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

ОТЧЁТ О ПРАКТИЧЕСКОЙ РАБОТЕ

Типы данных СУБД PostgreSQL
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

ВВЕДЕНИЕ

1 Цель работы

Изучить теоретический материал по теме «Типы данных СУБД PostgreSQL». Выполнить представленные задания.

2 Задачи

В рамках данной практической работы необходимо выполнить следующие задачи:

- изучить теоретический материал по указанной теме;
- выполнить задание;
- предоставить отчёт преподавателю.

3 Задание

Задание данной практической работы состоит из следующих частей:

- открыть книгу на е-курсах и выполнить задания из главы, указанной на курсе;

ХОД РАБОТЫ

1 Задание 2

На рисунках 1, 2, 3 представлен прогресс работы.

```
dbt_pw2=# create table test_numeric (
dbt_pw2(# measurement numeric,
dbt_pw2(# description text
dbt_pw2(# );
CREATE TABLE
dbt_pw2=#
```

Рисунок 1 – Создание таблицы

```
dbt_pw2=# INSERT INTO test_numeric VALUES ( 1234567890.0987654321, 'Точность 20 знаков, масштаб 10 знаков' );
INSERT 0 1
dbt_pw2=# INSERT INTO test_numeric VALUES ( 1234567890.0987654321, 'Точность 20 знаков, масштаб 10 знаков' );
INSERT 0 1
dbt_pw2=# INSERT INTO test_numeric VALUES ( 0.12345678901234567890, 'Точность 21 знак, масштаб 20 знаков' );
INSERT 0 1
dbt_pw2=# INSERT INTO test_numeric VALUES ( 1234567890, 'Точность 10 знаков, масштаб 0 знаков (целое число)' );
INSERT 0 1
```

Рисунок 2 – Вставка значений различной точности и масштаба

```
dbt_pw2=# SELECT * FROM test_numeric;
      measurement       |                    description
------------------------+-----------------------------------------------------
  1234567890.0987654321 | Точность 20 знаков, масштаб 10 знаков
  1234567890.0987654321 | Точность 20 знаков, масштаб 10 знаков
 0.12345678901234567890 | Точность 21 знак, масштаб 20 знаков
             1234567890 | Точность 10 знаков, масштаб 0 знаков (целое число)
(4 строки)
```

Рисунок 3 – Проверка результата

2 Задание 4

На рисунках 4, 5 представлен прогресс работы.

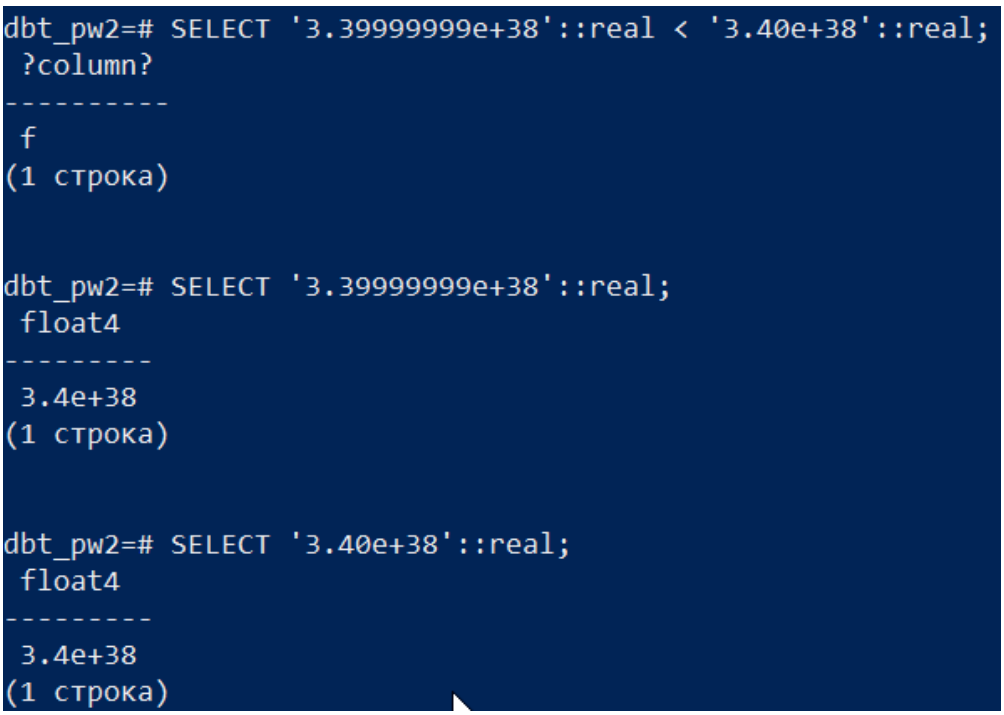

Рисунок 4 – Эксперименты с точностью, часть 1

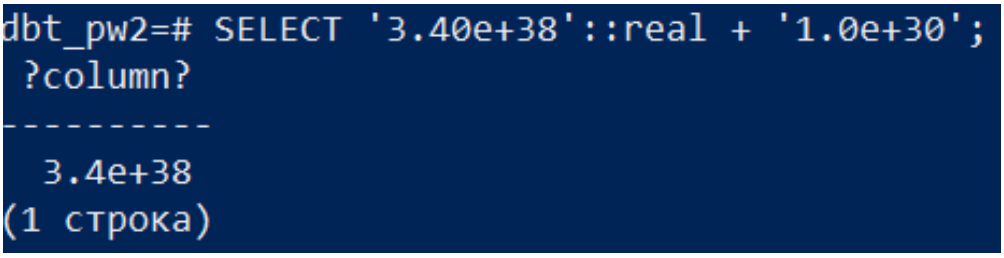

Рисунок 5 – Эксперименты с точностью, часть 2

3 Задание 6

На рисунках 6, 7 представлен прогресс работы. Проверяется, что NaN больше предельных значений того же типа.

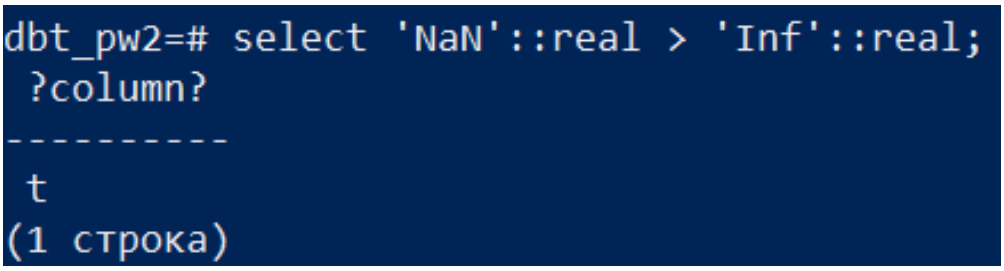

Рисунок 6 – Эксперименты с NaN, часть 1

```
dbt_pw2=# select 'NaN'::double precision > 'Inf'::double precision;
 ?column?
----------
 t
(1 строка)
```

Рисунок 7 – Эксперименты с NaN, часть 2

4 Задание 8

На рисунке 8 представлен прогресс работы. Ошибка возникает, потому-что по умолчанию вставка в таблицу без указания id происходит со значением этого поля, равным 1. После ошибки, postgresql пересчитывает значение последовательности и вставка срабатывает.

```
dbt_pw2=# CREATE TABLE test_serial ( id serial PRIMARY KEY, name text );
CREATE TABLE
dbt_pw2=# INSERT INTO test_serial ( name ) VALUES ( 'Вишневая' );
INSERT 0 1
dbt_pw2=# INSERT INTO test_serial ( id, name ) VALUES ( 2, 'Прохладная' );
INSERT 0 1
dbt_pw2=# INSERT INTO test_serial ( name ) VALUES ( 'Грушевая' );
ОШИБКА:  повторяющееся значение ключа нарушает ограничение уникальности "test_serial_pkey"
ПОДРОБНОСТИ:  Ключ "(id)=(2)" уже существует.
dbt_pw2=# INSERT INTO test_serial ( name ) VALUES ( 'Грушевая' );
INSERT 0 1
dbt_pw2=# INSERT INTO test_serial ( name ) VALUES ( 'Зеленая' );
INSERT 0 1
dbt_pw2=# DELETE FROM test_serial WHERE id = 4;
DELETE 1
dbt_pw2=# INSERT INTO test_serial ( name ) VALUES ( 'Луговая' );
INSERT 0 1
dbt_pw2=# SELECT * FROM test_serial;
 id |    name
----+------------
  1 | Вишневая
  2 | Прохладная
  3 | Грушевая
  5 | Луговая
(4 строки)
```

Рисунок 8 – Манипулирование строками

5 Задание 10

На рисунке 9 представлены ограничения типов. Они связаны с тем, что эти типы данных должны помещаться в 32 или 64 бита. Помимо этого, необходимо обеспечивать работу как с большими, так и маленькими данными.

Name	Storage Size	Description	Low Value	High Value	Resolution
`timestamp [ (p) ] [ without time zone ]`	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond
`timestamp [ (p) ] with time zone`	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond
`date`	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day
`time [ (p) ] [ without time zone ]`	8 bytes	time of day (no date)	00:00:00	24:00:00	1 microsecond
`time [ (p) ] with time zone`	12 bytes	time of day (no date), with time zone	00:00:00+1559	24:00:00-1559	1 microsecond
`interval [ fields ] [ (p) ]`	16 bytes	time interval	-178000000 years	178000000 years	1 microsecond

Рисунок 9 – Ограничения типов данных

6 Задание 12

На рисунках 10, 11, 12, 13, 14 показаны возможности изменения datestyle.

```
dbt_pw2=# SET datestyle TO 'MDY';
SET
dbt_pw2=# SELECT '18-05-2016'::date;
ОШИБКА:  значение поля типа date/time вне диапазона: "18-05-2016"
СТРОКА 1: SELECT '18-05-2016'::date;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
dbt_pw2=# SELECT '05-18-2016'::date;
    date
------------
 2016-05-18
(1 строка)


dbt_pw2=# SET datestyle TO DEFAULT;
SET
```

Рисунок 10 – Экспериментирование с datestyle, часть 1

```
dbt_pw2=# SET datestyle TO 'MDY';
SET
dbt_pw2=# SELECT '18-05-2016'::timestamp;
ОШИБКА:  значение поля типа date/time вне диапазона: "18-05-2016"
СТРОКА 1: SELECT '18-05-2016'::timestamp;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
dbt_pw2=# SELECT '05-18-2016'::timestamp;
      timestamp
---------------------
 2016-05-18 00:00:00
(1 строка)


dbt_pw2=# SET datestyle TO DEFAULT;
SET
```

Рисунок 11 – Экспериментирование с datestyle, часть 2

```
dbt_pw2=# SET datestyle TO 'Postgres, DMY';
SET
dbt_pw2=# SELECT '18-05-2016'::date;
    date
------------
 18-05-2016
(1 строка)


dbt_pw2=# SELECT '05-18-2016'::date;
ОШИБКА:  значение поля типа date/time вне диапазона: "05-18-2016"
СТРОКА 1: SELECT '05-18-2016'::date;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
dbt_pw2=# SELECT '18-05-2016'::timestamp;
        timestamp
--------------------------
 Wed 18 May 00:00:00 2016
(1 строка)


dbt_pw2=# SELECT '05-18-2016'::timestamp;
ОШИБКА:  значение поля типа date/time вне диапазона: "05-18-2016"
СТРОКА 1: SELECT '05-18-2016'::timestamp;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
dbt_pw2=# SET datestyle TO DEFAULT;
SET
```

Рисунок 12 – Экспериментирование с datestyle, часть 3

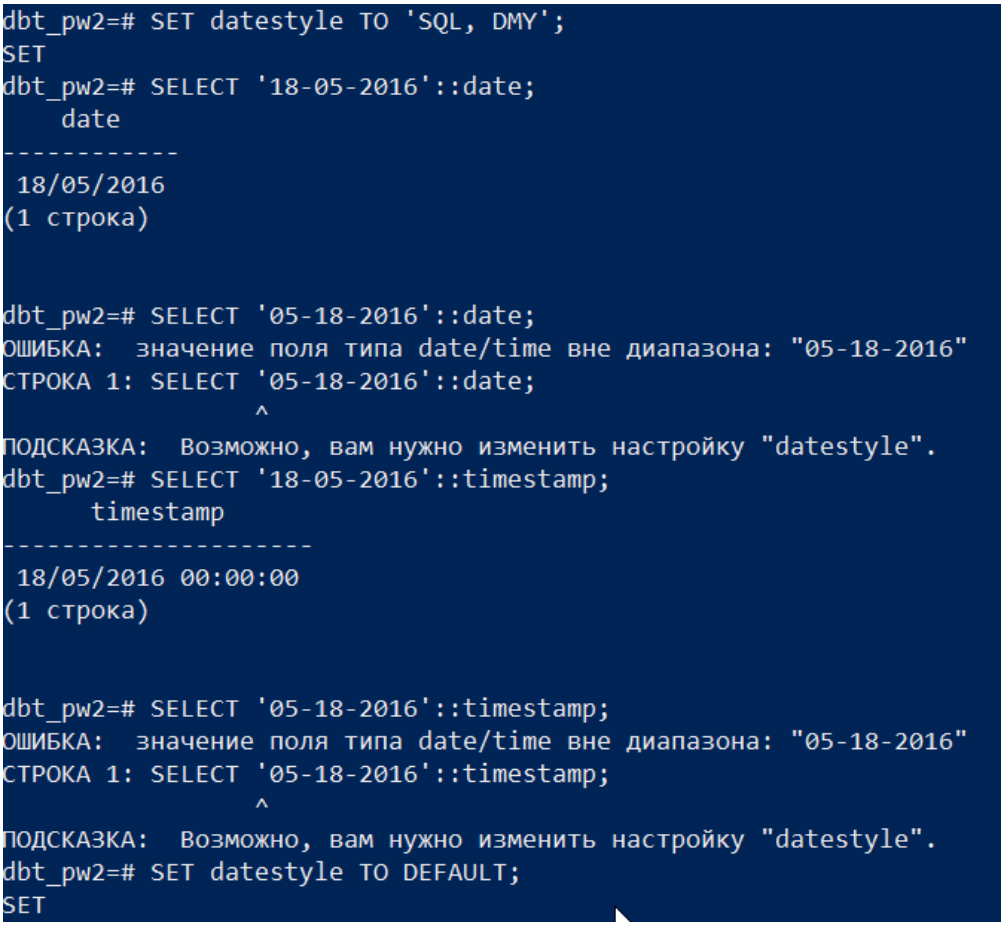

Рисунок 13 – Экспериментирование с datestyle, часть 4

```
dbt_pw2=# SET datestyle TO 'German, DMY';
SET
dbt_pw2=# SELECT '18-05-2016'::date;
    date
------------
 18.05.2016
(1 строка)


dbt_pw2=# SELECT '05-18-2016'::date;
ОШИБКА:  значение поля типа date/time вне диапазона: "05-18-2016"
СТРОКА 1: SELECT '05-18-2016'::date;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
dbt_pw2=# SELECT '18-05-2016'::timestamp;
      timestamp
---------------------
 18.05.2016 00:00:00
(1 строка)


dbt_pw2=# SELECT '05-18-2016'::timestamp;
ОШИБКА:  значение поля типа date/time вне диапазона: "05-18-2016"
СТРОКА 1: SELECT '05-18-2016'::timestamp;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
dbt_pw2=# SET datestyle TO DEFAULT;
SET
```

Рисунок 14 – Экспериментирование с datestyle, часть 5

7 Задание 14

На рисунках 15, 16, 17, 18 показаны возможности изменения datestyle через файл ”postgresql.conf”.

```
datestyle = 'iso, dmy'
```

Рисунок 15 – Экспериментирование с datestyle, часть 1

```
dbt_pw2=# SELECT '05-18-2016'::timestamp; SELECT current_timestamp;
ОШИБКА:  значение поля типа date/time вне диапазона: "05-18-2016"
СТРОКА 1: SELECT '05-18-2016'::timestamp;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
       current_timestamp
-------------------------------
 2025-02-26 00:40:50.539774+07
(1 строка)
```

Рисунок 16 – Экспериментирование с datestyle, часть 2

```
postgres=# SELECT pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 строка)
```

Рисунок 17 – Экспериментирование с datestyle, часть 2

```
dbt_pw2=# SELECT '05-18-2016'::timestamp; SELECT current_timestamp;
ОШИБКА:  значение поля типа date/time вне диапазона: "05-18-2016"
СТРОКА 1: SELECT '05-18-2016'::timestamp;
                 ^
ПОДСКАЗКА:  Возможно, вам нужно изменить настройку "datestyle".
       current_timestamp
-------------------------------
 26/02/2025 00:45:58.807414 +07
(1 строка)
```

Рисунок 18 – Экспериментирование с datestyle, часть 2

8 Задание 16

На рисунке 19 показана особенность postgresql по автоматической проверке даты на корректность.

```
dbt_pw2=# SELECT 'Feb 29, 2015'::date;
ОШИБКА:  значение поля типа date/time вне диапазона: "Feb 29, 2015"
СТРОКА 1: SELECT 'Feb 29, 2015'::date;
```

Рисунок 19 – Ввод недопустимого значения

9 Задание 18

При вычитании одной даты из другой результатом будет являться количество дней - разница между датами. Наиболее удобный формат для таких данных - integer.

На рисунке 20 представлен пример.

```
dbt_pw2=# SELECT pg_typeof( '2016-09-16'::date- '2016-09-01'::date );
 pg_typeof
-----------
 integer
(1 строка)
```

Рисунок 20 – Тип данных для разности двух дат

10 Задание 20

Если прибавить интервал к временной отметке, получится временная отметка. На рисунке 21 показан пример.

```
dbt_pw2=# SELECT ( current_timestamp + '1 mon'::interval );
            ?column?
-------------------------------
 26/03/2025 00:52:00.820481 +07
(1 строка)
```

Рисунок 21 – Сложение временнных отметки и интервала

11 Задание 22

На рисунке 22 показаны возможности стилистического оформления intervalstyle.

```
dbt_pw2=# SET intervalstyle = 'postgres';
SET
dbt_pw2=# SELECT '1 day 2 hours 3 minutes 4 seconds'::interval;
    interval
----------------
 1 day 02:03:04
(1 строка)


dbt_pw2=# SET intervalstyle = 'postgres_verbose';
SET
dbt_pw2=# SELECT '1 day 2 hours 3 minutes 4 seconds'::interval;
           interval
-------------------------------
 @ 1 day 2 hours 3 mins 4 secs
(1 строка)


dbt_pw2=# SET intervalstyle = 'sql_standard';
SET
dbt_pw2=# SELECT '1 day 2 hours 3 minutes 4 seconds'::interval;
 interval
-----------
 1 2:03:04
(1 строка)


dbt_pw2=# SET intervalstyle = 'iso_8601';
SET
dbt_pw2=# SELECT '1 day 2 hours 3 minutes 4 seconds'::interval;
  interval
------------
 P1DT2H3M4S
(1 строка)
```

Рисунок 22 – Эксперименты с intervalstyle

12 Задание 24

На рисунке 23 показан результат выполнения двух команд. Ошибка возникает, потому-что postgresql не понимает, что именно нужно вычесть: час, минуту, секунду. Исправляется приведением к типу INTERVAL. В случае с датой вычитание еденицы воспринимается, как вычитание одного дня, никаких проблем нет.

```
dbt_pw2=# SELECT ( '20:34:35'::time- 1 );
ОШИБКА:  оператор не существует: time without time zone - integer
СТРОКА 1: SELECT ( '20:34:35'::time- 1 );
                                  ^
ПОДСКАЗКА:  Оператор с данными именем и типами аргументов не найден. Возможно, вам следует добавить явные приведения типов.
dbt_pw2=# SELECT ( '2016-09-16'::date- 1 );
  ?column?
------------
 15/09/2016
(1 строка)
```

Рисунок 23 – Результат выполнения двух команд

13 Задание 26

На рисунке 24 показана работа с функцией ”date_trunc”.

```
dbt_pw2=# SELECT date_trunc('day', INTERVAL '5 days 12 hours 34 minutes 56 seconds');
 date_trunc
------------
 5 days
(1 строка)


dbt_pw2=# SELECT date_trunc('hour', INTERVAL '5 days 12 hours 34 minutes 56 seconds');
   date_trunc
-----------------
 5 days 12:00:00
(1 строка)


dbt_pw2=# SELECT date_trunc('day', TIMESTAMP '2023-10-05 14:30:45');
     date_trunc
---------------------
 05/10/2023 00:00:00
(1 строка)


dbt_pw2=# SELECT date_trunc('minute', TIMESTAMP '2023-10-05 14:30:45');
     date_trunc
---------------------
 05/10/2023 14:30:00
(1 строка)
```

Рисунок 24 – Эксперименты с date_trunc

14 Задание 28

На рисунке 25 показана работа с extract.

```
dbt_pw2=# SELECT EXTRACT(YEAR FROM TIMESTAMP '2023-10-05 14:30:45');
 extract
---------
    2023
(1 строка)


dbt_pw2=# SELECT EXTRACT(MONTH FROM TIMESTAMP '2023-10-05 14:30:45');
 extract
---------
      10
(1 строка)


dbt_pw2=# SELECT EXTRACT(DAY FROM INTERVAL '5 days 12 hours 34 minutes');
 extract
---------
       5
(1 строка)


dbt_pw2=# SELECT EXTRACT(HOUR FROM INTERVAL '5 days 12 hours 34 minutes');
 extract
---------
      12
(1 строка)
```

Рисунок 25 – Применение extract

15 Задание 30

На рисунке 26 показана результат выполнения множества команд. Некоторые из них выдают ошибку: 2, 3, 4, 6, 8, 9. Все они связаны с тем, что postgres не может привести указанные типы данных к тем, что были указаны в таблице. Где-то забыты кавычки, где-то используется неуместное выражение, где-то необходимо явное приведение.

```
dbt_pw2=# CREATE TABLE test_bool ( a boolean, b text );
CREATE TABLE
dbt_pw2=# INSERT INTO test_bool VALUES ( TRUE, 'yes' ); INSERT INTO test_bool VALUES ( yes, 'yes' ); INSERT INTO test_bo
ol VALUES ( 'yes', true ); INSERT INTO test_bool VALUES ( 'yes', TRUE ); INSERT INTO test_bool VALUES ( '1', 'true' ); I
NSERT INTO test_bool VALUES ( 1, 'true' ); INSERT INTO test_bool VALUES ( 't', 'true' ); INSERT INTO test_bool VALUES (
't', truth ); INSERT INTO test_bool VALUES ( true, true ); INSERT INTO test_bool VALUES ( 1::boolean, 'true' ); INSERT I
NTO test_bool VALUES ( 111::boolean, 'true' );
INSERT 0 1
ОШИБКА:  столбец "yes" не существует
СТРОКА 1: INSERT INTO test_bool VALUES ( yes, 'yes' );
                                         ^
INSERT 0 1
INSERT 0 1
INSERT 0 1
ОШИБКА:  столбец "a" имеет тип boolean, а выражение - integer
СТРОКА 1: INSERT INTO test_bool VALUES ( 1, 'true' );
                                         ^
ПОДСКАЗКА:  Перепишите выражение или преобразуйте его тип.
INSERT 0 1
ОШИБКА:  столбец "truth" не существует
СТРОКА 1: INSERT INTO test_bool VALUES ( 't', truth );
                                             ^
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

Рисунок 26 – Результат выполнения множества команд

16 Задание 32

На рисунках 27, 28, 29 показаны функции и операции, применяемые к массивам.

```
dbt_pw2=# SELECT string_to_array('1,2,3', ',');
 string_to_array
-----------------
 {1,2,3}
(1 строка)


dbt_pw2=# SELECT unnest(ARRAY[1, 2, 3]);
 unnest
--------
      1
      2
      3
(3 строки)


dbt_pw2=# SELECT ARRAY[1, 2, 3] @> ARRAY[2, 3];
 ?column?
----------
 t
(1 строка)


dbt_pw2=# SELECT ARRAY[2, 3] <@ ARRAY[1, 2, 3];
 ?column?
----------
 t
(1 строка)


dbt_pw2=# SELECT ARRAY[1, 2, 3] && ARRAY[3, 4, 5];
 ?column?
----------
 t
(1 строка)


dbt_pw2=# SELECT array_position(ARRAY[1, 2, 3, 2], 2);
 array_position
----------------
              2
(1 строка)
```

Рисунок 27 – Операции с массивами, часть 1

```
dbt_pw2=# SELECT array_ndims(ARRAY[[1, 2], [3, 4]]);
 array_ndims
-------------
           2
(1 строка)


dbt_pw2=# SELECT array_dims(ARRAY[[1, 2], [3, 4]]);
 array_dims
------------
 [1:2][1:2]
(1 строка)


dbt_pw2=# SELECT array_lower(ARRAY[1, 2, 3], 1);
 array_lower
-------------
           1
(1 строка)


dbt_pw2=# SELECT array_upper(ARRAY[1, 2, 3], 1);
 array_upper
-------------
           3
(1 строка)


dbt_pw2=# SELECT array_fill(7, ARRAY[3]);
 array_fill
------------
 {7,7,7}
(1 строка)


dbt_pw2=# SELECT array_to_string(ARRAY[1, 2, 3], ', ');
 array_to_string
-----------------
 1, 2, 3
(1 строка)
```

Рисунок 28 – Операции с массивами, часть 2

```
dbt_pw2=# SELECT ARRAY[1, 2] || ARRAY[3, 4];
 ?column?
-----------
 {1,2,3,4}
(1 строка)


dbt_pw2=# SELECT array_append(ARRAY[1, 2], 3);
 array_append
--------------
 {1,2,3}
(1 строка)


dbt_pw2=# SELECT array_prepend(0, ARRAY[1, 2]);
 array_prepend
---------------
 {0,1,2}
(1 строка)


dbt_pw2=# SELECT array_remove(ARRAY[1, 2, 3, 2], 2);
 array_remove
--------------
 {1,3}
(1 строка)


dbt_pw2=# SELECT array_replace(ARRAY[1, 2, 3, 2], 2, 99);
 array_replace
---------------
 {1,99,3,99}
(1 строка)


dbt_pw2=# SELECT array_length(ARRAY[1, 2, 3], 1);
 array_length
--------------
            3
(1 строка)
```

Рисунок 29 – Операции с массивами, часть 3

17 Задание 34

Изменение значений по ключу с помощью функци jsonb_set представлено на рисунке 30.

```
dbt_pw2=# select * from pilot_hobbies;
 pilot_name |                                  hobbies
------------+-------------------------------------------------------------------------------
 Ivan       | {"trips": 3, "sports": ["футбол", "плавание"], "home_lib": true}
 Petr       | {"trips": 2, "sports": ["теннис", "плавание"], "home_lib": true}
 Pavel      | {"trips": 4, "sports": ["плавание"], "home_lib": false}
 Boris      | {"trips": 0, "sports": ["футбол", "плавание", "теннис"], "home_lib": true}
(4 строки)


dbt_pw2=# UPDATE pilot_hobbies
dbt_pw2-# SET hobbies = jsonb_set(hobbies, '{home_lib}', 'false')
dbt_pw2-# WHERE pilot_name = 'Ivan';
UPDATE 1
dbt_pw2=# select * from pilot_hobbies;
 pilot_name |                                  hobbies
------------+-------------------------------------------------------------------------------
 Petr       | {"trips": 2, "sports": ["теннис", "плавание"], "home_lib": true}
 Pavel      | {"trips": 4, "sports": ["плавание"], "home_lib": false}
 Boris      | {"trips": 0, "sports": ["футбол", "плавание", "теннис"], "home_lib": true}
 Ivan       | {"trips": 3, "sports": ["футбол", "плавание"], "home_lib": false}
(4 строки)
```

Рисунок 30 – Применение jsonb_set

18 Задание 36

Добавление новых ключей json в таблице представлено на рисунке 31.

```
dbt_pw2=# update pilot_hobbies
dbt_pw2-# SET hobbies = hobbies || '{"new_key": "new_value"}'::jsonb
dbt_pw2-# WHERE pilot_name = 'Ivan';
UPDATE 1
dbt_pw2=# update pilot_hobbies
dbt_pw2-# SET hobbies = hobbies || '{"key1": "value1", "key2": "value2"}'::jsonb
dbt_pw2-# WHERE pilot_name = 'Petr';
UPDATE 1
dbt_pw2=# select * from pilot_hobbies;
 pilot_name |                                                  hobbies
------------+-----------------------------------------------------------------------------------------------------
 Pavel      | {"trips": 4, "sports": ["плавание"], "home_lib": false}
 Boris      | {"trips": 0, "sports": ["футбол", "плавание", "теннис"], "home_lib": true}
 Ivan       | {"trips": 3, "sports": ["футбол", "плавание"], "new_key": "new_value", "home_lib": false}
 Petr       | {"key1": "value1", "key2": "value2", "trips": 2, "sports": ["теннис", "плавание"], "home_lib": true}
(4 строки)
```

Рисунок 31 – Добавление новых ключей

ЗАКЛЮЧЕНИЕ

По результатам работы был изучен теоретический материал по теме «Типы данных СУБД PostgreSQL». Все поставленные цели и задачи были выполнены.