[SERVER-831] Positional Operator Matching Nested Arrays

Created: 25/Mar/10 Updated: 07/Dec/16

Status:	Open
Project:	Core Server
Component/s:	Querying, Write Ops
Affects Version/s:	None
Fix Version/s:	planned but not scheduled

Type:	New Feature	**Priority:**	Major - P3
Reporter:	Durran Jordan	**Assignee:**	Backlog - Query Team
Resolution:	Unresolved	**Votes:**	330
Labels:	query_triage		

Issue Links:	**Depends**			
	depends on	SERVER-27089	Extend the update subsystem to suppor...	Open
	depends on	~~SERVER-6399~~	Refactor update() code	Closed
	is depended on by	~~SERVER-828~~	Support for selecting array elements ...	Closed
	is depended on by	SERVER-3089	Ability to make use of a subdocument'...	Open
	Duplicate			
	is duplicated by	~~SERVER-9726~~	cant append for subdocument level 2	Closed
	is duplicated by	~~SERVER-14281~~	Positional Operator Points to Wrong I...	Closed
	is duplicated by	~~SERVER-11696~~	Updates not working at : Nested level...	Closed
	is duplicated by	~~SERVER-14164~~	MongoDB: Too many positional (i.e. '$...	Closed
	is duplicated by	~~SERVER-13929~~	Multi $(update) use or update in aggr...	Closed
	Related			
	is related to	SERVER-6864	positional operator projection inclus...	Open
Participants:	abdelouahab, Adam Reis, Alberto Lerner, Alexander, alex rockwell, Alex Young, Andreas B., Andrew Kalek, antoine, Anton Khodakivskiy, ashraf fayad, Asya Kamsky, Backlog - Query Team, Barry Dobson, basile, Binard, Bonozo Games, Brett, Bryan Migliorisi, Carl Banbury, Charlie Stigler, Chris Sedlmayr, Coen Hyde, Dani, Daniel Ciupe, Daniel Doubrovkine, Daniel Macias, Dan Pasette, Dave Geddes, David Creixell, david zhang, Dennis P, Devin Mooers, Durran Jordan, Eliot Horowitz, Ezekiel Victor, Frank Mayer, Freddy Löckli, Greg Braman, Guilherme Koehler, Guy Korland, Hans Petee, Hugh Watkins, I Need This Feature, Ioannis Chouklis, Ivan Fioravanti, Jake Clarkson, Jeff Whelpley, Joe, john.nishinaga@pat… , Jonathan, Jonathan Chambers, Joseph Blair, Joshua Austill, Joshua Chan, Justin Dearing, Keith Branton, KimShen, Kyle Richter, Manjunath Bhuyar, Marc MacLeod, Markus Graf, Markus Seeger, Matt Bodman, Megha Dev, Michael Fitchett, Michael Joyce, Michael Karpitsky, Miguel, Mikołaj Michalczyk, Mohammad Musa, Morten Herman Lan… , neeraj wadhwani, Neil Bartlett, Prashant Tiwari, Rainer Schreiber, Raul Guerrero, Ravindra M Rao, Remon van Vliet, Robert Weissmann, Roman, Rydal, Samuel Iten, Sam Weaver, Sander De Neve, Sebastian Lopez, Shane Hou, Shaun Berry, Snap Snerfing, Sparsh, Steve Babigian, Steve K, Sławomir Nowak, Tavacizade Torunoğlu, Timothy Pendergraft, Tolga Evcimen, Tony Ellard, Tony Mobily, Tyler Stroud, vijay kumar, vincent daubry, Vincent Huang, walay, wangdebing, Wen, Yannick Bétemps, yudho ahmad dipone… , Zoltan Altfatter			
Days since reply:	5 days ago			
Date of 1st Reply:	30/Mar/10 2:27 AM			
Last commenter:	Thomas Schubert			

Description

Issue Status as of Dec 06, 2016

We recognize that this feature is highly requested by the community; we are investigating implementation approaches and considering designs for the required query language extensions in SERVER-27089. We appreciate the challenge of maintaining applications with schemas including large arrays, and the work described in SERVER-27089 includes this feature request as well as other related improvements to array update capabilities. Please feel free to review SERVER-27089 for additional details around the requirements of this work and watch it for updates.

ORIGINAL DESCRIPTION

For cases where arrays can be nested within other arrays multiple levels deep, it would be great if the positional operator could match the appropriate array no matter how far deep the nesting goes. A syntax for this could be something like:

```
collection.update(
  { "_id" : 1, "medications._id" : 23, "medications.prescriptions._id" : 77 },
  { $set : { "medications.$0.prescriptions.$1.quantity" : 30 } },
  false,
  true
)
```

Here is a link to a gist with a sample document. In our current application we actually nest as many as 5 levels deep and some cases, and would like to be able to do atomic updates on fields at the very bottom of the tree for efficiency.

http://gist.github.com/342738

Comments

Comment by Coen Hyde [30/Mar/10]

I would like this feature. However is there a reason why we need the array sequence id? Why not just have 'medications.$.prescriptions.$.quantity' instead of 'medications.$0.prescriptions.$1.quantity'?

Comment by Durran Jordan [31/Mar/10]

Well if it's smart enough not to need them and just use $ that's even nicer.

Comment by Eliot Horowitz [13/Apr/10]

certainly can do it without the $0

Comment by antoine [13/Jul/10]

i really need to update nested array field ... !

Comment by Justin Dearing [17/Jan/11]

I just ran into this issue again. I can work around it for now since I can guarantee the position of the outer array.

If had a case where I could notavoid two round trips without this feature, the ability to lock a document would be useful.

Comment by wangdebing [31/Jan/11]

i really need to update nested array field ... !

Comment by Keith Branton [04/Feb/11]

While we clearly need a way to manipulate deeply nested objects atomically, the positional operator is very limited because it depends on the query providing its context - and can only be used once in an update. Not being aware of this jira I started another (http://jira.mongodb.org/browse/SERVER-2476), and now have a suggestion that could fix both problems.

If an array could be used for the field name in an update modifier, and bson queries were processed and iterated when encountered in the array, then we could use:

.update(
{ "_id":1}

, {$set:{["medications",{_id:23},"prescriptions",{_id:77},"quantity"]:30})

instead of the proposed

.update(
{ "_id":1, "medications._id":23, "medications.prescriptions._id":77 }

,
{$set:{medications.$0.prescriptions.$1.quantity" : 30 }})

and we could even do (using the gist examples)

.update(
{ "_id":1}

, {$set:{["medications",{_id:23},"prescriptions",{_id:77},"quantity"]:30,
["medications",{_id:23},"prescriptions",{_id:13},"quantity"]:6},
$push:{["medications",{_id:41},"prescriptions"],{_id:24,quantity:4,started:2009-03-01}}})

in a single operation, in addition to

.update(
{ "_id":1}

, {$set:{["medications",{_id:23},"prescriptions",{},"quantity"]:0}})

which would set quantity to 0 for all prescriptions for medication 23 and

.update(

{ "_id":1}

, {$set:{["medications",{_id:{$lte:7},"prescriptions",{_id:{$gt:60}},"quantity"]:0}})

which would set quantity to 0 where medication_id<=7 and prescription_id>60.

It would also be really useful if this approach could be used in queries too. Imagine being able to do this

.find({["medications",{_id:23},"prescriptions",{},"quantity"]:{$gt:120})

to find all patients who have been prescribed more than 120 for medication 23

I wonder too if the {} could be inferred - if you are using an array for a field name then all elements of arrays are automatically considered in the absence of a query

Am I onto something here or just out in left field?

Comment by Durran Jordan [04/Feb/11]

I like that idea since it's a bit more flexible in that multiple atomic updates could happen at different points in the hierarchy in a single call, instead of having to do multiples... The proposed syntax feels a little odd though to me.

Comment by wangdebing [07/Mar/11]

i really need to update nested array field ... !

Comment by Mohammad Musa [15/Apr/11]

We really need this to be implemented in MongoDB sooner than later. This seems natural to be part of MongoDB.

Without this we just can't implement voting functionality in our comments system.

So we are disabling our rating functionality and wait for 831 to be included in v1.9

This is a BIIIIIIIIIIIIG up vote for this jira.

Thanks
Musa

Comment by alex rockwell [11/May/11]

hope this one didn't fall off the radar

Comment by Frank Mayer [19/May/11]

I am stuck on the same issue. Has this fallen off the radar? I can't see it anywhere for version 1.9.1 or 1.9.2.
Thank you

Comment by Eliot Horowitz [19/May/11]

1.9 planning is things that might make it into 1.9
If not, they'll be on the short list for 2.1

Comment by Frank Mayer [19/May/11]

So, that leaves the question, when do you plan to release 2.1? (Just joking, but this would really be a big help in my current development. Too bad it's not supported yet.)

Comment by Remon van Vliet [20/May/11]

I hope this won't be bumped to 2.1+ given the fact that it basically means we cannot atomically update any field in any given document which should be possible. That said this shouldn't be rush implemented by simply adding allowing multiple ordered positional operators. I agree with Keith's sentiments that the positional operator is pretty limited/clumsy. In an update call the "criteria" parameter should only be responsible for selecting the document(s) the update has to be applied on. Deciding on which element of which (nested) array should be updated really belongs in the "update" parameter as Keith suggests.

Comment by Dave Geddes [27/Jun/11]

This is a show stopper. Mongo really needs this feature. Any update?

Comment by Marc MacLeod [03/Aug/11]

Huge +1 from me as well. My example case is a state->city->school nested structure and I would like to be able to update the school data with an update.

Comment by Rydal [16/Sep/11]

Definitely a huge 1 for me as well, I'm just updating the full document now and I have a todo assign to about 20 classes that need it. Anxiously waiting for 2.1!

Comment by Bonozo Games [29/Dec/11]

I am new to mongoDB and ran into this issue very early on. I naturally assumed I did not know how to write the update query. Imagine my surprise when I was informed, in the forums, it is an actual limitation of mongoDB! +1

Comment by Neil Bartlett [08/Jan/12]

Major +1. New user but hit this very early on. First major headache with Mongo though. Apart for this it has performed very well on my db which currently has approx 250k records in it.

Comment by Dave Geddes [13/Feb/12]

Is this still planned for 2.1? Need this feature badly.

Comment by Bryan Migliorisi [09/Mar/12]

Add me to the list of people who really need this feature, and with an estimate of less than a week, why not just get it done? 🙂

Comment by Michael Karpitsky [14/Mar/12]

I also need this feature

Comment by Daniel Ciupe [31/Mar/12]

I find the presence of the word 'desired' next to 2.1 a bit discouraging. This feature is an absolute must, and it has already been dragging beyond belief.

Comment by Frank Mayer [03/Apr/12]

Hi, there dear mongodevs!!! Any news on this issue?

Comment by Matt Bodman [23/Apr/12]

I also appeal to the most awesome mongodevs to implement this feature. Embedding is sure one of the coolest features on offer, but without a way to efficiently handle updates of deeply embedded documents, its perhaps more painful than its worth (at this stage). Keep up the great work!

Comment by Tyler Stroud [20/Jun/12]

+1 for this, I also ran into this exact issue--needing to use the positional op to access nested arrays. I'm glad to see it's been scheduled for 2.3 🙂

Comment by john.nishinaga@patdeegan.com [02/Aug/12]

+1 for this. This is a big deal if you're going to use MongoDB.

Comment by Sebastian Lopez [13/Sep/12]

+1 for this issue being addressed. Running into a use case where I need to update atomically a document in this fashion.

Comment by Steve Babigian [27/Sep/12]

+1 When people run into this issue (StackOverflow, Google Groups), I keep hearing people suggest that one reworks his schema to flatten it out or add another collection. To me, this goes against the core of what MongoDB and NoSQL data structures are all about! Otherwise, I might as well go back to SQL with 20 tables and a ton of JOINS. This is CORE functionality, please get it in there soon!

Comment by Michael Fitchett [27/Sep/12]

This ticket was created in 2010, it's almost 2013. I don't see this happening. It's obvious that people have a need for this functionalty. The reason it probably has not been implemented is do to some limitation with how MongoDB was originally built. For instance it's possible that a single document could become greater than the default 16MB file split which may cause an issue when trying to search multiple levels deep.

I decides a long time ago to just seperate out the data into multiple collections. MongoDB for my use has proven to be a better alternative to structured tables. Even without this functionalty. I just program around it for now.

I would however like to see this get added.

Comment by Steve K [10/Oct/12]

There is a bigger issue going on here than just adding support for multiple positional operators. I believe this ticket and other tickets exist for the most part due to a major lack of index support elsewhere.

Please see and VOTE for https://jira.mongodb.org/browse/SERVER-7313

Durran (original author of the ticket) appears to be like many of the users forced to setup his document using flat regular arrays to gain index performance. In his document, he could easily switch his flat medications/prescriptions arrays to associative arrays by using "id" field as the key and use the following command instead...

collection.update(
{...}

, {$set:
{"medications.23.prescriptions.77.quantity":30}

}, false, true);

Now the reason why he did not do that (just like many other users) is that it is not possible to index at nested levels if the data is setup as associative arrays instead of regular arrays.

I have run into the same users as Steve B has on the internet. I am actually one of those users who was forced to rework my schema in some cases while in other cases resort to a second synced flat structure for query purposes. I believe we should stop asking users to rework their schema and use the gawky positional operator. The better and more elegant solution is to add index support at the associative nested array levels and let users keep their object-oriented schema.

~~SERVER-7313~~ ticket has a link to a nice article about arrays and atomic issues. The article proposes associative arrays to workaround the atomic issues but unfortunately it still does not address the lack of index support.

Comment by Daniel Doubrovkine [12/Nov/12]

Wanted to pitch in. You hear rumors of MongoDB trashing data, which are, of course, untrue. I think this is the cause of a large % of those, because it's trashing data and most developers probably don't know it.

Here's an example with mongoid.

```
—
require "spec_helper"

class Sandwich
include Mongoid::Document
embeds_many :slices
end

class Slice
include Mongoid::Document
field :thickness
embedded_in :sandwich
end

describe "Concurrent updates" do

it "work" do
sandwich = Sandwich.create!
slice1 = Slice.create!(
{ sandwich: sandwich, thickness: 1 }

)
slice2 = Slice.create!(
{ sandwich: sandwich, thickness: 2 }

)

sandwich.slices.count.should == 2
sandwich.slices.first.thickness.should == 1
sandwich.slices.last.thickness.should == 2

Mongoid.default_session["sandwiches"].where(
{ _id: sandwich.id }

).update({
"$pull" => { "slices" => { _id: slice1.id }}
})
Mongoid.default_session["sandwiches"].find.first.should == {
"_id" => sandwich.id,
"slices" => [
{ "_id" => slice2.id, "thickness" => slice2.thickness }]
}

# => we don't expect mongoid to notice the concurrent removal
sandwich.slices.count.should == 2

slice2.update_attributes!({ thickness: 3 })

# => we do expect mongoid to update the correct record
Mongoid.default_session["sandwiches"].find.first.should == {
"_id" => sandwich.id,
"slices" => [{ "_id" => slice2.id, "thickness" => slice2.thickness }

]
}

end

end
—
```

What does this do?

```
—
1) Concurrent updates works
Failure/Error: }
expected: {"_id"=>"509fd8d13b5552dde5000001", "slices"=>[
{"_id"=>"509fd8d13b5552dde5000003", "thickness"=>3}]}
got: {"_id"=>"509fd8d13b5552dde5000001", "slices"=>[{"_id"=>"509fd8d13b5552dde5000003", "thickness"=>2}, {"thickness"=>3}]} (using ==)
Diff:
@@ -1,3 +1,3 @@
```

"_id" => "509fd8d13b5552dde5000001",
-"slices" => [{"_id"=>"509fd8d13b5552dde5000003", "thickness"=>3}

]
+"slices" => [
{"_id"=>"509fd8d13b5552dde5000003", "thickness"=>2}

,
{"thickness"=>3}

]
—

Ouch. This case can be solved at 1-level deep, with a lot of work for the ODM developer. But not nested, which is very frequent because of this bug. So MongoDB has different consistency guarantees for top-level documents, 1-level embedded documents and N-level embedded documents. It's bad.

More details at https://github.com/mongoid/mongoid/issues/2545

Comment by Jonathan Chambers [17/Jan/13]

This issue will be 3 years old this March, despite there being considerable demand for this feature. Could someone from 10gen at least comment on WHY it keeps getting postponed.

Comment by Alberto Lerner (Inactive) [17/Jan/13]

This is a feature we're certainly going to be tackling. Right now, there is a large reorganization effort involving the code that this feature sits atop. The effort is in an advanced stage, even if the resulting code only started to make it to the code base. Giving a precise time now is risky, but we're optimistic that most, if not all, what's necessary would be in place in the development cycle starting after 2.4 will be released (so, potentially in the 2.6 product).

Comment by Jonathan Chambers [17/Jan/13]

Thanks for the quick feedback Alberto. Good to hear you're making progress. Keeping my fingers crossed for 2.6...

Comment by Alex Young [01/Apr/13]

I strongly agree with Keith Branton and really like the way he suggested.

Like if a user has a book list which has 10 books to read, and now we want to change the priority, I have to do 10 times update with the position operator, it's awful especially when the list is in a multilevel subdocument.

data:

{
"_id": ObjectId("4fa43f4d1cf26a6a8952adf1"),
"userId": 1,
"facebookId": 1234,
"booksToRead": [
{
"_id": 1,
"name": "The Theory of Relativity",
"Order" : 0,
"referrer" : [
{ "userId": 2, name :"Tony", type : "general friend" }

,
{ "userId": 3, name : "Jimmy" type : "general friend" }

,
{ "userId": 4, name : "John" type : "close friend" }

]
},
{ "_id": 2, "name": Bibel", "Order" : 1, "referrer" : ["Alex","Sam","Luke"] }

,
{ "_id": 3, "name": "Pi and Richard Parker", "Order" : 2, "referrer" : ["An Li","Bill","Steve"] }

]
}]
}

It will be greate if we can locate each operation like $set, $push, $pull,
I really like Keith Branton's way, but I also suggest some change, just like:

db.user.update(
{"_id" : ObjectId("4fa43f4d1cf26a6a8952adf1"), "facebookId": 1234}

,
{$set:

[{
"booksToRead.${_id:1}.Order" : 1
},{
"booksToRead.$
{name:"\"Pi\" and Richard Parker"}

.Order" : 2, multi: true
},{
"booksToRead.$
{referrer:"Bill"}

.Order" : 0, multi: true
}]
}
)

Also, it's easy to modify array in deeplevel subdocument.

db.user.update(
{"_id" : ObjectId("4fa43f4d1cf26a6a8952adf1")}

,
{$set:[{
"booksToRead.${_id:1}.referrer.$
{\"userId\": 3}

.type : "general friend"
}]
}
)

Comment by Alex Young [02/Apr/13]

BTW, can we define a way to return some data after some operation like push.

db.user.update(
{"_id" : ObjectId("4fa43f4d1cf26a6a8952adf1")}

,
{$push:[{
"booksToRead.${_id:1}.referrer : {{
name : "Sam"
type : "close friend"
},
{ $return : id }

]
}
)

which gonna return a data like this:
{"_id" : ObjectId("4fa43f4d1cf26a6a8952adf1")}

,
{booksToRead:[{
"booksToRead.${_id:1}.referrer : {{
name : "Sam"
type : "close friend"
},
{ $return : id }

]
}

{
"booksToRead": [
{
"referrer" : [
{ "id": 5, name :"Tony", type : "general friend" }

]

Comment by Guy Korland [10/Apr/13]

Does the fact that the issue marked as "Fix Version/s:2.5.w" means it is already included in next 2.5 release?

Comment by Dan Pasette [10/Apr/13]

Guy Korland, that means it is a candidate feature for the 2.5 series. If the fixVersions is set to 2.5.0, 2.5.1, etc., it is slotted for a development release.

Comment by Barry Dobson [01/May/13]

I know this request is 3 years old, but without this feature it's a show stopper

Comment by Alberto Lerner (Inactive) [01/May/13]

Barry,

Noted. We are working on the supporting machinery necessary for this feature quite actively.

Alberto.

Comment by abdelouahab [18/May/13]

+1, because i thought it was my problem:
It seems that i can go further than one subdocument if i want to add it dynamicaly, here is the code:
db.users.update(
{"pup.cmn.id":id}

, {"$addToSet":{"pup.cmn":
{"abus":email}

}})
this give error:
OperationFailure: can't append to array using string field name: cmn
then, if i add positional element i get this:
db.users.update(
{"pup.cmn.id":id}

, {"$addToSet":{"pup.$.cmn":
{"abus":email}

}})
"cmn" :
[
{ "fto" : ObjectId("5190e8a53a5f3a0c102af045") "id" : "14.05.2013 12:29:53" }

,
{ "abus" : "u...@example.com" }

]
so as you can see, it will add it in the same level, and i dont want that, because the application will get errors.

Comment by Ezekiel Victor [07/Jun/13]

This is a huge issue. Even simple structures with minimal nesting are not feasible for production use because non-atomic find + update by array index is unreliable. +100 votes from me. My schema looks like classical normalized DB almost because of this limitation....

Comment by Joe [11/Aug/13]

Honestly, it is shameful that this feature has not been implemented yet. 10000 votes from me, I'm sure everyone is in favor of this.

Comment by Alberto Lerner (Inactive) [14/Aug/13]

Joe, Ezekiel,

We've just "swapped" the new update framework that this feature depends on today.

https://github.com/mongodb/mongo/commit/dda99507af1eef8368d5d26a226d4d94f50d5a30

In practice, this means that we have much better leeway now to improve how updates work. And this ticket is on the top of our list of improvements.

We've been considering a few alternatives for how to express such updates, from the language perspective. Might you want to add your uses cases to this ticket? Finding a simple, unambiguous language construct to express how to match nested arrays is harder than one may think. So if we have more use cases example that are important, that would help.

Comment by Joe [14/Aug/13]

Hey Alberto, thats awesome news. I posted a question on S.O. that explains my particular situation:

http://stackoverflow.com/questions/18173482/mongodb-update-deeply-nested-subdocument

If I have a schema such as this:

```
{id: 1,
 forecasts: [ {
             forecast_id: 123,
             name: "Forecast 1",
             levels: [
                { level: "proven",
                  configs: [
                            {
                              config: "Custom 1",
                              variables: [{ x: 1, y:2, z:3}]
                            },
                            {
                              config: "Custom 2",
                              variables: [{ x: 10, y:20, z:30}]
                            },
                    ]
                },
                { level: "likely",
                  configs: [
                            {
```

I am trying to update a particular config like this:

```
db.myCollection.update({"id": 1,
                        "forecasts.forecast-id": 123,
                        "forecasts.levels.level": "proven",
                        "forecasts.levels.configs.config": "Custom 1"
                         },
                         {"$set": {"forecasts.$.levels.$.configs.$": newData}}
                       )
```

What is the time frame for this getting into the latest incantation of mongo?

Comment by Daniel Doubrovkine [14/Aug/13]

Alberto, here's a blog post that's relevant for what you're asking: http://artsy.github.io/blog/2013/02/09/data-corruption-and-concurrent-updates-to-embedded-objects-with-mongoid/

Syntax-wise 99% of my scenarios find a document, then match it, deep down the hierarchy. So the current syntax of multiple positional operators is very difficult to work with - you have to remember all positions of parents. I think you need something that can match one or more EMBEDDED documents, then update those, NOT update starting from the parent.

Comment by Alberto Lerner (Inactive) [14/Aug/13]

For a query whose expression is the same as the $-nested expression, that works.

What would you expect to happen if the query was, say, a complex $or?

Comment by Daniel Doubrovkine [14/Aug/13]

I think there should be a clear understanding of what you're matching. Today it's very clearly the top level document. But if there was a way to clearly match an embedded document or a result set of top level or embedded documents at any one level, the whole $ mess could just go away.

I think what I am trying to say is that $set takes currently a top level path, but it could become something relative to the matched document.

db.coll.find(whatever, $set :
{ '@.x' => 'y' }

) where @ is the matched document(s).

Comment by Joe [14/Aug/13]

What are the proposed constructs / syntaxes for doing a nested update?

Comment by Alberto Lerner (Inactive) [14/Aug/13]

@Daniel, we'll continue matching from root. That's not the issue. The issue is that for some queries (that don't traverse arrays, or traverse them in alternative ways, as in a $or) there is not a clear answer of what semantics to use.

@Joe, There's one earlier in this thread.

Comment by Ezekiel Victor [15/Aug/13]

I think this would be unambiguous:

```
/*
Some contrived example:

// PetOwner collection
[
  {
    _id: ...,
    name: 'Bob',
    pets: [
      {
        name: 'Fluffy',
        animalType: 'cat',
        toys: ['string', 'wand']
      },
      {
        name: 'Mittens',
        animalType: 'cat',
        toys: ['mouse']
      }
    ]
```

Comment by Ezekiel Victor [15/Aug/13]

Sorry, that last example is not special; I think you can already do that anyway. Can you show us some ambiguities?

Comment by Coen Hyde [15/Aug/13]

I agree with @Ezekiel. Keep the format familiar. At the moment I think it is expected that "pets.$.toys.$" syntax should work regardless of any other possible syntax implementations.

Comment by Tony Mobily [27/Feb/14]

I haven't seen any activity in this for quite a while. This effectively means that deep updates are... well, not possible.
I see that it's now marked for 2.7. Can I make a vote, and ask to push it to 2.5 for 2.6?

I think this feature missing is *tragic*.

Comment by David Creixell [30/Mar/14]

I agree with @Tony. We are missing this feature since 2010, and is so critical. I can't believe that this is still unresolved.

Comment by KimShen [08/Apr/14]

Mark and follow, similar requirement like above

Comment by Roman [15/Apr/14]

we are very waiting for this feature!

Comment by Devin Mooers [16/Apr/14]

A big +1 for this issue. The workarounds for modifying many-levels-deep documents are not pretty or fast!

Comment by Carl Banbury [17/Apr/14]

Completely agree with Devin. We are basically having to avoid arrays in our documents because of this and other basic functionality missing for arrays.

Comment by Devin Mooers [17/Apr/14]

Yup. I'm now working around it by moving a major sub-collection to its own collection, so I now (luckily) only have to traverse one array level with the positional operator instead of two. Ironically this separate collection works out being a little more flexible for the app I'm working on, but having multiple positional operators still seems like a really killer feature that MongoDB deserves!

Comment by Jake Clarkson [23/Apr/14]

Agree with @Carl Banbury on this one, often this issue means our document readability suffers due to not being able to update the nested arrays.

Comment by Joshua Chan [27/May/14]

Another +1 for this issue as it allows us to keep things tidy (we don't have too many levels of nested arrays) in one document.

Comment by Mikołaj Michalczyk [01/Jun/14]

+1

Comment by Jeff Whelpley [04/Jun/14]

Huge +1 for me as well. This is a must have.

Comment by Yannick Bétemps [10/Jun/14]

big +1 for me too : this allows to make complex documents easier to maintain, this is also very useful with these document databases where you can not join tables : redundancy sometimes replaces the joints, so this feature makes it easier to maintain redundant data (ex. city / country names embedded in data etc)

Comment by walay [14/Jun/14]

+1 for me!
The feature of "medications.$.prescriptions.$.quantity" is a must have. Otherwise we will have to abandon using MongoDB!

Comment by yudho ahmad diponegoro [17/Jun/14]

+1 from me. This is what makes mongoDB mongoDB

Comment by neeraj wadhwani [17/Jun/14]

+1 ... Please make this happen.

Comment by basile [01/Jul/14]

+1 for me

Comment by Manjunath Bhuyar [08/Jul/14]

We also need this feature, currently its difficult to edit the production DB

Comment by Wen [16/Jul/14]

I need this feature, else I still have to use my SQL db.

Comment by Tavacizade Torunoğlu [18/Aug/14]

It's been 4 years and versions but still there is no improvement about this..

Comment by Raul Guerrero [26/Aug/14]

This would be great for our application so +1

Right now we have an N level binary tree that we store in MongoDB, and to be able to do queries on it, we have to convert and store our tree into a flat list defining each node's parent relationship and order to be able to use the aggregation framework, so we can convert the list back to a tree when we pull it from MongoDB for rendering in our UI.

We don't have that much overhead because we use Java and through HashMaps we can, very quickly and without processing overhead, convert the list to tree and viceversa, but if we want to access the tree from other languanges that don't have the same facilities as Java (like PHP), then we get major overheads doing that.

So just having the ability to store the tree as is and query it without having to turn it into a list, would be just awesome.

Comment by ashraf fayad [27/Aug/14]

+1 we need this feature too

Comment by vincent daubry [06/Sep/14]

+1 , as a newcomer to mongodb i almost immediately ran into this issue. This is very deceptive for people evaluating mongo for production use....

Comment by Markus Graf [28/Sep/14]

+1, I would also appreciate this. All users of Mongoid will get a problem with the positional operator using mongodb 2.6 (https://github.com/mongoid/mongoid/issues/3611). Breaking backward compatibility is jus not acceptable and this feature could solve this problem. After all I don't be able to use mongodb from 2.6 so if this does not change I have to switch database.

Comment by Markus Seeger [29/Sep/14]

+1 This is a common use case for our applications, and we consider it not being there a near show-stopper.

Comment by Markus Graf [29/Sep/14]

It turned out that I could use my application unter 2.6 with designing my database better. Now I don't have any array nesting deeper than 2 levels, but still it feels just not right and at some point the design could need a deeper nesting.

Comment by I Need This Feature [02/Oct/14]

I need this feature so much I created an account here just to let you know.

Comment by Greg Braman [16/Oct/14]

+1 on this too. Necessary for one of my company's use cases.

Comment by Tony Ellard [09/Nov/14]

+1 this as well. This would be a HUGE help. Thanks!

Comment by Kyle Richter [10/Nov/14]

+1 We could really use this.

Comment by Snap Snerfing [10/Nov/14]

+1 here, too.

Comment by Daniel Macias [10/Nov/14]

+1 Very useful.

Comment by Charlie Stigler [13/Nov/14]

Questions for any MongoDB folks reading this:

-Now that 2.8rc0 is out, does that mean this feature will officially not make 2.8?
-If that's the case, do things like this get added in minor releases (2.8.1)? Or does that mean this will not come out until at least 2.10?

Hoping this doesn't mean another year of waiting.

(sorry to add to the email spam of +1s, but I thought this was worth asking.))

Comment by Sam Weaver [16/Nov/14]

Hello Charlie.

Whilst we were keen to get this feature into 2.8, it unfortunately didn't make the cut. We understand there are a lot of votes and watchers on this ticket and its very highly desired. Rest assured we are addressing the state of this ticket, but in the mean time I have moved this to 2.9 desired.

Thank you for your understanding.

Kind Regards,
Sam

Comment by Shaun Berry [21/Nov/14]

+1 really could use this ASAP

Comment by Chris Sedlmayr [23/Jan/15]

Is there any movement on this at all since November?

Comment by Hugh Watkins [23/Jan/15]

This request is 5 years old, I seriously doubt it will ever get done.

Comment by Shane Hou [04/Feb/15]

So upon the release of 3.0, this feature turned into 3.1 desired... Was this decision based on some data statistics that few people ever need it or was it because of the difficulty to implement?

Comment by Joseph Blair [11/Mar/15]

+1. Isn't one of the main advantages of using Mongo supposed to be the support for nested arrays?

Comment by Andrew Kalek [13/Mar/15]

+1

Comment by Zoltan Altfatter [15/Mar/15]

+1

Comment by Vincent Huang [26/Mar/15]

+1

Comment by Andreas B. [31/Mar/15]

I almost feel bad for forgetting this issue's **5th** birthday. Happy belated birthday!

Comment by Megha Dev [22/May/15]

Are you guys planning to add support for this anytime soon?? It just seems SO essential for deeply nested documents.

Comment by Sławomir Nowak [05/Jun/15]

2015, and it still causes problems. I had to redesign my collection to make it works. +1, please fix it.

Comment by Ravindra M Rao [11/Jun/15]

+1

Comment by Prashant Tiwari [24/Jun/15]

Please, please, please fix this. We have only scant updates from project owners/maintainers throughout the history of this 5 year-old issue, even though one would think the number of +1s on this – and its importance as being a very crucial reason for using MongoDB – alone should warrant a P1 or at least a P2 level response.

Dear devs, please assure us that we can trust you that you indeed desire to fix this in 3.1 at last.

Comment by Michael Joyce [07/Jul/15]

+1 This is a must

Comment by Dennis P [17/Aug/15]

+1... also need this

Comment by Binard [17/Aug/15]

+1, need !

Comment by Rainer Schreiber [14/Sep/15]

+100, need this asap!

Comment by Ioannis Chouklis [14/Sep/15]

+1

Comment by Tolga Evcimen [02/Oct/15]

It is version 3.0 now, and this feature still lacks? I convinced my team to convert to mongodb. We have redesigned our schemes. We have started to development process with find queries, now we are at updates, and stuck 🙂 I felt like it's a shame. Now we will redesign our schemas, and queries too.

Comment by david zhang [16/Nov/15]
+1 need please finish this feature!!

Comment by david zhang [16/Nov/15]

no need too far ,two will be enough

Comment by Sander De Neve [16/Nov/15]

+1

Comment by Morten Herman Langkjaer [10/Dec/15]

Has come upon this problem on version 3.0
we nest our solution in 3 levels, and several threads are updating the model with atomic seperate operations, so when this is not implemented, it introduces race conditions, where state can be overridden because it was updated after one thread receives the object from mongo untill it is updated in mongodb
+1 to fix this

Comment by Miguel [05/Jan/16]

+1, I nested 3 levels thinking I won't need to update, suddenly I need to do it in the deepest branch and using a double for to go into the collection and then doing a complete update works but is not a nice and clean option, though.

Comment by Anton Khodakivskiy [10/Mar/16]

I moved away from MongoDB long time ago, but I keep the subscription to this ticket - 6 years and going strong!

Comment by Tony Mobily [10/Mar/16]

I haven't moved away but I too keep on watching the 5 or 6 features that are *amusingly* missing at this point; I am using MongoDb, but only with 1-depth tables which is kind of ridiculous.
Alberto Lerner, who has since left MongoDb, wrote in 2013:

> This is a feature we're certainly going to be tackling. Right now, there is a large reorganization effort involving the code that this feature sits atop. The effort is in an advanced stage, even if the resulting code only started to make it to the code base. Giving a precise time now is risky, but we're optimistic that most, if not all, what's necessary would be in place in the development cycle starting after 2.4 will be released (so, potentially in the 2.6 product).

Then Sam Weaver in 2014:

> Whilst we were keen to get this feature into 2.8, it unfortunately didn't make the cut. We understand there are a lot of votes and watchers on this ticket and its very highly desired. Rest assured we are addressing the state of this ticket, but in the mean time I have moved this to 2.9 desired. Thank you for your understanding.

I think this is getting silly, but I start to wonder about the state of MongoDB development. Really.

Comment by Ioannis Chouklis [13/Mar/16]

Well, I am still using MongoDB for a weekend project and I had this requirement. This issue just made me to take a step back and re-think my data model. I restructured it, and it worked. No big deal for a weekend project, but it could result to a headache for a company.

Comment by Joshua Austill [25/Apr/16]

I can't believe this is a thing. Has there been any updates since 2014 from the MongoDB team?

Comment by vijay kumar [13/Jun/16]

it is much needed feature and should be implemented as soon as possible.

Comment by Adam Reis [07/Aug/16]

No word since 2014 then? This issue, along with https://jira.mongodb.org/browse/SERVER-1243 are really preventing efficient data structures and update queries...

> I think this is getting silly, but I start to wonder about the state of MongoDB development. Really.

Agreed. I am now strongly considering to try a different NoSQL DB for my next project.
Commonly requested feature issues like these... open since 2010, that's getting a bit ridiculous.

Comment by Guilherme Koehler [11/Aug/16]

I'll have to add +1 to this.

It's nasty that this is not available.

As stated by others, we know things can be moved to different collections, but atomic updates are no longer possible.

Plus I don't think a DB limitation should be a valid reason to move things to different collections. If updating a document with two levels of arrays atomically is not possible in a clean and elegant way then what's the point ?

Is anyone on the team still looking at this ? Could you please give us some info ?

Comment by Jonathan [25/Aug/16]

> Commonly requested feature issues like these... open since 2010, that's getting a bit ridiculous.

No kidding. 6 years? Really Mongo?

Comment by Dani [06/Sep/16]

I got bitten by this sneaky limitation too. So much for atomic updates...

And it has priority "Major" WTF?????
Comment by Alexander [21/Sep/16]
I very want this feature... how soon it will be done?
Comment by Hans Petee [20/Oct/16]
We got stuck several times on this. It would be great to have some updates. If this is not fixable or needs a lot more time, please let us know, so that we can switch to another technology.
Comment by Ivan Fioravanti [25/Oct/16]
Any news on this one? Will this be part of 3.4 release?
Comment by Brett [27/Oct/16]
ignoring the community very disappointing mongodb!
Comment by Timothy Pendergraft [28/Oct/16]
I too would like to see this happen.
Comment by Sparsh [28/Oct/16]
I gave up hope somewhere around late 2014, the fact this is MAJOR and due for past 6 years. Whatever happens to low priorities
Comment by Robert Weissmann [01/Nov/16]
This would be a greate improvement and would feel natural and expected.
Comment by Samuel Iten [02/Nov/16]
Hello issue SERVER-831, my old friend. Would be nice to have a status update from the MongoDB Team.
Comment by Freddy Löckli [02/Nov/16]
please, I need this feature!
Comment by Asya Kamsky [06/Dec/16]
Hi All, We recognize that this feature is highly requested by the community; we are investigating implementation approaches and considering designs for the required query language extensions in SERVER-27089. We appreciate the challenge of maintaining applications with schemas including large arrays, and the work described in SERVER-27089 includes this feature request as well as other related improvements to array update capabilities. Please feel free to review SERVER-27089 for additional details around the requirements of this work and watch it for updates. Asya Kamsky Lead Product Manager MongoDB Server

Generated at Mon Dec 12 22:39:09 UTC 2016 using JIRA 6.4.14#64029-sha1:ae256fe0fbb912241490ff1cecfb323ea0905ca5.