LABORATORIUM NIERELACYJNE BAZY DANYCH

Data wykonania ćwiczenia:	08.04.2023
Rok studiów:	3
Semestr:	6
Grupa studencka:	2
Grupa laboratoryjna:	2B

Ćwiczenie nr.	5

Temat: Tworzenie indeksów i wykorzystywanie ich w zapytaniach

Osoby wykonujące ćwiczenia:

1. Igor Gawłowicz

Katedra Informatyki i Automatyki

Tworzenie indeksów i wykorzystywanie ich w zapytaniach

Na potrzeby przebadania efektywności indeksów w bazie danych stworzymy sobię nową kolekcję z wygenerowanymi 200 tysięcy prostych rekordów z kolejnymi liczbami.

```
from pymongo import MongoClient


client = MongoClient('mongodb://localhost:27017')

db = client['my_app']

collection = db['numbers']

query = {"student_surname": "Smith"}

for i in range(200000):
    collection.insert_one({"num": i})
```

Po kilku minutach nasze 200000 rekordów jest już w bazie

```
my_app> db.numbers.count()
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or
estimatedDocumentCount.
200000
```

Możemy także spradzić dowolny konkretny numer

```
my_app> db.numbers.find({num: 500})
[ { _id: ObjectId('6613d5d8c8e3c4af97dd5ddc'), num: 500 } ]
```

```
my_app> db.numbers.find({num: {"$gt": 199995}})
[
  { _id: ObjectId('6613d6a6c8e3c4af97e06924'), num: 199996 },
  { _id: ObjectId('6613d6a6c8e3c4af97e06925'), num: 199997 },
  { _id: ObjectId('6613d6a6c8e3c4af97e06926'), num: 199998 },
  { _id: ObjectId('6613d6a6c8e3c4af97e06927'), num: 199999 }
]
```

Możemy także ustalić górny i dolny limit

```
my_app> db.numbers.find({num: {"$lt": 199995, "$gt":199990}})
[
```

```
  { _id: ObjectId('6613d6a6c8e3c4af97e0691f'), num: 199991 },
  { _id: ObjectId('6613d6a6c8e3c4af97e06920'), num: 199992 },
  { _id: ObjectId('6613d6a6c8e3c4af97e06921'), num: 199993 },
  { _id: ObjectId('6613d6a6c8e3c4af97e06922'), num: 199994 }
]
```

Możemy teraz stworzyć indeks na naszej kolekcji

```
my_app> db.numbers.ensureIndex({num: 1})
[ 'num_1' ]
my_app> db.numbers.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { num: 1 }, name: 'num_1' }
]
```

Wyniki wykonania zapytania przed dodaniem indeksowania

```
my_app> db.numbers.find({num: {"$gt": 180000}}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'my_app.numbers',
    indexFilterSet: false,
    parsedQuery: { num: { '$gt': 180000 } },
    queryHash: '01E69096',
    planCacheKey: '01E69096',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'COLLSCAN',
      filter: { num: { '$gt': 180000 } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 19999,
    executionTimeMillis: 228,
    totalKeysExamined: 0,
    totalDocsExamined: 200000,
    executionStages: {
      stage: 'COLLSCAN',
      filter: { num: { '$gt': 180000 } },
      nReturned: 19999,
      executionTimeMillisEstimate: 19,
      works: 200001,
```

```
      advanced: 19999,
      needTime: 180001,
      needYield: 0,
      saveState: 200,
      restoreState: 200,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 200000
    }
  },
  command: {
    find: 'numbers',
    filter: { num: { '$gt': 180000 } },
    '$db': 'my_app'
  },
  serverInfo: {
    host: 'DESKTOP-6GVNM2J',
    port: 27017,
    version: '7.0.5',
    gitVersion: '7809d71e84e314b497f282ea8aa06d7ded3eb205'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted'
  },
  ok: 1
}
```

Wyniki wykonania zapytania po dodaniu indeksowania

```
my_app> db.numbers.find({num: {"$gt": 180000}}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'my_app.numbers',
    indexFilterSet: false,
    parsedQuery: { num: { '$gt': 180000 } },
    queryHash: '01E69096',
    planCacheKey: '5AEA6406',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
```

```
        stage: 'IXSCAN',
        keyPattern: { num: 1 },
        indexName: 'num_1',
        isMultiKey: false,
        multiKeyPaths: { num: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { num: [ '(180000, inf.0]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 19999,
    executionTimeMillis: 74,
    totalKeysExamined: 19999,
    totalDocsExamined: 19999,
    executionStages: {
      stage: 'FETCH',
      nReturned: 19999,
      executionTimeMillisEstimate: 11,
      works: 20000,
      advanced: 19999,
      needTime: 0,
      needYield: 0,
      saveState: 20,
      restoreState: 20,
      isEOF: 1,
      docsExamined: 19999,
      alreadyHasObj: 0,
      inputStage: {
        stage: 'IXSCAN',
        nReturned: 19999,
        executionTimeMillisEstimate: 8,
        works: 20000,
        advanced: 19999,
        needTime: 0,
        needYield: 0,
        saveState: 20,
        restoreState: 20,
        isEOF: 1,
        keyPattern: { num: 1 },
        indexName: 'num_1',
        isMultiKey: false,
        multiKeyPaths: { num: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
```

```
        indexBounds: { num: [ '(180000, inf.0]' ] },
        keysExamined: 19999,
        seeks: 1,
        dupsTested: 0,
        dupsDropped: 0
      }
    }
  },
  command: {
    find: 'numbers',
    filter: { num: { '$gt': 180000 } },
    '$db': 'my_app'
  },
  serverInfo: {
    host: 'DESKTOP-6GVNM2J',
    port: 27017,
    version: '7.0.5',
    gitVersion: '7809d71e84e314b497f282ea8aa06d7ded3eb205'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted'
  },
  ok: 1
}
```

Możemy teraz porównać czas wykonania zapytania

Przed: `executionTimeMillis: 228,`

Po: `executionTimeMillis: 74,`

Widzimy że czas wykonania jest 3-krotnie krótszy