**Indexes**

Indexes support the efficient execution of queries in MongoDB.

Without indexes, MongoDB must perform a collection scan, i.e. scan every document in a collection, to select those documents that match the query statement.

If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

The index stores the value of a specific field or set of fields, ordered by the value of the field.

MongoDB creates a [unique index](https://docs.mongodb.com/manual/core/index-unique/#index-type-unique) on the [\_id](https://docs.mongodb.com/manual/core/document/#document-id-field) field during the creation of a collection. The \_id index prevents clients from inserting two documents with the same value for the \_id field.

You cannot drop this index on the \_id field.

**Types of indexes**

**SingleFeildIndex**

**CompoundIndex**

**TextIndex**

**MultiKeyIndex**

**SparseIndex**

**SinlgeFeildIndex**

MongoDB allows to create an index on any of the fields

We need to specify the option for sort the field in ascending or descending order

1. Ascending order
2. Descending order

Ex1 : Db.player.createIndex({score:1}) // sorts the data in ascending order

**CompundIndex**

Single Index Strcuture holds multiple fields

Db.products.createIndex({“item”:1,”stock”:1}) // index on item and stock as well as stores the data in ascending order

**MultiKeyIndex**

Used to create Indexes on Array of subdocuments

MongoDB creates an index key for each element in the array.

Ex1 : Db.records.createIndex({location.state}) //location is an array subdocument ie subdocs

**TextIndexes**

MongoDB provides [text indexes](https://docs.mongodb.com/manual/core/index-text/#index-feature-text) to support text search queries on string content.

A [compound index](https://docs.mongodb.com/manual/core/index-compound/) can include text index keys

A collection can have most one text Index

Db.inventory.createIndex({description : “text”})

Db.inventory.createIndex({description : “text”, tilte :”text”})

**SparseIndex**

Sparse indexes only contain entries for documents that have the indexed field, even if the index field contains a null value.

The index skips over any document that is missing the indexed field.

The index is “sparse” because it does not include all documents of a collection.

A non-sparse indexes contain all documents in a collection, storing null values for those documents that do not contain the indexed field.

To create a sparse index

Db.collection.createIndex({feildName:1},{sparse:true})

Ex:

//Create a collection

db.myCollection.insert({"x":"3"})

db.myCollection.insert({"y":"2"})

db.myCollection.insert({"x":"1"})

//Create a SparseIndex

db.myCollection.createIndex({x: 1},{sparse:true})

//Query with out sparse

db.myCollection.find().sort({ x:-1 })

db.myCollection.find().sort({ x:1 })

//query to check with sparse . hint() specify field name of sparse index , search by using only sparse Index

db.myCollection.find().sort({ x:-1 }).hint({x:1})

**To Drop an Index**

db.Orders.dropIndex("Index Name ")