LABORATORIUM NIERELACYJNE BAZY DANYCH

Data wykonania ćwiczenia:	26.02.2023
Rok studiów:	3
Semestr:	6
Grupa studencka:	2
Grupa laboratoryjna:	2B

Ćwiczenie nr.	3

Temat: MongoDB. Filtrowanie i Sortowanie Danych. Indeksowanie

Osoby wykonujące ćwiczenia:

1. Igor Gawłowicz

Katedra Informatyki i Automatyki

Przygotowanie Środowiska.

Na ostatnich labach przygotowaliśmy środowisko a także stworzyliśmy testową kolekcje `courses`, gdzie wprowadziliśmy przykładowe dane dla 100 testowych studentów.

Filtrowanie Danych.

Przeprowadź 5 zapytań, aby wyświetlić tylko zażądane informacje. Wykorzystaj różne operatory porównania (np. $eq, $ne, $gt, $lt) do filtrowania danych

1. Znajdź wszystkich studentów, którzy otrzymali ocenę wyższą niż 4 z przedmiotu "Mathematics".

```
db.courses.find({"subjects.subject": "Mathematics", "subjects.grade": {$gt: 4}})
[
  {
    _id: ObjectId('65f83c77f23ac558846c9de5'),
    student_id: 0,
    student_name: 'Evelyn',
    student_surname: 'Roberts',
    subjects: [
      { subject: 'Mathematics', grade: 3, final: false },
      { subject: 'English Language', grade: 4, final: true },
      { subject: 'History', grade: 3, final: true },
      { subject: 'Science', grade: 2, final: true },
      { subject: 'Computer Science', grade: 3, final: true },
      { subject: 'Physical Education', grade: 5, final: true }
    ]
  },
  ...
```

2. Znajdź wszystkich studentów, którzy nie otrzymali oceny 5 z przedmiotu "English Language".

```
db.courses.find({"subjects.subject": "English Language", "subjects.grade": {$ne:
5}})
[
  {
    _id: ObjectId('65f83c77f23ac558846c9de7'),
    student_id: 2,
    student_name: 'Kristina',
    student_surname: 'Young',
    subjects: [
      { subject: 'Mathematics', grade: 4, final: false },
      { subject: 'English Language', grade: 3, final: true },
      { subject: 'History', grade: 2, final: true },
      { subject: 'Science', grade: 2, final: true },
      { subject: 'Computer Science', grade: 3, final: true },
      { subject: 'Physical Education', grade: 2, final: true }
    ]
```

```
  },
  ...
```

3. Znajdź wszystkich studentów, którzy otrzymali co najmniej jedną ocenę niższą niż 3.

```
db.courses.find({"subjects.grade": {$lt: 3}})
[
      {
    _id: ObjectId('65f83c77f23ac558846c9dfc'),
    student_id: 23,
    student_name: 'Jeffrey',
    student_surname: 'Glass',
    subjects: [
      { subject: 'Mathematics', grade: 3, final: false },
      { subject: 'English Language', grade: 3, final: true },
      { subject: 'History', grade: 2, final: false },
      { subject: 'Science', grade: 2, final: false },
      { subject: 'Computer Science', grade: 3, final: true },
      { subject: 'Physical Education', grade: 2, final: false }
    ]
  },
  ...
```

4. Znajdź wszystkich studentów, którzy otrzymali ocenę 5 z przedmiotu "Science".

```
db.courses.find({"subjects.subject": "Science", "subjects.grade": {$eq: 5}})
[
      {
    _id: ObjectId('65f83c77f23ac558846c9dfd'),
    student_id: 24,
    student_name: 'Amy',
    student_surname: 'Sexton',
    subjects: [
      { subject: 'Mathematics', grade: 4, final: true },
      { subject: 'English Language', grade: 5, final: true },
      { subject: 'History', grade: 4, final: true },
      { subject: 'Science', grade: 5, final: true },
      { subject: 'Computer Science', grade: 5, final: false },
      { subject: 'Physical Education', grade: 5, final: true }
    ]
  }
  ...
```

5. Znajdź wszystkich studentów, którzy nie otrzymali oceny końcowej (final: false) z co najmniej jednego przedmiotu.

```

```

```
db.courses.find({"subjects.final": false})

[
{
    _id: ObjectId('65f83c77f23ac558846c9df8'),
    student_id: 19,
    student_name: 'John',
    student_surname: 'Rodgers',
    subjects: [
      { subject: 'Mathematics', grade: 3, final: false },
      { subject: 'English Language', grade: 2, final: false },
      { subject: 'History', grade: 5, final: true },
      { subject: 'Science', grade: 3, final: true },
      { subject: 'Computer Science', grade: 4, final: false },
      { subject: 'Physical Education', grade: 3, final: false }
    ]
  }
  ...
```

Sortowanie Danych/Złożone Zapytania:

Zastosuj sortowanie. Przetestuj różne kierunki sortowania (rosnąco, malejąco).

Znajdź wszystkich studentów, którzy otrzymali ocenę wyższą niż 4 z przedmiotu "Mathematics", posortowanych malejąco według oceny.

```
db.courses.find({"subjects.subject": "Mathematics", "subjects.grade": {$gt:
4}}).sort({"subjects.grade": -1})

[
  {
    _id: ObjectId('65fa0fa4a88d6ebfd7d14a0e'),
    student_id: 0,
    student_name: 'Evelyn',
    student_surname: 'Roberts',
    subjects: [
      { subject: 'Mathematics', grade: 3, final: false },
      { subject: 'English Language', grade: 4, final: true },
      { subject: 'History', grade: 3, final: true },
      { subject: 'Science', grade: 2, final: true },
      { subject: 'Computer Science', grade: 3, final: true },
      { subject: 'Physical Education', grade: 5, final: true }
    ]
  },
  {
    _id: ObjectId('65fa0fa4a88d6ebfd7d14a0f'),
    student_id: 1,
    student_name: 'Joanne',
    student_surname: 'Thomas',
    subjects: [
```

```
      { subject: 'Mathematics', grade: 2, final: true },
      { subject: 'English Language', grade: 5, final: true },
      { subject: 'History', grade: 5, final: true },
      { subject: 'Science', grade: 4, final: false },
      { subject: 'Computer Science', grade: 5, final: true },
      { subject: 'Physical Education', grade: 2, final: false }
    ]
  },
  ...
```

Znajdź wszystkich studentów, którzy otrzymali ocenę 5 z przedmiotu "Science", posortowanych malejąco według nazwiska.

```
db.courses.find({"subjects.subject": "Science", "subjects.grade": {$eq:
5}}).sort({"student_surname": -1})

[
  {
    _id: ObjectId('65fa0fa4a88d6ebfd7d14a4b'),
    student_id: 61,
    student_name: 'Brandon',
    student_surname: 'Winters',
    subjects: [
      { subject: 'Mathematics', grade: 2, final: false },
      { subject: 'English Language', grade: 2, final: true },
      { subject: 'History', grade: 5, final: false },
      { subject: 'Science', grade: 5, final: true },
      { subject: 'Computer Science', grade: 4, final: true },
      { subject: 'Physical Education', grade: 5, final: true }
    ]
  },
  {
    _id: ObjectId('65fa0fa4a88d6ebfd7d14a6f'),
    student_id: 97,
    student_name: 'Mary',
    student_surname: 'Wilson',
    subjects: [
      { subject: 'Mathematics', grade: 5, final: true },
      { subject: 'English Language', grade: 4, final: true },
      { subject: 'History', grade: 2, final: true },
      { subject: 'Science', grade: 5, final: true },
      { subject: 'Computer Science', grade: 3, final: true },
      { subject: 'Physical Education', grade: 5, final: true }
    ]
  },
  ...
```

Indeksowanie

Zidentyfikuj pola, które powinny być zindeksowane dla efektywnego filtrowania i sortowania.

Zaimplementuj odpowiednie indeksy i sprawdź, jak wpływają one na wydajność zapytań.

Możemy zaimplementować odpowiednie indeksy w MongoDB dla tych pól i sprawdzić, jak wpływają one na wydajność zapytań.

```
db.courses.createIndex({"subjects.subject": 1})
db.courses.createIndex({"subjects.grade": 1})
db.courses.createIndex({"subjects.final": 1})
db.courses.createIndex({"student_surname": 1})
```

Nasz zakres danych jest zbyt mały żeby znacząco zaobserwować efektywność tych danych, jednak moglibyśmy to zrobić poniższym poleceniem:

```
db.courses.find({"subjects.subject": "Mathematics", "subjects.grade": {$gt:
3}}).sort({"subjects.grade": -1}).explain("executionStats")
```

W ten sposób wykonamy zapytanie a także otrzymamy dane o wszystkich parametrach użytych przy jego wykonaniu w tym np czasu.

Operacje CRUD.

Zaimplementuj podstawowe operacje CRUD (Create, Read, Update, Delete) dla kolekcji. Skorzystaj z oficjalnej dokumentacji MongoDB dla szczegółów dotyczących operacji filtrowania, sortowania i indeksowania

W celu tego przygotowałem prosty skrypt w pythonie, który zawiera każdą z tych funkcjonalności, wraz z komentarzami wyjaśniającymi poszczególne fragmenty kodu.

```
import pymongo
from pymongo import MongoClient

# Połączenie z lokalnym klientem MongoDB
client = MongoClient('localhost', 27017)

# Wybór bazy danych
db = client['mydatabase']

# Wybór kolekcji
collection = db['courses']

# Operacja Create (dodawanie nowego dokumentu)
def create_document(data):
    collection.insert_one(data)
    print("Dokument został pomyślnie dodany.")

# Operacja Read (odczyt wszystkich dokumentów)
def read_all_documents():
    cursor = collection.find({})
    for document in cursor:
```

```
        print(document)

# Operacja Read (odczyt dokumentu o określonym identyfikatorze)
def read_document(student_id):
    document = collection.find_one({"student_id": student_id})
    if document:
        print(document)
    else:
        print("Nie znaleziono dokumentu o podanym identyfikatorze.")

# Operacja Update (aktualizacja istniejącego dokumentu)
def update_document(student_id, new_data):
    result = collection.update_one({"student_id": student_id}, {"$set": new_data})
    if result.modified_count > 0:
        print("Dokument został pomyślnie zaktualizowany.")
    else:
        print("Nie udało się znaleźć dokumentu o podanym identyfikatorze.")

# Operacja Delete (usunięcie dokumentu)
def delete_document(student_id):
    result = collection.delete_one({"student_id": student_id})
    if result.deleted_count > 0:
        print("Dokument został pomyślnie usunięty.")
    else:
        print("Nie udało się znaleźć dokumentu o podanym identyfikatorze.")

# Przykładowe dane do dodania
sample_data = {
    "student_id": 101,
    "student_name": "John",
    "student_surname": "Doe",
    "subjects": [
        {"subject": "Math", "grade": 85, "final": True},
        {"subject": "Science", "grade": 75, "final": True}
    ]
}

# Wywołanie operacji CRUD
create_document(sample_data)
read_all_documents()
read_document(101)
update_document(101, {"student_name": "Jane"})
delete_document(101)

# Wyświetlenie wszystkich dokumentów po operacjach CRUD
print("Po operacjach CRUD:")
read_all_documents()

# Zamknięcie połączenia z klientem MongoDB
client.close()
```

Pytania kontrolne

1. Jakie są podstawowe operatory porównania używane do filtrowania danych w MongoDB?

Podstawowe operatory porównania używane do filtrowania danych w MongoDB to: $eq (równy), $ne (nierówny), $gt (większy niż), $lt (mniejszy niż), $gte (większy lub równy), $lte (mniejszy lub równy).

2. Jakie są korzyści z zastosowania indeksów w kontekście filtrowania i sortowania w MongoDB?

Korzyści z zastosowania indeksów w MongoDB w kontekście filtrowania i sortowania to przyspieszenie operacji odczytu danych, zmniejszenie czasu odpowiedzi zapytań oraz optymalizacja wydajności operacji wyszukiwania.

3. Jak można zastosować sortowanie do wyników zapytania w MongoDB?

Sortowanie wyników zapytania w MongoDB można zastosować za pomocą metody sort(), która przyjmuje parametr określający kierunek sortowania (rosnąco: 1, malejąco: -1) oraz pola, według którego sortujemy.

4. Jakie operatory logiczne można użyć do łączenia warunków filtrowania w MongoDB?

Do łączenia warunków filtrowania w MongoDB można użyć operatorów logicznych $and, $or i $not.

5. Jakie są różnice między sortowaniem rosnącym a malejącym w MongoDB?

Różnica między sortowaniem rosnącym a malejącym w MongoDB polega na tym, że sortowanie rosnące (asc) ustawia wyniki zapytania w kolejności od najmniejszego do największego, podczas gdy sortowanie malejące (desc) ustawia wyniki w odwrotnej kolejności.

6. Które pola w kolekcji powinny być zazwyczaj zindeksowane, aby poprawić wydajność zapytań filtrowania?

Pola, które często są filtrowane, sortowane lub używane w zapytaniach, powinny być zazwyczaj zindeksowane, aby poprawić wydajność zapytań filtrowania.

7. Jakie są różnice między operatorem $eq a operatorem $in w kontekście filtrowania w MongoDB?

Różnica między operatorem $eq a operatorem $in w kontekście filtrowania w MongoDB polega na tym, że $eq porównuje pole z jedną wartością, podczas gdy $in porównuje pole z tablicą wartości.

8. W jaki sposób można łączyć operacje filtrowania i sortowania w jednym zapytaniu?

Aby łączyć operacje filtrowania i sortowania w jednym zapytaniu w MongoDB, możemy użyć metody sort() po metodzie find(), połączonej z metodą sort().

9. Jak sprawdzić, czy dana kolekcja posiada indeks na określonym polu?

Aby sprawdzić, czy dana kolekcja posiada indeks na określonym polu, możemy użyć metody listIndexes() na kolekcji.

10. W jaki sposób efektywnie przeprowadzić zapytania, które wymagają filtrowania po kilku kryteriach jednocześnie w MongoDB?

Aby efektywnie przeprowadzić zapytania, które wymagają filtrowania po kilku kryteriach jednocześnie w MongoDB, można łączyć warunki za pomocą operatorów logicznych ($and, $or) oraz stosować odpowiednie indeksy dla pól używanych w filtrowaniu.