Лабораторная работа № 3

Создание Windows-приложения для работы с распределённой базой данных

Цель работы

Получить навыки создания Windows-приложения с графическим интерфейсом для работы с распределённой базой данных. Научиться создавать документы Word и Excel с данными из базы данных.

Создание проекта Windows Forms на C#

Для того чтобы создать новый проект C#, нужно зайти в меню **Файл** → **Создать** → **Проект**. В списке типов проектов выбрать язык C# и выбрать пункт «**Приложение Windows Forms**» (см. рисунок 1).

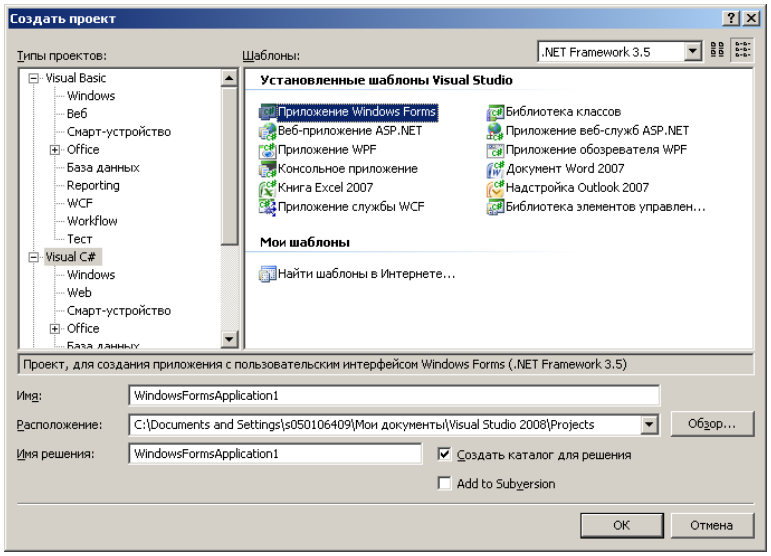

Рисунок 1 – Создание проекта

Работа с СУБД PostgreSQL

Драйвер Npgsql

Для работы с СУБД PostgreSQL потребуется драйвер Npgsql (скачать можно по ссылке [1]).

Библиотеки **Mono.Security.dll**, **Npgsql.dll** нужно поместить в папку с проектом. В проекте следует указать ссылки на эти библиотеки. Это делается так: в обозревателе решений нужно нажать правой кнопкой мыши на меню **Ссылки**, выбрать пункт **Добавить ссылку**. На вкладке **Обзор** нужно указать путь для библиотек (см. рисунок 2).

В коде формы (в обозревателе решений щёлкнуть правой кнопкой мыши на имя формы (по умолчанию **Form1.cs**) и выбрать **Перейти к коду**) нужно прописать строку

```
using Npgsql;
```

Подключение к базе данных

Для того, чтобы создать новое подключение к базе данных, нужно прописать строку подключения в любом месте программы перед обращением к базе данных. Строка подключения выглядит следующим образом:

```
NpgsqlConnection conn = new NpgsqlConnection (
  "server=<Сервер>; database=<база данных>;
   user Id=<Имя пользователя>; password=<Пароль>");
```

Вывод выборки данных в DataGridView

Для вывода выборки данных из базы данных используется элемент **DataGridView**. Этот элемент можно найти на панели элементов (по умолчанию вызывается **Ctrl+Alt+X**) в разделе **Данные** (см. рисунок 3). Нужно поместить этот элемент, а также кнопку (**Button** из раздела стандартные панели элементов) на форму (см. рисунок 4).

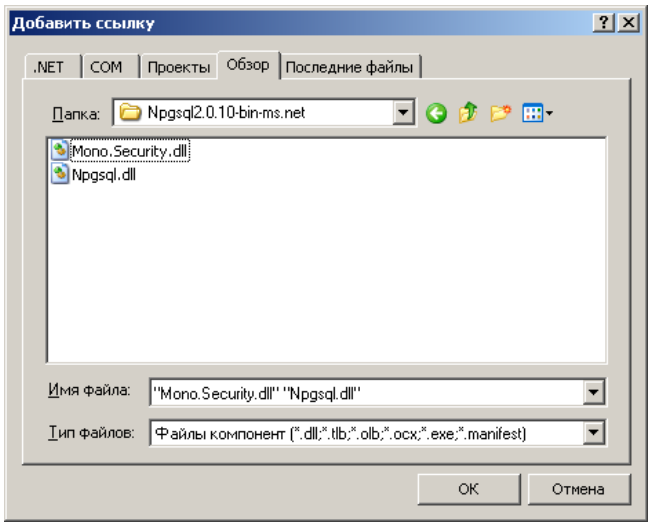

Рисунок 2 – Добавление библиотек в проект

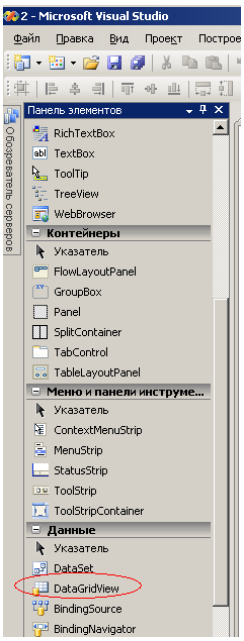

Рисунок 3 – Панель элементов

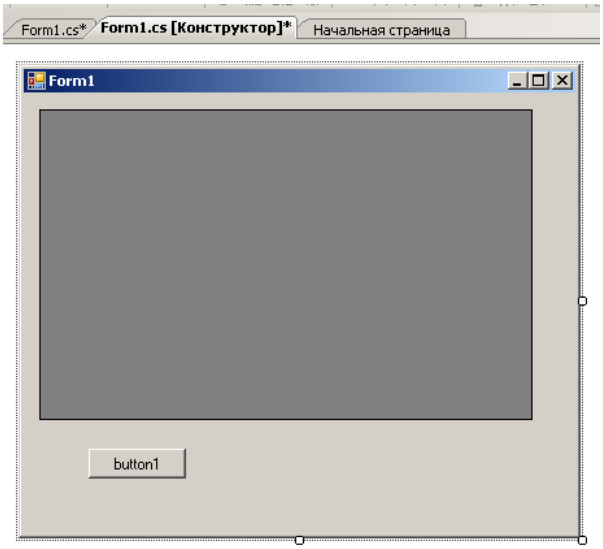

Рисунок 4 – Добавление DataGridView и кнопки на форму

Чтобы создать обработчик для кнопки, нужно сделать двойной щелчок по кнопке, после чего автоматически произойдёт переход к коду формы:

```
private void button1_Click(object sender, EventArgs e)
{
}
```

По нажатию кнопки должен происходить вывод данных в **DataGridView**. Для этого в обработчик нажатия кнопки следует прописать следующий код:

```
// Создадим новый набор данных
DataSet datasetmain = new DataSet();
// Открываем подключение
conn.Open();
// Очищаем набор данных
datasetmain.Clear();
// Sql-запрос
NpgsqlCommand command = new NpgsqlCommand("Текст запроса", conn);
```

```
// Новый адаптер нужен для заполнения набора данных
NpgsqlDataAdapter da = new NpgsqlDataAdapter(command);
// Заполняем набор данных данными, которые вернул запрос
da.Fill(datasetmain, "table1");
// Связываем элемент DataGridView1 с набором данных
dataGridView1.DataSource = datasetmain;
dataGridView1.DataMember = "table1";
// Закрываем подключение
conn.Close();
```

Добавление данных

Для начала нужно создать новую форму (в обозревателе решений щёлкнуть правой кнопкой по имени проекта и выбрать пункт **Добавить → Форма Windows**). В появившемся окне выбрать «**Форма Windows Forms**» (см. рисунок 5). Здесь же можно указать имя новой формы. Далее следует добавить на созданную форму новые элементы: текстовое поле (**TextBox**), выпадающий список (**ComboBox**) и кнопку, по нажатию на которую будут добавляться данные в базу (см. рисунок 6).

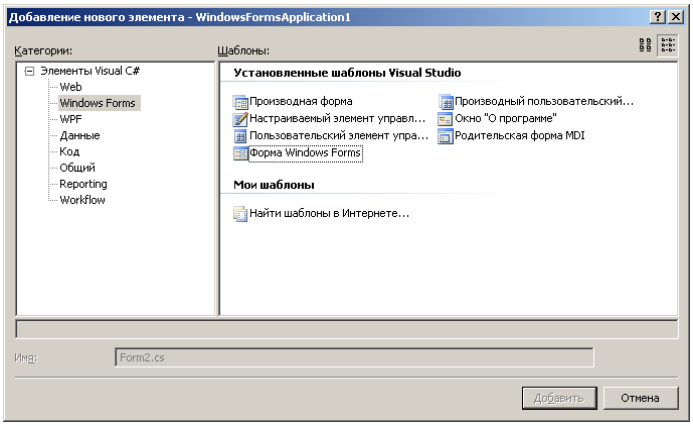

Рисунок 5 – Создание новой формы

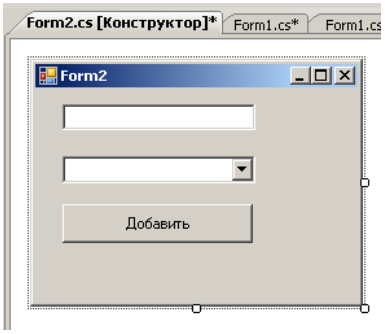

Рисунок 6 – Форма для добавления данных

Теперь необходимо сделать вывод данных из базы в выпадающий список. Для этого в обработчик загрузки формы нужно поместить следующий код:

```
DataSet set1 = new DataSet();
set1.Clear();
// Открываем подключение
conn.Open();
// Sql-запрос для формирования списка
NpgsqlCommand command1 =
  new NpgsqlCommand("select name ,id from <Имя таблицы>", conn);
// Адаптер
NpgsqlDataAdapter da1 = new NpgsqlDataAdapter(command1);
// Заполнение набора данных
da1.Fill(set1, "set2");
// Связывание списка и набора данных
comboBox1.DataSource = set1.Tables["set2"];
// Данные, которые будут отображены в списке
comboBox1.DisplayMember = "name";
// Значения, соотвествующие отображенным элементам списка
comboBox1.ValueMember = "id";
```

А в обработчик нажатия кнопки **Добавить** – следующий:

```
NpgsqlCommand command = new NpgsqlCommand(
  "insert into <Имя таблицы>(text,id_tovar) values(:text,:id_tovar)",
  Form2.conn);
// Данные из поля и списка добавляем в параметры запроса
command.Parameters.Add(new NpgsqlParameter("text", DbType.String));
command.Parameters[0].Value = Convert.ToInt32(richTextBox1.Text);
command.Parameters.Add(new NpgsqlParameter("id_tovar", DbType.Int32));
// Значение параметра=id, соответствующему выбранному значению из списка
command.Parameters[1].Value = Convert.ToInt32(comboBox1.SelectedValue);
```

```
// Выполнить команду
command.ExecuteNonQuery();
```

Редактирование данных

Редактирование производится аналогично, но для этого необходимо знать номер (**id**) редактируемой строки. Получить его можно так:

```
// Номер выбранной строки
int row = dataGridView1.CurrentRow.Index;
// Столбец, в который выводится id
// Например, id выводится в нулевой столбец
int column = 0;
// Идентификатор выбранной строки таблицы продажи
int id1 = Convert.ToInt32(dataGridView1[column, row].Value);
```

Удаление данных

Пример кода:

```
conn.Open();
// Запрос на удаление
NpgsqlCommand command = new NpgsqlCommand("Запрос на удаление", conn);
// Параметр запроса
command.Parameters.Add(new NpgsqlParameter("param", DbType.Int32));
int row = dataGridView1.CurrentRow.Index;
int column = 0;
// Параметр – идентификатор выбранной строки
command.Parameters[0].Value =
  Convert.ToInt32(dataGridView1[column, row].Value);
// Выполняем запрос
command.ExecuteNonQuery();
```

Распределённый запрос на удаление (добавление, редактирование) данных

Пример запроса выглядит следующим образом:

```
// Подключение к базе данных
select public.dblink_connect('<название соединения>',
  'dbname=<название базы данных> user=<имя пользователя> password=<пароль>');
// Выполнение запроса
select public.dblink_exec('<название соединения>',
  'delete from <Имя схемы.Имя таблицы> where id = :<Параметр>');
// Отключение соединения
select public.dblink_disconnect('<название соединения>');
```

Работа с Microsoft Word

Для работы с Microsoft Word и Excel необходимо добавить ссылки на соответствующие библиотеки (см. рисунок 7). В коде программы нужно написать следующее:

```
using Word = Microsoft.Office.Interop.Word;
using Excel = Microsoft.Office.Interop.Excel;
```

С точки зрения приложения, все объекты Word имеют иерархическую структуру. Объект **Application** – это COM-сервер и оболочка для других объектов. Он может содержать один или несколько объектов **Document**. Объекты **Document** могут содержать такие объекты, как **Paragraph**, **Table**, **Range**, **Bookmark**, **Chapter**, **Word**, **Sentence**, **Sections**, **Headers**, **Footers**,… Точнее говоря, объект **Application** может содержать коллекцию **Documents** – ссылок на объекты типа **Document**, а каждый объект типа **Document** – коллекцию **Paragraphs** или ссылок на объекты типа **Paragraph** и т.д.

Работа с документами, параграфами, символами, закладками и т.д. выполняется посредством использования свойств и методов этих объектов. Объекты при создании решений принято определять глобально для того, чтобы обеспечить доступ к ним из любой функции проекта.

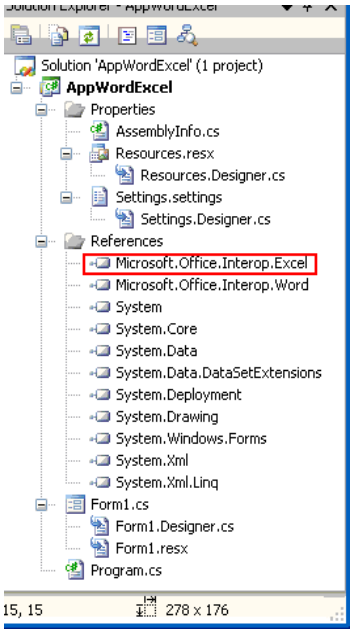

Рисунок 7 – Добавление библиотек для работы с Word и Excel

Создание документа

Глобальное определение основного объекта **`Word.Application`**:

```
private Word.Application wordapp;
```

Создание новых объектов **`Word`**, параграф и документ:

```
// Создаём объект Word. Это равносильно запуску Word
wordapp = new Word.Application();
// Делаем его видимым
wordapp.Visible = true;
// Создаём объект параграф
Word.Paragraph wordparagraph;
Word.Document doc = new Word.Document();
// Создание документа
doc = app.Documents.Add (ref object Template, ref object NewTemplate,
                         ref object DocumentType, ref object Visible);
```

Параметры метода **`Add`**:

- **`Template`** – имя шаблона, по которому создаётся новый документ. Если значение не указано, то используется шаблон **`Normal.dot`**;
- **`NewTemplate`** – при **`true`** новый документ открывается как шаблон. Значение по умолчанию – **`false`**;
- **`DocumentType`** – тип документа, может принимать одно из следующих значений констант типа **`Word.WdNewDocumentType`**:
  - **`wdNewBlankDocument`** – документ **`Word`** (по умолчанию);
  - **`wdNewEmailMessage`** – электронное сообщение;
  - **`wdNewWebPage`** – Web-страница;
  - **`wdNewXMLDocument`** – XML-документ;
- **`Visible`** – видимость документа. При **`true`** (по умолчанию) документ отображается.

В качестве параметра **`Template`** методу **`Add`** можно определить имя существующего документа или полное имя шаблона. Во втором случае осуществляется привязка к пути, по которому установлены приложения Microsoft Office, тот же эффект достигается, если используется параметр по умолчанию (напомним, **`Type`** – класс декларации типов, **`Type.Missing`** – отсутствие значения или значение по умолчанию. Кроме того, некоторые методы принимают необязательные параметры, которые не поддерживаются в C#, и в этом случае также используется **`Type.Missing`**, кото-

рый является ссылочным типом – *reference type*). Использование в качестве **Template** имени существующего файла полезно, когда потребуется добавлять какие-либо данные в уже существующий документ или бланк.

Вывод текста в документ

Вывод текста выполняется не просто в параграф, а в диапазон параграфа – объект **Range**. Объект **Range** – это непрерывная область документа, включающая позицию начального и конечного символов. Для пустого параграфа или если начальная и конечная позиции диапазона совпадают, **Range** представляет курсор ввода.

В документе можно определить диапазон, вызвав метод **Range** с передачей ему начального и конечного значений позиций символов (при определении позиции номера символов считаются от 0 и включают все символы, в том числе и непечатные). Выделенный диапазон можно "подсветить", используя метод **Select()**:

```
Object begin = 0;
Object end = 5;
Word.Range wordrange = worddocument.Range(ref begin, ref end);
wordrange.Select();
```

Создание таблицы

Информация об объектах **Table** хранится в виде ссылок на таблицы документа в свойстве **Tables**. Набор ссылок **Tables** доступен из объектов **Document**, **Selection** и **Range** и даже из объекта **Word.Table**. Это значит, что и создавать таблицы можно с использованием любого из этих объектов. Для создания таблиц используется метод **Add**:

```
Add
(
 Word.Range Range, // Объект Range - место формирования таблицы

 int NumRows,      // Число строк

 int NumColumns,   // Число столбцов

 /*  Определяет, изменяет ли Word автоматически размеры ячеек в таблицах, чтобы они соответ-
     ствовали содержанию ячеек. Может быть одна из Word.WdDefaultTableBehavior констант:
     wdWord8TableBehavior (нет) или wdWord9TableBehavior (да). По умолчанию -
     wdWord8TableBehavior.
 */
 ref object DefaultTableBehavior,

 /*  Автоподбор ширины столбцов, одна из следующих Word.WdAutoFitBehavior констант: wdAu-
     toFitContent - по содержимому, wdAutoFitFixed - фиксированная или wdAutoFitWindow по
     ширине окна. Если DefaultTableBehavior установлен в wdWord8TableBehavior, этот параметр
     игнорируется.
 */
 ref object AutoFitBehavior
)
```

Вывод информации в ячейки таблиц

Таблица или объект **Word.Table** состоит из ячеек – объектов **Cell**, ссылки на которые хранятся в наборе **Cells** данной таблицы. Ячейки таблицы считаются упорядоченными по координатам **X** и **Y**, нумерация с 1. Следующие строки кода ссылаются на ячейку, расположенную в первой строке и во втором столбце:

```
Word.Range wordcellrange = worddocument.Tables[1].Cell(1, 2).Range;
```

Для вывода текста в ячейку достаточно свойству **Text** объекта **Word.Range** присвоить значение типа **string**.

```
wordcellrange.Text = "Строка для вывода";
```

Сохранение документов

Документы Word можно сохранить программно и обычным для Word способом. В любом случае, перед выходом из Word необходимо вызвать метод **Quit**. Если свойство **DisplayAlerts** объекта **Word.Application** имеет значение **true**, Word предложит сохранить данные в том случае, когда после старта в документ были внесены какие-либо изменения.

Для сохранения документа можно использовать методы **Save()**, **SaveAs()** и **SaveAs2000**. Метод **Save()** не имеет параметров и при его вызове будет отображено диалоговое окно Word "**Сохранение документа**". Метод **SaveAs()** имеет множество параметров, большинство из которых можно не указывать, а использовать как параметры по умолчанию.

Параметры метода **SaveAs()**:

```
SaveAs
(
 ref fileName,          // Имя файла

 /*  Формат сохраняемого файла, одна из Word.WdSaveFormat констант wdFormatDocument,
     wdFormatWebArchive, wdFormatUnicodeText, wdFormatTextLineBreaks, wdFormatRTF,
     wdFormatText, wdFormatTemplate, wdFormatHTML, wdFormatFilteredHTML,
     wdFormatEncodedText, wdFormatDOSText, wdFormatDOSTextLineBreaks
 */
 ref fileFormat,

 /*  При true блокируется содержимое поля Заметки, находящегося на вкладке Документ в диало-
     говом окне Свойства меню Файл.
 */
 ref lockComments,

 ref password,          // Пароль доступа к документу при открытии

 /*  При true имя сохраняемого файла добавляется в список недавно открытых файлов в меню
     Файл.
 */
 ref addToRecentFiles,

 ref writePassword,     //Пароль для внесения изменений в документ

 /*  Если true - при открытии документа будет отображаться диалоговое окно с рекомендацией
     открывать документ только для чтения.
 */
 ref readOnlyRecommended,

 // При true – TrueType-шрифты сохраняются вместе с документом
 ref embedTrueTypeFonts,

 // При true сохраняет только родную графику (Windows)
 ref saveNativePictureFormat,

 // При true сохраняет только данные, введённые пользователем в формы
 ref saveFormsData,

 /*  Если в документе используется attached mailer (программа доставки электронной почты ад-
     ресату), то должно быть true для того, чтобы документ был сохранён.
 */
 ref saveAsAOCELetter,

 /*  Кодовая страница (набор символов) для документов, сохранённых как кодируемые текстовые
     файлы. Значение по умолчанию - системная кодовая страница. Задаётся как Mi-
     crosoft.Office.Core.MsoEncoding.msoEncodingUSASCII.
 */
 ref encoding,

 /*  Если документ сохраняется как текстовый файл, то при true разрешается вставка разрывов
     строк.
 */
 ref insertLineBreaks,
```

```
 /*  Если документ сохраняется как текстовый файл, то при true Word заменяет некоторые сим-
     волы текстом. Например, символ авторского права заменяет на (c).
 */
 ref allowSubstitutions,

 /*  Если документ сохраняется как текстовый файл, то одна из Word.WdLineEndingType констант
     (wdCRLF, wdLSPS, wdCROnly, wdLFCR, wdLFOnly), определяет, какие символы (перевод стро-
     ки, возврат каретки) используются для отделения строк друг от друга.
 */
 ref lineEnding,

 /*  При true Word добавляет к файлу символы управления вывода, чтобы сохранить двунаправ-
     ленное размещение текста в оригинале документа.
 */
 ref addBiDiMarks
);
```

Подробнее о работе с Microsoft Word в C# описано в [2].

Работа с Microsoft Excel

Сам сервер – объект **Application** или приложение Excel – может содержать одну книгу или более, ссылки на которые содержит свойство **Workbooks**. Книги – объекты **Workbook** – могут содержать одну страницу или более, ссылки на которые содержит свойство **Worksheets**, или несколько диаграмм – свойство **Charts**. Страницы (листы) – **Worksheet** – содержат объекты ячейки или группы ячеек, ссылки на которые становятся доступными через объект **Range**. Ниже в иерархии располагаются строки, столбцы,… Аналогично, для объекта **Chart** – серии линий, легенды,…

```
// Новое приложение Excel
app = new Excel.ApplicationClass();
```

Создание рабочих книг, листов и таблиц

Вторым в иерархии объектов **Excel.Application** является объект **Workbook**. Информация об объектах **Workbook** хранится в виде ссылок на открытые рабочие книги в свойстве **Workbooks**. Книга в приложение может быть добавлена только через добавление ссылки в коллекцию **Workbooks**, а ссылка на открытую книгу может быть получена различным образом (по имени, номеру, как ссылка на активную книгу):

```
// Новая рабочая книга
Excel.Workbook wb = app.Workbooks.Add(
                    Excel.XlWBATemplate.xlWBATWorksheet);
// Новый рабочий лист
Excel.Worksheet ws = (Excel.Worksheet) wb.ActiveSheet;
```

Аналогичного определения для групп ячеек и ячейки задать нельзя, т.к. отдельно данные объекты как самостоятельные в C# отсутствуют, а есть понятие области выделенных ячеек, которая может включать несколько ячеек (одну или более), с которыми можно выполнять действия. Поэтому для ячеек, с которыми выполняется действие, можно ввести следующее определение:

```
private Excel.Range excelcells;
```

Для выделения используется метод **get_Range**, который позволяет выделить группу ячеек через задание угловых ячеек диапазона, и есть возможность обратиться непосредственно к свойствам **Rows** и **Cells** – в любом случае выделенным будет диапазон ячеек.

Чтобы нарисовать таблицу в Excel, надо научиться рисовать рамки вокруг выбранной ячейки или объединённой группы ячеек.

Шаги рисования рамки будут следующие:

- выбрать ячейку или группу ячеек на листе документа;
- объединить ячейки;

- определить цвет линий обводки. Цвет может быть выбран как один из 56 цветов цветовой палитры Excel, и поэтому он задаётся через цветовой индекс (например, **`excelcells.Borders.ColorIndex = 3`** означает красный цвет). Некоторые значения **`ColorIndex`**:
  - 1 – белый;
  - 2 – черный;
  - 3 – красный;
  - 4 – зелёный;
  - 6 – жёлтый;
  - 41 – синий, и т.д.;
- выбрать стиль линии (**`Excel.XlLineStyle.xlContinuous`**). Стиль линии может быть одним из следующих: **`xlContinuous`**, **`xlDash`**, **`xlDashDot`**, **`xlDashDotot`**, **`xlDot`**, **`xlDouble`**, **`xlSlantDashDot`**, **`xlLineStyleNone`**;
- задать толщину линии (**`Excel.XlBorderWeight.lHairline`**). Толщина линии может быть одной из следующих: **`lHairline`**, **`xlMedium`**, **`xlThick`**, **`xlThin`**.

Можно рисовать линии по любой границе ячейки и не по границе ячейки, для чего необходимо задать расположение линии – вместо **`excelcells.Borders`** задать **`excelcells.Borders[направление]`**, где **`направление`** может быть одним из следующих:

- **`Excel.XlBordersIndex.xlDiagonalDown`**;
- **`Excel.XlBordersIndex.xlDiagonalxlDiagonalUp`**;
- **`Excel.XlBordersIndex.xlDiagonalUp`**;
- **`Excel.XlBordersIndex.xlEdgeBottom`**;
- **`Excel.XlBordersIndex.xlEdgeLeft`**;
- **`Excel.XlBordersIndex.xlEdgeRight`**;
- **`Excel.XlBordersIndex.xlEdgeTop`**;
- **`Excel.XlBordersIndex.xlInsideHorizontal`**;
- **`Excel.XlBordersIndex.xlInsideVertical`**.

Сохранение документов

Для сохранения документов можно использовать методы **`Save()`** и **`SaveAs()`** объекта **`Excel.Workbook`**. Метод **`Save()`** сохраняет рабочую книгу в папке "Мои документы" с именами, присваиваемыми документу по умолчанию ("Книга1.xls", "Книга2.xls",…) или в текущей директории с именем, под которым документ уже был сохранён.

Метод **`SaveAs()`** позволяет сохранить документ с указанием имени, формата файла, пароля, режима доступа и т.д. Данный метод, как и метод **`Save()`**, присваивает свойству **`Saved`** значение **`true`**. Метод **`SaveAs()`** имеет следующий синтаксис:

```
Workbook_object.SaveAs
(
 Filename,                // Имя сохраняемого файла
 FileFormat,              // Формат сохраняемого файла
 Password,                // Пароль доступа к файлу (до 15 символов)
 WriteResPassword,        // Пароль для доступа на запись
 ReadOnlyRecommended,     // При true режим только для чтения
 CreateBackup,            // При true создать резервную копию файла
 AccessMode,              // Режим доступа к рабочей книге
 ConflictResolution,      // Способ разрешения конфликтов
 // При true сохранённый документ добавляется в список ранее открытых файлов
 AddToMru,
 TextCodePage,            // Кодовая страница
 TextVisualLayout,        // Направление размещения текста
 Local                    // Идентификатор ExcelApplication
)
```

Подробнее о работе с Microsoft Excel в C# описано в [3].

Задание

Для созданной в лабораторной работе № 1 базы данных с оптимальным размещением таблиц по двум узлам написать Windows-приложение с графическим пользовательским интерфейсом, работающее с этой базой. Приложение должно уметь корректно обрабатывать вводимые данные, делать выборку данных из таблиц, вставлять, удалять и изменять данные в таблицах, расположенных в различных базах данных, сохраняя целостность распределённой базы данных.

Результат некоторых запросов приложение должно представить в виде отчётов, представляющих собой документы Word и/или Excel.

Примечание: При реализации приложения рекомендуется использовать платформу .NET.

Требования к оформлению отчёта

Отчёт по лабораторной работе должен включать в себя:

- титульный лист;
- краткое описание предметной области, для которой разработана база данных;
- ER-диаграммы баз данных для каждого узла;
- описание разработанного программного средства;
- программный код, написанный непосредственно студентами;
- тестирование программы;
- формируемые программой документы Word и Excel.

Отчёт не должен содержать орфографических, пунктуационных и смысловых ошибок. Все его разделы должны быть выдержаны в едином стиле оформления.

Критерии оценивания качества работы

1. Количество корректно обрабатываемых полей таблиц:
   1 – приложение обрабатывает данные в ***10*** полях таблиц, при этом ***6*** из этих полей являются внешними ключами для других таблиц, а ***3*** из них – внешними ключами для таблиц, размещённых в другой базе;
   0 – приложение обрабатывает данные в ***6*** полях таблиц, при этом ***3*** из этих полей служат внешними ключами для других таблиц, а ***1*** из них является внешним ключом для таблицы, размещённой в другой базе;
   л.р. не принимается – иначе.
   Примечание: Сложность выполнения лабораторной работы рассчитывается в количестве полей, которые обрабатываются, суммарно со всех использованных таблиц. Например, если необходима обработка 10 полей, можно взять одну таблицу, включающую в себя 5 полей, одну таблицу, включающую в себя 3 поля, и одну таблицу, включающую в себя 2 поля. В сумме это даст 10 полей. Из них 3 должны быть внешними ключами в локальной базе данных (ссылаются из student51 в student51), а 3 других – внешними ключами ко второй локальной базе данных (ссылаются из student51 в student52).
2. Выгрузка данных в Word и Excel:
   1 – приложение формирует хотя бы 1 отчёт с данными из базы как документ Word и хотя бы 1 отчёт – как документ Excel;
   0 – приложение формирует как документ Word или Excel хотя бы 1 отчёт с данными из базы;
   л.р. не принимается – иначе.
3. Содержание отчёта:
   1 – отчёт удовлетворяет всем требованиям;
   0 – отчёт удовлетворяет не всем требованиям.
4. Обработка ошибок:
   1 – все возможные ошибки и нестандартные ситуации (например, неудачная попытка открытия файла) обрабатываются программой, которая выдаёт соответствующее сообщение;
   0 – не все возможные ошибки обрабатываются программой.

5. Применение принципов структурного программирования:
   1 – все повторяющиеся либо логически целостные фрагменты программы выделены в качестве функций; работа каждой функции полностью определяется её параметрами (т.е. не используются глобальные переменные, все данные, нужные функции для работы, передаются ей через параметры); программа позволяет без перекомпиляции изменять все параметры, от которых зависит её работа; в тексте программы отсутствуют числовые константы (все необходимые константы объявляются как поименованные);
   0 – иначе (не выполняется что-либо из перечисленного).
6. Наличие комментариев в исходных кодах:
   1 – комментариев достаточно для документирования исходных кодов;
   0 – комментариев недостаточно.
7. Глубина понимания материала лабораторной работы каждым членом бригады:
   1 – быстрые и правильные ответы на все вопросы;
   0 – не на все вопросы ответы правильные и быстрые;
   л.р. не принимается – на половину вопросов ответы неправильные.

Список литературы

1. Драйвер Npgsql [Электронный ресурс]. – Режим доступа: http://pgfoundry.org/frs/?group_id=1000140.
2. Работа с MS Word в C# [Электронный ресурс]. – Режим доступа: http://wladm.narod.ru/C_Sharp/comword.html.
3. Работа с MS Excel в C# [Электронный ресурс]. – Режим доступа: http://wladm.narod.ru/C_Sharp/comexcel.html.