INDEXING USING Mongodb

Name : Arpit Chauhan

Roll No : 40

BEFORE INDEXING

```
arpitt> db.arp.find({Location:'Bhuj'}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'arpitt.arp',
    indexFilterSet: false,
    parsedQuery: { Location: { '$eq': 'Bhuj' } },
    queryHash: '2A2D6A2C',
    planCacheKey: '2A2D6A2C',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'COLLSCAN',
      filter: { Location: { '$eq': 'Bhuj' } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 22027,
    executionTimeMillis: 313,
    totalKeysExamined: 0,
    totalDocsExamined: 1004480,
    executionStages: {
      stage: 'COLLSCAN',
```

AFTER INDEXING

```
arpitt> db.arp.createIndex({Location:1})
Location_1
arpitt> db.arp.find({Location:'Bhuj'}).explain("executionStats")
{
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 22027,
  executionTimeMillis: 27,
  totalKeysExamined: 22027,
  totalDocsExamined: 22027,
  executionStages: {
```

<u>Key Differences:</u>

Winning Plan:

- Before Indexing: COLLSCAN (Collection Scan)
- After Indexing: IXSCAN (Index Scan)

Execution Time:

- Before Indexing: 313 milliseconds
- After Indexing: 27 milliseconds

Total Keys Examined:

- Before Indexing: 0 (No index was used, so no keys were examined)
- After Indexing: 22,027

Total Docs Examined:

- Before Indexing: 1,004,480
- After Indexing: 22,027

Overall, indexing significantly improved the query performance by reducing the execution time and the number of documents examined. It efficiently utilized the index to find relevant documents, leading to faster query execution.

To find existing indexes in MongoDB, you can use the getIndexes() method on a collection. Here's how you can do it

```
arpitt> db.arp.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { Location: 1 }, name: 'Location_1' }
]
```

To drop indexes in a collection in MongoDB, you can use the dropIndex() method. Here's how you can do it:

```
arpitt> db.arp.dropIndex("Location_1")
{ nIndexesWas: 2, ok: 1 }
```

If you want to drop all indexes except the default index on the _id field, you can use the dropIndexes() method:

```
arpitt> db.arp.dropIndex("Location_1")
{ nIndexesWas: 2, ok: 1 }
arpitt> db.arp.dropIndexes()
{
  nIndexesWas: 1,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
```