Advanced MongoDB

mongoDB

{ name: mongo, type: DB }

What is Mongo :

1. NoSql Database
2. Schema Less
3. Schema full
4. Document Oriented

Why Mongo:

1. Simple
2. fast
3. schema less

Other NoSql Db

Redis
Hbase
Neo4J,OrientDB

How To Design Blog Application Schema ?

MongoDB Data Model

```
{       title: ‘Too Big to  Fail’,
        text: ‘article text here...’,
        author: ‘pawan’,
        ts: Date(“05-Nov-09 10:33”),
        comments: [ { author: 'rahul',
                      comment: 'Great article!' },
        { author: 'amit',
        comment: ‘How fast is it?',
                      replies: [ {author: 'pooja',
        comment: 'scalable?' } ]
                }
        ]
,
        tags: [‘finance’, ‘economy’]
}
```

Querying

Simple :

```
db.getCollection('agg').find({"likes":{"$gte":100}})
```

Advanced :

```
    $gt, $lt, $gte, $lte, $ne, $in, $all, $size, $nin, $all
```

```
db.getCollection('agg').find({"likes":{"$ne":100}})
```

$INC

```
db.getCollection('agg').update({"_id" ObjectId("58d92efe3a2c66d18f33ef84")},{$inc:{ likes : 1}})
db.getCollection('agg').update({"by_user" : "tutorials point"},{$inc:{ likes : 1}})
db.getCollection('agg').updateMany({"by_user" : "tutorials point"},{$inc:{ likes : 1}})
```

Update/updateMany/   Multi true

```
db.getCollection('agg').update({"by_user" : "tutorials point"},{$inc:{ likes : 1}},{multi:true})
db.getCollection('agg').update({"by_user" : "tutorials point"},{"title" : "MongoDB Overview test"})
db.getCollection('agg').update({"by_user" : "tutorials point"},{$set:{"title" : "MongoDB Overview test"}})
```

$exists operator

```
db.getCollection('companies').find( { "rank.sasb": { $exists: true }
},{"rank.sasb":1} )
```

MongoDump

mongodump a Collection

/usr/bin/mongodump --db test --out /tmp/

```
mongodump  --db test --collection collection
```

mongodump a Database Excluding Specified Collections

```
mongodump  --db test --excludeCollection=orders
--excludeCollection=t1
```

Mongorestore

```
mongorestore --db test_dump /tmp/test/
```

MongoExport

Export in JSON Format

```
mongoexport --db test --collection t1 --out t1.json
```

Export in CSV Format

```
mongoexport --db test --collection t1 --type=csv --fields name --out
/tmp/t1.csv
```

MongoImport

Import Json File

```
mongoimport --db test_dump --collection t1 --file t1.json
```

Import CSV file

```
mongoimport --db test_dump --collection t1 --type csv --headerline
--file /tmp/t1.csv
```

Aggregation

The MongoDB, aggregation pipeline is a framework for data aggregation modeled on the concept of data processing pipelines. Documents enter as an input into multi-stage pipeline which transforms the documents into an aggregated results.

The MongoDB aggregation pipeline consists various stages. Each stage transforms the documents passes through the pipeline

Stages operators:

$project

$match

$group

MongoDB Aggregation Stages Operator - $project

- **$project - to include Specific Fields in Output Documents:**

  db.collection.aggregate([{ $project : { <field1> : 1 , <field2> : 1 } }]);

- **$project - to Suppress _id Field in the Output Documents:**

  db.collection.aggregate([{ $project : { _id: 0, <field1> : 1 } }]);

- **$project - to Include Computed Fields:**

  db.collection.aggregate([{ $project : {<field>: <expression>} }]);

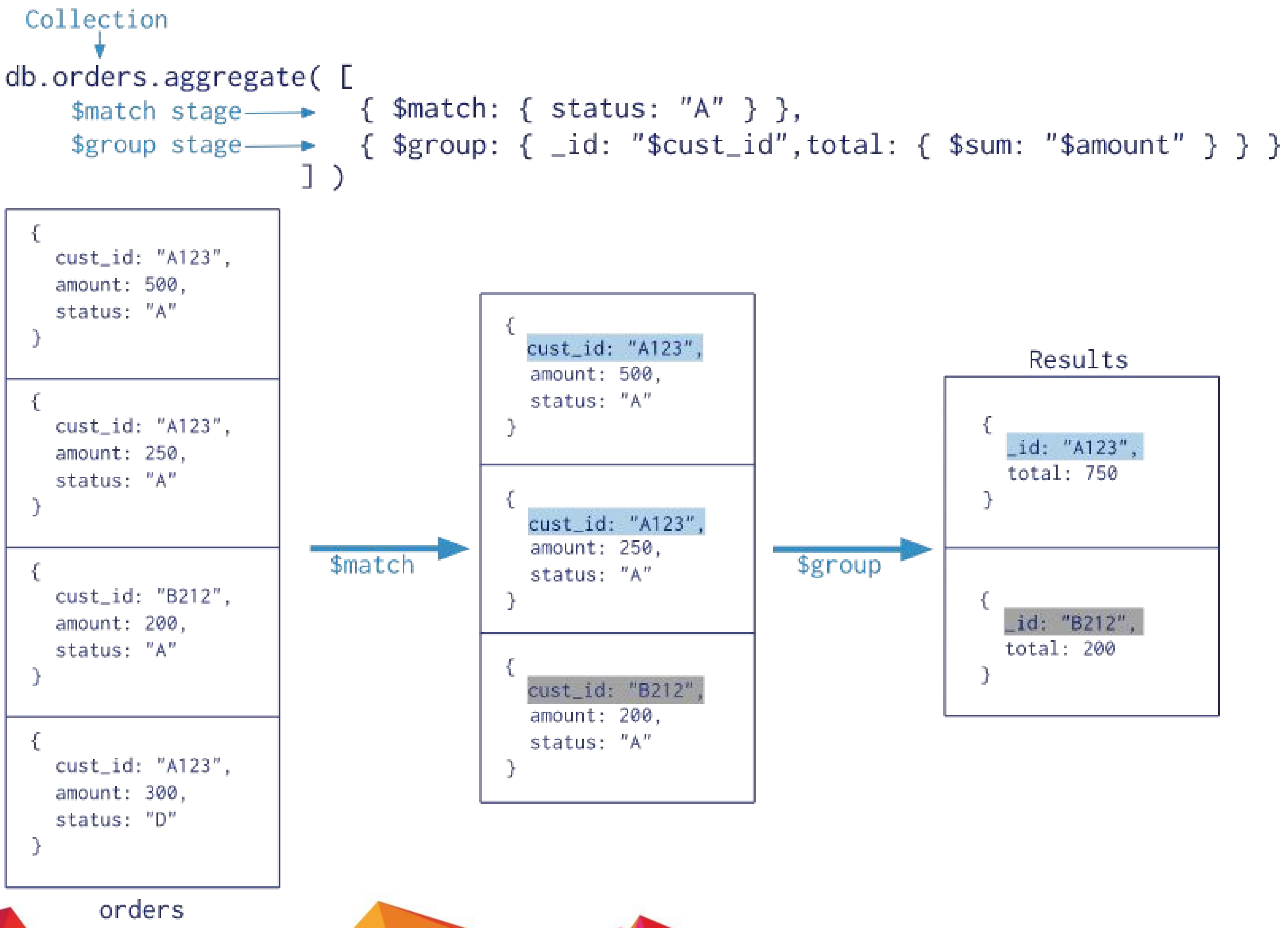
MongoDB Aggregation Stages Operator - $project

The MongoDB $match operator filters the documents to pass only those documents that match the specified condition(s) to the next pipeline stage.

- **$match - to Equality Match**

```
db.collection.aggregate([ { $match : { <query> } } } ])
```

Aggregation example with $match and $group



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

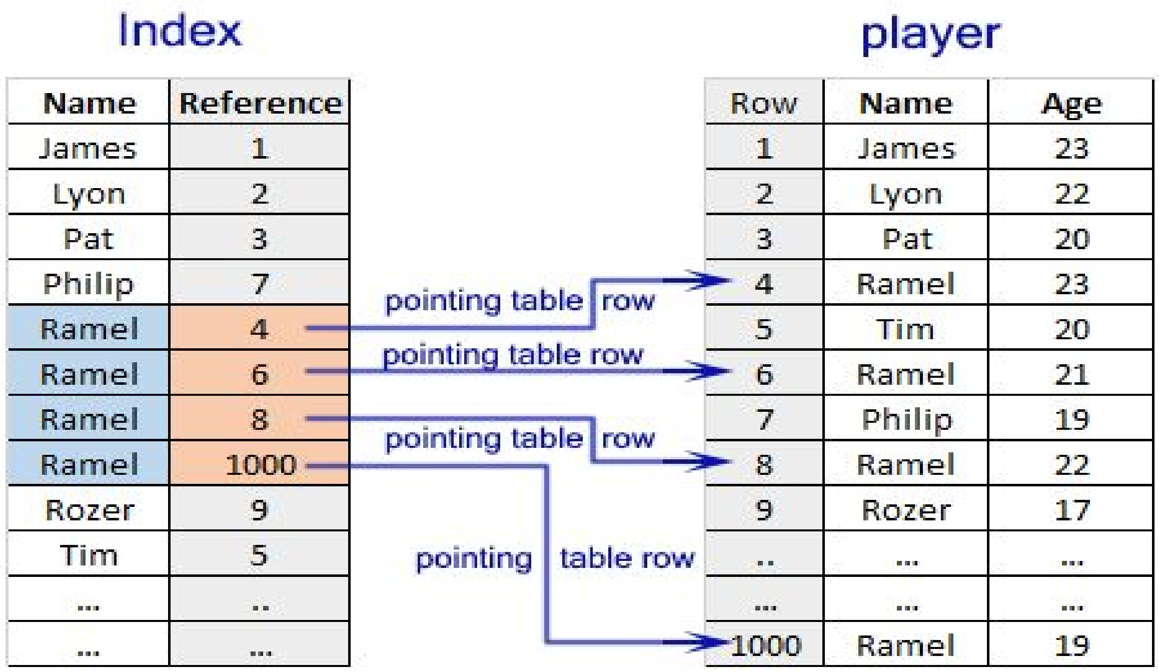
```
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
{
  cust_id: "A123",
  amount: 300,
  status: "D"
}
```

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

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

Indexes

Indexes are special data structures that store a small portion of the collection's data set in an easy to traverse form.



Index

Name	Reference
James	1
Lyon	2
Pat	3
Philip	7
Ramel	4
Ramel	6
Ramel	8
Ramel	1000
Rozer	9
Tim	5
...	..
...	...

player

Row	Name	Age
1	James	23
2	Lyon	22
3	Pat	20
4	Ramel	23
5	Tim	20
6	Ramel	21
7	Philip	19
8	Ramel	22
9	Rozer	17
..	...	...
...	...	...
1000	Ramel	19

Indexes

Default _id Index: MongoDB creates a unique index on the _id field during the creation of a collection. The _id index prevents clients from inserting two documents with the same value for the _id field. You cannot drop this index on the_id field.

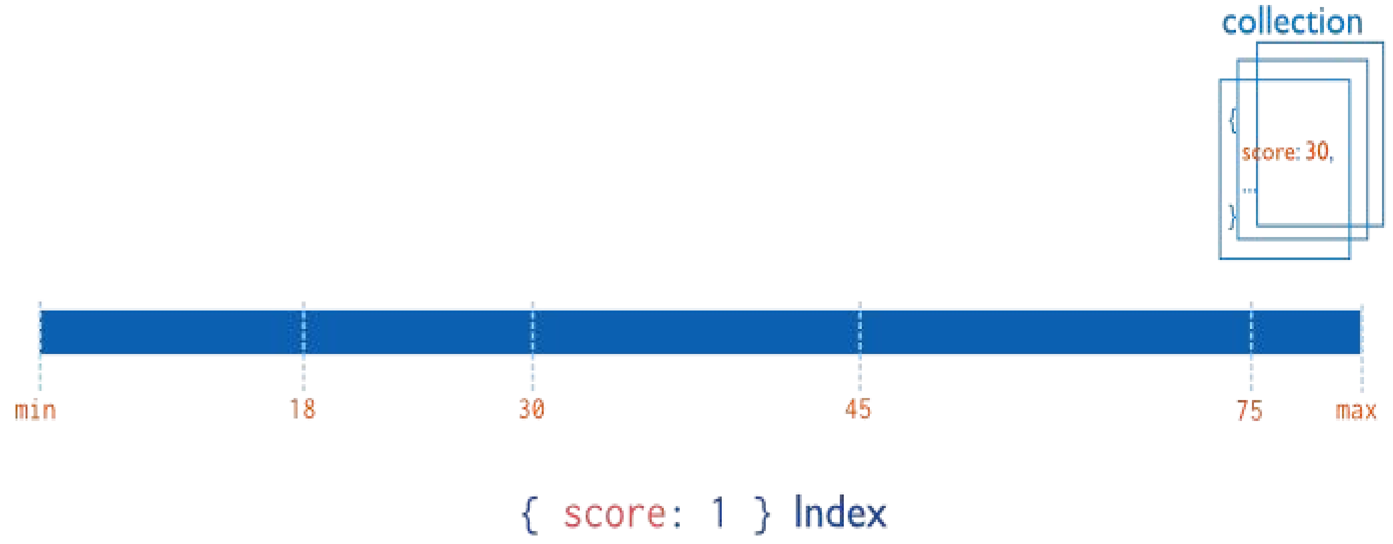
Create an Index:

db.collection.createIndex(<key and index type specification>, <options>)

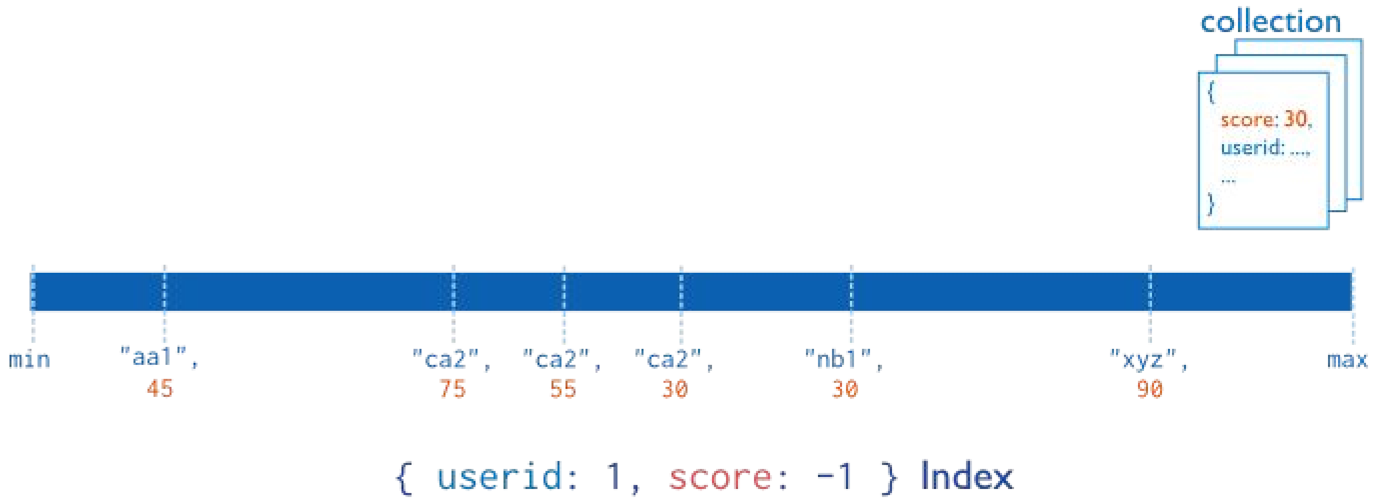
Index Types:

- Single Field
- Compound Index
- Multikey Index
- Text Indexes

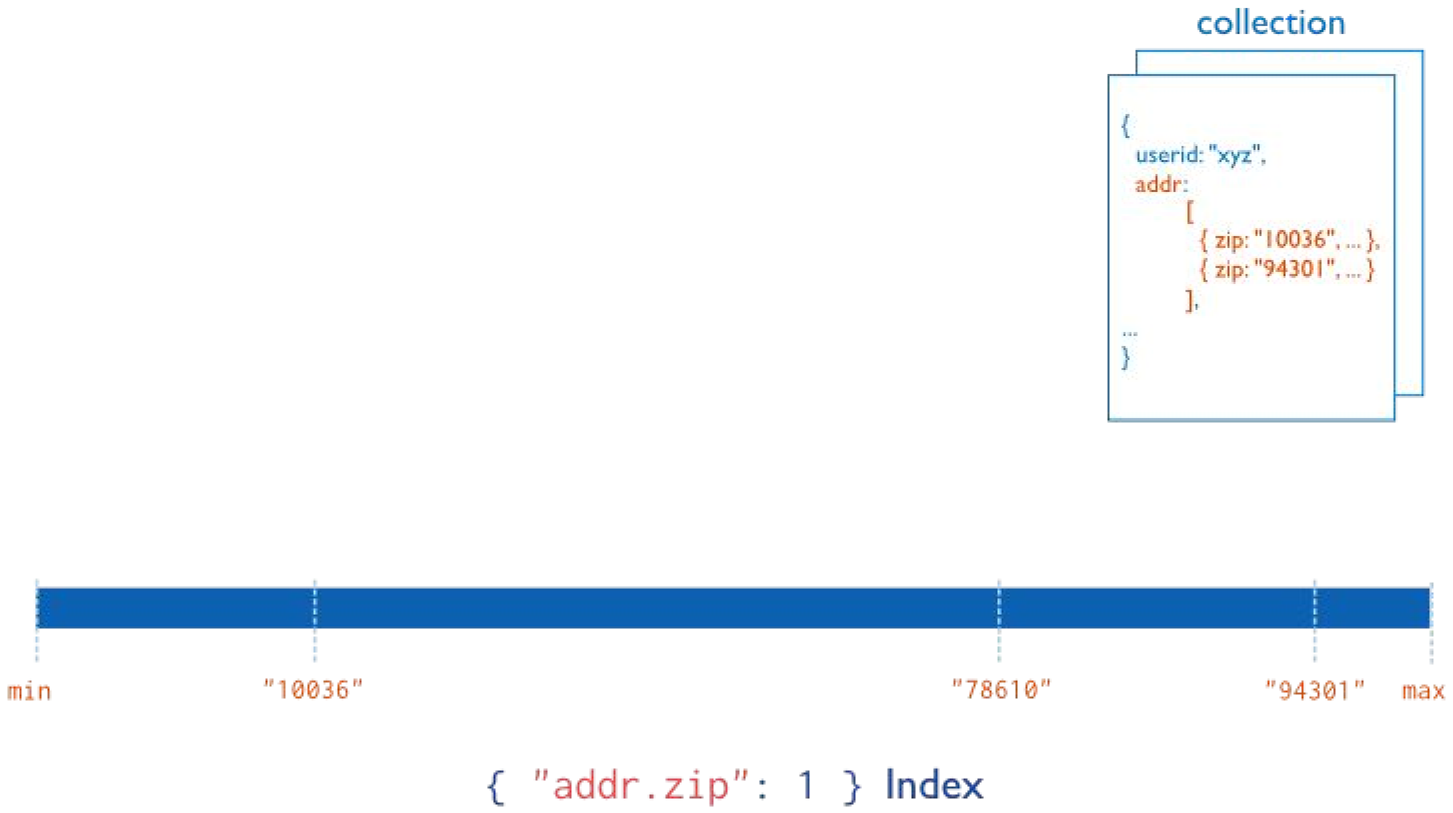
Index on Single Field



Compound Index



Multikey Index



Text Index

Another type of index that MongoDB provides is text index, that supports searching for string content in a collection. These text indexes do not store language-specific stop words (e.g. “the”, “a”, “or”) and restrict the words in a collection to only store root words.

```
db.collection.createIndex({<field>:”text”})
```

Queries on Array field

- Find
- Projection
- Insert: $push, $pushAll, $addToSet
- Update
- Remove

Exercise 1

To Find The user having tweets count and follower count greater that 1000

Excer

Exercise 2

Store the unique page view count and total page view count of Blog.

Exercise 3

Write a query to find user_id where retweet count > 10

Write Equivalent Query in Mongo

```
UPDATE users SET status = 'C'
WHERE age > 25
```

1. Insert Book collection

{title: "book1”, authors:{name:”James”, id:101}, price:500, language:”Hindi”, publish_date:”05/04/2014”}
{title: "book2”, authors:{name:”Abert”, id:102}, price:600, language:”Hindi”, publish_date:”05/05/2012”}
{title: "book3”, authors:{name:”Abert”, id:102}, price:600, language:”English”, publish_date:”05/04/2014”}
{title: "book4”, authors:{name:”Sam”, id:103}, price:1100, language:”English”,publish_date:”05/05/2012”}
{title: "book5”, authors:{name:”James”, id:101}, price:850, language:”English”, publish_date:”05/05/2012”}
{title: "book6”, authors:{name:”Albert”, id:102}, price:1250, language:”English”, publish_date:”05/04/2014”}

2. Inser Student collection shared in email

1. Find students whose zipcode is 63109 and school code is 102
2. Update each student code from 100 to 101 where zipcode is 63109.
3. Insert two more students in students array where zipcode is 63110.
4. Remove an element from students array whose age is greater than 25
5. Find out the total price of books group by their authors and language
6. Group by author name and then by language and display total_price and no of books published on the publish date “05/04/2014”
7. Display only blog title and comments in the output document using aggregate
8. Create compound index on price and author's name and show created index list
9. Create text index on books title and text search for any book title

Thank You!!