mongoDB TUTORIAL

MongoDB createIndex

ADVERTISEMENTS

Google Play
게임즈로 즐기는 오딘

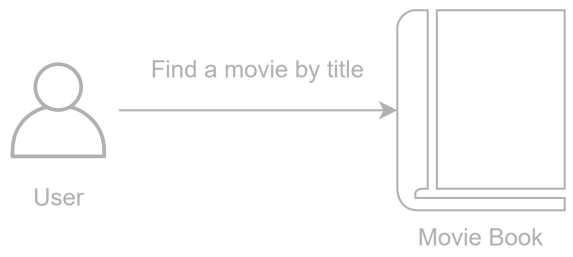
더욱 강해진 오딘을 Google Play 게임즈로
다양한 소환권과 함께 더 특별하게 즐기세요

Google Play Games

Summary: in this tutorial, you'll learn how to use MongoDB `createIndex()` method to create an index for a field in a collection to speed up queries.

A quick introduction to indexes

Suppose you have a book that contains a list of movies:



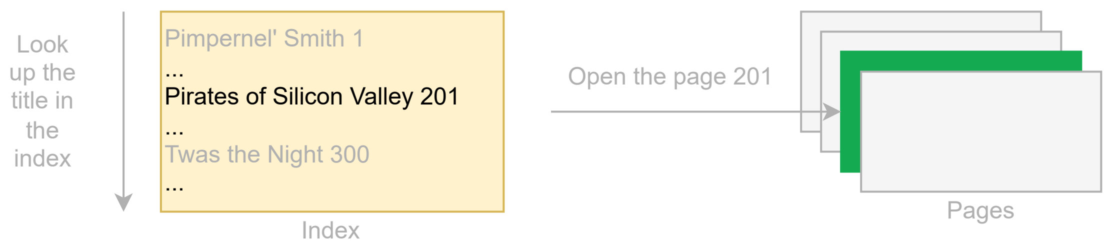
To find a movie with the title `Pirates of Silicon Valley`, you need to scan every page of the

ogle Play 게임즈로 즐기는 오딘
세서 오딘 플레이하고, 다양한 소환권과 오딘의 쿠폰 등 역대급 혜택을 놓치지
요 Google Play Games
자세히 알아보기

If the book has an index that maps titles with page numbers, you can look up the movie title in the index to find the page number:

```
Pimpernel' Smith 1
...
Pirates of Silicon Valley 201
...
Twas the Night 300
```

In this example, the movie with the title `Pirates of Silicon Valley` is located on page `201`. Therefore, you can open page 201 to get detailed information about the movie:



In this analogy, the index speeds up the search and makes it more efficient.

The MongoDB index works in a similar way. To speed up a query, you can create an index for a field of a collection.

However, when you insert, update, or delete the documents from the collection, MongoDB needs to update the index accordingly.

In other words, an index improves the speed of document retrieval at the cost of additional write and storage space to maintain the index data structure. Internally, MongoDB uses the B-tree structure to store the index.

Load sample data

We'll use the `movies` collection from the `mflix` sample database to demonstrate how the indexes work in MongoDB.

First, download the `movies.json` file and place it in a folder on your computer e.g., `c:\data\movies.json`

Second, import the `movies.json` file into the `mflix` database using the `mongoimport` tool:

```
mongoimport c:\data\movies.json -d mflix -c movies
```

List indexes of a collection

By default, all collections have an index on the `_id` field. To list the indexes of a collection, you use the `getIndexes()` method with the following syntax:

```
db.collection.getIndexes()
```

In this syntax, the `collection` is the name of the collection that you want to get the indexes. For example, the following shows the indexes of the `movies` collection in the `mflix` database:

```
db.sales.getIndexes()
```

Output:

```
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
```

The output shows the index name `'_id_'` and index key `_id`. The value 1 in the `key : { _id : 1 }` indicates the ascending order of the `_id` values in the index.

When an index contains one field, it's called a single field index. However, if an index holds references to multiple fields, it is called a compound index. This tutorial focuses on a single field index.

Explain a query plan

The following query finds the movie with the title `Pirates of Silicon Valley` :

```
db.movies.find({
    title: 'Pirates of Silicon Valley
})
```

To find the movie, MongoDB has to scan the `movies` collection to find the match.

Before executing a query, the MongoDB query optimizer comes up with one or more query execution plans and selects the most efficient one.

To get the information and execution statistics of query plans, you can use the `explain()` method:

```
db.collection.explain()
```

For example, the following returns the query plans and execution statistics for the query that finds the movies with the title `Pirates of Silicon Valley`:

```
db.movies.find({
   title: 'Pirates of Silicon Valley'
}).explain('executionStats')
```

The `explain()` method returns a lot of information. And you should pay attention to the following `winningPlan`:

```
...
    winningPlan: {
      stage: 'COLLSCAN',
      filter: { title: { '$eq': 'Pirates of Silicon Valley' } },
      direction: 'forward'
    },
...
```

The `winningPlan` returns the information on the plan that the query optimizer came up with. In this example, the query planner comes up with the `COLLSCAN` that stands for the collection scan.

Also, the `executionStats` shows that the result contains one document and the execution time is 9 milliseconds:

```
...
```

```
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 9,
    totalKeysExamined: 0,
    totalDocsExamined: 23539,
...
```

Create an index for a field in a collection

To create an index for the `title` field, you use the `createIndex()` method as follows:

```
db.movies.createIndex({title:1})
```

Output:

```
title_1
```

In this example, we pass a document to the `createIndex()` method. The `{ title: 1}` document contains the field and value pair where:

- The field is the index key (`year`).
- The value describes the type of index for the `year` field. The value `1` for descending index and `-1` for ascending index.

The `createIndex()` method returns the index name. In this example, it returns the `title_1` which is the concatenation of the field and value.

The following query shows the indexes of the `movies` collection:

```
db.movies.getIndexes()
```

Output:

```
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { title: 1 }, name: 'title_1' }
]
```

The output shows two indexes, one is the default index and another is the `year_1` index that we have created.

> By default, MongoDB names an index by concatenating the indexed keys and each key's direction in the index (i.e. 1 or -1) using underscores as a separator. For example, an index created on `{ title: 1 }` has the name `title_1`.

The following returns the query plans and execution statistics for the query that finds the movie with the title `Pirates of Silicon Valley`:

```
db.movies.find({
   title: 'Pirates of Silicon Valley'
})
```

This time the query optimizer uses the index scan (`IXSCAN`) instead of the collection scan (`COLLSCAN`):

```
...
winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { title: 1 },
        indexName: 'title_1',
        isMultiKey: false,
        multiKeyPaths: { title: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: {
          title: [
            '["Pirates of Silicon Valley", "Pirates of Silicon Valley"]'
          ]
        }
      }
    },
...
```

Also, the execution time (`executionTimeMillis`) was down to almost zero from `2` milliseconds:

```
...
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 0,
    totalKeysExamined: 1,
    totalDocsExamined: 1,
...
```

Summary

- An index improves the speed of document retrieval at the cost of additional write and storage space to maintain its data structure.
- Use the `createIndex()` method to create an index for a field in a collection.
- Use the `getIndexes()` method to list the indexes of a collection.
- Use the `explain()` method to get the information and execution statistics of query plans.

Was this tutorial helpful ?

ADVERTISEMENTS

ADVERTISEMENTS

ADVERTISEMENTS

ADVERTISEMENTS

ADVERTISEMENTS

ABOUT MONGODBTUTORIAL.COM

This MongoDB Tutorial helps you master MongoDB quickly.

Copyright © 2022 by mongodbtutorial.org Website. All Right Reserved.

This mongodbtutorial.org website has no relationship with MongoDB, Inc Also, MongoDB, Inc does not sponsor contents on this website.