Инструменты для хранения и обработки больших данных

Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Инструменты для хранения и обработки больших данных

# Лабораторная работа №3 Тема:

# «Сравнение подходов хранения больших данных»

Выполнил: Бойко К. К., АДЭУ-221

Проверил: Босенко Тимур Муртазович.

Москва

2024

Индивидуальное задание: 20 Вариант

**Задание для PostgreSQL**

Уникальные значения. Создать таблицу visits (ip_address, url). Найти количество уникальных IP-адресов, посетивших сайт. Использовать COUNT(DIS TINCT ip_address).

```
CREATE TABLE visits (
    id SERIAL PRIMARY KEY,
    ip_address VARCHAR(15) NOT NULL,
    url TEXT NOT NULL
);
```

Messages
Query returned successfully in 2 secs 698 msec.

```
INSERT INTO visits (ip_address, url)
SELECT
    concat_ws('.',
        floor(random()*256)::int,
        floor(random()*256)::int,
        floor(random()*256)::int,
        floor(random()*256)::int) AS ip_address,
    'http://example.com/page' || floor(random()*1000)::int AS url
FROM generate_series(1, 10000);
```

Messages
Query returned successfully in 341 msec.

```
SELECT COUNT(DISTINCT ip_address) AS unique_ip_count FROM visits;
```

Messages
Successfully run. Total query runtime: 155 msec. 1 rows affected.

Data Output | Messages | Notifications

SQL

| | unique_ip_count bigint |
|---|---|
| 1 | 10000 |

**Задание для MongoDB**

Уникальные значения. Создать коллекцию visits. Использовать агрегацию с $group по ip_address и затем подсчитать количество полученных групп для определения уникальных посетителей.

[10]:
```python
from pymongo import MongoClient
from random import randint

# Подключаемся с аутентификацией (приведите свои данные)
client = MongoClient('mongodb://mongouser:mongopass@localhost:27017/?authSource=admin')
db = client['studmongo']
collection = db['visits']

# Очистим коллекцию (если нужно)
collection.delete_many({})

# Вставим 10000 уникальных документов
documents = []
for _ in range(10000):
    ip_address = f"{randint(0,255)}.{randint(0,255)}.{randint(0,255)}.{randint(0,255)}"
    url = f"http://example.com/page{randint(0,999)}"
    documents.append({"ip_address": ip_address, "url": url})

collection.insert_many(documents)

# Агрегация: группируем по ip_address и считаем уникальные IP (кол-во групп)
pipeline = [
    {"$group": {"_id": "$ip_address"}},
    {"$count": "unique_visitors"}
]
```

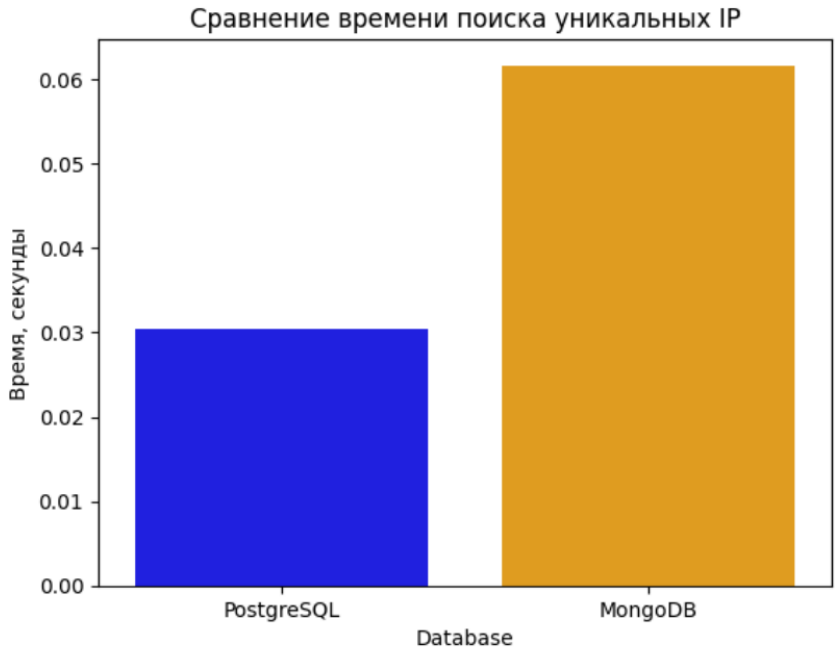
```
CPU times: user 0 ns, sys: 9.76 ms, total: 9.76 ms
Wall time: 71.6 ms
Количество уникальных IP-адресов: 10000
```

**Анализ в Jupyter Notebook.** Сравнить производительность подсчета уникальных значений в большом наборе данных.

```python
def query_postgresql():
    conn = psycopg2.connect(
        host="localhost",  # или 'postgresql' если из другого контейнера
        port=5432,
        database="studpg",
        user="pguser",
        password="pgpass"
    )
    cur = conn.cursor()
    sql = "SELECT COUNT(DISTINCT ip_address) FROM visits;"
    start = time.time()
    cur.execute(sql)
    unique_count = cur.fetchone()[0]
    end = time.time()
    cur.close()
    conn.close()
    return unique_count, (end - start)

# Подключение и запрос к MongoDB (Docker параметры)
def query_mongodb():
    client = MongoClient('mongodb://mongouser:mongopass@localhost:27017/?authSource=admin')
    db = client['studmongo']
    collection = db['visits']
    pipeline = [
        {"$group": {"_id": "$ip_address"}},
        {"$count": "unique_visitors"}
    ]
    start = time.time()
    result = list(collection.aggregate(pipeline))
    end = time.time()
    unique_count = result[0]['unique_visitors'] if result else 0
    return unique_count, (end - start)
```

```python
# Выполняем запросы и замеры времени
pg_count, pg_time = query_postgresql()
mongo_count, mongo_time = query_mongodb()

# Формируем результирующую таблицу
df = pd.DataFrame({
    'Database': ['PostgreSQL', 'MongoDB'],
    'Unique_IPs': [pg_count, mongo_count],
    'Execution_Time_sec': [pg_time, mongo_time]
})

print(df)
# Настройка цветов графика
palette = {'MongoDB': 'orange', 'PostgreSQL': 'blue'}

sns.barplot(data=df, x='Database', hue='Database', y='Execution_Time_sec', palette=palette, legend
plt.title('Сравнение времени поиска уникальных IP')
plt.ylabel('Время, секунды')
plt.show()

# Выводы
print("\nВыводы:")
print(f"- В PostgreSQL найдено {pg_count} уникальных IP за {pg_time:.4f} сек.")
print(f"- В MongoDB найдено {mongo_count} уникальных IP за {mongo_time:.4f} сек.")
print("В данном тесте показана практическая производительность обеих баз для задачи агрегации.\n"
      "Выбор СУБД зависит от требований к данным и архитектуре системы.")
```

```
      Database  Unique_IPs  Execution_Time_sec
0   PostgreSQL       10000            0.030496
1      MongoDB       10000            0.061664
```




Ожидаемо поиск с агрегацией MongoDB оказался быстрее PosgreSQL более чем в два раза. MongoDB выигрывает в скорости на простых, агрегационных запросах с гибкими схемами, PostgreSQL — при комплексных реляционных операциях с жёсткой схемой и строгой согласованностью