MongoDB Joins with MongooseJS

an AngularJS and Rails 4.1 Project The Weird Impact of Expertise o

Posted on 12 May
by J Cole Morrison

If you're going to have a cow about doing something relational in NoSQL, go have a cow about anything ever that was used for something other than what it was originally intended.

So you're using MongoDB and Node. Let's pretend you have two collections of documents, "Users" and "Comments." There are two ways to "relate" this data. More info at this link:

Embedded Data Models, which would be like so:

```
> db.users.findOne()

{
    "_id": ObjectId("randomstringofcharacters"),
    "firstname": "Some",
    "lastname": "Lastname",
    "username": "Pipituu",
    "comments": [
        {
            body: "Some comment",
            date: "some date"
        },
        {
            body: "Another comment",
            date: "some date"
        }
    ],
    "password": "abc123"
}
```

an AngularJS and Rails 4.1 Project The Weird Impact of Expertise o

So, these allow you to make one query and get everything vs. multiple ones. The problem with this model from a Mongo stand point is only that if it grows to large, you'll have to allocate more memory for it. However, for those who want to use MongoDB for relations, doing it like this is just a headache. If you want to "relate" the comments to another user, or to other comments, the best bet is to do it in a normalized style.

So we'd do it like so:

A User:

```
{
    "_id": ObjectId("randomstringofcharacters"),
    "firstname": "Some",
    "lastname": "Lastname",
    "username": "Pipituu",
    "comments": [
        ObjectId("idofsomecomment"),
        ObjectId("idofanothercomment")
    ],
    "password": "abc123"
}
```

A Comment:

```
{
    "_id": ObjectId("idofsomecomment"),
    "body": "Some comment",
    "parent": ObjectId("idofthiscommentsparent"),
    "moreMetaData": "more meta data"
}
```

Okay, enough of a primer, let's figure out how we'd join these together with Mongoose. And yes, you'll want to use something like Mongoose.js to simulate joins "easily." Otherwise, you'll need to query the User and then, in your programming language, use the Comment Ids to grab the related comments (which is all Mongoose does in the background).

Mongoose.js's Populate

This is the function one uses to populate documents. Let's pretend the data above is in our MongoDB. To get it join a User with all their appropriate Comment's together you'd just do the following:

This is assuming you've created a mongoose schema for Users and Comments.

```
// Find the user doc
Users.findById('randomstringofcharacters')

// This is just chaining with mongoose
// Here, we select the 'comments' field
// to be populated
    .populate('comments')

// finally, you just run it.
    .exec(function (err, client) {
        if (err) {
            console.log(err)
        }

        // This will now have comments embedded!
        return client;
    });
```

The client that you return will now have the comments embedded. A gotcha that you have to remember is that, every time you create a comment, you need to insert it's ID into the relevant user's comments array.

I'll be covering this more in-depth in the series of articles I put out on building apps with angular and express. By in-depth I mean, doing a step by step process of how it's done, instead of just a quick summary.

About the author

J Cole Morrison

Sacramento, CA

http://jcolemorrison.com

Comments

7 Comments **start.jcolemorrison.com** 1 **Login**

Recommend **Share** **Sort by Best**

Join the discussion…

Tony Gutierrez • 7 months ago

Not really a join as it requires dual tracking.

• Reply • Share ›

Alex Mills • 10 months ago

Nice article - would be good if you could explain how populate works - is it implemented in C and does it run on the MongoDB database server or somewhere in Node.js-land?

• Reply • Share ›

Stephen Bussey • 2 years ago

What if you were "joining" several thousand, tens, hundreds of thousands of records. This would pretty much explode with cursor unrolling, wouldn't it?

• Reply • Share ›

Cole Morrison Mod Stephen Bussey • 2 years ago

To be honest, I've yet to encounter a problem where I was joining two collections that required in the hundreds of thousands of numbers? I have reached in the thousands of documents to 1 document and it hasn't been a problem.

• Reply • Share ›

Stephen Bussey Cole Morrison • 2 years ago

Okay, I hope you never run into it...it's not very fun and usually leads to questioning of sanity for choosing the design

1 • Reply • Share ›

Steven Cooper • 2 years ago

Nice article but "joins" should never be used with Mongo cause it gives SQL'ers the wrong idea

• Reply • Share ›

Cole Morrison Mod Steven Cooper • 2 years ago

Thanks. Do you mean the word, or in general? If you mean the "word" then, it's used because it's an easy analogy into what it is that's being done without having to redundantly explain the concept. If you just mean in general... well then you skipped the beginning of the article. Mongo gets used for all sorts of things now days, just like "web documents" get used as "web applications."

1 • Reply • Share ›

ALSO ON **START.JCOLEMORRISON.COM**

Building an Angular and Express App Part 1

112 comments • 2 years ago•

Fadel Chafai — Thank for the tutorial, just for the bower_components folder bower put it on the client folder and when i start the …

Journaling a New Start(up)

1 comment • 2 years ago•

Aspire Subedi — inspirational, best of luck!

Yer a Hacker Harry!

3 comments • 2 years ago•

Steve — If you define magic as simply an arcane set of knowledge where you can create things, manipulate things, or …

How to Try Out Angular in your Existing Rails App

1 comment • 6 months ago•

Allen Maxwell — Thanks for the post. Very helpful. I'd love to see how you'd use resources to pull data from a database …

Subscribe Add Disqus to your site Add Disqus Add Privacy