

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

«Сравнение скорости CRUD операций (MySQL с PostgreSQL)»
тема

Руководитель

подпись, дата

А. Н. Пупков
инициалы, фамилия

Студент

КИ18-16Б, 031832652
номер группы, зачетной книжки

подпись, дата

А. С. Ядров
инициалы, фамилия

Красноярск 2020

1 Цель работы

Необходимо провести сравнение скорости и анализ CRUD операций для баз данных MySQL и PostgreSQL.

2 Задание

Теоретическая часть:

- Описать что такое CRUD операции;
- Описать, как работают хранилища данных, ссылаясь на соответствующую документацию;
- Найти информацию о том, как и почему скорость CRUD операций хранилищ отличается, провести сравнительный анализ для каждой операции с детальным и обоснованным объяснением;
- Сделать выводы о том, почему в данных хранилищах имеются различия в выполнении CRUD операций, чем это вызвано и как дизайн системы влияет на данный параметр.

Экспериментальная часть:

- Установить docker toolbox (или более свежее решение)
- Скачать контейнеры с соответствующими базами данных;
- Написать два простых скрипта выполняющих CRUD операции для каждой из пары баз данных и измеряющих время выполнения;
- Каждый эксперимент провести несколько раз, при этом:
- Нужно указать параметры (виртуальной) машины, на которой проводились исследования (кол-во RAM, CPU, потоков);
- Указать количество итераций для каждого эксперимента;
- Привести значения математического ожидания и дисперсии для каждого результата;
- Сделать графики с пояснениями;

- Сделать выводы о том, почему в данных хранилищах имеются различия в выполнении CRUD операций, чем это вызвано и как дизайн системы влияет на данный параметр.

3 Теоретический материал

CRUD - акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (*create*), чтение (*read*), модификация (*update*), удаление (*delete*) [1]. Иными словами CRUD - это аббревиатура основных команд для работы с базой данных, без этих команд база данных функционировать не может. Можно также сделать вывод о том, что все действия с информацией, хранящейся в бд, либо являются производными от CRUD, либо и являются одной из функцией CRUD.

Из языков программирования был выбран Python, так как он является высокоуровневым языком и богат на всякого рода библиотек. Таких как `datetime` [2] (работа со временем), `mysql` [3] (работа с MySQL), `psycopg2` [4] (работа с PostgreSQL).

Так как некоторого следующего материала не было в русскоязычной форме, для перевода был использован Гугл-переводчик [5].

MySQL – свободная реляционная система управления базами данных [6]. Программное обеспечение MySQL™ предоставляет очень быстрый, многопоточный, многопользовательский и надежный сервер базы данных SQL (язык структурированных запросов). Сервер MySQL предназначен для критически важных, высоконагруженных производственных систем, а также для встраивания в массовое программное обеспечение [7].

Поскольку MySQL – это реляционная база данных, то данные хранятся в виде таблиц, имеющих атрибуты и кортежи.

В работе были использован функционал БД MySQL соответствующей акрониму CRUD:

- INSERT – добавление строки в БД [8];

- SELECT – извлечение строк из БД [9];
- UPDATE — обновление столбцов в соответствии с их новыми значениями в строках существующей БД [10];
- DELETE - удаление строк из БД [11].

PostgreSQL - свободная реляционная система управления базами данных [12].

PostgreSQL - это мощная объектно-реляционная база данных с открытым исходным кодом, которая использует и расширяет язык SQL в сочетании с множеством функций, которые безопасно хранят и масштабируют самые сложные рабочие нагрузки данных.

PostgreSQL заработал прочную репутацию благодаря своей проверенной архитектуре, надежности, целостности данных, надежному набору функций, расширяемости и приверженности сообщества открытого исходного кода, стоящего за программным обеспечением, для последовательной разработки эффективных и инновационных решений [13].

Синтаксис функций CRUD в СУБД PostgreSQL:

- INSERT – запись в БД [14];
- SELECT – чтение записи из БД [15];
- UPDATE - изменение записи в БД [16];
- DELETE - удаление записи из БД [17].

В качестве хранилища контейнеров использовался Docker.

Docker - программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любую Linux-систему с поддержкой cgroups в ядре, а также предоставляет среду по управлению контейнерами [18].

Собственно в качестве контейнера выступают сами БД (MySQL и PostgreSQL).

4 Ход работы

Для начала работы было необходимо установить Docker (Рисунок 1.1, 1.2).



Рисунок 1.1 – Установка Docker.

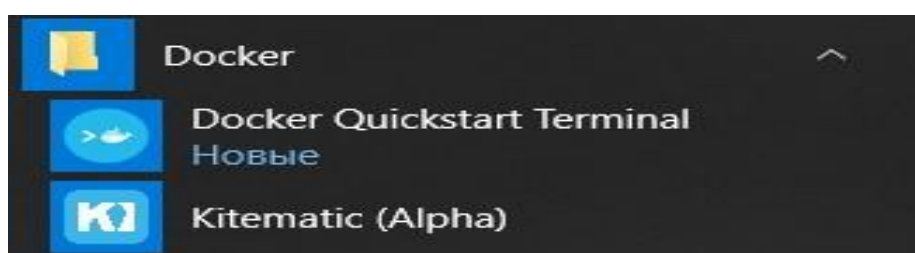


Рисунок 1.2 – Результат установки Docker.

После установки нужно развернуть контейнеры с БД (Рисунок 2.1, 2.2, 2.3).

```
User@LAPTOP-CP5TGHOG MINGW64 /c/Program Files/Docker Toolbox
$ docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
bf5952930446: Already exists
8254623a9871: Pull complete
938e3e06dac4: Pull complete
ea28ebf28884: Pull complete
f3cef38785c2: Pull complete
894f9792565a: Pull complete
1d8a57523420: Pull complete
6c676912929f: Pull complete
fff39fdb566b4: Pull complete
fff872988aba: Pull complete
4d34e365ae68: Pull complete
7886ee20621e: Pull complete
Digest: sha256:c358e72e100ab493a0304bda35e6f239db2ec8c9bb836d8a427ac34307d074ed
Status: Downloaded newer image for mysql:latest

User@LAPTOP-CP5TGHOG MINGW64 /c/Program Files/Docker Toolbox
$ docker run --name my-mysql -e MYSQL_ROOT_PASSWORD=qwerty -d mysql
0a2b8537e167a7b131136f1ee045f9c456a64dce0545d4d36976cf09ef794aab

User@LAPTOP-CP5TGHOG MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
0a2b8537e167	mysql	"docker-entrypoint.s..."	3 seconds ago	Up 4 seconds	3306/tcp, 33060/tcp	my-mysql

Рисунок 2.1 – Скачивание и развертывание MySQL.

```
User@LAPTOP-CP5TGHOG MINGW64 /c/Program Files/Docker Toolbox
$ docker run --name my_postgres -e POSTGRES_PASSWORD=qwerty -d postgres
6dde1de1a06a6d0f6d76d6c11d0e3418642e49d32575adfffcdf6b1a72cf995c

User@LAPTOP-CP5TGHOG MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6dde1de1a06a	postgres	"docker-entrypoint.s..."	2 seconds ago	Up 2 seconds	5432/tcp	my_postgres

Рисунок 2.2 – PostgreSQL.

Ключи (Рисунок 2.1, 2.2):

- **-d** – запустить контейнер в отдельном режиме;
- **-e** – пароль контейнера;
- **--name** – название контейнера.

```
User@LAPTOP-CP5TGHOG MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
0a2b8537e167	mysql	"docker-entrypoint.s..."	11 minutes ago	Up 11 minutes	3306/tcp, 33060/tcp	my-mysql
6dde1de1a06a	postgres	"docker-entrypoint.s..."	41 minutes ago	Up 3 seconds	5432/tcp	my_postgres

Рисунок 2.3 – Результат разворота контейнеров.

Проверим работоспособность баз данных (Рисунок 3.1, 3.2).

```
mysql> CREATE DATABASE test;
Query OK, 1 row affected (0.01 sec)

mysql> USE test;
Database changed
mysql> CREATE TABLE students (id INTEGER);
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO students VALUES (1), (2), (3);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM students;
+-----+
| id    |
+-----+
| 1     |
| 2     |
| 3     |
+-----+
3 rows in set (0.00 sec)

mysql> UPDATE students SET id = 3 WHERE id = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM students;
+-----+
| id    |
+-----+
| 3     |
| 2     |
| 3     |
+-----+
3 rows in set (0.00 sec)

mysql> DROP TABLE students;
Query OK, 0 rows affected (0.02 sec)
```

Рисунок 3.1 – Основные команды БД MySQL.


```

User@LAPTOP-CP5TGH0G MINGW64 /c/Program Files/Docker Toolbox
$ docker exec -it my_postgres bash
root@6dde1de1a06a:/# su - postgres
postgres@6dde1de1a06a:~$ psql
psql (12.4 (Debian 12.4-1.pgdg100+1))
Type "help" for help.

postgres=# CREATE TABLE students (id INTEGER);
CREATE TABLE
postgres=# INSERT INTO students VALUES (1), (2);
INSERT 0 2
postgres=# SELECT * FROM students;
 id
----
  1
  2
(2 rows)

postgres=# DELETE FROM students WHERE id = 1;
DELETE 1
postgres=# SELECT * FROM students;
 id
----
  2
(1 row)

postgres=# DROP TABLE students;
DROP TABLE
postgres=#

```

Рисунок 3.2 – Основные команды БД PostgreSQL.

Далее нужно написать «простые скрипты» которые будут выполнять операции CRUD (Листинг 1, 2).

Листинг 1 - Скрипт для MySQL

```

# Библиотека для работы со временем
from datetime import datetime
import time

# Библиотека для работы с mysql
import pymysql

# Подключиться к базе данных.
connection = pymysql.connect(host='127.0.0.1',
                             port=3306,

```



```

        user='root',
        password='qwerty',
        db='test')

print("connect successful!!")
cursor = connection.cursor()

def runC(version, count):
    # Происходит запись в БД count раз
    # вернет время выполнения скрипта
    start_time = datetime.now()
    for x in range(count):
        buf = "valueV" + str(version) + "_" + str(x)
        cursor.execute("INSERT INTO test (id) VALUES (%s)",
str(buf))
    return datetime.now() - start_time

def runR(version, count):
    # Происходит запись в БД count раз
    # вернет время выполнения скрипта
    start_time = datetime.now()
    for x in range(count):
        buf = "valueV" + str(version) + "_" + str(x)
        cursor.execute("SELECT id FROM test WHERE id = %s",
str(buf))
    return datetime.now() - start_time

def runU(version, count):
    # Происходит изменение значений в БД, count записей
    # вернет время выполнения скрипта
    start_time = datetime.now()
    for x in range(count):
        buf = "valueV" + str(version) + "_" + str(x)
        buf1 = "valueV" + str(version) + "_" + str(count - x)
        cursor.execute("UPDATE test SET id = %s WHERE id = %s",
(str(buf1), str(buf)))
    return datetime.now() - start_time

def runD(version, count):

```

```

# Происходит удаление count записей из БД
# вернет время выполнения скрипта
start_time = datetime.now()
for x in range(count):
    buf1 = "valueV" + str(version) + "_" + str(count - x)
    cursor.execute("DELETE from test WHERE ID = (%s)",
str(buf1))
    return datetime.now() - start_time

```

Листинг 2 — Скрипт для PostgreSQL

```

# Библиотека для работы со временем
from datetime import datetime
import time

# Библиотека для работы с Postgres
import psycopg2

# Подключение в БД
conn = psycopg2.connect(dbname='postgres', user='postgres',
                        password='qwerty', host='localhost')

cursor = conn.cursor()

cursor.execute("CREATE TABLE TEST(ID TEXT)")
def runC(version, count):
    # Происходит запись в БД count раз
    # вернет время выполнения скрипта
    start_time = datetime.now()
    for x in range(count):
        buf = "valueV" + str(version) + "_" + str(x)
        cursor.execute("INSERT INTO test (ID) VALUES (%s)",
(str(buf))
        return datetime.now() - start_time

def runR(version, count):
    # Происходит запись в БД count раз
    # вернет время выполнения скрипта
    start_time = datetime.now()

```

```

        for x in range(count):
            buf = "valueV" + str(version) + "_" + str(x)
            cursor.execute("SELECT ID from test WHERE ID=(%s)",
(str(buf))
            return datetime.now() - start_time

def runU(version, count):
    # Происходит изменение значений в БД, count записей
    # вернет время выполнения скрипта
    start_time = datetime.now()
    for x in range(count):
        buf = "valueV" + str(version) + "_" + str(x)
        buf1 = "valueV" + str(version) + "_" + str(count - x)
        cursor.execute("UPDATE test SET ID=(%s) where ID=(%s)",
(str(buf1), str(buf)))
    return datetime.now() - start_time

def runD(version, count):
    # Происходит удаление count записей из БД
    # вернет время выполнения скрипта
    start_time = datetime.now()
    for x in range(count):
        buf1 = "valueV" + str(version) + "_" + str(count - x)
        cursor.execute("DELETE from test WHERE ID = (%s)",
(str(buf1))
    return datetime.now() - start_time

```

Теперь все готово для проведения эксперимента. На момент эксперимента характеристики машины таковы (Рисунок 5.1).

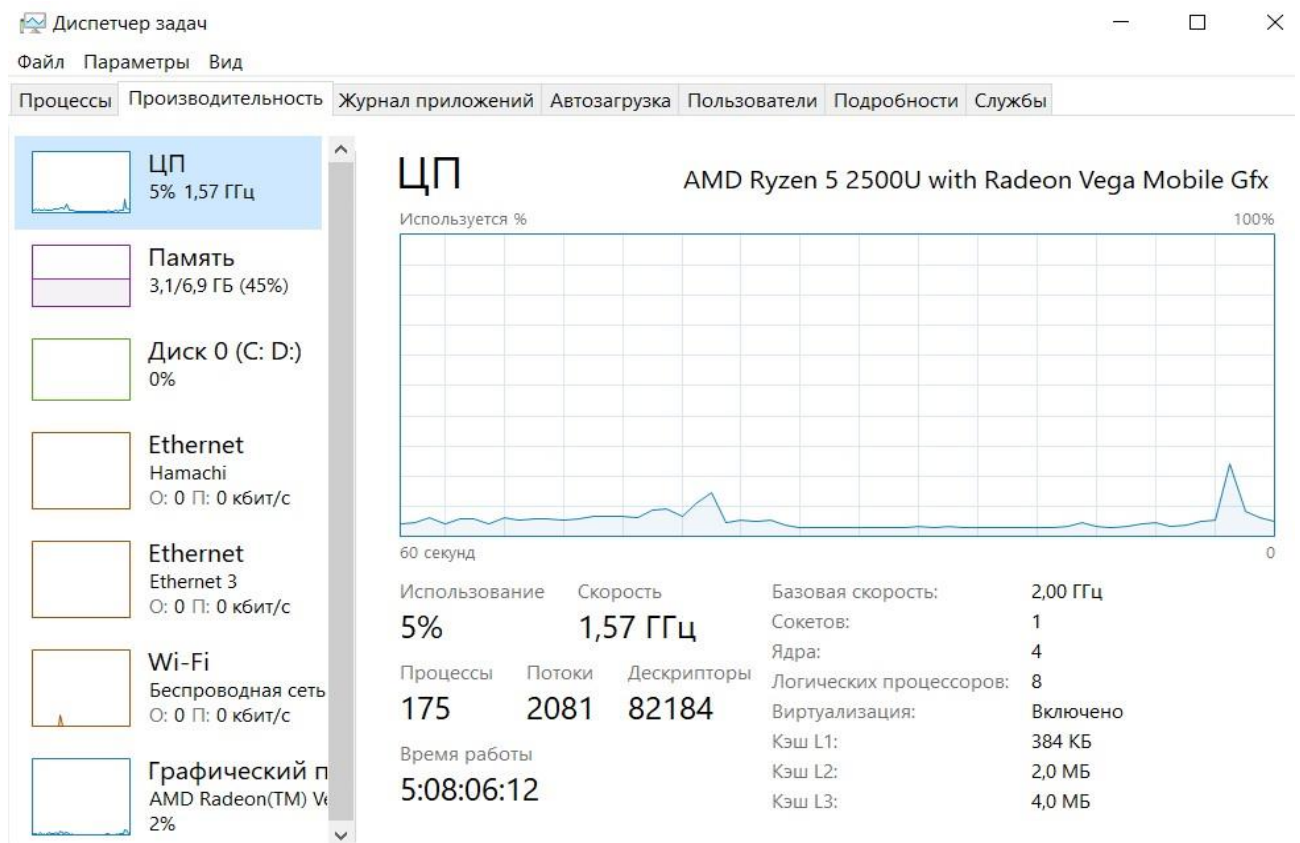
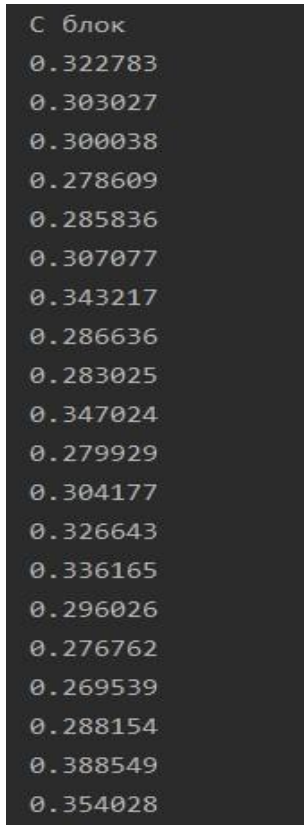


Рисунок 5.1 – Характеристики

Было выбрано 20000 записей и 20шт. итераций, то есть каждая итерация будет обрабатываться по 1000 записей. Все измерения измеряются в секундах.

Запустим скрипт для MySQL (Рисунок 6.1, 6.2, 6.3, 6.4).



С блок
0.322783
0.303027
0.300038
0.278609
0.285836
0.307077
0.343217
0.286636
0.283025
0.347024
0.279929
0.304177
0.326643
0.336165
0.296026
0.276762
0.269539
0.288154
0.388549
0.354028

Рисунок 6.1 – Блок операций С.

R блок

44.440802
42.202406
42.093025
42.186779
42.249276
42.327401
42.843016
42.668798
41.796164
41.077417
41.889907
42.343025
42.264902
42.452381
42.40897
42.327401
42.35865
42.233654
45.008248
43.186775

Рисунок 6.2 – Блок операций R.

U блок
52.45221
52.655349
52.905345
53.16849
52.7284
53.217844
53.077211
53.374087
53.124091
52.96785
53.332728
53.749102
48.499655
35.858766
36.171256
36.202505
35.890011
36.10875
39.620522
39.249331

Рисунок 6.3 – Блок операций U.


```
D блок
34.374431
31.280715
30.030735
28.577636
27.218286
25.827676
24.155841
22.968353
21.515264
20.421525
18.828809
17.484074
15.749709
14.640392
13.218529
11.937289
10.374829
 9.054231
 7.781101
 6.468666
```

Рисунок 6.4 – Блок операций D.

Запустим скрипт для PostgreSQL (Рисунок 6.5, 6.6, 6.7, 6.8).

```
С блок
0.111673
0.129303
0.127341
0.205709
0.208656
0.194921
0.174362
0.090119
0.105792
0.237058
0.261544
0.152809
0.118531
0.10383
0.103834
0.119508
0.127341
0.121466
0.132245
0.142036
```

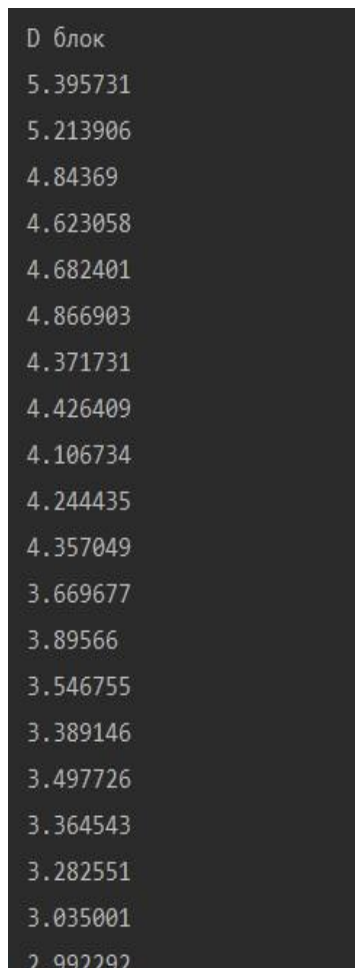
Рисунок 6.5 – Блок операций С.

```
Р блок
3.922832
4.156022
4.370367
4.381246
4.254261
4.243839
4.201958
4.217904
4.179622
4.28791
4.194095
4.245408
4.300978
4.223358
4.207523
4.281691
4.206194
4.218934
4.130939
4.101981
```

Рисунок 6.6 – Блок операций R.

U блок
4.245316
4.301815
4.433276
4.550583
4.514808
4.828357
4.509994
4.882941
4.857773
5.063787
5.06663
4.685981
4.727102
5.008822
5.095623
4.649362
4.447575
5.431504
5.260085
5.34113

Рисунок 6.7 – Блок операций U.



D блок
5.395731
5.213906
4.84369
4.623058
4.682401
4.866903
4.371731
4.426409
4.106734
4.244435
4.357049
3.669677
3.89566
3.546755
3.389146
3.497726
3.364543
3.282551
3.035001
2.992292

Рисунок 6.8 – Блок операций D.

Переведем полученные данные в таблицу (Таблица 1.1, 1.2). И заодно вычислим мат. ожидание и дисперсию.

MySQL					
Итер./этап	C, сек	R, сек	U, сек	D, сек	Кол-во записей, шт,
1	0,3228	44,4408	52,4522	34,3744	1000
2	0,3030	42,2024	52,6553	31,2807	1000
3	0,3000	42,0930	52,9053	30,0307	1000
4	0,2786	42,1868	53,1685	28,5776	1000
5	0,2858	42,2493	52,7284	27,2183	1000
6	0,3071	42,3274	53,2178	25,8277	1000
7	0,3432	42,8430	53,0772	24,1558	1000
8	0,2866	42,6688	53,3741	22,9684	1000
9	0,2830	41,7962	53,1241	21,5153	1000
10	0,3470	41,0774	52,9679	20,4215	1000
11	0,2799	41,8899	53,3327	18,8288	1000
12	0,3042	42,3430	53,7491	17,4841	1000
13	0,3266	42,2649	48,4997	15,7497	1000
14	0,3362	42,4524	35,8588	14,6404	1000
15	0,2960	42,4090	36,1713	13,2185	1000
16	0,2768	42,3274	36,2025	11,9373	1000
17	0,2695	42,3587	35,8900	10,3748	1000
18	0,2882	42,2337	36,1088	9,0542	1000
19	0,3885	45,0082	39,6205	7,7811	1000
20	0,3540	43,1868	39,2493	6,4687	1000
Сумма, х	6,1772	850,3590	944,3535	391,9081	20000
Сумма, х^2	1,926950617	36169,73805	45748,47779	9009,35509	
Мат. ожидание, сек	0,3089	42,5179	47,2177	19,5954	
ЗРСВ, сек	0,096347531	1808,486902	2287,42389	450,467755	
Дисперсия, сек	0,0010	0,7108	57,9150	66,4879	

Таблица 1.1 – Результаты работы для MySQL.

PostgreSQL					
Итер./этап	C, сек	R, сек	U, сек	D, сек	Кол-во записей, шт,
1	0,1117	3,9228	4,2453	5,3957	1000
2	0,1293	4,1560	4,3018	5,2139	1000
3	0,1273	4,3704	4,4333	4,8437	1000
4	0,2057	4,3812	4,5506	4,6231	1000
5	0,2087	4,2543	4,5148	4,6824	1000
6	0,1949	4,2438	4,8284	4,8669	1000
7	0,1744	4,2020	4,5100	4,3717	1000
8	0,0901	4,2179	4,8829	4,4264	1000
9	0,1058	4,1796	4,8578	4,1067	1000
10	0,2371	4,2879	5,0638	4,2444	1000
11	0,2615	4,1941	5,0666	4,3570	1000
12	0,1528	4,2454	4,6860	3,6697	1000
13	0,1185	4,3010	4,7271	3,8957	1000
14	0,1038	4,2234	5,0088	3,5468	1000
15	0,1038	4,2075	5,0956	3,8915	1000
16	0,1195	4,2817	4,6494	3,4977	1000
17	0,1273	4,2062	4,4476	3,3645	1000
18	0,1215	4,2189	5,4315	3,2826	1000
19	0,1322	4,1309	5,2601	3,0350	1000
20	0,1420	4,1020	5,3411	2,9923	1000
Сумма, х	2,9681	84,3271	95,9025	82,3077	20000
Сумма, x^2	0,48544802177	355,735562934	462,132853	348,1739	
Мат. ожидание, сек	0,1484	4,2164	4,7951	4,1154	
ЗРСВ, сек	0,02427240109	17,7867781467	23,1066426	17,408695	
Дисперсия, сек	0,0022	0,0091	0,1134	0,4723	

Таблица 1.2 – Результаты работы для PostgreSQL.

Сравним полученные результаты с помощью графиков (График 1.1, 1.2, 1.3, 1.4).

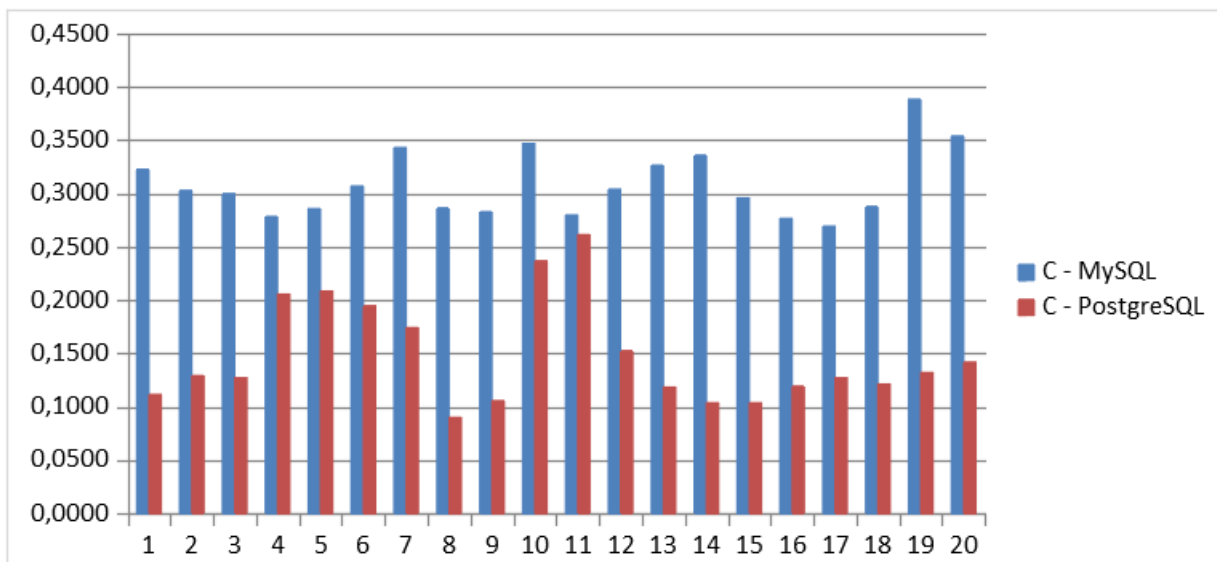
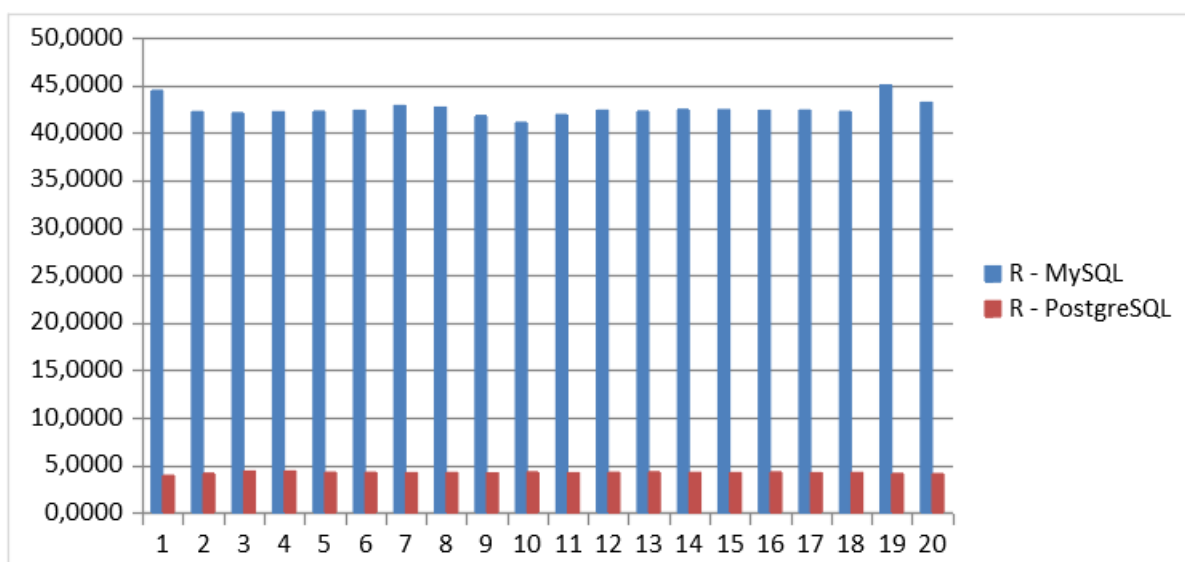


График 1.1 – Сравнение С блоков.



Сравнение R блоков.

График 1.2 – Сравнение

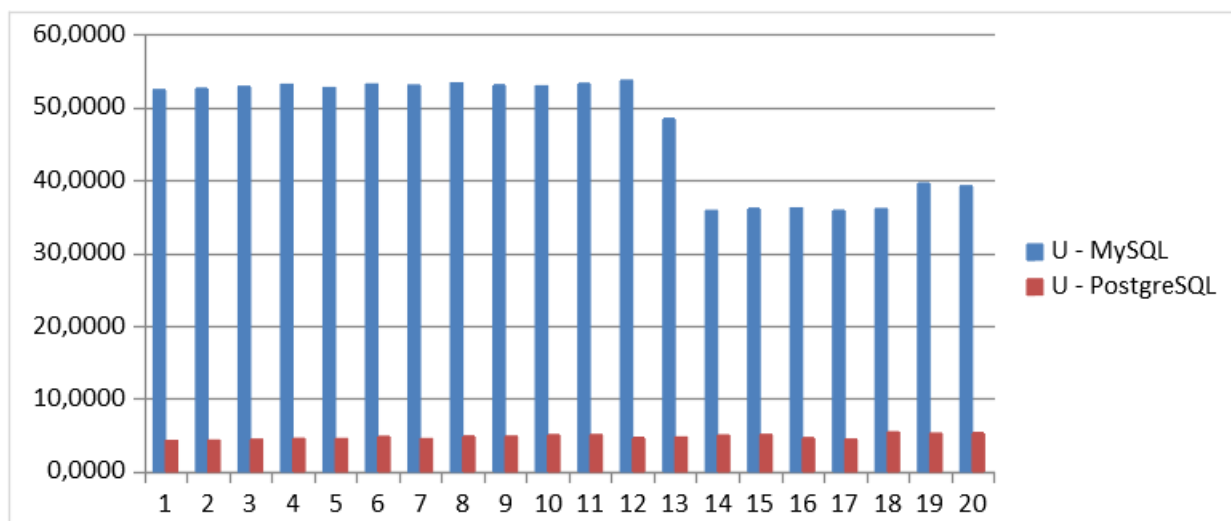


График 1.3 – Сравнение U блоков.

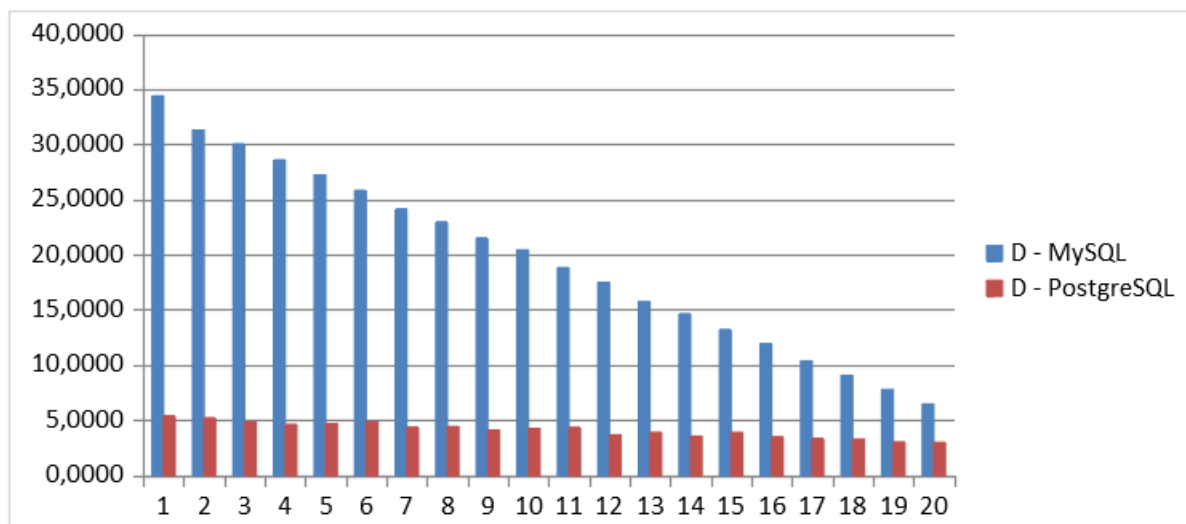


График 1.4 – Сравнение D блоков.

Исходя из графиков можно сделать вывод о том, что PostgreSQL лучше работает с большим объемом данных, нежели MySQL. Наглядно на графике блока D можно увидеть, как при уменьшении данных уменьшается скорость выполнения функций. Следовательно, скорость выполнения операций на MySQL напрямую зависит от кол-ва данных, в то время как скорость PostgreSQL почти не зависит от объема данных. Это может говорить о том, что PostgreSQL лучше подойдет для больших и объемных проектов, где также можно будет применить полный функционал этой СУБД. MySQL в это же время лучше подойдет лучше для небольших/средних проектов.

5 Вывод

Можно подвести итоги. PostgreSQL обрабатывает CRUD быстрее, о чем можно теперь судить. Эта СУБД больше подойдет для больших проектов с большим объемом данных, где эта постоянная скорость даст о себе знать. Ведь даже 0.1 секунда задержки при постоянных запросах за год может сыграть в убыток времени даже в 5 лет, а то и больше.

MySQL будет правильнее использовать в небольших/средних проектах, где не так важен огромный функционал, который присутствует в PostgreSQL

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. CRUD: [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/CRUD>
2. datetime: [Электронный ресурс]. URL: <https://docs.python.org/3/library/datetime.html>
3. PyMySQL: [Электронный ресурс]. URL: <https://python-scripts.com/pymysql>
4. psycopg2 2.8.5: [Электронный ресурс]. URL: <https://pypi.org/project/psycopg2/>
5. Гугл-переводчик: [Электронный ресурс]. URL: <https://translate.google.com/?hl=ru>
6. MySQL. Википедия: [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/MySQL>
7. MySQL. General Information: [Электронный ресурс]. URL: <https://dev.mysql.com/doc/refman/8.0/en/introduction.html>
8. INSERT. Язык запросов SQL: [Электронный ресурс]. URL: <https://sql-language.ru/insert.html>
9. Синтаксис оператора SELECT. Справочное руководство по MySQL: [Электронный ресурс]. URL: <http://www.mysql.ru/docs/man/SELECT.html>
10. Синтаксис оператора UPDATE. Справочное руководство по MySQL: [Электронный ресурс]. URL: <http://www.mysql.ru/docs/man/UPDATE.html>
11. Синтаксис оператора DELETE. Справочное руководство по MySQL: [Электронный ресурс]. URL: <http://www.mysql.ru/docs/man/DELETE.html>
12. PostgreSQL. Википедия: [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/PostgreSQL>

13. PostgreSQL: About: [Электронный ресурс]. URL:
<https://www.postgresql.org/about/>
14. SQL - Insert: [Электронный ресурс]. URL:
<https://postgrespro.ru/docs/postgresql/9.4/sql-insert>
15. SQL - Select: [Электронный ресурс]. URL:
<https://postgrespro.ru/docs/postgrespro/9.5/sql-select>
16. SQL - Update: [Электронный ресурс]. URL:
<https://postgrespro.ru/docs/postgresql/9.6/sql-update>
17. SQL - Delete: [Электронный ресурс]. URL:
<https://postgrespro.ru/docs/postgrespro/9.5/sql-delete>
18. Docker: [Электронный ресурс]. URL:
<https://ru.wikipedia.org/wiki/Docker>