|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ | | | | | | | | | | |
|  | | | |  | | |  | | | |
| ФГБОУ ВПО «Пермский государственный национальный исследовательский университет» | | | | | | | | | | |
|  | |  | | | | | | |  | |
|  | | ОТЧЕТ  по лабораторной работе «Древовидное представление» по дисциплине «Учебная практика по базам данных» | | | | | | |  | |
|  | | |  | | | | |  | | |
|  | Работу выполнил  студент гр. ПМИ-1,2  Васильевых М.Г. \_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 | | | |  | Проверил  ассистент кафедры МОВС  Турова И.А. \_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 | | | |  |
|  | |  | | | | | | |  | |
|  | | Пермь 2021 | | | | | | |  | |

Содержание

[1 Постановка задачи 3](#_Toc82955818)

[2 Описание предметной области и базы данных 4](#_Toc82955819)

[3 Описание функционала разработанной программы 5](#_Toc82955820)

[3.1. Внешний вид и функционал приложения 5](#_Toc82955821)

[3.2 Общий алгоритм решения задачи 5](#_Toc82955822)

[4 Реализация программы 7](#_Toc82955823)

[5 Тестирование программы 10](#_Toc82955824)

# 1 Постановка задачи

Создать и наполнить PostgreSQL базу данных для некоторой предметной области, состоящую из трёх таблиц, последовательно соединённых связью 1:M (аналогично страна – город – пользователь). Определить первичные и внешние ключи.

Написать программу, которая на графическом интерфейсе пользователя строит древовидный список по данным из ранее спроектированной базы данных, в котором элементы каждого из уровней иерархии отражают иерархию записей, хранящихся в базе данных (3 уровня).

# 2 Описание предметной области и базы данных

Для работы приложения нам потребуются три связанные таблицы из базы данных «Филиалы корпорации» - рисунок 1.1

В базе данных хранится информация о корпорации, количестве её работников, улица, город и страна где расположен филиал. Представленная база данных реляционного типа, в виде трёх последовательно связанных таблиц при помощи связи “Один ко многим”, структура представлена рисунком 1.1.

В первой таблице под названием “country\_branch” хранится информация о стране, где расположены филиалы данной корпорации.

В следующей таблице “city\_branch” хранится информация о городе, где располагаются филиалы данной корпорации.

И в последней таблице под названием “branches” хранится информация о филиале: адрес и количество работников.

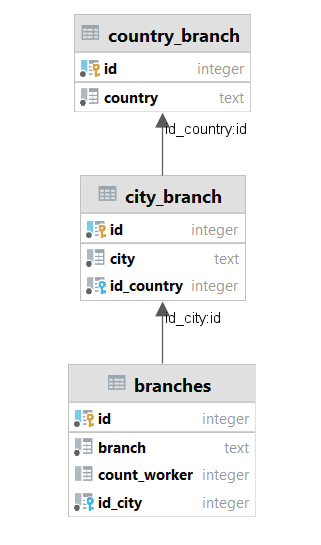


Рисунок 1.1 – Используемая БД

# 3 Описание функционала разработанной программы

3.1. Внешний вид и функционал приложения

При запуске приложения отображается главная форма. На ней расположен элемент типа TreeView.

Элементы верхнего уровня иерархии отображают названия стран, второго – названия городов, третьего – информацию о филиале <адрес филиала> <количество работников>. Внешний вид приложения представлен на рис. 1.

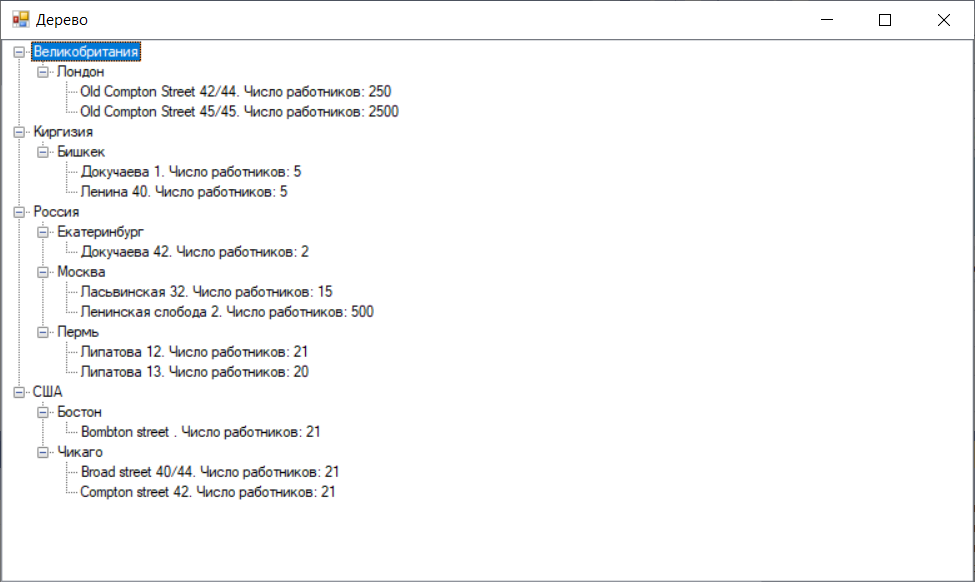


Рис. 1. Главная форма приложения

## 3.2 Общий алгоритм решения задачи

Алгоритм решения данной задачи можно разбить на несколько этапов:

1) С использованием трёх таблиц необходимо подготовить выборку

данных из них, в соответствии со столбцами

1. название страны
2. идентификатор страны
3. название города
4. идентификатор города
5. адрес филиала
6. идентификатор филиала
7. количество работников в филиале

При отборе данных необходимо устанавливать соответствие в

идентификаторах, а именно: идентификатор страны в таблице стран должен

совпадать с идентификатором страны в таблице города и т.д.

2)Переходя к построению древовидного представления, мы просматриваем каждую строчку поочередно, разбивая каждую строку БД на части, а именно на название страны, название города, адрес филиала, количество работников в филиале. Затем начиная с уровня стран, проверяем, был ли узел с таким же именем, если нет, то создаем новый, если был, то переходим к нему и проводим расширение иерархической структуры.

3)Аналогично поступаем с нижними уровнями, а именно, с названиями

городов и адресами филиалов, добавляя их на соответствующий уровень

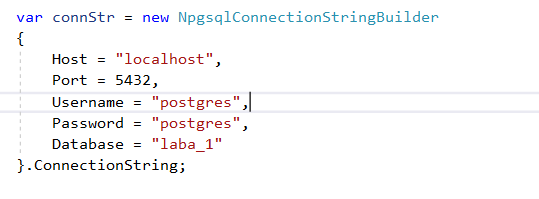
иерархии в дереве.

4)После добавления всех необходимых элементов сформируется дерево.

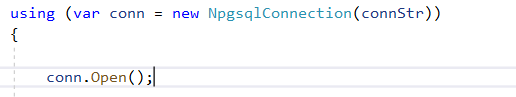
# 4 Реализация программы

В первую очередь, создается строка подключения к БД. Для этого

необходимо создать экземпляр класса NpgsqlConnectionStringBuilder в который будет передана информацию о БД, а именно: адрес сервера, инициализируемый каталог (каталог, содержащий БД) и данные для аутентификации, в качестве которого, в нашем случае, выступает проверка подлинности Windows.



Затем открываем соединение с БД, сначала создав экземпляр объекта

NpgsqlConnection, а затем применив к нему метод Open. 

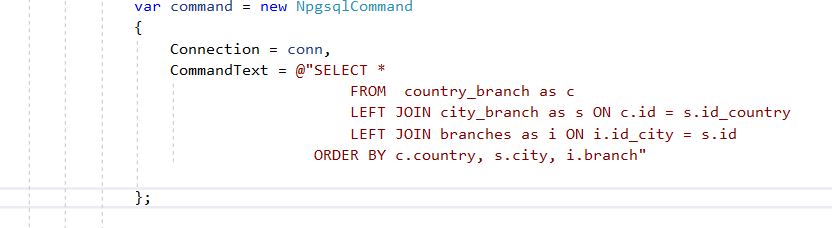
Затем делаем запрос в БД для получения инофрмации о филиале

А именно соединяем все таблицы при помощи LEFT JOIN что бы получить полную информацию о всех эллементов иерархии даже если у них нет потомков.

Сортировка использовалась, для более наглядного представления информации в БД, при её тестировании.

Сортировка не влияет на работу программы так как мы используем метод find для поиска смежный обьектов в бд на верхнем уровне иерархии.

Сортировка могла провить вклад в работу программы если бы мы заменили метод поиска на простое последовательное включение полей.



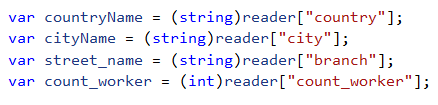
Затем создаём экземпляр класса необходимый для обращения к данным в БД.



Для реализации древовидного представления была использована структура данных treeView.

Заполнение происходит циклически при помощи объекта reader



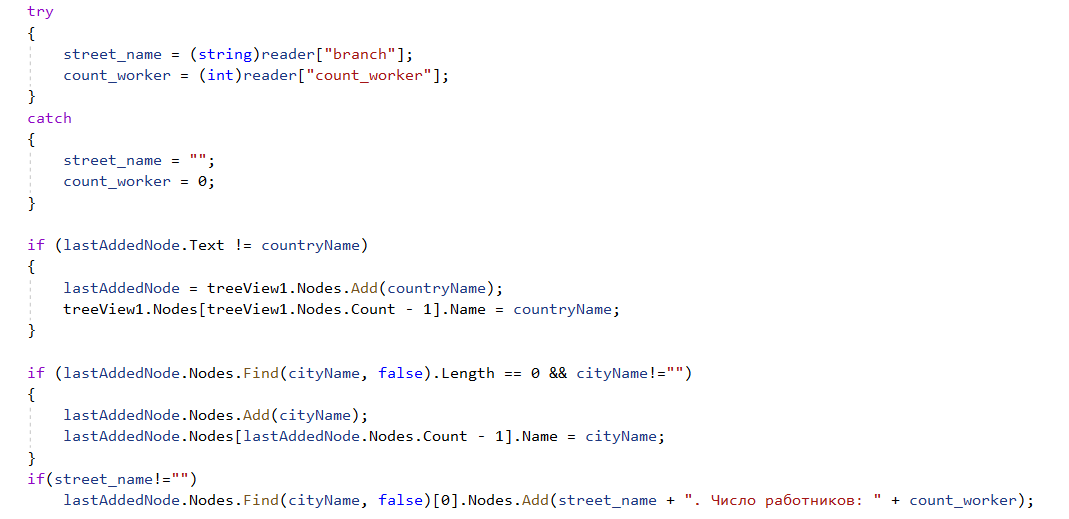


Для добавления в дерево используется объект lastaddedNode, с его помощью я добавляю элементы в коллекцию узлов дерева



Затем я проводил проверку наличия элемента в коллекции на определенном уровне через встроенный в данный класс метод find





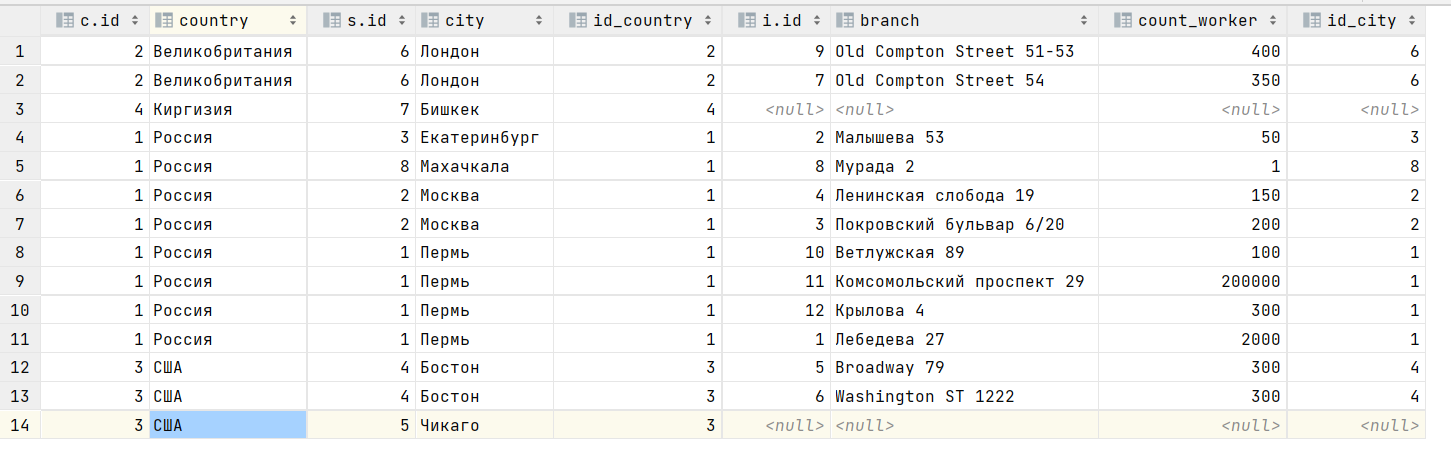
Если объекта с таким ключом не было я его добавляю, а если был, то прохожу вниз по иерархической ступени и выполняю те же действия.  
Для проверки пустых полей или неправильно использованных типов данный цикл был защищен обработчиком исключений try{} catch{}.

# 5 Тестирование программы

Тестирование программы можно разделить на 5 случаев:

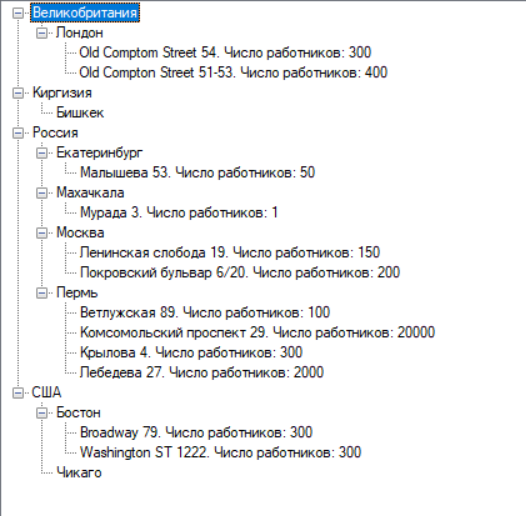
1. Древовидное отображение БД, в которой есть страны, города, улицы с одинаковыми названиями.
2. Древовидное отображение БД, в которой узлы верхнего уровня иерархии не имеют дочерние вершины.
3. Древовидное отображение БД, в которой часть узлов высшего уровня иерархии имеют дочерние вершины.
4. Древовидное отображение БД, в которой часть узлов высшего и среднего уровня иерархии имеют дочерние вершины.
5. Древовидное отображение БД, в которой все узлы имеют дочерние вершины.

Тест 1(тип 4)

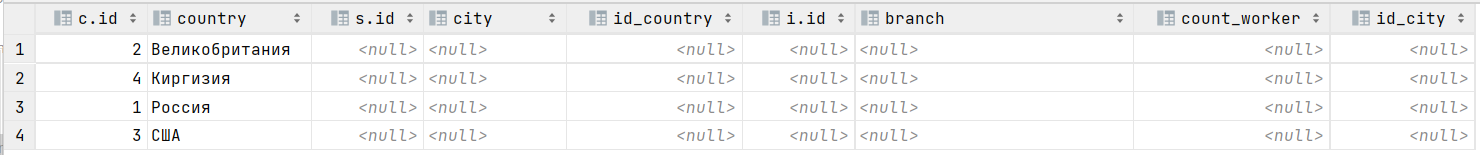
Входные данные:  


Ожидаемый результат: в дереве 4 узла на верхнем уровне, 8 средних узлов 12 узлов на нижнем уровне

Выходной результат:



Тест 2(тип 2)  
Входные данные:

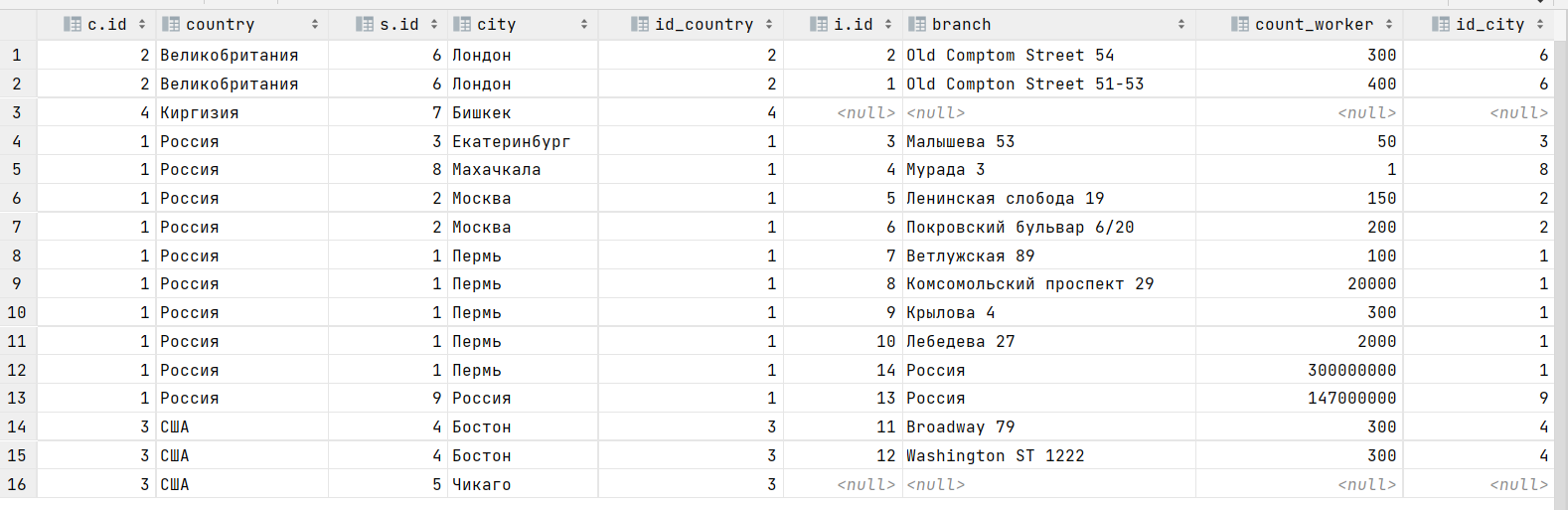


Ожидаемый результат: В дереве 4 узла на верхнем уровне, 0 на среднем, 0 на нижнем.

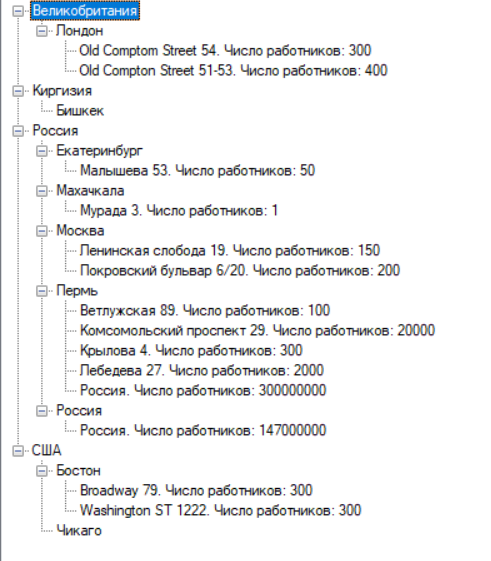
Выходной результат:



Тест 3(тип 1)  
Входные данные:

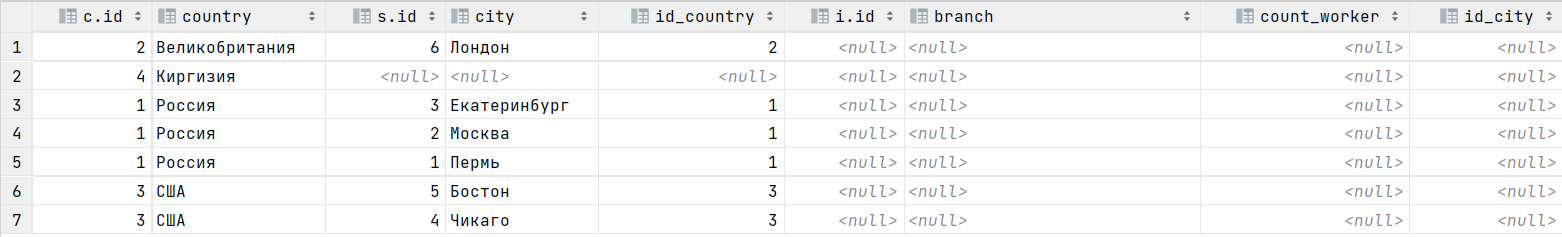


Ожидаемый результат: в дереве 4 узла на верхнем уровне, 9 средних узлов 14 узлов на нижнем уровне.

Выходной результат: 

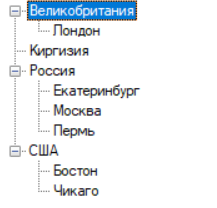
Тест 4(тип 3)

Входные данные:



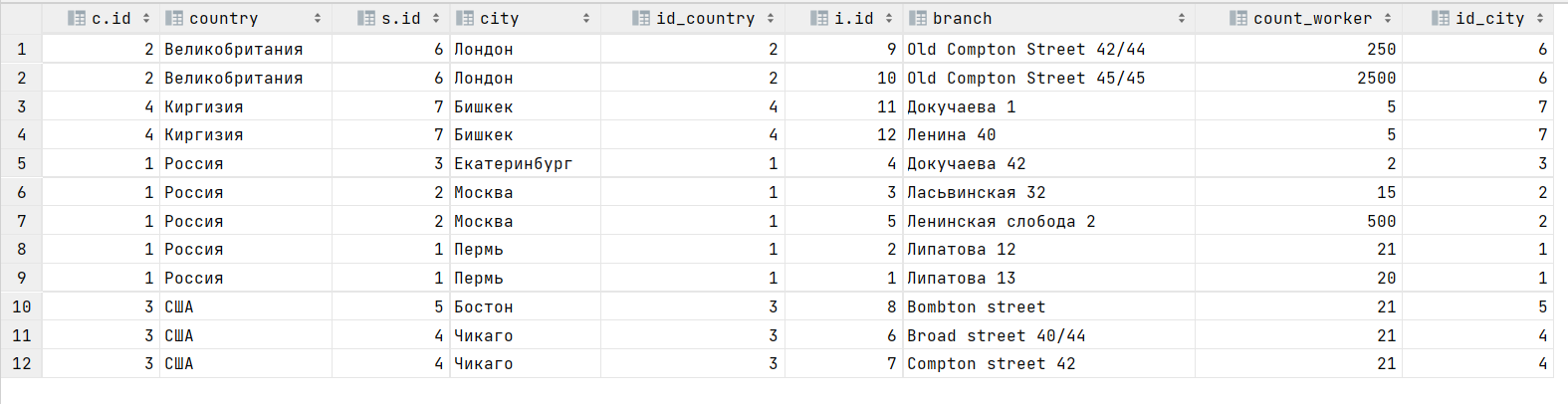
Ожидаемый результат: в дереве 4 узла на верхнем уровне, 6 средних узлов, 0 на нижем уровне.

Выходной результат:



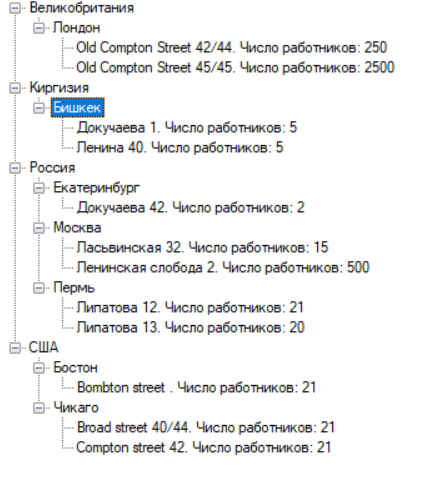
Тест 5(тип 5)

Входные данные:



Ожидаемый результат: в дереве 4 узла на верхнем уровне, 7 средних узлов 12 узлов на нижнем уровне.

Выходной результат:



Вывод: Программа успешно прошла тестирование, работает корректно.