

MongoDB教育訓練-06

What are Indexes for

- Speed up queries and updates
- Avoid disk I/O
- Reduce overall computation

How Do Indexes Work

- Data in an index is ordered.
 - Ordered data can be searched efficiently.
- Values in a MongoDB index point to document **identity**.
 - If a document moves on disk, the indexes don't need to be changed.

mongoDB 使用 B-tree索引

MongoDB Index misconceptions

- MongoDB is so fast it doesn't need indexes.
- Every field is automatically indexed.
- NoSQL uses hashes, not indexes

查詢至少要用到索引，不要full table scan(所以要至少建立可用索引)

When do you use an index

- A good index should support every query or update.
 - Scanning records is very inefficient, even if it is not all of them.
- Who determines the best index
 - The Developer, writing the query!
 - NOT a DBA after the fact

Types of Indexes

- Single-field indexes
- Compound indexes
- Multikey indexes
- Geospatial indexes
- Text indexes
- Hashed indexes
- Wildcard indexes

檢查執行計畫快取 -> (無) 有可能優化的索引找出來，同時跑看誰的效率快，加入快取

更新快取時機:

- 當前索引變慢(系統判斷)
- 當新的索引被創建
- server 重啟(memory 清除)

How MongoDB uses and chooses Indexes

- Checks in the PlanCache to see if an optimal index has been chosen before
- If not:
 - Picks all candidate indexes
 - Runs query using them to score which is most efficient
 - Adds its choice of best index to the PlanCache
- Plan cache entries are evicted when:
 - Using that index becomes less efficient
 - A new relevant index is added
 - The server is restarted

- 需要指定欄位
- 欄位可以是任何類型

Single Field Indexes

- Optimize finding values for a given field
- Specified as field name and direction
 - The direction is irrelevant for a single field index
- The field itself can be any data type.
- Indexing an Object type does not index all the values separately
 - It indexes the object essentially as a comparable blob.
 - You can use it for range searches against Objects or exact matches
- Indexing an Array is covered later.

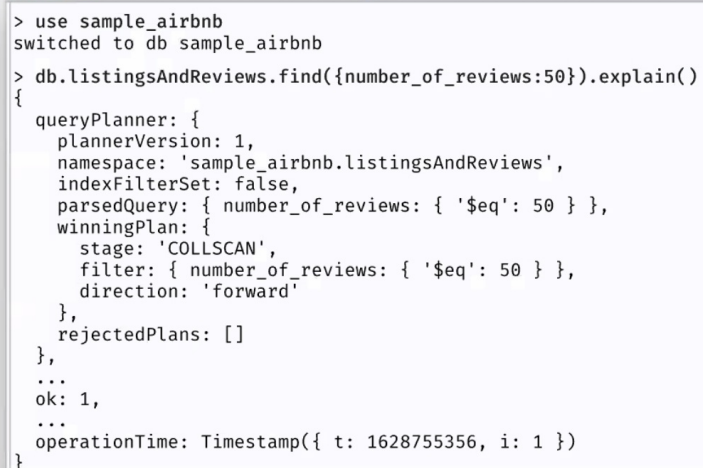
Index Demonstration

explain() on cursor shows the query mechanics.

COLLSCAN here - short for Collection Scan

Every document in the collection looked at.

Very Inefficient



```
> use sample_airbnb
switched to db sample_airbnb

> db.listingsAndReviews.find({number_of_reviews:50}).explain()
{
  queryPlanner: {
    plannerVersion: 1,
    namespace: 'sample_airbnb.listingsAndReviews',
    indexFilterSet: false,
    parsedQuery: { number_of_reviews: { '$eq': 50 } },
    winningPlan: {
      stage: 'COLLSCAN',
      filter: { number_of_reviews: { '$eq': 50 } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  ...,
  ok: 1,
  ...,
  operationTime: Timestamp({ t: 1628755356, i: 1 })
}
```

Explain 執行計畫

- queryPlanner : 只顯示執行計畫
- executionStats(最常用): 會把最優執行計畫執行一遍，並收集相關資訊
- allPlansExecution: 把所有執行計畫執行一遍，並收集相關資訊

Explain verbosity

"queryPlanner" : Shows the winning query plan but does not execute query

"executionStats" : Executes query and gathers statistics

"allPlansExecution": Runs all candidate plans and gathers statistics.

If you do not specify a verbosity - the default is "queryPlanner"

- nReturn: 返回數量
- totalKeyExamined : 找到索引
- totalDocExamined : 查找document數量
- stage:COLLSAN => 無index
- totalKeyExamined / nReturn 最好小於1000

Index Demonstration

We can see this looked at all 5,555 documents, returning 11 in 8 milliseconds

We can create an index to improve this.

Explain plans are complicated with nested stages of processing.

Key metrics are in bold here.

```
> use sample_airbnb
sample_airbnb
> db.listingsAndReviews.find({number_of_reviews:50}).explain(
"executionStats")

...
  executionStats: {
    executionSuccess: true,
    nReturned: 11,
    executionTimeMillis: 8,
    totalKeysExamined: 0,
    totalDocsExamined: 5555,
    executionStages: {
      stage: "COLLSAN",
      filter: {
        number_of_reviews: {
          '$eq' : 50
        }
      }
    }
  }
...

```

number_of_reviews: 1 (1: 升幂,-1:降幂)

Creating a simple index

In development, we can instantly create an index using createIndex()

In production, we need to look at the impact this will have.

There are better ways to do it in production.

```
> db.listingsAndReviews.createIndex({number_of_reviews:1})
number_of_reviews_1

```

通過index 取得資訊：FETCH

Index Demonstration

Here we see two stages:

- **IXSCAN** (Look through the index) returning a list of 11 documents identities
- **FETCH** (Read known document) getting the document itself

The total time was one millisecond

```
> db.listingsAndReviews.find({number_of_reviews:50}).explain(
  "executionStats")
...
  executionStats: {
    nReturned: 11,
    executionTimeMillis: 1,
    totalKeysExamined: 11,
    totalDocsExamined: 11,
    executionStages: {
      stage: 'FETCH',
      nReturned: 11,
      works: 12,
      docsExamined: 11,
      ...
      inputStage: {
        stage: 'IXSCAN',
        nReturned: 11,
        works: 12,
        ...
      }
    }
  }
  ...
```

- nReturn:totalKeyExamined:totalDocExamined 1:1:1 最佳情况

```
executionStats: {
  executionSuccess: true,
  nReturned: 11,
  executionTimeMillis: 1,
  totalKeysExamined: 11,
  totalDocsExamined: 11,
  executionStages: {
    stage: 'FETCH',
    nReturned: 11,
    executionTimeMillisEstimate: 0,
    works: 12,
    advanced: 11,
    needTime: 0,
    needYield: 0,
    saveState: 0,
    restoreState: 0,
    isEOF: 1,
    docsExamined: 11,
    alreadyHasObj: 0,
    inputStage: {
      stage: 'IXSCAN',
      nReturned: 11,
      executionTimeMillisEstimate: 0,
      works: 12,
      advanced: 11,
```

Explainable Operations

- find()
- aggregate()
- count()
- update()
- remove()
- findAndModify()

The newer style API for example updateOne() or updateMany() does not allow explain so you need to use update() - same functionality

Applying explain()

- A flag is sent to the server with the operation to say it's an explain command.
- We can set this on the cursor as we do for sort() or limit()
- What if the function we are calling does not return a cursor?
 - For example count() or update()
 - We need to set the flag on the collection object we call with.

```
peoplecoll = db.people
explainpeoplecoll = peoplecoll.explain()
explainpeoplecoll.count()
```

取得所有索引

```
Atlas atlas-vnfeuv-shard-0 [primary] sample_airbnb> db.listingsAndReviews.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_',
    {
      v: 2,
      key: { property_type: 1, room_type: 1, beds: 1 },
      name: 'property_type_1_room_type_1_beds_1',
      background: true
    }
  },
  { v: 2, key: { name: 1 }, name: 'name_1', background: true },
  {
    v: 2,
    key: { 'address.location': '2dsphere' },
    name: 'address.location_2dsphere',
    background: true,
    '2dsphereIndexVersion': 3
  },
  { v: 2, key: { number_of_reviews: 1 }, name: 'number_of_reviews_1' }
]
Atlas atlas-vnfeuv-shard-0 [primary] sample_airbnb>
```



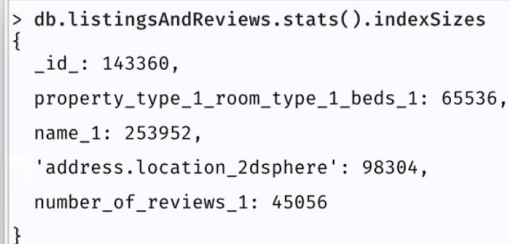
```
Atlas atlas-vnfeuv-shard-0 [primary] sample_airbnb> db.listingsAndReviews.dropIndex('number_of_reviews_1')
```

```
Atlas atlas-vnfeuv-shard-0 [primary] sample_airbnb> db.listingsAndReviews.dropIndex( { number_of_reviews: 1 } )
```

需要放得下index的記憶體（最低要求）

Index Sizes

We can call `stats()` method and look at the **indexSizes** key to see how large each index is in **bytes**.



```
> db.listingsAndReviews.stats().indexSizes
{
  _id_: 143360,
  property_type_1_room_type_1_beds_1: 65536,
  name_1: 253952,
  'address.location_2dsphere': 98304,
  number_of_reviews_1: 45056
}
```

`db.tablename.stats(檔案除以 ? (1024 => KB, 1024*1024 => MB)).indexSizes`

```
Atlas atlas-vnfeuv-shard-0 [primary] sample_airbnb> db.listingsAndReviews.stats().indexSizes
{
  _id_: 151552,
  property_type_1_room_type_1_beds_1: 65536,
  name_1: 253952,
  'address.location_2dsphere': 98304,
  number_of_reviews_1: 45056
}
Atlas atlas-vnfeuv-shard-0 [primary] sample_airbnb> db.listingsAndReviews.stats(1024*1024).indexSizes
{
  _id_: 0,
  property_type_1_room_type_1_beds_1: 0,
  name_1: 0,
  'address.location_2dsphere': 0,
  number_of_reviews_1: 0
}
Atlas atlas-vnfeuv-shard-0 [primary] sample_airbnb> db.listingsAndReviews.stats(1024).indexSizes
{
  _id_: 148,
  property_type_1_room_type_1_beds_1: 64,
  name_1: 248,
  'address.location_2dsphere': 96,
  number_of_reviews_1: 44
}
```

- `storageSize` : 物理檔案大小
- 刪除資料不會減少資料庫大小，只是把硬碟位置空出來，重複使用
- 釋放空間：compact <https://docs.mongodb.com/manual/reference/command/compact/>

```

Atlas atlas-vnfeuv-shard-0 [primary] sample_airbnb> db.listingsAndReviews.stats(1024)
{
  ns: 'sample_airbnb.listingsAndReviews',
  size: 92150,
  count: 5555,
  avgObjSize: 16986,
  storageSize: 52792,
  freeStorageSize: 40,
  capped: false,
  nindexes: 5,
  indexBuilds: [],
  totalIndexSize: 600,
  totalSize: 53392,
  indexSizes: {
    _id_: 148,
    property_type_1_room_type_1_beds_1: 64,
    name_1: 248,
    'address.location_2dsphere': 96,
    number_of_reviews_1: 44
  },
  scaleFactor: 1024,
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp(1, 1639707949),
    signature: {
      hash: Binary(Buffer.from("6408e9a5c0dd720fe20ac9904e89f1c4a3ecb58f", "hex"), 0),
      keyId: Long("6978151756584714241")
    }
  },
  operationTime: Timestamp(1, 1639707949)
}

```

Exercise

Use the **sample_airbnb** database and the **listingsAndReviews** collection.

1. Find the name of the host with the most total listings (this is an existing field)
2. Create an index to support the query.
3. Calculate how much more efficient it is now with this index.

Note - this is a tricky question for a number of reasons!