Dokumentowe bazy danych – MongoDB

ćwiczenie 2

Imiona i nazwiska autorów:

1. Bartłomiej Szubiak
2. Szymon Kubiczek
3. Konrad Armatys

Yelp Dataset

- www.yelp.com - serwis społecznościowy – informacje o miejscach/lokalach
- restauracje, kluby, hotele itd. `businesses`,
- użytkownicy odwiedzają te miejsca - "meldują się" `check-in`
- użytkownicy piszą recenzje `reviews` o miejscach/lokalach i wystawiają oceny oceny,
- przykładowy zbiór danych zawiera dane z 5 miast: Phoenix, Las Vegas, Madison, Waterloo i Edinburgh.

Zadanie 1 - operacje wyszukiwania danych

Dla zbioru Yelp wykonaj następujące zapytania

W niektórych przypadkach może być potrzebne wykorzystanie mechanizmu Aggregation Pipeline

https://www.mongodb.com/docs/manual/core/aggregation-pipeline/

1. Zwróć dane wszystkich restauracji (kolekcja `business`, pole `categories` musi zawierać wartość "Restaurants"), które są otwarte w poniedziałki (pole hours) i mają ocenę co najmniej 4 gwiazdki (pole `stars`). Zapytanie powinno zwracać: nazwę firmy, adres, kategorię, godziny otwarcia i gwiazdki. Posortuj wynik wg nazwy firmy.
2. Ile każda firma otrzymała ocen/wskazówek (kolekcja `tip`) w 2012. Wynik powinien zawierać nazwę firmy oraz liczbę ocen/wskazówek Wynik posortuj według liczby ocen (`tip`).
3. Recenzje mogą być oceniane przez innych użytkowników jako `cool`, `funny` lub `useful` (kolekcja `review`, pole `votes`, jedna recenzja może mieć kilka głosów w każdej kategorii). Napisz zapytanie, które zwraca dla każdej z tych kategorii, ile sumarycznie recenzji zostało oznaczonych przez te kategorie (np. recenzja ma kategorię `funny` jeśli co najmniej jedna osoba zagłosowała w ten sposób na daną recenzję)
4. Zwróć dane wszystkich użytkowników (kolekcja `user`), którzy nie mają ani jednego pozytywnego głosu (pole `votes`) z kategorii (`funny` lub `useful`), wynik posortuj alfabetycznie według nazwy użytkownika.
5. Wyznacz, jaką średnia ocenę uzyskała każda firma na podstawie wszystkich recenzji (kolekcja `review`, pole `stars`). Ogranicz do firm, które uzyskały średnią powyżej 3 gwiazdek.

   a) Wynik powinien zawierać id firmy oraz średnią ocenę. Posortuj wynik wg id firmy.

   b) Wynik powinien zawierać nazwę firmy oraz średnią ocenę. Posortuj wynik wg nazwy firmy.

Zadanie 1 - rozwiązanie

```
// zad 1
// zakładamy że Monday.open < Monday.close
db.business
  .find(
    {
      categories: "Restaurants",
      "hours.Monday": { $exists: true },
      stars: { $gte: 4.0 },
    },
    {
      _id: false,
      name: true,
      full_address: true,
      categories: true,
      hours: true,
      stars: true,
    }
  )
  .sort({ name: 1 });

// zad 2
db.business.aggregate([
  {
    $lookup: {
      from: "tip",
      let: {
        business_id: "$business_id",
      },
      pipeline: [
        {
          $match: {
            $expr: {
              $and: [
                { $ne: [{ $type: "$date" }, false] },
                { $regexMatch: { input: "$date", regex: /^2012/ } },
                { $eq: ["$business_id", "$$business_id"] },
```

```
              ],
            },
          },
        },
      ],
      as: "tips",
    },
  },
  {
    $group: {
      _id: "$business_id",
      totalTips: { $sum: { $size: "$tips" } },
      name: { $first: "$name" },
    },
  },
  {
    $project: {
      _id: 0,
      name: 1,
      totalTips: 1,
    },
  },
  {
    $sort: {
      totalTips: -1,
    },
  },
]);

// zad 3
db.review.aggregate([
  {
    $match: {
      "votes.useful": { $gte: 1 },
    },
  },
  {
    $group: {
      _id: null,
      count: { $sum: 1 },
    },
  },
  {
    $project: {
      _id: 0,
      count: 1,
    },
  },
]);

// zad 4
db.user
  .find({
    $and: [{ "votes.funny": { $eq: 0 } }, { "votes.useful": { $eq: 0 } }],
  })
  .sort({ name: 1 });

// zad 5 a)
db.review.aggregate([
  {
    $group: {
      _id: "$business_id",
      avg_rating: { $avg: "$stars" },
    },
  },
  {
    $match: {
      avg_rating: { $gt: 3 },
    },
  },
  {
    $project: {
      _id: 1,
      avg_rating: 1,
    },
  },
  {
    $sort: {
      _id: 1,
    },
  },
]);

// zad 5 b)
db.review.aggregate([
  {
    $lookup: {
      from: "business",
      localField: "business_id",
      foreignField: "business_id",
```

```
      as: "businesses",
    },
  },
  {
    $addFields: {
      name: "$businesses.name",
    },
  },
  {
    $unwind: "$name",
  },
  {
    $group: {
      _id: "$business_id",
      avg_rating: { $avg: "$stars" },
    },
  },
  {
    $match: {
      avg_rating: { $gt: 3 },
    },
  },
  {
    $project: {
      _id: 0,
      name: 1,
      avg_rating: 1,
    },
  },
  {
    $sort: {
      name: 1,
    },
  },
]);
```

Zadanie 2 - modelowanie danych

Zaproponuj strukturę bazy danych dla wybranego/przykładowego zagadnienia/problemu

Należy wybrać jedno zagadnienie/problem (A lub B)

Przykład A

- Wykładowcy, przedmioty, studenci, oceny
  - Wykładowcy prowadzą zajęcia z poszczególnych przedmiotów
  - Studenci uczęszczają na zajęcia
  - Wykładowcy wystawiają oceny studentom
  - Studenci oceniają zajęcia

Przykład B

- Firmy, wycieczki, osoby
  - Firmy organizują wycieczki
  - Osoby rezerwują miejsca/wykupują bilety
  - Osoby oceniają wycieczki

a) Warto zaproponować/rozważyć różne warianty struktury bazy danych i dokumentów w poszczególnych kolekcjach oraz przeprowadzić dyskusję każdego wariantu (wskazać wady i zalety każdego z wariantów)

b) Kolekcje należy wypełnić przykładowymi danymi

c) W kontekście zaprezentowania wad/zalet należy zaprezentować kilka przykładów/zapytań/zadań/operacji oraz dla których dedykowany jest dany wariantów

W sprawozdaniu należy zamieścić przykładowe dokumenty w formacie JSON (pkt a) i b)), oraz kod zapytań/operacji (pkt c)), wraz z odpowiednim komentarzem opisującym strukturę dokumentów oraz polecenia ilustrujące wykonanie przykładowych operacji na danych

Do sprawozdania należy kompletny zrzut wykonanych/przygotowanych baz danych (taki zrzut można wykonać np. za pomocą poleceń `mongoexport`, `mongdump` ...) oraz plik z kodem operacji zapytań (załącznik powinien mieć format zip).

Zadanie 2 - rozwiązanie przykład B (firmy, wycieczki, osoby)

Wariant I: 4 powiązane kolekcje

Kolekcja Companies

- `_id`: unikalny identyfikator firmy
- `name`: nazwa firmy
- `adress`: adres firmy **obiekt**
  - `country`: kraj
  - `postalCode`: adres pocztowy
  - `city`: miasto
  - `street`: ulica z adresem
- `trips`: tablica odwołań do wycieczek organizowanych przez tę firmę

Kolekcja Trips

- `_id`: unikalny identyfikator wycieczki
- `name`: nazwa wycieczki
- `description`: opis wycieczki
- `date`: data wycieczki
- `place`: miejsce, które odwiedza wycieczka
- `price`: cena wycieczki
- `ratings`: lista ocen wystawionych przez osoby
  - `user_id`: unikalny identyfikator osoby co wystawiła recenzje
  - `rating`: ocena
  - `comment`: komentarz

Kolekcja Persons

- `_id`: unikalny identyfikator osoby
- `name`: imię osoby
- `surname`: nazwisko osoby
- `adress`: adres osoby **obiekt**
  - `country`: kraj
  - `postalCode`: adres pocztowy
  - `city`: miasto
  - `street`: ulica z adresem

Kolekcja Reservations

- `_id`: unikalny identyfikator rezerwacji
- `person_id`: odwołanie do osoby, która dokonała rezerwacji
- `trip_id`: odwołanie do wycieczki, na którą została dokonana rezerwacja
- `seats_no`: liczba miejsc zarezerwowanych przez osobę
- `date`: data dokonania rezerwacji

Wariant II: Pojedyńcza hierarchiczna kolekcja

Kolekcja trips

- `_id`: unikalny identyfikator wycieczki
- `name`: nazwa wycieczki
- `description`: opis wycieczki
- `maxParticipantsNo`: Maksymalna liczba osób
- `company`: firma oferująca wycieczkę **obiekt**
  - `_id`: unikalny identyfikator firmy
  - `adress`: adres firmy **obiekt**
    - `country`: kraj
    - `postalCode`: adres pocztowy
    - `city`: miasto
    - `street`: ulica z adresem
- `date`: data wycieczki
- `place`: miejsce, do którego udaje się wycieczka
- `price`: cena wycieczki
- `ratings`: lista ocen wystawionych przez osoby
  - `rating`: ocena
  - `comment`: komentarz
- `participants`: tablica uczestników wycieczki, która zawiera **obiekty postaci**
  - `reservationDate`: data rezerwacji
  - `reservationId`: id rezerwacji
  - `_id`: unikalny identyfikator osoby
  - `name`: imię osoby
  - `surname`: nazwisko osoby
  - `adress`: adres osoby **obiekt**
    - `country`: kraj
    - `postalCode`: adres pocztowy
    - `city`: miasto
    - `street`: ulica z adresem

Wady i zalety poszczególnych wariantów

Wariant I:

- Wymaga bardziej złozonych zapytań, by otrzymać jakieś dane
- Zapobiega powielaniu danych
- Danymi jednej z kolekcji mozna zarządzać nie ingerując w dane innych kolekcji
- Wariant ten zapewnia większe bezpieczeństwo danych

Wariant II:

- Struktura bazy danych jest bardziej czytelna i swoją budową wskazuje na hierarchię obiektów
- Istnieje ryzyko przepełnienia maksymalnego rozmiaru dokumentu, choć w przypadku problemu firm i wycieczek nie powinno do tego dojść
- Mogą występować zduplikowane dane
- reservationId trzeba zewnętrznie poprawnie podawać

Decydując się na I wariant:

Inicjalizacja:

```
use travelAgency_db
// Stworzenie kolekcji 'Companies'
db.createCollection("Companies")

// Stworzenie kolekcji 'Trips'
db.createCollection("Trips")

// Stworzenie kolekcji 'Persons'
db.createCollection("Persons")

// Stworzenie kolekcji 'Reservations'
db.createCollection("Reservations")
```

Wstawienie przykładowych dokumentów do kolekcji 'Companies':

```
db.Companies.insertMany([
  {
    name: "Adventure Travel Agency",
    address: {
      country: "United States",
      postalCode: "90210",
      city: "Beverly Hills",
      street: "Rodeo Drive",
    },
    trips: [],
  },
  {
    name: "European Excursions Ltd.",
    address: {
      country: "United Kingdom",
      postalCode: "SW1A 1AA",
      city: "London",
      street: "Buckingham Palace Road",
    },
    trips: [],
  },
  {
    name: "Exotic Destinations Inc.",
    address: {
      country: "Australia",
      postalCode: "2000",
      city: "Sydney",
      street: "George Street",
    },
    trips: [],
  },
  {
    name: "Tropical Tours LLC",
    address: {
      country: "Costa Rica",
      postalCode: "10101",
      city: "San José",
      street: "Avenida Central",
    },
    trips: [],
  },
]);
```

Wstawienie przykładowych dokumentów do kolekcji 'Trips'

```
db.Trips.insertMany([
  {
    name: "Wakacje nad morzem",
    description: "Relaksująca wycieczka nad Morzem Bałtyckim",
    date: new Date("2024-07-01"),
    place: "Morze Bałtyckie",
    price: 500,
    ratings: [],
  },
  {
    name: "Wycieczka po Europie",
    description: "Niezapomniana podróż po najpiękniejszych miastach Europy",
    date: new Date("2024-09-15"),
    place: "Europa",
    price: 2000,
    ratings: [],
  },
  {
    name: "Wyprawa w góry",
    description: "Ekscytująca wspinaczka w Alpach",
```

```
    date: new Date("2024-08-10"),
    place: "Alpy",
    price: 800,
    ratings: [],
  },
  {
    name: "Safari w Afryce",
    description: "Safari po Parku Narodowym Serengeti",
    date: new Date("2024-10-20"),
    place: "Tanzania",
    price: 1500,
    ratings: [],
  },
]);
```

Wstawienie przykładowych dokumentów do kolekcji 'Persons'

```
db.Persons.insertMany([
  {
    name: "Jan",
    surname: "Kowalski",
    address: {
      country: "Poland",
      postalCode: "01-001",
      city: "Kraków",
      street: "ul. Floriańska 10",
    },
  },
  {
    name: "Anna",
    surname: "Nowak",
    address: {
      country: "France",
      postalCode: "75001",
      city: "Paris",
      street: "Avenue des Champs-Élysées",
    },
  },
  {
    name: "Mark",
    surname: "Smith",
    address: {
      country: "United States",
      postalCode: "10001",
      city: "New York",
      street: "Broadway",
    },
  },
  {
    name: "Maria",
    surname: "Garcia",
    address: {
      country: "Spain",
      postalCode: "28001",
      city: "Madrid",
      street: "Calle de Alcalá 1",
    },
  },
]);
```

Dodanie 'referencji' do wycieczek w kolekcji 'Companies'

```
db.Companies.updateOne(
  { name: "Adventure Travel Agency" },
  { $push: { trips: db.Trips.findOne({ name: "Wakacje nad morzem" })._id } }
);

db.Companies.updateOne(
  { name: "European Excursions Ltd." },
  { $push: { trips: db.Trips.findOne({ name: "Wycieczka po Europie" })._id } }
);

db.Companies.updateOne(
  { name: "Exotic Destinations Inc." },
  { $push: { trips: db.Trips.findOne({ name: "Wyprawa w góry" })._id } }
);

db.Companies.updateOne(
  { name: "Tropical Tours LLC" },
  { $push: { trips: db.Trips.findOne({ name: "Safari w Afryce" })._id } }
);
```

Dodanie przykładowych rezerwacji do kolekcji 'Reservations':

```
db.Reservations.insertMany([
  {
    person_id: db.Persons.findOne({ name: "Anna", surname: "Nowak" })._id,
    trip_id: db.Trips.findOne({ name: "Wyprawa w góry" })._id,
    seats_no: 1,
    date: new Date("2023-08-15"),
  },
  {
    person_id: db.Persons.findOne({ name: "Mark", surname: "Smith" })._id,
    trip_id: db.Trips.findOne({ name: "Safari w Afryce" })._id,
    seats_no: 3,
    date: new Date("2023-09-20"),
  },
  {
    person_id: db.Persons.findOne({ name: "Maria", surname: "Garcia" })._id,
    trip_id: db.Trips.findOne({ name: "Wakacje nad morzem" })._id,
    seats_no: 4,
    date: new Date("2023-10-25"),
  },
]);
```

Przykładowe operacje na bazie danych wykorzystujące własności wariantu I

1. Baza danych jest rozdzielona na kilka kolekcji, a relacje między jej dokumentami mozna modelować na bieząco

```
// Niech jedna wycieczka bedzie realizowana przez dwie firmy
db.Companies.updateOne(
  { name: "Adventure Travel Agency" },
  { $push: { trips: db.Trips.findOne({ name: "Wycieczka po Europie" })._id } }
);
```

2. Przez to, ze traktujemy bazę danych jako kilka równoległych kolekcji, mozemy jako punkt wyjścia w jej przeszukiwaniu obrać dowolną kolekcję; W wariancie drugim jest to utrudnione, bo niektóre dane są zagniedzone

```
// Pobranie informacji o firmach wraz z listą wycieczek
db.Companies.aggregate([
  {
    $lookup: {
      from: "trips",
      localField: "_id",
      foreignField: "company._id",
      as: "trips",
    },
  },
]);
```

Decydując się na 2 wariant:

Inicjalizacja

```
use travelAgency_db
// Stworzenie kolekcji 'trips'
db.createCollection("trips")
```

Wstawienie przykładowych dokumentów do kolekcji 'trips'

```
// dodanie wycieczek do bazy danycha
db.trips.insertMany([
  {
    name: "Wycieczka nad morze",
    description: "Wspaniała wycieczka nad morze",
    maxParticipantsNo: 10,
    company: {
      _id: 1,
      address: {
        country: "Polska",
        postalCode: "00-001",
        city: "Warszawa",
        street: "Aleje Jerozolimskie 100",
      },
    },
    date: new Date("2024-05-15"),
    place: "Kołobrzeg",
    price: 200,
    ratings: [],
    participants: [],
  },
  {
    name: "Wycieczka w góry",
    description: "Wspaniała wycieczka w góry Tatry",
```

```
    maxParticipantsNo: 8,
    company: {
      _id: 2,
      address: {
        country: "Polska",
        postalCode: "00-002",
        city: "Kraków",
        street: "Rynek Główny 1",
      },
    },
    date: new Date("2024-06-20"),
    place: "Tatry",
    price: 300,
    ratings: [],
    participants: [],
  },
  {
    name: "Wycieczka do zoo",
    description: "Wspaniała wycieczka do zoo",
    maxParticipantsNo: 15,
    company: {
      _id: 3,
      address: {
        country: "Polska",
        postalCode: "00-003",
        city: "Łódź",
        street: "Piotrkowska 100",
      },
    },
    date: new Date("2024-07-10"),
    place: "Zoo w łodzi",
    price: 100,
    ratings: [],
    participants: [],
  },
]);
```

Dodanie osób do istniejących wycieczek

```
// dodanie uczestników do wycieczek
db.trips.updateOne(
  { _id: 1 },
  {
    $push: {
      participants: {
        reservationDate: new Date("2023-09-21"),
        reservationId: 1,
        _id: 1,
        name: "Jan",
        surname: "Kowalski",
        address: {
          country: "Polska",
          postalCode: "00-001",
          city: "Warszawa",
          street: "Nowa 1",
        },
      },
    },
  }
);

db.trips.updateOne(
  { _id: 1 },
  {
    $push: {
      participants: {
        reservationDate: new Date("2024-05-20"),
        reservationId: 1,
        _id: 2,
        name: "Anna",
        surname: "Nowak",
        address: {
          country: "Polska",
          postalCode: "00-002",
          city: "Kraków",
          street: "Stara 2",
        },
      },
    },
  }
);

db.trips.updateOne(
  { _id: 2 },
  {
    $push: {
      participants: {
        reservationDate: new Date("2024-06-20"),
```

```
        reservationId: 2,
        _id: 3,
        name: "Adam",
        surname: "Nowicki",
        address: {
          country: "Polska",
          postalCode: "00-003",
          city: "Gdańsk",
          street: "Plażowa 3",
        },
      },
    },
  }
);
```

Przykładowe operacje na bazie danych wykorzystujące własności wariantu II

1. Dzięki drzewiastej strukturze tego wariantu mozemy w szybki sposób otrzymywać kompletne dane o wycieczkach

```
// Pozyskanie kompletnych informacji o wycieczkach o podanych parametrach
db.trips.findOne({ _id: 1 });
db.trips.find({ date: ISODate("2024-04-30") });
```

2. Taka, a nie inna struktura bazy danych pozwala na szybką manipulację na duzych danych. Pokazuje to fakt, ze usuwając wycieczkę automatycznie pozbywamy się wszystkich informacji o niej

```
// Przykładowy fragment kodu z NodeJS; tripId to id wycieczki do usunięcia, a url to adres bazy danych
async function deleteTrip(tripId) {
  const client = new MongoClient(url);
  await client.connect();
  try {
    const db = client.db(dbName);
    await db.collection("trips").deleteOne({ _id: tripId });
    console.log(`Wycieczka o ID ${tripId} została usunięta.`);
  } finally {
    await client.close();
  }
}
```

Punktacja:

zadanie	pkt
1	0,6
2	1,4
razem	2