УДК 004

Перенос данных из XML в реляционную базу данных SQL

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье рассматривается процесс переноса данных из XML-документов в реляционную базу данных SQL. Описывается методика извлечения информации из структурированных XML-файлов и их последующая трансформация и загрузка в таблицы SQL.
Ключевые слова:SQL, XML, импорт, C#, Visual Studio

Transferring data from XML to a relational SQL database

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article discusses the process of transferring data from XML documents to a relational SQL database. The method of extracting information from structured XML files and their subsequent transformation and loading into SQL tables is described.
Keywords: SQL, XML, import, C#, Visual Studio

Перенос данных из XML в реляционную базу данных SQL остается актуальным и важным процессом по нескольким причинам:

1. Интеграция данных: XML широко используется для обмена данными между различными системами и приложениями. Перенос данных из XML в SQL позволяет интегрировать информацию в реляционные базы данных для дальнейшего анализа и обработки.
2. Гибкость: XML предоставляет гибкую структуру для представления сложных данных, в то время как SQL базы данных обеспечивают мощные средства для управления и запроса данных. Комбинация этих двух технологий позволяет эффективно работать с данными.
3. Автоматизация: Использование XML файлов для динамического создания SQL таблиц упрощает и автоматизирует процесс разработки, сокращая время и усилия, необходимые для ручного кодирования.
4. Масштабируемость: Автоматизация процесса переноса данных позволяет легко масштабировать системы, добавляя новые данные без значительных изменений в коде.

5. Совместимость: XML и SQL являются стандартами, которые поддерживаются большинством платформ и языков программирования, что обеспечивает высокую совместимость и переносимость решений.
6. Обновление данных: Перенос данных из XML в SQL позволяет обновлять существующие базы данных новыми данными, сохраняя при этом целостность и актуальность информации.
7. Безопасность: Реляционные базы данных предлагают механизмы безопасности и контроля доступа, которые важны для защиты данных при их переносе и хранении.

В целом, перенос данных из XML в SQL базы данных остается ключевым элементом многих бизнес-процессов и IT-инфраструктур, обеспечивая эффективное управление данными и поддержку принятия обоснованных решений.

Целью данной статьи является создание простого метода переноса данных из XML в реляционную базу данных SQL в среде разработки «Visual Studio» на языке программирования C#.

В своей работе Н. Н. Додобоев, О. И. Кукарцева, Я. А. Тынченко рассмотрели вопросы появления различных языков программирования (в частности C#), определения особенностей этих языков, а также составления основных видов и классификаций языков программирования [1]. З. С. Магомадова рассмотрела языки программирования высокого уровня, особенности, недостатки и сложности в изучении, а также описала несколько легких алгоритмов [2]. В работе Г.М. Валитова, Б.Б. Чумакова рассматривается система импорта данных из баз данных СУБД Microsoft SQL Server, Microsoft Access, Oracle, Paradox и Visual FoxPro в СУБД MySQL. Система представляет собой Менеджер загрузки, необходимый для загрузки данных в хранилища данных из оперативных данных. И также в работе рассматривается создание общего интерфейса, позволяющего быстро внедрять новые драйверы для работы с файлами баз данных различных СУБД и производить импорт данных в MySQL с наименьшим количество затрат времени на разработку [3]. В статье А. Вайта рассматривается применение сценариев PowerShell для автоматизации процесса импорта данных из XML-файлов в таблицы реляционной базы данных SQL Server. Описываются методы и командлеты PowerShell, которые позволяют эффективно преобразовывать XML-данные и загружать их в SQL Server, обеспечивая при этом высокую скорость выполнения и точность переноса. Статья также подчеркивает важность использования PowerShell в задачах интеграции данных и демонстрирует, как данный инструмент может упростить и оптимизировать рабочие процессы разработчиков и администраторов баз данных [4].

Создаем проект «Windows Forms App» и называем необходимым именем см. рисунок 1.

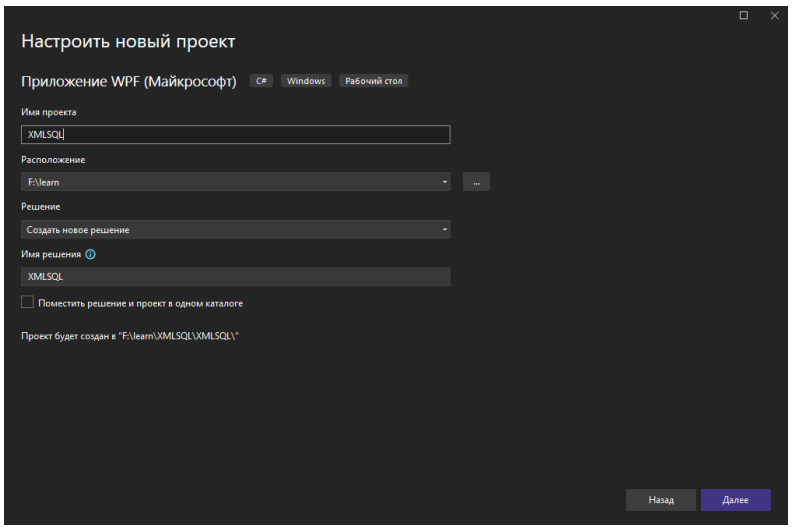

Рис. 1. Создание проекта

Далее заполняем форму необходимыми полями, добавляем: текстовое поле для присваивания пути к файлу XML, кнопку для просмотра и выбора файла XML на локальном диске, компонент OpenFileDialog для обработки выбора файла, кнопку импорта, которая будет выполнять основные функции проекта, и индикатор выполнения, показывающий прогресс выполнения функции нашего приложения см. рисунке 2.

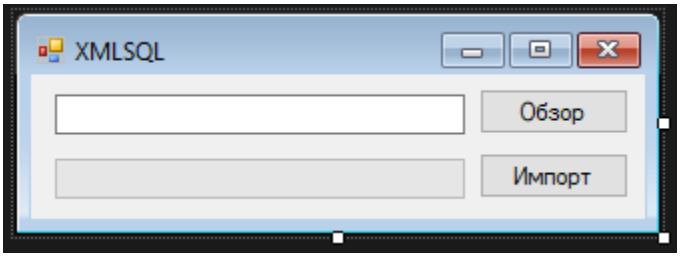

Рис. 2. Компоненты формы

После разработки формы приступаем к написанию кода. Сначала напишем фрагмент кода, отвечающий за интерфейс пользователя, позволяя пользователю выбрать XML-файл через стандартное диалоговое окно и сохранить путь к этому файлу в текстовое поле на форме см. рисунок 3-7.

```
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;
using System.IO;
using System.Data.SqlClient;
using System.Configuration;
using System.Xml;

namespace XMLtoDatabase
{
    Ссылок: 3
    public partial class Form1 : Form
    {

        string StrCon = ConfigurationManager.ConnectionStrings["strcon"].ToString();

        Ссылок: 1
        public Form1()
        {
            InitializeComponent();
        }


        Ссылок: 1
        private void btnBrowse_Click(object sender, EventArgs e)
        {
            if (OFD.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                txtFilePath.Text = OFD.FileName;

        }
```

Рис. 3. Настройка интерфейса пользователя

Далее переходим к основной логике программы. Напишем код позволяющий пользователю импортировать данные из XML-файла в реляционную базу данных SQL, автоматически создавая таблицу и заполняя данными. Данный фрагмент включает проверки и обработки ошибок для обеспечения корректности выполнения операций см. рисунок 4.

```
private void btnImport_Click(object sender, EventArgs e)
{
    string XMlFile = txtFilePath.Text;
    if (File.Exists(XMlFile))
    {

        DataTable dt = CreateDataTableXML(XMlFile);
        if (dt.Columns.Count == 0)
            dt.ReadXml(XMlFile);


        string Query = CreateTableQuery(dt);
        SqlConnection con = new SqlConnection(StrCon);
        con.Open();


        SqlCommand cmd = new SqlCommand("IF OBJECT_ID('dbo." + dt.TableName + "', 'U') IS NOT NULL DROP TABLE dbo." + dt.TableName + ";", con);
        cmd.ExecuteNonQuery();


        cmd = new SqlCommand(Query, con);
        int check = cmd.ExecuteNonQuery();
        if (check != 0)
        {

        using (var bulkCopy = new SqlBulkCopy(con.ConnectionString, SqlBulkCopyOptions.KeepIdentity))
        {

            foreach (DataColumn col in dt.Columns)
            {
                bulkCopy.ColumnMappings.Add(col.ColumnName, col.ColumnName);
            }

            bulkCopy.BulkCopyTimeout = 600;
            bulkCopy.DestinationTableName = dt.TableName;
            bulkCopy.WriteToServer(dt);
        }

            MessageBox.Show("Table Created Successfully");
        }
        con.Close();
```

Рис. 4. Основная логика программы

Далее создадим метод GetTableName принимающий путь к файлу в качестве аргумента и возвращающий имя таблицы, которое соответствует имени файла без расширения. Это делается с помощью класса FileInfo, который предоставляет свойства и методы для работы с файлами. Метод CreateTableQuery создает SQL-запрос для создания новой таблицы в базе

данных на основе структуры объекта DataTable. Метод устанавливает максимальное значение для progressBar1, которое соответствует количеству столбцов в DataTable, и инициализирует его значением 0. Затем метод перебирает все столбцы в DataTable, определяет их тип данных и формирует соответствующую часть SQL-запроса для каждого столбца. Если столбец имеет автоинкремент, добавляется соответствующий SQL-код. Также проверяется, разрешено ли значение NULL для столбца, и, если нет, добавляется ограничение NOT NULL см. рисунок 5.

```
public string GetTableName(string file)
{
    FileInfo fi = new FileInfo(file);
    string TableName = fi.Name.Replace(fi.Extension, "");

    return TableName;
}


Ссылок: 1
public string CreateTableQuery(DataTable table)
{
    string sqlsc = "CREATE TABLE " + table.TableName + "(";
    progressBar1.Maximum = table.Columns.Count;
    progressBar1.Value = 0;
    for (int i = 0; i < table.Columns.Count; i++)
    {
        sqlsc += "[" + table.Columns[i].ColumnName + "]";
        string columnType = table.Columns[i].DataType.ToString();
        switch (columnType)
        {
            case "System.Int32":
                sqlsc += " int ";
                break;
            case "System.Int64":
                sqlsc += " bigint ";
                break;
            case "System.Int16":
                sqlsc += " smallint";
                break;
            case "System.Byte":
                sqlsc += " tinyint";
                break;
            case "System.Decimal":
                sqlsc += " decimal ";
                break;
            case "System.DateTime":
                sqlsc += " datetime ";
                break;
            case "System.String":
            default:
                sqlsc += string.Format(" nvarchar({0}) ", table.Columns[i].MaxLength == -1 ? "max" : table.Columns[i].MaxLength.ToString());
                break;
        }
        if (table.Columns[i].AutoIncrement)
            sqlsc += " IDENTITY(" + table.Columns[i].AutoIncrementSeed.ToString() + "," + table.Columns[i].AutoIncrementStep.ToString() + ") ";
        if (!table.Columns[i].AllowDBNull)
            sqlsc += " NOT NULL ";
        sqlsc += ",";
```

Рис. 5. Метод GetTableName и метод CreateTableQuery

Создаем метод CreateDataTableXML принимающий путь к XML-файлу в качестве аргумента и выполняющий следующие действия:

1. Загружает XML-документ.
2. Создает новый объект DataTable.
3. Устанавливает имя таблицы, используя метод GetTableName, который извлекает имя файла без расширения.
4. Определяет структуру таблицы, используя первый дочерний узел корневого элемента XML-документа.
5. Устанавливает максимальное значение для progressBar1, равное количеству дочерних узлов структуры.
6. Добавляет столбцы в DataTable, используя имена узлов структуры.
7. Заполняет DataTable данными, извлекая текстовое содержимое каждого дочернего узла корневого элемента XML-документа.

Метод Progress обновляет индикатор прогресса progressBar1 и отображает процент выполнения в его центре. Данный метод выполняет следующие действия:

1. Проверяет, не достигло ли значение progressBar1 максимального.
2. Увеличивает значение progressBar1.
3. Рассчитывает процент выполнения.
4. Отображает процент выполнения на progressBar1 с помощью графического интерфейса.
5. Вызывает Application.DoEvents для обновления интерфейса пользователя см. рисунок 6-7.

```
            Progress();
        }
        return sqlsc.Substring(0, sqlsc.Length - 1) + "\n)";
    }


    Ссылок: 1
    public DataTable CreateDataTableXML(string XmlFile)
    {
        XmlDocument doc = new XmlDocument();

        doc.Load(XmlFile);

        DataTable Dt = new DataTable();

        try
        {
            Dt.TableName = GetTableName(XmlFile);
            XmlNode NodoEstructura = doc.DocumentElement.ChildNodes.Cast<XmlNode>().ToList()[0];
            progressBar1.Maximum = NodoEstructura.ChildNodes.Count;
            progressBar1.Value = 0;
            foreach (XmlNode columna in NodoEstructura.ChildNodes)
            {
                Dt.Columns.Add(columna.Name, typeof(String));
                Progress();
            }

            XmlNode Filas = doc.DocumentElement;
            progressBar1.Maximum = Filas.ChildNodes.Count;
            progressBar1.Value = 0;
            foreach (XmlNode Fila in Filas.ChildNodes)
            {
                List<string> Valores = Fila.ChildNodes.Cast<XmlNode>().ToList().Select(x => x.InnerText).ToList();
                Dt.Rows.Add(Valores.ToArray());
                Progress();
            }
        }
        catch (Exception ex)
        {

        }
        return Dt;
    }
```

Рис. 6. Метод CreateDataTableXML

```
Ссылок: 3
public void Progress()
{
    if (progressBar1.Value < progressBar1.Maximum)
    {
        progressBar1.Value++;
        int percent = (int)(((double)progressBar1.Value / (double)progressBar1.Maximum) * 100);
        progressBar1.CreateGraphics().DrawString(percent.ToString() + "%", new Font("Arial", (float)8.25, FontStyle.Regular), Brushes.Black,
            new PointF(progressBar1.Width / 2 - 10, progressBar1.Height / 2 - 7));

        Application.DoEvents();
    }
}

Ссылок: 1
private void Form1_Load(object sender, EventArgs e)
{

}
```

Рис. 7. Метод Progress

Проверяем работу программу см. рисунок 8-9.

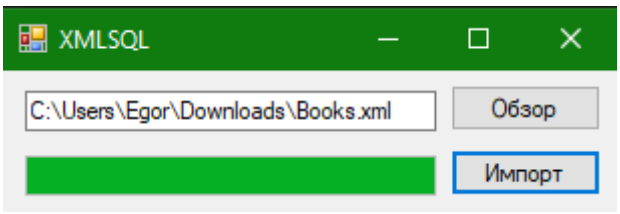

Рис. 8. Основное окно программы

	empno	ename	job	hiredate
▶	7369	SMITH	CLERK	17-DEC-1980
	7499	ALLEN	SALESMAN	20-FEB-1981
	7521	WARD	SALESMAN	22-FEB-1981
	7566	JONES	MANAGER	02-APR-1981
	7654	MARTIN	SALESMAN	28-SEP-1981
	7698	BLAKE	MANAGER	01-MAY-1981
	7782	CLARK	MANAGER	09-JUN-1981
	7788	SCOTT	ANALYST	19-APR-1987
	7839	KING	PRESIDENT	17-NOV-1981
	7844	TURNER	SALESMAN	08-SEP-1981
	7876	ADAMS	CLERK	23-MAY-1987
	7900	JAMES	CLERK	03-DEC-1981
	7902	FORD	ANALYST	03-DEC-1981
	7934	MILLER	CLERK	23-JAN-1982
*	*NULL*	*NULL*	*NULL*	*NULL*

Рис. 9. Созданная база данных

В этой статье были показаны основные концепции XML, DataTable, подключения к базам данных SQL и интеграции Progress Bar, а также работу Progress Bar. Этот проект может выступать в качестве подпроекта или модуля для любого другого проекта. Эту концепцию можно интегрировать в любой из проектов, в котором пользователь динамически создает таблицу SQL с помощью файлов XML. Это избавит от необходимости каждый раз писать код вручную.

Библиографический список

1. Додобоев Н. Н., Кукарцева О. И., Тынченко Я. А. Современные языки программирования // Современные технологии: актуальные вопросы, достижения и инновации. 2014. №5. С. 81-85.
2. Магомадова З. С. Языки программирования высокого уровня // Разработка и применение наукоёмких технологий в эпоху глобальных трансформаций. 2020. №8. С. 94-96.
3. Валитов, Г. М. Система импорта данных в СУБД MySQL на основе свободно распространяемых программных продуктов // Научный альманах. 2016. № 6-2(19). С. 34-37.
4. Вайт, А. Использование PowerShell для загрузки XML-данных в SQL Server // Windows 2000 Magazine/Re. 2014. № 6. С. 69.