Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Программная инженерия
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ

Индексы
тема

Преподаватель _____ А. Д. Вожжов
подпись, дата — инициалы, фамилия

Студент КИ23-17/1б, 032320521 _____ А. С. Лысаковский
номер группы, зачётной книжки — подпись, дата — инициалы, фамилия

Красноярск 2025

1 ВВЕДЕНИЕ

1.1 Цель работы

Изучить теоретический материал по теме «Индексы». Выполнить задания.

1.2 Задачи

В рамках данной практической работы необходимо выполнить следующие задачи:

1 изучить теоретический материал по предложенной теме;
2 выполнить задание;
3 предоставить отчёт преподавателю.

1.3 Задание

Задание данной практической работы состоит из следующих частей:

4 Выполнить задания из главы 8 из книги на е-курсах.

2 ХОД РАБОТЫ

2.1 Задание 1

На рисунках 1, 2 показан результат выполнения задания.

1. Предположим, что для какой-то таблицы создан уникальный индекс по двум столбцам: `column1` и `column2`. В таблице есть строка, у которой значение атрибута `column1` равно ABC, а значение атрибута `column2` — NULL. Мы решили добавить в таблицу еще одну строку с такими же значениями ключевых атрибутов, т. е. `column1` — ABC, а `column2` — NULL.

   Как вы думаете, будет ли операция вставки новой строки успешной или завершится с ошибкой? Объясните ваше решение.

Рисунок 1 – Задание

```
demo=# CREATE TEMP TABLE test_table (
demo(# column1 TEXT,    column2 TEXT,    UNIQUE (column1, column2));
CREATE TABLE
Время: 14,844 мс
demo=#
demo=# INSERT INTO test_table (column1, column2) VALUES ('ABC', NULL);
INSERT 0 1
Время: 1,102 мс
demo=# INSERT INTO test_table (column1, column2) VALUES ('ABC', NULL);
INSERT 0 1
Время: 0,226 мс
demo=# SELECT * FROM test_table;
 column1 | column2
---------+---------
 ABC     |
 ABC     |
(2 строки)
```

Рисунок 2 – Проверка гипотезы

Проблем не возникает, так как NULL != NULL в postgresql.

2.2 Задание 2

На рисунках 3-5 показан результат выполнения задания.

Проведите следующий эксперимент: выполните этот запрос несколько раз подряд при отсутствии индекса, а затем создайте индекс и опять выполните этот запрос несколько раз подряд.

```
SELECT count( * )
  FROM tickets
  WHERE passenger_name = 'IVAN IVANOV';
```

Вы увидите, что время выполнения *повторных* запросов к таблице сокращается, причем, когда создан индекс, оно сокращается на порядок. Как вы думаете, почему?

Рисунок 3 – Задание

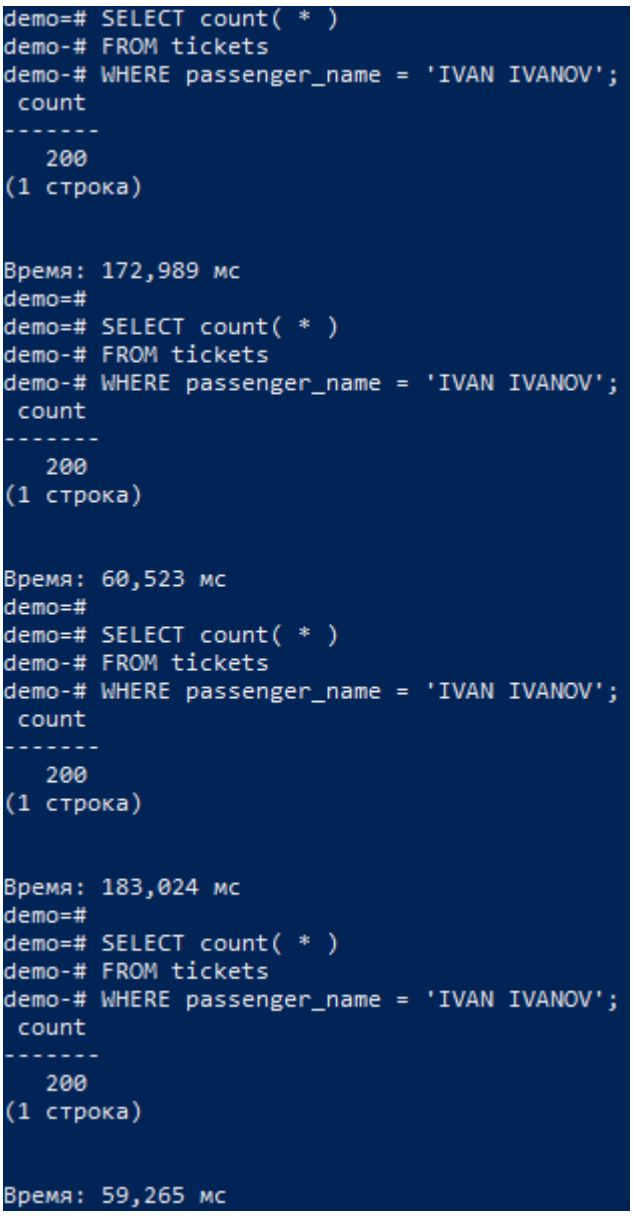

Рисунок 4 — Запросы с индексированием

```
demo=# CREATE INDEX ON tickets ( passenger_name );
CREATE INDEX
Время: 1451,020 мс (00:01,451)
demo=#
demo=# SELECT count( * )
demo-# FROM tickets
demo-# WHERE passenger_name = 'IVAN IVANOV';
 count
-------
   200
(1 строка)


Время: 0,905 мс
demo=# SELECT count( * )
demo-# FROM tickets
demo-# WHERE passenger_name = 'IVAN IVANOV';
 count
-------
   200
(1 строка)


Время: 0,364 мс
demo=# SELECT count( * )
demo-# FROM tickets
demo-# WHERE passenger_name = 'IVAN IVANOV';
 count
-------
   200
(1 строка)


Время: 0,434 мс
demo=# SELECT count( * )
demo-# FROM tickets
demo-# WHERE passenger_name = 'IVAN IVANOV';
 count
-------
   200
(1 строка)


Время: 0,445 мс
```

Рисунок 5 — Запросы без индексирования

Прирост скорости выполнения запроса в обоих случаях обусловлен сохранением части или всего количества в памяти результата из-за выполнения повторных запросов.

2.3 Задание 3

На рисунках 6-8 показан результат выполнения задания. Возвращаемый результат будет представлять таблицу вставленных данных.

Известно, что индекс значительно ускоряет работу, если при выполнении запроса из таблицы отбирается лишь небольшая часть строк. Если же эта доля велика, скажем, половина строк или более, то большого положительного эффекта от наличия индекса уже не будет, а возможно даже, что не будет практически никакого эффекта. Наша задача — проверить это утверждение на практике.

Обратимся к таблице «Перелеты» (`ticket_flights`). В ней имеется столбец «Класс обслуживания» (`fare_conditions`), который отличается от остальных тем, что в нем могут присутствовать лишь три различных значения: `Comfort`, `Business` и `Economy`.

Если секундомер в утилите psql выключен, то включите его.

Выполните запросы, подсчитывающие количество строк, в которых атрибут `fare_conditions` принимает одно из трех возможных значений. Каждый из запросов выполните три-четыре раза, поскольку время может немного изменяться, и подсчитайте среднее время. Обратите внимание на число строк, которые возвращает функция `count` для каждого значения атрибута. При этом среднее время выполнения запросов для трех различных значений атрибута `fare_conditions` будет различаться незначительно, поскольку в каждом случае СУБД просматривает все строки таблицы.

Рисунок 6 – Задание

```
demo=# SELECT count( * )
demo-# FROM ticket_flights
demo-# WHERE fare_conditions = 'Comfort';
 count
-------
 17291
(1 строка)


Время: 98,814 мс
demo=#
demo=# SELECT count( * )
demo-# FROM ticket_flights
demo-# WHERE fare_conditions = 'Business';
 count
--------
 107642
(1 строка)


Время: 72,298 мс
demo=#
demo=# SELECT count( * )
demo-# FROM ticket_flights
demo-# WHERE fare_conditions = 'Economy';
 count
--------
 920793
(1 строка)


Время: 79,468 мс
```

Рисунок 7 – Запросы без индексирования

```
demo=# CREATE INDEX ON ticket_flights (fare_conditions);
CREATE INDEX
Время: 634,005 мс
demo=# SELECT count( * )
demo-# FROM ticket_flights
demo-# WHERE fare_conditions = 'Comfort';
 count
-------
 17291
(1 строка)


Время: 1,920 мс
demo=# SELECT count( * )
demo-# FROM ticket_flights
demo-# WHERE fare_conditions = 'Business';
 count
--------
 107642
(1 строка)


Время: 8,443 мс
demo=# SELECT count( * )
demo-# FROM ticket_flights
demo-# WHERE fare_conditions = 'Economy';
 count
--------
 920793
(1 строка)


Время: 160,728 мс
```

Рисунок 8 – Запросы с индексированием

Наблюдается положительный прирост времени в 2-х из 3-х случаев.

2.4 Задание 4

На рисунках с 9-12 показан результат выполнения задания.

Для одной из таблиц создайте индекс по двум столбцам, причем по одному из них укажите убывающий порядок значений столбца, а по другому — возрастающий. Значения NULL у первого столбца должны располагаться в начале, а у второго — в конце. Посмотрите полученный индекс с помощью команд psql

```
\d имя_таблицы
\di+ имя_индекса
```

Обратите внимание, что первая команда выведет не только имя индекса, но также и имена столбцов, по которым он создан, а вторая команда выведет размер индекса.

Подберите запросы, в которых созданный индекс предположительно должен использоваться, а также запросы, в которых он использоваться, по вашему мнению, не будет. Проверьте ваши гипотезы, выполнив запросы. Объясните полученные результаты.

Рисунок 9 – Задание

```
demo=# CREATE INDEX bookings_date_amount_idxON bookings (book_date DESC NULLS FIRST, total_amount ASC NULLS LAST);
```

Рисунок 10 – Создание индекса

```
demo=# SELECT * FROM bookings ORDER BY book_date DESC NULLS FIRST, total_amount ASC NULLS LAST LIMIT 5;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 A093BB   | 2016-10-13 21:00:00+07 |     13000.00
 21E611   | 2016-10-13 21:00:00+07 |     16400.00
 2AFB36   | 2016-10-13 21:00:00+07 |     21400.00
 53C497   | 2016-10-13 21:00:00+07 |     67600.00
 F8601F   | 2016-10-13 21:00:00+07 |     93400.00
(5 строк)


Время: 50,922 мс
```

Рисунок 11 – Запрос 1

```
demo=# SELECT * FROM bookings WHERE book_date > '2023-02-01'ORDER BY book_date DESC NULLS FIRST, total_amount ASC NULLS
LAST LIMIT 5;
 book_ref | book_date | total_amount
----------+-----------+--------------
(0 строк)
```

Рисунок 12 – Запрос 2

2.5 Задание 5

На рисунках 13-17 показан результат выполнения задания.

В сложных базах данных целесообразно использование комбинаций индексов. Иногда бывают более полезны комбинированные индексы по нескольким столбцам, чем отдельные индексы по единичным столбцам. В реальных ситуациях часто приходится делать выбор, т. е. находить компромисс, между, например, созданием двух индексов по каждому из двух столбцов таблицы либо созданием одного индекса по двум столбцам этой таблицы, либо созданием всех трех индексов. Выбор зависит от того, запросы какого вида будут выполняться чаще всего. Предложите какую-нибудь таблицу в базе данных «Авиаперевозки» и смоделируйте ситуации, в которых вы приняли бы одно из этих трех возможных решений. Воспользуйтесь документацией на PostgreSQL.

Рисунок 13 – Задание

```
Время: 1,909 мс
demo=# SELECT * FROM ticket_flights
demo-# WHERE flight_id = 12345;
 ticket_no | flight_id | fare_conditions | amount
-----------+-----------+-----------------+--------
(0 строк)


Время: 66,892 мс
demo=# SELECT * FROM ticket_flightsWHERE amount BETWEEN 500.00 AND 1000.00;
ОШИБКА:  ошибка синтаксиса (примерное положение: "BETWEEN")
СТРОКА 1: SELECT * FROM ticket_flightsWHERE amount BETWEEN 500.00 AND ...
                                                 ^
Время: 15,627 мс
```

Рисунок 14 – Не индексированные запросы

```
demo=# CREATE INDEX ticket_flights_flight_id_idx ON ticket_flights (flight_id);
CREATE INDEX
Время: 374,944 мс
demo=# CREATE INDEX ticket_flights_amount_idx ON ticket_flights (amount);
CREATE INDEX
Время: 1004,813 мс (00:01,005)
```

Рисунок 15 – Индексация

```
demo=# SELECT * FROM ticket_flights WHERE flight_id = 12345;
 ticket_no | flight_id | fare_conditions | amount
-----------+-----------+-----------------+--------
(0 строк)


Время: 0,904 мс
demo=# SELECT * FROM ticket_flights WHERE amount BETWEEN 500.00 AND 1000.00;
 ticket_no | flight_id | fare_conditions | amount
-----------+-----------+-----------------+--------
(0 строк)


Время: 1,941 мс
```

Рисунок 16 – Индексированные запросы

```
demo=# CREATE INDEX ticket_flights_flight_amount_idx ON ticket_flights (flight_id, amount);
CREATE INDEX
Время: 844,405 мс
demo=# SELECT * FROM ticket_flights WHERE flight_id = 12345;
 ticket_no | flight_id | fare_conditions | amount
-----------+-----------+-----------------+--------
(0 строк)


Время: 0,688 мс
demo=# SELECT * FROM ticket_flights WHERE amount BETWEEN 500.00 AND 1000.00;
 ticket_no | flight_id | fare_conditions | amount
-----------+-----------+-----------------+--------
(0 строк)


Время: 0,628 мс
```

Рисунок 17 – Комбинированный индекс и запросы

2.6 Задание 6

На рисунке 18 показано задание.

Предложите какую-нибудь таблицу в базе данных «Авиаперевозки» и смоделируйте ситуацию, в которой было бы целесообразно использование индекса на основе функции или скалярного выражения от двух или более столбцов.

Рисунок 18 – Задание

Предположим, авиакомпания хочет анализировать продолжительность рейсов (разницу между «scheduled_arrival» и «scheduled_departure») для оптимизации расписания и выявления рейсов с определённой длительностью.

Кроме того, ей важно учитывать статус рейса («status»), чтобы, например, исключить отменённые рейсы («status = 'Cancelled'») из анализа.

На рисунках 19-20 показан результат выполнения задания.

```
demo=# CREATE INDEX flights_duration_status_idx
demo-# ON flights (
demo(#     (scheduled_arrival - scheduled_departure) DESC,
demo(#     status
demo(# );
CREATE INDEX
Время: 42,867 мс
```

Рисунок 19 – Создание индекса

```
demo=# SELECT flight_no, scheduled_departure, scheduled_arrival FROM flights WHERE (scheduled_arrival - scheduled_depart
ure) > INTERVAL '3 hours' AND status != 'Cancelled' ORDER BY (scheduled_arrival - scheduled_departure) DESC LIMIT 10;
 flight_no |  scheduled_departure   |   scheduled_arrival
-----------+------------------------+------------------------
 PG0168    | 2016-09-20 23:05:00+07 | 2016-09-21 07:55:00+07
 PG0168    | 2016-10-03 23:05:00+07 | 2016-10-04 07:55:00+07
 PG0168    | 2016-09-14 23:05:00+07 | 2016-09-15 07:55:00+07
 PG0168    | 2016-09-15 23:05:00+07 | 2016-09-16 07:55:00+07
 PG0168    | 2016-09-24 23:05:00+07 | 2016-09-25 07:55:00+07
 PG0168    | 2016-09-25 23:05:00+07 | 2016-09-26 07:55:00+07
 PG0168    | 2016-10-09 23:05:00+07 | 2016-10-10 07:55:00+07
 PG0168    | 2016-10-02 23:05:00+07 | 2016-10-03 07:55:00+07
 PG0168    | 2016-10-01 23:05:00+07 | 2016-10-02 07:55:00+07
 PG0168    | 2016-09-28 23:05:00+07 | 2016-09-29 07:55:00+07
(10 строк)


Время: 0,514 мс
demo=# DROP INDEX flights_duration_status_idx;
DROP INDEX
Время: 2,680 мс
demo=# SELECT flight_no, scheduled_departure, scheduled_arrival FROM flights WHERE (scheduled_arrival - scheduled_depart
ure) > INTERVAL '3 hours' AND status != 'Cancelled' ORDER BY (scheduled_arrival - scheduled_departure) DESC LIMIT 10;
 flight_no |  scheduled_departure   |   scheduled_arrival
-----------+------------------------+------------------------
 PG0168    | 2016-09-14 23:05:00+07 | 2016-09-15 07:55:00+07
 PG0168    | 2016-09-25 23:05:00+07 | 2016-09-26 07:55:00+07
 PG0168    | 2016-09-20 23:05:00+07 | 2016-09-21 07:55:00+07
 PG0168    | 2016-11-10 23:05:00+07 | 2016-11-11 07:55:00+07
 PG0168    | 2016-09-15 23:05:00+07 | 2016-09-16 07:55:00+07
 PG0168    | 2016-09-24 23:05:00+07 | 2016-09-25 07:55:00+07
 PG0168    | 2016-11-11 23:05:00+07 | 2016-11-12 07:55:00+07
 PG0168    | 2016-11-09 23:05:00+07 | 2016-11-10 07:55:00+07
 PG0168    | 2016-10-03 23:05:00+07 | 2016-10-04 07:55:00+07
 PG0168    | 2016-10-26 23:05:00+07 | 2016-10-27 07:55:00+07
(10 строк)
```

Рисунок 20 – Выполнение запроса с индексом и без

2.7 Задание 7

На рисунках 21-22 показан результат выполнения задания.

В разделе документации 5.3.5 «Внешние ключи» говорится о том, что в некоторых ситуациях бывает целесообразно создавать индекс по столбцам внешнего ключа ссылающейся таблицы. Это позволит ускорить выполнение операций DELETE и UPDATE над главной (ссылочной) таблицей.

Подумайте, есть ли такие таблицы в базе данных «Авиаперевозки», в отношении которых было бы целесообразно поступить так, как говорится в документации.

Рисунок 21 – Задание

```
demo=# CREATE INDEX flights_aircraft_code_idx ON flights (aircraft_code);
CREATE INDEX
Время: 28,925 мс
```

Рисунок 22 – Индекс для внешнего ключа

2.8 Задание 8

На рисунках 23-25 показан результат выполнения задания.

В тексте главы был показан пример использования частичного индекса для таблицы «Бронирования». Для его создания мы выполняли команду

```
CREATE INDEX bookings_book_date_part_key
  ON bookings ( book_date )
  WHERE total_amount > 1000000;
```

Проведите эксперимент с целью сравнения эффекта от создания частичного индекса с эффектом от создания обычного индекса по столбцу `total_amount`. Для этого удалите частичный индекс, а затем создайте обычный индекс.

```
DROP INDEX bookings_book_date_part_key;

CREATE INDEX bookings_total_amount_key
  ON bookings ( total_amount );
```

Теперь выполните тот же запрос к таблице `bookings`, который был приведен в тексте главы:

```
SELECT *
  FROM bookings
  WHERE total_amount > 1000000
  ORDER BY book_date DESC;
```

Сравните время выполнения с тем временем, которое было получено при использовании частичного индекса. Очень вероятно, что различия времени выполнения запроса будут незначительными.

Самостоятельно ознакомьтесь с разделом документации 11.8 «Частичные индексы» и попробуйте смоделировать ситуацию в предметной области «Авиаперевозки», когда частичный индекс дал бы больший эффект, чем обычный индекс.

Рисунок 23 – Задание

```
demo=# CREATE INDEX bookings_total_amount_key
demo-# ON bookings ( total_amount );
CREATE INDEX
Время: 279,207 мс
demo=#
demo=# SELECT *
demo-# FROM bookings
demo-# WHERE total_amount > 1000000
demo-# ORDER BY book_date DESC;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 D7E9AA   | 2016-10-06 08:29:00+07 |   1062800.00
 EF479E   | 2016-09-30 18:58:00+07 |   1035100.00
 3AC131   | 2016-09-28 04:06:00+07 |   1087100.00
 3B54BB   | 2016-09-02 20:08:00+07 |   1204500.00
 65A6EA   | 2016-08-31 09:28:00+07 |   1065600.00
(5 строк)


Время: 1,064 мс
```

Рисунок 24 – Обычный индекс и запрос

```
demo=# DROP INDEX bookings_total_amount_key;
DROP INDEX
Время: 3,022 мс
demo=# CREATE INDEX bookings_total_amount_part_key ON bookings (total_amount) WHERE total_amount > 1000000;
CREATE INDEX
Время: 52,040 мс
demo=# SELECT *
demo-# FROM bookings
demo-# WHERE total_amount > 1000000
demo-# ORDER BY book_date DESC;
 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 D7E9AA   | 2016-10-06 08:29:00+07 |   1062800.00
 EF479E   | 2016-09-30 18:58:00+07 |   1035100.00
 3AC131   | 2016-09-28 04:06:00+07 |   1087100.00
 3B54BB   | 2016-09-02 20:08:00+07 |   1204500.00
 65A6EA   | 2016-08-31 09:28:00+07 |   1065600.00
(5 строк)


Время: 0,865 мс
```

Рисунок 25 – Частичный индекс и запрос

Время не сильно отличается.

2.9 Задание 9

На рисунках 26-29 показан результат выполнения задания.

Когда выполняются запросы с поиском по шаблону LIKE или регулярными выражениями POSIX, тогда для того, чтобы использовался индекс, нужно предусмотреть следующее. Если параметры локализации системы отличаются от стандартной настройки «C» (например, «ru_RU.UTF-8»), тогда при создании индекса необходимо указать так называемый класс операторов. Существуют различные классы операторов, например, для столбца типа text это будет text_pattern_ops.

```
CREATE INDEX tickets_pass_name
  ON tickets ( passenger_name text_pattern_ops );
```

Индексы со специальными классами операторов пригодны не для всех типов запросов. Поэтому, возможно, потребуется создать еще и индекс с классом операторов по умолчанию. Самостоятельно изучите этот вопрос с помощью раздела документации 11.9 «Семейства и классы операторов».

Рисунок 26 – Задание

```
demo=# CREATE INDEX tickets_passenger_name_pattern_idx
demo-# ON tickets (passenger_name text_pattern_ops);
CREATE INDEX
Время: 282,359 мс
demo=#
demo=# SELECT * FROM tickets WHERE passenger_name LIKE 'IVAN%' LIMIT 10;
   ticket_no   | book_ref | passenger_id | passenger_name  |                                 contact_data
---------------+----------+--------------+-----------------+-----------------------------------------------------------
--------------
 0005432001031 | EA4481   | 6365 326222  | IVAN MEDVEDEV   | {"phone": "+70386792287"}
 0005432002050 | 997E15   | 1779 518628  | IVAN KUDRYASHOV | {"phone": "+70268002514"}
 0005432002074 | 66583B   | 6734 551438  | IVAN STEPANOV   | {"phone": "+70392144991"}
 0005432002081 | 305FED   | 6745 357385  | IVAN FROLOV     | {"email": "frolov_i_101966@postgrespro.ru", "phone": "+7088
6971455"}
 0005432003670 | E33E2B   | 7893 562577  | IVAN POPOV      | {"email": "ivanpopov.1985@postgrespro.ru", "phone": "+70798
991248"}
 0005432003783 | 95CF7C   | 3430 815878  | IVAN SERGEEV    | {"email": "sergeevi.08051969@postgrespro.ru", "phone": "+70
316279857"}
 0005432005201 | E2A3CC   | 5163 816189  | IVAN YAKOVLEV   | {"email": "ivan-yakovlev071973@postgrespro.ru", "phone": "+
70788739132"}
 0005432019836 | 76184F   | 6344 157785  | IVAN DMITRIEV   | {"phone": "+70142946091"}
 0005432019857 | D335CA   | 9284 067873  | IVAN VOLKOV     | {"email": "ivolkov-1976@postgrespro.ru", "phone": "+7091240
0382"}
 0005432019874 | 1EA898   | 2045 059431  | IVAN PAVLOV     | {"phone": "+70051298495"}
(10 строк)


Время: 1,121 мс
```

Рисунок 27 — Создание индекса и выполнение запроса

```
demo=# SELECT * FROM tickets WHERE passenger_name LIKE '%OV%' LIMIT 10;
   ticket_no   | book_ref | passenger_id |   passenger_name   |                                 contact_data
---------------+----------+--------------+--------------------+-----------------------------------------------------------
------------------------
 0005432020115 | 6668DA   | 5660 190512  | TATYANA YAKOVLEVA  | {"email": "yakovleva_tatyana_26091973@postgrespro.ru", "
phone": "+70111898460"}
 0005432020116 | 6668DA   | 8566 477919  | NATALYA KONOVALOVA | {"email": "konovalova.natalya101976@postgrespro.ru", "ph
one": "+70995865185"}
 0005432020117 | 48699C   | 0804 755038  | SVETLANA BELOVA    | {"phone": "+70976908603"}
 0005432020119 | 005696   | 9790 140874  | OLEG SCHERBAKOV    | {"phone": "+70766374027"}
 0005432020121 | 92DC06   | 7756 431675  | ELENA KRASNOVA     | {"email": "elena_krasnova09111982@postgrespro.ru", "phon
e": "+70378220929"}
 0005432020123 | 92DC06   | 9417 323021  | IRINA NAZAROVA     | {"email": "nazarova-i-23051972@postgrespro.ru", "phone"
 "+70912630972"}
 0005432020125 | 83A07B   | 4249 898314  | SOFIYA STEPANOVA   | {"email": "s.stepanova17071972@postgrespro.ru", "phone"
 "+70419184936"}
 0005432020126 | 83A07B   | 5870 778002  | NIKOLAY EGOROV     | {"email": "nikolayegorov-1972@postgrespro.ru", "phone":
"+70450475476"}
 0005432020127 | 67C35F   | 2376 383107  | NATALYA GAVRILOVA  | {"email": "natalya_gavrilova_1970@postgrespro.ru", "phon
e": "+70952097340"}
 0005432020129 | 4C714C   | 3613 609181  | DENIS GAVRILOV     | {"phone": "+70776381590"}
(10 строк)


Время: 0,544 мс
```

Рисунок 28 — Запрос 1

```
demo=# SELECT * FROM tickets WHERE passenger_name ~ '^IVAN [A-Z]+' LIMIT 10;
   ticket_no   | book_ref | passenger_id | passenger_name |                                 contact_data
---------------+----------+--------------+----------------+-----------------------------------------------------------
------------
 0005432020212 | DDD384   | 0236 149814  | IVAN SIDOROV   | {"phone": "+70388926062"}
 0005432020347 | 862FB5   | 1249 056409  | IVAN STEPANOV  | {"email": "stepanovi_24021969@postgrespro.ru", "phone": "+70
070604540"}
 0005432020348 | A113BD   | 7464 532030  | IVAN STEPANOV  | {"phone": "+70228805653"}
 0005432020386 | 867976   | 2367 383441  | IVAN SAVELEV   | {"email": "savelev.i_1978@postgrespro.ru", "phone": "+708717
78663"}
 0005432020457 | 24EE42   | 6715 865502  | IVAN MAKAROV   | {"phone": "+70782866929"}
 0005432020959 | 829075   | 1828 463905  | IVAN OSIPOV    | {"phone": "+70140186966"}
 0005432020591 | 78501B   | 5491 750097  | IVAN NIKOLAEV  | {"phone": "+70730489352"}
 0005432020610 | FB9240   | 5075 580388  | IVAN BELOV     | {"phone": "+70470493329"}
 0005432020715 | FFF6D4   | 1385 828610  | IVAN DENISOV   | {"email": "denisov_ivan1965@postgrespro.ru", "phone": "+7055
7387573"}
 0005432020950 | A05E72   | 2655 129648  | IVAN KUZNECOV  | {"phone": "+70629705980"}
(10 строк)


Время: 0,857 мс
```

Рисунок 29 — Запрос 2

3 ЗАКЛЮЧЕНИЕ

По результатам работы был изучен теоретический материал по теме «Индексы». Все поставленные цели и задачи были выполнены.