Dokumentowe bazy danych – MongoDB

Ćwiczenie 2 - zadanie do samodzielnego wykonania

Imię i nazwisko:

Materiały:

Książki

Np.

- Shannon Bradshaw, Eoin Brazil, Kristina Chodorow, MongoDB: The Definitive Guide. Powerful and Scalable Data Storage, O'Reily 2019
- Alex Giamas, Mastering MongoDB 4.x., Pact 2019

Dokumentacja

- https://www.mongodb.com/docs/manual/reference/program/mongo/

MongoDB University Courses

- https://university.mongodb.com/courses/catalog
- MongoDB Basics
  - https://university.mongodb.com/courses/M001/about
- The MongoDB Aggregation Framework
  - https://university.mongodb.com/courses/M121/about
- Data Modeling
  - https://university.mongodb.com/courses/M320/about

Yelp Dataset

www.yelp.com - serwis społecznościowy – informacje o miejscach/lokalach

- restauracje, kluby, hotele itd. (*businesses*),
- użytkownicy piszą recenzje (*reviews*) o miejscach i wystawiają oceny oceny,
- użytkownicy odwiedzają te miejsca - "meldują się" (*check-in*)
- Przykładowy zbiór danych zawiera dane z 5 miast: Phoenix, Las Vegas, Madison, Waterloo i Edinburgh.

Kolekcje:

- **42,153** businesses
- **320,002** business attributes
- **31,617** check-in sets
- **252,898** users
- **955,999** edge social graph
- **403,210** tips
- **1,125,458** reviews

business

```
{
    'type': 'business',
    'business_id': (encrypted business id),
    'name': (business name),
    'neighborhoods': [(hood names)],
    'full_address': (localized address),
    'city': (city),
    'state': (state),
    'latitude': latitude,
    'longitude': longitude,
    'stars': (star rating, rounded to half-stars),
    'review_count': review count,
    'categories': [(localized category names)]
    'open': True / False (corresponds to closed, not business hours),
    'hours': {
        (day_of_week): {
            'open': (HH:MM),
            'close': (HH:MM)
        },
        ...
    },
    'attributes': {
        (attribute_name): (attribute_value),
        ...
    },
}
```

review

```
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

user

```
{
    'type': 'user',
    'user_id': (encrypted user id),
    'name': (first name),
    'review_count': (review count),
    'average_stars': (floating point average, like 4.31),
    'votes': {(vote type): (count)},
    'friends': [(friend user_ids)],
    'elite': [(years_elite)],
    'yelping_since': (date, formatted like '2012-03'),
    'compliments': {
        (compliment_type): (num_compliments_of_this_type),
        ...
    },
    'fans': (num_fans),
}
```

check-in

```
{
    'type': 'checkin',
    'business_id': (encrypted business id),
    'checkin_info': {
        '0-0': (number of checkins from 00:00 to 01:00 on all Sundays),
        '1-0': (number of checkins from 01:00 to 02:00 on all Sundays),
        ...
        '14-4': (number of checkins from 14:00 to 15:00 on all Thursdays),
        ...
        '23-6': (number of checkins from 23:00 to 00:00 on all Saturdays)
    }, # if there was no checkin for a hour-day block it will not be in the dict
}
```

tip

```
{
    'type': 'tip',
    'text': (tip text),
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'date': (date, formatted like '2012-03-14'),
    'likes': (count),
}
```

Zadania

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/

a) Zwróć dane wszystkich restauracji (kolekcja businesss, 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.

Standardowe zapytanie spełniające kryteria podpunktu. Metoda `find()` składa się z 3 parametrów: zapytania, widoku zapytania oraz dodatkowych opcji, gdzie możemy np. posortować wyniki. Poniżej fragment otrzymanych dokumentów.

```
db.getCollection('business').find(
    {
        'categories': { $elemMatch: { $eq: 'Restaurants' } },
        'hours.Monday': { $exists: true },
        'stars': { $gte: 4 }
    },
    {
        'name': true,
        'full_address': true,
        'categories': true,
        'hours': true,
        'stars': true
    },
    {
        sort: [ 'name' ]
    }
);
```

```
yelp_mongodb_js> db.getCollection('business').find(
...     {
...         'categories': { $elemMatch: { $eq: 'Restaurants' } },
...         'hours.Monday': { $exists: true },
...         'stars': { $gte: 4 }
...     },
...     {
...         'name': true,
...         'full_address': true,
...         'categories': true,
...         'hours': true,
...         'stars': true
...     },
...     {
...         sort: [ 'name' ]
...     }
... );
[
  {
    _id: ObjectId("647e58bf00c163c7414ded87"),
    full_address: '67 Nicolson Street\nNewington\nEdinburgh EH8 9BZ',
    hours: {
      Monday: { close: '22:00', open: '10:00' },
      Tuesday: { close: '22:00', open: '10:00' },
      Friday: { close: '22:00', open: '10:00' },
      Wednesday: { close: '22:00', open: '10:00' },
      Thursday: { close: '22:00', open: '10:00' },
      Sunday: { close: '22:00', open: '10:00' },
      Saturday: { close: '22:00', open: '10:00' }
    },
    categories: [ 'Food', 'Desserts', 'Coffee & Tea', 'Indian', 'Restaurants' ],
    name: '10-to-10 In Delhi',
    stars: 4.5
  },
```

b) Ile hoteli znajduje się w każdym mieście. (pole *categories* musi zawierać wartość *Hotels & Travel* lub *Hotels)*. Wynik powinien zawierać nazwę miasta, oraz liczbę hoteli. Posortuj wynik malejąco wg liczby hoteli.

W tym zapytaniu korzystam z metody `aggregate()`, która przyjmuje tablicę obiektów odpowiadających za agregację zapytania. `$match` to filtr kategorii hoteli, natomiast `$group`, jak sama nazwa wskazuje, to agregator grupy – grupuje każde przedsiębiorstwo po mieście i zlicza liczbę takich hoteli w obrębie każdej grupy, a na koniec sortuje malejąco po tej liczbie malejąco, korzystając z `$sort`.

```
db.getCollection('business').aggregate(
    [
        { $match: { 'categories': {
            $in: [ 'Hotels & Travel', 'Hotels' ]
        } } },
        { $group: {
            '_id': '$city',
            'hotel_count': { $count: {} }
        } },
        { $sort: { 'hotel_count': -1 } }
    ]
);
```

```
yelp_mongodb_js> db.getCollection('business').aggregate(
...     [
...         { $match: { 'categories': { $in: [ 'Hotels & Travel', 'Hotels' ] } } },
...         { $group: { '_id': '$city', 'hotel_count': { $count: {} } } },
...         { $sort: { 'hotel_count': -1 } }
...     ]
... );
[
  { _id: 'Las Vegas', hotel_count: 485 },
  { _id: 'Phoenix', hotel_count: 250 },
  { _id: 'Edinburgh', hotel_count: 161 },
  { _id: 'Scottsdale', hotel_count: 122 },
  { _id: 'Madison', hotel_count: 67 },
  { _id: 'Tempe', hotel_count: 57 },
  { _id: 'Mesa', hotel_count: 53 },
  { _id: 'Henderson', hotel_count: 41 },
  { _id: 'Chandler', hotel_count: 30 },
  { _id: 'Glendale', hotel_count: 20 },
  { _id: 'Peoria', hotel_count: 12 },
  { _id: 'North Las Vegas', hotel_count: 12 },
  { _id: 'Surprise', hotel_count: 10 },
  { _id: 'Goodyear', hotel_count: 9 },
  { _id: 'Middleton', hotel_count: 7 },
  { _id: 'Casa Grande', hotel_count: 7 },
  { _id: 'Gila Bend', hotel_count: 6 },
  { _id: 'Wickenburg', hotel_count: 5 },
  { _id: 'Waterloo', hotel_count: 5 },
  { _id: 'Avondale', hotel_count: 5 }
]
Type "it" for more
```

c) Ile każda firma otrzymała ocen/wskazówek (kolekcja *tip*) w 2012. Wynik powinien zawierać nazwę firmy oraz liczbę ocen/wskazówek Wynik posortuj według liczby wskazówek (*tip*).

Filtruję za pomocą wyrażenia regularnego wszystkie dokumenty, które mają w dacie 2012 rok (posiadają na początku tekstu `2012`). Grupuję i zliczam wszystkie przefiltrowane wskazówki, dołączam kolekcję `business` do zwróconych wyników, aby wydobyć nazwę firmy (odpowiednik `JOIN` z relacyjnych baz danych), następnie sortuję malejąco po liczbie sugestii, a na końcu zmieniam widok, żeby zawierał tylko pola: id, nazwę firmy oraz liczbę posiadanych wskazówek. W tym podpunkcie można zauważyć, że kolejność agregacji ma olbrzymie znaczenie na wydajność zapytania, w tym przypadku złe ułożenie przyczyniało się do wydłużenia czasu obliczania zapytania do ponad 15 minut, np. `$sort` po `$project` z `$lookup`.

```
db.getCollection('tip').aggregate([
    { $match: { 'date': { $regex: /^2012/ } } },
    { $group: {
        '_id': '$business_id',
        'tip_count': { $count: {} }
    } },
    {
        $lookup: {
            from: 'business',
            localField: '_id',
            foreignField: 'business_id',
            as: 'business'
        }
    },
    { $sort: { 'tip_count': -1 } },
    {
        $project: {
            'business_name': { $first: '$business.name' },
            'tip_count': 1
        }
    }
]);
```

```
yelp_mongodb_js> db.getCollection('tip').aggregate([
...     { $match: { 'date': { $regex: /^2012/ } } },
...     { $group: {
...         '_id': '$business_id',
...         'tip_count': { $count: {} }
...     } },
...     {
...         $lookup: {
...             from: 'business',
...             localField: '_id',
...             foreignField: 'business_id',
...             as: 'business'
...         }
...     },
...     { $sort: { 'tip_count': -1 } },
...     { $project: { 'business_name': { $first: '$business.name' }, 'tip_count': 1 } }
... ]);
[
  {
    _id: 'jf67Z1pnwElRSXllpQHiJg',
    tip_count: 1084,
    business_name: 'McCarran International Airport'
  },
  {
    _id: 'hW0Ne_HTHEAgGF1rAdmR-g',
    tip_count: 622,
    business_name: 'Phoenix Sky Harbor International Airport'
  },
  {
    _id: '2e2e7WgqU1BnpxmQL5jbfw',
    tip_count: 430,
    business_name: 'Earl of Sandwich'
  },
  {
    _id: 'CsNOg-u_wCuXSt9Z-xU92Q',
    tip_count: 374,
    business_name: 'Las Vegas Athletic Club Southwest'
  },
```

d) 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)

Zamieniam obiekt `votes` na tablicę klucz-wartość, następnie rozpakowywuję tablicę za pomocą `$unwind`, pozbywam się wszystkich dokumentów z liczbą ocen z danej kategorii równą 0. `count` to liczba wszystkich niezerowych ocen, natomiast `total` sumuje liczbę wszystkich ocen, tj. jeśli w jednym dokumencie `v` będzie miało wartość 2, a w drugim 3 (`k` będzie równe) to `count` zwróci 2, a `total` 5.

```
db.getCollection('review').aggregate([
    {
        $project: {
            'votes': { $objectToArray: '$votes' }
        }
    },
    { $unwind: '$votes' },
    { $match: { 'votes.v': { $gt: 0 } } },
    {
        $group: {
            '_id': '$votes.k',
            'count': { $count: {} },
            'total': { $sum: '$votes.v' }
        }
    }
]);
```

```
yelp_mongodb_js> db.getCollection('review').aggregate([
...     {
...         $project: {
...             'votes': { $objectToArray: '$votes' }
...         }
...     },
...     { $unwind: '$votes' },
...     { $match: { 'votes.v': { $gt: 0 } } },
...     {
...         $group: {
...             '_id': '$votes.k',
...             'count': { $count: {} },
...             'total': { $sum: '$votes.v' }
...         }
...     }
... ]);
[
  { _id: 'useful', count: 197572, total: 443648 },
  { _id: 'cool', count: 120644, total: 253364 },
  { _id: 'funny', count: 95750, total: 209946 }
]
```

e) 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.

```
db.getCollection('user').aggregate([
    {
        $match: {
            'votes.funny': 0,
            'votes.useful': 0,
            'type': 'user'
        }
    },
    { $sort: { 'name': 1 } }
]);
```

```
yelp_mongodb_js> db.getCollection('user').aggregate([
...     {
...         $match: {
...             'votes.funny': 0,
...             'votes.useful': 0,
...             'type': 'user'
...         }
...     },
...     { $sort: { 'name': 1 } }
... ]);
[
  {
    _id: ObjectId("647e5b9300c163c7415b8bdd"),
    yelping_since: '2009-08',
    votes: { funny: 0, useful: 0, cool: 0 },
    review_count: 1,
    name: ' Bernard',
    user_id: 'xP3SPgfgW2vc5Zj5uV8SEA',
    friends: [],
    fans: 0,
    average_stars: 5,
    type: 'user',
    compliments: {},
    elite: []
  },
  {
    _id: ObjectId("647e5bc600c163c7415df5f3"),
    yelping_since: '2013-03',
    votes: { funny: 0, useful: 0, cool: 0 },
    review_count: 1,
    name: ',Maria',
    user_id: 'Os5f3TNpM7_A8IDNEdPX2g',
    friends: [],
    fans: 0,
    average_stars: 5,
    type: 'user',
    compliments: {},
    elite: []
  },
```

f) Wyznacz, jaką średnia ocenę uzyskała każda firma na podstawie wszystkich recenzji (kolekcja *review*, pole *stars*). Ogranicz do firm, które uzyskały średnią powyżej 3 gwiazdek.

przypadek 1: Wynik powinien zawierać id firmy oraz średnią ocenę. Posortuj wynik wg id firmy.

Na początku grupuję po polu `business_id`, agregując dodatkowo średnią opinii, wyświetlam tylko te opinie, które posiadają średnią większą niż 3, a na koniec sortuję wyniki po id firmy (`_id`).

```
db.getCollection('review').aggregate([
    {
        $group: {
            '_id': '$business_id',
            'stars_mean': { $avg: '$stars' }
        }
    },
    { $match: { 'stars_mean': { $gt: 3 } } },
    { $sort: { '_id': 1 } }
]);
```

```
yelp_mongodb_js> db.getCollection('review').aggregate([
...     {
...         $group: {
...             '_id': '$business_id',
...             'stars_mean': { $avg: '$stars' }
...         }
...     },
...     { $match: { 'stars_mean': { $gt: 3 } } },
...     { $sort: { '_id': 1 } }
... ]);
[
  { _id: '--XBxRlD92RaV6TyUnP8Ow', stars_mean: 3.6666666666666665 },
  { _id: '-0HGqwlfw3I8nkJyMHxAsQ', stars_mean: 4 },
  { _id: '-0QBrNvhrPQCaeo7mTo0zQ', stars_mean: 4.333333333333333 },
  { _id: '-1bOb2izeJBZjHC7NWxiPA', stars_mean: 3.824324324324324 },
  { _id: '-34jE_5dujSWMIOBudQsiQ', stars_mean: 4.75 },
  { _id: '-3JXOT-i2gDbASLgDeOSwQ', stars_mean: 4.75 },
  { _id: '-3WVw1TNQbPBzaKCaQQ1AQ', stars_mean: 3.7892156862745097 },
  { _id: '-3qWPkQAxsggQJYqgi5myA', stars_mean: 3.25 },
  { _id: '-3xbryp44xhpN4BohxXDdQ', stars_mean: 3.8151260504201683 },
  { _id: '-4BtfPW3_v092G8pqHrv4w', stars_mean: 3.625 },
  { _id: '-4zMr3jk0ykmsk5sxsQMlA', stars_mean: 4 },
  { _id: '-5IgAihccTd8_iENVgpd9A', stars_mean: 3.5 },
  { _id: '-5bLhMLNGbIVyoPOVl0yLg', stars_mean: 3.3333333333333335 },
  { _id: '-63VfA2tnYyzwRt81B1AKw', stars_mean: 4.0588235294117645 },
  { _id: '-65rmLRQ5JDwI-l8UDEV7Q', stars_mean: 3.8 },
  { _id: '-6Ft3hulif7O2sIdKiG0Og', stars_mean: 4.5 },
  { _id: '-6O53B-ksqSKzWM6Y9moEQ', stars_mean: 3.25 },
  { _id: '-6Roo-EHgSdUa4rP3tWyRw', stars_mean: 3.4571428571428573 },
  { _id: '-6c7Tfec-X0aDmzjKO6R-Q', stars_mean: 3.1666666666666665 },
  { _id: '-7DxAxnXkAlTXUrtByoXAg', stars_mean: 4.666666666666667 }
]
```

przypadek 2: Wynik powinien zawierać nazwę firmy oraz średnią ocenę. Posortuj wynik wg nazwy firmy.

Do zapytania z przypadku 1 dodałem relację z kolekcją `business`, wyłuskując tylko nazwę firmy za pomocą `pipeline`. Czas oczekiwania na zapytanie jest bardzo długi z powodu natury zapytania i struktury bazy Yelp, mianowicie sortowanie i pozyskiwanie danych spoza bazowej kolekcji za pomocą `$lookup` zdecydowanie zmniejsza wydajność.

```
db.getCollection('review').aggregate([
    {
        $group: {
            '_id': '$business_id',
            'stars_mean': { $avg: '$stars' }
        }
    },
    { $match: { 'stars_mean': { $gt: 3 } } },
    {
        $lookup: {
            from: 'business',
            localField: '_id',
            foreignField: 'business_id',
            pipeline: [{ $group: { '_id': '$name' } }],
            as: 'business'
        }
    },
    {
        $project: {
            '_id': 0,
            'business_name': { $first: '$business._id' },
            'stars_mean': 1
        }
    },
    { $sort: { 'business_name': 1 } }
]);
```

```
yelp_mongodb_js> db.getCollection('review').aggregate([
...     {
...         $group: {
...             '_id': '$business_id',
...             'stars_mean': { $avg: '$stars' }
...         }
...     },
...     { $match: { 'stars_mean': { $gt: 3 } } },
...     {
...         $lookup: {
...             from: 'business',
...             localField: '_id',
...             foreignField: 'business_id',
...             pipeline: [{ $group: { '_id': '$name' } }],
...             as: 'business'
...         }
...     },
...     {
...         $project: {
...             '_id': 0,
...             'business_name': { $first: '$business._id' },
...             'stars_mean': 1
...         }
...     },
...     { $sort: { 'business_name': 1 } }
... ]);
[
  { stars_mean: 4.125, business_name: '#1 Brothers Pizza' },
  { stars_mean: 4.055555555555555, business_name: '19th Donut Hole' },
  { stars_mean: 4.375, business_name: '1st Emergency Pet Care' },
  { stars_mean: 4.928571428571429, business_name: '1st Nails II' },
  {
    stars_mean: 3.9444444444444446,
    business_name: '1st Pet Veterinary Centers'
  },
  {
    stars_mean: 3.0714285714285716,
    business_name: '2 Men and A Pizza'
  },
  { stars_mean: 3.5, business_name: '2007 Nails' },
  { stars_mean: 3.2666666666666666, business_name: '24 Hour Fitness' },
  { stars_mean: 3.5454545454545454, business_name: '24 Hour Fitness' },
```

W sprawozdaniu należy umieścić zrzuty ekranów (z kodem poleceń oraz z uzyskanymi wynikami). Dodatkowo należy dołączyć plik tekstowy (najlepiej z rozszerzeniem .js) zawierający kod poleceń

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

Wybieram model A – baza danych uczelni.

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)

Wariant 1

Struktura bazy danych jest podzielona na 3 kolekcje: wykładowców, przedmioty i studentów. W tym modelu położyłem nacisk na redundancję danych i spójność, stosując podobny do relacyjnych baz danych model.

Zalety:

- Redundancja danych;
- Spójność;
- Duża wydajność dla poszczególnych kolekcji;
- Niewielki rozmiar dokumentów.

Wady:

- Średnia lub mała wydajność dla poszczególnych kolekcji przy zapytaniach agregacyjnych, które łączą kilka kolekcji;
- Zamodelowanie typowych dla tej bazy danych zapytań wymaga już łączenia kilku kolekcji;
- Mała czytelność z powodu posiadanych licznych obiektów `ObjectId`.

```
// Lecturers
const exampleLecturer1 = {
    "personalDetails": {
        "firstName": "John",
        "lastName": "Doe",
        "realLifeId": "532214321",
        "nationality": "Scottish"
        // pozostałe dane, nr telefonu, email uczelniany itd.
        // ...
    },
    "faculty": "Faculty of Computer Science, Electronics and
Telecomunications",
    "degree": "Doctor of Philosophy",
    "roles": [
        "Head of the department"
        // ...
    ],
    "taughtCourses": [
        ObjectId("_id_course_0"),
        ObjectId("_id_course_1")
        // ...
    ]
};
```

```
// Courses
const exampleCourse1 = {
    "name": "Data Structures and Algorithms",
    "academicYear": [ 2021, 2022 ],
    "description": "Algorithms, yeah.",
    "participants": [
        ObjectId("_id_student_0"),
        ObjectId("_id_student_1"),
        ObjectId("_id_student_2"),
        ObjectId("_id_student_3"),
        ObjectId("_id_student_4")
        // ...
    ],
    "reviews": [
        {
            "value": 10,
            "date": new Date(),
            "comment": "Yeah, cool.",
            "issuedBy": ObjectId("_id_anne_kowalski"),
        }
    ]
};
```

```
// Students
const exampleStudent1 = {
    "personalDetails": {
        "firstName": "Anne",
        "lastName": "Kowalski",
        "realLifeId": "00211388482",
        "nationality": "Polish"
        // pozostałe dane, nr telefonu, email uczelniany itd.
        // ...
    },
    "major": "Computer Science",
    "faculty": "Faculty of Computer Science, Electronics and
Telecomunications",
    "participatedCourses": [
        ObjectId("_id_course_0"),
        ObjectId("_id_course_1"),
        ObjectId("_id_course_2")
        // ...
    ],
    "grades": [
        {
            "value": 4.5,
            "date": new Date(),
            "issuedBy": ObjectId("_id_john_doe"),
            "courseId": ObjectId("_id_dsa_course")
        }
        // ...
    ]
};
```

Wariant 2

Struktura bazy danych jest podzielona także na 3 kolekcje: wykładowców, przedmioty i studentów. W tym modelu położyłem nacisk na wydajność modelu, stosując zagnieżdżone dokumenty, które zawierają pełne informacje.

Zalety:

- Duża wydajność zapytań bez względu na ich rodzaj;
- Posiadanie wszystkich danych w jednej kolekcji bez potrzebnej agregacji;

Wady:

- Bardzo duża powtarzalność tych samych danych poprzez powielenie dokumentów;
- Bardzo duży rozmiar dokumentów jak i samej bazy;

- Integralność między danymi jest utrudniona, w wypadku modyfikacji pojedynczego dokumentu w kolekcji należy zmodyfikować inne, gdzie występuje ten sam dokument także.
- Mała czytelność przez ilość danych w pojedynczym dokumencie.

Wariant 3

Wariant ten to wariant 2 z redukowaną liczbą pól, tablice obiektów w postaci `grades`, `reviews`, itd. zostały usunięte.

Zalety:
- Duża wydajność zapytań większości standardowych zapytań;
- Posiadanie wszystkich kluczowych danych w jednej kolekcji bez potrzebnej agregacji;
- Czytelność.

Wady:
- Duża powtarzalność tych samych danych poprzez powielenie dokumentów;
- Duży rozmiar dokumentów jak i samej bazy;
- Integralność między danymi jest utrudniona, w wypadku modyfikacji pojedynczego dokumentu w kolekcji należy zmodyfikować inne, gdzie występuje ten sam dokument także.
- Czasami trzeba zastosować agregację danych, łącząc kilka kolekcji, by pozyskać odpowiednie informacje.

Wariant 4

Struktura bazy danych danych z poprzednich punktów zostaje zmieniona, powstają 2 nowe kolekcje w postaci ocen i opinii. W tym modelu można zastosować łączenie relacji poprzez ich identyfikatory lub częściowe zagnieżdżanie dokumentów tak jak w wariancie 3.

b) Kolekcje należy wypełnić przykładowymi danymi

Zrobiłem to w punkcie a.

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).

Punktacja za zadanie (razem 2pkt)