

Лабораторная работа № 2

Нереляционные базы данных

Цель работы: получить навыки работы с нереляционной СУБД *MongoDB* и научиться связывать её со своей средой разработки.

Теоретическая часть

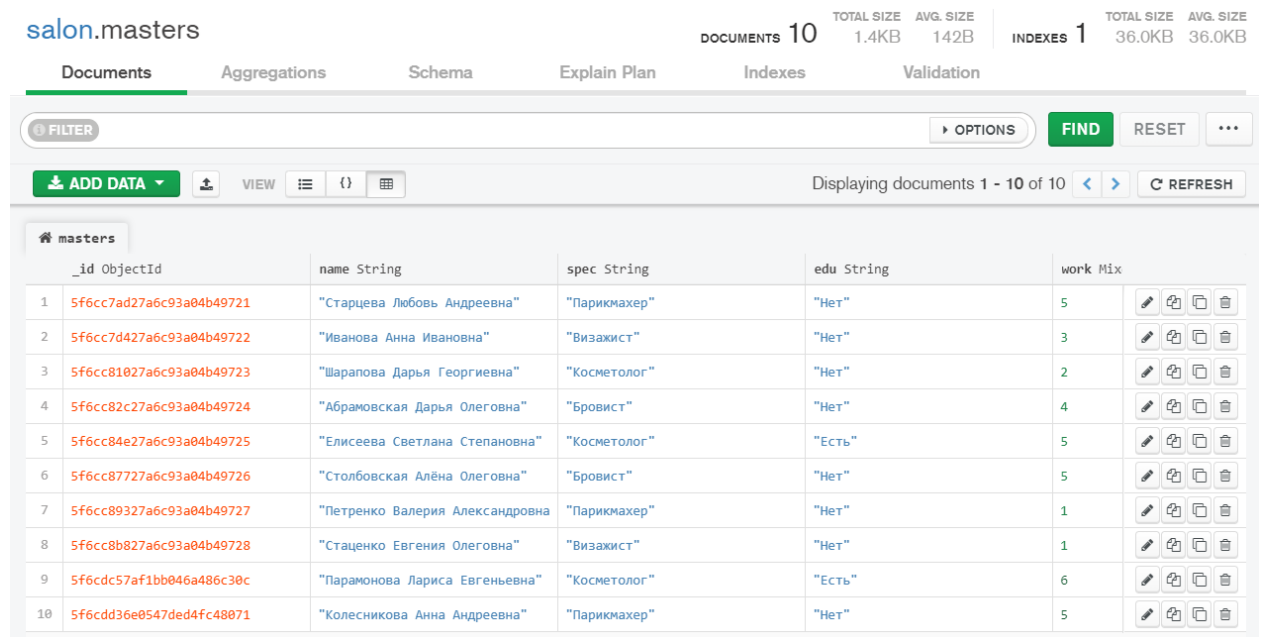
Нереляционная база данных — это база данных, в которой в отличие от большинства традиционных систем баз данных не используется табличная схема строк и столбцов. В этих базах данных применяется модель хранения, оптимизированная под конкретные требования типа хранимых данных. Например, данные могут храниться как простые пары "ключ — значение", документы *JSON* или граф, состоящий из ребер и вершин.

Все эти хранилища данных не используют реляционную модель. Кроме того, они, как правило, поддерживают определенные типы данных. Процесс запроса данных также специфический. Например, хранилища данных временных рядов рассчитаны на запросы к последовательностям данных, упорядоченных по времени, а хранилища данных графов — на анализ взвешенных связей между сущностями. Ни один из форматов не подходит в полной мере при выполнении задач управления данными о транзакциях.

Термин *NoSQL* применяется к хранилищам данных, которые не используют язык запросов *SQL*, а запрашивают данные с помощью других языков и конструкций. На практике *NoSQL* означает "нереляционная база данных", даже несмотря на то, что многие из этих баз данных поддерживают запросы, совместимые с *SQL*. Однако базовая стратегия выполнения запросов *SQL* обычно значительно отличается от применяемой в системе управления реляционной базой данных (реляционная СУБД).

Ход работы

С помощью СУБД *MongoDB* была создана нереляционная БД «*salon*», в которой была создана коллекция «*masters*», содержащая список мастеров салона красоты. Данная коллекция представлена на рисунке 1.



	_id ObjectId	name String	spec String	edu String	work Mix
1	5f6cc7ad27a6c93a04b49721	"Старцева Любовь Андреевна"	"Парикмахер"	"Нет"	5
2	5f6cc7d427a6c93a04b49722	"Иванова Анна Ивановна"	"Визажист"	"Нет"	3
3	5f6cc81027a6c93a04b49723	"Шарапова Дарья Георгиевна"	"Косметолог"	"Нет"	2
4	5f6cc82c27a6c93a04b49724	"Абрамовская Дарья Олеговна"	"Бровист"	"Нет"	4
5	5f6cc84e27a6c93a04b49725	"Елисеева Светлана Степановна"	"Косметолог"	"Есть"	5
6	5f6cc87727a6c93a04b49726	"Столбовская Алёна Олеговна"	"Бровист"	"Нет"	5
7	5f6cc89327a6c93a04b49727	"Петренко Валерия Александровна"	"Парикмахер"	"Нет"	1
8	5f6cc8b827a6c93a04b49728	"Стаценко Евгения Олеговна"	"Визажист"	"Нет"	1
9	5f6cdc57af1bb046a486c30c	"Парамонова Лариса Евгеньевна"	"Косметолог"	"Есть"	6
10	5f6cdd36e0547ded4fc48071	"Колесникова Анна Андреевна"	"Парикмахер"	"Нет"	5

Рисунок 1 – Коллекция «*masters*».

Коллекция содержит ФИО мастера, его специализацию, информацию о наличии медицинского образования, стаж работы.

В среде разработки *Visual Studio* было создано приложение, которое позволяет просматривать список мастеров, добавлять новых мастеров в коллекцию и редактировать информацию об уже существующих мастерах. Интерфейс приложения представлен на рисунке 2.

	Name	Spec	Edu	Work
▶	Старцева Любо...	Парикмахер	Нет	5
	Иванова Анна И...	Визажист	Нет	3
	Шарапова Дарь...	Косметолог	Нет	2
	Абрамовская Д...	Бровист	Нет	4
	Елисеева Светл...	Косметолог	Есть	5
	Столбовская Ал...	Бровист	Нет	5
	Петренко Валер...	Парикмахер	Нет	1
	Стаценко Евген...	Визажист	Нет	1
	Парамонова Па...	Косметолог	Есть	6
	Колесникова Ан...	Парикмахер	Нет	5

Id	<input type="text" value="5f6cc7ad27a6c93a04b49721"/>
ФИО	<input type="text" value="Старцева Любовь Андреевна"/>
Специальность	<input type="text" value="Парикмахер"/>
Образование	<input type="text" value="Нет"/>
Стаж работы	<input type="text" value="5"/>

Вставить

Обновить

Удалить

Рисунок 2 – Интерфейс программы.

Для добавления нового мастера нужно заполнить все поля, кроме *id* (он генерируется автоматически при добавлении в БД), и нажать на кнопку «Вставить». Для изменения информации об уже существующем мастере, необходимо выбрать его в списке (тогда вся информация о нём занесётся в элементы *textbox*), внести нужные изменения и нажать на кнопку «Обновить». Для удаления мастера из коллекции нужно выбрать его в списке и нажать на кнопку «Удалить». После каждой манипуляции список обновляется и можно увидеть все внесённые изменения. Те же изменения происходят и в самой БД – для того, чтобы их увидеть, нужно нажать на кнопку «*Refresh*» в окне работы с коллекцией в клиенте *MongoDB Compass*.

Вывод: в ходе выполнения лабораторной работы были получены навыки создания нереляционной БД в СУБД *MongoDB*, была настроена связь между созданной БД и приложением *WF*, созданным в среде разработки *Visual Studio*.

Листинг приложения

Form1.cs

```
using MongoDB.Bson;
using MongoDB.Driver;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MongoDB
{
    public partial class Form1 : Form
    {
        static MongoClient client = new MongoClient();
        static IMongoDatabase db = client.GetDatabase("salon");
        static IMongoCollection<Masters> collection =
db.GetCollection<Masters>("masters");
        public void ReadAllDocuments()
        {
            List<Masters> list =
collection.AsQueryable().ToList<Masters>();
            dataGridView1.DataSource = list;
            textBox5.Text =
dataGridView1.Rows[0].Cells[4].Value.ToString();
            textBox1.Text =
dataGridView1.Rows[0].Cells[0].Value.ToString();
            textBox2.Text =
dataGridView1.Rows[0].Cells[1].Value.ToString();
            textBox3.Text =
dataGridView1.Rows[0].Cells[2].Value.ToString();
            textBox4.Text =
dataGridView1.Rows[0].Cells[3].Value.ToString();
        }
    }
}
```

```

    }

    public Form1()
    {
        InitializeComponent();
        ReadAllDocuments();
    }

    private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        textBox5.Text =
dataGridView1.Rows[e.RowIndex].Cells[4].Value.ToString();
        textBox1.Text =
dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
        textBox2.Text =
dataGridView1.Rows[e.RowIndex].Cells[1].Value.ToString();
        textBox3.Text =
dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString();
        textBox4.Text =
dataGridView1.Rows[e.RowIndex].Cells[3].Value.ToString();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        Masters m = new Masters(textBox1.Text, textBox2.Text,
textBox3.Text, int.Parse(textBox4.Text));
        collection.InsertOne(m);
        ReadAllDocuments();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        var updateDef = Builders<Masters>.Update.Set("name",
textBox1.Text).Set("spec",
textBox2.Text).Set("edu",
textBox3.Text).Set("work", Convert.ToInt32(textBox4.Text));
        collection.UpdateOne(m => m.Id ==
ObjectId.Parse(textBox5.Text), updateDef);
        ReadAllDocuments();
    }

```

```

    }

    private void button3_Click(object sender, EventArgs e)
    {
        collection.DeleteOne(m => m.Id ==
ObjectId.Parse(textBox5.Text));
        ReadAllDocuments();
    }
}

```

Masters.cs

```

using MongoDB.Bson;
using MongoDB.Bson.Serialization.Attributes;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MongoDB
{
    class Masters
    {
        [BsonElement("name")]
        public String Name { get; set; }
        [BsonElement("spec")]
        public String Spec { get; set; }
        [BsonElement("edu")]
        public String Edu { get; set; }
        [BsonElement("work")]
        public int Work { get; set; }
        [BsonId]
        public ObjectId Id { get; set; }
    }
}

```

```
work)      public Masters(string name, string spec, string edu, int
           {
             Name = name;
             Spec = spec;
             Edu = edu;
             Work = work;
           }
         }
       }
```