**Task 5 (Partitioned Table)**

Папка задания в репозитории:

<https://github.com/artemgur/AdvancedPostgreSQL/tree/master/Task%205%20(Partitioned%20Table)>

Flyway-скрипты хранятся в следующей папке:

<https://github.com/artemgur/AdvancedPostgreSQL/tree/master/flyway/sql>

Flyway-скрипты, относящиеся к этому заданию, имеют номер версии вида 5.\*.

**Создание и заполнение таблиц**

Создание обычной таблицы:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/flyway/sql/V5.0__CreateOrdinaryTable.sql>

Создание partitioned таблицы (PARTITION BY HASH):

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/flyway/sql/V5.1.0__CreatePartitionedTable.sql>

Создание partitions:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/flyway/sql/V5.1.1__CreatePartitions.sql>

Заполнение обычной таблицы:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/flyway/sql/V5.2.0__InsertOrdinary.sql>

Заполнение partitioned table:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/flyway/sql/V5.2.1__InsertPartitioned.sql>

**Измерение времени выполнения SELECT**

В случае обычной таблицы с помощью команды:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/Task%205%20(Partitioned%20Table)/PgBenchSelectOrdinary.sh>

запускался скрипт:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/Task%205%20(Partitioned%20Table)/PgBenchSelectOrdinary.sql>

В случае partitioned table с помощью команды:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/Task%205%20(Partitioned%20Table)/PgBenchSelectPartitioned.sh>

запускался скрипт:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/Task%205%20(Partitioned%20Table)/PgBenchSelectPartitioned.sql>

**Результаты для обычной таблицы:**

* 0.108 ms — latency average
* 9225.000961 — tps (including connections establishing)
* 9225.244841 — tps (excluding connections establishing)

**Результаты для partitioned table:**

* 0.133 ms — latency average
* 7527.160457 — tps (including connections establishing)
* 7527.318383 — tps (excluding connections establishing)

Таким образом, SELECT в случае partitioned table работает медленнее.

**Визуализация распределения данных по partitions**

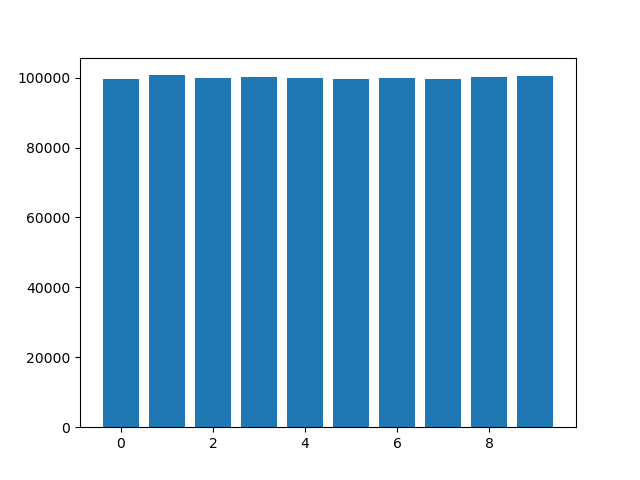
Я использовал Python. С помощью psycopg2 я получил нужные данные и построил график с помощью matplotlib.pyplot.

Python-скрипт:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/Task%205%20(Partitioned%20Table)/PartitionDataDistribution/main.py>

Полученная диаграмма:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/Task%205%20(Partitioned%20Table)/PartitionDataDistribution/Graph.png>



Так как я использовал распределение по хэшу, как и следовало ожидать, все partitions содержат почти равные количества строк.