Департамент образования города Москвы

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

Институт цифрового образования
Департамент информатики, управления и технологий

ПРАКТИЧЕСКАЯ РАБОТА №4

по дисциплине «Инструменты для хранения и обработки больших данных»
Направление подготовки 38.03.05 – бизнес-информатика
Профиль подготовки «Аналитика данных и эффективное управление»
(очная форма обучения)

Выполнила:
Студентка группы АДЭУ-221
Вознесенская В. Е.

Проверил:
Босенко Т. М., доцент

Москва
2025

СРАВНЕНИЕ ПОДХОДОВ ХРАНЕНИЯ БОЛЬШИХ ДАННЫХ

Генерируем данные (полный код предоставлен в GitHub)

```python
import csv
import json
import random
from datetime import datetime, timedelta
import os
import shutil
from google.colab import files

# Создаем папку для данных
os.makedirs("data", exist_ok=True)

# --- Категории ---
categories = [
    {"id": 1, "name": "Phones"},
    {"id": 2, "name": "Computers"},
    {"id": 3, "name": "Headphones"}
]

# --- Продукты ---
products = []

product_names = {
    1: [
        "iPhone 14 128GB", "iPhone 14 256GB", "iPhone 15 128GB", "iPhone 15 Pro",
        "Samsung Galaxy S23", "Samsung Galaxy S23 Ultra", "Google Pixel 8", "Google Pixel 8 Pro",
        "OnePlus 11", "Xiaomi 13 Pro"
    ],
    2: [
        "MacBook Pro 13 M2", "MacBook Pro 16 M2 Pro", "MacBook Air M2",
        "Dell XPS 13 9310", "Dell XPS 15 9520", "HP Spectre x360 14",
        "Lenovo ThinkPad X1 Carbon", "Asus ZenBook 14 OLED", "Acer Swift 5"
    ],
    3: [
        "Sony WH-1000XM5", "Sony WH-1000XM4", "Bose QC45", "Bose QC35 II",
        "Apple AirPods Max", "Apple AirPods Pro 2", "Sennheiser HD 560S", "JBL Live 660NC"
    ]
```

Загружаем данные в PostgreSQL:

```
studpg=> \q
mgpu@mgpu-vm:~/Downloads/idb/nosql-workshop/01-environment/docker$ docker exec -it postgresql ls -l /tmp
total 220
-rw-rw-r-- 1 1000 1000     46 Nov 11 06:59 categories.csv
-rw-rw-r-- 1 1000 1000 215557 Nov 11 06:59 orders.csv
-rw-rw-r-- 1 1000 1000    839 Nov 11 06:59 products.csv
mgpu@mgpu-vm:~/Downloads/idb/nosql-workshop/01-environment/docker$ docker exec -it postgresql psql -U pguser -d studpg
psql (16.10 (Debian 16.10-1.pgdg13+1))
Type "help" for help.

studpg=> \COPY categories FROM '/tmp/categories.csv' WITH (FORMAT csv, HEADER true);
COPY 3
studpg=> \COPY products   FROM '/tmp/products.csv'   WITH (FORMAT csv, HEADER true);
COPY 27
studpg=> \COPY orders     FROM '/tmp/orders.csv'     WITH (FORMAT csv, HEADER true);
COPY 10000
studpg=>
```

Выполняем простые запросы для проверки:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

----------------
             3
(1 row)

studpg=> SELECT COUNT(*) AS cnt_products FROM products;
 cnt_products
--------------
           27
(1 row)

studpg=> SELECT COUNT(*) AS cnt_orders FROM orders;
 cnt_orders
------------
      10000
(1 row)

studpg=>
```

```
studpg=> SELECT * FROM products ORDER BY id LIMIT 10;
 id |           name           | category_id |  price
----+--------------------------+-------------+---------
  1 | iPhone 14 128GB          |           1 |  639.75
  2 | iPhone 14 256GB          |           1 | 1191.25
  3 | iPhone 15 128GB          |           1 | 1267.78
  4 | iPhone 15 Pro            |           1 |  617.32
  5 | Samsung Galaxy S23       |           1 |  633.45
  6 | Samsung Galaxy S23 Ultra |           1 | 1103.89
  7 | Google Pixel 8           |           1 | 1231.32
  8 | Google Pixel 8 Pro       |           1 |  882.07
  9 | OnePlus 11               |           1 |  807.71
 10 | Xiaomi 13 Pro            |           1 |  965.02
(10 rows)
```

Далее файлы также были загружены в Mongo:

```
 ✔ Container mongo-express  Running
mgpu@mgpu-vm:~/Downloads/idb/nosql-workshop/01-environment/docker$ docker exec -it mongodb ls -l /tmp
total 952
-rw-rw-r-- 1    1000    1000    138 Nov 11 13:36 categories.json
srwx------ 1 mongodb mongodb      0 Nov 11 22:02 mongodb-27017.sock
-rw-rw-r-- 1    1000    1000 965524 Nov 11 13:37 orders.json
-rw-rw-r-- 1    1000    1000   2623 Nov 11 13:37 products.json
```

Просмотр для проверки:

```
---
> use studmongo
switched to db studmongo
> db.products.findOne()
{
        "_id" : ObjectId("69133dedfd28e505d45d3aaf"),
        "id" : 1,
        "name" : "iPhone 14 128GB",
        "category_id" : 1,
        "price" : 639.75
}

> db.orders.find().limit(5).pretty()
{
        "_id" : ObjectId("69133df996ea7d4450f29641"),
        "id" : 1,
        "product_id" : 24,
        "quantity" : 3,
        "order_date" : "2025-10-04"
}
{
        "_id" : ObjectId("69133df996ea7d4450f29642"),
        "id" : 2,
        "product_id" : 12,
        "quantity" : 1,
        "order_date" : "2025-10-27"
}
{
        "_id" : ObjectId("69133df996ea7d4450f29643"),
        "id" : 3,
        "product_id" : 18,
        "quantity" : 5,
        "order_date" : "2025-10-22"
}
{
        "_id" : ObjectId("69133df996ea7d4450f29644"),
        "id" : 4,
        "product_id" : 4,
        "quantity" : 2,
        "order_date" : "2025-10-21"
}
{
        "_id" : ObjectId("69133df996ea7d4450f29645"),
        "id" : 5,
        "product_id" : 13,
        "quantity" : 3,
```

Далее подключаемся к PostgreSQL через Jupiter:

[2]:
```python
import psycopg2

# параметры подключения
conn_params = {
    "host": "postgresql",
    "port": 5432,
    "dbname": "studpg",
    "user": "pguser",
    "password": "pgpass"
}

try:
    conn = psycopg2.connect(**conn_params)
    print("✅ Подключение к PostgreSQL успешно!")
except Exception as e:
    print("❌ Ошибка подключения:", e)
```

```
✅ Подключение к PostgreSQL успешно!
```

Согласно требованиям задания 1 варианта, выполняем запрос и засекаем время выполнения.

[4]:
```python
import pandas as pd
import psycopg2
import time


# сам SQL-запрос
query = """
WITH month_orders AS (
  SELECT
    o.product_id,
    p.name AS product_name,
    p.category_id,
    c.name AS category_name,
    SUM(o.quantity) AS total_qty
  FROM orders o
  JOIN products p ON p.id = o.product_id
  JOIN categories c ON c.id = p.category_id
  WHERE o.order_date >= (CURRENT_DATE - INTERVAL '30 days')
  GROUP BY o.product_id, p.name, p.category_id, c.name
)
SELECT
  category_name,
  product_name,
  total_qty
FROM (
  SELECT
    mo.*,
    ROW_NUMBER() OVER (PARTITION BY category_id ORDER BY total_qty DESC) AS rn
  FROM month_orders mo
) ranked
WHERE rn <= 5
ORDER BY category_name, total_qty DESC;
"""

# засечем время выполнения
start = time.time()
df_top = pd.read_sql(query, conn)
end = time.time()

# выводим результаты
print(f"⏱ Время выполнения запроса: {end - start:.4f} сек.\n")
display(df_top)

# закрываем соединение
conn.close()
```

```
⏱ Время выполнения запроса: 0.0994 сек.
```

	category_name	product_name	total_qty
0	Computers	Dell XPS 13 9310	716
1	Computers	Asus ZenBook 14 OLED	642
2	Computers	MacBook Pro 16 M2 Pro	630
3	Computers	Lenovo ThinkPad X1 Carbon	621
4	Computers	MacBook Air M2	608
5	Headphones	Bose QC45	672
6	Headphones	Apple AirPods Pro 2	654
7	Headphones	Sony WH-1000XM4	636
8	Headphones	JBL Live 660NC	631
9	Headphones	Apple AirPods Max	601
10	Phones	Samsung Galaxy S23	732
11	Phones	iPhone 15 128GB	688
12	Phones	iPhone 14 128GB	657
13	Phones	Samsung Galaxy S23 Ultra	657
14	Phones	iPhone 14 256GB	644

Далее подключаемся к Mongo и также выполняем запрос и засекаем время выполнения:

```
from pymongo import MongoClient

# создаём клиент
client = MongoClient(
    "mongodb://mongouser:mongopass@mongodb:27017/?authSource=admin"
)

# выбираем базу
db = client["studmongo"]

# проверка: сколько документов в коллекции products
print("Products:", db.products.count_documents({}))
print("Orders:", db.orders.count_documents({}))
print("Categories:", db.categories.count_documents({}))
```

```
Products: 27
Orders: 10000
Categories: 6
```

[9]:
```python
from datetime import datetime, timedelta
from pymongo import MongoClient
import time

client = MongoClient("mongodb://mongouser:mongopass@mongodb:27017/")
db = client["studmongo"]

# Конвертируем дату отсечения в строку формата YYYY-MM-DD
thirty_days_ago_str = (datetime.now() - timedelta(days=30)).strftime("%Y-%m-%d")

start_time = time.time()

pipeline = [
    {"$addFields": {
        # Преобразуем строковую дату в настоящий Date
        "order_date_dt": {"$dateFromString": {"dateString": "$order_date"}}
    }},
    {"$match": {"order_date_dt": {"$gte": datetime.strptime(thirty_days_ago_str, "%Y-%m-%d")}}},
    {"$lookup": {
        "from": "products",
        "localField": "product_id",
        "foreignField": "id",
        "as": "product_info"
    }},
    {"$unwind": "$product_info"},
    {"$lookup": {
        "from": "categories",
        "localField": "product_info.category_id",
        "foreignField": "id",
        "as": "category_info"
    }},
    {"$unwind": "$category_info"},
    {"$group": {
        "_id": {
            "category": "$category_info.name",
            "product": "$product_info.name"
        },
        "total_qty": {"$sum": "$quantity"}
    }},
    {"$sort": {"_id.category": 1, "total_qty": -1}},
    {"$group": {
        "_id": "$_id.category",
        "top_products": {
            "$push": {
                "product_name": "$_id.product",
                "total_qty": "$total_qty"
            }
        }
    }},
```

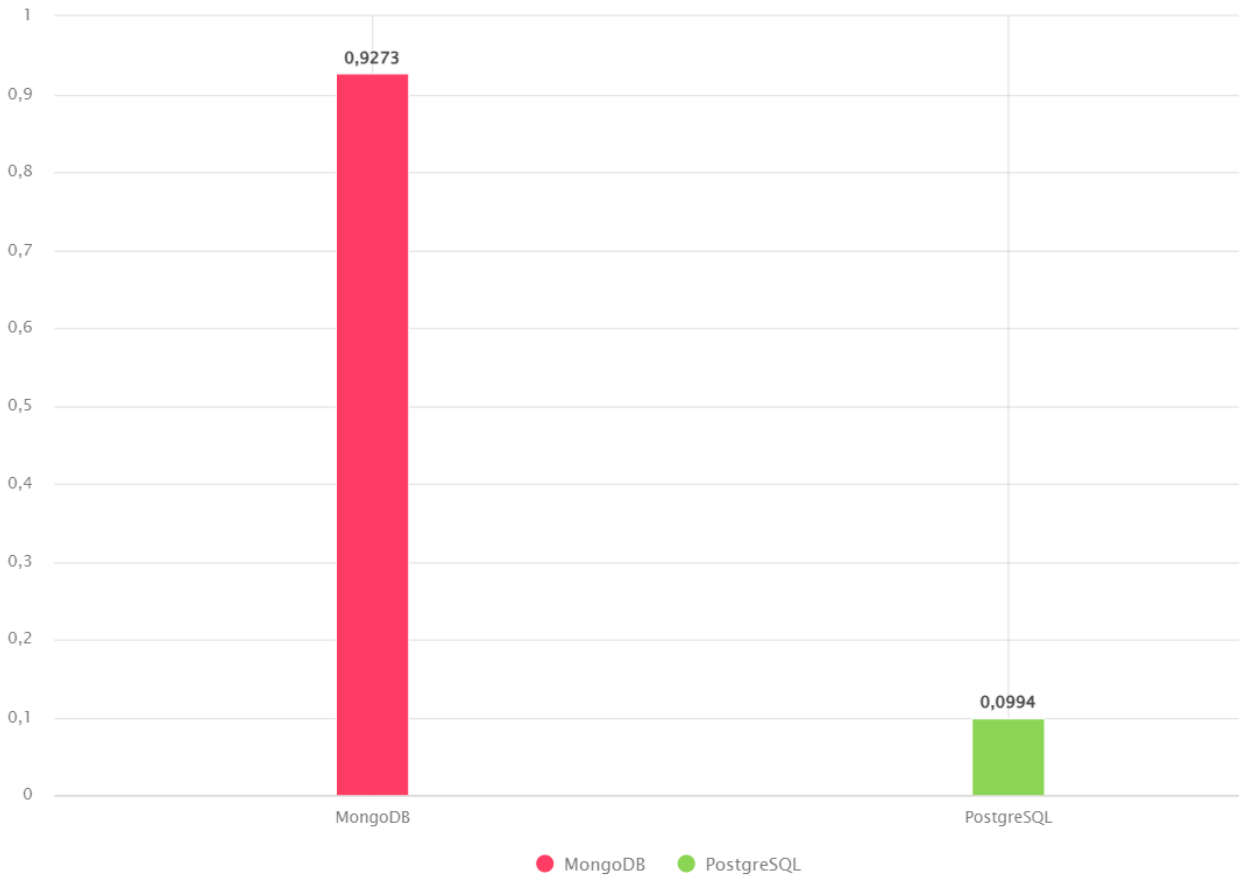
```
Категория: Computers
  Dell XPS 13 9310: 1432 шт.
  Asus ZenBook 14 OLED: 1284 шт.
  MacBook Pro 16 M2 Pro: 1260 шт.
  Lenovo ThinkPad X1 Carbon: 1242 шт.
  MacBook Air M2: 1216 шт.

Категория: Headphones
  Bose QC45: 1344 шт.
  Apple AirPods Pro 2: 1308 шт.
  Sony WH-1000XM4: 1272 шт.
  JBL Live 660NC: 1262 шт.
  Apple AirPods Max: 1202 шт.

Категория: Phones
  Samsung Galaxy S23: 1464 шт.
  iPhone 15 128GB: 1376 шт.
  Samsung Galaxy S23 Ultra: 1314 шт.
  iPhone 14 128GB: 1314 шт.
  iPhone 14 256GB: 1288 шт.

⏱ Время выполнения: 0.9273 секунд
```

Сравнение времени выполнения запросов:



Вывод:

В ходе работы были выполнены одинаковые агрегационные запросы в PostgreSQL и MongoDB для определения 5 самых продаваемых товаров в каждой категории за последний месяц.
Время выполнения запросов составило: PostgreSQL — 0.0994 с, MongoDB — 0.9273с. Результаты показывают, что при структурированных данных и использовании сложных агрегирующих операций PostgreSQL демонстрирует более высокую производительность и лучше подходит для аналитических отчётов. MongoDB остаётся эффективной при работе с гибкими, неструктурированными данными и больших объёмах, но при аналитических задачах с фильтрацией и соединениями её производительность ниже.