

A G H

**Akademia Górniczo-Hutnicza
im. Stanisława Staszica
w Krakowie**

Bazy danych

*Dokumentacja do projektu “JamesBondMovies -
Porównanie działania bazy grafowej i relacyjnej”*

Autorzy:

*Agata Nowara
Kinga Wierchomska*

Prowadzący:

*dr inż. Robert Marcjan
dr inż. Leszek Siwik*

Spis treści

Wprowadzenie	3
Neo4j - tworzenie bazy danych	3
Wierzchołki	4
Krawędzie	6
Microsoft SQL Server - tworzenie bazy	9
3.1 Wierzchołki	9
3.2 Krawędzie	11
Porównanie działania baz utworzonych w poprzednich krokach	14
4.1 Do każdego aktora grającego Jamesa Bonda podaj liczbę filmów, w których wystąpił. Wyniki posortuj malejąco	14
4.2 Podaj aktora grającego Jamesa Bonda, który miał w sumie najwięcej dziewczyn	16
4.3 Podaj Bonda mającego kontakt z największą liczbą reżyserów	17
4.4 Dla każdej "dziewczyny Bonda" podaj tytuł filmu, w którym wystąpiła oraz aktora grającego Bonda, któremu towarzyszyła.	19
4.5 Podaj "dziewczynę Bonda", która pojawiła się w największej ilości produkcji.	20
4.6 Czy jest jakaś aktorka ("dziewczyna Bonda"), która miała kontakt z więcej niż jednym Bondem	
	21
Neo4J	21
4.7 Dla każdego Bonda podaj ile razy prezentował się w samochodzie danej marki.	23
4.8 Podaj markę samochodów z największą liczbą różnych modeli ukazanych w filmach.	24
4.9 Podaj nazwę modelu samochodu, który ukazał się w największej ilości produkcji.	26
4.10 Pokaż reżyserów realizujących kilka filmów z serii wraz z ich liczbą	27
4.11 Dla filmu o największym box-office pokaż jego tytuł, rok premiery, reżysera, Bonda	29
4.12 Przy realizacji filmu, przez którego reżysera pojawiło się najwięcej aut. Podaj tytuł filmu oraz liczbę modeli	30
4.13 Ile pojawiło się samochodów w filmie z największym box-office	31
4.14 W czasie pracy, którego reżysera w sumie pojawiło się najwięcej aut	32
4.15 Podaj Bonda mającego kontakt, tylko z jednym reżyserem.	34
4.16 Podaj łączny box - office całej serii filmów o Jamesie Bondzie	35
4.17 Podaj średni box - office całej serii	36
4.18 Podaj markę i model wszystkich samochodów, w których nazwie wystąpiło słowo sedan lub coupe	37
4.19 Podaj tytuły wszystkich filmów zaczynających się od słowa "The"	38
4.20 Wypisz wszystkie osoby, które w nazwie posiadają "ros"	39
Porównanie / wnioski	40
Aplikacja webowa	41

1. Wprowadzenie

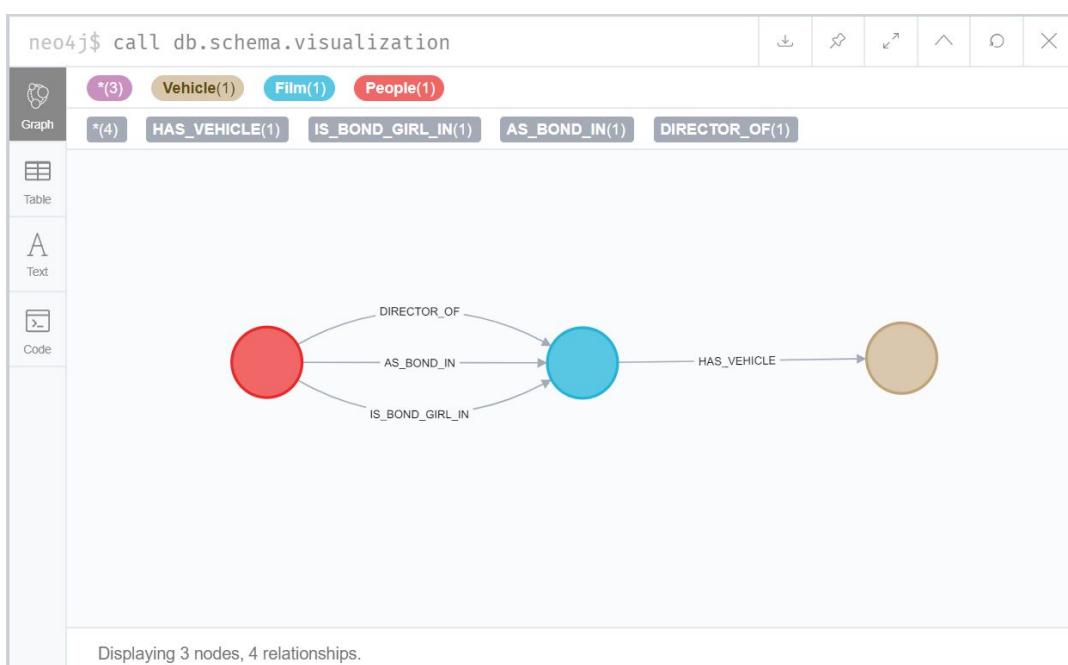
Tematem projektu jest porównanie działania bazy danych stworzonej z wykorzystaniem Neo4J, z relacyjną bazą danych zbudowaną z pomocą Microsoft SQL Server. W obydwu bazach zostanie zamodelowany graf. W bazie relacyjnej zdecydowano się na przetestowanie oferowanej od 2017 roku przez Microsoft SQL Server opcji przechowywania danych w postaci zbliżonej do grafowej. Koncepcja polega na tym, że wierzchołki grafu należące do tej samej grupy (np. wierzchołki reprezentujące ludzi) są przechowywane w jednej tabeli. Zawiera ona atrybuty opisujące dany węzeł oraz tzw node_id. W oddzielnych tabelach przechowywane są krawędzie modelowanego grafu. Każda tabela zawiera co najmniej trzy kolumny - tzw. edge_id, id węzła od, którego krawędź wychodzi (from_id) oraz id węzła, do którego wchodzi (to_id). Wymienione atrybuty jasno wskazują, że modelujemy graf skierowany. Do tabeli krawędzi można dodać także dodatkowe parametry, jednak z uwagi na strukturę zimportowanej przez nas grafowej bazy do Neo4J, nie skorzystano z takiej opcji.

Obydwie bazy przechowują dane dotyczące filmowej serii o Jamesie Bondzie. Znajdują się tu informacje o reżyserze danego filmu, aktorze grającym w nim główną rolę, "dziewczynach Bonda", jego "super samochodach" itd.

Przy tworzeniu tego projektu kładziemy nacisk na pokazanie, która baza danych lepiej radzi sobie w podanych sytuacjach oraz sprawdzenie czy różnica między ich wynikami jest znacząca.

2. Neo4j - tworzenie bazy danych

Została stworzona z wykorzystaniem istniejącej już bazy Neo4j ->
<https://neo4j.com/graphgist/know-more-about-james-bond-movie>



neo4j\$ call db.schema.nodeTypeProperties					
	nodeType	nodeLabels	propertyName	propertyTypes	mandatory
A Text	"::People"	["People"]	"Name"	["String"]	true
Code	"::People"	["People"]	"Role"	["String"]	false
	"::Vehicle"	["Vehicle"]	"Brand"	["String"]	true
	"::Vehicle"	["Vehicle"]	"Model"	["String"]	true
	"::Film"	["Film"]	"Year"	["Long"]	true
	"::Film"	["Film"]	"Box"	["Long"]	true
	"::Film"	["Film"]	"Name"	["String"]	true

Started streaming 7 records after 1 ms and completed after 26 ms.

a. Wierzchołki

Na początku stworzone zostały wierzchołki reprezentujące

- ludzi związanych z filmem
 - przykładowe utworzenie:

```
CREATE ( :People {Name:"Terence Young" })
```

- następnie do aktorek przyporządkowana została rola:

```
MATCH (a:People {Name:"Eunice Gayson"}) SET
a.Role="Sylvia Trench" ;
```

- filmy
 - przykładowe utworzenie:

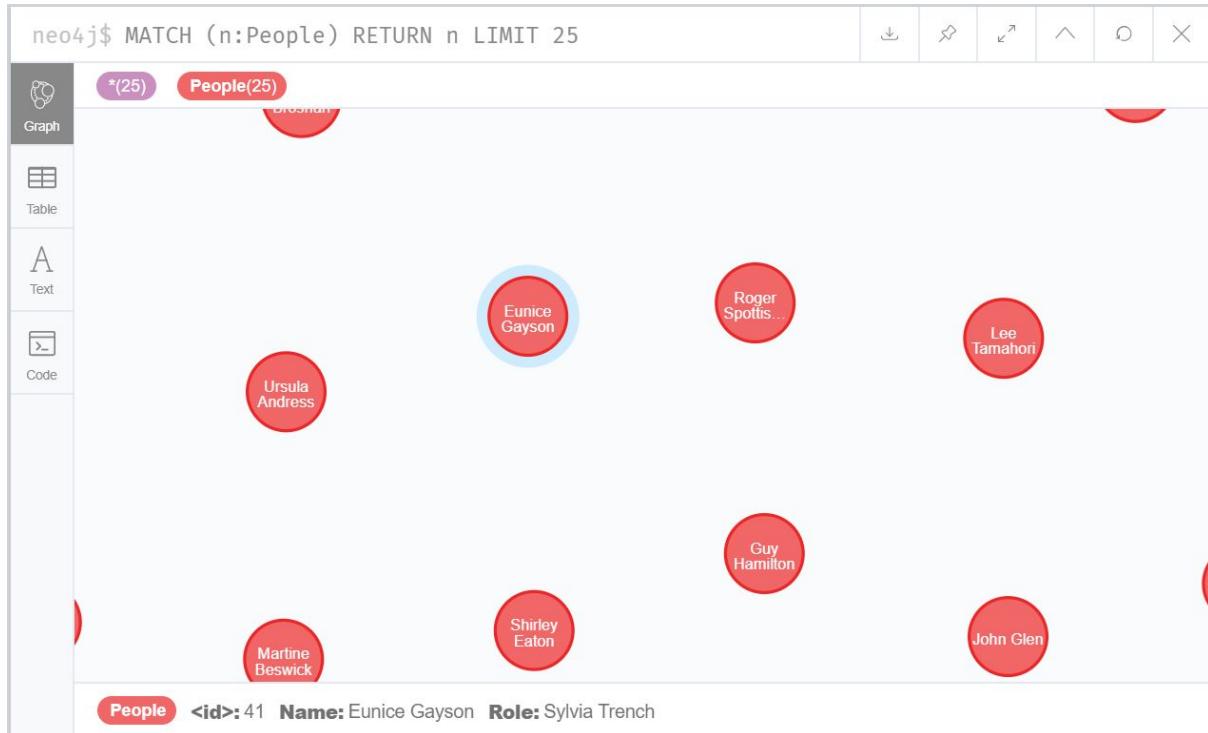
```
CREATE ( :Film {Name:"Dr. No", Year:1962,
Box:440759072 })
```

- samochody
 - przykładowe utworzenie:

```
CREATE ( :Vehicle {Brand:"Alfa", Model:"Alfa
Romeo GTV6" })
```

Efekt końcowy utworzenia wierzchołków

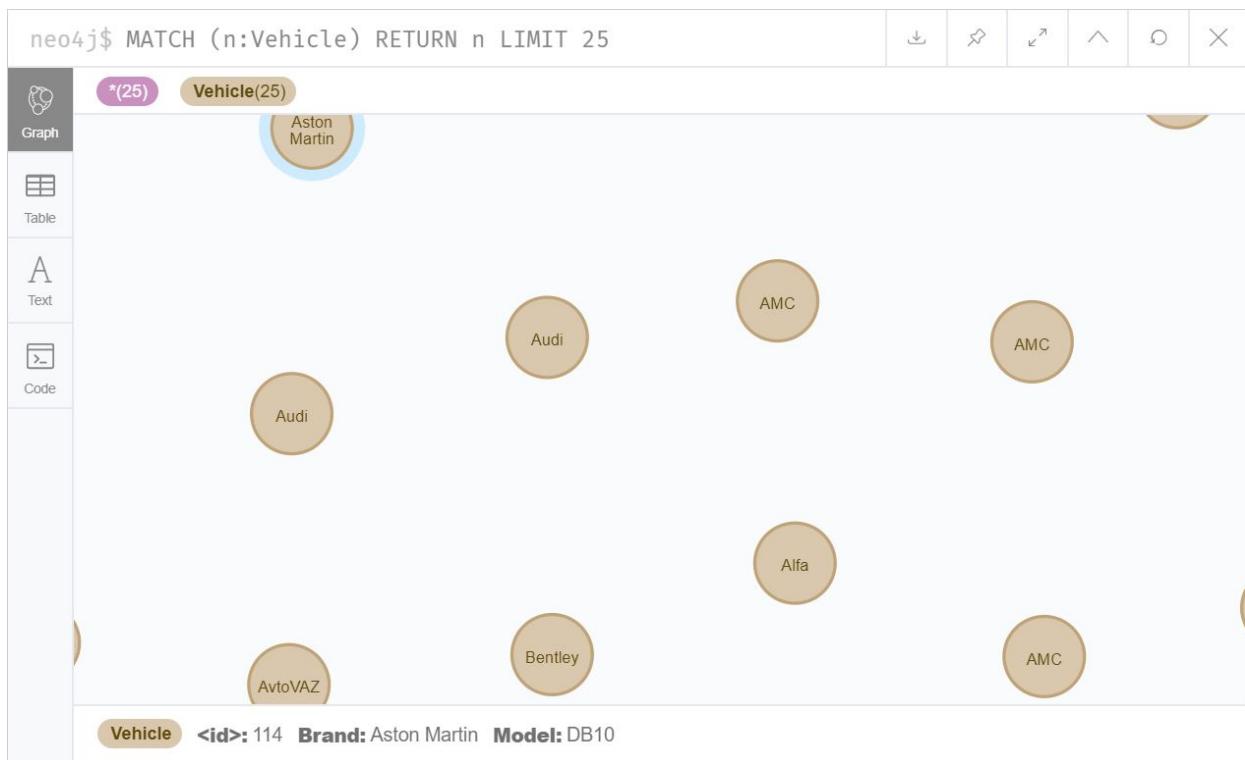
- ludzie związani z filmem



- filmy



- samochody



b. Krawędzie

Następnie utworzone zostały krawędzie między wierzchołkami reprezentujące relacje między nimi

- połączenie aktora grającego Bonda z filmem

- przykładowe utworzenie:

```
MATCH (a:People), (b:Film) WHERE a.Name="Sean Connery" AND b.Name="Dr. No" MERGE (a)-[r:AS_BOND_IN]->(b);
```

- połączenie aktorki grającej “dziewczynę Bonda” z filmem

- przykładowe utworzenie:

```
MATCH (a:Film), (b:People) WHERE a.Name="Dr. No" AND b.Name="Eunice Gayson" MERGE (b)-[:IS_BOND_GIRL_IN]->(a);
```

- połączenie reżysera z filmem

- przykładowe utworzenie:

```
MATCH (a:People), (b:Film) WHERE a.Name="Terence Young" AND b.Name="Dr. No" MERGE (a)-[r:DIRECTOR_OF]->(b);
```

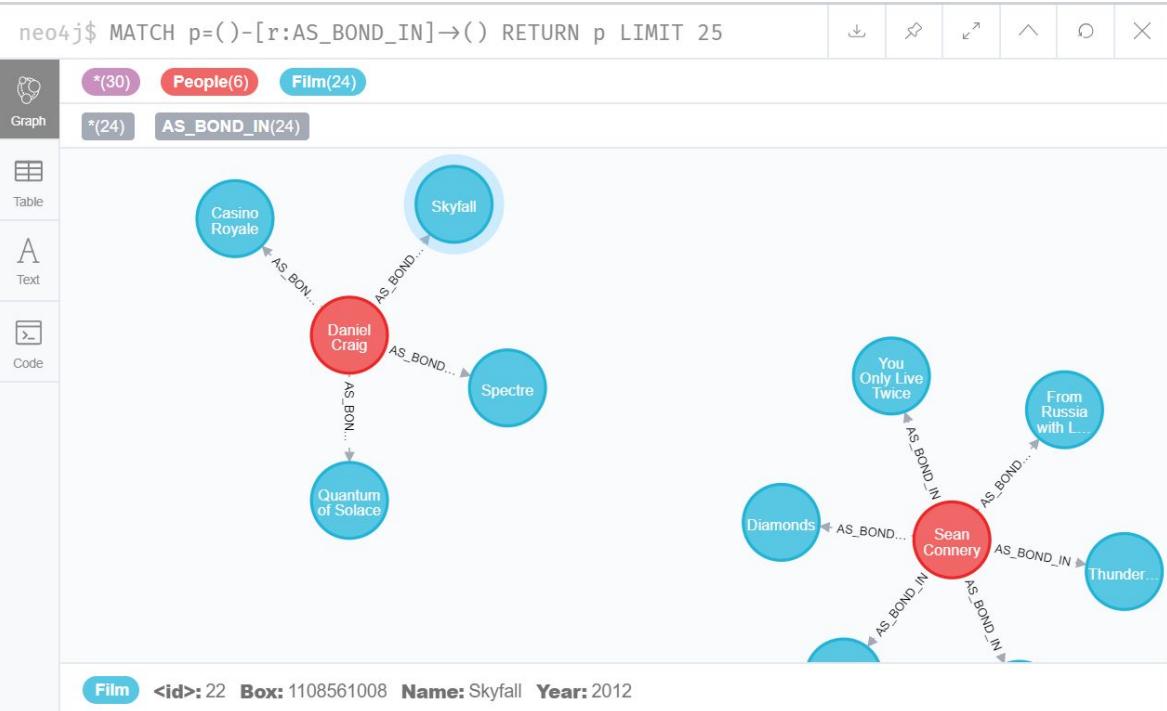
- połączenie samochodu z filmem

- przykładowe utworzenie:

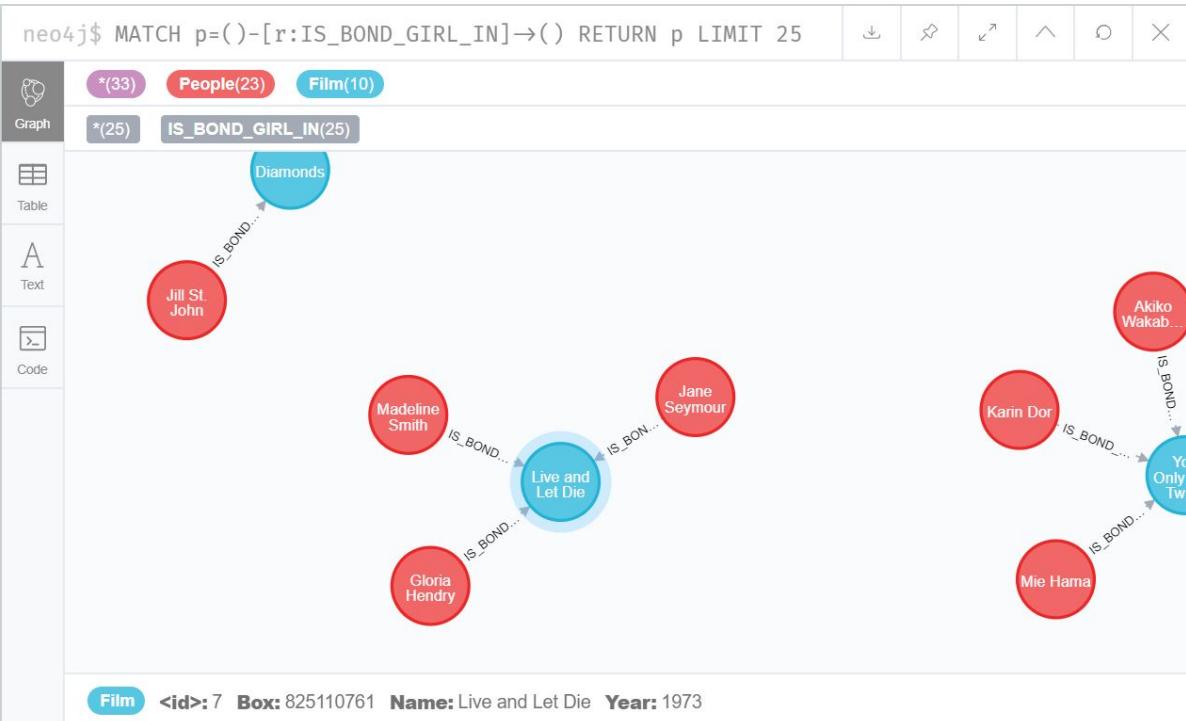
```
MATCH (a:Film), (b:Vehicle) WHERE
a.Name="Octopussy" AND b.Model="Alfa Romeo GTV6"
MERGE (a)-[:HAS_VEHICLE]->(b);
```

Efekt końcowy utworzenia krawędzi

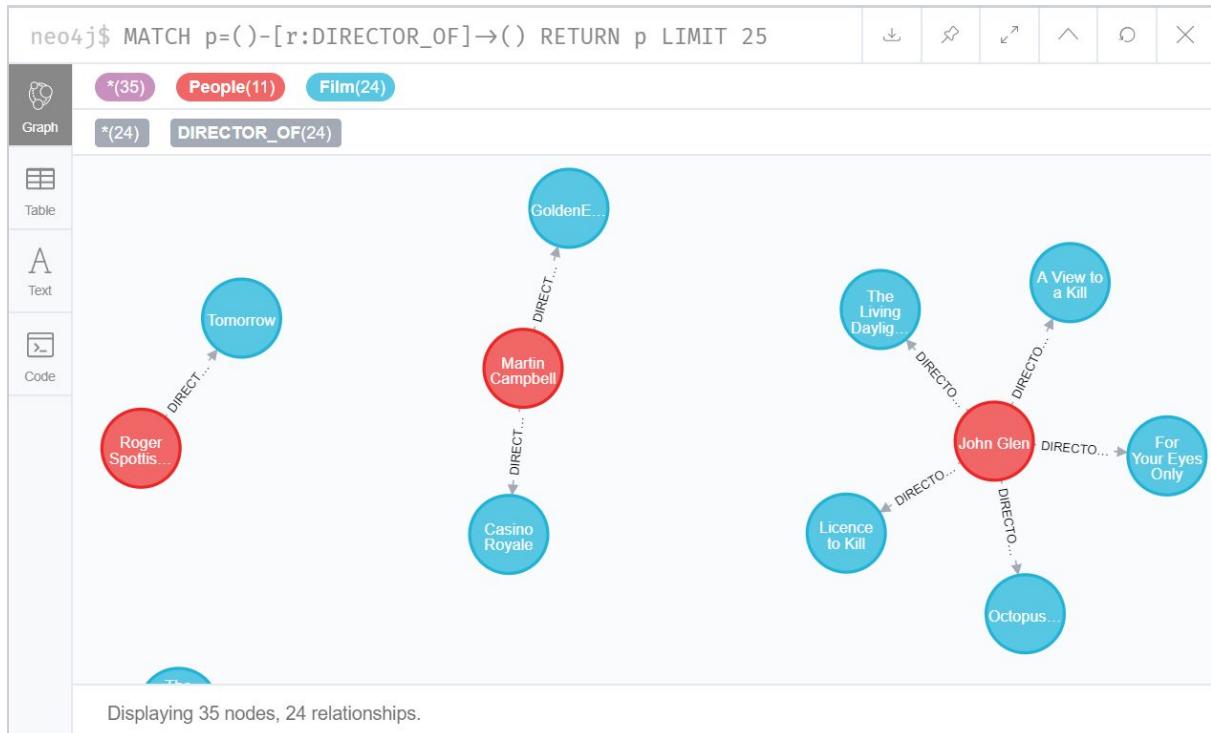
- połączenie aktora grającego Bonda z filmem



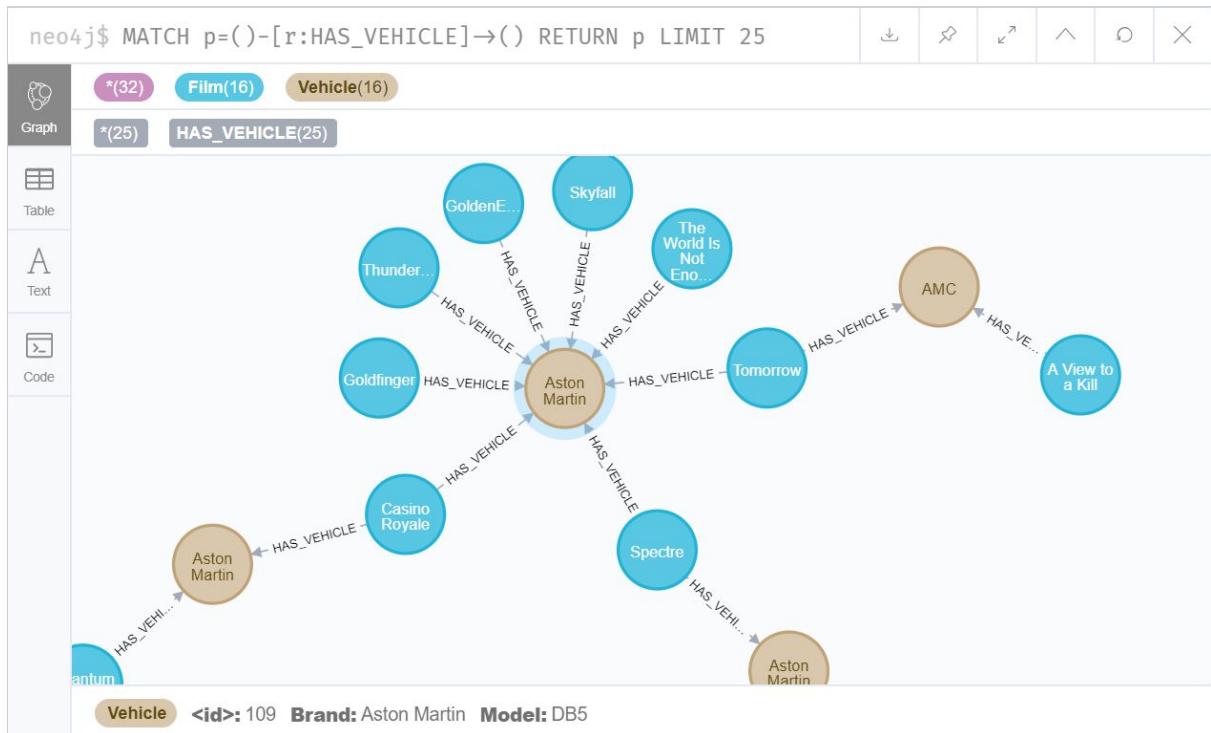
- połączenie aktorki grającej “dziewczynę Bonda” z filmem



- połączenie reżysera z filmem



- połączenie samochodu z filmem



3. Microsoft SQL Server - tworzenie bazy

W tym kroku starano się zamodelować graf korzystając tym razem z relacyjnej bazy danych Microsoft SQL Server. Sposób tworzenia grafu oparto na metodzie opisanej we wprowadzeniu.

Aby nie utracić grafowego charakteru powyżej zaprezentowanych danych, stworzono osobne tabele dla wierzchołków oraz krawędzi.

3.1 Wierzchołki

Na początku stworzone zostały tabele reprezentujące wierzchołki i przechowujące o nich dane

- tabela zawierająca **ludzi** związanych z filmem
 - schemat tworzenia tabeli People jako węzeł grafu

```
CREATE TABLE People (
    Id integer identity(1,1) primary key,
    Name varchar(100) not null,
    Role varchar(100)
) as node;
```

- przykładowe wprowadzenie danych do tabeli

```
INSERT INTO People VALUES
('Terence Young', null),
('Guy Hamilton', null),
('Lewis Gilbert', null),
('Peter R. Hunt', null),
('John Glen', null),
('Martin Campbell', null),
('Roger Spottiswoode', null),
('Michael Apted', null),
('Lee Tamahori', null),
('Marc Forster', null),
('Sam Mendes', null),
('Sean Connery', 'James Bond'),
('George Lazenby', 'James Bond'),
('Roger Moore', 'James Bond'),
('Timothy Dalton', 'James Bond'),
('Pierce Brosnan', 'James Bond'),
('Daniel Craig', 'James Bond'),
```

- pokazanie części zawartości tabeli **People**

	\$node_id_B56B7B9A51FC4EA2994817DF3556878E	Id	Name	Role
1	{"type":"node","schema":"dbo","table":"People","id":0}	1	Terence Young	<null>
2	{"type":"node","schema":"dbo","table":"People","id":1}	2	Guy Hamilton	<null>
3	{"type":"node","schema":"dbo","table":"People","id":2}	3	Lewis Gilbert	<null>
4	{"type":"node","schema":"dbo","table":"People","id":3}	4	Peter R. Hunt	<null>
5	{"type":"node","schema":"dbo","table":"People","id":4}	5	John Glen	<null>
6	{"type":"node","schema":"dbo","table":"People","id":5}	6	Martin Campbell	<null>
7	{"type":"node","schema":"dbo","table":"People","id":6}	7	Roger Spottiswoode	<null>
8	{"type":"node","schema":"dbo","table":"People","id":7}	8	Michael Apted	<null>
9	{"type":"node","schema":"dbo","table":"People","id":8}	9	Lee Tamahori	<null>
10	{"type":"node","schema":"dbo","table":"People","id":9}	10	Marc Forster	<null>
11	{"type":"node","schema":"dbo","table":"People","id":10}	11	Sam Mendes	<null>
12	{"type":"node","schema":"dbo","table":"People","id":11}	12	Sean Connery	James Bond
13	{"type":"node","schema":"dbo","table":"People","id":12}	13	George Lazenby	James Bond
14	{"type":"node","schema":"dbo","table":"People","id":13}	14	Roger Moore	James Bond
15	{"type":"node","schema":"dbo","table":"People","id":14}	15	Timothy Dalton	James Bond
16	{"type":"node","schema":"dbo","table":"People","id":15}	16	Pierce Brosnan	James Bond
17	{"type":"node","schema":"dbo","table":"People","id":16}	17	Daniel Craig	James Bond

- tabela zawierająca **filmy**

- schemat tworzenia tabeli Film jako węzeł grafu

```
CREATE TABLE Film (
    Id integer identity(1,1) primary key,
    Year int not null,
    Box varchar(100) not null,
    Name varchar(100) not null
) as node;
```

- przykładowe wprowadzenie danych do tabeli

```
INSERT INTO Film VALUES
(1962, '440759072', 'Dr. No'),
(1963, '576277964', 'From Russia with Love'),
(1964, '912257512', 'Goldfinger'),
(1965, '1014941117', 'Thunderball'),
(1967, '756544419', 'You Only Live Twice'),
(1969, '505899782', 'On Her Majesty''s Secret Service'),
(1971, '648514469', 'Diamonds Are Forever'),
(1973, '825110761', 'Live and Let Die'),
(1974, '448249281', 'The Man with the Golden Gun'),
```

- pokazanie części zawartości tabeli **Film**

	\$node_id_D59F9DC70DF549D890C360D6BBC2A9B9	Id	Year	Box	Name
1	{"type":"node","schema":"dbo","table":"Film","id":0}	1	1962	440759072	Dr. No
2	{"type":"node","schema":"dbo","table":"Film","id":1}	2	1963	576277964	From Russia with Love
3	{"type":"node","schema":"dbo","table":"Film","id":2}	3	1964	912257512	Goldfinger
4	{"type":"node","schema":"dbo","table":"Film","id":3}	4	1965	1014941117	Thunderball
5	{"type":"node","schema":"dbo","table":"Film","id":4}	5	1967	756544419	You Only Live Twice
6	{"type":"node","schema":"dbo","table":"Film","id":5}	6	1969	505899782	On Her Majesty's Secret Service
7	{"type":"node","schema":"dbo","table":"Film","id":6}	7	1971	648514469	Diamonds Are Forever
8	{"type":"node","schema":"dbo","table":"Film","id":7}	8	1973	825110761	Live and Let Die
9	{"type":"node","schema":"dbo","table":"Film","id":8}	9	1974	448249281	The Man with the Golden Gun

- tabela zawierająca pojazdy wykorzystane w filmach
 - schemat tworzenia tabeli **Vehicle** jako węzeł grafu

```
CREATE TABLE Vehicle (
    Id integer identity(1,1) primary key,
    Brand varchar(100) not null,
    Model varchar(100) not null
) as node;
```

- przykładowe wprowadzenie danych do tabeli

```
INSERT INTO Vehicle VALUES
('Alfa','Alfa Romeo GTV6'),
('Alfa','Alfa Romeo 159'),
('Alfa','Alfa Romeo 156'),
('AMC','AMC Hornet'),
('AMC','AMC Matador coupe'),
('AMC','AMC Matador sedan'),
```

- pokazanie części zawartości tabeli **Vehicle**

\$node_id_E661FA245412448C8177676B27C8D36A	Id	Brand	Model
1 {"type":"node","schema":"dbo","table":"Vehicle","id":0}	1	Alfa	Alfa Romeo GTV6
2 {"type":"node","schema":"dbo","table":"Vehicle","id":1}	2	Alfa	Alfa Romeo 159
3 {"type":"node","schema":"dbo","table":"Vehicle","id":2}	3	Alfa	Alfa Romeo 156
4 {"type":"node","schema":"dbo","table":"Vehicle","id":3}	4	AMC	AMC Hornet
5 {"type":"node","schema":"dbo","table":"Vehicle","id":4}	5	AMC	AMC Matador coupe
6 {"type":"node","schema":"dbo","table":"Vehicle","id":5}	6	AMC	AMC Matador sedan

3.2 Krawędzie

Stworzenie tabeli, których zadaniem jest pełnienie roli krawędzi w grafie. Rezultat ten uzyskujemy przez podanie ID dwóch węzłów, które łączy ze sobą dana krawędź. Każda krawędź posiada również unikalny identyfikator

- tabela odpowiadająca za połączenie filmu z aktorem grającym w nim Jamesa Bonda
 - schemat tworzenie tabeli będącej krawędzią między węzłami

```
CREATE TABLE AsBondIn AS EDGE;
```

- przykładowe dopasowania aktora do filmu

```
INSERT INTO AsBondIn VALUES
((SELECT $node_id FROM People WHERE ID = 12), (SELECT $node_id FROM Film WHERE ID = 1)),
((SELECT $node_id FROM People WHERE ID = 12), (SELECT $node_id FROM Film WHERE ID = 2)),
((SELECT $node_id FROM People WHERE ID = 12), (SELECT $node_id FROM Film WHERE ID = 3)),
((SELECT $node_id FROM People WHERE ID = 12), (SELECT $node_id FROM Film WHERE ID = 4)),
((SELECT $node_id FROM People WHERE ID = 12), (SELECT $node_id FROM Film WHERE ID = 5)),
((SELECT $node_id FROM People WHERE ID = 13), (SELECT $node_id FROM Film WHERE ID = 6)),
((SELECT $node_id FROM People WHERE ID = 12), (SELECT $node_id FROM Film WHERE ID = 7)),
((SELECT $node_id FROM People WHERE ID = 14), (SELECT $node_id FROM Film WHERE ID = 8)),
```

- pokazanie części zawartości tabeli AsBondIn (identyfikatory obydwu wierzchołków oraz identyfikator krawędzi)

\$edge_id_F765876B36C744039BEF767BD2FC5B43	\$from_id_3FC953DC864847B6BE7368876987716F	\$to_id_B1ED3EBDC9DE46F08474A9757996A402
1 {"type":"edge","schema":"dbo","table":"AsBondIn","id":0}	{"type":"node","schema":"dbo","table":"People","id":11}	{"type":"node","schema":"dbo","table":"Film","id":0}
2 {"type":"edge","schema":"dbo","table":"AsBondIn","id":1}	{"type":"node","schema":"dbo","table":"People","id":11}	{"type":"node","schema":"dbo","table":"Film","id":1}
3 {"type":"edge","schema":"dbo","table":"AsBondIn","id":2}	{"type":"node","schema":"dbo","table":"People","id":11}	{"type":"node","schema":"dbo","table":"Film","id":2}
4 {"type":"edge","schema":"dbo","table":"AsBondIn","id":3}	{"type":"node","schema":"dbo","table":"People","id":11}	{"type":"node","schema":"dbo","table":"Film","id":3}
5 {"type":"edge","schema":"dbo","table":"AsBondIn","id":4}	{"type":"node","schema":"dbo","table":"People","id":11}	{"type":"node","schema":"dbo","table":"Film","id":4}
6 {"type":"edge","schema":"dbo","table":"AsBondIn","id":5}	{"type":"node","schema":"dbo","table":"People","id":11}	{"type":"node","schema":"dbo","table":"Film","id":5}
7 {"type":"edge","schema":"dbo","table":"AsBondIn","id":6}	{"type":"node","schema":"dbo","table":"People","id":11}	{"type":"node","schema":"dbo","table":"Film","id":6}
8 {"type":"edge","schema":"dbo","table":"AsBondIn","id":7}	{"type":"node","schema":"dbo","table":"People","id":13}	{"type":"node","schema":"dbo","table":"Film","id":7}
9 {"type":"edge","schema":"dbo","table":"AsBondIn","id":8}	{"type":"node","schema":"dbo","table":"People","id":13}	{"type":"node","schema":"dbo","table":"Film","id":8}

- tabela odpowiadająca za połączenie filmu z aktorką grającą w nim “dziewczynę Bonda”
 - schemat tworzenie tabeli będącej krawędzią między węzłami

```
CREATE TABLE IsBondGirlIn AS EDGE;
```

- przykładowe dopasowanie aktorki do filmu

INSERT INTO IsBondGirlIn VALUES
((SELECT \$node_id FROM People WHERE ID = 18), (SELECT \$node_id FROM Film WHERE ID = 1)),
((SELECT \$node_id FROM People WHERE ID = 19), (SELECT \$node_id FROM Film WHERE ID = 1)),
((SELECT \$node_id FROM People WHERE ID = 20), (SELECT \$node_id FROM Film WHERE ID = 1)),
((SELECT \$node_id FROM People WHERE ID = 18), (SELECT \$node_id FROM Film WHERE ID = 2)),
((SELECT \$node_id FROM People WHERE ID = 21), (SELECT \$node_id FROM Film WHERE ID = 2)),
((SELECT \$node_id FROM People WHERE ID = 22), (SELECT \$node_id FROM Film WHERE ID = 2)),
((SELECT \$node_id FROM People WHERE ID = 23), (SELECT \$node_id FROM Film WHERE ID = 2)),
((SELECT \$node_id FROM People WHERE ID = 24), (SELECT \$node_id FROM Film WHERE ID = 3)),

- pokazanie części zawartości tabeli IsBondGirlIn (identyfikatory obydwu wierzchołków oraz identyfikator krawędzi)

\$edge_id_5A09EAE695CC48E0A51ACE87C7978D01	\$from_id_8CAD80C653D14896BEBE6D6D4D28F7EE	\$to_id_1CFEBF394EDA4066A23968448AA515FE
1 {"type":"edge","schema":"dbo","table":"IsBondGirlIn","id":0}	{"type":"node","schema":"dbo","table":"People","id":17}	{"type":"node","schema":"dbo","table":"Film","id":0}
2 {"type":"edge","schema":"dbo","table":"IsBondGirlIn","id":1}	{"type":"node","schema":"dbo","table":"People","id":18}	{"type":"node","schema":"dbo","table":"Film","id":0}
3 {"type":"edge","schema":"dbo","table":"IsBondGirlIn","id":2}	{"type":"node","schema":"dbo","table":"People","id":19}	{"type":"node","schema":"dbo","table":"Film","id":0}
4 {"type":"edge","schema":"dbo","table":"IsBondGirlIn","id":3}	{"type":"node","schema":"dbo","table":"People","id":17}	{"type":"node","schema":"dbo","table":"Film","id":1}
5 {"type":"edge","schema":"dbo","table":"IsBondGirlIn","id":4}	{"type":"node","schema":"dbo","table":"People","id":17}	{"type":"node","schema":"dbo","table":"Film","id":1}
6 {"type":"edge","schema":"dbo","table":"IsBondGirlIn","id":5}	{"type":"node","schema":"dbo","table":"People","id":20}	{"type":"node","schema":"dbo","table":"Film","id":1}
7 {"type":"edge","schema":"dbo","table":"IsBondGirlIn","id":6}	{"type":"node","schema":"dbo","table":"People","id":22}	{"type":"node","schema":"dbo","table":"Film","id":1}
8 {"type":"edge","schema":"dbo","table":"IsBondGirlIn","id":7}	{"type":"node","schema":"dbo","table":"People","id":23}	{"type":"node","schema":"dbo","table":"Film","id":2}
9 {"type":"edge","schema":"dbo","table":"IsBondGirlIn","id":8}	{"type":"node","schema":"dbo","table":"People","id":24}	{"type":"node","schema":"dbo","table":"Film","id":2}

- tabela odpowiadająca za połączenie filmu z jego reżyserem
 - schemat tworzenie tabeli będącej krawędzią między węzłami

```
CREATE TABLE DirectorOf AS EDGE;
```

- przykładowe dopasowanie aktorki do filmu

INSERT INTO DirectorOf VALUES
((SELECT \$node_id FROM People WHERE ID = 1), (SELECT \$node_id FROM Film WHERE ID = 1)),
((SELECT \$node_id FROM People WHERE ID = 1), (SELECT \$node_id FROM Film WHERE ID = 2)),
((SELECT \$node_id FROM People WHERE ID = 2), (SELECT \$node_id FROM Film WHERE ID = 3)),
((SELECT \$node_id FROM People WHERE ID = 1), (SELECT \$node_id FROM Film WHERE ID = 4)),
((SELECT \$node_id FROM People WHERE ID = 3), (SELECT \$node_id FROM Film WHERE ID = 5)),
((SELECT \$node_id FROM People WHERE ID = 4), (SELECT \$node_id FROM Film WHERE ID = 6)),
((SELECT \$node_id FROM People WHERE ID = 2), (SELECT \$node_id FROM Film WHERE ID = 7)),
((SELECT \$node_id FROM People WHERE ID = 2), (SELECT \$node_id FROM Film WHERE ID = 8)),

- pokazanie części zawartości tabeli DirectorOf (identyfikatory obydwu wierzchołków oraz identyfikator krawędzi)

\$edge_id_0CF040D3B15F455FB6C7B97015B43704	\$from_id_FFAAAD6DDE534E35AF17E770CE59AC9	\$to_id_171E6B1AFB8D44338716E0C4F24F2A86
1 {"type":"edge","schema":"dbo","table":"Director0f","id":0}	{"type":"node","schema":"dbo","table":"People","id":8}	{"type":"node","schema":"dbo","table":"Film","id":0}
2 {"type":"edge","schema":"dbo","table":"Director0f","id":1}	{"type":"node","schema":"dbo","table":"People","id":8}	{"type":"node","schema":"dbo","table":"Film","id":1}
3 {"type":"edge","schema":"dbo","table":"Director0f","id":2}	{"type":"node","schema":"dbo","table":"People","id":8}	{"type":"node","schema":"dbo","table":"Film","id":2}
4 {"type":"edge","schema":"dbo","table":"Director0f","id":3}	{"type":"node","schema":"dbo","table":"People","id":8}	{"type":"node","schema":"dbo","table":"Film","id":3}
5 {"type":"edge","schema":"dbo","table":"Director0f","id":4}	{"type":"node","schema":"dbo","table":"People","id":8}	{"type":"node","schema":"dbo","table":"Film","id":4}
6 {"type":"edge","schema":"dbo","table":"Director0f","id":5}	{"type":"node","schema":"dbo","table":"People","id":8}	{"type":"node","schema":"dbo","table":"Film","id":5}
7 {"type":"edge","schema":"dbo","table":"Director0f","id":6}	{"type":"node","schema":"dbo","table":"People","id":8}	{"type":"node","schema":"dbo","table":"Film","id":6}
8 {"type":"edge","schema":"dbo","table":"Director0f","id":7}	{"type":"node","schema":"dbo","table":"People","id":8}	{"type":"node","schema":"dbo","table":"Film","id":7}
9 {"type":"edge","schema":"dbo","table":"Director0f","id":8}	{"type":"node","schema":"dbo","table":"People","id":8}	{"type":"node","schema":"dbo","table":"Film","id":8}

- tabela odpowiadająca za połączenie filmu z jego reżyserem
 - schemat tworzenie tabeli będącej krawędzią między węzłami

```
CREATE TABLE HasVehicle AS EDGE;
```

- przykładowe dopasowanie filmu do modeli samochodów Jamesa Bonda

```
INSERT INTO HasVehicle VALUES
((SELECT $node_id FROM Film WHERE ID = 13), (SELECT $node_id FROM Vehicle WHERE ID = 1)),
((SELECT $node_id FROM Film WHERE ID = 22), (SELECT $node_id FROM Vehicle WHERE ID = 2)),
((SELECT $node_id FROM Film WHERE ID = 22), (SELECT $node_id FROM Vehicle WHERE ID = 3)),
((SELECT $node_id FROM Film WHERE ID = 9), (SELECT $node_id FROM Vehicle WHERE ID = 4)),
((SELECT $node_id FROM Film WHERE ID = 9), (SELECT $node_id FROM Vehicle WHERE ID = 5)),
((SELECT $node_id FROM Film WHERE ID = 9), (SELECT $node_id FROM Vehicle WHERE ID = 6)),
((SELECT $node_id FROM Film WHERE ID = 11), (SELECT $node_id FROM Vehicle WHERE ID = 7)),
((SELECT $node_id FROM Film WHERE ID = 11), (SELECT $node_id FROM Vehicle WHERE ID = 8)),
```

- pokazanie części zawartości tabeli HasVehicle (identyfikatory obydwu wierzchołków oraz identyfikator krawędzi)

\$edge_id_FBD70363D9394E9389284AF5E52B04F3	\$from_id_ECE17A6BDC1B4E158889B2CDC1E1EDCC2	\$to_id_ECB28A5F9BE247B2A859A053C47142FF
1 {"type":"edge","schema":"dbo","table":"HasVehicle","id":0}	{"type":"node","schema":"dbo","table":"Film","id":12}	{"type":"node","schema":"dbo","table":"Vehicle","id":0}
2 {"type":"edge","schema":"dbo","table":"HasVehicle","id":1}	{"type":"node","schema":"dbo","table":"Film","id":21}	{"type":"node","schema":"dbo","table":"Vehicle","id":1}
3 {"type":"edge","schema":"dbo","table":"HasVehicle","id":2}	{"type":"node","schema":"dbo","table":"Film","id":21}	{"type":"node","schema":"dbo","table":"Vehicle","id":2}
4 {"type":"edge","schema":"dbo","table":"HasVehicle","id":3}	{"type":"node","schema":"dbo","table":"Film","id":8}	{"type":"node","schema":"dbo","table":"Vehicle","id":3}
5 {"type":"edge","schema":"dbo","table":"HasVehicle","id":4}	{"type":"node","schema":"dbo","table":"Film","id":8}	{"type":"node","schema":"dbo","table":"Vehicle","id":4}
6 {"type":"edge","schema":"dbo","table":"HasVehicle","id":5}	{"type":"node","schema":"dbo","table":"Film","id":8}	{"type":"node","schema":"dbo","table":"Vehicle","id":5}
7 {"type":"edge","schema":"dbo","table":"HasVehicle","id":6}	{"type":"node","schema":"dbo","table":"Film","id":10}	{"type":"node","schema":"dbo","table":"Vehicle","id":6}
8 {"type":"edge","schema":"dbo","table":"HasVehicle","id":7}	{"type":"node","schema":"dbo","table":"Film","id":10}	{"type":"node","schema":"dbo","table":"Vehicle","id":7}
9 {"type":"edge","schema":"dbo","table":"HasVehicle","id":8}	{"type":"node","schema":"dbo","table":"Film","id":13}	{"type":"node","schema":"dbo","table":"Vehicle","id":8}

- wizualizacja stworzonych i omówionych powyżej tabel w Data Grip

IsBondGirlIn	Director0f
graph_id_E3D0B1FE63DE47B993F62B46BD7A0DCE bigint \$edge_id_5A09EA695CC48EOA51ACE87C7978D01 nvarchar(1000) from_obj_id_6986AE8D1E9B4A77816B369D6FB50D9 int from_id_357489394F264EFA6B4629EEBA61C5083E bigint \$from_id_8CAD80C653D14856BEBE6D6D4D28F7EE nvarchar(1000) to_obj_id_F4EC0CA45E18486FB58BB838AAB8E01 int to_id_A38DF4D6B836C48CBAA018756A0DG1EBE bigint \$to_id_1CFEBF394EDA4066A2396844BAA515FE nvarchar(1000)	graph_id_8727B825D15542C6A0A45A0D82CB09A bigint \$edge_id_0CF040D3B15F455FB6C7B97015B43704 nvarchar(1000) from_obj_id_1205E3C8E1B47C78DF1B702E6B1223 int from_id_7B0942D22E4C4761A1D58A0193FA3F78 bigint \$from_id_FFAAAD6DDE534E35AF17E770CE59AC9 nvarchar(1000) to_obj_id_66129F89A0E14E968BAB1EAFCB7ABD4 int to_id_7A66CDE3DEBB4E82B620CAD80237B3DF bigint \$to_id_171E6B1AFB8D44338716E0C4F24F2A86 nvarchar(1000)

HasVehicle	AsBondIn
graph_id_5A4D8431055C49CEA89ECC3CA4F89B90 bigint \$edge_id_FBD70363D9394E9389284AF5E52B04F3 nvarchar(1000) from_obj_id_67387E8E847C4936BA77CFDEF6F06FAO int from_id_E32AAC394A94EC9CE52180342055F bigint \$from_id_ECE17A6BDC1B4E158889B2CDC1E1EDCC2 nvarchar(1000) to_obj_id_BD16E1893B934C78AA8B21F026D9DE05 int to_id_A915749701094C43964C99D8F61529FE bigint \$to_id_ECB28A5F9BE247B2A859A053C47142FF nvarchar(1000)	graph_id_7F1F3B8CFF64350A65165FF8DEE7102 bigint \$edge_id_F76587E836C7440398EF767B02FC5B43 nvarchar(1000) from_obj_id_5900310465F7423C961DE5E62C7B69C1 int from_id_9323AE0852B480580220B7FC30E881 bigint \$from_id_3FC953DC864847B68EF7388876987716F nvarchar(1000) to_obj_id_7889E13AD0974A6EB6DBE77C1367037 int to_id_A7790DEFAD49D47F59B17E1F5C1F283F1 bigint \$to_id_B1ED3EBDC9DE46F08474A9757996A402 nvarchar(1000)

Film	People
graph_id_d6251E6481F2548C5A4F4349AF15EC370 bigint \$node_id_D59F9DC70DF549DB890C360D6BBC2A9B9 nvarchar(1000) Year int Box varchar(100) Name varchar(100)	graph_id_3740FF24D56044E0B46A4C75530F6042 bigint \$node_id_B56B7B9A51FC4EA2994817DF3556878E nvarchar(1000) Name varchar(100) Role varchar(100)

Vehicle	
graph_id_5227B8C312674FCE962A41CE6D5668DE bigint \$node_id_E661FA24541244BC81776B27C8D36A nvarchar(1000) Brand varchar(100) Model varchar(100)	

4. Porównanie działania baz utworzonych w poprzednich krokach

4.1 Do każdego aktora grającego Jamesa Bonda podaj liczbę filmów, w których wystąpił. Wyniki posortuj malejąco

- Neo4j

The screenshot shows the Neo4j browser interface. At the top, there is a code editor containing the following Cypher query:

```
1 MATCH (p:People)-[:AS_BOND_IN]-(m:Film)
2 RETURN p.Name AS Actor, count(p.Name) AS BondMovies ORDER BY BondMovies desc
```

Below the code editor is a results table. The table has two columns: "Actor" and "BondMovies". The data is as follows:

Actor	BondMovies
"Roger Moore"	7
"Sean Connery"	6
"Pierce Brosnan"	4
"Daniel Craig"	4
"Timothy Dalton"	2
"George Lazenby"	1

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **7.7 ms**

- Microsoft SQL Server - wariant I

```
6 select P.Name as Actor, count(P.Name) as NumberOfBondMovies
7 from People as P, Film as F, AsBondIn
8 where match (P - (AsBondIn) -> F)
9 group by P.Name
10 order by NumberOfBondMovies desc
```

The screenshot shows the Microsoft SQL Server Management Studio results pane. The results tab is selected, displaying the following data:

Actor	NumberOfBondMovies
Roger Moore	7
Sean Connery	6
Daniel Craig	4
Pierce Brosnan	4
Timothy Dalton	2
George Lazenby	1

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **9.4 ms**

- Microsoft SQL Server - wariant II

```

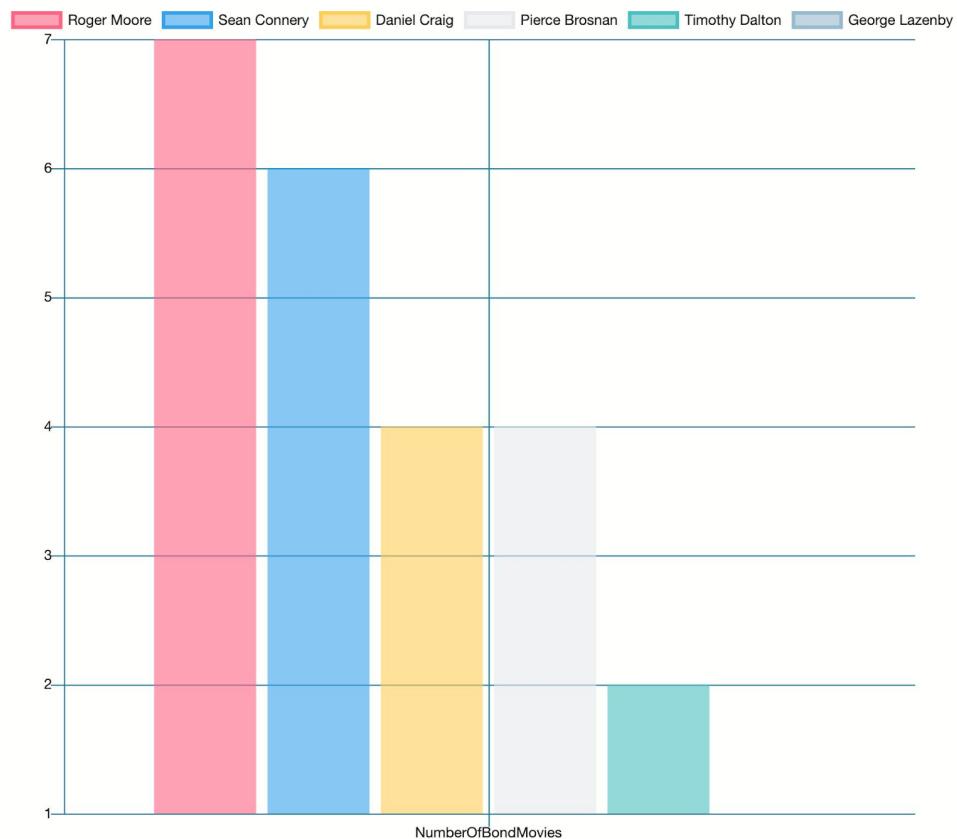
16  select P.Name as Actor, count(P.Name) as NumberOfBondMovies
17  from People as P
18  join AsBondIn ABI on P.$node_id=ABI.$from_id
19  join Film F on F.$node_id = ABI.$to_id
20  group by P.Name
21  order by NumberOfBondMovies desc
--
```

Results **Messages**

	Actor	NumberOfBondMovies
1	Roger Moore	7
2	Sean Connery	6
3	Daniel Craig	4
4	Pierce Brosnan	4
5	Timothy Dalton	2
6	George Lazenby	1

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **9.6 ms**

Liczba filmów przypadających na każdego aktora grającego Bonda



4.2 Podaj aktora grającego Jamesa Bonda, który miał w sumie najwięcej dziewczyn

- Neo4J

The screenshot shows the Neo4j browser interface. In the top query editor, there is a Cypher query:

```
neo4j$ MATCH (a:People)-[:IS_BOND_GIRL_IN]→
  (m:Film)←[:AS_BOND_IN]-(b:People)
  RETURN b.Name as Bond, count (distinct
  a.Name) as BondGirlsCount ORDER BY
  BondGirlsCount desc LIMIT 1
```

Below the query editor is a results table:

Bond	BondGirlsCount
"Roger Moore"	18

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **5.5 ms**

- Microsoft SQL Server - wariant I

```
25 select top 1 with ties Bond.Name as Bond, count(distinct Girl.Name) as BondGirlsCount
26 from People Bond, AsBondIn, People Girl, IsBondGirlIn, Film
27 where match (Girl - (IsBondGirlIn) -> Film <- (AsBondIn) - Bond)
28 group by Bond.Name
29 order by BondGirlsCount desc
30
```

The screenshot shows the Microsoft SQL Server Management Studio results pane. It has two tabs: "Results" and "Messages". The "Results" tab displays a table with one row:

Bond	BondGirlsCount
Roger Moore	18

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **9.4 ms**

- Microsoft SQL Server - wariant II

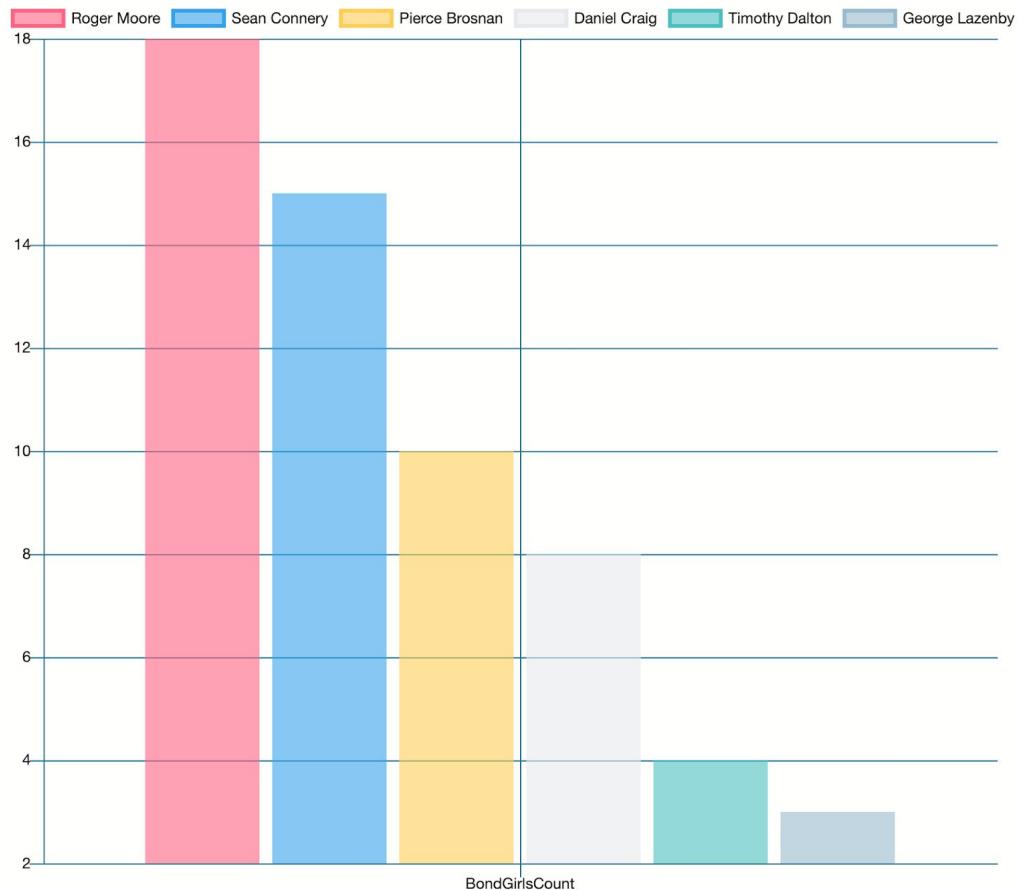
```
31 select top 1 with ties Bond.Name as Bond, count(distinct Girl.Name) as BondGirlsCount
32 from People Bond
33 join AsBondIn ABI on Bond.$node_id = ABI.$from_id
34 join Film F on ABI.$to_id = F.$node_id
35 join IsBondGirlIn IBGI on F.$node_id = IBGI.$to_id
36 join People Girl on Girl.$node_id = IBGI.$from_id
37 group by Bond.Name
38 order by BondGirlsCount desc
39
```

The screenshot shows the Microsoft SQL Server Management Studio results pane. It has two tabs: "Results" and "Messages". The "Results" tab displays a table with one row:

Bond	BondGirlsCount
Roger Moore	18

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **9.2 ms**

Liczba dziewczyn przypadająca na każdego aktora grającego Bonda



4.3 Podaj Bonda mającego kontakt z największą liczbą reżyserów

- Neo4J

```
neo4j$ MATCH (a:People)-[:DIRECTOR_OF]→  
  (m:Film)←[:AS_BOND_IN]-(b:People)  
  RETURN b.Name as Actor, count(distinct  
  a.Name) as DirectorCount ORDER BY  
  DirectorCount desc LIMIT 1
```

neo4j\$ MATCH (a:People)-[:DIRECTOR_OF]→(...)

Actor	DirectorCount
"Pierce Brosnan"	4

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **3.2 ms**

- Microsoft SQL Server - wariant I

```
41  select top 1 with ties Bond.Name as Actor, count(distinct Director.Name) as DirectorCount
42  from People Bond, AsBondIn, People Director, DirectorOf, Film
43  where match (Director - (DirectorOf) -> Film <- (AsBondIn) - Bond)
44  group by Bond.Name
45  order by DirectorCount desc
46
```

Results Messages

	Actor	DirectorCount
1	Pierce Bro...	4

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **8.8 ms**

- Microsoft SQL Server - wariant II

```
47  select top 1 with ties Bond.Name as Actor, count(distinct Director.Name) as DirectorCount
48  from People Bond
49  join AsBondIn ABI on Bond.$node_id = ABI.$from_id
50  join Film F on F.$node_id = ABI.$to_id
51  join DirectorOf DO on F.$node_id = DO.$to_id
52  join People Director on Director.$node_id = DO.$from_id
53  group by Bond.Name
54  order by DirectorCount desc
```

Results Messages

	Actor	DirectorCount
1	Pierce Brosnan	4

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **9.4 ms**

Liczba reżyserów, z którymi współpracował każdy aktor grający Bonda



4.4 Dla każdej “dziewczyny Bonda” podaj tytuł filmu, w którym wystąpiła oraz aktora grającego Bonda, któremu towarzyszyła.

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, there is a code editor window containing the following Cypher query:

```
1 MATCH (a:People)-[:IS_BOND_GIRL_IN]→
  (m:Film)←[:AS_BOND_IN]-(b:People)
2 RETURN a.Name as BondGirl, m.Name as Film,
  b.Name as Bond
```

Below the code editor is a results table with three columns: BondGirl, Film, and Bond. The table contains the following data:

BondGirl	Film	Bond
"Ursula Andress"	"Dr. No"	"Sean Connery"
"Zena Marshall"	"Dr. No"	"Sean Connery"
"Eunice Gayson"	"Dr. No"	"Sean Connery"
"Daniela Bianchi"	"From Russia with Love"	"Sean Connery"
"Martine Beswick"	"From Russia with Love"	"Sean Connery"
"Aliza Gur"	"From Russia with Love"	"Sean Connery"
"Eunice Gayson"	"From Russia with Love"	"Sean Connery"
"Honor Blackman"	"Goldfinger"	"Sean Connery"

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **3.3 ms**

- Microsoft SQL Server - wariant I

```
57 select Girl.Name as BondGirl, Film.Name as Name, Bond.Name as Bond
58   from People Girl, People Bond, Film, IsBondGirlIn, AsBondIn
59   where match (Girl - (IsBondGirlIn) -> Film <- (AsBondIn) - Bond)
60
```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) results pane. It displays the results of the provided T-SQL query. The results are presented in a table with three columns: BondGirl, Name, and Bond.

	BondGirl	Name	Bond
1	Eunice Gayson	Dr. No	Sean Connery
2	Zena Marshall	Dr. No	Sean Connery
3	Ursula Andress	Dr. No	Sean Connery
4	Eunice Gayson	From Russi...	Sean Connery
5	Aliza Gur	From Russi...	Sean Connery
6	Martine Beswi...	From Russi...	Sean Connery
7	Daniela Bianc...	From Russi...	Sean Connery
8	Shirley Eaton	Goldfinger	Sean Connery
9	Honor Blackman	Goldfinger	Sean Connery

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **9.3 ms**

- Microsoft SQL Server -wariant II

```

61  select Girl.Name as BondGirl, F.Name as Name, Bond.Name as Bond
62  from People Girl
63  join IsBondGirlIn IBGI on Girl.$node_id = IBGI.$from_id
64  join Film F on F.$node_id = IBGI.$to_id
65  join AsBondIn ABI on F.$node_id = ABI.$to_id
66  join People Bond on Bond.$node_id = ABI.$from_id
67

```

Results Messages

	BondGirl	Name	Bond
1	Eunice Gayson	Dr. No	Sean Connery
2	Zena Marshall	Dr. No	Sean Connery
3	Ursula Andress	Dr. No	Sean Connery
4	Eunice Gayson	From Russia with Love	Sean Connery
5	Aliza Gur	From Russia with Love	Sean Connery
6	Martine Beswick	From Russia with Love	Sean Connery
7	Daniela Bianchi	From Russia with Love	Sean Connery
8	Shirley Eaton	Goldfinger	Sean Connery
9	Honor Blackman	Goldfinger	Sean Connery

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **8.6 ms**

4.5 Podaj “dziewczynę Bonda”, która pojawiła się w największej ilości produkcji.

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, there is a code editor window containing a Cypher query:

```

neo4j$ MATCH (p:People)-[:IS_BOND_GIRL_IN]→
  (m:Film) RETURN p.Name AS Actor,
  count(p.Name) AS BondMovies ORDER BY
  BondMovies desc LIMIT 2

```

Below the code editor is a results table. The table has two columns: "Actor" and "BondMovies". The data is as follows:

Actor	BondMovies
"Maud Adams"	2
"Eunice Gayson"	2

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **2.6 ms**

- Microsoft SQL Server - wariant I

```
69 select top 1 with ties Girl.Name as Actor, count(Film.Name) as NumberOfBondMovies  
70 from People Girl, IsBondGirlIn, Film  
71 where match (Girl - (IsBondGirlIn) -> Film)  
72 group by Girl.Name  
73 order by NumberOfBondMovies desc
```

Results Messages

	Actor	NumberOfBondMovies
1	Eunice Gayson	2
2	Maud Adams	2

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **7.8 ms**

- Microsoft SQL Server - wariant II

```
75 select top 1 with ties Girl.Name as Actor, count(F.Name) as NumberOfBondMovies  
76 from People Girl  
77 join IsBondGirlIn IBGI on Girl.$node_id = IBGI.$from_id  
78 join Film F on F.$node_id = IBGI.$to_id  
79 group by Girl.Name  
80 order by NumberOfBondMovies desc
```

Results Messages

	Actor	NumberOfBondMovies
1	Eunice Gayson	2
2	Maud Adams	2

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **8.4 ms**

4.6 Czy jest jakaś aktorka ("dziewczyna Bonda"), która miała kontakt z więcej niż jednym Bondem

- Neo4J

```
1 MATCH (a:People)-[:IS_BOND_GIRL_IN]→  
      (m:Film)←[:AS_BOND_IN]-(b:People) WITH a,  
      count(distinct b.Name) as BondCount WHERE  
      BondCount > 1  
2 RETURN BondCount, a.Name
```



```
neo4j$ MATCH (a:People)-[:IS_BOND_GIRL_IN]→(...)
```



(no changes, no records)

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **4.3 ms**

- Microsoft SQL Server - wariant I

```
83  select Girl.Name as BondGirl, count(distinct Bond.Name) as DistinctBondCount
84  from People Girl, People Bond, IsBondGirlIn, AsBondIn, Film
85  where match (Girl - (IsBondGirlIn) -> Film <- (AsBondIn) - Bond)
86  group by Girl.Name
87  having count(distinct Bond.Name) > 1
88  order by DistinctBondCount desc
89
```

Results Messages

BondGirl	DistinctBondCount
----------	-------------------

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **9.0 ms**

- Microsoft SQL Server -wariant II

```
90  select Girl.Name as BondGirl, count(distinct Bond.Name) as DistinctBondCount
91  from People Girl
92  join IsBondGirlIn IBGI on Girl.$node_id = IBGI.$from_id
93  join Film F on F.$node_id = IBGI.$to_id
94  join AsBondIn ABI on F.$node_id = ABI.$to_id
95  join People Bond on Bond.$node_id = ABI.$from_id
96  group by Girl.Name
97  having count(distinct Bond.Name) > 1
98  order by DistinctBondCount desc
```

Results Messages

BondGirl	DistinctBondCo...
----------	-------------------

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **9.2 ms**

4.7 Dla każdego Bonda podaj ile razy prezentował się w samochodzie danej marki.

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, there is a code editor window containing the following Cypher query:

```
neo4j$ MATCH (p:People)-[:AS_BOND_IN]→(m:Film)-[:HAS_VEHICLE]→(v:Vehicle)  
RETURN p.Name as Bond, v.Brand as Brand, count(v.Brand) as BrandCount
```

Below the code editor is a results table. The table has three columns: Bond, Brand, and BrandCount. The data is as follows:

Bond	Brand	BrandCount
"Sean Connery"	"Sunbeam"	1
"Sean Connery"	"Chevrolet"	5
"Sean Connery"	"Ford"	16
"Sean Connery"	"Rolls-Royce"	3
"Sean Connery"	"Mercedes Benz"	2
"Sean Connery"	"Aston Martin"	2
"Sean Connery"	"Toyota"	1
"George Lazenby"	"Rolls-Royce"	1

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **3.6 ms**

- Microsoft SQL Server -wariant I

```
101  select Bond.Name, Vehicle.Brand, count(Vehicle.Brand)  
102  from People Bond, AsBondIn, Vehicle, HasVehicle, Film  
103  where match (Bond - (AsBondIn) → Film - (HasVehicle) → Vehicle)  
104  group by Bond.Name, Vehicle.Brand  
105  order by Bond.Name
```

The screenshot shows the Microsoft SQL Server Management Studio results pane. It displays a table with three columns: Name, Brand, and (No column name). The data is as follows:

	Name	Brand	(No column name)
1	Daniel Craig	Alfa	2
2	Daniel Craig	Aston Martin	6
3	Daniel Craig	Audi	2
4	Daniel Craig	Ford	5
5	Daniel Craig	Jaguar	6
6	Daniel Craig	Land Rover	8
7	Daniel Craig	Mercedes Be...	2
8	Daniel Craig	Rolls-Royce	1
9	Daniel Craig	Volkswagen	1

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **11 ms**

- Microsoft SQL Server - wariant II

```

107  select Bond.Name, V.Brand, count(V.Brand)
108  from People Bond
109  join AsBondIn ABI on Bond.$node_id = ABI.$from_id
110  join Film F on F.$node_id = ABI.$to_id
111  join HasVehicle HV on F.$node_id = HV.$from_id
112  join Vehicle V on V.$node_id = HV.$to_id
113  group by Bond.Name, V.Brand
114  order by Bond.Name
115

```

Results Messages

	Name	Brand	(No column name)
1	Daniel Craig	Alfa	2
2	Daniel Craig	Aston Martin	6
3	Daniel Craig	Audi	2
4	Daniel Craig	Ford	5
5	Daniel Craig	Jaguar	6
6	Daniel Craig	Land Rover	8
7	Daniel Craig	Mercedes Benz	2
8	Daniel Craig	Rolls-Royce	1
9	Daniel Craig	Volkswagen	1

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **10.6 ms**

4.8 Podaj markę samochodów z największą liczbą różnych modeli ukazanych w filmach.

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, a query is entered:

```
neo4j$ MATCH(m:Film)-[:HAS_VEHICLE]→(v:Vehicle) RETURN v.Brand as Brand, count(distinct v.Model) as ModelCount ORDER BY ModelCount desc LIMIT 1
```

Below the query, the results are displayed in a table:

Brand	ModelCount
"Ford"	25

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **3.8 ms**

- Microsoft SQL Server - wariant I

```

118  select top 1 with ties Vehicle.Brand as Brand, count(distinct Vehicle.Model) as DistinctModelCount
119  from Vehicle, HasVehicle, Film
120  where match (Film - (HasVehicle) -> Vehicle)
121  group by Brand
122  order by DistinctModelCount desc
123

```

Results		Messages
	Brand	DistinctModelCount
1	Ford	25

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania **-9.2 ms**

- Microsoft SQL Server - wariant II

```

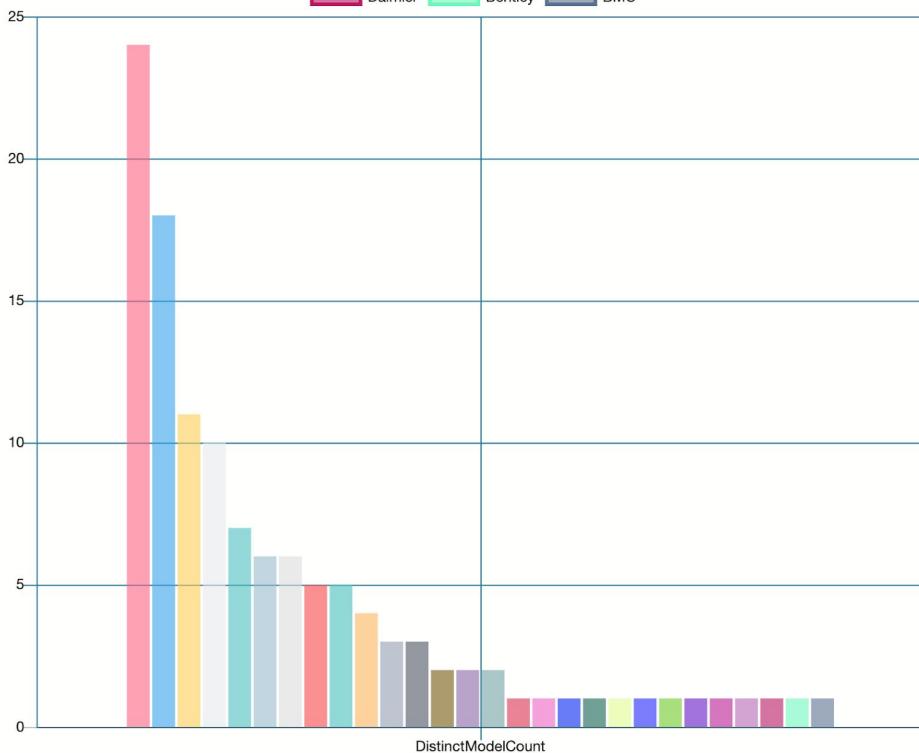
124  select top 1 with ties V.Brand as Brand, count(distinct V.Model) as DistinctModelCount
125  from Vehicle V
126  join HasVehicle HV on V.$node_id = HV.$to_id
127  join Film F on F.$node_id = HV.$from_id
128  group by Brand
129  order by DistinctModelCount desc

```

Results		Messages
	Brand	DistinctModelCount
1	Ford	25

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania **- 9.8 ms**

Liczba różnych modeli samochodów przypadających na jedną markę



4.9 Podaj nazwę modelu samochodu, który ukazał się w największej ilości produkcji.

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, a query is entered:

```
neo4j$ MATCH(m:Film)-[:HAS_VEHICLE]→(v:Vehicle) RETURN v.Model as Model, count(m.Name) as MoviesCount ORDER BY MoviesCount desc LIMIT 1
```

Below the query, the results are displayed in a table:

Model	MoviesCount
"DB5"	8

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **3.4 ms**

- Microsoft SQL Server - wariant I

```
132 select top 1 with ties Vehicle.Brand as Brand, Vehicle.Model as Model, count(Film.Name) as MoviesCount .  
133 from Vehicle, HasVehicle, Film  
134 where match (Film - (HasVehicle) -> Vehicle)  
135 group by Brand, Model  
136 order by MoviesCount desc  
137
```

Results			
Brand	Model	MoviesCount	
1	Aston Martin	DB5	8

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **7.9 ms**

- Microsoft SQL Server - wariant II

```
138 select top 1 with ties V.Brand as Brand, V.Model as Model, count(F.Name) as MoviesCount  
139 from Vehicle V  
140 join HasVehicle HV on V.$node_id = HV.$to_id  
141 join Film F on F.$node_id = HV.$from_id  
142 group by Brand, Model  
143 order by MoviesCount desc  
144
```

Results			
Brand	Model	MoviesCount	
1	Aston Martin	DB5	8

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **9.3 ms**

4.10 Pokaż reżyserów realizujących kilka filmów z serii wraz z ich liczbą

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, there is a code editor window containing a Cypher query:

```
neo4j$ MATCH (p:People)-[:DIRECTOR_OF]→(m:Film) WITH p, count(m.Name) as FilmsCount WHERE FilmsCount > 1 RETURN p.Name as Director, FilmsCount ORDER BY FilmsCount desc
```

Below the code editor is a results table. The table has two columns: "Director" and "FilmsCount". The data is as follows:

Director	FilmsCount
"John Glen"	5
"Guy Hamilton"	4
"Terence Young"	3
"Lewis Gilbert"	3
"Martin Campbell"	2
"Sam Mendes"	2

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **4.2 ms**

- Microsoft SQL Server - wariant I

```
146  select Director.Name as Director, count(Film.Name) as MoviesCount
147  from People Director, DirectorOf, Film
148  where match (Director - (DirectorOf) -> Film)
149  group by Director.Name
150  having count(Film.Name)>1
151  order by MoviesCount desc
152
```

The screenshot shows the SSMS Results pane. It displays the output of the SQL query. The results are presented in a table with two columns: "Director" and "MoviesCount". The data is as follows:

	Director	MoviesCount
1	John Glen	5
2	Guy Hamilton	4
3	Terence Young	3
4	Lewis Gilbert	3
5	Martin Campb...	2
6	Sam Mendes	2

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **11.2 ms**

- Microsoft SQL Server - wariant II

```

153  select Director.Name as Director, count(F.Name) as MoviesCount
154  from People Director
155  join DirectorOf D0 on Director.$node_id = D0.$from_id
156  join Film F on F.$node_id = D0.$to_id
157  group by Director.Name
158  having count(F.Name)>1
159  order by MoviesCount desc

```

[Results](#) [Messages](#)

	Director	MoviesCount
1	John Glen	5
2	Guy Hamilton	4
3	Terence Young	3
4	Lewis Gilbert	3
5	Martin Campbell	2
6	Sam Mendes	2

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **11.8 ms**

Liczba filmów przypadająca na jednego reżysera (powyżej 1)



4.11 Dla filmu o największym box-office pokaż jego tytuł, rok premiery, reżysera, Bonda

- Neo4J

The screenshot shows the Neo4j browser interface. In the top query editor, the following Cypher query is written:

```
neo4j$ MATCH (a:People)-[:DIRECTOR_OF]→
  (m:Film)←[:AS_BOND_IN]-(b:People)
  RETURN a.Name as Director, b.Name as
  Bond, m.Name as Title, m.Year as
  ProductionYear ORDER BY m.Box desc
  LIMIT 1
```

Below the query editor, the results table displays one row of data:

	Director	Bond	Title	ProductionYear
1	"Sam Mendes"	"Daniel Craig"	"Skyfall"	2012

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **3.9 ms**

- Microsoft SQL Server - wariant 1

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The query window contains the following T-SQL code:

```
162 select top 1 Director.Name as Director, Bond.Name as Bond, Film.Name as Title, Film.Year as ProductionYear -
163 from People Director, People Bond, AsBondIn, DirectorOf, Film
164 where match (Director - (DirectorOf) -> Film <- (AsBondIn) - Bond)
165 order by Convert(int, Film.Box) desc
166
```

Below the query window, the results pane shows the output:

	Director	Bond	Title	ProductionYear
1	Sam Mendes	Daniel Craig	Skyfall	2012

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **10.0 ms**

- Microsoft SQL Server - wariant 2

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The query window contains the following T-SQL code:

```
167 select top 1 Director.Name as Director, Bond.Name as Bond, F.Name as Title, F.Year as ProductionYear
168 from People Director
169 join DirectorOf DO on Director.$node_id = DO.$from_id
170 join Film F on F.$node_id = DO.$to_id
171 join AsBondIn ABI on ABI.$to_id = F.$node_id
172 join People Bond on Bond.$node_id = ABI.$from_id
173 order by Convert(int, F.Box) desc
```

Below the query window, the results pane shows the output:

	Director	Bond	Title	ProductionYear
1	Sam Mendes	Daniel Craig	Skyfall	2012

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania -**10.3 ms**

4.12 Przy realizacji filmu, przez którego reżysera pojawiło się najwięcej aut.
Podaj tytuł filmu oraz liczbę modeli

- Neo4J

The screenshot shows the Neo4j browser interface. In the top query editor, the following Cypher query is written:

```
neo4j$ MATCH (p:People)-[:DIRECTOR_OF]→(m:Film)-[:HAS_VEHICLE]→(v:Vehicle)  
RETURN p.Name as Director, m.Name as Title, count(v.Model) as ModelCount  
ORDER BY ModelCount desc LIMIT 1
```

Below the query editor, the results table is displayed:

Director	Title	ModelCount
"Marc Forster"	"Quantum of Solace"	11

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **3.4 ms**

- Microsoft SQL Server - wariant I

```
201 select top 1 with ties Director.Name as Director, Film.Name as Title, count(Vehicle.Model) as CarModelCount  
202 from People Director, DirectorOf, Film, Vehicle, HasVehicle  
203 where match (Director -> DirectorOf) -> Film -> Vehicle  
204 group by Director.Name, Film.Name  
205 order by CarModelCount desc  
206
```

Results		
Director	Title	CarMode...
1 Marc Forster	Quantum of Solace	11
2 John Glen	The Living Daylights	11

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **12.3 ms**

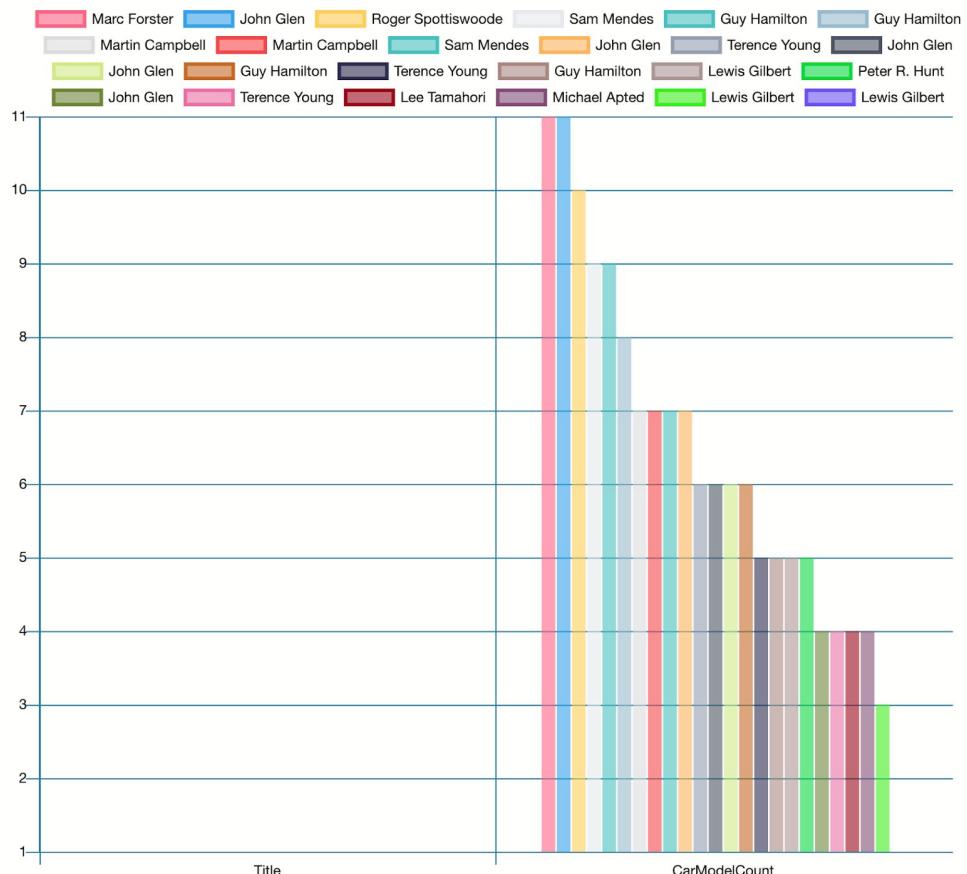
- Microsoft SQL Server - wariant II

```
207 select top 1 with ties Director.Name as Director, F.Name as Title, count(V.Model) as CarModelCount  
208 from People Director  
209 join DirectorOf DO on Director.$node_id = DO.$from_id  
210 join Film F on F.$node_id = DO.$to_id  
211 join HasVehicle HV on F.$node_id = HV.$from_id  
212 join Vehicle V on V.$node_id = HV.$to_id  
213 group by Director.Name, F.Name  
214 order by CarModelCount desc
```

Results		
Director	Title	CarModelCount
1 Marc Forster	Quantum of Solace	11
2 John Glen	The Living Daylights	11

Na podstawie 10 pomiarów uzyskano średni czas wykonania zapytania - **12.8 ms**

Liczba wykorzystanych samochodów przy realizacji filmu przez danego reżysera



4.13 Ile pojawiło się samochodów w filmie z największym box-office

- Neo4J

```
neo4j$ MATCH (m:Film)-[:HAS_VEHICLE]→
  (v:Vehicle) RETURN m.Name, m.Box,
  count(v.Brand) as VehicleCount ORDER
  BY m.Box desc LIMIT 1
```

m.Name	m.Box	VehicleCount
"Skyfall"	1108561008	7

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wynosił: **5.6 ms**

- Microsoft SQL Server - wariant I

```

217  select top 1 Film.Name as Title, Film.Box as BoxOffice, count(Vehicle.Model) as CarModelCount
218  from Film, Vehicle, HasVehicle
219  where match (Film - (HasVehicle) -> Vehicle)
220  group by Film.Name, Film.Box
221  order by Convert(int, Film.Box) desc
222

```

Results Messages

Title	BoxOffice	CarModelCount
Skyfall	1108561008	7

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **12.8 ms**

- Microsoft SQL Server - wariant II

```

223  select top 1 F.Name as Title, F.Box as BoxOffice, count(V.Model) as CarModelCount
224  from Film F
225  join HasVehicle HV on F.$node_id = HV.$from_id
226  join Vehicle V on V.$node_id = HV.$to_id
227  group by F.Name, F.Box
228  order by Convert(int, F.Box) desc
229

```

Results Messages

Title	BoxOffice	CarModelCount
Skyfall	1108561008	7

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **13.8 ms**

4.14 W czasie pracy, którego reżysera w sumie pojawiło się najwięcej aut

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, there is a code editor window containing a Cypher query:

```

neo4j$ MATCH (p:People)-[:DIRECTOR_OF]→
  (m:Film)-[:HAS_VEHICLE]→(v:Vehicle)
  RETURN p.Name as Director,
  count(v.Model) as ModelCount ORDER BY
  ModelCount desc LIMIT 1

```

Below the code editor is a results table. The table has two columns: "Director" and "ModelCount". There is one row of data:

Director	ModelCount
"John Glen"	34

On the left side of the interface, there are two buttons: "Table" and "Text".

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **3.2 ms**

- Microsoft SQL Server - wariant I

```
231  select top 1 with ties Director.Name as Director, count(Vehicle.Model) as CarModelCount
232  from People Director, DirectorOf, Film, Vehicle, HasVehicle
233  where match (Director - (DirectorOf) -> Film - (HasVehicle) -> Vehicle)
234  group by Director.Name
235  order by CarModelCount desc
236
```

Results Messages

Director	CarModelCount
1 John Glen	34

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **14.3 ms**

- Microsoft SQL Server - wariant II

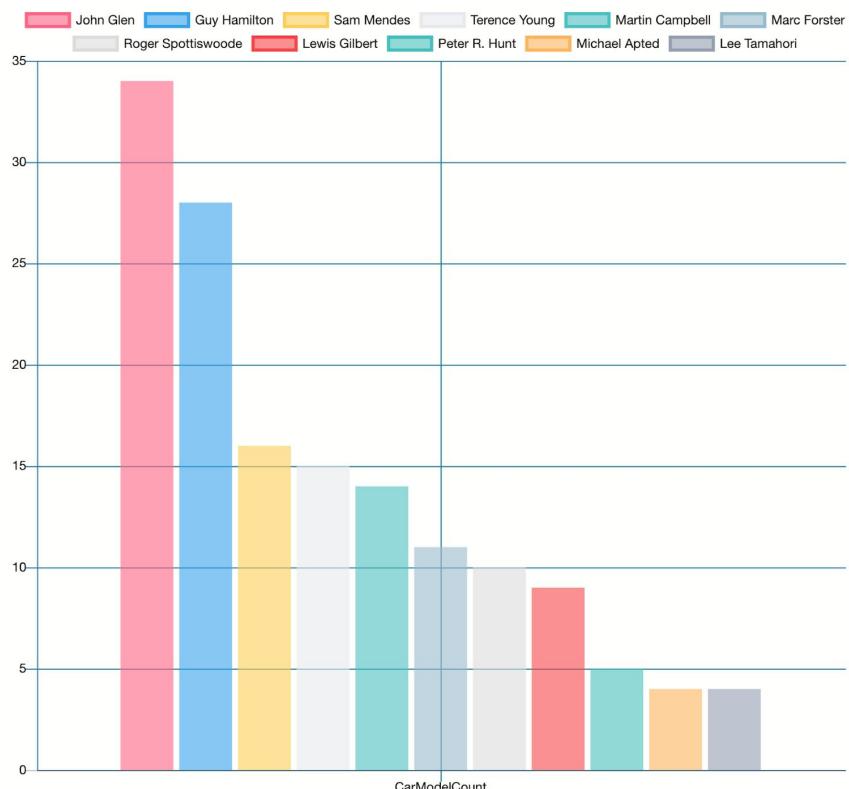
```
237  select top 1 with ties Director.Name as Director, count(V.Model) as CarModelCount
238  from People Director
239  join DirectorOf DO on Director.$node_id = DO.$from_id
240  join Film F on F.$node_id = DO.$to_id
241  join HasVehicle HV on F.$node_id = HV.$from_id
242  join Vehicle V on V.$node_id = HV.$to_id
243  group by Director.Name
244  order by CarModelCount desc
245
```

Results Messages

Director	CarModelCount
1 John Glen	34

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **12.9 ms**

Łączna liczba samochodów, wykorzystanych w filmach realizowanych przez danego reżysera



4.15 Podaj Bonda mającego kontakt, tylko z jednym reżyserem.

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, there is a code editor window containing a Cypher query:

```
neo4j$ MATCH (a:People)-[:AS_BOND_IN]→  
  (m:Film)←[:DIRECTOR_OF]-(b:People)  
  WITH a, count(distinct b.Name) AS  
  DirectorCount WHERE DirectorCount = 1  
  RETURN a.Name, DirectorCount
```

Below the code editor is a results table. The table has two columns: "a.Name" and "DirectorCount". There is one row of data:

a.Name	DirectorCount
"George Lazenby"	1

On the left side of the interface, there is a sidebar with three tabs: "Table" (selected), "Text", and "Code".

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **4.5 ms**

- Microsoft SQL Server - wariant I

```
264 select Bond.Name as Bond, Director.Name as Director  
265 from People Bond, People Director, AsBondIn, DirectorOf, Film  
266 where match (Director - (DirectorOf) -> Film <- (AsBondIn) -> Bond) and Bond.Name in  
267 (  
268   select Bond.Name as Bond  
269   from People Bond, People Director, AsBondIn, DirectorOf, Film  
270   where match (Director - (DirectorOf) -> Film <- (AsBondIn) -> Bond)  
271   group by Bond.Name  
272   having count(Director.Name)=1  
273 )
```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. At the top, there is a code editor window containing the provided T-SQL query. Below the code editor is a results grid. The grid has two columns: "Bond" and "Director". There is one row of data:

Bond	Director
George Lazenby	Peter R. Hunt

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **13.6 ms**

- Microsoft SQL Server - wariant II

```

276  select Bond.Name as Bond, Director.Name as Director
277  from People Director
278  join DirectorOf D0 on Director.$node_id = D0.$from_id
279  join Film F on F.$node_id = D0.$to_id
280  join AsBondIn ABI on ABI.$to_id = F.$node_id
281  join People Bond on Bond.$node_id = ABI.$from_id
282  where Bond.Name in
283  (
284      select Bond.Name as Bond
285      from People Director
286      join DirectorOf D0 on Director.$node_id = D0.$from_id
287      join Film F on F.$node_id = D0.$to_id
288      join AsBondIn ABI on ABI.$to_id = F.$node_id
289      join People Bond on Bond.$node_id = ABI.$from_id
290      group by Bond.Name
291      having count(Director.Name)=1
292  )

```

Results		Messages
	Bond	Director
1	George Lazenby	Peter R. Hunt

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **14.5 ms**

4.16 Podaj łączny box - office całej serii filmów o Jamesie Bondzie

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, a query is entered in the command line:

```
neo4j$ MATCH (m:Film) RETURN sum(m.Box) as BoxSum
```

Below the command line, the results are displayed in a table:

BoxSum
14699091551

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **2.8 ms**

- Microsoft SQL Server

```

1  select SUM(CAST(Box as BIGINT))
2  from Film
3

```

Results		Messages
	(No column name)	
1	14699091551	

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **6.7 ms**

4.17 Podaj średni box - office całej serii

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, there is a code input field containing:

```
neo4j$ MATCH (m:Film) RETURN avg(m.Box) as BoxAvg
```

Below the input field are several icons: a star, a clipboard, and a right-pointing arrow. The main area displays the results of the query:

```
neo4j$ MATCH (m:Film) RETURN avg(m.Box) a... ↓ ⚡ ↵ ⌂ ×
```

A sidebar on the left has two tabs: "Table" (selected) and "Text". The "Table" tab shows a single row with the header "BoxAvg" and the value "612462147.9583335".

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **2.3 ms**

- Microsoft SQL Server

```
4 select AVG(CAST(Box as BIGINT))  
5 from Film
```

The screenshot shows the Microsoft SQL Server Management Studio interface. It features two tabs at the top: "Results" (selected) and "Messages".

The results pane displays a table with one row:

	(No column name)
1	612462147

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **6.5 ms**

4.18 Podaj markę i model wszystkich samochodów, w których nazwie wystąpiło słowo sedan lub coupe

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, there is a code editor containing the following Cypher query:

```
1 MATCH (v:Vehicle) WHERE v.Model CONTAINS
  'sedan' RETURN v.Brand, v.Model
2 UNION
3 MATCH (v:Vehicle) WHERE v.Model CONTAINS
  'coupe' RETURN v.Brand, v.Model
```

Below the code editor is a results table. The table has two columns: "v.Brand" and "v.Model". The data is as follows:

v.Brand	v.Model
"AMC"	"AMC Matador sedan"
"Ford"	"Galaxie 500sedan"
"Chevrolet"	"Chevrolet Impalasedan"
"Chevrolet"	"Vauxhall PA Crestasedan"
"AMC"	"AMC Matador coupe"

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **4.5 ms**

- Microsoft SQL Server

```
7 select Brand, Model
8 from Vehicle
9 where Model like '%sedan%' OR Model like '%coupe%'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. At the top, there is a results pane titled "Results". Below it is a table with two columns: "Brand" and "Model". The data is as follows:

Brand	Model
AMC	AMC Matador coupe
AMC	AMC Matador sedan
Ford	Galaxie 500sedan
Chevrolet	Chevrolet Impalasedan
Chevrolet	Vauxhall PA Crestasedan

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **6.0 ms**

4.19 Podaj tytuły wszystkich filmów zaczynających się od słowa "The"

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, a query is entered in the command line:

```
neo4j$ MATCH (m:Film) WHERE m.Name STARTS WITH 'The' RETURN m.Name
```

Below the command line, the results are displayed in a table. The table has one column labeled "m.Name". The results are:

m.Name
"The Man with the Golden Gun"
"The Spy Who Loved Me"
"The Living Daylights"
"The World Is Not Enough"

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **2.4 ms**

- Microsoft SQL Server

```
11  select Name as Title
12  from Film
13  where Name like 'The%'
```

The screenshot shows the Microsoft SQL Server Management Studio interface. At the top, a query is entered in the query editor:

```
11  select Name as Title
12  from Film
13  where Name like 'The%'
```

Below the query editor, the results are displayed in a table. The table has one column labeled "Title". The results are:

Title
1 The Man with the Golden G...
2 The Spy Who Loved Me
3 The Living Daylights
4 The World Is Not Enough

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **5.7 ms**

4.20 Wypisz wszystkie osoby, które w nazwie posiadają "ros"

- Neo4J

The screenshot shows the Neo4j browser interface. At the top, there is a code editor containing the following Cypher query:

```
1 MATCH (p:People) WHERE p.Name CONTAINS  
'Ros' RETURN p.Name  
2 UNION  
3 MATCH (p:People) WHERE p.Name CONTAINS  
'ros' RETURN p.Name
```

Below the code editor is a results table with one column labeled "p.Name". The table contains two rows of data:

p.Name
"Rosamund Pike"
"Pierce Brosnan"

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **3.9 ms**

- Microsoft SQL Server

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, there is a code editor with the following T-SQL query:

```
15 select Name  
16 from People  
17 where Name like '%ros%'  
18
```

Below the code editor are two tabs: "Results" and "Messages". The "Results" tab is selected, showing a table with one column labeled "Name". The table contains two rows of data:

Name
Pierce Brosnan
Rosamund Pike

Po wykonaniu 10 pomiarów średni czas wykonania zapytania wyniósł: **5.5 ms**

3. Porównanie / wnioski

Powyższe przykłady jasno pokazały, że w przypadku danych o budowie grafowej baza Neo4J sprawia się znacznie lepiej niż relacyjna baza danych zamodelowana z wykorzystaniem Microsoft SQL Server. Przeciętny czas wykonania się zapytania obejmującego relacje między różnymi węzłami jest od 3-4 razy krótszy przy wykorzystaniu predefiniowanej do tego celu bazy Neo4J. Jednakże, wyniki zapytań dotyczących sumy, średniej, bądź zwykłego filtrowania informacji z jednej tabeli też uzyskano szybciej korzystając z Neo4J. Różnica ta była jednak zdecydowanie mniejsza.

Sama baza Microsoft SQL Server dostarcza nam również różnych narzędzi do radzenia sobie z problemem, jakim jest grafowy charakter danych. Tradycyjnie możemy uzyskać szukane informacje przez wykorzystanie klauzuli JOIN albo podzapytań. Jednak od wersji Microsoft SQL Server 2017 dostępna jest nowa komenda MATCH znacznie ułatwiająca dopasowanie dwóch węzłów połączonych ze sobą pewną relacją.

Nie tylko skraca to i ułatwia zapis, ale jednocześnie w większości przypadków minimalnie zmniejsza czas wykonania się zapytania.

Z kolei jeżeli chodzi o modelowanie grafu w bazie to schemat postępowania w obydwu bazach jest dość podobny. Wszystko sprowadza się do stworzenia wierzchołków i krawędzi, czyli relacji łączących poszczególne węzły. Niewątpliwie schematy otrzymane po utworzeniu takich baz są o wiele bardziej czytelne w przypadku wykorzystania technologii Neo4J.

Oczywiście bazy relacyjne mają w kilku miejscach przewagę. Przede wszystkim są dużo bardziej popularne, dlatego w razie wystąpienia jakiegoś problemu, pytań znacznie łatwiej uzyskać informację i poradzić sobie z przeszkodami. Bardzo możliwe, że w przypadku przerobienia grafu na typową relacyjną postać, uzyskane wyniki wyglądałyby całkiem inaczej. Celem tej pracy było jednak porównanie działania grafu zamodelowanego w Neo4J, z oferowanym od 2017 roku sposobem przechowywanie danych w formie grafu w relacyjnej bazie.

4. Aplikacja webowa

Aby użytkownicy mogli korzystać ze stworzonej bazy danych zbudowana została prosta aplikacja webowa używająca bazy JamesBondMovies w Neo4j. Wykorzystując framework Express oraz Neo4j Javascript Driver, udało się stworzyć aplikację widoczną poniżej.

The screenshot shows a web application interface for "James Bond Movies". At the top, there's a navigation bar with the title "007 James Bond Movies", a search input field containing "Skyfall", and a "Search" button. Below the navigation is a table titled "Search Results" with columns: Movie, Released, and Box office. The table lists several James Bond movies with their release years and box office earnings. To the right of the table is a large, dark image of a James Bond wristwatch with the "007" logo and the word "OCEAN" visible on its dial. A "Details" button is positioned above the watch image.

Movie	Released	Box office
Dr. No	1962	440759072
From Russia with Love	1963	576277964
Goldfinger	1964	912257512
Thunderball	1965	1014941117
You Only Live Twice	1967	756544419
On Her Majesty's Secret Service	1969	505899782
Diamonds Are Forever	1971	648514469
Live and Let Die	1973	825110761
The Man with the Golden Gun	1974	448249281
The Spy Who Loved Me	1977	692713752
Moonraker	1979	655872400
For Your Eyes Only	1981	486468881
Ostatni szansa	1983	102011352

Po wpisaniu tytułu interesującego nas filmu, otrzymujemy:

This screenshot shows the same application interface after searching for "Skyfall". The search results table now only contains one row for "Skyfall" (2012). To the right of the table is a detailed view for "Skyfall". It includes a list of facts about the movie, such as it being a 2012 spy movie and Daniel Craig starring as James Bond. Below this, under "Bond girls:", there's a list of actors: Bérénice Marlohe as Sévérine and Tonia Sotiropoulou as Bond's Lover. Under "Bond cars:", there's a list of vehicles: Mercedes Benz S400, Land Rover Land Rover Discovery 4, Land Rover Range Rover (L322), Land Rover Land Rover Defender, Jaguar XJ, Audi Audi A5, and Aston Martin DBS. A "See all movies" button is located at the bottom right of the detailed view area.

007™ James Bond Movies

Skyfall | Search

Search Results		
Movie	Released	Box office
Moonraker	1979	655872400

Details

- Moonraker is a 1979 spy movie, which grossed over \$655872400 worldwide.
- Roger Moore starring as James Bond.
- **Bond girls:**
 - Lois Chiles as Holly Goodhead
 - Emily Bolton as Manuela
 - Corinne Cléry as Corinne Dufour
- Directed by Lewis Gilbert.
- **Bond cars:**
 - Lafer MP Lafer Cabriolet
 - Rolls-Royce Silver Shadow
 - Chevrolet Corvette Veraneioambulance
 - AMC Jeep Wagoneer
 - AMC AMC Concord

[See all movies](#)



Możemy także powrócić do listy wszystkich filmów klikając przycisk "See all movies", co spowoduje powrót do takiego wyglądu strony:

007™ James Bond Movies

Skyfall | Search

Search Results		
Movie	Released	Box office
Dr. No	1962	440759072
From Russia with Love	1963	576277964
Goldfinger	1964	912257512
Thunderball	1965	1014941117
You Only Live Twice	1967	756544419
On Her Majesty's Secret Service	1969	505899782
Diamonds Are Forever	1971	648514469
Live and Let Die	1973	825110761
The Man with the Golden Gun	1974	448249281
The Spy Who Loved Me	1977	692713752
Moonraker	1979	655872400
For Your Eyes Only	1981	486468881
Ogoniok	1983	426244359

Details

