|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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** | | | | | |
| Программирование на стороне сервера в среде 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

## Цель работы

Изучить теоретический материал по теме «Программирование на стороне сервера в среде СУБД PostgreSQL». Выполнить задания.

## Задачи

В рамках данной практической работы необходимо выполнить следующие задачи:

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

## Задание

Задание данной практической работы состоит из следующих частей:

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

# ХОД РАБОТЫ

## Задание 1

Вызов функции можно использовать в подзапросе в предложении FROM.

Напишите такой запрос с подзапросом на примере функции generate\_students\_data().

На рисунке 1 показан результат выполнения задания.

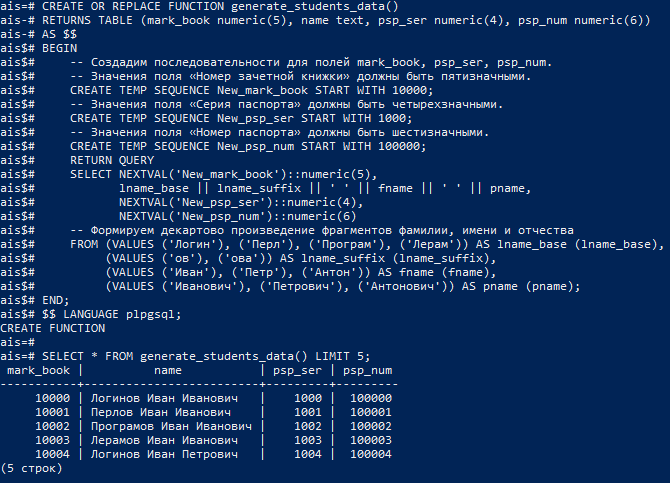


Рисунок 1 – Создание функции, применение её в запросе

## Задание 2

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

На рисунке 2 показан результат выполнения задания.



Рисунок 2 – Удаление функции

## Задание 3

Для чего нужны модификаторы (ключевые слова) IN и OUT перед именами параметров функций?

Ответ. Ключевые слова IN и OUT служат для указания входных и выходных параметров для функции.

## Задание 4

С помощью интерактивного терминала psql посмотрите список всех функций, созданных в вашей базе данных.

На рисунке 3 показан результат выполнения задания.

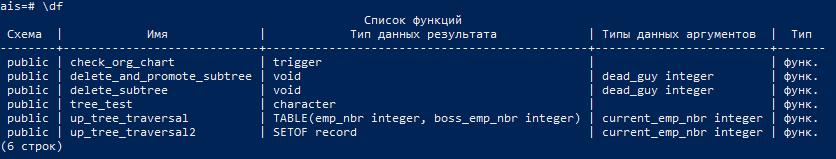


Рисунок 3 – Список всех функций

## Задание 5

Каким образом можно сделать так, чтобы функция возвратила табличное значение?

Чтобы функция возвратила табличное значение, необходимо в RETURNS указать TABLE (…), где в скобках перечислить возвращаемые столбцы.

## Задание 6

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.

На рисунках 4, 5 показан результат выполнения задания.

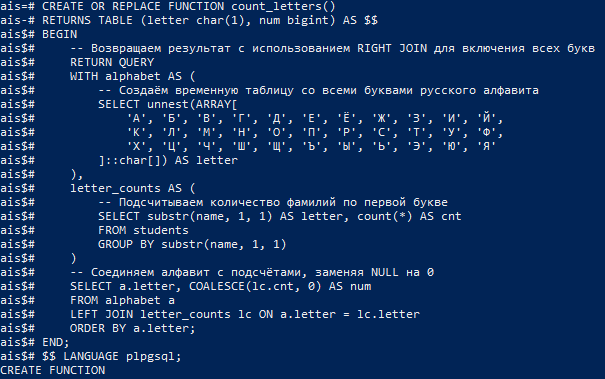


Рисунок 4 – Функция

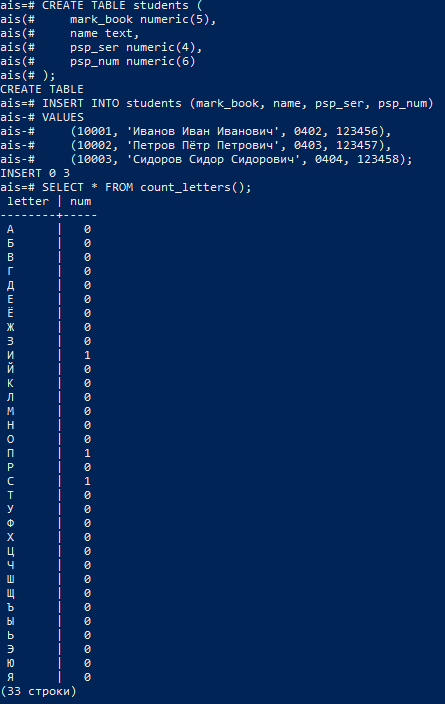


Рисунок 5 – Результат

## Задание 7

Что такое триггер?

Триггер – это набор действий, выполняемый в случае возникновения определённых событий, привязанный к таблицам или представлениям. Событиями могут быть выполняемые операции над таблицей: SELECT, INSERT, UPDATE, DELETE.

## Задание 8

Какие особенности имеет триггерная функция?

Триггер-функция может выполняться до операции и после операции. В представлениях заменяет собой операцию по умолчанию. На уровне затрагивания триггер-функция может выполняться для каждой строки операции или единожды для всей операции в целом, даже если строк не было получено. Триггерная функция лишь вызывает другую функцию, которая выполняет всю работу. В триггерной функции доступны новые (NEW) и старые (OLD) представления строк для взаимодействия.

## Задание 9

Чем триггеры уровня строки (row-level) отличаются от триггеров уровня команды (statement-level)?

Row-level говорит выполняться триггер-функции каждую строчку. Statement-level говорит выполняться триггер-функции единожды за операцию.

## Задание 10

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

На рисунках с 57 по 60 показан результат работы.

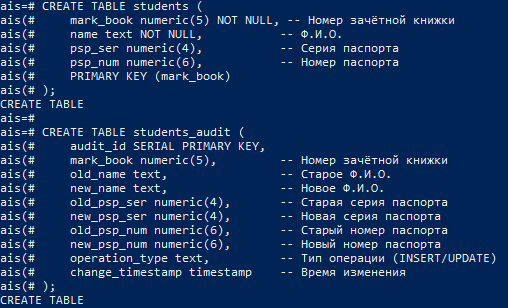


Рисунок 6 – Создание таблиц

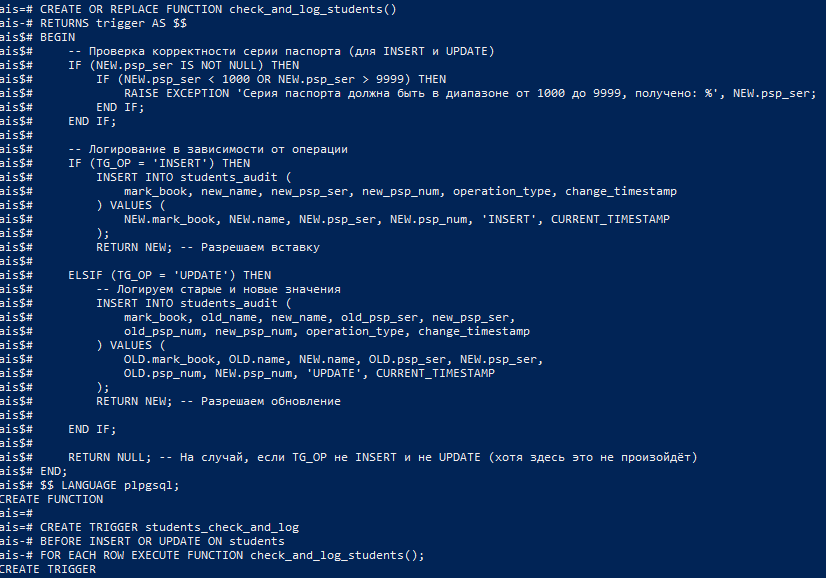


Рисунок 7 – Создание и привязка триггер-функции

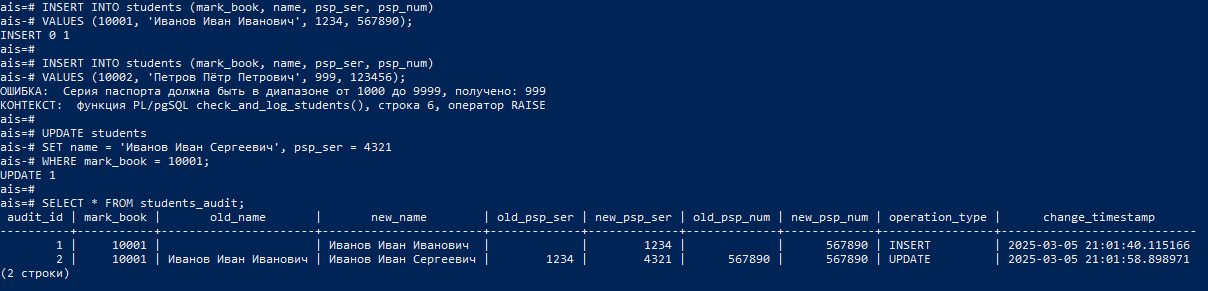


Рисунок 8 – Тестирование функции

## Задание 11

В базе данных ais создайте таблицы, функции и триггеры, необходимые для изучения метода хранения иерархий в реляционных базах данных.

Для этого можно поступить следующим образом:

Сначала создайте текстовый файл с именем, например, adj\_list.sql, содержащий все команды и определения функций, приведенные в тексте пособия.

Затем выполните команду в среде операционной системы:

psql -d ais -f adj\_list.sql

Возьмём структуры из главы 4. На рисунке 9 представлен прогресс работы.

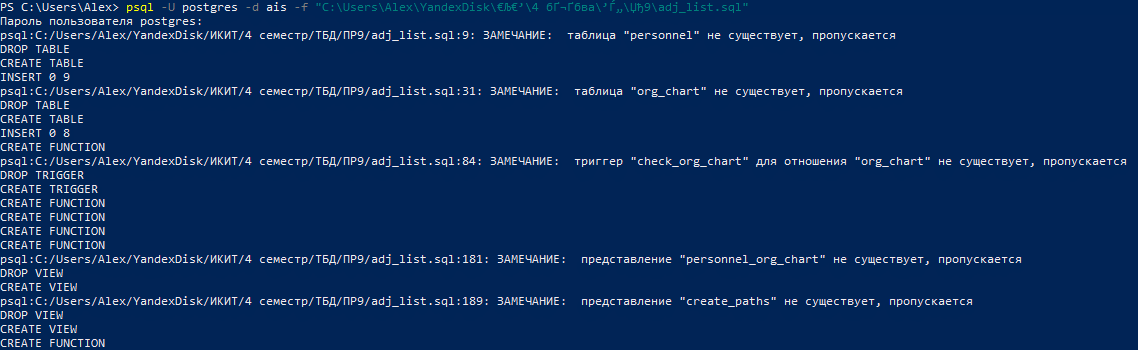


Рисунок 9 – Прогресс работы

## Задание 12

Сделайте выборки данных из таблиц «Персонал» и «Организационная 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:

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"

Примечание:

Если не указан параметр -U, утилита psql подключается к базе данных от имени пользователя ОС. Возможно, потребуется использовать параметр -U, если в БД нет соответствующей учетной записи.

На рисунках 10, 11 представлен прогресс работы.

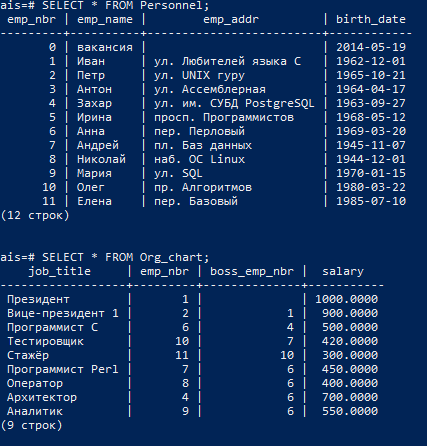


Рисунок 10 – Запросы, часть 1

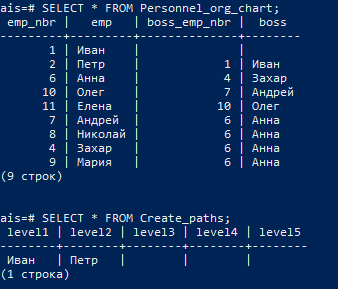


Рисунок 11 – Запросы, часть 2

## Задание 13

Выполните проверку структуры дерева на отсутствие циклов командой:

SELECT \* FROM tree\_test();

При первоначальной проверке без изменений в таблице "Организационная структура" функция покажет корректную древовидную структуру.

Для тестирования:

Создайте короткий цикл в таблице

Создайте длинный цикл в таблице

После каждого изменения проверяйте структуру:

SELECT \* FROM tree\_test();

На рисунках с 12 по 14 показан прогресс работы.

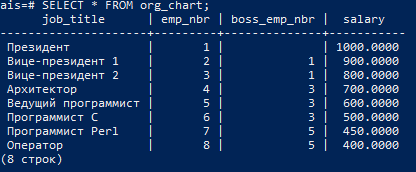


Рисунок 12 – Исходная таблица

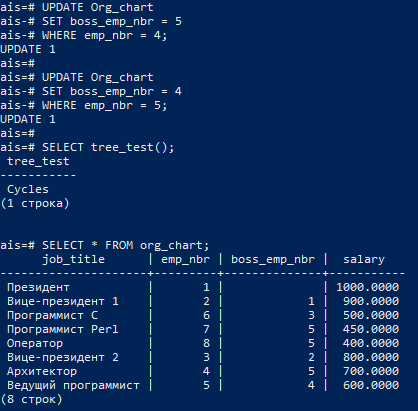


Рисунок 13 – Создание короткого цикла

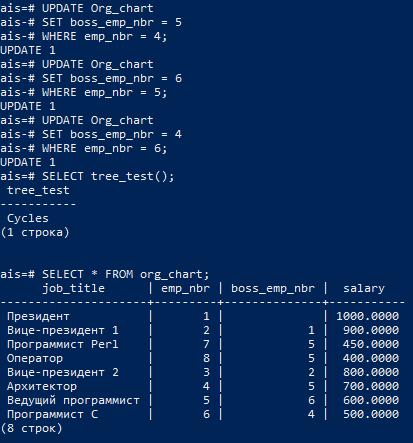


Рисунок 14 – Создание длинного цикла

## Задание 14

Выполните обход дерева организационной структуры снизу вверх, начиная с конкретного узла, можно с помощью функции up\_tree\_traversal() либо функции up\_tree\_traversal2().

Сначала сделайте 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 возвращает 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 результата. Не забудьте, что текст подзапроса заключается в скобки, поэтому появляются двойные скобки:

SELECT \* FROM up\_tree\_traversal((SELECT ... FROM Personnel WHERE ...));

Завершите эту команду и выполните ее с различными именами работников.

На рисунке 15 показан прогресс работы.

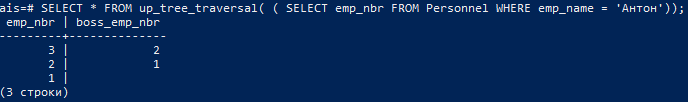


Рисунок 15 – Обход дерева по имени сотрудника

## Задание 15

Выполните операцию удаления поддерева с помощью функции delete\_subtree(). Параметром функции является код работника.

SELECT \* FROM delete\_subtree(6);

Аналогично работе с функцией up\_tree\_traversal() используйте подзапрос для получения кода работника по его имени:

SELECT \* FROM delete\_subtree((SELECT emp\_id FROM Personnel WHERE emp\_name = 'Иванов'));

После удаления поддерева проверьте изменения в организационной структуре с помощью представлений:

SELECT \* FROM Personnel\_org\_chart;

SELECT \* FROM Create\_paths;

На рисунках 16, 17 показан прогресс работы.

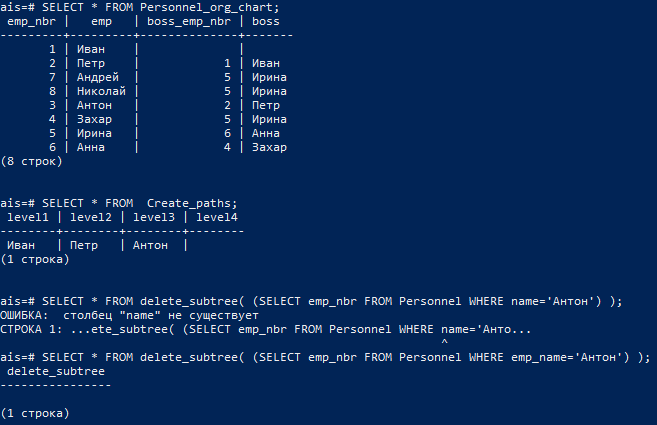


Рисунок 16 – Удаление сотрудника по имени

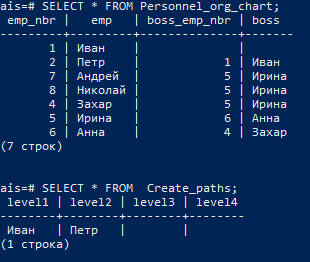


Рисунок 17 – Проверка

## Задание 16

Если в таблице «Организационная структура» осталось мало данных, выполните следующие действия:

Дополните таблицу необходимыми данными

Выполните удаление элемента иерархии с продвижением дочерних элементов:

SELECT \* FROM delete\_and\_promote\_subtree(5);

Для удобства можно использовать подзапрос для получения кода работника по имени (аналогично функции up\_tree\_traversal()):

SELECT \* FROM delete\_and\_promote\_subtree(

(SELECT emp\_id FROM Personnel WHERE emp\_name = 'Фамилия')

);

После выполнения операции проверьте изменения в организационной структуре:

SELECT \* FROM Personnel\_org\_chart;

SELECT \* FROM Create\_paths;

На рисунках с 18 по 20 представлен прогресс работы.

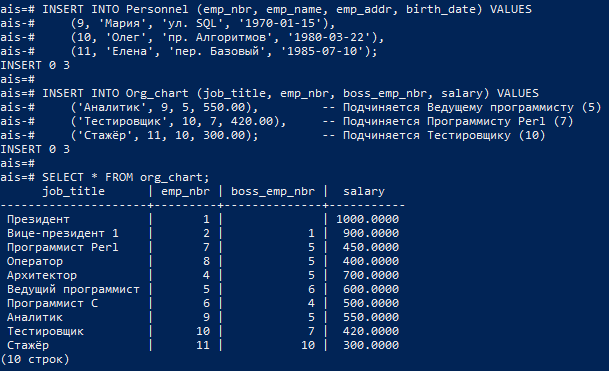


Рисунок 18 – Добавление сотрудников

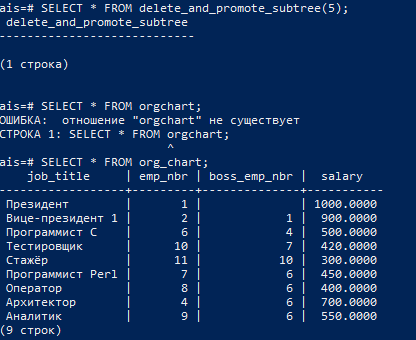


Рисунок 19 – Удаление по номеру

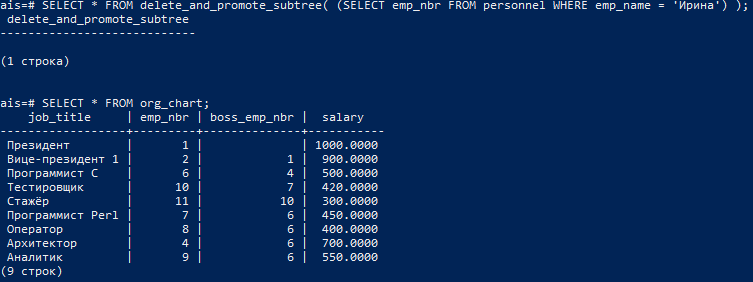


Рисунок 20 – Удаление по имени

## Задание 17

Представление Create\_paths позволяет отобразить только четыре уровня иерархии. Модифицируйте его так, чтобы оно могло работать с пятью уровнями иерархии.

На рисунке 19 показан прогресс работы.

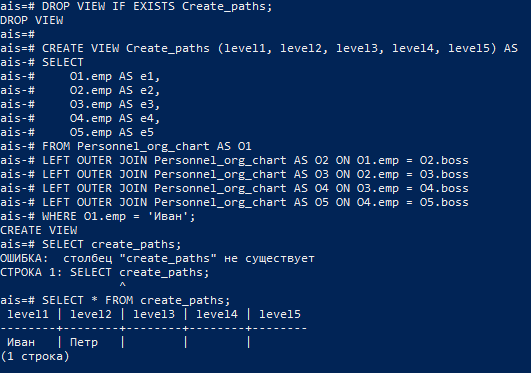


Рисунок 21 – Результат

## Задание 18

Самостоятельно ознакомьтесь с курсорами (cursors) - средством работы с таблицами базы данных. Используйте техническую документацию PostgreSQL, главу «PL/pgSQL – SQL Procedural Language».

На рисунках 22, 23 показан прогресс работы.

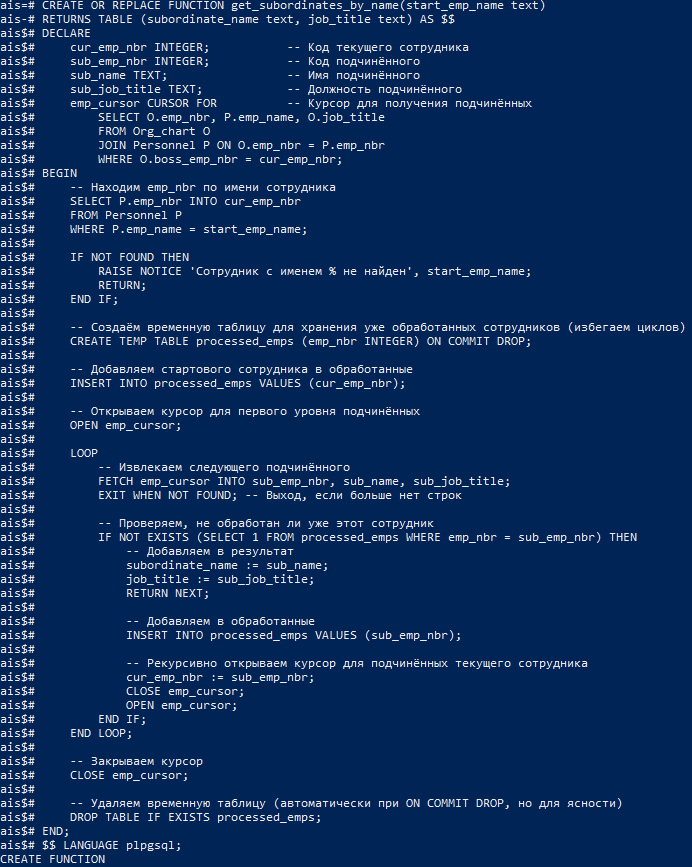


Рисунок 22 – Функция

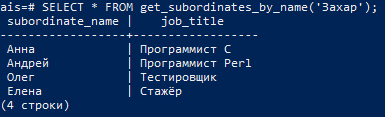


Рисунок 23 – Результат

## Задание 19

Самостоятельно ознакомьтесь с правилами (rules) - средством работы с таблицами базы данных. Используйте техническую документацию PostgreSQL, главу «The Rule System».

На рисунках с 24 по 26 показан прогресс работы.

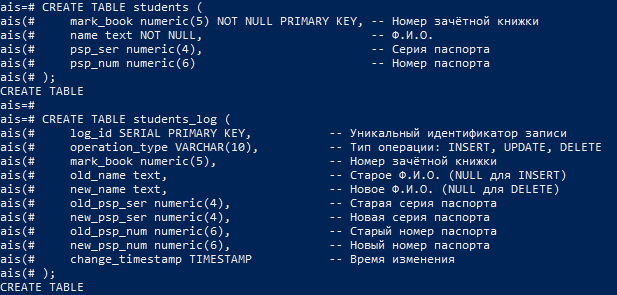


Рисунок 24 – Создание таблиц

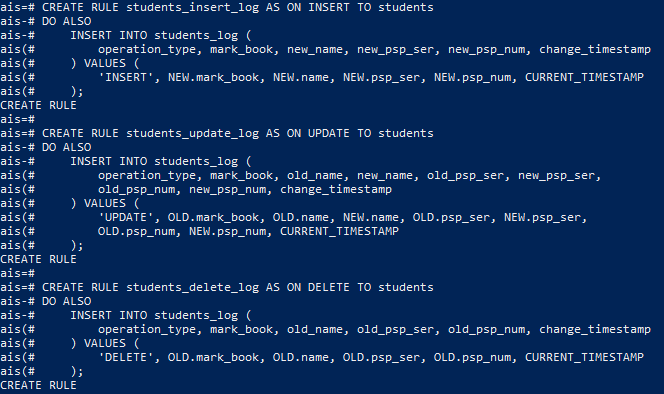


Рисунок 25 – Создание правил

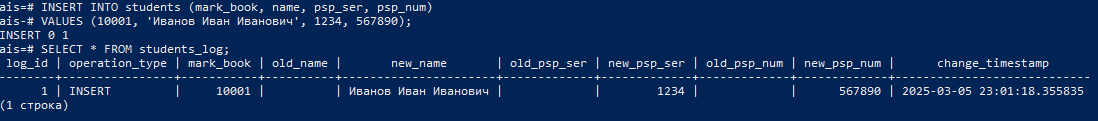


Рисунок 26 – Проверка

# ЗАКЛЮЧЕНИЕ

По результатам работы был изучен теоретический материал по теме «Программирование на стороне сервера в среде СУБД PostgreSQL». Все поставленные цели и задачи были выполнены.