**МиНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. В.Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Базы данных»

предметная область: Сеть кафе

|  |
| --- |
| Выполнил:  ст. гр. ПВ-31  Пересыпкина И.Г.  Проверил: старший  преподаватель Гарибов А.И. |

Белгород 2018

**Содержание**

[1. Цель выполнения курсового проекта 3](#_Toc535051074)

[2. Описание предметной области 3](#_Toc535051075)

[3. Инфологическая модель 3](#_Toc535051076)

[4. Описание структуры базы данных 5](#_Toc535051077)

[5. Структура базы данных 6](#_Toc535051078)

[6. Подключение к базе данных, инструменты 7](#_Toc535051079)

[7. Описание классов и объектов для работы с базой данных 7](#_Toc535051080)

[8. Добавление, удаление, получение и обновление записей в БД 8](#_Toc535051081)

[9. Импорт и экспорт данных. XML-файлы и XSD-схема 9](#_Toc535051082)

[10. Триггеры, представления и хранимые процедуры 10](#_Toc535051083)

[Список литературы 14](#_Toc535051084)

[Приложение A — SQL-запросы формирования БД 15](#_Toc535051085)

[Приложение Б — Добавление, удаление, получение, обновление 17](#_Toc535051086)

[Приложение В — Импорт и экспорт данных 19](#_Toc535051087)

[Приложение Г — Примеры работы программы 22](#_Toc535051088)

# Цель выполнения курсового проекта

Целями данного курсового проекта являются: закрепление знаний и навыков, полученных в рамках курса «Базы данных» и получение практического опыта проектирования и реализации приложений по технологии «клиент—сервер».

Основная задача в контексте темы «Сеть кафе» заключается в создании базы данных, инициализации всех параметров таблиц и необходимых методов для работы с этими данными посредствам ORM в связки с пользовательским приложением.

# Описание предметной области

В данном курсовом проекте в качестве предметной была выбрана область «сеть кафе». В создаваемой базе данных кафе «Bulvar» должна храниться информация, отражающая основные направления деятельности кафе. Кафе представляет собой предприятие общественного питания с широким ассортиментом блюд. Сотрудник - это человек без которого невозможна работа кафе, каждый сотрудник выполняет поставленную задачу. Меню кафе, содержит сведения о предлагаемых блюдах. Каждое блюдо имеет свое название, для него указан рецепт, который содержит перечисление ингредиентов и название каждого продукта, входящего в блюдо. Для каждого поставщика продуктов указаны наименование фирмы-поставщика, название поставляемого продукта, частота и объем поставок в месяц.

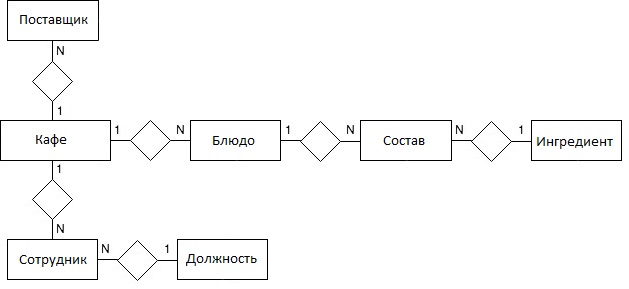
# Инфологическая модель

После изучения предметной области были выделены следующие сущности:

* кафе;
* поставщик;
* блюдо;
* состав;
* ингредиент;
* сотрудник;
* должность.

Учитывая всё вышеперечисленное, построим инфологическую модель для нашей предметной области (рисунок 1).

Рисунок 1 — Инфологическая модель без учета атрибутов



Однако, у каждой определённой нами сущности имеется набор атрибутов, то есть существенных свойств объекта, представляющих интерес для пользователя. Как правило, атрибуты позволяют отличать конкретные экземпляры объектов друг от друга.

Можно выделить атрибуты для каждой из описанных нами сущностей:

* Кафе: владелец, название, адрес, телефон, рейтинг;
* Поставщик: название, продукт, частота поставок, объем поставок;
* Блюдо: название, цена;
* Ингредиент: продукт;
* Состав: вес;
* Сотрудик: ФИО, паспорт, образование, опыт работы;
* Должность: название, зарплата;

# Описание структуры базы данных

Реализация базы данных проводится путем создания физической базы на сервере (MS SQL Server 2017).

Структура данных «Кафе»:

* owner: владелец кафе ФИО (nvarchar), пустые значения не допускаются;
* title: название кафе (nvarchar), пустые значения не допускаются;
* address: адрес кафе (nvarchar), пустые значения не допускаются;
* phone: номер телефона (int), пустые значения допускаются;
* rating: рейтинг кафе (int), пустые значения допускаются;
* supplierID: идентификатор поставщика, принадлежащего кафе (int);
* menuID: идентификатор меню (int);
* serveID: идентификатор сотрудника, принадлежащего кафе (int).

Структура данных «Поставщик»:

* title: название фирмы поставщика (nvarchar), пустые значения не допускаются;
* product: продукт поставляемый поставщиком (nvarchar), пустые значения не допускаются;
* frequency of deliveries: частота поставок (int), пустые значения допускаются;
* volume of deliveries: объем поставок (int), пустые значения допускаются.

Структура данных «Блюдо»:

* title: название блюда (nvarchar), пустые значения не допускаются;
* price: цена блюда (int), пустые значения не допускаются;
* cafeID: идентификатор кафе, принадлежащего блюда (int).

Структура данных «Состав»:

* numingredient: вес ингредиента (int), пустые значения не допускаются;
* ingredientID: идентификатор ингредиента, принадлежащего состава (int);
* dishID: идентификатор блюда, принадлежащего состава (int).

Структура данных «Ингредиент»:

* ingredient: название продукта (nvarchar), пустые значения не допускаются;
* compositionID: идентификатор состава, принадлежащего ингредиента (int).

Структура данных «Сотрудник»:

* name: ФИО работника кафе (nvarchar), пустые значения не допускаются;
* passport: серия, номер паспорта (int), пустые значения не допускаются;
* education: образование сотрудника (nvarchar), пустые значения допускаются;
* experience: опыт работы (int), пустые значения не допускаются;
* postID: идентификатор должности, принадлежащего сотрудника (int).

Структура данных «Должность»:

* title: название должности (nvarchar), пустые значения не допускаются;
* salary: зарплата работника (int), пустые значения не допускаются.

# Структура базы данных

По описанному выше легко определяются 7 таблиц, с которыми нам предстоит иметь дело при работе с базой данных. Можно выделить следующие атрибуты:

* кафе (cafe);
* поставщик (supplier);
* блюдо (dish);
* состав (composition);
* ингредиент (ingredient);
* сотрудник (serve);
* должность (post).

Изобразим в виде схемы структуру базы данных (рисунок 2).

Скрипт создания БД описан в приложении А.

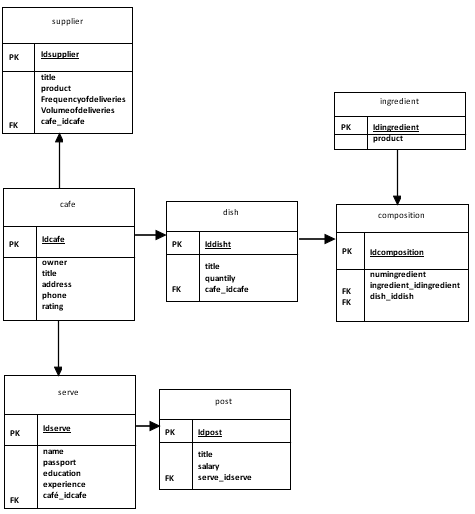


Рисунок 2 — табличное представление базы данных

# Подключение к базе данных, инструменты

Взаимодействие базы данных и приложения организовано на основе native поддержки MS SQL Server средствами языка C#.

В программе используется ORM (Object Relation Mapping), при котором на основе существующей базы данных разрабатываются классы, сопоставляющиеся с соответствующими таблицами базы. Эти классы можно изменять по своему усмотрению, добавляя в них необходимые CRUD-методы. CRUD — четыре базовые функции, используемые при работе с хранилищами данных: create, read, update, delete. Проведение аналогии с языком SQL позволяет заметить, что это методы: insert, select, update и delete.

Для подключения к базе данных используется класс ConnectionString, который представлен в листинге 1.

Листинг 1 — класс ConnectionString

class ConnectionString

{

public bool Open(SqlConnection conn)

{

try

{

conn.ConnectionString = "Data Source=DESKTOP-5E5L14V;Initial"

+" Catalog=cafe;Integrated Security=True";

conn.Open();

return true;

}

catch (System.Data.SqlClient.SqlException)

{

return false;

}

}

}

Для открытия и закрытия подключения использовались методы open(), close() классa SqlConnection.

# Описание классов и объектов для работы с базой данных

Для взаимодействия с базой данных «Сеть кафе» используются классы, именованные так же, как и таблицы, которые содержит данная база. Для работы с базой данных используются семь классов (Cafe, Supplier, Dish, Composition, Ingredient, Serve, Post). Поля классов соответствуют столбцам одноимённых таблиц. Каждое поле класса снабжено свойствами, аналогичными полям таблицы базы данных, и методами get и set для последующего удобства в корректировании и совершенствовании кода. Кроме этого, классы содержат методы модификации – удаление, изменение, добавление нового элемента. Операция добавления может вызываться как для экземпляра класса соответствующей таблицы, так и для класса в целом. Операции удаления и обновления для отдельного экземпляра вызываться не могут, поскольку нельзя гарантировать, что указанные в нём данные действительно являются одной из записей, хранимых в базе данных. С помощью операции получения данных можно запросить как весь список, так и единственную запись, выбираемую по полю id (только в тех классах, где соответствующее поле в таблице уникально). Вышеописанные механизмы реализуются с помощью классов SqlConnection, SqlDataReader, SqlCommand, отправляющих Insert/Delete/Update/Select-запросы для указанной записи или записей.

В листинге 2 представлено описание класса – Cafe.

Класс Café содержит свойства idcafe, owner, title, address, phone и rating, аналогичные полям таблицы базы данных Café. Кроме того, класс содержит методы получения всего списка кафе, поиска кафе по id, address, а также методы модификации такие как удаление, изменение, добавление новой записи.

Листинг 2 — Класс Cafe

public class Cafe

{

static private string connString = "Data Source=DESKTOP-5E5L14V;"

+"Initial Catalog=cafe;Integrated Security=True";

public int idcafe { get; set; }

public string owner { get; set; }

public string title { get; set; }

public string address { get; set; }

public int? phone { get; set; }

public int? rating { get; set; }

private static Cafe Read(SqlDataReader cafeReader);

public static List<Cafe> Get();

public static Cafe Get(int cafeId);

public static void Insert(Cafe cafe);

public static void Delete(int id);

public static void Update(int id, Cafe newCafe);

public static int GetPostCafe(int cafeID);

public static int GetCountServe(int cafeID);

public void change(int id, Cafe newCafe);

public void add( Cafe cafe);

public Cafe search(string address);

}

Методы класса позволяют работать с SQL Server, записывая данные в таблицы или, наоборот, считывая их оттуда. Например, метод Get в классе Cafe используется для получения полного списка кафе и реализован с помощью SQL-запроса на выборку данных. В частности, этот метод используется для отображения информации о кафе (см. скрипт создания в приложении Б).

Метод Get который принимает параметр id работает схожим образом, но возвращают лишь одно значение. Из входного параметра можно понять, что данный метод возвращает информацию, полученную по индивидуальному номеру кафе.

Помимо основных методов были добавлены методы такие как GetPostCafe этот метод является реализацией хранимой процедуры который возвращает среднюю зарплату по данному кафе. Метод GetCountServe является реализацией представления и возвращает суммарное количество сотрудников каждого кафе.

Методы Delete, Update и Insert, как и все прочие, используют SQL-запросы. С реализацией этих методов можно ознакомиться в приложении Б. Данные методы удаляют, обновляют или удаляют данные в базе данных. Метод Insert, использует инструкцию INSERT INTO языка SQL. В метод передается список данных и после выполнения метода insert, данные заносятся в таблицу базы данных, им присваивается индивидуальный номер. Метод Delete, использует инструкцию DELETE FROM языка SQL. В метод передается идентификатор поля данного класса и после выполнения метода delete, данное поле в таблице базы данных удаляется. Метод Update, использует инструкцию Update (название таблицы) SET языка SQL. В метод передается идентификатор поля данного класса и список данных, после выполнения метода update, данные в таблице базы данных обновляются.

Все остальные классы содержат соответствующие методы. Изменяются лишь значения, возвращаемые методами (такими, как Get, Read, Get с передаваемым параметром id), текст SQL-запросов (чтобы взаимодействие происходило с указанными таблицами), а также происходит небольшая модификация методов добавления, обновления и удаления.

Отличия от класса Cafe имеет класс Composition, представленный в листинге 3. Данный класс, так же имеет такие типы данных, как строковый и целочисленный.

Листинг 3 — Класс Composition

public class Composition

{

static private string connString = "Data Source=DESKTOP-5E5L14V; "

+"Initial Catalog=cafe;Integrated Security=True";

public int idcomposition { get; set; }

public int numingredient { get; set; }

public Ingredient ingredient { get; set; }

public Dish dish { get; set; }

private static Composition Read(SqlDataReader comReader);

public static List<Composition> Get();

public static void Insert(Composition composition);

public static void Delete(int id);

public static void Update(int id, Composition newComposition);

}

Можно заметить, что данный класс содержит свойства Ingredient, и Dish, которые в свою очередь являются уже реализованными классами. Такой подход позволяет быстро получать необходимую информацию, например, когда необходимо название блюда, его можно получить следующим образом: Dish.title. Точно такой подход реализован и в самой таблице Dish, в которой содержится поле Cafe. Кроме указанных ранее классов, существуют так же классы Supplier, Serve, Post. В данных классах реализованы соответствующие связи: для Supplier и Serve – связь с кафе, в Post – связь с сотрудниками кафе. В листинге приложении В представлен класс Composition. В нем можно увидеть, что метод, получающий все записи таблицы, реализован отличным образом: после считывания индивидуального номера состава вызывается метод Read, который вызывает методы получения ингредиентов и блюда по индивидуальному номеру. Это инициализирует свойства Ingredient и Dish. Информация инициализируется методом get класса Ingredient и Dish, а затем присваивается свойству класса Composition.

Все классы содержат соответствующие методы. После реализации необходимых классов, требовалось отобразить информацию из базы данных на формах приложения.

Для перехода между сущностями использовался элемент TabControl. Для отображения информации базы данных в приложении используются элементы DataGridView, ComboBox и TextBox.

Для более подробной информации по реализации описанных механизмов смотрите соответствующие разделы.

# Импорт и экспорт данных. XML-файлы и XSD-схема

В программе реализован импорт и экспорт данных при помощи XML-файлов для таблиц Cafe и Supplier. Для этого используются классы XmlDocument, XmlReader и XmlSchema. XSD-схема предварительно описана вручную и содержит в себе все характеристики указанных таблиц (столбцы, их тип и необходимость; ограничения дозволенных минимального и максимального количества появлений таблицы в обрабатываемом XML-файле). Таблицы Cafe и Supplier имеют связь «один-ко-многим». Соответственно, таблица Supplier содержит внешний ключ для реализации данной связи в базе данных. В таком случае значение внешнего ключа таблицы Supplier можно не хранить в документе – связь между таблицами полностью реализуется вложенностью элементов. Полный текст XSD-схемы приведён в листинге 4.

Листинг 4 — Текст XSD-схемы

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="XMLSchema1"

targetNamespace="http://tempuri.org/XMLSchema1.xsd"

elementFormDefault="qualified"

xmlns="http://tempuri.org/XMLSchema1.xsd"

xmlns:mstns="http://tempuri.org/XMLSchema1.xsd"

xmlns:xs="http://www.w3.org/2001/XMLSchema"

>

<xs:element name="BDCafe">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Cafe">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Supplier">

<xs:complexType>

<xs:attribute name="title" type="xs:string" use="required" />

<xs:attribute name="product" type="xs:string" use="required" />

<xs:attribute name="Frequencyofdeliveries" type="xs:int" use="optional" />

<xs:attribute name="Volumeofdeliveries" type="xs:int" use="optional" />

<xs:attribute name="cafeAddress" type="xs:int" use="required" />

</xs:complexType>

</xs:element>

</xs:sequence>

<xs:attribute name="owner" type="xs:string" use="required" />

<xs:attribute name="title" type="xs:string" use="required" />

<xs:attribute name="address" type="xs:string" use="required" />

<xs:attribute name="phone" type="xs:int" use="optional" />

<xs:attribute name="rating" type="xs:int" use="optional" />

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Кроме того, были разработаны методы для реализации импорта и экспорта с учётом разработанной схемы, находящиеся в исходном коде программы. Для этого был создан класс XML (листинг 5).

Листинг 5 — Класс XML

class XML

{

public static string targetNamespace = "http://tempuri.org/XMLSchema1.xsd";

public static string xds = "XMLSchema1.xsd";

private static bool GetCafeSup(Cafe cafe, Supplier supplier);

public static bool export(string f\_name);

static public void import(string f\_name);

}

Для формирования XML-файла реализован метод export, основывающийся на схеме, которая указана в пути xsd. С помощью ранее реализованных методов получается список всех кафе, а затем формируется XmlElemtn, в который и вносится необходимая информация. Формирование происходит с помощью внутренних циклов. Данные циклы являются последовательными, а не вложенными. Пример формирования представлен в приложении B.

При импорте загружаемый файл проверяется на соответствие XSD-схемы и на наличие содержимых записей в таблице. Сам импорт осуществляется с помощью процедуры import, которая принимает импортируемый XML. В случае, если какая-либо запись уже существует в базе данных, проводится обновление существующей записи, иначе происходит добавление не существующей записи в базу данных.

Реализация вышеописанных механизмов приведена в приложении В.

# Триггеры, представления и хранимые процедуры

В программе также задействованы такие механизмы баз данных, как триггеры, представления и хранимые процедуры.

Представление используется для выполнения SELECT-запроса, который возвращает таблицу данных из физических таблиц, указанных в тексте запроса, и представляет данные в виде табличного набора данных. Например, в данном проекте представление формируется с помощью соответствующего SQL-запроса и выбирает из таблиц Cafe, Serve и Post необходимую информацию, после чего получает краткую статистику о средней заработной плате сотрудников каждого кафе. Реализация данного механизма для класса Cafe приведена в листинге 6.

Листинг 6 — Реализация представления

public static int GetPostCafe(int cafeID)

{

SqlConnection conn = new SqlConnection(connString);

SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText = "select s.[средняя зп] from salarylist as s"

+" where s.idcafe = @cafeID";

cmd.Parameters.AddWithValue("cafeID", cafeID);

conn.Open();

int result = (int)cmd.ExecuteScalar();

conn.Close();

return result;

}

В базе данных были реализованы хранимые процедуры – объекты базы данных, представляющие собой набор инструкций и хранящиеся на сервере. Хранимая процедура в программе используется для выполнения вычислений соответствующих полей. Например, для подсчета рабочего персонала в каждом кафе, и в данном примере вызывается каждый раз, когда пользователю нужно будет привести краткую статистику по всем кофейням. Реализация данного механизма для класса Cafe приведена в листинге 7.

Листинг 7 — Реализация хранимой процедуры

public static int GetCountServe(int cafeID)

{

SqlConnection conn = new SqlConnection(connString);

SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText = "EXEC CountServe @cafeID";

cmd.Parameters.AddWithValue("cafeID", cafeID);

conn.Open();

int result = (int)cmd.ExecuteScalar();

conn.Close();

return result;

}

Триггер вызывается при возникновении в заданной таблице определенных действий и не по запросу, а при возникновении события на сервере базы данных. В базе данных был реализован триггер, представленный в листинге 8. Он необходим для удаления связующих значений полей таких таблиц как Café, Serve, Dish.

Данный триггер срабатывает при удалении записи в базе данных (на это указывает инструкция INSTEAD OF DELETE).

Сразу после удаления записи в таблице Serve происходит удаление записи из таблицы Post по соответствующему идентифицированному полю Post.serve\_idserve, которое равно полю Serve.idserve. При удалении записей у таблиц Café и Dish срабатывает схожий триггер, который так же удаляет связующие записи таблиц.

Листинг 8 — Реализация триггеров

USE cafe

GO

CREATE TRIGGER serve\_DELETE

ON serve

INSTEAD OF DELETE

AS

BEGIN

DECLARE @ServID int

SELECT @ServID = idserve FROM deleted

DELETE FROM post WHERE serve\_idserve = @ServID

DELETE FROM serve WHERE idserve = @ServID

END

CREATE TRIGGER cafe\_DELETE

ON cafe

INSTEAD OF DELETE

AS

BEGIN

DECLARE @cafeID int

SELECT @cafeID = idcafe FROM deleted

DELETE FROM supplier WHERE supplier.cafe\_idcafe = @cafeID

DELETE FROM serve WHERE serve.cafe\_idcafe = @cafeID

DELETE FROM cafe WHERE idcafe = @cafeID

END

CREATE TRIGGER dish\_DELETE

ON dish

INSTEAD OF DELETE

AS

BEGIN

DECLARE @dishID int

SELECT @dishID = iddish FROM deleted

DELETE FROM composition WHERE dish\_iddish = @dishID

DELETE FROM dish WHERE iddish = @dishID

END

# Список литературы

1. Бен-Ган И. Microsoft SQL Server 2012. Основы T-SQL – М.: Эксмо, 2015. – 298 с.
2. Дейт К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL. – М.: Символ-плюс, 2010. – 474 с.
3. Кириллов В.В., Громов Г.Ю. Введение в реляционные базы данных. – СПб.: БХВ-Петербург, 2009. – 464 с.
4. Котов О.М. Язык C#. Краткое описание и введение в технологии программирования. Учебное пособие. – Екатеринбург: Изд-во Урал.ун-та, 2014. – 208 с.
5. Лобел Л., Браст Дж. Э., Форте С. Разработка приложений на основе Microsoft SQL Server 2008. – СПб.: БХВ-Петербург, 2010. – 295 с.
6. Сарка Д., Талмейдж Р., Бен-Ган И. Microsoft SQL Server 2012. Создание запросов. – М.: Изд-во «Русская редакция», 2014. – 347 с.
7. Хендерсон К. Microsoft SQL Server: структура и реализация. Профессиональное руководство. – СПб.: Вильямс, 2012. – 1056 с.
8. Эндрю Стиллмен, Дженнифер Гринн. Изучаем C#. 3-е издание. – СПб.: Питер, 2014. – 816 с.
9. SQL Server. Microsoft. [Электронный ресурс]. URL: <https://www.microsoft.com/ru-ru/server-cloud/products/sql-server/overview.aspx> (дата обращения 20.12.2018).
10. Microsoft Developer Network. [Электронный ресурс] URL: <https://msdn.microsoft.com/ru-ru/library/system.configuration.configurationmanager(v=vs.110).aspx> (дата обращения 22.12.2018)

# Приложение A — SQL-запросы формирования БД

CREATE TABLE cafe(

idcafe INT PRIMARY KEY,

owner VARCHAR(255) NOT NULL ,

title VARCHAR(255) NOT NULL ,

address VARCHAR(255) NOT NULL ,

phone INT NULL ,

rating INT NULL ,

)

CREATE TABLE composition(

idcomposition INT PRIMARY KEY,

numingredient INT NOT NULL ,

ingredient\_idingredient INT NULL REFERENCES ingredient (idingredient)

ON DELETE SET NULL

ON UPDATE CASCADE,

dish\_iddish INT NULL REFERENCES dish (iddish)

ON DELETE SET NULL

ON UPDATE CASCADE,

)

CREATE TABLE dish(

iddish INT PRIMARY KEY,

title VARCHAR(255) NOT NULL ,

quantily INT NOT NULL ,

cafe\_idcafe INT NULL REFERENCES cafe (idcafe)

ON DELETE SET NULL

ON UPDATE CASCADE

)

CREATE TABLE ingredient (

idingredient INT PRIMARY KEY,

product VARCHAR(255) NOT NULL ,

)

CREATE TABLE post (

idpost INT NOT NULL PRIMARY KEY,

title VARCHAR(255) NOT NULL ,

salary INT NOT NULL,

serve\_idserve INT NULL REFERENCES serve (idserve)

ON DELETE SET NULL

ON UPDATE CASCADE

)

CREATE TABLE serve(

idserve INT PRIMARY KEY,

name VARCHAR(255) NOT NULL ,

passport INT NOT NULL ,

education VARCHAR(45) NULL ,

experience INT NOT NULL ,

cafe\_idcafe INT NULL REFERENCES cafe (idcafe)

ON DELETE SET NULL

ON UPDATE CASCADE

)

CREATE TABLE supplier (

idsupplier INT PRIMARY KEY,

title VARCHAR(255) NOT NULL ,

product VARCHAR(255) NOT NULL ,

Frequencyofdeliveries INT NULL ,

Volumeofdeliveries INT NULL ,

cafe\_idcafe INT NULL REFERENCES cafe (idcafe)

ON DELETE SET NULL

ON UPDATE CASCADE

)

# Приложение Б — Группа методов класса Café и Composition

public class Cafe

{

public int idcafe { get; set; }

public string owner { get; set; }

public string title { get; set; }

public string address { get; set; }

public int? phone { get; set; }

public int? rating { get; set; }

private static Cafe Read(SqlDataReader cafeReader)

{

Cafe newOne = new Cafe();

newOne.idcafe = cafeReader.GetInt32(0);

newOne.owner = cafeReader.GetString(1);

newOne.title = cafeReader.GetString(2);

newOne.address = cafeReader.GetString(3);

try

{

newOne.phone = cafeReader.GetInt32(4);

}

catch (System.Data.SqlTypes.SqlNullValueException)

{

newOne.phone = null;

}

try

{

newOne.rating = cafeReader.GetInt32(5);

}

catch (System.Data.SqlTypes.SqlNullValueException)

{

newOne.rating = null;

}

return newOne;

}

public static List<Cafe> Get()//считываем с БД

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn);

SqlCommand cmd = new SqlCommand("SELECT idcafe, owner, title, "

+" address, phone, rating FROM cafe", conn);

SqlDataReader cafeReader = cmd.ExecuteReader();

List<Cafe> cafes = new List<Cafe>();

while (cafeReader.Read())

{

cafes.Add(Read(cafeReader));

}

cafeReader.Close();

conn.Close();

return cafes;

}

public static Cafe Get(int cafeId)

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn);

SqlCommand cmd = new SqlCommand("SELECT idcafe, owner, title, "

+"address, phone, rating FROM cafe", conn);

Cafe cafe;

conn.Open();

SqlDataReader cafeReader = cmd.ExecuteReader();

while (cafeReader.Read())

{

int currID = cafeReader.GetInt32(0);

if (currID != cafeId)

continue;

cafe = Read(cafeReader);

cafeReader.Close();

conn.Close();

return cafe;

}

cafeReader.Close();

conn.Close();

throw new System.MissingMemberException();

}

public static void Insert(Cafe cafe)//добавить

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn);

SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText = "INSERT INTO cafe(owner, title, address";

string values = "VALUES( @\_owner, @\_title, @\_address";

cmd.Parameters.AddWithValue("\_owner", cafe.owner);

cmd.Parameters.AddWithValue("\_title", cafe.title);

cmd.Parameters.AddWithValue("\_address", cafe.address);

if (cafe.phone != null)

{

cmd.CommandText += ", phone";

values += ", @\_phone";

cmd.Parameters.AddWithValue("\_phone", cafe.phone);

}

if (cafe.rating != null)

{

cmd.CommandText += ", rating";

values += ", @\_rating";

cmd.Parameters.AddWithValue("\_rating", cafe.rating);

}

cmd.CommandText += ") " + values + ");";

cmd.ExecuteNonQuery();

conn.Close();

}

public static void Delete(int id)

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn);

SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText = @"DELETE FROM cafe

WHERE idcafe = @id";

cmd.Parameters.AddWithValue("id", id);

cmd.ExecuteNonQuery();

conn.Close();

}

public static void Update(int id, Cafe newCafe)

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn);

SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText = "Update cafe SET owner = @\_owner, "

+" title = @\_title, address = @\_address";

cmd.Parameters.AddWithValue("\_owner", newCafe.owner);

cmd.Parameters.AddWithValue("\_title", newCafe.title);

cmd.Parameters.AddWithValue("\_address", newCafe.address);

if (newCafe.phone != null)

{

cmd.CommandText += ", phone = @\_phone";

cmd.Parameters.AddWithValue("\_phone", newCafe.phone);

}

else

{

cmd.CommandText += ", phone = NULL";

}

if (newCafe.rating != null)

{

cmd.CommandText += ", rating = @\_rating";

cmd.Parameters.AddWithValue("\_rating", newCafe.rating);

}

else

{

cmd.CommandText += ", rating = NULL";

}

cmd.CommandText += " WHERE idcafe = @id";

cmd.Parameters.AddWithValue("id", id);

cmd.ExecuteNonQuery();

conn.Close();

}

public static int GetPostCafe(int cafeID)

{

SqlConnection conn = new SqlConnection(connString);

SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText = "select s.[средняя зп] from salarylist as "

+"s where s.idcafe = @cafeID";

cmd.Parameters.AddWithValue("cafeID", cafeID);

conn.Open();

int result = (int)cmd.ExecuteScalar();

conn.Close();

return result;

}

public static int GetCountServe(int cafeID)

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn);

SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText = "EXEC CountServe @cafeID";

cmd.Parameters.AddWithValue("cafeID", cafeID);

int result = (int)cmd.ExecuteScalar();

conn.Close();

return result;

}

public void change(int id, Cafe newCafe)

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn);

SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText = "Update cafe SET owner = @\_owner, "

+"title = @\_title, address = @\_address";

cmd.Parameters.AddWithValue("\_owner", newCafe.owner);

cmd.Parameters.AddWithValue("\_title", newCafe.title);

cmd.Parameters.AddWithValue("\_address", newCafe.address);

if (newCafe.phone != null)

{

cmd.CommandText += ", phone = @\_phone";

cmd.Parameters.AddWithValue("\_phone", newCafe.phone);

}

else

{

cmd.CommandText += ", phone = NULL";

}

if (newCafe.rating != null)

{

cmd.CommandText += ", rating = @\_rating";

cmd.Parameters.AddWithValue("\_rating", newCafe.rating);

}

else

{

cmd.CommandText += ", rating = NULL";

}

cmd.CommandText += " WHERE idcafe = @id";

cmd.Parameters.AddWithValue("id", id);

cmd.ExecuteNonQuery();

conn.Close();

}

public void add( Cafe cafe)

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn);

SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText = "INSERT INTO cafe(owner, title, address";

string values = "VALUES( @\_owner, @\_title, @\_address";

cmd.Parameters.AddWithValue("\_owner", cafe.owner);

cmd.Parameters.AddWithValue("\_title", cafe.title);

cmd.Parameters.AddWithValue("\_address", cafe.address);

if (cafe.phone != null)

{

cmd.CommandText += ", phone";

values += ", @\_phone";

cmd.Parameters.AddWithValue("\_phone", cafe.phone);

}

if (cafe.rating != null)

{

cmd.CommandText += ", rating";

values += ", @\_rating";

cmd.Parameters.AddWithValue("\_rating", cafe.rating);

}

cmd.CommandText += ") " + values + ");";

cmd.ExecuteNonQuery();

conn.Close();

}

public Cafe search(string address)

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn);

SqlCommand cmd = new SqlCommand("SELECT \* FROM cafe WHERE"

+" cafe.address = @address ", conn);

cmd.Parameters.AddWithValue("address", address);

SqlDataReader cafeReader = cmd.ExecuteReader();

if (cafeReader.Read())

{

return Read(cafeReader);

}

else

return null;

}

}

public class Composition

{

public int idcomposition { get; set; }

public int numingredient { get; set; }

public Ingredient ingredient { get; set; }

public Dish dish { get; set; }

private static Composition Read(SqlDataReader comReader)

{

Composition newOne = new Composition();

newOne.idcomposition = comReader.GetInt32(0);

newOne.numingredient = comReader.GetInt32(1);

newOne.ingredient = Ingredient.Get(comReader.GetInt32(2));

newOne.dish = Dish.Get(comReader.GetInt32(3));

return newOne;

}

public static List<Composition> Get()

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn);

SqlCommand cmd = new SqlCommand("SELECT idcomposition, numingredient, "

+" ingredient\_idingredient, dish\_iddish FROM composition", conn);

SqlDataReader comReader = cmd.ExecuteReader();

List<Composition> composition = new List<Composition>();

while (comReader.Read())

{

composition.Add(Read(comReader));

}

comReader.Close();

conn.Close();

return composition;

}

public static void Insert(Composition composition)

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn); SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText = "INSERT INTO composition(numingredient, "

+" ingredient\_idingredient, dish\_iddish";

string values = "VALUES( @\_numingredient, @\_ingredient\_idingredient, "

+" @\_dish\_iddish";

cmd.Parameters.AddWithValue("\_numingredient", composition.numingredient);

cmd.Parameters.AddWithValue("\_ingredient\_idingredient","

+" composition.ingredient.idingredient);

cmd.Parameters.AddWithValue("\_dish\_iddish", composition.dish.iddish);

cmd.CommandText += ") " + values + ");";

cmd.ExecuteNonQuery();

conn.Close();

}

public static void Delete(int id)

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn); SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText = @"DELETE FROM composition

WHERE idcomposition = @id";

cmd.Parameters.AddWithValue("id", id);

cmd.ExecuteNonQuery();

conn.Close();

}

public static void Update(int id, Composition newComposition)

{

ConnectionString str = new ConnectionString();

SqlConnection conn = new SqlConnection();

str.Open(conn);

SqlCommand cmd = new SqlCommand("", conn);

cmd.CommandText ="Update composition SET numingredient=@\_numingredient, "

+" ingredient\_idingredient = @\_ingredient\_idingredient, "

+" dish\_iddish = @\_dish\_iddish";

cmd.Parameters.AddWithValue("\_numingredient","

+" newComposition.numingredient);

cmd.Parameters.AddWithValue("\_ingredient\_idingredient","

+" newComposition.ingredient.idingredient);

cmd.Parameters.AddWithValue("\_dish\_iddish", newComposition.dish.iddish);

cmd.CommandText += " WHERE idcomposition = @id";

cmd.Parameters.AddWithValue("id", id);

cmd.ExecuteNonQuery();

conn.Close();

}

}

# Приложение В — Импорт и экспорт данных

Экспорт:

public static bool export(string f\_name)

{

XmlDocument doc = new XmlDocument();

XmlElement cafeElement, SupElemnt, root;

XmlDeclaration declaration;

XmlAttribute attribute;

doc.Schemas.Add(targetNamespace, xds);

declaration = doc.CreateXmlDeclaration("1.0", "UTF-8", null);

root = doc.CreateElement("BDCafe");

doc.AppendChild(root);

List<Cafe> cafeList;

cafeList = Cafe.Get();

List<Supplier> supList;

supList = Supplier.Get();

foreach (Cafe cafe in cafeList)

{

cafeElement = doc.CreateElement("Cafe");

attribute = doc.CreateAttribute("owner");

attribute.Value = cafe.owner;

cafeElement.Attributes.Append(attribute);

attribute = doc.CreateAttribute("title");

attribute.Value = cafe.title;

cafeElement.Attributes.Append(attribute);

attribute = doc.CreateAttribute("address");

attribute.Value = cafe.address;

cafeElement.Attributes.Append(attribute);

attribute = doc.CreateAttribute("phone");

attribute.Value = cafe.phone.ToString();

cafeElement.Attributes.Append(attribute);

attribute = doc.CreateAttribute("rating");

attribute.Value = cafe.rating.ToString();

cafeElement.Attributes.Append(attribute);

foreach (Supplier supplier in supList)

{

if (GetCafeSup(cafe, supplier) == true)

{

SupElemnt = doc.CreateElement("Supplier");

attribute = doc.CreateAttribute("title");

attribute.Value = supplier.title;

SupElemnt.Attributes.Append(attribute);

attribute = doc.CreateAttribute("product");

attribute.Value = supplier.product;

SupElemnt.Attributes.Append(attribute);

attribute = doc.CreateAttribute("Frequencyofdeliveries");

attribute.Value = supplier.FrequencyOfDeliveries.ToString();

SupElemnt.Attributes.Append(attribute);

attribute = doc.CreateAttribute("Volumeofdeliveries");

attribute.Value = supplier.VolumeOfDeliveries.ToString();

SupElemnt.Attributes.Append(attribute);

attribute = doc.CreateAttribute("cafeAddress");

attribute.Value = supplier.cafe.address; //ID

SupElemnt.Attributes.Append(attribute);

cafeElement.AppendChild(SupElemnt);

}

}

root.AppendChild(cafeElement);

}

doc.Save(f\_name);

try

{

doc.DocumentElement.SetAttribute("xmlns", targetNamespace);

doc.Validate(null);

return true;

}

catch

{

return false;

}

}

Импорт:

static public void import(string f\_name)

{

XmlDocument doc = new XmlDocument();

doc.Load(f\_name);

XmlElement root = doc.DocumentElement;

doc.Schemas.Add(targetNamespace, xds);

if (string.IsNullOrEmpty(doc.DocumentElement.NamespaceURI))

{

doc.DocumentElement.SetAttribute("xmlns", targetNamespace);

doc.LoadXml(doc.InnerXml);

}

foreach (XmlNode n in root)

{

Cafe c = new Cafe();

List<Cafe> cafeList;

cafeList = Cafe.Get();

c.owner = n.Attributes["owner"].Value;

c.title = n.Attributes["title"].Value;

c.address = n.Attributes["address"].Value;

c.phone = Int32.Parse(n.Attributes["phone"].Value);

c.rating = Int32.Parse(n.Attributes["rating"].Value);

Cafe np = new Cafe();

np = np.search(c.address);

if (np == null)

{

c.add(c);

cafeList = Cafe.Get();

c.idcafe = cafeList.LastOrDefault().idcafe;

}

else

{

c.change(np.idcafe, c);//обновляет

c.idcafe = np.idcafe;

}

foreach (XmlNode t in n)

{

Supplier s = new Supplier();

if (t.Attributes["cafeAddress"].Value != c.address) break;

s.title = t.Attributes["title"].Value;

s.product = t.Attributes["product"].Value;

s.FrequencyOfDeliveries = Int32.Parse(t.Attributes["Frequencyofdeliveries"].Value);

s.VolumeOfDeliveries = Int32.Parse(t.Attributes["Volumeofdeliveries"].Value);

s.cafe = Cafe.Get(c.idcafe);

Supplier sp = new Supplier();

sp = sp.search(s.title);

if (sp == null)

{

s.add(s);

}

else

{

s.change(sp.idsupplier, s);

}

}

}

}

# Приложение Г — Примеры работы программы

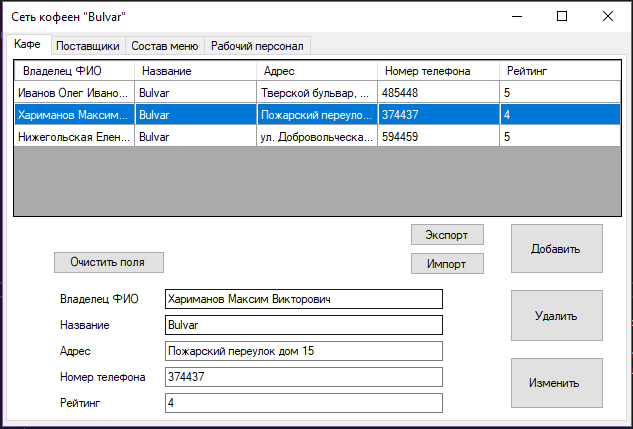


Рисунок 3 — Стартовое окно

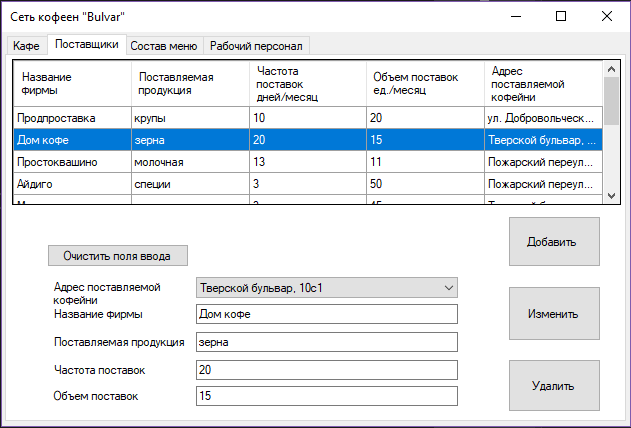
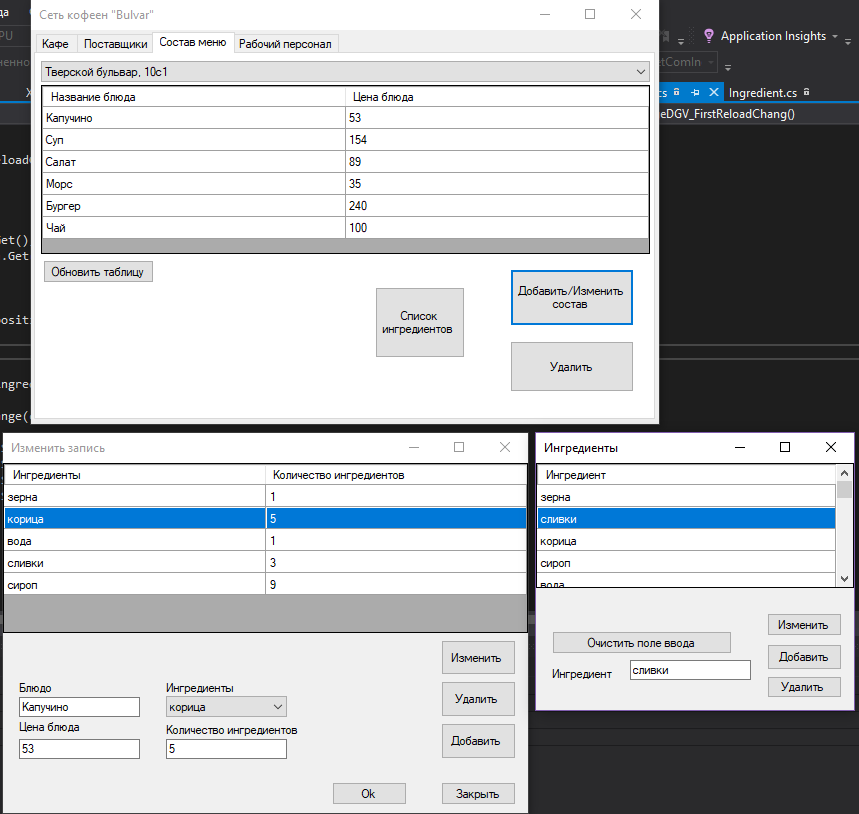


Рисунок 4 — Окно “Поставщики”

Рисунок 5 — Окно “Состав меню”

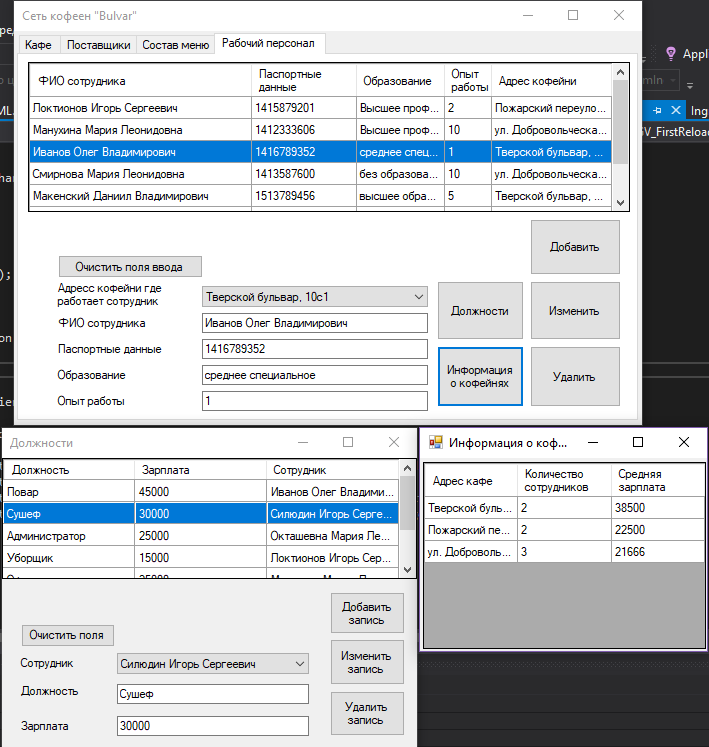


Рисунок 6 — Окно “Рабочий персонал”