MongoDB
Indexes & Storage

Team Emertxe

Indexes

Indexes

An index is a data structure that improves the speed of data retrieval operations on a database table.

In MongoDB uses index to reduce the number of documents to be scanned in a given collection ,if no index is define in the MongoDB then it has to scan every document of a given collection.

Indexes

MongoDB defines indexes at the collection level and supports indexes on any field or sub-field of the documents in a MongoDB collection.

MongoDB can return sorted results by using the ordering in index.

MongoDB indexes use a B-tree data structure.

Default Index

MongoDB creates a unique index named _id acts as a primary key to access any document in a collection.

The _id index prevents the insertion of two documents with the same value for the _id field.

Creating Index

Syntax :

- db.collection_name.createIndex({ field : value })

Creating Index
Example

```
> db.Student.createIndex({ id : 1 })
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1
}
```

Index Types

- Single Field Index
- Compound Index
- MultiKey Index
- Geospatial Index
- Text Index
- Hash Index

Single Field Index

MongoDB provide ascending/descending indexes on a single field.

Field value 1 specifies that orders item in ascending order.

Field value -1 specifies that orders item in descending order.

Compound Index

Compound Index is a single index structure holds references to multiple fields within a collection documents .

- Example

```
> db.Student.createIndex( name : 1 , marks :-1} )
```

MultiKey Index

Multikey Index is used to index the content stored in arrays.

MongoDB creates separate index entry for every element of the array.

MultiKey Index
Example

```
> db.StudAddress.insert( { stud_id  : 01 , addr :  [ {city : "ranchi"}  ,
{city: "mumbai" } , { city : "delhi" } ] });
WriteResult({ "nInserted" : 1 })
> db.StudAddress.createIndex( { "addr.city" : 1 } );
{
      "createdCollectionAutomatically" : false,
      "numIndexesBefore" : 1,
      "numIndexesAfter" : 2,
      "ok" : 1
}
```

The above example create the multikey index for {“addr:city” : 1}

Exercise

- Write a MongoDB query to create an ascending index on Employee collection on employee id field.
- Write a MongoDB query to create compound index on Employee collection on (id and name) field.

Geospatial Index

Geospatial Index is used to support the queries required for the geospatial coordinate data.

There are two indexes :

- 2d Indexes
- 2dsphere

2d indexes that uses planar geometry when returning results

2dsphere indexes that use spherical geometry to return results.

Text Index

Text Index is used to search for string content in a collection.

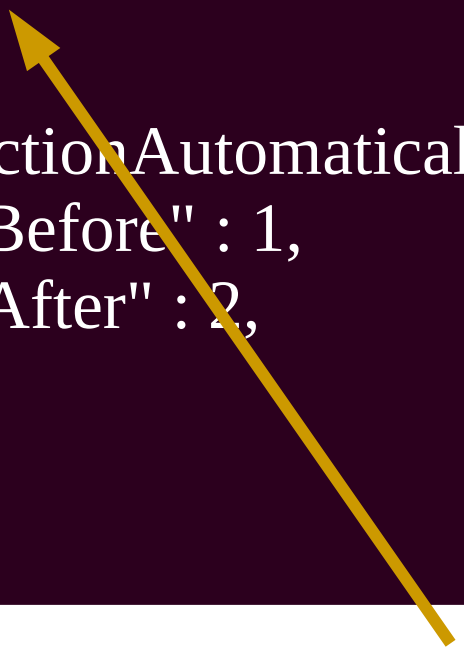
- Example

```
>db.Student.createIndex( { course : “text” })
```

Specify the text Index name

```
> db.person.createIndex( { name  : "text" , city : "text" } , { name : "id1" } )

{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1
}
```



Text Index Name

Wildcard Text Indexes

We can use the Wildcard specifier with text index .

With Wildcard text index ,MongoDB indexes every field that contains string data for each document in the collection.

Example

```
db.collection.createIndex( {“$**” , “text” } )
```

Text Index

- Drop a Text Index

  >db.collection.dropIndex(“text index name “)

- To list all Indexes

  > db.collection.getIndexes()

Exercise

- Write a MongoDB query to create a text index on employee collection.
- Write a MongoDB query to delete text index.

Hash Index

Hash Index maintain entries with hashes of the values of the indexed field.

Hashed indexes support sharding using shard keys .

Example :

```
> db.collection.createIndex( {_id : “hashed” })
```

Index Properties

Unique Indexes

Unique Indexes are used to remove duplicate values for the indexed field.

Partial Indexes

Partial index will index only the document in a collection that meet a specified filter expression.

Partial Indexes are preferred over sparse index.

Index Properties

Sparse Indexes

The Sparse Indexes will contain entries for documents that have the indexed field.

The index skips the document that do not have indexed field.

Index Properties

TTL (Time -to-Live) Indexes

TTL indexes are used to remove documents from a collection after a certain amount of time.

Data Expiration is useful for machine generated event data , logs and session information.

Example :

```
> db.eventlog.createIndex ( { “lastmodifieddate”  : 1 } , { expireAfterSeconds : 3600 } )
```

Index Intersection

In MongoDB we can use the intersection of indexes to fulfill queries.

Multiple/nested index intersection can be used to resolve a query.

In compound query conditions , one index can fulfill a part of query condition and another index can fulfill another part of query condition.

Index Intersection
Example

```
> db.Student.createIndex( {name : 1 , marks : 1} );
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 5,
        "numIndexesAfter" : 6,
        "ok" : 1
}
> db.Student.find( { name : "Mac"   , marks : { $gt : 15 } } )
{ "_id" : ObjectId("58c6344a97a24d8a5d676497"), "name" :
"Mac",
"id" : 2, "marks" : 68, "status" : "A" }
```

Storage

Storage Engine

The Storage Engine is the primary component of MongoDB database responsible for managing how data is stored in memory and on disk.

MongoDB support multiple Storage Engine as different engines for specific workloads.

Storage Engine

WiredTiger is the default storage engine introduced in MongoDB 3.2.

WiredTiger provide document level , concurrency model , checkpoint and compression and Encryption at Rest.

Journal

The Journal is a log that helps a database to recover in the event of hard shutdown.

Journaling writes data first to the journal and then to the core data files.

MongoDB enables Journaling by default for 64 bit .

Journal files are allocated and exists as files in the data directory.

GridFS

GridFS is a storage system that is used to handle large files that exceed the BSON-document size limit of 16 MB.

GridFS divides the file into parts or chunks (Each chunk as a separate document)

GridFS divides a file into chunks of 255kB with the exception of the last chunk.

The last chunk is as large as necessary.

GridFS

GridFS uses two collections to store files.

- Chunks : stores the binary chunks.
- Files : stores the file's metadata.

By default GridFS uses two collection with a bucket named fs :

- fs.files
- fs.chunks

References

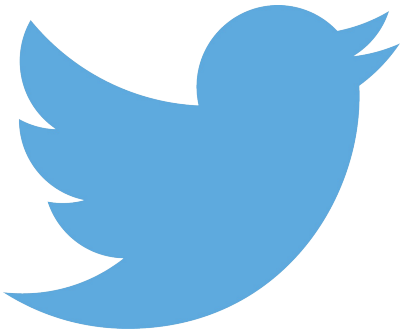
- https://www.wikimedia.org/
- https://docs.mongodb.com/manual/

Stay connected

About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,
No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046
T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe



https://twitter.com/EmertxeTweet

https://www.slideshare.net/EmertxeSlides

EMERTXE

Thank You