Lab 4

Rafał Włodarczyk

2024-12-18

Exercise 1 - Run with docker

```
docker pull mongodb/mongodb-community-server:latest
docker run --name mongodb -p 27017:27017 -d mongodb/mongodb-community-server:latest
mongosh --port 27017
```

Exercise 2 - Basic operations

1. Create a database with mongosh

```
use library # only creates db when data is stored
```

2. Create `authors` and `books` collection:

```
db.createCollection('authors')
db.createCollection('books')
show collections
```

3. Now insert into the `authors` collection.

```
db.authors.insertOne({
  "_id": ObjectId("65711ccdcb2d05e2c973fe85"),
  "name": { "first": "J.R.R", "last": "Tolkien" },
  "country": "UK",
  "birth": new Date('Jan 3, 1892'),
  "death": new Date('Sep 2, 1973')
})
```

4. Now insert into the `books` collection.

```
db.books.insertOne({
"_id": ObjectId("65712008cb2d05e2c973fe86"),
"title": "The Hobbit",
"isbn": "978-0-261-10295-6",
"publication_year": 1937,
"language": "English",
"author": ObjectId("65711ccdcb2d05e2c973fe85"),
"publisher":
    {
    "name": "George Allen & Unwin",
    "country": "UK"
    }
})
```

Now solve:

> Wstaw dane dotyczące czterech różnych autorów (co najmniej dwóch pochodzących z Polski) i dla każdego autora dane o kilku różnych napisanych przez niego książkach (dodaj co najmniej trzy książki tego samego autora napisane po polsku).

5. Lets insert 4 authors:

```
db.authors.insertMany([
  {
    "_id": ObjectId("65711ccdcb2d05e2c973fe87"),
    "name": { "first": "Henryk", "last": "Sienkiewicz" },
    "country": "Poland",
    "birth": new Date('May 5, 1846'),
    "death": new Date('Nov 15, 1916')
  },
  {
    "_id": ObjectId("65711ccdcb2d05e2c973fe88"),
    "name": { "first": "Adam", "last": "Mickiewicz" },
    "country": "Poland",
    "birth": new Date('Dec 24, 1798'),
    "death": new Date('Nov 26, 1855')
  },
  {
    "_id": ObjectId("65711ccdcb2d05e2c973fe89"),
    "name": { "first": "Juliusz", "last": "Słowacki" },
    "country": "Poland",
    "birth": new Date('Sep 4, 1809'),
    "death": new Date('Apr 3, 1849')
  },
  {
    "_id": ObjectId("65711ccdcb2d05e2c973fe90"),
    "name": { "first": "William", "last": "Shakespeare" },
    "country": "UK",
      "birth": new Date('Apr 26, 1564'),
      "death": new Date('Apr 23, 1616')
    }
  ])
```

6. Let's insert 5 books

```
db.books.insertMany([
  {
    "_id": ObjectId("65712008cb2d05e2c973fe87"),
    "title": "Quo Vadis",
    "isbn": "978-0-261-10295-6",
    "publication_year": 1896,
    "language": "Polish",
    "author": ObjectId("65711ccdcb2d05e2c973fe87"),
    "publisher": {
      "name": "Zakład Narodowy im. Ossolińskich",
      "country": "Poland"
    }
  },
  {
    "_id": ObjectId("65712008cb2d05e2c973fe88"),
    "title": "Krzyżacy",
    "isbn": "978-0-261-10295-6",
    "publication_year": 1900,
    "language": "Polish",
    "author": ObjectId("65711ccdcb2d05e2c973fe87"),
    "publisher": {
      "name": "Zakład Narodowy im. Ossolińskich",
      "country": "Poland"
    }
  },
  {
    "_id": ObjectId("65712008cb2d05e2c973fe89"),
    "title": "Potop",
    "isbn": "978-0-261-10295-6",
    "publication_year": 1886,
    "language": "Polish",
    "author": ObjectId("65711ccdcb2d05e2c973fe87"),
    "publisher": {
      "name": "Zakład Narodowy im. Ossolińskich",
      "country": "Poland"
    }
  },
  {
    "_id": ObjectId("65712008cb2d05e2c973fe90"),
    "title": "Pan Tadeusz",
    "isbn": "978-0-261-10295-6",
    "publication_year": 1834,
    "language": "Polish",
    "author": ObjectId("65711ccdcb2d05e2c973fe88"),
    "publisher": {
      "name": "Zakład Narodowy im. Ossolińskich",
      "country": "Poland"
    }
  }
])
```

7. Now solve

> Przetestuj, co się stanie, jeżeli data śmierci to null (nie jest znana), albo jeżeli:
>
> - pomylisz nazwę jednego pola,
> - podasz błędny typ wstawianych danych,
> - jedno pole zupełnie pominiesz.
>
> Czy można wstawić dokument o innym schemacie?
>
> Czy można wprowadzić schemat wymuszający poprawność danych?
>
> Jeśli tak, to w jaki sposób?

8. Let's insert a document with `death` set to `null`:

```
db.authors.insertOne({
  "_id": ObjectId("65711ccdcb2d05e2c973fe91"),
  "name": { "first": "Umberto", "last": "Eco" },
  "country": "Italy",
  "birth": new Date('Jan 5, 1932'),
  "death": null
})
```

It works.

It will not work after validator has been set.

Validator response:

```
MongoServerError: Document failed validation
Additional information: {
  failingDocumentId: ObjectId('65711ccdcb2d05e2c973fe91'),
  details: {
    operatorName: '$jsonSchema',
    schemaRulesNotSatisfied: [
      {
        operatorName: 'properties',
        propertiesNotSatisfied: [
          {
            propertyName: 'death',
            description: 'must be a date and is required',
            details: [
              {
                operatorName: 'bsonType',
                specifiedAs: { bsonType: 'date' },
                reason: 'type did not match',
                consideredValue: null,
                consideredType: 'null'
              }
            ]
          }
        ]
      }
    ]
  }
}
```

9. Let's insert a document with a typo in the field name:

```
db.authors.insertOne({
  "_id": ObjectId("65711ccdcb2d05e2c973fe92"),
  "name": { "first": "Umberto", "last": "Eco" },
  "country": "Italy",
  "birth": new Date('Jan 5, 1932'),
  "death": new Date('Feb 19, 2016'),
  "deat": new Date('Feb 19, 2016')
})
```

It works.

10. Let's insert a document with a wrong type of data:

```
db.authors.insertOne({
  "_id": ObjectId("65711ccdcb2d05e2c973fe93"),
  "name": { "first": "Umberto", "last": "Eco" },
  "country": "Italy",
  "birth": new Date('Jan 5, 1932'),
  "death": new Date('Feb 19, 2016'),
  "deat": "Feb 19, 2016"
})
```

It works

11. Let's insert a document with a missing field:

```
db.authors.insertOne({
  "_id": ObjectId("65711ccdcb2d05e2c973fe94"),
  "name": { "first": "Umberto", "last": "Eco" },
  "birth": new Date('Jan 5, 1932'),
  "death": new Date('Feb 19, 2016')
})
```

It works

12. In order to impose a schema on the data, we can use JSON schema validation. We can create a schema for the `authors` collection:

```
# remove collection
db.authors.drop()

db.createCollection('authors',
{
  validator: {
    $jsonSchema: {
      bsonType: "object",
        required: [
          "name",
          "country",
          "birth",
          "death"
        ],
          properties: {
        name: {
          bsonType: "object",
            required: [
              "first",
              "last"
            ],
              properties: {
            first: {
              bsonType: "string",
                description: "must be a string and is required"
            },
            last: {
              bsonType: "string",
                description: "must be a string and is required"
            }
          }
        },
        country: {
          bsonType: "string",
            description: "must be a string and is required"
        },
        birth: {
          bsonType: "date",
            description: "must be a date and is required"
        },
        death: {
          bsonType: "date",
            description: "must be a date and is required"
        }
      }
    }
  }
}
)
```

Now let's try again.

13. Now solve:

> Dodaj kolekcję reviews, która będzie przechowywać recenzje książek:
>
> - referencję do książki (po odpowiednim _id),
> - dane recenzenta,
> - ocenę w skali 1-5,
> - tekst recenzji.
>
> Wstaw dane dotyczące co najmniej trzech recenzji dla jednej książki (jedna z ocen poniżej 3 i jedna z ocen wynosząca 5). Wstaw recenzje kilku książek wystawione przez tego samego autora.
>
> Jaki wpływ na wstawianie i wyszukiwanie danych ma przyjęta przez Ciebie metoda przechowywania informacji o recenzencie?

14. Add `reviews`

```
db.createCollection('reviews', {
    validator: {
        $jsonSchema: {
            bsonType: "object",
            required: [ "book", "reviewer", "rating", "description" ],
            properties: {
                book: {
                    bsonType: "objectId",
                    description: "must be a objectId and is required"
                },
                reviewer: {
                    bsonType: "string",
                    description: "must be a string and is required"
                },
                rating: {
                    bsonType: "int",
                    minimum: 1,
                    maximum: 5,
                    description: "must be an integer in [ 1, 2, 3, 4, 5 ] and is required"
                },
                description: {
                    bsonType: "string",
                    description: "must be a string and is required"
                }
                }
            }
        }
})
```

15. Insert 3 reviews for one book

```
db.reviews.insertMany([
    {
        "book": ObjectId("65712008cb2d05e2c973fe87"),
        "reviewer": "John Doe",
        "rating": 5,
        "description": "Best book ever!"
    },
    {
        "book": ObjectId("65712008cb2d05e2c973fe87"),
        "reviewer": "John Doe",
        "rating": 3,
        "description": "Best Mediocrecy!"
    },
    {
        "book": ObjectId("65712008cb2d05e2c973fe87"),
        "reviewer": "Jane Doe",
        "rating": 5,
        "description": "Sehr gut!"
    },
    {
        "book": ObjectId("65712008cb2d05e2c973fe88"),
        "reviewer": "Jink Doe",
        "rating": 4,
        "description": "Great book!"
    },
    {
        "book": ObjectId("65712008cb2d05e2c973fe89"),
        "reviewer": "Jack Doe",
        "rating": 3,
        "description": "Decent book."
    }
]);
```

The impact on selecting and inserting data is that we can now use the `book` field to reference the book. This way we can easily find all reviews for a given book.

16. Now solve:

> Dodaj dla każdego dokumentu w kolekcji authors nowe pole: awards – tablicę nagród (np. nazwa nagrody, rok otrzymania), z możliwością pustej tablicy, jeśli autor nie otrzymał nagród.

> Dodaj nowe pole w kolekcji books: genres – tablicę stringów reprezentującą gatunki literackie (np. "Fantasy", "Horror").

Add awards to authors:

```
db.authors.updateMany({}, { $set: { awards: [] } })
```

Add genres to books:

```
db.books.updateMany({}, { $set: { genres: [] } })

db.books.updateOne(
    { "_id": ObjectId("65712008cb2d05e2c973fe87") },
    { $addToSet: { genres: "Ancient" } }
)
db.books.updateOne(
    { "_id": ObjectId("65712008cb2d05e2c973fe88") },
    { $addToSet: { genres: "Classical" } }
)
db.books.updateOne(
    { "_id": ObjectId("65712008cb2d05e2c973fe89") },
    { $addToSet: { genres: "Classical" } }
)
db.books.updateOne(
    { "_id": ObjectId("65712008cb2d05e2c973fe90") },
    { $addToSet: { genres: "Fantasy" } }
)
```

Exercise 3 - Search Queries

Solve:

> Wyszukaj wszystkie książki napisane przez autora o konkretnym imieniu i nazwisku. (1 pkt)

```
# find by id
db.books.find({ "author": ObjectId("65711ccdcb2d05e2c973fe87") })

# find by first fetching id from authors
db.authors.find({ "name.first": "Henryk", "name.last": "Sienkiewicz" })

# (FINAL) joined query
db.books.find({ "author": db.authors.findOne({ "name.first": "Henryk",
"name.last": "Sienkiewicz" })._id })
```

> Wyszukaj wszystkie książki napisane po polsku w gatunku "Fantasy". (1 pkt)

```
db.books.find({ "language": "Polish", "genres": "Fantasy" })
```

> Wyszukaj wszystkie książki, których średnia ocena w recenzjach to co najmniej 4. (1 pkt)

```
db.books.aggregate([
    {
        $lookup: {
            from: "reviews",
            localField: "_id",
            foreignField: "book",
            as: "book_reviews"
        }
    },
    {
        $addFields: {
            average_rating: {
                $avg: "$book_reviews.rating"
            }
        }
    },
    {
        $match: {
            average_rating: { $gte: 4 }
        }
    },
    {
        $project: {
            _id: 0,
            title: 1,
            isbn: 1,
            publication_year: 1,
            language: 1,
            author: 1,
            average_rating: 1
        }
    }
]);
```

> Za pomocą aggregate wyszukaj dane o książkach napisanych przez polskich autorów, wraz z nazwiskami tych autorów i średnią oceną książek. (2 pkt)

```
db.books.aggregate([
    {
        $lookup: {
            from: "authors",
            localField: "author",
            foreignField: "_id",
            as: "author_info"
        }
    },
    {
        $unwind: "$author_info"
    },
    {
        $match: {
            "author_info.country": "Poland"
        }
    },
    {
        $lookup: {
            from: "reviews",
            localField: "_id",
            foreignField: "book",
            as: "reviews"
        }
    },
    {
        $addFields: {
            average_rating: {
                $cond: {
                    if: { $gt: [{ $size: "$reviews" }, 0] },
                    then: { $avg: "$reviews.rating" },
                    else: null
                }
            }
        }
    },
    {
        $project: {
            _id: 0,
            title: 1,
            isbn: 1,
            publication_year: 1,
            language: 1,
            author: {
                first: "$author_info.name.first",
                last: "$author_info.name.last"
            },
            average_rating: 1
        }
    }
])
```