Indexes and It's Types In MongoDB

AJIT YADAV (DBA)

Mongodb Index

In MongoDB, indexes improve the performance of search queries by allowing the database to quickly locate data without scanning every document in a collection.

Indexes in MongoDB are special data structures that store a small portion of the collection's data set in an easy-to-traverse form. They significantly im
of query execution by allowing MongoDB to limit the number of documents it must scan to return query results. Without indexes, MongoDB must scan
document in a collection, leading to slower query performance.

Creating an Index

To create an index in MongoDB, you can use the createIndex() method. The syntax for creating a single field index is as follows:

```
db.COLLECTION_NAME.createIndex({KEY: 1})
```

Types of Indexes in MongoDB:

Default _id index
Every collection automatically has an index on the _id field.

Single Field Index
Indexes a single field.

```
db.collection.createIndex({ fieldName: 1 }) // 1 for ascending, -1 for descending
```

Compound Index
Index on multiple fields.

```
db.collection.createIndex({ field1: 1, field2: -1 })
```

Multikey Index
Automatically created when indexing array fields.

Text Index
For full-text search in string content.

```
db.collection.createIndex({ description: "text" })
```

Hashed Index
Useful for sharding.

```
db.collection.createIndex({ userId: "hashed" })
```

Wildcard Index
Indexes all fields dynamically.

```
db.collection.createIndex({ "$**": 1 })
```

Unique Indexes
Prevents duplicate values.

```
db.users.createIndex({ email: 1 }, { unique: true })
```

Partial Indexes
Indexes only documents matching a filter.

```
db.orders.createIndex(
  { status: 1 },
  { partialFilterExpression: { status: { $eq: "active" } } }
)
```

View Existing Indexes
```
db.collection.getIndexes()
```

Drop an Index
```
db.collection.dropIndex({ fieldName: 1 })
```

Sparse Indexes
Indexes only documents with the indexed field.

```
db.users.createIndex({ phoneNumber: 1 }, { sparse: true })
```

TTL Indexes (Time-To-Live)
Automatically deletes documents after a certain time.

```
db.sessions.createIndex({ createdAt: 1 }, { expireAfterSeconds: 3600 })
```

Performance Tool

explain()
Analyzes how a query uses indexes.
```
db.collection.find({ name: "John" }).explain("executionStats")
```

db.collection.stats()
Shows storage stats, including index size.

How to manage index

List Indexes: db.collection.getIndexes()

Drop All Indexes: db.collection.dropIndexes()

Rebuild Indexes (if corruption suspected): db.collection.reIndex()

Best Practices while creating index

Only index fields used in queries, filters, or sorting.

Avoid too many indexes — each one slows writes.

Use compound indexes carefully — order matters.

Consider covered queries (where the index contains all the needed fields).

Use hint() to force a specific index (for performance testing).