



# 深入浅出

# Geodatabase 编程

## ——基本理论与最佳实践

张煜

zhangyu.gis@gmail.com



# 目的

- 深入理解 Geodatabase 编程中的常用的 ArcObjects 对象
- 掌握数据库功能编程最佳实践
- 提高系统效率(Efficiency)和鲁棒性(Robustness)

# 假设



- 您具备 .....
- ArcGIS 使用经验
- Geodatabase 基本知识
- GDB API 编程经验
- C# 或 C++ 或 Java... 编程基础



# 内容提纲

- Geodatabase 回顾
- Geodatabase 编程常见操作
  - 数据库连接
  - 表、要素类访问、创建
  - 数据增删改查
- ArcObjects + .NET 编程要点



# Geodatabase 回顾



# 什么是 Geodatabase ?



Geodatabase = Geog*raphy* + Database

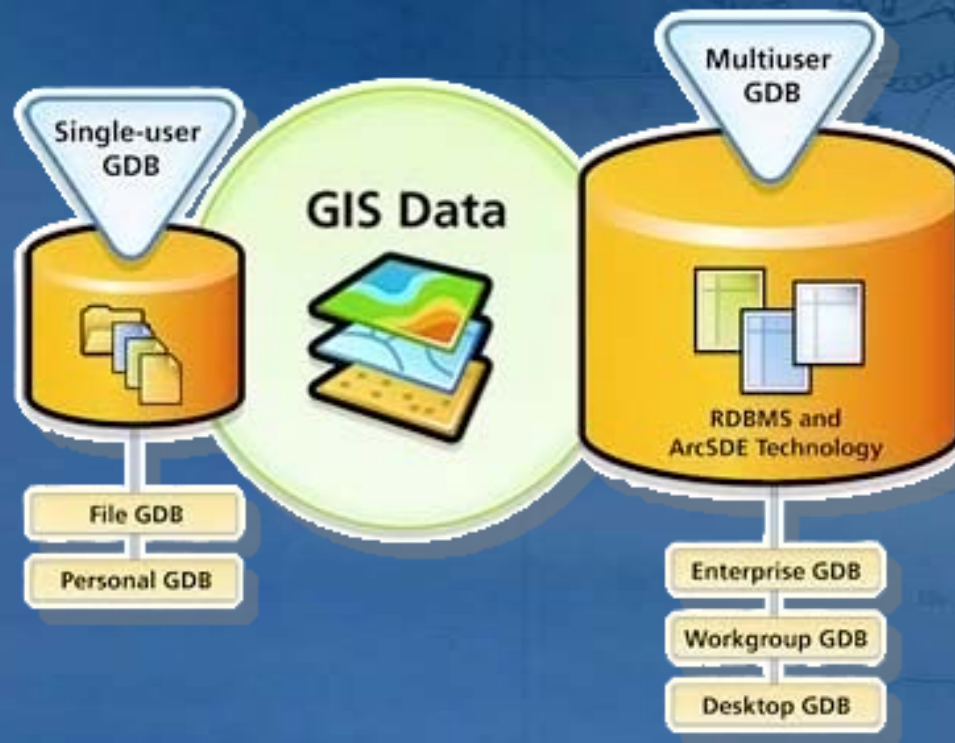
- 一种数据结构、数据格式  
the native **data structure** for ArcGIS and is the primary **data format** used for editing and data management
- 一个数据集 ( Dataset ) 集合的物理实例  
a physical instance of a collection of datasets



# Geodatabase 的架构

- 以一系列简单表表达、管理
  - 地理信息数据对象
  - 空间关系、GIS行为、空间完整性规则
- 事务模型 ( Transaction Model )
  - 管理 GIS 数据 workflow
- 直接支持多种地理数据
  - Geodatabase 物理格式
  - shapefile、coverage
  - CAD、TIN、GML等

# Geodatabase 的类型







# Geodatabase 的类型比较 ( 9.3 )

	Personal GDB	File GDB	ArcSDE GDB
存储格式	微软 Access	包含若干二进制文件的文件夹	DBMS
存储容量	2 GB	1 TB 每个表*	版本决定
平台	Windows	跨平台	版本决定
用户数	单读单写	单读多写	多读多写

# ArcSDE 版本比较 ( 9.3 )



	Personal	Workgroup	Enterprise
所属产品	ArcGIS Desktop(ArcEditor/ ArcInfo) ArcGIS Engine	ArcGIS Server (Workgroup)	ArcGIS Server (Enterprise)
并发用户量	3	10	DBMS决定
DBMS	SQL Server Express ( 1G RAM, 1 CPU )	SQL Server Express ( 1G RAM, 1 CPU )	Oracle/SQL Server/DB2/Infomix/PostgreSQL
存储容量	最大4G	最大4G	无限制



# 数据库选择——最佳实践

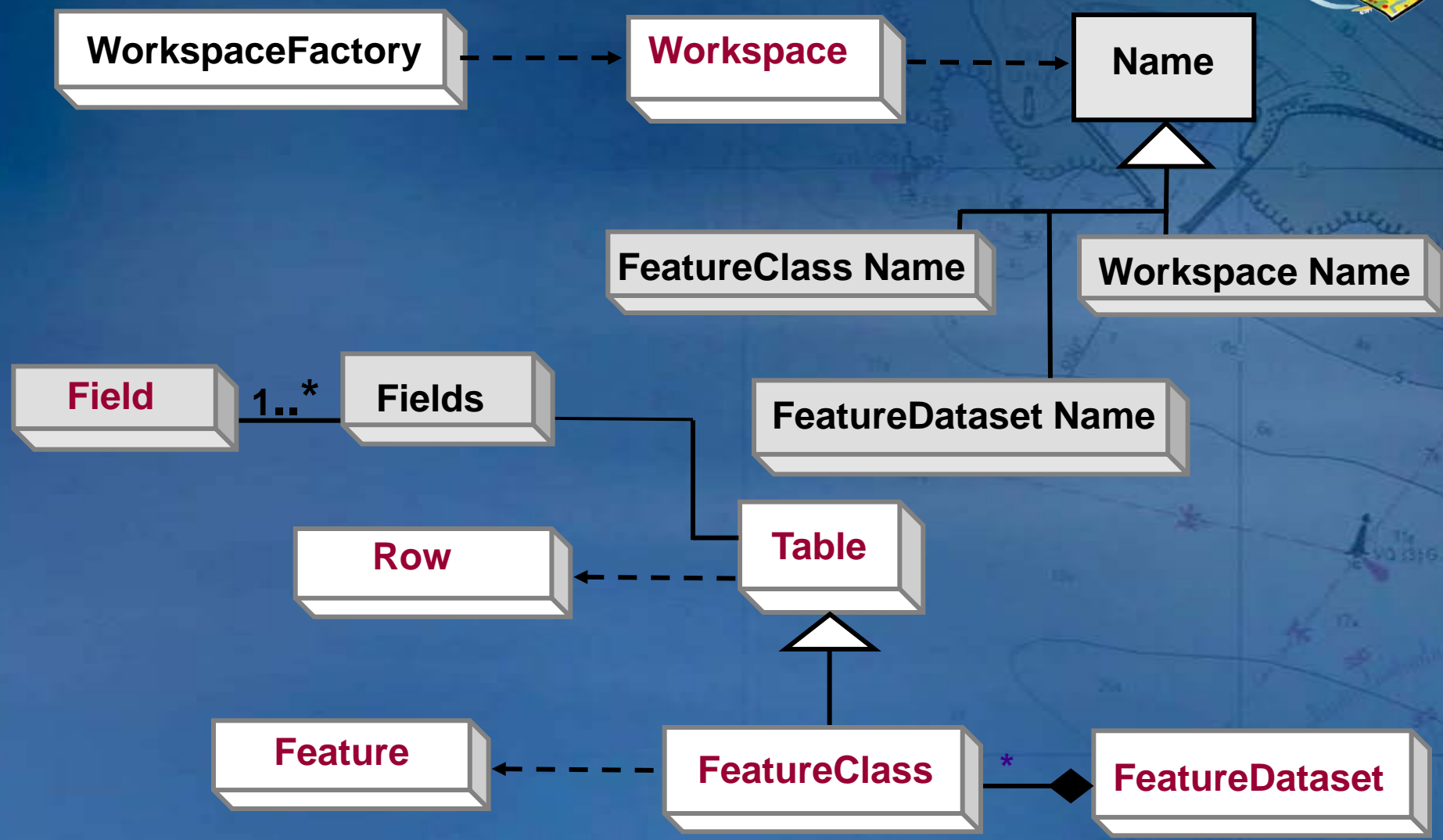
- 多种类型组合使用
  - 发布、编辑分库
- 少量用户读操作建议使用 File GDB
- 必须使用 ArcSDE GDB 的场景：
  - 用到Version、Replica、Archive 功能
  - 多用户管理
  - 并发访问
  - .....



# Geodatabase 编程常见对象



# Geodatabase 对象模型图

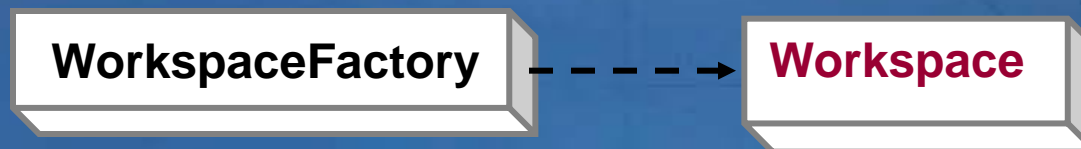






# Workspace & WorkspaceFactory

- Factory Pattern
- Workspace
  - 一个数据库或数据源
  - 包含多个 dataset
- WorkspaceFactory
  - 工作空间调度者
  - 单例对象
- WorkspaceFactory 类型决定 Workspace 类型

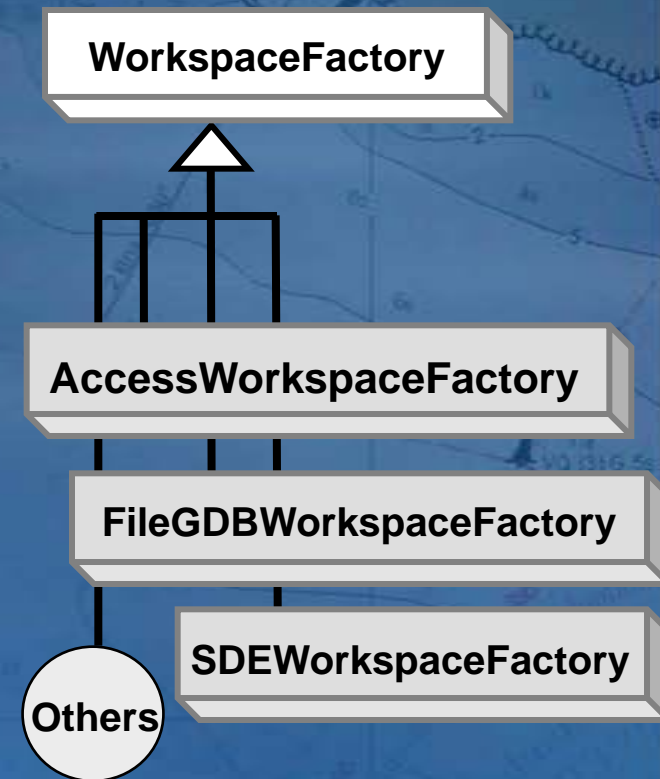




# 数据库访问

## 1. 根据要访问的 Geodatabase 类型实例化相应的 WorkspaceFactory

```
// File GDB
IWorkspaceFactory2 wsf = new
FileGDBWorkspaceFactoryClass();
// Personal GDB
IWorkspaceFactory2 wsf = new
AccessWorkspaceFactoryClass();
// SDE
IWorkspaceFactory2 wsf = new
SdeWorkspaceFactoryClass();
// Scratch GDB
IScratchWorkspaceFactory2 wsf = new
ScratchWorkspaceFactoryClass();
// InMemory GDB
IWorkspaceFactory2 wsf = new
InMemoryWorkspaceFactoryClass();
```





# 数据库访问 (续)

## 2. 存在性、正确性检查

- IWorkspaceFactory2.IsWorkspace()

## 3. 连接数据库

- IWorkspaceFactory2.Open\*()
  - Open()
  - OpenFromFile()
  - OpenFromString()



# Personal / File Geodatabase 访问

```
// 实例化工作空间工厂
IWorkspaceFactory2 wsf = new AccessWorkspaceFactoryClass();
// 工作空间
IWorkspace ws;

// geodatabase 全路径 (含扩展名)
string fileName = @"C:\Data\PersonalGDB.mdb"; // Personal
// string fileName = @"C:\Data\FileGDB.gdb";   // File

// 方法一
ws = wsf.OpenFromFile(fileName, 0);

// 方法二
IPropertySet2 propertySet = new PropertySetClass();
propertySet.SetProperty("DATABASE", fileName);
ws = wsf.Open(propertySet, 0);

// 方法三
ws = wsf.OpenFromString("DATABASE=" + fileName, 0);
```





# ArcSDE Geodatabase 访问

```
// 实例化工作空间工厂
IWorkspaceFactory2 workspaceFactory = new SdeWorkspaceFactoryClass();
// 工作空间
IWorkspace ws;

// 方法一
string connectionFileName = @"C:\Connection to localhost.sde";
ws = workspaceFactory.OpenFromFile(connectionFileName, 0);

// 方法二
// 设置所需连接属性
IPropertySet propertySet = new PropertySetClass();
propertySet.SetProperty("SERVER", server);
propertySet.SetProperty("INSTANCE", instance);
propertySet.SetProperty("DATABASE", database);
propertySet.SetProperty("USER", user);
propertySet.SetProperty("PASSWORD", password);
propertySet.SetProperty("VERSION", version);
ws = workspaceFactory.Open(propertySet, 0);

// 方法三
string connectionString = @"SERVER=ESRI;INSTANCE=5151;USER=sde; _
    PASSWORD=sde;VERSION=sde.DEFAULT";
ws = workspaceFactory.OpenFromString(connectionString, 0);
```





# Workspace 连接池

- 每一 WorkspaceFactory 维护一个当前已连接、激活的 Workspace 连接池
- 每次调用 Open\* 方法，首先检查连接池中是否存在打开的 workspace
  - 如果有，直接引用
  - 如果没有，连接数据库，并引用





# Dataset 访问、创建

- IFeatureWorkspace.Open\*() 参数：名称
  - OpenTable()
  - OpenFeatureClass()
  - OpenFeatureDataset()
  - OpenRelationshipClass()
- Dataset 创建是一种 DDL 命令
  - DDL ( Data definition language )：修改数据库 schema 的数据库命令，包括表、字段等结构编辑
  - 不要在编辑会话 ( Edit Session ) 中执行
- IFeatureWorkspace.Create\*()
  - CreateTable()
  - CreateFeatureClass()
  - CreateFeatureDataset()
  - CreateRelationshipClass()

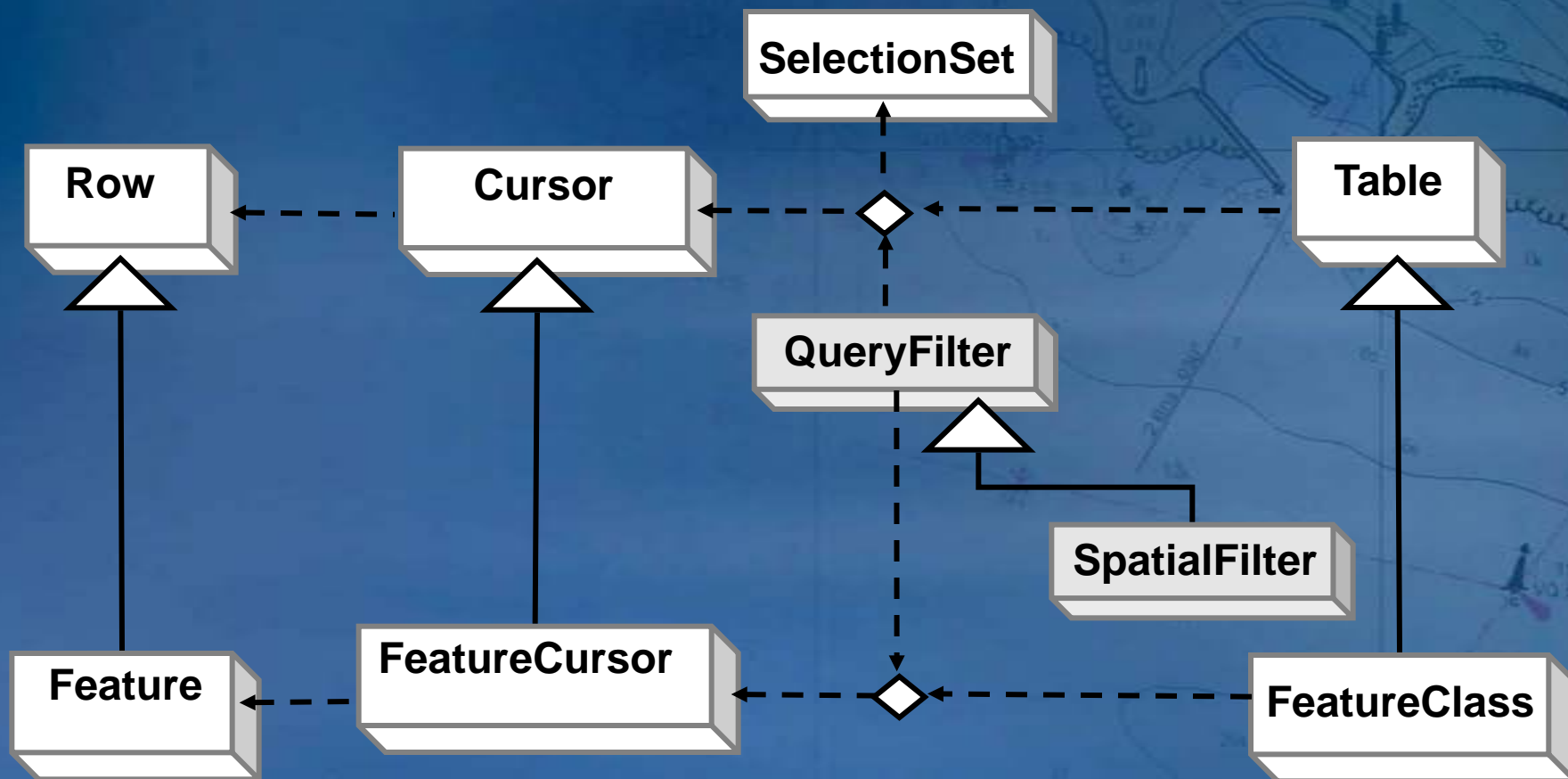


# 对象类、要素类 创建

- name : string
  - 存在性检查 IWorkspace2.NameExist
- fields : IFields
  - 必须字段 ( Required Fields )
    - 表 : OID
    - 要素类 : OID、Shape
  - 可以通过 IObjectClassDescription.RequiredFields 创建



# 数据访问





# 数据查询、更新

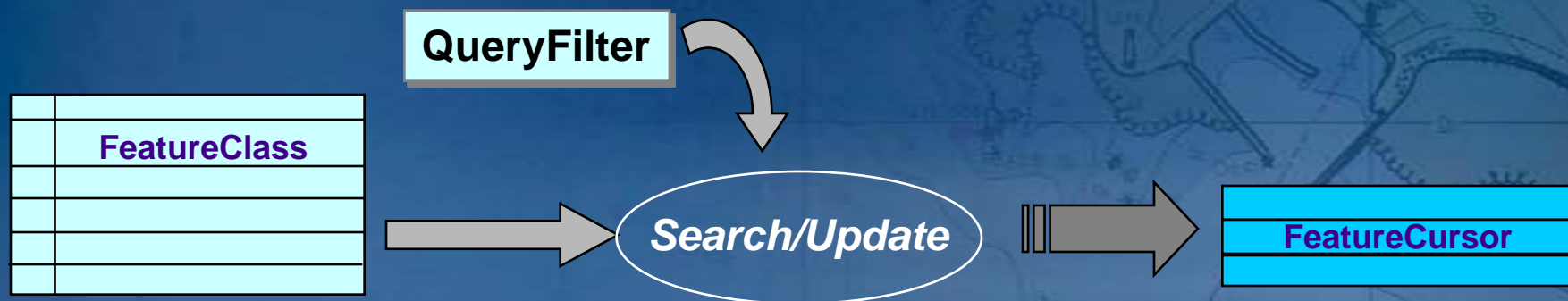


Table + QueryFilter = Cursor -> Row  
FC + SpatialFilter = FeatureCursor -> Feature

```
ITable.Search ( QueryFilter, Recycling)  
ITable.Update ( QueryFilter, Recycling)
```

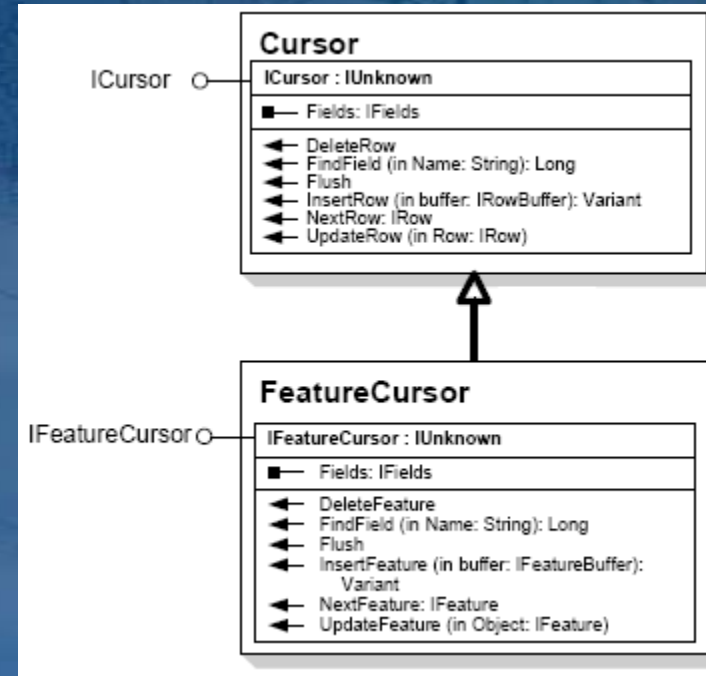


# Cursor



# 游标 ( Cursor )

- 用一个对象管理一个记录集合
- 可用于数据的查询、编辑
- 类型：
  - 由获取游标的方法决定
  - 类游标
    - ITable.Search -> 查询游标：全能
    - ITable.Insert -> 插入游标：增加
    - ITable.Update -> 更新游标：删改
  - 查询定义游标
    - IQueryDef.Evaluate -> 查询定义游标：查询
    - 多表





# 使用游标访问记录

- 初始化游标时，游标指针指“第一条”记录之上
- NextRow/NextFeature 操作返回记录（行、要素）

```
ICursor myCursor;  
myCursor = myTable.Search(queryFilter, true);
```

myCursor 初始化

内存 回收

```
myRow = myCursor.NextRow();  
myRow = myCursor.NextRow();  
myRow = myCursor.NextRow();  
...  
myRow = myCursor.NextRow();
```

Row 3

Row 2

Row 3

null

遍历完所有记录，返回 null



# 内存回收 True or False





# 内存回收——最佳实践

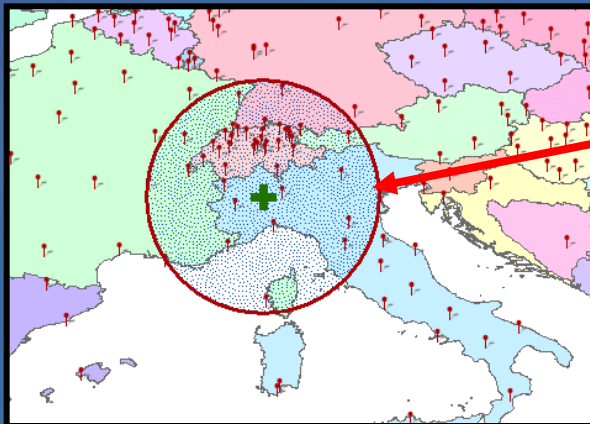
- 内存回收
  - 大多数查询 search cursor
- 内存不回收
  - 查询结果需要保持纪录唯一性时
    - 如以 ArrayList 保存所有记录结果
  - 编辑



# filter 过滤器

# QueryFilter & SpatialFilter

- QueryFilter
  - 仅属性过滤
  - SubFields 或 AddField 约束返回结果
  - WhereClause : 相当于 SQL 中的 Where 子句
- SpatialFilter
  - 属性过滤 和 空间过滤
    - 如 : 过滤与中国的邻国中 , 面积大于100万平方公里的国家



## Geometry 几何体

- Point
- Line
- Polygon
- Envelope
- GeometryBag



## SpatialRel 空间关系

- Inside
- Contains
- Intersects



例如：查询【Cities】表中人口大于100万的城市，并以城市名排序，结果中包含城市名和人口数

Select Name,Population From Cities Where  
Population>1000000 OrderBy Name

Select 【字段1, 字段2...】	----- SubFields
From 【表】	----- Table/FeatureClass
Where 【where子句】	----- WhereClause
OrderBy 【字段1】	----- ??????



# 查询结果排序

- 方法一：使用 ITableSort 接口
  - Cursor : ICursor 待排序的结果集
  - Rows : ICursor 排序后的结果集
  - Ascending 升序、降序
  - CaseSensitive 大小写敏感
  - Fields 要排序的字段，先后顺序决定排序优先级
- 方法二：使用 IQueryFilterDefinition 接口
  - PostfixClause
    - OrderBy
  - 9.3版本中，File Geodatabase 不支持





# 如何获取最大值



# Where语句的SQL支持

- 函数
  - 统计 ( AVG, COUNT, MIN, MAX, SUM ... )
  - 数值运算 ( SIN, COS, FLOOR, LOG, CEILING, POWER ... )
  - 日期 ( CURRENT\_DATE ... )
  - 字符串 ( LOWER, UPPER ... )
- 子句
  - "GDP2006" > (SELECT MAX("GDP2005") FROM countries)
- 服务端查询，性能较优
- 不同的 Geodatabases 对 SQL 的支持不同\*

\*参考ArcGIS Desktop Help -> SQL Reference

# 查询——最佳实践



- 尽可能使用 File Geodatabase 做查询
- 将空间查询转换为属性查询
- 设置 IQueryFilter.SubFields, 使仅返回所需的字段
- 用于空间查询的几何体要有空间参考
- 查询回收内存



# 数据创建（记录法）

- 步骤

1. IFeatureClass.CreateFeature : IFeature
  - 此时已在表中增加了一条记录
2. 设置属性值
3. 创建几何体、设定 IFeature.Shape
4. 调用 IFeature.Store 保存值
  - 仅仅是保存记录每个字段的值（包含几何字段）

- 适用场景

- 交互编辑
- 少量编辑



# 数据创建（代码）

```
// 创建要素
IFeature feature = featureClass.CreateFeature();

// 设置属性值
int nameFieldIndex = featureClass.FindField("Name");
feature.set_Value(nameFieldIndex, "中国");

// 设置几何体
feature.Shape = point;

// 保存
feature.Store();
```





# 数据创建（游标法）

- 大量简单数据批量导入
  - 使用 Cursor
- 基本步骤（以Row为例，Feature类似）
  - 使用 ITable.Insert(true) 方法创建Insert Cursor
  - 使用 ITable.CreateRowBuffer 方法创建行缓冲\*
  - 行缓冲相当于一个行对象，直接修改其值
  - 每次修改后，使用 ICursor.InsertRow插入行缓冲
  - 周期性使用 ICursor.Flush 将缓冲中的所有行写入表

\*只有在编辑会话中使用缓冲才有效



# 插入游标

- 用于批量插入记录
  - 对于简单数据，此法比 IFeature.Store 快
    - 绕过事件
  - 对于复杂数据，没有明显差异
    - 需要调用 CreateRow 和 Store 方法
    - 维护行为、复杂关系
  - 缓冲 (buffer) 是关键
    - 客户端缓冲要插入的记录
    - 何时写入服务端数据库
      - 插入游标销毁时
      - 执行 Flush 方法时
      - 建议使用 Flush 方法，易于精确捕捉写入异常



# 数据创建（批量导入代码）

```
public void InsertFeaturesUsingCursor(IFeatureClass featureClass, List<IGeometry>
    geometryList)
{
    IFeatureBuffer featureBuffer = featureClass.CreateFeatureBuffer();
    int typeFieldIndex = featureClass.FindField("TYPE");
    IFeatureCursor insertCursor = featureClass.Insert(true);
    featureBuffer.set_Value(typeFieldIndex, "Primary Highway");
    for (int i = 0; i < 1000; i++)
    {
        featureBuffer.Shape = geometryList[i];
        try
        {
            insertCursor.InsertFeature(featureBuffer);
        }
        catch (COMException comExc)
        {
        }
    }
    try
    {
        insertCursor.Flush();
    }
    catch (COMException comExc)
    {
    }
    finally
    {
        Marshal.ReleaseComObject(insertCursor);
    }
}
```



# 数据修改与删除

- 两种方法无明显差异
- 应使用 内存不回收
- 决定游标类型的四个主要因素：
  - 编辑会话、应用程序、数据复杂情况、Geodatabase类型

	ArcMap	Engine - Simple	Engine - Complex
<b>Inside edit sessions</b>	Search Cursor	ArcSDE: Search Cursor Local GDB: Update Cursor	ArcSDE: Search Cursor Local GDB: Update Cursor
<b>Outside edit sessions</b>	Search Cursor	Update Cursor	ArcSDE: Search Cursor Local GDB: Update Cursor



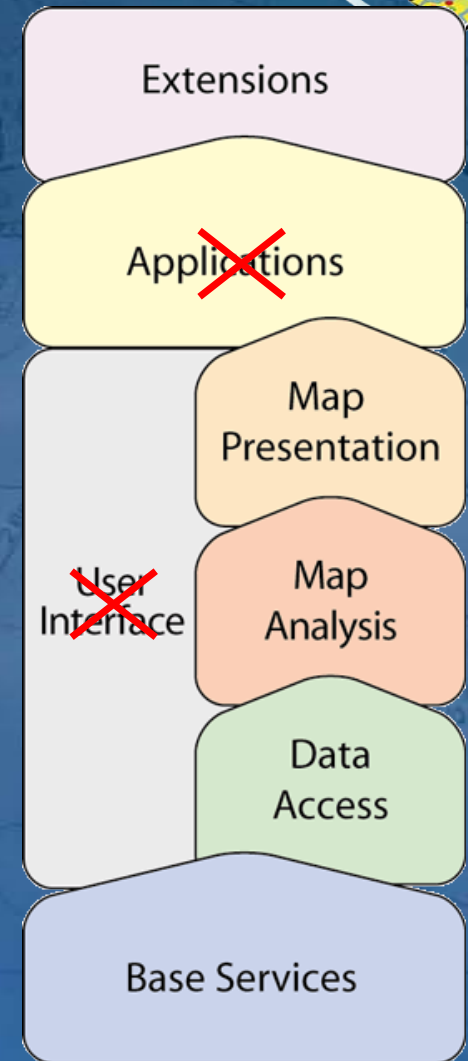
# ArcObjects + .NET 编程要点





# ArcObjects共享

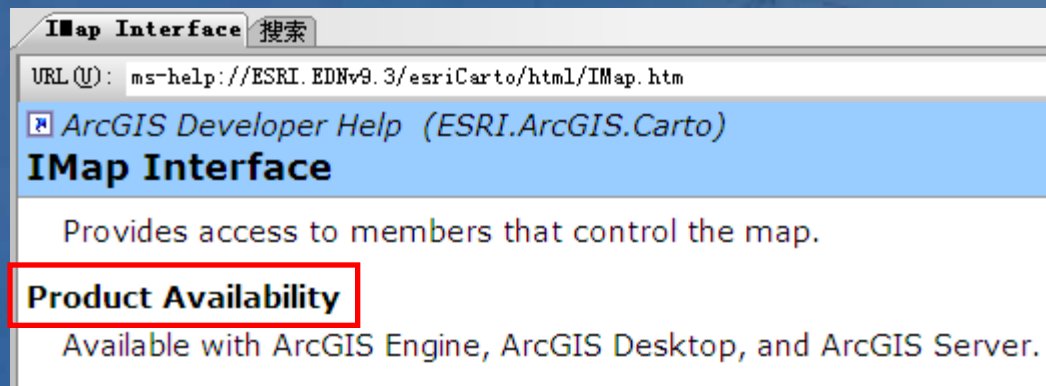
- 共享核心 ArcObjects 类库
  - Carto、Geodatabase、Geometry、DatasourceGDB、Display...
- Desktop “专供” “独享” 类库
  - Applications ( Arc\*: ArcMap、ArcCatalog、ArcGlobe... )
  - User Interface ( \*UI: ArcMapUI、DisplayUI、OutputUI... )
    - 特例: SystemUI





# 如何确定所属产品？

- 方法一：
  - 产品->类库->接口、类
  - 可行，但不符合思维习惯
- 方法二：
  - 接口、类->Product Availability



# 最佳实践



- 先确定接口、类所属产品
- 再进行编码

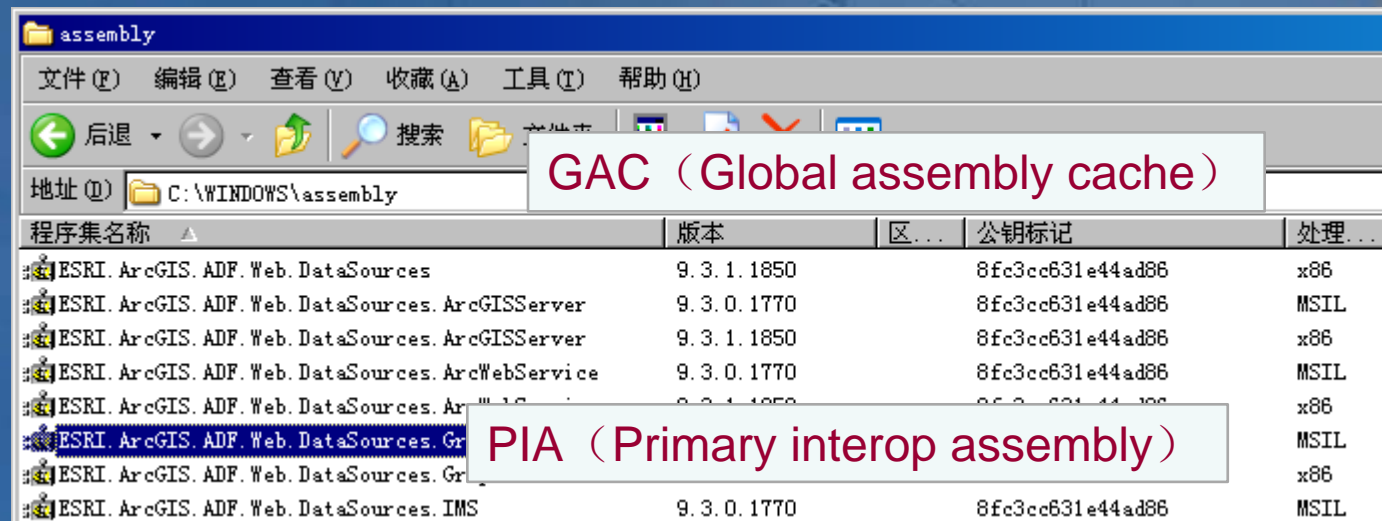


# 编程语言如何选择？



# .NET 中的 COM 对象

- COM代码在 .NET Framework 之外执行
  - 非托管代码
- 托管 ( Managed ) 对象和非托管 ( Unmanaged ) 对象
  - 必须通过互操作程序集 ( Interop Assemblies ) 通讯
  - 无缝通讯
  - 性能损耗







# COM操作性能比较测试

- 测试内容：**100万**次几何操作 ILine.PutCoords()
- 测试环境：3 GHz CPU , .NET Framework 2.0

.NET	C++
约30秒	约5秒



Java  
C++

VB6 .NET

- 工程主体代码以 C# 为主
- 复杂空间计算、分析、循环体以 C++ 编写



# COM对象如何释放？



# COM对象释放

- 每一个 COM 对象必须实现 IUnknown 接口
  - AddRef(), Release(), QueryInterface()
  - RefCount
- 要释放哪些对象
  - 理论上, 所有的 AO 对象
  - Cursor 对象必须释放
  - 由 Cursor 衍生的对象





# COM对象释放 (续)

- 释放方法
  - System.Runtime.InteropServices.Marshal.ReleaseComObject()
  - ESRI.ArcGIS.ADF.ComReleaser.ReleaseCOMObject()  
(彻底释放、推荐方法)
  - 一旦对象使用完立刻释放，特别是循环体内创建的Cursor

```
IQueryFilter queryFilter = new QueryFilterClass();  
queryFilter.WhereClause = "Area = " + i.ToString();  
IFeatureCursor featCursor = featClass.Search(queryFilter, true);  
' 使用 featCursor.  
' ...  
ESRI.ArcGIS.ADF.ComReleaser.ReleaseCOMObject(featCursor);
```

- Cursor 使用后，立刻使用  
ESRI.ArcGIS.ADF.ComReleaser.R  
eleaseCOMObject() 释放

# 总结



- 以上仅是从 AO 角度的讲解，对于 Geodatabase 编程，首选方式是直接调用 GP 工具（ArcToolbox 中的工具）
- AO 是复杂度极高的类库，同一功能会有多种不同的实现方式，这需要以 ArcGIS Desktop 为性能参照，研读文档，多做分析比较

# 感谢聆听！

反馈建议请 mail to:  
[zhangyu.gis@gmail.com](mailto:zhangyu.gis@gmail.com)