|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ | | | | | | | | |
|  | | |  | | |  | | |
| ФГБОУ ВО «Пермский государственный  национальный исследовательский университет» | | | | | | | | |
|  | | |  | | |  | | |
|  | | ОТЧЕТ  по лабораторной работе № 1 «Построение древовидного представления» по дисциплине «Учебная практика по БД и СУБД» | | | | |  | |
|  | | |  | | |  | | |
|  | Работу выполнил  студент гр. \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Зимин И. В.\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2018 | | |  | Проверил  \_\_\_\_\_\_\_\_\_\_\_ кафедры МОВС  (доц., ст. преп., асс.)  Постаногов И. С.\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_ 2018 | | |  |
|  |  | | |  |  | | |  |
| Пермь 2018 | | | | | | | | |

1 Постановка задачи

Задание было сформулировано следующим образом:

Написать программу, которая на графическом интерфейсе пользователя строит древовидный список по данным из спроектированной ранее базы данных, 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. 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.

Важным моментом в данной формулировке является то, что для верхних уровней иерархии может не быть записей в нижних уровнях.

2 Проектирование базы данных

Для выполнения данного задания я воспользовался ранее спроектированной в среде MS SQL Server Management Studio 2014 базой данных «Автомобили», в которой содержится 3 таблицы, последовательно связанных связью типа «Один ко многим»

В данной предметной области выделены следующие сущности: «Страна-производитель», «Марка автомобиля» и «Модель автомобиля».

Страны-производители имеют название и год основания. Марки автомобилей имеют название и год основания. Модели автомобилей имеют название и год выпуска.

База данных была спроектирована таким образом, что каждой стране, марке и модели было предано уникальное значение, так называемый идентификатор. Например, не может Россия и Китай иметь одинаковый идентификатор, но уникальность идентификатора не гарантирует, что названия стран и других объектов будут различными. Таким же образом обстоят дела и с марками, и с моделями автомобилей.

Помимо вышеперечисленных характеристик нужно было завести дополнительные для того, чтобы организовать связь между данными сущностями.

Одна страна может производить автомобили нескольких марок, но с другой стороны одна марка не может принадлежать сразу нескольким странам. Поэтому для марки автомобиля нужна дополнительная характеристика, показывающая принадлежность к стране-производителю.

Одна марка автомобиля может выпускать несколько моделей, но конкретная модель не может выпускаться сразу несколькими марками. Поэтому для модели автомобиля требуется дополнительная характеристика, показывающая принадлежность к марке автомобиля.

Так же стоит отметить, что года основания/выпуска могут принимать значение NULL, так как могут быть неизвестны.

Диаграмма спроектированной таким образом базы данных приведена на рисунке 2.1.

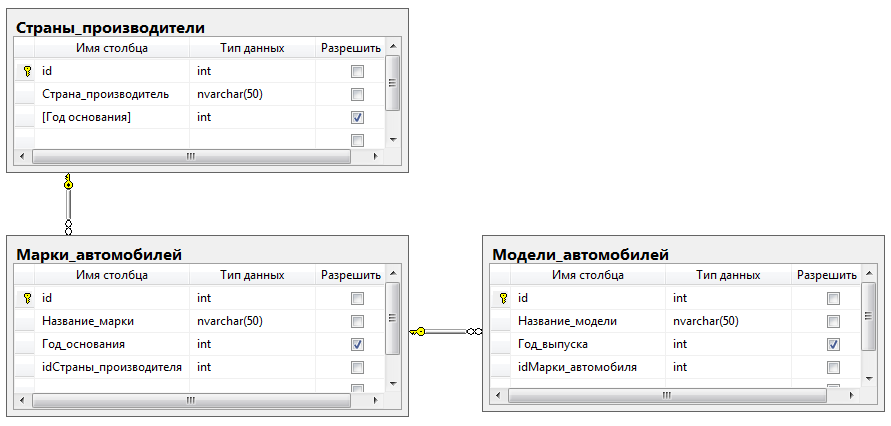


Рисунок 2.1 – Диаграмма базы данных «Автомобили»

3 Основное описание

Для решения данного задания необходимо было осуществить связь определённого языка программирования с данными из ранее спроектированной базы данных. Какой язык выбрать? Зависит от предпочтений. Лично мой выбор пал на один из современных языков программирования C#. Разработка приложения проводилась в среде Visual Studio 2015.

Приложение будет выводить данные в древовидном виде. При запуске приложения будет выведен список стран-производителей. Страны, в которых производится хотя бы одна марка, будет возможность «развернуть» и отобразить список этих марок, принадлежащих к этой стране. При данном действии, сразу за названием появятся некоторые дополнительные характеристики, это будет выглядеть следующим образом: «Название» - «кол-во элементов внутри пункта, которые имеют дочерние вершины» / «кол-во элементов, внутри пункта, которые не имеют дочерних вершин». По необходимости можно «свернуть» список марок. Марки автомобилей, которые выпускают хотя бы одну модель будет так же возможность «развернуть» и по необходимости «свернуть».

На форме расположен только элемент управления, который отображает данные из базы данных в древовидной структуре. Примерный вариант графического интерфейса правильно разработанного приложения изображен на рисунке 3.1.

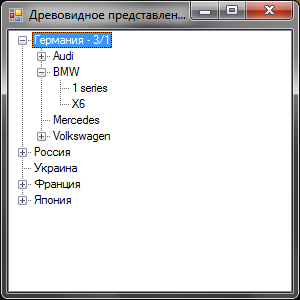


Рисунок 3.1 – Графический интерфейс приложения «Древовидное представление».

4 Описание алгоритма

1. Имеется таблица с данными, которые получены из описанной в разделе 3 базы данных. В данной таблице имеются следующие поля: «название страны» «название марки автомобиля» «название модели автомобиля» «идентификатор страны» «идентификатор марки автомобиля».

При этом таблица последовательно отсортирована по названию страны, идентификатору страны, названию марки автомобиля, идентификатору марки автомобиля.

1. Последовательно передвигаемся по строкам данной таблицы, пока не дойдём до последней. Получив название страны-производителя, мы добавляем его в список стран на нулевой уровень дерева. Если есть модель, производимая данной страной, то добавляем ее на первый уровень и, если есть модель, выпускаемая данной маркой, добавляем ее на второй уровень. При этом нужно избежать повторения стран с одинаковым идентификатором на 0-м уровне и повторений марок с одинаковым идентификатором на 1-м уровне.
2. Подготовить методы для отлавливания нажатия кнопки «сворачивания/разворачивания» узла дерева. Если узел на 0-м уровне был развернут, то приписываем к названию нужные числовые характеристики, которые вычисляются путём подсчёта дочерних вершин, содержащих дочерние вершины и общего числа дочерних вершин. Если узел свёрнут, то удаляем числовые характеристики.

5 Описание SQL – запроса

Для того чтобы простроить древовидное представление, необходимо было произвести выборку данных из БД, описанной в разделе [2].

Чтобы это сделать я написал SQL-запрос, представленный на листинге 5.1.

SELECT Страна\_производитель AS 'country', Название\_марки AS 'nameMark', Название\_модели AS 'nameModel', Страны\_производители.id AS 'idCountry', Марки\_автомобилей.id AS 'idMark'

FROM Страны\_производители LEFT OUTER JOIN Марки\_автомобилей ON Страны\_производители.id = idСтраны\_производителя LEFT OUTER JOIN Модели\_автомобилей ON Марки\_автомобилей.id = idМарки\_автомобиля

ORDER BY idCountry, idMark

В блоке SELECT перечислены столбцы, значения которых нам необходимо получить, для более удобного использования их в дальнейшем я придал им короткие, уникальные имена при помощи ключевого слова AS. К таковым столбцам относятся:

1. Название страны производителя
2. Название марки автомобиля
3. Название модели автомобиля
4. Идентификатор страны производителя
5. Идентификатор марки автомобиля

В блоке FROM дважды используются явные операции соединения всех трёх таблиц для этого было использованы ключевые слова LEFT JOIN. Воспользовавшись им в первый раз, мы получили таблицу из 7 столбцов (все столбцы из таблицы стран-производителей и все столбцы из таблицы марок автомобилей) причем в результат пойдут все строки из первой таблицы, благодаря ключевому слову LEFT и в строках принадлежащих второй таблице, могут находиться пустые строки, заполненные значением NULL, это будет означать что идентификатор страны не совпадает с идентификатором принадлежности к стране. Аналогичным образом подсоединяем к полученной таблице ещё одну таблицу «модели автомобилей». В итоге получаем одну большую таблицу из 10 столбцов и определённого количества строк, удовлетворяющим условиям запроса.

Из этих 10 столбцов выбираем нужные 5 и с помощью ORDER BY сортируем по идентификатору страны-производителя, а затем по идентификатору марки автомобиля. Это необходимо чтобы сущности не перемешивались.

6 Реализация приложения. Общий порядок действий.

Далеко не всем нравится читать пояснение той или иной задачи, опираясь

на какой-либо конкретный язык программирования. В этом разделе я приведу основной порядок действий, которого стоит придерживаться.

1. Выполнить подключение к БД.

В данном пункте нужно каким-либо способом подключиться к БД, например, создав строку подключения и с помощью нее уже осуществить, непосредственно, само подключение. После его создания необходимо открыть это подключение и реализовать корректное закрытие.

1. Создать команду для выборки данных из БД.

Команда обычно содержит подключение к БД и сам текст SQL-запроса.

1. Считывание данных из БД

Необходимо объявить некий reader, то есть переменную в которую поочерёдно в цикле будут записываться строчки результатов запроса, из которых можно выделить нужный столбец.

1. Добавление в дерево

Данные получаемые при помощи reader необходимо поместить в дерево, надлежащим образом.

1. Числовые характеристики

Необходимо организовать методы, которые могли бы отловить «свёртывание/развёртывание» узла и при помощи свойств древовидного представления посчитать характеристики, требуемые по заданию.

7 Реализация приложения на языке C#

Для подключения к БД необходимо создать строку подключения, которая формируется посредством создания объекта типа SqlConnectionStringBuilder. После чего необходимо создать объект класса SqlConnection, которому передать в конструктор строку подключения, тем самым мы создадим подключение к БД. Далее требуется выполнить описанный в разделе 5 SQL-запрос, для этого нужно создать и проинициализировать объект класс SqlCommand и выполнить его с помощью метода ExecuteReader(). Последовательный просмотр записей БД реализуется с помощью SqlDataReader. Полученные данные наполняют элемент TreeView.

1. Представление данных:

Для древовидного представления в приложении Windows Form использован элемент TreeView.

1. Создаём строку подключения:

SqlConnectionStringBuilder – класс для создания строки подключения.

Инициализируем поля класса:

1. DataSource – задает сетевой адрес экземпляра SQL Server, с которым устанавливается соединение;
2. IntegratedSecurity – определяет способ проверки подлинности. При true - используются учетные данные текущей учетной записи Windows;
3. InitialCatalog – имя БД для подключения.

Свойство ConnectionString вернет строку подключения, которую сохраняем.

Создеание строки подключение приведено на листинге 7.1.

string sConstr = new SqlConnectionStringBuilder()

{

DataSource = ".",

IntegratedSecurity = true,

InitialCatalog = "Laba3"

}.ConnectionString;

Листинг 7.1 - Создание строки подключения.

1. Создаем подключение:

SqlConnection – класс, предоставляющий открытое подключение к БД SQL Server. Создаем объект типа SqlConnection и инициализируем, передав в конструктор класса строку подключения. Данное действие лучше выполнять при помощи конструкции using(), тем самым обеспечив корректное закрытие подключения. Как это выполняется показано на листинге 7.2. В теле конструкции мы открываем подключение, после чего помещаем все элементы кода, которые использует данное подключение.

using (SqlConnection sConn = new SqlConnection(sConstr)){ //Код };

Листинг 7.2. Конструкция Using

1. Создаём команду:

SqlCommand – класс, определяющий запрос к БД.

Инициализируем поля:

1. Connection – задает объект SqlConnection;
2. CommandText – текст запроса к БД.
3. Извлекаем данные из БД:

SqlDataReader – класс, реализующий чтение потока строк последовательного доступа из БД.

Чтобы выполнить запрос к БД, необходимо воспользоваться методом ExecuteReader() из класса SqlCommand. Выполнение данного метода вернет объект типа SqlDataReader, который будет использоваться для извлечения информации из БД.

Создаем цикл, который выполняется пока не дойдем до конца БД. Переход к следующей записи БД выполняется с помощью метода Read() из класса SqlDataReader.

Для получения значения конкретного элемента необходимо обратиться к объекту SqlDataReader по индексу, передав в качестве индекса имя столбца типа String (reader["название столбца"]). Полученное значение имеет тип object, поэтому необходимо выполнять явное приведение к нужному типу для сохранения значения в переменной.

Если объект в БД отсутствует, то при чтении он имеет тип DBNull, поэтому для проверки того, что получаемое значение отлично от NULL используется конструкция вида:

if (!(reader["название столбца"] is DBNull)) {}.

Для того чтобы избежать повторов одинаковых полей в структуре дерева требуется запоминать идентификаторы из прошлой строки, для этого используются переменные lastCountryId и lastIdMark, которым изначально присвоено значение -1, что бы при первом сравнении с полученным из БД идентификатором равенство было невозможным, так как идентификатор число натуральное.

Полный цикл считывания данных из БД и добавления в дерево приведен на листинге 7.3.

SqlDataReader reader = sCommand.ExecuteReader();

TreeNode lastCountryNode = new TreeNode();

TreeNode lastNameMark = new TreeNode();

int lastCountryId = -1;

int lastIdMark = -1;

while (reader.Read()) {

string country = (string)reader["country"];

int idCountry = (int)reader["idCountry"];

if (idCountry != lastCountryId) {

lastCountryNode = tvAuto.Nodes.Add(country);

lastCountryId = idCountry;

}

if (reader["nameMark"] is DBNull) continue;

string nameMark = (string)reader["nameMark"];

int idMark = (int)reader["idMark"];

if (idMark != lastIdMark) {

lastNameMark = lastCountryNode.Nodes.Add(nameMark);

lastIdMark = idMark;

}

if (reader["nameModel"] is DBNull) continue;

string nameModel = (string)reader["nameModel"];

lastNameMark.Nodes.Add(nameModel);

}

Листинг 7.3 – считывание данных и добавление в дерево.

1. Вычисление кол-ва вершин:

Используются встроенные события элемента TreeView.

1. BeforeExpand – срабатывает при разворачивании узла дерева.
2. BeforeCollapse – срабатывает при сворачивании узла дерева.

8 Тестирование приложения

Чтобы полностью покрыть тестами все возможные заполнения БД, можно условно разделить все тесты на 3 класса, следующим образом:

1-й класс. Относящиеся к 0-му уровню, то есть к странам-производителям.

1. Страна не выпускает ни одной марки автомобиля
2. Страна выпускает одну марку автомобиля
3. Страна выпускает более одной марки.
4. Несколько стран имеют одинаковое название.

2-й класс. Относящиеся к 1-му уровню, то есть к маркам автомобилей

1. Нет моделей данной марки
2. Одна модель данной марки
3. Более одной модели данной марки
4. Несколько марок авто имеют одинаковые названия
5. Несколько марок авто выпускают модели с одинаковыми именами

3-й класс. Проверка числовых характеристик

1. Стране принадлежат только те марки, которые выпускают хотя бы одну модель
2. Стране принадлежат только те марки, которые не выпускают ни одной модели
3. Стране принадлежат марки как выпускающие модели, так и не выпускающие

Минимально наполним нашу БД, чтобы охватить проверку всех вышеперечисленных тестов.

Данные о странах-производителях предоставлены в таблице 8.1.

Страна с id равным 1 была включена, чтобы проверить тест №3 в 1-м классе.

Страна с id равным 2 была включена, чтобы проверить тест №2 в 1-м классе.

Страна с id равным 3 была включена, чтобы проверить тест №1 в 1-м классе.

Два Китая позволят нам проверить 4-й тест в 1-м классе.

|  |  |  |
| --- | --- | --- |
| Id | Страна\_производитель | Год основания |
| 1 | Россия | 1991 |
| 2 | Китай | 1949 |
| 3 | Китай | 1961 |

Таблица 8.1 – Страны-производители

Данные о марках автомобилей предоставлены в таблице 8.2.

Марка с id равным 1 была включена, чтобы проверить тест №3 из класса 2

Марка с id равным 2 была добавлены, чтобы проверить тест №2 из класса 2

Марка с id равным 3 была включена, чтобы проверить тест №1 из класса 2.

Две марки Lada позволяют проверить тест №4 из раздела 2

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Марка\_автомобиля | Год основания | idCтраны\_производителя |
| 1 | Lada | 1966 | 1 |
| 2 | УАЗ | 1992 | 1 |
| 3 | Lada | 1999 | 2 |

Таблица 8.1 – Марки автомобилей.

Данные о моделях автомобилей предоставлены в таблице 8.3.

Для теста №5 из раздела 2 были созданы 2 модели с именем Vesta

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Модель\_автомобиля | Год выпуска | idМарки\_автомобиля |
| 1 | Vesta | 2013 | 1 |
| 2 | Х-Ray | 2015 | 1 |
| 3 | Vesta | 2014 | 3 |

Таблица 8.1 – Модели автомобилей.

Суммарно все три таблицы дают результаты по всем 3-м классам тестов.

Приложение тестирование прошло успешно!