

Департамент образования и науки города Москвы

Государственное автономное образовательное учреждения высшего
образования города Москвы «Московский городской педагогический
университет»

Институт цифрового образования

Департамент информатики, управления и технологий

Лабораторная работа 4.1.

Сравнение подходов хранения больших данных.

Выполнил студент группы АДЭУ-221

Джамалова Сабина Шахиновна

Проверил доцент

Босенко Тимур Муртазович

Москва

2025

Оглавление

Введение	3
Шаг 1. Подготовка окружения с помощью Docker	6
Шаг 2: Настройка баз данных	7
Шаг 3: Выполнение заданий	9
Выводы:	17

Введение

Цель работы: сравнить производительность и эффективность различных подходов к хранению и обработке больших данных на примере реляционной базы данных PostgreSQL и документо-ориентированной базы данных MongoDB.

Вариант задания: 6.

Задание 1 (PostgreSQL). Полнотекстовый поиск. Создать таблицу documents с полем content (тип TEXT). Заполнить 10 000 записей текстом. Выполнить поиск по ключевому слову с использованием to_tsvector.

Задание 2 (MongoDB). Полнотекстовый поиск. Создать коллекцию documents с текстовым полем content. Создать текстовый индекс. Выполнить поиск по ключевому слову с использованием оператора \$text.

Задание 3 (Jupyter Notebook). Сравнить производительность полнотекстового поиска. Сравнить удобство настройки и использования в обеих СУБД.

Краткая теоретическая справка:

PostgreSQL — это объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом, поддерживающая язык SQL.

MongoDB — это документо-ориентированная СУБД, хранящая данные в формате BSON (близком к JSON).

Ключевые различия PostgreSQL и MongoDB – Таблица

Критерий	PostgreSQL	MongoDB
Тип СУБД	Реляционная (SQL)	Документо-ориентированная (NoSQL)
Структура данных	Таблицы, строки, связи	Документы, коллекции
Язык запросов	SQL	BSON / JSON + MongoDB Query Language

Схема данных	Жёсткая (определяется заранее)	Гибкая (можно менять структуру документов)
Поддержка транзакций	Полная, ACID	Полная (с версии 4.0)
Полнотекстовый поиск	Сложный, морфологический (через tsvector, GIN)	Простой, через \$text
Подходит для	Структурированных данных, аналитики	Гибких моделей данных, быстрого прототипирования

Архитектура решения и потоки данных:

PostgreSQL: структура данных

В PostgreSQL была создана одна таблица documents со следующей структурой:

Поле	Тип	Описание
id	UUID	Уникальный идентификатор записи
title	TEXT	Заголовок документа
author	TEXT	Автор документа
content	TEXT	Основное содержимое текста
category	TEXT	Категория документа
date	DATE	Дата создания документа
tags	TEXT[]	Список тегов

MongoDB: структура данных

В MongoDB использовалась коллекция documents, где каждый документ имел следующую структуру:

```
{
  "id": "UUID",
  "title": "Some title",
  "author": "Author Name",
  "content": "Generated text with keyword student...",
  "category": "new",
  "date": ISODate("2024-10-01T00:00:00Z"),
  "tags": ["data", "education"]
}
```

Потоки данных и архитектура эксперимента:

1. Генерация данных (Python):

С помощью библиотеки Faker создавались 10 000 случайных документов с полями title, author, content, category, date и tags. Данные хранились в DataFrame (pandas).

2. Загрузка в PostgreSQL:

- Подключение через SQLAlchemy.
- Данные из DataFrame передавались в таблицу documents методом to_sql().
- Создавался индекс GIN по полю content.

3. Загрузка в MongoDB:

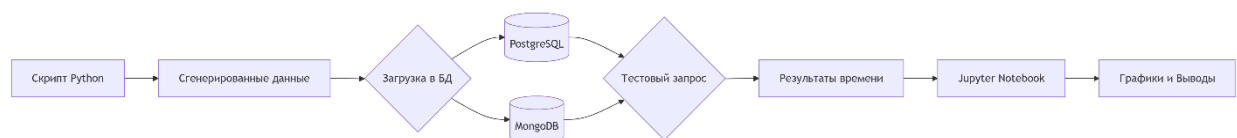
- Подключение через pymongo.
- Датафрейм преобразовывался в список словарей.
- Вставка документов в коллекцию методом insert_many().
- Создавался текстовый индекс (create_index([("content", "text"))]).

4. Выполнение запросов:

- В обеих БД выполнялся поиск по ключевому слову "student".
- Засекалось время выполнения запроса с помощью time.time().

5. Анализ в Jupyter Notebook:

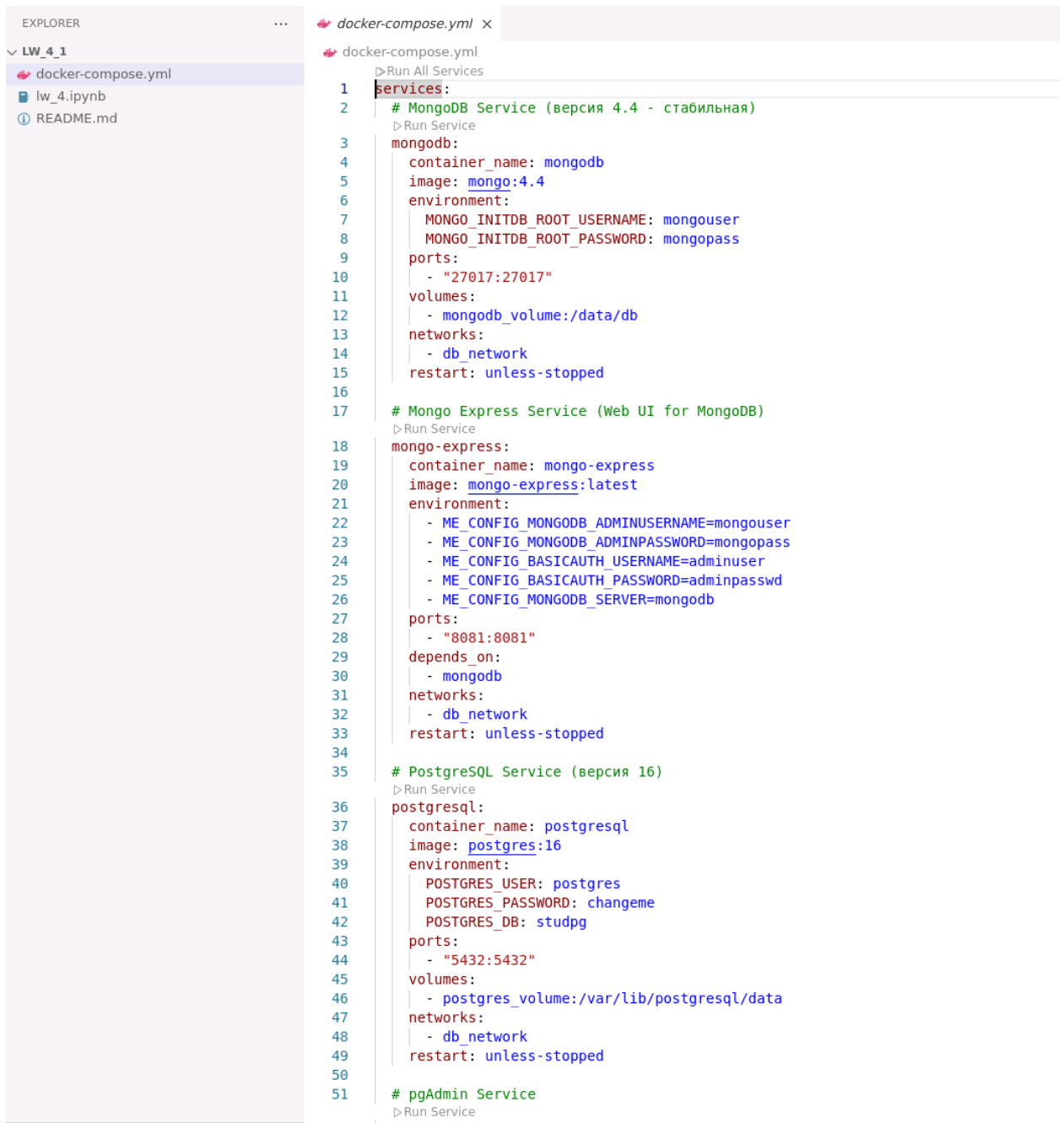
- Время вставки и поиска фиксировалось в таблице Pandas.
- На основе данных строился столбчатый график (matplotlib / pandas.plot).
- Формулировались выводы о производительности и удобстве работы.



Выполнение:

Шаг 1. Подготовка окружения с помощью Docker

1.1. Файл docker-compose.yml:



```
1 services:
2   # MongoDB Service (версия 4.4 - стабильная)
3   mongodb:
4     container_name: mongodb
5     image: mongo:4.4
6     environment:
7       MONGO_INITDB_ROOT_USERNAME: mongouser
8       MONGO_INITDB_ROOT_PASSWORD: mongopass
9     ports:
10      - "27017:27017"
11     volumes:
12      - mongodb_volume:/data/db
13     networks:
14      - db_network
15     restart: unless-stopped
16
17   # Mongo Express Service (Web UI for MongoDB)
18   mongo-express:
19     container_name: mongo-express
20     image: mongo-express:latest
21     environment:
22       - ME_CONFIG_MONGODB_ADMINUSERNAME=mongouser
23       - ME_CONFIG_MONGODB_ADMINPASSWORD=mongopass
24       - ME_CONFIG_BASICAUTH_USERNAME=adminuser
25       - ME_CONFIG_BASICAUTH_PASSWORD=adminpasswd
26       - ME_CONFIG_MONGODB_SERVER=mongodb
27     ports:
28      - "8081:8081"
29     depends_on:
30      - mongodb
31     networks:
32      - db_network
33     restart: unless-stopped
34
35   # PostgreSQL Service (версия 16)
36   postgresql:
37     container_name: postgresql
38     image: postgres:16
39     environment:
40       POSTGRES_USER: postgres
41       POSTGRES_PASSWORD: changeme
42       POSTGRES_DB: studpg
43     ports:
44      - "5432:5432"
45     volumes:
46      - postgres_volume:/var/lib/postgresql/data
47     networks:
48      - db_network
49     restart: unless-stopped
50
51   # pgAdmin Service
```

1.2. Запуск докера:

```

● mgpu@mgpu-vm:~/Downloads/BigDataAnalytic-main/lw/2025/lw_4_1$ sudo docker compose up -d
[sudo] password for mgpu:
[+] Running 57/57
  ✓ postgresql Pulled
  ✓ pgadmin Pulled
  ✓ jupyter Pulled
[+] Running 8/8
  ✓ Network lw_4_1_db_network      Created
  ✓ Volume "lw_4_1_mongodb_volume" Created
  ✓ Volume "lw_4_1_postgres_volume" Created
  ✓ Container postgresql           Started
  ✓ Container jupyter              Started
  ✓ Container mongodb              Started
  ✓ Container pgadmin              Started
  ✓ Container mongo-express        Started
○ mgpu@mgpu-vm:~/Downloads/BigDataAnalytic-main/lw/2025/lw_4_1$

```

1.3. Установка библиотек Python:

```

● mgpu@mgpu-vm:~/Downloads/BigDataAnalytic-main/lw/2025/lw_4_1$ pip install pycopg2-binary pymongo pandas matplotlib sqlalchemy
Collecting pycopg2-binary
  Downloading pycopg2-binary-2.9.10-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.0 MB)
    |████████████████████| 3.0 MB 1.2 MB/s
Collecting pymongo
  Downloading pymongo-4.10.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (930 kB)
    |████████████████████| 930 kB 2.6 MB/s
Collecting pandas
  Downloading pandas-2.0.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.4 MB)
    |████████████████████| 12.4 MB 15.2 MB/s
Collecting matplotlib
  Downloading matplotlib-3.7.5-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (9.2 MB)
    |████████████████████| 9.2 MB 1.0 MB/s
Collecting sqlalchemy
  Downloading sqlalchemy-2.0.43-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.2 MB)
    |████████████████████| 3.2 MB 1.3 MB/s
Collecting dnspython<3.0.0,>=1.16.0
  Downloading dnspython-2.6.1-py3-none-any.whl (307 kB)
    |████████████████████| 307 kB 3.3 MB/s
Collecting numpy>=1.20.3; python_version < "3.10"
  Downloading numpy-1.24.4-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
    |████████████████████| 17.3 MB 1.1 MB/s
Collecting python-dateutil>=2.8.2

```

Шаг 2: Настройка баз данных

Проверка доступов:

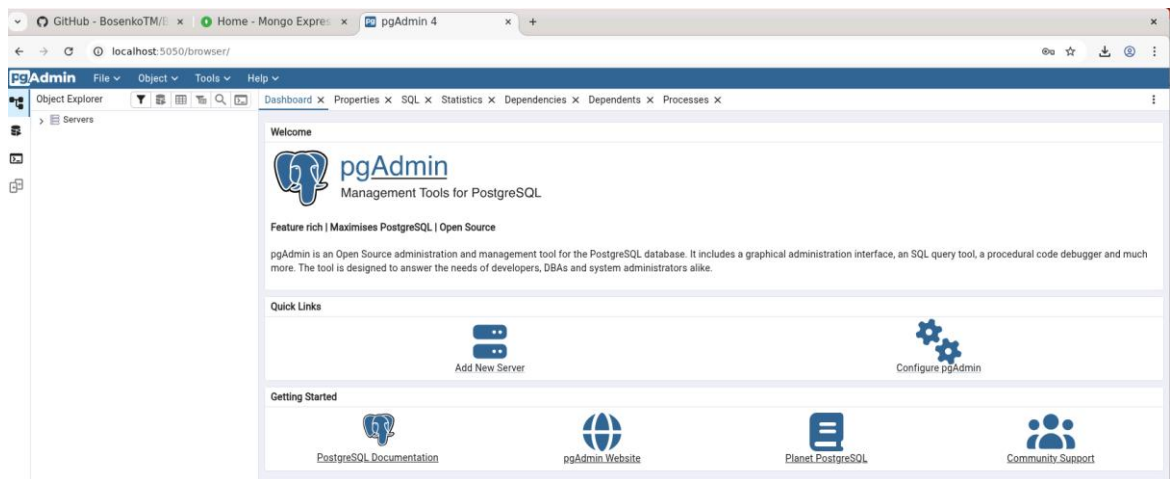
The screenshot shows the pgAdmin 4 web interface in a browser window. The address bar shows 'localhost:8081'. The interface has a sidebar with 'Mongo Express' selected. The main content area is divided into two sections: 'Databases' and 'Server Status'.

Databases Section:

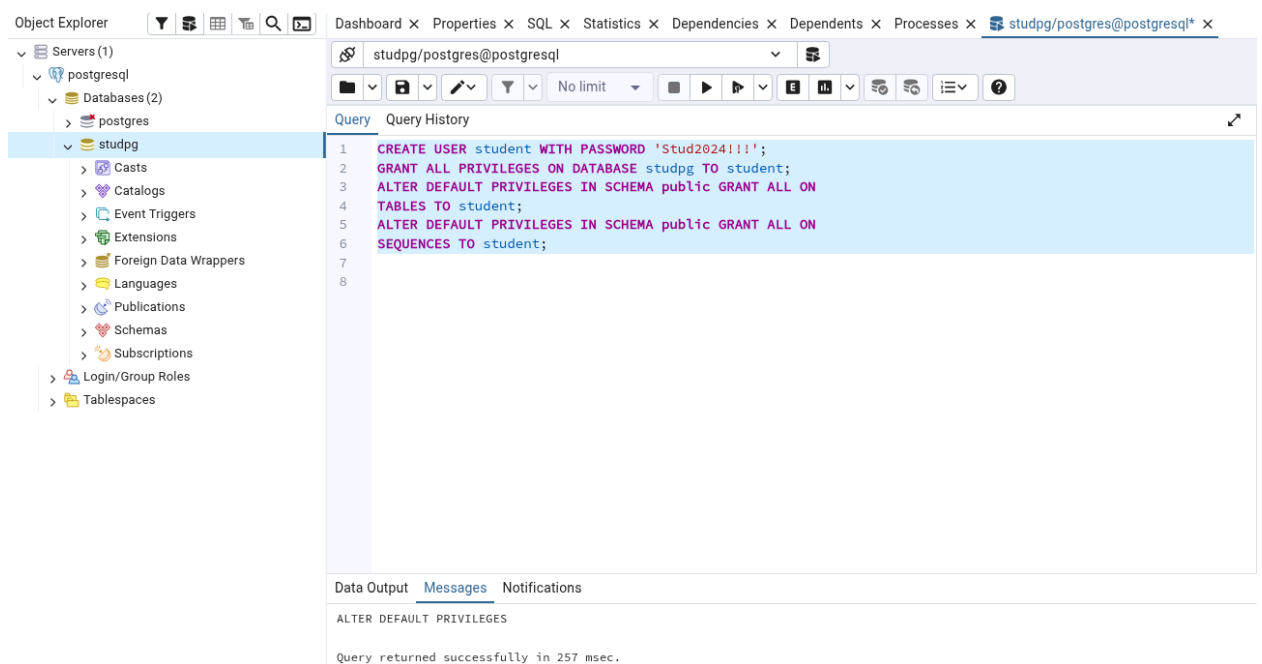
Database Name	View	Del
admin	View	Del
config	View	Del
local	View	Del

Server Status Section:

Server Status	
Hostname	3d2494e0ce43
Uptime	1352 seconds
Server Time	Sat, 04 Oct 2025 20:30:54 GMT
MongoDB Version	4.4.29
Node Version	18.20.3
VB Version	10.2.154.26-node.37
Current Connections	3
Active Clients	0
Clients Reading	0
Read Lock Queue	0
Available Connections	838857
Queued Operations	0
Clients Writing	0
Write Lock Queue	0
Disk Flushes	
Time Spent Flushing	ms
Last Flush	
Average Flush Time	ms
Total Inserts	0
Total Updates	1
Total Queries	6
Total Deletes	0



Работа в PostgreSQL:




Работа в MongoDB:
















- ✓ Успешное подключение к MongoDB
- ✓ Подключение через Docker сервис 'mongodb'
- 📥 Загрузка данных в MongoDB...
- ✓ Загружено 10,000 пользователей
- ✓ Загружено 1,000 товаров
- ✓ Загружено 100,000 просмотров
- ✓ Созданы индексы для оптимизации запросов

Mongo Express

Databases			Database Name	+ Create Database
 View	admin	 Del		
 View	config	 Del		
 View	local	 Del		
 View	studmongo	 Del		

 Mongo Express Database: studmongo ▾

Viewing Database: studmongo

Collections					Collection Name	+ Create collection
 View	 Export	 [JSON]	 Import	products	 Del	
 View	 Export	 [JSON]	 Import	users	 Del	
 View	 Export	 [JSON]	 Import	views	 Del	

Database Stats	
Collections (incl. system.namespaces)	3
Data Size	8.01 MB
Storage Size	3.11 MB
Avg Obj Size #	72.2 Bytes
Objects #	111000
Indexes #	8
Index Size	3.51 MB

Шаг 3: Выполнение заданий

1. Генерация данных:

```
import random
from faker import Faker
import pandas as pd
fake = Faker()

N = 10_000
categories = ["new", "old", "research", "study"]
```

```

tags_list = ["data", "analysis", "project", "experiment",
"report", "student", "education"]

rows = []

keyword = "student"

for _ in range(N):
    # Заголовок и автор
    title = fake.sentence(nb_words=random.randint(3,6))
    author = fake.name()

    # Основной текст: 40-120 слов
    length = random.randint(40,120)
    words = [fake.word() for _ in range(length)]
    # С вероятностью 20% ключевое слово 'student'
    if random.random() < 0.2:
        pos = random.randrange(length)
        words[pos] = keyword
    content = ' '.join(words).capitalize() + '.'

    # Категория, дата и теги
    category = random.choice(categories)
    date = fake.date_this_decade()
    tags = random.sample(tags_list, k=random.randint(1,3))
    rows.append({
        "id": fake.uuid4(),
        "title": title,
        "author": author,
        "content": content,
        "category": category,
        "date": date,
        "tags": tags
    })

df = pd.DataFrame(rows)
df.head(13)

```

Результат генерации:

	id	title	author	content	category	date	tags
0	92cf3ea4-3154-4655-8859-07dda698fba9	Someone sell pretty.	Teresa Cole	Democrat important water four style hospital s...	new	2024-10-01	[report, student, data]
1	7132d653-bd85-4285-b78e-051729fed027	We growth important ever serve interview.	Veronica Becker	Data upon exactly tree friend take despite nic...	study	2023-09-25	[student, analysis]
2	9d3c6a10-4191-4c8f-a446-eb2eba7256ae	Capital method style firm indicate yard drive.	Richard Dunn	Including piece my future yet strategy pass re...	old	2025-01-19	[education, data, report]
3	0e519cc3-d141-45b5-8bfc-e60e71a57c09	Wind general compare wait.	Derrick Nguyen	For place cover site service commercial year s...	old	2022-11-19	[education, analysis, student]
4	d5f017c4-b86f-47b1-84ac-a3d6746182ab	Federal success project long herself page.	Brady Padilla	Describe beautiful so value really seek while ...	research	2023-02-01	[experiment]
5	6f73afd0-60ed-4b97-aed6-a32828c6bb9e	Pull officer imagine.	Joshua Jordan	Series others quality myself piece force mr bo...	research	2023-02-07	[data, student]
6	eb0b5822-e2fa-4dac-b654-6d3342630803	Near PM perform build.	Tim Martinez	Create measure choice cut list top skill assum...	research	2022-07-10	[project]
7	3232c70c-24e7-4d23-b9ad-04fed27c19a4	Look contain important others.	Jon Parker	Physical sell wrong far image practice beyond ...	old	2021-08-02	[education]
8	83c43b3b-77a9-4b5e-bf28-984a0f0bac79	Real produce probably enter sometimes bank.	Tracey Barnes	Onto phone hard owner company well bag fact in...	new	2021-04-25	[project]
9	bc0c6269-0f24-4c1c-a88e-a9a4992007e0	Find force central great leg fight politics.	Theresa Anderson	What middle matter interesting crime stay town...	old	2020-04-11	[student, analysis]
10	0d6edba9-a5a3-44e4-95a9-8e1ea22fa813	Mean soon.	Nicole Shaw	Democrat large role against occur professor ou...	old	2020-01-21	[report]
11	e31a4c79-7e72-48dc-9d62-7ab0b5ebac78	Keep anyone product.	Carrie Watts	Admit me those high more because actually stud...	old	2023-04-04	[education, student, analysis]
12	32202f6e-e7aa-4e9f-8ebc-3058fec4745b	To cell structure you.	Robert Gibson	Far as can vote between age suggest natural th...	research	2023-06-07	[data]

2. Работа с PostgreSQL:

Создание таблицы для дальнейшего ввода данных:

```
cur.execute("DROP TABLE IF EXISTS documents;")

cur.execute("""

CREATE TABLE documents (

    id UUID PRIMARY KEY,

    title TEXT,

    author TEXT,

    content TEXT,

    category TEXT,

    date DATE,

    tags TEXT[]

);

""")

conn.commit()
```

Tables (1)

documents

Columns (7)

id

title

author

content

category

date

tags

> Constraints

> Indexes

> RLS Policies

> Rules

> Triggers

Добавление GIN-индекса:

```
cur.execute("""

CREATE INDEX idx_documents_tsv

ON documents USING GIN(to_tsvector('english', content));

""")

conn.commit()
```

Зачем он нужен? GIN-индекс хранит для каждого слова список документов, в которых оно встречается. Использую английский как конфигурацию языка.

Вставка сгенерированных данных:

```
insert_query = """
INSERT INTO documents (id, title, author, content, category,
date, tags)
VALUES (%s, %s, %s, %s, %s, %s, %s)
"""

records = df.apply(lambda row: (
    row['id'], row['title'], row['author'], row['content'],
    row['category'], row['date'], row['tags']
), axis=1).tolist()

start_insert = time.time()
cur.executemany(insert_query, records)
conn.commit()

elapsed_insert = time.time() - start_insert

print(f"Вставка {len(df)} документов заняла {elapsed_insert:.2f} сек")
```

Вставка 10000 документов заняла 5.71 сек

Проверка количеством:

Query		Query History
1	SELECT COUNT(*) FROM documents;	
Data Output		Messages Notifications
<div><div>≡+</div><div> SQL</div></div>		
	count	bigint
1	10000	

Реализация запроса. Поиск по ключевому слову с использованием to_tsvector:

```
keyword = 'student'

start_search = time.time()

cur.execute("""
```

```

SELECT COUNT(*) FROM documents
WHERE to_tsvector('english', content) @@ to_tsquery(%s)
""", (keyword,))
count = cur.fetchone()[0]
elapsed_search = time.time() - start_search

print(f"Найдено {count} документов с ключевым словом
'{keyword}', поиск занял {elapsed_search:.4f} сек")

Найдено 2690 документов с ключевым словом 'student', поиск занял 0.0199 сек

```

Время выполнения запроса – 0.0199 секунд.

3. Работа с MongoDB:

Подключение к MongoDB:

```

from pymongo import MongoClient

client =
MongoClient("mongodb://mongouser:mongopass@mongodb:27017/")

# База данных studmongo
db = client["studmongo"]

```

Вставка документов из DataFrame:

```

import datetime

# Преобразование столбца date в datetime.datetime
df['date'] = df['date'].apply(lambda d:
datetime.datetime(d.year, d.month, d.day))

# Преобразование DataFrame в список словарей
records = df.to_dict(orient='records')

# Вставка всех документов
import time
start_insert = time.time()

```

```
collection.insert_many(records)

elapsed_insert = time.time() - start_insert

print(f"Вставка {len(records)} документов заняла
{elapsed_insert:.2f} сек")
```

Вставка 10000 документов заняла 7.12 сек

Создание текстового индекса для полнотекстового поиска:

```
collection.create_index([("content", "text")])
```

MongoDB автоматически учитывает стоп-слова и стемминг для английского текста.

Проверка в веб-интерфейсе:

Viewing Database: studmongo

Viewing Collection: documents

_id	id	title	author	content	category	date	tags
 68e2710467cd849b24d8afe5	92cf3ea4-3154-4655-8859-07dda698fba9	Someone sell pretty.	Teresa Cole	Democrat important water four style hospital size...	new	Tue Oct 01 2024 00:00:00 GMT+0000 (Coordinated Universal Time)	report,student,data
 68e2710467cd849b24d8afe6	7132d653-bd85-4285-b78e-051729fed027	We growth important ever serve interview.	Veronica Becker	Data upon exactly tree friend take despite nice k...	study	Mon Sep 25 2023 00:00:00 GMT+0000 (Coordinated Universal Time)	student,analysis
 68e2710467cd849b24d8afe7	9d3c6a10-4191-4c8f-a446-eb2eba7256ae	Capital method style firm indicate yard drive.	Richard Dunn	Including piece my future yet strategy pass recor...	old	Sun Jan 19 2025 00:00:00 GMT+0000 (Coordinated Universal Time)	education,data,report
 68e2710467cd849b24d8afe8	0e519cc3-d141-45b5-8bfc-e60e71a57c09	Wind general compare wait.	Derrick Nguyen	For place cover site service commercial year some...	old	Sat Nov 19 2022 00:00:00 GMT+0000 (Coordinated Universal Time)	education,analysis,student
 68e2710467cd849b24d8afe9	d5f017c4-b86f-47b1-84ac-a3d6746182ab	Federal success project long herself page.	Brady Padilla	Describe beautiful so value really seek while exp...	research	Wed Feb 01 2023 00:00:00 GMT+0000 (Coordinated Universal Time)	experiment

Реализация запроса. Поиск по ключевому слову с использованием оператора \$text:

```
start_search = time.time()

count = collection.count_documents({"$text": {"$search":
"student"}})

elapsed_search = time.time() - start_search

print(f"Найдено {count} документов с ключевым словом 'student',
поиск занял {elapsed_search:.4f} сек")
```

Найдено 2690 документов с ключевым словом 'student', поиск занял 0.0951 сек

Время выполнения запроса – 0.0951 секунд.

4. Анализ в Jupyter Notebook:

Таблица:

```
import pandas as pd

# Таблица для сравнения

data = {

    "СУБД": ["PostgreSQL", "MongoDB"],

    "Вставка, сек": [5.71, 7.12],

    "Поиск 'student', сек": [0.0199, 0.0951],

}

results_df = pd.DataFrame(data)

results_df
```

	СУБД	Вставка, сек	Поиск 'student', сек
0	PostgreSQL	5.71	0.0199
1	MongoDB	7.12	0.0951

График для сравнения времени, затраченного на вставку и поиск:

```
# --- График 1: вставка ---

plt.figure(figsize=(7,4))

plt.bar(results_df["СУБД"], results_df["Вставка, сек"],
color=['skyblue', 'salmon'])

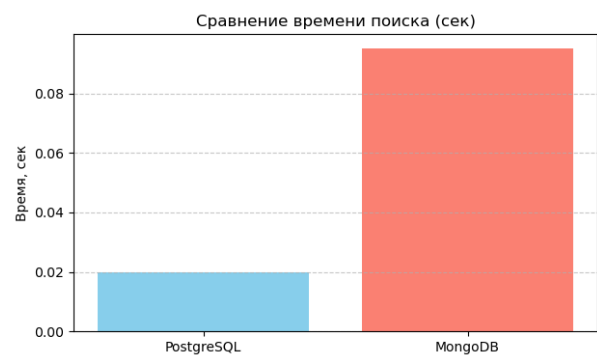
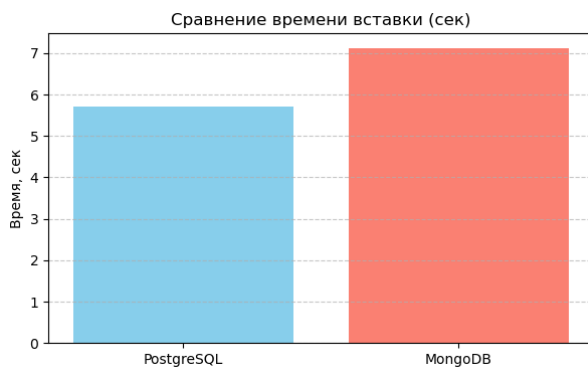
plt.title("Сравнение времени вставки (сек)")

plt.ylabel("Время, сек")
```

```
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```

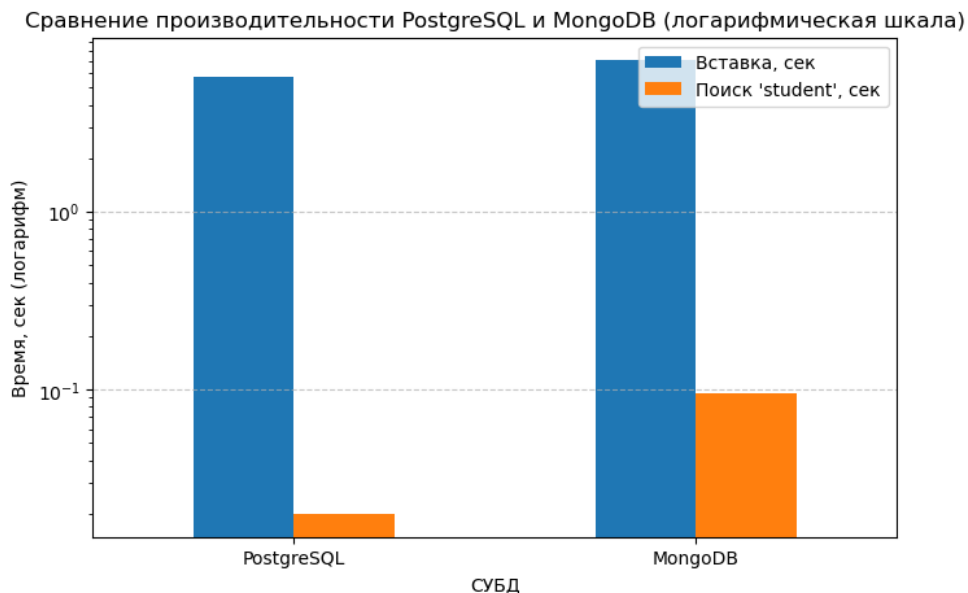
```
# --- График 2: поиск ---
```

```
plt.figure(figsize=(7,4))
plt.bar(results_df["СУВД"], results_df["Поиск 'student', сек"],
color=['skyblue','salmon'])
plt.title("Сравнение времени поиска (сек)")
plt.ylabel("Время, сек")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



Логарифмическая шкала:

```
plt.figure(figsize=(8,5))
results_df.plot(
    x="СУВД",
    y=["Вставка, сек", "Поиск 'student', сек"],
    kind="bar",
    figsize=(8,5),
    logy=True, # логарифмическая шкала
    title="Сравнение производительности PostgreSQL и MongoDB
(логарифмическая шкала) "
)
plt.ylabel("Время, сек (логарифм)")
plt.xticks(rotation=0)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```

Выводы:

1. Производительность:

- При вставке 10 000 документов PostgreSQL показал немного лучшее время загрузки данных (≈ 5.7 с против 7.1 с в MongoDB).
- При выполнении полнотекстового поиска по ключевому слову *"student"* PostgreSQL оказался значительно быстрее (≈ 0.02 с против 0.09 с в MongoDB).

Это объясняется тем, что PostgreSQL использует встроенные механизмы индексирования GIN и функцию `to_tsvector`, оптимизированные для полнотекстового поиска по большим текстовым полям.

2. Удобство настройки:

- В PostgreSQL требуется вручную создать индекс типа GIN с использованием `to_tsvector(language, field)` и выполнять запросы через `to_tsquery()`. Это чуть сложнее, но обеспечивает большую гибкость (поддержка морфологии, языков, весов слов и т.д.).

- В MongoDB настройка проще: достаточно создать текстовый индекс (`create_index([("field", "text")])`) и использовать `$text` в запросе. Однако система поиска MongoDB менее гибкая и не поддерживает лингвистические особенности на уровне PostgreSQL.

3. Общее сравнение:

- PostgreSQL — лучше для сложных аналитических задач, где важны точность и скорость поиска по большим текстовым массивам.
- MongoDB — проще в использовании и подходит для приложений, где важна гибкость схемы данных и лёгкость интеграции.

Лично мне комфортнее и привычнее работать в PostgreSQL, поскольку его синтаксис и инструменты более знакомы, понятны и логичны. Именно эта СУБД стала предпочтительной для меня.

В данном задании PostgreSQL оказался быстрее MongoDB по двум причинам. Во-первых, PostgreSQL использует реляционную структуру с заранее определённой схемой и поддержкой GIN-индекса для полнотекстового поиска. Этот индекс оптимизирован для быстрого поиска по большим текстовым полям с учётом морфологии и ранжирования релевантности. Во-вторых, PostgreSQL хорошо оптимизирован для транзакций и работы с типами данных, что ускоряет вставку и выборку записей.

MongoDB, напротив, хранит данные в формате документов с гибкой схемой, что делает её удобной для прототипирования, но поиск текста менее оптимизирован. Текстовый индекс и оператор `$text` ускоряют поиск, но не учитывают морфологию слов и не применяют продвинутые алгоритмы ранжирования, поэтому поиск идёт медленнее и возвращает немного отличающиеся результаты.

Таким образом, особенности архитектуры — реляционная структура и мощные индексы в PostgreSQL против документо-ориентированной гибкой схемы в MongoDB — напрямую влияют на производительность и точность полнотекстового поиска в этом конкретном задании.