OPEN SOURCE
DEPARTMENT

Mongo Database

Scale the web

Course Material

You can access course material via this URL:

http://tinyurl.com/iti-mongo

Agenda

- Mongo aggregation framework
- Schema Modeling

Mongo Aggregation

Mongo Aggregation

- Aggregations operations process data records and return computed results.
- aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result.
- MongoDB provides three ways to perform aggregation: **the aggregation pipeline**, **single purpose aggregation methods** and **the map-reduce function.**
- MongoDB provides a rich set of aggregation operations that examine and perform calculations on the data sets.
- Running data aggregation on the Mongod instance simplifies application code and limits resource requirements.

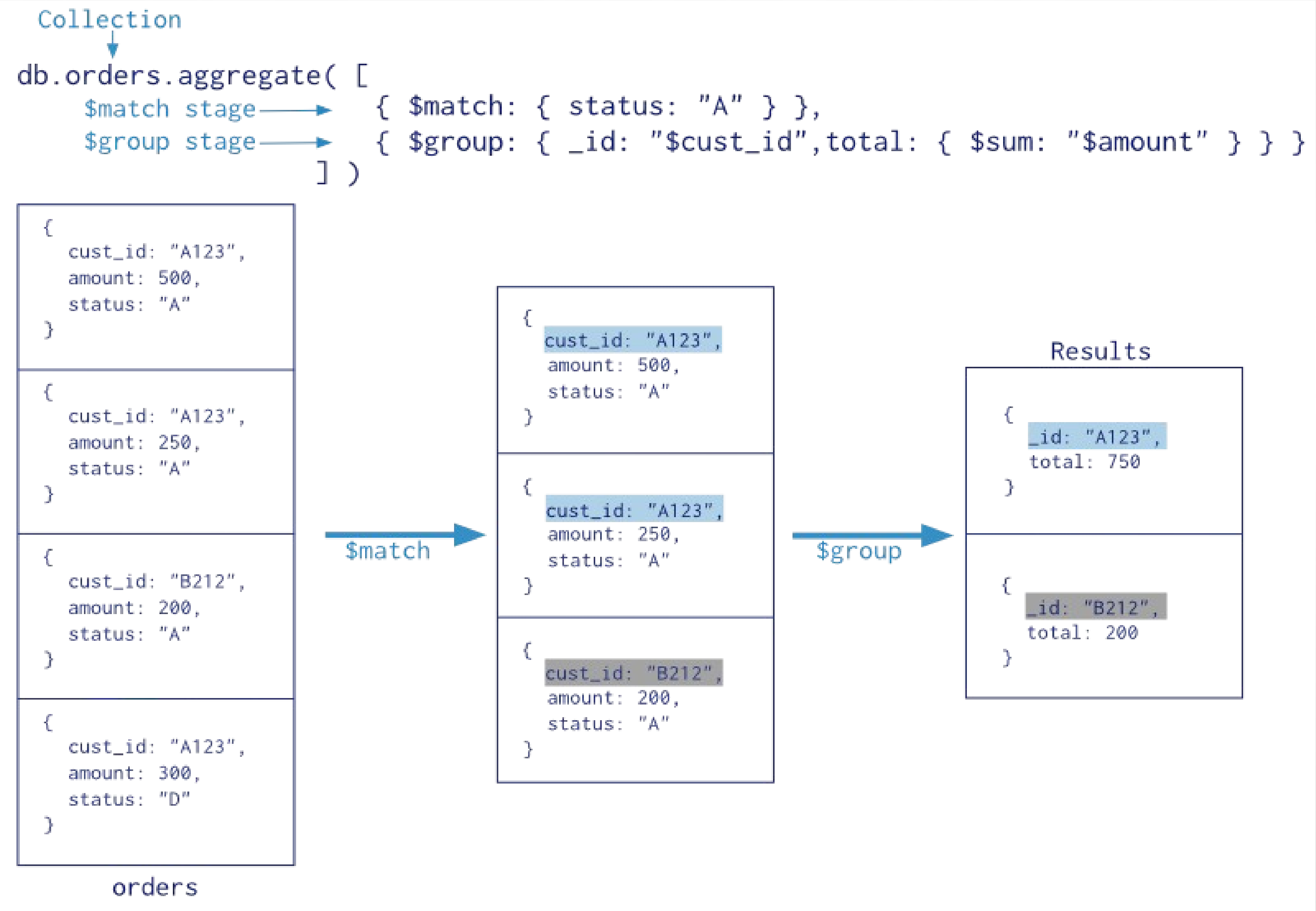
Mongo Aggregation (Pipeline)

- The aggregation pipeline is a framework for data aggregation modeled on the concept of data processing pipelines.

- Documents enter a multi-stage pipeline that transforms the documents into an aggregated results.

- Pipeline stages do not need to produce one output document for every input document; e.g., some stages may generate new documents or filter out documents. Pipeline stages can appear multiple times in the pipeline.

Mongo Aggregation (Pipeline)



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

orders

```
{
   cust_id: "A123",
   amount: 500,
   status: "A"
}
{
   cust_id: "A123",
   amount: 250,
   status: "A"
}
{
   cust_id: "B212",
   amount: 200,
   status: "A"
}
{
   cust_id: "A123",
   amount: 300,
   status: "D"
}
```

$match

```
{
   cust_id: "A123",
   amount: 500,
   status: "A"
}
{
   cust_id: "A123",
   amount: 250,
   status: "A"
}
{
   cust_id: "B212",
   amount: 200,
   status: "A"
}
```

$group

Results

```
{
   _id: "A123",
   total: 750
}
{
   _id: "B212",
   total: 200
}
```

Mongo Aggregation (Pipeline)

Example:

```
{
  "_id": "10280",
  "city": "NEW YORK",
  "state": "NY",
  "pop": 5574,
  "loc": [
    -74.016323,
    40.710537
  ]
}
```

Mongo Aggregation (Pipeline)

return all states with a population greater than 10 million

```
db.zipcodes.aggregate( { $group :
                                { _id : "$state",
                                 totalPop : { $sum : "$pop" }
                                }
                       },
                      { $match :
                             { totalPop : { $gte : 10*1000*1000 } }
                       } )
```

Mongo Aggregation (Pipeline)

return the average populations for cities in each state

```
db.zipcodes.aggregate( { $group :
                              { _id : { state : "$state", city : "$city" },
                               pop : { $sum : "$pop" } }
                        },
                        { $group :
                              { _id : "$_id.state",
                               avgCityPop : { $avg : "$pop" } }
                         } )
```

Mongo Aggregation (Pipeline)

return the smallest and largest cities by population for each state

```
db.zipcodes.aggregate(  { $group:
                                 { _id: { state: "$state", city: "$city" },
                                   pop: { $sum: "$pop" } } },
                        { $sort: { pop: 1 } },
                        { $group:
                                   { _id : "$_id.state",
                                     biggestCity:  { $last: "$_id.city" },
                                     biggestPop:   { $last: "$pop" },
                                     smallestCity: { $first: "$_id.city" },
                                     smallestPop:  { $first: "$pop" }
                                   }
                        })
```

Mongo Aggregation (Pipeline)

return the smallest and largest cities by population for each state

```
{ $project:
                    { _id: 0,
                      state: "$_id",
                      biggestCity:  { name: "$biggestCity",  pop: "$biggestPop" },
                      smallestCity: { name: "$smallestCity", pop: "$smallestPop" }
}
```

Mongo Aggregation (Pipeline)

$project

Passes along the documents with only the specified fields to the next stage in the pipeline.
The specified fields can be existing fields from the input documents or newly computed fields.

$out

Takes the documents returned by the aggregation pipeline and writes them to a specified collection.
The $out operator must be the last stage in the pipeline.

$skip

Skips over the specified number of documents that pass into the stage and passes the remaining documents to the next stage in the pipeline.

$limit

Limits the number of documents passed to the next stage in the pipeline.

Mongo Aggregation (Pipeline)

$sort

Sorts all input documents and returns them to the pipeline in sorted order.

$match

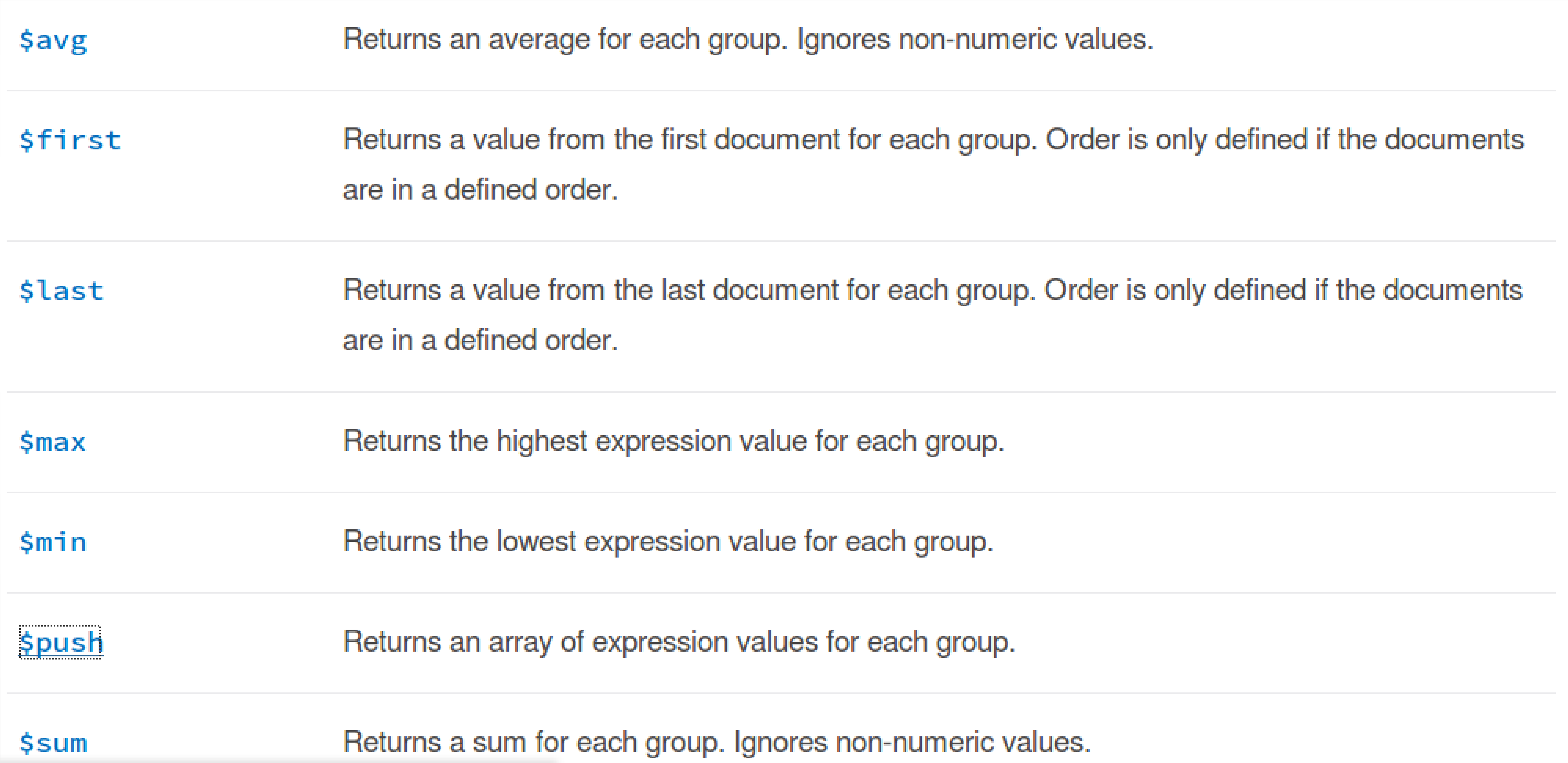
Filters the documents to pass only the documents that match the specified condition(s) to the next pipeline stage.

$group

- Groups documents by some specified expression and outputs to the next stage a document for each distinct grouping.
- The output documents contain an _id field which contains the distinct group by key.
- The output documents can also contain computed fields that hold the values of some accumulator expression grouped by the $group‘s _id field.
- $group does not order its output documents.

Mongo Aggregation (Pipeline)

$group Operators



`$avg`	Returns an average for each group. Ignores non-numeric values.
`$first`	Returns a value from the first document for each group. Order is only defined if the documents are in a defined order.
`$last`	Returns a value from the last document for each group. Order is only defined if the documents are in a defined order.
`$max`	Returns the highest expression value for each group.
`$min`	Returns the lowest expression value for each group.
`$push`	Returns an array of expression values for each group.
`$sum`	Returns a sum for each group. Ignores non-numeric values.

Mongo Aggregation (Single Purpose Methods)

count()

MongoDB can return a count of the number of documents that match a query.

distinct()

The distinct operation takes a number of documents that match a query and returns all of the unique values for a field in the matching documents.

Schema Modeling

Schema Modeling

- One of the challenges that comes with moving to MongoDB is figuring how to best model your data.

- Data in **MongoDB** has a flexible schema. Collections do not enforce document structure. Decisions that affect how you model data can affect application performance and database capacity

- most developers have internalized the rules of thumb for designing schemas for RDBMSs, these rules don't always apply to MongoDB.

- The simple fact that documents can represent rich, schema-free data structures means that we have a lot of viable alternatives to the standard, normalized, relational model.

Schema Modeling

- **Model One-to-One Relationships with Embedded Documents:**
  Presents a data model that uses embedded documents to describe one-to-one relationships between connected data.

- **Model One-to-Many Relationships with Embedded Documents**
  Presents a data model that uses embedded documents to describe one-to-many relationships between connected data.

- **Model One-to-Many Relationships with Document References**
  Presents a data model that uses references to describe one-to-many relationships between documents.