|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ | | | | | | | | |
|  | | |  | | |  | | |
| ФГБОУ ВО «Пермский государственный национальный исследовательский университет» | | | | | | | | |
|  | | |  | | |  | | |
|  | | ОТЧЕТ | | | | |  | |
|  | | по практической работе № 8 «Нестандартный запрос» по дисциплине «Учебная практика по БД и СУБД»» | | | | |  | |
|  | | |  | | |  | | |
|  |  | | |  | Выполнил:  студент механико-математического факультета  2 курса  Мироненко А.О.  Группа ПМИ-5,6 | | |  |
|  |  | | |  |  | | |  |
| Пермь 2018 | | | | | | | | |

1. Постановка задачи

Нестандартный запрос предназначен для создания определённых ограничений на выбор данных из БД, причём пользователю предоставляется интерфейс для создания такого ограничения. Необходимо реализовать поставленную задачу, должны присутствовать 4 вкладки. На вкладке «Поля» - все поля из всех. Поля можно сгруппировать по смыслу, разделив специальными строчками, как показано на рисунке. Здесь мы выбираем те поля, которые будут заголовками полей в таблице – результате выполнения запроса. Далее, пользователь должен иметь возможность наложить определённые условия на результат запроса: в поле «Имя поля» должны попадать все поля из всех таблиц, а не только те, которые были выбраны на предыдущей вкладке, в поле «Критерий» - операции сравнения, доступные для типа данных выбранного поля, в поле «Выражение» должны подчитываться значения выбранного поля «Имя поля» из соответствующей таблицы БД. Условия могут быть связаны через И, ИЛИ. Далее, на вкладке «порядок» - возможность задать сортировку по выбранным полям. На вкладке «Результат» - результат выполнения запроса

**Дополнительные таблицы:**

1. «Транслит»

Используется на первой вкладке для вывода всех полей всех таблиц БД, а также для учёта типа поля при формировании условия на вкладке «Условия». Таблица содержит два поля с названиями перечисленных полей. Одно используется при формировании части SELECT, а другое, переведённое на русский, для вывода на форму для наглядного описания имён полей.

1. «Таблица-связь»

Данная таблица используется для формирования строки соединения между какими-либо двумя таблицами.

1. Проектирование БД

Для решения поставленной задачи требовалось воспользоваться уже ранее спроектированной БД «Shops». Были взяты 4 таблицы, представленные на рисунке 1

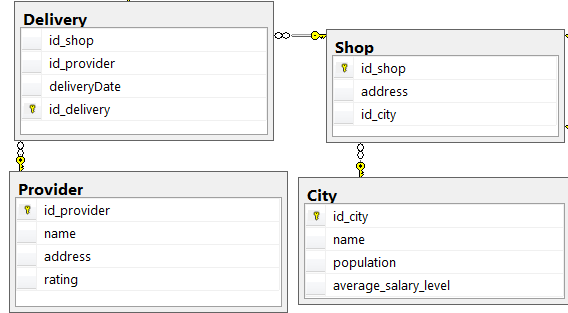
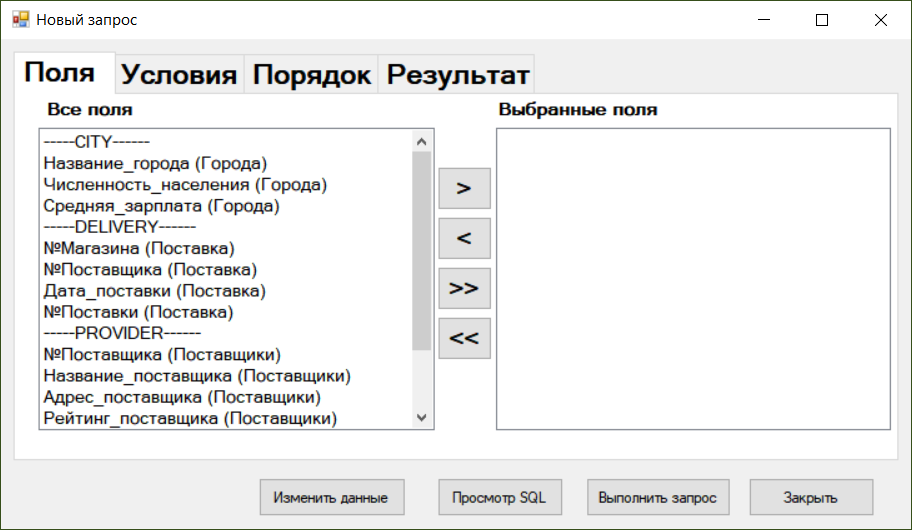


Рисунок 1 – Таблицы для реализации проекта

Приложение выглядит следующим образом:



1. Реализация на C#

Для подключения к БД была создана строка подключения вида:

private readonly string \_sConnStr = new SqlConnectionStringBuilder

{

// Берем параметры из конфигурационного файла

DataSource = Properties.Settings.Default.ProjectDataSource,

InitialCatalog = Properties.Settings.Default.ProjectInitialCatalog,

IntegratedSecurity = true

}.ConnectionString;

Где имя сервера и имя БД были взяты из конфигурационного файла App.config

Все поля из таблицы хранились в словаре типа Column:

private class Column

{

public string NameTable { get; set; }

public string Name { get; set; }

public string Type { get; set; }

public static Dictionary<string, string> FriendlyNames;

public string FullColumnName => $"[{NameTable}].[{Name}]";

}

Для хранения данных из вспомогательной таблицы с метаданными о связях между таблицами БД Хранились в двух однотипных словарях

// Лист соединенных напрямую таблиц

Dictionary<Tuple<string, string>, string> joinsDict = new Dictionary<Tuple<string, string>, string>();

В данном листе хранятся данные следующим образом:

Ключ: (таблица1, таблица2)

Значение: условие соединения таблиц

// Лист с промежуточными соединениями

Dictionary<Tuple<string, string>, string> farJoinsDict = new Dictionary<Tuple<string, string>, string>();

В данном листе хранятся данные следующим образом:

Ключ: (таблица1, таблица2)

Значение: промежуточная соединяющая таблица

Составление запроса:

Создаем команду

var sCommand = new SqlCommand { CommandText = "SELECT DISTINCT " };

Преобразуем все выбранные столбцы к списку типа Column

var items = lbxSelectFields.Items.Cast<Column>().ToList();

Создаем строку для предложения SELECT

var columns = items.Select(t => t.FullColumnName + " as '" + t.DisplayName + "'").Distinct().ToList();

var tables = items.Select(t => t.NameTable).Distinct().ToList();

sCommand.CommandText += string.Join(", ", columns);

Получаем все таблицы в которых содержатся выбранные столбцы:

var tables = items.Select(t => t.NameTable).Distinct().ToList();

Добавляем в список tables таблицы, которые были использованы в условиях.

foreach (ListViewItem row in lvConditions.Items)

{

Column clmn = null;

foreach (var item in fields)

if (item.DisplayName == row.SubItems[0].Text)

clmn = item;

tables.Add(clmn.NameTable);

}

tables = tables.Distinct().ToList();

Получаем список соединений

var joinConditions = GetJoinConditions(tables, out var allTables);

Для этого написана функция GetJoinConditions в которой мы про собираем все прямые соединения таблиц из tables а тык же делаем обход и строим соединения между таблицами между котороыми нет прямой связи при помощи функции

string CreatePath(Tuple<string, string> tuple, string finishTable)

{

var curTuple = Tuple.Create(tuple.Item2, finishTable);

if (!farJoinsDict.ContainsKey(curTuple)) return tuple.Item1 + " " + tuple.Item2 + " " + finishTable;

return tuple.Item1 + " " + CreatePath(Tuple.Create(tuple.Item2, farJoinsDict[curTuple]), finishTable);

}

Которая возвращает полный путь от одной таблицы к другой.

Как например путь между таблицами City и Provider:

City Shop Delivery Provder

Далее мы разбиваем данную строку по пробелам и добавляем таблицы в tables, которых еще там не было.

Таким образом мы получили все нужные нам таблицы.

И можем составить блок FROM

sCommand.CommandText += " FROM " + string.Join(", ", allTables);

Осталось составить блок WHERE:

Для хранения условия используем словарь

var conditionsParam = new List<string>();

Далее пробегаем по списку условий и кладем строки типа

[Имя таблицы].[Имя столбца] критерий параметр связка

conditionsParam.Add($"{column.FullColumnName} {criterion} {paramName} {predicate}");

Далее сортировка:

// Лист для сортировки

var orderList = new List<string>();

// Проходим по полям листбокса сортировки

foreach (var item in lbxSort.Items)

{

var field = (Column)item;

// sortList[field.DisplayName] = либо ASC либо DESC

orderList.Add(field.FullColumnName + sortList[field.DisplayName]);

}

Далее необходимо составить блок WHERE

if (allConditionsParam.Any())

sCommand.CommandText += " WHERE " + string.Join(" ", allConditionsParam);

// Если в листе для сортировки есть что нибудь, тогда добаляем ORDER BY и поля сортировки

if (orderList.Any()) sCommand.CommandText += " ORDER BY " + string.Join(", ", orderList);

После того как текст команды составлен, осуществляем запрос и помещаем данные в таблицу на листе результатов:

var table = new DataTable();

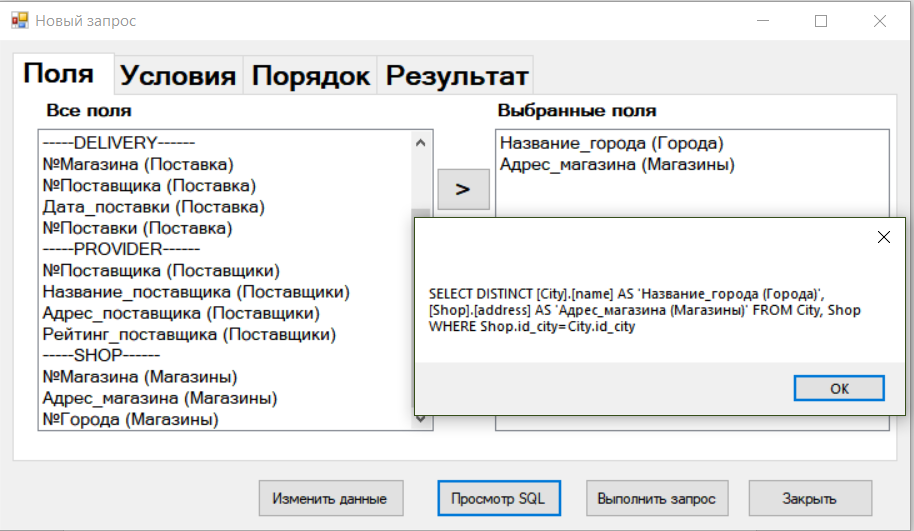
table.Load(sCommand.ExecuteReader());

dgvResult.DataSource = table;

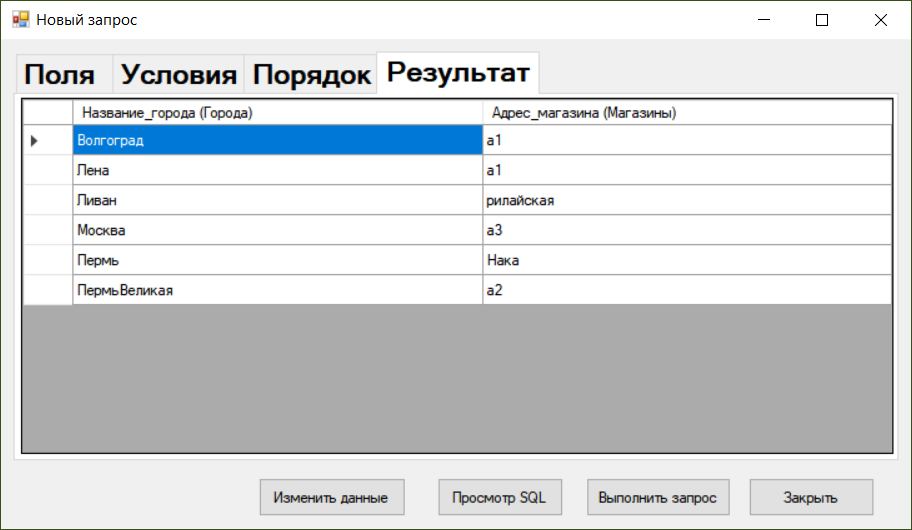
// Переходим на вкладку "Результат"

tbcMain.SelectedTab = tbpResult;

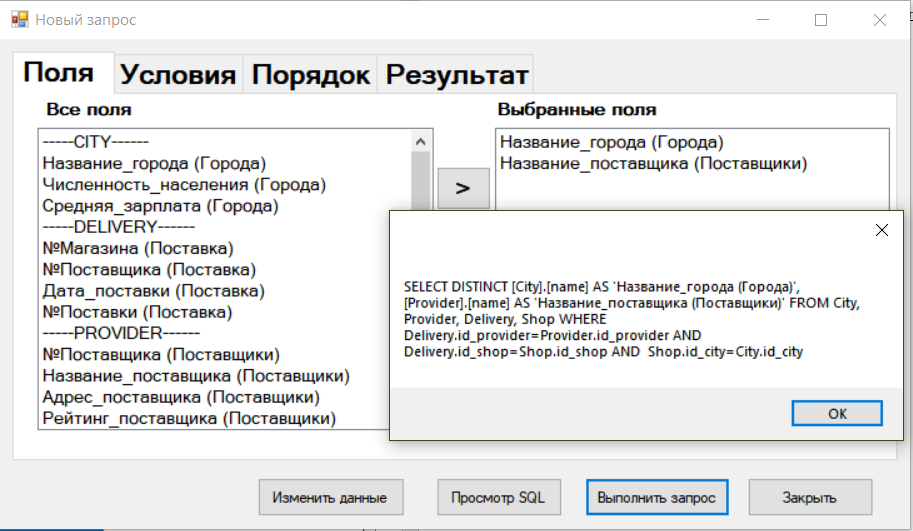
1. Тестирование
2. Проверка построения запроса из напрямую связанных таблиц



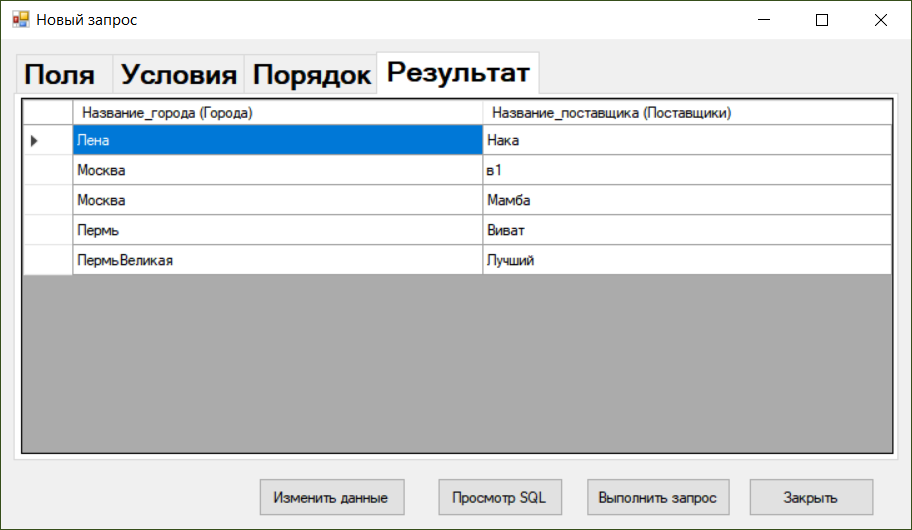
Результат



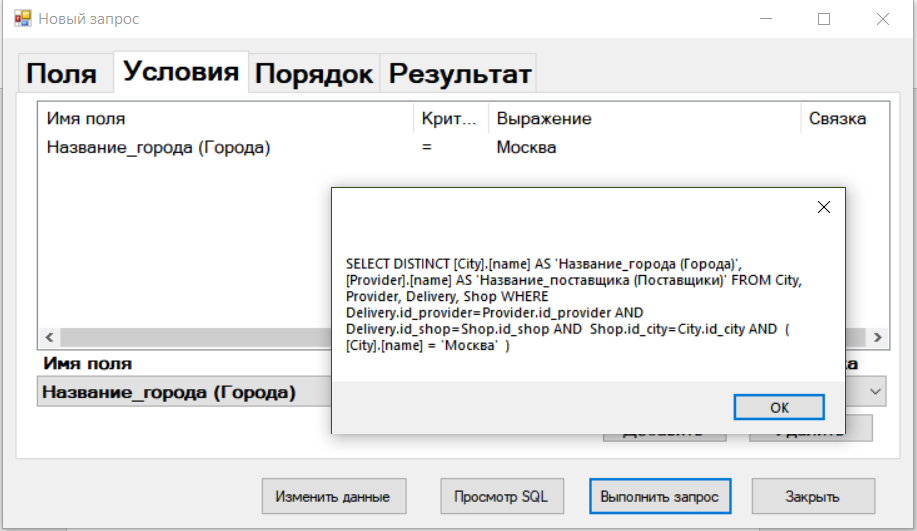
1. Проверка на правильность построения связей между таблицами, которые связаны не на прямую



Результат:



1. Проверка выполнения условия



Результат

