

Ex.No:6 Indexing with suitable example using MongoDB

NAME = DHEERAJ VERMA

ROLL NO = 22BCS025

AIM: To **optimize query performance** by enabling the database to retrieve data more efficiently.

```
Command Prompt
2024-08-07T11:22:17.509+0530 Failed: invalid JSON input. Position: 20. Character: N
2024-08-07T11:22:17.509+0530 0 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\EMCLAB\Downloads>mongoimport --db=db2 --collection=person --file=persons2.json --jsonArray
2024-08-07T11:23:59.830+0530 connected to: mongodb://localhost/
2024-08-07T11:23:59.836+0530 2 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\EMCLAB\Downloads>mongoimport --db=db2 --collection=person --file=persons.json --jsonArray
2024-08-07T11:24:52.364+0530 connected to: mongodb://localhost/
2024-08-07T11:24:52.366+0530 Failed: invalid JSON input. Position: 18. Character: N
2024-08-07T11:24:52.366+0530 0 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\EMCLAB\Downloads>mongoimport --db=db2 --collection=person --file=persons2.json --jsonArray
2024-08-07T11:26:22.739+0530 connected to: mongodb://localhost/
2024-08-07T11:26:22.742+0530 2 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\EMCLAB\Downloads>mongoimport --db=db2 --collection=person --file=persons2.json --jsonArray
2024-08-07T11:26:57.298+0530 connected to: mongodb://localhost/
2024-08-07T11:26:57.300+0530 Failed: invalid JSON input. Position: 20. Character: N
2024-08-07T11:26:57.301+0530 0 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\EMCLAB\Downloads>mongoimport --db=db2 --collection=person --file=persons2.json --jsonArray --legacy
2024-08-07T11:27:46.298+0530 connected to: mongodb://localhost/
2024-08-07T11:27:46.304+0530 2 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\EMCLAB\Downloads>mongoimport --db=db2 --collection=person --file=persons.json --jsonArray --legacy
2024-08-07T11:28:04.013+0530 connected to: mongodb://localhost/
2024-08-07T11:28:04.045+0530 1000 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\EMCLAB\Downloads>
```

```
Command Prompt - mongo
}
{
  "_id" : ObjectId("66b30cecec984aa777e69871"),
  "index" : 19,
  "name" : "Mcguire Vincent",
  "isActive" : false,
  "registered" : ISODate("2017-01-28T01:53:06Z"),
  "age" : 24,
  "gender" : "male",
  "eyeColor" : "green",
  "favoriteFruit" : "strawberry",
  "company" : {
    "title" : "ELENTRIX",
    "email" : "mcguirevincent@elentrix.com",
    "phone" : "+1 (924) 572-3321",
    "location" : {
      "country" : "USA",
      "address" : "347 Sandford Street"
    }
  },
  "tags" : [
    "do",
    "aute",
    "incidunt"
  ]
}
Type "it" for more
> db.person.find().count()
1000
>
```

```
Command Prompt - mongo
> db.person.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
>
```

"v" : 2:

This field denotes the version of the index. MongoDB uses different versions to represent changes and improvements in index functionality. Version 2 is quite common and indicates a standard index version.

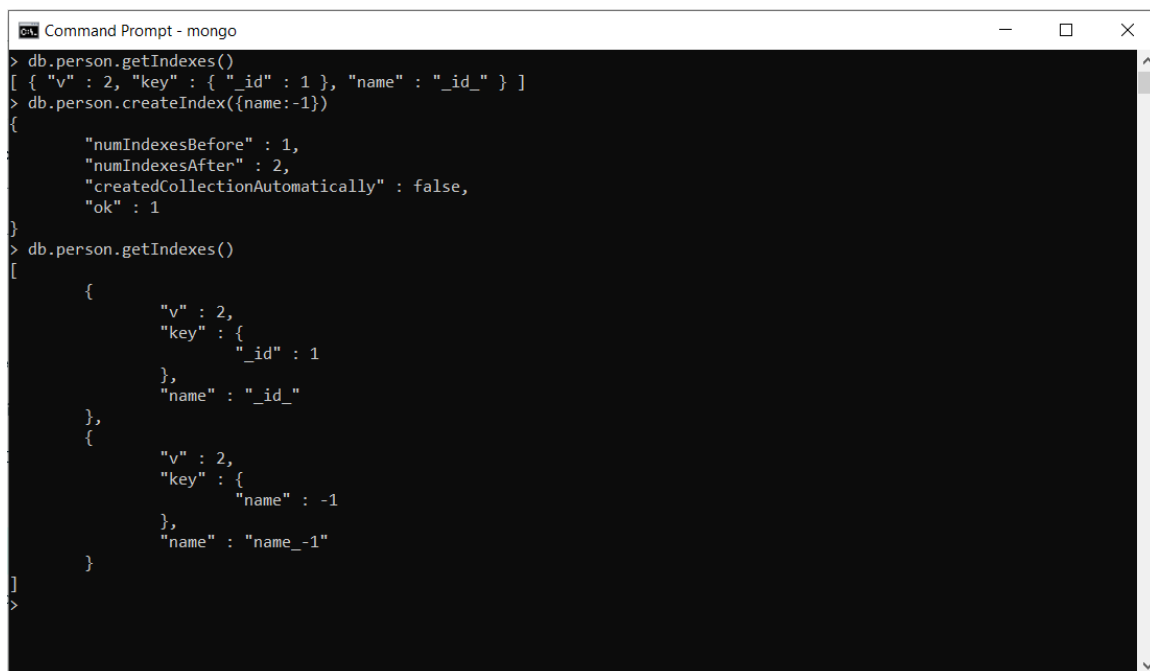
"key" : { "_id" : 1 }:

This field describes the key pattern of the index. Here, "_id" : 1 indicates that the index is on the _id field of the documents in the collection. The 1 signifies that the index is in ascending order. If it were -1, it would be in descending order.

The _id field is a unique identifier for each document in a MongoDB collection. By default, MongoDB creates a unique index on the _id field, ensuring that each document has a unique _id value.

"name" : "_id_":

This field shows the name of the index. MongoDB automatically names the index on the _id field as _id_. This name is generated by MongoDB to help identify and manage the index.



```
Command Prompt - mongo
> db.person.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
> db.person.createIndex({name:-1})
{
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
> db.person.getIndexes()
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "name" : -1
    },
    "name" : "name_-1"
  }
]
```

```
Command Prompt - mongo
> db.person.update({},{$rename:{"age":"personAge"}},false,true)
WriteResult({ "nMatched" : 1000, "nUpserted" : 0, "nModified" : 999 })
> db.person.createIndex({personAge:1})
{
  "numIndexesBefore" : 2,
  "numIndexesAfter" : 3,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
> db.person.getIndexes()
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "name" : -1
    },
    "name" : "name_-1"
  },
  {
    "v" : 2,
    "key" : {
      "personAge" : 1
    },
    "name" : "personAge_1"
  }
]
```

```
Command Prompt - mongo
> db.person.createIndex({index:1},{unique:true})
{
  "numIndexesBefore" : 3,
  "numIndexesAfter" : 4,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
> db.person.getIndexes()
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "name" : -1
    },
    "name" : "name_-1"
  },
  {
    "v" : 2,
    "key" : {
      "personAge" : 1
    },
    "name" : "personAge_1"
  },
  {
    "v" : 2,
    "key" : {
      "index" : 1
    },
    "name" : "index_1",
    "unique" : true
  }
]
```

Command Prompt - mongo

```
> db.person.createIndex({name:1,eyeColor:1})
{
  "numIndexesBefore" : 4,
  "numIndexesAfter" : 5,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
> db.person.getIndexes()
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "name" : -1
    },
    "name" : "name_-1"
  },
  {
    "v" : 2,
    "key" : {
      "personAge" : 1
    },
    "name" : "personAge_1"
  },
  {
    "v" : 2,
    "key" : {
      "index" : 1
    },
    "name" : "index_1",
    "unique" : true
  },
  {
    "v" : 2,
    "key" : {
      "name" : 1,
      "eyeColor" : 1
    },
    "name" : "name_1_eyeColor_1"
  }
]
```

```
Command Prompt - mongo
> db.person.explain().find({age:{ $gt:25}})
{
  "explainVersion" : "1",
  "queryPlanner" : {
    "namespace" : "db2.person",
    "indexFilterSet" : false,
    "parsedQuery" : {
      "age" : {
        "$gt" : 25
      }
    },
    "queryHash" : "48B283C4",
    "planCacheKey" : "5C3F837A",
    "maxIndexedOrSolutionsReached" : false,
    "maxIndexedAndSolutionsReached" : false,
    "maxScansToExplodeReached" : false,
    "winningPlan" : {
      "stage" : "COLLSCAN",
      "filter" : {
        "age" : {
          "$gt" : 25
        }
      },
      "direction" : "forward"
    },
    "rejectedPlans" : [ ]
  },
  "command" : {
    "find" : "person",
    "filter" : {
      "age" : {
        "$gt" : 25
      }
    },
    "$db" : "db2"
  },
  "serverInfo" : {
    "host" : "EMC02",
    "port" : 27017,
    "version" : "5.0.27",
    "gitVersion" : "49571988f1fea870e803f71a3ef8417173f3fbb1"
  },
  "serverParameters" : {
    "internalQueryFacetBufferSizeBytes" : 104857600,
    "internalQueryFacetMaxOutputDocSizeBytes" : 104857600,
    "internalLookupStageIntermediateDocumentMaxSizeBytes" : 104857600,
    "internalDocumentSourceGroupMaxMemoryBytes" : 104857600,
    "internalQueryMaxBlockingSortMemoryUsageBytes" : 104857600,
    "internalQueryProhibitBlockingMergeOnMongoS" : 0,
    "internalQueryMaxAddToSetBytes" : 104857600,
    "internalDocumentSourceSetWindowFieldsMaxMemoryBytes" : 104857600
  },
  "ok" : 1
}
```

```
Command Prompt - mongo

> db.person.createIndex({age:1},{partialFilterExpression:{age:{$gt:25}}})
{
  "numIndexesBefore" : 5,
  "numIndexesAfter" : 6,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
> db.person.getIndexes()
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "name" : -1
    },
    "name" : "name_-1"
  },
  {
    "v" : 2,
    "key" : {
      "personAge" : 1
    },
    "name" : "personAge_1"
  },
  {
    "v" : 2,
    "key" : {
      "index" : 1
    },
    "name" : "index_1",
    "unique" : true
  },
  {
    "v" : 2,
    "key" : {
      "name" : 1,
      "eyeColor" : 1
    },
    "name" : "name_1_eyeColor_1"
  },
  {
    "v" : 2,
    "key" : {
      "age" : 1
    },
    "name" : "age_1",
    "partialFilterExpression" : {
      "age" : {
        "$gt" : 25
      }
    }
  }
]
```

In MongoDB, partial indexes are a feature that allows you to create indexes that only apply to a subset of documents in a collection. This can be useful for optimizing performance and saving storage space, especially when our queries only need to filter on specific conditions.

`{ age: 1 }:`

This specifies the key pattern for the index. In this case, the index will be on the age field in ascending order (1).

`{ partialFilterExpression: { age: { $gt: 25 } } }:`

This specifies a partial filter expression that defines which documents the index should cover. Here, the index will only include documents where the age field is greater than 25.

Components of the Output:

`"v" : 2:`

The version of the index. Version 2 is a common index version, indicating a standard index with partial filtering capabilities.

```
"key" : { "age" : 1 }:
```

This specifies the index is on the age field in ascending order.

```
"name" : "age_1":
```

The name of the index. MongoDB automatically names the index based on the indexed fields and their sort order. Here, age_1 signifies an ascending index on the age field.

```
"partialFilterExpression" : { "age" : { "$gt" : 25 } }:
```

This specifies the partial filter expression used to create the index. It means that only documents with an age value greater than 25 will be included in this index.

Output Explanation of explain():

```
explainVersion : "1":
```

Indicates the version of the explain output format.

```
queryPlanner:
```

This section describes the query plan used by MongoDB.

```
namespace : "db2.person":
```

Specifies the database and collection being queried.

```
indexFilterSet : false`:
```

Indicates whether any index filters were applied. In this case, no filters were applied.

```
parsedQuery:
```

Shows the parsed query filter. Here, it's { "age" : { "\$gt" : 25 } }, meaning the query is looking for documents where age is greater than 25.

```
queryHash and planCacheKey:
```

These are internal identifiers used for caching and optimizing query execution.

```
maxIndexedOrSolutionsReached:
```


Indicates whether the query planner reached the maximum number of possible indexed OR solutions. This was not reached in this case.

maxIndexedAndSolutionsReached:

Indicates whether the query planner reached the maximum number of possible indexed AND solutions. This was not reached in this case.

maxScansToExplodeReached:

Indicates whether the query planner reached the maximum number of index scans allowed. This was not reached in this case.

winningPlan:

This section describes the plan MongoDB chose to execute the query.

stage : "FETCH":

The query planner decided to perform a FETCH stage, which retrieves documents that match the query condition from the index scan results.

inputStage:

This describes the index scan stage used as input to the fetch operation.

stage : "IXSCAN":

Indicates that MongoDB will perform an index scan on the age index.

keyPattern:

{ "age" : 1 } specifies the index is on the age field in ascending order.

indexName : "age_1":

The name of the index being used is age_1.

isMultiKey : false`:

Indicates whether the index is a multikey index (indexes arrays). Here, it is not.

multiKeyPaths:

Shows the paths for multi-key indexes. Since this is not a multi-key index, it's empty.

isUnique:

Indicates whether the index enforces uniqueness. It does not in this case.

isSparse:

Indicates whether the index is sparse (only includes documents with the indexed field). This index is not sparse.

isPartial : true`:

Indicates that this is a partial index and only includes documents where age is greater than 25.

indexVersion : 2`:

The version of the index, as previously discussed.

direction : "forward"``:

Indicates the direction of the index scan. Here, it is in ascending order.

indexBounds:

Shows the bounds of the index scan. Here, age is scanned from (25.0, inf.0], meaning values greater than 25.

rejectedPlans:

Lists any query plans that were considered but rejected. In this case, there are no rejected plans.

command:

Shows the actual command that was explained. In this case, it is a find operation on the person collection with a filter where age is greater than 25.

serverInfo:

Provides information about the MongoDB server that executed the query, including the host, port, and version.

serverParameters:

Lists various server parameters related to query execution and memory usage.

ok:

Indicates whether the explain command was successful. A value of 1 means it was successful.

```
Command Prompt - mongo
> db.person.find({$and:[{personAge:{>25}},{personAge:{<40}}]}).count()
654
> db.person.find({$and:[{personAge:{>25}},{personAge:{<40}}]}).explain("executionStats")
{
  "explainVersion" : "1",
  "queryPlanner" : {
    "namespace" : "db2.person",
    "indexFilterSet" : false,
    "parsedQuery" : {
      "$and" : [
        {
          "personAge" : {
            "$lt" : 40
          }
        },
        {
          "personAge" : {
            "$gt" : 25
          }
        }
      ]
    },
    "maxIndexedOrSolutionsReached" : false,
    "maxIndexedAndSolutionsReached" : false,
    "maxScansToExplodeReached" : false,
    "winningPlan" : {
      "stage" : "FETCH",
      "inputStage" : {
        "stage" : "IXSCAN",
        "keyPattern" : {
          "personAge" : 1
        },
        "indexName" : "personAge_1",
        "isMultiKey" : false,
        "multiKeyPaths" : {
          "personAge" : [ ]
        },
        "isUnique" : false,
        "isSparse" : false,
        "isPartial" : false,
        "indexVersion" : 2,
        "direction" : "forward",
        "indexBounds" : {
          "personAge" : [
            "(25.0, 40.0)"
          ]
        }
      }
    },
    "rejectedPlans" : [ ]
  },
  "executionStats" : {
    "executionSuccess" : true,
    "nReturned" : 654,
    "executionTimeMillis" : 1,
```

The executionStats in MongoDB's explain output provides detailed information about how a query was executed, including performance metrics and statistics about the query execution.

Components of executionStats:

executionSuccess:

true: Indicates that the query executed successfully without errors.

nReturned:

654: The number of documents returned by the query.

executionTimeMillis:

1: The total time taken to execute the query, in milliseconds. This is the overall time MongoDB spent processing the query.

totalKeysExamined:

0: The number of index keys examined. Since this is a collection scan (COLLSCAN), no index keys were examined.

totalDocsExamined:

1000: The total number of documents examined during the query execution. MongoDB scanned 1000 documents to find those that matched the query criteria.

This section provides details on the query execution stages, specifically focusing on the stage chosen and its performance metrics.

stage:

COLLSCAN: Indicates that a collection scan was performed. This means MongoDB scanned the entire collection because no relevant index was used.

filter:

Contains the query conditions applied during the scan:

json

```
"$and": [
  { "personAge": { "$lt": 40 } },
  { "personAge": { "$gt": 25 } }
]
```

MongoDB applied these conditions to each document during the scan.

nReturned:

654: The number of documents that matched the filter criteria and were returned as results.

executionTimeMillisEstimate:

0: The estimated time for the execution stage. It may be 0 because it is an estimate or the time was too small to record.

works:

1002: The number of operations or "works" MongoDB performed to execute the stage. This includes both processing and moving through the collection.

advanced:

654: The number of documents processed by the stage and returned as results.

needTime:

347: The number of times MongoDB needed more time to continue processing the query. This indicates periods where the query execution was delayed.

needYield:

0: The number of times MongoDB needed to yield control to other operations. In this case, no yielding was needed.

saveState:

1: The number of times MongoDB saved the state of the execution. This is used for managing query execution.

restoreState:

1: The number of times MongoDB restored the state of the execution.

isEOF:

1: Indicates that the end of the query results was reached.

direction:

forward: The direction of the scan through the collection. In this case, it was a forward scan.

docsExamined:

1000: The total number of documents examined in the collection scan.

Command Prompt - mongo

```
> db.person.dropIndex("personAge_1")
{ "nIndexesWas" : 6, "ok" : 1 }
> db.person.getIndexes()
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "name" : -1
    },
    "name" : "name_-1"
  },
  {
    "v" : 2,
    "key" : {
      "index" : 1
    },
    "name" : "index_1",
    "unique" : true
  },
  {
    "v" : 2,
    "key" : {
      "name" : 1,
      "eyeColor" : 1
    },
    "name" : "name_1_eyeColor_1"
  },
  {
    "v" : 2,
    "key" : {
      "age" : 1
    },
    "name" : "age_1",
    "partialFilterExpression" : {
      "age" : {
        "$gt" : 25
      }
    }
  }
]
```

```
Command Prompt - mongo
> db.person.dropIndexes()
{
  "nIndexesWas" : 5,
  "msg" : "non-_id indexes dropped for collection",
  "ok" : 1
}
> db.person.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
>
```