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 №9

Программирование на стороне сервера в среде СУБД PostgreSQL
тема

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

Студент КИ23-17/1б 032318988 _____ Александров Е.А.
номер зачетной книжки — подпись, дата — инициалы, фамилия

Красноярск 2025

1 Цель

Изучить основы программирования на стороне сервера в среде СУБД PostgreSQL. Выполнить указанные в файле задания.

2 Ход работы

Результат выполнения заданий показан на рисунках с 1 по 11.

Какая команда используется для удаления функции? Удалите какую-нибудь вашу функцию.

```
ais=# DROP FUNCTION IF EXISTS generate_students_data;
DROP FUNCTION
```

Рисунок 1 – Задание 2

Модифицируйте функцию count_letters(), подсчитывающую количество фамилий в таблице students («Студенты»), начинающихся на каждую букву. Сделайте так, чтобы в случае отсутствия в таблице фамилий, начинающихся с каких-то букв, в выводе функции эти буквы были представлены нулевыми (пустыми) значениями.

```
ais=# CREATE OR REPLACE FUNCTION count_letters()
ais-# RETURNS TABLE (letter char(1), num bigint) AS $$
ais$# BEGIN
ais$#     -- Возвращаем результат с использованием RIGHT JOIN для включения всех букв
ais$#     RETURN QUERY
ais$#     WITH alphabet AS (
ais$#         -- Создаём временную таблицу со всеми буквами русского алфавита
ais$#         SELECT unnest(ARRAY[
ais$#             'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й',
ais$#             'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф',
ais$#             'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы', 'Ь', 'Э', 'Ю', 'Я'
ais$#         ]::char[]) AS letter
ais$#     ),
ais$#     letter_counts AS (
ais$#         -- Подсчитываем количество фамилий по первой букве
ais$#         SELECT substr(name, 1, 1) AS letter, count(*) AS cnt
ais$#         FROM students
ais$#         GROUP BY substr(name, 1, 1)
ais$#     )
ais$#     -- Соединяем алфавит с подсчётами, заменяя NULL на 0
ais$#     SELECT a.letter, COALESCE(lc.cnt, 0) AS num
ais$#     FROM alphabet a
ais$#     LEFT JOIN letter_counts lc ON a.letter = lc.letter
ais$#     ORDER BY a.letter;
ais$# END;
ais$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

Рисунок 2 – Задание 6

Напишите триггер уровня строки (row-level) для таблицы «Студенты» или таблицы «Успеваемость».

```
ais=# CREATE TABLE students (
ais(#     mark_book numeric(5) NOT NULL, -- Номер зачётной книжки
ais(#     name text NOT NULL,            -- Ф.И.О.
ais(#     psp_ser numeric(4),            -- Серия паспорта
ais(#     psp_num numeric(6),            -- Номер паспорта
ais(#     PRIMARY KEY (mark_book)
ais(# );
CREATE TABLE
ais=#
ais=# CREATE TABLE students_audit (
ais(#     audit_id SERIAL PRIMARY KEY,
ais(#     mark_book numeric(5),          -- Номер зачётной книжки
ais(#     old_name text,                 -- Старое Ф.И.О.
ais(#     new_name text,                 -- Новое Ф.И.О.
ais(#     old_psp_ser numeric(4),        -- Старая серия паспорта
ais(#     new_psp_ser numeric(4),        -- Новая серия паспорта
ais(#     old_psp_num numeric(6),        -- Старый номер паспорта
ais(#     new_psp_num numeric(6),        -- Новый номер паспорта
ais(#     operation_type text,           -- Тип операции (INSERT/UPDATE)
ais(#     change_timestamp timestamp     -- Время изменения
ais(# );
CREATE TABLE
```

Рисунок 3 – Задание 10

```
ais=# CREATE OR REPLACE FUNCTION check_and_log_students()
ais-# RETURNS trigger AS $$
ais$# BEGIN
ais$#     -- Проверка корректности серии паспорта (для INSERT и UPDATE)
ais$#     IF (NEW.psp_ser IS NOT NULL) THEN
ais$#         IF (NEW.psp_ser < 1000 OR NEW.psp_ser > 9999) THEN
ais$#             RAISE EXCEPTION 'Серия паспорта должна быть в диапазоне от 1000 до 9999, получено: %', NEW.psp_ser;
ais$#         END IF;
ais$#     END IF;
ais$#
ais$#     -- Логирование в зависимости от операции
ais$#     IF (TG_OP = 'INSERT') THEN
ais$#         INSERT INTO students_audit (
ais$#             mark_book, new_name, new_psp_ser, new_psp_num, operation_type, change_timestamp
ais$#         ) VALUES (
ais$#             NEW.mark_book, NEW.name, NEW.psp_ser, NEW.psp_num, 'INSERT', CURRENT_TIMESTAMP
ais$#         );
ais$#         RETURN NEW; -- Разрешаем вставку
ais$#
ais$#     ELSIF (TG_OP = 'UPDATE') THEN
ais$#         -- Логируем старые и новые значения
ais$#         INSERT INTO students_audit (
ais$#             mark_book, old_name, new_name, old_psp_ser, new_psp_ser,
ais$#             old_psp_num, new_psp_num, operation_type, change_timestamp
ais$#         ) VALUES (
ais$#             OLD.mark_book, OLD.name, NEW.name, OLD.psp_ser, NEW.psp_ser,
ais$#             OLD.psp_num, NEW.psp_num, 'UPDATE', CURRENT_TIMESTAMP
ais$#         );
ais$#         RETURN NEW; -- Разрешаем обновление
ais$#
ais$#     END IF;
ais$#
ais$#     RETURN NULL; -- На случай, если TG_OP не INSERT и не UPDATE (хотя здесь это не произойдёт)
ais$# END;
ais$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
ais=#
ais=# CREATE TRIGGER students_check_and_log
ais-# BEFORE INSERT OR UPDATE ON students
ais-# FOR EACH ROW EXECUTE FUNCTION check_and_log_students();
CREATE TRIGGER
```

Рисунок 4 – Задание 10

```
ais=# INSERT INTO students (mark_book, name, psp_ser, psp_num)
ais-# VALUES (10001, 'Иванов Иван Иванович', 1234, 567890);
INSERT 0 1
ais=#
ais=# INSERT INTO students (mark_book, name, psp_ser, psp_num)
ais-# VALUES (10002, 'Петров Пётр Петрович', 999, 123456);
ОШИБКА:  Серия паспорта должна быть в диапазоне от 1000 до 9999, получено: 999
КОНТЕКСТ:  функция PL/pgSQL check_and_log_students(), строка 6, оператор RAISE
ais=#
ais=# UPDATE students
ais-# SET name = 'Иванов Иван Сергеевич', psp_ser = 4321
ais-# WHERE mark_book = 10001;
UPDATE 1
ais=#
ais=# SELECT * FROM students_audit;
 audit_id | mark_book |       old_name       |       new_name       | old_psp_ser | new_psp_ser | old_psp_num | new_psp_num | operation_type |      change_timestamp
----------+-----------+----------------------+----------------------+-------------+-------------+-------------+-------------+----------------+----------------------------
        1 |     10001 |                      | Иванов Иван Иванович |             |        1234 |             |      567890 | INSERT         | 2025-03-05 21:01:40.115166
        2 |     10001 | Иванов Иван Иванович | Иванов Иван Сергеевич |        1234 |        4321 |      567890 |      567890 | UPDATE         | 2025-03-05 21:01:58.898971
(2 строки)
```

Рисунок 5 – Задание 10

Сделайте выборки данных из таблиц «Персонал» и «Органи зационная структура», а также реконструируйте организационную структуру с помощью двух представлений (view). Команды можно выполнять не только в среде интерактивного терминала psql, но также и из командной строки операционной системы. Выполните эти команды в командной строке операционной системы: psql -d ais -c "SELECT * FROM Personnel" psql -d ais -c "SELECT * FROM

Org_chart" psql -d ais -c "SELECT * FROM Personnel_org_chart" psql -d ais -c "SELECT * FROM Create_paths"

```
ais=# SELECT * FROM Personnel;
 emp_nbr | emp_name |         emp_addr          | birth_date
---------+----------+---------------------------+------------
       0 | вакансия |                           | 2014-05-19
       1 | Иван     | ул. Любителей языка C     | 1962-12-01
       2 | Петр     | ул. UNIX гуру             | 1965-10-21
       3 | Антон    | ул. Ассемблерная          | 1964-04-17
       4 | Захар    | ул. им. СУБД PostgreSQL   | 1963-09-27
       5 | Ирина    | просп. Программистов      | 1968-05-12
       6 | Анна     | пер. Перловый             | 1969-03-20
       7 | Андрей   | пл. Баз данных            | 1945-11-07
       8 | Николай  | наб. ОС Linux             | 1944-12-01
       9 | Мария    | ул. SQL                   | 1970-01-15
      10 | Олег     | пр. Алгоритмов            | 1980-03-22
      11 | Елена    | пер. Базовый              | 1985-07-10
(12 строк)


ais=# SELECT * FROM Org_chart;
    job_title     | emp_nbr | boss_emp_nbr |  salary
------------------+---------+--------------+-----------
 Президент        |       1 |              | 1000.0000
 Вице-президент 1 |       2 |            1 |  900.0000
 Программист C    |       6 |            4 |  500.0000
 Тестировщик      |      10 |            7 |  420.0000
 Стажёр           |      11 |           10 |  300.0000
 Программист Perl |       7 |            6 |  450.0000
 Оператор         |       8 |            6 |  400.0000
 Архитектор       |       4 |            6 |  700.0000
 Аналитик         |       9 |            6 |  550.0000
(9 строк)
```

Рисунок 6 – Задание 12

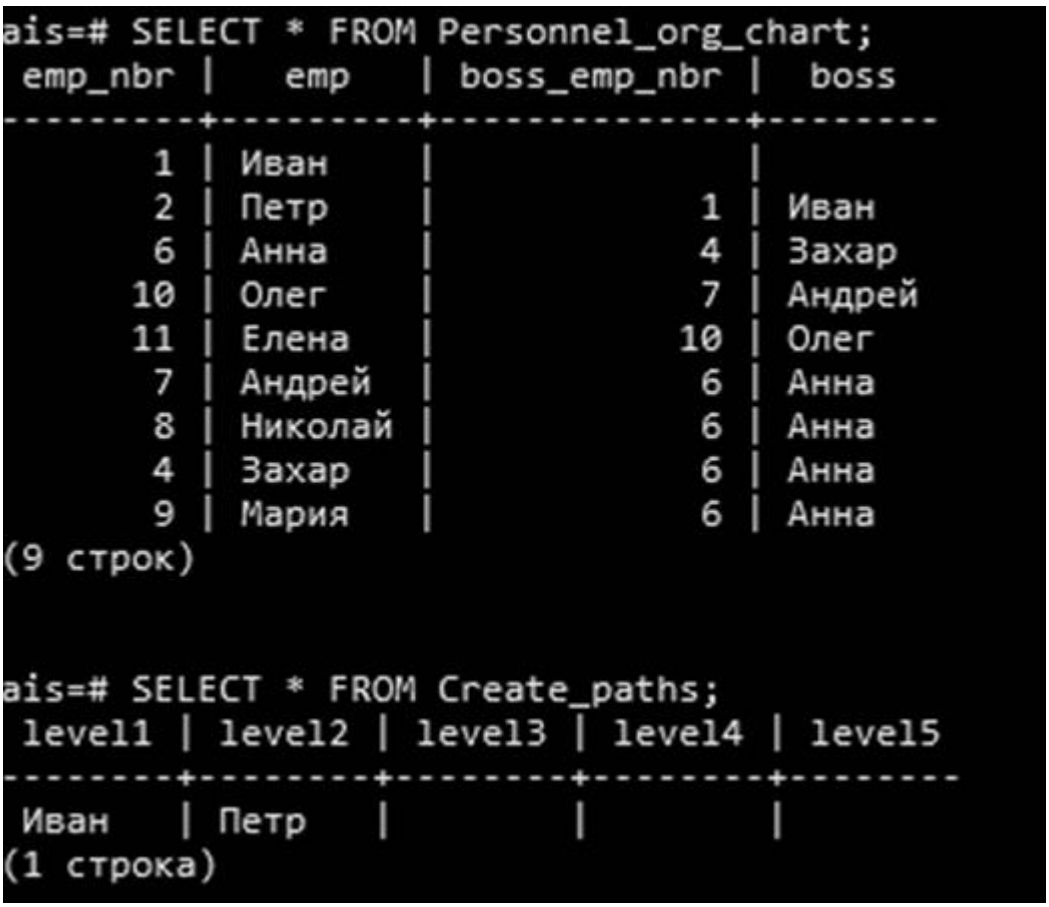

Рисунок 7 – Задание 12

Выполните обход дерева организационной структуры снизу вверх, начиная с конкретного узла, можно с помощью функции up_tree_traversal() либо функции up_tree_traversal2(). Сначала сде лайте это с помощью первой из функций: SELECT * FROM up_tree_traversal(6); Параметром этих функций является код работника. Измените код работника и повторите команду. Теперь воспользуйтесь второй функцией. Учтите, что она воз вращает SETOF RECORD, поэтому команда будет более сложной: SELECT * FROM up_tree_traversal2(6) AS (emp int, boss int); Очевидно, что для использования числового кода работника нужно знать этот код. Удобнее иметь дело с именем работника. Поэтому можно в качестве параметра этих функций использовать подзапрос, возвращающий код работника в качестве своего резуль тата. Не забудьте, что текст подзапроса заключается в скобки, поэто му появляются двойные скобки: SELECT * FROM up_tree_traversal((SELECT … FROM Personnel WHERE …));

```
ais=# SELECT * FROM up_tree_traversal( ( SELECT emp_nbr FROM Personnel WHERE emp_name = 'Антон'));
 emp_nbr | boss_emp_nbr
---------+--------------
       3 |            2
       2 |            1
       1 |
(3 строки)
```

Рисунок 8 – Задание 14

Если в таблице «Организационная структура» осталось мало данных, то дополните ее данными и выполните удаление элемента иерархии и продвижение дочерних элементов на один уровень вверх (т. е. к «бабушке»). SELECT * FROM delete_and_promote_subtree(5); Аналогично работе с функцией up_tree_traversal() используйте под запрос для получения кода работника по его имени. После удаления элемента иерархии посмотрите, что стало с организационной структурой, с помощью двух представлений Personnel_org_chart и Create_paths.

```
ais=# INSERT INTO Personnel (emp_nbr, emp_name, emp_addr, birth_date) VALUES
ais-#     (9, 'Мария', 'ул. SQL', '1970-01-15'),
ais-#     (10, 'Олег', 'пр. Алгоритмов', '1980-03-22'),
ais-#     (11, 'Елена', 'пер. Базовый', '1985-07-10');
INSERT 0 3
ais=#
ais=# INSERT INTO Org_chart (job_title, emp_nbr, boss_emp_nbr, salary) VALUES
ais-#     ('Аналитик', 9, 5, 550.00),         -- Подчиняется Ведущему программисту (5)
ais-#     ('Тестировщик', 10, 7, 420.00),     -- Подчиняется Программисту Perl (7)
ais-#     ('Стажёр', 11, 10, 300.00);         -- Подчиняется Тестировщику (10)
INSERT 0 3
ais=#
ais=# SELECT * FROM org_chart;
      job_title      | emp_nbr | boss_emp_nbr |  salary
---------------------+---------+--------------+-----------
 Президент           |       1 |              | 1000.0000
 Вице-президент 1    |       2 |            1 |  900.0000
 Программист Perl    |       7 |            5 |  450.0000
 Оператор            |       8 |            5 |  400.0000
 Архитектор          |       4 |            5 |  700.0000
 Ведущий программист |       5 |            6 |  600.0000
 Программист C       |       6 |            4 |  500.0000
 Аналитик            |       9 |            5 |  550.0000
 Тестировщик         |      10 |            7 |  420.0000
 Стажёр              |      11 |           10 |  300.0000
(10 строк)
```

Рисунок 9 – Задание 16

```
ais=# SELECT * FROM delete_and_promote_subtree(5);
 delete_and_promote_subtree
----------------------------

(1 строка)


ais=# SELECT * FROM orgchart;
ОШИБКА:  отношение "orgchart" не существует
СТРОКА 1: SELECT * FROM orgchart;
                       ^
ais=# SELECT * FROM org_chart;
    job_title     | emp_nbr | boss_emp_nbr |  salary
------------------+---------+--------------+-----------
 Президент        |       1 |              | 1000.0000
 Вице-президент 1 |       2 |            1 |  900.0000
 Программист C    |       6 |            4 |  500.0000
 Тестировщик      |      10 |            7 |  420.0000
 Стажёр           |      11 |           10 |  300.0000
 Программист Perl |       7 |            6 |  450.0000
 Оператор         |       8 |            6 |  400.0000
 Архитектор       |       4 |            6 |  700.0000
 Аналитик         |       9 |            6 |  550.0000
(9 строк)
```

Рисунок 10 – Задание 16

```
ais=# SELECT * FROM delete_and_promote_subtree( (SELECT emp_nbr FROM personnel WHERE emp_name = 'Ирина') );
 delete_and_promote_subtree
----------------------------

(1 строка)


ais=# SELECT * FROM org_chart;
    job_title     | emp_nbr | boss_emp_nbr |  salary
------------------+---------+--------------+-----------
 Президент        |       1 |              | 1000.0000
 Вице-президент 1 |       2 |            1 |  900.0000
 Программист C    |       6 |            4 |  500.0000
 Тестировщик      |      10 |            7 |  420.0000
 Стажёр           |      11 |           10 |  300.0000
 Программист Perl |       7 |            6 |  450.0000
 Оператор         |       8 |            6 |  400.0000
 Архитектор       |       4 |            6 |  700.0000
 Аналитик         |       9 |            6 |  550.0000
(9 строк)
```

Рисунок 11 – Задание 16

3 Заключение

В ходе практической работы были изучены основы программирования на стороне сервера в среде СУБД PostgreSQL, были выполнены указанные в файле задания.