.NET作业4

161250010 陈俊达

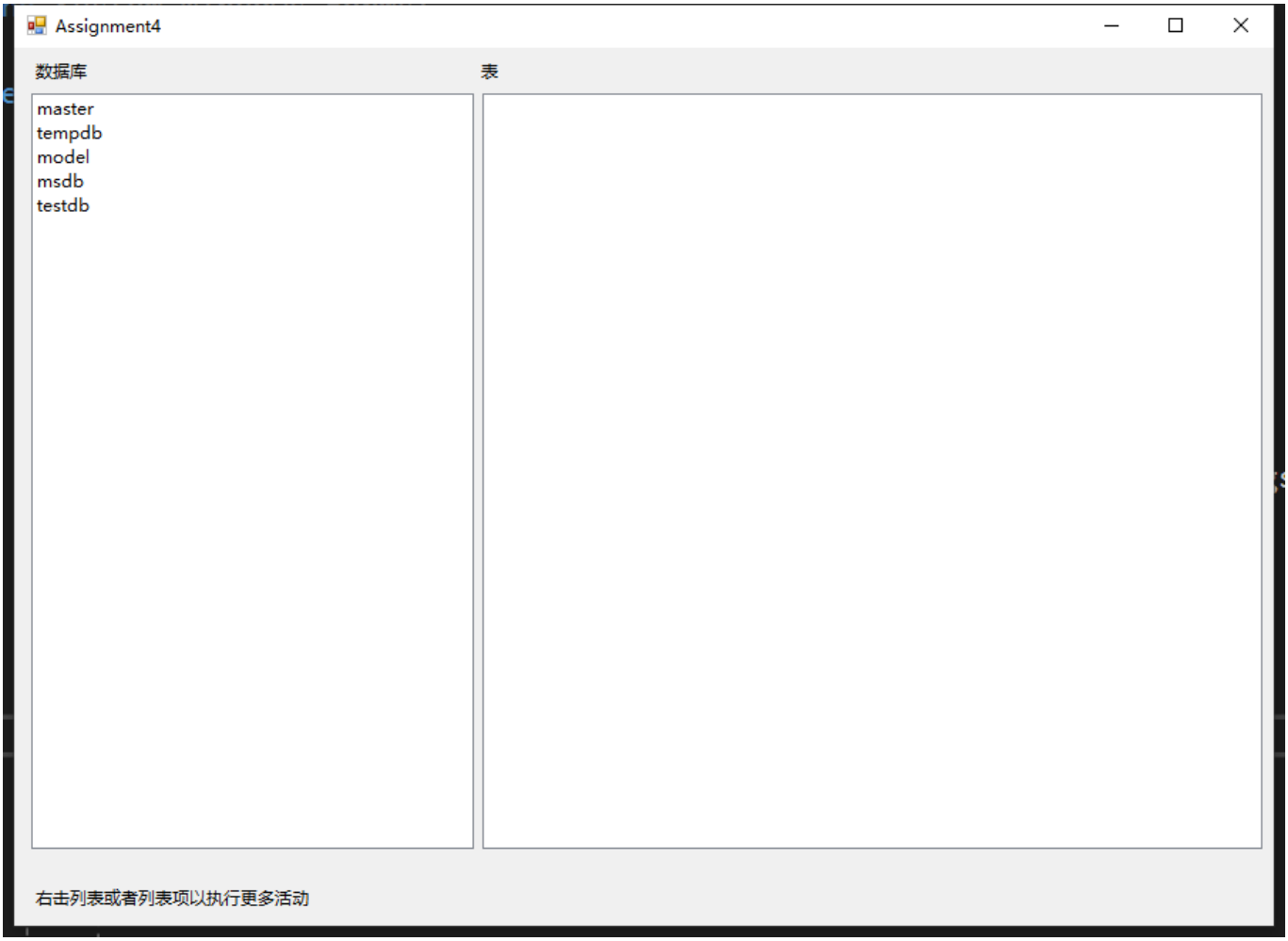
需求确认

由于需求描述比较模糊，系统内有的地方的具体实现依赖我的个人想法。以下为需求中没有描述的、但是对系统功能有密切关系的几个点的说明。

需求	模糊点	实现
以树的形式显示表的内容	树的具体形式	首先使用DataSet的GetXml功能拿到Xml，然后将XML解析成树并显示
增加、删除树节点和修改表格	是否需要数据和数据库同步	添加、删除树节点后均和数据库同步；修改表格点击下方保存后与数据库同步，并刷新两个视图

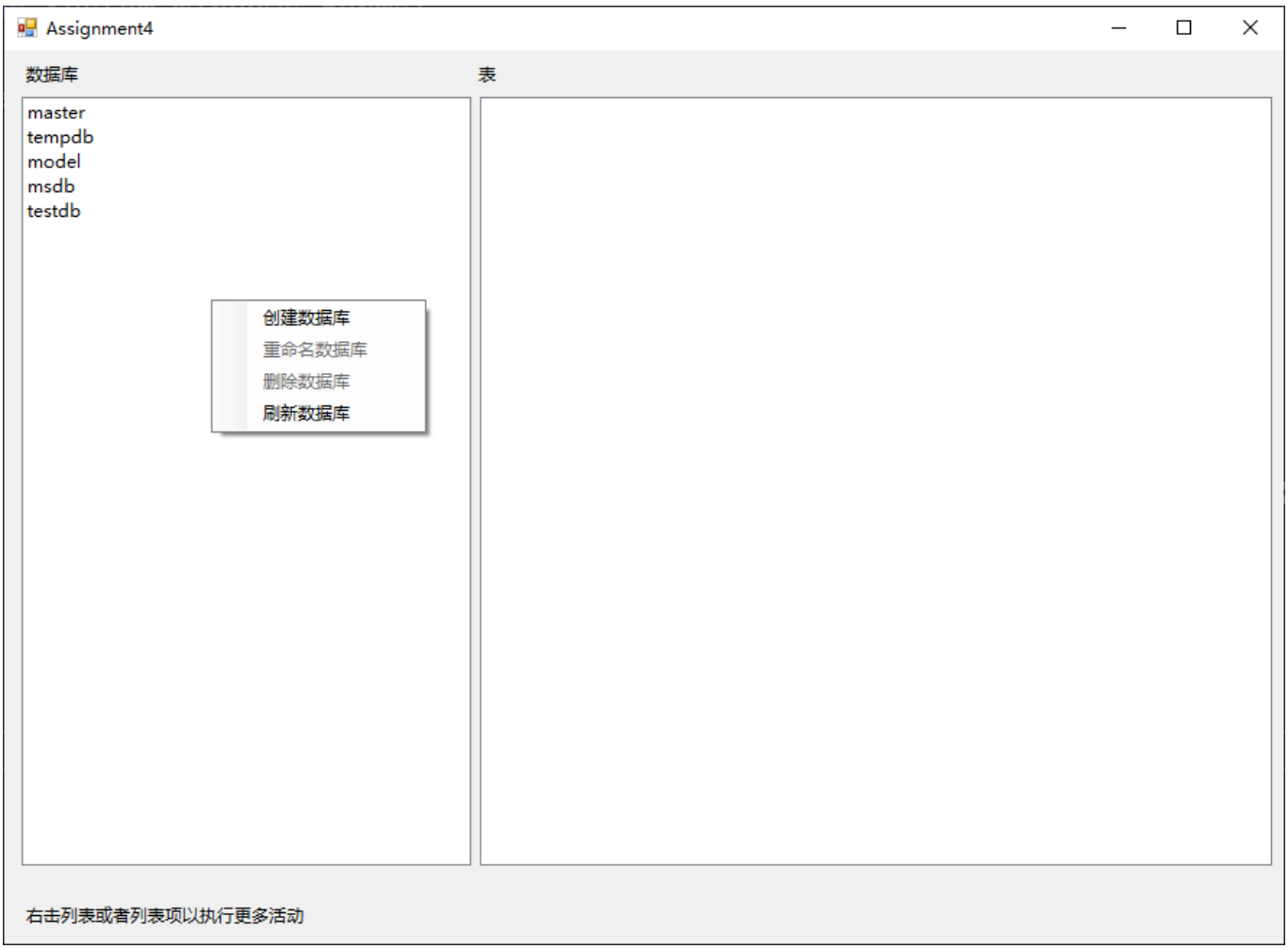
截图

主界面

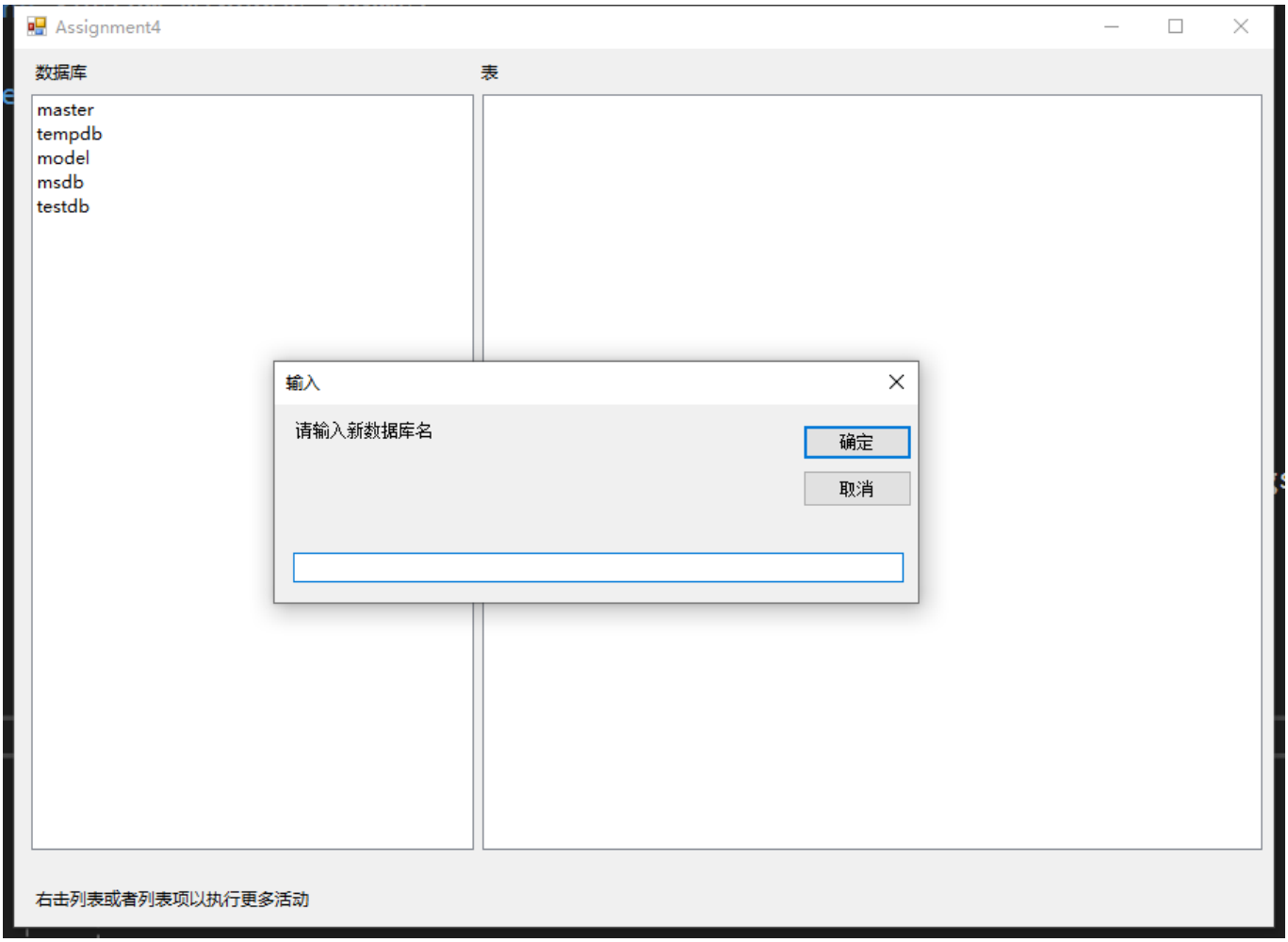


数据库部分

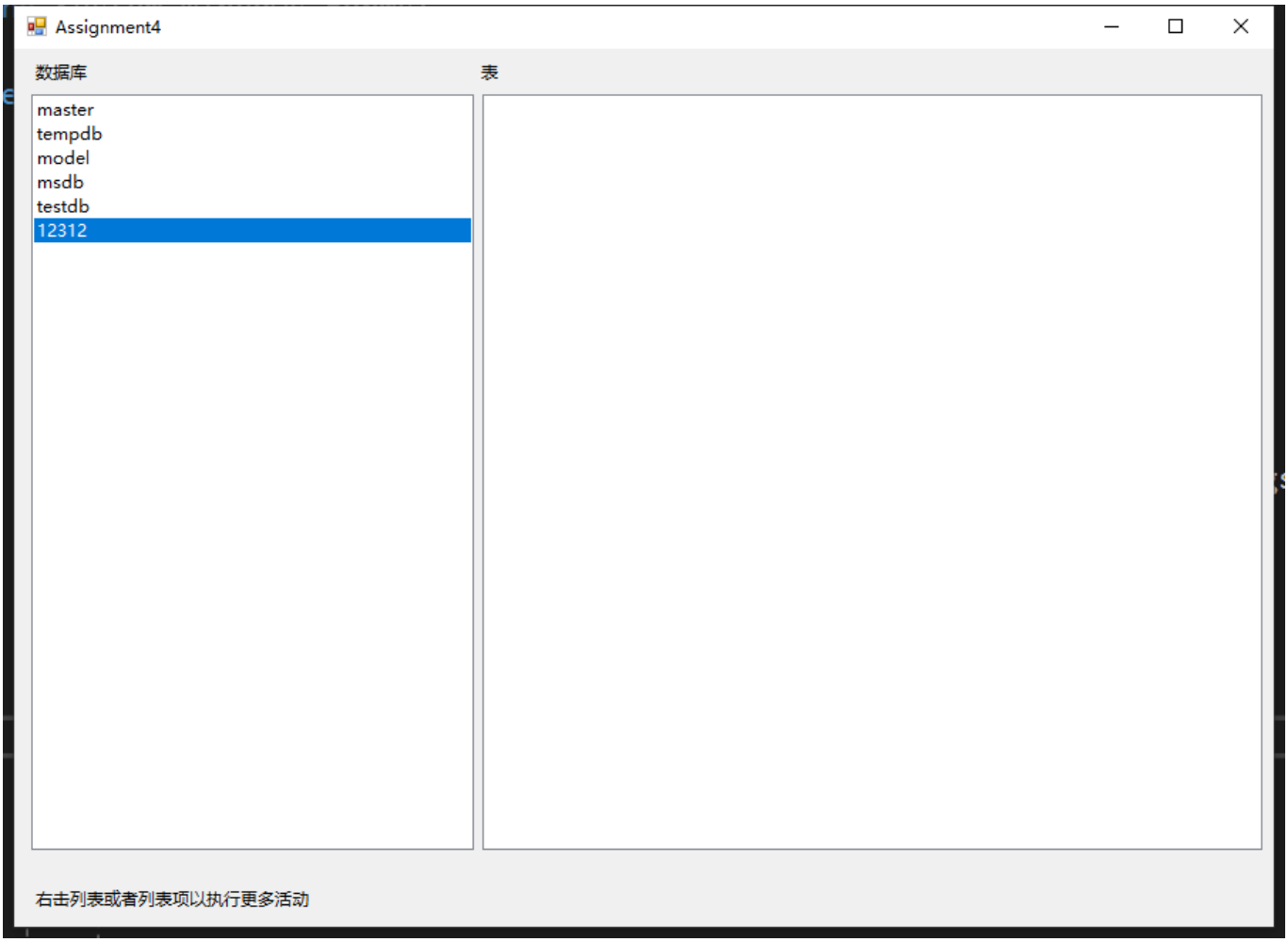
没有选择数据库的情况下右键单击数据库列表



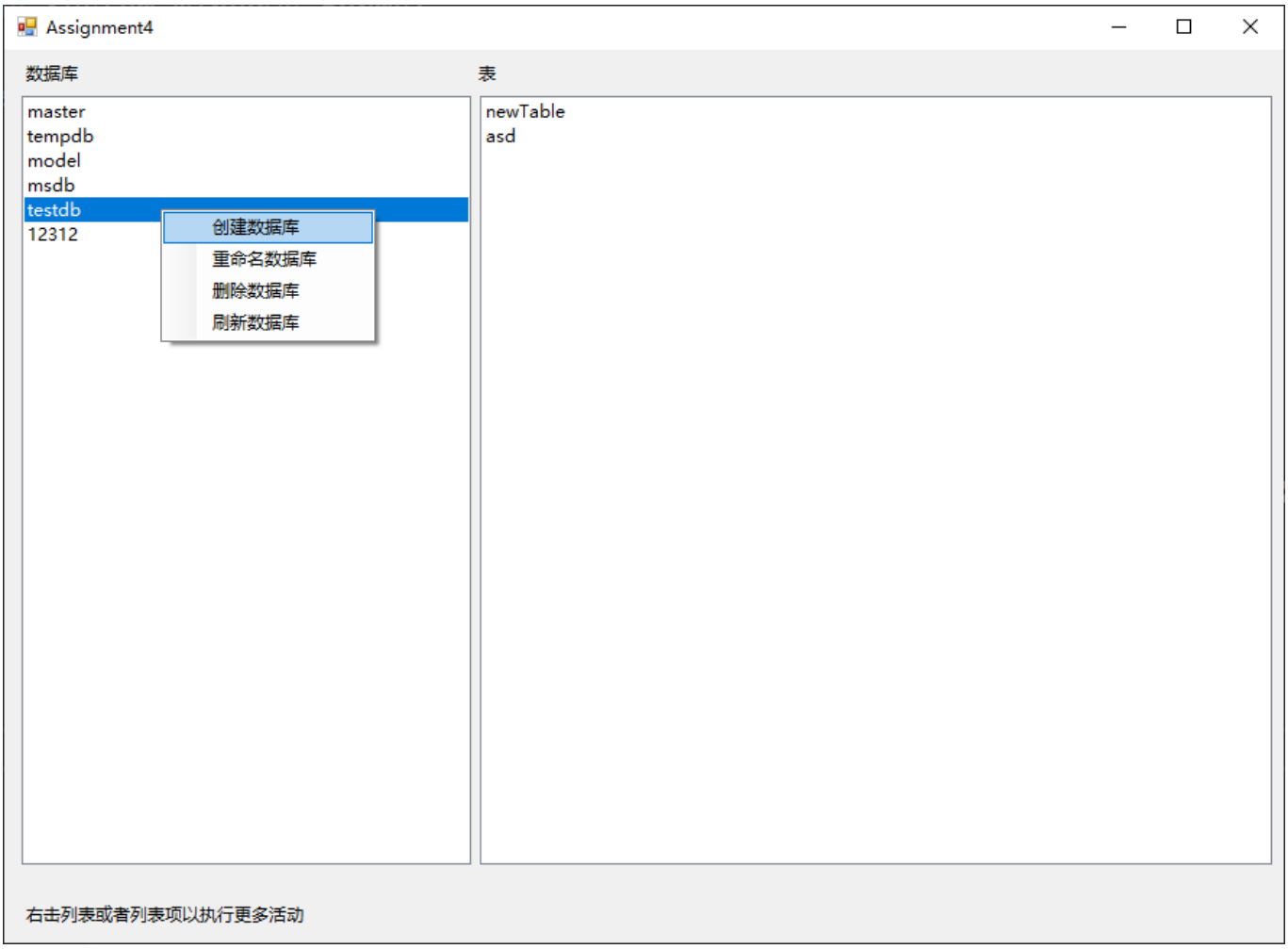
增加数据库


Assignment4
数据库
表
master
tempdb
model
msdb
testdb
输入
请输入新数据库名
确定
取消
右击列表或者列表项以执行更多活动

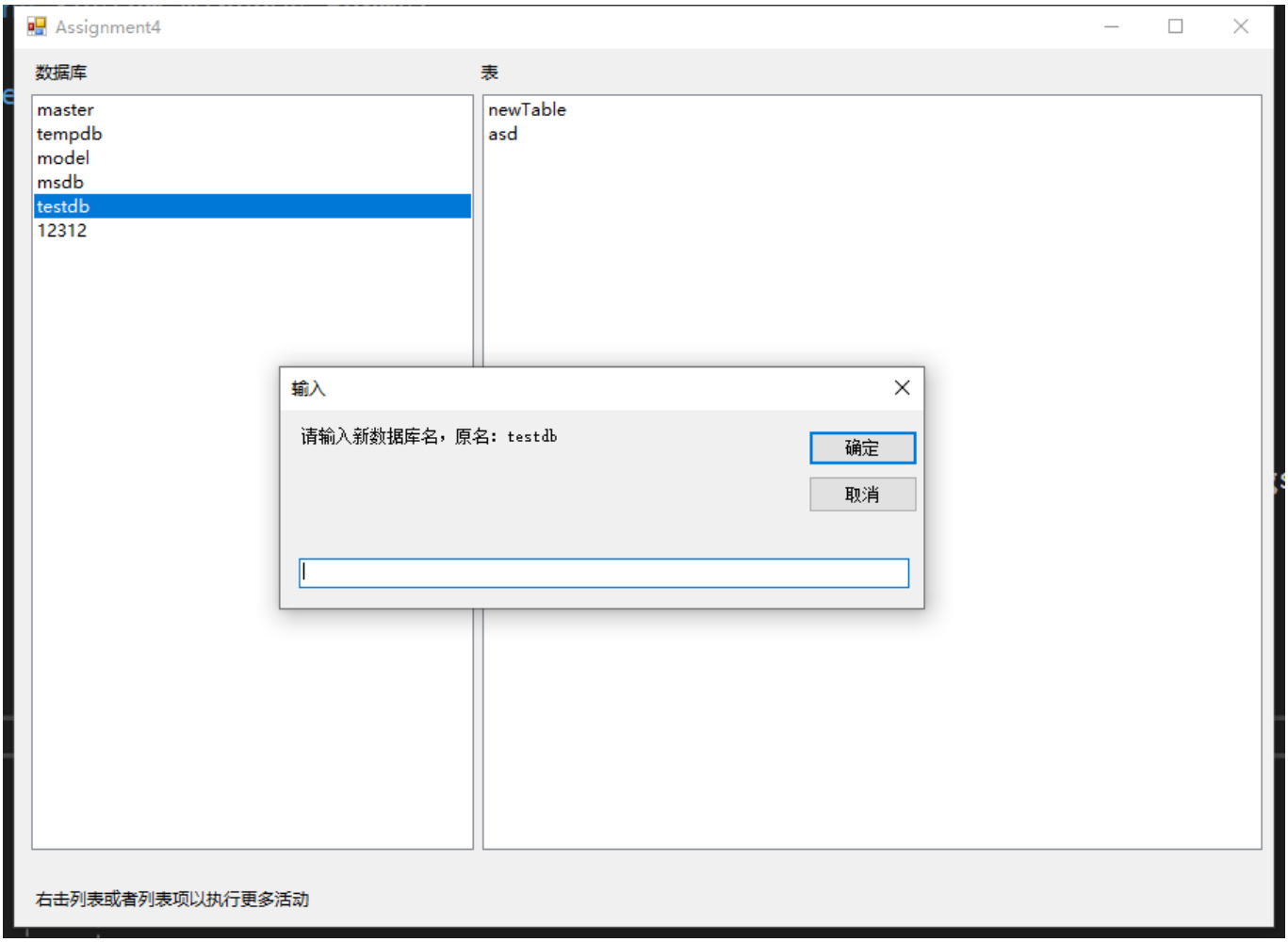
数据库已经增加



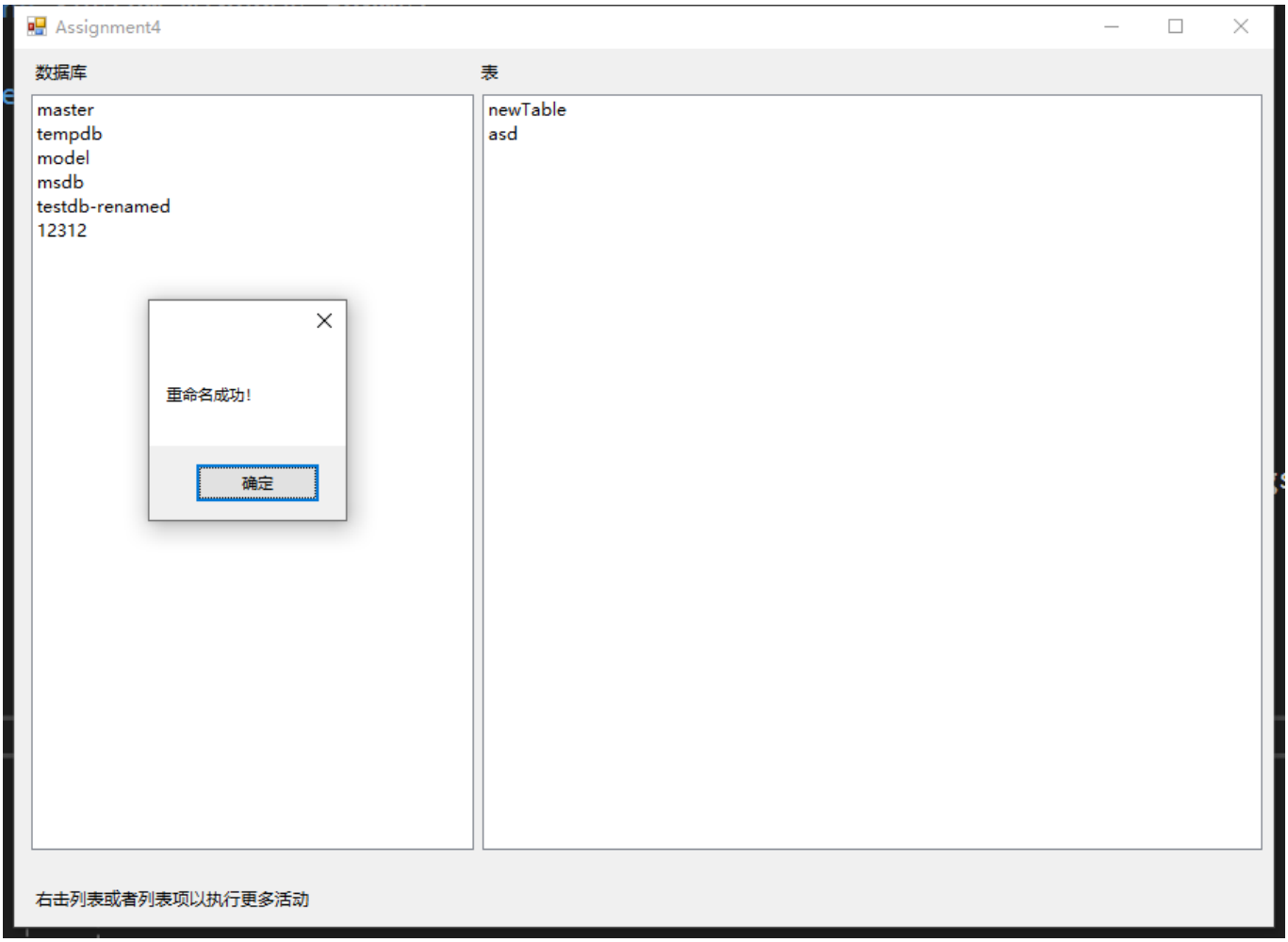
选择数据库后，右键单击数据库列表


Assignment4
数据库
表
master
tempdb
model
msdb
testdb
12312
newTable
asd
创建数据库
重命名数据库
删除数据库
刷新数据库
右击列表或者列表项以执行更多活动

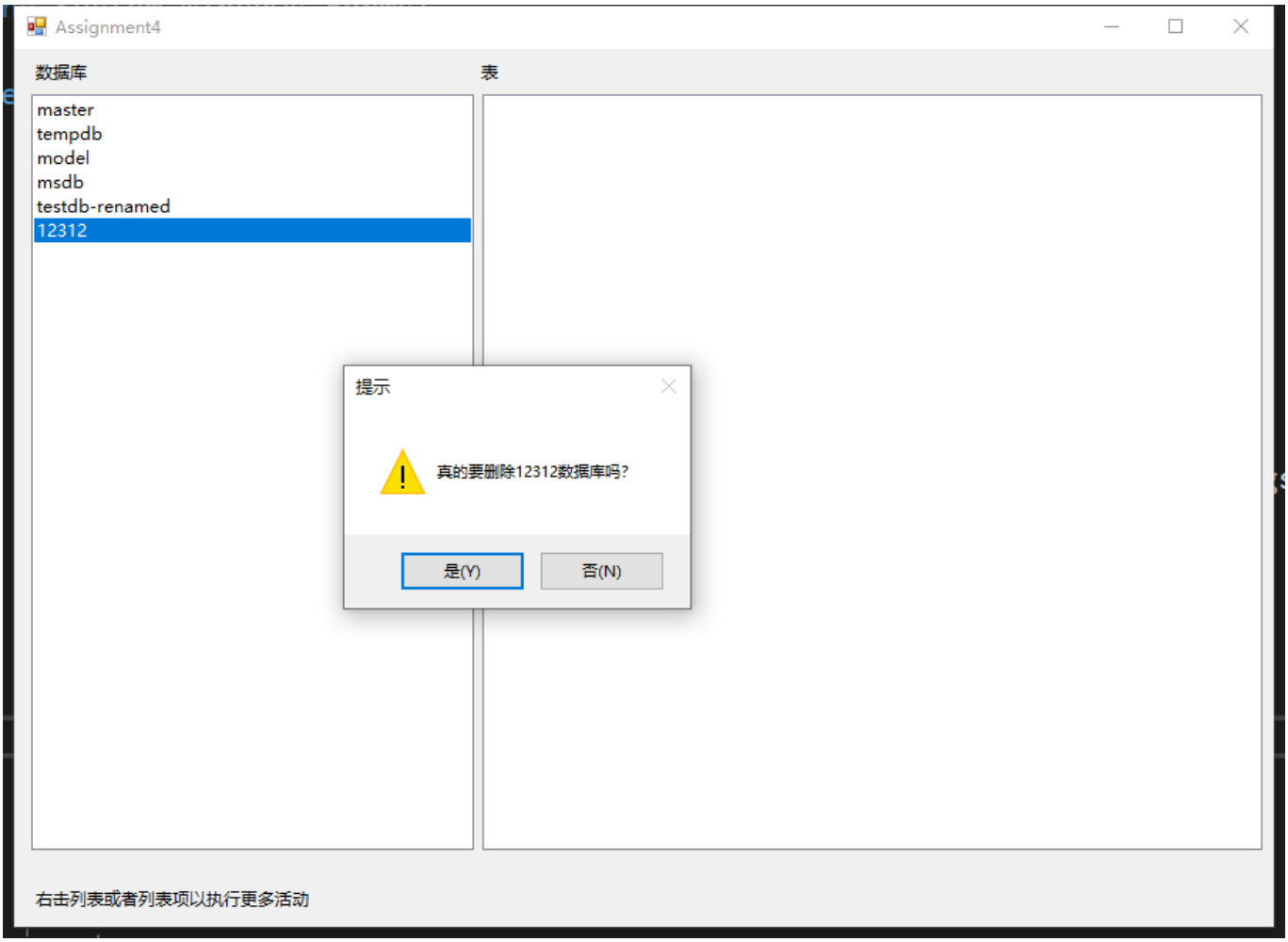
重命名数据库


Assignment4
数据库
表
master
tempdb
model
msdb
testdb
12312
newTable
asd
输入
请输入新数据库名，原名：testdb
确定
取消
右击列表或者列表项以执行更多活动

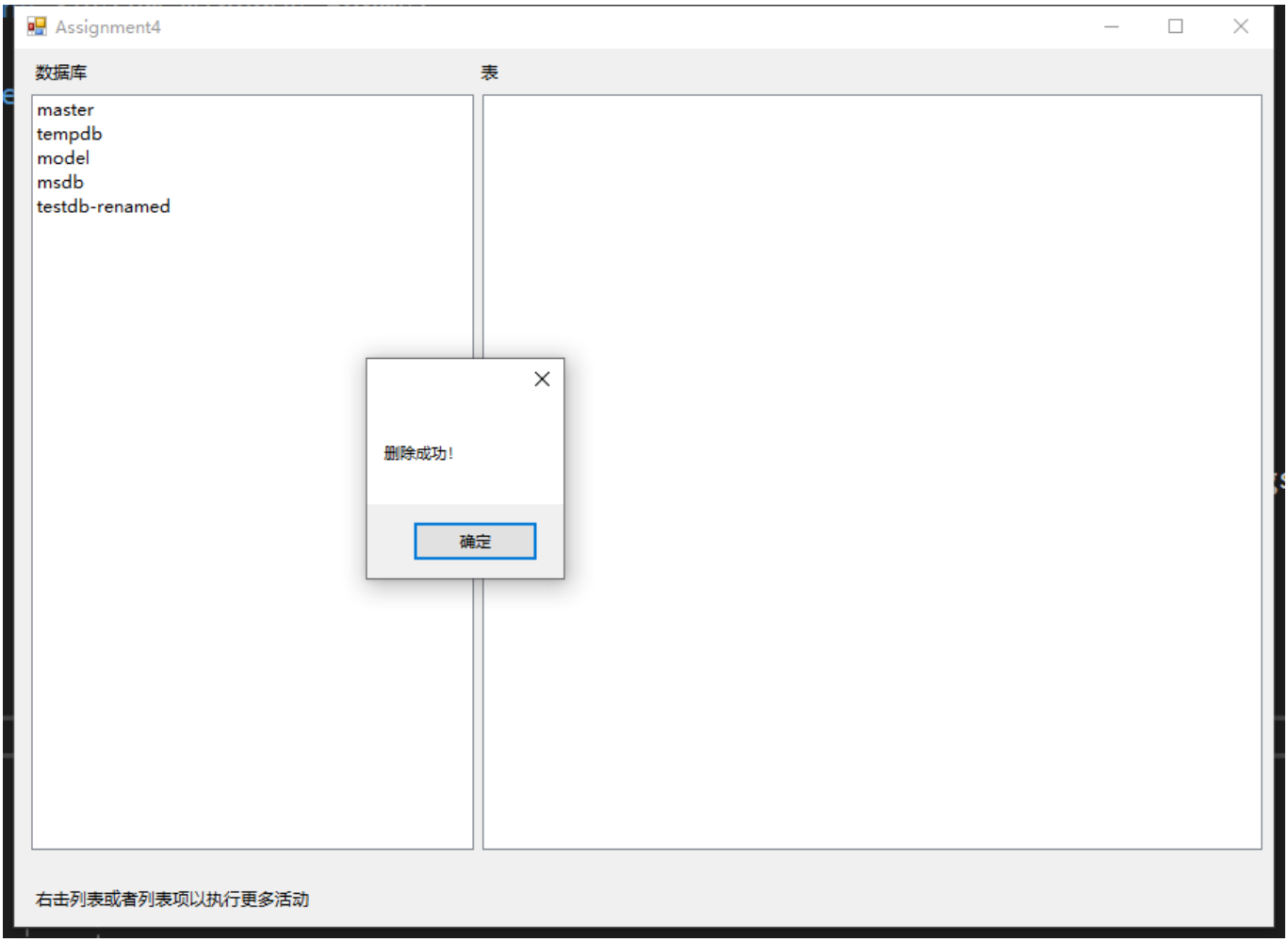
数据库已经重命名


Assignment4
数据库
表
master
tempdb
model
msdb
testdb-renamed
12312
newTable
asd
重命名成功!
确定
右击列表或者列表项以执行更多活动

删除数据库

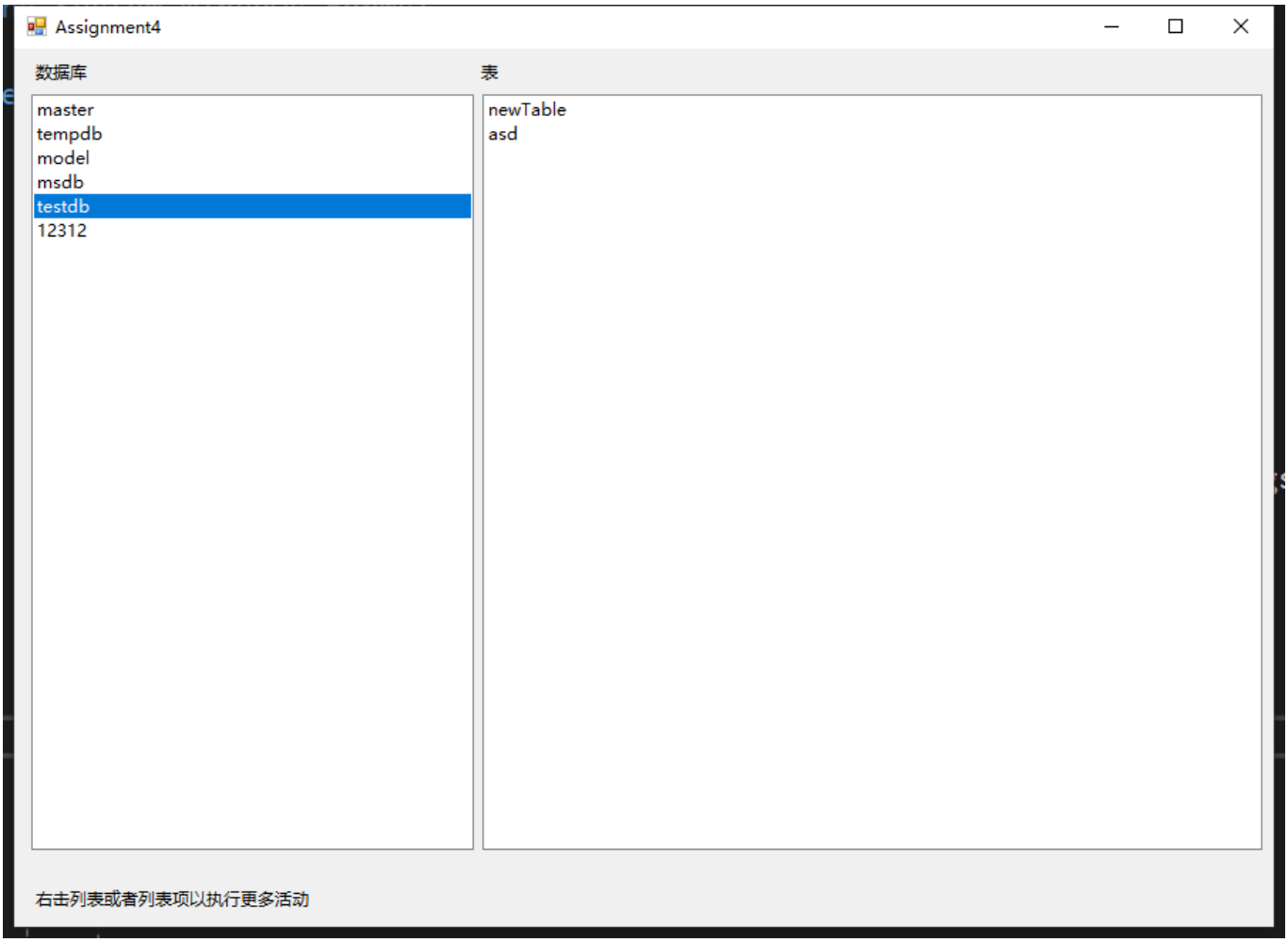

Assignment4
数据库
表
master
tempdb
model
msdb
testdb-renamed
12312
提示
真的要删除12312数据库吗?
是(Y)
否(N)
右击列表或者列表项以执行更多活动

已经删除数据库

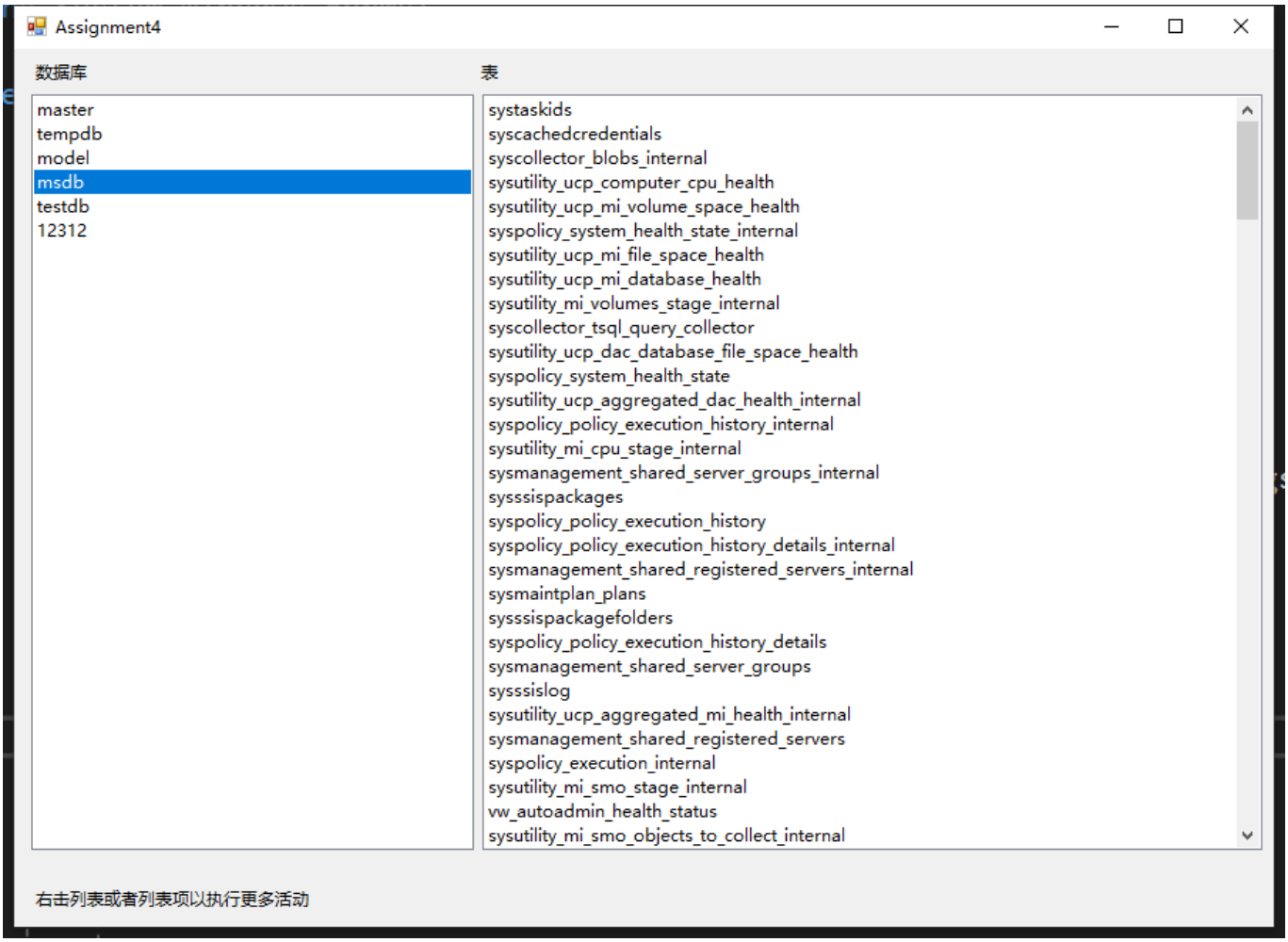


表

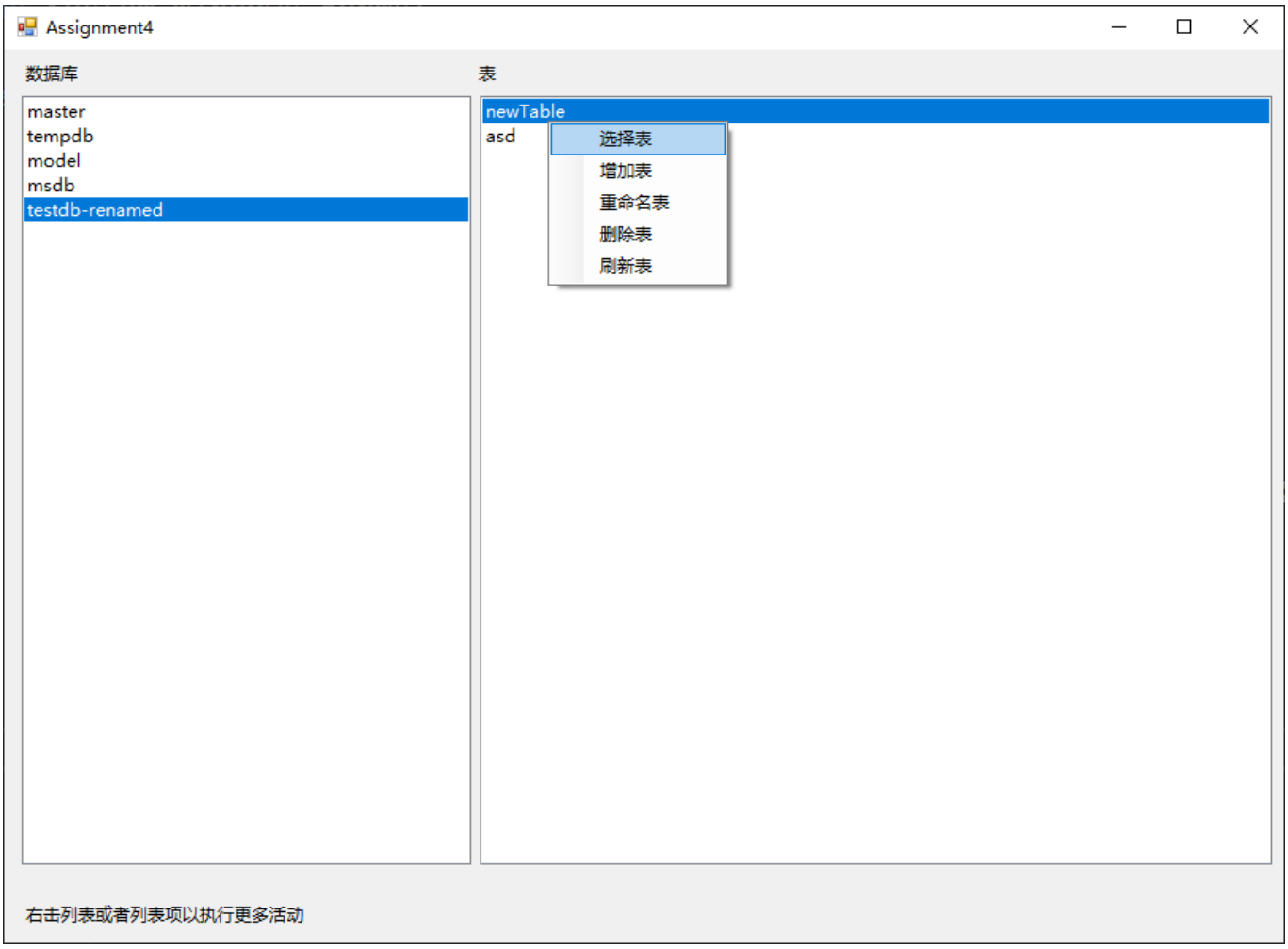
选择数据库后，右侧显示选中数据库的表



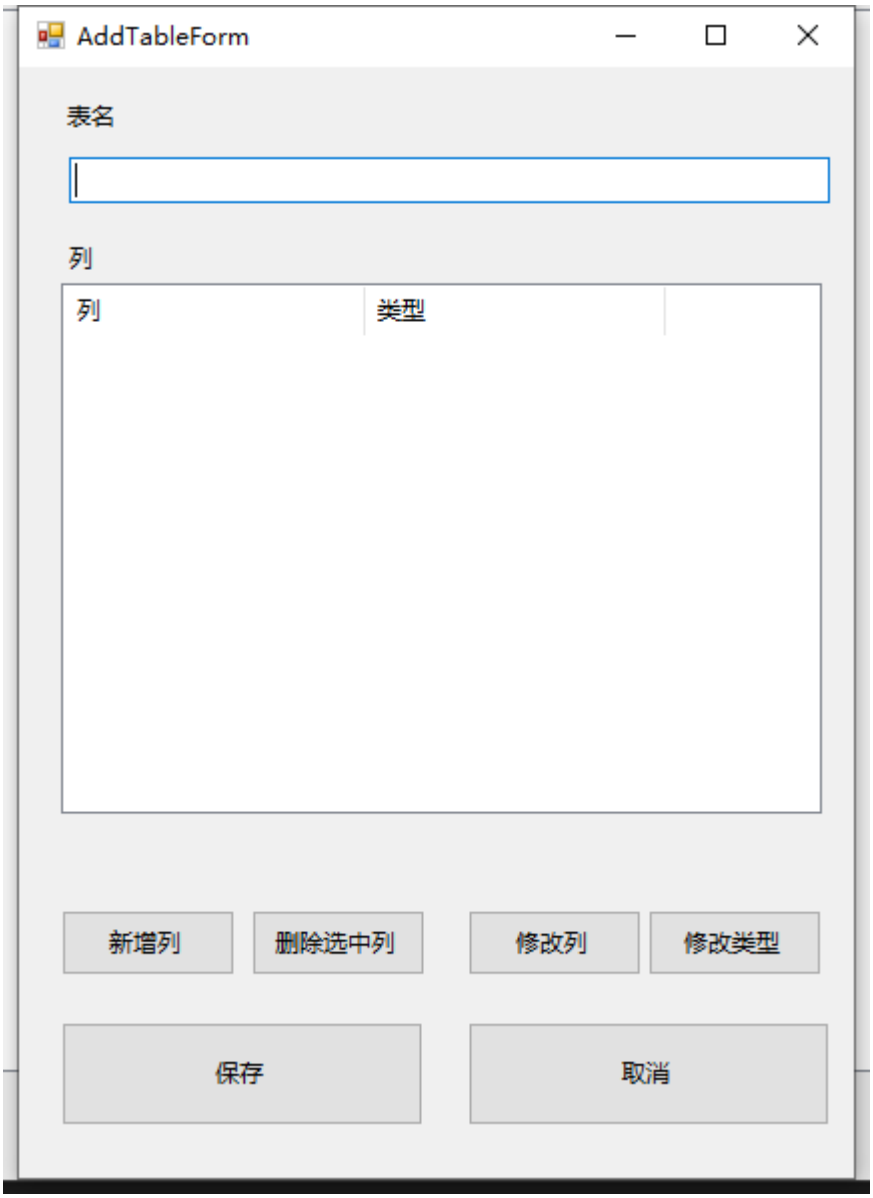
选择其他数据库可显示其他数据库的表



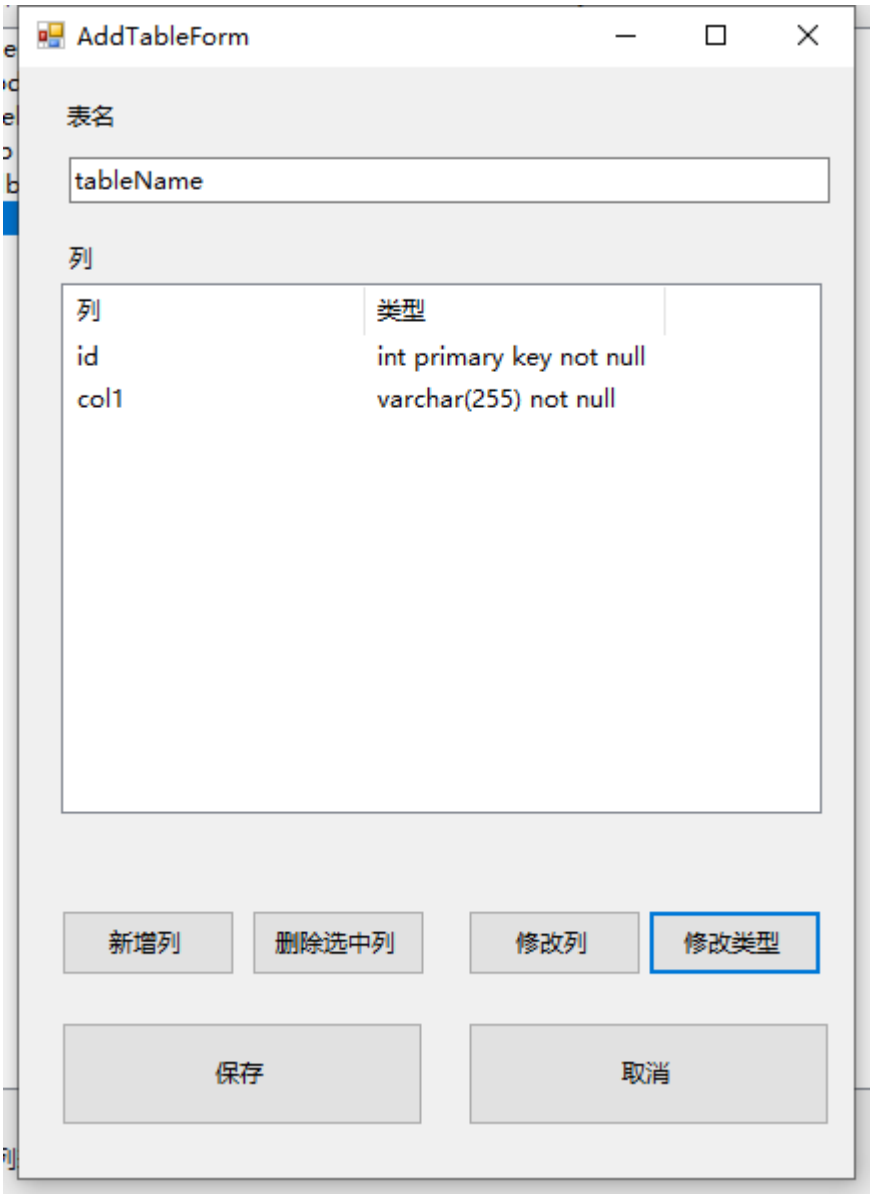
点击一个表后，右击表列表



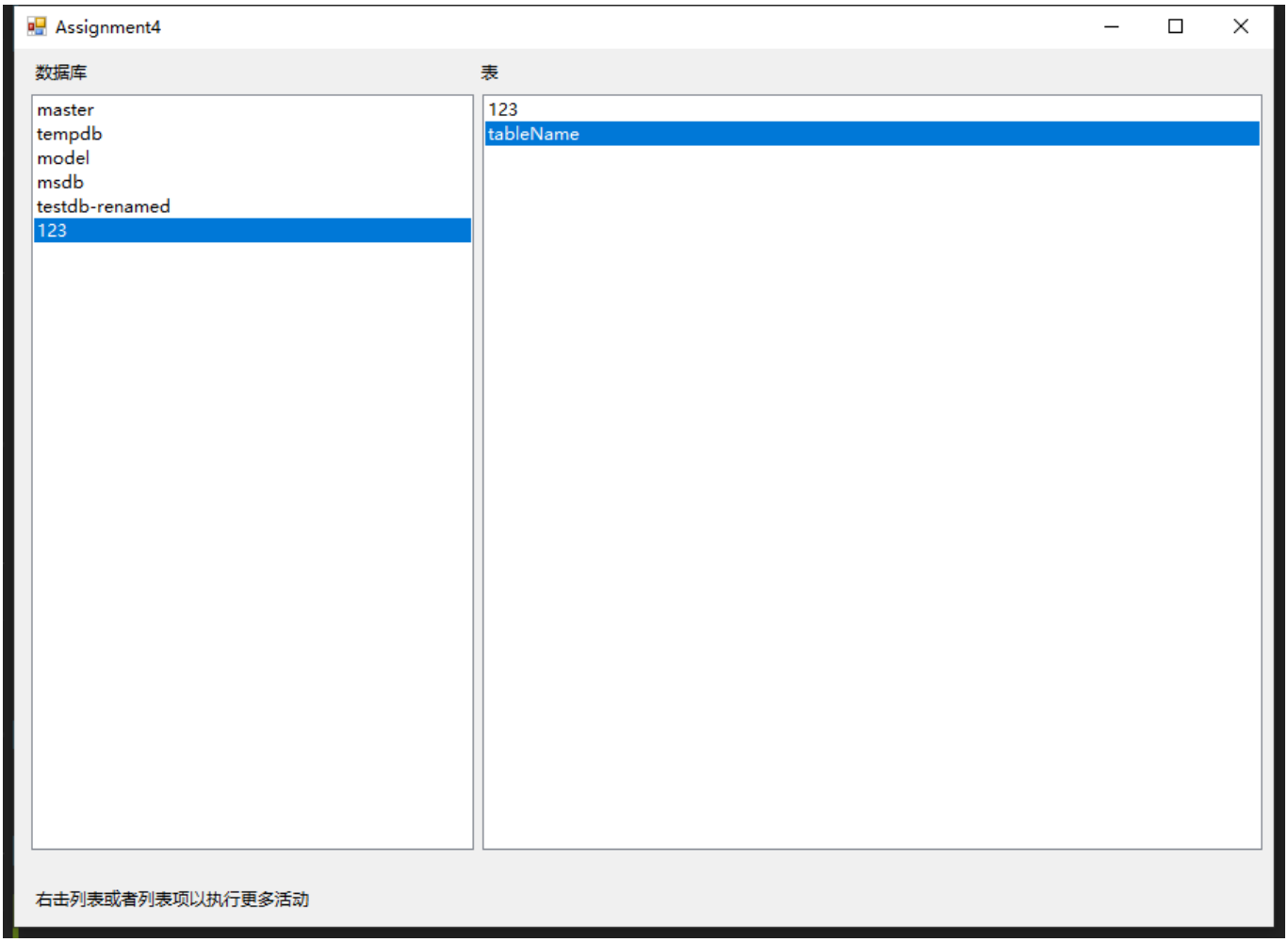
增加表，弹出表信息输入窗口



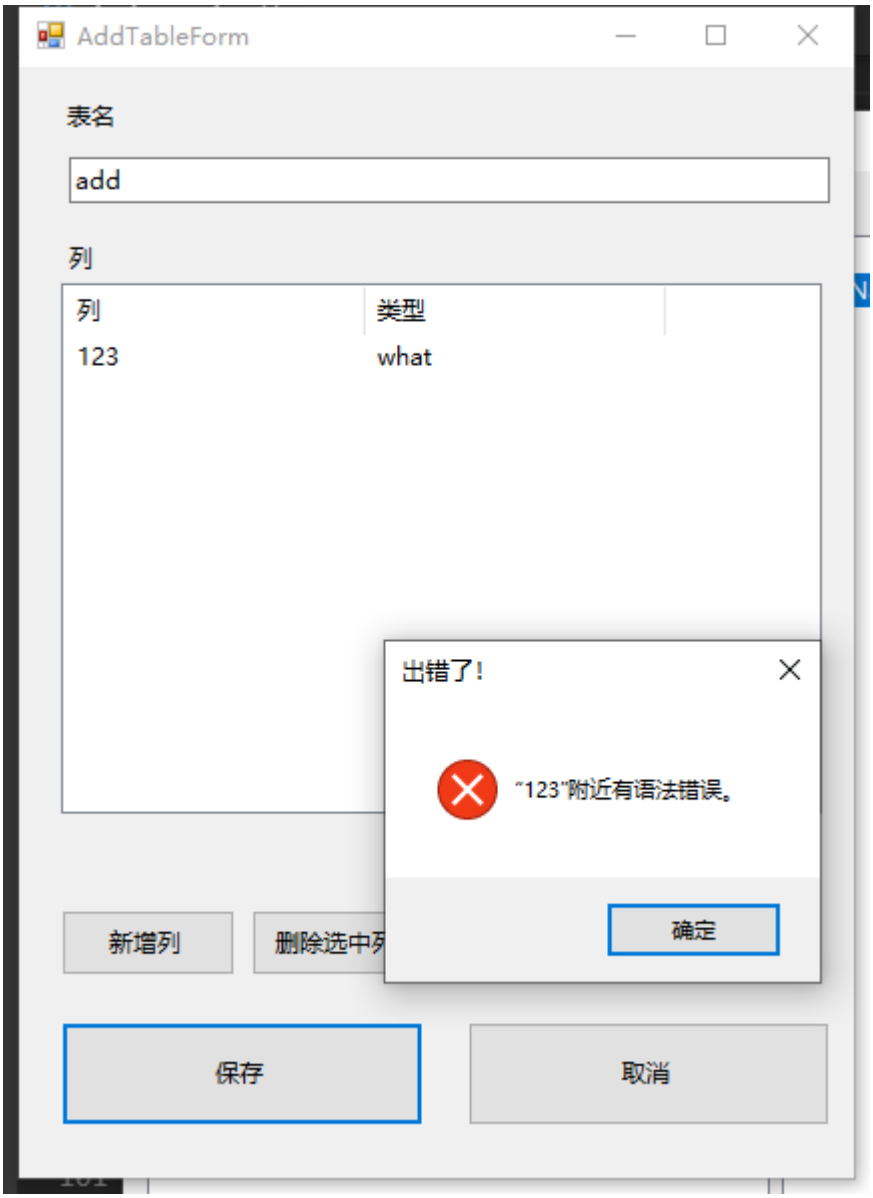
输入表名，添加表列信息


AddTableForm
表名
tableName
列
列
类型
id
int primary key not null
col1
varchar(255) not null
新增列
删除选中列
修改列
修改类型
保存
取消

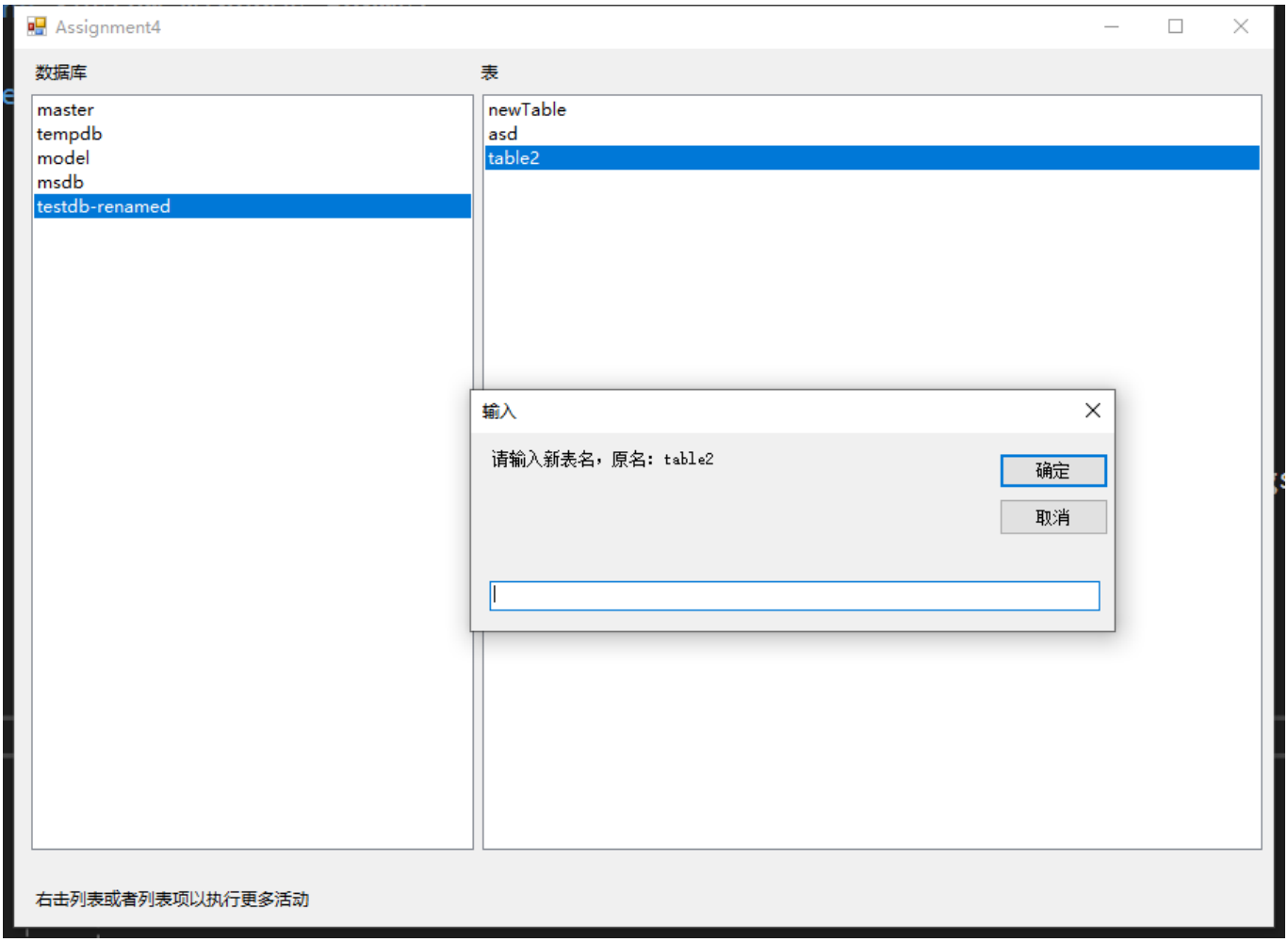
表已经增加



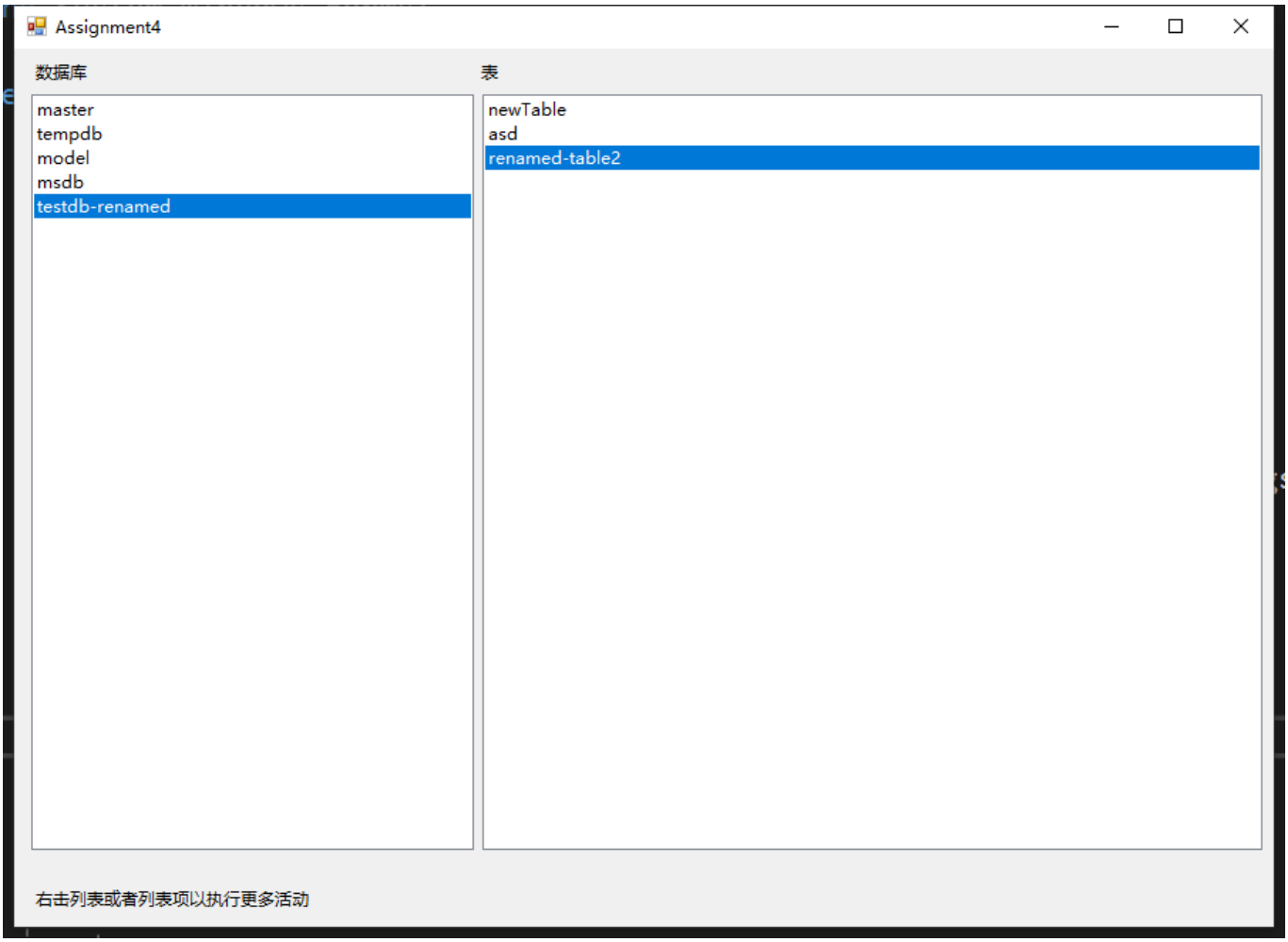
若输入表信息有误，将会弹出对应错误提示


AddTableForm
表名
add
列
列
类型
123
what
出错了!
"123"附近有语法错误。
确定
新增列
保存
取消

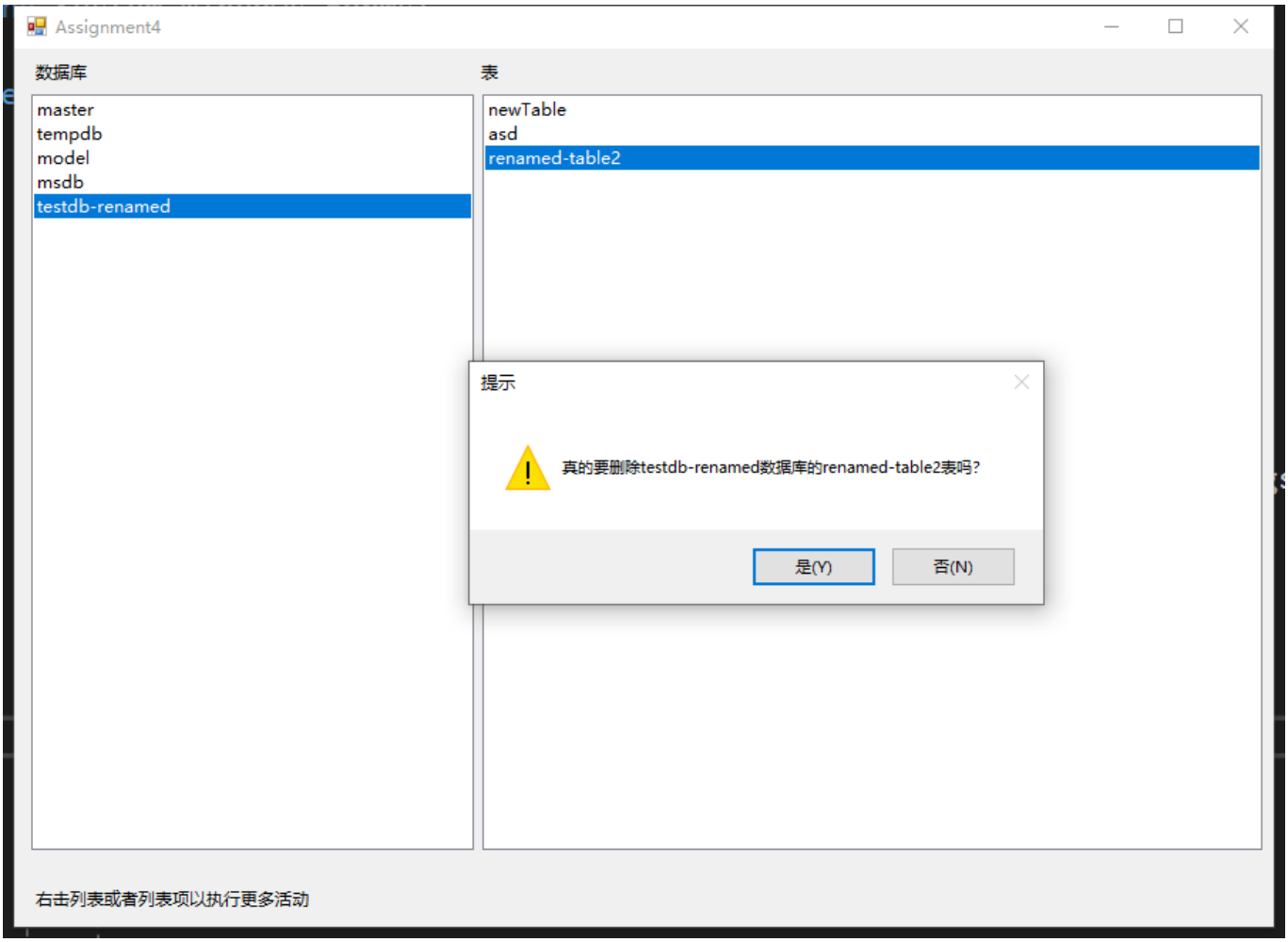
重命名表


Assignment4
数据库
表
master
tempdb
model
msdb
testdb-renamed
newTable
asd
table2
输入
请输入新表名，原名：table2
确定
取消
右击列表或者列表项以执行更多活动

重命名成功

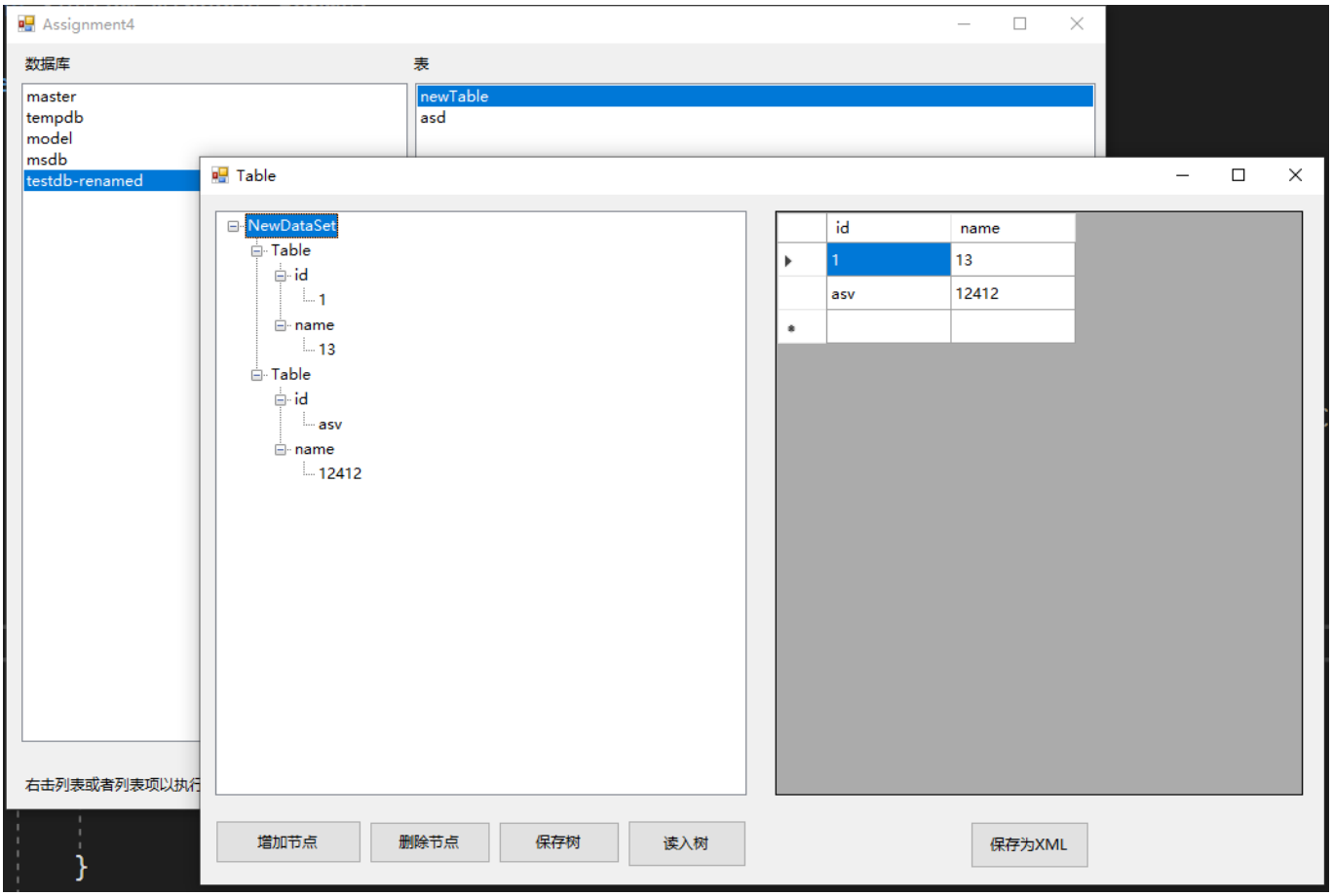

Assignment4
数据库
表
master
tempdb
model
msdb
testdb-renamed
newTable
asd
renamed-table2
右击列表或者列表项以执行更多活动

删除表



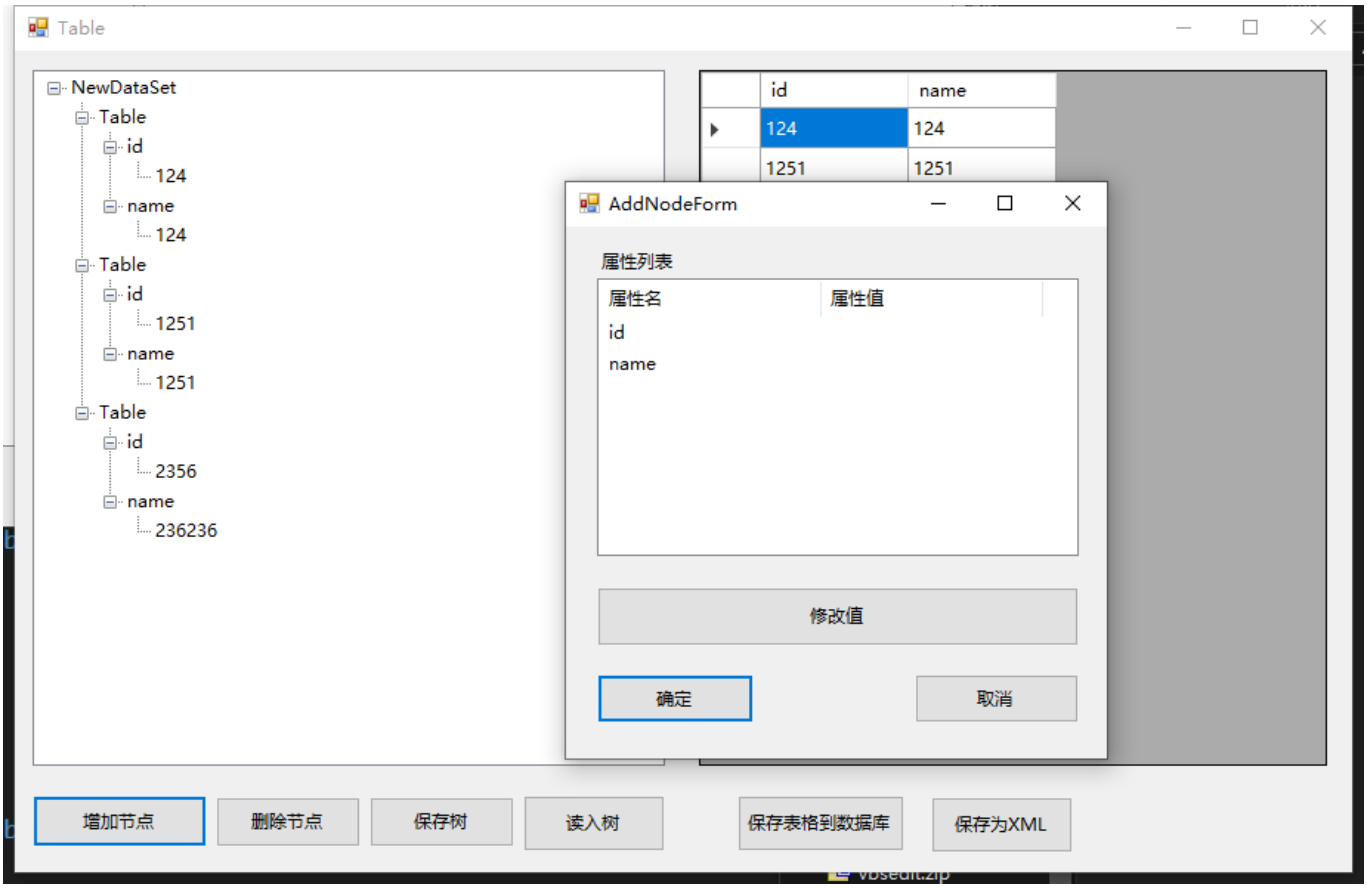
选择表

双击，或者选中表后右键单击列表选择“选择表”，进入新的界面

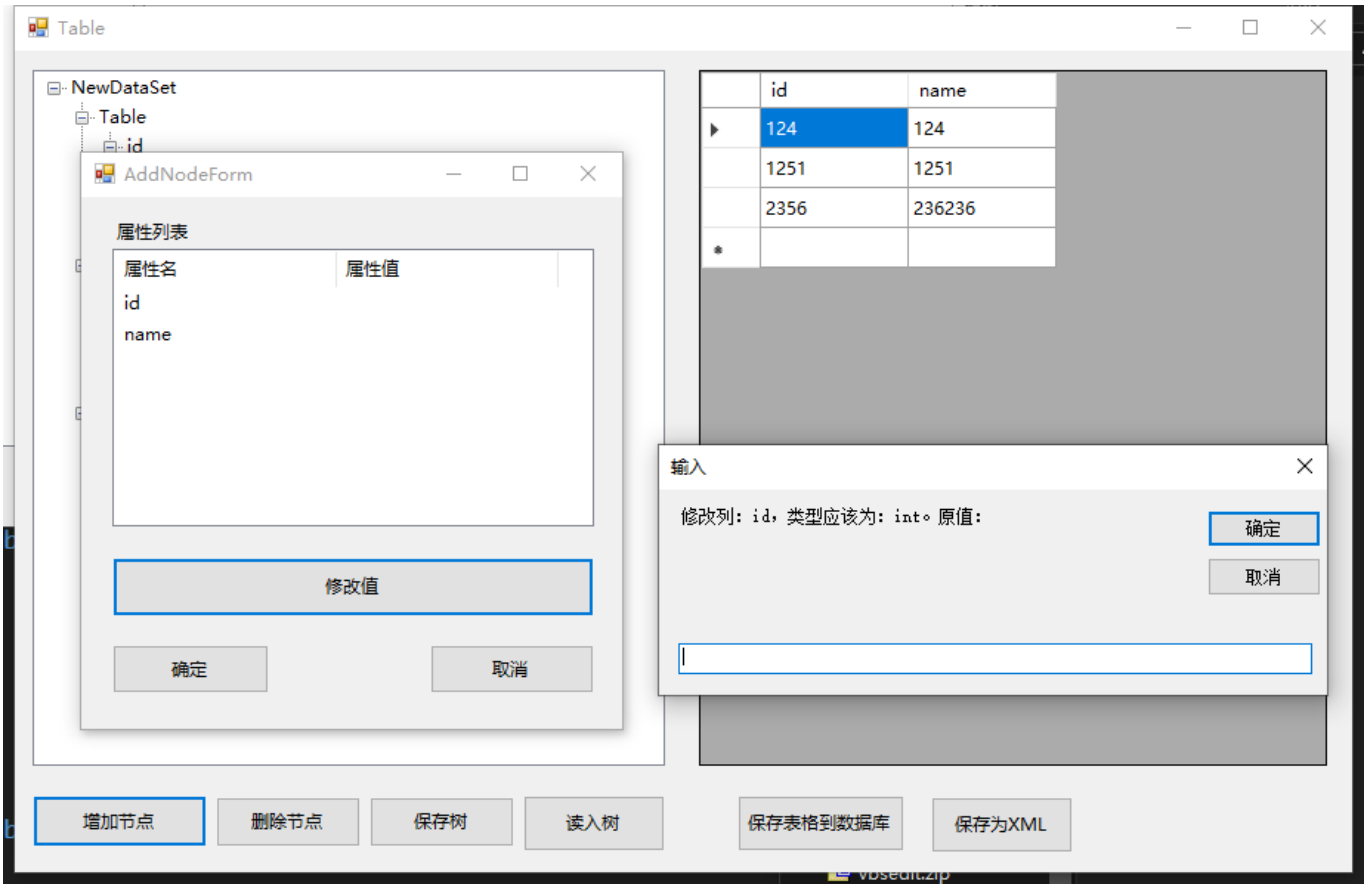


树

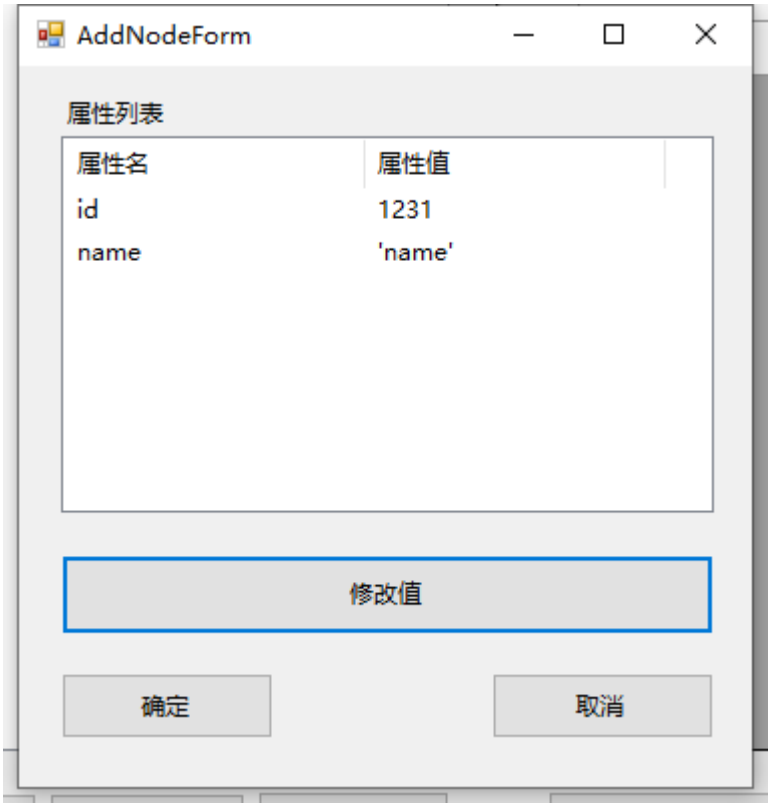
点击增加节点，进入节点增加窗口

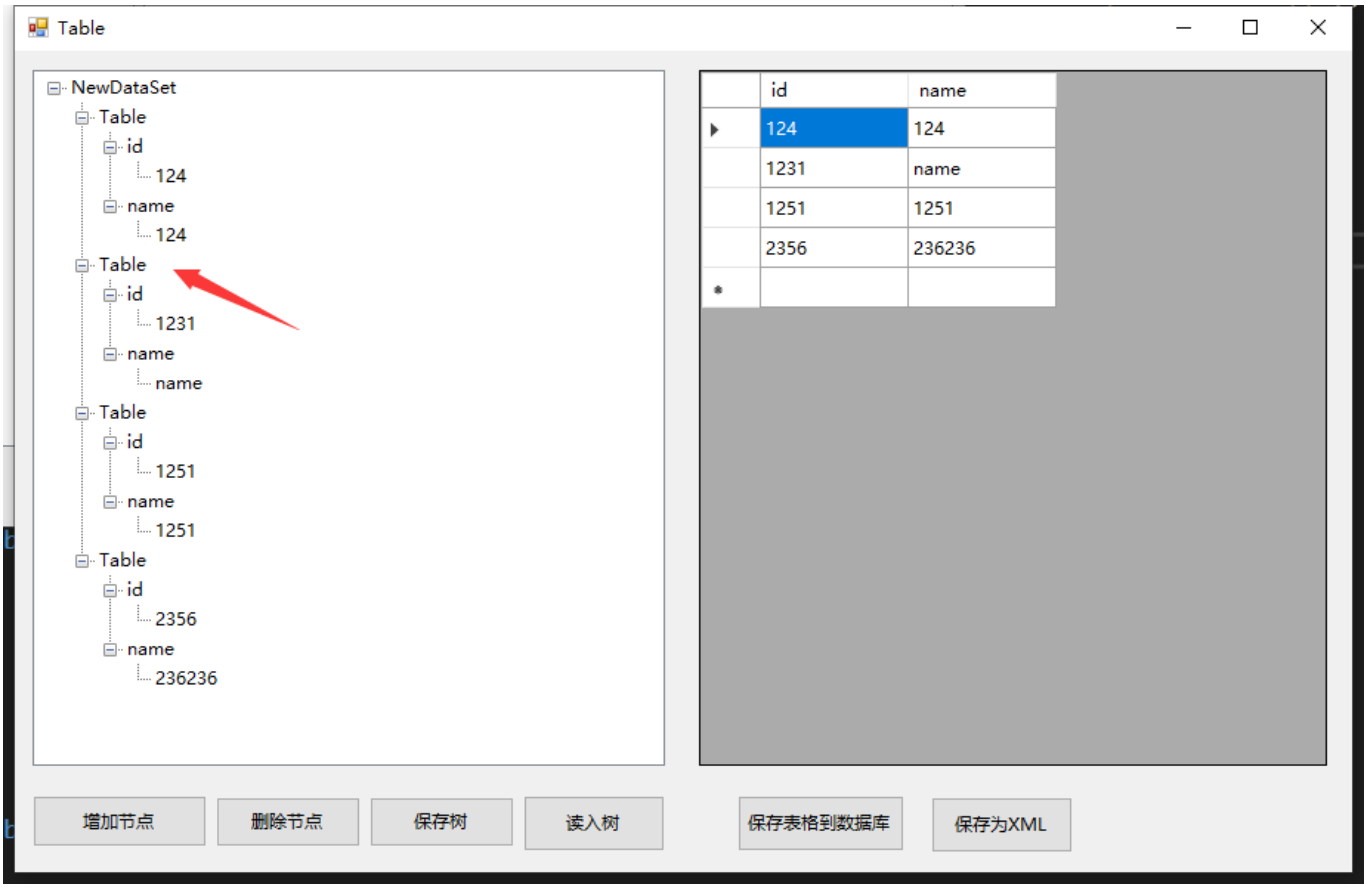


选中一个属性名，点击修改值，可以修改对应列的值

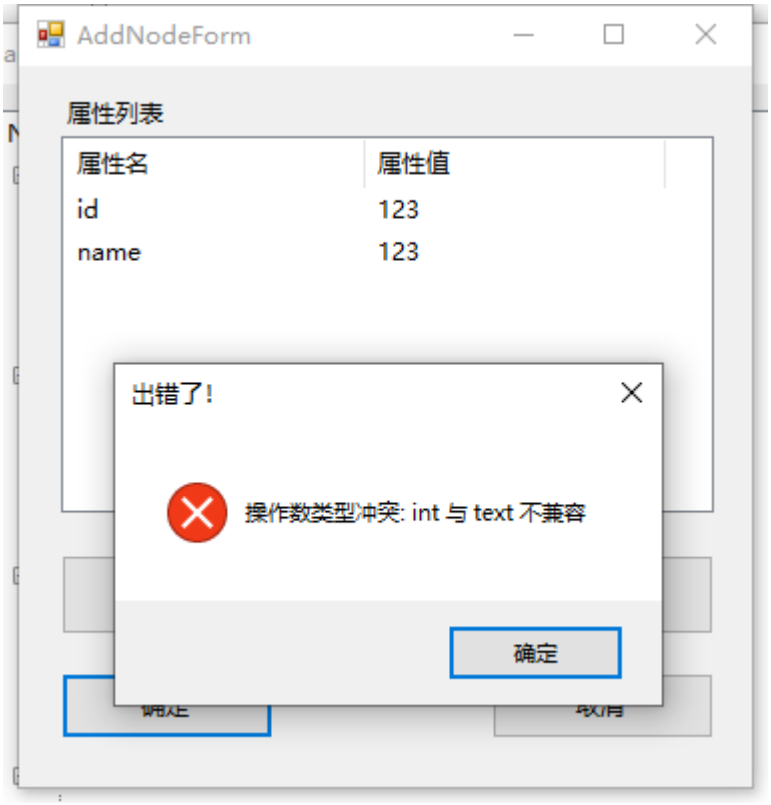


输入各个列的信息后点击保存，节点增加成功。

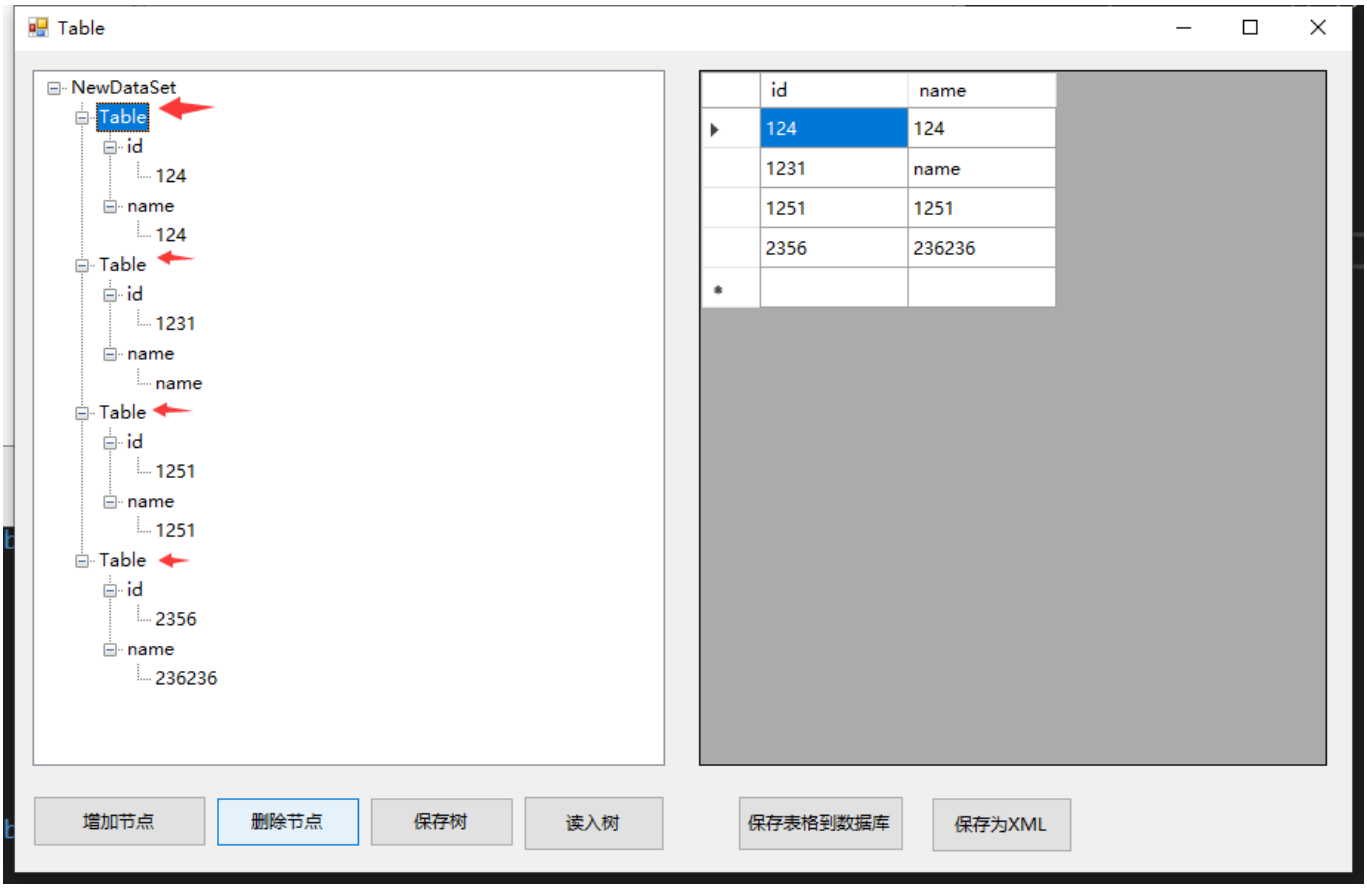




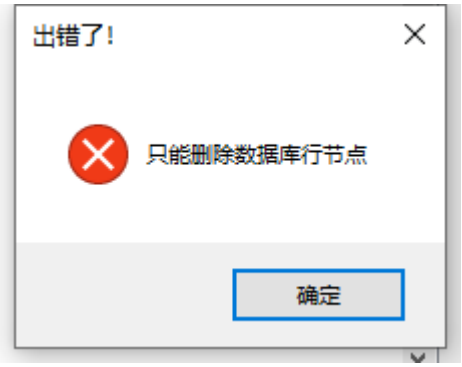
当输入列的值不符对应类型，会报错



选择一个数据库行对应的节点，点击删除节点，可删除节点

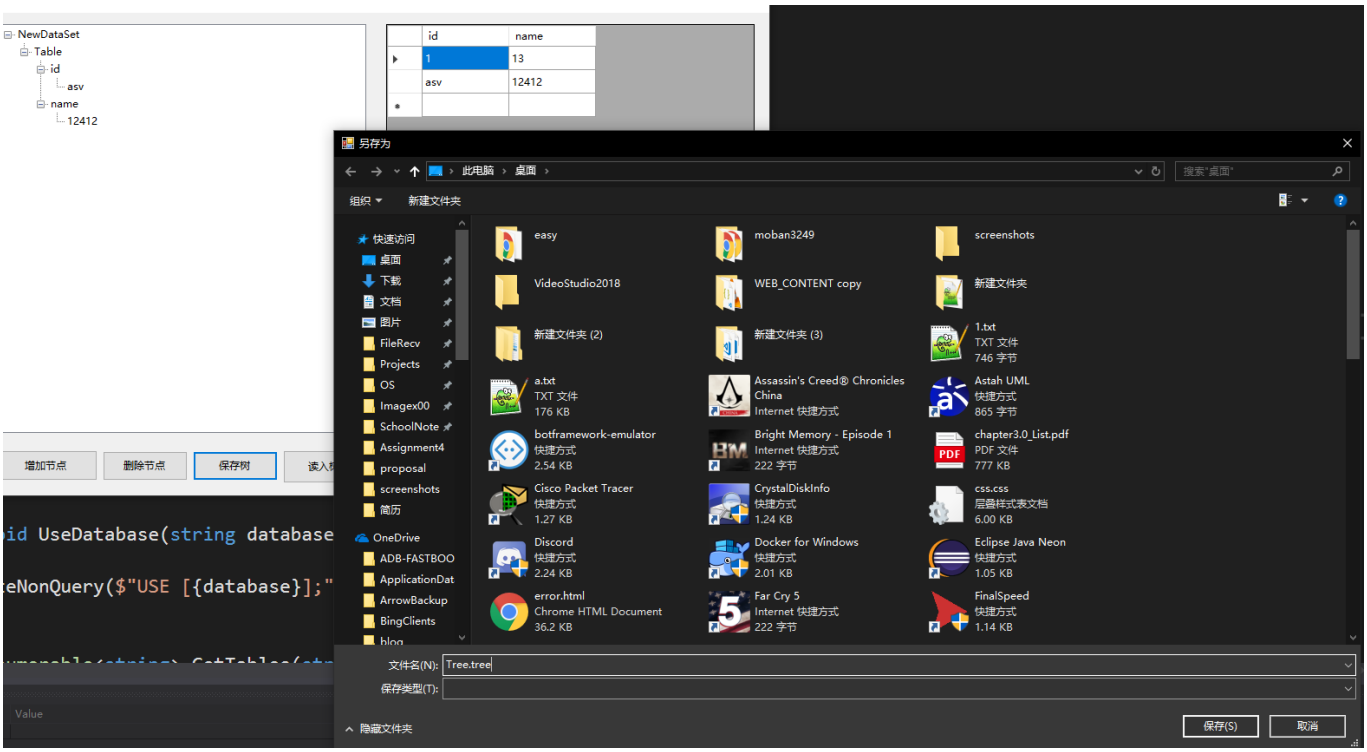


若点击了其他节点，再选择删除节点，会报错

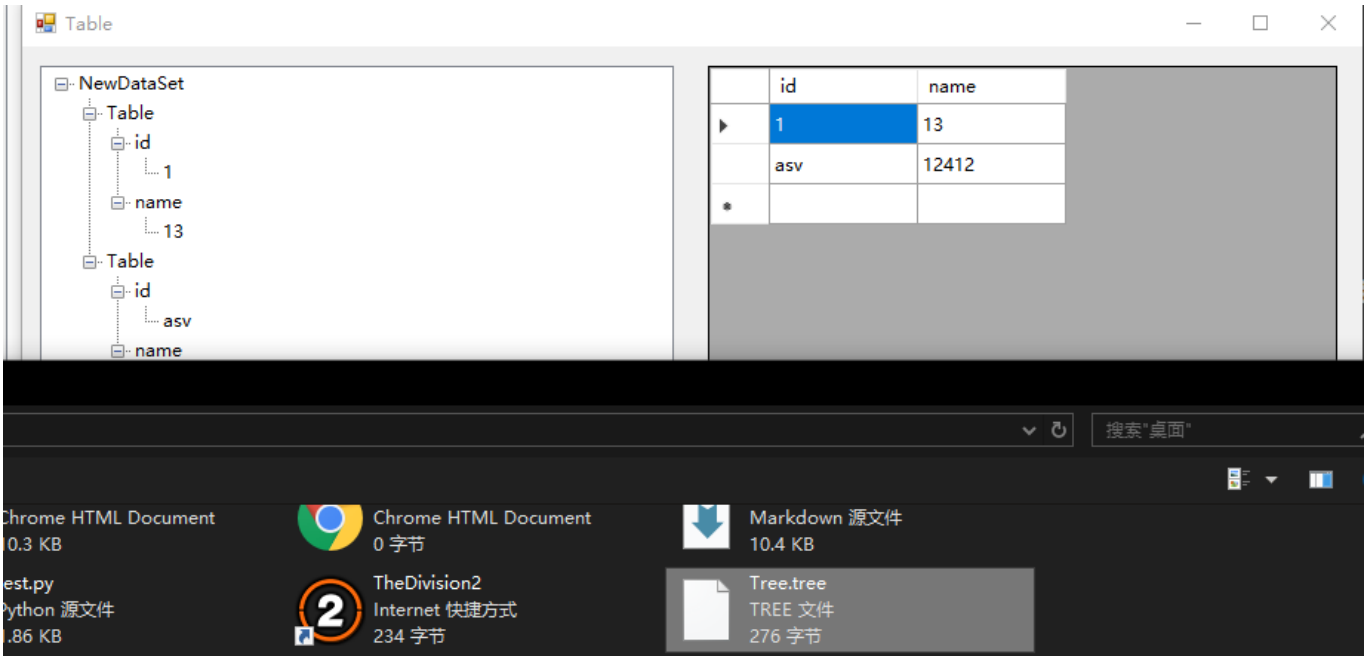


增加节点或者删除节点后，右边DataGrid会自动刷新，使数据和数据库保持统一。

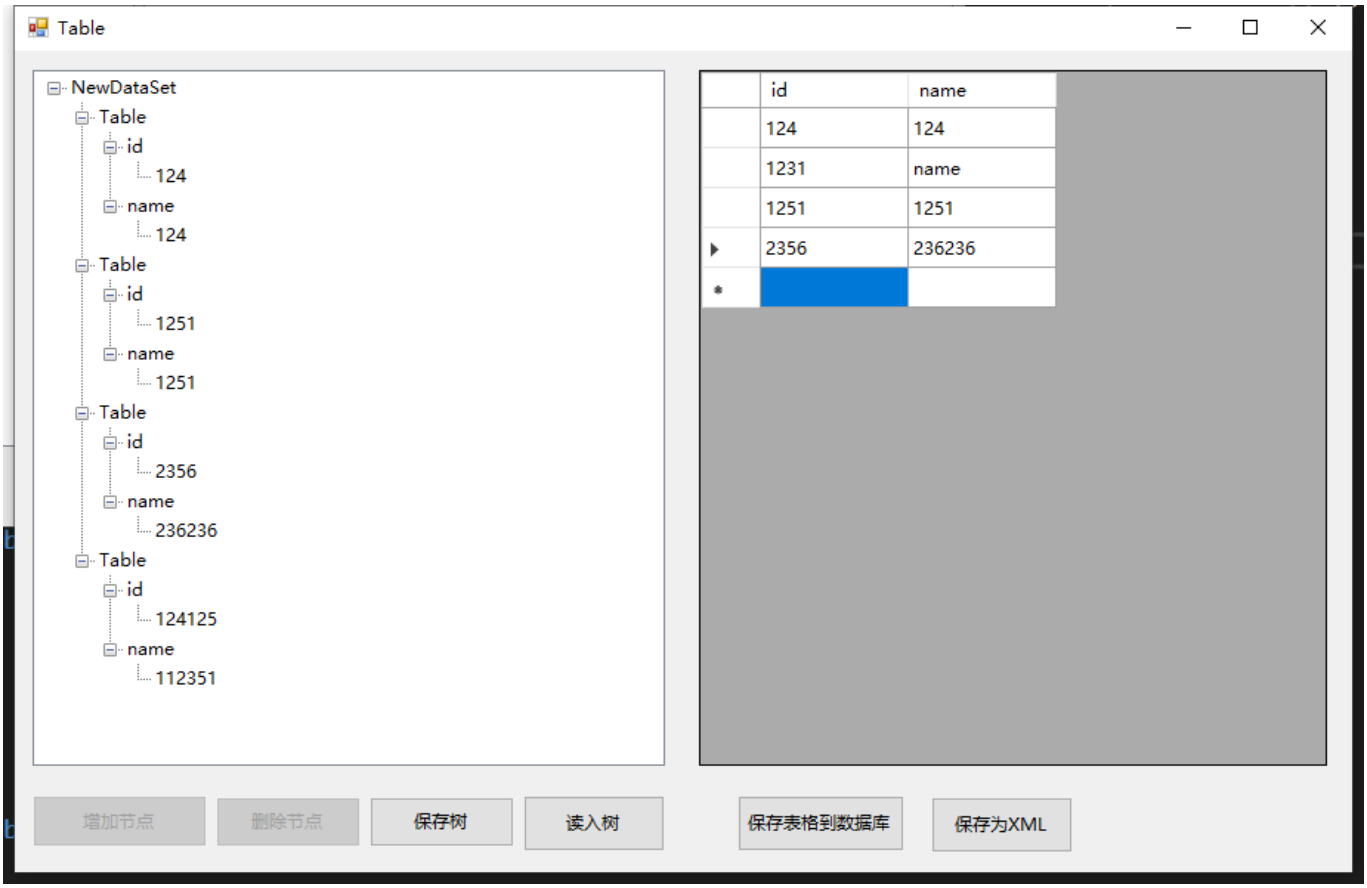
点击保存树，选择目标路径和文件，可将树信息保存



点击读入树，可读入之前保存的树的信息



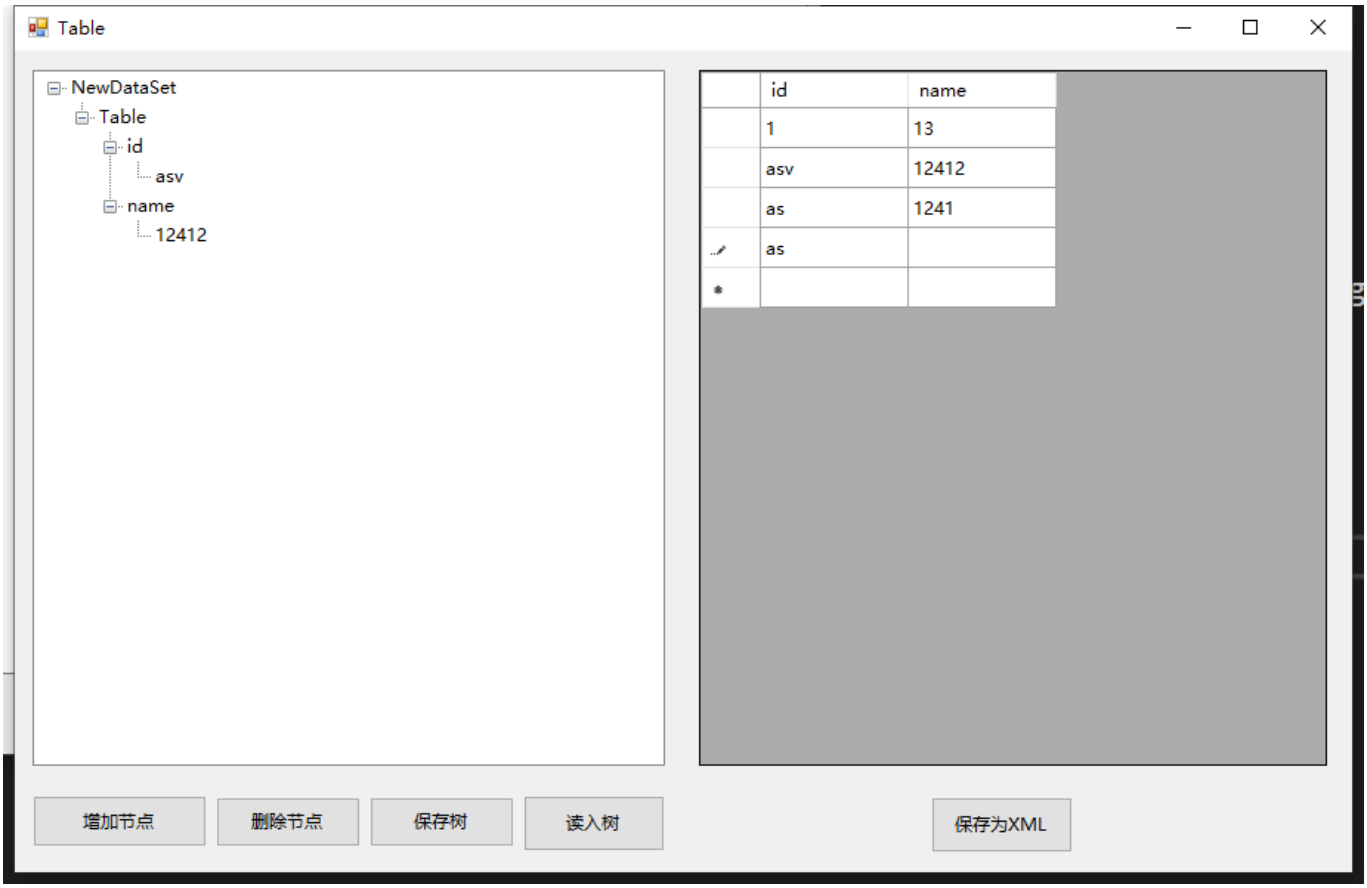
之前的树替换了现在的树



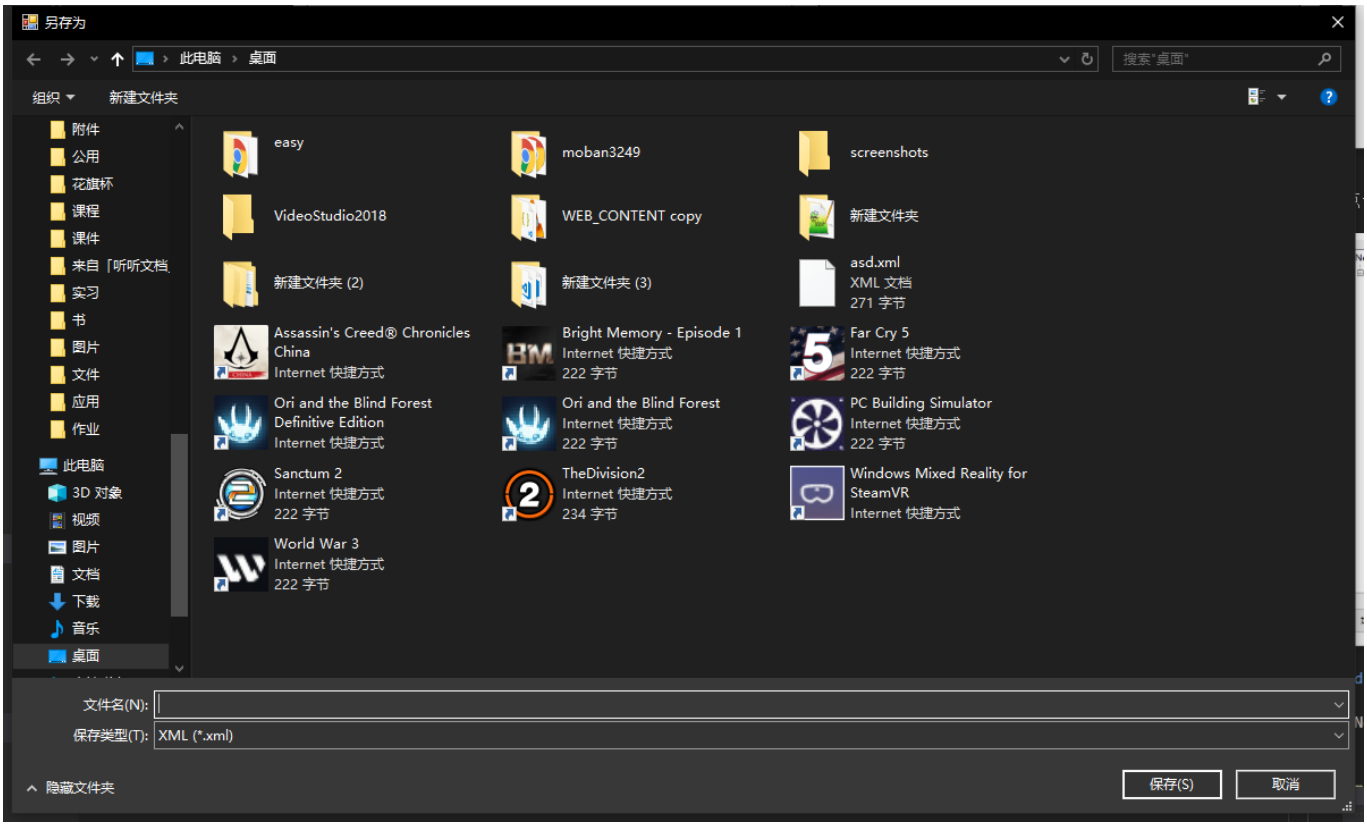
读入树之后，将会禁止增加节点和删除节点功能。

DataGrid

可以自由在右侧修改表的信息



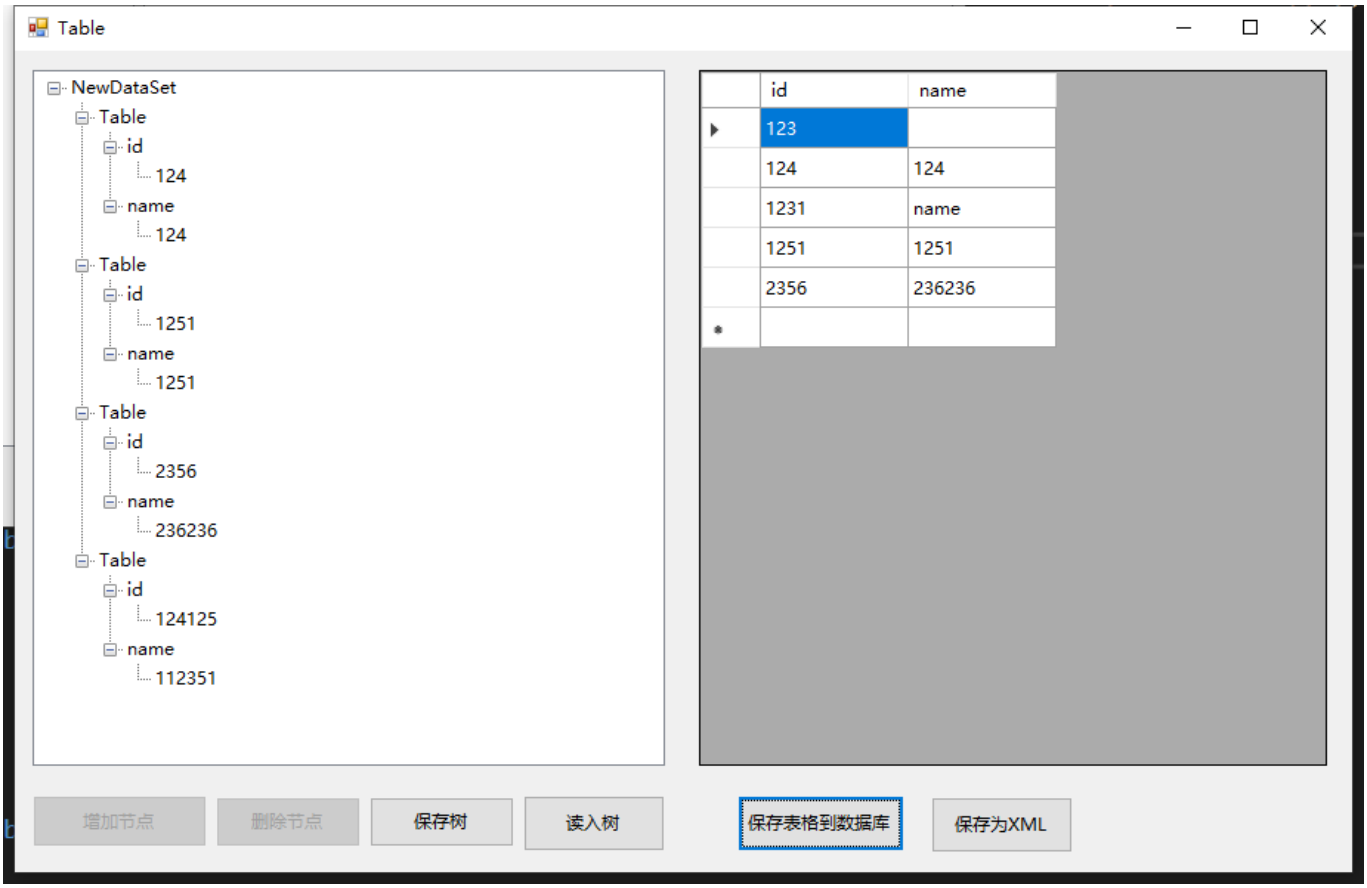
点击保存XML可以将表信息保存到本地



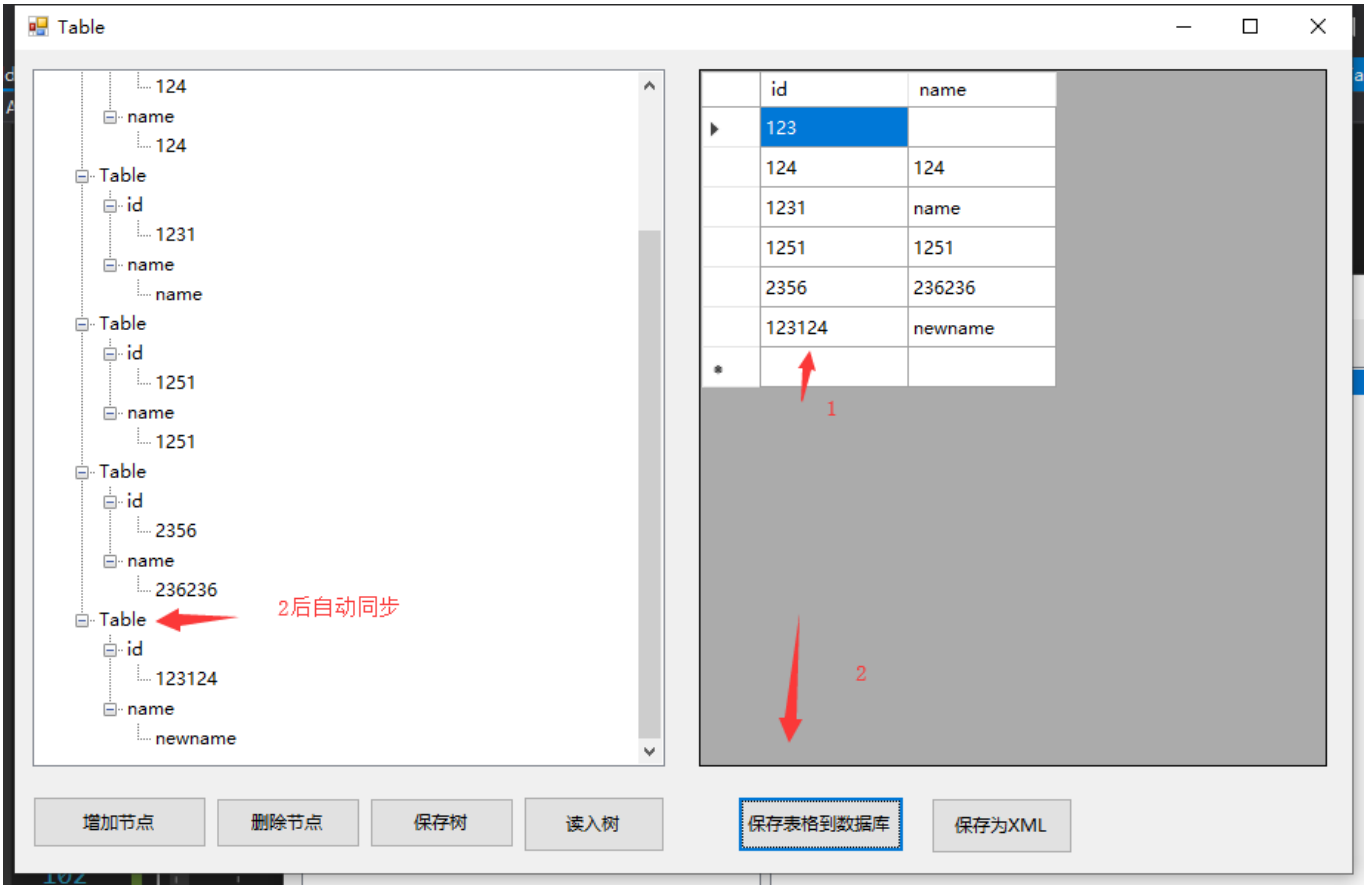
XML信息

```
<NewDataSet>
  <Table>
    <id>1          </id>
    <name>13</name>
  </Table>
  <Table>
    <id>asv        </id>
    <name>12412</name>
  </Table>
  <Table>
    <id>as</id>
    <name>1241</name>
  </Table>
  <Table>
    <id>as</id>
  </Table>
</NewDataSet>
```

修改DataGrid的数据后，点击“保存表格到数据库”，可将表格信息和数据库进行同步。



若左边树并没有从文件读入树，那么左边的树也会自动刷新以匹配数据库数据。



运行说明

1. 本系统支持的数据库的为SQL Server，已经在SQL Server Express 2017上进行测试。若没有安装SQL Server，请先安装并正确配置
2. 使用Visual Studio 2017打开本项目
3. 修改`app.config`里的connectionString，确保初始Database未设置或者设置为master

```
<connectionStrings>
  <add name="default" providerName="System.Data.SqlClient"
       connectionString="Server=MSI-
VICCRUBS\SQLEXPRESS;Database=master;Trusted_Connection=True;Integrated
Security=SSPI;" />
</connectionStrings>
```

4. 运行程序！

实现说明

这里针对几个重要功能的实现进行说明。

数据库操作

数据库相关操作是通过`DataService`对象实现的。

在主Form加载的时候会加载`DataService`对象，在这个对象构造时会使用`SqlConnection`，从`app.config`配置文件中读取连接字符串，连接数据库并打开连接。在整个软件生命周期中只会进行一次连接。

Services\DatabaseService.cs

```
public DatabaseService()
{
    conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["default"].ConnectionString);
    conn.Open();
}
```

这个对象连接上数据库后，可以通过对象的实例方法进行所有数据库操作。

获得数据库信息和表信息是通过`GetSchema`方法获得的。

```
public IEnumerable<string> GetTables(string database)
{
    UseDatabase(database); // 即运行 USE [database] 切换到目标数据库
    var tables = conn.GetSchema("Tables");
    return tables.AsEnumerable()
        .Select(r => r.Field<string>("table_name"));

}
```

获得表的信息，是通过`select * from table`后，将数据填入DataSet获得的。

```
public DataSet GetDataSet(string database, string table)
{
    UseDatabase(database);
    var adapter = new SqlDataAdapter($"SELECT * FROM [{table}]", conn);

    var dataSet = new DataSet();
    adapter.Fill(dataSet);
    return dataSet;

}
```

树的显示

本软件中，数据树的表示是通过将上述获得的DataSet转成GetXML后，使用和作业3相同的算法，将XML转换成树显示的。

```
private void SetTreeViewData(DataSet dataSet)
{
    string xml = dataSet.GetXml();
    document.LoadXml(xml);
    UpdateList(); // 使用document刷新树
}
```

树的持久化

本软件中，树的持久化的过程分为以下几步

1. 将树对应的`XmlDocument`保存进一个自定义类型`SerializableXmlDocument`中
2. 使用`BinaryFormatter`序列化`SerializableXmlDocument`对象，并下写入文件

反序列化为序列化过程反过来，即

1. 读入文件，使用`BinaryFormatter`反序列化`SerializableXmlDocument`对象
2. 从`SerializableXmlDocument`对象中拿到XmlDocument对象，并进行显示

SerializableXmlDocument类定义如下。

为了将XmlDocument序列化，此类实现了`ISerializable`接口，在`GetObjectData`中将XmlDocument转换为Xml字符串，并保存进info中。

为了正确反序列化，此类还增加了特殊的构造方法`SerializableXmlDocument(SerializationInfo, StreamingContext)`。在反序列化时，这个特殊的构造方法会被调用。在这个方法中，首先拿到序列化时生成的Xml字符串，然后使用此字符串还原XmlDocument。

SerializableXmlDocument.cs

```
[Serializable]
public class SerializableXmlDocument : ISerializable
{
    private const string KEY = "xml";
    public XmlDocument Document { get; set; }

    internal SerializableXmlDocument(SerializationInfo si, StreamingContext
context)
    {
        var xmlString = si.GetString(KEY) as string;
        Document = new XmlDocument();
        Document.LoadXml(xmlString);
    }

    public SerializableXmlDocument(XmlDocument document)
    {
        Document = document;
    }

    public void GetObjectData(SerializationInfo info, StreamingContext context)
    {
        using (StringWriter sw = new StringWriter())
        {
            using (XmlTextWriter tx = new XmlTextWriter(sw))
            {
                Document.WriteTo(tx);
                string strXmlText = sw.ToString();
                info.AddValue(KEY, strXmlText);
            }
        }
    }
}
```

DataGrid数据绑定

使用DataSet获得数据的一个巨大的好处时可以将DataGrid可以直接和DataSet绑定，支持显示和编辑数据。所以在表信息窗口初始化时，系统会获得表数据的DataSet并将其绑定到DataGrid上。这样就很方便的做到了数据库的显示和编辑。

```
var dataSet = databaseService.GetDataSet(database, table);
dgData.DataSource = dataSet;
dgData.DataMember = "Table";
```

DataGrid数据到XML

由于DataGrid使用DataSet作为数据源，而DataSet自己就支持将数据转换为XML字符串。所以将DataGrid数据保存到XML只需要其对应DataSet的GetXml方法，然后将XML字符串写入文件即可。

```
public void SaveToXml(DataSet dataSet, string path)
{
    var xml = dataSet.GetXml();
    using (var writer = new StreamWriter(path, false))
    {
        writer.Write(xml);
    }
}
```

TreeView同步到数据库

增加节点时，将会将输入的各个列的值集合成一个Dictionary<string, string>，

```
var dict = new Dictionary<string, string>();
foreach (ListViewItem pair in lvAttributes.Items)
{
    dict.Add(pair.SubItems[0].Text, pair.SubItems[1].Text);
}
```

然后根据这些数据，拼接成一个INSERT语句，交由数据库执行。

```
ExecuteNonQuery($"INSERT INTO [{table}] ({string.Join(",",values.Keys.Select((x)
=> $"[{x}]"))}) VALUES ({string.Join(",", values.Values)});");
```

删除节点时，会首先在XmlDocument上删除这个节点，

```
selectedNode.ParentNode.RemoveChild(selectedNode);
```

然后清空对应表数据，再将XmlDocument转换成DataSet，然后同步进数据库。

```
var dataSet = dgData.DataSource as DataSet;

var xmlReader = new XmlNodeReader(document);

dataSet.ReadXml(xmlReader);

UpdateDatabase(dataSet, true);
```

DataGrid同步到数据库

在获取数据的时候，将会获得DataAdapter。DataGrid要同步到数据库时，只需要获得DataGrid对应的DataSet，然后使用DataAdapter，调用其Update方法，将DataGrid传入，即可将DataGrid的数据同步到数据

库。

```
private void btnSaveGridToDb_Click(object sender, EventArgs e)
{
    if (dataAdapter == null)
    {
        return;
    }

    dgData.EndEdit();

    UpdateDatabase(dgData.DataSource as DataSet, false);

    UpdateViews();
}

private void UpdateDatabase(DataSet dataSet, bool deletedAll)
{
    if (deletedAll)
    {
        databaseService.DeleteAll(database, table);

    }
    dataAdapter.Update(dataSet);

}
```