Kolegium Nauk Przyrodniczych
Uniwersytet Rzeszowski

Przedmiot:

Inteligentne metody eksploracji danych

Projekt 1

Wykonał:

Faustyna Misiura 119041

Jakub Jakubowski 125125

Prowadzący: dr inż. Piotr Grochowalski

Rzeszów 2024

Cel projektu:

Projekt ma na celu porównanie wydajności zapytań w dwóch różnych typach baz danych: relacyjnej bazie danych PostgreSQL oraz nierelacyjnej bazie danych MongoDB. Celem jest analiza takich parametrów jak czas wykonania zapytań, zużycie pamięci RAM oraz wydajność procesora CPU dla zapytań wykonywanych na tych bazach. Wszystkie wyniki będą zbierane i przetwarzane przy pomocy skryptów w Pythonie, a ostateczne dane zostaną zapisane w pliku Excel i zaprezentowane w interfejsie użytkownika opartym na React i Flasku.

Opis architektury:

Projekt składa się z trzech głównych komponentów:

1. **Skrypty do wykonania zapytań do baz danych:**
    - **PostgreSQL**: Skrypty wykonują zapytania na bazie danych PostgreSQL, mierząc czas wykonania zapytania, zużycie pamięci RAM oraz wydajność CPU.
    - **MongoDB**: Skrypty wykonują zapytania do bazy MongoDB, wykonując podobne pomiary.
2. **Skrypty do analizy i zapisania wyników:**
    - Po wykonaniu zapytań, wyniki są zapisywane w pliku `result.txt`, a następnie przetwarzane przez skrypty, które wyodrębniają interesujące dane (czas wykonania zapytań, RAM, CPU) i zapisują je w formacie Excel (`database_performance_comparison.xlsx`).
3. **Interfejs użytkownika (UI):**
    - Interfejs użytkownika jest wykonany w **React** i umożliwia wizualizację wyników analiz. Serwer aplikacji jest oparty na **Flasku**, który dostarcza dane w formacie JSON do front-endu React.

Szczegóły implementacji:

1. Skrypty wykonujące zapytania:

Skrypty w Pythonie łączą się z bazą danych PostgreSQL i MongoDB, wykonują zapytania SQL i zapytania w MongoDB, a następnie mierzą czas wykonania zapytania, zużycie pamięci RAM i wydajność CPU. Skrypty korzystają z takich bibliotek jak:

- `psutil` – do monitorowania zużycia RAM i CPU.

- `psycopg2` – do łączenia się z bazą PostgreSQL.
- `pymongo` – do łączenia się z bazą MongoDB.
- `time` – do mierzenia czasu wykonania zapytań.

2. Skrypt do analizy wyników:

Po wykonaniu zapytań, wyniki są zapisywane w pliku `result.txt`. Skrypt analizuje ten plik i wyciąga istotne dane, takie jak:

- Czas wykonania zapytania (w sekundach).
- Średnie i maksymalne zużycie RAM (w MB).
- Średnia i maksymalna wydajność CPU (w procentach).

Dane są zapisywane w pliku Excel (`database_performance_comparison.xlsx`) przy użyciu biblioteki `pandas`.

3. Interfejs użytkownika:

React i **Flask** zostały użyte do stworzenia front-endu i back-endu aplikacji webowej:

- **React (front-end)**:
  - Aplikacja React umożliwia wyświetlanie wyników analizy w formie tabeli.
  - Użytkownicy mogą interaktywnie przeglądać wyniki dla każdej bazy danych (PostgreSQL i MongoDB).
  - Komponenty React łączą się z serwerem Flask, aby pobrać dane z pliku Excel i zaprezentować je w przeglądarkach.
- **Flask (back-end)**:
  - Flask zapewnia serwer, który obsługuje zapytania HTTP (GET) z front-endu i przekazuje dane (z pliku Excel) w formacie JSON.
  - Serwer Flask pobiera dane z pliku Excel i konwertuje je na format JSON, aby frontend mógł je łatwo wyświetlić.

Struktura aplikacji:

1. **Backend (Flask)**:
   - Endpoint `/api/results` – odpowiada za pobieranie wyników z pliku Excel i zwracanie ich w formacie JSON.
2. **Frontend (React)**:
   - Komponenty: `Table` (do wyświetlania wyników) i `App` (główna aplikacja, która komunikuje się z Flaskiem).
   - React fetchuje dane z Flaskowego serwera i renderuje je w tabeli.

4. Plik wynikowy:

Wszystkie wyniki są zapisywane w pliku Excel (`database_performance_comparison.xlsx`), który zawiera:

- Kolumny dla bazy danych (PostgreSQL, MongoDB).
- Kolumny z danymi dotyczącymi zapytań:
    - Czas wykonania zapytania.
    - Średnie i maksymalne zużycie RAM.
    - Średnia i maksymalna wydajność CPU.

Szczegóły implementacji skryptów

Szczegółowa implementacja skryptów znajduje sie w repozytorium zdalnym na github:

IMEDLaboratories/data_mining: inteligent data mining methods

Interpretacja wyników porównania baz danych PostgreSQL i MongoDB

W przedstawionych danych porównano wyniki zapytań dla dwóch baz danych: **PostgreSQL** i **MongoDB**. Oto szczegółowa analiza wyników i wnioski z eksperymentu.

1. Czas wykonania zapytań:

- **PostgreSQL**: Czas wykonania zapytań w PostgreSQL waha się od **0,0282 sekundy** (najmniejszy czas dla zapytania 2) do **2,8869 sekundy** (największy czas dla zapytania 10).
- **MongoDB**: Czas wykonania zapytań w MongoDB jest zdecydowanie krótszy, wynoszący od **0,0009 sekundy** do **0,6502 sekundy**, z typowym czasem wykonania poniżej 0,5 sekundy.

Wnioski: MongoDB wykazuje znacznie szybsze czasy wykonania zapytań w porównaniu do PostgreSQL. Czas wykonania zapytań w PostgreSQL bywa znacznie dłuższy, szczególnie dla zapytań wymagających większej obróbki danych (np. zapytania 10, 12 dla bazy CLINIC).

2. Zużycie pamięci RAM:

- **PostgreSQL**: Zużycie pamięci RAM w PostgreSQL waha się głównie w okolicach **45 MB**, z małymi odchyleniami (max. do **45,28 MB**). Zapytania PostgreSQL nie powodują większych wahań w zużyciu pamięci, co wskazuje na dość stabilne zarządzanie pamięcią.
- **MongoDB**: Zużycie pamięci RAM w MongoDB jest wyższe, wynoszące około **55 MB** dla większości zapytań. Zapytania w MongoDB generują wyższe zużycie pamięci (np. zapytania dla bazy **CLINIC MongoDB** zaczynają się od **54,56 MB** i rosną do **57 MB** w przypadku zapytań 1-12).

Wnioski: MongoDB ma wyższe zużycie pamięci RAM w porównaniu do PostgreSQL. Może to wynikać z innej struktury przechowywania danych w MongoDB, który może przechowywać dane w pamięci podręcznej w większym zakresie. Dla bazy MongoDB pamięć jest używana w sposób bardziej intensywny.

3. Wydajność CPU:

- **PostgreSQL**: Wydajność CPU w PostgreSQL waha się od **0,93%** (zapytanie 7) do **50%** (zapytanie 3). W przypadku zapytań intensywnie przetwarzających dane (np. zapytania 3, 6, 9) wydajność CPU jest stosunkowo wysoka.
- **MongoDB**: Wydajność CPU w MongoDB jest znacznie niższa, przy średnich wartościach bliskich **0%** (np. zapytanie 2 w MongoDB), ale w przypadkach zapytań obciążających system (np. zapytanie 32) wydajność CPU dochodzi do **32%**.

Wnioski: MongoDB wykazuje niższe zużycie CPU w większości zapytań. Jednak w przypadku zapytań, które są bardziej złożone (np. zapytanie 32 w MongoDB, zapytanie 3 w PostgreSQL), wykorzystanie CPU w MongoDB może wzrosnąć, ale ogólnie pozostaje ono na niższym poziomie niż w przypadku PostgreSQL. Może to sugerować, że MongoDB lepiej radzi sobie z obciążeniem obliczeniowym, szczególnie przy prostych zapytaniach.

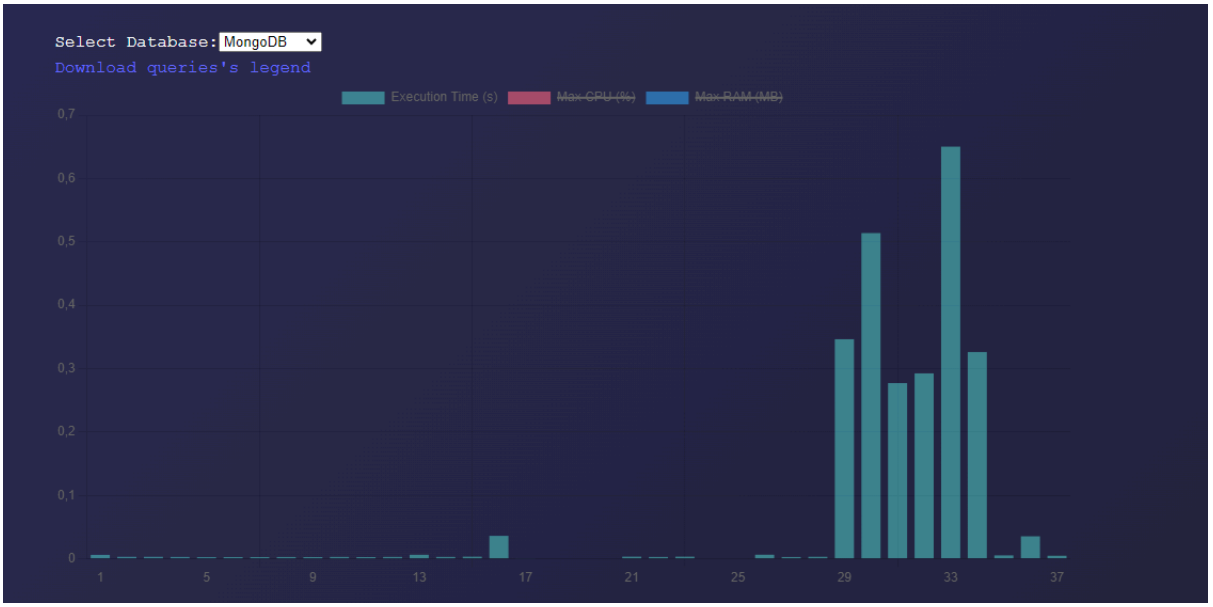
4. Porównanie na podstawie zapytań:

- **PostgreSQL**: Baza PostgreSQL, mimo dłuższego czasu wykonania zapytań, jest bardziej wydajna pod względem CPU w zapytaniach bardziej złożonych. Jest to typowe dla relacyjnych baz danych, które są zoptymalizowane do przetwarzania złożonych zapytań wymagających dużej obróbki danych.
- **MongoDB**: MongoDB wykazuje bardzo szybkie wykonanie zapytań, zwłaszcza przy prostych operacjach, jednak zużywa więcej pamięci RAM i ma niższą wydajność CPU przy prostych zapytaniach. Wydajność CPU wzrasta, gdy zapytania są bardziej złożone, ale nadal jest mniejsza niż w przypadku PostgreSQL.

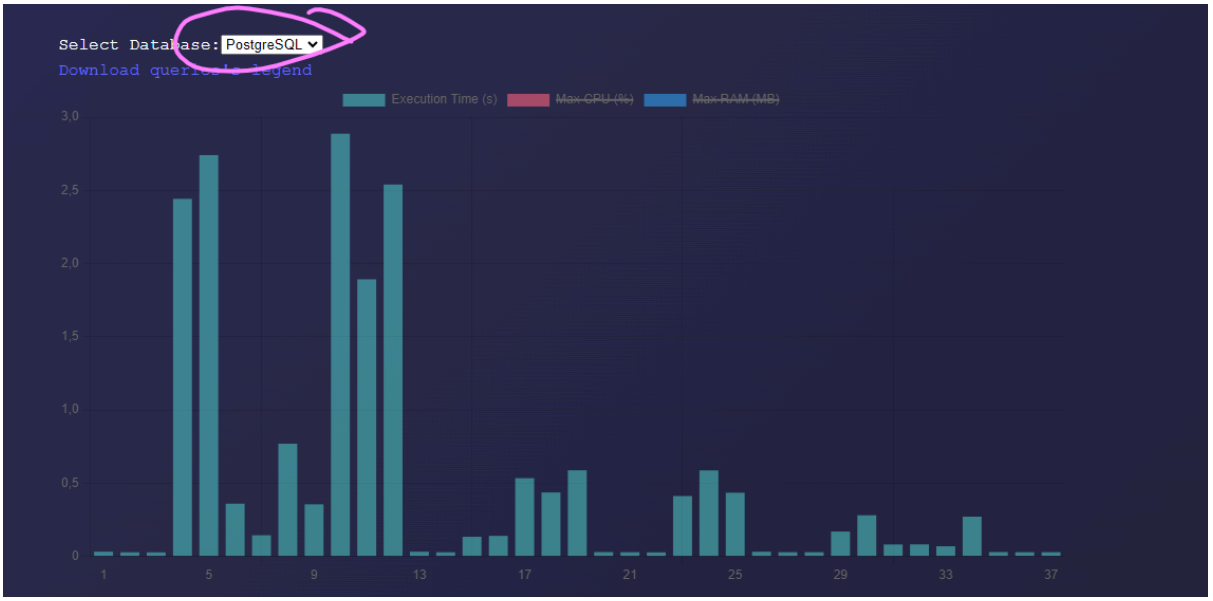
Podsumowanie wniosków:

1. **Szybkość zapytań**: MongoDB jest znacznie szybsze w wykonaniu zapytań, zwłaszcza przy prostych operacjach, co może być wynikiem braku potrzeby skomplikowanego przetwarzania danych.
2. **Zużycie pamięci RAM**: MongoDB zużywa więcej pamięci RAM niż PostgreSQL. Może to wynikać z różnych mechanizmów przechowywania danych i caching w MongoDB.
3. **Wydajność CPU**: PostgreSQL jest bardziej wymagający pod względem CPU, szczególnie przy bardziej złożonych zapytaniach, jednak MongoDB wydaje się być bardziej wydajna pod względem obciążenia CPU w przypadku prostych zapytań.
4. **Optymalizacja zapytań**: PostgreSQL lepiej radzi sobie z bardziej złożonymi zapytaniami, które wymagają intensywnego przetwarzania danych, podczas gdy MongoDB sprawdza się przy szybszych zapytaniach, które nie wymagają złożonej obróbki danych.

Na podstawie tych wyników można stwierdzić, że **MongoDB** będzie bardziej efektywne w przypadku zapytań o niskiej złożoności, gdzie ważna jest szybkość wykonania, podczas gdy **PostgreSQL** jest lepszym wyborem dla bardziej złożonych zapytań, które wymagają intensywnego przetwarzania danych, powyzesz wnioski zostały udokumentowane interraktywnie także w GUI:

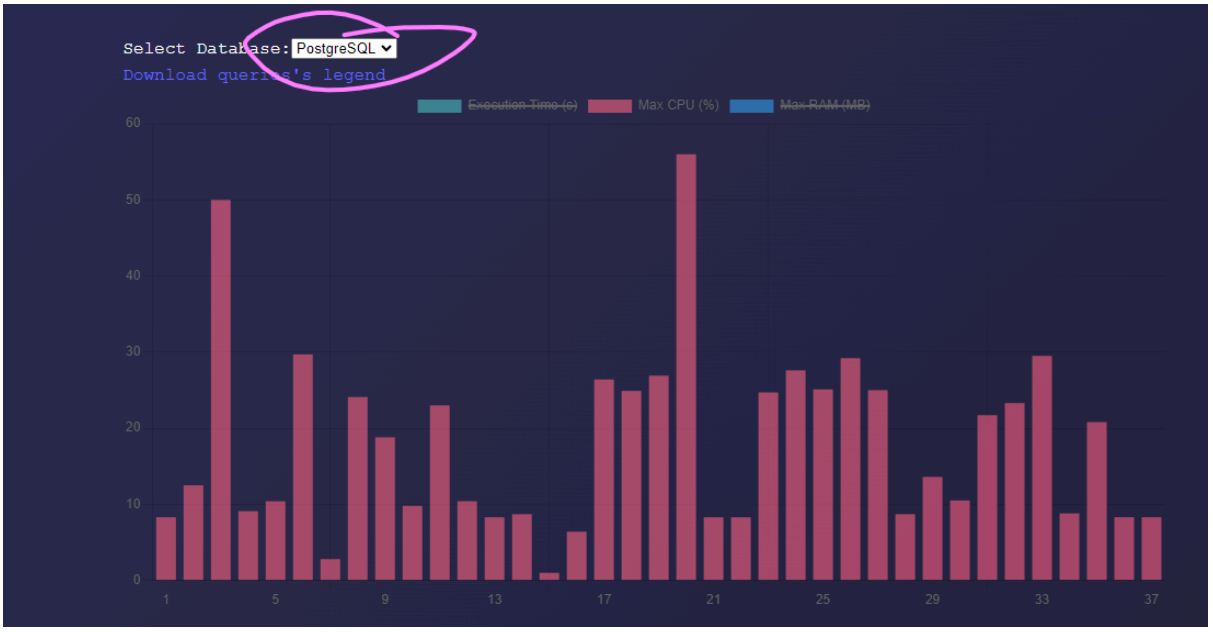


Rys.1 Wyniki dla MongoDB czasu trwania zapytania

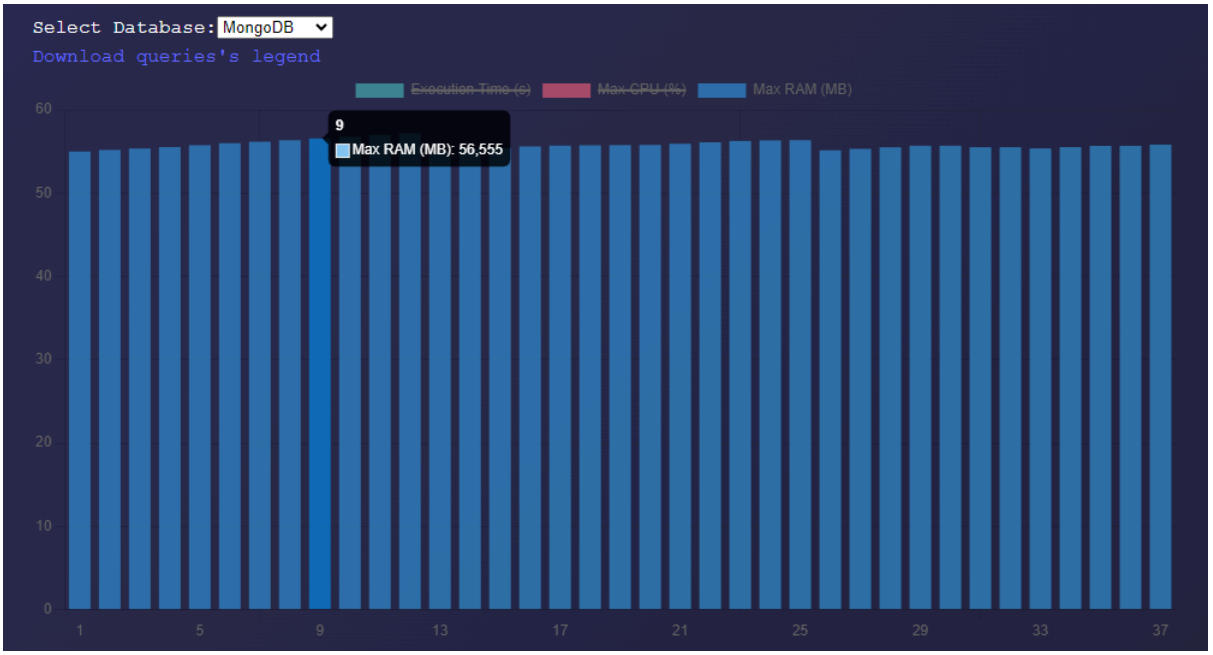


Rys.2 Wyniki czasu trwania zapytania w zależności od złożoności tego zapytania dla PostgreSQL

Rys.3 Wyniki dla zajętości CPU(mierzone w %) dla MongoDb

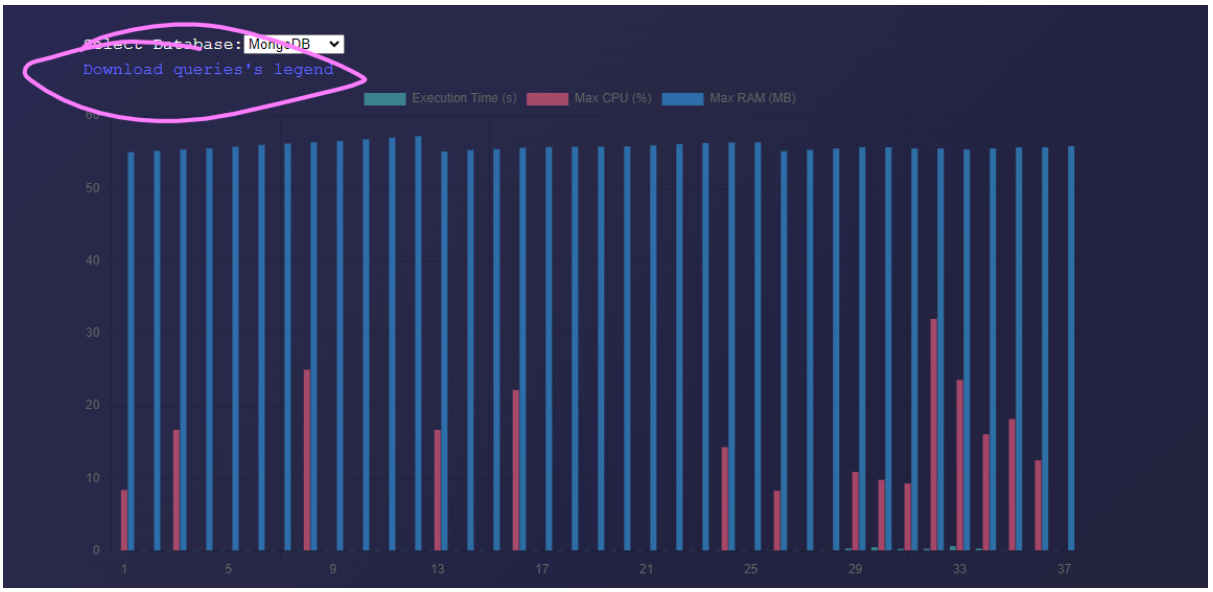


Rys. 4 Wyniki dla PostresSQL wydajności CPU



Rys.5 Wyniki zużycia RAM (celem większego porównania należy kliknąć na poszczególne słupek wykresu - pojawia się możliwość podejrzenia dokładniejszej danej)

Ponadto jest możliwość podglądu legendy zastosowanej numeracji zapytań:



Rys. 6 Podgląd osi poziomej numeracji poszczególnych zapytań