Speed It Up! Building Efficient Indexes in MongoDB

Naveen Kumar
Senior DevOps Engineer, O9 Solutions
MongoDB User Group Leader - Bangalore

Agenda

Why indexes matter

Single vs. compound indexes

ESR Rule

Measuring index usage

Dropping index impact

Other types of indexes

Meet the Hungry Customer

Let’s hunt down the best spots for Biryani

Quick Check on Basics

```
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.students.find().pretty()
[
  {
    _id: ObjectId('687a4c6963c02df13b718dc4'),
    name: 'Alice',
    age: 22,
    grade: 'A',
    city: 'Chennai'
  },
  {
    _id: ObjectId('687a4c6963c02df13b718dc5'),
    name: 'Bob',
    age: 24,
    grade: 'B',
    city: 'Mumbai'
  },
```

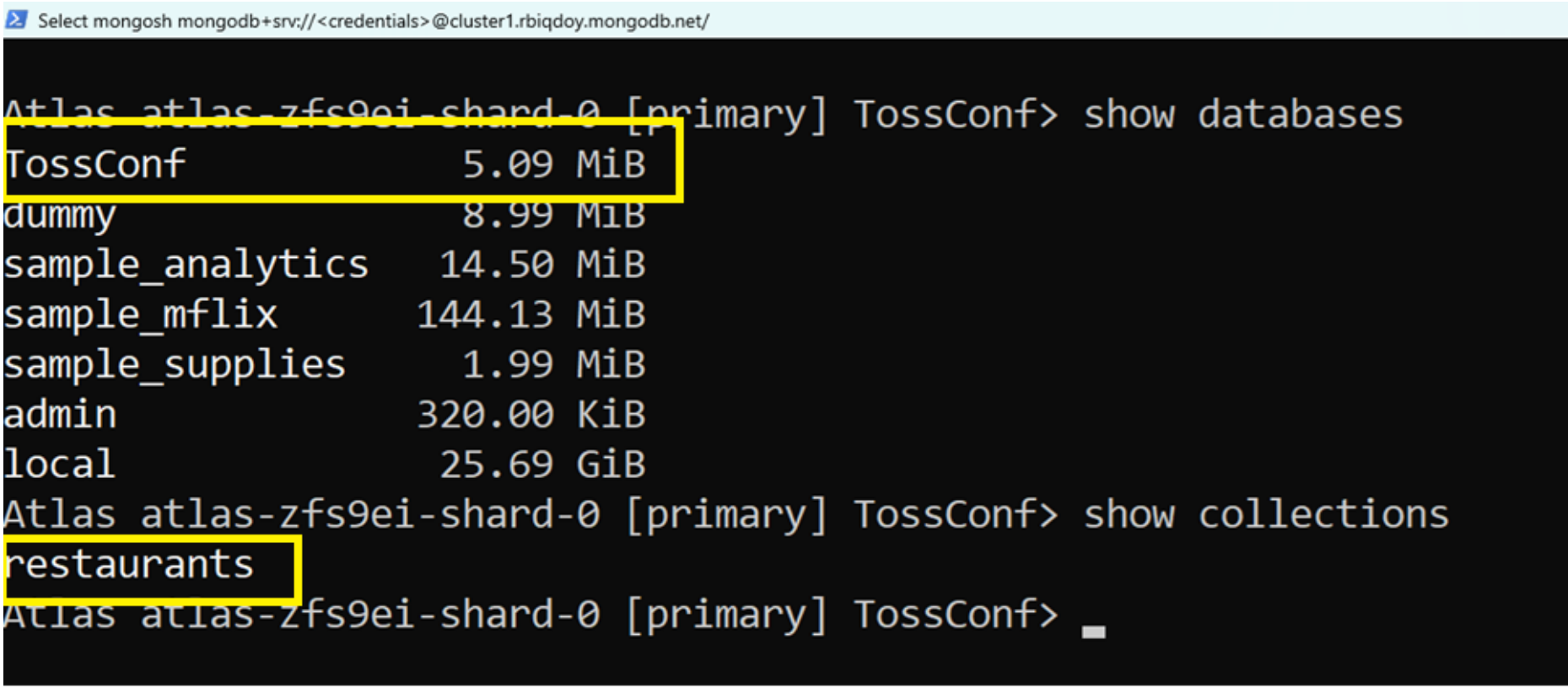
Quick Check on Basics (Continued)

```
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.students.find({ city: "Chennai" })
[
  {
    _id: ObjectId('687a4c6963c02df13b718dc4'),
    name: 'Alice',
    age: 22,
    grade: 'A',
    city: 'Chennai'
  },
  {
    _id: ObjectId('687a4c6963c02df13b718dc6'),
    name: 'Charlie',
    age: 23,
    grade: 'A',
    city: 'Chennai'
  },
```

Quick Check on Basics (Continued)

```
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.students.find({},{ name: 1, grade: 1, _id: 0 })
[
  { name: 'Alice', grade: 'A' },
  { name: 'Bob', grade: 'B' },
  { name: 'Charlie', grade: 'A' },
  { name: 'David', grade: 'C' },
  { name: 'Eva', grade: 'B' }
]
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.students.find({city: "Chennai" },{ name: 1, _id: 0 })
[ { name: 'Alice' }, { name: 'Charlie' }, { name: 'Eva' } ]
```

Exploring Our Dataset



Exploring Our Dataset (Continued)

```
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.restaurants.find().limit(2).pretty()
[
  {
    _id: ObjectId('6870afee2e4aeb9b57840bc2'),
    'Restaurant Name': 'KFC',
    Category: [ 'American', 'Snacks', 'Biryani' ],
    Rating: 3.9,
    'Cost for two': 400,
    Veg: false,
    city: 'Delhi',
    Area: 'Paharganj',
    Locality: 'KFC Paharganj',
    Address: 'KFC Restaurant 2154, Desh Bandhu Gupta Road, Near Paharganj Police Station Front of Bank of Baroda, Bazar Sangatrashan, Ch
    'Long Distance Delivery': 0
  },
  {
    _id: ObjectId('6870afee2e4aeb9b57840bc3'),
    'Restaurant Name': "McDonald's",
    Category: [ 'American' ],
    Rating: 4.3,
    'Cost for two': 400,
    Veg: false,
    city: 'Delhi',
    Area: 'Kashmere Gate',
    Locality: 'Delhi ISBT DMRC (GF)',
    Address: 'Delhi ISBT DMRC,ISBT DMRC Railway Station. Kashmere Gate, Delhi- 110006',
    'Long Distance Delivery': 0
  }
]
Atlas atlas-zfs9ei-shard-0 [primary] TossConf>
```

Crafting the Query

city: ‘Chennai'

Category: "Biryani"

Cost: 400 - 800

Give me the top-rated restaurants

Code for the Use Case

```
db.restaurants.find({
    city: 'Chennai',
    Category: 'Biryani',
    "Cost for two": { $gte: 400, $lte: 800 }
  }).sort({ Rating: -1 })
```

Unlocking Index Power

```
1 db.restaurants.find({city: 'Chennai'}).explain('executionStats');
```

Reviewing the Explain Plan


```
winningPlan: {
    isCached: false,
    stage: 'COLLSCAN',
    filter: { city: { '$eq': 'Chennai' } },
    direction: 'forward'
  },
  rejectedPlans: []
},
executionStats: {
  executionSuccess: true,
  nReturned: 374,
  executionTimeMillis: 17,
  totalKeysExamined: 0,
  totalDocsExamined: 31804,
```

Creating a Single-Field Index


```
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.restaurants.createIndex({ City: 1 })
City_1
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.restaurants.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { City: 1 }, name: 'City_1' }
]
```

Revisiting the Explain Plan

```
winningPlan: {
    isCached: false,
    stage: 'FETCH',
    inputStage: {
      stage: 'IXSCAN',
      keyPattern: { city: 1 },
      indexName: 'city_1',
      isMultiKey: false,
      multiKeyPaths: { city: [] },
      isUnique: false,
```

Revisiting the Explain Plan (Continued)

executionStats: {
executionSuccess: true,
nReturned: 374,
executionTimeMillis: 0,
totalKeysExamined: 374,
totalDocsExamined: 374,

Adding a New Filter to the Query

```
db.restaurants.find({
    city: 'Chennai',
    Category: 'Biryani'
  }).explain('executionStats');
```

Can we make use of the city_1 index?

```
winningPlan: {
    isCached: false,
    stage: 'FETCH',
    filter: { Category: { '$eq': 'Biryani' } },
    inputStage: {
      stage: 'IXSCAN',
      keyPattern: { city: 1 },
      indexName: 'city_1',
```

What’s the Problem Now?

```
executionStats: {
    executionSuccess: true,
    nReturned: 59,
    executionTimeMillis: 0,
    totalKeysExamined: 374,
    totalDocsExamined: 374,
```

Let’s Try Indexing on Category

Understanding Compound Indexes

- Index on multiple fields
- Support queries that match on the prefix of the index field
- Max Fields: Up to 32 fields per compound index

Creating a Compound Index

db.restaurants.createIndex({Category: 1 , City: 1})

```
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.restaurants.createIndex({Category: 1 , City: 1})
Category_1_City_1
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.restaurants.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { city: 1 }, name: 'city_1' },
  { v: 2, key: { Category: 1, City: 1 }, name: 'Category_1_City_1' }
]
```

Re-running the Query with the New Index

Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.restaurants.find({ city: “Chennai”, Category: “Biryani”})

executionStats: {
executionSuccess: true,
nReturned: 59,
executionTimeMillis: 1,
totalKeysExamined: 374,
totalDocsExamined: 374,

Create Index Script -> **db.restaurants.createIndex({Category: 1 , City: 1})**

Updated Version -> db.restaurants.createIndex({Category: 1 , city: 1})

Refining the Index

Atlas atlas-zfs9ei-shard-0 [primary] TossConf> **db.restaurants.createIndex({Category: 1 , city: 1})**
Category_1_city_1
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.restaurants.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { city: 1 }, name: 'city_1' },
  { v: 2, key: { Category: 1, City: 1 }, name: 'Category_1_City_1' },
  { v: 2, key: { Category: 1, city: 1 }, name: 'Category_1_city_1' }
]

executionStats: {
    executionSuccess: true,
    nReturned: 59,
    executionTimeMillis: 1,
    totalKeysExamined: 59,
    totalDocsExamined: 59,

Reviewing the Final Query

db.restaurants.find({city: “Chennai”,Category: “Biryani”,"Cost for two": { $gte: 400, $lte: 800 }}).sort({ Rating: -1 })

```
db.restaurants.find({
  city: 'Chennai',
  Category: 'Biryani',
  "Cost for two": { $gte: 400, $lte: 800 }
}).sort({ Rating: -1 })
```

Building a Compound Index for Our Query

Query

```
db.restaurants.find({city: 'Chennai',Category: 'Biryani',"Cost for two": { $gte: 400, $lte: 800 }}).sort({ Rating: -1 })
```

Index

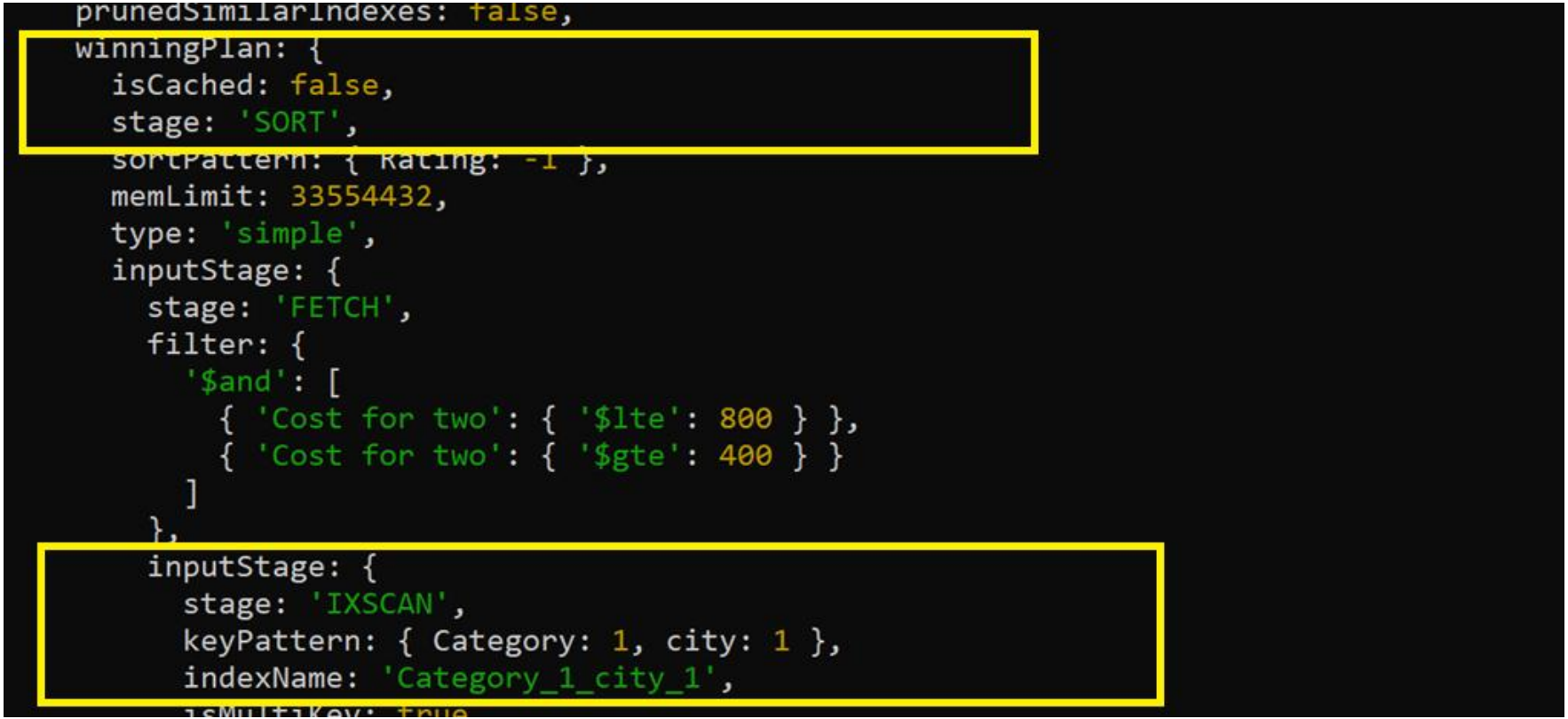
```
db.restaurants.createIndex({Rating:1,"Cost for two":1, city:1, Category:1})
```

New Index Created

Atlas atlas-zfs9ei-shard-0 [primary] TossConf> **db.restaurants.createIndex({Rating:1,"Cost for two":1, city:1, Category:1})**
Rating_1_Cost for two_1_city_1_Category_1
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> **db.restaurants.getIndexes()**

```
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { city: 1 }, name: 'city_1' },
  { v: 2, key: { Category: 1, City: 1 }, name: 'Category_1_City_1' },
  { v: 2, key: { Category: 1, city: 1 }, name: 'Category_1_city_1' },
  {
    v: 2,
    key: { Rating: 1, 'Cost for two': 1, city: 1, Category: 1 },
    name: 'Rating_1_Cost for two_1_city_1_Category_1'
  }
]
```

Revisiting the explain plan



ESR Rule

E -> Equality
S -> Sort
R -> Range

db.restaurants.find({**city: 'Chennai',Category: 'Biryani'**,**"Cost for two": { $gte: 400, $lte: 800 }**}).**sort({ Rating: -1 })**

Index Redesign
db.restaurants.createIndex({**city:1, Category:1**, Rating:1,"Cost for two":1})
db.restaurants.createIndex({**Category:1, city:1**, Rating:1,"Cost for two":1})

Index Prefix Compression

Cardinality refers to the uniqueness of values in a column (or field) in a database table

Index with low-cardinality prefix will be smaller in size.

For a single field index, you should chose high-cardinality prefix

Prefix and Supported Queries

Index Definition -> db.collectionname.createIndex({“a”:1,”b”:1,”c”:1})

Supported Queries

db.collectionname.find({“a”:10})

db.collectionname.find({“a”:12,”b”:20})

db.collectionname.find({“a”:5,”b”:4,”c”:6})

Unsupported Queries

db.collectionname.find({“b”:22})

db.collectionname.find({“b”:33,”c”:4})

db.collectionname.find({”c”:12})

Redesigned Index: Creation

```
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.restaurants.createIndex({city:1, Category:1, Rating:1,"Cost for two":1})
city_1_Category_1_Rating_1_Cost for two_1
Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.restaurants.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { city: 1 }, name: 'city_1' },
  { v: 2, key: { Category: 1, City: 1 }, name: 'Category_1_City_1' },
  { v: 2, key: { Category: 1, city: 1 }, name: 'Category_1_city_1' },
  {
    v: 2,
    key: { Rating: 1, 'Cost for two': 1, city: 1, Category: 1 },
    name: 'Rating_1_Cost for two_1_city_1_Category_1'
  },
  {
    v: 2,
    key: { city: 1, Category: 1, Rating: 1, 'Cost for two': 1 },
    name: 'city_1_Category_1_Rating_1_Cost for two_1'
  }
]
```

Explain Plan Output

```
winningPlan: {
    isCached: true,
    stage: 'FETCH',
    inputStage: {
      stage: 'IXSCAN',
      keyPattern: { city: 1, Category: 1, Rating: 1, 'Cost for two': 1 },
      indexName: 'city_1_Category_1_Rating_1_Cost for two_1',
      isMultiKey: true,
      multiKeyPaths: {
        city: [],
        Category: [ 'Category' ],
        Rating: [],
        'Cost for two': []
      },
```

Explain Plan Output (Continued)

executionStats: {
executionSuccess: true,
nReturned: 19,
executionTimeMillis: 0,
totalKeysExamined: 31,
totalDocsExamined: 19,

Measure Index Usage

Atlas atlas-zfs9ei-shard-0 [primary] TossConf> db.restaurants.aggregate([{ $indexStats: {} }])

```
{
  name: 'Category_1_City_1',
  key: { Category: 1, City: 1 },
  accesses: { ops: Long('0'), since: ISODate('2025-07-14T07:47:08.137Z') },
  host: 'ac-je7r8g7-shard-00-01.rbiqdoy.mongodb.net:27017'
},
{
  name: 'Category_1_city_1',
  key: { Category: 1, city: 1 },
  accesses: { ops: Long('4'), since: ISODate('2025-07-14T09:11:26.944Z') },
  host: 'ac-je7r8g7-shard-00-01.rbiqdoy.mongodb.net:27017'
},
{
  name: 'city_1_Category_1_Rating_1_Cost for two_1',
  key: { city: 1, Category: 1, Rating: 1, 'Cost for two': 1 },
  accesses: { ops: Long('7'), since: ISODate('2025-07-15T14:44:17.285Z') },
  host: 'ac-je7r8g7-shard-00-01.rbiqdoy.mongodb.net:27017'
},
{
  name: 'Rating_1_Cost for two_1_city_1_Category_1',
  key: { Rating: 1, 'Cost for two': 1, city: 1, Category: 1 },
  accesses: { ops: Long('0'), since: ISODate('2025-07-15T13:58:02.200Z') },
  host: 'ac-je7r8g7-shard-00-01.rbiqdoy.mongodb.net:27017'
}
```

Measure Index Usage (Continued)

Index Name	Fields in Index Key	Ops (Accesses)
_id_	{ _id: 1 }	0
city_1	{ city: 1 }	1
Category_1_City_1	{ Category: 1, City: 1 }	0
Category_1_city_1	{ Category: 1, city: 1 }	4
city_1_Category_1_Rating_1_Cost for two_1	{ city: 1, Category: 1, Rating: 1, 'Cost for two': 1 }	7
Rating_1_Cost for two_1_city_1_Category_1	{ Rating: 1, 'Cost for two': 1, city: 1, Category: 1 }	0

Impact of Dropping an Index

- Slower Query Performance (If a query was using the index).
- Recreation of the index will also be a resource bottleneck.
- Impact on Sorting
- Query Plan Cache Invalidation

Before You Drop an Index

- Hide the index
- Monitor queries and performance
- If no degradation after a few days/weeks → safe to drop
- If things slow down → unhide it instantly

Other Index Types

- Multikey Index
- Geospatial Index
- Text Index
- Hashed Index
- Wildcard Index
- Other key parameters (Partial, Unique, TTL, Hidden, Sparse)

Resources

- MongoDB Docs: Indexes
- MongoDB Docs: createIndex()
- MongoDB Docs: Unique Indexes
- MongoDB Docs: Measure Index Use
- MongoDB Docs: getIndexes()
- MongoDB Docs: Multikey Indexes
- MongoDB Docs: Compound Indexes
- MongoDB Docs: Indexing Strategies

Connect with me

Email: contactnaveent@gmail.com
LinkedIn: https://www.linkedin.com/in/tnaveen-kumar/