

Wydział Informatyki Politechniki Białostockiej Zaawansowane bazy danych i hurtownie danych Pracownia specjalistyczna	Data realizacji: 24.05.2020r.
Zadanie nr 5 Temat: MongoDB Grupa PS 3 Imię i nazwisko 1. Adam Bajguz 2. Michał Kierzkowski	Prowadzący: dr hab. inż. Agnieszka Drużdżel, prof. PB Ocena:

1 REALIZACJA ZADANIA 1

Do realizacji zadania nr 1 wykorzystano następujące polecenie „mongoimport --type tsv --db IMDB --collection nazwa_kolekcji --headerline --file ścieżka_do_pliku”, które pozwoliło na załadowanie wszystkich wymaganych plików jako kolekcje do bazy MongoDB. Poprawność załadowania danych zaprezentowana jest na Rys. 1.

```

Authors:
- Adam Bajguz
- Michał Kierzkowski
=====

---[HomeworkTask01]-----

{
  "CollectionNames": [
    "Rating",
    "Crew",
    "Cast",
    "Name",
    "Title"
  ]
}

public async Task<object> RunAsync()
{
    IAsyncCursor<string> collections = await _databaseContext.Db.ListCollectionNamesAsync();

    Result result = new Result
    {
        CollectionNames = collections.ToEnumerable()
    };
    return result;
}

public sealed class Result
{
    public IEnumerable<string>? CollectionNames { get; set; }
}

```

Rys. 1. Prezentacja poprawności załadowania danych wraz z kodem odpowiedzialnym za wypisanie kolekcji

2 REALIZACJA ZADANIA 1

W celu realizacji zadania nr 2 utworzono kod przedstawiony na Rys. 2.1. Wynik działania tego kodu zaprezentowano na Rys. 2.2.

```
11 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
public async Task<object> RunAsync()
{
    IAsyncCursor<string> collections = await _dbContext.Db.ListCollectionNamesAsync();

    List<CollectionInfo> collectionSizes = new List<CollectionInfo>();
    await foreach (var x in collections)
    {
        struct IMDB.Application.HomeworkTasks.HomeworkTask02.CollectionInfo
        {
            IMongoCollection<object> collection = _dbContext.Db.GetCollection<object>(x);
            IMongoQueryable<object> mongoQueryable = collection.AsQueryable();

            int size = await mongoQueryable.CountAsync();
            object firstObject = await mongoQueryable.FirstOrDefaultAsync();

            collectionSizes.Add((x, size, firstObject));
        });

    Result result = new Result
    {
        CollectionSizes = collectionSizes
    };
    return result;
}

2 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
public sealed class Result
{
    1 reference | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public IEnumerable<CollectionInfo>? CollectionSizes { get; set; }
}

8 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
public struct CollectionInfo
{
    6 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public string CollectionName { get; }
    6 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public int Size { get; }
    6 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public object Entity { get; }

    1 reference | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public CollectionInfo(string collectionName, int size, object entity)
    {
        CollectionName = collectionName;
        Size = size;
        Entity = entity;
    }

    0 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public override bool Equals(object? obj)
    {
        return obj is CollectionInfo other &&
            CollectionName == other.CollectionName &&
            Size == other.Size && Entity == other.Entity;
    }

    0 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public override int GetHashCode()
    {
        return HashCode.Combine(CollectionName, Size, Entity);
    }

    0 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public void Deconstruct(out string collectionName, out int size, out object entity)
    {
        collectionName = CollectionName;
        size = Size;
        entity = Entity;
    }

    public static implicit operator (string CollectionName, int Size, object Entity)(CollectionInfo value)
    {
        return (value.CollectionName, value.Size, value.Entity);
    }

    public static implicit operator CollectionInfo((string CollectionName, int Size, object Entity) value)
    {
        return new CollectionInfo(value.CollectionName, value.Size, value.Entity);
    }
}
```

Rys. 2.1. Fragment najistotniejszego kodu realizującego zadanie nr 2

```

-----[HomeworkTask02]-----
{
  "CollectionSizes": [
    {
      "CollectionName": "Rating",
      "Size": 1041611,
      "Entity": {
        "_id": "5ec56740e74eb62ee9cb6d43",
        "tconst": "tt0000002",
        "averageRating": 6.0,
        "numVotes": 197
      }
    },
    {
      "CollectionName": "Crew",
      "Size": 6814741,
      "Entity": {
        "_id": "5ec567288498e8287e7e3810",
        "tconst": "tt0000008",
        "directors": "nm0005690",
        "writers": ""
      }
    },
    {
      "CollectionName": "Cast",
      "Size": 39311661,
      "Entity": {
        "_id": "5ec5683f7d25a63dec54593c",
        "tconst": "tt0000001",
        "ordering": 1,
        "nconst": "nm1588970",
        "category": "self",
        "job": "",
        "characters": "[\\\"Self\\\"]"
      }
    },
    {
      "CollectionName": "Name",
      "Size": 10100399,
      "Entity": {
        "_id": "5ec56778b0fb0498743a03ad",
        "nconst": "nm0000001",
        "primaryName": "Fred Astaire",
        "birthYear": 1899,
        "deathYear": 1987,
        "primaryProfession": "soundtrack,actor,miscellaneous",
        "knownForTitles": "tt0072308,tt0053137,tt0050419,tt0043044"
      }
    },
    {
      "CollectionName": "Title",
      "Size": 10894425,
      "Entity": {
        "_id": "5ec568396273c2189603f451",
        "tconst": "tt0000001",
        "titleType": "short",
        "primaryTitle": "Carmencita",
        "originalTitle": "Carmencita",
        "isAdult": 0,
        "startYear": 1894,
        "endYear": 0.0,
        "runtimeMinutes": 1,
        "genres": "Documentary,Short"
      }
    }
  ]
}

```

Rys. 2.2. Wynik działania kodu z Rys. 2.1

3 REALIZACJA ZADANIA 3

W celu realizacji zadania nr 3 utworzono kod przedstawiony na Rys. 3.1. Wynik działania tego kodu zaprezentowano na Rys. 3.2. Podczas realizacji zadania, ze względu na występowanie w bazie originalTitle i primaryTitle wybrano originalTitle jako tytuł po którym sortowano wyniki.

```
public async Task<object> RunAsync()
{
    IMongoQueryable<Title> queryable = _dbContext.Titles.AsQueryable();

    IOrderedMongoQueryable<Title> partialQuery = queryable.Where(x => x.StartYear == 2005 &&
        x.Genres != null && x.Genres.ToLower().Contains("romance") &&
        x.RuntimeMinutes > 100 && x.RuntimeMinutes < 120)
        .OrderBy(x => x.OriginalTitle);

    var list = await partialQuery.Take(5)
        .Select((x) => new { x.OriginalTitle, x.PrimaryTitle, x.StartYear, x.Genres, x.RuntimeMinutes })
        .ToListAsync();

    int numberOfMatchingElements = await partialQuery.CountAsync();

    Result result = new Result
    {
        Titles = list,
        NumberOfMatchingElements = numberOfMatchingElements
    };
    return result;
}

2 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
public sealed class Result
{
    1 reference | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public object? Titles { get; set; }
    1 reference | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public int NumberOfMatchingElements { get; set; }
}
```

Rys. 3.1. Fragment najistotniejszego kodu realizującego zadanie nr 3

```
-----[HomeworkTask03]-----
{
  "Titles": [
    {
      "OriginalTitle": "1735 Km",
      "PrimaryTitle": "1735 Km",
      "StartYear": 2005,
      "Genres": "Comedy,Drama,Romance",
      "RuntimeMinutes": 107
    },
    {
      "OriginalTitle": "1735 Km",
      "PrimaryTitle": "1735 Km",
      "StartYear": 2005,
      "Genres": "Comedy,Drama,Romance",
      "RuntimeMinutes": 107
    },
    {
      "OriginalTitle": "3° kälter",
      "PrimaryTitle": "3° kälter",
      "StartYear": 2005,
      "Genres": "Drama,Romance",
      "RuntimeMinutes": 104
    },
    {
      "OriginalTitle": "8-gatsu no Kurisumasu",
      "PrimaryTitle": "Christmas in August",
      "StartYear": 2005,
      "Genres": "Drama,Romance",
      "RuntimeMinutes": 102
    },
    {
      "OriginalTitle": "A Lot Like Love",
      "PrimaryTitle": "A Lot Like Love",
      "StartYear": 2005,
      "Genres": "Comedy,Drama,Romance",
      "RuntimeMinutes": 107
    }
  ],
  "NumberOfMatchingElements": 262
}
```

Rys. 3.2. Wynik działania kodu z Rys. 3.1

4 REALIZACJA ZADANIA 4

W celu realizacji zadania nr 4 utworzono kod przedstawiony na Rys. 4.1. Wynik działania tego kodu zaprezentowano na Rys. 4.2.

```
namespace IMDB.Application.HomeworkTasks
{
    using System.Threading.Tasks;
    using IMDB.Application.Interfaces;
    using IMDB.Domain.Entities;
    using MongoDB.Driver;
    using MongoDB.Driver.Linq;

    // [HomeworkExclude]
    1 reference | Adam Bajguz, 11 minutes ago | 1 author, 1 change
    public sealed class HomeworkTask04 : IHomeworkTask
    {
        private readonly IDatabaseContext _databaseContext;

        0 references | Adam Bajguz, 11 minutes ago | 1 author, 1 change
        public HomeworkTask04(IDatabaseContext databaseContext)
        {
            _databaseContext = databaseContext;
        }

        11 references | Adam Bajguz, 11 minutes ago | 1 author, 1 change
        public async Task<object> RunAsync()
        {
            IMongoQueryable<Title> queryable = _databaseContext.Titles.AsQueryable();

            var list = await queryable.Where(x => x.StartYear == 1930 &&
                                                x.Genres != null && x.Genres.ToLower().Contains("comedy"))
                                    .OrderByDescending(x => x.RuntimeMinutes)
                                    .Select((x) => new { x.OriginalTitle, x.RuntimeMinutes, x.Genres })
                                    .ToListAsync();

            Result result = new Result
            {
                Titles = list,
            };
            return result;
        }

        2 references | Adam Bajguz, 11 minutes ago | 1 author, 1 change
        public sealed class Result
        {
            1 reference | Adam Bajguz, 11 minutes ago | 1 author, 1 change
            public object? Titles { get; set; }
        }
    }
}
```

Rys. 4.1. Fragment najistotniejszego kodu realizującego zadanie nr 4

```

----[HomeworkTask04]-----
{
  "Titles": [
    {
      "OriginalTitle": "Ojôsan",
      "RuntimeMinutes": 135,
      "Genres": "Comedy"
    },
    {
      "OriginalTitle": "O Babao",
      "RuntimeMinutes": 120,
      "Genres": "Comedy,Musical"
    },
    {
      "OriginalTitle": "O Babao",
      "RuntimeMinutes": 120,
      "Genres": "Comedy,Musical"
    },
    {
      "OriginalTitle": "Madam Satan",
      "RuntimeMinutes": 116,
      "Genres": "Comedy,Musical,Romance"
    },
    {
      "OriginalTitle": "Just Imagine",
      "RuntimeMinutes": 113,
      "Genres": "Comedy,Fantasy,Musical"
    },
    {
      "OriginalTitle": "Liebling der Götter",
      "RuntimeMinutes": 112,
      "Genres": "Comedy"
    },
    {
      "OriginalTitle": "Liebling der Götter",
      "RuntimeMinutes": 112,
      "Genres": "Comedy"
    }
  ]
}

```

Rys. 4.2. Fragment wyników działania kodu z Rys. 4.1

```

2020-05-23 12:24:41.008 +02:00 [INF] <IMDB 19068:4> (~) Command Started: aggregate, Command {
  .."aggregate" : "Title",
  .."pipeline" : [{
  ..  .."$match" : {
  ..  ..  .."$and" : [{
  ..  ..  ..  .."startYear" : 1930
  ..  ..  ..  }, {
  ..  ..  ..  .."genres" : {
  ..  ..  ..  ..  .."$ne" : null
  ..  ..  ..  ..}
  ..  ..  ..  }, {
  ..  ..  ..  .."genres" : /comedy/is
  ..  ..  ..  }]
  ..  ..}
  ..  }, {
  ..  .."$sort" : {
  ..  ..  .."runtimeMinutes" : -1
  ..  ..}
  ..  }, {
  ..  .."$project" : {
  ..  ..  .."OriginalTitle" : "$originalTitle",
  ..  ..  .."RuntimeMinutes" : "$runtimeMinutes",
  ..  ..  .."Genres" : "$genres",
  ..  ..  .."_id" : 0
  ..  ..}
  ..  }],
  .."cursor" : { },
  .."$db" : "IMDB",
  .."lsid" : {
  ..  .."id" : CSUUID("c23d408c-e5b9-4080-b735-d396e81cf8c6")
  ..}
}

```

Rys. 4.2. Wpis z logów pokazujący rzeczywiste zapytanie zbudowane na bazie Expression Tree z LINQ

Należy zauważyć że użycie ToLower() lub ToUpper() tak naprawdę powoduje zastosowanie porównywania case insensitive, co pokazuje wpis z logów na Rys. 4.1. Parametr **i** po wyrażeniu regularnym np. /comedy/ oznacza że wielkość liter nie ma znaczenie, natomiast **s** umożliwia znakowi kropki (.) dopasowanie wszystkich znaki, w tym znaku nowej linii.

5 REALIZACJA ZADANIA 5

W celu realizacji zadania nr 5 utworzono kod przedstawiony na Rys. 5.1. Wynik działania tego kodu zaprezentowano na Rys. 5.2.

```
1 reference | Adam Bajguz, 6 minutes ago | 1 addition, 3 changes
public sealed class HomeworkTask05 : IHomeworkTask
{
    private readonly IDatabaseContext _databaseContext;

    0 references | Adam Bajguz, 16 minutes ago | 1 author, 1 change
    public HomeworkTask05(IDatabaseContext databaseContext)
    {
        _databaseContext = databaseContext;
    }

    11 references | Adam Bajguz, 16 minutes ago | 1 author, 1 change
    public async Task<object> RunAsync()
    {
        IMongoQueryable<Title> queryable = _databaseContext.Titles.AsQueryable();

        var list = await queryable.Where(x => x.StartYear == 1942 && (x.OriginalTitle == "Casablanca" || x.PrimaryTitle == "Casablanca"))
            .Join(_databaseContext.Crews.AsQueryable(),
                title => title.TConst,
                crew => crew.TConst,
                (title, crew) => new { Title = title, Crew = crew })
            .Join(_databaseContext.Names.AsQueryable(),
                x => x.Crew.Directors,
                names => names.NConst,
                (titleCrew, name) => new { TitleCrew = titleCrew, Name = name })
            .Select((x) => new { x.Name.PrimaryName, x.Name.BirthYear })
            .ToListAsync();

        Result result = new Result
        {
            Data = list,
        };
        return result;
    }

    2 references | Adam Bajguz, 16 minutes ago | 1 author, 1 change
    public sealed class Result
    {
        1 reference | Adam Bajguz, 16 minutes ago | 1 author, 1 change
        public object? Data { get; set; }
    }
}
```

Rys. 5.1. Fragment najistotniejszego kodu realizującego zadanie nr 5

```
Microsoft Visual Studio Debug Console
Authors:
- Adam Bajguz
- Michał Kierzkowski
=====

---[HomeworkTask05]---

{
  "Data": [
    {
      "PrimaryName": "Michael Curtiz",
      "BirthYear": 1886
    }
  ]
}
```

Rys. 5.2. Wynik działania kodu z Rys. 5.1

6 REALIZACJA ZADANIA 6

W celu realizacji zadania nr 6 utworzono kod przedstawiony na Rys. 6.1. Wynik działania tego kodu zaprezentowano na Rys. 6.2. Podczas realizacji zadania, ze względu na występowanie w bazie originalTitle i primaryTitle wybrano originalTile jako tytuł po którym sortowano wyniki.

```
public sealed class HomeworkTask06 : IHomeworkTask
{
    private readonly IDatabaseContext _databaseContext;

    0 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public HomeworkTask06(IDatabaseContext databaseContext)
    {
        _databaseContext = databaseContext;
    }

    11 references | Adam Bajguz, Less than 5 minutes ago | 1 author, 1 change
    public async Task<object> RunAsync()
    {
        IMongoQueryable<Title> queryable = _databaseContext.Titles.AsQueryable();

        var list = await queryable.Where(x => x.StartYear >= 2007 && x.StartYear <= 2009)
            .GroupBy(x => x.TitleType)
            .Select(n => new
            {
                TitleType = n.Key,
                Count = n.Count()
            })
            .ToListAsync();

        Result result = new Result
        {
            TitlesStatistics = list,
        };
        return result;
    }

    2 references | Adam Bajguz, 17 minutes ago | 1 author, 1 change
    public sealed class Result
    {
        1 reference | Adam Bajguz, 17 minutes ago | 1 author, 1 change
        public object? TitlesStatistics { get; set; }
    }
}
```

Rys. 6.1. Fragment najistotniejszego kodu realizującego zadanie nr 6

```

-----[HomeworkTask06]-----
{
  "TitlesStatistics": [
    {
      "TitleType": "tvMiniSeries",
      "Count": 2213
    },
    {
      "TitleType": "short",
      "Count": 87714
    },
    {
      "TitleType": "tvShort",
      "Count": 1497
    },
    {
      "TitleType": "tvMovie",
      "Count": 15818
    },
    {
      "TitleType": "tvSpecial",
      "Count": 4431
    },
    {
      "TitleType": "movie",
      "Count": 47136
    },
    {
      "TitleType": "tvSeries",
      "Count": 25031
    },
    {
      "TitleType": "tvEpisode",
      "Count": 658272
    },
    {
      "TitleType": "videoGame",
      "Count": 2996
    },
    {
      "TitleType": "video",
      "Count": 57505
    }
  ]
}

```

Rys. 6.2. Wynik działania kodu z Rys. 6.1

7 REALIZACJA ZADANIA 7

W celu realizacji zadania nr 7 utworzono kod przedstawiony na Rys. 7.1. Wynik działania tego kodu zaprezentowano na Rys. 7.2. Zapytanie zwraca 13577 dokumentów.

```
1 namespace IMDB.Application.HomeworkTasks
2 {
3     using System.Linq;
4     using System.Threading.Tasks;
5     using IMDB.Application.Interfaces;
6     using IMDB.Domain.Entities;
7     using MongoDB.Driver;
8     using MongoDB.Driver.Linq;
9
10    ///[HomeworkExclude]
11    1 reference | Adam Bajguz, 39 minutes ago | 1 author, 6 changes
12    public sealed class HomeworkTask07 : IHomeworkTask
13    {
14        private readonly IDatabaseContext _databaseContext;
15
16        0 references | Adam Bajguz, 17 hours ago | 1 author, 1 change
17        public HomeworkTask07(IDatabaseContext databaseContext)
18        {
19            _databaseContext = databaseContext;
20        }
21
22        11 references | Adam Bajguz, 39 minutes ago | 1 author, 4 changes
23        public async Task<object> RunAsync()
24        {
25            IMongoQueryable<Title> queryable = _databaseContext.Titles.AsQueryable();
26
27            int numberOfMatchingElements = await queryable.Where(x => x.StartYear >= 1994 && x.StartYear <= 1996 &&
28                x.Genres.ToLower().Contains("document"))
29                .Select(e => e.TConst)
30                .Distinct()
31                .CountAsync();
32
33            var list = await queryable.Where(x => x.StartYear >= 1994 && x.StartYear <= 1996 &&
34                x.Genres.ToLower().Contains("document"))
35                .Take(100)
36                .Select((x) => new { x.TConst, x.PrimaryTitle, x.OriginalTitle, x.StartYear })
37                .Join(_databaseContext.Ratings.AsQueryable(),
38                    title => title.TConst,
39                    rating => rating.TConst,
40                    (title, rating) => new { title.PrimaryTitle, title.OriginalTitle, title.StartYear, rating.AverageRating })
41                .OrderByDescending(x => x.AverageRating)
42                .Take(10)
43                .ToListAsync();
44
45            Result result = new Result
46            {
47                TitlesWithRatings = list,
48                NumberOfMatchingElements = numberOfMatchingElements
49            };
50            return result;
51        }
52
53        2 references | Adam Bajguz, 17 hours ago | 1 author, 1 change
54        public sealed class Result
55        {
56            1 reference | Adam Bajguz, 17 hours ago | 1 author, 1 change
57            public object? TitlesWithRatings { get; set; }
58            1 reference | Adam Bajguz, 17 hours ago | 1 author, 1 change
59            public int NumberOfMatchingElements { get; set; }
60        }
61    }
62 }
```

Rys. 7.1. Fragment najistotniejszego kodu realizującego zadanie nr 7

Ze względu na bardzo długi czas zajmowany prawdopodobnie przez złączenia, które w tej bazie są oparte na wartościach tekstowych. Wynik przedstawiony na Rys. 7.2 jest oparty na 100 filmach – odkomentowania linia nr 32. Możliwym rozwiązaniem tego problemu była by konwersja pól i wartości z tekstowych na liczbowe i dodanie indeksu, jednakże polecenie nie określa czy taka operacja jest dopuszczalna. Zatem podjęto decyzję o zaprezentowaniu działania kodu na wspomnianym podzbiore 100 filmów.

```

-----[HomeworkTask07]-----
{
  "TitlesWithRatings": [
    {
      "PrimaryTitle": "A színész és a halál",
      "OriginalTitle": "A színész és a halál",
      "StartYear": 1995,
      "AverageRating": 9.3
    },
    {
      "PrimaryTitle": "A színész és a halál",
      "OriginalTitle": "A színész és a halál",
      "StartYear": 1995,
      "AverageRating": 9.3
    },
    {
      "PrimaryTitle": "Baseball",
      "OriginalTitle": "Baseball",
      "StartYear": 1994,
      "AverageRating": 9.2
    },
    {
      "PrimaryTitle": "Baseball",
      "OriginalTitle": "Baseball",
      "StartYear": 1994,
      "AverageRating": 9.2
    },
    {
      "PrimaryTitle": "The Lynchburg Story",
      "OriginalTitle": "The Lynchburg Story",
      "StartYear": 1994,
      "AverageRating": 8.8
    },
    {
      "PrimaryTitle": "The Lynchburg Story",
      "OriginalTitle": "The Lynchburg Story",
      "StartYear": 1994,
      "AverageRating": 8.8
    },
    {
      "PrimaryTitle": "Movie Magic",
      "OriginalTitle": "Movie Magic",
      "StartYear": 1994,
      "AverageRating": 8.7
    },
    {
      "PrimaryTitle": "Movie Magic",
      "OriginalTitle": "Movie Magic",
      "StartYear": 1994,
      "AverageRating": 8.7
    },
    {
      "PrimaryTitle": "Time Team",
      "OriginalTitle": "Time Team",
      "StartYear": 1994,
      "AverageRating": 8.5
    },
    {
      "PrimaryTitle": "Time Team",
      "OriginalTitle": "Time Team",
      "StartYear": 1994,
      "AverageRating": 8.5
    }
  ],
  "NumberOfMatchingElements": 13577
}

```

Rys. 7.2. Wynik działania kodu z Rys. 7.1

8 REALIZACJA ZADANIA 8

W celu realizacji zadania nr 8 utworzono kod przedstawiony na Rys. 8.1. Wynik działania tego kodu zaprezentowano na Rys. 8.2.

```
[[HomeworkExclude]
1 reference | Adam Bajguz, 8 minutes ago | 1 author, 4 changes
public sealed class HomeworkTask08 : IHomeworkTask
{
    private readonly IDatabaseContext _databaseContext;

    0 references | Adam Bajguz, 15 hours ago | 1 author, 1 change
    public HomeworkTask08(IDatabaseContext databaseContext)
    {
        _databaseContext = databaseContext;
    }

    11 references | Adam Bajguz, 5 minutes ago | 1 author, 4 changes
    public async Task<object> RunAsync()
    {
        IMongoQueryable<Rating> queryableRating = _databaseContext.Ratings.AsQueryable();
        IMongoQueryable<Title> queryableTitle = _databaseContext.Titles.AsQueryable();

        var partialQuery = queryableRating.Where(x => x.AverageRating == 10)
            .Join(_databaseContext.Titles.AsQueryable(),
                rating => rating.TConst,
                title => title.TConst,
                (rating, title) => new { Rating = rating, Title = title })
            .Select((x) => new { x.Title.Id, x.Rating.AverageRating });

        var listRatings = await queryableRating.Where(x => x.AverageRating == 10)
            .Take(10)
            .Join(_databaseContext.Titles.AsQueryable(),
                rating => rating.TConst,
                title => title.TConst,
                (rating, title) => new { Rating = rating, Title = title })
            .Select((x) => new { x.Title.Id, x.Rating.AverageRating })
            .ToListAsync();

        await partialQuery.ForEachAsync(async (x) =>
        {
            FilterDefinition<Title> filter = Builders<Title>.Filter.Eq(title => title.Id, x.Id);
            UpdateDefinition<Title> update = Builders<Title>.Update.Set("max", 1);

            UpdateResult updateResult = await _databaseContext.Titles.UpdateOneAsync(filter, update);
        });

        var listMax = await queryableTitle.Where(x => x.Max == 1)
            .Take(10)
            .ToListAsync();

        Result result = new Result
        {
            TitlesWithRatings = listRatings,
            TitlesWithMax = listMax
        };
        return result;
    }
}

2 references | Adam Bajguz, 6 minutes ago | 1 author, 2 changes
public sealed class Result
{
    1 reference | Adam Bajguz, 12 hours ago | 1 author, 1 change
    public object? TitlesWithRatings { get; set; }
    1 reference | Adam Bajguz, 6 minutes ago | 1 author, 1 change
    public object? TitlesWithMax { get; set; }
}
```

Rys. 8.1. Fragment najistotniejszego kodu realizującego zadanie nr 8

```

-----[HomeworkTask08]-----
{
  "TitlesWithRatings": [
    {
      "Id": "5ec5683e6273c21896052abb",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec5683f6273c2189605705b",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec568446273c21896069cb2",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec568446273c21896069cb5",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec568496273c2189606fa8e",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec568496273c2189606fa91",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec5684c6273c2189607bbe2",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec5684c6273c2189607bbe3",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec5684c6273c2189607db50",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec5684c6273c2189607db51",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec5684d6273c2189608165c",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec5684d6273c2189608165e",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec5684e6273c21896083e63",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec5684e6273c218960847c3",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec5684f6273c218960874c6",
      "AverageRating": 10.0
    },
    {
      "Id": "5ec5684f6273c218960874ca",
      "AverageRating": 10.0
    }
  ],
  "TitlesWithMax": [
    {
      "Id": "5ec5683e6273c21896052abb",
      "TConst": "tt0050536",
      "TitleType": "tvMovie",
      "PrimaryTitle": "Illusionen",
      "OriginalTitle": "Illusionen",
      "IsAdult": 0,
      "StartYear": 1957,
      "EndYear": 0,
      "RuntimeMinutes": 82,
      "Genres": "\\N",
      "Max": 1
    },
    {
      "Id": "5ec5683f6273c2189605705b",
      "TConst": "tt0061857",
      "TitleType": "tvMovie",
      "PrimaryTitle": "Der Kaktusgarten",
      "OriginalTitle": "Der Kaktusgarten",
      "IsAdult": 0,
      "StartYear": 1967,
      "EndYear": 0,
      "RuntimeMinutes": 90,
      "Genres": "\\N",
      "Max": 1
    },
    {
      "Id": "5ec568446273c21896069cb2",
      "TConst": "tt0111263",
      "TitleType": "short",
      "PrimaryTitle": "Spiral Tribe",
      "OriginalTitle": "Spiral Tribe",
      "IsAdult": 0,
      "StartYear": 1994,
      "EndYear": 0,
      "RuntimeMinutes": 6,
      "Genres": "Short",
      "Max": 1
    },
    {
      "Id": "5ec568446273c21896069cb5",
      "TConst": "tt0111263",
      "TitleType": "short",
      "PrimaryTitle": "Spiral Tribe",
      "OriginalTitle": "Spiral Tribe",
      "IsAdult": 0,
      "StartYear": 1994,
      "EndYear": 0,
      "RuntimeMinutes": 6,
      "Genres": "Short",
      "Max": 1
    },
    {
      "Id": "5ec568496273c2189606fa8e",
      "TConst": "tt0160316",
      "TitleType": "video",
      "PrimaryTitle": "Renegades 2",
      "OriginalTitle": "Renegades 2",
      "IsAdult": 1,
      "StartYear": 1995,
      "EndYear": 0,
      "RuntimeMinutes": 91,
      "Genres": "Adult",
      "Max": 1
    },
    {
      "Id": "5ec568496273c2189606fa91",
      "TConst": "tt0127236",
      "TitleType": "video",
      "PrimaryTitle": "Renegades 2",
      "OriginalTitle": "Renegades 2",
      "IsAdult": 1,
      "StartYear": 1993,
      "EndYear": 0,
      "RuntimeMinutes": 59,
      "Genres": "Adult",
      "Max": 1
    },
    {
      "Id": "5ec5684c6273c2189607bbe2",
      "TConst": "tt0160316",
      "TitleType": "movie",
      "PrimaryTitle": "Girls Loving Girls",
      "OriginalTitle": "Girls Loving Girls",
      "IsAdult": 1,
      "StartYear": 1996,
      "EndYear": 0,
      "RuntimeMinutes": 60,
      "Genres": "Adult",
      "Max": 1
    },
    {
      "Id": "5ec5684c6273c2189607bbe3",
      "TConst": "tt0160316",
      "TitleType": "movie",
      "PrimaryTitle": "Girls Loving Girls",
      "OriginalTitle": "Girls Loving Girls",
      "IsAdult": 1,
      "StartYear": 1996,
      "EndYear": 0,
      "RuntimeMinutes": 60,
      "Genres": "Adult",
      "Max": 1
    },
    {
      "Id": "5ec5684c6273c2189607db50",
      "TConst": "tt0165902",
      "TitleType": "video",
      "PrimaryTitle": "On Location in Palm Springs",
      "OriginalTitle": "On Location in Palm Springs",
      "IsAdult": 1,
      "StartYear": 1993,
      "EndYear": 0,
      "RuntimeMinutes": 59,
      "Genres": "Adult",
      "Max": 1
    },
    {
      "Id": "5ec5684c6273c2189607db51",
      "TConst": "tt0165902",
      "TitleType": "video",
      "PrimaryTitle": "On Location in Palm Springs",
      "OriginalTitle": "On Location in Palm Springs",
      "IsAdult": 1,
      "StartYear": 1993,
      "EndYear": 0,
      "RuntimeMinutes": 59,
      "Genres": "Adult",
      "Max": 1
    }
  ]
}

```

Rys. 8.2. Wynik działania kodu z Rys. 8.1 przedstawiający część danych które uległy zmianie

9 REALIZACJA ZADANIA 9

W celu realizacji zadania nr 9 utworzono kod przedstawiony na Rys. 9.1. Wynik działania tego kodu zaprezentowano na Rys. 9.2. Z uwagi na niejasność w poleceniu wybrano 10 dokumentów a nie 5. Zapytanie zwraca 94865 dokumentów.

```
namespace IMDB.Application.HomeworkTasks
{
    using System.Linq;
    using System.Threading.Tasks;
    using IMDB.Application.Interfaces;
    using IMDB.Domain.Entities;
    using MongoDB.Driver;
    using MongoDB.Driver.Linq;

    ///[HomeworkExclude]
    1 reference | Adam Bajguz, 11 minutes ago | 1 author, 3 changes
    public sealed class HomeworkTask09 : IHomeworkTask
    {
        private readonly IDatabaseContext _databaseContext;

        0 references | Adam Bajguz, 14 hours ago | 1 author, 1 change
        public HomeworkTask09(IDatabaseContext databaseContext)
        {
            _databaseContext = databaseContext;
        }

        11 references | Adam Bajguz, 11 minutes ago | 1 author, 3 changes
        public async Task<object> RunAsync()
        {
            IMongoQueryable<Name> queryable = _databaseContext.Names.AsQueryable();

            await _databaseContext.Names.Indexes.DropOneAsync("primaryName_text");

            CreateIndexModel<Name> indexModel = new CreateIndexModel<Name>(Builders<Name>.IndexKeys.Text(x => x.PrimaryProfession));
            await _databaseContext.Names.Indexes.CreateOneAsync(indexModel);

            IAsyncCursor<MongoDB.Bson.BsonDocument> indexes = await _databaseContext.Names.Indexes.ListAsync();

            var partialQuery = queryable.Where(x => x.BirthYear >= 1950 && x.BirthYear <= 1980 &&
                (x.PrimaryProfession.Contains("actor") ||
                 x.PrimaryProfession.Contains("director")));

            var list = await partialQuery.Take(10)
                .Select((x) => new { x.PrimaryName, x.BirthYear, x.PrimaryProfession })
                .ToListAsync();

            int count = await partialQuery.CountAsync();

            Result result = new Result
            {
                Indexes = indexes.ToEnumerable(),
                People = list,
                Count = count
            };
            return result;
        }

        2 references | Adam Bajguz, 33 minutes ago | 1 author, 1 change
        public sealed class Result
        {
            1 reference | Adam Bajguz, 33 minutes ago | 1 author, 1 change
            public object? Indexes { get; set; }
            1 reference | Adam Bajguz, 33 minutes ago | 1 author, 1 change
            public int Count { get; set; }
            1 reference | Adam Bajguz, 33 minutes ago | 1 author, 1 change
            public object? People { get; set; }
        }
    }
}
```

Rys. 9.1. Fragment najistotniejszego kodu realizującego zadanie nr 9

```

-----[HomeworkTask09]-----
{
  "Indexes": [
    [
      {
        "Name": "v",
        "Value": 2
      },
      {
        "Name": "key",
        "Value": [
          {
            "Name": "_id",
            "Value": 1
          }
        ]
      }
    ],
    {
      "Name": "name",
      "Value": "_id_"
    },
    {
      "Name": "ns",
      "Value": "IMDB.Name"
    }
  ],
  [
    {
      "Name": "v",
      "Value": 2
    },
    {
      "Name": "key",
      "Value": [
        {
          "Name": "_fts",
          "Value": "text"
        },
        {
          "Name": "_ftsx",
          "Value": 1
        }
      ]
    }
  ],
  {
    "Name": "name",
    "Value": "primaryProfession_text"
  },
  {
    "Name": "ns",
    "Value": "IMDB.Name"
  },
  {
    "Name": "weights",
    "Value": [
      {
        "Name": "primaryProfession",
        "Value": 1
      }
    ]
  },
  {
    "Name": "default_language",
    "Value": "english"
  },
  {
    "Name": "language_override",
    "Value": "language"
  },
  {
    "Name": "textIndexVersion",
    "Value": 3
  }
],
  "Count": 94865,
  "People": [
    {
      "PrimaryName": "Elena Koreneva",
      "BirthYear": 1953,
      "PrimaryProfession": "actress,casting_director,soundtrack"
    },
    {
      "PrimaryName": "Brad Pitt",
      "BirthYear": 1963,
      "PrimaryProfession": "actor,producer,soundtrack"
    },
    {
      "PrimaryName": "Pamela Anderson",
      "BirthYear": 1967,
      "PrimaryProfession": "actress,producer,director"
    },
    {
      "PrimaryName": "Patricia Arquette",
      "BirthYear": 1968,
      "PrimaryProfession": "actress,director,producer"
    },
    {
      "PrimaryName": "Rowan Atkinson",
      "BirthYear": 1955,
      "PrimaryProfession": "actor,writer,soundtrack"
    },
    {
      "PrimaryName": "Dan Aykroyd",
      "BirthYear": 1952,
      "PrimaryProfession": "writer,actor,producer"
    },
    {
      "PrimaryName": "Kevin Bacon",
      "BirthYear": 1958,
      "PrimaryProfession": "actor,producer,soundtrack"
    },
    {
      "PrimaryName": "Antonio Banderas",
      "BirthYear": 1960,
      "PrimaryProfession": "actor,soundtrack,producer"
    },
    {
      "PrimaryName": "Luc Besson",
      "BirthYear": 1959,
      "PrimaryProfession": "writer,producer,director"
    },
    {
      "PrimaryName": "Matthew Broderick",
      "BirthYear": 1962,
      "PrimaryProfession": "actor,soundtrack,director"
    }
  ]
}
-----

```

Rys. 9.2. Wynik działania kodu z Rys. 9.1

10 REALIZACJA ZADANIA 10

W celu realizacji zadania nr 10 utworzono kod przedstawiony na Rys. 10.1. Wynik działania tego kodu zaprezentowano na Rys. 10.2. Zapytanie zwraca 617 dokumentów, a liczba indeksów wynosi dwa.

```
namespace IMDB.Application.HomeworkTasks
{
    using System.Collections.Generic;
    using System.Linq;
    using System.Threading.Tasks;
    using IMDB.Application.Interfaces;
    using IMDB.Domain.Entities;
    using MongoDB.Bson;
    using MongoDB.Driver;
    using MongoDB.Driver.Linq;

    ///[HomeworkExclude]
    1 reference | Adam Bajguz, 44 minutes ago | 1 author, 4 changes
    public sealed class HomeworkTask10 : IHomeworkTask
    {
        private readonly IDatabaseContext _databaseContext;

        0 references | Adam Bajguz, 16 hours ago | 1 author, 1 change
        public HomeworkTask10(IDatabaseContext databaseContext)
        {
            _databaseContext = databaseContext;
        }

        11 references | Adam Bajguz, 44 minutes ago | 1 author, 4 changes
        public async Task<object> RunAsync()
        {
            IMongoQueryable<Name> queryable = _databaseContext.Names.AsQueryable();

            //await _databaseContext.Names.Indexes.DropOneAsync("primaryProfession_text");

            CreateIndexModel<Name> indexModel = new CreateIndexModel<Name>(Builders<Name>.IndexKeys.Text(x => x.PrimaryName));
            await _databaseContext.Names.Indexes.CreateOneAsync(indexModel);

            IAsyncCursor<MongoDB.Bson.BsonDocument> indexes = await _databaseContext.Names.Indexes.ListAsync();

            var partialQuery = queryable.Where(x => x.PrimaryName.Contains("Fonda") ||
                                                    x.PrimaryName.Contains("Coppola"));

            var list = await partialQuery.Take(5)
                .Select((x) => new { x.PrimaryName, x.PrimaryProfession })
                .ToListAsync();

            int count = await partialQuery.CountAsync();

            IEnumerable<BsonDocument> enumerable = indexes.ToEnumerable();
            Result result = new Result
            {
                Indexes = enumerable,
                IndexesCount = (await _databaseContext.Names.Indexes.ListAsync()).ToEnumerable().Count(),
                People = list,
                Count = count
            };
            return result;
        }

        2 references | Adam Bajguz, 1 hour ago | 1 author, 1 change
        public sealed class Result
        {
            1 reference | Adam Bajguz, 1 hour ago | 1 author, 1 change
            public object? Indexes { get; set; }
            1 reference | 0 changes | 0 authors, 0 changes
            public int IndexesCount { get; set; }
            1 reference | Adam Bajguz, 1 hour ago | 1 author, 1 change
            public int Count { get; set; }
            1 reference | Adam Bajguz, 1 hour ago | 1 author, 1 change
            public object? People { get; set; }
        }
    }
}
```

Rys. 10.1. Fragment najistotniejszego kodu realizującego zadanie nr 10

```

-----[HomeworkTask10]-----
{
  "Indexes": [
    [
      {
        "Name": "v",
        "Value": 2
      },
      {
        "Name": "key",
        "Value": [
          {
            "Name": "_id",
            "Value": 1
          }
        ]
      }
    ],
    {
      "Name": "name",
      "Value": "_id_"
    },
    {
      "Name": "ns",
      "Value": "IMDB.Name"
    }
  ],
  [
    {
      "Name": "v",
      "Value": 2
    },
    {
      "Name": "key",
      "Value": [
        {
          "Name": "_fts",
          "Value": "text"
        },
        {
          "Name": "_ftsx",
          "Value": 1
        }
      ]
    }
  ],
  {
    "Name": "name",
    "Value": "primaryName_text"
  },
  {
    "Name": "ns",
    "Value": "IMDB.Name"
  },
  {
    "Name": "weights",
    "Value": [
      {
        "Name": "primaryName",
        "Value": 1
      }
    ]
  },
  {
    "Name": "default_language",
    "Value": "english"
  },
  {
    "Name": "language_override",
    "Value": "language"
  },
  {
    "Name": "textIndexVersion",
    "Value": 3
  }
],
  "IndexesCount": 2,
  "Count": 617,
  "People": [
    {
      "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"
    }
  ]
}
-----

```

Rys. 10.2. Wynik działania kodu z Rys. 10.1

11 WNIOSKI

- W zwracanych danych występowały wartości „\N” w miejscu liczb całkowitych. W celu rozwiązania problemu zostały one zmapowane na null.
- Podczas realizacji zadań zauważono że tytuły filmów są zapisane w polu primaryTitle oraz originalTitle. W zadaniu nr 3 i 7 z uwagi na niejednoznaczność treści wypisano obydwa te tytuły.
- Zastosowanie oficjalnej biblioteki do obsługi MongoDB z NuGet w konsolowej aplikacji .NET Core 3.1.4 pozwoliło na szybką realizację zadań w sposób zbliżony do tego jak taka realizacja mogła by wyglądać przy stosowaniu bazy relacyjnej i biblioteki EntityFramework lub EntityFrameworkCore.
- Zastosowanie LINQ, które oficjalna biblioteka MongoDB do .NET wspiera natywnie, pozwoliło na nałożenie składniowej abstrakcji na etapie dostępu do danych. Ponadto w przeciwieństwie do drugiej możliwej do zastosowania metody, tj. używanie metod operujących na m.in. filtrach, składnia jest prostsza i przypomina SQL, chociaż w rzeczywistości i tak zapisane wyrażenia są tłumaczone i wykonywane w języku MQL.
- Metoda Count() / CountAsync() zwracająca dokładną liczbę elementów w kolekcji wykonuje się długo, a szybkość wykonania zależy od rozmiaru kolekcji. Rozwiązaniem tego mogłoby być zastosowanie metody EstimatedCount() / EstimatedCountAsync(), lecz wówczas liczba elementów mogłaby nie odpowiadać rzeczywistej.
- Metoda EstimatedCount() ma złożoność $O(1)$, a Count() – $O(n)$. Jest to spowodowane metadanymi używanymi do zliczania. Zauważono że druga metoda ma znaczne zapotrzebowanie na pamięć operacyjną.
- Użycie ToLower() lub ToUpper() powoduje zastosowanie porównywania nie uwzględniającego wielkości liter. Z punktu widzenia .NET i programisty to zachowanie dziwne, gdyż zezwala na zwrócenie true przez (...).ToUpper().Contains("coppola"), chociaż Contains domyślnie w normalnym zastosowaniu wewnątrz .NET używa porównywania bajtowego z uwzględnieniem wielkości liter StringComparison.Ordinal. Oznacza to że oficjalna biblioteka MongoDB zawiera zachowania, które mogą prowadzić do błędnego wykonywania kodu, który na pozór wygląda dobrze i powinien się tłumaczyć na MQL w całkowicie inny sposób.
- Zauważono, że MongoDB może wykorzystać wszystkie rdzenie procesora na komputerze wielordzeniowym do operacji odczytu, ale dla operacji zapisu w obrębie jedno procesu MongoDB prawdopodobnie może wykorzystywać tylko jeden rdzeń.
- Zauważono, że MongoDB podczas operacji zapisu na dużych kolekcjach ma znaczne zapotrzebowanie na RAM. Przykładowo kolekcja około 10 milionów elementów i rozmiarze około 3GB podczas wykonywania zapytania zmieniającego jedno pole powoduje użycie nawet 8GB pamięci operacyjnej. Jednakże zużycie dysku i procesora w porównaniu do pamięci operacyjnej pozostaje niskie, co może sugerować że MongoDB nie jest optymalnym systemem bazodanowym do zapisu/aktualizacji danych.
- „Złączenia” kolekcji po „kluczu” tekstowym zajmują bardzo dużo czasu i zasobów sprzętowych.