Politechnika Białostocka

Sprawozdanie z pracowni specjalistycznej

Zaawansowane bazy danych i hurtownie danych

Zadanie numer: **5**

Temat: BAZY DANYCH TYPU NOSQL, MONGODB

Wykonujący ćwiczenie: Mateusz Czarniecki

Studia dzienne

Kierunek: Informatyka, stopień II

Semestr: I Grupa zajęciowa: PS**1**

Prowadzący ćwiczenie: prof. dr hab. inż. Agnieszka Drużdżel

Zadania zostały wykonane w języku C#, w oparciu o bibliotekę MongoDB.Driver.

Zadanie 1

Korzystając z mongoimport załaduj dane o filmach:

(a) z pliku title.basics.tsv.gz (https://datasets.imdbws.com/) do bazy IMDB oraz kolekcji Title
(b) z pliku title.principals.tsv.gz (https://datasets.imdbws.com/) do do bazy IMDB oraz kolekcji Cast
(c) z pliku title.crew.tsv.gz (https://datasets.imdbws.com/) do do bazy IMDB oraz kolekcji Crew
(d) z pliku title.ratings.tsv.gz (https://datasets.imdbws.com/) do do bazy IMDB oraz kolekcji Rating
(e) z pliku name.basics.tsv.gz (https://datasets.imdbws.com/) do do bazy IMDB oraz kolekcji Name

Po zaimportowaniu plików pobranych z bazy filmów IMDB, łącze się z bazą i pobieram 5 kolekcji: Title, Cast, Crew, Ratings, Name.

```
static MongoClient dbClient;
static IMongoDatabase db;
static IMongoCollection<BsonDocument> titleCollection;
static IMongoCollection<BsonDocument> castCollection;
static IMongoCollection<BsonDocument> crewCollection;
static IMongoCollection<BsonDocument> ratingCollection;
static IMongoCollection<BsonDocument> nameCollection;
```

```
dbClient = new MongoClient("mongodb://localhost:27017");
db = dbClient.GetDatabase("imdb");

titleCollection = db.GetCollection<BsonDocument>("Title");
castCollection = db.GetCollection<BsonDocument>("Cast");
crewCollection = db.GetCollection<BsonDocument>("Crew");
ratingsCollection = db.GetCollection<BsonDocument>("Ratings");
nameCollection = db.GetCollection<BsonDocument>("Name");
```

Zadanie 2

Sprawdź liczbę dokumentów w kolekcjach Title/Cast/Crew/Rating/Name. Wyświetl pierwszy dokument każdej kolekcji.

W tym zadaniu do obliczenia liczby dokumentów użyłem metody *CountDocumets* na każdej kolekcji. Metoda ta przyjmuje filtr, lecz zgodnie z treścią zadania podałem tam pustą klasę *BsonDocument*, by dostać liczbę wszystkich dokumentów z kolekcji.

By pobrać pierwszy dokument z każdej kolekcji użyłem metody *Find* do zwrócenia wszystkich dokumentów, a następnie metody *FirstOrDefault*, która zwraca pierwszy dokument lub w przypadku, gdy w kolekcji nie znajduje się żaden dokument zwracana jest wartość null. By dokumenty były wypisane w przejrzysty sposób podałem obiekt *JsonWirtterSettings* do metody *ToJson* z właściwością *Indent* na *true* (podobnie jak funkcja *pretty()* w mongo).

Kod programu:

```
var titleCount = titleCollection.CountDocuments(new BsonDocument());
var castCount = castCollection.CountDocuments(new BsonDocument());
var crewCount = crewCollection.CountDocuments(new BsonDocument());
var ratingCount = ratingsCollection.CountDocuments(new BsonDocument());
var nameCount = nameCollection.CountDocuments(new BsonDocument());

var firstTitle = titleCollection.Find(new BsonDocument()).FirstOrDefault()
    .ToJson(new JsonWriterSettings() { Indent = true});
var firstCast = castCollection.Find(new BsonDocument()).FirstOrDefault()
    .ToJson(new JsonWriterSettings() { Indent = true });
var firstCrew = crewCollection.Find(new BsonDocument()).FirstOrDefault()
    .ToJson(new JsonWriterSettings() { Indent = true });
var firstRating = ratingsCollection.Find(new BsonDocument()).FirstOrDefault()
    .ToJson(new JsonWriterSettings() { Indent = true });
var firstName = nameCollection.Find(new BsonDocument()).FirstOrDefault()
    .ToJson(new JsonWriterSettings() { Indent = true });

Console.WriteLine("Ilość dokumentów Title:  " + titleCount);
Console.WriteLine("Ilość dokumentów Cast:   " + castCount);
Console.WriteLine("Ilość dokumentów Crew:   " + crewCount);
Console.WriteLine("Ilość dokumentów Rating: " + ratingCount);
Console.WriteLine("Ilość dokumentów Name:   " + nameCount);
Console.WriteLine("\n----------------------------------------\n");
Console.WriteLine("Pierwszy dokument Title:\n" + firstTitle + "\n");
Console.WriteLine("Pierwszy dokument Cast:\n" + firstCast + "\n");
Console.WriteLine("Pierwszy dokument Crew:\n" + firstCrew + "\n");
Console.WriteLine("Pierwszy dokument Rating:\n" + firstRating + "\n");
Console.WriteLine("Pierwszy dokument Name:\n" + firstCrew + "\n");
```

Wynik:

```
Ilość dokumentów Title:  6823665
Ilość dokumentów Cast:   39361835
Ilość dokumentów Crew:   6823665
Ilość dokumentów Rating: 1042583
Ilość dokumentów Name:   10107370

----------------------------------------

Pierwszy dokument Title:
{
  "_id" : ObjectId("5ec7ecaa8df0e088df183dca"),
  "tconst" : "tt0000001",
  "titleType" : "short",
  "primaryTitle" : "Carmencita",
  "originalTitle" : "Carmencita",
  "isAdult" : 0,
  "startYear" : 1894,
  "endYear" : "\\N",
  "runtimeMinutes" : 1,
  "genres" : "Documentary,Short"
}

Pierwszy dokument Cast:
{
  "_id" : ObjectId("5ec7f008ee6a6a4a306baf0f"),
  "tconst" : "tt0000001",
  "ordering" : 2,
  "nconst" : "nm0005690",
  "category" : "director",
  "job" : "\\N",
  "characters" : "\\N"
}

Pierwszy dokument Crew:
{
  "_id" : ObjectId("5ec7ef3de9e6ad455a28c7eb"),
  "tconst" : "tt0000001",
  "directors" : "nm0005690",
  "writers" : "\\N"
}

Pierwszy dokument Rating:
{
  "_id" : ObjectId("5ec7f29772547865c5143013"),
  "tconst" : "tt0000001",
  "averageRating" : 5.5999999999999996,
  "numVotes" : 1614
}

Pierwszy dokument Name:
{
  "_id" : ObjectId("5ec7ef3de9e6ad455a28c7eb"),
  "tconst" : "tt0000001",
  "directors" : "nm0005690",
  "writers" : "\\N"
}
```

Zadanie 3

Wybierz 5 pierwszych dokumentów z kolekcji Title, które były wyprodukowane w roku 2005, są z kategorii filmów Romance, ich czas trwania jest większy niż 100 minut, ale nie przekracza 120 minut. Zwracane dokumenty powinny zawierać tytuł, rok produkcji, kategorię oraz czas trwania. Dane uporządkuj rosnąco wg tytułu filmu. Sprawdź również, ile dokumentów zwróciłoby zapytanie po wyłączeniu ograniczenia w postaci 5 pierwszych dokumentów.

W zadaniu filtr i projekcję zbudowałem za pomocą obiektu Builders. Przy wykonywaniu zapytania do metody *Find* przekazałem zbudowany filtr zapytania, a do metody *Project* zbudowaną projekcję.

```
var builderFilter = Builders<BsonDocument>.Filter;
var builderProjection = Builders<BsonDocument>.Projection;

var filter = builderFilter.Eq(x => x["startYear"], 2005) &
    builderFilter.Regex(x => x["genres"], new BsonRegularExpression(".*Romance.*")) &
    builderFilter.Gt(x => x["runtimeMinutes"], 100) &
    builderFilter.Lte(x => x["runtimeMinutes"], 120);

var projection = builderProjection.Include(x => x["primaryTitle"])
    .Include(x => x["startYear"]).Include(x => x["genres"])
    .Include(x => x["runtimeMinutes"]).Exclude(x => x["_id"]);

var resultWithoutLimit = titleCollection.Find(filter).Project(projection)
    .SortBy(x => x["primaryTitle"]);
var resultCount = resultWithoutLimit.CountDocuments();
var resultWithLimit = resultWithoutLimit.Limit(5).ToList()
    .ToJson(new JsonWriterSettings() { Indent = true });

Console.WriteLine("Ilość dokumnetów: " + resultCount);
Console.WriteLine(resultWithLimit);
```

Wynik:


```
Ilość dokumnetów: 166
[{
    "primaryTitle" : "1735 Km",
    "startYear" : 2005,
    "runtimeMinutes" : 107,
    "genres" : "Comedy,Drama,Romance"
  }, {
    "primaryTitle" : "2 Young",
    "startYear" : 2005,
    "runtimeMinutes" : 107,
    "genres" : "Drama,Romance"
  }, {
    "primaryTitle" : "3° kälter",
    "startYear" : 2005,
    "runtimeMinutes" : 104,
    "genres" : "Drama,Romance"
  }, {
    "primaryTitle" : "A Boy Who Went to Heaven",
    "startYear" : 2005,
    "runtimeMinutes" : 110,
    "genres" : "Romance"
  }, {
    "primaryTitle" : "A Few People, a Little Time",
    "startYear" : 2005,
    "runtimeMinutes" : 104,
    "genres" : "Biography,Drama,Romance"
  }]
```

Zadanie 4

Wybierz filmy z kolekcji Title z roku 1930, które są z kategorii filmów Comedy. Zwracane dokumenty powinny zawierać oryginalny tytuł filmu, czas trwania filmu oraz kategorię. Dane uporządkuj malejąco wg czasu trwania filmu.

```
var builderFilter = Builders<BsonDocument>.Filter;
var builderProjection = Builders<BsonDocument>.Projection;

var filter = builderFilter.Eq(x => x["startYear"], 1930) &
    builderFilter.Regex(x => x["genres"], new BsonRegularExpression(".*Comedy."));

var projection = builderProjection.Include(x => x["originalTitle"])
    .Include(x => x["runtimeMinutes"]).Include(x => x["genres"])
    .Exclude(x => x["_id"]);

var result = titleCollection.Find(filter).Project(projection)
    .SortBy(x => x["runtimeMinutes"]);

var resultCount = result.CountDocuments();
var resultWithLimit = result.Limit(5).ToList()
    .ToJson(new JsonWriterSettings() { Indent = true });

Console.WriteLine("Ilość dokumentów: " + resultCount);
Console.WriteLine(resultWithLimit);
```

Wynik:

```
Ilość dokumentów: 661
[{
    "originalTitle" : "Minnie's Yoo Hoo",
    "runtimeMinutes" : 5,
    "genres" : "Animation,Comedy,Family"
  }, {
    "originalTitle" : "The Shaming of the True",
    "runtimeMinutes" : 5,
    "genres" : "Comedy,Short"
  }, {
    "originalTitle" : "Autumn",
    "runtimeMinutes" : 6,
    "genres" : "Animation,Comedy,Family"
  }, {
    "originalTitle" : "The Barnyard Concert",
    "runtimeMinutes" : 6,
    "genres" : "Animation,Comedy,Family"
  }, {
    "originalTitle" : "The Booze Hangs High",
    "runtimeMinutes" : 6,
    "genres" : "Animation,Comedy,Family"
  }]
```

Zadanie 5

Korzystając z kolekcji Cast (lub Crew), Title oraz Name, sprawdź kto był reżyserem filmu Casablanca z 1942 roku. Podaj imię i nazwisko reżysera oraz jego datę urodzenia. Jeśli to możliwe, skorzystaj tylko z jednego polecenia.

Funkcja Aggregate sprawia, że możemy wywoływać funkcje agregujące takie jak Match, Lookup, Unwind.

```
var builderFilter = Builders<BsonDocument>.Filter;
var builderProjection = Builders<BsonDocument>.Projection;

var filterTitle = builderFilter.Eq(x => x["primaryTitle"], "Casablanca") &
    builderFilter.Eq(x => x["startYear"], 1942);

var filterCast = builderFilter.Regex(x => x["cast.category"],
    new BsonRegularExpression(".*director.*"));

var projection = builderProjection.Include(x => x["primaryName"])
    .Include(x => x["birthDate"]);

var result = titleCollection.Aggregate().Match(filterTitle)
    .Lookup("Cast", "tconst", "tconst", "cast").Unwind("cast")
    .Lookup("Name", "cast.nconst", "nconst", "name").Unwind("name")
    .Match(filterCast).Project(x => new
    {
        primaryName = x["name.primaryName"],
        birthYear = x["name.birthYear"]
    }).FirstOrDefault().ToJson(new JsonWriterSettings() { Indent = true });

Console.WriteLine(result);
```

Wynik:

```
{
  "primaryName" : "Michael Curtiz",
  "birthYear" : 1886
}
```

Zadanie 6

Sprawdź ile filmów różnego typu (pole titleType) było wprodukowanych w latach 2007-2009. Wynik zapytania powinien zwracać nazwę typu oraz liczbę filmów.

```
var builderFilter = Builders<BsonDocument>.Filter;

var filter = builderFilter.Gte(x => x["startYear"], 2007) &
    builderFilter.Lte(x => x["startYear"], 2009);

var result = titleCollection.Aggregate().Match(filter)
    .Group(x => x["titleType"], y => new
    {
        titleType = y.Key,
        howMany = y.Count()
    }).ToList().ToJson(new JsonWriterSettings() { Indent = true });

Console.WriteLine(result);
```

Wynik:


```
[{
    "titleType" : "tvMovie",
    "howMany" : 9668
  }, {
    "titleType" : "short",
    "howMany" : 53235
  }, {
    "titleType" : "tvSpecial",
    "howMany" : 2845
  }, {
    "titleType" : "tvShort",
    "howMany" : 901
  }, {
    "titleType" : "tvSeries",
    "howMany" : 15618
  }, {
    "titleType" : "videoGame",
    "howMany" : 1986
  }, {
    "titleType" : "tvMiniSeries",
    "howMany" : 1373
  }, {
    "titleType" : "movie",
    "howMany" : 29207
  }, {
    "titleType" : "video",
    "howMany" : 35184
  }, {
    "titleType" : "tvEpisode",
    "howMany" : 407483
  }]
```

Zadanie 7

W oparciu o kolekcje Title oraz Rating sprawdź średnią ocenę filmów dokumentalnych wprodukowanych w latach 1994-1996. Wyświetl tytuł filmu, rok produkcji oraz jego średnią ocenę. Dane uporządkuj malejąco wg średniej oceny.

a) **sprawdź, ile takich dokumentów zwróci zapytanie**
b) **wyświetl tylko 10 pierwszych dokumentów spełniających powyższe warunki.**

By zapytanie nie wykonywało się „zbyt długo" i zwróciło wynik dodałem indeksy w kolekcjach Title i Ratings po polu tconst.

```
var builderFilter = Builders<BsonDocument>.Filter;
var builderProjection = Builders<BsonDocument>.Projection;

var filter = builderFilter.Gte(x => x["startYear"], 1994) &
    builderFilter.Lte(x => x["startYear"], 1996);

titleCollection.Indexes.CreateOne(new CreateIndexModel<BsonDocument>("{tconst: 1}"));
ratingsCollection.Indexes.CreateOne(new CreateIndexModel<BsonDocument>("{tconst: 1}"));

var result = titleCollection.Aggregate().Match(filter)
    .Lookup("Ratings", "tconst", "tconst", "ratings")
    .SortByDescending(x => x["ratings.averageRating"])
    .Limit(10).Project(x => new
    {
        primaryTitle = x["primaryTitle"],
        startYear = x["startYear"],
        averageRating = x["ratings.averageRating"],
    }).ToList().ToJson(new JsonWriterSettings() { Indent = true });

Console.WriteLine(result);
```

Wynik:


```
[{
    "primaryTitle" : "Spiral Tribe",
    "startYear" : 1994,
    "averageRating" : [10.0]
  }, {
    "primaryTitle" : "So You Wanna Be a Porn Star 1: The Russians Are Cumming",
    "startYear" : 1994,
    "averageRating" : [10.0]
  }, {
    "primaryTitle" : "He's Heavy, He's My Brother",
    "startYear" : 1994,
    "averageRating" : [10.0]
  }, {
    "primaryTitle" : "Girls Loving Girls",
    "startYear" : 1996,
    "averageRating" : [10.0]
  }, {
    "primaryTitle" : "A Series of Arena's Special Purpose Films",
    "startYear" : 1994,
    "averageRating" : [10.0]
  }, {
    "primaryTitle" : "Tongue",
    "startYear" : 1995,
    "averageRating" : [10.0]
  }, {
    "primaryTitle" : "Far East",
    "startYear" : 1994,
    "averageRating" : [10.0]
  }, {
    "primaryTitle" : "Johnston... Johnston",
    "startYear" : 1995,
    "averageRating" : [10.0]
  }, {
    "primaryTitle" : "Brujaci",
    "startYear" : 1994,
    "averageRating" : [10.0]
  }, {
    "primaryTitle" : "Renegades 2",
    "startYear" : 1995,
    "averageRating" : [10.0]
  }]
```

Zadanie 8

Dla każdego filmu (kolekcja Title), który ma najwyższą średnią ocenę (10.0), dodaj pole max z wartością równą 1. W poleceniu skorzystaj z kolekcji Rating, która zawiera informacje o średniej ocenie filmu.

```
var builderFilter = Builders<BsonDocument>.Filter;
var builderProjection = Builders<BsonDocument>.Projection;

var filter = builderFilter.Eq(x => x["averageRating"], 10.0);

var projection = builderProjection.Include(x => x["tconst"])
    .Exclude(x => x["_id"]);

var titleTconstWithMaxAvgRating = ratingsCollection.Find(filter)
    .Project(projection).ToList();

var updateFilter = builderFilter.In(x => x["tconst"],
    titleTconstWithMaxAvgRating.Select(x => x["tconst"]));


var modifiedCount = titleCollection.UpdateMany(updateFilter, "{$set: {max: 1}}")
    .ModifiedCount;

Console.WriteLine("Zmodyfikowano: " + modifiedCount);
```

Wynik (w wyniku została pokazana tylko ilość zmodyfikowanych dokumentów z kolekcji Title):

```
Zmodyfikowano: 3407
```

Zadanie 9

Stwórz indeks tekstowy dla pola primaryProfession w kolekcji Name. Następnie używając tego indeksu znajdź 5 pierwszych osób urodzonych w latach 1950-1980, które są aktorami lub reżyserami. Sprawdź również liczbę zwracanych dokumentów po wyłączeniu ograniczenia w postaci 10 pierwszych dokumentów. Zapytanie powinno zwrócić imię i nazwisko osoby, datę urodzenia oraz profesję.

```
var builderFilter = Builders<BsonDocument>.Filter;
var builderProjection = Builders<BsonDocument>.Projection;

var filter = builderFilter.Gte(x => x["birthYear"], 1950) &
    builderFilter.Lte(x => x["birthYear"], 1980) &
    builderFilter.Regex(x => x["primaryProfession"],
    new BsonRegularExpression("(.*actor.*|.*actress.*|.*director.*)"));

var projection = builderProjection.Include(x => x["primaryName"])
    .Include(x => x["birthYear"]).Include(x => x["primaryProfession"])
    .Exclude(x => x["_id"]);

nameCollection.Indexes.CreateOne(new CreateIndexModel<BsonDocument>
    ("{primaryProfession: \"text\"}"));

var result = nameCollection.Find(filter);

var resultCount = result.CountDocuments();
var resultWithLimit = result.Limit(5).Project(projection).ToList()
    .ToJson(new JsonWriterSettings() { Indent = true });

Console.WriteLine("Ilość dokumentów: " + resultCount);
Console.WriteLine(resultWithLimit);
```

Wynik:


```
Ilość dokumentów: 136330
[{
    "primaryName" : "Margaux Hemingway",
    "birthYear" : 1954,
    "primaryProfession" : "actress,miscellaneous"
  }, {
    "primaryName" : "Li Gong",
    "birthYear" : 1965,
    "primaryProfession" : "actress"
  }, {
    "primaryName" : "Elena Koreneva",
    "birthYear" : 1953,
    "primaryProfession" : "actress,casting_director,soundtrack"
  }, {
    "primaryName" : "Brad Pitt",
    "birthYear" : 1963,
    "primaryProfession" : "actor,producer,soundtrack"
  }, {
    "primaryName" : "Gillian Anderson",
    "birthYear" : 1968,
    "primaryProfession" : "actress,producer,soundtrack"
  }]
```

Zadanie 10

Utwórz indeks tekstowy dla pola primaryName w kolekcji Name. Następnie używając tego indeksu znajdź dokumenty opisujące osoby o nazwisku Fonda oraz Coppola. Przy wyszukiwaniu włącz opcję, która będzie uwzględniać wielkie/małe litery.

a) **Ile dokumentów zwraca zapytanie?**
b) **Wyświetl 5 pierwszych dokumentów (dokument powinien powinien zwracać dwa pola: primaryName, primaryProfession).**
c) **Sprawdź, ile indeksów posiada kolekcja Name?**

```
var builderFilter = Builders<BsonDocument>.Filter;
var builderProjection = Builders<BsonDocument>.Projection;

var filter = builderFilter.Regex(x => x["primaryName"],
    new BsonRegularExpression("(.*Fonda.*|.*Coppola.*)"));

var projection = builderProjection.Include(x => x["primaryName"])
    .Include(x => x["primaryProfession"]).Exclude(x => x["_id"]);

nameCollection.Indexes.CreateOne(new CreateIndexModel<BsonDocument>
    ("{primaryName: \"text\"}"));

var result = nameCollection.Find(filter);
var resultCount = result.CountDocuments();
var resultWithLimit = result.Limit(5).Project(projection).ToList()
    .ToJson(new JsonWriterSettings() { Indent = true });
var indexesCount = nameCollection.Indexes.List().ToList().Count;

Console.WriteLine("Ilość dokumentów: " + resultCount);
Console.WriteLine("Ilość indeksów: " + indexesCount);
Console.WriteLine(resultWithLimit);
```

Wynik:


Ilość dokumentów: 605
Ilość indeksów: 2
[{
"primaryName" : "Henry Fonda",
"primaryProfession" : "actor,producer,soundtrack"
}, {
"primaryName" : "Francis Ford Coppola",
"primaryProfession" : "producer,director,writer"
}, {
"primaryName" : "Bridget Fonda",
"primaryProfession" : "actress,soundtrack"
}, {
"primaryName" : "Jane Fonda",
"primaryProfession" : "actress,producer,soundtrack"
}, {
"primaryName" : "Sofia Coppola",
"primaryProfession" : "actress,director,writer"
}]

Wnioski

Za pomocą biblioteki MongoDb.Driver dla frameworka .NET w prosty sposób mamy możliwość połączenia się z bazą MongoDB. Biblioteka oferuje wiele funkcji, dzięki którym możemy wyszukiwać dokumenty w danej kolekcji, modyfikować kolekcje, łączyć kolekcję itd. Poprzez używanie indeksów będziemy mogli szybciej wyszukiwać elementy. Przydaje się to w takich dużych bazach jak ta z filmami.