CSCI235 Database Systems

MongoDB Indexing

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Indexing

Outline

Overview of indexing

Indexes significantly reduce amount of time needed to access the documents

Without indexes all the documents in a collection must be accessed

Single-key index is the most appropriate for `{"key":"value"}` query conditions

For query conditions over multiple keys, e.g. `{$and:[{"key1":"value1"},{"key2":"value2"}]}` compound index is the best option

If we have a compound index on (key1, key2) then the second index on key1 is not really needed, however it may still speed up an access a bit

If we have a compound index on (key1, key2) then the second index on key2 speeds up access a lot

An order of keys in a compound index is very important

Indexing

Outline

Single key indexes

An index on `"_id"` is an automatically created single-key index

Equality search over the values of `"_id"` is the fasted possible search

The following command creates a single key unique index on a key `code`

createIndex()

```
db.department.createIndex( {"code": 1}, {"unique":true} )
```

The index is unique because it enforces uniqueness of the values associated with key `code`, i.e. each document in a collection has a different value associated with a key `code`

An attempt to insert two documents with the same value of key `code` fails enforcing a key constraint

Unique index should be create before inserting any data

A unique index cannot be created on a collection where duplicate keys exist

Single key indexes

The following command creates a single key nonunique index on a key `budget`

createIndex()
```
db.department.createIndex( {"budget": 1}, {"unique": false} )
```

Indexing

Outline

Compound key index

The following commands create the single key nonunique indexes on the keys `budget` and `total_staff_number`

createIndex()
```
db.department.createIndex( {"budget":1}, {"unique":false} )
```
createindex()
```
db.department.createIndex( {"total_staff_number":1}, {"unique":false} )
```

A query like

createIndex()
```
db.department.find({"budget":2000, :"total_staff_number":5})
```

uses only one of the indexes created above

A query optimizer picks the more efficient index (with higher selectivity)

To use both indexes we can traverse each index separately and calculate intersection of disk locations found

Compound key index

The following commands create a compound key nonunique index on the keys `budget` and `total_staff_number`

createIndex()
```
db.department.createIndex( {"budget":1, "total_staff_number":1}, {"unique":false} )
```

A compound index is a single index where each entry is composed of more than one key

A compound index is used by a query

find()
```
db.department.find({"budget":2000, :"total_staff_number":5})
```

Indexing

Outline

Sparse index

MongoDB indexes are dense by default

In a dense index for every document there is an index key even the document lacks a key

Then there exists a null entry in an index and it is possible to use an index for a query like

```
find()
db.department.find( {"budget":null} )
```

Dense index is inconvenient when:

- unique index on a field that doesn't appear in every document in a collection is needed
- a large number of documents in a collection have no indexed key

In a sparse index, only documents that have a value for the indexed key are indexed

```
createIndex()
db.department.createIndex( {"total_staff_number":1},
                                                  {"unique":false, "sparse":true} )
```

Indexing

Outline

Multikey index

In multikey index multiple entries in the index reference the same document

Multikey index is useful for indexing fields whose values are arrays

createIndex()

```
db.department.createIndex( {"course.code":1} )
```

Each value in this `courses.code` array will appear in the index

A query on any array values can use the index to locate the document

Indexing

Outline

Hashed index

In hashed index the keys become the arguments of a hash function and a results of of hash function determine location of a document in a hash bucket

The hashed values will determine the ordering of the documents

```
                                                   Indexing
db.department.createIndex( {"name":"hashed"} )
```

Hashed indexes have the following restructions:

- equality queries can be processed with an index
- range queries cannot use hashed index
- multikey hashed indexes are not allowed
- floating-point values are cast to an integer before being hashed; 1.4 and 1.5 will have the same value in hashed index

Hashed indexes are used for sharding

Indexing

Outline

Geospacial index

Geospacial index allows to find the documents that are close to a given location, based on latitude and longitude values stored in each document

Geospacial index can be used to efficiently calculate geographic distances, including the curvature of the earth

MongoDB supports different kinds of indexes, however, only the first two types of indexes listed below can be combined to a compound index

- 1: Ascending B*-tree index
- -1: Descending B*-tree index
- "hashed": Hashtable index; very fast for lookup by exact value, especially in very large collections. But it is not usable for inexact queries (`"$gt"`, `"$regex"` or similar)
- "text": Text index designed for searching for words in strings with natural language
- "2d": Geospatial index on a flat plane
- "2dsphere": Geospatial index on a sphere

Indexing

Outline

Index administration

Listing the indexes

find()

```
db.department.getIndexes()
```

Results

```
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "test.department"
        }
]
```

Index administration

Creating an index

createIndex()

```
db.department.createIndex( {"name":"hashed"} )
```

Results

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

Index administration

Listing the indexes

getindexes()

```
db.department.getIndexes()
```

Results

```
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "test.department"
        },
        {
                "v" : 2,
                "key" : {
                        "name" : "hashed"
                },
                "name" : "name_hashed",
                "ns" : "test.department"
        }
]
```

Index administration

Delete an index `name_hashed`

dropIndex()

```
db.department.dropIndex("name_hashed")
```

getIndexes()

```
db.department.getIndexes()
```

Results

```
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "test.department"
        }
]
```

Index administration

Creation of indexes before loading data allows indexes to be built incrementally as the data is inserted

Creation of an index on an already loaded collection may take a long time

It is possible to create an index in the background without closing a database system

```
createIndex()
db.department.createIndex( {"total_staff_number":1},{"background":true} )
```

It is possible to create an index in offline by taking a replica node offline, building an index, and taking node online allowing the node to catch up with master replica node

When ready we can promote a node to primary and take another secondary node offline, etc.

Index administration

It is possible to re-build indexes in order to defragment them after a lot of updates

reIndex()

```
db.department.reIndex()
```

Results

```
{
        "nIndexesWas" : 1,
        "nIndexes" : 1,
        "indexes" : [
                {
                        "v" : 2,
                        "key" : {
                                "_id" : 1
                        },
                        "name" : "_id_",
                        "ns" : "test.department"
                }
        ],
        "ok" : 1
}
```

References

Chodorow K. MongoDB The Definitive Guide, O'Reilly, 2013

Banker K., Bakkum P., Verch S., Garret D., Hawkins T., MongoDB in Action, 2nd ed., Manning Publishers, 2016

MongoDB Manual, Indexes `https://docs.mongodb.com/manual/indexes/`