mongoDB

AGGREGATION FRAMEWORK

Emily Stolfo **@EmStolfo**

PIPELINE

PIPELINE

- Process a stream of documents
  - Original input is a collection
  - Final output is a result document
- Series of operators
  - Filter or transform data
  - Input/output chain

```
ps ax | grep mongod | head -n 1
```

PIPELINE OPERATORS

- `$match`
- `$project`
- `$group`
- `$unwind`
- `$sort`
- `$limit`
- `$skip`

WHAT ABOUT MAP REDUCE?

$MATCH

- Filter documents
- Place early in the pipeline if possible
- Uses existing **query syntax**
- No `$where` or **geospatial operations**

$MATCH

MATCHING FIELD VALUES

```
{
  title: "The Great Gatsby",
  pages: 218,
  language: "English"
}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

```
{
  title: "Atlas Shrugged",
  pages: 1088,
  language: "English"
}
```

▶

```
{ $match: {
  language: "Russian"
}}
```

▼

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

$MATCH

MATCHING WITH QUERY OPERATORS

```
{
  title: "The Great Gatsby",
  pages: 218,
  language: "English"
}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

```
{
  title: "Atlas Shrugged",
  pages: 1088,
  language: "English"
}
```

```
{ $match: {
  pages: { $gt: 1000 }
}}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

```
{
  title: "Atlas Shrugged",
  pages: 1088,
  language: "English"
}
```

$PROJECT

- Reshape documents
- Include, exclude or rename fields
- Inject computed fields
- Create sub-document fields

$PROJECT

INCLUDING AND EXCLUDING FIELDS

```
{
  _id: 375,
  title: "The Great Gatsby",
  ISBN: "9781857150193",
  available: true,
  pages: 218,
  subjects: [
    "Long Island",
    "New York",
    "1920s"
  ],
  language: "English"
}
```

```
{ $project: {
  _id: 0,
  title: 1,
  language: 1
}}
```

```
{
  title: "The Great Gatsby",
  language: "English"
}
```

$PROJECT

RENAMING AND COMPUTING FIELDS

```
{
  _id: 375,
  title: "The Great Gatsby",
  ISBN: "9781857150193",
  available: true,
  pages: 218,
  subjects: [
    "Long Island",
    "New York",
    "1920s"
  ],
  language: "English"
}
```

```
{ $project: {
  upperTitle: {
    $toUpper: "$title"
  },
  lang: "$language"
}}
```

```
{
  _id: 375,
  upperTitle: "THE GREAT GATSBY",
  lang: "English"
}
```

$PROJECT

CREATING SUB-DOCUMENT FIELDS

```
{
  _id: 375,
  title: "The Great Gatsby",
  ISBN: "9781857150193",
  available: true,
  pages: 218,
  subjects: [
    "Long Island",
    "New York",
    "1920s"
  ],
  language: "English"
}
```

```
{ $project: {
  title: 1,
  stats: {
    pages: "$pages",
    language: "$language",
  }
}}
```

```
{
  _id: 375,
  title: "The Great Gatsby",
  stats: {
    pages: 218,
    language: "English"
  }
}
```

EXPRESSIONS

EXPRESSIONS

- Return computed values
- Used with `$project`

$PROJECT - BOOLEAN OPERATORS

- Input array of one or more values
  - `$and`, `$or`
- Inverse values with `$not`
- BSON standard for converting non-booleans
  - `null`, `undefined`, zero values → `false`
  - Non-zero values, strings, dates, objects → `true`

$PROJECT - COMPARISON OPERATORS

- Compare numbers, strings, and dates
- Input array with two operands
  - `$cmp`, `$eq`, `$ne`
  - `$gt`, `$gte`, `$lt`, `$lte`

$PROJECT - ARITHMETIC OPERATORS

- Input array of one or more numbers
  - `$add`, `$multiply`
- Input array of two operands
  - `$subtract`, `$divide`, `$mod`

$PROJECT - STRING OPERATORS

- `$strcasecmp` case-insensitive comparison
  - `$cmp` is case-sensitive
- `$toLower` and `$toUpper` case change
- `$substr` for sub-string extraction
- Not encoding aware
- Assumes Latin alphabet

$PROJECT - DATE OPERATORS

- Extract date components
  - `$dayOfYear`, `$dayOfMonth`, `$dayOfWeek`
  - `$year`, `$month`, `$week`
  - `$hour`, `$minute`, `$second`

$PROJECT - CONDITIONAL OPERATORS

- `$cond` ternary operator
- `$ifNull`

$GROUP

- Documents are grouped by an ID that can be:
  - Field path reference ($)
  - Object with multiple references
  - Constant value
- Each field uses group aggregation function
  - `$max`, `$min`, `$avg`, `$sum`
  - `$addToSet`, `$push`
  - `$first`, `$last`

$GROUP

CALCULATING AN AVERAGE

```
{
  title: "The Great Gatsby",
  pages: 218,
  language: "English"
}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

```
{
  title: "Atlas Shrugged",
  pages: 1088,
  language: "English"
}
```

▶

```
{ $group: {
  _id: "$language",
  avgPages: { $avg: "$pages" }
}}
```

▼

```
{
  _id: "Russian",
  avgPages: 1440
}
```

```
{
  _id: "English",
  avgPages: 653
}
```

$GROUP

SUMMING FIELDS AND COUNTING

```
{
  title: "The Great Gatsby",
  pages: 218,
  language: "English"
}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

```
{
  title: "Atlas Shrugged",
  pages: 1088,
  language: "English"
}
```

```
{ $group: {
  _id: "$language",
  numTitles: { $sum: 1 },
  sumPages: { $sum: "$pages" }
}}
```

```
{
  _id: "Russian",
  numTitles: 1,
  sumPages: 1440
}
```

```
{
  _id: "English",
  numTitles: 2,
  sumPages: 1306
}
```

$GROUP

COLLECTING DISTINCT VALUES

```
{
  title: "The Great Gatsby",
  pages: 218,
  language: "English"
}
```

```
{
  title: "War and Peace",
  pages: 1440,
  language: "Russian"
}
```

```
{
  title: "Atlas Shrugged",
  pages: 1088,
  language: "English"
}
```

▶

```
{ $group: {
  _id: "$language",
  titles: {
    $addToSet: "$title"
  }
}}
```

▼

```
{
  _id: "Russian",
  titles: [
    "War and Peace"
  ]
}
```

```
{
  _id: "English",
  titles: [
    "Atlas Shrugged",
    "The Great Gatsby"
  ]
}
```

$UNWIND

- Often used to operate on an array field
- Yield documents for each array value
  - Nothing for absent or empty fields
  - Error for non-array fields
- Complements `$group`

$UNWIND

YIELDING MULTIPLE DOCUMENTS FROM ONE

```
{
  title: "The Great Gatsby",
  ISBN: "9781857150193",
  subjects: [
    "Long Island",
    "New York",
    "1920s"
  ]
}
```

▶

```
{ $unwind: "$subjects" }
```

▼

```
{
  title: "The Great Gatsby",
  ISBN: "9781857150193",
  subjects: "Long Island"
}
```

```
{
  title: "The Great Gatsby",
  ISBN: "9781857150193",
  subjects: "New York"
}
```

```
{
  title: "The Great Gatsby",
  ISBN: "9781857150193",
  subjects: "1920s"
}
```

$SORT

- Sort documents by one or more fields
- Uses familiar **cursor format**
- Waits for earlier pipeline operator to return
- In-memory unless early and indexed

$SORT

SORT ALL DOCUMENTS IN THE PIPELINE

```
{ title: "The Great Gatsby" }
{ title: "Brave New World" }
{ title: "The Grapes of Wrath" }
{ title: "Animal Farm" }
{ title: "Lord of the Flies" }
{ title: "Fathers and Sons" }
{ title: "Invisible Man" }
{ title: "Fahrenheit 451" }
```

▶

```
{ $sort: { title: 1 }}
```

▼

```
{ title: "Animal Farm" }
{ title: "Brave New World" }
{ title: "Fahrenheit 451" }
{ title: "Fathers and Sons" }
{ title: "Invisible Man" }
{ title: "Lord of the Flies" }
{ title: "The Grapes of Wrath" }
{ title: "The Great Gatsby" }
```

$LIMIT

LIMIT DOCUMENTS THROUGH THE PIPELINE

$SKIP

SKIP OVER DOCUMENTS IN THE PIPELINE

```
{ title: "The Great Gatsby" }
{ title: "Brave New World" }
{ title: "The Grapes of Wrath" }
{ title: "Animal Farm" }
{ title: "Lord of the Flies" }
{ title: "Fathers and Sons" }
{ title: "Invisible Man" }
{ title: "Fahrenheit 451" }
```

```
{ $skip: 5 }
```

```
{ title: "Fathers and Sons" }
{ title: "Invisible Man" }
{ title: "Fahrenheit 451" }
```

EXTENDING THE FRAMEWORK

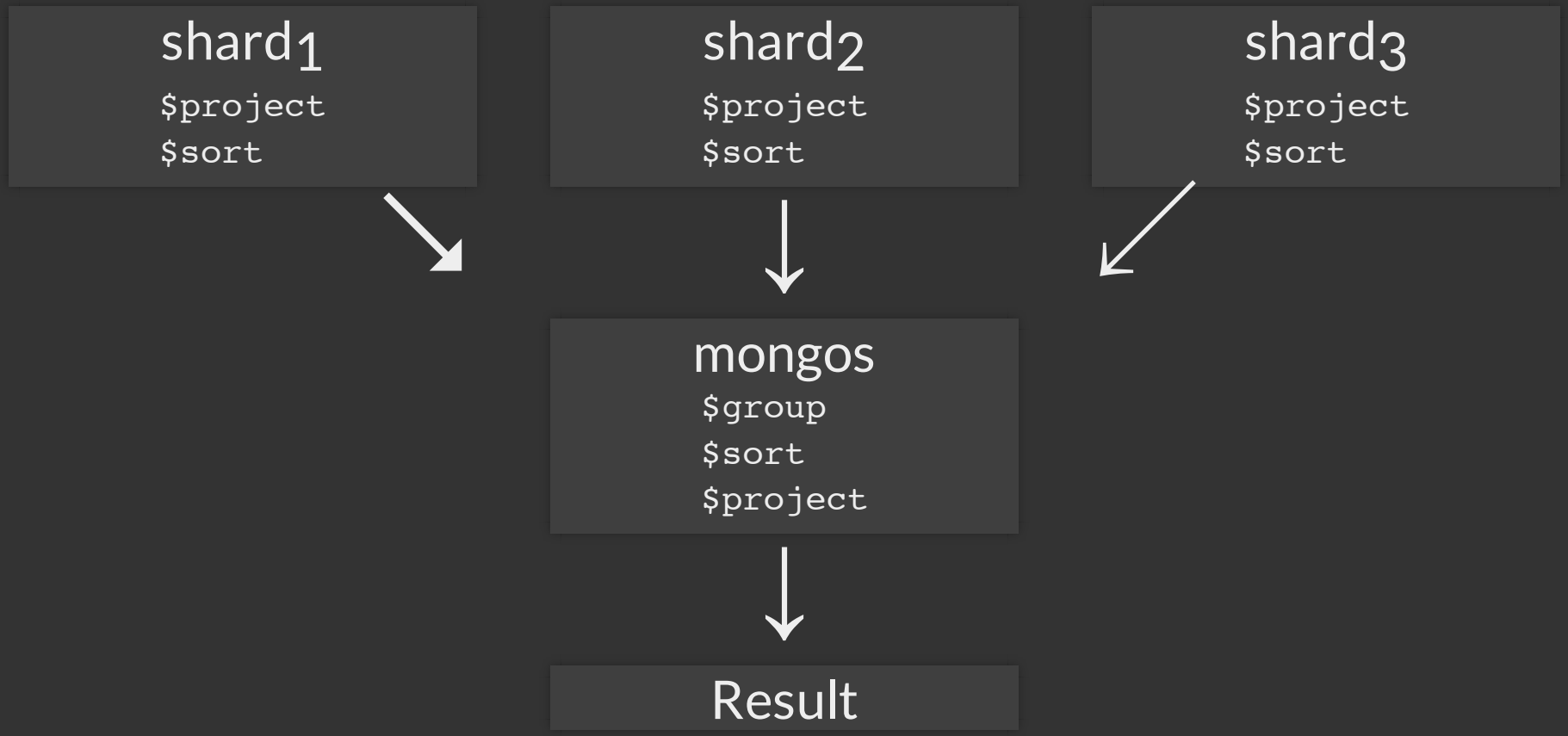
- Future pipeline operators and expressions
- `$out` and `$tee` for output control
  - https://jira.mongodb.org/browse/SERVER-3253

SHARDING

SHARDING

- All operators up to first `$group` or `$sort` pushed to each shard
  - Shards execute operators
  - **mongos** merges results and continues
- Early `$match` may excuse shards
- CPU and memory implications for **mongos**

SHARDING


shard1
$project
$sort
shard2
$project
$sort
shard3
$project
$sort
mongos
$group
$sort
$project
Result

MEMORY USAGE

- Final output must be less than 16mb
- `$group` and `$sort` entirely in-memory
- Operation memory usage limits
  - Warning at >5%
  - Error at >10%

EXAMPLES