Експериментальні вивчення можливостей сортування, фільтрації, пошуку даних з використанням технології ADO .NET
На прикладі ADO_SortFilterFind

У цьому прикладі продемонструємо всі способи сортування, фільтрації та пошуку даних, які застосовуються в **ADO.NET**. Кожен із методів перевірявся на таблиці з одним мільйоном записів.

Для початку роботи в конструкторі форм **Designer** спроектуємо форму, де розмістимо необхідні елементи керування (рис. 1).

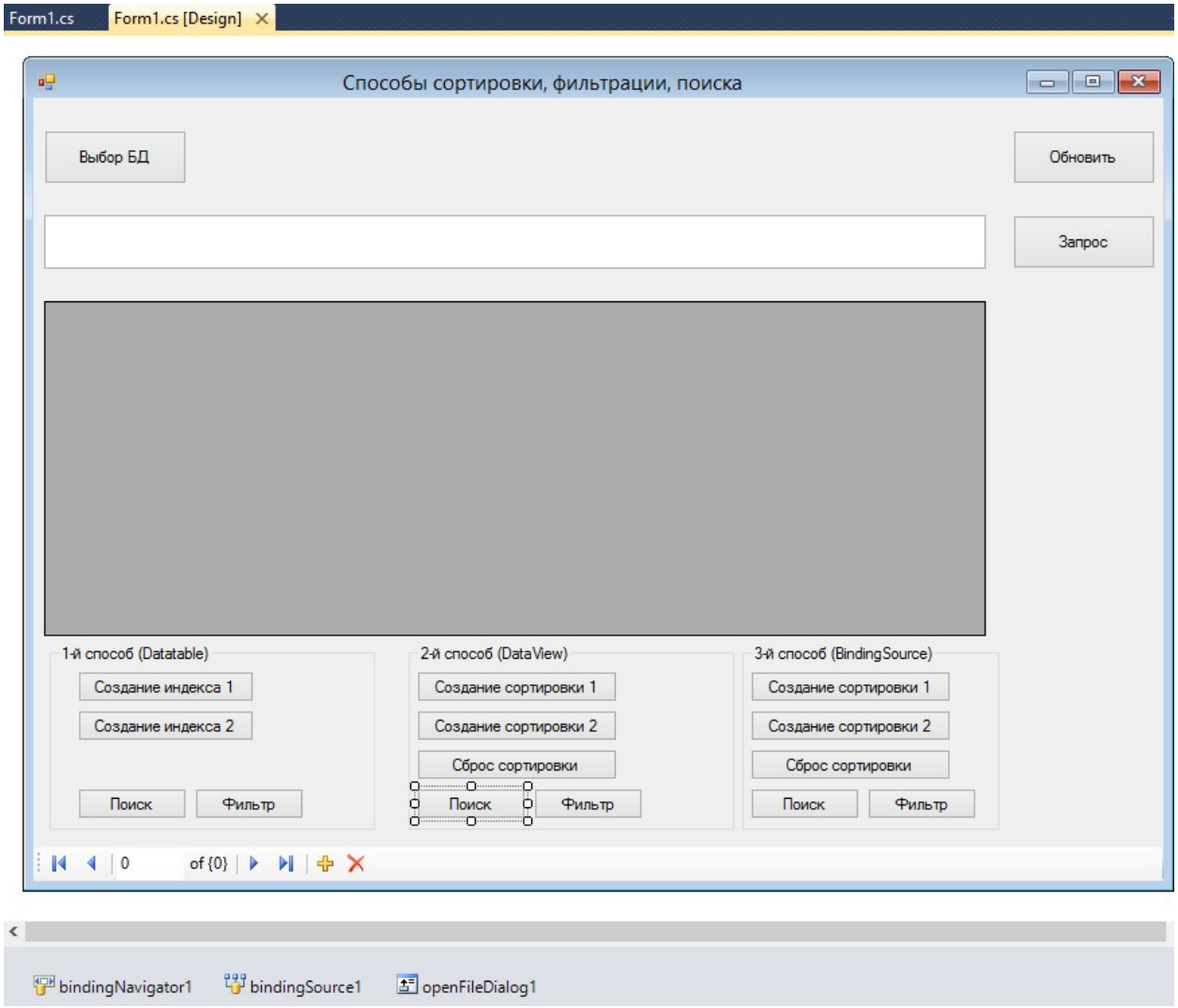

Рис. 1. Конструювання додатку ADO_SortFilterFind

Наведемо повний текст програмного коду програми ADO_SortFilterFind:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
namespace Ado22
{
    public partial class Form1 : Form
    {
        //Объект Connection с драйвером OleDb
        OleDbConnection conn;
        OleDbDataAdapter da;
        DataSet ds;

        public Form1()
        {
            InitializeComponent();
            conn = null;
            da = null;
            ds = new DataSet();
        }

        //Открытие БД
        private void button1_Click(object sender, EventArgs e)
        {
            //Открытие файла *.mdb
            openFileDialog1.FileName = "otl.mdb";
            openFileDialog1.Filter = "*.mdb|*.mdb";
            if (openFileDialog1.ShowDialog() != DialogResult.OK) return;

            //Формирование объекта Connection
            try
            {
                string source = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
                                 openFileDialog1.FileName + ";Mode=ReadWrite";
                if (conn != null) conn.Dispose();
                conn = new OleDbConnection(source);
                conn.Open();
                textBox1.Text = "Select * From otl_tab";
            }

            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }
        //Получение результата SQL-запроса
        private void button2_Click(object sender, EventArgs e)
        {
            try
            {
                //Создание объекта Recordset
                if (da != null) da.Dispose();
                da = new OleDbDataAdapter(textBox1.Text, conn);

                //Сохранение данных через объект OleDbCommandBuilder
                OleDbCommandBuilder bulder = new OleDbCommandBuilder(da);
                //Символы в которые будут заключены поля и таблицы БД
                //при формировании запросов
                bulder.QuotePrefix = "[";
                bulder.QuoteSuffix = "]";

                //Сформированные запросы на Insert, Delete, Update
                string str = bulder.GetInsertCommand().CommandText+"\n\n";
                str = str + bulder.GetDeleteCommand().CommandText + "\n\n";
                str = str + bulder.GetUpdateCommand().CommandText;
                MessageBox.Show(str);

                if (ds.Tables.Count>0) ds.Tables.RemoveAt(0);
```

```
        da.Fill(ds);

        bindingSource1.DataSource = ds.Tables[0];
        dataGridView1.DataSource = bindingSource1;
        bindingNavigator1.BindingSource = bindingSource1;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

//Обновление данных
private void button3_Click(object sender, EventArgs e)
{
    try
    {
        da.Update(ds.Tables[0]);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

//Работа с индексами, сортировкой и фильтрацией с помощью объекта DataTable
////////////////////////////////////////////////////////////////////////////

//Primarykey - создание индекса (уникальное поле таблицы)
//Find - поиск точного значения по индексированному полю
//Select - фильтрация данных, работает как с индексом так и без него (с индексом
//фильтрация данных работает мгновенно)
//Данный способ является самым быстрым и оптимальным - он позволяет хранить все
//индексы и производить быстрый и эффективный поиск

//Создание индекса 1
private void button4_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str;
      //Создание индексного файла по 1-му полю таблицы
      //При создании следующих индексов-все предыдущие сохраняются.
      //При создании индекса PrimaryKey значения в столбце должны быть уникальны
        ds.Tables[0].PrimaryKey = new DataColumn[] { ds.Tables[0].Columns[0] };

        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds /
               1000.0).ToString() + " секунд";
        MessageBox.Show(str);
    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

//Создание индекса 2
```

```
private void button5_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str;

        //Создание индексного файла по 2-му полю таблицы
        ds.Tables[0].PrimaryKey = new DataColumn[] { ds.Tables[0].Columns[1] };
        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds /
              1000.0).ToString() + " секунд";
        MessageBox.Show(str);

    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

//Поиск с помощью метода Find в таблице с последующим переходом на значение
private void button7_Click(object sender, EventArgs e)
{

    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str="";

 //Поиск на точное совпадение - делается по индексному полю (Primarykey) таблицы
        //При существующем индексе делается мгновенно
        DataRow zap = ds.Tables[0].Rows.Find(textBox1.Text);
        DataView datav = ds.Tables[0].DefaultView;

        if (zap != null)
        {
            int pos = datav.Table.Rows.IndexOf(zap);
            dataGridView1.CurrentCell = dataGridView1.Rows[pos].Cells[0];

        }
        else
            str = "Значение не найдено\n";

        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = str + "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds
                    / 1000.0).ToString() + " секунд";
        MessageBox.Show(str);

    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
```

```
    }
}

//Фильтрация данных с помощью метода Select в таблице
//Фильтр работает с индексными файлами и без них.
//Фильтр с учетом индекса делается мгновенно!!!
private void button8_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str="";

        DataRow [] datar = ds.Tables[0].Select(textBox1.Text);
        DataTable table = ds.Tables[0];

        DataTable newTable = new DataTable();
        DataRow vr;
        int i;

        for (i = 0; i < table.Columns.Count;i++)
    newTable.Columns.Add(table.Columns[i].ColumnName,table.Columns[i].DataType);

        foreach (DataRow row in datar)
        {

            vr = newTable.NewRow();

            for (i = 0; i < table.Columns.Count; i++)
                vr[i]=row[i];

            newTable.Rows.Add(vr);
        }

        //dataGridView1.DataSource = null;
        dataGridView1.DataSource = newTable;

        //dataGridView  1. DataSource = ;

        /*
        DataView datav = ds.Tables[0].DefaultView;
        if (datar.Length > 0)
        {
            int pos = datav.Table.Rows.IndexOf(datar[0]);
            dataGridView1.CurrentCell = dataGridView1.Rows[pos].Cells[0];
        }
        else
            str = "Нет значений, которые удовлетворяют фильтру\n";
        */

        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = str + "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds
                    / 1000.0).ToString() + " секунд";
        MessageBox.Show(str);
    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
```

```
}

//Работа сортировкой, поиском и фильтрацией с помощью объекта DataView
//Sort - сортировка по выбранному полю или совокупности полей(влияет на
// представление данных и в текущий момент времени активна только одна)
//Find - точный поиск, который осуществляется по выбранной сортировке (работает
//быстро)
//RowFilter - фильтрация данных (работает независимо от сортировки) - всегда
 //работает медленно

//Сортировка и фильтрация с помощью DataView предназначена для перестраивания
//представления данных, генерации отчетов и т.д.
//Для вычислительных операций эффективнее всего использование методов объекта
//DataTable

//Сортировка по 1-му полю
private void button9_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str;

        //Сортировка по первому полю
        DataView datav = ds.Tables[0].DefaultView;
        datav.Sort = ds.Tables[0].Columns[0].ColumnName;

        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds /
              1000.0).ToString() + " секунд";
        MessageBox.Show(str);
    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

//Сортировка по 2-му полю
private void button10_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str;

        //Сортировка по второму полю
        DataView datav = ds.Tables[0].DefaultView;
        datav.Sort = ds.Tables[0].Columns[1].ColumnName;

        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds /
               1000.0).ToString() + " секунд";
        MessageBox.Show(str);
```

```
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

//Сброс сортировки
private void button11_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str;
        //Сброс сортировки
        DataView datav = ds.Tables[0].DefaultView;
        datav.Sort = "";

        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds /
               1000.0).ToString() + " секунд";
        MessageBox.Show(str);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

//Поиск с помощью метода Find по отсортированному полю с использованием DataView
private void button12_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str="";

        //Поиск с помощью метода Find по отсортированному полю
        DataView datav = ds.Tables[0].DefaultView;
        int pos = datav.Find(textBox1.Text);
        if (pos >= 0)
            dataGridView1.CurrentCell = dataGridView1.Rows[pos].Cells[0];
        else
            str = "Значение не найдено\n";
        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = str + "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds
                    / 1000.0).ToString() + " секунд";
        MessageBox.Show(str);
    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

```
//Фильтрация данных с помощью метода RowFilter - всегда выполняется медленно
private void button13_Click(object sender, EventArgs e)
{

    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str = "";

     //Фильтрация данных с помощью метода RowFilter - всегда выполняется медленно
        DataView datav = ds.Tables[0].DefaultView;
        datav.RowFilter = textBox1.Text;
        if (datav.Count < 1)
            str = "Нет значений, которые удовлетворяют фильтру\n";

        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = str + "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds
        / 1000.0).ToString() + " секунд";
        MessageBox.Show(str);
    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

//Работа сортировкой, поиском и фильтрацией с помощью объекта BindingSource
//////////////////////////////////////////////////////////////////////

//Сортировка по 1-му полю
private void button14_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str;

        //Сортировка по первому полю
        bindingSource1.Sort = ds.Tables[0].Columns[0].ColumnName;

        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds /
               1000.0).ToString() + " секунд";
        MessageBox.Show(str);
    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

//Сортировка по 2-му полю
```

```
private void button15_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str;

        //Сортировка по второму полю
        bindingSource1.Sort = ds.Tables[0].Columns[1].ColumnName;

        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds /
               1000.0).ToString() + " секунд";
        MessageBox.Show(str);
    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

//Сброс сортировки
private void button17_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str;
        //Сброс сортировки
        bindingSource1.Sort = "";

        dt2 = DateTime.Now;
        tt = dt2 - dt1;

        str = "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds /
                1000.0).ToString() + " секунд";
        MessageBox.Show(str);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
//Поиск с помощью метода Find  без учета сортировки (осуществляется
//последовательный поиск - всегда медленно)
private void button16_Click(object sender, EventArgs e)
{
    try
    {
        DateTime dt1, dt2;
        TimeSpan tt;
        dt1 = DateTime.Now;
        ////////////////////////////////////////
        string str = "";
        //Поиск с помощью метода Find  без учета сортировки (осуществляется
```

```
            //последовательный поиск - всегда медленно)
            int pos = bindingSource1.Find("id",textBox1.Text);
            if (pos >= 0)
                dataGridView1.CurrentCell = dataGridView1.Rows[pos].Cells[0];
            else
                str = "Значение не найдено\n";

            dt2 = DateTime.Now;
            tt = dt2 - dt1;

            str = str + "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds
                  / 1000.0).ToString() + " секунд";
            MessageBox.Show(str);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    //Фильтрация данных с помощью метода Filter - всегда выполняется медленно
    private void button18_Click(object sender, EventArgs e)
    {
        try
        {
            DateTime dt1, dt2;
            TimeSpan tt;
            dt1 = DateTime.Now;
            ////////////////////////////////////////
            string str = "";

            //Фильтрация данных с помощью метода Filter - всегда выполняется медленно
            bindingSource1.Filter = textBox1.Text;
            DataView datav = ds.Tables[0].DefaultView;
            if (datav.Count < 1)
                str = "Нет значений, которые удовлетворяют фильтру\n";

            dt2 = DateTime.Now;
            tt = dt2 - dt1;

            str = str + "Запрос выполнен в течение " + (tt.Seconds + tt.Milliseconds
            / 1000.0).ToString() + " секунд";
            MessageBox.Show(str);
        }

        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

}
}
```

Існує три способи роботи з сортуванням, фільтрацією та пошуком даних:

1. Використовуючи методи класу **DataTable**.
2. Використовуючи методи класу представлення **DataView**.
3. Використовуючи методи зв’язувача **BindingSource**.

При першому способі користувачеві необхідно створити індекси для полів, де відбуватиметься пошук або фільтрація даних. Можна створювати індекси, які складаються з кількох полів. Поле, за яким створюється індекс,

має бути унікальним (властивість **PrimaryKey**). Як **PrimaryKey** необхідно вказати масив колонок (**DataColumn**).

Приклад створення індексу першого поля таблиці:

```
ds.Tables[0].PrimaryKey = new DataColumn[] { ds.Tables[0].Columns[0] };
```

Важливим є те, що всі індекси таблиці зберігаються, але створюються при початковому присвоєння. Так, індекси створюються лише один раз, а потім просто перемикаються.

Пошук здійснюється методом **Find** властивості таблиці **Rows** та повертає рядок типу **DataRow**. Якщо пошук невдалий – значення **null**. Під час пошуку вказується значення вибраного ключового поля. Метод **Find** шукає точне значення за активним ключем (властивість **PrimaryKey**).

Приклад пошуку:

```
DataRow zap = ds.Tables[0].Rows.Find(500000);
```

Для фільтрації даних використовується функція **Select**, яка повертає масив рядків **DataRow[]**, що задовольняють заданій умові. Фільтрування зі створеними індексними файлами працює миттєво. При цьому не має значення, який активний індекс обрано. Фільтр може працювати і без індексів, але це буде неефективно і займе набагато більше часу.

Приклад фільтрації даних:

```
DataRow [] datar = ds.Tables[0].Select("id>500000 and id<700000");
```

Висновок

Для рішень різноманітних обчислювальних завдань цей спосіб є найефективнішим. Користувач один раз створює необхідні індекси та робить пошук та фільтрацію з максимальною швидкістю.

Другий і третій способи працюють з представленням (об’єкт **DataView**), яке має відображатися в елементах керування, наприклад, **DataGridView**. Це зручно для організації звітів, де потрібно візуалізувати (представити у наочному вигляді) дані.

Об’єкт **DataView** містить такі властивості та методи:

1. **Sort** – властивість, яка задає сортування і перебудовує об’єкт представлення (сортування може бути задане по кількох полях). У даний момент активне тільки одне сортування.

Приклад сортування:

```
DataView datav = ds.Tables[0].DefaultView;
datav.Sort = ds.Tables[0].Columns[0].ColumnName;
```

2. **Find** – метод точного пошуку за ключем сортування (Sort) – може містити кілька полів для пошуку. **Без встановлення сортування Find не**

працюватиме! При успішному пошуку повертається позиція знайденого рядка представлення, інакше – значення -1.

Приклад пошуку:

```
DataView datav = ds.Tables[0].DefaultView;
int pos = datav.Find(textBox1.Text);
```

3. **RowFilter** – метод фільтрації даних. На великих обсягах даних цей метод працює дуже довго, незважаючи на встановлену властивість **Sort**. Не виявлено особливих відмінностей при роботі методу з сортуванням та без нього.

Приклад фільтрації даних:

```
DataView datav = ds.Tables[0].DefaultView;
datav.RowFilter = "id>500000 and id<700000";
```

При роботі з елементом **BindingSource** спостерігаємо аналогічні результати сортування та фільтрації даних. Щодо пошуку, то він робиться послідовно і без урахування сортування. Такий пошук застосовувати недоцільно.

Приклад сортування через **BindingSource**:

```
bindingSource1.Sort = ds.Tables[0].Columns[0].ColumnName;
```

Пиклад фільтрації за допомогою **BindingSource**:

```
bindingSource1.Filter = "id>500000 and id<700000";
```

Приклад пошуку за допомогою **BindingSource**:

```
int pos = bindingSource1.Find("id",500000);
```

Загальний висновок:

Для ефективного пошуку та фільтрації даних, а також створення відразу кількох індексів слід користуватися методами та властивостями класу **DataTable** (**PrimaryKey** – створення та вибір індексу, **Find** – пошук, **Select** – фільтрація). При цьому і фільтр, і пошук працюватимуть миттєво.

Для зміни представлення даних в елементах керування, скажімо для формування звітів, можна використовувати методи класу **DataView** (**Sort**, **Find**, **RowFilter**) або **BindingSource** зв’язувача (**Sort**, **Find**, **Filter**). При цьому буде не висока швидкість на великих обсягах даних, оскільки головна мета сортування та фільтрації перебудувати представлення існуючої таблиці для відображення у відповідному елементі.

На рис. 2. наведемо результат виконання програми ADO_SortFilterFind.

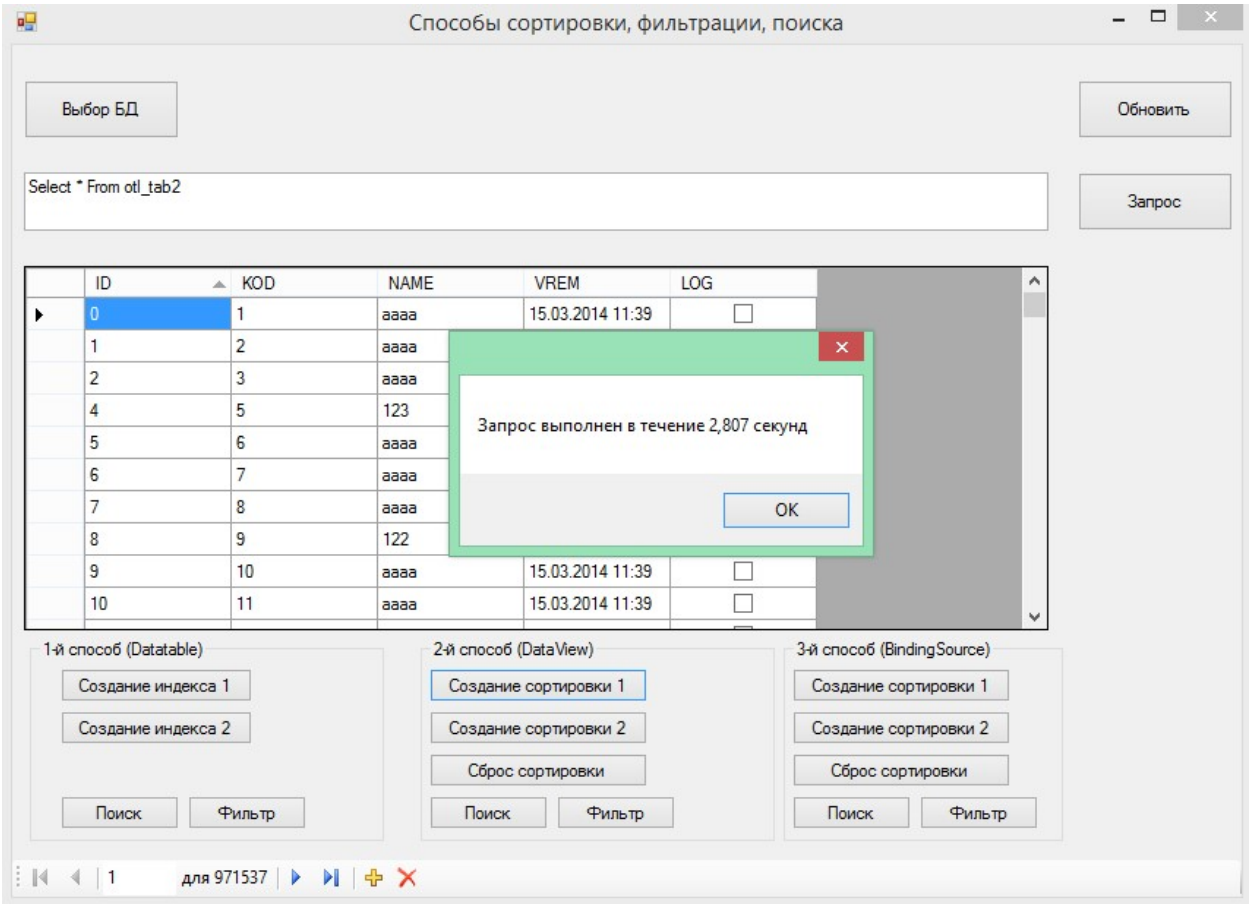

Рис. 2. Результат роботи програми ADO_SortFilterFind