Many To Many Joins

(Note: for this example, I borrowed from something I did very early on, so the case style of the collection and field names is not standard!)

We can also use the lookup operator to query a many to many relationship.  Let's say I have two people and share two phone numbers.  We'll create the data like this:

Hide   Copy Code

db.Person.insert({ID: 1, LastName: "Clifton", FirstName: "Marc"})

db.Person.insert({ID: 2, LastName: "Wagers", FirstName: "Kelli"})

db.Phone.insert({ID: 1, Number: "518-555-1212"})

db.Phone.insert({ID: 2, Number: "518-123-4567"})

Note that I'm using my own ID's here, which is purely illustrative, it makes it easier to write the test examples, but this technique works equally well with the object ID "\_id" field.

Next, we'll create the many-to-many association document:

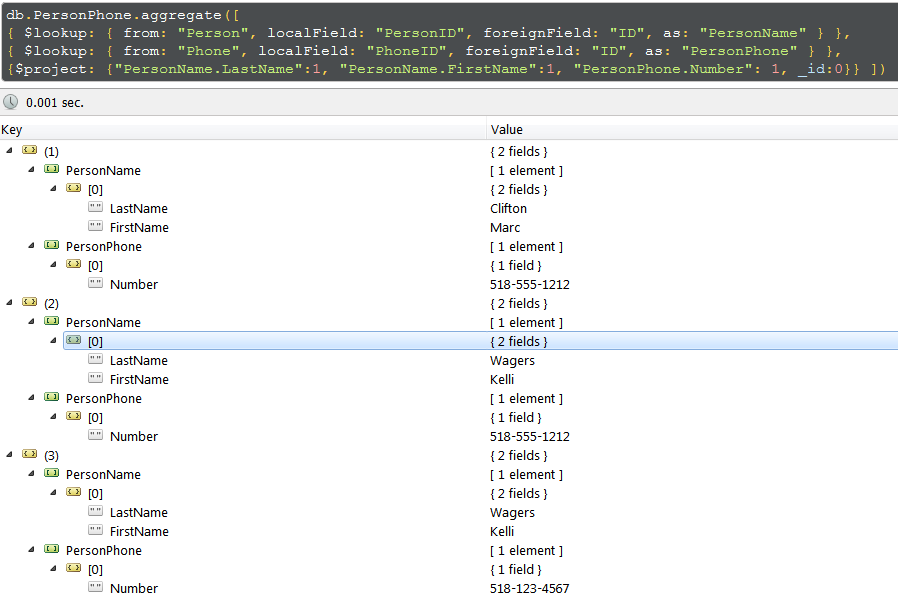
Hide   Copy Code

db.PersonPhone.insert({ID: 1, PersonID: 1, PhoneID: 1})

db.PersonPhone.insert({ID: 2, PersonID: 2, PhoneID: 1})

db.PersonPhone.insert({ID: 3, PersonID: 2, PhoneID: 2})

This results in a document with 3 records:



We can unwind the arrays and alias the document field paths for a somewhat simpler layout:

Hide   Copy Code

db.PersonPhone.aggregate([

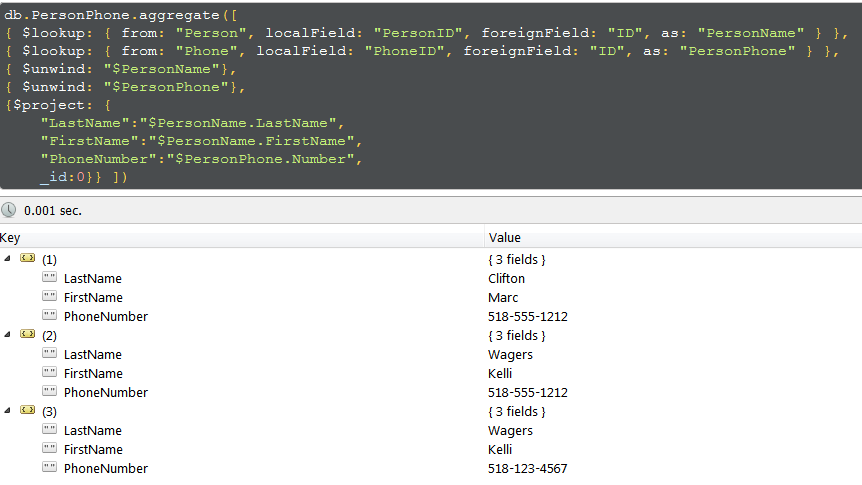
{ $lookup: { from: "Person", localField: "PersonID", foreignField: "ID", as: "PersonName" } },

{ $lookup: { from: "Phone", localField: "PhoneID", foreignField: "ID", as: "PersonPhone" } },

{ $unwind: "$PersonName"},

{ $unwind: "$PersonPhone"},

{$project: {"LastName":"$PersonName.LastName", "FirstName":"$PersonName.FirstName", "PhoneNumber":"$PersonPhone.Number", \_id:0}} ])



Nested Lookups

What if you have three collections, where collection A has a "foreign key" to collection B, and collection B has a "foreign key" to collection C?  Let's go back to our country code / country name collections and add a phone number which a separate country code, so we can then do a join like this:

phone number + country code + country name

It's somewhat artificial but it will clearly illustrate the "trick" to get this to work.  First, create a couple phone records:

Hide   Copy Code

db.phone.insert([{number: "555-1212", countryCode: 1}, {number: "851-1234", countryCode: 20}])

Note that we must unwind the each resulting document before proceeding with the next lookup:

Hide   Copy Code

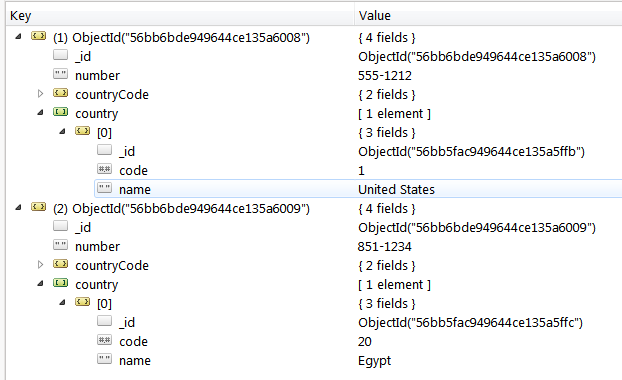
db.phone.aggregate([

{ $lookup: {from: "countryCode", localField: "countryCode", foreignField: "code", as: "countryCode"} },

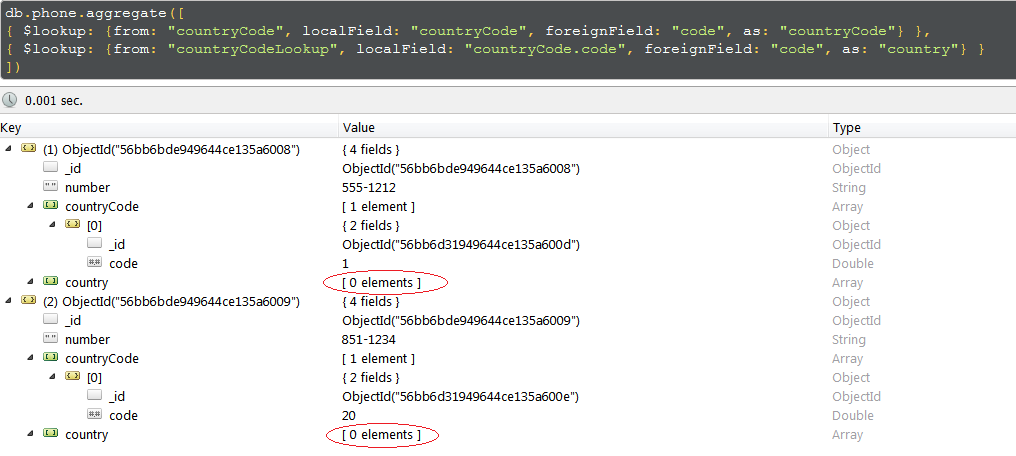
{ $unwind: "$countryCode"},

{ $lookup: {from: "countryCodeLookup", localField: "countryCode.code", foreignField: "code", as: "country"} }

])



If we don't do this, we don't get any records in our nested document:



Why is this?

As Blakes Seven [wrote on Stack Overflow](http://stackoverflow.com/a/34968489/2276361):

*The $lookup aggregation pipeline stage will not work directly with an array. The main intent of the design is for a "left join" as a "one to many" type of join ( or really a "lookup" ) on the possible related data. But the value is intended to be singular and not an array.  Therefore you must "de-normalise" the content first prior to performing the $lookup operation in order for this to work. And that means using $unwind*

It took me a bit find that solution!

Again, we can further denormalize the data with a final unwind and an alias field path projection:

Hide   Copy Code

db.phone.aggregate([

{ $lookup: {from: "countryCode", localField: "countryCode", foreignField: "code", as: "countryCode"} },

{ $unwind: "$countryCode"},

{ $lookup: {from: "countryCodeLookup", localField: "countryCode.code", foreignField: "code", as: "country"} },

{ $unwind: "$country"},

{ $project: {"number": "$number", "countryCode": "$countryCode.code", "country": "$country.name", "\_id":0} }

])

