УДК 004

РАБОТА С БАЗАМИ ДАННЫХ НА C# НА ПРИМЕРЕ MICROSOFT ACCESS

Кевбрин Владислав Андреевич
студент

Кузнецов Дмитрий Александрович
студент

Мордовский государственный университет им. Н.П. Огарева, Саранск

author@apriori-journal.ru

Аннотация. В данной статье рассмотрены основные особенности работы с базами данных на подключенном уровне на языке программирования C#.

Ключевые слова: базы данных; программирование; C#; .NET; ADO.NET; SQL; OLE DB.

WORKING WITH DATABASES ON C# AT EXAMPLE OF MICROSOFT ACCESS

Kevbrin Vladislav Andreevich
student

Kuznetsov Dmitriy Alexandrovich
student

Ogarev Mordovian State University, Saransk

Abstract. In this article the main features of work with databases at the connected level in the C# programming language are considered.

Key words: databases; programming; C#; .NET; ADO.NET; SQL; OLE DB.

Платформа .NET определяет ряд пространств имен, которые позволяют непосредственно взаимодействовать с локальными и удаленными реляционными базами данных. Все вместе эти пространства имен известны как ADO.NET. Платформа .NET поддерживает множество различных поставщиков данных, каждый из которых оптимизирован на взаимодействие с конкретной СУБД (Microsoft SQL Server, Oracle, MySQL и т.д.). Отсюда и возникают проблемы при работе с разными базами данных.

С точки зрения программиста, основной сборкой является System.Data.dll. Работа с Microsoft Access возможна через сборку System.Data.OleDb, реализующую классы и методы для работы с интерфейсом OLE DB (*Object Linking and Embedding, Database*).

Для анализа используется заранее подготовленная база данных о заказах автомобилей. Она нормализована по стандарту 3NF [1].

База данных, используемая в примере, состоит из 2 таблиц. Первая содержит информацию о заказе автомобиля: код заказчика, фамилию заказчика, имя заказчика, модель заказанного автомобиля. Вторая таблица содержит информацию о заказанных моделях автомобиля, а именно: уникальный код модели, название модели, цвет, цену.

Основными классами, при работе с данной базой данных на подключенном уровне, являются:

- OleDbConnection – представляет открытое подключение к источнику данных;
- OleDbCommand – представляет набор операторов SQL или хранимую процедуру, применяемую к источнику данных;
- OleDbDataReader – предоставляет способ чтения потока строк данных из источника только в прямом порядке.

Библиотеки ADO.NET можно применять тремя концептуально разными способами: в подключенном режиме, в автономном режиме и с помощью технологии Entity Framework. В работе рассматривается подключенный режим.

Подключенный уровень ADO.NET позволяет взаимодействовать с базой данных с помощью объектов подключения, чтения данных и команд конкретного поставщика данных. При необходимости подключиться к базе данных и прочитать записи с помощью объекта чтения данных нужно выполнить следующие шаги:

1. Создать, настроить и открыть объект подключения (экземпляр класса OleDbConnection);
2. Создать и настроить объект команды (экземпляр класса OleDbCommand), указав объект подключения в аргументе конструктора или через свойство Connection;
3. Вызвать метод OleDbCommand.ExecuteReader() настроенного объекта команды;
4. Обработать каждую запись с помощью метода OleDbDataReader.Read() объекта чтения данных.

Первое что требуется сделать – это установить сеанс с источником данных с помощью объекта подключения. Создается объект подключения с помощью конструктора класса OleDbConnection. У объектов подключения .NET имеется форматированная строка подключения, которая содержит ряд пар имя/значение, разделенных точками с запятой. Информация в строке подключения содержит имя машины, к которой нужно подключиться, необходимые параметры безопасности, имя базы данных на этой машине и другую информацию, зависящую от поставщика.

Самым простым методом создания объекта подключения является создание с заранее заданной строкой, содержащей адрес местоположения базы данных в системе. Создание строки на языке программирования C# похоже на создание строки на языке программирования C [2], поддерживаются базовые операции (конкатенации строк и т.д.). Например:

```
//Задание строки подключения
string connStr = @"Provider=Microsoft.ACE.OLEDB.12.0;”+
”DataSource= D:\TestApplication\ "+
"BD\Test.accdb";
//Создание объекта подключения
OleDbConnection conn = new OleDbConnection();
conn.ConnectionString = connStr;
//Открытие подключения
conn.Open();
```

В результате выполнения данного кода создается подключение к базе данных Test.accdb, созданной посредством Microsoft Access, находящейся в директории, путь к которой указан в строке connStr. Часть строки @"Provider=Microsoft.ACE.OLEDB.12.0;” содержит дополнительные данные по подключению: свойством Provider мы задаем необходимый провайдер для подключения базы данных. Текущий провайдер позволяет нам работать с базами данных, созданными в Microsoft Access 2010 и более поздних версиях. Метод OleDbConnection.Open() открывает подключение к базе данных со значениями свойств, определяемыми объектом ConnectionString. Для закрытия подключения достаточно вызвать метод OleDbConnection.Close() от объекта подключения (в примере таковым является conn).

Во многих случаях, когда требуется указать множество параметров строки подключения, удобнее использовать объект типа ConnectionStringBuilder. Поставщики данных ADO.NET, разработанные Microsoft, поддерживают объекты построителей строк подключения, которые позволяют устанавливать пары имя/значение с помощью строго типизированных свойств.

При работе с базами данных на подключенном уровне основным объектом, с которым работает программист, является объект команд. При создании объекта команды можно сразу задать SQL-запрос, пере-

дав его с помощью параметра конструктора (или непосредственно свойства CommandText) [3]. Кроме того, при создании объекта команды необходимо указать подключение, которое будет в нем применяться. Это тоже можно сделать либо через параметр конструктора, либо с помощью свойства Connection, как в следующем фрагменте кода:

```
// Создание объекта команды с помощью аргументов конструктора.
string result = "Select * From Orders”;
OleDbCommand myCommand = new OleDbCommand(result, cn);
// Создание еще одного объекта команды с помощью свойств.
SqlCommand testCommand = new SqlCommand();
testCommand.Connection = cn;
testCommand.CommandText = result;
```

Объект команд используется для выполнения многих запросов, в том числе для добавления, изменения, удаления строк в базу данных.

Для вывода информации из базы данных лучше всего использовать объект чтения данных, который будет максимально быстро выдавать по одной записи. Объекты чтения данных удобны, если нужно быстро просмотреть большой объем данных без необходимости представлять их в памяти. Нужно так же не забывать, что объекты чтения данных поддерживают открытое подключение к источнику данных, пока сеанс не будет явно закрыт. Объект чтения данных можно получить из объекта команды с помощью вызова ExecuteReader(). В объекте чтения данных имеет индексатор, который обеспечивает доступ к столбцам текущей записи. В качестве индекса можно использовать либо имя столбца, либо целочисленный индекс, начиная от нуля.

```
//Создание объекта команды
myCommand = new OleDbCommand("Select Orders.OrderId, "+ "Orders.LastName, Orders.Name, Car.CarModel, Car.ColorName, "+ "Car.Price From Orders, Car where Orders.CarModel=Car.CarId", cn);
//Создание объекта чтения данных
    using (OleDbDataReader reader = myCommand.ExecuteReader())
    {
        while (reader.Read())
        {
            Console.WriteLine("{0} {1} {2}: {3} to {4} for ${5}",
            reader["OrderId"], reader["LastName"], reader["Name"],
            reader["CarModel"], reader["ColorName"],  ["Price"]     );
        }
    }
```

В приведенном выше фрагменте кода, создается новый объект команд и в него загружается сложный запрос. Этот запрос выводит информацию из двух таблиц, связь которых обеспечивается через поля CarModel (из таблицы Oreders) и CarId (из таблицы Car). Далее создается объект чтения типа OleDbDataReader и в цикле на консоль выводится информация.

Для внесения изменений в базу данных (добавление, изменение и удаление записей) необходимо использовать объект команд, для каждого случая строка с командой будет разной. Синтаксис команд схож с языком SQL [4, 5], но имеет незначительные изменения. Для добавления записи – INSERT INTO, для изменения записи – UPDATE, для удаления записи – DELETE. Например, изменение модели автомобиля:

```
        Console.WriteLine("Чей заказ изменить (номер заказа )?");
        //Считываем с консоли индетификатор изменяемого заказа
        string Id = Console.ReadLine();
        Console.WriteLine("Введите модель автомобиля: ");
        //Считываем с консоли новое название модели автомобиля
        string carmodel = Console.ReadLine();
        //Создаем строку с оператором UPDATE
        result = String.Format("update Car set CarModel = '{0}' where CarId in"
+
        "(select CarModel from Orders where OrderId = {1})", carmodel, Id);
        //Создаем объект команд
        myCommand = new OleDbCommand(result, cn);
        //Вызываем выполнение команды
        myCommand.ExecuteNonQuery();
```

В приведенном примере, с помощью String.Format, создается строка с командой UPDATE, в которой указывается, что мы меняем модель автомобиля в таблице Car, где уникальный код будет содержаться в изменяемом заказе. Остальные типы изменений выполняются аналогично приведенному примеру. Использование String.Format позволяет нам создавать строку схожим образом, как это делалось на языке C операторами sprintf и sprint_s [6; 7]. Такой метод создания запросов применим только при относительно простых запросах, в более сложных случаях, рекомендуется использовать объекты класса OleDbParameter.

Выполнение команд UPDATE, DELETE, INSERT происходит посредством вызова метода OleDbCommand.ExecuteNonQuery().

Работа с базами данных на подключенном уровне на языке программирования C# очень похожа на работу запросами на SQL, но эта схожесть ограничивается лишь выполняемыми командами. Для того чтобы правильно работать с базами данных, необходимо знать информацию и уметь

использовать объекты, которые используется непосредственно для выполнения команд. Таким образом, для полноценного изучения принципов работы с базами данных на C# нужно комплексно изучить все три режима работы. Это можно осуществить в виде практических занятий, подкрепленных теоретическими знаниями о необходимых объектах [8].

Список использованных источников

1. Таланов В. М., Федосин С. А. Проектирование информационных систем и баз данных. Учеб. пособие. Саранск: Изд-во СВМО, 2013. 72 с.
2. Александров Э.Э., Афонин В.В. Введение в программирование на языке С. Саранск: Изд-во Мордов. ун-та, 2009. 316 с.
3. Эндрю Троелсен Язык программирования C# 5.0 и платформа .NET 4.5. М.: ООО «И.Д. Вильямс», 2013. 1312 с.
4. Аббакумов А.А., Акимов В.Л., Егунова А.И., Лещанкин К.А., Таланов В.М. Базы данных (MS ACCESS, MYSQL). Саранск: Изд-во Средневолжского математического общества, 2011. 112 с.
5. Аббакумов А.А., Егунова А.И., Таланов В. М. Базы данных (MS SQL Server). Учеб. пособие. Саранск: Изд-во СВМО, 2015. 66 с.
6. Александров Э.Э., Афонин В.В. Программирование на языке С в Microsoft Visual Studio 2010 [Электронный ресурс]. Режим доступа: http://www.intuit.ru/department/pl/prcmsvs2010 (дата обращения: 29.11.2015)
7. Александров Э.Э., Афонин В.В. Программирование на языке С в Microsoft Visual Studio 2010. Саранск: Изд-во Мордов. ун-та, 2010. 424 с.
8. Афонин В.В., Федосин С.А. О структурировании лабораторно-практических занятий при изучении дисциплин программирования // Образовательные технологии и общество. 2014. Т. 17. № 4. С. 497-506. [Электронный ресурс]. Режим доступа: http://ifets.ieee.org/russian/periodical/izgurn.html (дата обращения: 29.11.2015).