MongoDB – Part 2

Abdu Alawini

University of Illinois at Urbana-Champaign

CS411: Database Systems

Some slides were adopted from D. Maier. S. Davidson with permission

Learning Objectives

After this lecture, you should be able to:

- Write MongoDB cursor queries.
- Write aggregation queries
- Query documents by reference.

Cursor methods

- .count, .pretty, .sort etc are all examples of cursor methods
- **.toArray** returns an array that contains all documents from a cursor

```
nsf= db.awards.find({"by": "National Science Foundation").toArray()
if (nsf.length >0) {printjson (nsf[0])}
```

- **.forEach** applies a JavaScript function to each document from the cursor (similar to .map)

```
db.awards.find(). forEach
 ( function(myDoc) { printjson ("Award: " + myDoc.name); });
```

Aggregation

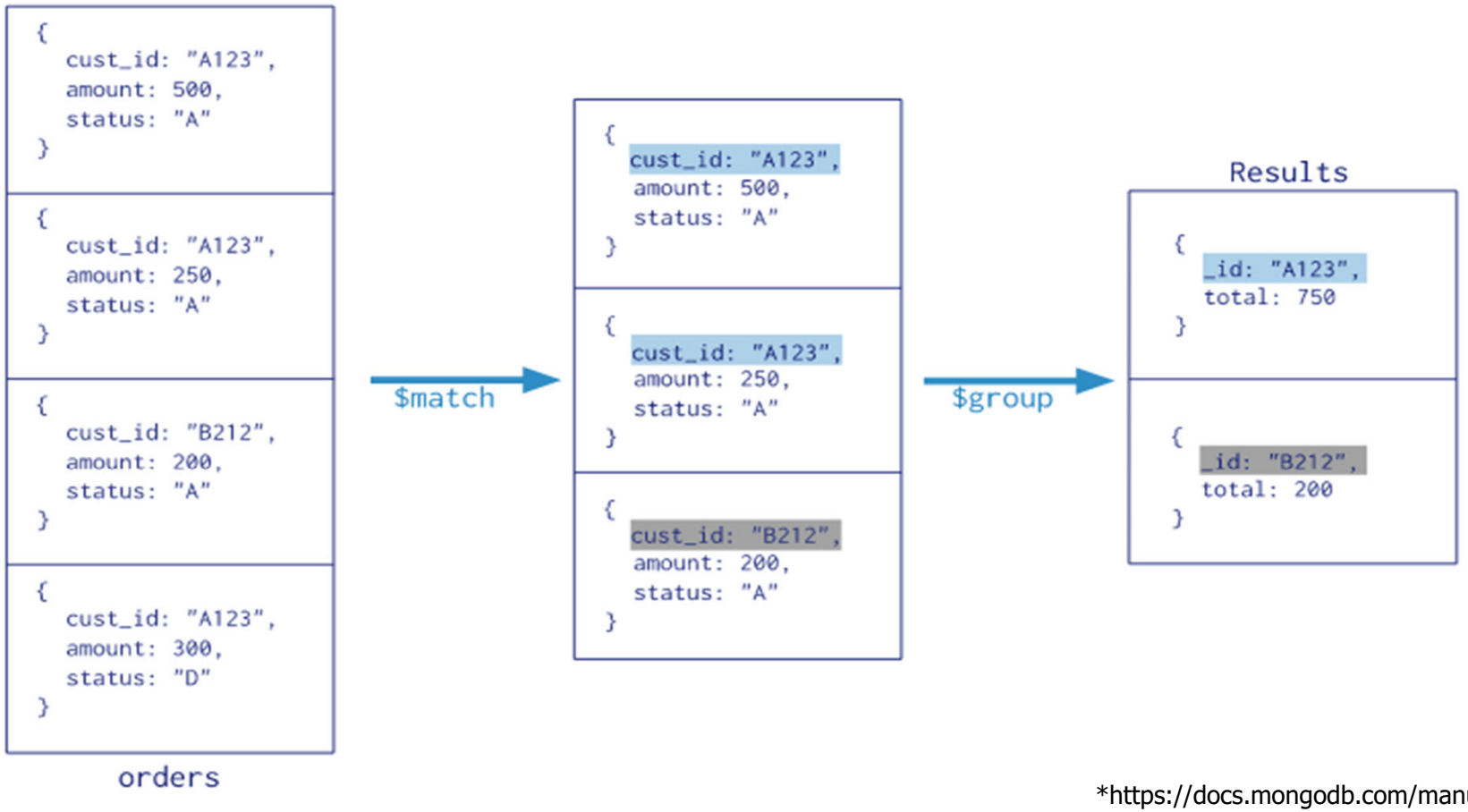
- A framework to provide "group-by" and aggregate functionality without the overhead of map-reduce.

- Conceptually, documents from a collection pass through an aggregation pipeline, which transforms the objects as they pass through (similar to UNIX pipe "|")

- Operators include: $project, $match, $group, $sort, $skip, $limit, $unwind

Aggregation Example*

```
Collection
db.orders.aggregate( [
    $match stage ──►  { $match: { status: "A" } },
    $group stage ──►  { $group: { _id: "$cust_id",total: { $sum: "$amount" } } }
                   ] )
```



*https://docs.mongodb.com/manual/aggregation/

Aggregation: $group

- Every group expression must specify an _id field.
- Suppose we wanted to find how many people were born each year

```
> db.people.aggregate( { $group :
    { _id : "$birthyear", birthsPerYear : { $sum : 1}}})
```

```
{ "_id" : 1924, "birthsPerYear" : 1 }
```

- Contrast with aggregate operation over entire result

```
> db.people.count( )
> db.people.find({“name.first”: “John”}).count( )
> db.people.count({“name.first”: “John”})
```

Aggregation: $unwind

- Deconstructs an array field to output a document for each element.

```
Posts:   {
         _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),
         author : "Kevin",
         date : new Date("February 2, 2012"),
         text : "About MongoDB...",
         birthyear:  1980,
         tags : [ "tech", "databases" ]
         }
```

```
>db.posts.aggregate( { $project : { author : 1, tags : 1 }}, { $unwind : "$tags" } )
```

Result of unwind

```
>db.posts.aggregate( { $project : { author : 1, tags : 1 }}, { $unwind : "$tags" } )
```

```
{ "_id" : ObjectId("4c4ba5c0672c685e5e8aabf3"),
        "author" : "Kevin",
        "tags" : ”tech" },
{ "_id" : ObjectId("4c4ba5c0672c685e5e8aabf3"),
        "author" : "Kevin",
        "tags" : ”databases" }
```

“Joins” in MongoDB

1. Embedded Relationships
   - “Do joins while write, not on reads.”

2. Referenced Relationships ($lookup)
   - Use a left outer-joins to a collection in the same database to search (filter) in documents from the (joined) second collection for processing.

Relationships: Embedded

```
{ _id: 1,
  name: { first: "John", last: "Backus" },
  birthyear: 1924,
  contribs: [ "Fortran", "ALGOL",
              "Backus Naur Form", "FP" ],
  awards: [ {title: "National Medal of Science" ,
             by: "National Science Foundation",
             year: 1975 },
            {title: "Turing Award",
             by: "ACM" ,
             year: 1977} ,
            {title: "Career Award",
             by: "NIH" ,
             year: 1980} ,
             {title: "Career Success Award",
             by: "NASA" ,
             year: 1982} ]  }
```

Aggregation: $lookup

- For each input doc, the $lookup stage adds a new array field whose elements are the matching documents from the 2nd (joined) collection
- Syntax:

```
db.colName.aggregate{
   $lookup:
     {
       from: <collection to join>,
       localField: <field from the input documents(colName)>,
       foreignField: <field from the documents of the "from" collection>,
       as: <output array field to be added to colName>
     }
}
```

$lookup Example

customers :


```
{"_id" : ObjectId("60299bc2713966e59be2071f"),
    "CustomerID" : "FRANR",
     "Address" : 54,
    "City" : "rue Royale",
     …
    "Fax" : "40.32.21.21",
    "field11" : "40.32.21.20"}
```


orders:


```
{"_id" : ObjectId("60299b3d37036677a386fdob"),
    "OrderID" : 10268,
    "CustomerID" : " FRANR",
    "EmployeeID" : 8,
    "OrderDate" : "1996-07-30 00:00:00.000",
    …}
```

$lookup Example: Query

```
db.customers.aggregate([
        {$lookup:
                {
                 from:"orders",
                 localField:"CustomerID",
                 foreignField:"CustomerID",
                 as:"Cust_Orders"
                 }
            }
])
```

$lookup Example: Result


```
{
    "_id" : ObjectId("60299bc2713966e59be2071f"),
    "CustomerID" : "FRANR",
     "Address" : 54,
    "City" : "rue Royale",
     …
    "Fax" : "40.32.21.21",
    "field11" : "40.32.21.20“,
    "Cust_Orders" : [
                    {"_id" : ObjectId("60299b3d3703667a386fdob"),
                     "OrderID" : 10268,
                     "CustomerID" : " FRANR",
                     "EmployeeID" : 8,
                     "OrderDate" : "1996-07-30 00:00:00.000",
                     …}
                    ]
}
```