Running advanced MongoDB queries in R with rmongodb

As MySQL is driving me nuts I'm trying to make myself acquainted with my first "NoSQL" DBMS and it happened to be MongoDB. I'm connecting to it via rmongodb.

The more I play around with rmongodb, the more questions/problems come up with respect to running advanced queries.

First I present some example data before I go into detail about the different types of queries that I can't seem to specify correctly.

Example Data

The example is taken from the MongoDB website and has been simplified a bit.

```
pkg <- "rmongodb"
if (!require(pkg, character.only=TRUE)) {
    install.packages(pkg)
    require(pkg, character.only=TRUE)
}

# Connect to DB
db <- "test"
ns <- "posts"
mongo <- mongo.create(db=db)

# Insert document to collection 'test.users'
b <- mongo.bson.from.list(list(
    "_id"="alex",
    name=list(first="Alex", last="Benisson"),
    karma=1.0,
    age=30,
    test=c("a", "b")
``
```

```
))
mongo.insert(mongo, "test.users", b)

# Insert document to collection 'test.posts'
b <- mongo.bson.from.list(list(
        "_id"="abcd",
        when=mongo.timestamp.create(strptime("2011-09-19 02:00:00",
            "%Y-%m-%d %H:%M:%s"), increment=1),
        author="alex",
        title="Some title",
        text="Some text.",
        tags=c("tag.1", "tag.2"),
        votes=5,
        voters=c("jane", "joe", "spencer", "phyllis", "li"),
        comments=list(
            list(
```

Two questions related to inserting JSON/BSON objects:

1. Document 'test.posts', field `voters` : is it correct to use `c()` in this case?
2. Document 'test.posts', field `comments` : what's the right way to specify this, `c()` or `list()` ?

Top Level Queries: they work a treat

Top level queries work just fine:

```
# Get all posts by 'alex' (only titles)
res <- mongo.find(mongo, "test.posts", query=list(author="alex"),
    fields=list(title=1L))
out <- NULL
while (mongo.cursor.next(res))
    out <- c(out, list(mongo.bson.to.list(mongo.cursor.value(res))))

> out
[[1]]
                      _id                    title
                   "abcd"          "No Free Lunch"
```

Question 1: Basic Sub Level Queries

How can run a simple "sub level queries" (as opposed to top level queries) that need to reach into arbitrarily deep sublevels of a JSON/BSON style MongoDB object? These sub level queries make use of MongoDB's dot notation and I can't seem to figure out how to map that to a valid rmongodb query

In plain MongoDB syntax, something like

```
> db.posts.find( { comments.who : "meghan" } )
```

would work. But I can't figure out how to do that with rmongodb functions

Here's what I tried so far

```
# Get all comments by 'meghan' from 'test.posts'

#--------------------
# Approach 1)
#--------------------
res <- mongo.find(mongo, "test.posts", query=list(comments=list(who="meghan")))
out <- NULL
while (mongo.cursor.next(res))
    out <- c(out, list(mongo.bson.to.list(mongo.cursor.value(res))))

> out
NULL
# Does not work

#--------------------
# Approach 2)
#--------------------
buf <- mongo.bson.buffer.create()
mongo.bson.buffer.start.object(buf, "comments")
mongo.bson.buffer.append(buf, "who", "meghan")
mongo.bson.buffer.finish.object(buf)
query <- mongo.bson.from.buffer(buf)
res <- mongo.find(mongo, "test.posts", query=query)
out <- NULL
while (mongo.cursor.next(res))
    out <- c(out, list(mongo.bson.to.list(mongo.cursor.value(res))))

> out
NULL
# Does not work
```

Question 2: Queries Using $ Operators

These work

Query 1

```
buf <- mongo.bson.buffer.create()
mongo.bson.buffer.start.object(buf, "age")
mongo.bson.buffer.append(buf, "$lte", 30)
mongo.bson.buffer.finish.object(buf)
criteria <- mongo.bson.from.buffer(buf)
criteria

> mongo.find.one(mongo, "test.users", query=criteria)
```

```
    _id : 2      alex
    name : 3
        first : 2    Alex
        last : 2     Benisson

    karma : 1    1.000000
    age : 1      30.000000
    test : 4
        0 : 2    a
        1 : 2    b
```

Query 2

```
buf <- mongo.bson.buffer.create()
mongo.bson.buffer.start.object(buf, "test")
mongo.bson.buffer.append(buf, "$in", c("a", "z"))
mongo.bson.buffer.finish.object(buf)
criteria <- mongo.bson.from.buffer(buf)
criteria
mongo.find.one(mongo, "test.users", query=criteria)
```

However, notice that an atomic set will result in a return value of `NULL`

```
mongo.bson.buffer.append(buf, "$in", "a")
# Instead of 'mongo.bson.buffer.append(buf, "$in", c("a", "z"))'
```

Trying the same with sub level queries I'm lost again

```
buf <- mongo.bson.buffer.create()
mongo.bson.buffer.start.object(buf, "name")
mongo.bson.buffer.start.object(buf, "first")
mongo.bson.buffer.append(buf, "$in", c("Alex", "Horst"))
mongo.bson.buffer.finish.object(buf)
mongo.bson.buffer.finish.object(buf)
criteria <- mongo.bson.from.buffer(buf)
criteria <- mongo.bson.from.buffer(buf)
> criteria
    name : 3
        first : 3
            $in : 4
                0 : 2    Alex
                1 : 2    Horst

> mongo.find.one(mongo, "test.users", query=criteria)
NULL
```

r | mongodb | rmongodb

edited Sep 13 '12 at 12:12
Stennie
18.8k 3 32 56

asked May 29 '12 at 11:58
Rappster
2,495 17 36

The author (Gerald Lindsly) posted a short blog here - r-bloggers.com/rmongodb-r-driver-for-mongodb. It's a

little old but he seems to be keen for feedback so it could be worth dropping him an email (his address is a the bottom of the link)? AFAIK the driver is one of the community developed ones. – Mark Hillick May 29 '12 at 14:07

Thanks Mark, I did scan through the post, but just saw that it has quite an extensive example application linked to. Right, it's community developed. I eventually did try to contact Gerald, but also thought that putting this at SO would be a good idea. – Rappster May 29 '12 at 14:13

Cool, I'm having a look at it but I'm very new to R so I doubt I can help much. Let me know how you get on. – Mark Hillick May 29 '12 at 14:27

add a comment

3 Answers

Either c() or list() can be ok. Depends on whether the components are named and whether they all have the same type (for list). Best thing to do is look at the generated BSON and see if you are getting what you want. For the best control of the generated object use mongo.bson.buffer and the functions that operate on it. In fact this is why the sub-queries are failing. 'comments' is being created as a subobject rather than an array. mongo.bson.from.list() is handy but it doesn't give you the same control and sometimes it guesses wrong about what to generate from complicated structures.

The query on the other set of data can be corrected like so though:

```
buf <- mongo.bson.buffer.create()
mongo.bson.buffer.start.object(buf, "name.first")
mongo.bson.buffer.append(buf, "$in", c("Alex", "Horst"))
mongo.bson.buffer.finish.object(buf)
criteria <- mongo.bson.from.buffer(buf)
```

Note that you definitely need to use a buffer here since R will choke on the dotted name.

I hope this straightens out your problem. Let me know if you have any further questions.

answered Jun 9 '12 at 14:12

Gerald Lindsly
206 2 3

Great, thanks a lot Gerald! I'll check it out with my code tomorrow. – Rappster Jun 11 '12 at 15:39

I try to tackle one thing at a time, but still can't seem to figure out the correct BSON structure for the `posts` collection ;-) I added an "answer" where I tried to follow your suggestions to make it as easy as possible for you to have a hack at it. Thanks a lot for your support! – Rappster Jun 12 '12 at 10:20

Also: how do I know if I ended up with a sub-object or an array? The resulting BSON object in R looks quite similar in both situations. – Rappster Jun 12 '12 at 10:22

add a comment

I'm still not very clear on what's the preferred way here on SO to progress once a question has been posted but one wishes to elaborate a bit more, possibly adding further questions and answer approaches.

As I was often told not to blow up my original question with future edits, in this "answer" I'm simply taking the suggestions by Gerald Lindsly and try to put it into actual code (because it still didn't work out for me):

Preparations

```
pkg <- "rmongodb"
if (!require(pkg, character.only=TRUE)) {
    install.packages(pkg)
    require(pkg, character.only=TRUE)
}

# Connect to DB
db <- "test"
ns <- "posts"
mongo <- mongo.create(db=db)

# Make sure we start with an empty collection
mongo.drop(mongo, paste(db, ns, sep="."))
```

Insert document

As Gerald has pointed out in his answer, `mongo.bson.from.list()` sometimes makes wrong guesses about the resulting BSON structure, so I tried to go ahead an explicitly create BSON array objects:

```
buf <- mongo.bson.buffer.create()

# 'REGULAR' APPENDING
mongo.bson.buffer.append(buf, "_id", "abcd")
mongo.bson.buffer.append(buf, "when",
    mongo.timestamp.create(strptime("2011-09-19 02:00:00",
        "%Y-%m-%d %H:%M:%s"), increment=1))
mongo.bson.buffer.append(buf, "author", "alex")
mongo.bson.buffer.append(buf, "title", "Some title")
mongo.bson.buffer.append(buf, "text", "Some text.")
mongo.bson.buffer.append(buf, "tags", c("tag.1", "tag.2"))
mongo.bson.buffer.append(buf, "votes", 5)
# /

# VOTERS ARRAY
mongo.bson.buffer.start.array(buf, "voters")
```

```
mongo.bson.buffer.start.array(buf, "voters")
voters <- c("jane", "joe", "spencer", "phyllis", "li")
i=1
for (i in seq(along=voters)) {
    mongo.bson.buffer.append(buf, as.character(i), voters[i])
}
mongo.bson.buffer.finish.object(buf)
# /

# COMMENTS ARRAY
mongo.bson.buffer.start.array(buf, "comments")

mongo.bson.buffer.start.object(buf, "1")
mongo.bson.buffer.append(buf, "who", "jane")
mongo.bson.buffer.append(buf, "when",
    mongo.timestamp.create(strptime("2011-09-19 04:00:00",
            "%Y-%m-%d %H:%M:%s"), increment=1))
mongo.bson.buffer.append(buf, "comment", "some comment.")
mongo.bson.buffer.finish.object(buf)
```

Basic sub-level query

```
# Get all comments by 'meghan' from 'test.posts'

#--------------------
# Approach 1)
#--------------------
res <- mongo.find(mongo, "test.posts", query=list(comments=list(who="meghan")))
out <- NULL
while (mongo.cursor.next(res))
    out <- c(out, list(mongo.bson.to.list(mongo.cursor.value(res))))

> out
NULL
# Does not work

#--------------------
# Approach 2)
#--------------------
buf <- mongo.bson.buffer.create()
mongo.bson.buffer.start.object(buf, "comments")
mongo.bson.buffer.append(buf, "who", "meghan")
mongo.bson.buffer.finish.object(buf)
query <- mongo.bson.from.buffer(buf)
res <- mongo.find(mongo, "test.posts", query=query)
out <- NULL
while (mongo.cursor.next(res))
    out <- c(out, list(mongo.bson.to.list(mongo.cursor.value(res))))

> out
NULL
# Does not work
```

I still must be doing something wrong here when specifying the document ;-)

answered Jun 12 '12 at 10:18


Rappster
2,495 17 36

add a comment

Regarding atomic queries and the $in operator, I got Query 2 from your first question to work as follows:

```
buf <- mongo.bson.buffer.create()
mongo.bson.buffer.start.object(buf, "test")
mongo.bson.buffer.start.array(buf, "$in")
mongo.bson.buffer.append(buf, "a", "a")
mongo.bson.buffer.finish.object(buf)
mongo.bson.buffer.finish.object(buf)
criteria <- mongo.bson.from.buffer(buf)
criteria
```

I guess explicitly starting and ending the array does the trick, if the array is going to end up holding only one element.

One thing that can be useful is monitoring the mongod console or log (after starting mongod with the -v option). Running your old query, you'll see:

```
Tue Nov 20 16:09:04 [conn23] User Assertion: 12580:invalid query
Tue Nov 20 16:09:04 [conn23] assertion 12580 invalid query ns:test.users query:{ test
Tue Nov 20 16:09:04 [conn23] problem detected during query over test.users : { $err:
Tue Nov 20 16:09:04 [conn23] query test.users query: { test: { $in: "a" } } ntoreturn
```

Running the modified query, it looks ok:

```
Tue Nov 20 16:10:14 [conn23] query test.users query: { test: { $in: [ "a" ] } } ntore
```

answered Nov 21 '12 at 0:10


Dan Tenenbaum
384 3 13

That looks promising, thanks! Have been waiting for some input for quite some time now, I'll check this out tomorrow! – Rappster Nov 22 '12 at 10:01

add a comment

Not the answer you're looking for? Browse other questions tagged r mongodb rmongodb **or ask your own question.**