NBD MongoDB lab. 2

Tasks for the student are denoted with “Taskx” where x is a number. Solutions to such tasks should be submitted to eNauczanie. “Task*” denotes practice tasks not requiring submission.

Please open command terminal and move to the directory with MongoDB binaries.

Create data_db directory than run:

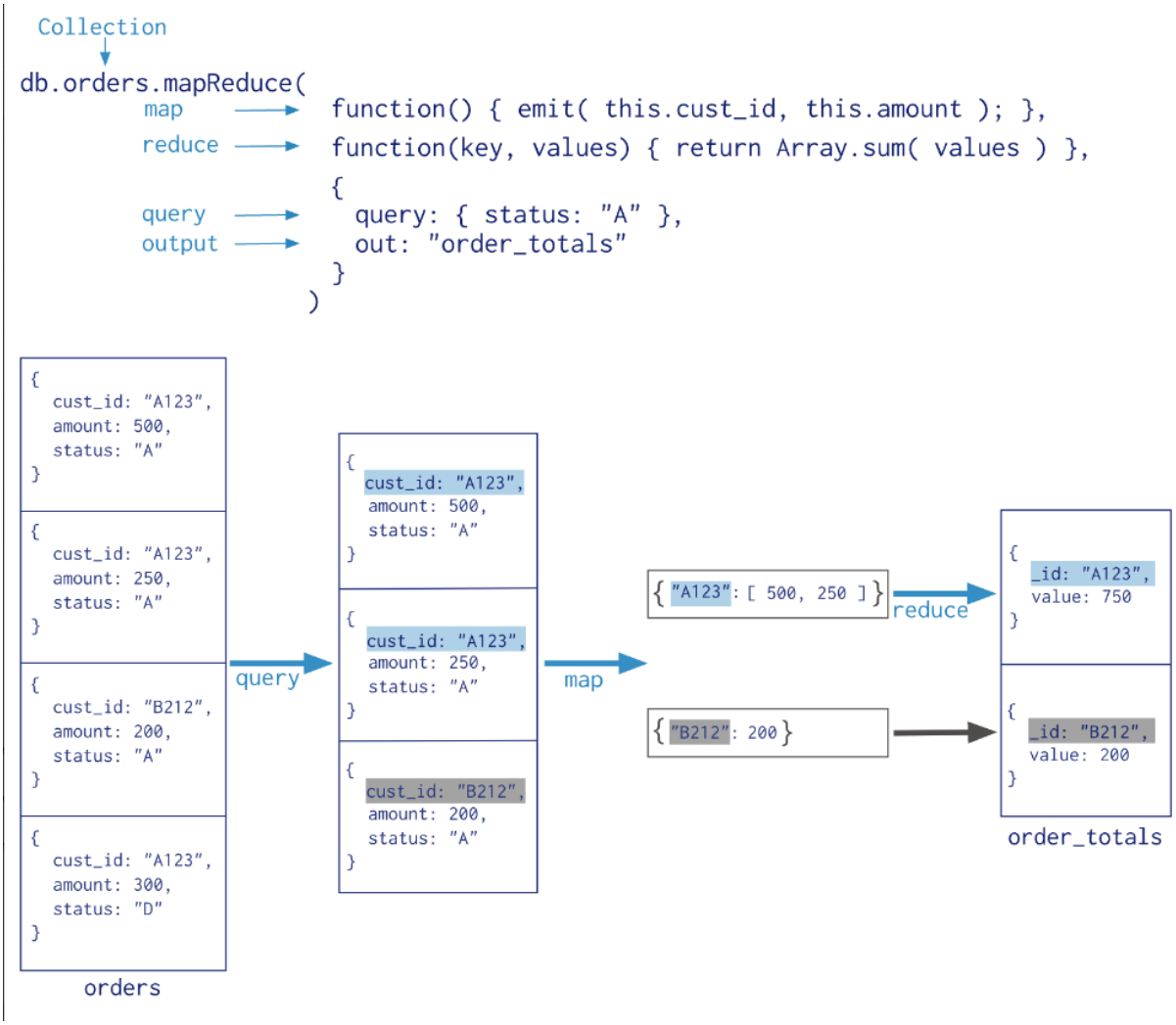
mongod --dbpath .\data_db (or choose other empty directory).

Open new command terminal window and run

mongosh (will connect with localhost:27017, other address can be given)

Add "packages", "inventory", "sillinesses", "orders" and "stock" collections from “Sample documents for lab 2” file in eNauczanie.

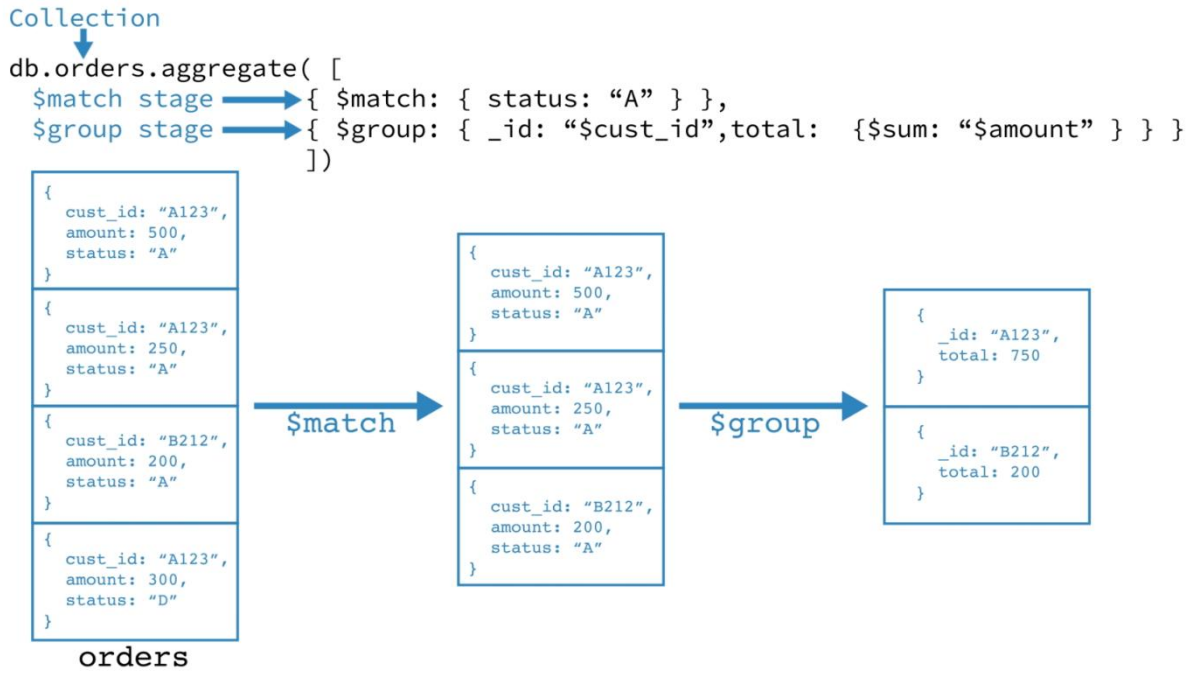
Map-Reduce in MongoDB:



In map-reduce operation map function is applied to each document in a collection that matches the query and emits key-value pairs. Reduce operation is than applied to keys with multiple values.

Optionally, output of reduce can be passed through finalize function for further condensing/processing. All Map-Reduce functions in MongoDB are JavaScript.

Aggregation pipeline in MongoDB:



Framework for data aggregation based on data processing pipelines concept. Documents from a single collection are passed through a series of stages, each stage transforming the passing documents. Neither number of the documents, nor the documents format needs to remain constant between entering and leaving a stage. Documents leaving the last stage are passed to the user, unless $out or $merge stages are used.

Aggregation stages(with description) can be found in:
https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/

Notable among them are:

- $match – filtering out unwanted documents,
- $project – rebuilding documents (removing fields, adding new fields, renaming old fields...; can be use flatten arrays by use of accumulator expressions),
- $unwind – splitting array elements into separate documents,
- $group – analogous to SQL "GROUP BY" phrase, use to apply accumulator expressions to multiple documents ($avg, $min, $sum, $addToSet),
- $lookup – left outer join with another collection (must be unsharded),
- $out – outputs documents to collection (final stage),
- $count – counts documents (final stage).

Efficiency consideration: use $match, $limit and $skip as early in the stream as possible.

As a revision the following task will be done by students. The first student to do a task correctly will earn points.

--Querying documents

Write mongo shell query for collection packages that would:

- find documents
  - without "destination" field,
  - containing an array in field "items" that
    - contains a document whose "type" filed equals one of: "electric", "electronic", "electromechanical"
    - and contains a document whose "quantity" field is divisible by 4
- return the following for at most 5 matched documents:
  - no "_id" field,
  - "name" field
  - the first document in array "items":
    - whose "type" is not one of: "electric", "electronic", "electromechanical"
    - and whose "quantity" is not divisible by 4

```
db.packages.find({
        destination: {$exists: false},
        "items.type": {$in: ["electric", "electronic", "electromechanical"]},
        "items.quantity": {$mod: [4,0]}
},
{_id: 0,
        name: 1,
        items: {$elemMatch: {type: {$nin: ["electric", "electronic", "electromechanical"]},
        quantity: {$not: {$mod: [4,0]}}}}
}).limit(5)
```

--Updating document

Write mongo shall query for collection inventory that would update the first document whose type equals "other" in the following way:

- change the value of the field "type" to "other type"
- if "weight" is greater than 200, set it to 200,
- increase "quantity" by 1.

```
db.inventory.updateOne({type: "other"},{$set: {type: "other type"}, $min: {weight: 200}, $inc: {quantity: 1}})
```

--Delete documents

Write a mongo shell query that would delete all documents that:

- have array "bob" of size 4,
- and "coolness" field that is greater or equal 10.

```
db.sillinesses.deleteMany({bob:{$size: 4}, coolness:{$gte: 10}})
```

--Aggregation Pipeline

Collection "orders" stores current order documents.

Collection "stock" lists quantities of items in store.

Write a mongo shell aggregate statement that will identify items without enough supply in store to serve current orders and list their names.

```
db.orders.aggregate([
	{$unwind: "$orderedItems"},
	{$group: {_id: "$orderedItems.name", required: {$sum: "$orderedItems.qty"}}},
	{$lookup: {from:"stock", localField: "_id", foreignField: "item", as: "supply"}},
	{$project: {_id: 0, item: "$_id", missing: {$let: {vars: {sup: { $arrayElemAt: [ "$supply", 0 ] }},
in: {$subtract: ["$required", {$ifNull: ["$$sup.qty", 0]}]}}}}},
	{$match: {missing: {$gt: 0}}}
])
```

Now it is time for an individual task. On eNauczanie you will find “Sample collections for the individual task” and “Individual task”. Please solve the tasks described in the “Individual task” by yourself and submit the solutions to eNauczanie. Collections for the task are provided in “Sample collections for the individual task” and “Individual task”.