" : ObjectId("4f4494b885926af0ecf9295f"), "company" : "contoso", "poc" : [ { "email" :
"me.too@comapny.com" }, { "email" : "me.too@company.com" } ] }
Moreover, this happens either at insert or at update.
> db.test.update({"_id" : ObjectId("4f44949685926af0ecf9295d")}, {$push: { poc: {email:
'me@company.com'}}})
> db.test.find()
{ "_id" : ObjectId("4f4494b885926af0ecf9295f"), "company" : "contoso", "poc" : [ { "email" :
"me.too@comapny.com" }, { "email" : "me.too@company.com" } ] }
{ "_id" : ObjectId("4f44949685926af0ecf9295d"), "company" : "contoso", "poc" : [ { "email" :
"me@comapny.com" }, { "email" : "me@company.com" }, { "email" : "me@company.com" } ] }
>
```

Is this a **bug** or a **by-design-feature** I missed spotting in the documentation?

There is an open issue regarding the same problem unique indexes not enforced within array of single document . You can vote for it.

Also there is a nice workaround suggested by Kyle Banker in this similar post Unique indexes on embedded documents

Update

This is not only related to the embedded fields, we can reproduce the same for array fields too.

```
>db.uniqueTest.insert({a:[1],x:1})
>db.uniqueTest.createIndex({a:1}, {unique: true})
> db.uniqueTest.find()
{ "_id" : ObjectId("4f44c6252434860b44986b02"), "a" : [ 1 ], "x":1 }
and it throws an error if we try to create a new document with the same value (correct behavior )
> db.uniqueTest.insert({a:[1],x:3})
```

```
E11000 duplicate key error index: stack.uniqueTest.$a_1 dup key: { : 1.0 }
```

But this works fine if we put the same value inside the array (no errors, silently accepts the duplicate value inside the array)

```
> db.uniqueTest.insert({a:[2],x:2})
> db.uniqueTest.update({x:2},{ $push:{a:2}})
{ "_id" : ObjectId("4f44c65f2434860b44986b05"), "a" : [ 2, 2 ], "x" : 2 }
```

But not for this

```
> db.uniqueTest.update({x:2},{ $push:{a:1}})
```

```
E11000 duplicate key error index: stack.uniqueTest.$a_1 dup key: { : 1.0 }
```

☎ Call us on

✉ Drop us a Query



Q150) Updating a sub-document in mongodb?

How do I target an subdocument in the authors array as shown below, in order to update it?

```
collection.update({'_id': "4f44af6a024342300e000001"}, {$set: { 'authors.?' }} )
```

The document:

```
{
  _id: "4f44af6a024342300e000001",
  title: "A book",
  created: "2012-02-22T14:12:51.305Z"
  authors: [{ "_id": "4f44af6a024342300e000002?"}]
}
```

By specifying actual position of embedded document like this:

```
// update _id field of first author
collection.update({'_id': "4f44af6a024342300e000001"},
{$set: { 'authors.0._id': "1" }} )
```

Or via positional operator:

```
// update _id field of first matched by _id author
collection.update({'_id': "4f44af6a024342300e000001",
//you should specify query for embedded document
'authors._id' : "4f44af6a024342300e000002" },
// you can update only one nested document matched by query
{$set: { 'authors.$. _id': "1" }} )
```

Q151) Update MongoDB collection using \$toLower

I have an existing MongoDB collection containing user names. The user names contain both lower case and upper case letters. I want to update all the user names so they only contain lower case letters. I have tried this script, but it didn't work

```
db.myCollection.find().forEach(
function(e) {
e.UserName = $toLower(e.UserName);
db.myCollection.save(e);
}
)
```

Any information on getting this to work will be really appreciated, Scott

MongoDB does not have a concept of \$toLower as a command. The solution is to run a big forloop over the data and issue the updates individually.

You can this in any driver or from the shell:

```
db.myCollection.find().forEach(
function(e) {
e.UserName = e.UserName.toLowerCase();
```

```
db.myCollection.save(e);
```



Call us on

✉ Drop us a Query

}

)

You can also replace the save with an atomic update:

```
db.myCollection.update({_id: e._id}, {$set: {UserName: e.UserName.toLowerCase() } })
```

Again, you could also do this from any of the drivers, the code will be very similar.

EDIT: Remon brings up a good point. The \$toLowerCase command does exist as part of the aggregation framework, but this has nothing to do with updating. The documentation for updating is here.

Q152) Mongo ObjectIDs: Safe to use in the wild?

I'm designing an API that interacts with MongoDB.

Now the question is, if it is **safe** to use the raw ObjectID to query for objects etc. Could any security issues arise when using the OIDs directly (e.g. in queries), or should I encrypt/decrypt them before they leave my server environment?

Look at BSON Object ID specification here and you will know **if it is safe for you to use**.

If you try to protect from users sending different URLs from scripts (fuskators) then it seems for me it has weak security. There won't be too many 'machine', 'pid' part combinations. 'time' part can be calculated if attacker can have an idea how data was inserted (especially if using batch). 'inc' – very weak.

I won't trust ObjectIDs as the only security.

Please note there can't be a right answer to the question "is it safe" in general. You must decide yourself.

1. But keep in mind that such URL-based security will fall to dust when users will share URLs they visited. Even best your encryption won't help.

Q153) In MongoDB, How to toggle a boolean field in one document with atomic operation?

Is there any way to toggle the boolean field of ONE document in MongoDB with atomic operation? Say, (In python)

```
cl.update({"_id": ...}, {"$toggle": {"field": 1}})
```

Right now, I don't think it's possible to do this with one operation. The bitwise operators (<https://www.mongodb.org/display/DOCS/Updating#Updating-%24bit>) don't have a '\$xor' yet although I've a patch for it.

Right now the workaround I think of is by always using '\$inc':

```
cl.update( { "_id": ... }, { '$inc' : { 'field' : 1 } } );
```

Then instead of checking for true or false, you can do check whether an item is "true":

```
cl.find( { "_id": ..., 'field' : { '$mod' : [ 2, 1 ] } } );
```

IE, you using the modulo operator to see whether it's even or uneven with even being "unset", and uneven being "set". If you want to have the opposite behaviour (ie, find all items that don't have the flag set), then use

```
[ 2, 0 ];
```

Q154) Working with special characters in a Mongo collection



I have a collection I'm unable to drop, I'm assuming that the "-" in its name is a special character. In MongoDB, what is the best way to escape special characters?

```
> db.tweets.drop();
```

true

BUT

```
> db.tweets-old.drop();
```

ReferenceError: old is not defined (shell):1

I've tried to escape with quotes (both single and double) and a slash, but nothing works.

The following works:

```
db["tweets-old"].drop();
```

It's called the square bracket notation, which allows you to use special characters in property names.

Q155) Add new field to all documents in a nested array

I have a database of person documents. Each has a field named photos, which is an array of photo documents. I would like to add a new 'reviewed' flag to each of the photo documents and initialize it to false.

This is the query I am trying to use:

```
db.person.update({ "_id" : { $exists : true } }, { $set : { photos.reviewed : false } }, false, true)
```

However I get the following error:

SyntaxError: missing : after property id (shell):1

Is this possible, and if so, what am I doing wrong in my update?

Here is a full example of the 'person' document:

Skip code block

```
{
  "_class" : "com.foo.Person",
  "_id" : "2894",
  "name" : "Pixel Spacebag",
  "photos" : [
    {
      "_id" : null,
      "thumbUrl" : "https://site.com/a_s.jpg",
      "fullUrl" : "https://site.com/a.jpg"
    },
    {
      "_id" : null,
      "thumbUrl" : "https://site.com/b_s.jpg",
      "fullUrl" : "https://site.com/b.jpg"
    }
  ]
}
```



Bonus karma for anyone who can tell me a cleaner way to update "all documents" without using the
 Call us on Drop us a Query


```
query { “_id” : { $exists : true } }
```

Is this possible, and if so, what am I doing wrong in my update?

No. In general MongoDB is only good at doing updates on top-level objects.

The exception here is the \$ positional operator. From the docs: Use this to find an array member and then manipulate it.

However, in your case you want to modify all members in an array. So that is not what you need.

Bonus karma for anyone who can tell me a cleaner way to update “all documents”

Try `db.coll.update(query, update, false, true)`, this will issue a “multi” update. That `lasttrue` is what makes it a multi.

Is this possible,

You have two options here:

1. Write a for loop to perform the update. It will basically be a nested for loop, one to loop through the data, the other to loop through the sub-array. If you have a lot of data, you will want to write this in your driver of choice (and possibly multi-thread it).
2. Write your code to handle `reviewed` as nullable. Write the data such that if it comes across a photo with `reviewed` undefined then it must be false. Then you can set the field appropriately and commit it back to the DB.

Method #2 is something you should get used to. As your data grows and you add fields, it becomes difficult to “back-port” all of the old data. This is similar to the problem of issuing a schema change in SQL when you have 1B items in the DB.

Instead just make your code resistant against the null and learn to treat it as a default.

Again though, this is still not the solution you seek.

Q156) How to listen for changes to a MongoDB collection?


I’m creating a sort of background job queue system with MongoDB as the data store. How can I “listen” for inserts to a MongoDB collection before spawning workers to process the job? Do I need to poll every few seconds to see if there are any changes from last time, or is there a way my script can wait for inserts to occur? This is a PHP project that I am working on, but feel free to answer in Ruby or language agnostic.

What you are thinking of sounds a lot like triggers. MongoDB does not have any support for triggers, however some people have “rolled their own” using some tricks. The key here is the oplog.

When you run MongoDB in a Replica Set, all of the MongoDB actions are logged to an operations log (known as the oplog). The oplog is basically just a running list of the modifications made to the data. Replica Sets function by listening to changes on this oplog and then applying the changes locally.


Does this sound familiar?

I cannot detail the whole process here, it is several pages of documentation, but the tools you need are available.

First some write-ups on the oplog – Brief description – Layout of the local collection (which contains the oplog) 

You will also want to leverage tailable cursors. These will provide you with a way to listen for changes

 Call us on

 Drop us a Query

instead of polling for them. Note that replication uses tailable cursors, so this is a supported feature. MongoDB has what is called capped collections and tailable cursors that allows MongoDB to push data to the listeners.

A capped collection is essentially a collection that is a fixed size and only allows insertions. Here's what it would look like to create one:

```
db.createCollection("messages", { capped: true, size: 100000000 })
```

Ruby example of using tailable cursors:

```
coll = db.collection('my_collection')
cursor = Mongo::Cursor.new(coll, :tailable => true)
loop do
  if doc = cursor.next_document
    puts doc
  else
    sleep 1
  end
end
```

Additional Resources:

Ruby/Node.js Tutorial which walks you through creating an application that listens to inserts in a MongoDB capped collection.

An article talking about tailable cursors in more detail.

PHP, Ruby, Python, and Perl examples of using tailable cursors.

Q157) MongoDB – Query on nested field with index

I am trying to figure out how I must structure queries such that they will hit my index. I have documents structured like so:

```
{ "attributes" : { "make" : "Subaru", "color" : "Red" } }
```

With an index of: `db.stuff.ensureIndex({"attributes.make":1})`

What I've found is that querying using dot notation hits the index while querying with a document does not.

Example:

Skip code block

```
db.stuff.find({"attributes.make":"Subaru"}).explain()
```

```
{
  "cursor" : "BtreeCursor attributes.make_1",
  "nscanned" : 2,
  "nscannedObjects" : 2,
  "n" : 2,
  "millis" : 0,
  "nYields" : 0,
```

```
"nChunkSkips" : 0,
```

📞 Call us on



✉ Drop us a Query

```

"isMultiKey" : false,
"indexOnly" : false,
"indexBounds" : {
  "attributes.make" : [
    "Subaru",
    "Subaru"
  ]
}
}

```

vs

Skip code block

```

db.stuff.find({attributes:{make:"Subaru"}}).explain()
{
  "cursor" : "BasicCursor",
  "nscanned" : 2,
  "nscannedObjects" : 2,
  "n" : 0,
  "millis" : 1,
  "nYields" : 0,
  "nChunkSkips" : 0,
  "isMultiKey" : false,
  "indexOnly" : false,
  "indexBounds" : {
  }
}

```

Is there a way to get the document style query to hit the index? The reason is that when constructing queries from my persistent objects it's much easier to serialize them out as documents as opposed to something using dot notation.

I'll also add that we're using a home grown data mapper layer built w/ Jackson. Would using something like Morphia help with properly constructing these queries?

Did some more digging and this thread explains what's going with the sub-document query. My problem above was that to make the sub-document based query act like the dot-notation I needed to use elemMatch.

Skip code block

```

db.stuff.find({"attributes":{"$elemMatch" : {"make":"Subaru"}}}).explain()
{
  "cursor" : "BtreeCursor attributes.make_1",
  "nscanned" : 2,

```

Call us on



✉ Drop us a Query

```

"nscannedObjects" : 2,
"n" : 0,
"millis" : 2,
"nYields" : 0,
"nChunkSkips" : 0,
"isMultiKey" : false,
"indexOnly" : false,
"indexBounds" : {
  "attributes.make" : [
    [
      "Subaru",
      "Subaru"
    ]
  ]
}
}

```

Q158) Save Subset of MongoDB Collection to Another Collection

I have a set like so

```

{date: 20120101}
{date: 20120103}
{date: 20120104}
{date: 20120005}
{date: 20120105}

```

How do I save a subset of those documents with the date '20120105' to another collection?

i.e `db.subset.save(db.full_set.find({date: "20120105"}));`

Here's the shell version:

```

db.full_set.find({date:"20120105?}).forEach(function(doc){
db.subset.insert(doc);
});

```

There's no direct equivalent of SQL's insert into ... select from

You have to take care of it yourself. Fetch documents of interest and save them to another collection.

You can do it in the shell, but I'd use a small external script in Ruby. Something like this:

require 'mongo'

```

db = Mongo::Connection.new.db('mydb')
source = db.collection('source_collection')
target = db.collection('target_collection')
source.find(date: "20120105").each do |doc|
target.insert doc

```

end
📞 Call us on



✉ Drop us a Query

Actually, there is an equivalent of SQL's insert into ... select from in MongoDB. First, you convert multiple documents into an array of documents; then you insert the array into the target collection

```
db.subset.insert(db.full_set.find({date:"20120105?}).toArray())
```

Q159) What characters are NOT allowed in MongoDB field names?

I figured out that of course . and SPACE aren't allowed. Are there other forbidden characters ?

You can use any (UTF8) character in the field name which aren't special (contains ".", or starts with "\$").

<https://jira.mongodb.org/browse/SERVER-3229>

<https://stackoverflow.com/a/7976235/311220>

It's generally best to stick with lowercase alphanumeric with underscores though.

Q160) Is MongoDB somehow limited to a single core?

Perhaps I have misconfigured MongoDB somehow, but even under heavy load I don't see it using more than one core. For example, top is currently showing:

Skip code block

```
Tasks: 145 total, 1 running, 144 sleeping, 0 stopped, 0 zombie
Cpu0 : 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu1 : 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu2 : 0.0%us, 0.0%sy, 0.0%ni,100.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu3 :100.0%us, 0.0%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 41182768k total, 40987476k used, 195292k free, 109956k buffers
Swap: 2097144k total, 1740288k used, 356856k free, 28437928k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
16297 mongod 15 0 521g 18g 18g S 99.8 47.2 2929:32 mongod
1 root 15 0 10372 256 224 S 0.0 0.0 0:03.39 init
```

Is there something I can do to get Mongo to use the other cores more effectively? If it's relevant, I currently have a big M/R running which seems to have put a lot of read queries in "waiting" mode.

MongoDB can saturate all cores on a multi-core machine for read operations, but for write operations and map-reduce MongoDB can only utilize a single core per mongod process.

The limitation for single-core MapReduce is due to the Javascript interpreter that MongoDB utilizes. This is something that is supposed to be fixed in the future but in the interim you can use Hadoop to execute the MapReduce and store the result set in your MongoDB database.

Another option which has seen mixed results is to run a single mongod process for every core on the instance this will not increase performance for a single database unless they are configured to run in a sharded setup.

Q161) Mongo count really slow when there are millions of records

```
//FAST
```

```
db.datasources.find().count()
```

```
12036788
```

```
//SLOW
db.count('sources')
```



✉ Drop us a Query

```
db.datasources.find({nid:19882}).count()
```

```
10161684
```

Index on nid

Any way to make the second query faster? (It is taking about 8 seconds)

Count queries, indexed or otherwise, are slow due to the fact that MongoDB still has to do a full b-tree walk to find the appropriate number of documents that match your criteria. The reason for this is that the MongoDB b-tree structure is not “counted” meaning each node does not store information about the amount of elements in the node/subtree.

The issue is reported here <https://jira.mongodb.org/browse/SERVER-1752> and there is currently no workaround to improve performance other than manually maintaining a counter for that collection which obviously comes with a few downsides.

Also note that the `db.col.count()` version (so no criteria) can take a big shortcut and doesn’t actually perform a query hence it’s speed. That said it does not always report the same value as a count query would that should return all elements (it won’t be in sharded environments with high write throughput for example). Up for debate whether or not that’s a bug. I think it is.

Note that in 2.3+ a significant optimization was introduced that should (and does) improve performance of counts on indexed fields. See: <https://jira.mongodb.org/browse/SERVER-7745>

Q162) MongoDB not equal to

I’m trying to display a query in MongoDB where a text field is not ” (blank)

```
{ 'name' : { $not : " " }}
```

However I get the error invalid use of \$not

I’ve looked over the documentation but the examples they use are for complicated cases (with regexp and \$not negating another operator).

How would I do the simple thing I’m trying to do?

Use \$ne — \$not should be followed by the standard operator:

An examples:

use test

switched to db test

```
db.test.insert({author : 'me', post: ""})
```

```
db.test.insert({author : 'you', post: "how to query"})
```

```
db.test.find({'post': {$ne : ""}})
```

```
{ "_id" : ObjectId("4f68b1a7768972d396fe2268"), "author" : "you", "post" : "how to query" }
```

And now \$not:

```
db.test.find({'post': {$not: {$ne : ""}}})
```

```
{ "_id" : ObjectId("4f68b19c768972d396fe2267"), "author" : "me", "post" : "" }
```

Use \$ne instead of \$not

https://docs.mongodb.org/manual/reference/operator/ne/#op._S_ne



```
db.collections.find({"name": {"$ne": ""}});
```

If you want to do multiple \$ne then do

```
db.users.find({name : {"$nin": ["mary", "dick", "jane"]}})
```

Q163) Multiple \$elemMatch expressions for matching array values using \$all in MongoDB?

In the answer to a question I found a interesting solution for searching array values using \$elemMatch.

If we have the following documents in our collection:

```
{
  foo : [ { bar : "xy", baz : 1 },
  { bar : "a", baz : 10 } ]
},
{
  foo : [ { bar : "xy", baz : 5 },
  { bar : "b", baz : 50 } ]
}
```

The following query will match only the first document:

```
db.test.find({
  foo : { "$all" : [ { "$elemMatch" : { bar : "xy", baz : 1 } }, { "$elemMatch" : { bar : "a", baz: 10 } } ] }
});
```

I tried it with several other examples and it really works. But the official documentation for \$all operator doesn't say anything about combining these two queries.

Is this the intended behavior or a bug? Or is this just a problem that the documentation does not cover this use case?

This is the intended behavior. The documentation doesn't cover this use case and we are working on it to make it better. Its difficult, however, to document every possible query combination.

Q164) How to remove duplicate entries from an array?

How to remove duplicate entries from an array?

In below example "Algorithms in C++" is added twice.

\$unset modifier removes a particular field but how to remove an entry from a field?

```
> db.users.find()
{ "_id" : ObjectId("4f6cd3c47156522f4f45b26f"),
  "favorites" : { "books" : [ "Algorithms in C++",
    "The Art of Computer Programmning",
    "Graph Theory",
    "Algorithms in C++" ] },
  "name" : "robert" }
```

What you have to do is use map reduce to detect and count duplicate tags .. then use \$set to replace the entire books based on { "_id" : ObjectId("4f6cd3c47156522f4f45b26f"),

This has been discussed sevel times here .. please seee

[Removing duplicate records using MapReduce](#)

 [Drop us a Query](#)

Fast way to find duplicates on indexed column in mongodb

<https://csanz.posterous.com/look-for-duplicates-using-mongodb-mapreduce>

<https://www.mongodb.org/display/DOCS/MapReduce>

How to remove duplicate record in MongoDB by MapReduce?

Thanks :)

As of MongoDB 2.2 you can use the aggregation framework with an \$unwind, \$group and \$project stage to achieve this:

```
db.users.aggregate([{$unwind: '$favorites.books'},
{$group: {_id: '$_id',
books: {$addToSet: '$favorites.books'},
name: {$first: '$name'}}},
{$project: {'favorites.books': '$books', name: '$name'}}
])
```

Note the need for the \$project to rename the favorites field, since \$group aggregate fields cannot be nested.

Q165) Order of \$lt and \$gt in MongoDB range query

Today I have noticed that the order in which the \$lt and \$gt operators are given seem to matter in MongoDB 2.0.2.

I have a database of games. "player" is an array of two strings representing both players, "endedAtMS" is a timestamp when the game has ended. I have created this index:

```
db.games.ensureIndex({player:1,endedAtMS:-1})
```

To get 30 of my games which were finished in a certain time range, ordered by the time the games where finished, I do:

Skip code block

```
db.games.find({ "player" : "Stefan" ,
"endedAtMS" : { "$lt" : 1321284969946 ,
"$gt" : 1301284969946}}).
sort({endedAtMS:-1}).
limit(30).
explain()
{
"cursor" : "BtreeCursor player_1_endedAtMS_-1",
"nscanned" : 30,
"nscannedObjects" : 30,
"n" : 30,
"millis" : 0,
"nYields" : 0,
"nChunkSkips" : 0,
"isMultiKey" : true,
Call us on
```



✉ Drop us a Query


```

“indexOnly” : false,
“indexBounds” : {
“player” : [
[
“Stefan”,
“Stefan”
]
],
“endedAtMS” : [
[
1321284969946,
-1.7976931348623157e+308
]
]
}
}

```

All seems to work fine. However when I change the order of \$lt and \$gt in the query above I get this:

Skip code block

```

db.games.find({ “player” : “Stefan” ,
“endedAtMS” : { “$gt”:1301284969946,
“$lt” : 1321284969946}}).
sort({endedAtMS:-1}).
limit(30).
explain()
{
“cursor” : “BtreeCursor player_1_endedAtMS_-1”,
“nscanned” : 126,
“nscannedObjects” : 126,
“n” : 30,
“millis” : 1,
“nYields” : 0,
“nChunkSkips” : 0,
“isMultiKey” : true,
“indexOnly” : false,
“indexBounds” : {
“player” : [
[
“Stefan”,
“Stefan”

```



Call us on

Drop us a Query

```

],
"endedAtMS" : [
[
1.7976931348623157e+308,
1301284969946
]
]
}
}

```

As you can see 126 docs need to be scanned to get the 30 docs for the result. If you take a look at the indexBounds in the explain output it seems that only the first operator is used to limit the search space in the index.

What do I miss? Why is Mongo only using one operator to limit the search space?

This is a known issue. The short answer is that it has to do with the fact that a multikey index is used ("player" is an array), and the index cannot be constrained on both upper and lower bounds.

This is explained in more detail in the Jira case: <https://jira.mongodb.org/browse/SERVER-4155> - "Index bound incorrect?"

There is an open Jira ticket to improve this behavior: <https://jira.mongodb.org/browse/SERVER-4180> - "Wrong indexbounds picked for a date range query (regression)" which is slated to be released in version 2.1.2 (this version is subject to change). Please vote for it!

Q166) Foreign key like relationship in Mongo DB

How can I implement a foreign key like relationship in Mongo DB?

hiya see this: MongoDB normalization, foreign key and joining && further
 ==>https://books.google.com/books/about/Document_Design_for_MongoDB.html?id=TbIHkgEACAAJ&redir_esc=y
 hope it helps! cheerios!

MongoDB doesn't support server side foreign key relationships, normalization is also discouraged. You should embed your child object within parent objects if possible, this will increase performance and make foreign keys totally unnecessary. That said it is not always possible, so there is a special construct called DBRef which allows to reference objects in a different collection. This may be then not so speedy because DB has to make additional queries to read objects but allows for kind of foreign key reference. Still you will have to handle your references manually. Only while looking up your DBRef you will see if it exists, the DB will not go through all the documents to look for the references and remove them if the target of the reference doesn't exist any more. But I think removing all the references after deleting the book would require a single query per collection, no more, so not that difficult really.

Edit update

<https://levycarneiro.com/tag/mongodb/>



levycarneiro.com/tag/mongodb [quote] So you create 4 collections: Clients, Suppliers, Employees and Contacts. You connect them all together via a db reference. This acts like a foreign key. But, this is not the mongoDB way to do things. Performance will penalized. [unquote]

Q167) How to get a specific embedded document inside a MongoDB collection?

I have a collection Notebook which has embedded array document called Notes. The sample document looks like as shown below.

Skip code block

```
{
  "_id" : ObjectId("4f7ee46e08403d063ab0b4f9"),
  "name" : "MongoDB",
  "notes" : [
    {
      "title" : "Hello MongoDB",
      "content" : "Hello MongoDB"
    },
    {
      "title" : "ReplicaSet MongoDB",
      "content" : "ReplicaSet MongoDB"
    }
  ]
}
```

I want to find out only note which has title "Hello MongoDB". I am not getting what should be the query. Can anyone help me.

I don't believe what you are asking is possible, at least without some map-reduce maybe?

See here: [Filtering embedded documents in MongoDB](#)

That answer suggests you change your schema, to better suit how you'd like to work with the data.

You can use either "dot notation" or \$elemMatch to get back the correct, document that has the matching "note title" ...

```
> db.collection.find({ "notes.title" : "Hello MongoDB"}, { "notes.title" : 1?});
```

or ...

```
> db.collection.find({ "notes" : { "$elemMatch" : { "title" : "Hello MongoDB" } } });
```

But you will get back the whole array, not just the array element that caused the match.

Also, something to think about ... with your current setup it would be hard to do any operations on the items in the array.

If you don't change your schema (as the answer linked to suggests) ... I would consider adding "ids" to each element in the array so you can do things like delete it easily if needed.

You can do this with mongo version higher 2.2



the query like this:

```
db.coll.find({ 'notes.title': 'Hello MongoDB' }, {'notes.$': 1});
```

you can try with \$elemMatch like Justin Jenkins

Q168) can you have mongo \$push prepend instead of append?

I'd like to have push add at the beginning of my set rather than appended to the back when I do a mongo \$push.

Is it possible to do an atomic push update that adds it as the first element rather than the last?

A similar question was asked a few days ago. Unfortunately, the short answer is, "no", but there is an open request for this feature.

<https://jira.mongodb.org/browse/SERVER-2191> - "\$push() to front of array"

There is some more information as well as a possible work-around on the other thread: "Use MongoDB array as stack" - Use MongoDB array as stack

Hopefully the above will be useful and help you to find an acceptable work-around.

Use negative index with \$set for prepend, tested in mongo v2.2:

Skip code block

```
> db.test.insert({'array': [4, 5, 6]})
```

```
> db.test.find()
```

```
{ "_id" : ObjectId("513ad0f8afdf1e6736e49eb"),
```

```
  "array" : [ 4, 5, 6 ] }
```

```
//prepend 3
```

```
> db.test.update({"_id" : ObjectId("513ad0f8afdf1e6736e49eb")},
```

```
{ '$set': {'array.-1': 3} })
```

```
> db.test.find()
```

```
{ "_id" : ObjectId("513ad0f8afdf1e6736e49eb"),
```

```
  "array" : [ 3, 4, 5, 6 ] }
```

```
//prepend 2
```

```
> db.test.update({"_id" : ObjectId("513ad0f8afdf1e6736e49eb")},
```

```
{ '$set': {'array.-1': 2} })
```

```
> db.test.find()
```

```
{ "_id" : ObjectId("513ad0f8afdf1e6736e49eb"),
```

```
  "array" : [ 2, 3, 4, 5, 6 ] }
```

```
//prepend 1
```

```
> db.test.update({"_id" : ObjectId("513ad0f8afdf1e6736e49eb")},
```

```
{ '$set': {'array.-1': 1} })
```

```
> db.test.find()
```

```
{ "_id" : ObjectId("513ad0f8afdf1e6736e49eb"),
```

```
  "array" : [ 1, 2, 3, 4, 5, 6 ] }
```

As of MongoDB v2.5.3, there is a new \$position operator that you can include along with the \$each operator as part of your \$push query to specify the location in the array at which you would like to insert

📞 Call us on

✉ Drop us a Query

a value.

Here's an example from the docs page to add the elements 20 and 30 at the array index of 2::

```
db.students.update( { _id: 1 },
{ $push: { scores: {
$each: [ 20, 30 ],
$position: 2
}
}
}
)
```

Reference:https://docs.mongodb.org/master/reference/operator/update/position/#up._S_position

Q169) How can I create unique IDs for embedded documents in MongoDB?

So I need to reference particular subdocuments uniquely from items in my collection. For instance:

```
User = {
'name': 'jim',
'documents': [
{'id': 0001, 'title': "My document"},
{'id': 0002, 'title': "My second document!"},
]
}
```

So I need to be able to auto-create IDs for new documents, preferably not at the application level (since there will be race conditions in the actual development scenario).

Is there a way to use mongo's autogenerated ObjectId (used in the _id field at the collection level), or something similar?

Yes, using mongo's ObjectId is the way to go. The only thing is: you have to generate them yourself, in the application code. They are meant to be globally unique, different workers won't generate two identical ObjectIds, so there's no race condition in that sense.

All official drivers should provide a way to generate ObjectId. Here's how it is in Ruby:

```
oid = BSON::ObjectId.new
```

All drivers have functionality for generating ObjectIds.

In the shell you just do new ObjectId():

```
> db.test.insert({x:new ObjectId()});
> db.test.find();
{ "_id" : ObjectId("4f88592a06c05e4de90d0bc1"), "x" : ObjectId("4f88592a06c05e4de90d0bc0") }
```

In Java it's new ObjectId() as well. See the API docs for your driver to see the specific syntax.

Q170) Deleting a key/value from existing MongoDB entry

I need to delete a certain key and value from every entry in a particular collection. I've looked into remove and that seems to be for only entire entries. Looking at update, I don't believe updating a particular key with null or an empty string would achieve what I'm trying to do. I'm very much a beginner

with mongodb, so please excuse my ignorance.

Long story short, how can I turn

```
{
  "_id" : 1234,
  "name" : "Chris",
  "description" : "Awesome"
}
```

into

```
{
  "_id" : 1234,
  "name" : "Chris"
}
```

without deleting the entry and creating a new one, or using any non-mongodb commands? Thanks!

Try \$unset in a call to update().

In other words,

```
db.collection_name.update({ _id: 1234 }, { $unset : { description : 1 } });
```

should work. Link.

Q171) Problems with Mongo value-in-array query: \$not operator doesn't work, and ObjectIds cannot be selected

I am trying to select a document which does NOT contain a value in a document's array.

I'm having two problems, which I'll present separately:

(1) I cannot get the \$not operator to work with a value-in-array query: For example, if I have the following document in my collection:

```
{ _id: ObjectId("00000000000000000000000000000000"), mylist: [ "red", "green", "blue" ] }
```

I can select this document using:

```
db.myCol.find({mylist:"red"})
```

However, I would like to select this document by testing for the absence of orange:

```
db.myCol.find({$not:{mylist:"orange"}})
```

Q172) Why does this not work?

(2) I cannot get the value in array query to work if the array values are ObjectIds:

```
{ _id: Object("00000000000000000000000000000000"), mylist: [ ObjectId("11111111111111111111111111111111") ] }
```

The following does NOT retrieve this document:

```
myCol.find({mylist:ObjectId("11111111111111111111111111111111")})
```

Can someone suggest what I might be doing wrong?

There is no \$not that works like, you want \$ne for simple cases like yours:

```
db.myCol.find({ mylist: { $ne: 'orange' } })
```

Your second should (and does) work fine for me, perhaps you're using the wrong number of ones. ^

Q173) Updating a document field in mongo based on another field's value

📞 Call us on

✉ Drop us a Query

Suppose I have two fields F1 and F2. I want to update F1 to become F1 + “, ” + F2. Can I do so with a single update command in mongo?

No, you can't do that. You can't use expressions in mongodb updates. The only way is to fetch this document to the client, compose new field value there and issue a prepared update statement.

Q174) mongodb map reduce to get uniques by date/day

trying to group by date and number

my document

```
{ "_id" : ObjectId("4f956dee76ddb26752026e8f"), "request" : "default", "options" : "1", "keyword" : "somekey", "number" : "5b234b79-4d70-437e-8eef-32a2941af40a", "date" : "20120423200446", "time" : 1335193066 }
```

my query

```
map = "function() { var date = new Date(this.time * 1000); var key = date.getFullYear() + date.getMonth() + date.getDate(); emit({day : key, number: this.number}, {count: 1}); }"
```

```
reduce = "function(key, values) { var count = 0; values.forEach(function(v) { count += v['count']; }); return {count: count}; }"
```

```
db.txtweb.mapReduce(map, reduce, {out: "pageview_results"});
```

my error

```
uncaught exception: map reduce failed:{ "errmsg" : "ns doesn't exist", "ok" : 0 }
```

I cannot figure out whats wrong, but I think it is do something with the date functionality.

Any ideas.

ns doesn't exist means that you're accessing a non-existent database or collection. Check their names

Script below group this.amount by day (without time)

Skip code block

```
db.payments.mapReduce(
function() {
var k = clone(this.payment_time);
k.setMinutes(0);
k.setSeconds(0);
k.setMilliseconds(0);
emit(k, this.amount);
},
function (key, vals) {
var total = 0;
for (var i = 0; i < vals.length; i++) {
total += vals[i];
}
return total;
},
```

{
📞 Call us on



✉ Drop us a Query

```

out : {merge : "resultName" }
}
);
}
};

```

Q175) mongo - how to query a nested json

I am a complete mongo newbie. I am using mongo hub for mac. I need to query the for following json -

```

{ "_id" : ObjectId( "abcd" ),
  "className" : "com.myUser",
  "reg" : 12345,
  "test" : [
    { "className" : "com.abc",
      "testid" : "pqrs" } ] }

```

and find records where testid is pqrs. How would I go about doing that?

You can type {'test.testid': 'pqrs'} in the query field of Mongo Hub.

Q176) Mongoddb find a document with all subdocuments satisfying a condition

I have Game collection in my DB:

```

var game = {
  players: [{username:"user1?", status:"played"},
    {username:"user2?", status:"accepted"}]
}

```

As far as I understand query like this: `db.games.find({"players.status":"played"})` will give me all games where at least one player has status "played". How can I find games with ALL players having status "played"?

If you only have one other status than "played" use the query:

```
db.games.find({"players.status":{"$ne":"accepted"}})
```

You can adjust the query to handle more status values, as long as they are all known at the time of the query.

Q177) MongoDB - Update an object in nested Array

Assume we have the following collection, which I have few questions about:

Skip code block

```

{
  "_id" : ObjectId("4faaba123412d654fe83hg876"),
  "user_id" : 123456,
  "total" : 100,
  "items" : [
    {
      "item_name" : "my_item_one",
      "item_id" : 1
    }
  ]
}

```

Call Us on

✉ Drop us a Query


```

    "price" : 20
  },
  {
    "item_name" : "my_item_two",
    "price" : 50
  },
  {
    "item_name" : "my_item_three",
    "price" : 30
  }
]
}

```

1 - I want to increase the price for "item_name":"my_item_two" **and if it doesn't exists**, it should be appended to the "items" array.

2 - How can I update two fields at the same time. For example, increase the price for "my_item_three" and at the same time increase the "total" (with the same value).

I prefer to do this on the MongoDB side, otherwise I have to load the document in client-side (Python) and construct the updated document and replace it with the existing one in MongoDB.

UPDATE This is what I have tried and works fine **IF THE Object Exists** :

```

db.test_invoice.update({user_id : 123456 , "items.item_name":"my_item_one"} , {$inc: {"items.$.price":
10}})

```

But if the key doesn't exist it does nothing. Also it only updates the nested object. There is no way with this command to update the "total" field as well.

For question #1, let's break it into two parts. First, increment any document that has "items.item_name" equal to "my_item_two". For this you'll have to use the positional "\$" operator. Something like:

```

db.bar.update( {user_id : 123456 , "items.item_name" : "my_item_two" } ,
{$inc : {"items.$.price" : 1} } ,
false ,
true);

```

Note that this will only increment the first matched subdocument in any array (so if you have another document in the array with "item_name" equal to "my_item_two", it won't get incremented). But this might be what you want.

The second part is trickier. We can push a new item to an array without a "my_item_two" as follows:

```

db.bar.update( {user_id : 123456, "items.item_name" : {$ne : "my_item_two" } } ,
{$addToSet : {"items" : {'item_name' : "my_item_two" , 'price' : 1 } } } ,
false ,
true);

```

For your question #2, the answer is easier. To increment the total and the price of item_three in any document that contains "my_item_three," you can use the \$inc operator on multiple fields at the same time. Something like:

Call us on

Drop us a Query

```
db.bar.update( { "items.item_name" : { $ne : "my_item_three" } } ,
{ $inc : { total : 1 , "items.$.price" : 1 } ,
false ,
true );
```

Q178) To what extent are 'lost data' criticisms still valid of MongoDB?

I'm referring to the following

<https://pastebin.com/raw.php?i=FD3xe6Jt>

Which, if still valid would be worrying to some extent. The article primarily references v1.6 and v1.8 and since then v2 has been released. Are the shortcomings discussed in the article still outstanding as of the current release?

That particular post was debunked, point by point by the MongoDB CTO and co-founder, Eliot Horowitz, here:

<https://news.ycombinator.com/item?id=3202959>

There is also a good summary here:

<https://www.betabeat.com/2011/11/10/the-trolls-come-out-for-10gen/>

The short version is, it looks like this was basically someone trolling for attention (successfully), with no solid evidence or corroboration. There have been genuine incidents in the past, which have been dealt with as the product evolved (see the introduction of journaling in 1.8 for example) or as more specific bugs were found and fixed.

Disclaimer: I do work for MongoDB (formerly 10gen), and love the fact that philnate got here and refuted this independently first – that probably says more about the product than anything else :)

Update: August 19th 2013

I've seen quite a bit of activity on this answer recently, which I assume is related to the announcement of the bug in SERVER-10478 – it is most certainly an edge case, but I would still recommend anyone using sharding with large documents to upgrade ASAP to v2.2.6 and v2.4.6 which include the fix for this issue. Never heard of those severe problems in recent versions. What you need to consider is that MongoDB has no decade of development as relational Systems in the back. Further it may be true that MongoDB doesn't offer that much functionality to avoid data loss at all. But even with relational Systems you won't be ever sure that you'll never loose any data. It highly depends on your system configuration (so with Replication and manual data backups you should be quite safe).

As a general guideline to avoid Beta Bugs or bugs from early versions, avoid to use fresh versions in productions (there's a reason why debian is so popular for servers). If MongoDB would suffer such severe problems (all the time) the list of users would be smaller:<https://www.mongodb.org/display/DOCS/Production+Deployments> Additionally I don't really trust this pastebin message, why is this published anonymously? Is this person company shamed to tell that they used mongodb, do they fear 10gen? Where a links to those Bug reports (or did 10gen delete them from JIRA?)

So lets talk shortly about those points:

1. Yep MongoDB operates normally in fire and forget mode. But you can modify this behavior with
 Call us on  Drop us a Query

several options: <https://www.mongodb.org/display/DOCS/getLastError+Command>. So only because MongoDB defaults to it, it doesn't mean you can't change it to your needs. But you need to live less performance if you don't fire and forget within your app, as you're adding a roundtrip.

2. Never heard of such problems, except those caused to own failure...but that can happen with relational systems as well. I guess this point only talks about Master-Slave Replication. Replica-Sets are much more stable. Some links from the web where other dbms caused data loss due to malfunction as well: <https://blog.lastinfirstout.net/2010/04/bit-by-bug-data-loss-running-oracle-on.html>

<https://dbaspot.com/oracle-server/430465-parallel-cause-data-lost-another-oracle-bug.html>

<https://bugs.mysql.com/bug.php?id=18014> (Those posted links aren't in any favor of a given system or should imply anything else than showing that there are bugs in other systems as well, who can cause data loss.)

3. Yes actually there's Locking at instance level, I don't think that in sharded environment this is a global one, I think this will be at instance level for each shard separate, as there's no need to lock other instances as there are no consistency checks needed. The upcoming Version 2.2 will lock at DB Level, tickets for Collection Level and maybe extend or document exists as well (<https://jira.mongodb.org/browse/SERVER-4328>). But locking at deeper levels may affect the actual performance of MongoDB, as a lock management is expensive.

4. Moving chunks shouldn't cause much problems as rebalancing should take a few chunks from each node and move them to the new one. It never should cause ping/pong of chunks nor does rebalancing start just because of a difference of one or two chunks. What can be problematic is when your shard key is chosen wrong. So you may end up with all new entries inserted to one node rather than all. So you would see more often rebalancing which can cause problems, but that would be not due to mongo rather than your poorly chosen shardkey.

5. Can't comment on this one

6. Not 100% sure, but I think Replicasets were introduced in 1.6, so as told earlier never use the latest version for production, except you can live with loss of data. As with every new feature there's the possibility of bugs. Even extensive testing may not reveal all problems. Again always run some manual backup for god's sake, except you can live with data loss.

7. Can't comment on this. But in reality software may contain severe bugs. Many games suffer those problems as well and there are other areas as well where banana software was quite well known or is. Can't Comment about something concrete as this was before my MongoDB time.

8. Replication can cause such problems. Depending on the replication strategy this may be a problem and another system may fit better. But without a really really write intensive workload you may not encounter such problems. Indeed it may be problematic to have 3 replicas polling changes from one master. You could cure the problem by adding more shards.

As a general conclusion: Yeah it may be that those problems were existent, but MongoDB did much in this direction and further I doubt that other DBMS never had the one or other problem itself. Just take traditional relational dbms, would those scale well to web-scale there would be no need for Systems like MongoDB, HBase and what else. You can't get a system which fits all needs. So you have to live with the downsides of one or try to build a combined system of multiple to get what you need.

Call us on

Drop us a Query

Disclaimer: I'm not affiliated with MongoDB or 10gen, I'm just working with MongoDB and telling my opinion about it.

I can't speak for every case, only my own. However I don't work for 10gen or its competitors, and I have lost data when using MongoDB, so here goes.

The main criticisms of Mongo around the last time I used it (2010) were:

1. Supposedly stable versions of Mongo had major data-losing bugs that weren't made explicit to users. Eg, prior to 1.8 non-clustered configurations were likely to lose data. This was documented by Mongo, but not to the extent a data losing bug in a stable-versioned DB would normally be.

The main defence of that criticism was:

2. Users were informed of this danger, albeit not so explicitly. Users should read all the documentation before they begin.

In my own case, I was using 1.7 in a single server configuration but aware of the risk. I shut down the DB to take a back up. The act of shutting down the DB itself lost my data, 10gen assisted (for free) but were unable to recover the data.

Q179) Mongo DB Delete a field and value

In mongo shell how would I delete all occurrences of "id" : "1" the value of the field is always different. Would I use the \$unset operator? Would that delete the value and the field?

You're saying remove all occurrences of the field, right? If so, then it should be like this:

```
db.collection.update(
{ id: { $exists: true } }, // criteria
{ $unset: { id: 1 } },    // modifier
false,                  // no need to upsert
true                     // multi-update
);
```

Q180) Checking if a field contains a string

I'm looking for an operator, which allows me to check, if the value of a field contains a certain string. Something like:

```
db.users.findOne({$contains:{"username":"son"}})
```

Is that possible?

You can do it with the following code.

```
db.users.findOne({"username" : {$regex : ".*son.*"}});
```

As Mongo shell support regex, that's completely possible.

```
db.users.findOne({"username" : /.son./});
```

If we want the query to be case-insensitive, we can use "i" option, like shown below:

```
db.users.findOne({"username" : /.son./i});
```

See: <https://www.mongodb.org/display/DOCS/Advanced+Queries#AdvancedQueries-RegularExpressions>

<https://www.mongodb.org/display/DOCS/SQL+to+Mongo+Mapping+Chart>

<https://www.mongodb.org/manual/en/mongo.sqltomongo.php>

✉ Drop us a Query

MySQL

```
SELECT * FROM users WHERE username LIKE "%Son%"
```

MongoDB

```
db.users.find({username:/Son/})
```

Q181) What's the fastest way to copy a collection within the same database?

I want to copy a collection within the **same** database and give it a different name – basically take a snapshot.

What's the best way to do this? Is there a command, or do I have to copy each record in turn?

I'm aware of the cloneCollection command, but it seems to be for copying to another server only.

I'm also aware of mongoimport and mongoexport, but as I'm doing this via PHP I'd prefer not to make calls out to the shell.

You have a few options, but the fastest is:

```
mongoexport -d db -c sourcecollection | mongoimport -d db -c targetcollection -drop
```

or in php:

```
`mongoexport -d db -c sourcecollection | mongoimport -d db -c targetcollection -drop`;
```

after that you have

```
mongo db < script.js
```

where, as shown in the mongo docs, script.js contains something like:

```
db.myoriginal.find().forEach( function(x){db.mycopy.insert(x)} );
```

The slowest (by an order of magnitude or more) way to copy a collection will be to use the native php driver – simply because of moving information around. But you could issue the above mongo query if you absolutely want to avoid cli calls using the db execute function.

Q182) mongodb: upserting: only set value if document is being inserted

Considering a simple mongo document structure:

```
{ _id, firstTime, lastTime }
```

The client needs to insert a document with a known ID, or update an existing document. The 'lastTime' should always be set to some latest time. For the 'firstTime', if a document is being inserted, then the 'firstTime' should be set to current time. However, if the document is already created, then 'firstTime' remain unchanged. I would like to do it purely with upserts (to avoid look ups).

I've crawled the <https://www.mongodb.org/display/DOCS/Updating>, but I just don't see how that particular operation can be done.

I don't believe this is something unreasonable, there are \$push and \$addToSet operations that effectively do that on array fields, just nothing that would do the same on simple fields. It's like there should be something like \$setIf operation.

I ran into the exact same problem and there was no simple solution for <2.4 however since 2.4 the **\$setOnInsert** operator let's you do exactly that.

```
db.collection.update( ,
```

```
{ $setOnInsert: { "firstTime": } },
```

📞 Call us on



✉ Drop us a Query

```
{ upsert: true }
)
```

See the 2.4 release notes of setOnInsert for more info.

Q183) Change collection name in mongodb

Changing the name of a collection in mongodb can be achieved by copy the documents in the collection to a new one and delete the original.

But is there a simpler way to change the name of a collection in mongodb?

```
db.oldname.renameCollection("newname")
```

Really? No google search?

<https://www.mongodb.org/display/DOCS/renameCollection+Command>

```
> db.oldname.renameCollection("newname")
```

Q184) what's the diff between findAndModify and update in MongoDB?

I'm a little bit confused by the findAndModify method in MongoDB. What's the advantage of it over update method? For me, it seems that it just returns the item first and then update it. But Why do I need to return the item first? I read the **MongoDB: the definitive guide** and it says that It is handy for manipulating queues and performing other operations that need get-and-set style atomicity. But I didn't understand how it achieves this. Can somebody explain this to me?

If you fetch an item and then update it, there may be an update by another thread between those two steps. If you update an item first and then fetch it, there may be another update in-between and you will get back a different item than what you updated.

Doing it "atomically" means you are guaranteed that you are getting back the exact same item you are updating - i.e. no other operation can happen in between.

findAndModify returns the document, update does not.

If I understood Dwight Merriman (one of the original authors of mongoDB) correctly, using update to modify a single document i.e. {"multi":false} is also atomic. Currently, it should also be faster than doing the equivalent update using findAndModify.

One useful class of use cases is counters and similar cases. For example, take a look at this code (one of the MongoDB tests): find_and_modify4.js.

Thus, with findAndModify you increment the counter and get its incremented value in one step. Compare: if you (A) perform this operation in two steps and somebody else (B) does the same operation between your steps then A and B may get the same last counter value instead of two different (just one example of possible issues).

Q185) Getting an error, "Error: couldn't connect to server 127.0.0.1 shell/mongo.js" & when trying to run mongod on mac osx lion

I am using Mac OS X Lion and i just did a fresh install of MongoDB using macports and when i try to run mongod for the first time i get the following error

MongoDB shell version: 2.0.5

connection to: test

✉ Drop us a Query

Fri Jun 1 11:20:33 Error: couldn't connect to server 127.0.0.1 shell/mongo.js:84

exception: connect failed

can anybody please help with this? thanks

when i run mongod i get:

Skip code block

hisham-agil:~ hisham\$ mongod

mongod -help for help and startup options

Fri Jun 1 11:24:47 [initandlisten] MongoDB starting : pid=53452 port=27017 dbpath=/data/db/ 64-bit
host=hisham-agil.local

Fri Jun 1 11:24:47 [initandlisten] db version v2.0.5, pdfile version 4.5

Fri Jun 1 11:24:47 [initandlisten] git version: nogitversion

Fri Jun 1 11:24:47 [initandlisten] build info: Darwin gamma.local 11.3.0 Darwin Kernel Version 11.3.0: Thu
Jan 12 18:48:32 PST 2012; root:xnu-1699.24.23~1/RELEASE_I386 i386 BOOST_LIB_VERSION=1_49

Fri Jun 1 11:24:47 [initandlisten] options: {}

Fri Jun 1 11:24:47 [initandlisten] exception in initAndListen: 10296 dbpath (/data/db/) doesnot exist,
terminating

Fri Jun 1 11:24:47 dbexit:

Fri Jun 1 11:24:47 [initandlisten] shutdown: going to close listening sockets...

Fri Jun 1 11:24:47 [initandlisten] shutdown: going to flush diaglog...

Fri Jun 1 11:24:47 [initandlisten] shutdown: going to close sockets...

Fri Jun 1 11:24:47 [initandlisten] shutdown: waiting for fs preallocator...

Fri Jun 1 11:24:47 [initandlisten] shutdown: lock for final commit...

Fri Jun 1 11:24:47 [initandlisten] shutdown: final commit...

Fri Jun 1 11:24:47 [initandlisten] shutdown: closing all files...

Fri Jun 1 11:24:47 [initandlisten] closeAllFiles() finished

Fri Jun 1 11:24:47 dbexit: really exiting now

You're running the mongo client without starting the server first. Try running mongod first.

You'll have to specify where the mongo "workspace" is using the -dbpath switch, such as mongod -
dbpath /mongo/db. The specified folder should exist.

By default mongod will try to use the directory /data/db for the database files.

In your case that directory does not exist.

Before starting mongod you should create those directories and make sure they are writable by the
same user that is running mongod process.

Q186) MongoDB - how to query for a nested item inside a collection?

I have some data that looks like this:

Skip code block

[

{

"_id" : ObjectId("4e2f2af16f1e7e4c2000000a"),

Call us on



Drop us a Query

```

“advertisers” : [
{
“created_at” : ISODate(“2011-07-26T21:02:19Z”),
“category” : “Infinity Pro Spin Air Brush”,
“updated_at” : ISODate(“2011-07-26T21:02:19Z”),
“lowered_name” : “conair”,
“twitter_name” : “”,
“facebook_page_url” : “”,
“website_url” : “”,
“user_ids” : [ ],
“blog_url” : “”,
},

```

and I was thinking that a query like this would give the id of the advertiser:

```

var start = new Date(2011, 1, 1);
> var end = new Date(2011, 12, 12);
> db.agencies.find( { “created_at” : { $gte : start , $lt : end } } , { _id : 1 , program_ids: 1 , advertisers {
name : 1 } } ).limit(1).toArray();

```

But my query didn't work. Any idea how I can add the fields inside the nested elements to my list of fields I want to get?

Thanks!

Use dot notation (e.g. advertisers.name) to query and retrieve fields from nested objects:

```

db.agencies.find( { “advertisers.created_at” : { $gte : start , $lt : end } } , { _id : 1 , program_ids: 1 ,
“advertisers.name”: 1 } } ).limit(1).toArray();

```

Reference: Retrieving a Subset of Fields and Dot Notation

Q187) How do I abort a query in mongo javascript shell

Executing queries in the Javascript shell of Mongo feels a lot like executing commands right in the bash shell. Because of this, my fingers keep trying to Ctrl+C my way out of a command that I want to “abort” and land back at the command prompt. Unfortunately, in the mongo shell at 2.0.4, when I Ctrl+C, it drops me out of the javascript shell, rather than just aborting the command and giving me the command prompt.

Is there a way in the javascript shell to abandon the current command without executing it and land back at the command prompt?

What I've always naturally done is type a (and then immediately hit enter three times. The shell will assume you are just hitting enter to break up your command, but if you hit enter 2 more times it will exit back to the shell. It's wonky, but it works.

Q188) Where does mongodb stand in the CAP theorem?

Everywhere I look, I see that MongoDB is CP. But when I dig in I see it is eventually consistent. Is it ~~CP~~ when you use safe=true? If so, does that mean that when I write with safe=true, all replicas will be updated before getting the result?

 Drop us a Query

MongoDB is strongly consistent by default – if you do a write and then do a read, assuming the write was successful you will always be able to read the result of the write you just read. This is because MongoDB is a single-master system and all reads go to the primary by default. If you optionally enable reading from the secondaries then MongoDB becomes eventually consistent where it's possible to read out-of-date results.

MongoDB also gets high-availability through automatic failover in replica sets:
<https://www.mongodb.org/display/DOCS/Replica+Sets>

This should help answer the question, along with other NoSQL and other persistent storage systems.

Q189) MongoDB: Billions of documents in a collection

I need to load 6.6 billion bigrams into a collection but I can't find any information on the best way to do this.

Loading that many documents onto a single primary key index would take forever but as far as I'm aware mongo doesn't support the equivalent of partitioning?

Would sharding help? Should I try and split the data set over many collections and build that logic into my application?

It's hard to say what the optimal bulk insert is — this partly depends on the size of the objects you're inserting and other immeasurable factors. You could try a few ranges and see what gives you the best performance. As an alternative, some people like using mongoimport, which is pretty fast, but your import data needs to be json or csv. There's obviously mongorestore, if the data is in BSON format.

Mongo can easily handle billions of documents and can have billions of documents in the one collection but remember that the maximum document size is 16mb. There are many folk with billions of documents in MongoDB and there's lots of discussions about it on the MongoDB Google User Group. Here's a document on using a large number of collections that you may like to read, if you change your mind and want to have multiple collections instead. The more collections you have, the more indexes you will have also, which probably isn't what you want.

Here's a presentation from Craigslist on inserting billions of documents into MongoDB and the guy's blogpost.

It does look like sharding would be a good solution for you but typically sharding is used for scaling across multiple servers and a lot of folk do it because they want to scale their writes or they are unable to keep their working set (data and indexes) in RAM. It is perfectly reasonable to start off with a single server and then move to a shard or replica-set as your data grows or you need extra redundancy and resilience.

However, there are other users use multiple mongods to get around locking limits of a single mongod with lots of writes. It's obvious but still worth saying but a multi-mongod setup is more complex to manage than a single server. If your IO or cpu isn't maxed out here, your working set is smaller than RAM and your data is easy to keep balanced (pretty randomly distributed), you should see improvement (with sharding on a single server). As a FYI, there is potential for memory and IO contention. With 2.2 having improved concurrency with db locking, I suspect that there will be much less of a reason for such a deployment.

📞 Call us on

✉ Drop us a Query

You need to plan your move to sharding properly, i.e. think carefully about choosing your shard key. If you go this way then it's best to pre-split and turn off the balancer. It will be counter-productive to be moving data around to keep things balanced which means you will need to decide up front how to split it. Additionally, it is sometimes important to design your documents with the idea that some field will be useful for sharding on, or as a primary key.

Here's some good links –

1. Choosing a Shard Key
2. Blog post on shard keys
3. Overview presentation on sharding
4. Presentation on Sharding Best Practices

You can absolutely shard data in MongoDB (which partitions across N servers on the shard key). In fact, that's one of it's core strengths. There is no need to do that in your application.

For most use cases, I would strongly recommend doing that for 6.6 billion documents. In my experience, MongoDB performs better with a number of mid-range servers rather than one large one.

Q190) Why are key names stored in the document in MongoDB

I'm curious about this quote from Kyle Banker's MongoDB In Action:

Its important to consider the length of the key names you choose, since key names are stored in the documents themselves. This contrasts with an RDBMS, where column names are always kept separate from the rows they refer to. So when using BSON, if you can live with dob in place of date_of_birth as a key name, you'll save 10 bytes per document. That may not sound like much, but once you have a billion such documents, you'll have saved nearly 10 GB of storage space just by using a shorter key name. This doesn't mean you should go to unreasonable lengths to ensure small key names; be sensible. But if you expect massive amounts of data, economizing on key names will save space.

I am interested in the reason why this is not optimized on the database server side. Would a in-memory lookup table with all key names in the collection be too much of a performance penalty that is not worth the potential space savings?

What you are referring to is often called “key compression”*. There are several reasons why it hasn't been implemented:

1. If you want it done, you can currently do it at the Application/ORM/ODM level quite easily.
2. It's not necessarily a performance** advantage in all cases think collections with lots of key names, and/or key names that vary wildly between documents.
3. It might not provide a measurable performance** advantage at all until you have millions of documents.
4. If the server does it, the full key names still have to be transmitted over the network.
5. If compressed key names are transmitted over the network, then readability really suffers using the javascript console.
6. Compressing the entire JSON document might offer an even better performance advantage.

Like all features, there's a cost benefit analysis for implementing it, and (at least so far) other features have offered more “bang for the buck”.

📞 Call us on

✉ Drop us a Query

Full document compression is being considered for a future MongoDB version.

* An in-memory lookup table for key names is basically a special case of LZW style compression that's more or less what most compression algorithms do.

** Compression provides both a space advantage and a performance advantage. Smaller documents means that more documents can be read per IO, which means that in a system with fixed IO, more documents per second can be read.

I believe one of the original reasons behind storing the key names with the documents is to allow a more easily scalable schema-less database. Each document is self-contained to a greater extent, in that if you move the document to another server (for example, via replication or sharding) you can index the contents of the document without having to reference separate or centralized metadata such as a mapping of key names to more compact key IDs.

Since there is no enforced schema for a MongoDB collection, the field names can potentially be different for every document in the same collection. In a sharded environment, inserts to each shard are (intentionally) independent so at a document level the raw data could end up differing unless the key mapping was able to be consistent per shard.

Depending on your use case, the key names may or may not consume a significant amount of space relative to the accompanying data. You could always workaround the storage concern from the application / ODM implementation by mapping YourFriendlyKeyNames to shorter DB key equivalents.

There is an open MongoDB Jira issue and some further discussion to have the server tokenize field names, which you can vote on to help prioritize including this feature in a future release.

MongoDB's current design goals include performance with dynamic schemas, replication & high availability, auto-sharding, and in-place updates .. with one potential tradeoff being some extra disk usage.

Q191) How can I copy collection to another database in MongoDB?

Is there some simple way to fulfill this?

At the moment there is no command in MongoDB that would do this. Please note the JIRA ticket with related feature request: <https://jira.mongodb.org/browse/SERVER-732>.

You could do something like:

```
db..find().forEach(function(d){ db.getSiblingDB("")[].insert(d); });
```

Please note that with this, the two databases would need to share the same mongod for this to work.

Besides this, you can do a mongodump of a collection from one database and then mongorestore the collection to the other database.

The best way is to do a mongodump then mongorestore.

You can select the collection via:

```
mongodump -d some_database -c some_collection
```

[Optionally, zip the dump (zip some_database.zip some_database/* -r) and scp it elsewhere]

Then restore it:

```
mongorestore -d some_other_db -c some_or_other_collection dump/some_collection.bson
```

Existing data in some_or_other_collection will be preserved. That way you can "append" a collection

☎ Call us on

✉ Drop us a Query



from one database to another.

Prior to version 2.4.3, you will also need to add back your indexes after you copy over your data. Starting with 2.4.3, this process is automatic, and you can disable it with `-noIndexRestore`.

I would abuse the connect function in mongo cli mongo doc. so that means you can start one or more connection. if you want to copy customer collection from test to test2 in same server. first you start mongo shell

use test

```
var db2 = connect('localhost:27017/test2')
```

do a normal find and copy the first 20 record to test2.

```
db.customer.find().limit(20).forEach(function(p) { db2.customer.insert(p); });
```

or filter by some criteria

```
db.customer.find({"active": 1}).forEach(function(p) { db2.customer.insert(p); });
```

just change the localhost to IP or hostname to connect to remote server. I use this to copy test data to a test database for testing.

Actually, there is a command to move a collection from one database to another. It's just not called "move" or "copy".

To copy a collection, you can clone it on the same db, then move the clone.

To clone:

```
> use db1
```

```
> db.source_collection.find().forEach( function(x){db.collection_copy.insert(x)} );
```

To move:

```
> use admin
```

switched to db admin

```
> db.runCommand({renameCollection: 'db1.source_collection, to: 'db2.target_collection'}) // who'd think rename could move?
```

The other answers are better for copying the collection, but this is especially useful if you're looking to move it.

I'd usually do:

```
use sourcedatabase;
```

```
var docs=db.sourcetable.find();
```

```
use targetdatabase;
```

```
docs.forEach(function(doc) { db.targettable.insert(doc); });
```


Q192) In Mongo what is the difference between sharding and replication?

Replication seems to be a lot simpler than sharding, unless I am missing the benefits of what sharding is actually trying to achieve. Don't they both provide horizontal scaling?

In the context of scaling MongoDB:

replication creates additional copies of the data and allows for automatic failover to another node. 1.

Replication may help with horizontal scaling of reads if you are OK to read data that potentially isn't the latest.

 Call us on

 Drop us a Query

2. sharding allows for horizontal scaling of data writes by partitioning data across multiple servers using a shard key. It's important to choose a good shard key. For example, a poor choice of shard key could lead to "hot spots" of data only being written on a single shard.

A sharded environment does add more complexity because MongoDB now has to manage distributing data and requests between shards — additional configuration and routing processes are added to manage those aspects.

Replication and sharding are typically combined to create a sharded cluster where each shard is supported by a replica set.

From a client application point of view you also have some control in relation to the replication/sharding interaction, in particular:

1. Read preferences
2. Write concerns

Replication is a mostly traditional master/slave setup, data is synced to backup members and if the primary fails one of them can take its place. It is a reasonably simple tool. It's primarily meant for redundancy, although you can scale reads by adding replica set members. That's a little complicated, but works very well for some apps.

Sharding sits on top of replication, usually. "Shards" in MongoDB are just replica sets with something called a "router" in front of them. Your application will connect to the router, issue queries, and it will decide which replica set (shard) to forward things on to. It's significantly more complex than a single replica set because you have the router and config servers to deal with (these keep track of what data is stored where).

If you want to scale Mongo horizontally, you'd shard. 10gen likes to call the router/config server setup auto-sharding. It's possible to do a more ghetto form of sharding where you have the app decide which DB to write to as well.

Q193) What does Mongo sort on when no sort order is specified?

When we run a Mongo find() query without any sort order specified, what does the database internally use to sort the results?

According to the documentation on the mongo website:

When executing a find() with no parameters, the database returns objects in forward natural order.

For standard tables, natural order is not particularly useful because, although the order is often close to insertion order, it is not guaranteed to be. However, for Capped Collections, natural order is guaranteed to be the insertion order. This can be very useful.

However for standard collections (non capped collections), what field is used to sort the results? Is it the `_id` field or something else?

Edit:

Basically, I guess what I am trying to get at is that if I execute the following search query:

```
db.collection.find({"x":y}).skip(10000).limit(1000);
```

At two different points in time: **t1** and **t2**, will I get different result sets:

1. When there have been no additional writes between t1 & t2?

📞 Call us on

✉ Drop us a Query

2. When there have been new writes between t1 & t2?

3. There are new indexes that have been added between t1 & t2?

I have run some tests on a temp database and the results I have gotten are the same (**Yes**) for all the 3 cases – but I wanted to be sure and I am certain that my test cases weren't very thorough.

By definition the sort **defaults to undefined**, and so is the return order of documents. If there is no query then it will use the natural order. The results are returned in the **order they are found**, which may coincide with insertion order (but isn't guaranteed to be) or the order of the index(es) used.

Some examples that will affect storage (natural) order:

1. if documents are updated and don't fit in their currently allocated space, they will be moved

new documents may be inserted in available gaps created by deleted or moved documents

2. If an index is used, docs will be returned in the order they are found. If more than one index is used then the order depends internally on which index first identified the document during the de-duplication process.

If you **want a specific order** then you **must** include a sort with your query.

The exception noted for capped collections' natural order is because documents can't move and are stored in insertion order. The ordering is part of the capped collection feature that ensures the oldest documents "age out" first. Additionally, documents cannot be deleted or moved in a capped collection (see Usage and Restrictions for more info).

Q194) How to reclaiming deleted space without `db.repairDatabase()`?

I want to shrink data files size by reclaiming deleted space, but I can't run db.repairDatabase(), because free disk space is not enough.

The original answer to this question is here: Reducing MongoDB database file size

There really is nothing outside of repair that will reclaim space. The compact should allow you to go much longer on the existing space. Otherwise, you will have to migrate to a bigger drive.

One way to do this is to use an off-line secondary from your Replica Set. This should give you a whole maintenance window to migrate, repair, move back and bring back up.

If you are not running a Replica Set, it's time to look at doing just that.

You could run the compact command on a single collection, or one by one in all the collections you want to shrink.

<https://www.mongodb.org/display/DOCS/Compact+Command>

```
db.runCommand( { compact : 'mycollectionname' } )
```

As noted in comments, I was mistaken, compact does not actually reclaim disk space, it only defragments and rebuilds collection indexes.

Instead though, you could use "-repairpath" option if you have another drive available which has available freespace.

For example:

```
mongod -dbpath /data/db -repair -repairpath /data/db0
```

Shown here: <https://docs.mongodb.org/manual/tutorial/recover-data-following-unexpected-shutdown/> 

Q195) creating a different database for each collection in MongoDB 2.2

 Drop us a Query

MongoDB 2.2 has a write lock per database as opposed to a global write lock on the server in previous versions. So would it be ok if i store each collection in a separate database to effectively have a write lock per collection.(This will make it look like MyISAM's table level locking). Is this approach faulty? There's a key limitation to the locking and that is the local database. That database includes a theoplog collection which is used for replication.

If you're running in production, you should be running with Replica Sets. If you're running with Replica Sets, you need to be aware of the write lock effect on that database.

Breaking out your 10 collections into 10 DBs is useless if they all block waiting for the oplog.

Before taking a large step to re-write, please ensure that the oplog will not cause issues.

Also, be aware that MongoDB implements DB-level security. If you're using any security features, you are now creating more DBs to secure.

Q196) How to stop mongo DB in one command

I need to be able to start/stop MongoDB on the cli. It is quite simple to start:

```
./mongod
```

But to stop mongo DB, I need to run open mongo shell first and then type two commands:

```
$ ./mongo
```

```
use admin
```

```
db.shutdownServer()
```

So I don't know how to stop mongo DB in one line. Any help?

Starting and Stopping MongoDB is covered in the MongoDB manual. This section of the manual explains the various options of stopping MongoDB through the shell, cli, drivers etc. It also details the risks of incorrectly stopping MongoDB (such as data corruption) and talks about the different kill signals.

Additionally, if you have installed MongoDB using a package manager for Ubuntu or Debian then you can stop mongod as follows:

1. Upstart:sudo service mongod stop

2. Sysvinit:sudo /etc/init.d/mongod stop

Or on Red Hat based systems:

1. service mongod stop

Or on Windows if you have installed as a service named MongoDB:

1. net stop MongoDB

To learn more about the problems of an unclean shutdown, how to best avoid such a scenario and what to do in the event of an unclean shutdown, please see: Recover Data after an Unexpected Shutdown.

If you literally want a one line equivalent to the commands in your original question, you could alias:

```
mongo -eval "db.getSiblingDB('admin').shutdownServer()"
```

Mark's answer on starting and stopping MongoDB via services is the more typical (and recommended) administrative approach.

```
mongod -dbpath /path/to/your/db -shutdown
```

More info at official: <https://docs.mongodb.org/manual/tutorial/manage-mongod-processes/>



Q197) MongoDB select count(distinct x) on an indexed column – count unique results for large data sets

I have gone through several articles and examples, and have yet to find an efficient way to do this SQL query in MongoDB (where there are millions of documents)

First attempt

(e.g. from this almost duplicate question – Mongo equivalent of SQL's SELECT DISTINCT?)

```
db.myCollection.distinct("myIndexedNonUniqueField").length
```

Obviously I got this error as my dataset is huge

Thu Aug 02 12:55:24 uncaught exception: distinct failed: {

"errmsg" : "exception: distinct too big, 16mb cap",

"code" : 10044,

"ok" : 0

}

Second attempt

I decided to try and do a group

```
db.myCollection.group({key: {myIndexedNonUniqueField: 1},
```

```
initial: {count: 0},
```

```
reduce: function (obj, prev) { prev.count++; } });
```

But I got this error message instead:

exception: group() can't handle more than 20000 unique keys

Third attempt

I haven't tried yet but there are several suggestions that involve mapReduce

e.g.

1. this one [how to do distinct and group in mongodb?](#) (not accepted, answer author / OP didn't test it)
2. this one [MongoDB group by Functionalities](#) (seems similar to Second Attempt)
3. this one <https://blog.emmettshear.com/post/2010/02/12/Counting-Uniques-With-MongoDB>
4. this one <https://groups.google.com/forum/?fromgroups#!topic/mongodb-user/trDn3jJqtE>
5. this one https://cookbook.mongodb.org/patterns/unique_items_map_reduce/

Also

It seems there is a pull request on GitHub fixing the .distinct method to mention it should only return a count, but it's still open: <https://github.com/mongodb/mongo/pull/34>

But at this point I thought it's worth to ask here, what is the latest on the subject? Should I move to SQL or another NoSQL DB for distinct counts? or is there an efficient way?

Update:

This comment on the MongoDB official docs is not encouraging, is this accurate?

<https://www.mongodb.org/display/DOCS/Aggregation#comment-430445808>

Update2:

Seems the new Aggregation Framework answers the above comment... (MongoDB 2.1/2.2 and above, development preview available, not for production)

<https://docs.mongodb.org/manual/applications/aggregation/>

✉ Drop us a Query

1) The easiest way to do this is via the aggregation framework. This takes two “\$group” commands: the first one groups by distinct values, the second one counts all of the distinct values

Skip code block

```
pipeline = [
  { $group: { _id: "$myIndexedNonUniqueField" } },
  { $group: { _id: 1, count: { $sum: 1 } } }
];
//
// Run the aggregation command
//
R = db.runCommand(
{
  "aggregate": "myCollection",
  "pipeline": pipeline
}
);
printjson(R);
```

2) If you want to do this with Map/Reduce you can. This is also a two-phase process: in the first phase we build a new collection with a list of every distinct value for the key. In the second we do a count() on the new collection.

Skip code block

```
var SOURCE = db.myCollection;
var DEST = db.distinct
DEST.drop();
map = function() {
  emit( this.myIndexedNonUniqueField , {count: 1});
}
reduce = function(key, values) {
  var count = 0;
  values.forEach(function(v) {
    count += v['count'];    // count each distinct value for lagniappe
  });
  return {count: count};
};
//
// run map/reduce
//
res = SOURCE.mapReduce( map, reduce,
{ out: 'distinct',
```

verbose: true
 Call us on



 Drop us a Query

```

}
);
print( "distinct count= " + res.counts.output );
print( "distinct count=", DEST.count() );

```

Note that you cannot return the result of the map/reduce inline, because that will potentially overrun the 16MB document size limit. You can save the calculation in a collection and then count() the size of the collection, or you can get the number of results from the return value of mapReduce().

```

db.myCollection.aggregate(
{$group : {_id : "$myIndexedNonUniqueField"} },
{$group: {_id:1, count: {$sum : 1 }}});

```

straight to result:

```

db.myCollection.aggregate(
{$group : {_id : "$myIndexedNonUniqueField"} },
{$group: {_id:1, count: {$sum : 1 }}})
.result[0].count;

```

Q198) MongoDB: Find Subdocument in Array Matching Parameters

In MongoDB I would like to find a document based on the values of a subdocument meeting certain parameters. Specifically I have a document structured like this:

Skip code block

```

{
  name: "test",
  data: [{
    name: "test1",
    start: 0,
    end: 2
  },
  {
    name: "test2",
    start: 15
    end: 18
  }]
}

```

How can I tell MongoDB to only return my document if the start time for a data subdocument is less than 5 and the end time for the same subdocument is greater than 5? Currently, if I do


```

db.foo.findOne({
  'data.start': { $lte: 5 },
  'data.end': { $gte: 5 }
})

```

it will return my document always because 5 is greater than 0 and less than 18. How can I tell MongoDB

 Call us on

 Drop us a Query



to only return my document if 5 (or whatever value) is greater than 0 and less than 2 OR greater than 15 and less than 18?

You want to use \$elemMatch.

```
db.foo.findOne({ data: { $elemMatch : {
start: { $lte: 5 },
end: { $gte: 5 }
}}
})
```

Q199) How to efficiently perform “distinct” with multiple keys?

For example, there is a collection like this:

```
{market: 'SH', code: '000001', date: '2012-01-01', price: 1000}
{market: 'SZ', code: '000001', date: '2012-01-01', price: 1000}
{market: 'SH', code: '000001', date: '2012-01-02', price: 1000}
{market: 'SZ', code: '000001', date: '2012-01-02', price: 1000}
{market: 'SH', code: '000002', date: '2012-01-03', price: 1000}
...
```

This collection contains tens of millions documents.

I want to call distinct with two keys:

```
collection.distinct('market', 'code');
```

and get result :

```
[{market: 'SH', code:'000001'}, {market: 'SZ', code:'000001'}, {market: 'SH', code:'000002'}]
```

As native distinct command accept only one key, I try to implement it by using map-reduce. But map-reduce is far too slow to native distinct. In my one-key distinct test, map-reduce spend about ten times longer than native distinct.

Is there a efficient way to implement multikey distinct?

If you are willing to wait for the upcoming 2.2 release of MongoDB, you can run this query efficiently using the aggregation framework:

```
collection = db.tb;
result = collection.aggregate(
[
{"$group": { "_id": { market: "$market", code: "$code" } } }
]
);
printjson(result);
```

On a million-record collection on my test machine, this ran in 4 seconds, while the map/reduce version took over a minute.

Q200) MongoDB: Trade-offs of dropping a collection vs. removing all of its documents



What are the trade-offs of dropping a MongoDB collection vs. removing all of its documents (assuming the collection will be re-created immediately)?

If you go through a remove all the documents from a collection, then you'll be doing a lot more work (freeing the document's storage, clearing the index entries that point to the document, and so on). If you instead just drop the collection, it'll just be reclaiming the extents that the collection and its indexes use.

One other difference is that dropping the collection will also remove the collection's indexes.

A benefit of simply dropping a collection is that it is much faster than removing all of a collection's documents. If your collection will be "re-created immediately" anyway (assuming that includes index re-creation), then this is probably the most-attractive option.

The authors of the book MongoDB: The Definitive Guide (Kristina Chodorow and Michael Dirolf) ran an experiment where they provided a Python script which timed a drop vs. a remove of 1000000 records. The results came in at 46.08 seconds for the remove and .01 seconds for the drop. Now while the exact times may differ based-on hardware and other factors, it nonetheless illustrates the point that the drop is significantly faster.

reference: Chodorow K., Dirolf M. (2010). MongoDB: The Definitive Guide. O'Reilly Media, Inc. Sebastapol, CA., pp.25

Related Page: MongoDB Vs MySQL - Which Is A Better Database? (<https://mindmajix.com/mongodb-vs-mysql>)

Q201) MongoDB : find value in Array with multiple criteria

I have the following documents:

```
{_id : 1, numbers : [-1000, 1000]}
```

```
{_id : 2, numbers : [5]}
```

I'm trying to get a query that will find a document that has a value in the numbers array that is between -10 and 10 (in this case, _id : 2). However, when I try the following:

```
db.foo.find({numbers : $and : [{ $gt : -10 }, { $lt : 10 } ]})
```

it returns all documents. Is this possible to do without map-reduce? Thanks, -JWW

You can use \$elemMatch to check if an element in an array matches a specified match expression.

In this case, you can use it to get a document whose numbers array has an element that is between -10 and 10:

```
db.foo.find( { numbers : { $elemMatch : { $gt : -10 , $lt : 10 } } } );
```

This will just return the _id : 2 document.

Q202) MongoDB 2.1 Aggregate Framework Sum of Array Elements matching a name

This is a question about the best way to add up a series of data in an array where I have to match another element. I'm trying to use the 2.2 Aggregation framework and it's possible I can do this with a simple group.

So for a given set of documents I'm trying to get an output like this;

Skip code block

```
{
```

Call us on

✉ Drop us a Query

```
{
  "_id" : null,
  "numberOf": 2,
  "Sales" : 468000,
  "profit" : 246246,
}
],
"ok" : 1
}
```

Now, I originally had a list of documents, containing values assigned to named properties, eg;
Skip code block

```
[
{
  _id : 1,
  finance: {
    sales: 234000,
    profit: 123123,
  }
},
{
  _id : 2,
  finance: {
    sales: 234000,
    profit: 123123,
  }
}
]
```

This was easy enough to add up, but the structure didn't work for other reasons. For instance, there are may other columns like "finance" and I want to be able to index them without creating thousands of indexes, so I need to convert to a structure like this;

Skip code block

```
[
{
  _id : 1,
  finance: [
    {
      "k": "sales",
      "v": {
        "description": "sales over the year",

```

Call us on

✉ Drop us a Query

```

v: 234000,
}
},
{
  "k": "profit",
  "v": {
    "description": "money made from sales",
    v: 123123,
  }
}
]
}
,
{
  _id : 2,
  finance: [
    {
      "k": "sales",
      "v": {
        "description": "sales over the year",
        v: 234000,
      }
    },
    {
      "k": "profit",
      "v": {
        "description": "money made from sales",
        v: 123123,
      }
    }
  ]
}
]

```

I can index finance.k if I want, but then I'm struggling to build an aggregate query to add up all the numbers matching a particular key. This was the reason I originally went for named properties, but this really needs to work in a situation whereby there are thousands of "k" labels.

Does anyone know how to build an aggregate query for this using the new framework? I've tried this;

Skip code block

```
db.projects.aggregate([
```



Call us on

Drop us a Query

```

$match: {
// QUERY
$and: [
// main query
{}],
]
}
},
{
$group: {
_id: null,
"numberOf": { $sum: 1 },
"sales": { $sum: "$finance.v.v" },
"profit": { $sum: "$finance.v.v" },
}
},
])
but I get;
{
"errmsg" : "exception: can't convert from BSON type Array to double",
"code" : 16005,
"ok" : 0
}

```

**** For extra kudos, I'll need to be able to do this in a MapReduce query as well.**

You can use the aggregation framework to get sales and profit and any other value you may be storing in your key/value pair representation.

For your example data:

Skip code block

```

var pipeline = [
{
"$unwind" : "$finance"
},
{
"$group" : {
"_id" : "$finance.k",
"numberOf" : {
"$sum" : 1
},
"total" : {
"$sum" : "$finance.v.v"
}
}
}
]

```

Call us on

✉ Drop us a Query

```

}
}
}
]
R = db.tb.aggregate( pipeline );
printjson(R);
{
  "result" : [
    {
      "_id" : "profit",
      "numberOf" : 2,
      "total" : 246246
    },
    {
      "_id" : "sales",
      "numberOf" : 2,
      "total" : 468000
    }
  ],
  "ok" : 1
}

```

If you have additional k/v pairs then you can add a match which only passes through k values in ["sales","profit"].

You will have to use '\$unwind' to break out the values in the array, which will mean that you can't get the sum of the sales and the profit in a single aggregation command. Given that, the query itself is easy:

Skip code block

```

var pipeline = [
  { "$unwind": "$finance" },
  { "$match": { "finance.k": "sales" } },
  { $group:
    { _id: null,
      numberOf: { "$sum": 1 },
      sales: { "$sum": "$finance.v.v" }
    }
  },
  ];
R = db.tb.aggregate( pipeline );
printjson(R);
{

```

 Call us on

 Drop us a Query


```
{
  "_id" : null,
  "numberOf" : 2,
  "sales" : 236340
},
{
  "ok" : 1
}
```

You can run a similar query for profit, just substitute “profit” for “sales” in the “\$match” operator.

Oh, and here’s the map/reduce example:

Skip code block

```
map = function() {
  var ret = { sales: 0.0 , profit: 0.0, count: 1 };
  // iterate over 'finance[]' array
  this.finance.forEach( function (i) {
    if ( i.k == "sales" ) ret.sales = i.v.v ;
    if ( i.k == "profit" ) ret.profit = i.v.v ;
  } );
  emit( 1, ret );
}

reduce = function(key, values) {
  var ret = { sales: 0.0 , profit: 0.0, count: 0 };
  values.forEach(function(v) {
    ret.sales += v.sales;
    ret.profit += v.profit;
    ret.count += v.count;
  });
  return ret;
};

//
// run map/reduce
//
res = SOURCE.mapReduce( map, reduce );
```

Q203) Select * group by in mongo aggregation

I am trying to do something that I think is quite simple. Suppose I have a series of records in mongo that have a common key, and variable number of attributes. I want to select all attributes and group by name across the records. For example

```
{ Name: George, x: 5, y: 3 }
```

```
{ Name: George, z: 9 }
```

📞 Call us on



✉ Drop us a Query

```
{ Name: Rob, x: 12, y: 2 }
```

I would like to produce a CSV that looks like this:

```
Name  X  Y  Z
```

```
George 5  3  9
```

```
Rob    12 2
```

Tried

```
DB.data.aggregate({ $group : { _id : "$Name" } })
```

Unfortunately I get back all the names as records but not the union of all the possible attributes.

If you want to combine the attributes, you'll need to add those to the group. For example, using `$addToSet` to find the unique values of the x,y,z attributes grouped by each name:

```
db.data.aggregate(  
  { $group : {  
    _id : "$Name",  
    x: { $addToSet: "$x" },  
    y: { $addToSet: "$y" },  
    z: { $addToSet: "$z" },  
  } }  
)
```

Returns:

Skip code block

```
{  
  "result" : [  
    {  
      "_id" : "Rob",  
      "x" : [  
        12  
      ],  
      "y" : [  
        2  
      ],  
      "z" : [ ]  
    },  
    {  
      "_id" : "George",  
      "x" : [  
        5  
      ],  
      "y" : [  
        3
```



```

"z" : [
9
]
}
],
"ok" : 1
}

```

Q204) Creating custom Object ID in MongoDB

I am creating a service for which I will use MongoDB as a storage backend. The service will produce a hash of the user input and then see if that same hash (+ input) already exists in our dataset.

The hash will be unique yet random (= non-incremental/sequential), so my question is:

Is it -legitimate- to use a random value for an Object ID? Example:

```
$object_id = new Mongoid(HEX-OF-96BIT-HASH);
```

Or will MongoDB treat the ObjectId differently from other server-produced ones, since a "real" ObjectId also contains timestamps, machine_id, etc?

What are the pros and cons of using a 'random' value? I guess it would be statistically slower for the engine to update the index on inserts when the new _id's are not in any way incremental – am I correct on that?

Yes it is perfectly fine to use a random value for an object id, if some value is present in _id field of a document being stored, it is treated as objectId.

Since _id field is always indexed, and primary key, you need to make sure that different objectId is generated for each object. There are some guidelines to optimize user defined object ids :

<https://www.mongodb.org/display/DOCS/Optimizing+Object+IDs#OptimizingObjectIDs-Use+the+collection's+natural+primary+key+in+the+id+field>.

Whether it is good or bad depends upon it's uniqueness. Of course the ObjectId provided by MongoDB is quite unique so this is a good thing. So long as you can replicate that uniqueness then you should be fine.

There are no inherent risks/performance losses by using your own ID. I guess using it in string form might use up more index/storage/querying power but there you are using it in Mongoid (ObjectId) form which should preserve the strengths of not storing it in a simple string.

While any values, including hashes, can be used for the _id field, I would recommend against using random values for two reasons:

1. You may need to develop a collision-management strategy in the case you produce identical random values for two different objects. In the question, you imply that you'll generate IDs using a some type of a hash algorithm. I would not consider these values "random" as they are based on the content you are digesting with the hash. The probability of a collision then is a function of the diversity of content and the hash algorithm. If you are using something like MD5 or SHA-1, I wouldn't worry about the algorithm, just the content you are hashing. If you need to develop a collision-management strategy then you definitely should not use random or hash-based IDs as collision management in a clustered environment

📞 Call us on

✉ Drop us a Query

is complicated and requires additional queries.

2. Random values as well as hash values are purposefully meant to be dispersed on the number line. That (a) will require more of the B-tree index to be kept in memory at all times and (b) may cause variable insert performance due to B-tree rebalancing. MongoDB is optimized to handle ObjectIDs, which come in ascending order (with one second time granularity). You're likely better off sticking with them.

Q205) How to resolve error :dbpath (/data/db/) does not exist permanently in MongoDB

I have installed mongodb in my Ubuntu 10.04.

I know that when it comes to start the mongodb server with the command "mongod", then it expects /data/db folder and it can be easily resolved by creating "/data/db/". One more way is to provide your own path using mongod -dbpath "path", when we intend to give our own custom path for db.

But while going through <https://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/> link i found that there is a configuration file.

I made the following changes to it.

mongodb.conf

```
dbpath=/EBS/Work/mongodb/data/db/
logpath=/EBS/Work/mongodb/mongodb.log
logappend=true
```

But still when I try to start the server with "mongod" it throws the same **error i.e error :dbpath (/data/db/) does not exist**. I wanted to know that how can I permanently redirect my dbpath to my own custom folder cause everytime you don't want to type the path using "mongod -dbpath path". Rather we look to make some changes in configuration file.

Assuming you have followed the instructions to install a packaged version of MongoDB from 10gen, you should be starting and stopping mongod using service.

To start mongod:

```
sudo service mongodb start
```

To stop mongod:

```
sudo service mongodb stop
```

If you use the service command to start and stop, it should be using the configuration file: /etc/mongodb.conf.

Starting mongod from the command line

If you run mongod directly instead of using the service definition, you will also have to specify a configuration file as a command line parameter if you want one to be used:

```
mongod -config /etc/mongodb.conf
```

here is how i solve this problem. followed their official Doc . HERE. long story short , I created a directory inside /srv and executed the command mongod -dbpath /srv/mongodb/ RAN like a champ.

Skip code block

```
~$ mongod
```

```
Tue Jun 3 20:27:39.564 [initandlisten] MongoDB starting : pid=5380 port=27017 dbpath=/srv/mongodb/
```

```
64-bit host= -SVE1411EGXB
```

📞 Call us on

✉ Drop us a Query

```

Tue Jun 3 20:27:39.564 [initandlisten] db version v2.4.10
Tue Jun 3 20:27:39.564 [initandlisten] git version: e3d78955d181e475345ebd60053a4738a4c5268a
Tue Jun 3 20:27:39.564 [initandlisten] build info: Linux ip-10-2-29-40 2.6.21.7-2.ec2.v1.2.fc8xen #1 SMP
Fri Nov 20 17:48:28 EST 2009 x86_64 BOOST_LIB_VERSION=1_49
Tue Jun 3 20:27:39.564 [initandlisten] allocator: tcmalloc
Tue Jun 3 20:27:39.564 [initandlisten] options: { dbpath: "/srv/mongodb/" }
Tue Jun 3 20:27:39.565 [initandlisten] exception in initAndListen: 10296
*****

ERROR: dbpath (/srv/mongodb/) does not exist.
Create this directory or give existing directory in -dbpath.
See https://dochub.mongodb.org/core/startingandstoppingmongo
*****

, terminating
Tue Jun 3 20:27:39.565 dbexit:
Tue Jun 3 20:27:39.565 [initandlisten] shutdown: going to close listening sockets...
Tue Jun 3 20:27:39.565 [initandlisten] shutdown: going to flush diaglog...
Tue Jun 3 20:27:39.565 [initandlisten] shutdown: going to close sockets...
Tue Jun 3 20:27:39.565 [initandlisten] shutdown: waiting for fs preallocator...
Tue Jun 3 20:27:39.565 [initandlisten] shutdown: lock for final commit...
Tue Jun 3 20:27:39.565 [initandlisten] shutdown: final commit...
Tue Jun 3 20:27:39.565 [initandlisten] shutdown: closing all files...
Tue Jun 3 20:27:39.565 [initandlisten] closeAllFiles() finished
Tue Jun 3 20:27:39.565 dbexit: really exiting now
~$ mongod -dbpath /srv/mongodb/
Tue Jun 3 20:27:55.616 [initandlisten] MongoDB starting : pid=5445 port=27017 dbpath=/srv/mongodb/
64-bit host= -SVE1411EGXB
Tue Jun 3 20:27:55.616 [initandlisten] db version v2.4.10
Tue Jun 3 20:27:55.616 [initandlisten] git version: e3d78955d181e475345ebd60053a4738a4c5268a
Tue Jun 3 20:27:55.616 [initandlisten] build info: Linux ip-10-2-29-40 2.6.21.7-2.ec2.v1.2.fc8xen #1 SMP Fri
Nov 20 17:48:28 EST 2009 x86_64 BOOST_LIB_VERSION=1_49
Tue Jun 3 20:27:55.616 [initandlisten] allocator: tcmalloc
Tue Jun 3 20:27:55.616 [initandlisten] options: { dbpath: "/srv/mongodb/" }
Tue Jun 3 20:27:55.617 [initandlisten] exception in initAndListen: 10296
~$ sudo service mongodb start
start: Job is already running: mongodb
~$ sudo service mongodb stop
mongodb stop/waiting
~$ cd /srv/
~$-$/srv$ ls
~/srv$ mkdir mongodb
~/srv$ cd mongodb

```



mkdir: cannot create directory mongodb: Permission denied

~\$ cd - /srv

~\$ sudo chgrp /srv

~\$ sudo chmod 775 /srv

~\$ cd /srv/

/srv\$ ls

/srv\$ mkdir mongodb

/srv\$ ls mongodb

/srv\$ cd

~\$ mongod -dbpath /srv/mongodb/

Tue Jun 3 20:40:57.457 [initandlisten] MongoDB starting : pid=6018 port=27017 dbpath=/srv/mongodb/
64-bit host= -SVE1411EGXB

Tue Jun 3 20:40:57.457 [initandlisten] db version v2.4.10

Tue Jun 3 20:40:57.457 [initandlisten] git version: e3d78955d181e475345ebd60053a4738a4c5268a

Tue Jun 3 20:40:57.457 [initandlisten] build info: Linux ip-10-2-29-40 2.6.21.7-2.ec2.v1.2.fc8xen #1 SMP

Fri Nov 20 17:48:28 EST 2009 x86_64 BOOST_LIB_VERSION=1_49

Tue Jun 3 20:40:57.457 [initandlisten] allocator: tcmalloc

Tue Jun 3 20:40:57.457 [initandlisten] options: { dbpath: "/srv/mongodb/" }

Tue Jun 3 20:40:57.520 [initandlisten] journal dir=/srv/mongodb/journal

Tue Jun 3 20:40:57.521 [initandlisten] recover : no journal files present, no recovery needed

Tue Jun 3 20:41:00.545 [initandlisten] preallocatesFaster=true 36.86

Tue Jun 3 20:41:03.489 [initandlisten] preallocatesFaster=true 35.06

Tue Jun 3 20:41:07.456 [initandlisten] preallocatesFaster=true 34.44

Tue Jun 3 20:41:07.456 [initandlisten] preallocatesFaster check took 9.935 secs

Tue Jun 3 20:41:07.456 [initandlisten] preallocating a journal file /srv/mongodb/journal/prealloc.0

Tue Jun 3 20:41:10.009 [initandlisten] File Preallocator Progress: 985661440/1073741824 91%

Tue Jun 3 20:41:22.273 [initandlisten] preallocating a journal file /srv/mongodb/journal/prealloc.1

Tue Jun 3 20:41:25.009 [initandlisten] File Preallocator Progress: 933232640/1073741824 86%

Tue Jun 3 20:41:37.119 [initandlisten] preallocating a journal file /srv/mongodb/journal/prealloc.2

Tue Jun 3 20:41:40.093 [initandlisten] File Preallocator Progress: 1006632960/1073741824 93%

Tue Jun 3 20:41:52.450 [FileAllocator] allocating new datafile /srv/mongodb/local.ns, filling with zeroes...

Tue Jun 3 20:41:52.450 [FileAllocator] creating directory /srv/mongodb/_tmp

Tue Jun 3 20:41:52.503 [FileAllocator] done allocating datafile /srv/mongodb/local.ns, size: 16MB, took 0.022 secs

Tue Jun 3 20:41:52.517 [FileAllocator] allocating new datafile /srv/mongodb/local.O, filling with zeroes...

Tue Jun 3 20:41:52.537 [FileAllocator] done allocating datafile /srv/mongodb/local.O, size: 64MB, took 0.02 secs

Tue Jun 3 20:41:52.538 [websvr] admin web console waiting for connections on port 28017

Tue Jun 3 20:41:52.538 [initandlisten] waiting for connections on port 27017

☎ Call us on

✉ Drop us a Query

Q206) install mongodb-10gen failed with apt-get

Install mongodb-10gen as <https://docs.mongodb.org/manual/tutorial/install-mongodb-on-debian/> but got error below:

dpkg: error processing /var/cache/apt/archives/mongodb-10gen_2.2.0_amd64.deb (-unpack):
trying to overwrite '/usr/bin/mongoimport', which is also in package mongodb-clients 1:1.4.4-3
configured to not write apport reports

dpkg-deb: subprocess paste killed by signal (Broken pipe)

Errors were encountered while processing:

/var/cache/apt/archives/mongodb-10gen_2.2.0_amd64.deb

E: Sub-process /usr/bin/dpkg returned an error code (1)

A bug here <https://jira.mongodb.org/browse/SERVER-6910>

apt-get remove mongodb-clients first, and then the installation of mongodb-10gen should work.

Q207) Mongodb Explain for Aggregation framework

Is there an explain function for the Aggregation framework in MongoDB? I can't see it in the documentation.

If not is there some other way to check, how a query performs within the aggregation framework?

I know with find you just do

```
db.collection.find().explain()
```

But with the aggregation framework I get an error

Skip code block

```
db.collection.aggregate(
{ $project : { "Tags._id" : 1 } },
{ $unwind : "$Tags" },
{ $match: { $or: [{ "Tags._id": "tag1?" }, { "Tags._id": "tag2?" } ] } },
{
$group:
{
_id : { id: "$_id" },
"count": { $sum: 1 }
}
},
{ $sort: { "count": -1 } }
).explain()
```

There isn't an explain() feature for the Aggregation Framework as at MongoDB 2.2.0, however you should read up on the documentation regarding Optimizing Performance.

The following pipeline operators take advantage of an index when they occur at the beginning of the pipeline:

\$match \$sort \$limit \$skip

The above operators can also use an index when placed before the following aggregation operators:

Call us on

Drop us a Query

`$project $unwind $group`

So for your specific example, the pipeline will execute faster if you add an initial `$match` at the beginning to reduce the number of documents that need to be projected/unwound for the subsequent `$match`.

Update

MongoDB 2.6 (which is currently a Release Candidate) now includes an `explain` option for the aggregation pipeline.

Skip code block

```
db.collection.aggregate([
  { $project : { "Tags._id" : 1 } },
  { $unwind : "$Tags" },
  { $match: { $or: [{ "Tags._id": "tag1?" }, { "Tags._id": "tag2?" } ] } },
  { $group: {
    _id : "$_id",
    count: { $sum: 1 }
  } },
  { $sort: { "count": -1 } }
],
{
  explain: true
})
```

Starting with version 2.6.x mongodb allows users to do `explain` with aggregation framework.

All you need to do is to add `explain : true`

```
db.records.aggregate(
[ ...your pipeline...],
{ explain: true }
)
```

Thanks to Rafa, I know that it was possible to do even in 2.4, but only through `runCommand()`. But now you can use `aggregate` as well.

Related Page: MongoDB Aggregate (<https://mindmajix.com/mongodb-aggregate>)

Q208) Run script against replica set in MongoDB

I have replica set of 3 nodes and I want to run a cleanup script against it every end of day. What I would do if there was only single node would be a simple bash script:

```
~/mongo/bin/mongo host:port cleanupScript.js
```

But since I want to run the same script against replica-set I can't use this approach. I would need to somehow find which node is primary and run the script against that node.

So the question: Is there a way how to run the script against whole replica set and let the mongoprocess pick the primary node and execute on it?

Thanks!
Call us on

✉ Drop us a Query

The mongo shell can connect directly to a replica set – this works with 2.4 (current), 2.2 (previous) and 2.0 (the version before that).

Assuming you have a replica set called myrs and your hosts are host1:27017, etc. use the following syntax:

```
mongo -host myrs/host1:27017
```

The shell will figure out the rest, including connecting to the primary and if the primary steps down or goes away it will reconnect to the new primary once it's elected.

Unfortunately, the 'mongo' shell does not support making connections to a replica set: only to individual nodes. You have two choices if you want to do this:

1. Use a different language, which supports making a connection to a replica set. (PHP, Python, Perl, Java and Ruby all support this.)
2. Have a driver script that runs an 'rs.status()' command, parses the output of that command, and determines the current primary. If it's not the node you've connected to, then re-connect to the correct primary node

I wish I had a better answer for you.

I normally set priority for a node in replicaset. This gives me freedom to chose which node should get read and write load.

In your case, I think, if you set priority for the nodes then you can always run your script against the highest priority node as that node will be primary almost all the time.

Setting priority is quite simple and straight forward. You can check this link <https://docs.mongodb.org/manual/tutorial/force-member-to-be-primary/>

I hope this will solve your problem.

OKAY.... Probably this is what you need..

```
#!/bin/bash
```

```
PRIMARY=`~/mongo/bin/mongo localhost:27017 -eval "printjson(rs.isMaster())" | grep "primary" | cut -d"'" -f4`
```

```
echo "$PRIMARY"
```

```
~/mongo/bin/mongo "$PRIMARY" cleanupScript.js
```

Run this on any node and it will give you the "server:port" of the primary. Give full path to the mongo executable.. just to avoid errors.

PS: the whole command is in the backticks. Somehow those are not visible so thought of telling you.

Q209) Couldn't connect to server 127.0.0.1:27017

I'm getting the following error:

```
alex@alex-K43U:/ $ mongo
```

```
MongoDB shell version: 2.2.0
```

```
connecting to: test
```

```
Thu Oct 11 11:46:53 Error: couldn't connect to server 127.0.0.1:27017 src/mongo/shell/mongo.js:91
```

```
exception: connect failed
```

```
alex@alex-K43U:/ $
```

📞 Call us on



✉ Drop us a Query

This is what happens when I try to start mongodb:

* Starting database mongod [fail]

I already tried mongo -repair

I made chown and chmod to var, lib, and data/db and log mongod.

Not sure what else to do. Any suggestions?

mongodb.log:

Skip code block

**** SERVER RESTARTED ****

Thu Oct 11 08:29:40

Thu Oct 11 08:29:40 warning: 32-bit servers don't have journaling enabled by default. Please use -journal if you want durability.

Thu Oct 11 08:29:40

Thu Oct 11 08:29:41 [initandlisten] MongoDB starting : pid=1052 port=27017 dbpath=/var/lib/mongod 32-bit host=alex-K43U

Thu Oct 11 08:29:41 [initandlisten]

Thu Oct 11 08:29:41 [initandlisten] ** NOTE: when using MongoDB 32 bit, you are limited to about 2 gigabytes of data

Thu Oct 11 08:29:41 [initandlisten] ** see <https://blog.mongodb.org/post/137788967/32-bit-limitations>

Thu Oct 11 08:29:41 [initandlisten] ** with -journal, the limit is lower

Thu Oct 11 08:29:41 [initandlisten]

Thu Oct 11 08:29:41 [initandlisten] db version v2.2.0, pdfile version 4.5

Thu Oct 11 08:29:41 [initandlisten] git version: f5e83eae9cfbec7fb7a071321928f00d1b0c5207

Thu Oct 11 08:29:41 [initandlisten] build info: Linux domU-12-31-39-01-70-B4 2.6.21.7-2.fc8xen #1 SMP Fri Feb 15 12:39:36 EST 2008 i686 BOOST_LIB_VERSION=1_49

Thu Oct 11 08:29:41 [initandlisten] options: { config: "/etc/mongod.conf", dbpath: "/var/lib/mongod", logappend: "true", logpath: "/var/log/mongod/mongod.log" }

Thu Oct 11 08:29:41 [initandlisten] Unable to check for journal files due to: boost::filesystem::basic_directory_iterator constructor: No such file or directory: "/var/lib/mongod/journal"

Unclean shutdown detected.

Please visit <https://dochub.mongodb.org/core/repair> for recovery instructions.

Thu Oct 11 08:29:41 [initandlisten] exception in initAndListen: 12596 old lock file, terminating

Thu Oct 11 08:29:41 dbexit:

Thu Oct 11 08:29:41 [initandlisten] shutdown: going to close listening sockets...

Thu Oct 11 08:29:41 [initandlisten] shutdown: going to flush diaglog...

Thu Oct 11 08:29:41 [initandlisten] shutdown: going to close sockets...

Thu Oct 11 08:29:41 [initandlisten] shutdown: waiting for fs preallocator...

Call us on

✉ Drop us a Query

Thu Oct 11 08:29:41 [initandlisten] shutdown: closing all files...

Thu Oct 11 08:29:41 [initandlisten] closeAllFiles() finished

Thu Oct 11 08:29:41 dbexit: really exiting now

EDIT:

I removed the lock then did mongod repair and got this error:

Thu Oct 11 12:05:37 [initandlisten] exception in initAndListen: 10309 Unable to create/openlock file: /data/db/mongod.lock errno:13 Permission denied Is a mongod instance already running?, terminating so I did it with sudo:

Skip code block

alex@alex-K43U:~\$ sudo mongod -repair

Thu Oct 11 12:05:42

Thu Oct 11 12:05:42 warning: 32-bit servers don't have journaling enabled by default. Please use -journal if you want durability.

Thu Oct 11 12:05:42

Thu Oct 11 12:05:42 [initandlisten] MongoDB starting : pid=5129 port=27017 dbpath=/data/db/ 32-bit host=alex-K43U

Thu Oct 11 12:05:42 [initandlisten]

Thu Oct 11 12:05:42 [initandlisten] ** NOTE: when using MongoDB 32 bit, you are limited to about 2 gigabytes of data

Thu Oct 11 12:05:42 [initandlisten] ** see <https://blog.mongodb.org/post/137788967/32-bit-limitations>

Thu Oct 11 12:05:42 [initandlisten] ** with -journal, the limit is lower

Thu Oct 11 12:05:42 [initandlisten]

Thu Oct 11 12:05:42 [initandlisten] db version v2.2.0, pdf file version 4.5

Thu Oct 11 12:05:42 [initandlisten] git version: f5e83eae9cfbec7fb7a071321928f00d1b0c5207

Thu Oct 11 12:05:42 [initandlisten] build info: Linux domU-12-31-39-01-70-B4 2.6.21.7-2.fc8xen #1 SMP Fri Feb 15 12:39:36 EST 2008 i686 BOOST_LIB_VERSION=1_49

Thu Oct 11 12:05:42 [initandlisten] options: { repair: true }

Thu Oct 11 12:05:42 [initandlisten] Unable to check for journal files due to: boost::filesystem::basic_directory_iterator constructor: No such file or directory: "/data/db/journal"

Thu Oct 11 12:05:42 [initandlisten] finished checking dbs

Thu Oct 11 12:05:42 dbexit:

Thu Oct 11 12:05:42 [initandlisten] shutdown: going to close listening sockets...

Thu Oct 11 12:05:42 [initandlisten] shutdown: going to flush diaglog...

Thu Oct 11 12:05:42 [initandlisten] shutdown: going to close sockets...

Thu Oct 11 12:05:42 [initandlisten] shutdown: waiting for fs preallocator...

Thu Oct 11 12:05:42 [initandlisten] shutdown: closing all files...

Thu Oct 11 12:05:42 [initandlisten] closeAllFiles() finished

Thu Oct 11 12:05:42 [initandlisten] shutdown: removing fs lock...

Thu Oct 11 12:05:42 dbexit: really exiting now

Call us on

✉ Drop us a Query

But still having the same problem.

The log indicates that mongod is terminating because there is an old lock file.

If you are not and were not running with journaling, remove the lock file, run repair, and start mongod again.

If you are or were running with journaling turned on, see the relevant Mongo DB docs. Note that they say "If you are running with Journaling you should not do a repair to recover to a consistent state." So if you were journaling, the repair may have made things worse.

Skip code block

Step 1: Remove lock file.

```
sudo rm /var/lib/mongodb/mongod.lock
```

Step 2: Repair mongod.

```
mongod -repair
```

Step 3: start mongod.

```
sudo start mongod
```

or

```
sudo service mongod start
```

Step 4: Check status of mongod.

```
sudo status mongod
```

or

```
sudo service mongod status
```

Step 5: Start mongo console.

```
mongo
```

Did you run mongod before running mongo?

I followed installation instructions for mongod from <https://docs.mongodb.org/manual/tutorial/install-mongodb-on-os-x/> and I had the same error as you only when I ran mongo before actually running the mongo process with mongod. I thought installing mongod would also launch it but you need to launch it manually with mongod before you do anything else that needs mongod.

try

```
sudo service mongod start
```

I solved my problem by this

Q210) Operate on all databases from the mongo shell

We have a system with many different mongo databases. I regularly want to write ad-hoc queries that will apply to all (or a subset) of them, without having a priori knowledge of what databases are there.

I can do show dbs, which will visually print a list, but is there a way to do something like:

```
var db_list = listDatabases();  
for (i = 0; i < db_list.length; i++) {  
  do_something(db_list[i])  
}
```



My problem with show dbs is that it doesn't capture any return values, so I can't do anything productive

☎ Call us on

✉ Drop us a Query

with the output.

You can use the 'listDatabases' admin command for that:

```
var db_list = db.adminCommand('listDatabases');
```

That returns an object that looks like this:

Skip code block

```
{
  "databases" : [
    {
      "name" : "test",
      "sizeOnDisk" : 2097152000,
      "empty" : false
    },
    {
      "name" : "local",
      "sizeOnDisk" : 1,
      "empty" : true
    }
  ],
  "totalSize" : 8487174144,
  "ok" : 1
}
```

Q211) I accidentally named a collection "stats" in MongoDB and now cannot rename it

Oops.

I use mongoose, and accidentally created a collection "stats". I didn't realize this was going to be an issue until a few weeks later, so I now need to rename (rather than just delete) the collection...

However, my attempts have hit a predictable problem:

```
PRIMARY> db.stats.find();
```

```
Thu Oct 18 10:39:43 TypeError: db.stats.find is not a function (shell):1
```

```
PRIMARY> db.stats.renameCollection('statssnapshots');
```

```
Thu Oct 18 10:39:45 TypeError: db.stats.renameCollection is not a function (shell):1
```

Try db["stats"].find() and db["stats"].renameCollection('statssnapshots').

You could also do db.getCollection("stats").find().

Q212) Set max on mongo capped collection

I have an existing collection that I need to convert into a Capped Collection using the method listed:

```
> db.runCommand({"convertToCapped": "mycoll", size: 100000});
```

However, the max field is not accepted as a parameter

```
> db.mycoll.convertToCapped
```

```
function (bytes) {
```

```
  if (bytes) {
```



}
}

✉ Drop us a Query

```
throw "have to specify # of bytes";
}
return this._dbCommand({convertToCapped: this._shortName, size: bytes});
}
```

Any idea how to set this?

max is only an option in the createCollection method, not convertToCapped:

```
db.createCollection("mycoll", {capped:true, size:100000, max:100});
```

There's a cloneCollectionAsCapped, but it doesn't look like there's a max doc option there either:

<https://docs.mongodb.org/manual/reference/command/cloneCollectionAsCapped/>

You may need to create a new capped collection with the max parameter and transfer data and indices from the existing collection. See <https://learnmongo.com/posts/easily-move-documents-between-collections-or-databases/>

Q213) Does MongoDB reuse deleted space?

First off, I know about this question:

Auto compact the deleted space in mongodb?

My question is not about shrinking DB file sizes though, but more about the reuse of deleted space. Say I have 100K documents in a collection, I then delete 50K of those. Will Mongo reuse the space within its data file that the deleted documents have freed? Or are they simply "marked" as deleted?

I don't care so much about the actual size of the file on disk, its more about "does it just grow and grow".

When documents are deleted the space left behind is put into a free list, yes. However, to use the space there will need to be similarly sized documents inserted later, and MongoDB will need to find an appropriate space for that document within a certain time frame (once it times out looking at the list, it will just append) otherwise the space re-use is not going to happen very often. This deletion is done within the data files, so there is no disk space reclamation happening here – all of this is done internally within the existing data files.

If you subsequently do a repair, or resync a secondary from scratch, the data files are rewritten and the space on disk will be reclaimed (any padding on docs is also removed). This is where you will see actual space reclamation on-disk. For any other actions (compact included) the on disk usage will not change and may even increase.

With 2.2 you can now use the collMod command and the use PowersOf2Sizes option to make the re-use of deleted space more likely. This means that the initial space allocation for a document is a bit less efficient (512 bytes for a 400 byte doc for example) but means that when a new doc is inserted it is more likely to be able to re-use that space. If you are deleting (or growing and hence moving) documents a lot, then this will be more efficient in the long term.

For anyone that is interested, one of the people that wrote a lot of the storage code (Mathias Stearn) has a great presentation about the storage internals, which can be found here



Q214) Can not delete collection from mongodb

📞 Call us on

✉ Drop us a Query

Can not delete the collection from the shell,

The thing that the collection is available and my php script is accessing it (selecting|updating)

But when I used:

```
db._registration.drop()
```

it gives me an error:

Date, JS Error: TypeErrorL db._registration has no properties (shell): 1

The problem is not with deleting the collection. The problem is with accessing the collection. So you would not be able to update, find or do anything with it from the shell. As it was pointed in mongodb JIRA, this is a bug when a collection has characters like `_`, `-` or `.`

Nevertheless this type of names for collections is acceptable, but it cause a problem in shell.

You can delete it in shell with this command:

```
db.getCollection("_registration").drop()
```

but I would rather rename it (of course if it is possible and will not end up with a lot of changing).

Q215) explain() in MongoDB: differences between “nscanned” and “nscannedObjects”

I cannot get the exact difference between “nscanned” and “nscannedObjects” in the MongoDB’s explain query output.

On MongoDB Explain documentation I can read:

nscanned Number of items (documents or index entries) examined. Items might be objects or index keys. If a “covered index” is involved, nscanned may be higher than nscannedObjects.

nscannedObjects Number of documents scanned.

What’s the different between these two fields? And more specific what does exactly mean when I have a query, which uses a BtreeCursor (an index), and these two fields have two different values, for example:

```
{
  "cursor" : "BtreeCursor a_1_b_1",
  "isMultiKey" : false,
  "n" : 5,
  "nscannedObjects" : 5,
  "nscanned" : 9,
  (...)
}
```

I know what a “covered index” is. I would like to understand exactly what the query did in the example above. Did it pass through (“scanned”) 9 elements (nscanned = 9), where all of them are index entries and read (“examined”) the value of only 5 of them (nscannedObjects = 5) to produce the result set?

This means that :

The query returned 5 documents – n

scanned 9 documents from the index – nscanned

and then read 5 full documents from the collection – nscannedObjects

Similar example is given at :

<https://docs.mongodb.org/manual/core/read-operations/#measuring-index-use>

☎ Call us on

✉ Drop us a Query

Q216) MongoDB Database vs Collection

I am designing a system with MongoDB (64 bit version) to handle a large amount of users (around 100,000) and each user will have large amounts of data (around 1 million records).

What is the best strategy of design?

Dump all records in single collection

Have a collection for each user

Have a database for each user.

Many Thanks,

So you're looking at somewhere in the region of 100 billion records (1 million records * 100,000 users).

The preferred way to deal with large amounts of data is to create a sharded cluster that splits the data out over several servers that are presented as single logical unit via the mongo client.

Therefore the answer to your question is put all your records in a single sharded collection.

The number of shards required and configuration of the cluster is related to the size of the data and other factors such as the quantity and distribution of reads and writes. The answers to those questions are probably very specific to your unique situation, so I won't attempt to guess them.

I'd probably start by deciding how many shards you have the time and machines available to set up and testing the system on a cluster of that many machines. Based on the performance of that, you can decide whether you need more or fewer shards in your cluster

Q217) MongoDB upsert – insert or update

I can't find this in the documentation in any of the obvious places. I'd like to know if it is possible to know if mongo executed an insert or update in the upsert operation? Thanks.

Yes there is, on a save call (or getLastError) the update function will return an array with an upsert field and a updatedExisting field.

You can read the PHP version of this here:

<https://php.net/manual/en/mongocollection.insert.php> towards the bottom.

As it says within the documentation on upserted:

If an upsert occurred, this field will contain the new record's _id field. For upserts, either this field or updatedExisting will be present (unless an error occurred).

So upserted contains the _id of the new record if a insert was done or it will increment updatedExisting if it updated a record.

I am sure a similar thing appears in all drivers.

Edit

It will actually be a boolean in the updatedExisting field of true or false

For reference only, in node.js:

```
collection.update( source, target, { upsert: true }, function(err, result, upserted) {
```

```
...
```

```
});
```

For reference only, in node.js using Mongoose 3.6:

```
model.update( findquery, updatequery, { upsert: true }, function(err, numberAffected, rawResponse) {
```

Can us on

Drop us a query

...

});

Where rawResponse looks like this when it has updated an existing document:

```
{ updatedExisting: true,
  n: 1,
  connectionId: 222,
  err: null,
  ok: 1 }
```

And it looks like this when it has created a new document:

```
{ updatedExisting: false,
  upserted: 51eebc080eb3e2208a630d8e,
  n: 1,
  connectionId: 222,
  err: null,
```

(Both cases would return numberAffected = 1)

Q218) How to insert an element to MongoDB internal list?

I have the following doc stored in MongoDB:

Skip code block

```
{
  name: 'myDoc',
  list: [
    {
      id:1
      items:[
        {id:1, name:'item1'},
        {id:2, name:'item2'}
      ]
    },
    {
      id:2
      items:[
        {id:1, name:'item1'},
        {id:3, name:'item3'}
      ]
    }
  ]
}
```

I found a way to add an element to 'list' using '\$addToSet' but I couldn't find a way to add to a specific list of 'items' an item.

📞 Call us on

✉ Drop us a Query

e.g. I get the following:

```
{id:5, name:'item5'}
```

and I want to add it to the element's item in the list with id:2.

One way of doing it would be with \$push:

```
db.col.update(
{ name: 'doc', 'list.id': 2 },
{$push: {'list.$.items': {id: 5, name: 'item5'}}}
)
```

<https://docs.mongodb.org/manual/reference/operator/push/>

You can also replace \$push with other operators like (possibly) \$addToSet to get the exact results you are looking for.

You can use this: -

```
> var toInsert = {id: 5, name: 'item6'}
> db.abc.update(
{ name: 'myDoc', list: { $elemMatch: { id: 2 } } },
{ $addToSet: { 'list.$.items': toInsert } }
)
```

The query part will find the document from the list array with id = 2. Then we use \$ positional element to add a new element at that array index.

See positional \$ operator

You can also replace list: {\$elemMatch: {id: 2}} with just 'list.id': 2.

But using \$elemMatch will be better, when you want to update based on multiple elements of your array.

For e.g., when your matching criteria is id and some another field in the array say length: -

```
list: {$elemMatch: {id: 2, length: 5}}
```

Q219) Fastest way to remove duplicate documents in mongodb

I have approximately 1.7M documents in mongodb (in future 10m+). Some of them represent duplicate entry which I do not want. Structure of document is something like this:

```
{
  _id: 14124412,
  nodes: [
    12345,
    54321
  ],
  name: "Some beauty"
}
```

Document is duplicate if it has at **least one node same** as another document with **same name**. What is the fastest way to remove duplicates?

Assuming you want to permanently delete docs that contain a duplicate name + nodes entry from the collection, you can add a unique index with the dropDups: true option:

☎ Call us on

✉ Drop us a Query

```
db.test.ensureIndex({name: 1, nodes: 1}, {unique: true, dropDups: true})
```

As the docs say, use extreme caution with this as it will delete data from your database. Back up your database first in case it doesn't do exactly as you're expecting.

Related Page: MongoDB Update Document (<https://mindmajix.com/mongodb-update-document>)

Q220) MongoDB \$query operator ignores index?

I created a test MongoDB collection "sampleCollection" with documents which looks like:

```
"_id" : ObjectId("510929e041cb2179b41ace1c"),
```

```
"stringField" : "Random string0",
```

```
"longField" : NumberLong(886)
```

and has index on field "stringField". When I execute

```
db.sampleCollection.find({"stringField":"Random string0?}).explain()
```

everything is ok:

Skip code block

```
"cursor" : "BtreeCursor stringField_1",
```

```
"isMultiKey" : false,
```

```
"n" : 2,
```

```
"nscannedObjects" : 2,
```

```
"nscanned" : 2,
```

```
"nscannedObjectsAllPlans" : 2,
```

```
"nscannedAllPlans" : 2,
```

```
"scanAndOrder" : false,
```

```
"indexOnly" : false,
```

```
"nYields" : 0,
```

```
"nChunkSkips" : 0,
```

```
"millis" : 0,
```

```
"indexBounds" : {
```

```
"stringField" : [
```

```
[
```

```
"Random string0",
```

```
"Random string0"
```

```
]
```

```
]
```

```
}
```

but

```
db.sampleCollection.find({$query:{"stringField":"Random string0?}}).explain()
```

gets me

Skip code block

```
"cursor" : "BasicCursor",
```

```
"isMultiKey" : false,
```

📞 Call us on



✉ Drop us a Query

```

“n” : 0,
“nscannedObjects” : 4,
“nscanned” : 4,
“nscannedObjectsAllPlans” : 4,
“nscannedAllPlans” : 4,
“scanAndOrder” : false,
“indexOnly” : false,
“nYields” : 0,
“nChunkSkips” : 0,
“millis” : 0,
“indexBounds” : {
}

```

This is not looks like a problem, but I’m using **org.springframework.data.mongodbframework** in production and usage of query constructions is an only way to write repositories. And thus I have a db which completely ignores indexed data.

Is it correct? Or I misunderstood something?

That was funny i cannot decide to say it is a bug or not it is up to you:

There are two available syntax: <https://docs.mongodb.org/manual/reference/operator/query/>

When you using:

```
db.collection.find( { age : 25 } )
```

also will

```
db.collection.find( { age : 25 } ).explain()
```

```
db.collection.find( { age : 25 } ).hint(someindex)
```

work fine.

When you using your solution (the other syntax):

```
db.collection.find( { $query: { age : 25 } } )
```

the output of

```
db.sampleCollection.find({$query:{"stringField":"Random stringO?}}).explain()
```

Will show like the query not using the index

if you also use .hint for the index it will omit the result. :) (That is i do not really understand)

Fortunately there is another syntax for these operations too: you can use:

```
db.sampleCollection.find({$query:{"stringField":"Random stringO?"}, $explain:1})
```

it will have the right output and showed for me the usage of the index. Also there is similar syntax for \$hint.

You can check the documentation here: <https://docs.mongodb.org/manual/reference/meta-query-operators/>

I found this really interesting so i turned on the profiler:

i made a test collection (queryTst) with around 250k docs each with only _id and an age field in the structure with an index on age.

For this query:

☎ Call us on


✉ Drop us a Query

```
db.queryTst.find({$query:{"age":16},$explain:1})
```

i got:

Skip code block

```
{
  "cursor" : "BtreeCursor age_1",
  "isMultiKey" : false,
  "n" : 2,
  "nscannedObjects" : 2,
  "nscanned" : 2,
  "nscannedObjectsAllPlans" : 2,
  "nscannedAllPlans" : 2,
  "scanAndOrder" : false,
  "indexOnly" : false,
  "nYields" : 0,
  "nChunkSkips" : 0,
  "millis" : 0,
  "indexBounds" : {
    "age" : [
      [
        16,
        16
      ]
    ]
  },
  "allPlans" : [
    {
      "cursor" : "BtreeCursor age_1",
      "n" : 2,
      "nscannedObjects" : 2,
      "nscanned" : 2,
      "indexBounds" : {
        "age" : [
          [
            16,
            16
          ]
        ]
      }
    }
  ]
}
```

 Call us on

 Drop us a Query



```

“oldPlan” : {
“cursor” : “BtreeCursor age_1”,
“indexBounds” : {
“age” : [
[
16,
16
]
]
}
},
“server” : “”
}

```

for this:

```
db.queryTst.find({$query:{“age”:16},$explain:1}).explain()
```

i got:

Skip code block

```

“cursor” : “BasicCursor”,
“isMultiKey” : false,
“n” : 0,
“nscannedObjects” : 250011,
“nscanned” : 250011,
“nscannedObjectsAllPlans” : 250011,
“nscannedAllPlans” : 250011,
“scanAndOrder” : false,
“indexOnly” : false,
“nYields” : 0,
“nChunkSkips” : 0,
“millis” : 103,
“indexBounds” : {
},

```

in the profiler log: for the first

Skip code block

```

{
“ts” : ISODate(“2013-01-30T20:35:40.526Z”),
“op” : “query”,
“ns” : “test.queryTst”,
“query” : {
“$query” : {
“age” : 16
}
}
}

```

Call us on



✉ Drop us a Query

```
},
"$explain" : 1
},
"ntoreturn" : 0,
"ntoskip" : 0,
"nscanned" : 2,
"keyUpdates" : 0,
"numYield" : 0,
"lockStats" : {
  "timeLockedMicros" : {
    "r" : NumberLong(368),
    "w" : NumberLong(0)
  },
  "timeAcquiringMicros" : {
    "r" : NumberLong(8),
    "w" : NumberLong(5)
  }
},
"nreturned" : 1,
"responseLength" : 567,
"millis" : 0,
"client" : "127.0.0.1",
"user" : ""
}
```

for the second:

Skip code block

```
{
  "ts" : ISODate("2013-01-30T20:35:47.715Z"),
  "op" : "query",
  "ns" : "test.queryTst",
  "query" : {
    "query" : {
      "$query" : {
        "age" : 16
      }
    },
    "$explain" : 1
  },
  "$explain" : true
},
"ntoreturn" : 0,
```

Call us on



✉ Drop us a Query

```

"ntoskip" : 0,
"nscanned" : 250011,
"keyUpdates" : 0,
"numYield" : 0,
"lockStats" : {
  "timeLockedMicros" : {
    "r" : NumberLong(104092),
    "w" : NumberLong(0)
  },
  "timeAcquiringMicros" : {
    "r" : NumberLong(13),
    "w" : NumberLong(5)
  }
},
"nreturned" : 1,
"responseLength" : 373,
"millis" : 104,
"client" : "127.0.0.1",
"user" : ""
}

```

which somehow means to me that is the explain() cause the table scan in the mixed syntax.

this is not a bug, I happened to have the same question you had, and was about to report it as an issue in mongodb, but reading the documentation it specifies that you shouldn't mix cursor methods (like the explain()) with this query format, you should use \$explain, like in the example provided in this page:

<https://docs.mongodb.org/manual/reference/operator/query/>

Q223). How to index an \$or query with sort

Suppose I have a query that looks something like this:

Skip code block

```

db.things.find({
  deleted: false,
  type: 'thing',
  $or: [{
    'creator._id': someid
  }, {
    'parent._id': someid
  }, {
    'somerelation._id': someid
  }]
}

```

}).sort({
📞 Call us on



✉ Drop us a Query


```
'date.created': -1
```

```
}}
```

That is, I want to find documents that meets one of those three conditions and sort it by newest. However, \$or queries do not use indexes in parallel when used with a sort. Thus, how would I index this query?

<https://docs.mongodb.org/manual/core/indexes/#index-behaviors-and-limitations>

You can assume the following selectivity:

1. deleted- 99%

2. type- 25%

3. creator._id,parent._id, somerelation._id - < 1%

Now you are going to need more than one index for this query, there is no doubt about that.

The question is what indexes?

Now you have to take into consideration that none of your \$ors will be able to sort their data cardinally in a optimal manner using the index due to a bug in MongoDBs query optimiser: <https://jira.mongodb.org/browse/SERVER-1205>.

So you know that the \$or will have some performance problems with a sort and that putting the sort field into the \$or clause indexes is useless atm.

So condering this the first index you want is one that covers the base query you are making. As @Leonid said you can make this into a compound index, however I would not do it the order he has done it. Instead I would do:

```
db.col.ensureIndex({type:-1,deleted:-1,date.created:-1})
```

I am very unsure about the deleted field being in the index at all due to it's super low selectivity, it could in fact create a less performant operation (this is true for most databases including SQL) being in the index rather than being taken out. This part will need testing by you, maybe the field should be last (?)

Subscribe For Free Demo

Phone *

E-mail Address *

SUBSCRIBE

Free Demo for Corporate & Online Trainings.



About The Author

📞 Call us on

✉ Drop us a Query



Prasanthi is an expert writer in MongoDB, and has written for various reputable online and print publications. At present, she is working for Mindmajix, and writes content not only on MongoDB, but also on Sharepoint, Uipath, AWS, and Azure.



(<https://www.dmca.com/Protection/Status.aspx?ID=5b159fca-d195-4289-83e3-b69a0818b38d&refurl=https://mindmajix.com/mongodb-interview-questions>)

Social Share



(<https://www.facebook.com/mindmajix/shareArticle?u=https://statu&title=https://mindmajix.com/mongodb-interview-questions&summary=&source=>

<https://statu&title=https://mindmajix.com/mongodb-interview-questions&summary=&source=>

PREVIOUS ([HTTPS://MINDMAJIX.COM/JAVA-SPRING-INTERVIEW-questions&summary=&source=](https://MINDMAJIX.COM/JAVA-SPRING-INTERVIEW-questions&summary=&source=))

NEXT ([HTTPS://MINDMAJIX.COM/MSBI-INTERVIEW-QUESTIONS&summary=&source=](https://MINDMAJIX.COM/MSBI-INTERVIEW-QUESTIONS&summary=&source=))

DROP US A QUERY

Full Name

E-mail Address

Course Name

US



+1 -

Phone Number *

Message

I'm not a robot

reCAPTCHA
Privacy - Terms

CONTACT US



- ▶ ForeScout Administrator (<https://mindmajix.com/forescout-administrator/>)
- ▶ Peoplesoft CRM (<https://mindmajix.com/peoplesoft-crm/>)
- ▶ Hyperion FDQM (<https://mindmajix.com/hyperion-fdqm/>)
- ▶ Regression (<https://mindmajix.com/regression/>)
- ▶ SAP SD (<https://mindmajix.com/sap-sd/>)
- ▶ PeopleSoft HRMS Functional (<https://mindmajix.com/peoplesoft-hrms-functional/>)
- ▶ BOXI (<https://mindmajix.com/boxi/>)
- ▶ Bigdata Greenplum DBA (<https://mindmajix.com/bigdata-greenplum-dba/>)
- ▶ Azure (<https://mindmajix.com/microsoft-azure/>)
- ▶ IBM API Connect (<https://mindmajix.com/ibm-api-connect/>)

[VIEW MORE \(HTTPS://MINDMAJIX.COM/BLOG-CATEGORY\)](https://mindmajix.com/blog-category)

Popular Courses in 2019

MongoDB Training (<https://mindmajix.com/mongodb-training>)

★★★★★

👤 1489 Learners

Selenium Training (<https://mindmajix.com/selenium-training>)

★★★★★

👤 4807 Learners



Majix

Mindmajix - Online global training platform connecting individuals with the best trainers around the globe. With the diverse range of courses, Training Materials, Resume formats and On Job Support, we have it all covered to get into IT Career. Instructor Led Training - Made easy.

f (<https://www.facebook.com/MindMajixTechnologies>) **t** (<https://twitter.com/mindmajix>)

g+ (<https://plus.google.com/+MindmajixTechnologies>) **in** (<https://www.linkedin.com/company/mindmajix-technologies-pvt-ltd->)



Call us on (<https://www.youtube.com/channel/UCkKemMaRnFPINLHZ0zOYfpA>)

✉ Drop us a Query

Company

Home (<https://mindmajix.com/>)

About Us (<https://mindmajix.com/about>)

Courses (<https://mindmajix.com/all-courses>)

Sample Resumes (<https://mindmajix.com/sample-resumes>)

Blog (<https://mindmajix.com/blog>)

Contact Us (<https://mindmajix.com/contact>)

Reviews (<https://mindmajix.com/reviews>)

Write for us (<https://mindmajix.com/write-for-us>)

Trending Courses

Tableau Training (<https://mindmajix.com/tableau-training>)

Oracle DBA training (<https://mindmajix.com/oracle-dba-training>)

Qlikview Training (<https://mindmajix.com/qlikview-training>)

Docker Training (<https://mindmajix.com/docker-training>)

JBoss Training (<https://mindmajix.com/jboss-training>)

Informatica Training (<https://mindmajix.com/informatica-training>)

Cassandra Training (<https://mindmajix.com/apache-cassandra-training>)

Blockchain Training (<https://mindmajix.com/blockchain-training>)

Contact info

📍 244 Fifth Avenue, Suite 1222 New York(NY) United States (US) - 10001

📞 USA : +1 917 456 8403

✉ info@mindmajix.com (<mailto:info@mindmajix.com>)

📍 #811, 10th A Main, Suite No.506 1st Floor, Indira Nagar Bangalore, India - 560038

📞 India : +91 905 240 3388

✉ info@mindmajix.com (<mailto:info@mindmajix.com>)



📞 Call us on


✉ Drop us a Query

Copyright © 2019 Mindmajix Technologies Inc. All Rights Reserved

Privacy Policy (<https://mindmajix.com/terms-of-service-and-privacy-policy>)


Terms of use (<https://mindmajix.com/terms-of-service-and-privacy-policy>)

Refund Policy (<https://mindmajix.com/terms-of-service-and-privacy-policy>)

 **Call us on**

 **Drop us a Query**



 **Call us on**

 **Drop us a Query**