Systemy baz danych - Sprawozdanie

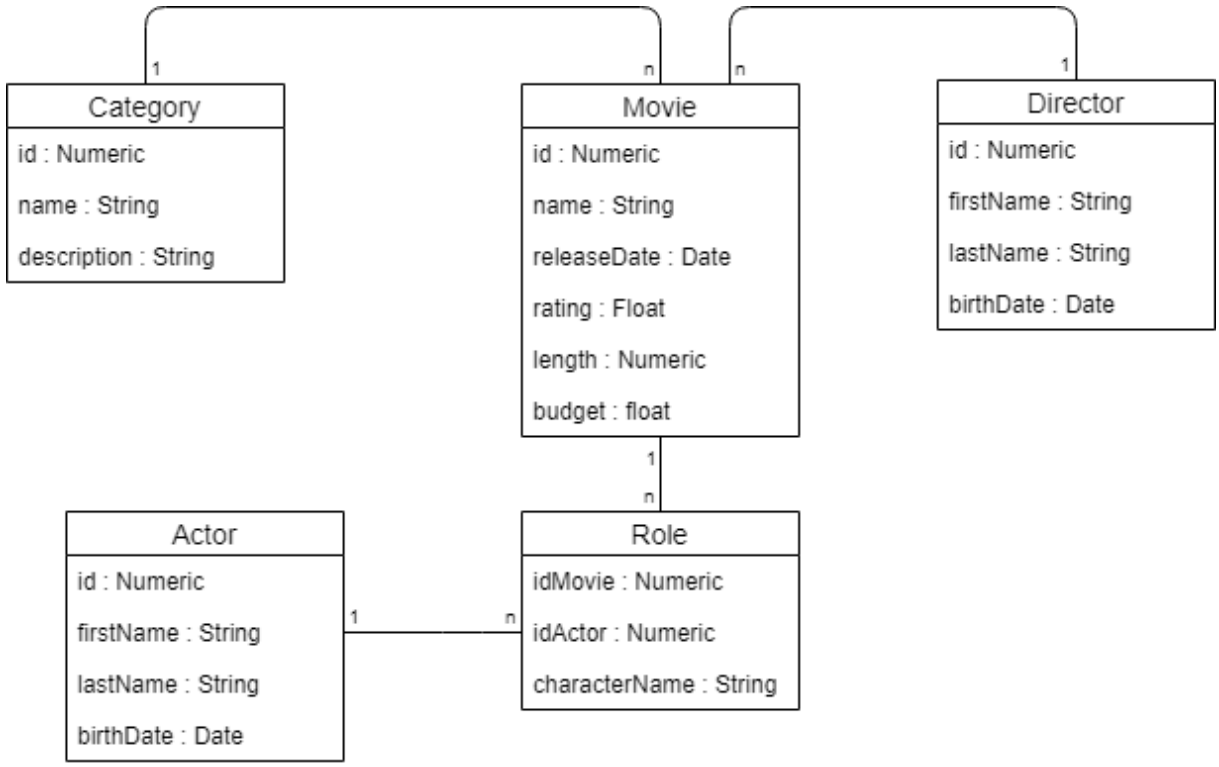
Wykonali : Jarosław Dąbrowski, Artur Dusiński

Grupa : I8E1S4

Temat : Rozproszone bazy danych

1. Projekt i implementacja relacyjnej bazy danych

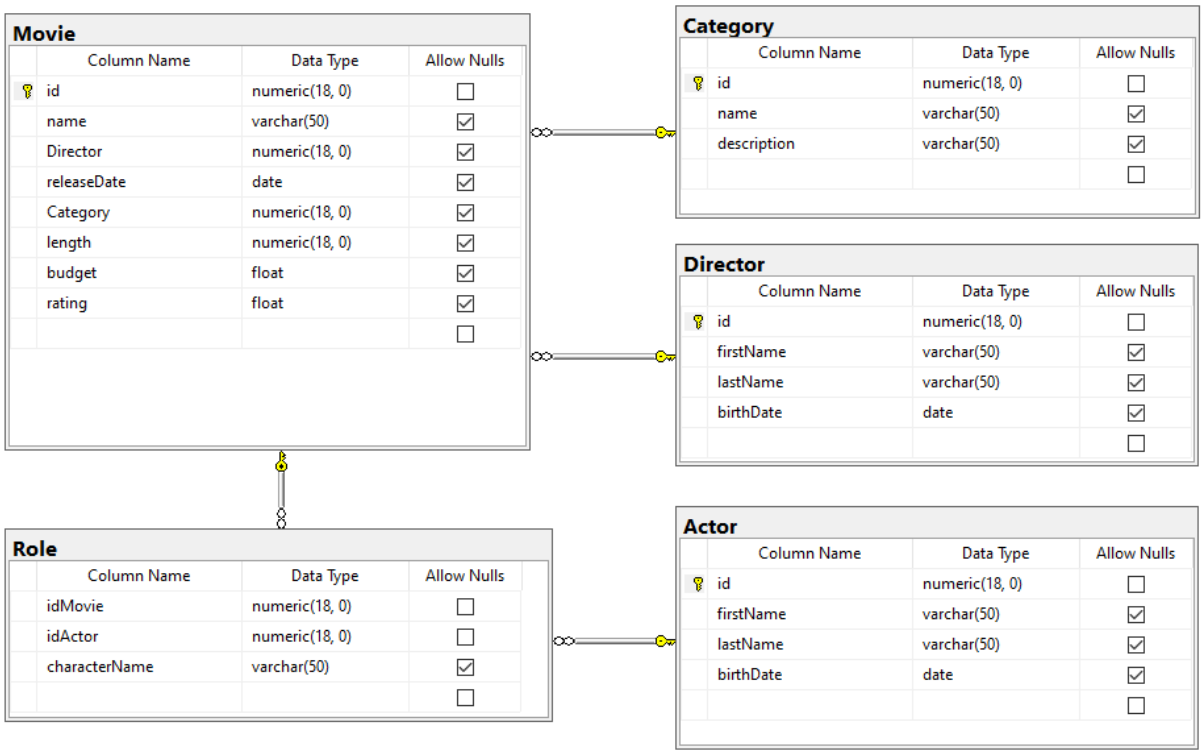
Projektowana baza danych zawiera informacje o filmach oraz o osobach zaangażowanych w produkcje filmowe. Poniżej przedstawiono model logiczny bazy:



Rysunek 1 Model logiczny projektowanej bazy danych

Movie	Podstawowe dane o filmach (nazwa, data premiery, ocena, długość, budżet).
Category	Lista kategorii filmów wraz z opisami. Każdy film może należeć do jednej kategorii.
Director	Podstawowe informacje o reżyserach (imię, nazwisko, data urodzenia). Każdy film może mieć jednego reżysera.
Actor	Podstawowe informacje o aktorach (imię, nazwisko, data urodzenia). W każdym filmie może grać wielu aktorów oraz każdy aktor może grać w wielu filmach.
Role	Intersekcja pomiędzy encjami Actor oraz Movie. W roli przechowywana jest również nazwa granego bohatera

Na podstawie modelu logicznego utworzono model fizyczny. Wykorzystano do tego system MS SQL oraz narzędzie Microsoft SQL Server Management Studio. Poniżej został przedstawiony model fizyczny:



Rysunek 2 Model fizyczny bazy danych

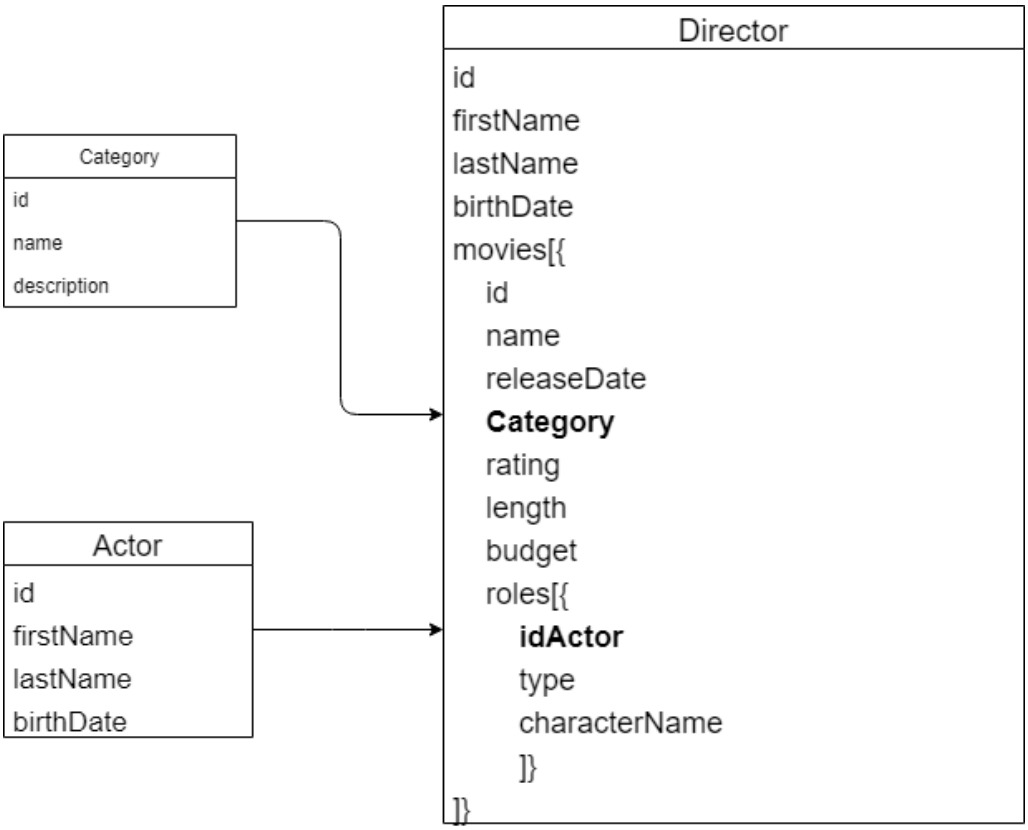
Na podstawie modelu fizycznego został wygenerowany skrypt odpowiedzialny za utworzenie tabel baz danych:

```
USE [SBD_LAB_4]
GO
/****** Object:  Table [dbo].[Actor]    Script Date: 21.06.2019 18:18:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Actor](
	[id] [numeric](18, 0) NOT NULL,
	[firstName] [varchar](50) NULL,
	[lastName] [varchar](50) NULL,
	[birthDate] [date] NULL,
 CONSTRAINT [PK_Actor] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Category]    Script Date: 21.06.2019 18:18:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Category](
	[id] [numeric](18, 0) NOT NULL,
	[name] [varchar](50) NULL,
	[description] [varchar](50) NULL,
 CONSTRAINT [PK_Category] PRIMARY KEY CLUSTERED
(
	[id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
```

```
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Director]    Script Date: 21.06.2019 18:18:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Director](
        [id] [numeric](18, 0) NOT NULL,
        [firstName] [varchar](50) NULL,
        [lastName] [varchar](50) NULL,
        [birthDate] [date] NULL,
 CONSTRAINT [PK_Director] PRIMARY KEY CLUSTERED
(
        [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Movie]    Script Date: 21.06.2019 18:18:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Movie](
        [id] [numeric](18, 0) NOT NULL,
        [name] [varchar](50) NULL,
        [Director] [numeric](18, 0) NULL,
        [releaseDate] [date] NULL,
        [Category] [numeric](18, 0) NULL,
        [length] [numeric](18, 0) NULL,
        [budget] [float] NULL,
        [rating] [float] NULL,
 CONSTRAINT [PK_Movie] PRIMARY KEY CLUSTERED
(
        [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Role]    Script Date: 21.06.2019 18:18:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Role](
        [idMovie] [numeric](18, 0) NOT NULL,
        [idActor] [numeric](18, 0) NOT NULL,
        [characterName] [varchar](50) NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Movie]  WITH CHECK ADD  CONSTRAINT [FK_Movie_Category] FOREIGN KEY([Category])
REFERENCES [dbo].[Category] ([id])
GO
ALTER TABLE [dbo].[Movie] CHECK CONSTRAINT [FK_Movie_Category]
GO
ALTER TABLE [dbo].[Movie]  WITH CHECK ADD  CONSTRAINT [FK_Movie_Director] FOREIGN KEY([Director])
REFERENCES [dbo].[Director] ([id])
GO
ALTER TABLE [dbo].[Movie] CHECK CONSTRAINT [FK_Movie_Director]
GO
ALTER TABLE [dbo].[Role]  WITH CHECK ADD  CONSTRAINT [FK_Role_Actor] FOREIGN KEY([idActor])
REFERENCES [dbo].[Actor] ([id])
GO
ALTER TABLE [dbo].[Role] CHECK CONSTRAINT [FK_Role_Actor]
GO
ALTER TABLE [dbo].[Role]  WITH CHECK ADD  CONSTRAINT [FK_Role_Movie] FOREIGN KEY([idMovie])
REFERENCES [dbo].[Movie] ([id])
GO
ALTER TABLE [dbo].[Role] CHECK CONSTRAINT [FK_Role_Movie]
GO
```

2. Projekt i implementacja bazy NoSQL

Do utworzenie nierelacyjnej bazy danych wybrano system MongoDB. Dane składowane są w tym przypadku w dokumentach JSON co daje możliwość zastosowania hierarchii danych. Poniżej przedstawiono schemat kolekcji w bazie danych:



Rysunek 3 Schemat logiczny nierelacyjnej bazy danych dla MongoDB

W kolekcji Director zawarto listę jego filmów, a dla każdego filmu listę bohaterów.

Jako że jest to baza bezschematowa, nie ma potrzeby tworzenia modelu fizycznego oraz skryptu uruchomieniowego. Rekordy będą wprowadzone do bazy dynamicznie podczas wykonywania skryptu generującego dane.

3. Wypełnienie bazy danych testowymi danymi

Dla celów obliczeniowych przygotowano specjalny skrypt „run_tests”. Jest on odpowiedzialny za generowanie zestawów danych:

- Rekordy filmu dodawane są na bieżąco, przy dodaniu każdego 1000 rekordów uruchamiane są testy szybkości wyszukiwania informacji dla bazy relacyjnej oraz MongoDB
- Przy dodawaniu rekordów filmów dodawani są również reżyserowie oraz role. Dla każdego reżysera przypisywanych jest losowo od 5 do 20 filmów, a dla każdego filmu przypisywanych jest od 3 do 10 ról
- Liczba rekordów dla kategorii oraz aktorów jest wartością stałą ustalaną na początku programu (100 kategorii oraz 100 aktorów)

4. Zapytanie testowe

4.1. Zapytanie testowe – baza relacyjna

Do testów szybkości baz danych wykorzystane będzie zapytanie SQL. Zapytanie wylicza:

- Sumę ile łącznie zarobiły filmy każdego reżysera
- Średnią oceną filmów każdego reżysera
- Ile ról osadzona łącznie dla filmów każdego reżysera

```
SELECT
      d.id as 'ID',
      d.firstName as 'Imię',
      d.lastName as 'Nazwisko',
      d.birthDate  'Data urodzenia',
      sum(m.budget) as 'Suma budżetów filmów',
      avg(m.rating) as 'Średnia ocena filmów',
      (
            SELECT COUNT(r.idMovie)
            FROM Movie m, Role r
            WHERE m.id = r.idMovie
            AND m.Director = d.id
      ) as 'Łączna liczba bohaterów w filmach'
FROM Director d INNER JOIN Movie m
ON (m.Director = d.id)
GROUP BY d.id, d.firstName, d.lastName, d.birthDate
HAVING sum(m.budget) > 100000
ORDER BY 'Suma budżetów filmów'
```

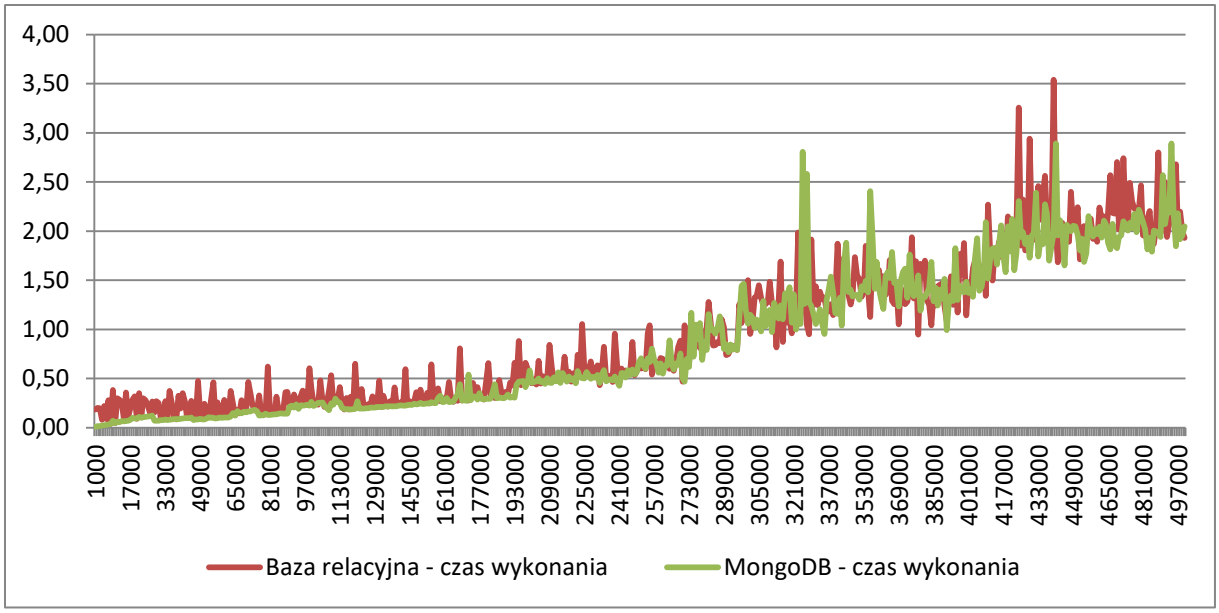
4.2. Zapytanie testowe – MongoDB

Dla systemu MongoDB wykorzystane zostało analogiczne zapytanie wyliczające dokładnie te same wartości co zapytanie SQL. Dzięki temu jak zaprojektowaliśmy bazę danych w MongoDB nie ma potrzeby łączenia kluczy jak w przypadku bazy SQL. Poniżej przedstawiono implementację :

```
db.Director.aggregate([
    {"$group" : {"_id" : "$lastName",
        "count":{"$sum":"$movies.budget"},
        "count":{"$avg":"$movies.rating"},
        "count":{"$movies.roles"}
    }}
])
```

5. Testy szybkości wykonania zapytań

Za pomocą wcześniej opisanego skryptu wykorzystującego zapytania testowe przetestowano obie bazy danych (relacyjną oraz MongoDB). Testy polegały na zebraniu informacji czasowych, ile trwa wyszukanie informacji w zależności od liczby rekordów tabeli „Movie”. Wyniki zostały zapisane w arkuszu kalkulacyjnym dołączonym do sprawozdania. Na podstawie tabeli wstawiono wykres zależności czasu od ilości rekordów :



Rysunek 4 Czas wykonania zapytania w zależności od ilości rekordów w tabeli Movie

Jak widać na powyższym wykresie, dla większości przeprowadzonych testów czasy dostępów były minimalnie krótsze dla systemu MongoDB (w szczególności dla małych zbiorów danych).

6. Podsumowanie pracy

- Zalety MongoDB :
  - Dzięki wykorzystaniu modelu dokumentowego, dostęp do danych jest minimalnie szybszy niż w przypadku tradycyjnych baz relacyjnych
  - Możliwość deklarowania skomplikowanych struktur danych wraz z agregacją danych. W relacyjnych bazach danych opcja ta jest niewykorzystywana
  - Możliwość horyzontalnego skalowania z podziałem danych na przedziały. Takie rozwiązanie daje również możliwość uruchamiania MongoDB na wielu serwerach (rozproszona baza danych).
  - Dzięki wykorzystaniu bezschematowej bazy danych, nie ma potrzeby tworzenia modelu fizycznego oraz
- Zalety bazy relacyjnej:
  - Relacyjne bazy danych są najczęstszym rozwiązaniem wykorzystywanym na rynku
  - Możliwość wykorzystania język zapytań SQL
  - Relacyjne bazy danych charakteryzują się bardzo prostą strukturą. Są łatwe w skalowaniu oraz w rozszerzaniu o kolejne tabele.

7. Załączniki

Projekt zawierający wszystkie skrypty, diagramy i dokumentację można znaleźć w repozytorium GitHub : https://github.com/jdabrowski11926/SBD_LAB4