|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Министерство науки и Высшего образования РФ | | | | | | | | |  |
|  |  | | | | | | | | |  |
|  | | Пермский государственный национальный исследовательский университет | | | | | |  | | |
|  | | | |  | | |  | | | |
|  | | ОТЧЕТ  задание № 1 "Древовидное представление" по дисциплине "Учебная практика по базам данных" | | | | | |  | | |
|  | | | |  | | |  | | | |
|  | | | Работу выполнил  студент гр. ПМИ-1,2  Ковыляев Д.А.\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_2020 | |  | Проверил  старший преподаватель кафедры МОВС  Постаногов И.С. \_\_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_2020 | | |  | |
|  | | | |  | | |  | | | |
|  | | | | Пермь 2020 | | |  | | | |

Содержание

[1 Постановка задачи. Задание № 1 «Древовидное представление» 3](#_Toc31371417)

[2 Определение идеи алгоритма, выбор методов решения и структур данных 4](#_Toc31371418)

# Постановка задачи. Задание № 1 «Древовидное представление»

Написать программу, которая на графическом интерфейсе пользователя строит древовидный список по данным из спроектированной ранее базы данных в формате SQLite, состоящей из трёх таблиц, последовательно связанных связью типа 1:М, в котором элементы каждой из трёх уровней иерархии отражают иерархию записей, хранящихся в базе данных. При раскрытии вершины на самом верхнем уровне иерархии к имени вершины приписывается строка " – A/B", где А – число непосредственных дочерних вершин, у которых есть дочерние вершины, а В – число непосредственных дочерних вершин без своих дочерних.

Провести тестирование корректности работы программы. Удостовериться, что на каждом уровне иерархии не объединяются различные сущности с одинаковым текстовым представлением (названием).

Написать отчёт о проделанной работе.

**P.S.** Не забудьте также отразить записи с верхних уровней иерархии, для которых нет соответствующих записей на нижних уровнях иерархии.

**P.P.S.** Использование метода *ToString*() для приведения типов, а также конструкции *try/catch* запрещено.

# Определение идеи алгоритма, выбор методов решения и структур данных

Для решения задачи была выбрана система программирования Microsoft Visual Studio Community 2019 C#. В качестве СУБД для базы данных SQLite был выбран DataGrip.

## Была составлена БД в формате SQLite, содержащая информацию о серверах. В БД есть 3 таблицы: сервера, игроки и предметы. Таблицы связаны связью типа 1:М соответственно. Диаграмму БД можно увидеть на рисунке 1:

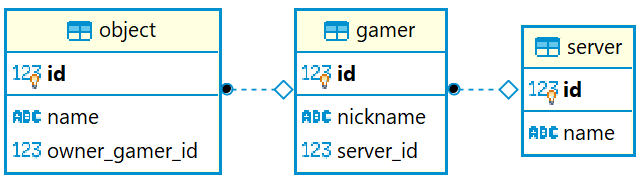


Рисунок 1 – Визуализация Server\_tree в DBeaver

У каждого сервера есть имя и идентификатор. Соответственно таблица server содержит столбцы name и id. name не может быть null.

У игрока есть идентификатор, никнейм и идентификатор сервера, на котором находится игрок. В таблице gamer это столбцы id, nickname и server\_id соответственно. nickname и server\_id не могут быть null.

У предмета есть идентификатор, название и идентификатор владельца. В таблице object это столбцы id, name и owner\_gamer соответственно. name и owner\_gamer не могут быть null.

Идентификатор id во всех таблицах является первичным ключом.

## Построение дерева осуществляется при помощи элемента TreeView.

## Для выполнения части задания (при раскрытии вершины на самом верхнем уровне иерархии к имени вершины приписывается строка " – A/B", где А – число непосредственных дочерних вершин, у которых есть дочерние вершины, а В – число непосредственных дочерних вершин без своих дочерних) было решено создать Dictionary с ключами-ссылками на вершины иерархии – сервера и значениями специально написанного класса ServerInfo.

ServerInfo, как говорит название класса, хранит некоторую информацию о сервере: имя сервера, количество потомков (игроков) и количество потомков, у которых имеются потомки (предметы).

Класс ServerInfo имеет два метода. Первый увеличивает счётчик количества потомков на 1, второй увеличивает счётчик количество потомков, у которых имеются потомки.

## Получение информации из БД происходит с помощью SQL запроса. Запрашиваемыми элементами являются:

Имена серверов и их идентификаторы;

Никнеймы игроков и их идентификаторы;

Имена объектов.

Запрашиваемая информация сортируется таким образом, чтобы было возможно построить дерево в TreeView за один проход.

Использование LEFT JOIN при соединении таблиц в запросе позволяет выводить сервера, на которых отсутствуют игроки, а также игроков, у которых отсутствуют предметы.

## Построение дерева осуществляется так, что на каждом этапе добавления очередной строчки из таблицы, полученной из БД, имеется позиция вставки элемента в TreeView, меняющаяся по следующим правилам:

* если появился уникальный сервер, то добавляем его в Dictionary и делаем текущим местом вставки вершин-игроков (делаем текущим сервером). Добавляем соответственно корневую вершину в TreeViev;
* если появился уникальный игрок, то добавляем новую вершину к текущему серверу и делаем игрока текущим;
* пришедший из таблицы предмет добавляем в дерево к вершине текущего игрока.

При добавлении вершин в дерево так же происходит увеличение счётчиков класса ServerInfo для соответствующих серверов.

## Обработка открытия вершины дерева сделана следующим образом: если открытая вершина является сервером, то меняем текст приписывая информацию из ServerInfo этого сервера.

Закрытие сервера-вершины меняет текст вершины на название сервера.

## Работа с пользователем ведётся через окно Server\_tree, имеющее в своей основе элемент TreeView.

# Тестирование

Проведя анализ возможных входных данных согласно критериям тестирования программы по типу «черного ящика», было составлено 8 тестов, для каждого из них ниже дано описание и результат прохождения данного теста программой.

1. БД без записей.

Ожидаемый результат: отсутствие дерева на экране.

Результат работы программы представлен на рисунке 2. Программы выдала корректный результат.

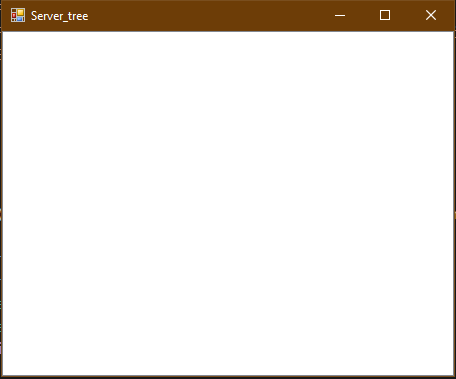


Рисунок 2 – Результат теста

1. В БД присутствуют 2 записи серверов. Второй сервер пустой. На первом сервере находятся 3 игрока без предметов. У двух игроков одинаковый никнейм. Число А на первом сервере равно 0, число B равно 3.

Ожидаемый результат: сервера выводятся в отсортированном порядке. Второй сервер выводится без игроков. Первый сервер с игроками. Два игрока с одинаковыми никами не являются одной вершиной. Игроки отсортированы.

Результат работы программы представлен на рисунке 3:

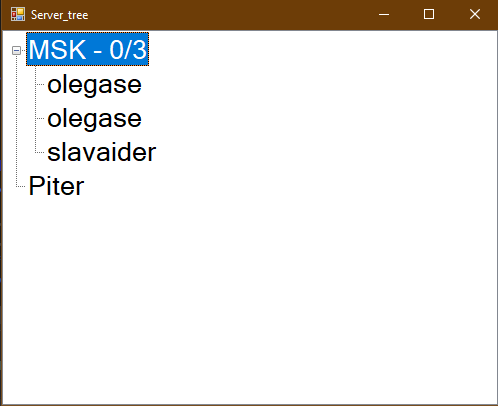


Рисунок 3 – Результат теста

1. Игроки из предыдущего теста получают предметы.

Ожидаемы результат: предметы принадлежат их владельцам в соответствии с БД. Число А на первом сервере равно 2, число B равно 1.

Результат работы программы представлен на рисунке 4. Программы выдала корректный результат.

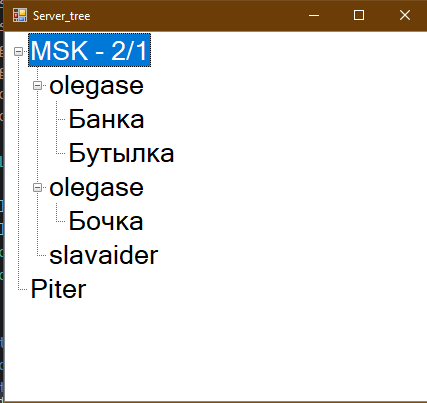


Рисунок 4 – Результат теста

1. Переименуем в БД «Бутылка» в «Банка».

Ожидаемый результат: в TreeView «Бутылка» сменится на «Банка».

Результат работы программы представлен на рисунке 5:

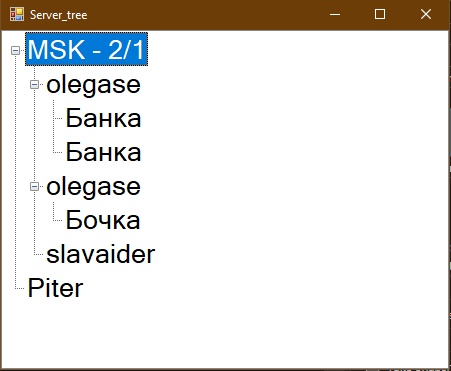


Рисунок 5 – Результат теста

1. Добавим «olegase», находящемуся в дереве ниже, предмет «Банка».

Ожидаемый результат: у первого игрока два предмета «Банка», у второго один.

Результат работы программы представлен на рисунке 6:

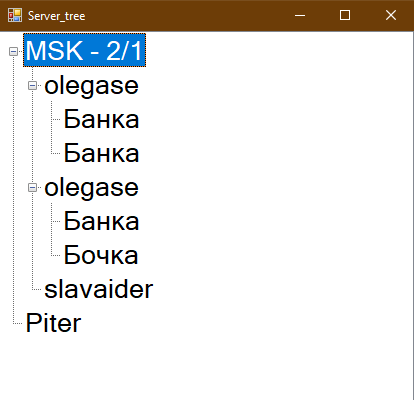


Рисунок 6 – Результат теста

1. Работа на различных данных.

SQL-запрос для составления БД прилагается в файле «Server\_tree\_SQL.txt».

Результат работы программы представлен на рисунке 7. Программы выдала корректный результат.



Рисунок 7 – Результат теста

1. Открытие вершины.

SQL-запрос для составления БД прилагается в файле «Server\_tree\_SQL.txt».

Ожидаемый результат: «MSK» сменится на «MSK – 3/0».

Работа программы представлена на рисунке 8

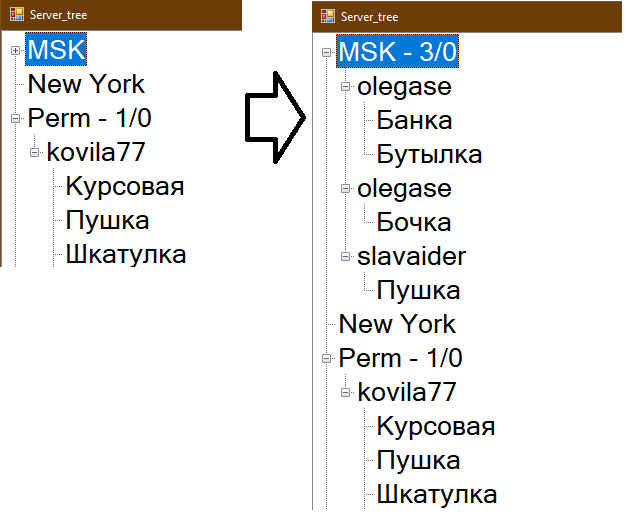


Рисунок 8 – Открытие вершины

1. Закрытие вершины.

SQL-запрос для составления БД прилагается в файле «Server\_tree\_SQL.txt».

Ожидаемый результат: «MSK – 3/0» сменится на «MSK».

Работа программы представлена на рисунке 9:

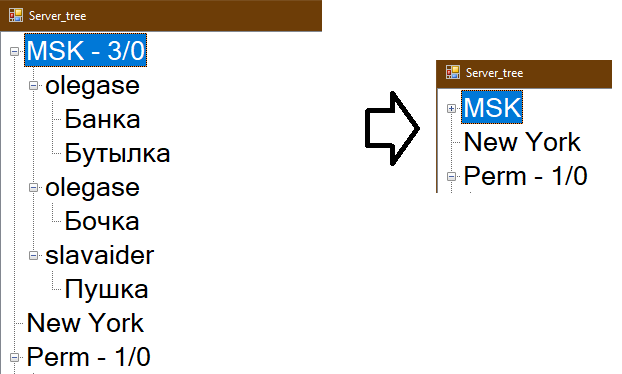


Рисунок 9 – Закрытие вершины