|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

***\_\_\_\_\_АНАЛИЗ ЭФФЕКТИВНОСТИ ОБРАБОТКИ\_\_\_ ЗАПРОСОВ ПРИ РАЗНЫХ ИНДЕКСАХ ДЛЯ БД\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_ИУ5И-32М\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Гун Шэншо \_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Ю.Е. Гапанюк\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2023 г.*

**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_В.Е. Терехов\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме *АНАЛИЗ ЭФФЕКТИВНОСТИ ОБРАБОТКИ ЗАПРОСОВ ПРИ РАЗНЫХ ИНДЕКСАХ ДЛЯ БД*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_\_\_ИУ5И-32М\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гун Шэншо \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_учебная\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_Кафедра ИУ5\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к \_4\_ нед., 50% к \_8\_ нед., 75% к \_12\_ нед., 100% к \_16\_ нед.

Техническое задание \_Данное исследование посвящено анализу эффективности различных индексов, таких как B-дерево, Хеш, GiST, SP-GiST, GIN и BRIN, в контексте базы данных. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на 26 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**Руководитель НИР**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_Ю.Е. Гапанюк\_\_**

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Гун Шэншо \_\_**

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**Оглавление**

[Введение 4](#_Toc150304593)

[1. Создать базу данных в Pgadmin4 5](#_Toc150304594)

[3.Выполнить к разным таблицам БД три разные запроса и вывести их планы (БЕЗ ИНДЕКСА и С РАЗНЫМИ ИНДЕКСАМИ). 9](#_Toc150304596)

[4. Экспериментальная оценка 21](#_Toc150304600)

[5. Анализ эффективности 24](#_Toc150304601)

[Заключение 25](#_Toc150304602)

[Список литературы 26](#_Toc150304603)

Введение

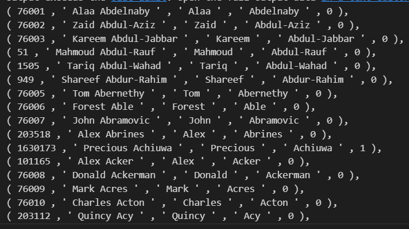
Сегодня все больше сервисов используют запросы к большим базам данных. Примерами таких сервисов являются поисковые системы (например, Яндекс, Google и Baidu), онлайн-сервисы баз данных (например, HUAWEI Cloud и Oracle Database) и сервисы для обработки бизнес-запросов (например, онлайн-регистрация и авторизация). Поэтому эффективное управление большими базами данных является одной из главных проблем в современном мире [1-3].

Высокая скорость отклика и быстрый доступ к данным - это важные критерии при проектировании базы данных. С ростом объема данных скорость ответа на запросы становится одной из главных проблем, которые необходимо решать в системе. Индексирование - это часто наиболее удобный, простой и эффективный способ повышения производительности запросов к базе данных. Большинство проблем с производительностью запросов к базе данных можно решить с помощью индексации.

Производительность системы баз данных непосредственно зависит от скорости запроса и обработки данных. Создание правильного индекса является наиболее распространенным методом настройки базы данных для повышения скорости запроса и обработки данных. Использование индексов может значительно сократить количество операций ввода-вывода с диска, что является важным фактором для определения скорости доступа. Создание правильного и эффективного индекса может существенно повысить производительность базы данных, ускорить запрос данных и уменьшить время отклика системы.

1. Создать базу данных в Pgadmin4

Потом что сначала наши данные сохраняются в виде .csv, поэтому сначала надо преобразовать их в вид, который подходит к созданию базы данных в Pgdamin 4. То в такой вид:



Исходный код для преобразования видов записей:

#%% преобразовать записей таблицы player.csv

import pandas as pd

df = pd.read\_csv(r'C:\Users\Gong Shengshuo\Desktop\Python\1\1.1\player.csv')

# вывести данные в нужном виде

for i in range(len(df)):

    print("(",df.id[i],", '",df.full\_name[i],"' ,","'",df.first\_name[i],"' ,","'",df.last\_name[i],"' ,",df.is\_active[i],'),')

# %%преобразовать записей таблицы officials.csv

import pandas as pd

df = pd.read\_csv(r'C:\Users\Gong Shengshuo\Desktop\Python\1\1.1\officials.csv')

# вывести данные в нужном виде

for i in range(len(df)):

    print("(",df.game\_id[i],",",df.official\_id[i],", '",df.first\_name[i],"' ,","'",df.last\_name[i],"' ,",df.jersey\_num[i],'),')

# %% преобразовать записей таблицы game\_info.csv

import pandas as pd

df = pd.read\_csv(r'C:\Users\Gong Shengshuo\Desktop\Python\1\1.1\game\_info.csv')

# вывести данные в нужном виде

for i in range(len(df)):

    print("(",df.game\_id[i],", '",df.game\_date[i],"' ,","'",df.attendance[i],"' ,","'",df.game\_time[i],"'",'),')

# %% преобразовать записей таблицы common\_player\_info.csv

import pandas as pd

df = pd.read\_csv(r'C:\Users\Gong Shengshuo\Desktop\Python\1\1.1\common\_player\_info.csv')

# вывести данные в нужном виде

for i in range(len(df)):

    print("(",df.person\_id[i],

          ", '",df.first\_name[i],"' ,","'",df.last\_name[i],"' ,","'",df.birthdate[i],"' ,",

          "'",df.school[i],"' ,","'",df.country[i],"' ,","'",df.height[i],"' ,",df.weight[i],',',

          df.season\_exp[i],',',

          "'",df.position[i],"' ,",df.team\_id[i],", '",df.team\_name[i],"' ,","'",df.team\_city[i],"' ,",

          "'",df.playercode[i],"' ,",df.from\_year[i],",",df.to\_year[i],",",

          "'",df.draft\_year[i],"' ,","'",df.draft\_round[i],"'",'),')

# %% преобразовать записей таблицы play\_by\_play.csv

import pandas as pd

df = pd.read\_csv(r'C:\Users\Gong Shengshuo\Desktop\Python\1\1.1\play\_by\_play.csv')

# вывести данные в нужном виде

for i in range(600001,700000):

    print("(","'",df.game\_id[i],"' ,",

          "'",df.eventnum[i],"' ,","'",df.eventmsgtype[i],"' ,","'",df.period[i],"' ,",

          "'",df.wctimestring[i],"' ,",

          "'",df.player1\_id[i],"'",

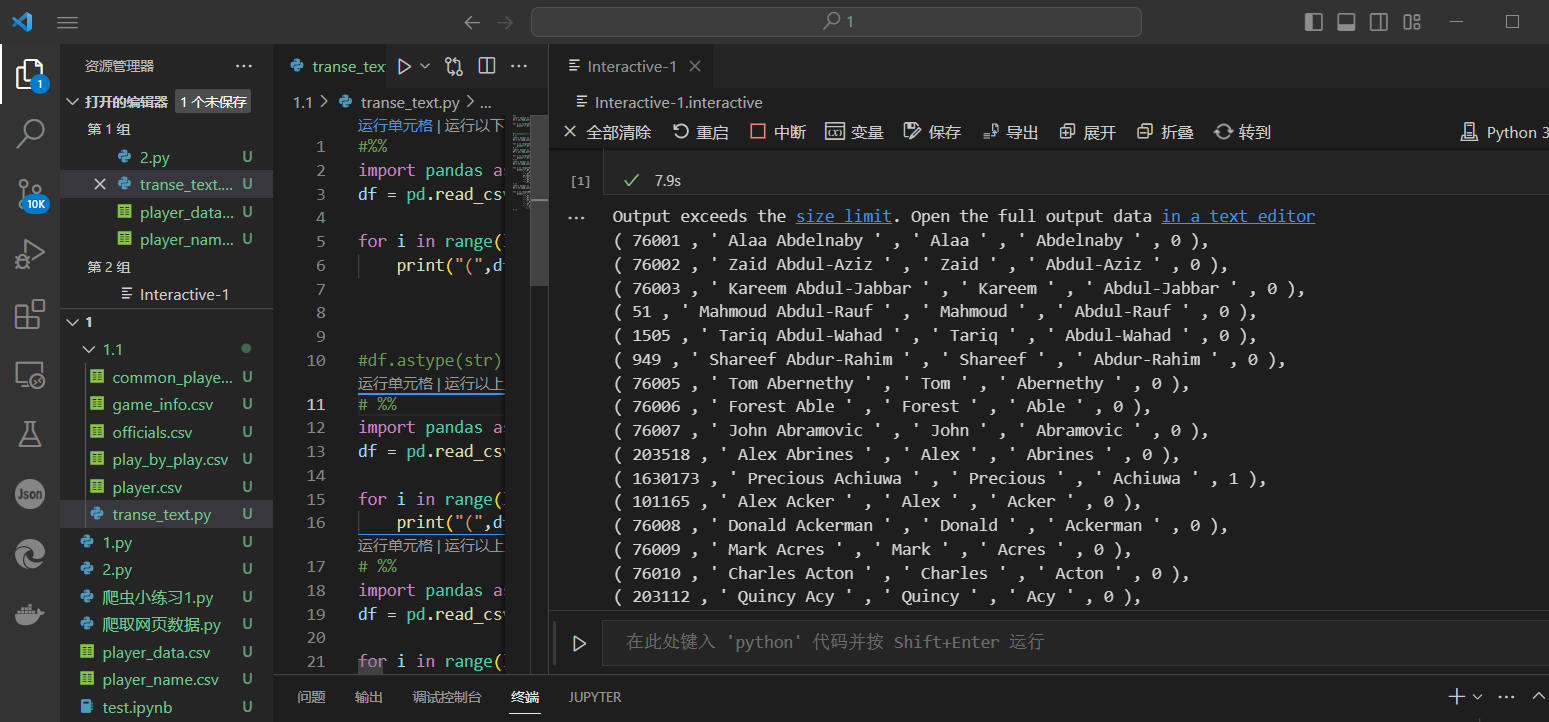
          ", '",df.player1\_name[i],"' ,","'",df.player1\_team\_id[i],"' ,","'",df.player1\_team\_city[i],"' ,","'",df.player1\_team\_nickname[i],"' ,",

          "'",df.player2\_id[i],"'",

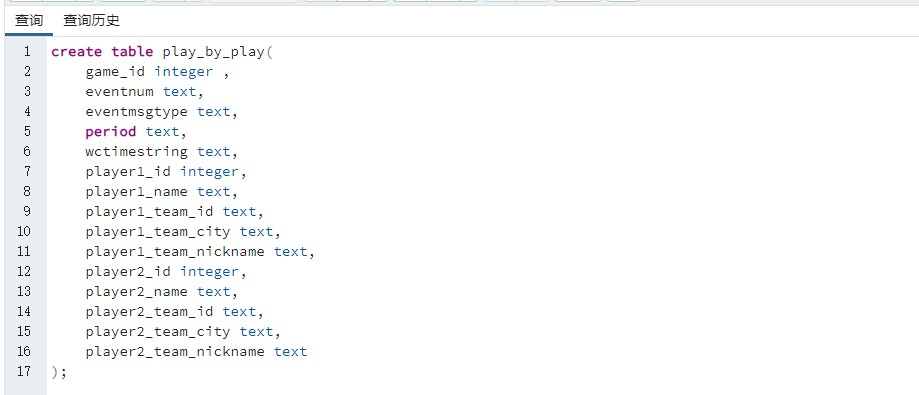
          ", '",df.player2\_name[i],"' ,","'",df.player2\_team\_id[i],"' ,","'",df.player2\_team\_city[i],"' ,","'",df.player2\_team\_nickname[i],"'",

            '),')

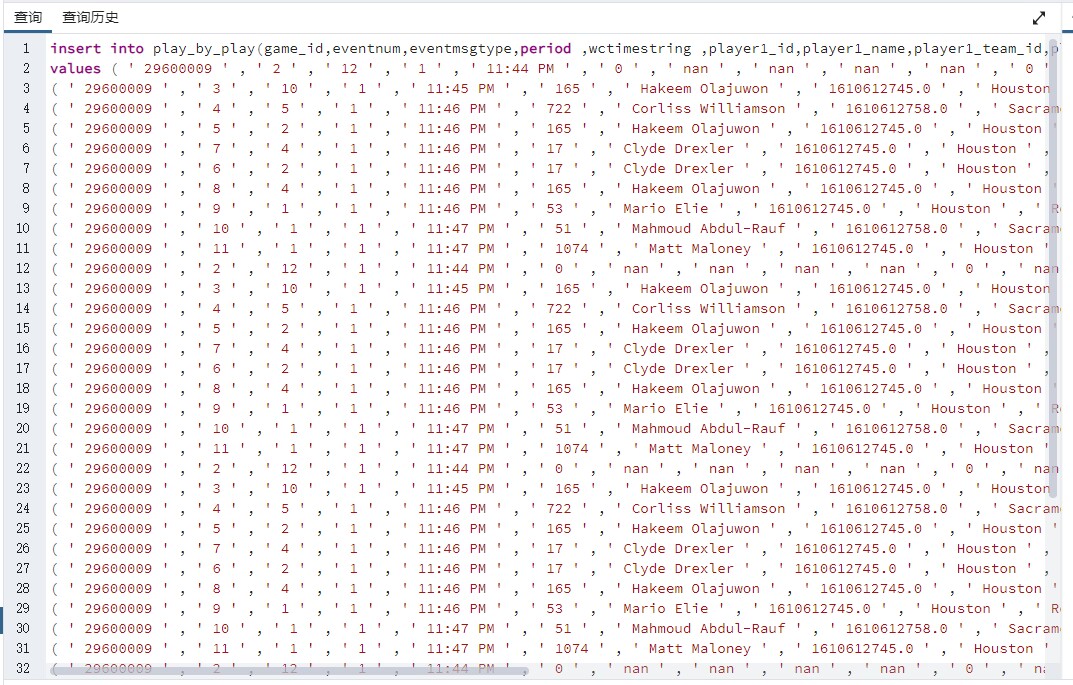
Вид выходных данных:



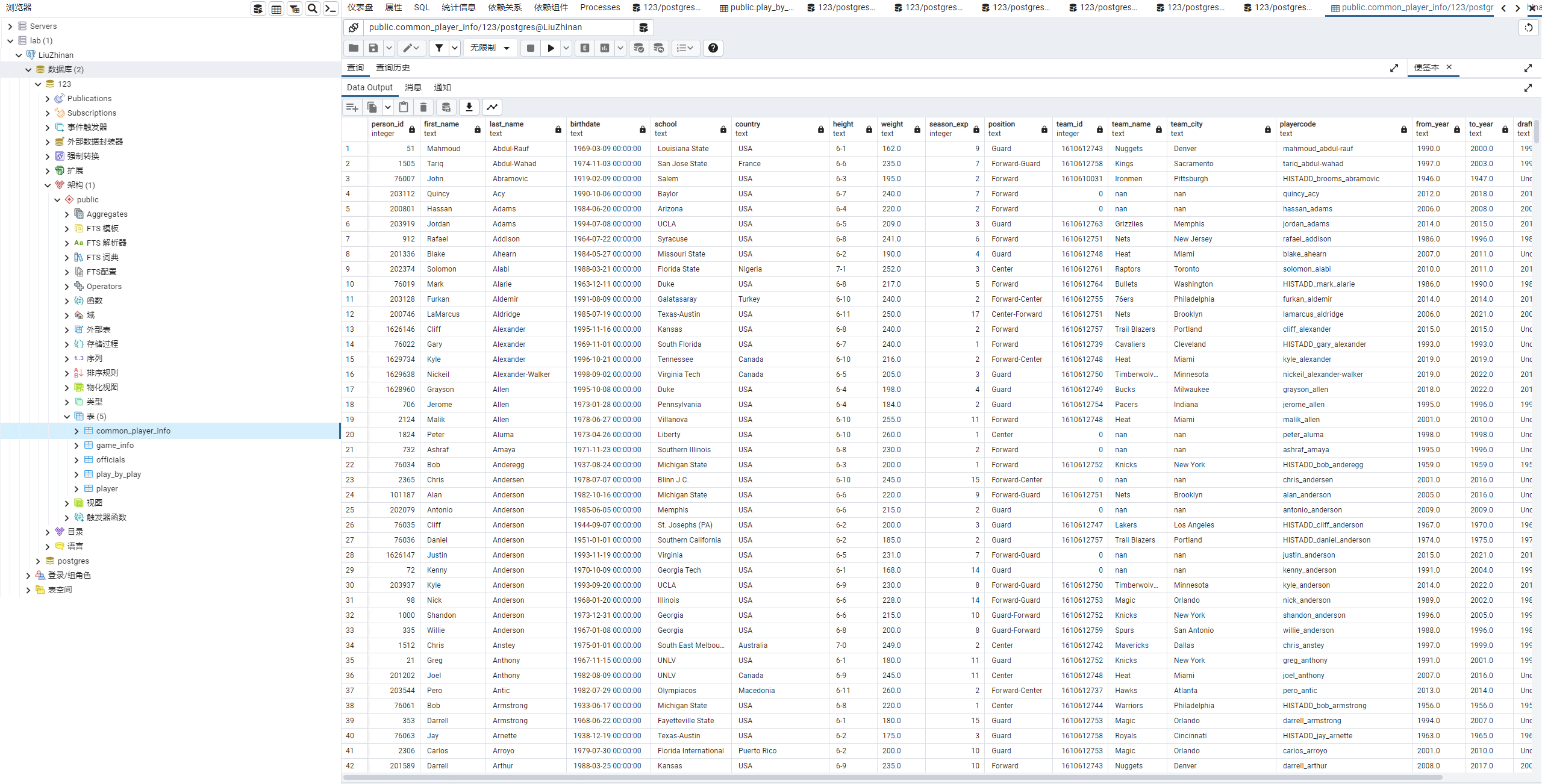
Потом надо создать таблицы в Pgadmin4:



Дальше добавить в нее записи порядком тысячи или сотни тысячи:



Готовая база данных:



**Описание базы данных**

Наша БД состоит из пяти таблиц -- common\_player\_info, game\_info, play\_by\_play, officials, player.

Common\_player\_info состоит из 18 столбцов, включает 4902 записи.

Game\_info состоит из 4 столбцов, включает 60796 записей.

Play\_by\_play состоит из 15 столбцов, включает более миллиона записей.

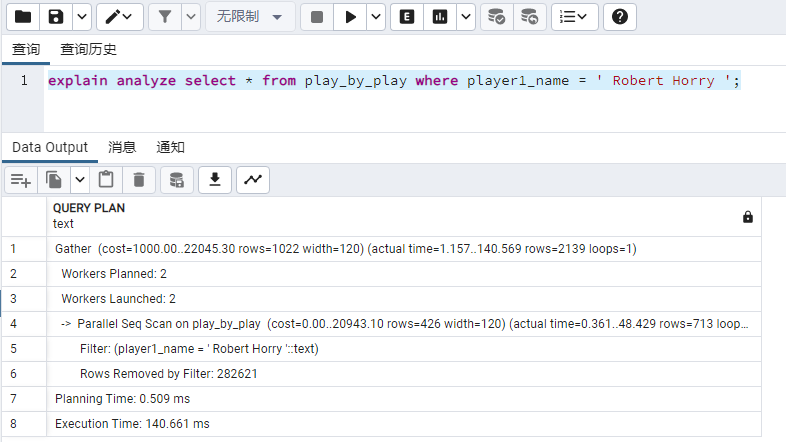
Officials состоит из 5 столбцов, включает 61789 записей.

Player состоит из 5 столбцов, включает 4810 записей.

3.Выполнить к разным таблицам БД три разные запроса и вывести их планы (БЕЗ ИНДЕКСА и С РАЗНЫМИ ИНДЕКСАМИ).

**Первый запрос**

**БЕЗ ИНДЕКСА**



**• B-дерево**

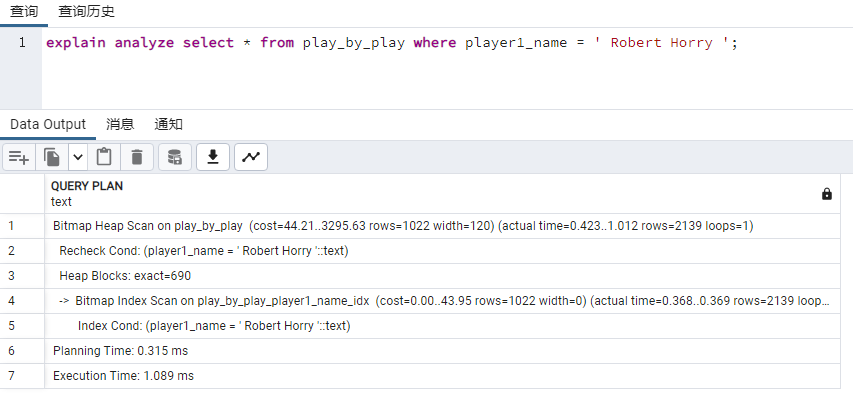
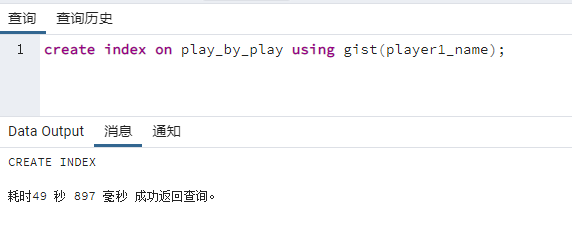


**• Хеш**

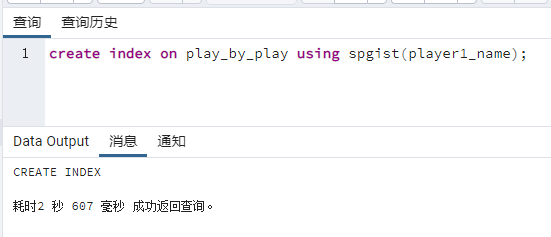


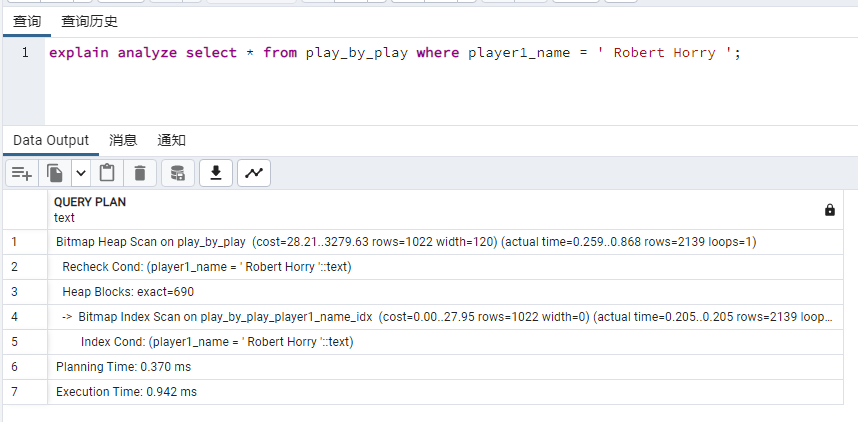


**• GIST**

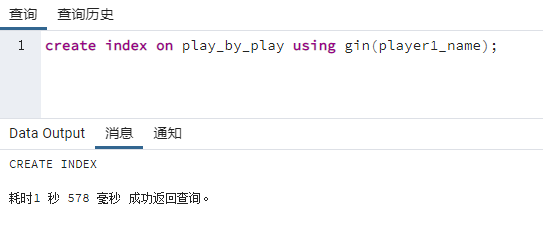


**• SP-GIST**

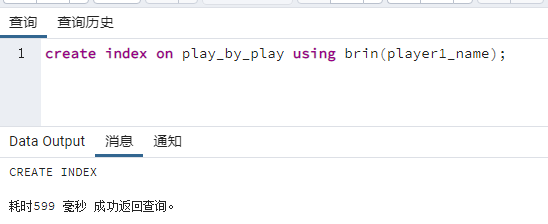




**• GIN**



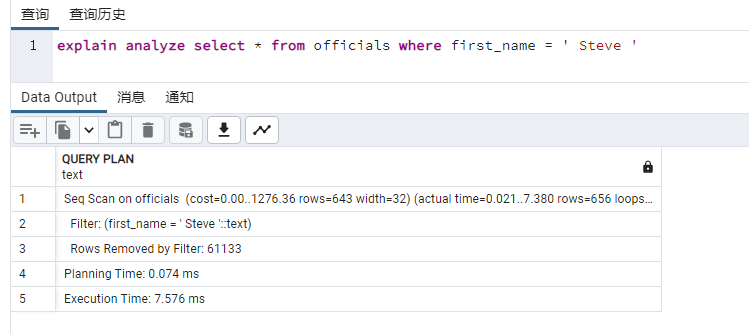
**• BRIN**



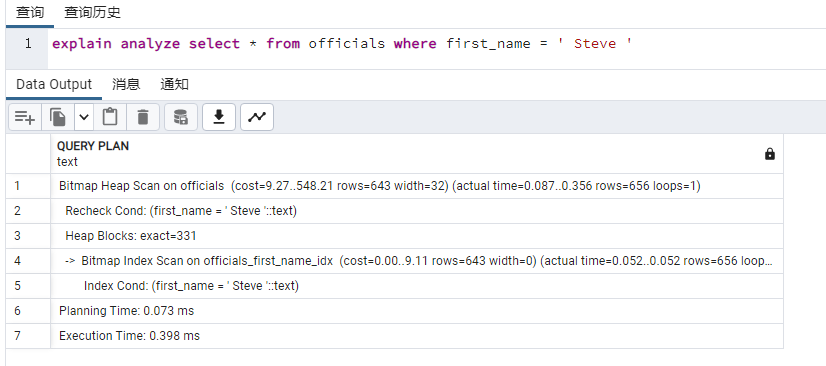
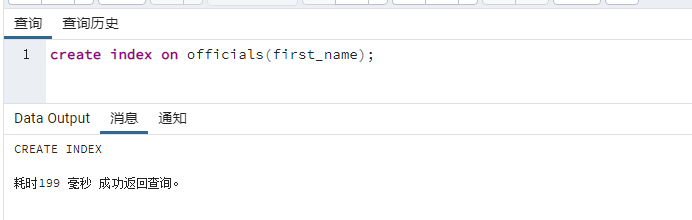


**Второй запрос**

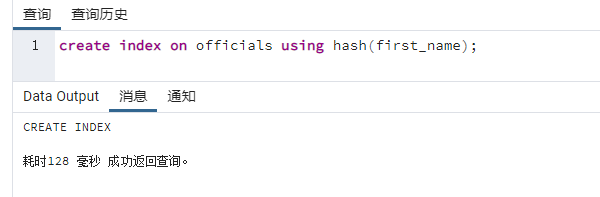
**БЕЗ ИНДЕКСА**

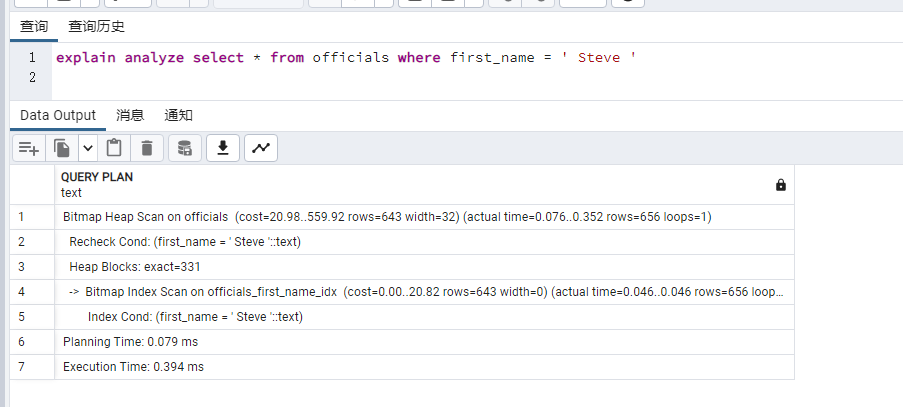


**• B-дерево**

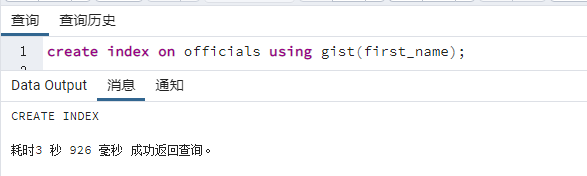
****

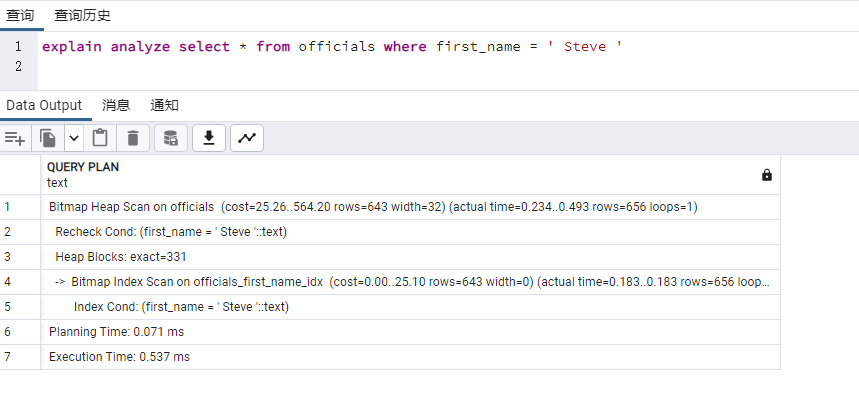
**• Хеш**

****

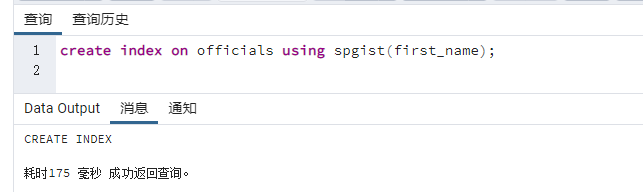
****

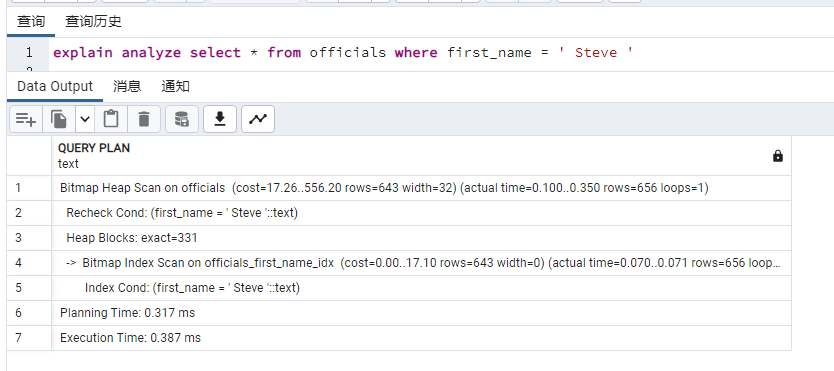
**• GIST**

****

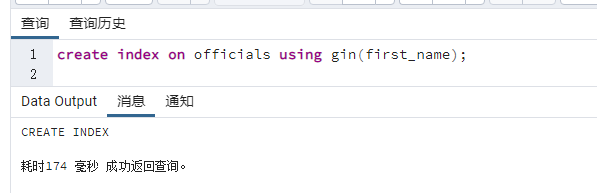
****

**• SP-GIST**

****

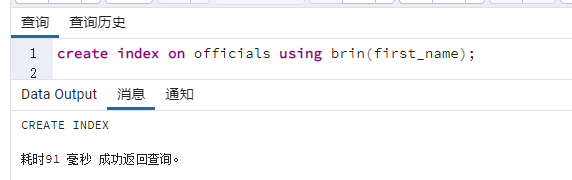
****

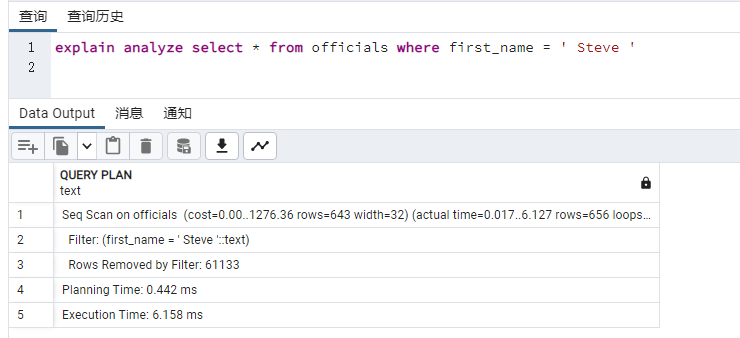
**• GIN**

****

****

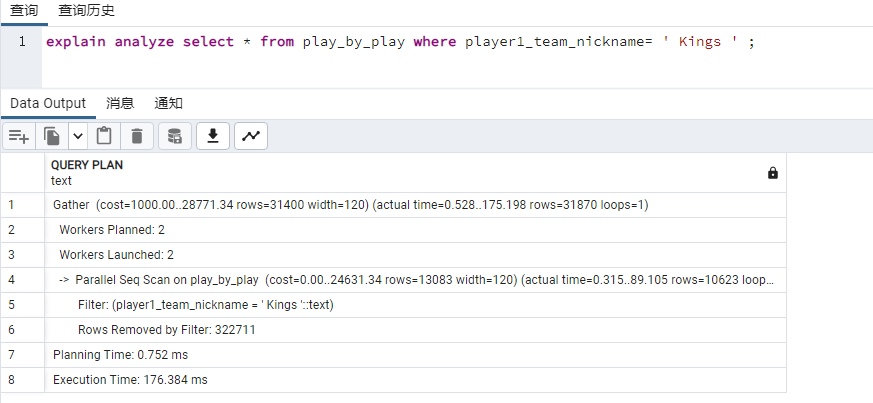
**• BRIN**

****

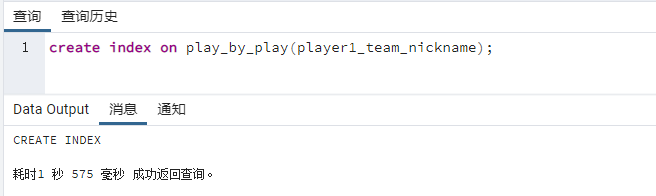
****

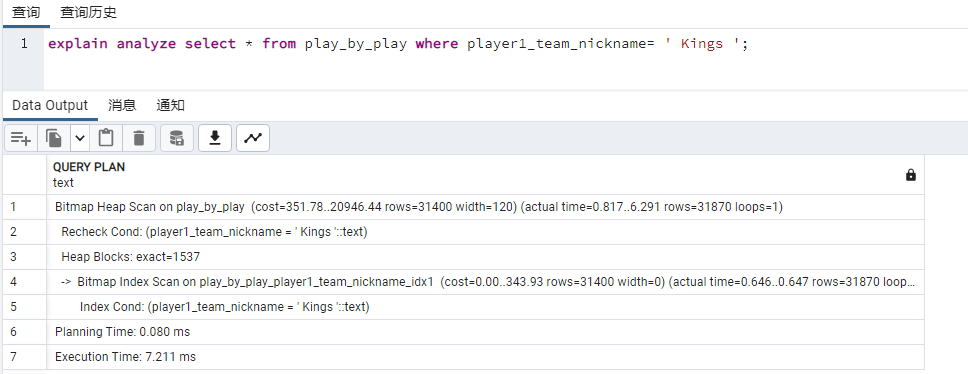
**Третий запрос**

**БЕЗ ИНДЕКСА**

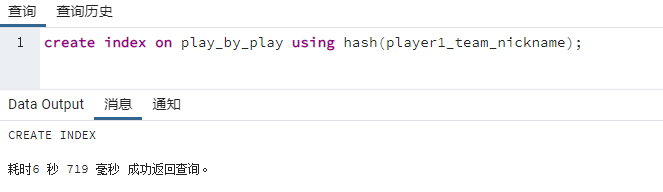


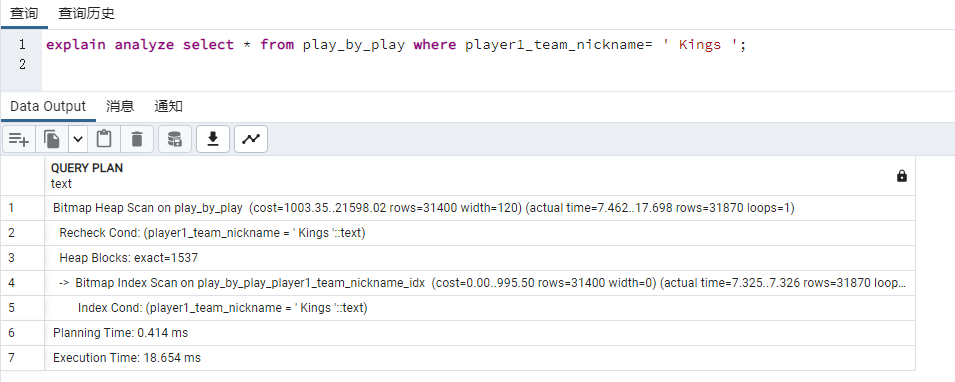
**• B-дерево**



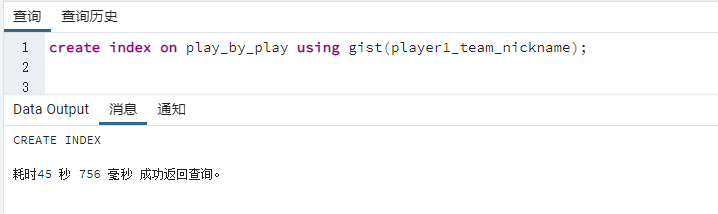


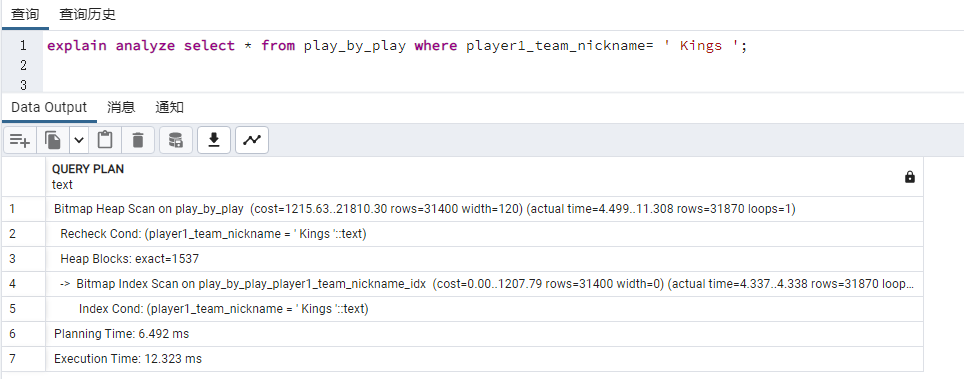
**• Хеш**



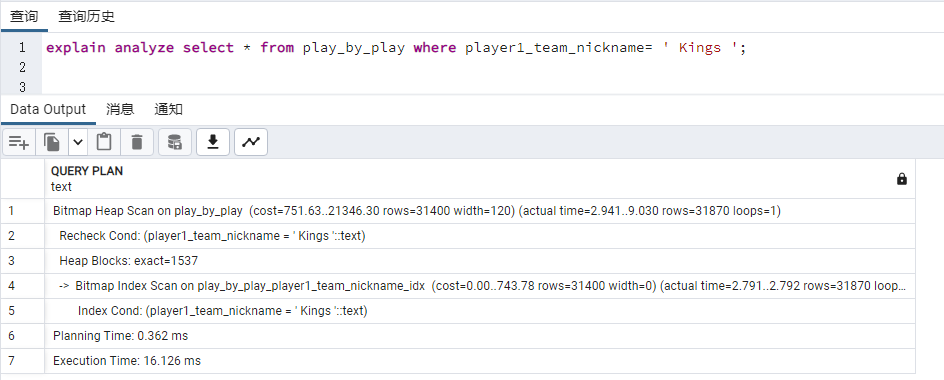
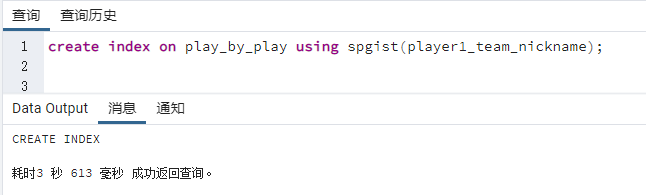


**• GIST**

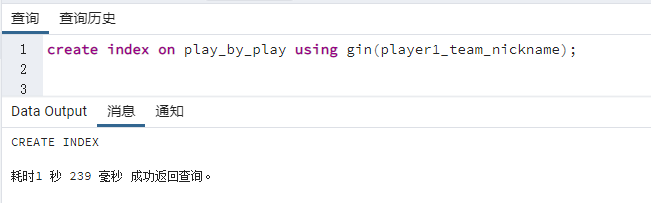


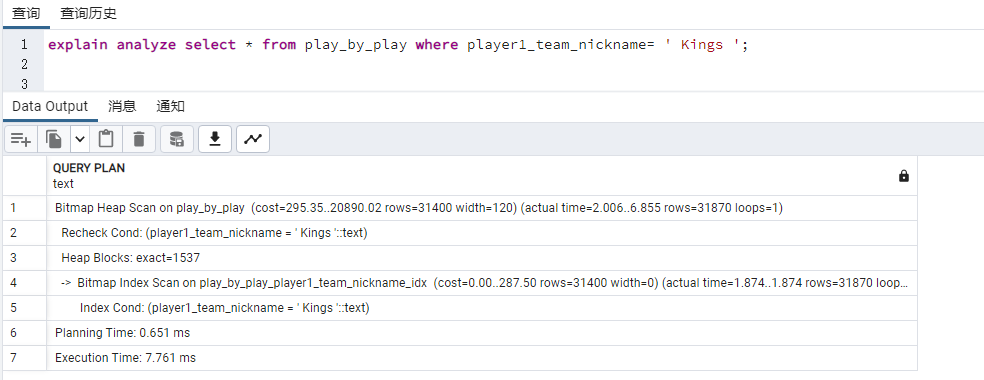


**• SP-GIST**

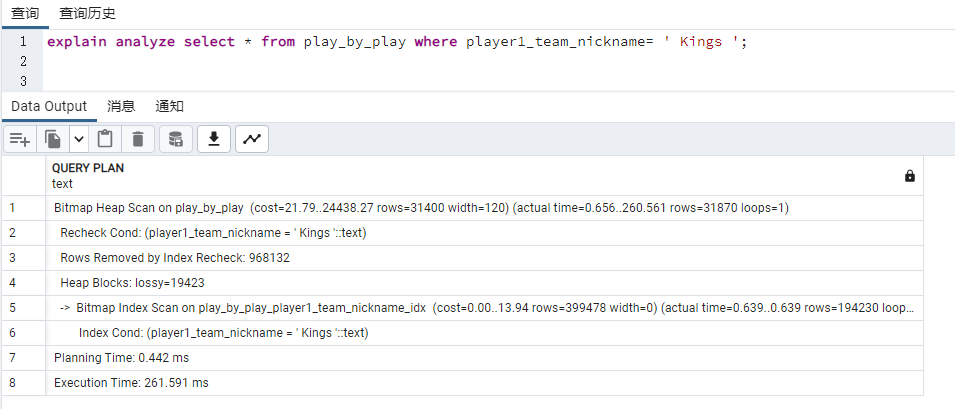
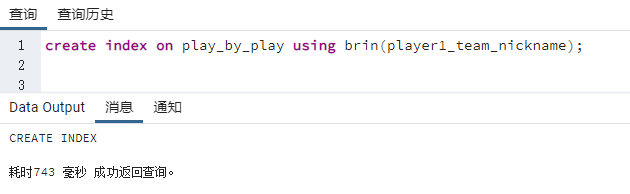


**• GIN**





**• BRIN**



4. Экспериментальная оценка

У нас есть база данных, состоящая из пяти таблиц - common\_player\_info, game\_info, play\_by\_play, officials, player.

Таблица Common\_player\_info состоит из 18 столбцов и включает 4902 записи.

Таблица Game\_info состоит из 4 столбцов и включает 60796 записей.

Таблица Play\_by\_play состоит из 15 столбцов и включает более миллиона записей.

Таблица Officials состоит из 5 столбцов и включает 61789 записей.

Таблица Player состоит из 5 столбцов и включает 4810 записей.

Для проведения исследования мы использовали компьютер i7-7700HQ (2,80 ГГц) с процессором Intel (R) Core (TM), 16 ГБ оперативной памяти и 64-разрядной операционной системой Windows 10. Для выполнения запросов мы использовали утилиту PgAdmin 4. Язык запроса, который мы использовали, - SQL. Все результаты, представленные в таблице, являются средним значением за 20 измерений.

Мы использовали утилиту PgAdmin 4 для создания индексов и выполнения запросов в двух таблицах. Первый запрос: SELECT \* FROM play\_by\_play WHERE player1\_name = 'Robert Horry'. Второй запрос: SELECT \* FROM officials WHERE jersey\_num > 23 AND jersey\_num < 34. В конце мы использовали оператор EXPLAIN ANALYZE, чтобы вывести QUERY PLAN запроса, и получили результаты [10].

Таблица 1. Время выполнения первого запроса, ms

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Без индекса | B-дерево | Хеш | GiST | SP-GiST | GIN | BRIN |
| 1 | 173.126 | 4.609 | 3.282 | 5.746 | 3.804 | 3.586 | 196.938 |
| 2 | 165.101 | 1.072 | 1.334 | 1.469 | 1.251 | 1.127 | 181.639 |
| 3 | 164.645 | 1.120 | 1.277 | 1.288 | 1.212 | 1.056 | 161.869 |
| 4 | 142.084 | 0.977 | 1.198 | 1.561 | 1.139 | 1.124 | 157.087 |
| 5 | 145.455 | 1.102 | 1.222 | 1.374 | 1.164 | 1.142 | 181.516 |
| 6 | 141.479 | 1.136 | 1.214 | 1.388 | 1.218 | 1.133 | 141.103 |
| 7 | 161.559 | 1.189 | 1.239 | 1.491 | 1.410 | 1.223 | 173.750 |
| 8 | 146.068 | 0.966 | 1.799 | 1.403 | 1.173 | 1.316 | 173.758 |
| 9 | 142.547 | 0.995 | 1.422 | 1.533 | 1.516 | 1.405 | 272.208 |
| 10 | 237.617 | 1.368 | 1.716 | 1.821 | 1.513 | 1.065 | 169.944 |
| 11 | 222.319 | 0.999 | 1.371 | 1.557 | 1.119 | 1.090 | 277.353 |
| 12 | 202.847 | 1.193 | 1.349 | 1.427 | 1.096 | 1.599 | 181.426 |
| 13 | 206.047 | 1.296 | 1.526 | 1.344 | 1.405 | 1.468 | 152.230 |
| 14 | 171.365 | 1.079 | 1.404 | 1.522 | 1.577 | 1.104 | 172.429 |
| 15 | 180.604 | 1.349 | 1.665 | 1.385 | 1.561 | 1.544 | 179.639 |
| 16 | 188.487 | 1.097 | 1.410 | 1.950 | 1.112 | 1.169 | 250.849 |
| 17 | 129.209 | 1.026 | 1.702 | 1.27 | 1.131 | 1.231 | 156.131 |
| 18 | 142.897 | 1.152 | 1.152 | 1.417 | 1.184 | 1.117 | 164.660 |
| 19 | 144.326 | 0.963 | 1.292 | 1.545 | 1.343 | 1.248 | 159.000 |
| 20 | 156.613 | 0.968 | 1.296 | 1.249 | 1.523 | 1.081 | 148.673 |
| Среднее Значение | 168.219 | 1.282 | 1.494 | 1.687 | 1.422 | 1.341 | 182.610 |
| Среднеквадратичное | 29.131 | 0.773 | 0.450 | 0.946 | 0.571 | 0.539 | 37.920 |



Рис. 1. Среднее время выполнения первого запроса

Таблица 2. Время выполнения второго запроса, ms

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Без индекса | B-дерево | Хеш | GiST | SP-GiST | GIN | BRIN |
| 1 | 5.622 | 1.155 | 1.900 | 4.365 | 1.79 | 1.999 | 5.578 |
| 2 | 4.141 | 0.531 | 0.647 | 0.72 | 0.699 | 0.451 | 7.536 |
| 3 | 4.602 | 0.436 | 0.46 | 0.557 | 0.398 | 0.402 | 4.572 |
| 4 | 4.364 | 0.422 | 0.533 | 0.502 | 0.403 | 0.633 | 6.114 |
| 5 | 4.003 | 0.561 | 0.449 | 0.481 | 0.399 | 0.395 | 5.229 |
| 6 | 5.197 | 0.420 | 0.407 | 0.803 | 0.653 | 0.457 | 4.021 |
| 7 | 6.908 | 0.530 | 0.439 | 0.539 | 0.419 | 0.392 | 4.865 |
| 8 | 4.151 | 1.086 | 0.533 | 0.542 | 0.396 | 0.442 | 4.889 |
| 9 | 3.958 | 0.420 | 0.460 | 0.499 | 0.425 | 0.409 | 7.129 |
| 10 | 4.120 | 0.464 | 0.451 | 0.495 | 0.444 | 0.396 | 4.181 |
| 11 | 4.463 | 0.373 | 0.457 | 1.055 | 0.432 | 0.408 | 4.515 |
| 12 | 6.989 | 0.444 | 0.416 | 0.515 | 0.548 | 0.442 | 4.166 |
| 13 | 4.565 | 0.427 | 0.546 | 0.563 | 0.421 | 0.403 | 4.873 |
| 14 | 4.620 | 0.374 | 0.609 | 0.559 | 0.412 | 0.462 | 4.430 |
| 15 | 3.964 | 0.573 | 0.461 | 0.529 | 0.410 | 0.511 | 4.894 |
| 16 | 4.303 | 0.383 | 0.412 | 0.846 | 0.411 | 0.394 | 4.124 |
| 17 | 4.323 | 0.408 | 0.420 | 0.495 | 0.419 | 0.414 | 4.764 |
| 18 | 5.279 | 0.387 | 0.424 | 0.593 | 0.619 | 0.396 | 4.268 |
| 19 | 3.961 | 0.555 | 0.798 | 0.553 | 0.434 | 0.412 | 4.184 |
| 20 | 4.449 | 0.541 | 0.407 | 0.564 | 0.440 | 0.404 | 4.234 |
| Среднее Значение | 4.699 | 0.525 | 0.561 | 0.789 | 0.462 | 0.511 | 4.928 |
| Среднеквадратичное | 0.871 | 0.209 | 0.322 | 0.833 | 0.091 | 0.346 | 0.955 |



Рис. 2. Среднее время выполнения второго запроса

5. Анализ эффективности

Согласно QUERY PLAN, при выполнении первого запроса без использования индекса, сначала происходит Seq Scan или Table Scan, оператор получает строки из таблицы последовательно. Затем фильтр принимает входные данные и возвращает только те строки, которые удовлетворяют критерию фильтрации (player1\_name = 'Robert Horry'). В результате выполнение запроса занимает 168,219 ms.

При использовании индекса B-дерево в запросе сначала происходит Heap Scan, и по условию (player1\_name = 'Robert Horry') находится 690 heap blocks. Затем выполняется Index Scan и возвращаются только те строки, которые удовлетворяют условию. В результате выполнение запроса занимает 1,282 ms.

При использовании индекса "BRIN" в запросе сначала происходит Gather, как и в случае, когда индекса нет. Затем выполняется Heap Scan, и по условию (player1\_name = 'Robert Horry') находится 12529 heap blocks, что значительно больше, чем при использовании других индексов. Затем выполняется Index Scan и возвращаются только те строки, которые удовлетворяют условию [8]. В результате выполнение запроса занимает 182,610 ms.

Первый запрос имеет одинаковый QUERY PLAN при использовании индексов B-дерево, Хеш, GiST, P-GIST и GIN, но немного различается время выполнения из-за небольших отличий в actual time of Index Scan [4-7]. GiST - это сокращение от "Generalized Search Tree" и представляет собой сбалансированное дерево поиска. В случае, если поддерживаются операторы "больше", "меньше" и "равно", GiST не так эффективен, как индекс B-дерево, поэтому результат индексации GiST в данном эксперименте хуже, чем у индекса B-дерево. Однако метод индексации GiST полезен для типов данных, которые не имеют смысла для этих операторов, например, географических данных, текстовых документов, изображений и т.д.

Заключение

В данной работе исследовались различные индексы для базы данных баскетболистов Национальной баскетбольной ассоциации (НБА), включая B-дерево, Хеш, GiST, SP-GiST, GIN и BRIN. Результаты исследования показали, что после создания индексов B-дерево, Хеш, GiST, SP-GiST и GIN эффективность поиска данных значительно увеличилась, а BRIN-индекс не подходит для данной базы данных. В работе была произведена анализ эффективности обработки запросов для разных типов индексов, основанный на QUERY PLAN [9]. Эксперименты показали, что B-дерево может работать с условиями на равенство и проверками диапазонов для данных, которые можно отсортировать в определенном порядке, а хеш-индексы работают только с простыми условиями равенства.

В будущем мы можем создавать более современные базы данных, которые будут хранить такие типы данных, как географические данные, текстовые документы, изображения и другие. Основываясь на полученных результатах, мы можем разработать новую систему индексации, которая будет более подходящей для нашей базы данных.

Список литературы

1. Фролов, К. М., & Казакова, Р. И. (2014). B-дерево как индексная структура данных. In *Информационные технологии в науке и образовании. Проблемы и перспективы* (p. 159).
2. Веретенников, А. Б. Эффективное создание текстовых индексов. *Проблемы теоретической и прикладной математики: Труды*, *39*, 348-350.
3. Хайков, Д. В., & Пипенко, Р. С. (2022). СРАВНЕНИЕ СТРУКТУР ХРАНЕНИЯ ДАННЫХ «ДВОИЧНОЕ ДЕРЕВО»,«КРАСНО-ЧЕРНОЕ ДЕРЕВО» И «B-ДЕРЕВО». In *НОВОЕ СЛОВО В НАУКЕ И ОБРАЗОВАНИИ* (pp. 7-10).
4. Иванова, Е. В. (2014). Использование распределенных колоночных хеш-индексов для обработки запросов к сверхбольшим базам данных. In *Научный сервис в сети Интернет: многообразие суперкомпьютерных миров. Труды Международной суперкомпьютерной конференции. М.: Изд-во МГУ* (pp. 102-104).
5. РЕДЮК, С., & САТТАРОВ, А. (2015). Реализация метода последовательного обхода GiST индекса в СУБД PostgreSQL. In *XIX Международная телекоммуникационная конференция молодых ученых и студентов" МОЛОДЕЖЬ И НАУКА"* (pp. 34-34).
6. Aref, W. G., & Ilyas, I. F. (2001). Sp-gist: An extensible database index for supporting space partitioning trees. *Journal of Intelligent Information Systems*, *17*, 215-240.
7. Барахнин, В. Б., Кожемякина, О. Ю., & Борзилова, Ю. С. (2020). Оптимизация SQL-запросов на примере работы поискового модуля системы комплексного анализа художественных текстов. *Cloud of Science*, *7*(4), 749-763.
8. Меджидов, Р. Г. (2019). Анализ многоколоночных индексов баз данных. In *Актуальные проблемы прикладной математики, информатики и механики* (pp. 420-422).
9. Rizzolo, F., & Vaisman, A. A. (2008). Temporal XML: modeling, indexing, and query processing. *The VLDB Journal*, *17*, 1179-1212.
10. Li, K., & Li, G. (2018). Approximate query processing: What is new and where to go? A survey on approximate query processing. *Data Science and Engineering*, *3*, 379-397.