拓扑检查技术文档

拓扑检查

组员

邱洪发、秦卫付、周青鑫、杨安邦、王嘉奇

指导老师

张丰

目录

拓扑是结合了一组编辑工具和技术的规则集合，它使地理数据库能够更准确地构建几何关系模型。

ArcGIS 通过一组用来定义要素共享地理空间方式的规则和一组用来处理在集成方式下共享几何的要素的编辑工具来实施拓扑。拓扑以一种或多种关系的形式保存在地理数据库中，这些关系定义一个或多个要素类中的要素共享几何的方式。参与构建拓扑的要素仍是简单要素类，拓扑不会修改要素类的定义，而是用于描述要素的空间关联方式。

一、准备

1.1 目的

使用给定的面 shp 数据进行拓扑检查，要求实现以下几个功能：

（1）图层拓扑检查，检查的规则包括：1 图层内面无自相交，2 图层内面与面无重叠，3 图层内面与面无缝隙

（2） 拓扑错误列表显示，包括拓扑错误类型，产生错误相关要素的 ID

（3） 点击任一拓扑错误，图上能够突出显示拓扑错误。

1.2arcgis 中的实现步骤与效果

创建拓扑

（1）打开 ArcCatalog。

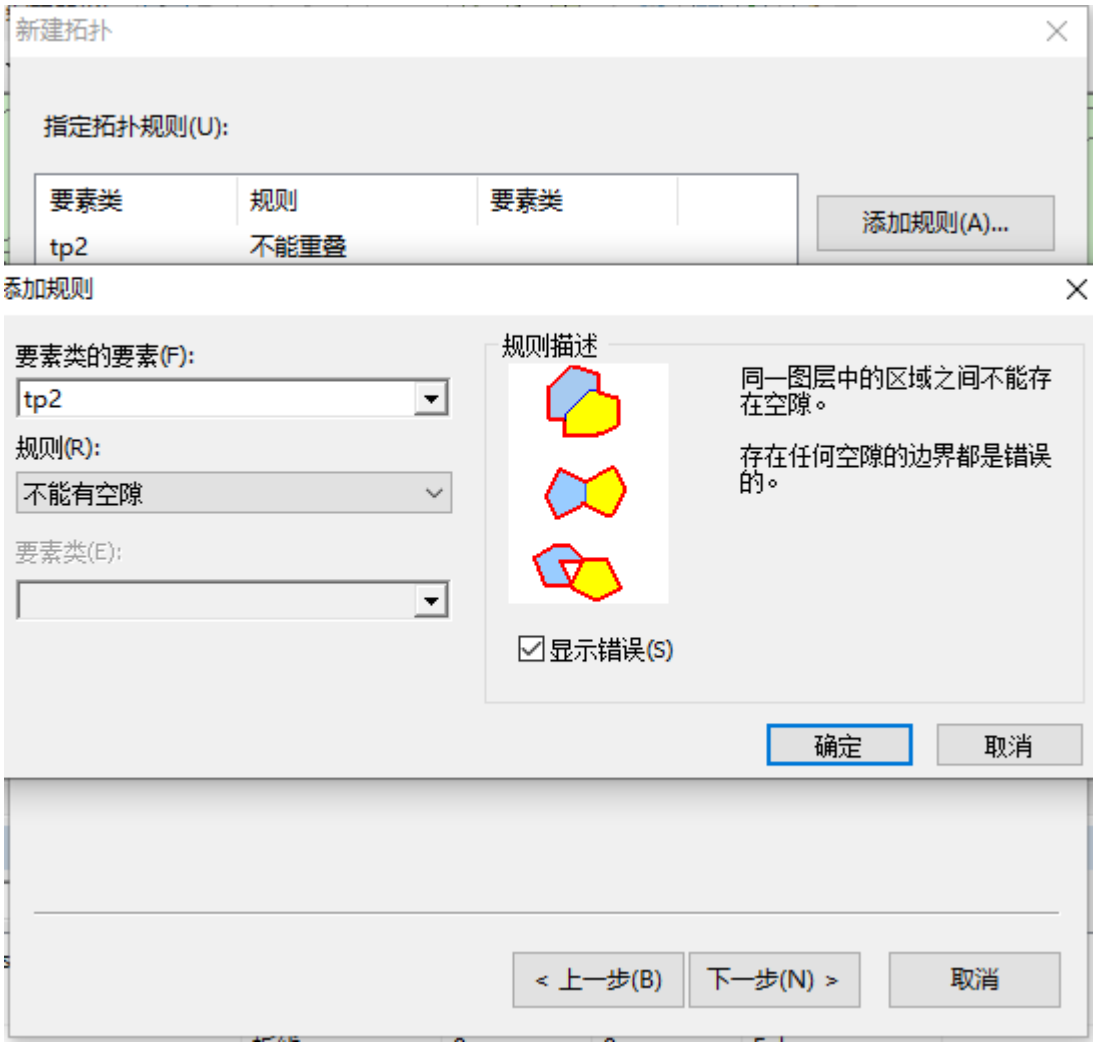
（2）点击连接到文件地理数据库的要素集，在右侧空白处右键点击，弹出菜单列表，点击“新建”—“拓扑”

（3）弹出“创建拓扑”界面，点击下一步，输入拓扑名称和拓扑容差值，点击下一步。

（4）选择要参与拓扑的图层，点击下一步，设置各图层的等级，点击下一步

（5）添加拓扑规则

（6）ArcGIS 提供了很多的拓扑规则以供选择，比如“面要素间不能有空隙”、“面要素不能重叠”等，添加好之后，点击下一步，显示拓扑的摘要信息，点击完成。



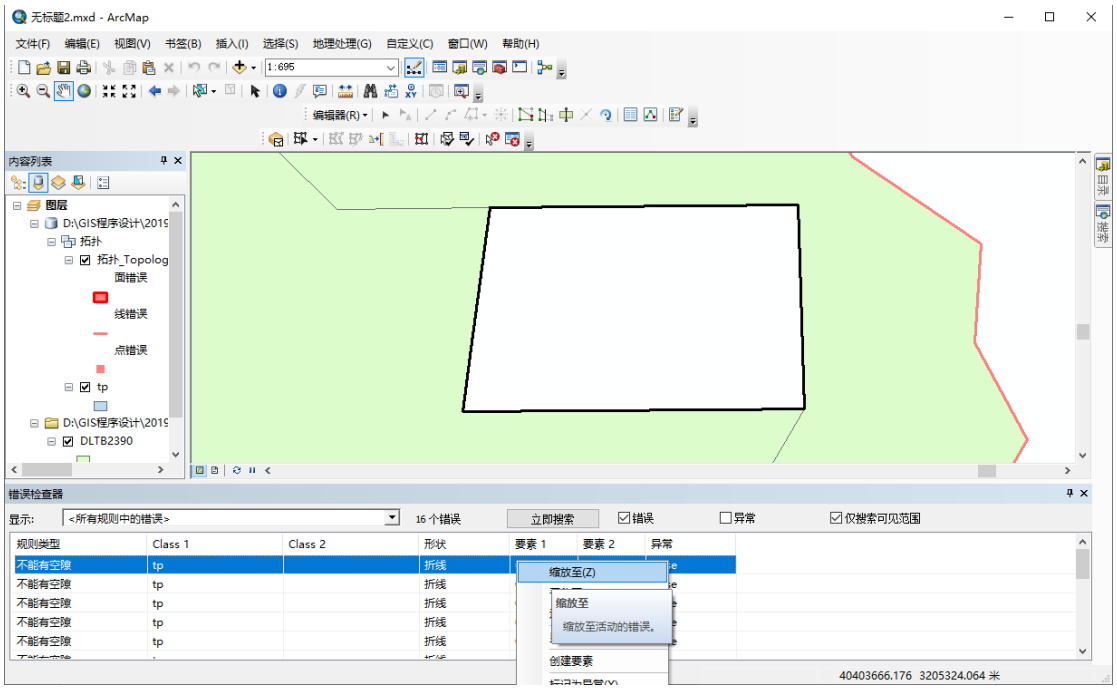
（7）等待拓扑完成，完成后会提示是否要进行验证，点击是，等待拓扑验证完成，拓扑就建好了。

（8）将拓扑手动加载到地图中显示。

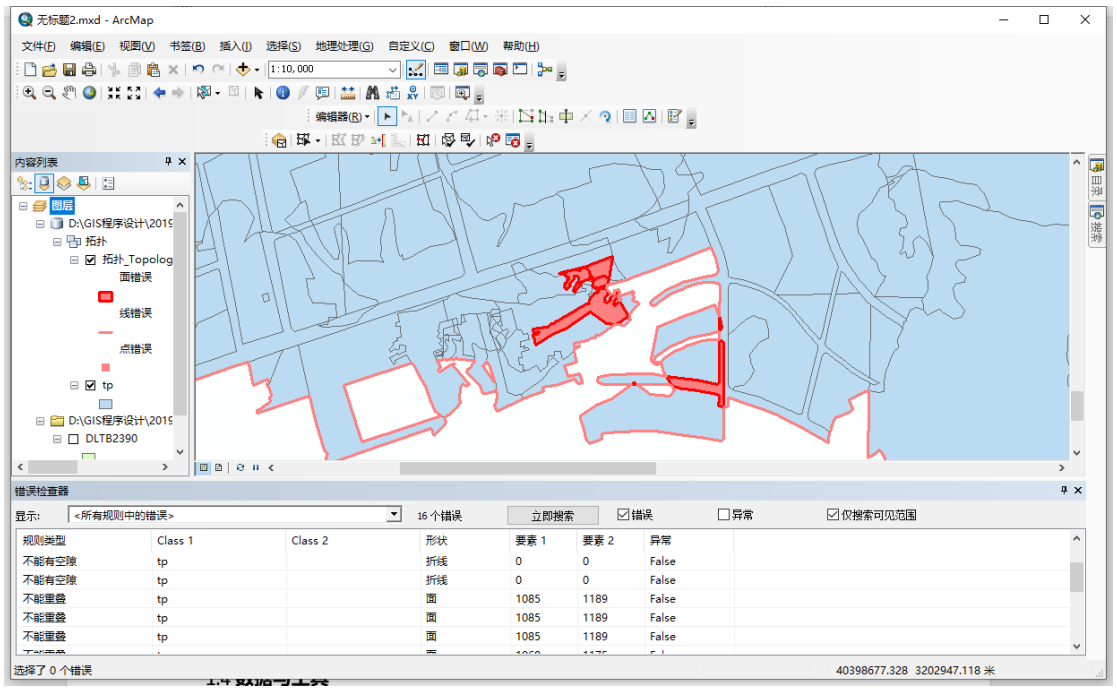
读取拓扑

（1） 在 arcgis 中选择编辑进行拓扑检查的要素类，此时拓扑工具条可用，点击打开错误检查器，搜索当前拓扑中的错误信息。

（2） 右键点击某个错误要素即可缩放到相应要素。



（3）对参与拓扑的图层进行编辑后点击工具条上的“验证当前范围内的拓扑”按钮，更新拓扑信息，点击搜索，出现新的拓扑错误。



（4）arcgis 在搜索拓扑错误时支持按不同错误类型进行搜索、仅搜索可见范围。在进行拓扑验证时支持验证当前范围中的拓扑、框选范围验证。

1.3ArcEngine 中的实现想法

ArcEngine 中拓扑检查有以下两种常用的方法：调用 GP 工具（CheckGeometry）检查数据的几何、调用 ITopologicalOperator5 接口进行拓扑检查。

为更好地理解拓扑检查的流程，我们采用通过接口创建拓扑进行拓扑检查的方法。参考 ArcGIS 中在 ArcCatalog 内创建拓扑的方法，我们制定了以下几个步骤：

（1） 创建数据库。我们选择个人地理数据库（mdb），在用户指定的位置创建数据库。

（2） 创建数据集。拓扑的创建需要在数据集中，数据集（DataSet）存放在第一步创建好的 mdb 中。

（3） 添加要素类。在创建好的数据集中导入我们需要进行拓扑分析的 shp 文件。

（4） 创建拓扑。打开第二步创建好的数据集，我们需要在该数据集下创建拓扑。拓扑的创建我们简化为设定参与拓扑的要素类、添加拓扑规则、完成拓扑创建、验证拓扑几个步骤。

参考 ArcGIS 中读取拓扑错误信息以及验证拓扑的操作，我们计划实现以下功能：

（1） 读取拓扑结果。打开已经创建好的拓扑，搜索指定错误类型、指定范围内的错误信息，并以表格形式呈现。

（2） 显示单个拓扑错误。支持在错误信息表格上进行操作，单击错误信息高亮错误要素、右键支持缩放到该要素。

（3） 实时拓扑验证。支持在参与拓扑的要素类被编辑后，重新验证拓扑，并更新拓扑错误信息。或者用户指定验证某一范围内的拓扑，更新拓扑信息。

具体的实现方法以及涉及的接口与函数将在下面的文档中详细介绍。

1.4 数据与工具

操作环境：win10_64 位操作系统。

软件工具：VS 2015(2017)、ArcGIS 10.2 标准版套件。

编程语言：C#。

数据：本次参与拓扑检查的数据为单一面要素 shp 文件“DLTB2390.shp”。

二、实践

2.1 创建拓扑前的操作

在 ArcMap 中，拓扑的创建和验证都需在地理空间数据库中进行，故在创建拓扑之前，需实现数据库（Geodatabase）的创建。而拓扑及其相关矢量要素类都存放在数据集（Feature dataset）中，所以也需要实现向数据库中添加数据集的功能。添加完数据集后，还需向数据集中导入相关要素类（Feature class）,如此才能进行创建拓扑等后续操作。所以在创建拓扑前，通常需要进行创建数据库、创建数据集和添加要素类中的几步操作。

相关组件介绍

·IworkspaceFactory：提供对创建和打开工作空间并提供工作空间工厂信息成员的访问

IWorkspaceFactory Interf... ✕ 管理内容(M)

ArcGIS Developer Help (ESRI.ArcGIS.Geodatabase)

IWorkspaceFactory Interface

Provides access to members that create and open workspaces and supply workspace factory information. **Note:** the IWorkspaceFactory interface has been superseded by IWorkspaceFactory2. Please consider using the more recent version.

Product Availability

Available with ArcGIS Engine, ArcGIS Desktop, and ArcGIS Server.

When To Use

Use **IWorkspaceFactory** when you need to create a new workspace, connect to an existing workspace or find information about a workspace.

Members

	All	Description
←	ContainsWorkspace	Indicates if parentDirectory contains a valid workspace, or is a valid file-system workspace.
←	Copy	Copies a workspace to the specified destination folder.
←	Create	Creates a new workspace specified by the directory, file name, and connection properties.
←	GetClassID	The class ID of the WorkspaceFactory.
←	GetWorkspaceName	Retrieves the workspace name of a workspace from the given list of file names.
←	IsWorkspace	True if the specified file identifies a workspace supported by the workspace factory.
←	Move	Moves a workspace to the specified destination folder.
←	Open	Opens the workspace specified by the connection properties.
←	OpenFromFile	Opens the workspace specified by the given file name.
←	ReadConnectionPropertiesFromFile	The connection properties from the specified file.
■	WorkspaceDescription	A singular or plural description of the type of workspace the workspace factory opens/creates.
■	WorkspaceType	The type of workspace the workspace factory opens/creates.

·IWorkspaceName：实现对提供工作空间名称信息的成员的访问。

IWorkspaceName Interface ✕ 管理内容(M)

ArcGIS Developer Help (ESRI.ArcGIS.Geodatabase)

IWorkspaceName Interface

Provides access to members that supply workspace name information. **Note:** the IWorkspaceName interface has been superseded by IWorkspaceName2. Please consider using the more recent version.

Product Availability

Available with ArcGIS Engine, ArcGIS Desktop, and ArcGIS Server.

When To Use

Use **IWorkspaceName** when you are browsing for workspaces and do not want the overhead of instantiating Workspace objects.

Members

	All	Description
■	BrowseName	The browse name of the WorkspaceName.
■	Category	The category of the WorkspaceName.
■	ConnectionProperties	The connection properties of the WorkspaceName.
■	PathName	The path name of the WorkspaceName.
■	Type	The type of the associated workspace.
■	WorkspaceFactory	The workspace factory of the WorkspaceName.
■	WorkspaceFactoryProgID	The ProgID of the WorkspaceName's workspace factory.

·IName: 提供对使用名称对象的成员的访问。

IName Interface 管理内容(M)

ArcGIS Developer Help (ESRI.ArcGIS.esriSystem)

IName Interface

Provides access to members that work with Name objects.

Product Availability

Available with ArcGIS Engine, ArcGIS Desktop, and ArcGIS Server.

When To Use

Use **IName** to open a workspace object.

Members

	All	Description
■—	NameString	The name string of the object.
←—	Open	Opens the object referred to by this name.

·IWorkspace：提供对具有关于工作空间信息的成员的访问。

IWorkspace Interface 管理内容(M)

ArcGIS Developer Help (ESRI.ArcGIS.Geodatabase)

IWorkspace Interface

Provides access to members that have information about the workspace.

Product Availability

Available with ArcGIS Engine, ArcGIS Desktop, and ArcGIS Server.

Members

	All	Description
■—	ConnectionProperties	The connection properties of the workspace.
■—	DatasetNames	The DatasetNames in the workspace.
■—	Datasets	The datasets in the workspace.
←—	ExecuteSQL	Executes the specified SQL statement.
←—	Exists	Checks if the workspace exists.
←—	IsDirectory	TRUE if the workspace is a file system directory.
■—	PathName	The file system full path of the workspace.
■—	Type	The Type of the Workspace.
■—	WorkspaceFactory	The factory that created the workspace.

·IFeatureWorkspace：提供对创建和打开各种类型的数据集和其他工作空间级别对象的成员的访问。

IFeatureWorkspace Interf... 管理内容(M)

ArcGIS Developer Help (ESRI.ArcGIS.Geodatabase)

IFeatureWorkspace Interface

Provides access to members that create and open various types of datasets and other workspace level objects.

Product Availability

Available with ArcGIS Engine, ArcGIS Desktop, and ArcGIS Server.

Description

The **IFeatureWorkspace**interface is used to access and manage datasets that are a key component of a feature based geodatabase; Tables and ObjectClasses, FeatureClasses, FeatureDatasets, and RelationshipClasses. All of the Open methods (such as **OpenTable**) take a dataset name as input. When working with an enterprise geodatabase, the name may be fully qualified (for example, "database.owner.tablename" or "owner.tablename") using the qualification character appropriate to the underlying database (see **ISQLSyntax**)). If the input name is not fully qualified, then it is qualified using the currently connected user for the workspace.

When working with geodatabases (personal, file or ArcSDE) the workspace keeps a running object table of instantiated datasets. Multiple calls to open an already instantiated dataset will return a reference to the already instantiated dataset.

When To Use

IFeatureWorkspace is the main interface for creating and opening objects and object classes with a workspace.

Members

	All	Description
←—	CreateFeatureClass	Creates a new standalone feature class under the workspace.
←—	CreateFeatureDataset	Creates a new feature dataset.
←—	CreateQueryDef	Create a query definition object.
←—	CreateRelationshipClass	Creates a new relationship class.
←—	CreateTable	Creates a new table.
←—	OpenFeatureClass	Opens an existing feature class.
←—	OpenFeatureDataset	Opens an existing feature dataset.
←—	OpenFeatureQuery	Opens a feature dataset containing a single feature class defined by the specified Query.
←—	OpenRelationshipClass	Opens an existing relationship class.
←—	OpenRelationshipQuery	The table of a relationship join query.
←—	OpenTable	Opens an existing table.

·IFeatureClass：提供对控制要素类的行为和属性的成员的访问。

IFeatureClass Interface 管理内容(M)

	Description
AddField	Adds a field to this object class.
AddIndex	Adds an index to this object class.
AliasName	The alias name of the object class.
AreaField	The geometry area field.
CLSID	The GUID for the COM Class (CoClass) corresponding to instances of this object class.
CreateFeature	Create a new feature, with a system assigned object ID and null property values.
CreateFeatureBuffer	Create a feature buffer that can be used with an insert cursor.
DeleteField	Deletes a field from this object class.
DeleteIndex	Deletes an index from this object class.
EXTCLSID	The GUID for the COM Class (CoClass) corresponding to the class extension for this object class.
Extension	The extension for this object class.
ExtensionProperties	The extension properties for this object class.
FeatureClassID	The unique ID for the Feature Class.
FeatureCount	The number of features selected by the specified query.
FeatureDataset	The feature dataset that contains the feature class.
FeatureType	The type of features in this feature class.
Fields	The fields collection for this object class.
FindField	The index of the field with the specified name.
GetFeature	Get the feature with the specified object ID.
GetFeatures	Get a cursor of Rows given a set of object ids.
HasOID	Indicates if the class has an object identity (OID) field.
Indexes	The indexes collection for this object class.
Insert	Returns a cursor that can be used to insert new features.
LengthField	The geometry length field.
ObjectClassID	The unique ID for the object class.
OIDFieldName	The name of the field corresponding to the OID.
RelationshipClasses	The relationship classes in which this object class participates in for the specified role.
Search	Returns an object cursor that can be used to fetch feature objects selected by the specified query.
Select	Returns a selection That contains the object ids selected by the specified query.
ShapeFieldName	The name of the default sShape field.
ShapeType	The type of the default Shape for the features in this feature class.
Update	Returns a cursor that can be used to update features selected by the specified query.

·ISpatialReference：提供对控制空间引用的成员的访问。

ISpatialReference Interface 管理内容(M)

The spatial reference is not defined when creating a new instance of a geometry. It is the developer's responsibility to define a spatial reference that makes sense for the geometry and for the operation. To achieve precise and predictable results using the geometry library, it is essential that the spatial reference of geometries within a workflow is well defined. When performing a spatial operation using two or more geometries, for example an intersection, the coordinate systems of the two geometries must be equal. If the coordinate systems of the geometries are different or undefined, the operation could produce unexpected results.

Members

	Description
Abbreviation	The abbreviated name of this spatial reference component.
Alias	The alias of this spatial reference component.
Changed	Notify this object that some of its parts have changed (parameter values, z unit, etc.).
FactoryCode	The factory code (WKID) of the spatial reference.
GetDomain	The XY domain extent.
GetFalseOriginAndUnits	Get the false origin and units.
GetMDomain	The measure domain extent.
GetMFalseOriginAndUnits	Get the measure false origin and units.
GetZDomain	The Z domain extent.
GetZFalseOriginAndUnits	Get the Z false origin and units.
HasMPrecision	Returns true when m-value precision information has been defined.
HasXYPrecision	Returns true when (x,y) precision information has been defined.
HasZPrecision	Returns true when z-value precision information has been defined.
IsPrecisionEqual	Returns TRUE when the precision information for the two spatial references is the same.
Name	The name of this spatial reference component.
Remarks	The comment string of this spatial reference component.
SetDomain	The XY domain extent.
SetFalseOriginAndUnits	Set the false origin and units.
SetMDomain	The measure domain extent.
SetMFalseOriginAndUnits	Set the measure false origin and units.
SetZDomain	The Z domain extent.
SetZFalseOriginAndUnits	Set the Z false origin and units.
ZCoordinateUnit	The unit for the Z coordinate.

·IFeatureDataset：提供在特征数据集中创建新要素类的访问。

IFeatureDataset Interface 管理内容(M)

ArcGIS Developer Help (ESRI.ArcGIS.Geodatabase)

IFeatureDataset Interface

Provides access to create a new feature class in a feature dataset.

Product Availability

Available with ArcGIS Engine, ArcGIS Desktop, and ArcGIS Server.

Members

	All	Description
	BrowseName	The browse name of the dataset.
	CanCopy	True if this dataset can be copied.
	CanDelete	True if this dataset can be deleted.
	CanRename	True if this dataset can be renamed.
	Category	The category of the dataset.
	Copy	Copies this dataset to a new dataset with the specified name.
	CreateFeatureClass	Creates a new FeatureClass in this FeatureDataset.
	Delete	Deletes this dataset.
	FullName	The associated name object.
	Name	The name of the Dataset.
	PropertySet	The set of properties for the dataset.
	Rename	Renames this Dataset.
	Subsets	Datasets contained within this dataset.
	Type	The type of the Dataset.
	Workspace	The workspace containing this dataset.

·IEnumDatasetName：提供对通过数据集名称枚举的成员的访问。

IEnumDatasetName Interf... 管理内容(M)

ArcGIS Developer Help (ESRI.ArcGIS.Geodatabase)

IEnumDatasetName Interface

Provides access to members that enumerate through Dataset Names.

Product Availability

Available with ArcGIS Engine, ArcGIS Desktop, and ArcGIS Server.

Members

	All	Description
	Next	Retrieves the next feature class in the enumeration sequence.
	Reset	Resets the enumeration sequence to the beginning.

·IDatasetName：提供对提供数据集名称信息的成员的访问。

IDatasetName Interface 管理内容(M)

ArcGIS Developer Help (ESRI.ArcGIS.Geodatabase)

IDatasetName Interface

Provides access to members that supply dataset name information. **Note:** the IDatasetName interface has been superseded by IDatasetName2. Please consider using the more recent version.

Product Availability

Available with ArcGIS Engine, ArcGIS Desktop, and ArcGIS Server.

Members

	All	Description
	Category	The category of the dataset.
	Name	The name of the dataset.
	SubsetNames	Subset names contained within this dataset name.
	Type	The type of the dataset.
	WorkspaceName	The WorkspaceName of the DatasetName.

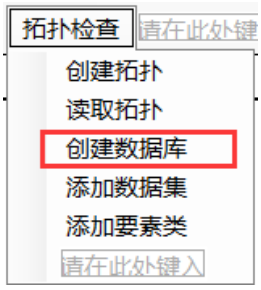
2.1.1 创建数据库

创建数据库主要通过 IWorkspaceFactory 的 Create 方法进行，程序将 public IWorkspace CreateAccessWorkspace(String databasename)封装在 DataBaseOperator 类中，实现在指定位置创建数据库的功能。

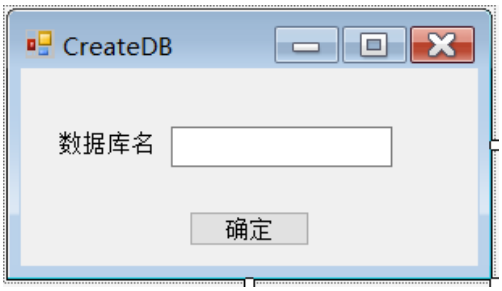
1.添加控件及窗体设计

在主窗体的【拓扑检查】菜单下添加【创建数据库】菜单，并将控件名改

为“miCreateDB”

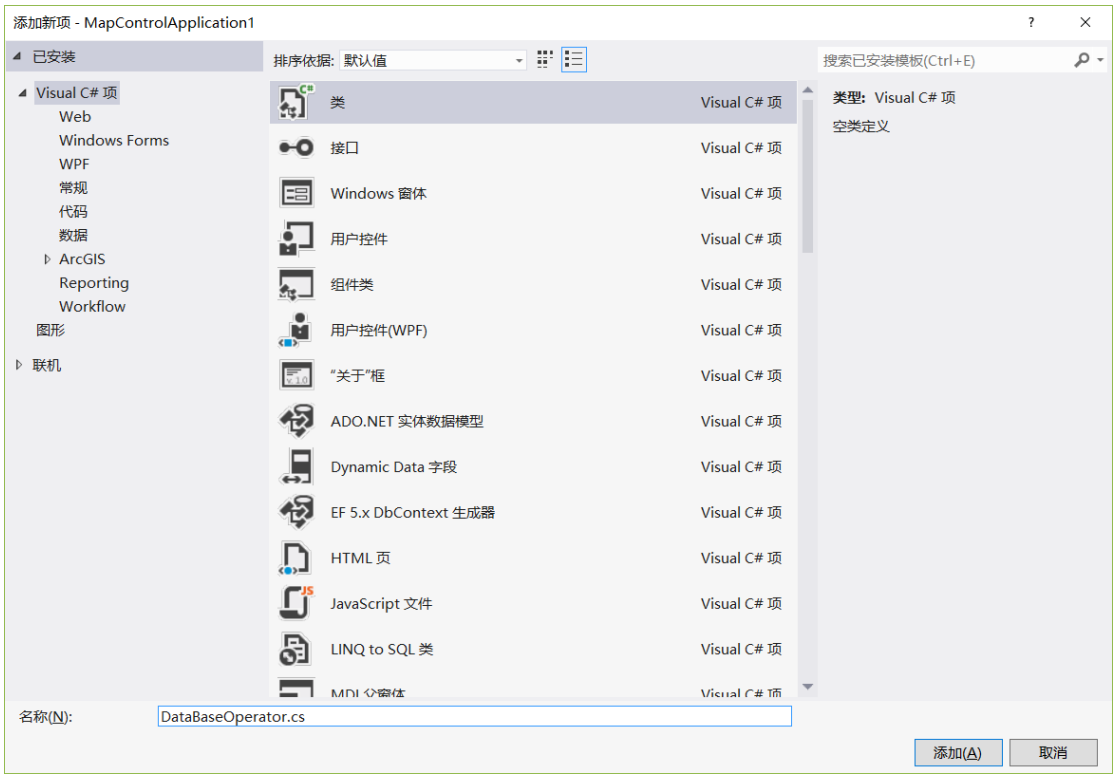


新建 Windows 窗体“CreateDB”，设计如下，其中文本框的名称为“tbxDBName”，“确定”按钮的名称为“btnSure”。



2.添加类

新建 DataBaseOperator 类，用于封装创建拓扑前的与数据操作有关的函数。



3.完善类的功能

向 DataBaseOperator 类中添加 public IWorkspace CreateAccessWorkspace(String databasename)函数，用于实现在指定位置创建数据库的功能，数据库的位置在函数中通过创建 FolderBrowserDialog 类的对象打开相关文件夹。函数代码如下：

```
public IWorkspace CreateAccessWorkspace(String databasename)
        {
            //选择数据库存储的位置
            String path = "";
            FolderBrowserDialog folderBrowserDialog = new FolderBrowserDialog();
            folderBrowserDialog.Description = "选择所要创建数据库的位置";
            folderBrowserDialog.ShowNewFolderButton = false;
```

```
            if (folderBrowserDialog.ShowDialog() == DialogResult.OK)
                path = folderBrowserDialog.SelectedPath;
            Type factoryType =
Type.GetTypeFromProgID("esriDataSourcesGDB.AccessWorkspaceFactory");
            IWorkspaceFactory workspaceFactory =
(IWorkspaceFactory)Activator.CreateInstance(factoryType);
            IWorkspaceName workspaceName = workspaceFactory.Create(path,
databasename+".mdb", null, 0);
            IName name = (IName)workspaceName;
            IWorkspace workspace = (IWorkspace)name.Open();
            return workspace;
        }
```

4.实现数据库的创建

在“CreateDB”窗体的代码中新建 MainForm 的对象，并为“确定”按钮添加响应函数：当 tbxDBName 文本框的内容不为空时，将文本框的内容传回 MainForm 中的函数 createDB（String dbName），在 createDB（String dbName）函数中调用 DataBaseOperator 类中的 CreateAccessWorkspace 函数创建数据库。

“CreateDB”窗体的代码如下：

```
public partial class CreateDB : Form
    {
        public MainForm cDB;
        public CreateDB(MainForm mf)
        {
            InitializeComponent();
```

```
            cDB = mf;
        }
        private void btnSure_Click(object sender, EventArgs e)
        {
            if (tbxDBName.Text != "" )
            {
                cDB.createDB(tbxDBName.Text);
                MessageBox.Show("数据库创建成功！");
                this.Close();
            }
            else MessageBox.Show("请输入名称！");
        }
    }
```

MainForm 中 createDB 函数代码如下：

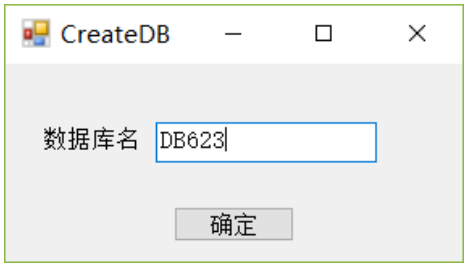
```
//创建数据库
        public void createDB(String dbName)
        {
            //以下代码为创建数据库
            DataBaseOperator dataBaseOperator = new DataBaseOperator();
            //创建数据库
            IWorkspace workspace =
dataBaseOperator.CreateAccessWorkspace(dbName);
            }
```

MainForm 中【创建数据库】按钮单击响应函数如下：

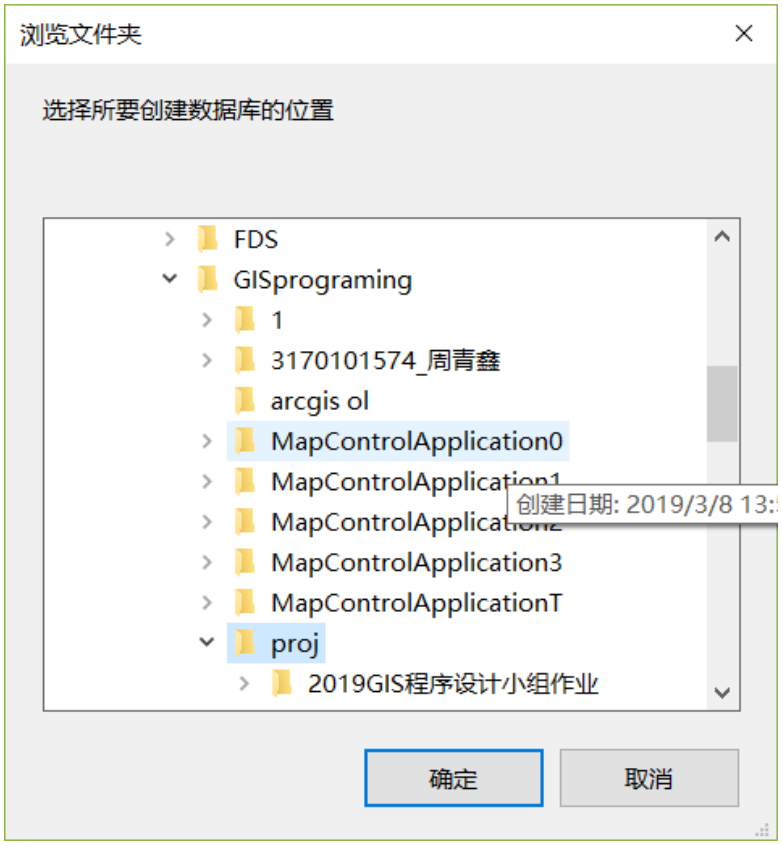
```
//调用CreateDB窗口
        private void miCreateDB_Click(object sender, EventArgs e)
        {
            CreateDB createDB = new CreateDB(this);
            createDB.Show();
           }
```

5.运行结果与异常处理

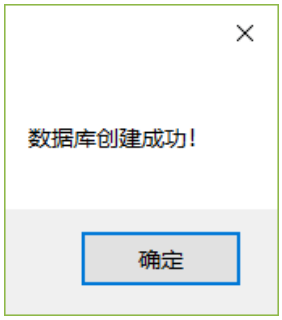
单击【拓扑检查】菜单下的【创建数据库】按钮，弹出如下对话框：



输入数据库名称并点击确定后，弹出“浏览文件夹”对话框，可选择要创建数据库的位置

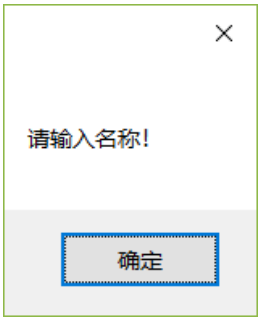


单击【确定】后数据库创建成功，弹出如下消息提示：



在文件夹中可以看到刚创建的数据库“db643”（也可在 ArcCatalog 中查看）

若在创建数据库时为输入数据库名就点击【确定】按钮，则会弹出以下消息提示框：

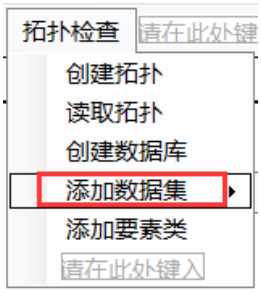


2.1.2 创建数据集

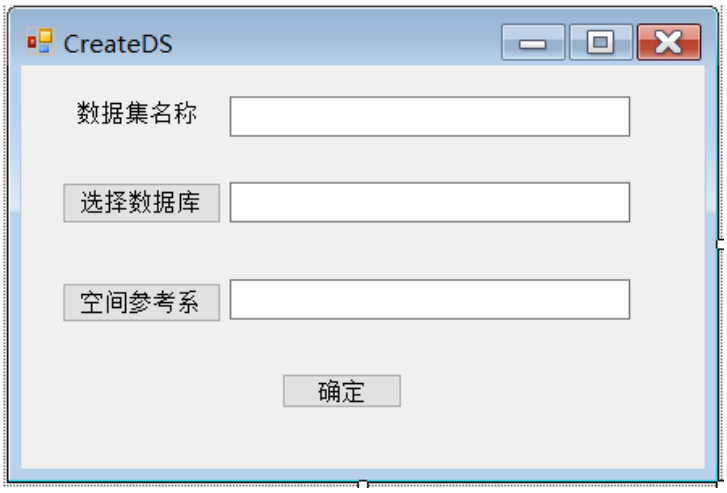
创建数据集前需要输入数据集的名称，选择数据集要存放的数据库位置并指定数据集的空间参考系。其中创建数据集的工作主要通过 DataBaseOperator 类的 public IFeatureDataset CreateFeatureDataset(IWorkspace workspace, string fdsName, ISpatialReference fdsSR)函数完成，而选择数据库和指定空间参考系的函数都直接封装在“CreateDS”窗体中。

1.添加控件及窗体设计

在主窗体的【拓扑检查】菜单下新建【添加数据集】菜单，并将控件命名为“miAddDataSet”。



新建 Windows 窗体“CreateDS”，窗体设计如下：其中【选择数据库】按钮的名称为“btnSelectDB”，【空间参考系】按钮的名称为“btnSetSpaRef”，【确定】按钮的名称为“btnSure”。



2.完善类的功能

向 DataBaseOperator 类中添加 public IFeatureDataset CreateFeatureDataset(IWorkspace workspace, string fdsName, ISpatialReference fdsSR)函数，用于在指定 IWorkspace 中创建数据集 ，返回创建的数据集。代码如下：

```
public IFeatureDataset CreateFeatureDataset(IWorkspace workspace, string fdsName, ISpatialReference fdsSR)
        {
            IFeatureWorkspace featureWorkspace = (IFeatureWorkspace)workspace;
            return featureWorkspace.CreateFeatureDataset(fdsName, fdsSR);
        }
```

3.实现数据集的创建

“CreateDS”窗体内建立了 MainForm 类的对象 cDS，并建立了两个全局变量 ISpatialReference spaRef 和 IWorkspace workspace 分别用于指定数据集的空间参考系和工作空间。

数据库的选择通过 private void btnSelectDB_Click(object sender, EventArgs e)

函数进行，代码如下：

```
//选择存放数据集的数据库
        private void btnSelectDB_Click(object sender, EventArgs e)
        {
            DataBaseOperator dataBaseOperator = new DataBaseOperator();

            String DBPath = "";
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.InitialDirectory = "C://";
            openFileDialog.FilterIndex = 1;
            openFileDialog.RestoreDirectory = true;
            openFileDialog.Title = "选择需要打开数据库的位置";
            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                DBPath = openFileDialog.FileName;
                int start = DBPath.LastIndexOf('\\');//获取数据库名称开始的字符在DBPath中的位置
                tbxSelectDB.Text = DBPath.Substring(start+1);//把数据库名称添加到tbxSelectDB中
            }

            Type factoryType = Type.GetTypeFromProgID(
                "esriDataSourcesGDB.AccessWorkspaceFactory");
```

```
            IWorkspaceFactory workspaceFactory =
(IWorkspaceFactory)Activator.CreateInstance
                (factoryType);
            workspace = workspaceFactory.OpenFromFile(DBPath, 0);// 通过路径应路径打开数据库
        }
```

空间参考系默认为当前地图的空间参考系，也可以指定成某个带有空间参考系的 shp 文件的空间参考系。代码如下：

```
//设置空间参考系
        private void btnSetSpaRef_Click(object sender, EventArgs e)
        {
            //选择要加入数据集的要素
            String IN_ShapePath = "";
            OpenFileDialog openFileDialog = new OpenFileDialog(); //选择需要添加的要素的位置
            openFileDialog.Title = "选择要加入数据集的数据";
            openFileDialog.InitialDirectory = "C://";
            openFileDialog.FilterIndex = 1;
            openFileDialog.RestoreDirectory = true;
            if (openFileDialog.ShowDialog() == DialogResult.OK)
                IN_ShapePath = openFileDialog.FileName;
            Geoprocessor GP_Tool = new Geoprocessor();
            string Temp_Direction =
System.IO.Path.GetDirectoryName(IN_ShapePath);//该Shp文件的目录
```

```
            FileInfo fi = new FileInfo(IN_ShapePath);
            string Temp_Name = fi.Name; //  该shp的文件名
            String Temp_Name2 = System.IO.Path.GetFileNameWithoutExtension(IN_ShapePath);//无扩展名的shp名称
            IWorkspaceFactory workspaceFactory = new ShapefileWorkspaceFactoryClass();
            IWorkspace workspace = workspaceFactory.OpenFromFile(Temp_Direction, 0);//打开存储shapefile的文件夹
            IFeatureWorkspace featureWorkspace = workspace as IFeatureWorkspace;
            IFeatureClass feature = featureWorkspace.OpenFeatureClass(Temp_Name);
            spaRef = (feature as IGeoDataset).SpatialReference;
            tbxSpaRef.Text = spaRef.Name;
        }
```

点击【确定】按钮后，会根据输入信息的情况将信息传回 MainForm 中的 AddDataSet 函数，创建相应的数据库并给出相关信息。若未指定空间参考系，会默认将数据集的空间参考系指定为当前地图的空间参考系，若未选择数据库或未输入数据集的名称，则会弹出提示消息框。代码如下：

```
private void btnSure_Click(object sender, EventArgs e)
        {
            if (tbxDSName.Text != "" && tbxSelectDB.Text != "" && tbxSpaRef.Text != "")
            {
```

```
                cDS.AddDataSet(tbxDSName.Text, spaRef, workspace, true);
                MessageBox.Show("在数据库" + tbxSelectDB.Text + "中创建数据集成功！" + '\n' + "空间参考系为：" + tbxSpaRef.Text);
                this.Close();
            }
            else if (tbxDSName.Text != "" && tbxSelectDB.Text != "" && tbxSpaRef.Text == "")
            {
                cDS.AddDataSet(tbxDSName.Text, spaRef, workspace, false);
                MessageBox.Show("在数据库" + tbxSelectDB.Text + "中创建数据集成功！" + '\n' + "空间参考系为当前地图空间参考系");
                this.Close();
            }
            else if (tbxDSName.Text == "")
                MessageBox.Show("请输入数据集名称！", "警告", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk);
            else if (tbxSelectDB.Text == "")
                MessageBox.Show("请选择数据库！", "警告", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk);
        }
```

MainForm 中的【创建数据集】的单击响应函数用于调用“CreateDS”窗口，代码如下：

```
//调用CreateDS窗口
        private void miAddDataSet_Click(object sender, EventArgs e)
```

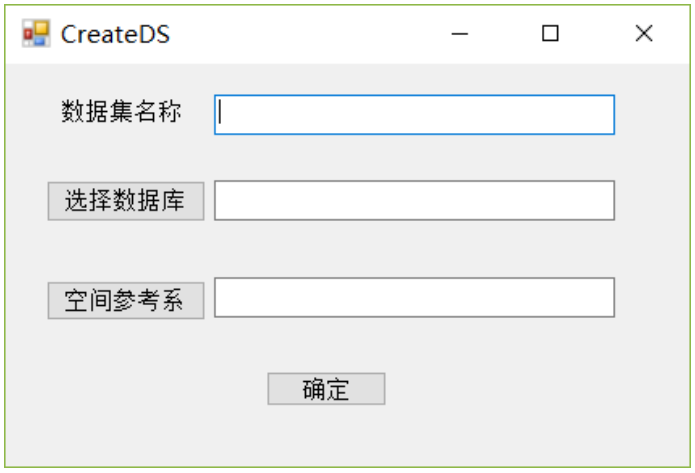
```
        {
            CreateDS createDS = new CreateDS(this);
            createDS.Show();
        }
```

MainForm 中的 AddDataSet 函数用于接收“CreateDS”窗口的信息并创建数据集，代码如下：

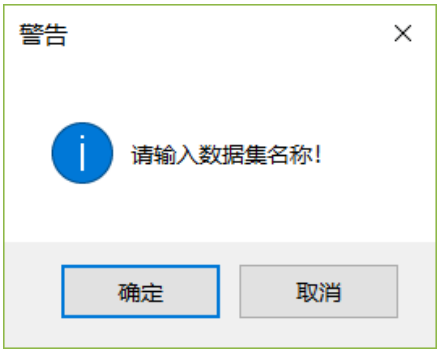
```
//添加数据集
        public void AddDataSet(String dsName, ISpatialReference spRef,
IWorkspace workSpace, Boolean haveSpaRef)
        {
            DataBaseOperator dataBaseOperator = new DataBaseOperator();
            if (haveSpaRef == false)
            {
                spRef = axMapControl1.Map.SpatialReference;
            }
            //创建数据集
            IFeatureDataset featureDataset =
dataBaseOperator.CreateFeatureDataset(
                workSpace, dsName, spRef);
        }
```

4.运行结果与异常处理

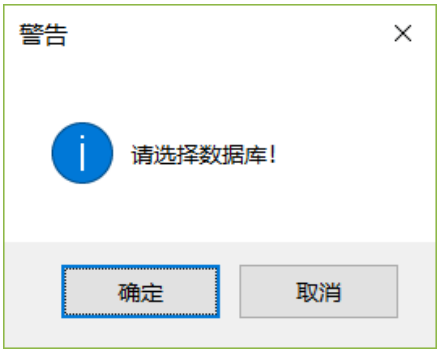
点击【拓扑检查】菜单下的【创建数据集】按钮，弹出以下窗口：



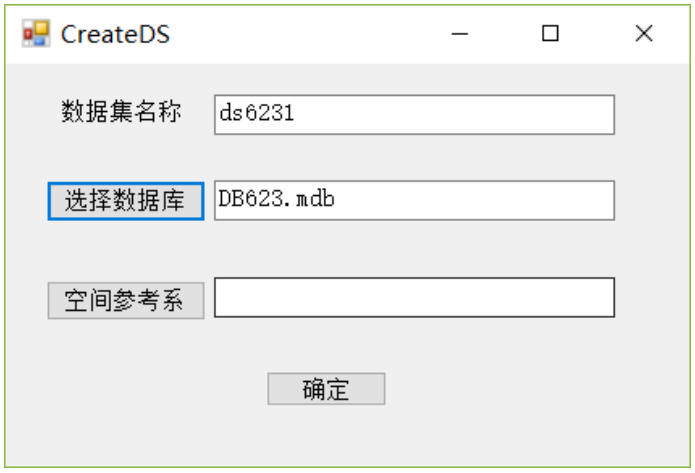
若未输入数据集名称就点击【确定】，则会弹出以下消息提示框：



若未选择数据集要存放的数据库，则会弹出以下消息提示框：



选择了数据库后，数据库的名称会显示在【选择数据库】后的文本框中，效果如图：



若未指定数据集的空间参考系，则会以当前地图的空间参考系作为数据集的空间参考系，点击【确定】按钮后成功创建数据集。

若指定了空间参考系，则空间参考系的名称会显示在【空间参考系】后的文本框中，效果如图：

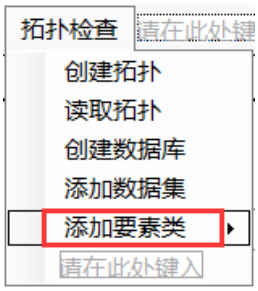
单击【确定】后完成数据集的创建，并弹出消息提示框：

2.1.3 添加要素类

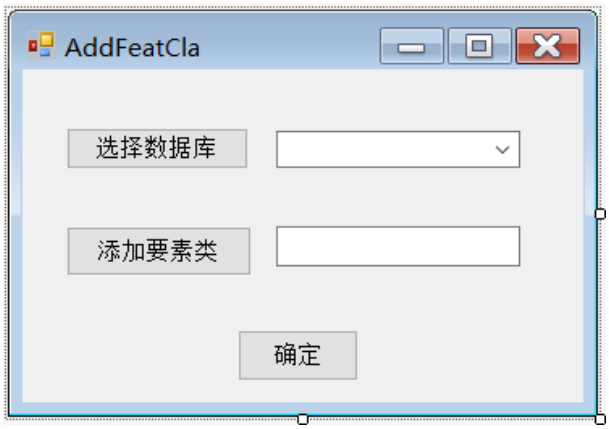
在实现添加要素类之前，需要先指定数据库和数据集，最后向选定的数据集中添加要素类。相关函数均封装在“AddFeatCla”类中。

1.添加控件及窗体设计

在主窗体的【拓扑检查】菜单下新建【添加要素类】菜单，并命名为“miAddFeatClass”。



新建 Windows 窗体“AddFeatCla”，用于实现向指定数据集添加要素类的功能。窗体设计如下，其中【选择数据库】按钮的名称为“btnSelectDB”，其后存放数据库内数据集名称的 comboBox 的名称为“cbDataSet”，【添加要素类】按钮的名称为“addShp”，其后用于显示要素类名称的文本框的名称为“tbxAddShp”，【确定】按钮的名称为“btnSure”。



2.完善类的功能

向 DataBaseOperator 类中添加 public String AddShpToSet (IFeatureDataset ds) 函数，用于向指定数据集中添加要素类，同时返回要素类的路径字符串。代码如下：

```
public String AddShpToSet(IFeatureDataset ds)
        {
            //选择要加入数据集的要素
            String IN_ShapePath = "";
            OpenFileDialog openFileDialog = new OpenFileDialog(); //选择需要添加的要素的位置
            openFileDialog.Title = "选择要加入数据集的数据";
            openFileDialog.InitialDirectory = "C://";
            openFileDialog.FilterIndex = 1;
            openFileDialog.RestoreDirectory = true;
            if (openFileDialog.ShowDialog() == DialogResult.OK)
                IN_ShapePath = openFileDialog.FileName;
            IFeatureDataset FDS_Featuredataset = ds;//目标featuredataset
            Geoprocessor GP_Tool = new Geoprocessor();
            string Temp_Direction = System.IO.Path.GetDirectoryName(IN_ShapePath);//该Shp文件的目录
```

```
            FileInfo fi = new FileInfo(IN_ShapePath);
            string Temp_Name = fi.Name; //  该shp的文件名
            String Temp_Name2 =
System.IO.Path.GetFileNameWithoutExtension(IN_ShapePath);//无扩展名的shp名称
            IWorkspaceFactory workspaceFactory = new
ShapefileWorkspaceFactoryClass();
            IWorkspace workspace =
workspaceFactory.OpenFromFile(Temp_Direction, 0);//打开存储shapefile的文件夹
            IFeatureWorkspace featureWorkspace = workspace as
IFeatureWorkspace;
            IFeatureClass feature =
featureWorkspace.OpenFeatureClass(Temp_Name);//通过名称打开featureclass，名字需要完整
            //实现添加的函数
            try
            {
                FeatureClassToFeatureClass Temp_FCToFC = new
FeatureClassToFeatureClass(feature, FDS_Featuredataset, Temp_Name2);
                GP_Tool.Execute(Temp_FCToFC, null);
                MessageBox.Show("要素集添加成功！");
                return openFileDialog.FileName;
            }
            catch (COMException comExc)
            {
                MessageBox.Show(String.Format(
                    "要素集添加失败: {0} 错误信息: {1}",
comExc.ErrorCode,
                    comExc.Message));
                return null;
```

```
        }
    }
```

3.实现数据集的创建

“AddFeatCla”窗体内建立了 public static IFeatureWorkspace featureWorkspace，public String DBPathOut = ""和 IFeatureDataset featureDataset 三个全局变量，分别用于存放工作空间，记录数据库的位置和存放保存了要素类的数据集。

选择数据库部分通过 IWorkspaceFactory 接口的 OpenFromFile 函数打开数据库，再通过 IEnumDatasetName 接口获取数据库中所有的数据集名称。代码如下：

```
private void btnSelectDB_Click(object sender, EventArgs e)
        {
            String DBPath = "";
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.InitialDirectory = "C://";
            openFileDialog.FilterIndex = 1;
            openFileDialog.RestoreDirectory = true;
            if (openFileDialog.ShowDialog() == DialogResult.OK)
                DBPath = openFileDialog.FileName;
            DBPathOut = DBPath;
            Type factoryType = Type.GetTypeFromProgID(
                "esriDataSourcesGDB.AccessWorkspaceFactory");
            IWorkspaceFactory workspaceFactory =
(IWorkspaceFactory)Activator.CreateInstance
                (factoryType);
            IWorkspace workspace = workspaceFactory.OpenFromFile(DBPath,
0);// 通过路径应路径打开数据库
```

```
            //选择数据库后获取所有数据集名称，并添加到复选框
            featureWorkspace = (IFeatureWorkspace)workspace;
            IEnumDatasetName enumDatasetName =
workspace.DatasetNames[ESRI.ArcGIS.Geodatabase.esriDatasetType.esriDTFeature
Dataset];
            enumDatasetName.Reset();
            IDatasetName datasetName = enumDatasetName.Next();
            while (datasetName != null)
            {
                cbDataSet.Items.Add(datasetName.Name.ToString());
                datasetName = enumDatasetName.Next();
            }
        }
```

选择数据集。当选完数据库后，就可以在【选择数据库】后的列表中选择数据库了，当 comboBox 的选项更改后通过相应的路径打开数据集。代码如下：

```
private void cbDataSet_SelectedIndexChanged(object sender, EventArgs e)
        {
            Type factoryType = Type.GetTypeFromProgID(
                "esriDataSourcesGDB.AccessWorkspaceFactory");
            IWorkspaceFactory workspaceFactory =
(IWorkspaceFactory)Activator.CreateInstance
                (factoryType);
            IWorkspace workspace =
workspaceFactory.OpenFromFile(DBPathOut, 0);// 通过路径应路径打开数据库
            IFeatureWorkspace featureWorkspace =
(IFeatureWorkspace)workspace;
            String datasetName = cbDataSet.SelectedItem.ToString();
             featureDataset =
```

```
featureWorkspace.OpenFeatureDataset(datasetName);//通过名称打开相应的数据集
        }
```

确定了数据库和数据集后，就可以调用 DataBaseOperator 类中的 AddShpToSet 函数，向数据集添加要素类。代码如下：

```
        private void addShp_Click(object sender, EventArgs e)
        {
            DataBaseOperator dataBaseOperator1 = new DataBaseOperator();
            String path = dataBaseOperator1.AddShpToSet(featureDataset);
            int start = path.LastIndexOf('\\');//获取shp文件名开始的字符在path中的位置
            tbxAddShp.Text = path.Substring(start + 1);//把shp名称添加到tbxAddShp中
        }
```

单击【确定】后“AddFeatCla”窗体关闭：

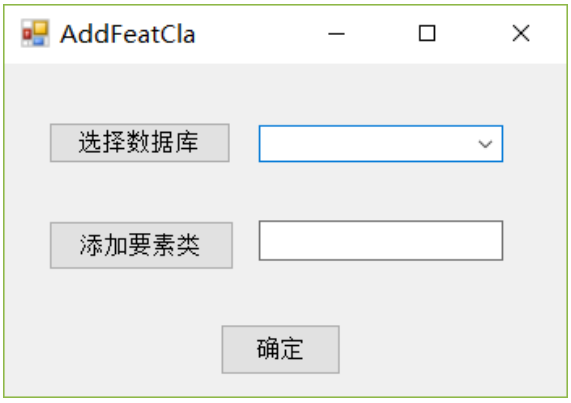
```
private void btnSure_Click(object sender, EventArgs e)
        {
            this.Close();
        }
```

MainForm 中的【添加要素类】菜单的单击响应函数：

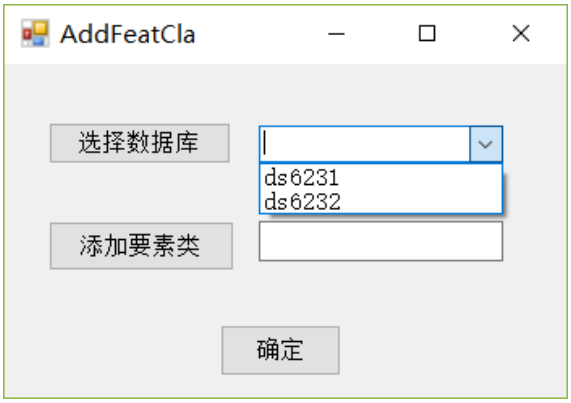
```
//调用AddFeatCla窗口添加要素类
        private void miAddFeatClass_Click(object sender, EventArgs e)
        {
            AddFeatCla addFeatCla = new AddFeatCla();
            addFeatCla.Show();
        }
```

4.运行结果与异常处理

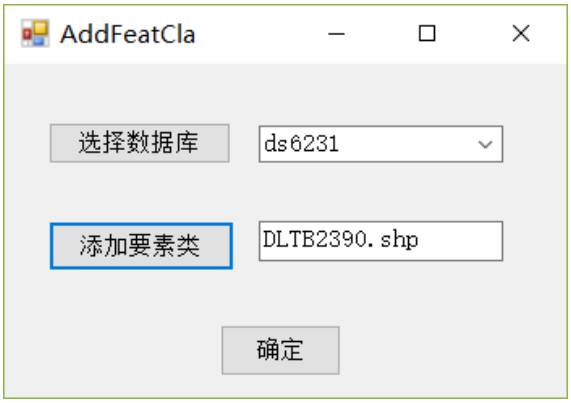
单击【添加要素类】后弹出以下窗口：



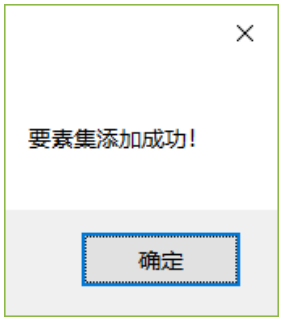
选择数据库后，数据库中的数据集全部被添加到 comboBox 中：



选择了数据集后，点击【添加要素类】，向数据集中添加要素类，要素类的名称将会显示在文本框中：



并弹出以下消息提示窗口：



在 AddShpToSet 函数中针对要素类添加失败进行了异常处理，相关代码段如下：

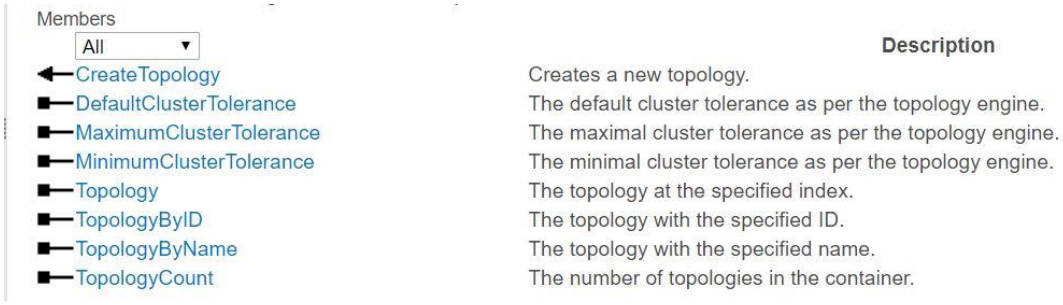
```
try
                {
                    FeatureClassToFeatureClass Temp_FCToFC = new
FeatureClassToFeatureClass(feature, FDS_Featuredataset, Temp_Name2);
                    GP_Tool.Execute(Temp_FCToFC, null);
                    MessageBox.Show("要素集添加成功！");
                    return openFileDialog.FileName;
                }
                catch (COMException comExc)
                {
                    MessageBox.Show(String.Format(
                        "要素集添加失败: {0} 错误信息: {1}",
comExc.ErrorCode,
                        comExc.Message));
                    return null;
                }
```

2.2 创建拓扑

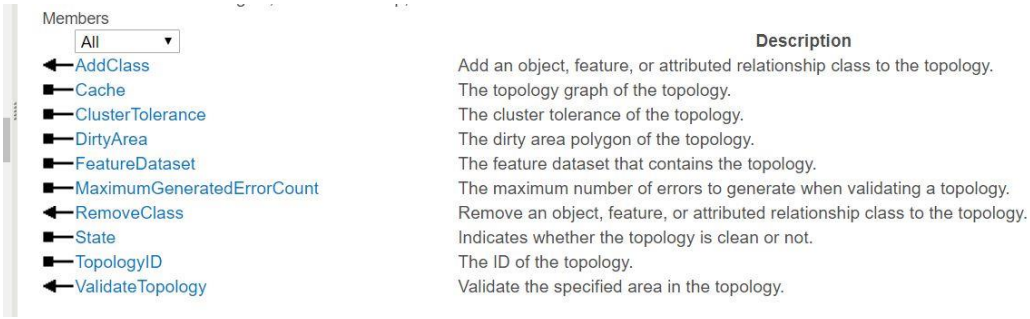
2.2.1 介绍

拓扑的创建必须在一个数据库的某个数据集之中，数据库和数据集的操作详

见上文。创建拓扑是主要用到的接口有 ITopologyContainer，该接口对象可由 IFeatureDataset 转化而来。下图为 ITopologyContainer 接口。



另外一个常用的接口为 ITopology，通过该接口可实现拓扑规则、要素等的添加，下图为接口详情。



在进行拓扑操作时，首先要创建一个拓扑对象。

2.2.2 添加类

添加一个 Topology 类，存放拓扑对象，用于实现对拓扑的创建编辑读取操作，类的权限设为 public，添加相关引用；

```
using ESRI.ArcGIS.Geodatabase;

using ESRI.ArcGIS.Geometry;
```

2.2.3 完善类的功能

向 topology 类中添加 AddRuleToTopology 成员函数，用于添加一个要素类拓扑规则，并实现其重载，添加两个类相关的拓扑规则。

代码如下

```
public void AddRuleToTopology(ITopology topology, esriTopologyRuleType
ruleType,

                              String ruleName, IFeatureClass
```

```
featureClass0)
        {
            // Create a topology rule.
            ITopologyRule topologyRule = new TopologyRuleClass();
            topologyRule.TopologyRuleType = ruleType;
            topologyRule.Name = ruleName;
            topologyRule.OriginClassID = featureClass0.FeatureClassID;
            topologyRule.AllOriginSubtypes = true;
            // Cast the topology to the ITopologyRuleContainer interface and
add the rule.
            ITopologyRuleContainer topologyRuleContainer =
(ITopologyRuleContainer)topology;
            if (topologyRuleContainer.get_CanAddRule(topologyRule))
            {
                topologyRuleContainer.AddRule(topologyRule);
            }
            else
            {
                MessageBox.Show("不能添加特殊规则！ ");
                return;
            }
        }
        //添加拓扑规则——两个图层
        public void AddRuleToTopology(ITopology topology,
esriTopologyRuleType ruleType,
                                      String ruleName, IFeatureClass
originClass, int originSubtype, IFeatureClass
                                       destinationClass, int
destinationSubtype)
```

```
        {
            // Create a topology rule.
            ITopologyRule topologyRule = new TopologyRuleClass();
            topologyRule.TopologyRuleType = ruleType;
            topologyRule.Name = ruleName;
            topologyRule.OriginClassID = originClass.FeatureClassID;
            topologyRule.AllOriginSubtypes = false;
            topologyRule.OriginSubtype = originSubtype;
            topologyRule.DestinationClassID =
destinationClass.FeatureClassID;
            topologyRule.AllDestinationSubtypes = false;
            topologyRule.DestinationSubtype = destinationSubtype;
            // Cast the topology to the ITopologyRuleContainer interface and
add the rule.
            ITopologyRuleContainer topologyRuleContainer =
(ITopologyRuleContainer)topology;
            if (topologyRuleContainer.get_CanAddRule(topologyRule))
            {
                topologyRuleContainer.AddRule(topologyRule);
            }
            else
            {
                MessageBox.Show("不能添加特殊规则！");
                return;
            }
        }
```

添加 ValidateTopology 成员函数，用于验证拓扑规则。

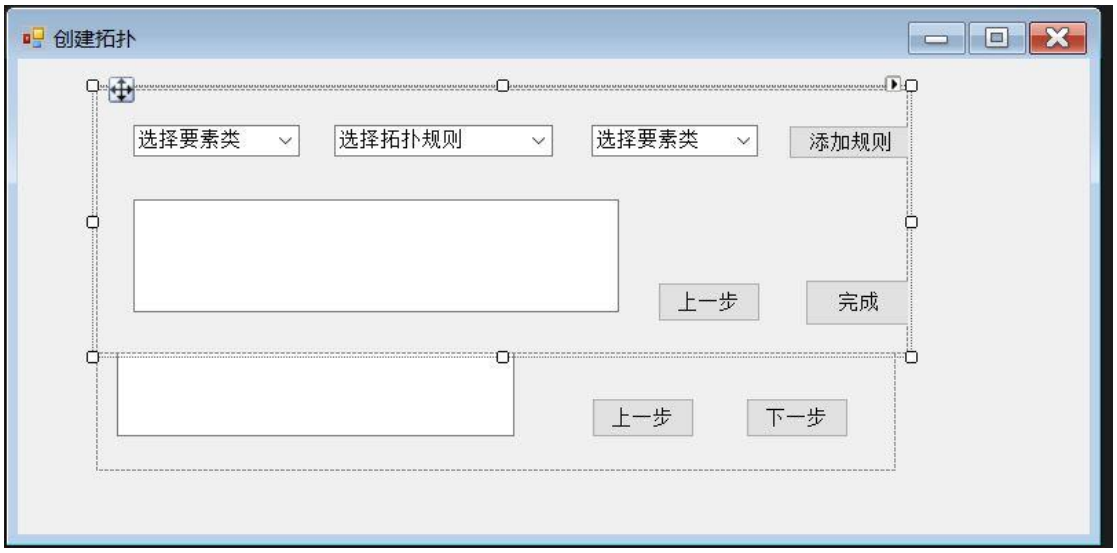
代码如下

```
public void ValidateTopology(ITopology topology, IEnvelope envelope)
        {
            // Get the dirty area within the provided envelope.
            IPolygon locationPolygon = new PolygonClass();
            ISegmentCollection segmentCollection = (ISegmentCollection)locationPolygon;
            segmentCollection.SetRectangle(envelope);
            IPolygon polygon = topology.get_DirtyArea(locationPolygon);
            // If a dirty area exists, validate the topology.
            if (!polygon.IsEmpty)
            {
                // Define the area to validate and validate the topology.
                IEnvelope areaToValidate = polygon.Envelope;
                IEnvelope areaValidated = topology.ValidateTopology(areaToValidate);
            }
```

2.2.4 窗体设计

添加一个新窗体，命名为 CreateTopology。添加相应的控件 bottom，combox，labe 等。

添加相应的成员函数完成数据库的选择、数据集的选择、要素的选择、要素等级的选择和添加、拓扑规则的选择。其函数分别为 btChooseDB_Click、cbDatesetName_SelectedIndexChanged、tbAddFeatueclass_Click、btAddRule_Click。



2.2.5 窗体主要功能的实现

窗体的主要功能是输入拓扑名称、选择数据库、数据集等。其实现见下列函数。

选择数据库

选择需要创建 topology 的数据库、选择数据库之后读取数据库中的所有数据集，并将其名称添加到复选框。

```
private void btChooseDB_Click(object sender, EventArgs e)
 {
      String DBPath = "";
      OpenFileDialog openFileDialog = new OpenFileDialog();
      openFileDialog.InitialDirectory = "C://";
      openFileDialog.FilterIndex = 1;
      openFileDialog.RestoreDirectory = true;
      if (openFileDialog.ShowDialog() == DialogResult.OK)
          DBPath = openFileDialog.FileName;
      DBPathOut = DBPath;
      Type factoryType = Type.GetTypeFromProgID(
          "esriDataSourcesGDB.AccessWorkspaceFactory");
```

```
                IWorkspaceFactory workspaceFactory =
(IWorkspaceFactory)Activator.CreateInstance
                    (factoryType);
                try
                {
                    IWorkspace workspace =
workspaceFactory.OpenFromFile(DBPath, 0);// 通过路径应路径打开数据库
                    //选择数据库后获取所有数据集名称，并添加到复选框
                    cbDatesetName.Items.Clear();
                    featureWorkspace = (IFeatureWorkspace)workspace;
                    IEnumDatasetName enumDatasetName =
workspace.DatasetNames[ESRI.ArcGIS.Geodatabase.esriDatasetType.esriDTFe
atureDataset];
                    enumDatasetName.Reset();
                    IDatasetName datasetName = enumDatasetName.Next();
                    while (datasetName != null)
                    {

cbDatesetName.Items.Add(datasetName.Name.ToString());
                        datasetName = enumDatasetName.Next();
                    }
                }
                catch { return; }
                }
```

选择数据集

数据集选择状态更改后更新信息，当数据集被选定时，读取数据集中的所有要素信息，并将其添加至复选框。

```
private void cbDatesetName_SelectedIndexChanged(object sender,
EventArgs e)
```

```
        {
            String datasetName = cbDatesetName.SelectedItem.ToString();
            featureDataset = featureWorkspace.OpenFeatureDataset(datasetName);//通过名称打开相应的数据集
            //通过featuredataset获取feature
            IFeatureClassContainer featureClassContainer = (IFeatureClassContainer)featureDataset;
            IEnumFeatureClass enumFeatureClass = featureClassContainer.Classes;
            IFeatureClass featureClass = enumFeatureClass.Next();
            while (featureClass != null)
            {
                cbFeatureClass.Items.Add(featureClass.AliasName);
                cbFeatureClass1.Items.Add(featureClass.AliasName);
                cbFeatureClass2.Items.Add(featureClass.AliasName);
                featureClass = enumFeatureClass.Next();
            }
            ITopologyWorkspace TopoWorkSpace = featureWorkspace as ITopologyWorkspace;
            try//若不存在同名拓扑则添加
            {
                ITopology Topology1 = TopoWorkSpace.OpenTopology(tbTopologyName.Text + "_Topology");
                MessageBox.Show("已存在该拓扑，无法添加！");
            }
            catch
            {
                return;
            }
```

}当数据集被选定后，点击下一页在该数据集创建拓扑

```
private void panel1NextBt_Click(object sender, EventArgs e)
{
    if (tbTopologyName.Text == "")
    {
        MessageBox.Show("拓扑名称不能为空！");
        return;
    }
    Type factoryType = Type.GetTypeFromProgID(
        "esriDataSourcesGDB.AccessWorkspaceFactory");
    IWorkspaceFactory workspaceFactory =
(IWorkspaceFactory)Activator.CreateInstance
        (factoryType);
    IWorkspace workspace =
workspaceFactory.OpenFromFile(DBPathOut, 0);// 通过路径应路径打开数据
库
    featureWorkspace = (IFeatureWorkspace)workspace;

    //创建 topology
    ISchemaLock schemaLock = (ISchemaLock)featureDataset;
    try
    {

schemaLock.ChangeSchemaLock(esriSchemaLock.esriExclusiveSchemaLock);
        // Create the topology.
        ITopologyContainer topologyContainer =
(ITopologyContainer)featureDataset;
        String topologyName = tbTopologyName.Text;
        //下一行为测试代码
```

```
                    // MessageBox.Show(topologyName + "_Topology" +
"@" + topologyContainer.DefaultClusterTolerance.ToString());
                    topology =
topologyContainer.CreateTopology(topologyName + "_Topology",
                        topologyContainer.DefaultClusterTolerance, -1, "");
                }
                catch (COMException comExc)
                {
                    MessageBox.Show(String.Format(
                        "拓扑创建失败: {0}  错误信息: {1}",
comExc.ErrorCode,
                        comExc.Message));
                }
                finally
                {

schemaLock.ChangeSchemaLock(esriSchemaLock.esriSharedSchemaLock);
                }
                panel1.Visible = false;
                panel2.Visible = true;
            }
```

2.2.6 添加图层和拓扑规则

添加要素类

通过复选框，选择相应的要素类，将其添加至拓扑，调用 topology 对象里的 addclass 函数实现

```
    private void tbAddFeatueclass_Click(object sender, EventArgs e)
      {//
```

```
            try
            {
                IFeatureClass featureClass =
featureWorkspace.OpenFeatureClass(cbFeatureClass.SelectedItem.ToString());
                if (featureClass == null)
                {
                    MessageBox.Show("null");
                    return;
                }
                topology.AddClass(featureClass, int.Parse(tbRank.Text), 1, 1,
false);
                tbAddedFeatureclass.Text += ("\r\n" + featureClass.AliasName
+ "    " + tbRank.Text);
                tbAddedFeatureclass.Refresh();
            }
            catch (COMException comExc)
            {
                MessageBox.Show(String.Format(
                    "添加要素类失败: {0}  错误信息: {1}",
comExc.ErrorCode,
                    comExc.Message));
            }
              }
```

添加拓扑规则

选择拓扑规则，点击“添加规则”按钮时将其添加至拓扑，调用 topology 类里的 Addruletotopology 实现

```
    private void btAddRule_Click(object sender, EventArgs e)
     {
```

```
            esriTopologyRuleType ruleType =
esriTopologyRuleType.esriTRTAny;
            switch(cbRule.SelectedIndex)
            {
                case 0:ruleType =
esriTopologyRuleType.esriTRTAreaNoOverlap;break;
                case 1:ruleType =
esriTopologyRuleType.esriTRTAreaNoGaps;break;
                case 2:ruleType =
esriTopologyRuleType.esriTRTAreaNoOverlapArea;break;
                case 3:ruleType =
esriTopologyRuleType.esriTRTLineNoSelfIntersect;break;
            }
            IFeatureClass featureClass =
featureWorkspace.OpenFeatureClass(cbFeatureClass1.SelectedItem.ToString());
            myTopology.AddRuleToTopology(topology, ruleType,
                    cbRule.SelectedItem.ToString(), featureClass);
            tbAddedRule.Text += ("\r\n" + featureClass.AliasName +" "+
cbRule.SelectedItem.ToString());
            tbAddedRule.Update();
                }
```

2.2.7 验证拓扑

当规则添加完毕时点击完成，验证拓扑规则，调用 topology 类里的 ValidateTopology 函数实现

```
        private void btDone_Click(object sender, EventArgs e)
         {
            //以下为验证拓扑的代码
            // Get an envelope with the topology's extents and validate the
topology.
```

```
IGeoDataset geoDataset = (IGeoDataset)topology;
IEnvelope envelope = geoDataset.Extent;
try
{
    myTopology.ValidateTopology(topology, envelope);// 验证拓扑
    //将拓扑结果加载到地图上
    if (topology != null)
    {
        ITopologyLayer topolayer = new TopologyLayerClass();
        topolayer.Topology = topology;
        ILayer layer = topolayer as ILayer;
        layer.Name = tbTopologyName.Text + "_Topology";
        MapContro1.Map.AddLayer(layer);
        MapContro1.Refresh();
    }
    MessageBox.Show("拓扑创建完成！已将拓扑结果添加到图层中！");
    this.Close();
}
catch (COMException comExc)
{
    MessageBox.Show(String.Format(
        "拓扑验证失败: {0} 错误信息: {1}", comExc.ErrorCode,
        comExc.Message));
}
this.Close();
    }
```

运行结果与异常处理

2.3 读取拓扑

读取拓扑结果并在数据表以及地图中显示拓扑错误是拓扑检查中较为重要的一部分，该部分内容用于交互显示验证得到的拓扑错误，可自由框选查询区域内的错误，选择数据表中的错误条目会进行跳转显示，同时也能实现对编辑修改后的拓扑进行实时验证的功能。

用到的接口主要：ITopologyContainer；ITopology；IErrorFeatureContainer；

All	Description
ErrorFeature	An error feature with that matches the passed in parameters.
ErrorFeatures	An enumeration of error features that match the passed in parameters.
ErrorFeaturesByGeometryType	An enumeration of error features that match the passed in parameters.
ErrorFeaturesByRuleType	An enumeration of error features that match the passed in parameters.

ITopologyRuleContainer；

All	Description
AddRule	Adds a topology rule.
CanAddRule	Indicates if the topology rule can be added to the topology.
DeleteRule	Deletes a topology rule.
DemoteFromRuleException	Demotes a topology error from an exception to an error.
PromoteToRuleException	Promotes a topology error to an exception.
Rule	The topology rule with the corresponding ID.
RuleByGUID	The rule with the globally unique ID.
Rules	An enumeration of all the topology rules.
RulesByClass	An enumeration of all the rules for a given class.
RulesByClassAndSubtype	An enumeration of all the rules for a given class and subtype.

ITopologyRule；

All	Description
AllDestinationSubtypes	Indicates if all destination subtypes are specified for the topology rule.
AllOriginSubtypes	Indicates if all origin subtypes are specified for the topology rule.
DestinationClassID	Destination ClassID of the topology rule.
DestinationSubtype	Destination subtype of the topology rule.
DestinationSubtypeSpecified	Indicates if a destination subtype has been specified.
ErrorShapeTypes	Indicates the shape types of errors for the topology rule.
GUID	GUID of the topology rule.
Name	Name of the topology rule.
OriginClassID	Origin ClassID of the topology rule.
OriginSubtype	Origin subtype of the topology rule.
OriginSubtypeSpecified	Indicates if an origin subtype has been specified.
TopologyRuleType	Topology rule type of the topology rule.
TriggerErrorEvents	Indicates if error events are triggered for the topology rule.

ITopologicalOperator

All	Description
Boundary	The boundary of this geometry. A polygon's boundary is a polyline. A polyline's boundary is a multipoint. A point or multipoint's boundary is an empty point or multipoint.
Buffer	Constructs a polygon that is the locus of points at a distance less than or equal to a specified distance from this geometry.
Clip	Constructs the intersection of this geometry and the specified envelope.
ClipDense	Constructs the intersection of this geometry and the specified envelope; densifies lines in output contributed by the clipping envelope.
ConstructUnion	Defines this geometry to be the union of the inputs. More efficient for unioning multiple geometries than calling Union repeatedly.
ConvexHull	Constructs the convex hull of this geometry.
Cut	Splits this geometry into a part left of the cutting polyline, and a part right of it.
Difference	Constructs the geometry containing points from this geometry but not the other geometry.
Intersect	Constructs the geometry that is the set-theoretic intersection of the input geometries. Use different resultDimension values to generate results of different dimensions.
IsKnownSimple	Indicates whether this geometry is known (or assumed) to be topologically correct.
IsSimple	Indicates whether this geometry is known (or assumed) to be topologically correct, after explicitly determining this if the geometry is not already known (or assumed) to be simple.
QueryClipped	Redefines clippedGeometry to be the intersection of this geometry and the clipping envelope.
QueryClippedDense	Redefines clippedGeometry to be the intersection of this geometry and the clipping envelope; densifies lines in the output contributed by the clipping envelope.
Simplify	Makes this geometry topologically correct.
SymmetricDifference	Constructs the geometry that contains points from either but not both input geometries.
Union	Constructs the geometry that is the set-theoretic union of the input geometries.

下面介绍在 VS2015 中编写代码，实现读取拓扑的功能。

2.3.1 读取拓扑结果

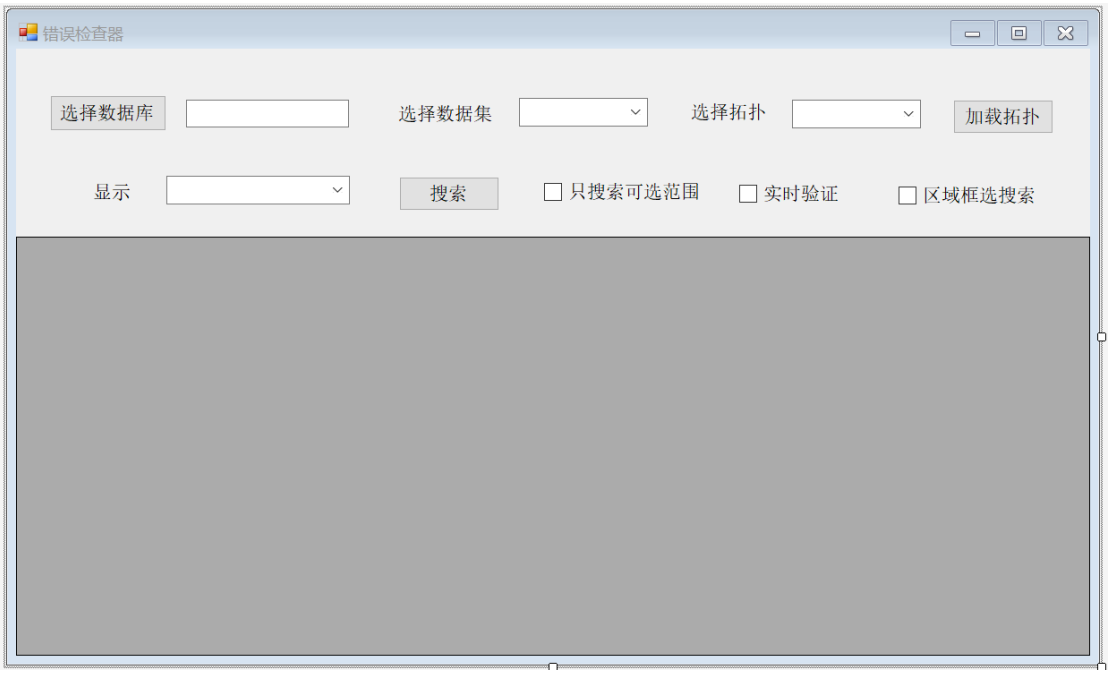
读取拓扑结果是在 ReadTopology 窗体中实施的，通过选择加载拓扑，搜索其中的拓扑错误。

1.添加菜单项

在主窗体的菜单栏“拓扑检查”中添加“读取拓扑”菜单项，其控件名为“tbReadTopology”。

2.读取拓扑窗体设计

参照 ArcMap 中设计如下 ReadTopology 窗体：



可分为三个部分，加载拓扑，搜索拓扑错误以及拓扑错误的数据表显示。

3. ReadTopology 窗体类

为该类导入部分类库，代码如下：

```
using ESRI.ArcGIS.Controls;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.Geometry;
```

```
using ESRI.ArcGIS.DataSourcesGDB;
```

初始化声明需要的变量，构造函数需要传入 AxMapControl 类型的参数，用于与地图控件的联系，代码如下：

```
public partial class ReadTopology : Form
    {
        IFeature feature;//当前选中的要素
        Topology top = new Topology();
        ITopology topology;
        AxMapControl MapContro1;
        IFeatureWorkspace featureWorkspace;
        public ReadTopology(AxMapControl MapContro)
        {
            MapContro1 = MapContro;
            InitializeComponent();
            GridView1.MultiSelect = false;
            cbTopologyRule.Text = "All Rules";
        }
```

4. 添加“选择数据库”点击事件

为“选择数据库”按钮生成点击事件响应函数，实现数据库选择，代码如下：

```
//点击打开数据库按钮
        private void btOpenDB_Click(object sender, EventArgs e)
        {
            String DBPath = "";
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.InitialDirectory = "C://";
            openFileDialog.FilterIndex = 1;
            openFileDialog.RestoreDirectory = true;
```

```
            if (openFileDialog.ShowDialog() == DialogResult.OK)
                DBPath = openFileDialog.FileName;
            string DBName = openFileDialog.SafeFileName;
            Type factoryType = Type.GetTypeFromProgID(
                "esriDataSourcesGDB.AccessWorkspaceFactory");
            IWorkspaceFactory workspaceFactory =
(IWorkspaceFactory)Activator.CreateInstance
                (factoryType);
            IWorkspace workspace = workspaceFactory.OpenFromFile(DBPath,
0);// 通过路径应路径打开数据库
            //选择数据库后获取所有数据集名称，并添加到复选框
            tbDBName.Clear();
            cbDSName.Items.Clear();
            cbTPName.Items.Clear();
            tbDBName.Text = DBName;
            featureWorkspace = (IFeatureWorkspace)workspace;
            IEnumDatasetName enumDatasetName =
workspace.DatasetNames[ESRI.ArcGIS.Geodatabase.esriDatasetType.esriDTFeature
Dataset];
            enumDatasetName.Reset();
            IDatasetName datasetName = enumDatasetName.Next();
            while (datasetName != null)
            {
                cbDSName.Items.Add(datasetName.Name.ToString());
                datasetName = enumDatasetName.Next();
            }
        }
```

5. 增加“数据集变更”函数

与“选择数据库”按钮相应的操作有选择其中的数据集和拓扑要素，当下拉框选中的对象发生改变时触发相应事件更新 featureclass 数据，代码如下：

```
//切换数据集时更新featureclass数据
        private void cbDSName_SelectedIndexChanged(object sender, EventArgs e)
        {
            cbTPName.Items.Clear();
            //通过名称打开相应的数据集
            IFeatureDataset featureDataset = featureWorkspace.OpenFeatureDataset(cbDSName.SelectedItem.ToString());
            //获取数据集中的拓扑名称
            IFeatureDatasetName2 featureDatasetNames = featureDataset.FullName as IFeatureDatasetName2;
            IEnumDatasetName enumDatasetName= featureDatasetNames.TopologyNames;
            IDatasetName datasetName = enumDatasetName.Next();
            if (datasetName != null)
            {
                if(datasetName.Name!=null)
                cbTPName.Items.Add(datasetName.Name);
            }
        }
```

6. 添加“加载拓扑”点击事件

为“加载拓扑”按钮生成点击事件响应函数，实现拓扑的加载。需要分析有无异常，如无拓扑等，并对它们进行处理。无异常则加载拓扑到地图上，同时也遍历该拓扑的拓扑规则，添加到选择拓扑规则的下拉框中以便后续搜索拓扑错误，代码如下：

```
private void btLoadTopology_Click(object sender, EventArgs e)
        {
```

```
            if (featureWorkspace == null || cbDSName.Text == "" ||
cbTPName.Text == "")
            {
                MessageBox.Show("数据为空！ ");
                return;
            }
            try
            {
                topology =
top.OpenTopologyFromFeatureWorkspace(featureWorkspace, cbDSName.Text,
cbTPName.Text);
            }
            catch
            {
                MessageBox.Show("未找到该拓扑！ ");
                return;
            }
            //将拓扑结果加载到地图上
            if (topology != null)
            {
                ITopologyLayer topolayer = new TopologyLayerClass();
                topolayer.Topology = topology;
                ILayer layer = topolayer as ILayer;
                layer.Name = cbTPName.Text;
                int i;
                for ( i= 0; i < MapContro1.Map.LayerCount; i++)
                {
                    string name = MapContro1.Map.Layer[i].Name;
                    if (name == layer.Name)
```

```
                    break;
                }
                if (i == MapContro1.Map.LayerCount)
                {
                    MapContro1.Map.AddLayer(layer);
                    MapContro1.Refresh();
                }
            }
            try
            {
                //通过名称打开相应的数据集
                IFeatureDataset featureDataset =
featureWorkspace.OpenFeatureDataset(cbDSName.SelectedItem.ToString());
                IErrorFeatureContainer errorFeatureContainer =
(IErrorFeatureContainer)topology;
                Topology myTopology = new Topology();
                DataTable dataTable = new DataTable();
                IGeoDataset g = (IGeoDataset)topology;
                IEnvelope env = g.Extent;
                dataTable = top.DisplayAllErrorFeatures(topology, env);
                GridView1.DataSource = dataTable;
                ZoomToEnvelope(env, MapContro1.ActiveView);
                //初始化错误类型列表
                ITopologyRuleContainer topologyRuleContainer =
(ITopologyRuleContainer)topology;
                IEnumRule enumRule = topologyRuleContainer.Rules;
                // 遍历拓扑规则.
                enumRule.Reset();
                IRule rule = null;
```

```
                while ((rule = enumRule.Next()) != null)
                {
                    ITopologyRule topologyRule = (ITopologyRule)rule;

cbTopologyRule.Items.Add(topologyRule.TopologyRuleType);
                }
            }
            catch
            {
                MessageBox.Show("拓扑错误未加载！");
                return;
            }
        }
```

7. 添加“搜索”点击事件

为“搜索”按钮生成点击事件响应函数，实现拓扑错误的搜索和显示。与之关联的有三个勾选框，分别是只搜索可选范围、实时验证、区域框选搜索。事件响应时判断是否勾选，再根据所选的拓扑规则搜索相应类型的拓扑错误。只搜索可选范围，包围盒赋值为 env = MapContro1.ActiveView.Extent 是搜索地图当前视图内的错误。具体的实时验证等写在第三点中。事件函数代码如下：

```
private void btErrorSearch_Click(object sender, EventArgs e)
        {
            DataTable dataTable = new DataTable();
            IGeoDataset g = (IGeoDataset)topology;
            IEnvelope env;
            if (cbCurrentExtent.Checked)
            {
                env = MapContro1.ActiveView.Extent;
            }
```

```
            else env = g.Extent;
            //实时验证框被勾选后进行实时验证
            if (cbvaliTopo.Checked)
            {
                top.ValidateTopology(topology, env);
                MapContro1.Refresh();
            }

            switch (cbTopologyRule.Text)
            {
                case "esriTRTAreaNoOverlap":
                    dataTable = top.DisplayErrorFeatureByRuleType(topology,
esriTopologyRuleType.esriTRTAreaNoOverlap, env);
                    break;
                case "esriTRTAreaNoGaps":
                    dataTable = top.DisplayErrorFeatureByRuleType(topology,
esriTopologyRuleType.esriTRTAreaNoGaps, env);
                    break;
                case "esriTRTAreaNoOverlapArea":
                    dataTable = top.DisplayErrorFeatureByRuleType(topology,
esriTopologyRuleType.esriTRTAreaNoOverlapArea, env);
                    break;
                default:
                    dataTable = top.DisplayAllErrorFeatures(topology, env);
                    break;
            }
            GridView1.DataSource = dataTable;
            ZoomToEnvelope(env, MapContro1.ActiveView);
        }
```

8. 增加“缩放到包围盒”函数

每次搜索错误之后需要缩放到指定范围，代码如下：

```
//缩放到指定范围
        public static void ZoomToEnvelope(IEnvelope env, IActiveView activeView)
        {
            activeView.Extent = env;
            activeView.Refresh();
            activeView.ScreenDisplay.UpdateWindow();
        }
```

9. 增加“根据规则搜索拓扑错误”函数

搜索拓扑错误需要在 Topology 类中添加两个函数，分别是根据选中的规则以及直接搜索所有错误，代码如下：

```
//给定拓扑和拓扑规则类型，返回指定规则的错误要素信息
        public DataTable DisplayErrorFeatureByRuleType(ITopology topology, esriTopologyRuleType ruleType,IEnvelope searchExtent)
        {
            //获取坐标系
            IErrorFeatureContainer errorFeatureContainer = (IErrorFeatureContainer)topology;
            IGeoDataset geoDataset = (IGeoDataset)topology;
            ISpatialReference spatialReference = geoDataset.SpatialReference;
            ITopologyRuleContainer topologyRuleContainer = (ITopologyRuleContainer)topology;
            //遍历拓扑规则
            IEnumRule enumRule = topologyRuleContainer.Rules;
            enumRule.Reset();
            IRule rule = null;
```

```
DataRow dataRow;
int i = 1;
DataTable dataTable = new DataTable();
DataColumn dataColunm = new DataColumn();
dataColunm.ColumnName = "序号";
dataColunm.DataType = System.Type.GetType("System.Int32");
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
dataColunm.ColumnName = "错误ID";
dataColunm.DataType = System.Type.GetType("System.Int32");
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
dataColunm.ColumnName = "规则类型";
dataColunm.DataType = System.Type.GetType("System.String");
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
dataColunm.ColumnName = "形状";
dataColunm.DataType = System.Type.GetType("System.String");
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
dataColunm.ColumnName = "原要素集ID";
dataColunm.DataType = System.Type.GetType("System.Int32");
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
dataColunm.ColumnName = "原OID";
dataColunm.DataType = System.Type.GetType("System.Int32");
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
```

```
                dataColunm.ColumnName = "目标要素集ID";
                dataColunm.DataType = System.Type.GetType("System.Int32");
                dataTable.Columns.Add(dataColunm);
                dataColunm = new DataColumn();
                dataColunm.ColumnName = "目标OID";
                dataColunm.DataType = System.Type.GetType("System.Int32");
                dataTable.Columns.Add(dataColunm);
                while ((rule = enumRule.Next()) != null)
                {
                    //获取当前拓扑规则的拓扑错误并遍历
                    ITopologyRule topologyRule = (ITopologyRule)rule;
                    IEnumTopologyErrorFeature enumTopologyErrorFeature =
errorFeatureContainer.get_ErrorFeatures(spatialReference, topologyRule,
searchExtent, true, true);
                    ITopologyErrorFeature topologyErrorFeature = null;
                    while ((topologyErrorFeature =
enumTopologyErrorFeature.Next()) != null&&
                        topologyErrorFeature.TopologyRuleType==ruleType)
                    {
                        dataRow = dataTable.NewRow();
                        dataRow[0] = i++;
                        dataRow[1] = topologyErrorFeature.ErrorID;
                        dataRow[2] = topologyErrorFeature.TopologyRuleType;
                        dataRow[3] = topologyErrorFeature.ShapeType;
                        dataRow[4] = topologyErrorFeature.OriginClassID;
                        dataRow[5] = topologyErrorFeature.OriginOID;
                        dataRow[6] = topologyErrorFeature.DestinationClassID;
                        dataRow[7] = topologyErrorFeature.DestinationOID;
                        dataTable.Rows.Add(dataRow);
```

```
                }
            }
            return dataTable;
        }
//显示所有错误要素
        public DataTable DisplayAllErrorFeatures(ITopology topology, IEnvelope searchExtent)
        {
            //获取坐标系
            IErrorFeatureContainer errorFeatureContainer = (IErrorFeatureContainer)topology;
            IGeoDataset geoDataset = (IGeoDataset)topology;
            ISpatialReference spatialReference = geoDataset.SpatialReference;
            ITopologyRuleContainer topologyRuleContainer = (ITopologyRuleContainer)topology;
            //遍历拓扑规则
            IEnumRule enumRule = topologyRuleContainer.Rules;
            enumRule.Reset();
            IRule rule = null;
            DataRow dataRow;
            int i = 1;
            DataTable dataTable = new DataTable();
            DataColumn dataColunm = new DataColumn();
            dataColunm.ColumnName = "序号";
            dataColunm.DataType = System.Type.GetType("System.Int32");
            dataTable.Columns.Add(dataColunm);
            dataColunm = new DataColumn();
            dataColunm.ColumnName = "错误ID";
            dataColunm.DataType = System.Type.GetType("System.Int32");
```

```
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
dataColunm.ColumnName =   "规则类型";
dataColunm.DataType = System.Type.GetType("System.String");
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
dataColunm.ColumnName =   "形状";
dataColunm.DataType = System.Type.GetType("System.String");
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
dataColunm.ColumnName = "原要素集ID";
dataColunm.DataType = System.Type.GetType("System.Int32");
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
dataColunm.ColumnName =   "原OID";
dataColunm.DataType = System.Type.GetType("System.Int32");
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
dataColunm.ColumnName = "目标要素集ID";
dataColunm.DataType = System.Type.GetType("System.Int32");
dataTable.Columns.Add(dataColunm);
dataColunm = new DataColumn();
dataColunm.ColumnName = "目标OID";
dataColunm.DataType = System.Type.GetType("System.Int32");
dataTable.Columns.Add(dataColunm);
while ((rule = enumRule.Next()) != null)
{
    //获取当前拓扑规则的拓扑错误并遍历
```

```
                ITopologyRule topologyRule = (ITopologyRule)rule;
                IEnumTopologyErrorFeature enumTopologyErrorFeature =
errorFeatureContainer.get_ErrorFeatures(spatialReference, topologyRule,
searchExtent, true, true);
                ITopologyErrorFeature topologyErrorFeature = null;
                while ((topologyErrorFeature =
enumTopologyErrorFeature.Next()) != null)
                {
                    dataRow = dataTable.NewRow();
                    dataRow[0] = i++;
                    dataRow[1] = topologyErrorFeature.ErrorID;
                    dataRow[2] = topologyErrorFeature.TopologyRuleType;
                    dataRow[3] = topologyErrorFeature.ShapeType;
                    dataRow[4] = topologyErrorFeature.OriginClassID;
                    dataRow[5] = topologyErrorFeature.OriginOID;
                    dataRow[6] = topologyErrorFeature.DestinationClassID;
                    dataRow[7] = topologyErrorFeature.DestinationOID;
                    dataTable.Rows.Add(dataRow);
                }
            }
            return dataTable;
        }
```

10.添加读取拓扑事件

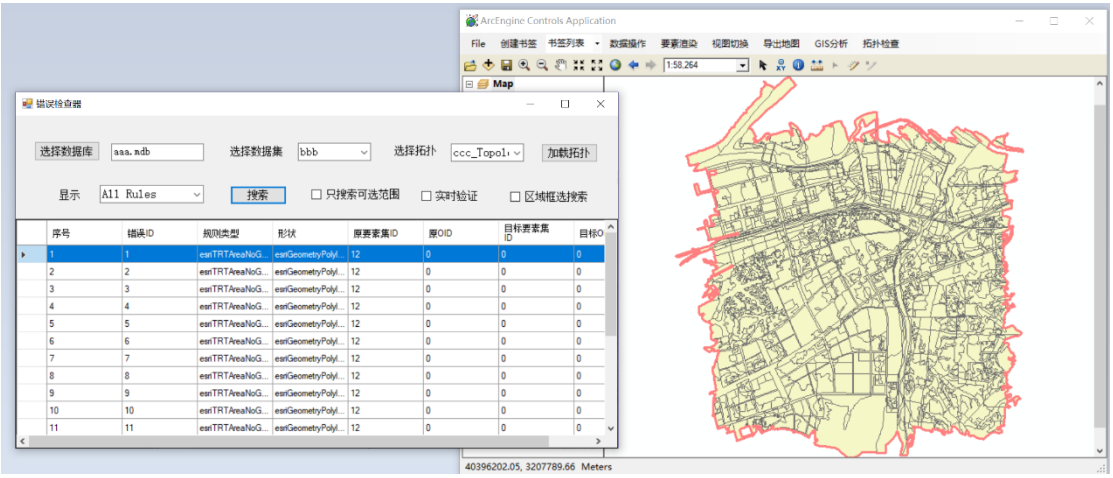
为“读取拓扑”菜单生成“点击”事件响应，实现 ReadTopology 窗体显示，代码如下：

```
        ReadTopology readTopology;
        private void tbReadTopology_Click(object sender, EventArgs e)
        {
```

```
            readTopology = new ReadTopology(axMapControl1);
            readTopology.Show();
        }
```

11.实现效果

打开创建好的拓扑，拓扑自动加载到地图上，点击搜索按钮搜索所有的错误信息。



2.3.2 显示单个拓扑错误

显示单个拓扑错误是在 ReadTopology 窗体和 AxMapControl1 控件中交互实施的，在 ReadTopology 窗体数据表中选中单个拓扑错误条目，地图上会跳转到对应的错误要素位置，并闪烁提示。同时当视图偏离该要素时也可以右键错误条目，点击“缩放到图层”进行跳转。

1. 添加“数据表”条目点击事件

为“数据表”生成条目点击事件响应函数，实现显示单个拓扑错误功能。选中拓扑错误条目时根据错误 ID 找到要素并跳转视图，闪烁提示，代码如下：

```
//选中表格内容进行缩放及闪烁
        private void GridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (GridView1.CurrentRow == null) return;
            DataGridViewRow dgvr = GridView1.CurrentRow;
```

```
            int errorID = Convert.ToInt32(dgvr.Cells["错误
ID"].Value.ToString());
            try
            {
                feature = top.GetErrorFeatureByID(topology, errorID);
            }
            catch
            {
                return;
            }
            if (feature == null)
                return;
            else
            {
                ZoomToGeometry(feature.Shape, MapContro1.ActiveView);
                ITopologicalOperator ipTO =
(ITopologicalOperator)feature.Shape;
                IGeometry geometry = ipTO.Buffer(0.5);
                MapContro1.FlashShape(geometry, 2, 300, null);
            }
        }
```

2. 添加“右键缩放”点击事件

为“右键缩放”生成点击事件响应函数，实现缩放到要素。右键拓扑错误条目点击缩放到要素，跳转并闪烁提示，代码如下：

```
//右键缩放及闪烁
        private void ZoomToLayer_Click(object sender, EventArgs e)
        {
            if (GridView1.CurrentRow == null) return;
```

```
            DataGridViewRow dgvr = GridView1.CurrentRow;
            int errorID = Convert.ToInt32(dgvr.Cells["错误
ID"].Value.ToString());
            feature = top.GetErrorFeatureByID(topology, errorID);
            if (feature == null)
            {
                MessageBox.Show("未能获取到要素！");
                return;
            }
            ZoomToGeometry(feature.Shape, MapContro1.ActiveView);
            ITopologicalOperator ipTO = (ITopologicalOperator)feature.Shape;
            IGeometry geometry = ipTO.Buffer(0.5);
            MapContro1.FlashShape(geometry, 2, 300, null);
        }
```

3. 增加“缩放到要素”函数

实现缩放到要素，并刷新，代码如下：

```
        //缩放到要素
        public static void ZoomToGeometry(IGeometry geometry, IActiveView
activeView)
        {
            if (geometry == null)
            {
                MessageBox.Show("未能获取到要素！");
                return;
            }
            IEnvelope env = geometry.Envelope;
            env.Expand(2, 2, true);
            activeView.Extent = env;
```

```
            activeView.Refresh();
            activeView.ScreenDisplay.UpdateWindow();
        }
```

4. 增加“根据错误 ID 查找要素”函数

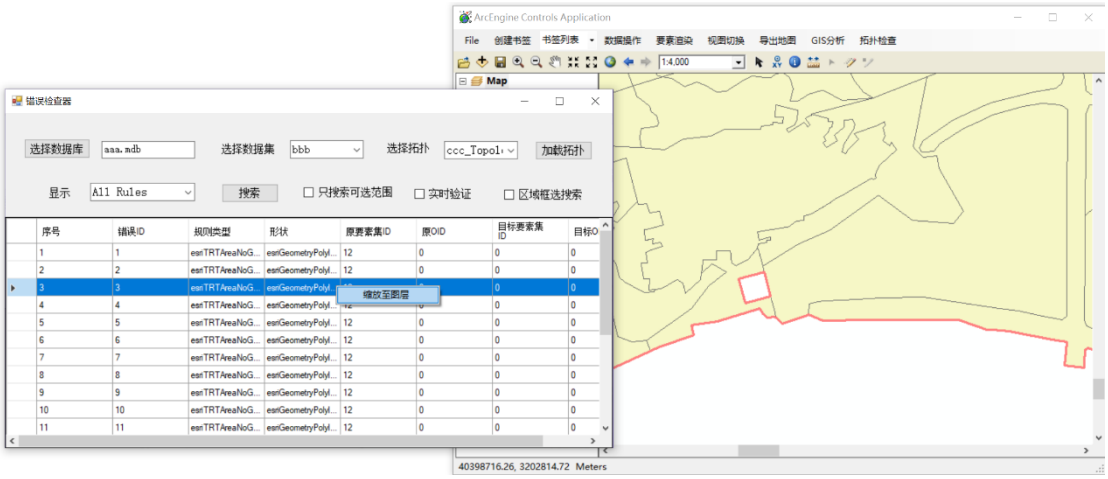
选中拓扑错误条目找到对应的拓扑错误要素需要在 Topology 类中添加函数，是通过错误 ID 来找到错误要素的，代码如下：

```
//返回指定错误序号的错误要素
        public IFeature GetErrorFeatureByID(ITopology topology, int errorID)
        {
            IFeature feature=null;
            //获取坐标系
            IErrorFeatureContainer errorFeatureContainer =
(IErrorFeatureContainer)topology;
            IGeoDataset geoDataset = (IGeoDataset)topology;
            ISpatialReference spatialReference = geoDataset.SpatialReference;
            ITopologyRuleContainer topologyRuleContainer =
(ITopologyRuleContainer)topology;
            //遍历拓扑规则
            IEnumRule enumRule = topologyRuleContainer.Rules;
            enumRule.Reset();
            IRule rule = null;
            while ((rule = enumRule.Next()) != null)
            {
                //获取当前拓扑规则的拓扑错误并遍历
                ITopologyRule topologyRule = (ITopologyRule)rule;
                IEnumTopologyErrorFeature enumTopologyErrorFeature =
errorFeatureContainer.get_ErrorFeatures(spatialReference, topologyRule,
geoDataset.Extent, true, true);
```

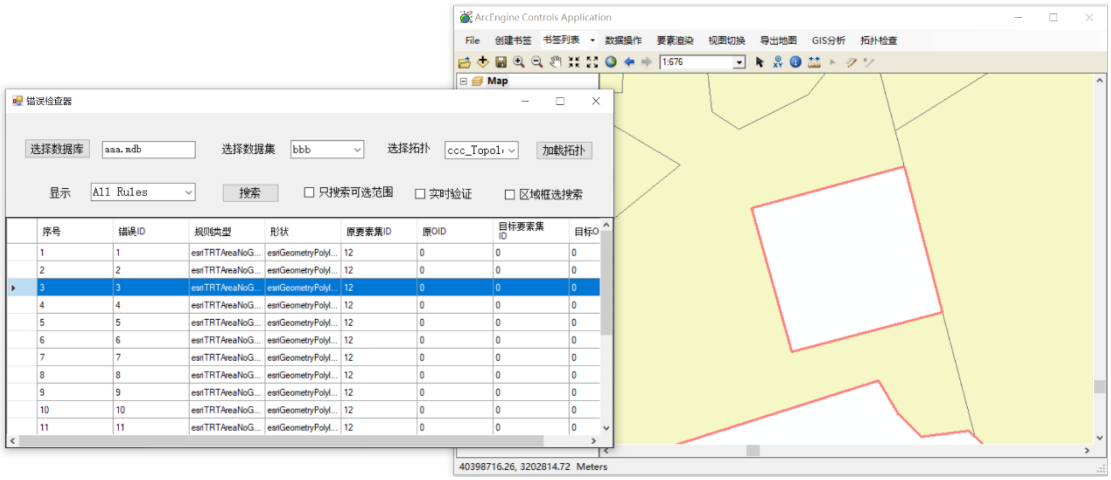
```
                ITopologyErrorFeature topologyErrorFeature = null;
                while ((topologyErrorFeature =
enumTopologyErrorFeature.Next()) != null)
                {
                    if (topologyErrorFeature.ErrorID == errorID)
                    {
                        feature = (IFeature)topologyErrorFeature;
                        break;
                    }
                }
            }
            return feature;
        }
```

5.实现效果

点击选中记录后右键缩放至指定要素。



左键单击记录，被选中的几何将闪烁 0.3 秒。



2.3.3 实时验证和区域框选等

实时验证和区域框选等使得拓扑错误查询功能更为完善。

1. 增加“验证拓扑”函数

实时验证需要在 Topology 中添加一个 ValidateTopology 函数，代码如下：

```
public void ValidateTopology(ITopology topology, IEnvelope envelope)
        {
            // Get the dirty area within the provided envelope.
            IPolygon locationPolygon = new PolygonClass();
            ISegmentCollection segmentCollection =
(ISegmentCollection)locationPolygon;
            segmentCollection.SetRectangle(envelope);
            IPolygon polygon = topology.get_DirtyArea(locationPolygon);
            // If a dirty area exists, validate the topology.
            if (!polygon.IsEmpty)
            {
                // Define the area to validate and validate the topology.
                IEnvelope areaToValidate = polygon.Envelope;
                IEnvelope areaValidated =
topology.ValidateTopology(areaToValidate);
```

```
        }
    }
```

2. 添加“区域框选搜索”勾选状态改变事件

为“区域框选搜索”生成勾选状态改变事件响应函数，判断是否更新包围盒。

区域框选搜索需要新声明变量作为判断依据，在事件中判断，代码如下：

```
    public bool check=false;
    private void cbZoneCheck_CheckedChanged(object sender, EventArgs e)
    {
        if (cbZoneCheck.Checked)
            check = true;
        else check = false;
    }
```

3. 增加“空间检查”函数

区域框选搜索需要非事件响应的函数用以实时的区域错误搜索，代码如下：

```
public void ZoneCheck(IEnvelope env)
    {
        DataTable dataTable = new DataTable();
        IGeoDataset g = (IGeoDataset)topology;
        if (cbCurrentExtent.Checked)
        {
            cbCurrentExtent.Checked = false;
        }
        //实时验证框被勾选后进行实时验证
        if (cbvaliTopo.Checked)
        {
```

```
                top.ValidateTopology(topology, env);
                MapContro1.Refresh();
            }

            switch (cbTopologyRule.Text)
            {
                case "esriTRTAreaNoOverlap":
                    dataTable = top.DisplayErrorFeatureByRuleType(topology,
esriTopologyRuleType.esriTRTAreaNoOverlap, env);
                    break;
                case "esriTRTAreaNoGaps":
                    dataTable = top.DisplayErrorFeatureByRuleType(topology,
esriTopologyRuleType.esriTRTAreaNoGaps, env);
                    break;
                case "esriTRTAreaNoOverlapArea":
                    dataTable = top.DisplayErrorFeatureByRuleType(topology,
esriTopologyRuleType.esriTRTAreaNoOverlapArea, env);
                    break;
                default:
                    dataTable = top.DisplayAllErrorFeatures(topology, env);
                    break;
            }
            GridView1.DataSource = dataTable;
            ZoomToEnvelope(env, MapContro1.ActiveView);
        }
```

4. 补充地图控件鼠标按下事件

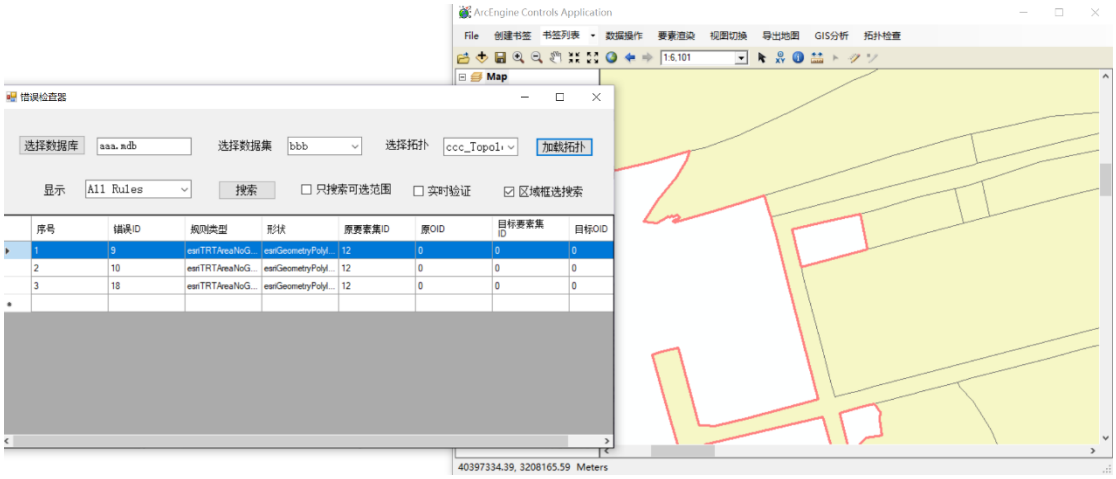
对 axMapControl1_OnMouseDown 的函数做出修改，当其 TrackRectangle() 改变时，刷新包围盒，将包围盒作为参数传入 ReadTopology 中的 ZoneCheck 这

一非事件响应函数进行实时的区域错误搜索。

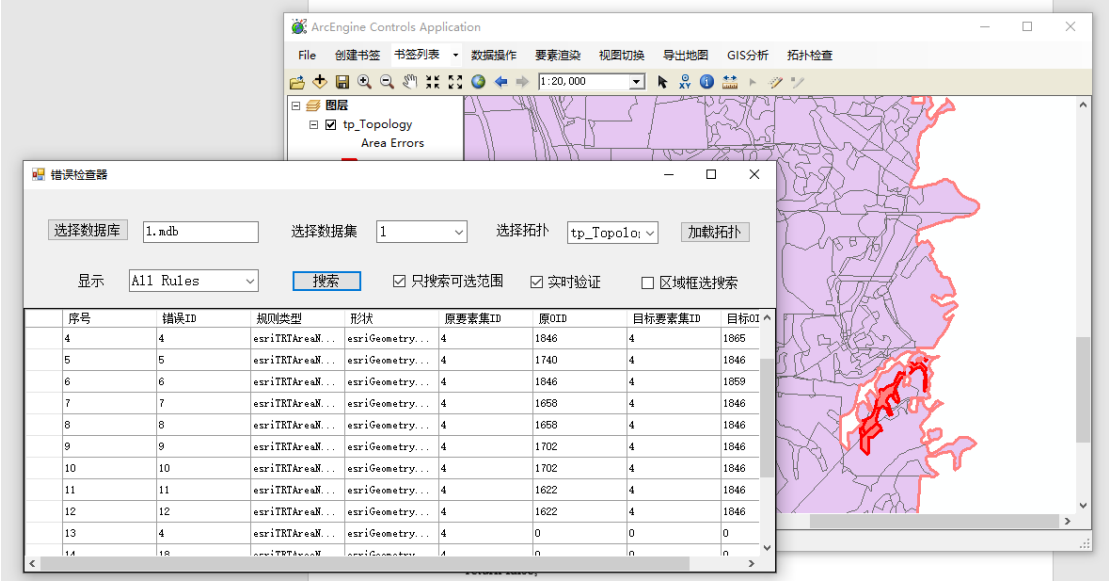
```
private void axMapControl1_OnMouseDown(object sender,
IMapControlEvents2_OnMouseDownEvent e)
        {
            if(readTopology!=null&&readTopology.check)
            {
                IEnvelope env = axMapControl1.TrackRectangle();
                env.Expand(2, 2, true);
                readTopology.ZoneCheck(env);
            }
```

5. 实现效果

勾选区域框选搜索后，在地图控件上使用鼠标框选范围，更新错误信息表，并缩放到框选范围。



要素类被编辑后，再次验证拓扑，正常显示拓扑错误。



三、总结与反思

3.1 总结

本程序基本达到要求，各项功能工作正常，操作流程简洁明了，程序效率较高。

3.2 不足之处

（1）程序中尚存在部分 BUG，操作不当会引发程序崩溃。后续可以进一步完善代码中的异常处理部分。

（2）程序未能实现面自相交的拓扑检查，原因是 ITopologicalOperator5 接口中并未包含面自相交的规则。实现面自相交检查的方法有以下两种：

将面要素转变为线要素，再使用线不能自相交的规则。

判断图形是否是已知图形，代码如下：

```
// 判断是否自相交；TRUE 表示自相交，FALSE 表示不自相交
public static bool IsSelfCross(IGeometry pGeometry)
{
    ITopologicalOperator3 pTopologicalOperator2 = pGeometry as ITopologicalOperator3;
    pTopologicalOperator2.IsKnownSimple_2 = false;
    esriNonSimpleReasonEnum reason =
```

```
esriNonSimpleReasonEnum.esriNonSimpleOK;
        if (!pTopologicalOperator2.get_IsSimpleEx(out reason))
        {
            if (reason ==
esriNonSimpleReasonEnum.esriNonSimpleSelfIntersections)
            {
                return true;
            }
        }
        return false;
 }
/*
pTopologicalOperator2.IsKnownSimple_2 = false;//布尔型，指示此几何图形是否是已知的（或假设）是拓扑正确。这里赋值 false,就是非已知的几何图形。
pTopologicalOperator2.get_IsSimpleEx(out reason)//返回布尔值，指示该几何图形是否为简单的。如果返回的是 false，则可以对输出的"reason"参数检查审查。
*/
```

（3）界面设计不够美观，操作流程略显复杂。创建拓扑这一步骤分成了四个小的步骤：创建数据库、创建数据集、添加要素类、创建拓扑。每个小步骤之间没有明确的操作提示，对于初步接触者不够友好。

（4）代码重复利用率较低。在读取拓扑的相关代码中，多次用到了 DataTable 的导出，但代码未能重复利用。

（5）部分操作的流程设计有问题。比如在“添加要素类”过程中，选择了 shp 文件后就直接进行了添加，而非在点击确定后进行添加，这使得用户无法更改选项，也增加了出现 BUG 的几率。

四、附件

1.MapControlApplication1.rar 压缩包内是 VS 工程项目文件。