ПРЕДСТАВЛЕНИЕ ПОДМНОЖЕСТВА ДАННЫХ В ИНТЕРФЕЙСЕ ПОЛЬЗОВАТЕЛЯ

Разработка интерфейса приложений требует разнообразных средств для сортировки, фильтрации, поиска, изменения и навигации. Класс ***DataView*** реализует отбор табличных данных из объектов ***DataTable*** по заданным критериям с возможностью привязки отобранных данных к элементам интерфейса. Класс *DataView* являются настраиваемым представлением *DataTable*. *DataView* не сохраняет данные, а представляет связанное представление соответствующего *DataTable*. Изменения данных *DataView* влияют на содержание *DataTable*. Изменения данных *DataTable* влияют на все связанные с ним *DataView*.

Отображение таблиц и полей: объект DataView

Объект ***DataView*** обладает свойствами, которые позволяют настраивать способ отображения данных из объекта ***DataTable*** следующим образом:

- Изменять порядок сортировки (нисходящий или восходящий) по одному или нескольким полям.
- Использовать выражение для фильтрации записей, которое указывает критерии отображения записей на основе значений полей.
- Применять фильтр состояния записей, который указывает критерии отображения записей на основе состояния записи.

Объект ***DataView*** – динамическое представление данных таблицы-источника, то есть все изменения в таблице-источнике немедленно отображаются в объекте ***DataView***. Каждый объект ***DataView*** является представлением только одной таблицы и не может быть объединением нескольких таблиц. Не смотря на то, что объект ***DataView*** очень похож на таблицу, он не может использоваться как таблица, в нем нельзя исключать поля, которые присутствуют в таблице-источнике, нельзя включать дополнительные поля, например, вычисляемые поля, которых нет в таблице-источнике. У каждой таблицы может быть сколько угодно представлений и для всех этих представлений таблица является источником полей и строк.

Первый способ создания объекта ***DataView*** – использовать свойство ***DefaultView*** непосредственно таблицы-источника.

```
ds.Tables["Salaries"].DefaultView.RowFilter=”Salary > 10000”;
ds.Tables["Emp"].DefaultView.Sort=”Department”;
```

Второй способ - явное создание объекта класса ***DataView*** для конкретной таблицы с последующей настройкой свойств ***RowFilter***, ***Sort*** и ***RowStateFilter*** объекта.

```
DataView dv = new DataView(ds.Tables["Emp"], "", "FIO", DataViewRowState.CurrentRows);
```

Этой кодовой строкой было создано представление для таблицы Emp, в котором нет фильтра, сортировка по полю FIO в возрастающем порядке и отображены текущие записи таблицы.

Объект ***DataView*** поддерживает модель редактирования, аналогичную модели редактирования объекта ***DataTable***, то есть в представлении можно удалять, добавлять, редактировать данные.

Создайте в вашем проекте еще одну форму для изучения объекта ***DataView*** (рисунок 1).

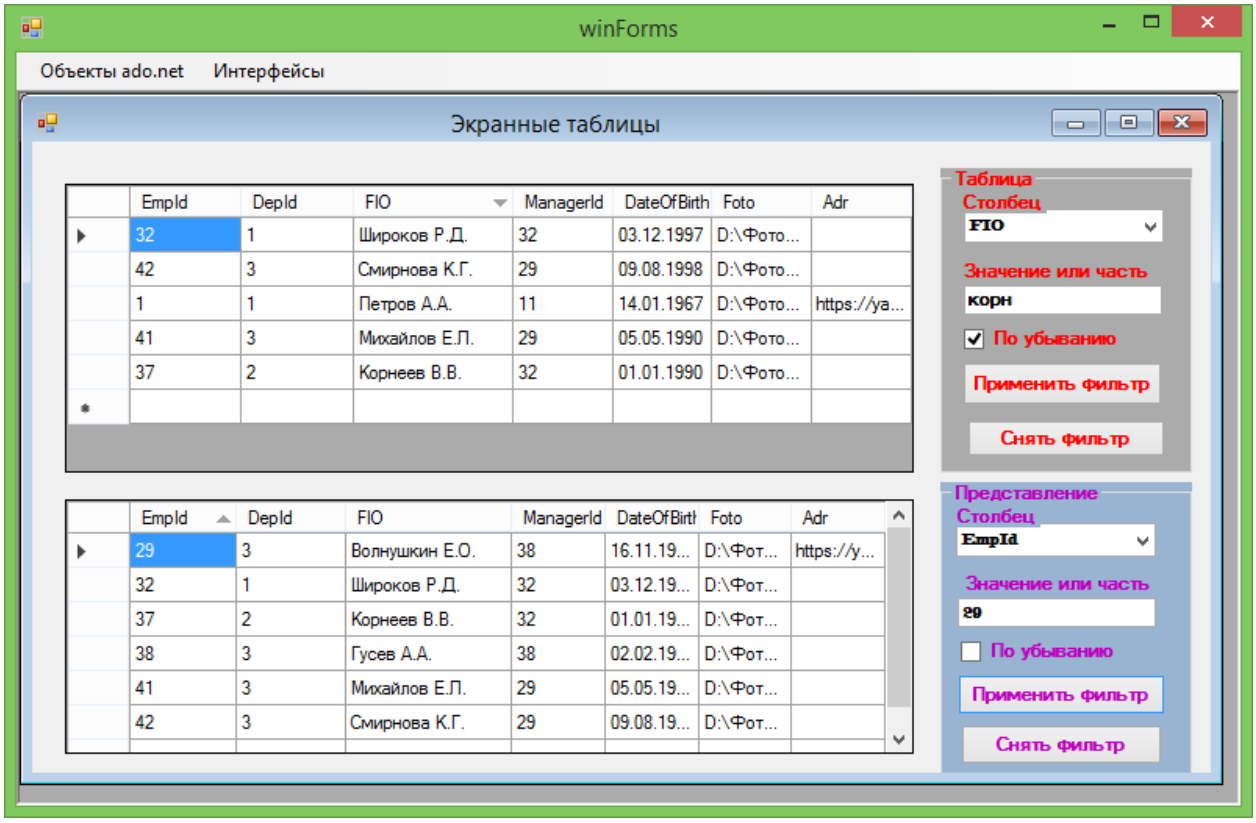

Рисунок 1 - Интерфейс, построенный для изучения объекта DataView

Сформируйте наполнение данными элементов интерфейса при загрузке формы:

```
        private void Form8_Load(object sender, EventArgs e)
        {
cnn = new SqlConnection(ConfigurationManager.ConnectionStrings["Employees"].ConnectionString);
            da = new SqlDataAdapter("select * from Emp", cnn);
            da.Fill(ds, "Emp");
            dataGridView1.DataSource = ds.Tables["Emp"];

            //Создаем представление для таблицы Emp
            dv = new DataView(ds.Tables["Emp"],"","FIO",DataViewRowState.CurrentRows);

            //Чтение списка названий столбцов таблицы в comboBoxы
            foreach (DataColumn col in ds.Tables["Emp"].Columns)
            {
                comboBox1.Items.Add(col.ColumnName);
                comboBox2.Items.Add(col.ColumnName);
            }

            //Установка значений по умолчанию
            comboBox1.SelectedItem = "FIO";
            comboBox2.SelectedItem = "FIO";
            ds.Tables["Emp"].DefaultView.Sort = "FIO";
            dv.Sort = "FIO";

            //Связывание сеток с 2-мя различными представлениями таблицы
            dataGridView1.DataSource = ds.Tables["Emp"].DefaultView;
            dataGridView2.DataSource = dv;
        }
```

```
        //Верхняя кнопка Применить фильтр
        private void button1_Click(object sender, EventArgs e)
        {
            //Указание фильтра
string filter = String.Format("{0}>='{1}'",comboBox1.SelectedItem.ToString(),textBox1.Text);
            ds.Tables["Emp"].DefaultView.RowFilter = filter;
            //Указание сортировки
            string sort = comboBox1.SelectedItem.ToString();
            if (checkBox1.Checked) sort += " DESC";
            ds.Tables["Emp"].DefaultView.Sort = sort;
        }
```

Свойство ***Sort*** предназначено для вывода записей в порядке возрастания (ascending, ASC) или убывания (descending, DESC) по значениям заданного поля. В принципе, элемент DataGridView сам по себе поддерживает сортировку – достаточно просто щелкнуть по заголовку поля. Однако это требует действий от пользователя, тогда как объект DataView может предоставлять данные уже в готовом виде.

Свойство ***RowFilter*** позволяет формировать условие фильтрации записей. Это свойство, задаваемое строкой, применяется в качестве средства фильтрации на базе критериев, определенных значением строки. Ее синтаксис подобен конструкции WHERE стандартного SQL, но относится к данным, уже полученным из базы. ***DataView*** не позволяет фильтровать столбцы, а только строки.

```
//Нижняя кнопка Применить фильтр
private void button3_Click(object sender, EventArgs e)
{
    //Указание фильтра
    string filter = String.Format("{0}>='{1}'", comboBox2.SelectedItem.ToString(),
      textBox2.Text);
    dv.RowFilter = filter;
    //Указание сортировки
    string sort = comboBox2.SelectedItem.ToString();
    if (checkBox2.Checked) sort += " DESC";
    dv.Sort = sort;
}
```

Пример использования объекта DataView

Рассмотрим практический пример построения интерфейса с использованием DataView.

В приведенном ниже примере программного кода используются различные события элемента ComboBox:

- событие ***SelectionChangeCommitted*** - происходит при изменении выделенного элемента пользователем, если это изменение отображается в объекте ComboBox;
- событие ***SelectedIndexChanged*** – происходит при изменении свойства SelectedIndex.

Можно создать обработчики этих событий и обеспечить специальную обработку для ComboBox. Следует учитывать особенности этих событий, связанных с тем как ComboBox настроен и как пользователь изменяет выбранный элемент: изменяется ли индекс программным способом или пользователем.

Программирование ComboBox позволяет решать вопросы синхронизации отображения данных в других элементах формы на основе текущего состояния ComboBox. Обработчики событий для этого элемента можно использовать для загрузки данных в другие элементы формы.

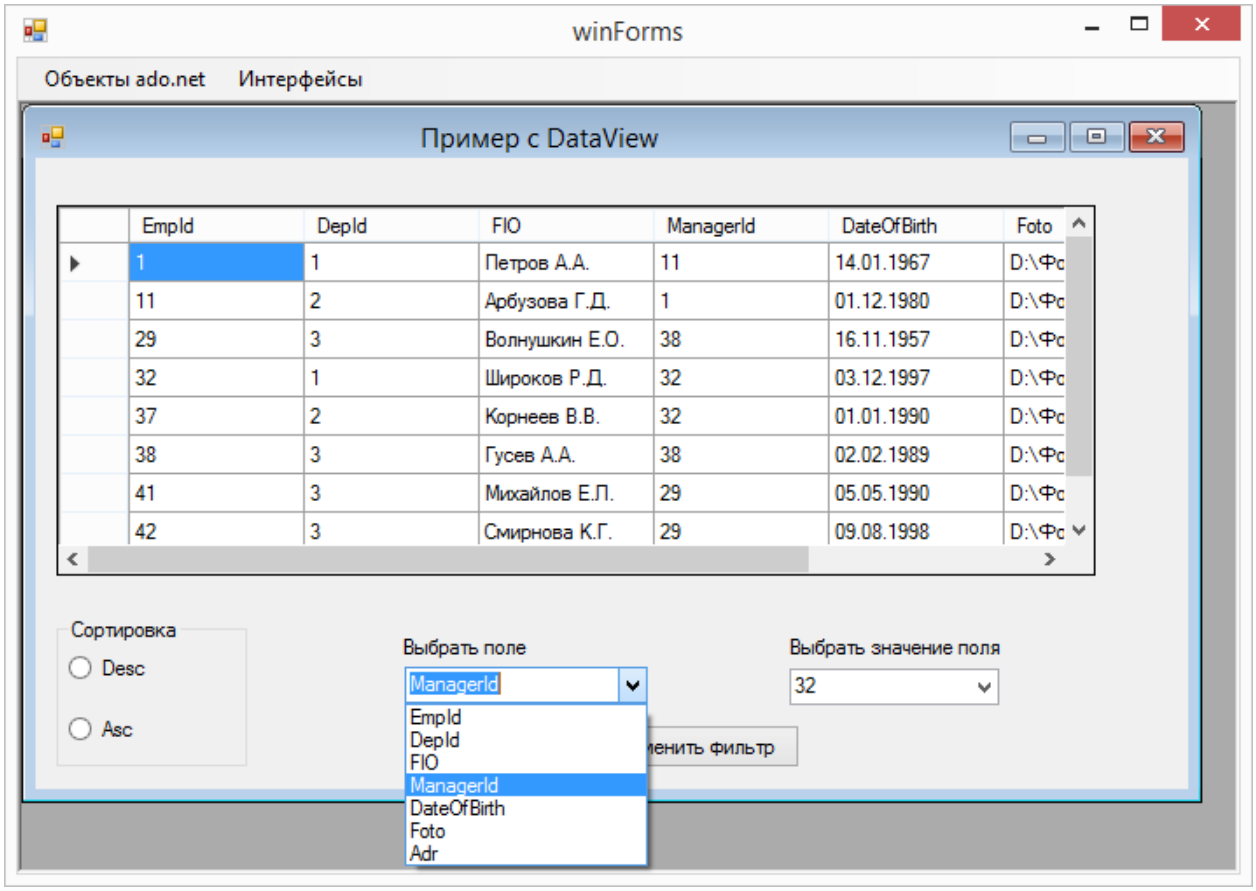

Рисунок 2 – Применение объекта DataView для организации просмотра данных

```
private void Form9_Load(object sender, EventArgs e)
{
    cnn = new SqlConnection(ConfigurationManager.ConnectionStrings
      ["Employees"].ConnectionString);
    da = new SqlDataAdapter("select * from Emp", cnn);
    da.Fill(ds, "Emp");
    dv = new DataView(ds.Tables[0]);
    dataGridView1.DataSource = dv;
    //Список всех полей таблицы Emp
    foreach( DataColumn col in ds.Tables[0].Columns)
    { comboBox1.Items.Add(col.ColumnName); }
    //Начальная настройка полей формы
    comboBox2.DataSource = ds.Tables[0];
    comboBox2.DisplayMember = "Fio";
    comboBox2.ValueMember = "Fio";
    comboBox2.SelectedIndex = 2;
    bild = new SqlCommandBuilder(da);
}
```

```
//Выбор поля таблицы
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    comboBox2.DisplayMember = comboBox1.SelectedItem.ToString();
    comboBox2.ValueMember = comboBox1.SelectedItem.ToString();
}
//Выбор значения
private void comboBox2_SelectionChangeCommitted(object sender, EventArgs e)
{
    dv.RowFilter = String.Format("{0}='{1}'", comboBox1.SelectedItem,
      comboBox2.SelectedValue);
}
```

```
//Сортировка по убыванию
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    dv.Sort = comboBox1.SelectedItem.ToString()+" DESC";
}
```

Задание для самостоятельной работы

Для задания 1 самостоятельно разработайте кнопки по удалению параметров фильтрации и сортировки, т.е. установлению значений по умолчанию. Сформируйте различные варианты строки для свойства *RowFilter*, в том числе, вариант расширенного фильтрования с отношениями.

Для задания 2 самостоятельно напишите программный код сортировки по возрастанию, отмены фильтрации, добавления вычисляемого столбца.