

# Benchmark baz danych

Aleks Zieliński, Filip Kalinowski

June 19, 2025

# Rozdziały

<b>1</b>	<b>Informacje wstępne</b>	<b>3</b>
1.1	Założenia projektu . . . . .	3
1.2	Użyte technologie i sprzęt . . . . .	3
1.3	Schemat logiczny bazy danych . . . . .	4
1.4	Przykładowe dane z plików JSON . . . . .	4
1.5	Baza danych i dane w PostgreSQL . . . . .	5
<b>2</b>	<b>Instalacja baz danych</b>	<b>6</b>
2.1	PostgreSQL . . . . .	6
2.2	MongoDB . . . . .	7
<b>3</b>	<b>Listy zapytań</b>	<b>10</b>
3.1	Oryginalne pomysły zapytań . . . . .	10
3.2	Finalne zapytania . . . . .	10
<b>4</b>	<b>Zapytania w MongoDB</b>	<b>11</b>
4.1	Zapytanie 1 . . . . .	11
4.2	Zapytanie 2 . . . . .	12
4.3	Zapytanie 3 . . . . .	13
4.4	Zapytanie 4 . . . . .	14
4.5	Zapytanie 5 . . . . .	16
4.6	Zapytanie 6 . . . . .	18
4.7	Zapytanie 7 . . . . .	19
4.8	Zapytanie 8 . . . . .	20
<b>5</b>	<b>Zapytania w PostgreSQL</b>	<b>21</b>
5.1	Zapytanie 1 . . . . .	21
5.2	Zapytanie 2 . . . . .	21
5.3	Zapytanie 3 . . . . .	22
5.4	Zapytanie 4 . . . . .	22
5.5	Zapytanie 5 . . . . .	23
5.6	Zapytanie 6 . . . . .	23
5.7	Zapytanie 7 . . . . .	24
5.8	Zapytanie 8 . . . . .	24
<b>6</b>	<b>Porównanie czasowe zapytań</b>	<b>25</b>
<b>7</b>	<b>Uwagi końcowe</b>	<b>25</b>

# 1 Informacje wstępne

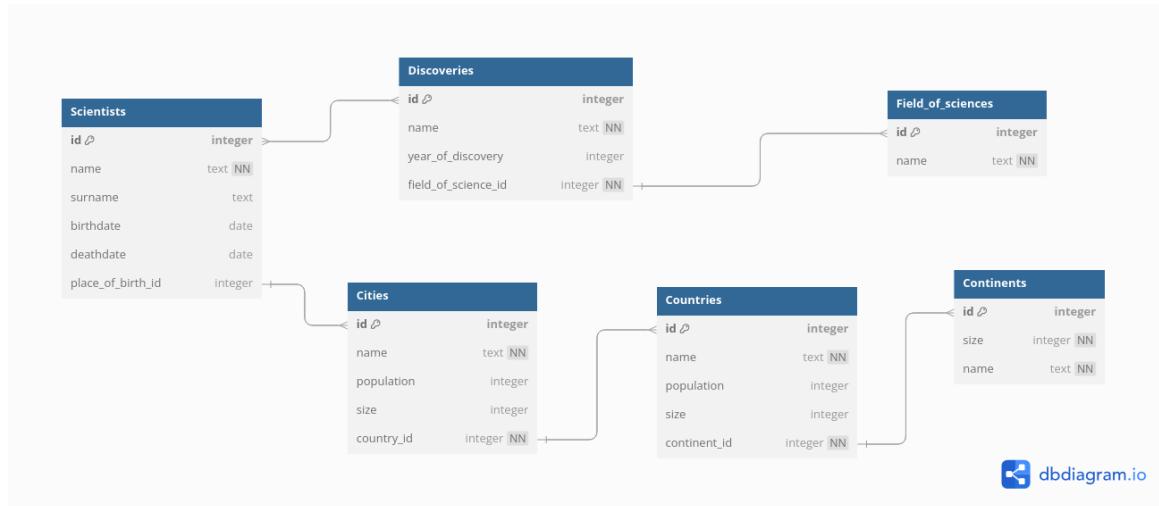
## 1.1 Założenia projektu

1. Zaprojektować schemat dokumentów JSON dla wybranego tematu. Należy mieć min 2 typy dokumentów na osobę w projekcie. Minimum jeden typ musi mieć dokumenty zagnieżdżone
2. Opracowany model, wraz z propozycją najważniejszych/najbardziej rozbudowanych poleceń należy zatwierdzić u wykładowcy (ew. poprawić wg. sugestii)
3. Dla każdego typu dokumentu wygenerować min 100 sensownych instancji. Jeden z typów powinien mieć minimum 500 instancji. Dane można generować automatami typu generatedata.com bądź napisać 4 skrypt w pythonie ( dodatkowe punkty).
4. Tak przygotowane dokumenty należy zainportować do:
  - Wybranej bazy NoSQL
  - PostgreSQLa do kolumn typu JSONB
  - PostgreSQLa przy jednoczesnej konwersji JSON na tabele
5. Wszystkie 3 procesy importowania należy udokumentować screenshotami i zapisanymi wykorzystanymi poleceniami
6. Należy utworzyć min 4 zapytania na osobę realizujące najważniejsze problemy wyszukiwania w tworzonej bazie.
7. Każde z zapytań musi mieć 3 wersje dla 3 sytuacji z pkt 4.
8. Należy przeprowadzić eksperyment pomiaru czasu wykonania tych zapytań. Aby eksperyment był rzetelnie przeprowadzony, zarówno baza NoSQLowa, jak i PostgreSQL muszą operować w podobnych warunkach: albo obie bazy są postawione w osobnych dockerach, albo zainstalowane w tym samym OS, ale uruchamiane jedna na raz (proces instalacji musi być udokumentowany). Wyniki czasowe należy przedstawić w tabelce

## 1.2 Użyte technologie i sprzęt

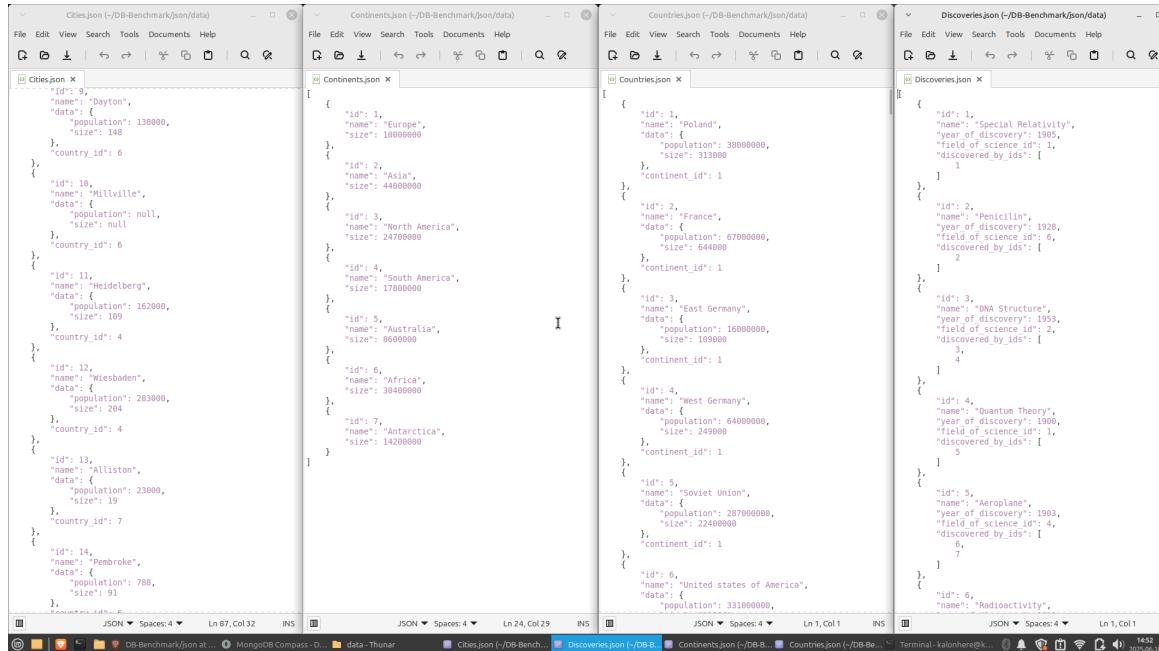
- PostgreSQL - wersja 16.9
- MongoDB - wersja 8.0.10
- Laptop:
  - OS - Linux Mint 22.1
  - CPU - Intel i7-9750H
  - GPU - Nvidia GTX 1660
  - RAM - 16GB 2667 MHz

### 1.3 Schemat logiczny bazy danych



Zdjęcie. 1: Schemat logiczny zrobiony w serwisie dbdiagram.io

### 1.4 Przykładowe dane z plików JSON



Zdjęcie. 2: Pliki JSON od lewej: Cities, Continents, Countries, Discoveries

```

Field_of_sciences.json
[{"id":1,"name":"Physics"}, {"id":2,"name":"Biology"}, {"id":3,"name":"Chemistry"}, {"id":4,"name":"Technology"}, {"id":5,"name":"Medicine"}, {"id":6,"name":"Mathematics"}]

Scientists.json
[{"id": 1, "full_name": {"name": "Albert", "surname": "Einstein"}, "dates": {"birthdate": "1879-03-14", "deathdate": "1955-04-18"}, "place_of_birth_id": 1}, {"id": 2, "full_name": {"name": "Alexander", "surname": "Fleming"}, "dates": {"birthdate": "1881-08-06", "deathdate": "1955-03-11"}, "place_of_birth_id": 2}, {"id": 3, "full_name": {"name": "Francis", "surname": "Crick"}, "dates": {"birthdate": "1916-06-08", "deathdate": "2004-07-28"}, "place_of_birth_id": 3}, {"id": 4, "full_name": {"name": "James", "surname": "Watson"}, "dates": {"birthdate": "1928-04-06", "deathdate": "2024-12-01"}, "place_of_birth_id": 4}, {"id": 5, "full_name": {"name": "John", "surname": "Goodenough"}, "dates": {"birthdate": "1922-07-24", "deathdate": "2023-08-15"}, "place_of_birth_id": 5}]

```

Zdjęcie. 3: Pliki JSON od lewej: Field\_of\_sciences, Scientists

## 1.5 Baza danych i dane w PostgreSQL

```

Scientists.sql
CREATE TABLE Scientists (
    "id" bigint,
    "full_name.name" text,
    "full_name.surname" text,
    "dates.birthdate" date,
    "dates.deathdate" date,
    "place_of_birth_id" bigint
);

INSERT INTO Scientists
("id","full_name.name","full_name.surname","dates.birthdate","
dates.deathdate","place_of_birth_id")
VALUES
(1,'Albert','Einstein','1879-03-14','1955-04-18',1),
(2,'Alexander','Fleming','1881-08-06','1955-03-11',2),
(3,'Francis','Crick','1916-06-08','2004-07-28',3);

Discoveries.sql
CREATE TABLE discoveries (
    "id" bigint,
    "name" text,
    "year_of_discovery" bigint,
    "field_of_science_id" bigint,
    "discovered_by_ids" json
);

INSERT INTO discoveries
("id","name","year_of_discovery","field_of_science_id","discovered_by_
ids")
VALUES
(1,'Special Relativity',1905,1,['1']),
(2,'Penicillin',1928,6,['2']),
(3,'DNA Structure',1953,2,['3,4']),
(4,'Quantum Theory',1900,1,['5']);

Field_of_sciences.sql
CREATE TABLE Field_of_sciences (
    "id" bigint,
    "name" text
);

INSERT INTO Field_of_sciences
("id","name")
VALUES
(1,'Physics'),
(2,'Biology'),
(3,'Chemistry'),
(4,'Technology'),
(5,'Medicine'),
(6,'Mathematics');

Continents.sql
CREATE TABLE Continents (
    "id" bigint,
    "name" text,
    "size" bigint
);

INSERT INTO Continents
("id","name","size")
VALUES
(1,'Europe',10000000),
(2,'Asia',44000000),
(3,'North America',24700000),
(4,'South America',17800000),
(5,'Australia',8600000),
(6,'Africa',38400000),
(7,'Antarctica',14200000);

Discovered_by.sql
CREATE TABLE discovered_by (
    "discovery_id" bigint,
    "scientist_id" bigint
);
-- Klucz glowny, para (discovery_id, scientist_id), nigdy nie bedzie dwóch takich samych
INSERT INTO discovered_by
("discovery_id","scientist_id")
VALUES
(1,1),
(2,2),
(3,3),
(3,4),
(4,5),
(5,6),
(5,7),
(6,8),
(6,9),
(7,10);

Cities.sql
CREATE TABLE Cities (
    "id" bigint,
    "name" text,
    "data.population" bigint NULL,
    "data.size" bigint NULL,
    "country_id" bigint
);

INSERT INTO Cities
("id","name","data.population","data.size",
"country_id")
VALUES
(1,'Ulm',129000,118,4),
(2,'Lochfield',34700,14,8),
(3,'Northampton',243500,61,8),
(4,'Chicago',2746400,606,6),
(5,'Kiel',249000,155,4),
(6,'Warsaw',1862000,517,1),
(7,'Vienna',1973400,414,31),
(8,'Lichfield',326000,9,8)

Countries.sql
CREATE TABLE Countries (
    "id" bigint,
    "name" text,
    "data.population" bigint,
    "data.size" bigint,
    "continent_id" bigint
);

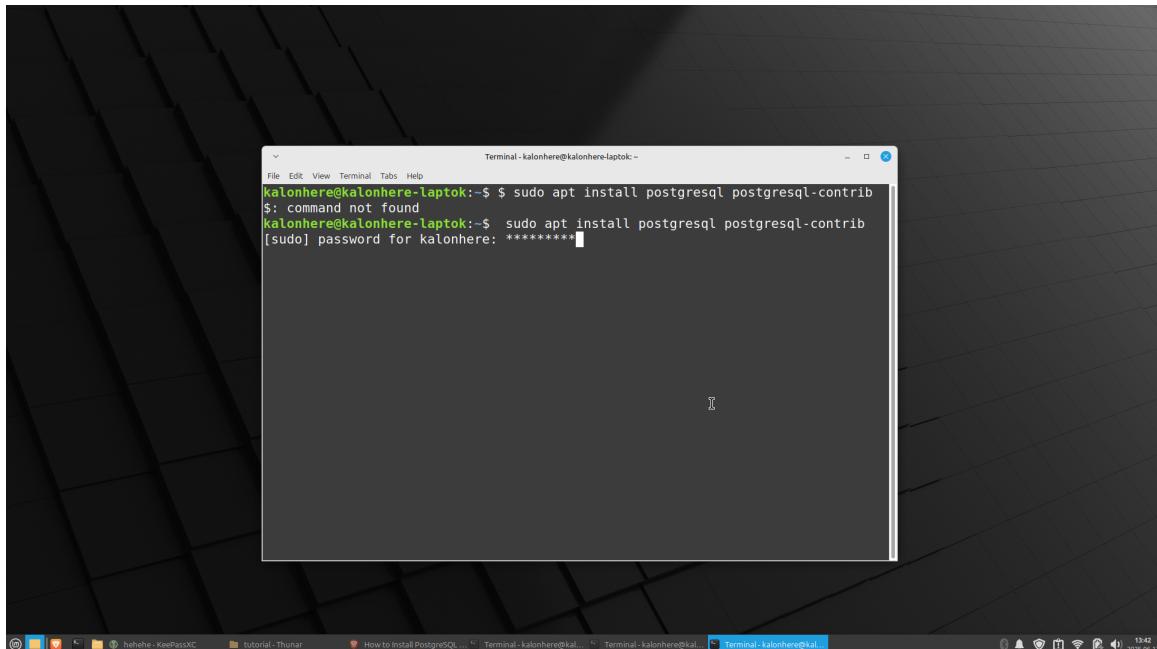
INSERT INTO Countries
("id","name","data.population","data.size","cont-
inent_id")
VALUES
(1,'Poland',38000000,313000,1),
(2,'France',67000000,644000,1),
(3,'East Germany',16000000,109000,1),
(4,'West Germany',64000000,249000,1),
(5,'Soviet Union',28700000,22400000,1),
(6,'United States of America',
33100000,980000,3),
(7,'Canada',37000000,10000000,3)

```

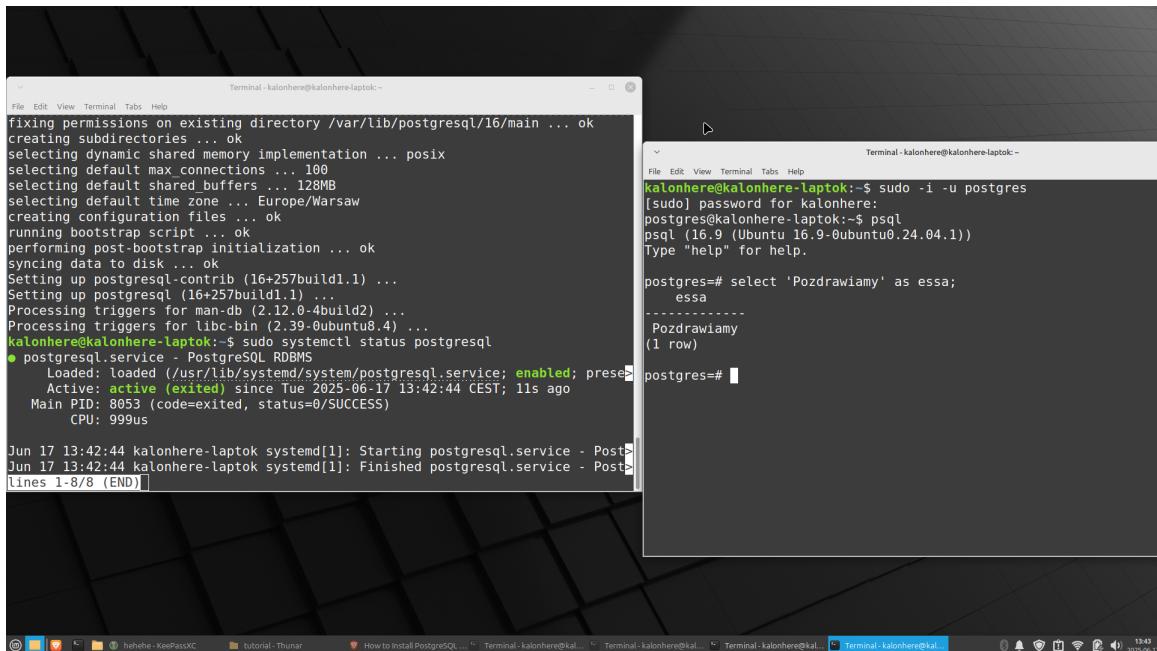
Zdjęcie. 4: Wszystkie JSONy po konwersji na sql

## 2 Instalacja baz danych

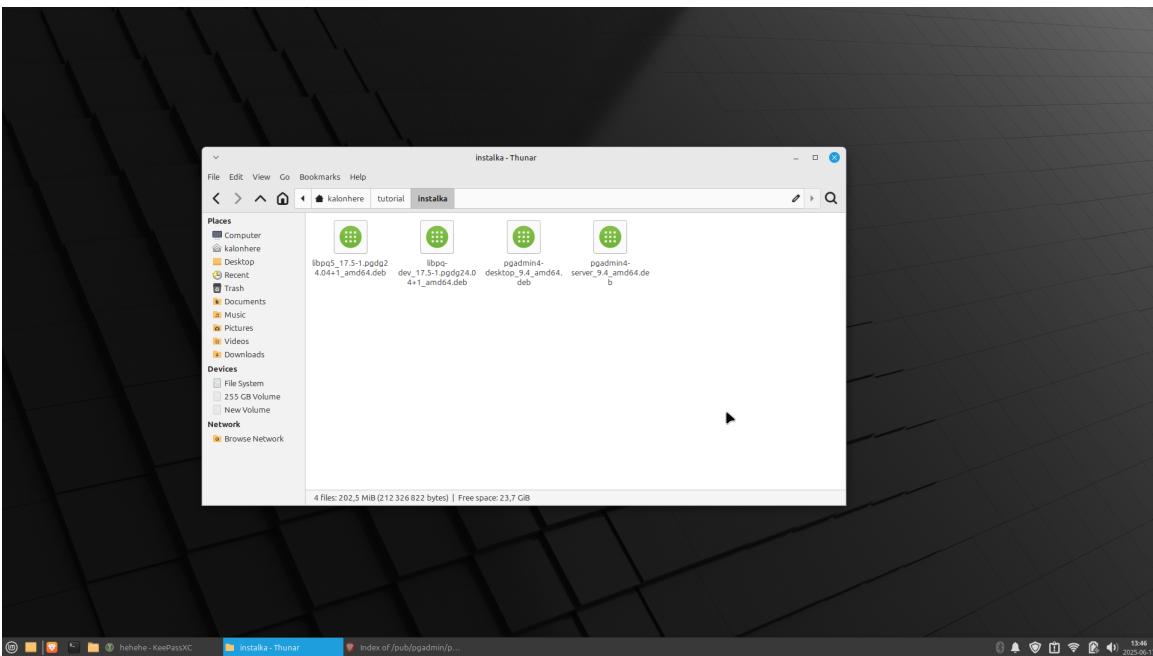
### 2.1 PostgreSQL



Zdjęcie. 5: Instalacja za pomocą apt

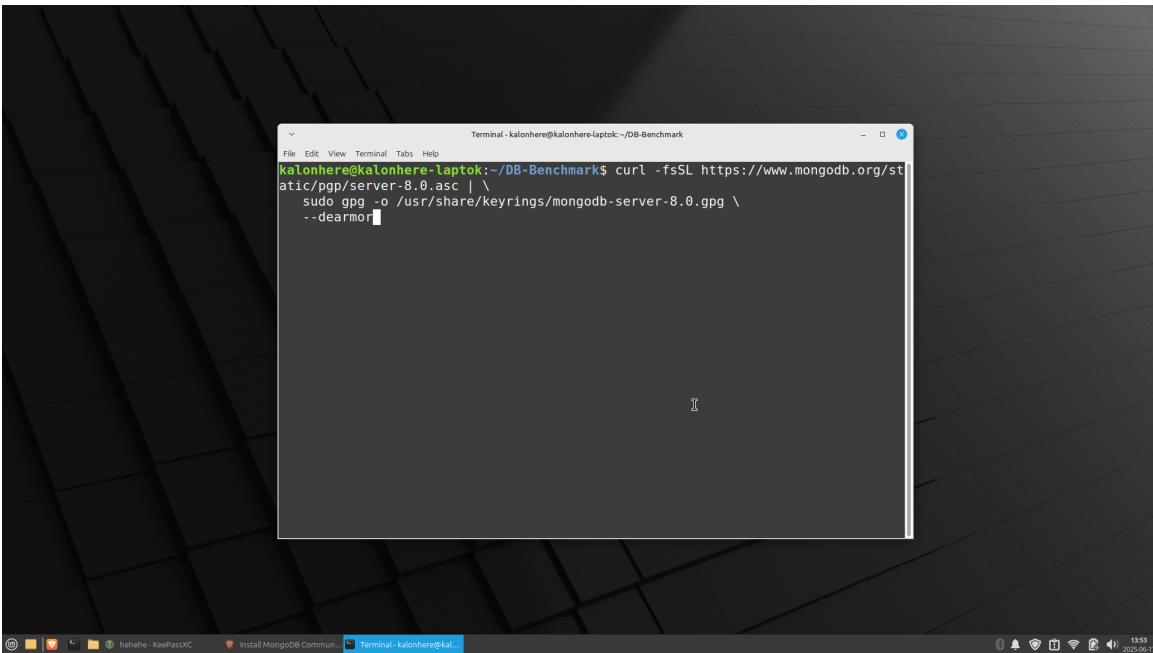


Zdjęcie. 6: Po lewej - komenda systemctl aby sprawdzić czy działa, po prawej - logowanie na user, uruchomienie psql i test działania

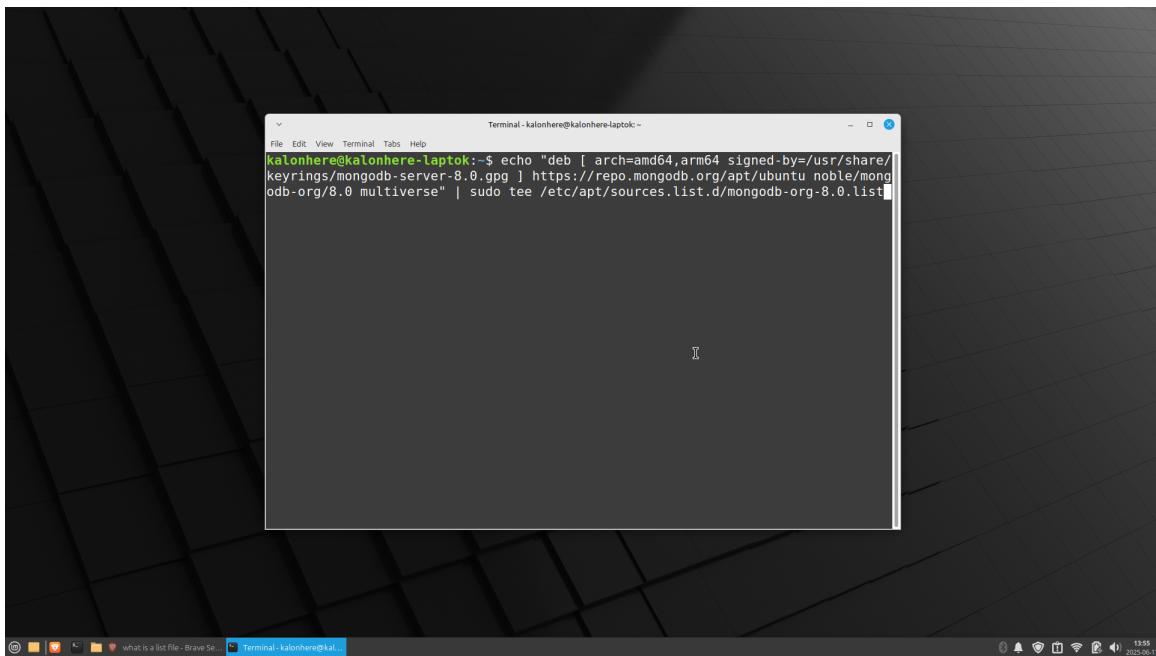


Zdjęcie. 7: Instalacja pgadmina z serwera ftp PostgreSQL

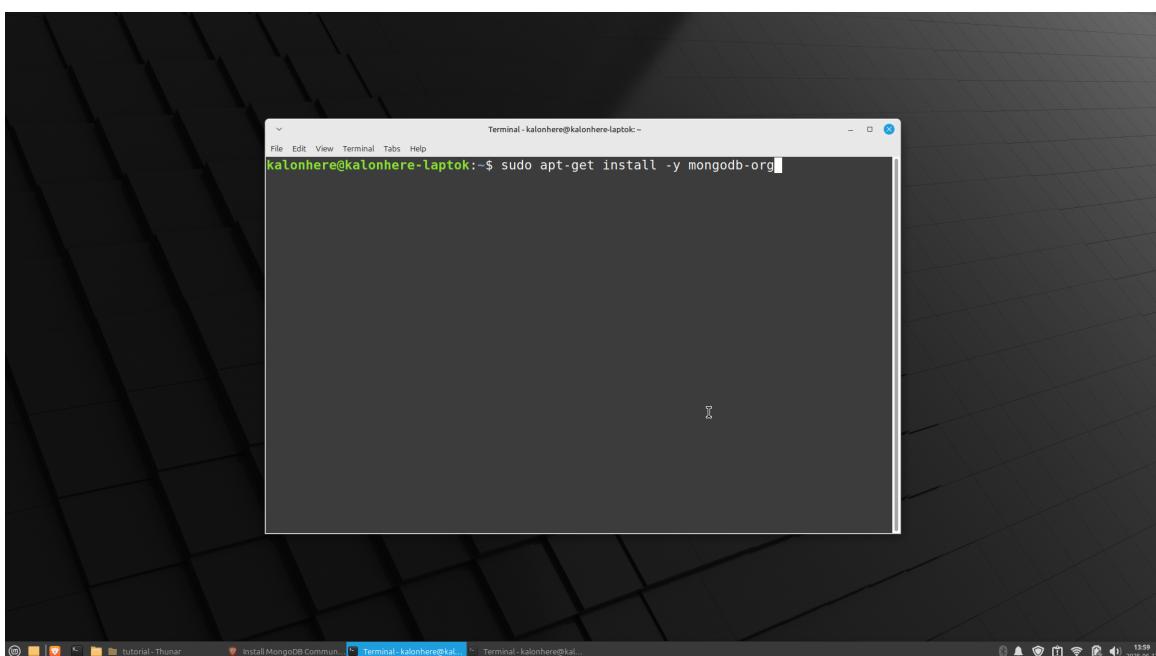
## 2.2 MongoDB



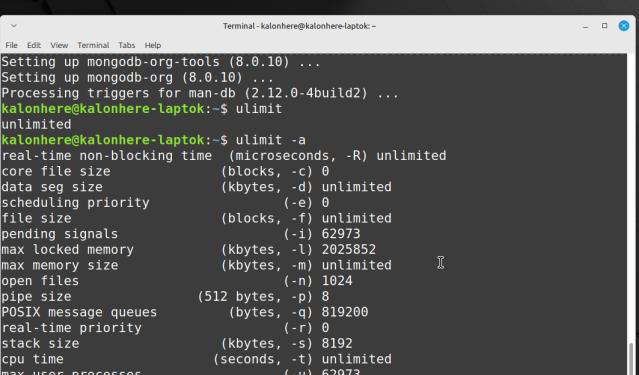
Zdjęcie. 8: Importowanie kluczy gpg do MongoDB



Zdjęcie. 9: Stworzenie pliku z listą źródeł do pobierania repo/package



Zdjęcie. 10: Instalacja MongoDB



```
Terminal - kalonhere@kalonhere-laptok:~  
File Edit View Terminal Tabs Help  
Setting up mongodb-org-tools (8.0.10) ...  
Setting up mongodb-org (8.0.10) ...  
Processing triggers for man-db (2.12.0-4build2) ...  
kalonhere@kalonhere-laptok:~$ ulimit  
unlimited  
real-time non-blocking time (microseconds, -R) unlimited  
core file size (blocks, -c) 0  
data seg size (kbytes, -d) unlimited  
scheduling priority (-e) 0  
file size (blocks, -f) unlimited  
pending signals (-i) 62973  
max locked memory (kbytes, -l) 2025852  
max memory size (kbytes, -m) unlimited  
open files (-n) 1024  
pipe size (512 bytes, -p) 8  
POSIX message queues (bytes, -q) 819200  
real-time priority (-r) 0  
stack size (kbytes, -s) 8192  
cpu time (seconds, -t) unlimited  
max user processes (-u) 62973  
virtual memory (kbytes, -v) unlimited  
file locks (-x) unlimited  
kalonhere@kalonhere-laptok:~$ ulimit -n 64000
```

**Zdjęcie. 11:** Zwiększenie limitu otwartych plików z 1024 do 64000

**Zdjęcie. 12:** Uruchomienie MongoDB, sprawdzenie statusu i włączenie autouruchamiania

### **3 Listy zapytań**

#### **3.1 Oryginalne pomysły zapytań**

1. Wybierz naukowców którzy pochodzą z azji i odkryli więcej niż 5 rzeczy z biologii lub chemii
2. Wybierz państwo które ma najwięcej odkryć z fizyki oraz nazwa państwa kończy się na "a"
3. Zlicz ile odkryć z matematyki ma każdy kontynent
4. Znajdź naukowców którzy nie żyją i mają wiecej odkryć niż 3 z wszystkiego oprócz fizyki, dodatkowo wypisz liczbę ich odkryć poza fizyką
5. Znajdź miasto których naukowcy mają najwięcej odkryć w chemii
6. Znajdź najstarsze i najmłodsze odkrycie z ZSRR z chemii
7. Znajdź miasto które nie ma żadnych odkryć w biologii po 1960 roku
8. Znajdź średnią liczbę wszystkich odkryć na rok w każdym państwie w europie
9. Wypisz dziedzine w której każde państwo z azji dokonało najwięcej odkryć
10. Wypisz średnią liczbę odkryć na naukowca z każdego kontynetu
11. Znajdź państwo z największą średnią liczbą odkryć na osobę
12. Znajdź miasto z którego pochodzący naukowcy dokonali najwięcej odkryć z fizyki

#### **3.2 Finalne zapytania**

1. Wybierz państwo którego nazwa kończy się na "a" i ma najwięcej odkryć z fizyki
2. Znajdź najstarsze i najmłodsze odkrycie z USA z biologii
3. Znajdź państwa które mają odkrycia z biologii po 1960 roku
4. Znajdź średnią liczbę wszystkich odkryć na państwo w każdym kontynencie
5. Wypisz dziedzine w której każde państwo z europy dokonało najwięcej odkryć
6. Wypisz średnią liczbę odkryć na naukowca z każdego kontynetu
7. Znajdź państwo z największą średnią liczbą odkryć na kilometr kwadratowy
8. Znajdź miasto z którego pochodzi najwięcej fizyków

## 4 Zapytania w MongoDB

### 4.1 Zapytanie 1

```
[{$match: {
    name: { $regex: "a$", $options: "i" }
},
{$lookup: {
    from: "Cities",
    localField: "id",
    foreignField: "country_id",
    as: "cities"
}
},
{ $unwind: "$cities" },
{$lookup: {
    from: "Scientists",
    localField: "cities.id",
    foreignField: "place_of_birth_id",
    as: "scientists"
}
},
{ $unwind: "$scientists" },
{$lookup: {
    from: "Discoveries",
    let: { scientistId: "$scientists.id" },
    pipeline: [
        {
            $match: {
                $expr: {
                    $and: [
                        { $eq: ["$field_of_science_id", 1] },
                        { $in: ["$$scientistId", "$discovered_by_ids"] }
                    ]
                }
            }
        ],
        as: "discoveries"
    }
},
{ $unwind: "$discoveries" },
{$group: {
    _id: { country_id: "$id", country: "$name", discovery_id: "$discoveries.id", discovery_name: "$discoveries.name" }
},
{$group: {
    _id: { country_id: "$_id.country_id", country: "$_id.country" },
    discoveries: { $addToSet: { id: "$_id.discovery_id", name: "$_id.discovery_name" } },
    count: { $sum: 1 }
}
},
{ $sort: { count: -1 } },
{ $limit: 1 },
{$project: {
    _id: 0,
    country: "$_id.country",
    count: 1,
    discoveries: 1
}
}]]
```

## 4.2 Zapytanie 2

```
[{$match: {
    field_of_science_id: 2
},
{$unwind: "$discovered_by_ids"
},
{$lookup: {
    from: "Scientists",
    localField: "discovered_by_ids",
    foreignField: "id",
    as: "scientist"
}
},
{$unwind: "$scientist" },
{$lookup: {
    from: "Cities",
    localField: "scientist.place_of_birth_id",
    foreignField: "id",
    as: "birth_city"
}
},
{$unwind: "$birth_city" },
{$match: {
    "birth_city.country_id": 6
}
},
{$group: {
    _id: {
        id: "$id",
        name: "$name",
        year_of_discovery: "$year_of_discovery"
    }
}
},
{$sort: { "_id.year_of_discovery": 1 } },
{$group: {
    _id: null,
    discoveries: { $push: "$_id" }
}
},
{$project: {
    _id: 0,
    oldest: { $arrayElemAt: ["$discoveries", 0] },
    youngest: { $arrayElemAt: ["$discoveries", -1] }
}
}]]
```

### 4.3 Zapytanie 3

```
[{$match: {
    field_of_science_id: 2,
    year_of_discovery: { $gt: 1960 }
},
{ $unwind: "$discovered_by_ids",
{$lookup: {
    from: "Scientists",
    localField: "discovered_by_ids",
    foreignField: "id",
    as: "scientist"
}
},
{ $unwind: "$scientist" },
{$lookup: {
    from: "Cities",
    localField: "scientist.place_of_birth_id",
    foreignField: "id",
    as: "city"
}
},
{ $unwind: "$city" },
{$group: {
    _id: {
        country_id: "$city.country_id",
        discovery_id: "$id",
        discovery_name: "$name"
    }
}
},
{$lookup: {
    from: "Countries",
    localField: "_id.country_id",
    foreignField: "id",
    as: "country"
}
},
{ $unwind: "$country" },
{$group: {
    _id: "$country.name",
    discoveries: { $addToSet: { id: "$_id.discovery_id", name: "$_id.discovery_name" } },
    count: { $sum: 1 }
}
},
{$sort: { count: -1 } },
{$project: {
    _id: 0,
    country: "$_id",
    count: 1,
    discoveries: 1
}
}]]
```

#### 4.4 Zapytanie 4

```
[{$lookup: {
  from: "Scientists",
  localField: "discovered_by_ids",
  foreignField: "id",
  as: "scientist"
}
},
{ $unwind: "$scientist" },
{$lookup: {
  from: "Cities",
  localField: "scientist.place_of_birth_id",
  foreignField: "id",
  as: "city"
}
},
{ $unwind: "$city" },
{$lookup: {
  from: "Countries",
  localField: "city.country_id",
  foreignField: "id",
  as: "country"
}
},
{ $unwind: "$country" },
{$lookup: {
  from: "Continents",
  localField: "country.continent_id",
  foreignField: "id",
  as: "continent"
}
},
{ $unwind: "$continent" },
{$group: {
  _id: {
    continent_id: "$continent.id",
    continent_name: "$continent.name",
    country_id: "$country.id",
    country_name: "$country.name",
    discovery_id: "$id"
  }
}
},
{$group: {
  _id: {
    continent_id: "$_id.continent_id",
    continent_name: "$_id.continent_name",
    country_id: "$_id.country_id",
    country_name: "$_id.country_name" },
    discoveries_count: { $sum: 1 }
  }
}]
```

```
},
{$group: {
  _id: {
    continent_id: "$_id.continent_id",
    continent_name: "$_id.continent_name"},
    average_discoveries_per_country: { $avg: "$discoveries_count" },
    countries: {
      $push: {
        country_id: "$_id.country_id",
        country_name: "$_id.country_name",
        discoveries_count: "$discoveries_count"
      }
    }
  }
},
{
  $project: {
    _id: 0,
    continent_id: "$_id.continent_id",
    continent_name: "$_id.continent_name",
    average_discoveries_per_country: { $round: ["$average_discoveries_per_country", 2] },
  }
},
{$sort: { continent_name: 1 } }
]
```

## 4.5 Zapytanie 5

```
[{$unwind: "$discovered_by_ids"
},
{$lookup: {
    from: "Scientists",
    localField: "discovered_by_ids",
    foreignField: "id",
    as: "scientist"
}
},
{ $unwind: "$scientist" },
{$lookup: {
    from: "Cities",
    localField: "scientist.place_of_birth_id",
    foreignField: "id",
    as: "city"
}
},
{ $unwind: "$city" },
{$lookup: {
    from: "Countries",
    localField: "city.country_id",
    foreignField: "id",
    as: "country"
}
},
{ $unwind: "$country" },
{$match: {
    "country.continent_id": 1
}
},
{$group: {
    _id: {
        country_id: "$country.id",
        country_name: "$country.name",
        field_of_science_id: "$field_of_science_id"
    },
    discoveries_count: { $sum: 1 }
}
},
{$sort: {
    "_id.country_name": 1,
    discoveries_count: -1
}
},
{$group: {
    _id: "$_id.country_name",
    field_of_science_id: { $first: "$_id.field_of_science_id" },
    discoveries_count: { $first: "$discoveries_count" }
}
},
```

```
    {$lookup: {
        from: "Field_of_sciences",
        localField: "field_of_science_id",
        foreignField: "id",
        as: "science"
    },
    { $unwind: "$science" },
    {$project: {
        _id: 0,
        country: "$_id",
        field_of_science: "$science.name",
        discoveries_count: 1
    }
},
{$sort: { country: 1 } }
```

## 4.6 Zapytanie 6

```
[{$lookup: {
    from: "Cities",
    localField: "place_of_birth_id",
    foreignField: "id",
    as: "city"
},
{$unwind: "$city" },
{$lookup: {
    from: "Countries",
    localField: "city.country_id",
    foreignField: "id",
    as: "country"
},
{$unwind: "$country" },
{$lookup: {
    from: "Continents",
    localField: "country.continent_id",
    foreignField: "id",
    as: "continent"
},
{$unwind: "$continent" },
{$lookup: {
    from: "Discoveries",
    let: { scientistId: "$id" },
    pipeline: [
        {
            $match: {
                $expr: { $in: ["$$scientistId", "$discovered_by_ids"] }
            }
        ],
        as: "discoveries"
    }
},
{$addFields: {
    discoveries_count: { $size: "$discoveries" }
}},
{$group: {
    _id: "$continent.name",
    total_discoveries: { $sum: "$discoveries_count" },
    scientists_count: { $sum: 1 }
}},
{$project: {
    _id: 0,
    continent: "$_id",
    avg_discoveries_per_scientist: {
        $cond: [
            { $eq: ["$scientists_count", 0] },
            0,
            { $round: [{ $divide: ["$total_discoveries", "$scientists_count"] }, 2] }
        ]
    },
    total_discoveries: 1,
    scientists_count: 1
}},
{$sort: { continent: 1 } }
]]
```

## 4.7 Zapytanie 7

```
[{$unwind: "$discovered_by_ids"
},
{$lookup: {
    from: "Scientists",
    localField: "discovered_by_ids",
    foreignField: "id",
    as: "scientist"
}
},
{$unwind: "$scientist" },
{$lookup: {
    from: "Cities",
    localField: "scientist.place_of_birth_id",
    foreignField: "id",
    as: "city"
}
},
{$unwind: "$city" },
{$lookup: {
    from: "Countries",
    localField: "city.country_id",
    foreignField: "id",
    as: "country"
}
},
{$unwind: "$country" },
{$group: {
    _id: {
        country_id: "$country.id",
        country_name: "$country.name",
        discovery_id: "$id"
    },
    country_size: { $first: "$country.data.size" }
}
},
{$group: {
    _id: {
        country_id: "$_id.country_id",
        country_name: "$_id.country_name"
    },
    discoveries_count: { $sum: 1 },
    country_size: { $first: "$country_size" }
}
},
{$addFields: {
    avg_discoveries_per_km2: {
        $cond: [
            { $gt: ["$country_size", 0] },
            { $divide: ["$discoveries_count", "$country_size"] },
            null
        ]
    }
},
{$sort: { avg_discoveries_per_km2: -1 } },
{$limit: 1 },
{$project: {
    _id: 0,
    country: "$_id.country_name",
    discoveries_count: 1,
    size: "$country_size",
    avg_discoveries_per_km2: { $round: ["$avg_discoveries_per_km2", 6] }
}
}]]
```

## 4.8 Zapytanie 8

```
[{$lookup: {
    from: "Discoveries",
    let: { scientistId: "$id" },
    pipeline: [
        {
            $match: {
                field_of_science_id: 1,
                $expr: { $in: ["$$scientistId", "$discovered_by_ids"] }
            }
        }
    ],
    as: "physics_discoveries"
},
{$match: {
    "physics_discoveries.0": { $exists: true }
},
{$group: {
    _id: "$place_of_birth_id",
    physicists_count: { $sum: 1 }
},
{$lookup: {
    from: "Cities",
    localField: "_id",
    foreignField: "id",
    as: "city"
},
{$unwind: "$city" },
{$sort: { physicists_count: -1 } },
{$limit: 1 },
{$project: {
    _id: 0,
    city: "$city.name",
    country_id: "$city.country_id",
    physicists_count: 1
}
}]]
```

## 5 Zapytania w PostgreSQL

### 5.1 Zapytanie 1

```
select * from scientists
inner join discovered_by on scientists.id = discovered_by.scientist_id
inner join discoveries on discovered_by.discovery_id = discoveries.id
inner join field_of_sciences on discoveries.field_of_science_id = field_of_sciences.id
where place_of_birth_id in
(select id from cities where country_id in
(select id from countries where name like '%a'))
and field_of_sciences.name like 'Physics';
```

### 5.2 Zapytanie 2

```
select discoveries.name, discoveries.year_of_discovery
from scientists
inner join discovered_by on scientists.id = discovered_by.scientist_id
inner join discoveries on discovered_by.discovery_id = discoveries.id
inner join field_of_sciences on discoveries.field_of_science_id = field_of_sciences.id
inner join cities on scientists.place_of_birth_id = cities.id
inner join countries on cities.country_id = countries.id
where lower(countries.name) = 'united states of america'
and lower(field_of_sciences.name) = 'biology'
and discoveries.year_of_discovery =
(select min(discoveries.year_of_discovery)
from scientists
inner join discovered_by on scientists.id = discovered_by.scientist_id
inner join discoveries on discovered_by.discovery_id = discoveries.id
inner join field_of_sciences on discoveries.field_of_science_id = field_of_sciences.id
inner join cities on scientists.place_of_birth_id = cities.id
inner join countries on cities.country_id = countries.id
where lower(countries.name) = 'united states of america'
and lower(field_of_sciences.name) = 'biology'
)
union all
select discoveries.name, discoveries.year_of_discovery
from scientists
inner join discovered_by on scientists.id = discovered_by.scientist_id
inner join discoveries on discovered_by.discovery_id = discoveries.id
inner join field_of_sciences on discoveries.field_of_science_id = field_of_sciences.id
inner join cities on scientists.place_of_birth_id = cities.id
inner join countries on cities.country_id = countries.id
where lower(countries.name) = 'united states of america'
and lower(field_of_sciences.name) = 'biology'
and discoveries.year_of_discovery =
(select max(discoveries.year_of_discovery)
from scientists
inner join discovered_by on scientists.id = discovered_by.scientist_id
inner join discoveries on discovered_by.discovery_id = discoveries.id
inner join field_of_sciences on discoveries.field_of_science_id = field_of_sciences.id
inner join cities on scientists.place_of_birth_id = cities.id
inner join countries on cities.country_id = countries.id
where lower(countries.name) = 'united states of america'
and lower(field_of_sciences.name) = 'biology'
);
```

### 5.3 Zapytanie 3

```
select * from scientists
inner join discovered_by on scientists.id = discovered_by.scientist_id
inner join discoveries on discovered_by.discovery_id = discoveries.id
inner join field_of_sciences on discoveries.field_of_science_id = field_of_sciences.id
where place_of_birth_id in
(select id from cities where country_id in
(select id from countries))
and field_of_sciences.name like 'Biology'
and year_of_discovery > 1960;
```

### 5.4 Zapytanie 4

```
select
    continents.name as continent,
    avg(country_discoveries.discovery_count) as average_discoveries_per_country
from
(
    select
        countries.id as country_id,
        count(*) as discovery_count
    from
        scientists
        join countries on scientists.place_of_birth_id = countries.id
        join discovered_by on scientists.id = discovered_by.scientist_id
    group by
        countries.id
) as country_discoveries
join countries on country_discoveries.country_id = countries.id
join continents on countries.continent_id = continents.id
group by
continents.name;
```

## 5.5 Zapytanie 5

```
with country_discoveries as (
    select
        lower(countries.name) as country,
        lower(field_of_sciences.name) as field,
        count(distinct discoveries.id) as discovery_count
    from scientists
    inner join cities on scientists.place_of_birth_id = cities.id
    inner join countries on cities.country_id = countries.id
    inner join discovered_by on scientists.id = discovered_by.scientist_id
    inner join discoveries on discovered_by.discovery_id = discoveries.id
    inner join field_of_sciences on discoveries.field_of_science_id = field_of_sciences.id
    where countries.continent_id = 1
    group by countries.name, field_of_sciences.name
), max_discoveries as (
    select
        country,
        max(discovery_count) as max_count
    from country_discoveries
    group by country
)
select
    cd.country,
    cd.field,
    cd.discovery_count
from country_discoveries cd
inner join max_discoveries md
    on cd.country = md.country and cd.discovery_count = md.max_count
order by cd.country, cd.field;
```

## 5.6 Zapytanie 6

```
select
    lower(continents.name) as continent,
    round(
        count(distinct discovered_by.discovery_id)::numeric
        / count(distinct scientists.id), 6
    ) as average_discoveries_per_scientist
from scientists
inner join cities on scientists.place_of_birth_id = cities.id
inner join countries on cities.country_id = countries.id
inner join continents on countries.continent_id = continents.id
inner join discovered_by on scientists.id = discovered_by.scientist_id
group by continents.name
order by continent asc;
```

## 5.7 Zapytanie 7

```
select
    lower(c.name) as country,
    round(cd.discovery_count::numeric / c."data.size", 6) as discoveries_per_km2
from
    countries c
    join (
        select
            co.id as country_id,
            count(distinct db.discovery_id) as discovery_count
        from
            scientists s
            join cities ci on s.place_of_birth_id = ci.id
            join countries co on ci.country_id = co.id
            join discovered_by db on s.id = db.scientist_id
        group by
            co.id
    ) cd on c.id = cd.country_id
where
    c."data.size" is not null and c."data.size" > 0
order by
    discoveries_per_km2 desc
limit 1;
```

## 5.8 Zapytanie 8

```
select
    lower(cities.name) as city,
    count(distinct scientists.id) as physicist_count
from scientists
inner join cities on scientists.place_of_birth_id = cities.id
inner join discovered_by on scientists.id = discovered_by.scientist_id
inner join discoveries on discovered_by.discovery_id = discoveries.id
inner join field_of_sciences on discoveries.field_of_science_id = field_of_sciences.id
where lower(field_of_sciences.name) = 'physics'
group by cities.name
order by physicist_count desc
limit 1;
```

## 6 Porównanie czasowe zapytań

Po wykonaniu każdego zapytania odpowiednio, otrzymaliśmy następujące wyniki:

Zapytanie	MongoDB	PostgreSQL	PostgreSQL-JSONB
1	22ms	57ms	n/a
2	4ms	59ms	n/a
3	1ms	57ms	n/a
4	41ms	67ms	n/a
5	49ms	80ms	n/a
6	54ms	80ms	n/a
7	70ms	79ms	n/a
8	42ms	67ms	n/a

## 7 Uwagi końcowe

- Brak udokumentowania procesu instalacji MongoDB Compass - zapomnieliśmy
- Brak udokumentowania procesu importowania do MongoDB - jedno kliknięcie w MongoDB Compass
- Brak udokumentowania procesu importowania do PostgreSQL JSONB - proszę zrozumieć, Filip po tym będzie miał traumę
- Brak zrobienia testów z poleceń PostgreSQL JSONB - brak czasu i trauma Filipa po importowaniu (udało się zimporotować ale trauma wygrała i nie udało się zrobić poleceń)
- Brak udokumentowania procesu importowania do PostgreSQL - proste wklejenie sql z pliku do pgAdmin
- Słabe/Bezsensowne zapytania - nie dostaliśmy akcepta na czas :c
- Wiemy, że zrobiliśmy kiepskie JSONy w sensie do MongoDB ale chcieliśmy móc je prosto przekonwertować do PostgreSQL i nie mieliśmy nigdy doświadczenie w projektowaniu ich jako bazy danych
- Wszystkie JSONy pisaliśmy sami, zawierają odkrycia z 20 wieku, miasta i państwa posiadają najnowsze dane na temat populacji i wielkości, co do ich lokacji to korzystaliśmy z mapy z okresu zimnej wojny
- Cały projekt z plikami - <https://github.com/Aleks-Zielinski/DB-Benchmark>