Index Management

Index Creation

- *> db.users.createIndex({"username" : 1})*
  - Here '1' indicates the sorting order. 1 indicates ascending -1 indicates descending.
- **Compound Index**
- > db.students.createIndex({student_id: 1, class_id: 1})
  - The first argument to the command is a list of key-value pairs, where each pair consists of a field name and sort order, and the optional second argument is a set of options to control the indexes

Listing Indexes of a table

- *> db.users.getIndexes()*
- For each index, it displays the version, indexed fields and their sort order, the index name, and a namespace made up of the index name and database name.

```
MongoDB Enterprise > db.movies.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "name" : 1
                },
                "name" : "name_1"
        }
]
MongoDB Enterprise >
```

Index Names

- MongoDB assigns a default name to an index if a name is not provided explicitly. The default name of an index consists of the field name and the sort order, separated by underscores.
- However, you can also create an index with a specific name. To do so, you can use the **name** attribute to provide a custom name to the index, as follows:

```
db.theaters.createIndex(
   {theaterId : -1},
   {name : "myTheaterIdIndex"}
);
```

Dropping Indexes

- It is important to note that MongoDB does not allow updating an existing index. Thus, to fix an incorrectly created index, we need to drop it and recreate it correctly.
- dropIndex
  - db.collection.dropIndex(indexNameOrSpecification)
  - db.movies.dropIndex(
    {“name_1”}
)

Dropping Multiple Indexes

- db.collection.dropIndexes()
- This command can be used to drop all the indexes on a collection except the default **_id** index. You can use the command to drop a single index by passing either the index name or the index specification document. You can also use the command to delete a group of indexes by passing an array of index names.
- db.theaters.dropIndexes()

Hiding Indexes

- MongoDB provides a way to hide indexes from the query planner.
- To hide an index, the **hideIndex()** command can be used on the collection, as follows:
  - db.collection.hideIndex(indexNameOrSpecification)
- The **unhideIndex()** function takes a single argument, which can either be the index name or an index specification document.

Type of Indexes

- Default Index
  - each document in a collection has a primary key (namely, the _id field) and is indexed by default. MongoDB uses this index to maintain the uniqueness of the _id field, and it is available on all the collections.
- **Single-Key Indexes**
  - An index created using a single field from a collection is called a single-key index.
- **Compound Indexes**
  - An index created using more than a field from a collection is called a compound Index

Type of Indexes

- **Multikey Indexes**
  - An index created on the fields of an array type is called a multikey index. When an array field is passed as an argument to the **createIndex** function, MongoDB creates an index entry for each element of the array.
  - db.collectionName.createIndex({ arrayFieldName: sortOrder })
- **Text Indexes**
  - An index defined on a string field or an array of string elements is called a text index. Text indexes are not sorted, meaning that they are faster than normal indexes. The syntax to create a text index is as follows:
  - db.collectionName.createIndex({ fieldName : "text"})

Type of Indexes

- **Multikey Indexes**
  - An index created on the fields of an array type is called a multikey index. When an array field is passed as an argument to the **createIndex** function, MongoDB creates an index entry for each element of the array.

  - db.collectionName.createIndex({ arrayFieldName: sortOrder })

- **Text Indexes**
  - An index defined on a string field or an array of string elements is called a text index. Text indexes are not sorted, meaning that they are faster than normal indexes. The syntax to create a text index is as follows:
  - db.collectionName.createIndex({ fieldName : "text"})

Type of Indexes

- **Indexes on Nested Documents**

  Using a dot (**.**) notation, you can create an index on any of the nested document fields, just like any other field in the collection, as in the following example:

```
db.theaters.createIndex(
   { "location.address.zipcode" : 1}
)
```

Properties of Indexes

- **Unique Index**
  - A unique index property restricts the duplication of the index key. This is useful if you want to maintain the uniqueness of a field in a collection.

```
db.collection.createIndex(
    { field: type},
    { unique: true }
)
```

The **{ unique: true }** option is used to create a unique index.

Properties of Indexes

```
db.movies.createIndex(
    {title: 1, type:1},
    {
        partialFilterExpression: {
            year : { $gt: 1950}
        }
    }
)
```

Providing hints

- MongoDB query planner picks an index for a query depending on its own internal logic. When there are multiple indexes available to perform a query execution, the query planner uses its default query optimization technique to select and use the most appropriate index. However, we can use a **hint()** function to specify which index should be used for the execution:

```
db.users.find().hint(
    { index }
)
```