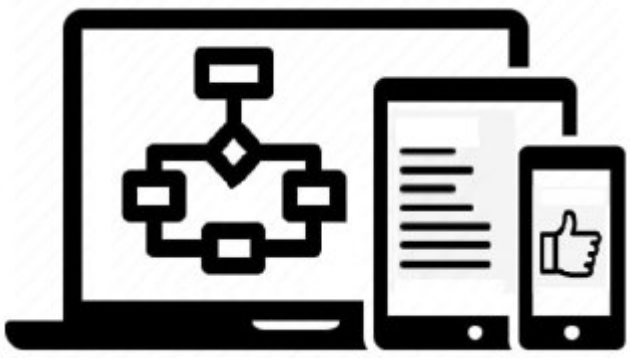
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Методические рекомендации к лабораторным работам для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» дневной и заочной форм обучения

Часть 2



Могилев 2023

УДК 004.65
ББК 32.973.26-0.18.2
С89

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «31» августа 2023 г., протокол № 1

Составители: канд. техн. наук, доц. К. В. Захарченков;
канд. техн. наук, доц. Т. В. Мрочек

Рецензент Ю. С. Романович

Методические рекомендации содержат описание четырех лабораторных работ», выполняемых во втором семестре изучения дисциплины «Системы управления базами данных». Рассматриваются основы работы с Microsoft SQL Server и языком Transact-SQL, а также ADO.NET и ASP.NET.

Учебное издание

СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Часть 2

Ответственный за выпуск	В. В. Кутузов
Корректор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Часть 2

6 Язык SQL. Работа с курсорами

Цель: научиться создавать курсоры в MS SQL Server Management Studio.

Теоретические положения

SQL-сервер способен возвращать в результате выполнения запроса сотни тысяч строк. Клиентские приложения не всегда могут справиться с такими объемами данных, т. к. для их хранения требуется много памяти. Решением этой проблемы является использование курсора.

Курсоры SQL-сервер представляют собой механизм обмена данными между сервером и клиентом. Курсор позволяет клиентским приложениям работать не с полным набором данных, а только с одной или несколькими строками.

Набор данных, имеющихся в таблице, называется полным набором строк. Набор строк, возвращаемый командой SELECT, называется результирующим набором. Курсоры работы с результирующим набором данных расширяют возможности пользователей по их обработке.

Скорость выполнения операций обработки данных с помощью курсора заметно ниже, чем у стандартных средств SQL-сервер.

Типы и поведение курсоров. Существует четыре основных типа курсоров, различающихся по предоставляемым возможностям. Тип курсора определяется на стадии его создания и не может быть изменен.

1 *Статические курсоры*. При открытии статического курсора (курсора моментального снимка) сервер выбирает все данные, соответствующие заданным критериям, и сохраняет полный результирующий набор строк в системной БД tempbd. На время открытия курсора сервер устанавливает блокировку на все строки, включенные в полный результирующий набор курсоров. Статический курсор не изменяется после создания и всегда отображает тот набор данных, который существовал в БД на момент его открытия. Внесение изменений в статические курсоры невозможно, т. к. нет гарантии существования неизменности строк данных, на основе которых построен курсор.

2 *Динамические курсоры*. При использовании динамических курсоров не создается полная копия исходных данных, а выполняется динамическая выборка данных из исходных таблиц при обращении пользователя к тем или иным данным. Сервер блокирует строки на время выборки и все изменения, вносимые пользователями в полный результирующий набор курсора, будут видны в курсоре при выборке. Если пользователь внес изменения в данные уже после из выборки курсором, то эти изменения не будут отражаться в курсоре.

3 *Последовательные курсоры*. Они не разрешают выполнять выборку данных в обратном направлении, т. е. пользователь может выбирать строки данных только от начала к концу.

4 *Ключевые курсоры*. Курсоры, зависящие от набора ключей (ключевые), построены на основе уникальных идентификаторов. Ключевой курсор представляет собой набор ключей, идентифицирующих строки полного результирующего набора курсора. Так как сохраняется информация только о ключевых полях строк, ключевые курсоры отражают все изменения, вносимые другими пользователями.

Управление курсорами.

При работе с курсором можно выделить пять основных операций.

1 Создание курсора. Перед тем как использовать курсор, его необходимо создать.

2 Открытие курсора. Сразу после создания курсор не содержит никаких данных так же, как переменная после объявления не содержит значения. Операция открытия курсора наполняет курсор данными.

3 Выборка из курсора и изменение строк данных с помощью курсора. После того как в курсор занесены данные, можно приступать к его использованию. В зависимости от того, какой тип курсора был объявлен при его создании, можно выполнять только чтение или еще и изменение данных.

4 Закрытие курсора. Когда все операции обработки данных завершены, и надобность в курсоре отпадает, его необходимо закрыть. При закрытии курсора сервер освобождает пространство в системной БД tempbd, выделенное курсором при его открытии.

5 Освобождение курсора. Операция удаляет курсор как объект.

Более подробно курсоры описаны в конспекте лекций и в [1–3, 5].

Далее рассмотрены примеры создания курсора для просмотра информации о студентах и выдачи информации об их числе.

```
DECLARE curs1 CURSOR
GLOBAL          /* Создается глобальный курсор, который
                 будет существовать до закрытия данного соединения*/
SCROLL          /* Создает прокручиваемый курсор */
KEYSET          /* Будет создан ключевой курсор */
TYPE_WARNING
FOR
SELECT          /* Какие поля будут показаны в курсоре */
Student.s_number_of_library_ticket, Student.s_last_name, Student.s_first_name,
Student.s_patronymic, Student.s_year_of_entrance, Student.s_faculty
FROM Student                    /* Из какой таблицы выбираются данные */
FOR READ ONLY                   /* Только для чтения */
OPEN GLOBAL curs1               /* открываем глобальный курсор */
DECLARE @@Counter int           /* объявляем переменную*/
SET @@Counter =@@CURSOR_ROWS    /*присваиваем ей число рядов
курсора */
Select @@Counter                /* выводим результат на экран */
CLOSE curs1                     /* закрываем курсор */
DEALLOCATE curs1                /* освобождаем курсор */
```

Курсор для просмотра заказов и подсчета общего количества заказанных книг.

```
DECLARE curs2 CURSOR
GLOBAL SCROLL KEYSET
TYPE_WARNING    /* Сервер будет информировать пользователя о неявном изменении типа курсора, если он несовместим с запросом SELECT */
FOR
SELECT                /* Что будет показано в курсоре */
Lecturer.l_last_name, Lecturer.l_first_name, Lecturer.l_patronymic, Book.b_author_last_name, Book.b_title, Order_of_book.ob_quantity, Book.b_price
FROM Book INNER JOIN (Lecturer INNER JOIN Order_of_book ON Lecturer.l_number_of_library_ticket = Order_of_book.ob_l_number_of_library_ticket)
ON Book.b_id_book = Order_of_book.ob_b_id_book
FOR UPDATE       /* Курсор для обновления */
OPEN GLOBAL curs2
DECLARE
  @@l_name varchar(20),
  @@f_name varchar(20),
  @@patronim varchar(20),
  @@author  varchar(20),
  @@title varchar(20),
  @@qty int,
  @@price int,
  @@Counter int,
  @@Var1 int
SET @@Counter = 1
SET @@Var1 = 0
WHILE @@COUNTER< @@CURSOR_ROWS        /* Пока счетчик просмотренных строк меньше их общего числа */
BEGIN
FETCH curs2 INTO @@l_name, @@f_name, @@patronim, @@author, @@title, @@qty    /* Просматриваем строки и значения */
SET @@Counter = @@Counter + 1            /* Меняем значение счетчика при переходе к другой строке */
SET @@Var1 = @@Var1 + @@qty * @@price    /* Суммируем стоимости заказанных книг */
END
SELECT @@Var1                /* выводим сумму на экран */
CLOSE curs2
DEALLOCATE curs2
```

Задание

В разрабатываемой БД необходимо реализовать пять курсоров статического и динамического типов, содержащих не менее пяти различных функций.

Содержание отчета: тема и цель работы; SQL-код пяти курсоров.

Контрольные вопросы

1 Что такое курсор? Какие типы курсоров различают?
2 Что такое полный и результирующий наборы строк?
3 Какие основные операции выделяют при работе с курсором? С помощью каких команд T-SQL реализуются основные операции?

7 Взаимодействие СУБД MS SQL Server с системой программирования MS Visual Studio.NET

Цель: изучить основы создания приложения Windows Forms в среде Visual Studio .Net для ввода, изменения и удаления данных в базе данных.

Теоретические положения

Вопросы, изучаемые в данной теме, описаны в [1], а также на https://metanit.com/sharp/adonetcore/. После создания приложения Windows Forms нужно установить соединение с БД. Для этого выбирается пункт меню Tools / Connect to Database. Появляется окно Add Connection, где в пункте Server name задается имя сервера, а в пункте Select or enter a database name – имя БД из выпадающего списка. Для проверки правильности подключения к БД можно нажать кнопку Test Connection в левом нижнем углу окна Add Connection.

Далее в Solution Explorer из контекстного меню WindowsFormApplication для создания набора данных выбирается пункт Add → New Item → Data → Dataset. Нажать кнопку «Далее» на странице «Сохранение подключения в файле конфигурации приложения». Затем развернуть узел Tables в Server Explorer и выбрать таблицы в окне источников данных и перетащить их на форму. Для подчиненных таблиц в качестве источника данных выбирается не сама вторая таблица, а связь (Binding Source) между таблицами.

Источник данных нужно открыть в конструкторе набора данных для добавления или изменения объектов, составляющих набор данных.

Для импорта необходимых пространств имен служит инструкция using System.Data.SqlClient.

Строка подключения состоит из пар ключ=значение, которые отделяются друг от друга точкой с запятой. Вот некоторые частные разновидности параметров, которые можно указать в строке подключения:

– Data Source (или Server) – указывает имя сервера SQL Server, к которому будет установлено подключение. Это может быть название локального сервера, например, ./SQLEXPRESS (если установлен MS SQL Server Express), localhost (при подключении к базе данных на том же компьютере, на котором выполняется код), либо сетевой адрес;

– Initial Catalog (или Database) – указывает имя базы данных, с которой будет установлено соединение;

– Integrated Security – определяет, будет ли использоваться аутентификация Windows для подключения. Значение True указывает на использование аутентификации Windows, в то время как значение False указывает на использование аутентификации SQL Server, требующей указания имени пользователя и пароля;

– User ID – имя пользователя базы данных, используемое для аутентификации, если используется аутентификация SQL Server;

– Password – Пароль пользователя базы данных, используемый для аутентификации, если используется аутентификация SQL Server.

Определение строки подключения зависит от режима аутентификации.

В случае, если используется аутентификация Windows, то указывается параметр Integrated Security=True. Здесь применяется доверительное подключение (trusted connection), где не требуются логин и пароль (например, при подключении к локальному серверу SQL Server, который запущен на том же компьютере). В качестве альтернативного названия параметра может использоваться Trusted_Connection=True. Trusted_Connection может принимать значения true, false, yes, no и sspi (обычно эквивалентно значению True и использует аутентификацию Windows, если она доступна). По умолчанию значение false.

Если значение Integrated Security=True, то для аутентификации будет использоваться текущая учетная запись Windows. Подходит для подключения к локальному серверу. Строка подключения в общем виде может выглядеть следующим образом:

"Data Source=SERVER_NAME;Initial Catalog=DATABASE_NAME;Integrated Security=True";

В случае, если используется аутентификация SQL Server, требуется указать отдельно имя пользователя и пароль:

"Data Source=SERVER_NAME;Initial Catalog=DATABASE_NAME;User ID=USERNAME;Password=PASSWORD";Integrated Security=False";

Строка подключения может содержать и другие дополнительные параметры, специфичные для конкретного провайдера данных или базы данных:

– Encrypt – указывает, следует ли использовать шифрование соединения (True – использование шифрования, а False – на отключение);

– TrustServerCertificate – определяет, следует ли доверять сертификату сервера при шифрованном соединении (принимает значения True или False);

– Connection Timeout – определяет время (в секундах), в течение которого приложение будет ожидать установления соединения с базой данных, прежде чем сгенерировать исключение.

Чтобы добавить логику обновления в приложение, следует:

– дважды щелкнуть кнопку «Сохранить» на BindingNavigator, чтобы открыть редактор кода для обработчика событий bindingNavigatorSaveItem_Click;

– заменить код в обработчике событий на вызов методов Update связанных адаптеров таблиц TableAdapter [1].

Задание

Необходимо создать приложение Windows Forms в среде Visual Studio .Net для ввода, изменения и удаления данных в таблицах базы данных.

Содержание отчета: тема и цель работы; прокомментированный код на C#; скриншоты приложения.

Контрольные вопросы

1 Как установить соединение с БД?
2 Как выполняется удаление и обновление записей в Windows-формах?

8 Работа с базами данных с использованием технологии ADO.NET

Цель: изучить основы технологии ADO.NET и компоненты доступа к данным в MS Visual Studio .Net.

Теоретические положения

ADO.NET – это набор средств Microsoft .NET Framework, позволяющих приложению легко управлять и взаимодействовать со своим файловым или серверным хранилищем данных.

Для работы с различными СУБД подключаются (using) соответствующие пространства имен: для MS Access – System.Data.OleDb; для MS SQL – System.Data.SqlClient.

В объектной модели ADO.NET можно выделить несколько уровней.

Уровень данных. Это, по сути дела, базовый уровень, на котором располагаются сами данные (например, таблицы БД MS SQL Server, представления, хранимые процедуры и функции). На данном уровне обеспечивается физическое хранение информации на магнитных носителях и манипуляция с данными на уровне исходных таблиц (выборка, сортировка, добавление, удаление, обновление и т. п.).

Уровень бизнес-логики. Это набор объектов, определяющих, с какой БД предстоит установить связь и какие действия необходимо будет выполнить с содержащейся в ней информацией. Для установления связи с БД используется объект DataConnection. Для хранения команд, выполняющих какие-либо действия над данными, используется объект DataAdapter. И, наконец, если выполнялся процесс выборки информации из БД, для хранения результатов выборки исполь-

зуется объект DataSet. Объект DataSet представляет собой набор данных, «вырезанных» из таблиц основного хранилища, который может быть передан любой программе-клиенту, способной отобразить эту информацию пользователю, либо выполнить какие-либо манипуляции с полученными данными.

Каждый объект DataAdapter обеспечивает обмен данными между одной таблицей источника данных (базы данных) и одним объектом DataTable в наборе данных DataSet. Если DataSet сдержит несколько таблиц (объектов DataTable), то необходимо иметь и несколько адаптеров данных.

Используя объект DataAdapter, можно читать, добавлять, модифицировать и удалять записи в источнике данных. Чтобы определить, как каждая из этих операций должна произойти, DataAdapter поддерживает следующие свойства: SelectCommand – описание команды, которая обеспечивает выборку нужной информации из базы данных; InsertCommand – описание команды, которая обеспечивает добавление записей в базу данных; UpdateCommand – описание команды, которая обеспечивает обновление записей в базе данных; DeleteCommand – описание команды, которая обеспечивает удаление записей из базы данных. Каждая из команд реализована в виде SQL-запроса или хранимой процедуры. Эти свойства являются самостоятельными объектами и относятся к элементам класса OleDbCommand или SqlCommand. Данные объекты поддерживают свойство CommandText, содержащее описание SQL-запроса или хранимой процедуры.

Уровень приложения. Это набор объектов, позволяющих хранить и отображать данные на компьютере конечного пользователя. Для хранения информации используется объект DataSet, а для отображения данных имеется набор элементов управления (DataGrid, TextBox, ComboBox, Label и т. д.).

Обмен данными между приложениями и уровнем бизнес-логики происходит с использованием формата XML, а средой передачи данных служат либо локальная сеть (Интранет), либо глобальная сеть (Интернет).

В ADO.NET для манипуляции с данными могут использоваться команды, реализованные в виде SQL-запросов или хранимых процедур (DataCommand). Например, если нужно получить некий набор информации БД, формируется команда SELECT или вызывается хранимая процедура по ее имени.

При получении набора строк из БД может использоваться следующая последовательность действий:

– открыть соединение (connection) с БД;

– вызвать на исполнение метод или команду, указав ей в качестве параметра текст SQL-запроса или имя хранимой процедуры;

– закрыть соединение с базой данных.

Связь с базой данных остается активной только на достаточно короткий срок – на период выполнения запроса или хранимой процедуры.

Когда команда вызывается на исполнение, она возвращает либо данные, либо код ошибки. Если в команде содержался SQL-запрос на выборку SELECT, то команда может вернуть набор данных. Можно выбрать из БД только определенные строки и колонки, используя объект DataReader, который работает достаточно быстро, т. к. использует курсоры read-only, forward-only.

Более подробно применение технологии ADO.NET описано в конспекте лекций, в [1, 4] и на https://metanit.com/sharp/ado.php.

Задание

Необходимо обеспечить вывод на форму приложения Windows Forms результатов работы всех представлений и хранимых процедур, разработанных ранее в создаваемой базе данных, и продемонстрировать срабатывание триггеров.

Содержание отчета: тема и цель работы; прокомментированный код C# выполнения задания.

Контрольные вопросы

1 Как осуществляется взаимодействие с БД через объект DataSet? Как создать набор данных (DataSet) и заполнить его результатами запроса?

2 Для чего предназначены объекты SqlConnection, OleDbConnection, DataAdapter, DataReader? Перечислить методы и свойства этих объектов.

3 Каким образом можно обрабатывать хранимые процедуры с использованием ADO.NET?

9 Работа с базами данных с использованием технологии ASP.NET

Цель: изучение основ использования технологии ASP.NET при работе с БД.

Теоретические положения

Работа с технологией ASP.NET рассматривается в конспекте лекций, в [1, 4] и на https://metanit.com/sharp/mvc.php.

В данной лабораторной работе рассмотрен пример построения файла *.aspx, который отобразит таблицу Inventory базы данных AutoLot с использованием подключенного уровня. Следует запустить Visual Studio и создать новую веб-форму (Web Form), выбрав пункт меню File / New / File.

Далее следует сохранить файл под именем Default.aspx в новом каталоге на жестком диске, чтобы позже его можно было легко найти (например, C:\MyCode\SinglePageModel).

Можно воспользоваться проводником Windows, чтобы создать подкаталог bin внутри папки SinglePageModel. Специально именованный подкаталог bin – это зарегистрированное имя для механизма исполняющей среды ASP.NET. В папку \bin корня веб-сайта можно поместить любые приватные сборки, используемые веб-приложением. Для рассматриваемого примера нужно поместить копию AutoLotDAL.dll в папку C:\MyCode\SinglePageModel\bin.

Далее следует выбрать вкладку Standard (Стандартные) в панели инструментов Visual Studio и перетащить элементы управления Button, Label и GridView на поверхность визуального конструктора страницы (виджет GridView находится на вкладке Data (Данные) панели инструментов) между открывающим и закрывающим элементами form. Можно воспользоваться окном Properties для установки различных визуальных свойств и назначения каждому виджету подходящего имени через атрибут ID.

Теперь следует найти на странице раздел <form>. Каждый веб-элемент управления определен с использованием дескриптора <asp:>. После этого префикса дескриптора указывается имя веб-элемента управления ASP.NET (Label, GridView и Button). Перед закрывающим дескриптором заданного элемента находится серия пар «имя/значение», которые соответствуют настройкам, проведенным в окне Properties:

```
<form id="forml" runat="server">
<div>
<asp:Label ID="lblInfo" runat="server"
Text="Click on the Button to Fill the Grid">
</asp:Label>
<br />
<br />
<asp: GridView ID="carsGndView" runat="server">
</asp: GndView>
<br />
<asp:Button ID="btnFillData" runat="server" Text="Fill Grid" />
</div>
</form>
```

Веб-элементы управления – это объекты, обрабатываемые на веб-сервере, который автоматически возвращает их HTML-представление в исходящем HTTP-запросе. Помимо этого главного преимущества, веб-элементы управления ASP.NET имитируют модель программирования настольных приложений, при которой имена свойств, методов и событий обычно совпадают с их аналогами из Windows Forms/WPF.

Добавление логики доступа к данным. Следует обработать событие Click для типа Button, используя окно Properties среды Visual Studio (через значок с изображением молнии) либо раскрывающиеся списки в верхней части окна визуального конструктора. После этого определение Button будет обновлено добавлением атрибута OnClick, которому присвоено имя обработчика события Click:

```
<asp:Button ID="btnFillData" runat="server"
Text="Fill Grid" OnClick="btnFillData_Click"/>
```

Теперь в блоке <script> серверной стороны внутри файла *.aspx необходимо реализовать обработчик события Click. Для этого нужно добавить следующий код, следя за тем, чтобы входные параметры в точности соответствовали цели делегата System. EventHandler:

```
<script runat="server">
protected void btnFillData_Click(object sender, EventArgs args)
</script>
```

Следующий шаг связан с наполнением GridView посредством функциональности сборки AutoLoDAL.dll. Для этого с помощью директивы <%@Import%> нужно указать, что будет использоваться пространство имен AutoLotConnectedLayer. Вдобавок нужно информировать исполняющую среду ASP.NET о том, что эта однофайловая страница ссылается на сборку AutoLoDAL.dll, через директиву <%@Assembly%>. Ниже показана оставшаяся существенная логика страницы из файла Default.aspx (при необходимости изменить строку соединения):

```
<%@ Page Language="C#" %>
<%@ Import Namespace = "AutoLotConnectedLayer" %>
<%@ Assembly Name ="AutoLotDAL" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtmll/DTD/xhtmll-transitional.dtd">
<script runat="server">
void btnFillData_Click(object sender, EventArgs args)
{
InventoryDAL dal = new InventoryDAL();
dal.OpenConnection(@"Data Source=(local)\SQLEXPRESS;" +
"Initial Catalog=AutoLot; Integrated Secunty=True") ;
carsGndView. DataSource = dal .GetAllInventory () ;
carsGndView. DataBind () ; dal.CloseConnection ();
}
</script>
<html xmlns="http://www.w3.огд/1999/xhtml" >
</html>
```

Далее нужно сохранить файл *.aspx. Если щелкнуть правой кнопкой мыши в любом месте визуального конструктора *.aspx и выбрать в контекстном меню пункт View in Browser (Просмотр в браузере), то это запустит веб-сервер разработки ASP.NET, который, в свою очередь, развернет страницу.

После обработки страницы сначала будут видны элементы управления Label и Button. Однако после щелчка на кнопке произойдет обратная отправка на веб-сервер и веб-элементы управления визуализируют соответствующие HTML-дескрипторы. В открывшемся диалоговом окне нужно выбрать подходящий

шаблон и после щелчка на кнопке ОК просмотреть сгенерированное объявление элемента управления.

```
<asp:GridView ID="carsGridView" runat="server" BackColor="White"
BorderColor="#E7E7FF" BorderStyle="None" BorderWidth="lpx" CellPad-
ding="3 GridLines="Honzontal">
<AlternatingRowStyle BackColor="#F7F7F7" />
<FooterStyle BackColor="#B5C7DE" ForeColor="#4A3C8C" />
<HeaderStyle BackColor~"#4A3C8C" Font-Bold="True" ForeColor="#F7F7F7"
/> <PagerStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" Horizonta-
lAlign="Right" /> <RowStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" />
<SelectedRowStyle BackColor="#738A9C" Font-Bold=„True" Fore-
Color="#F7F7F7" /> <SortedAscendingCellStyle BackColor="#F4F4FD" />
<SortedAscendingHeaderStyle BackColor="#5A4C9D" />
<SortedDescendingCellStyle BackColor="#D8D8F0" />
<SortedDescendingHeaderStyle BackColor="#3E3277" />
</asp:GridView>
```

Для использования базы данных необходимо иметь надежный и безопасный способ подключения. На платформе .NET этот способ реализуется с помощью пространства имен System.Data и одной строки подключения.

Задание

Необходимо создать web-форму для доступа ко всем таблицам базы данных.

Содержание отчета: тема и цель работы; код C# выполнения задания.

Контрольные вопросы

1 Охарактеризовать сущность процесса создания Web-приложения на основе ASP.NET Web Forms с использованием технологии Entity Framework в среде разработки Visual Studio.

2 Какова структура ASP.NET приложения?

3 Как подключиться к базе данных в ASP.NET?

4 Как добавить новую запись в таблицу базы данных с помощью ASP.NET?

5 Как использовать ORM (Object-Relational Mapping) в ASP.NET для взаимодействия с базой данных?

Список литературы

1 **Агальцов, В. П.** Базы данных [Электронный ресурс]: учебник: в 2 т. Т. 2: Распределенные и удаленные базы данных / В. П. Агальцов. – Москва: ФОРУМ; ИНФРА-М, 2021. – 271 с. – Режим доступа: https://znanium.com/catalog/document?id=377105. – Дата доступа: 05.09.2023.

2 **Бен-Ган, И.** Microsoft SQL Server 2012. Создание запросов: учебный курс Microsoft: пер. с англ. / И. Бен-Ган, Д. Сарка, Р. Талмейдж. – Москва: Русская редакция, 2015. – 720 с. : ил.

3 **Бондарь, А. Г.** Microsoft SQL Server 2014 / А. Г. Бондарь. – Санкт-Петербург: БХВ-Петербург, 2015. – 592 с.: ил.

4 **Рихтер, Дж.** CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# [Электронный ресурс]: учебник / Дж. Рихтер. – 4-е изд. – Санкт-Петербург: Питер, 2020. – 896 с. – Режим доступа: https://znanium.com/catalog/product/1733758. – Дата доступа: 05.09.2023.

5 **Тарасов, С. В.** СУБД для программиста: базы данных изнутри [Электронный ресурс] / С. В. Тарасов. – Москва: СОЛОН-Пресс, 2020. – 320 с. – Режим доступа: https://znanium.com/catalog/product/1227737. – Дата доступа: 05.09.2023.