

OGC 标准介绍

吴泳锋

2010/6/17

目录

目录.....	1
I. OGC 与 OGC 标准.....	7
• OGC	7
• OGC 标准	8
II. SFS-简单要素标准	11
• 概述	11
• 几何对象模型.....	12
• WKT 描述的几何对象	13
• WKB 描述的几何对象.....	15
• WKT 描述的空间参考	16
• SQL 预定义 schema	18
• SQL 几何对象存储.....	19
• SQL 空间操作.....	21
1. 所有几何对象支持	21
2. Point 对象支持	26
3. Curve 对象支持	27
4. LineString 对象支持.....	27
5. Surface 对象支持.....	27
6. Polygon 对象支持	27

7. GeomCollection 对象支持.....	28
8. MultiCurve 对象支持.....	28
9. MultiSurface 对象支持	28
• ArcGIS 对 SFS 的支持.....	28
III. GML-地理标记语言.....	30
• 概述	30
• GML Schema.....	30
• GML 示例	31
IV. SLD-图层样式描述	35
• 概述	35
• SLD Schema	35
• SLD 简单例子	35
• SE 示例.....	36
V. KML-我从 Google 来	39
• 概述	39
• KML Schema	39
• KML 示例	41
• ArcGIS 对 KML 的支持.....	42
1. ArcToolbox 输出 KML.....	42
2. ArcGIS Server 发布 KML 服务.....	46
VI. OWS-OGC Web 服务通用标准	48

• 概述	48
• 服务涉及的基本元素	48
1. HTTP 请求规则	48
2. HTTP 响应规则	49
3. SOAP	49
• GetCapabilities 操作	50
• 其它一般操作	51
VII. WFS-要素 Web 服务	52
• 概述	52
• WFS 种类与操作	52
• GetCapabilities 操作	53
1. KVP 格式请求	53
2. XML 格式请求	53
3. 响应示例	54
• DescribeFeatureType 操作	57
1. KVP 格式请求	57
2. XML 格式请求	58
3. 响应示例	58
• GetFeature 操作	59
1. KVP 格式请求	59
2. XML 格式请求	60

3. 响应示例.....	61
• Transaction 操作.....	61
1. KVP 格式请求.....	61
2. XML 格式请求.....	62
3. 响应示例.....	64
• ArcGIS Server 对 WFS 的支持.....	64
VIII. WMS-地图 Web 服务	69
• 概述	69
• WMS 种类与操作	69
• GetCapabilities 操作	69
• GetMap 操作	70
• GetFeatureInfo 操作.....	71
• ArcGIS Server 对 WMS 的支持.....	71
IX. WCS-栅格 Web 服务.....	74
• 概述	74
• WCS 的操作.....	74
• GetCapabilities 操作	74
1. KVP 格式请求.....	74
2. XML 格式请求.....	75
3. 响应示例.....	75
• DescribeCoverage 操作	76

1. KVP 格式请求.....	76
2. XML 格式请求.....	76
3. 响应示例.....	77
• GetCoverage 操作	78
1. KVP 格式请求.....	78
2. XML 格式请求.....	78
3. 响应示例.....	79
• ArcGIS Server 对 WCS 的支持	79
X. WMTS-切片地图 Web 服务.....	82
• 概述	82
• WMTS 的原理和操作.....	82
• GetCapabilities 操作	83
1. KVP 格式请求.....	83
2. SOAP 格式请求	84
3. RESTful 格式请求.....	84
• GetTile 操作	84
1. KVP 格式请求.....	84
2. SOAP 格式请求	85
3. RESTful 格式请求.....	86
• GetFeatureInfo 操作.....	86
1. KVP 格式请求.....	86

2. SOAP 格式请求	86
3. RESTful 格式请求.....	87
XI. 附录：ArcGIS 支持的 OGC 标准列表	88

I. OGC 与 OGC 标准

- OGC



<http://www.opengeospatial.org/>

OGC 全称 Open Geospatial Consortium，自称是一个非盈利的、国际化的、自愿协商的标准化组织，它的主要目的就是制定与空间信息、基于位置服务相关的标准。这些标准就是 OGC 的“产品”，而这些标准的用处就在于使不同厂商、不同产品之间可以通过统一的接口进行互操作。

在 GIS 领域，OGC 已经是一个比较“官方”的标准化机构了，它不但包括了 ESRI、Google、Oracle 等业界强势企业作为其成员，同时还和 W3C、ISO、IEEE 等协会或组织结成合作伙伴关系。因此，OGC 的标准虽然并不带有强制性，但是因为其背景和历史的原因，它所制定的标准天然地具有一定的权威性。

所以，我们也可以看到，很多国内的部门或行业要进行地理空间信息的共享或发布时，言必称 OGC 标准，就和这个原因有关。但是，事实上我们对 OGC 和 OGC 标准并不需要盲目崇拜和迷信，从 RESTful 服务规范的长期缺失¹、KML 的空降等可以看出来，OGC 还是有一些缺失和不足的地方。

¹ 这里有篇 OGC 的 CTO 关于此事的解释，有兴趣的可以看看：

http://portal.opengeospatial.org/files/?artifact_id=32222

• OGC 标准

OGC 的标准基本上就是 OGC 所有的成果，而所谓的标准就是一些接口或编码的技术文档。不同的厂商、各种 GIS 产品都可以对照这些文档来定义开放服务的接口、空间数据存储的编码、空间操作的方法等。

除了正式发布的标准 ([OpenGIS® Standards](#)), OGC 的工作成果还包括一些其它类型的文档，比如讨论稿 ([Discussion Papers](#))、抽象规范 ([Abstract Specification](#))、最佳实践文档 ([Best Practices Documents](#))、OGC 参考模型 ([OGC Reference Model \(ORM\)](#))、白皮书 ([White Papers](#)) 等。在 OGC 的工作中，标准的制定可能由待讨论的“讨论稿”开始，在形成一定的统一意见后形成“抽象规范”，进一步具体化到“标准”，在这个过程中同时也可能形成“最佳实践文档”供印证，而“OGC 参考模型”则是描述“抽象规范”、“标准”、“最佳实践文档”之间的关系。当然，其中最关心的还是 OGC 标准文档。

所以，就让我们先看看目前大概有哪些 OGC 标准吧：

OGC 标准	常用简称	说明
Cat: ebRIM App Profile: Earth Observation Products		
Catalogue Service	CS	用以发现、浏览服务器上数据、服务的元数据
CityGML		用以交换城市 3D 模型
Coordinate Transformation Service	CT	用以提供坐标系统及其转化的服务
Filter Encoding	FES	提供 XML 编码的过滤表达
GML in JPEG 2000		GML 和 JPEG 2000 编码图像的结合
Geographic Objects	GOS	通过 UML 和 Java 来描述抽象地理对象
Geography Markup Language	GML	提供 XML 编码的地理数据集
Geospatial eXtensible Access Control Markup Language	GeoXACML	
Grid Coverage Service		栅格服务
KML	KML	提供 XML 编码的地理数据集 (从 Google 引入)

Location Services	OpenLS	位置服务
Observations and Measurements		
Sensor Model Language		
Sensor Observation Service		
Sensor Planning Service		
Simple Features	SFS	简单要素对象的通用描述
Simple Features CORBA		
Simple Features OLE/COM		
Simple Features SQL		简单要素对象在 SQL 语句中的描述
Styled Layer Descriptor	SLD	用以对地理数据进行符号化
Symbology Encoding	SE	对符号进行编码
Transducer Markup Language	TML	
Web Coverage Processing Service	WCPS	栅格处理 Web 服务
Web Coverage Service	WCS	栅格 Web 服务
Web Feature Service	WFS	要素 Web 服务
Web Map Context		地图 Web 服务的组合
Web Map Service	WMS	地图 Web 服务
Web Map Tile Service	WMTS	切片地图 Web 服务
Web Processing Service	WPS	地理处理 Web 服务
Web Service Common	OWS	描述了 OGC Web 服务的通用规范

表 1 OGC 标准概览

其中，一些标准存在多个历史版本，比如 WMS 就有 1.3.0/1.1.1/1.1/1.0 等版本。在这些版本中，一般需要关注的就是最新版本，历史版本一般都被废弃，这从 OGC 网站上就可以看出区别：


Version	Document Title (click to download)	Document #	Type
1.3.0	Web Map Service	03-109r1	D-RP
1.3.0	OpenGIS Web Map Service (WMS) Implementation Specification	06-042	IS
	Web Map Services - Application Profile for EO Products (0.3.3)	07-063r1	BP
	Web Map Services - Application Profile for EO Products (0.2.0)	07-063	D-BP
	OpenGIS Web Map Services - Application Profile for EO Products (0.1.0)	06-093	D-DP
	DGIWG WMS 1.3 Profile and systems requirements for interoperability for use within a military environment (0.9.0)	09-102	BP
	OpenGIS Tiled WMS Discussion Paper (0.3.0)	07-057r2	D-DP
1.1.1	Web Map Service	01-068r3	D-IS
1.1	Web Map Service	01-047r2	D-IS
1.0	Web Map Service	00-028	D-IS
0.9	WMS - Proposed Animation Service Extension	06-045r1	DP
0.0.3	WMS Part 2: XML for Requests using HTTP Post	02-017r1	DP
0.1.0	WMS Change Request: Support for WSDL & SOAP	04-050r1	DP

这些标准中，一般我们接触的都集中在数据交换和服务互操作方面，比如

GML、KML 和 WFS、WMS 等，其实这也是一些标准存在意义较大的场合。下面，就逐个介绍一下其中的一些常用标准。

II. SFS-简单要素标准

- 概述

SFS (OpenGIS® Simple Features Interface Standard) 的当前版本是  1.2.0。事实上 SFS 中包括两部分内容 第一部分是描述简单要素的通用模型 ([Part 1: Common architecture](#))、另一部分是描述前一部分模型在 SQL 中的实现 ([Part 2: SQL option](#))¹。我们平时所熟知的 WKT、WKB 等就在第一部分中叙述，而在空间 SQL 语句中常见的 AsText、Intersects 操作等则在第二部分中有定义。

下面就 SFS 中比较重要的内容进行详细的说明，包括第一部分中的 几何对象模型、WKT 描述的几何对象、WKB 描述的几何对象、WKT 描述的空间参考 和第二部分中的 SQL 预定义 schema、SQL 几何类型、SQL 空间操作。

¹ 原本还有关于 CORBA 和 OLE/COM 的实现，当前版本已经不支持了。

• 几何对象模型

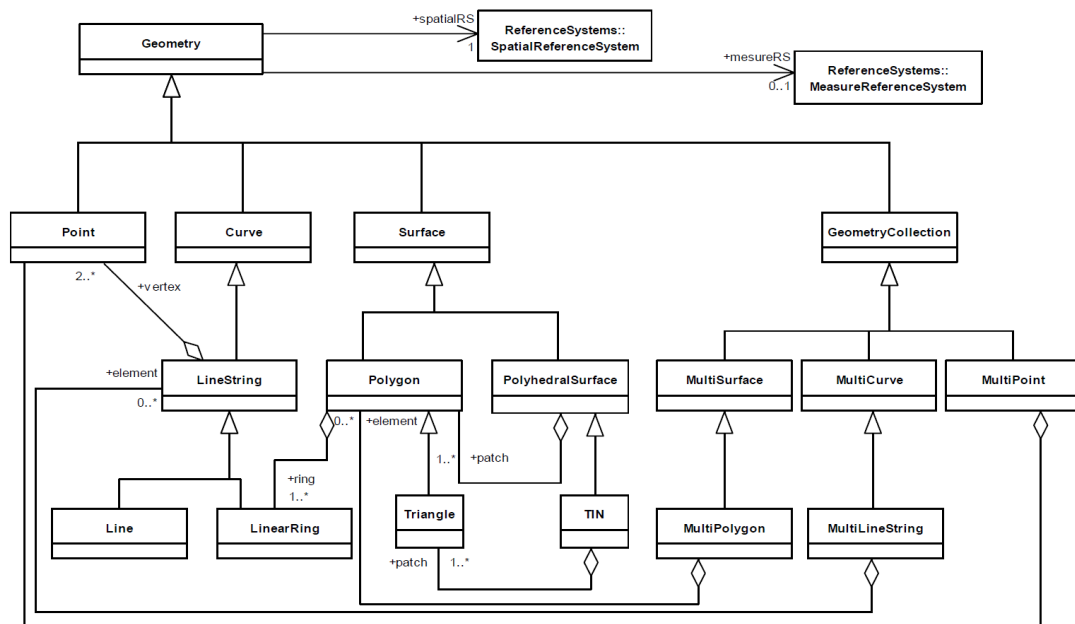


图 1 SFS 中定义的几何对象模型

图 1 显示的是 SFS 中几何对象的关系结构，简单要素中的几何对象主要就是定义了点、线、面和多点、多线、多面。另外，几何对象还涉及一系列的操作，如图 2 所示。

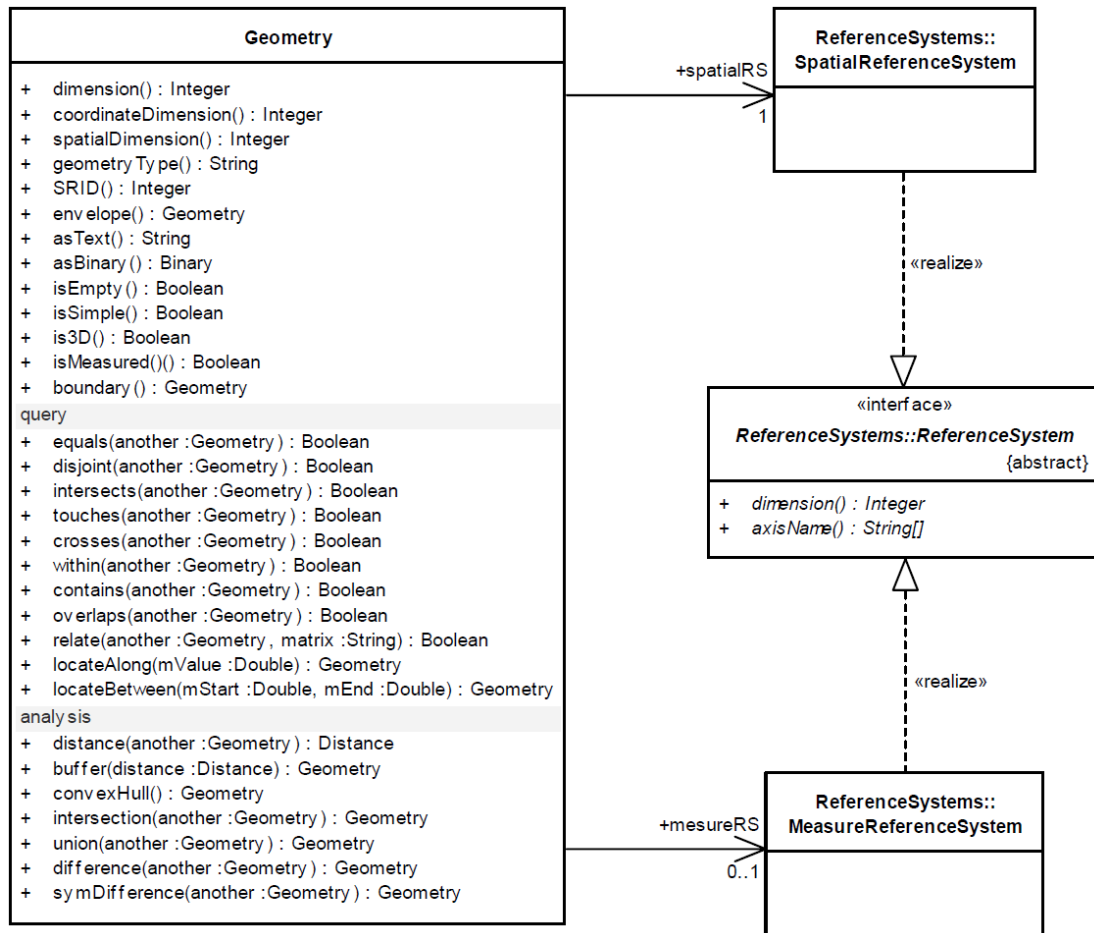


图 2 Geometry 的操作

● WKT 描述的几何对象

WKT (Well-known Text) 可以通过文本来描述几何对象。下面的例子可

以比较快速、直观地说明什么是 WKT :

几何类型	WKT 例子	说明
Point	Point (10 10)	点
LineString	LineString (10 10, 20 20, 30 40)	有 3 个节点的线
Polygon	Polygon ((10 10, 10 20, 20 20, 20 15, 10 10))	只有 1 个外环的多边形
MultiPoint	MultiPoint ((10 10), (20 20)	多点

)	
MultiLineString	MultiLineString ((10 10, 20 20), (15 15, 30 15))	多线
MultiPolygon	MultiPolygon (((10 10, 10 20, 20 20, 20 15, 10 10)), ((60 60, 70 70, 80 60, 60 60)))	多面
GeometryCollection	GeometryCollection (POINT (10 10), POINT (30 30), LINESTRING (15 15, 20 20))	几何集合
PolyhedralSurface	PolyhedralSurface Z (((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)), ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)), ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)))	多个表面构成的立方体
Tin	Tin Z (((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 0 0 1, 0 0 0)), ((1 0 0, 0 1 0, 0 0 1, 1 0 0)),)	4 个三角形构成的 TIN 网格
Point	Point Z (10 10 5)	三维点
Point	Point ZM (10 10 5 40)	带 M 值的三维点
Point	Point M (10 10 40)	带 M 值的二维点

表 2 WKT 描述几何对象示例

• WKB 描述的几何对象

WKB (Well-known Binary) 通过序列化的字节对象来描述几何对象。在 WKB 中主要涉及两种数值类型：一种是 uint32 ,占 4 个字节 ,用以存储节点数、几何对象类型等信息；另一种是 double ,占 8 个字节 ,用以存储节点坐标值。其中的几何对象类型对应的整数可以参考下表：

Type	Code	Type	Code	Type	Code	Type	Code
Geometry	0	Geometry Z	1000	Geometry M	2000	Geometry ZM	3000
Point	1	Point Z	1001	Point M	2001	Point ZM	3001
LineString	2	LineString Z	1002	LineString M	2002	LineString ZM	3002
Polygon	3	Polygon Z	1003	Polygon M	2003	Polygon ZM	3003
MultiPoint	4	MultiPoint Z	1004	MultiPoint M	2004	MultiPoint ZM	3004
MultiLineString	5	MultiLineString Z	1005	MultiLineString M	2005	MultiLineString ZM	3005
MultiPolygon	6	MultiPolygon Z	1006	MultiPolygon M	2006	MultiPolygon ZM	3006
GeometryCollection	7	GeometryCollection Z	1007	GeometryCollection M	2007	GeometryCollection ZM	3007
CircularString	8	CircularString Z	1008	CircularString M	2008	CircularString ZM	3008
CompoundCurve	9	CompoundCurve Z	1009	CompoundCurve M	2009	CompoundCurve ZM	3009
CurvePolygon	10	CurvePolygon Z	1010	CurvePolygon M	2010	CurvePolygon ZM	3010
MultiCurve	11	MultiCurve Z	1011	MultiCurve M	2011	MultiCurve ZM	3011
MultiSurface	12	MultiSurface Z	1012	MultiSurface M	2012	MultiSurface ZM	3012
Curve	13	Curve Z	1013	Curve M	2013	Curve ZM	3013
Surface	14	Surface Z	1014	Surface M	2014	Surface ZM	3014
PolyhedralSurface	15	PolyhedralSurface Z	1015	PolyhedralSurface M	2015	PolyhedralSurface ZM	3015
TIN	16	TIN Z	1016	TIN M	2016	TIN ZM	3016

图 3 WKB 中几何类型对应的整数值

除此之外，WKB 在第一位还存储了一个额外的字节用来标识字节序¹ (0=Big-Indian , 1=Little-Indian)。因此，对于一个点(不带 M 值的二维点)来说，它的 WKB 描述应该类似下面的结构，总共占据 21 个字节：



图 4 WKB 描述点的字节结构

对于有 2 个节点的线来说，WKB 描述应该包含 41 个字节：

¹ 关于字节序可以参考：<http://zh.wikipedia.org/zh/%E5%AD%97%E8%8A%82%E5%BA%8F>

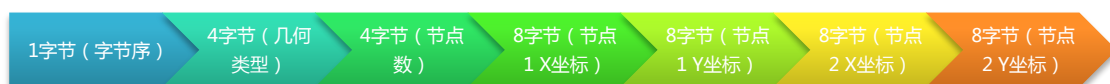


图 5 WKB 描述线的字节结构

对于仅有 1 个外环，由 3 个节点构成的多边形来说，WKB 描述则应该包含 77 个字节：



图 6 WKB 描述多边形的字节结构

• WKT 描述的空间参考

WKT 除了可以描述几何对象，也可以描述空间参考。通过 2 个例子可以很直观地看到如何通过文本来描述空间参考。

对于一个地理坐标系，比如最常见的 WGS84 坐标系统，WKT 描述是这样的：

```
GEOGCS
[
  "GCS_WGS_1984",
  DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],
  PRIMEM["Greenwich",0.0],
  UNIT["Degree",0.0174532925199433],
  AUTHORITY["EPSG",4326]
]
```

“GEOGCS”表明其后紧随的 “[]” 中描述的是一个地理坐标系统。该坐标系统名称为 “GCS_WGS_1984”；采用的大地基准面为 “D_WGS_1984”，该基准面近似椭球体的长轴为 6378137.0 米、扁率为 298.257223563；以格林威治

0 度经线为起始经线；地图单位为度，该单位的转换因子为 0.

0174532925199433($\pi/180$) ;最后 ,该坐标系统在 EPSG²中的编码为“4326”。

对于一个投影坐标系，比如 WGS84 Web Mercator (Auxiliary Sphere)
坐标系统，WKT 描述是这样的：

```
PROJCS
[
  "WGS_1984_Web_Mercator_Auxiliary_Sphere",
  GEOGCS
  [
    "GCS_WGS_1984",
    DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],
    PRIMEM["Greenwich",0.0],
    UNIT["Degree",0.0174532925199433]
  ],
  PROJECTION["Mercator_Auxiliary_Sphere"],
  PARAMETER["False_Easting",0.0],
  PARAMETER["False_Northing",0.0],
  PARAMETER["Central_Meridian",0.0],
  PARAMETER["Standard_Parallel_1",0.0],
  PARAMETER["Auxiliary_Sphere_Type",0.0],
  UNIT["Meter",1.0],
  AUTHORITY["EPSG",3857]
]
```

类似的，“PROJCS”代表这是一个投影坐标系。投影坐标系中必然会包括一个地理坐标系，这里的地理坐标系就是“GCS_WGS_1984”，这个地理坐标系的定义和上面的类似。下面紧跟着的是投影的相关参数，

“Mercator_Auxiliary_Sphere”是采用投影的名称 这个投影坐标系以 0 度经线为中央经线进行投影；坐标系的单位为米（显然，转换因子就为 1.0），而该坐标系的 EPSG 编码为“3857”。

¹ 转换因子意为一个单位所代表的米（线性单位）或所代表的弧度数（角度单位）。

² 在线 EPSG 编码参考：<http://spatialreference.org/ref/epsg/>

- SQL 预定义 schema

在空间数据库中，需要一些表来存储和管理几何字段、空间参考等信息，因此 OGC 首先规定了在数据库中需要的 Schema 对象¹：

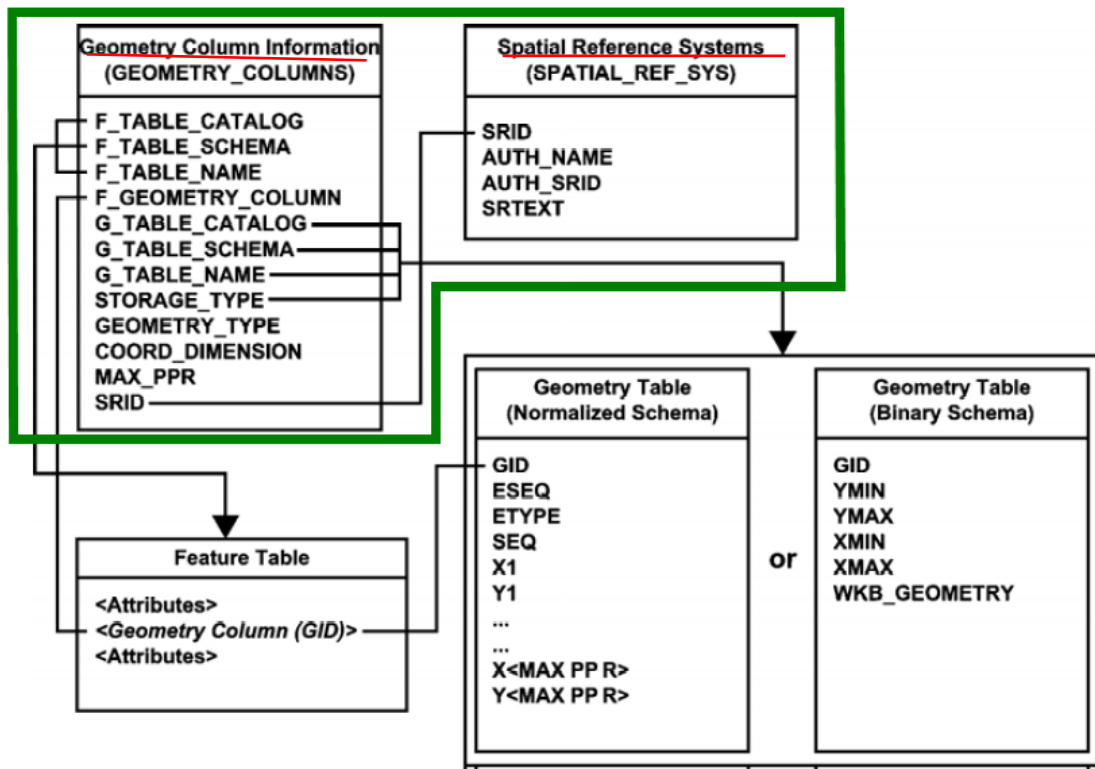


图 7 OGC 空间数据库中需要预定义的 schema

可见 数据库中必须要有一个记录几何字段信息的 GEOMETRY_COLUMNS 表和一个记录空间参考信息的 SPATIAL_REF_SYS 表。从支持 OGC 标准的空间数据库中，我们都可以找到这两张表。当然，有些空间数据库直接使用 “GEOMETRY_COLUMNS” 和 “SPATIAL_REF_SYS” 作为这两张表的名称，比如 PostGIS，有些则采用其它名字，比如 Oracle Spatial 采用

¹ 兼容 ISO 《SQL Multimedia and Application Packages(SQL/MM)—Part 3: Spatial》中的空间信息 Schema 短名字视图定义。

“OGIS_GEOMETRY_COLUMNS” 和

“OGIS_SPATIAL_REFERENCE_SYSTEMS”、ArcSDE for Oracle 则采用

“GEOMETRY_COLUMNS” 和 “SPATIAL_REFERENCES”，不一而同。

• SQL 几何对象存储

图 7 中显示在 OGC 标准中几何信息存储在一个 Geometry 表中，这个表可以用常规字段或 WKB 两种方式存储几何对象，Geometry 表通过 GID 字段关联到 Feature 表的几何字段。事实上，OGC 标准中还有一种定义，Feature 表的几何字段也可以是 SQL UDT（自定义类型），也就是不需要额外的 Geometry 表来存储几何信息，而直接存储在 Feature 表的几何字段中。大多数数据库都是采用这种自定义类型的方式存储几何信息，比如 ArcSDE 中的 ST_Geometry 类型、PostGIS 中的 Geometry 和 ST_Geometry 类型。

自定义类型可以采用 SFS 标准中定义的几何类型，也可以采用 SQL/MM¹的定义，比如 PostGIS 对这两种定义都进行了支持，下图是 SFS 和 SQL/MM 几何类型定义的一个对应关系：

¹ ISO 《SQL Multimedia and Application Packages》标准，其中的《Part3:Spatial》定义了和空间数据有关的内容。

	SQL with geometry type	ISO/IEC 13249-3:2003 (SQL/MM-Spatial)
Geometry Types	Point Curve Linestring Surface Polygon PolyhedralSurface GeomCollection Multipoint Multicurve Multilinestring Multisurface Multipolygon	ST_Point ST_Curve ST_Linestring ST_Circularstring ST_CompoundCurve ST_Surface ST_CurvePolygon ST_Polygon ST_PolyhedralSurface ST_Collection ST_Multipoint ST_MultiCurve ST_Multilinestring ST_Multisurface ST_Multipolygon

图 8 SFS 和 SQL/MM 几何类型的对应关系

用户既可以遵循 SFS 的定义，使用类似 “Geometry”、“Point” 这样的命名；也可以遵循 SQL/MM 定义，采用 “ST_” 作为前缀进行命名，如 “ST_Geometry”、“ST_Point”。

SQL 几何类型的继承关系可参考图 9，可以发现 SQL 中实现的几何对象模型和图 1 所示的通用几何对象模型非常类似。

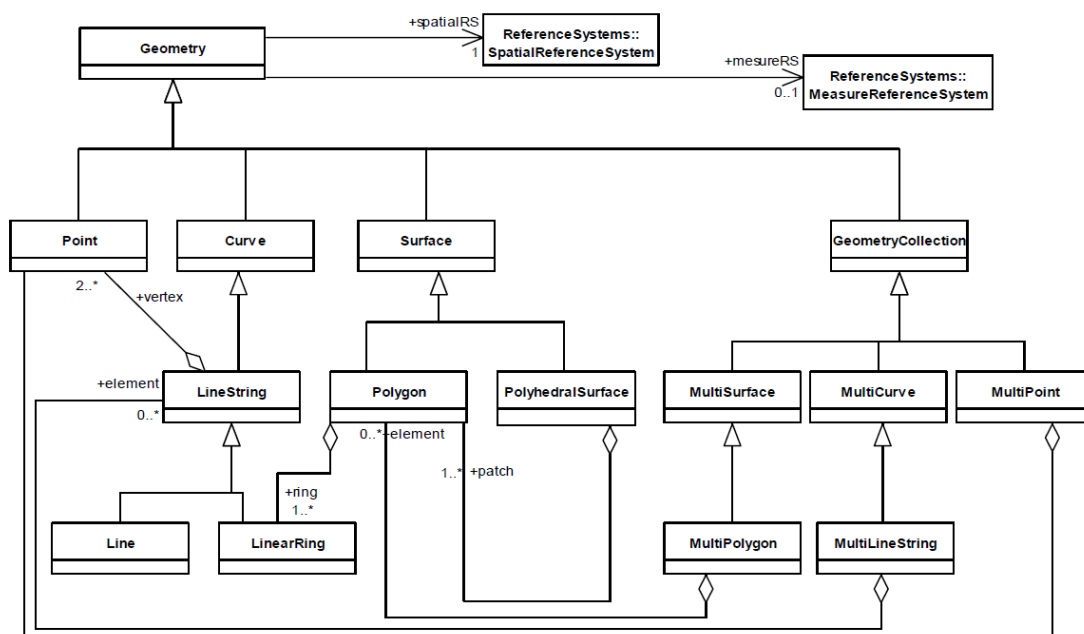


图 9 SQL 几何对象模型

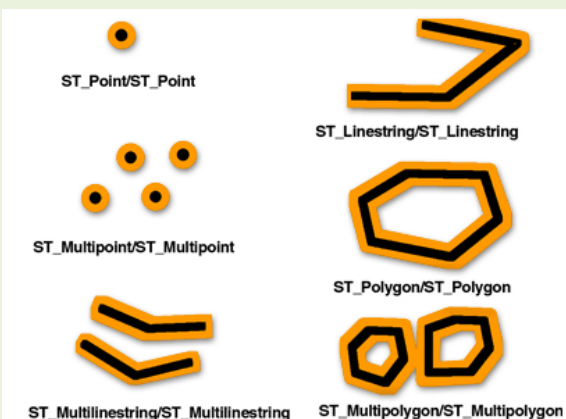
• SQL 空间操作

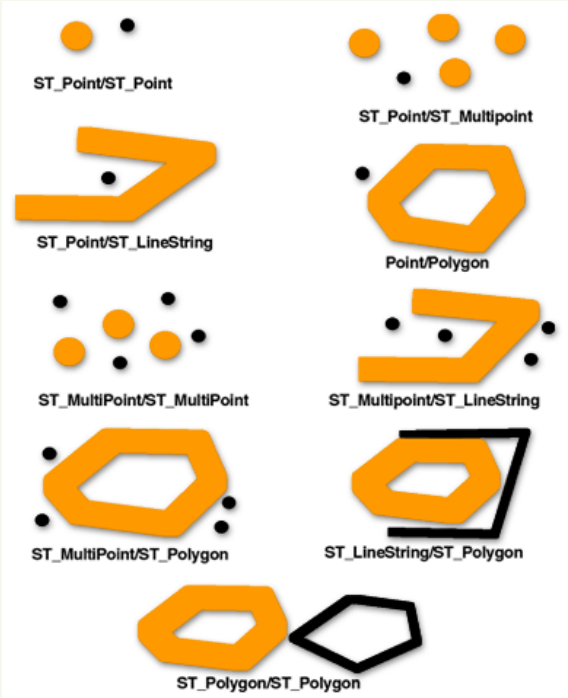
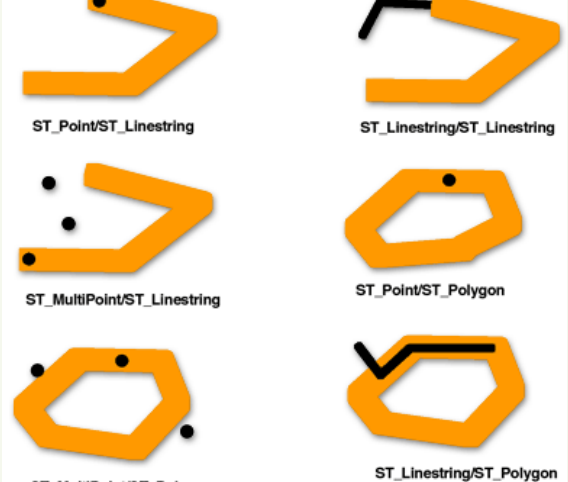
以 SQL/MM 定义为例，在 SFS 中规定了以下的操作。

1. 所有几何对象支持

几何对象构造	说明
ST_WKTTToSQL	从 WKT 构造几何对象
ST_WKBToSQL	从 WKB 构造几何对象

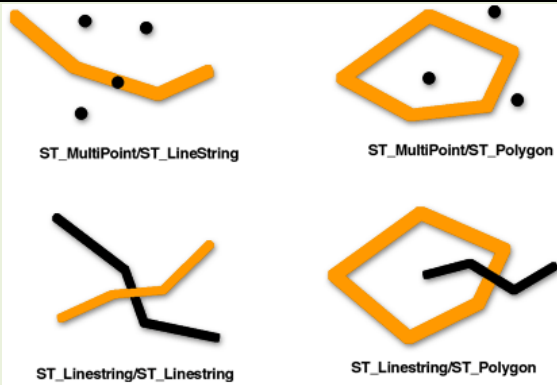
几何信息获取	说明
ST_AsText	获取 WKT 描述
ST_AsBinary	获取 WKB 描述
ST_Dimension	获取维数
ST_GeometryType	获取几何类型
ST_SRID	获取空间参考 ID
ST_IsEmpty	是否为空
ST_IsSimple	是否是简单对象
ST_Boundary	获取边界
ST_Envelope	获取矩形范围

空间关系判断	说明
ST_Equals	

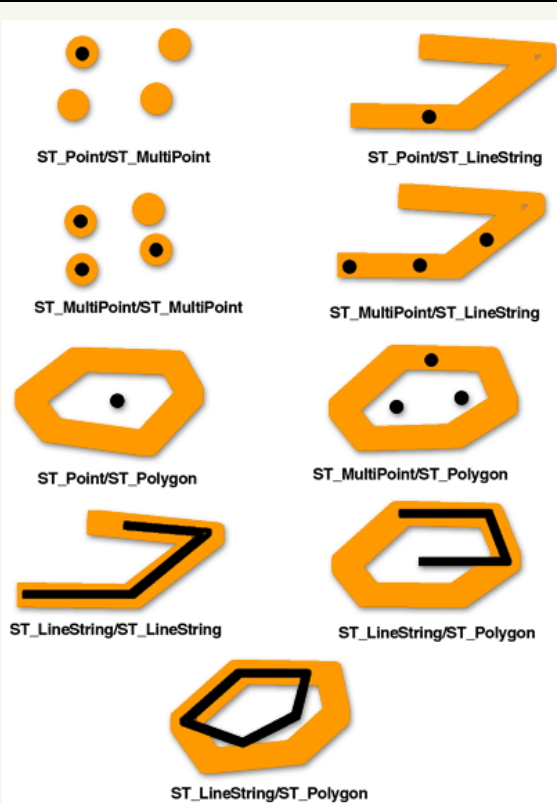
ST_Disjoint	
ST_Intersects	<p>任意部分有相交，等价于判断空间关系的 DE-9IM¹字符串表达是否是以下之一：</p> <p>T*****</p> <p>*T*****</p> <p>***T*****</p> <p>****T****</p>
ST_Touches	

¹ <http://docs.codehaus.org/display/GEOTDOC/Point+Set+Theory+and+the+DE-9IM+Matrix>

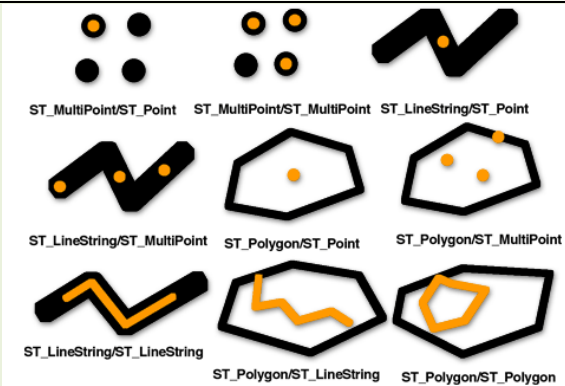
ST_Crosses



ST_Within



ST_Contains

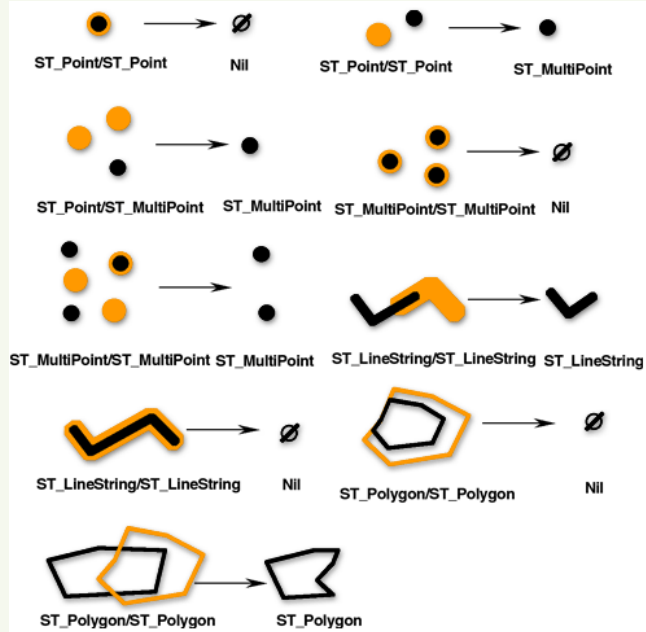


ST_Overlaps	
ST_Relate	判断是否满足 DE-9IM 字符串表达关系

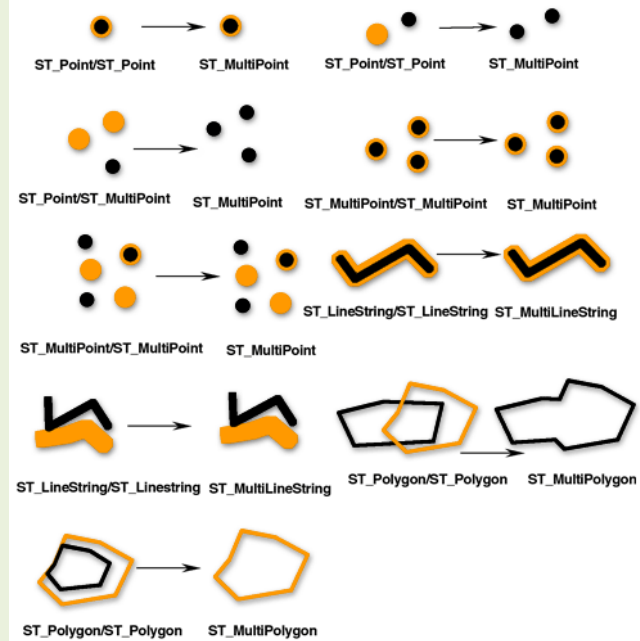
距离计算	说明
ST_Distance	几何对象间的最短距离

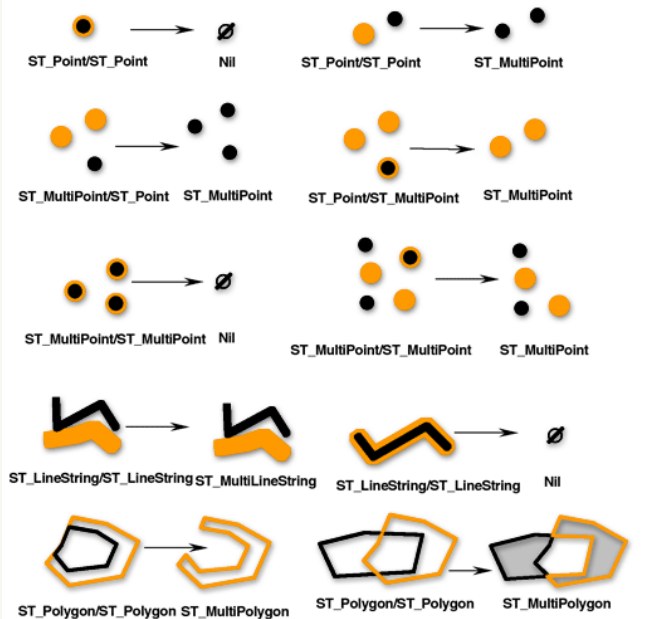
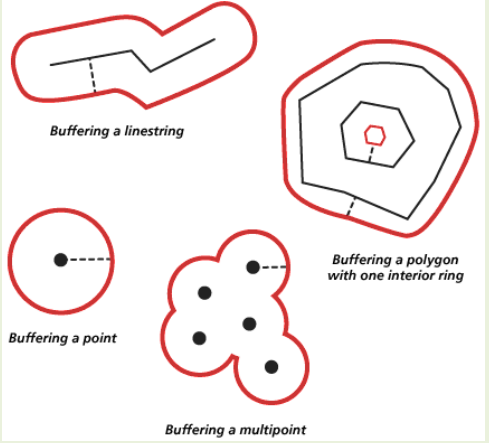
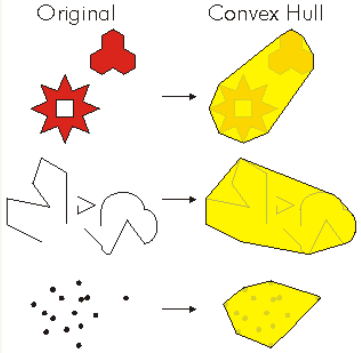
几何运算	说明
ST_Intersection	

ST_Difference



ST_Union



ST_SymDifference	 <p>ST_Point/ST_Point Nil</p> <p>ST_Point/ST_Point ST_MultiPoint</p> <p>ST_MultiPoint/ST_Point ST_MultiPoint</p> <p>ST_Point/ST_MultiPoint ST_MultiPoint</p> <p>ST_MultiPoint/ST_MultiPoint Nil</p> <p>ST_MultiPoint/ST_MultiPoint ST_MultiPoint</p> <p>ST_LineString/ST_LineString ST_MultiLineString</p> <p>ST_LineString/ST_LineString Nil</p> <p>ST_Polygon/ST_Polygon ST_MultiPolygon</p> <p>ST_Polygon/ST_Polygon ST_MultiPolygon</p>
ST_Buffer	 <p>Buffering a linestring</p> <p>Buffering a point</p> <p>Buffering a polygon with one interior ring</p> <p>Buffering a multipoint</p>
ST_ConvexHull	 <p>Original Convex Hull</p>

2.Point 对象支持

SQL 操作	说明
--------	----

ST_X	获取 X 值
ST_Y	获取 Y 值
ST_Z	获取 Z 值
ST_M	获取 M 值

3. Curve 对象支持

SQL 操作	说明
ST_StartPoint	获取起始点
ST_EndPoint	获取终点
ST_IsRing	是否是环
ST_Length	获取长度

4. LineString 对象支持

SQL 操作	说明
ST_NumPoints	节点数
ST_PointN	获取第 n 个节点

5. Surface 对象支持

SQL 操作	说明
ST_Centroid	获取中心点
ST_PointOnSurface	获取面上一点
ST_Area	获取面积

6. Polygon 对象支持

SQL 操作	说明
ST_ExteriorRing	获取外环
ST_NumInteriorRing	获取内环数

ST_InteriorRingN	获取第 n 个内环
------------------	-----------

7. GeomCollection 对象支持

SQL 操作	说明
ST_NumGeometries	获取几何对象数
ST_GeometryN	获取第 n 个几何对象

8. MultiCurve 对象支持

SQL 操作	说明
ST_IsClosed	是否闭合
ST_Length	获取长度

9. MultiSurface 对象支持

SQL 操作	说明
ST_Centroid	获取中心点
ST_PointOnSurface	获取面上一点
ST_Area	获取面积

• ArcGIS 对 SFS 的支持

ArcGIS 10 之前的全系列产品都支持 SFS 标准，版本为 1.1。下面通过几个 ArcSDE (Oracle) 的空间 SQL 操作看一下：

```
select st_astext(st_geometry('POINT(116 39)',0)) from dual;
```

```
select shape from TEST where  
st_envintersects(shape,40251885,4019516,40255159,4021607)=1;
```

```
select shape from TEST where st_intersects(shape, st_geometry('POLYGON((0  
0,180 -90,180 90,0 0))',2))=1;
```

```
select st_relate (g1, g2, 'T**F**FFF*') equals, st_relate (g1, g3, 'T**F**FFF*')  
not_equals from RELATE_TEST;
```

```
select sum (st_area (st_difference (lot, footprint))) from FOOTPRINTS bf, LOTS  
where bf.building_id = lots.lot_id;
```

III. GML-地理标记语言

- 概述

GML (OpenGIS® Geography Markup Language Encoding Standard) 当前版本是 3.2.1。它是一种基于 XML 的地理要素描述语言标准，用以在不同的软件或系统间交换空间数据，比如后面会介绍的 WFS 标准就使用 GML 作为输入和输出格式。GML 同时也是 ISO 标准¹。

- GML Schema

GML 标准其实就是通过 XML Schema(XSD²)来定义了 GML 文档的结构，这些定义都可以访问在线的地址：<http://schemas.opengis.net/gml/> 得到。目前 3.2.1 版本的 GML 中包含 7 个顶级 XSD (其中一个是废弃类型，为了向前兼容)，这些 XSD 下还有其它子 XSD，它们组合成为如图 10 的结构。

¹ http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32554

² XML Schema Definition

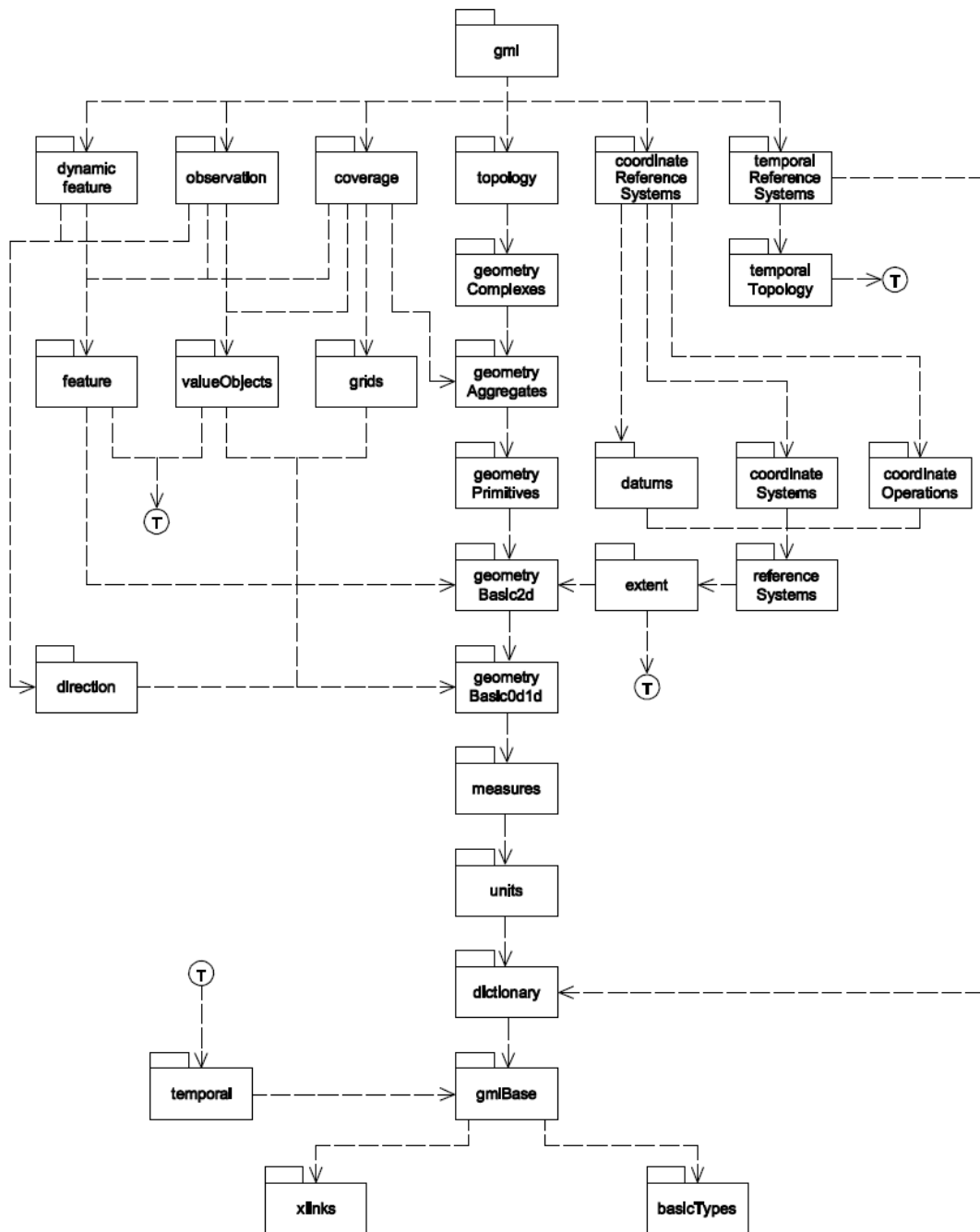


图 10 GML XSD 依赖关系

• GML 示例

下面是一个包含点、线、面数据的 GML 的简单的例子，通过这个例子可以

看到 GML 的组织情况：

```
<?xml version="1.0" encoding="UTF-8"?>
<gml:FeatureCollection xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:fme="http://www.safe.com/gml/fme"
  xsi:schemaLocation="http://www.safe.com/gml/fme MyGML.xsd">
  <gml:boundedBy>
    <gml:Envelope srsName="EPSG:4326" srsDimension="2">
      <gml:lowerCorner>-0.012611143330844 -0.050087654172727</gml:lowerCorner>
      <gml:upperCorner>0.051158411625351 0.019154661096934</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <gml:featureMember>
    <fme:point gml:id="id9e2fb700-bbdb-48a5-8d99-f68b89886f4a">
      <fme:Id>0</fme:Id>
      <gml:pointProperty>
        <gml:Point srsName="EPSG:4326" srsDimension="2">
          <gml:pos>0.027363801567048 -0.028672505120255</gml:pos>
        </gml:Point>
      </gml:pointProperty>
    </fme:point>
  </gml:featureMember>
  <gml:featureMember>
    <fme:line gml:id="id4e0efc7b-d652-4d3c-8466-1a2419bca6d2">
      <fme:Id>0</fme:Id>
      <gml:curveProperty>
        <gml:LineString srsName="EPSG:4326" srsDimension="2">
          <gml:posList>0.051158411625351 0.019154661096934 0.039736998797366
-0.050087654172727</gml:posList>
        </gml:LineString>
      </gml:curveProperty>
    </fme:line>
  </gml:featureMember>
  <gml:featureMember>
    <fme:polygon gml:id="idf9b78df6-c402-4bc9-afd4-22762401c565">
      <fme:Name>0</fme:Name>
      <gml:surfaceProperty>
        <gml:Surface srsName="EPSG:4326" srsDimension="2">
          <gml:patches>
            <gml:PolygonPatch>
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList>0.016656227040926 -0.024151529209121
0.003569191508859 0.005829679464227 -0.012611143330844 -0.01177833197886 -0.004283029810438
-0.042473378954071 0.018321849744836 -0.049373815870979 0.016656227040926
-0.024151529209121</gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:PolygonPatch>
          </gml:patches>
        </gml:Surface>
      </gml:surfaceProperty>
    </fme:polygon>
  </gml:featureMember>
</gml:FeatureCollection>
```

```

        </gml:Surface>
      </gml:surfaceProperty>
    </fme:polygon>
  </gml:featureMember>
</gml:FeatureCollection>

```

在这个例子中描述了一个 GML 的 FeatureCollection ,由于这是由 FME 导出的数据 ,因此其中还包含 “fme” 的命名空间和 “fme:point”、“fme:line”、“fme:polygon” 类型 , 这些类型在这个 XML 包含的 XSD 中进行了描述 , 实际对应的类型为基于 GML 标准的点、线、面 :

```

<?xml version= "1.0" encoding= "UTF-8"?>
<schema xmlns= "http://www.w3.org/2001/XMLSchema"
  xmlns:gml= "http://www.opengis.net/gml"
  xmlns:xlink= "http://www.w3.org/1999/xlink"
  xmlns:fme= "http://www.safe.com/gml/fme"
  targetNamespace= "http://www.safe.com/gml/fme"
  elementFormDefault= "qualified">
  <import namespace= "http://www.opengis.net/gml"
    schemaLocation= "http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <element name= "point" type= "fme:pointType" substitutionGroup= "gml:_Feature"/>
  <complexType name= "pointType">
    <complexContent>
      <extension base= "gml:AbstractFeatureType">
        <sequence>
          <element name= "Id" minOccurs= "0" type= "integer"/>
          <element ref= "gml:pointProperty" minOccurs= "0"/>
          <element ref= "gml:multiPointProperty" minOccurs= "0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <element name= "line" type= "fme:lineType" substitutionGroup= "gml:_Feature"/>
  <complexType name= "lineType">
    <complexContent>
      <extension base= "gml:AbstractFeatureType">
        <sequence>
          <element name= "Id" minOccurs= "0" type= "integer"/>
          <element ref= "gml:curveProperty" minOccurs= "0"/>
          <element ref= "gml:multiCurveProperty" minOccurs= "0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <element name= "polygon" type= "fme:polygonType" substitutionGroup= "gml:_Feature"/>
  <complexType name= "polygonType">
    <complexContent>
      <extension base= "gml:AbstractFeatureType">

```

```

<sequence>
  <element name= "Name" minOccurs= "0">
    <simpleType>
      <restriction base= "string">
        <maxLength value= "30"/>
      </restriction>
    </simpleType>
  </element>
  <element ref= "gml:surfaceProperty" minOccurs= "0"/>
  <element ref= "gml:multiSurfaceProperty" minOccurs= "0"/>
</sequence>
</extension>
</complexContent>
</complexType>
</schema>

```

而描述数据范围的 “gml:boundedBy” 属性则直接由 “gml:Envelope” 类型的对象来描述；另外，数据的坐标节点、空间参考等属性则直接由 GML 的类型进行描述。

更简单一点，如果我们需要自己写一个 GML 来描述一个地理数据，写出来的文件内容可能是这样的：

```

<City> <!--描述某城市的数据-->
  <gml:boundedBy>
    <gml:Envelope>
      <gml:pos>100 200</gml:pos>
      <gml:pos>230 250</gml:pos>
    </gml:Envelope>
  </gml:boundedBy>
  <gml:featureMember>
    <Bridge gml:id= "bridge 1"> <!--某桥对象-->
      <span>100</span> <!--属性信息：该桥的跨度-->
      <height>200</height> <!--属性信息：该桥的高度-->
      <gml:curveProperty> <!--几何信息-->
        <gml:LineString gml:id= "line 24">
          <gml:pos>100 200</gml:pos>
          <gml:pos>200 200</gml:pos>
        </gml:LineString>
      </gml:curveProperty>
    </Bridge>
  </gml:featureMember>
</City>

```

IV. SLD-图层样式描述

• 概述

SLD (OpenGIS® Styled Layer Descriptor) 当前版本是 1.1.0。SLD 是一种描述地图图层样式的标准，一般用于 WMS。一个地图不仅包含数据源组成，还需要对数据进行符号化和渲染，SLD 就是这个定义地图图层符号化和渲染信息的标准。

• SLD Schema

和 GML 类似，SLD 也通过一些 XML Schema 来定义 SLD 文档的结构，这些定义可以通过访问 <http://schemas.opengis.net/sld/> 得到。SLD 的 Schema 还依赖其它的 OGC 标准的 Schema，其中包括：GML¹、Filter Encoding²、Symbology Encoding³。

• SLD 简单例子

下面是一个 SLD 的简单例子，这里对一个多边形图层进行了填充样式的定义：

```
<?xml version="1.0" encoding="UTF-8"?>
```

¹ <http://schemas.opengis.net/gml/>

² <http://schemas.opengis.net/filter/>

³ <http://schemas.opengis.net/se>

```

<StyledLayerDescriptor version= "1.1.0"
  xsi:schemaLocation= "http://www.opengis.net/sld StyledLayerDescriptor.xsd"
  xmlns= "http://www.opengis.net/sld"
  xmlns:ogc= "http://www.opengis.net/ogc"
  xmlns:se= "http://www.opengis.net/se"
  xmlns:xlink= "http://www.w3.org/1999/xlink"
  xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance">
  <NamedLayer>
    <se:Name>OCEANSEA_1M:Foundation</se:Name>
    <UserStyle>
      <se:Name>GEOSYM</se:Name>
      <IsDefault>1</IsDefault>
      <se:FeatureTypeStyle>
        <se:FeatureTypeName>Foundation</se:FeatureTypeName>
        <se:Rule>
          <se:Name>main</se:Name>
          <se:PolygonSymbolizer
            uom= "http://www.opengeospatial.org/sld/units/pixel">
            <se:Name>MySymbol</se:Name>
            <se:Description>
              <se:Title>Example Symbol</se:Title>
              <se:Abstract>This is just a simple example.</se:Abstract>
            </se:Description>
            <se:Geometry>
              <ogc:PropertyName>GEOMETRY</ogc:PropertyName>
            </se:Geometry>
            <se:Fill>
              <se:SvgParameter name= "fill">#96C3F5</se:SvgParameter>
            </se:Fill>
          </se:PolygonSymbolizer>
        </se:Rule>
      </se:FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>

```

其中关键的还是在于 Symbology Encoding (SE) 标准定义的符号化和渲染的方式，由于 SE 和 SLD 的依赖关系，因此，在这里也顺便通过几个简单的例子对 SE 进行一下介绍。

• SE 示例

在 SE 标准中给出了几个符号化定义的例子，非常的浅显易懂，这几个例子分别定义了点、线、面、文本等内容应该使用什么样的符号化方式显示出来，而

在其中出现的 Mark、Stroke、Fill 等元素也很容易理解，下面就从这几个例子大致了解一下 SE 标准的定义。

点数据符号化：

```
<PointSymbolizer version="1.1.0"
  xsi:schemaLocation="http://www.opengis.net/se Symbolizer.xsd" xmlns="http://www.opengis.net/se"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  uom="http://www.opengeospatial.org/se/units/metre">
  <Name>MyPointSymbolizer</Name>
  <Description>
    <Title>Example Pointsymbolizer</Title>
    <Abstract>This is just a simple example of a point symbolizer.
    </Abstract>
  </Description>
  <Graphic>
    <Mark>
      <WellKnownName>star</WellKnownName>
      <Fill>
        <SvgParameter name="fill">#ff0000</SvgParameter>
      </Fill>
    </Mark>
    <Size>8.0</Size>
  </Graphic>
</PointSymbolizer>
```

线数据符号化：

```
<LineSymbolizer version="1.1.0"
  xsi:schemaLocation="http://www.opengis.net/se Symbolizer.xsd" xmlns="http://www.opengis.net/se"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  uom="http://www.opengeospatial.org/se/units/metre">
  <Name>MyLineSymbolizer</Name>
  <Description>
    <Title>Example Symbol</Title>
    <Abstract>This is just a simple example of a line symbolizer.
    </Abstract>
  </Description>
  <Stroke>
    <SvgParameter name="stroke">#0000ff</SvgParameter>
    <SvgParameter name="stroke-width">2</SvgParameter>
  </Stroke>
</LineSymbolizer>
```

面数据符号化：

```
<PolygonSymbolizer version="1.1.0"
  xsi:schemaLocation="http://www.opengis.net/se Symbolizer.xsd" xmlns="http://www.opengis.net/se"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:xlink="http://www.w3.org/1999/xlink"
```

```

xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
uom= "http://www.opengeospatial.org/se/units/pixel">
<Name>MyPolygonSymbolizer</Name>
<Description>
  <Title>Example PolygonSymbolizer</Title>
  <Abstract>This is just a simple example of a polygon symbolizer.
  </Abstract>
</Description>
<Fill>
  <SvgParameter name= "fill">#aaaaff</SvgParameter>
</Fill>
<Stroke>
  <SvgParameter name= "stroke">#0000aa</SvgParameter>
</Stroke>
</PolygonSymbolizer>

```

文本标注：

```

<TextSymbolizer version= "1.1.0"
  xsi:schemaLocation= "http://www.opengis.net/se Symbolizer.xsd" xmlns= "http://www.opengis.net/se"
  xmlns:ogc= "http://www.opengis.net/ogc" xmlns:xlink= "http://www.w3.org/1999/xlink"
  xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
  uom= "http://www.opengeospatial.org/se/units/pixel">
  <Name>MyTextSymbolizer</Name>
  <Description>
    <Title>Example TextSymbolizer</Title>
    <Abstract>This is just an example of a text symbolizer using the
      FormatNumber function.</Abstract>
  </Description>
  <Geometry>
    <ogc:PropertyName>locatedAt</ogc:PropertyName>
  </Geometry>
  <Label>
    <ogc:PropertyName>hospitalName</ogc:PropertyName>
  </Label>
  <Font>
    <SvgParameter name= "font-family">Arial</SvgParameter>
    <SvgParameter name= "font-family">Sans-Serif</SvgParameter>
    <SvgParameter name= "font-style">italic</SvgParameter>
    <SvgParameter name= "font-size">10</SvgParameter>
  </Font>
  <Halo />
  <Fill>
    <SvgParameter name= "fill">#000000</SvgParameter>
  </Fill>
</TextSymbolizer>

```

V. KML-我从 Google 来

- 概述

KML(OpenGIS® KML Encoding Standard)从 2.2.0 版本开始由 Google 提交到 OGC 并被接受为标准，当前 OGC KML 的版本也就是 2.2.0。

KML 和 GML 在名称上类似，但是功能有很大不同。GML 主要用于地理数据的交换；而 KML 主要用于地理数据的可视化，它不仅包括地理数据的描述，还包括数据的符号化方式、用户视角的控制等信息。

- KML Schema

KML 的 Schema 可以从 <http://schemas.opengis.net/kml/> 访问到。所有 KML 中耳熟能详的 Placemark、LookAt 等名称都可以在这里找到定义，相对 OGC 给出的文档，更好的关于 KML Schema 的参考在 Google Code 上 http://code.google.com/intl/zh-CN/apis/kml/documentation/kml_tut.html，这里还有关于 Schema 对象的关系，如图 11。

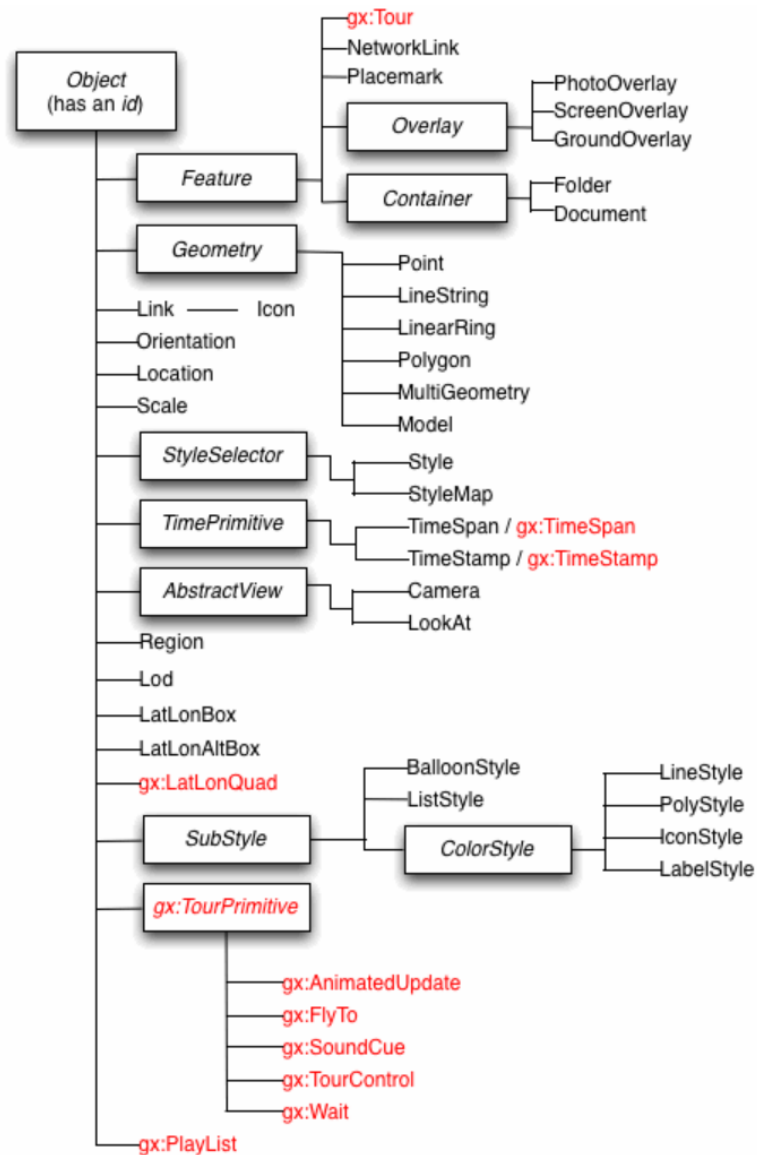


图 11 KML Schema 对象关系图

注意，在图 11 中的方框中的对象是逻辑上的对象，并不真实存在于 Schema 定义中。还有，红色标注的对象是 Google 对 KML 2.2.0 的扩展，在 Google Earth 5.0 以上版本中被支持，因此，对于需要了解 KML 标准的人来说，这些红色的内容都可以被忽略。

• KML 示例

关于地理数据的描述，下面是一个最简单的描述“地标”的 KML，它包含了一个点要素信息：

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Simple placemark</name>
    <description>Attached to the ground. Intelligently places itself
      at the height of the underlying terrain.</description>
    <Point>
      <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

下面是另外一个定义了显示样式的多边形数据：

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <Style id="transBluePoly">
      <LineStyle>
        <width>1.5</width>
      </LineStyle>
      <PolyStyle>
        <color>7dff0000</color>
      </PolyStyle>
    </Style>
    <Placemark>
      <name>Building 41</name>
      <styleUrl>#transBluePoly</styleUrl>
      <Polygon>
        <extrude>1</extrude>
        <altitudeMode>relativeToGround</altitudeMode>
        <outerBoundaryIs>
          <LinearRing>
            <coordinates>-122.0857412771483,37.42227033155257,17
              -122.0858169768481,37.42231408832346,17
              -122.085852582875,37.42230337469744,17
              -122.0858799945639,37.42225686138789,17
              -122.0858860101409,37.4222311076138,17
              -122.0858069157288,37.42220250173855,17
              -122.0858379542653,37.42214027058678,17
              -122.0856732640519,37.42208690214408,17
              -122.0856022926407,37.42214885429042,17
              -122.0855902778436,37.422128290487,17
              -122.0855841672237,37.42208171967246,17
```

```

-122.0854852065741,37.42210455874995,17
-122.0855067264352,37.42214267949824,17
-122.0854430712915,37.42212783846172,17
-122.0850990714904,37.42251282407603,17
-122.0856769818632,37.42281815323651,17
-122.0860162273783,37.42244918858722,17
-122.0857260327004,37.42229239604253,17
-122.0857412771483,37.42227033155257,17
</coordinates>
</LinearRing>
</outerBoundaryIs>
</Polygon>
</Placemark>
</Document>
</kml>

```

这是一个定义了视点属性的点要素：

```

<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Machu Picchu, Peru</name>
    <LookAt>
      <longitude>-72.503364</longitude>
      <latitude>-13.209676</latitude>
      <altitude>0</altitude>
      <range>14794.882995</range>
      <tilt>66.768762</tilt>
      <heading>71.131493</heading>
    </LookAt>
    <Point>
      <coordinates>-72.516244,-13.162806,0</coordinates>
    </Point>
  </Placemark>
</kml>

```

• ArcGIS 对 KML 的支持

ArcGIS 10 中支持 KML 2.2.0 版本。

1.ArcToolbox 输出 KML

我们可以通过 ArcToolbox 中的“To Kml” 工具箱将地图或图层输出为 KML

格式。比如使用“Map To Kml”工具，如图 12 所示：

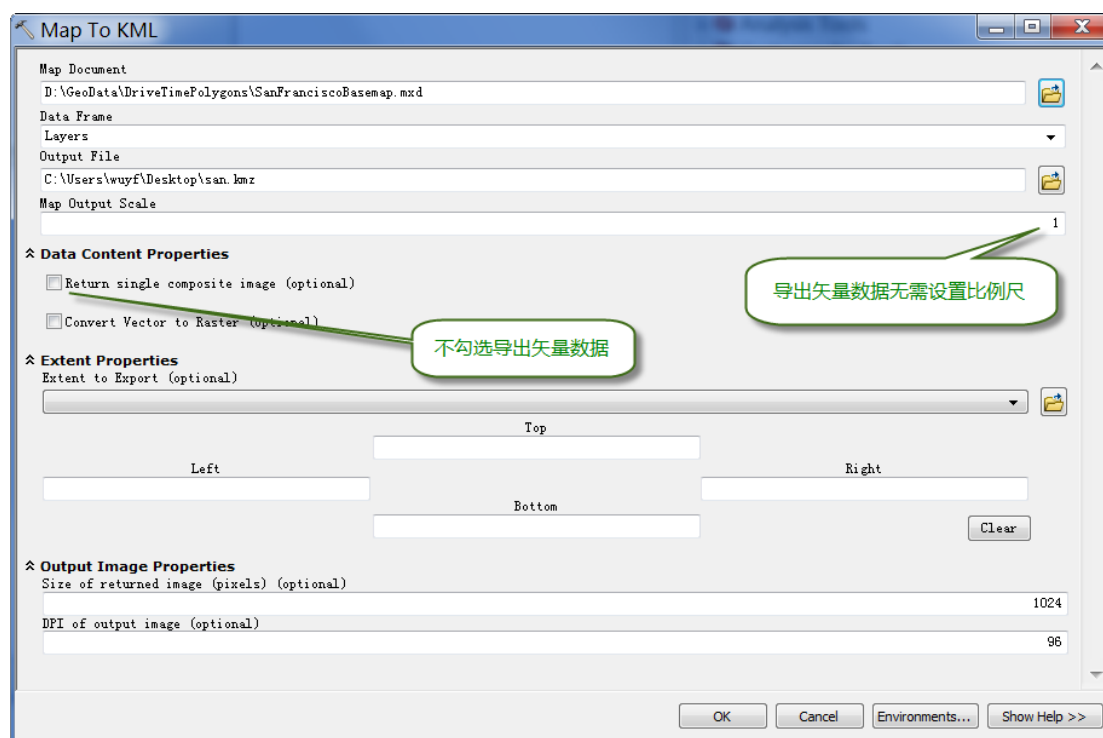


图 12 使用 ArcToolbox 导出 KML

这样导出的 KML 为 KMZ 文件，如果在 Google Earth 中直接打开导出的 KMZ 文件，可以看到在 Google Earth 中显示的地图和 MXD 中的基本一致。“基本一致”表示还稍有不同，比如在图 13 中蓝色的多边形对象在导出 KML 并加载到 Google Earth¹后会被下层的多边形覆盖而导致不可见。不过，这个是 Google Earth 的问题，因为从图层属性中可以看到，这些蓝色多边形所在的“Lakes”图层数据都存在，可惜没被正确显示，如图 14 的对比效果。

¹ 版本：5.1

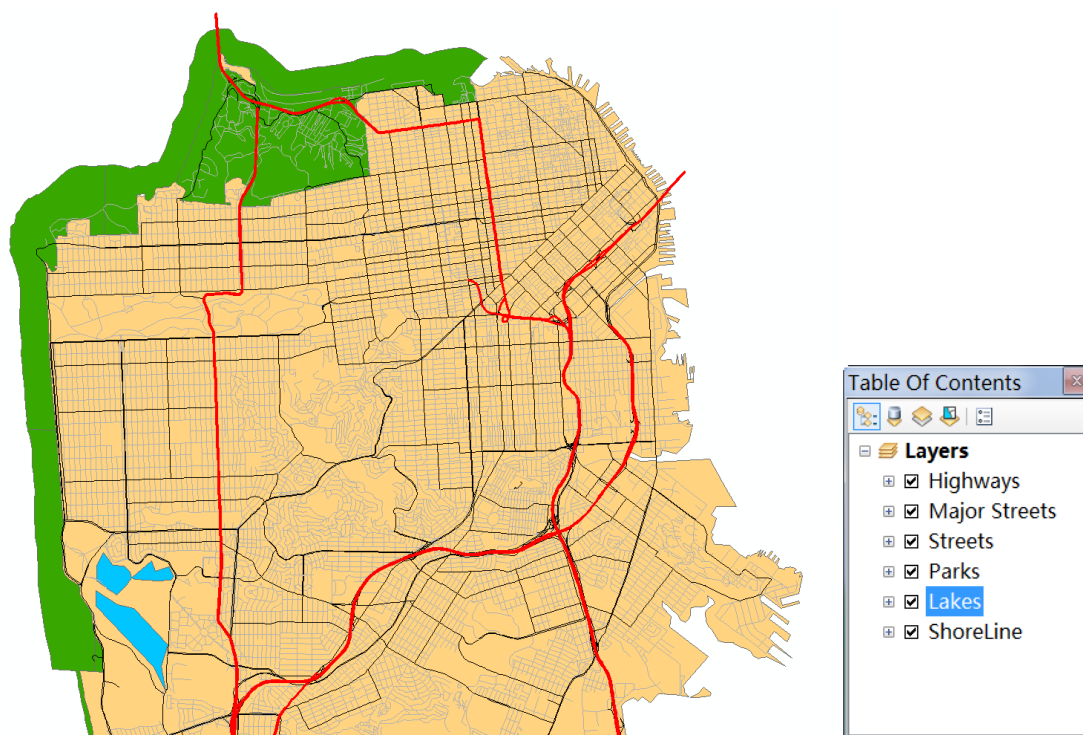


图 13 ArcMap 中显示的地图效果

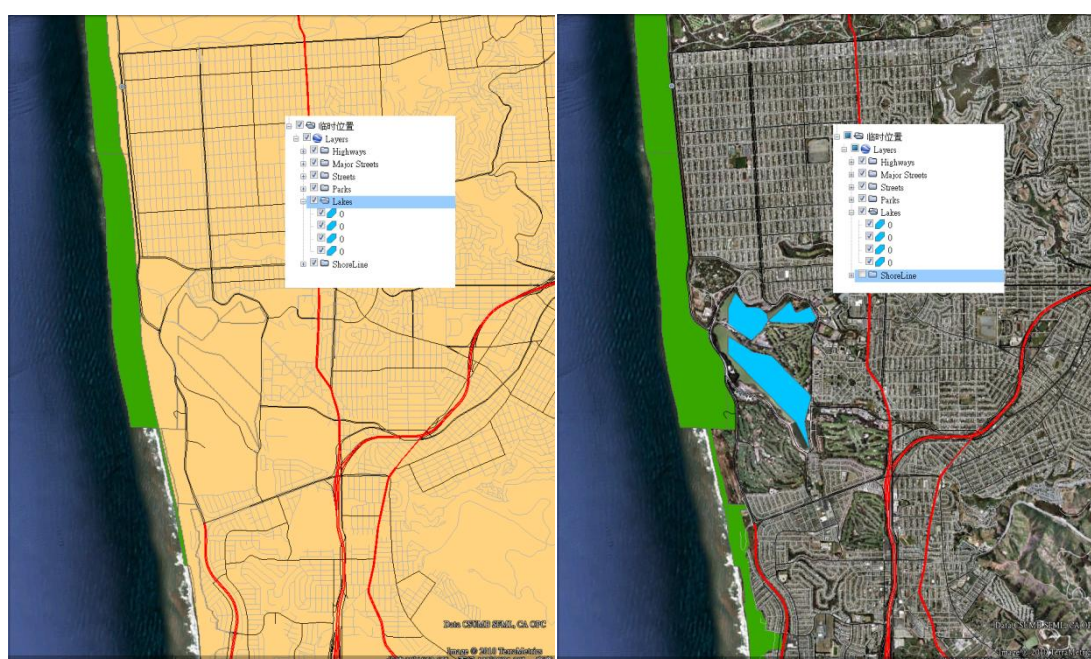


图 14 Google Earth 中显示 KML 的覆盖问题

由于我们选择了导出矢量的数据，因此如果解压缩导出的 KMZ 文件，可以看到其中只包含了一个 doc.kml 文件，有兴趣可以自己打开这个 XML 文件对照 KML 标准看一下。如果导出图片格式，KMZ 中将会包含一个输出的图片，在

doc.xml 则是简单地将这个图片引用进来，这样的 KML 在 Google Earth 中显示并放大后会出现如图 16 的锯齿。

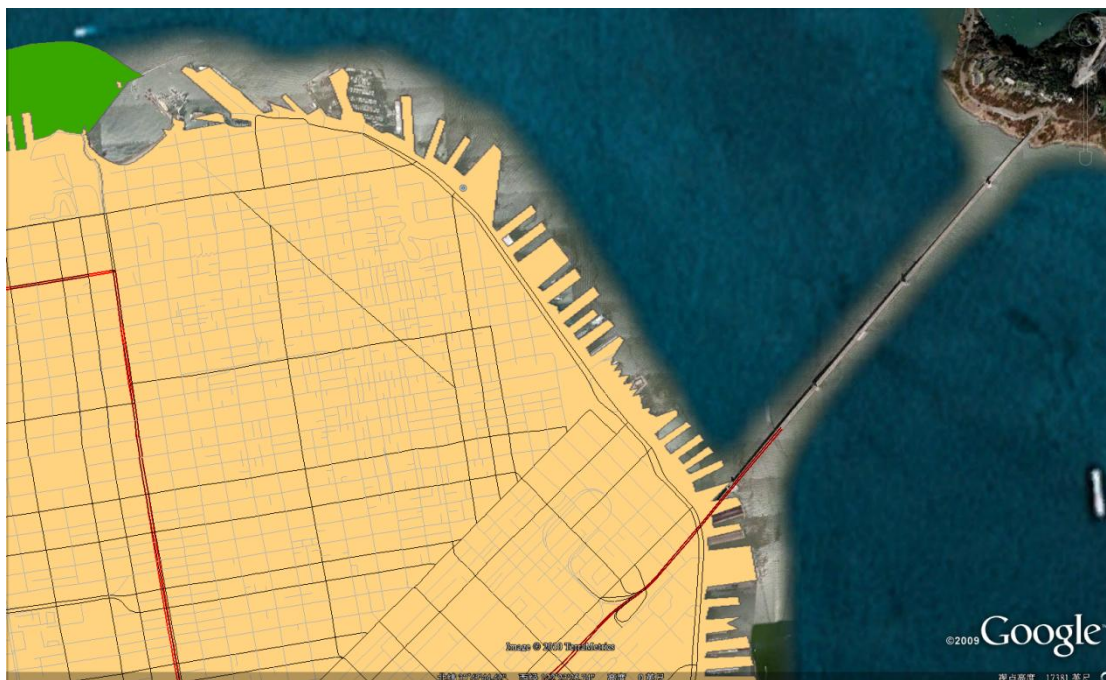


图 15 Google Earth 中显示 ArcGIS 导出的矢量 KML

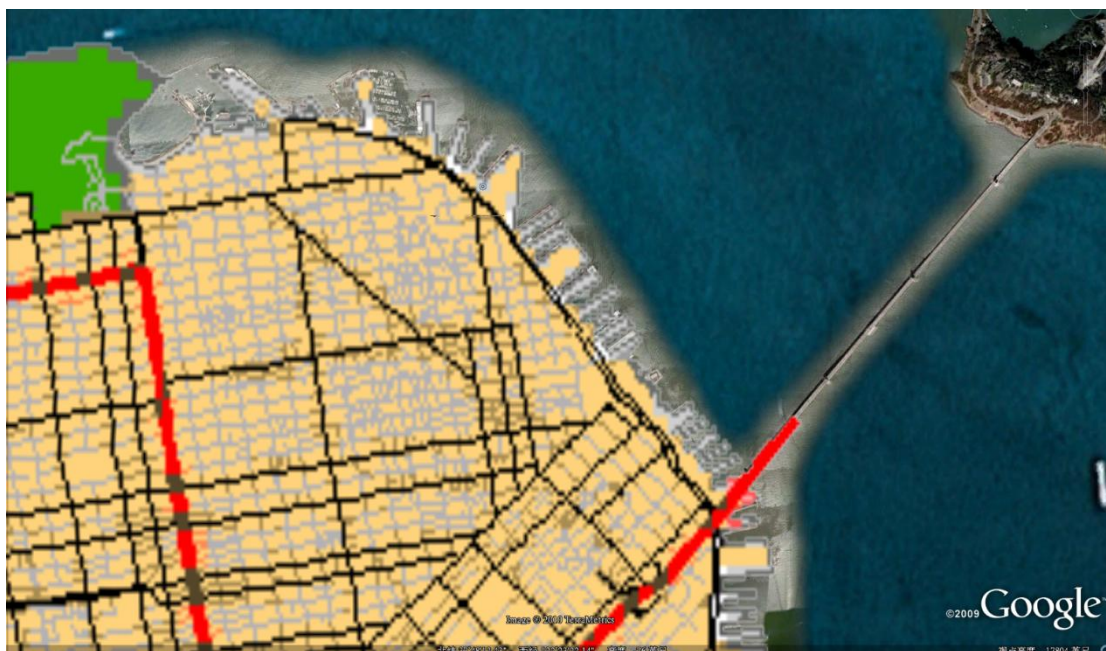


图 16 导出图片结果的 KML 在 Google Earth 中放大的效果

2.ArcGIS Server 发布 KML 服务

在 ArcGIS Server 中,通过勾选 KML 这个 Capability 可以启用服务的 KML 访问,如图 17。

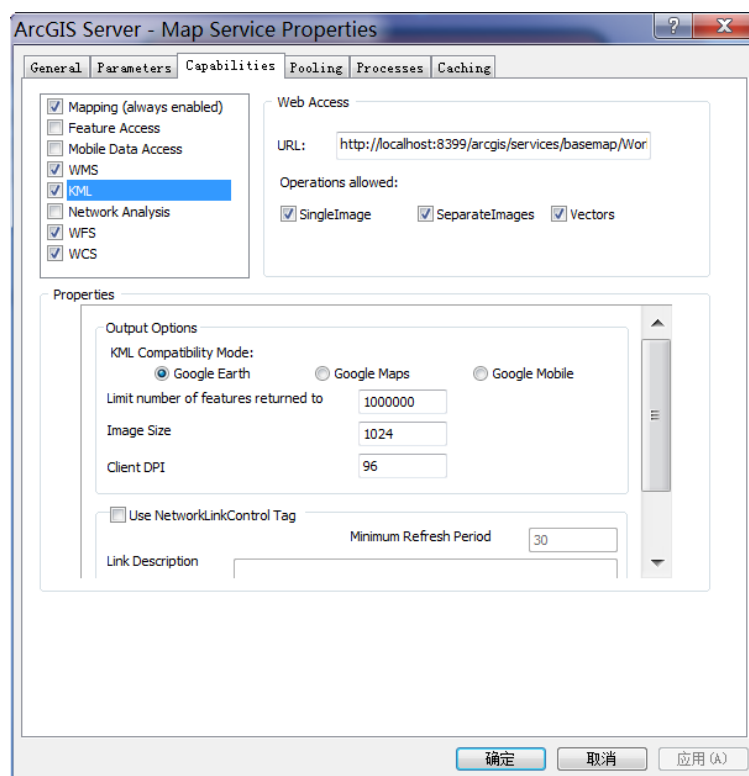


图 17 ArcGIS Server 发布 KML 服务

当发布完这个 KML 服务后,我们就可以直接向 ArcGIS Server 的 REST 接口发送返回 KMZ 格式的请求,然后将这个请求的地址添加到 Google Earth 的网络链接中。

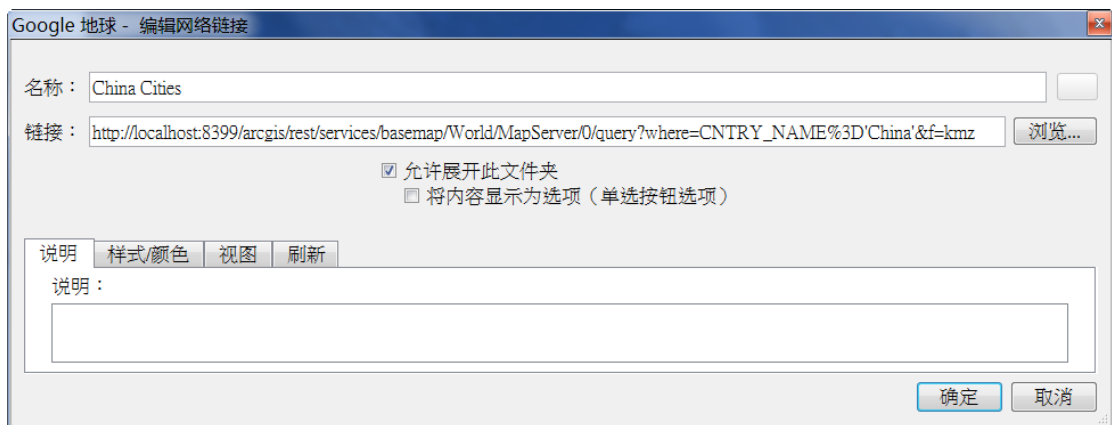


图 18 将 ArcGIS Server 发布的 KML 服务添加到 Google Earth 的效果

VI. OWS-OGC Web 服务通用标准

• 概述

OWS (OGC Web Service Common Implementation Specification) 当前版本是 2.0.0。OWS 描述了 Web 服务通用的一些接口规范，包括请求和响应的内容、请求的参数和编码等。目前，OWS 包括 WFS、WMS、WCS，因此，后续的《VII.WFS-要素 Web 服务》、《VIII.WMS-地图 Web 服务》、《IX.WCS-栅格 Web 服务》都和本章节有关系。

• 服务涉及的基本元素

1.HTTP 请求规则

OWS 可以通过 GET 和 POST 两种方式对服务进行请求。而请求的参数编码也有两种：一种是键值对应 (KVP¹)、另一种是 XML 对象 (XML)。它们的组合情况如下：

	GET	POST
KVP	非 MIME (URL)	MIME : application/x-www-form-urlencoded ²
XML	不支持	MIME : text/xml ³

因此，比如某 OWS 服务的 GetCapabilities 操作，可能会有以下 3 种请求方式，当然，不同种类的服务并不一定实现所有的这些组合：

¹ Key-Value Pair

² http://en.wikipedia.org/wiki/POST_%28HTTP%29

³ http://en.wikipedia.org/wiki/XML_and_MIME

	URL	请求体
GET	http://host:port/path?SERVICE=WFS&REQUEST=GetCapabilities	
POST	http://host:port/path	SERVICE:WFS REQUEST:GetCapabilities
	http://host:port/path	<?xml version="1.0" ?> <GetCapabilities service="WFS" xmlns=http://www.opengis.net/wfs xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:schemaLocation="http://www.opengis.net/wfs .. /wfs/1.1.0/WFS.xsd">

2.HTTP 响应规则

服务处理完请求后，生成结果或异常信息，然后以 MIME 的方式返回客户端。

3.SOAP

在 OWS 服务和 OWS 服务之间可以采用 SOAP 消息进行数据交互(POST)。SOAP 消息是符合 SOAP 规范的 XML ,它需要包括 SOAP 信封、SOAP 消息头、SOAP 消息体等组成部分：

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body>
    ...
    <soap:Fault>...</soap:Fault>
```

```
</soap:Body>  
</soap:Envelope>
```

• GetCapabilities 操作

每个 OWS 服务都包括 GetCapabilities 操作，这个操作返回这个服务的元数据信息。GetCapabilities 包括 2 个必须的参数和若干可选参数，这些参数的取值见下表：

参数	是否必须	取值（示例）
SERVICE	是	WFS 或 WMS 或 WCS
REQUEST	是	GetCapabilities
ACCEPTVERSIONS		1.1.0,1.0.0
SECTIONS		Contents
UPDATESEQUENCE		
ACCEPTFORMATS		text/xml
ACCEPTLANGUAGES		en-US,zh-CN

以下是一个 WFS 使用 KVP 格式的 GetCapabilities 操作：

```
http://www.someserver.com/wfs?  
SERVICE=WFS&  
REQUEST=GetCapabilities
```

以下是一个 WCS 使用 KVP 格式的 GetCapabilities 操作：

```
http://hostname:port/path?  
SERVICE=WCS&  
REQUEST=GetCapabilities&  
ACCEPTVERSIONS=1.0.0 0.8.3&  
SECTIONS=Contents&  
UPDATESEQUENCE=XYZ123&  
ACCEPTFORMATS=text/xml&  
ACCEPTLANGUAGES=en-CA fr-CA
```

以下是一个 WCS 使用 XML 格式的 GetCapabilities 操作：

```
<?xml version="1.0" encoding="UTF-8"?>  
<GetCapabilities xmlns="http://www.opengis.net/ows/2.0"  
  xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.opengis.net/ows/2.0 fragmentGetCapabilitiesRequest.xsd"  
  service="WCS" updateSequence="XYZ123">  
  <AcceptVersions>
```

```

    <Version>1.0.0</Version>
    <Version>0.8.3</Version>
  </AcceptVersions>
  <Sections>
    <Section>Contents</Section>
  </Sections>
  <AcceptFormats>
    <OutputFormat>text/xml</OutputFormat>
  </AcceptFormats>
  <AcceptLanguages>
    <Language>en-CA</Language>
    <Language>fr-CA</Language>
  </AcceptLanguages>
</GetCapabilities>

```

- 其它一般操作

除了 GetCapabilities，不同的 OWS 还包括其它不同操作以实现各自的功能，这些操作都包括以下 3 个必选参数和其它可选参数。当然，不同的操作还包括自身功能有关的其它必选参数和可选参数。

参数	是否必须	取值（示例）
SERVICE	是	WFS 或 WMS 或 WCS
REQUEST	是	GetMap
VERSION	是	1.3.0
ACCEPTLANGUAGES		en-US,zh-CN

以下是一个 WCS 使用 KVP 格式的 GetCoverage 操作：

```

http://hostname:port/path?
SERVICE=WCS&
REQUEST=GetCoverage&
VERSION=1.0.0&
AcceptLanguages=en fr

```

VII. WFS-要素 Web 服务

• 概述

WFS (OpenGIS® Web Feature Service) 当前版本是 1.1.0。WFS 标准定义了一些操作,这些操作允许用户在分布式的环境下通过 HTTP 对空间数据进行查询、编辑等操作。

WFS 服务要求服务的接口必须由 XML 描述,另外数据交互必须由 GML 进行,数据过滤采用 CQL¹语言。

• WFS 种类与操作

当一个客户端想要访问 WFS 服务时,一般会涉及到以下的流程:

1. 通过操作获取 WFS 服务支持的操作和要素类(Feature Type ,可以理解为 WFS 中的数据集)。
2. (可能) 通过操作获取 WFS 服务支持的要素类的定义。
3. 客户端发送某个操作的请求。
4. WFS 服务处理请求。
5. WFS 服务返回处理的结果和状态。

上面几个步骤中所提到的“操作”包括:

1. GetCapabilities (获取服务中的要素类及支持的操作)
2. DescribeFeatureType (描述要素类的信息)
3. GetFeature (获取要素)
4. GetGmlObject (通过 XLink 获取 GML 对象)
5. Transaction (创建、更新、删除数据的事务操作)
6. LockFeature (在事务过程中锁定要素)

但是,这些操作并不是必须全部实现,而是实现全部或部分。根据所支持的操作不同,WFS 可以分为 3 类:

¹ OGC Common Query Language, 参考《OGC Catalogue Service 2.0.2》标准的 6.2 章节。

1. Basic WFS (就是最常被提及的 WFS , 必须支持 GetCapabilities/ DescribeFeatureType/ GetFeature 操作 , 在功能上意味着提供一个只读的数据服务)
2. XLink WFS (必须在 Basic WFS 基础上加上 GetGmlObject 操作)
3. Transaction WFS (也有称为 WFS-T , 必须在 Basic WFS 基础上加上 Transaction 操作以支持编辑数据 , 另外也可以加上 GetGmlObject/LockFeature 操作)

关于服务涉及的基本元素 , 可以参考前面的章节 : 《服务涉及的基本元素》。

注意 , 在后面的内容中 , 服务的操作只介绍 Basic WFS 和 Transaction WFS 中需要实现的操作 , 也就是说 GetCapabilities、DescribeFeatureType、GetFeature 和 Transaction 操作。

• GetCapabilities 操作

1.KVP 格式请求

GetCapabilities 操作需要以下的参数 :

参数	是否必须	默认值
SERVICE	是	WFS
REQUEST=GetCapabilities	是	

以下是一个 WFS 使用 KVP 格式的 GetCapabilities 操作示例 :

```
http://www.someserver.com/wfs?
SERVICE=WFS&
REQUEST=GetCapabilities
```

2.XML 格式请求

以下是一个 WFS 使用 XML 格式的 GetCapabilities 操作示例 :

```
<?xml version="1.0"?>
<GetCapabilities service="WFS" xmlns="http://www.opengis.net/wfs"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd" />
```

3. 响应示例

以下是一个 WFS 的 GetCapabilities 操作的响应示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:WFS_Capabilities xmlns:ows="http://www.opengis.net/ows"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs ../wfs.xsd" version="1.1.0"
  updateSequence="0">
  <!--
    =====
  -->
  <!-- SERVICE IDENTIFICATION SECTION -->
  <!--
    =====
  -->
  <ows:ServiceIdentification>
    <ows:ServiceType>WFS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
    <ows:Title>OGC Member WFS</ows:Title>
    ...
  </ows:ServiceIdentification>
  <!--
    =====
  -->
  <!-- SERVICE PROVIDER SECTION -->
  <!--
    =====
  -->
  <ows:ServiceProvider>
    <ows:ProviderName>BlueOx</ows:ProviderName>
    <ows:ServiceContact>
      ...
    </ows:ServiceContact>
  </ows:ServiceProvider>
  <!--
    =====
  -->
  <!-- OPERATIONS METADATA SECTION -->
  <!--
    =====
  -->
  <ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
      ...
    </ows:Operation>
    <ows:Operation name="DescribeFeatureType">
      ...
    </ows:Operation>
  </ows:OperationsMetadata>

```

```

<ows:Operation name= "GetFeature">
    ...
</ows:Operation>
<ows:Operation name= "GetFeatureWithLock">
    ...
</ows:Operation>
<ows:Operation name= "GetGMLObject">
    ...
</ows:Operation>
<ows:Operation name= "LockFeature">
    ...
</ows:Operation>
<ows:Operation name= "Transaction">
    ...
</ows:Operation>
<ows:Parameter name= "srsName">
    <ows:Value>EPSG:4326</ows:Value>
</ows:Parameter>
<ows:Constraint name= "DefaultMaxFeatures">
    <ows:Value>10000</ows:Value>
</ows:Constraint>
...
</ows:OperationsMetadata>
<!--
=====
-->
<!-- FEATURE TYPE LIST SECTION -->
<!--
=====
-->
<wfs:FeatureTypeList>
    <wfs:FeatureType xmlns:bo= "http://www.BlueOx.org/BlueOx">
        <wfs:DefaultSRS>EPSG:62696405</wfs:DefaultSRS>
        <wfs:OutputFormats>
            <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
        </wfs:OutputFormats>
        <ows:WGS84BoundingBox>
            <ows:LowerCorner>-180 -90</ows:LowerCorner>
            <ows:UpperCorner>180 90</ows:UpperCorner>
        </ows:WGS84BoundingBox>
        ...
    </wfs:FeatureType>
</wfs:FeatureTypeList>
<!--
=====
-->
<!-- SERVES GML OBJECT TYPE LIST SECTION -->
<!--
=====
-->
<wfs:ServesGMLObjectTypeList>
    <wfs:GMLObjectType xmlns:bo= "http://www.BlueOx.org/BlueOx">
        <wfs:Name>bo:OxType</wfs:Name>

```



```

    <wfs:Title>Babe's Lineage</wfs:Title>
    <wfs:OutputFormats>
      <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
      <wfs:Format>text/xhtml</wfs:Format>
    </wfs:OutputFormats>
  </wfs:GMLObjectType>
</wfs:ServesGMLObjectTypeList>
<!--
=====
-->
<!-- SUPPORTS GML OBJECT TYPE LIST SECTION -->
<!--
=====
-->
<wfs:SupportsGMLObjectTypeList>
  <wfs:GMLObjectType>
    <wfs:Name>gml:PointType</wfs:Name>
    <wfs:OutputFormats>
      <wfs:Format>text/xml; subtype=gml/3.1.1</wfs:Format>
      <wfs:Format>text/xhtml</wfs:Format>
    </wfs:OutputFormats>
  </wfs:GMLObjectType>
  ...
</wfs:SupportsGMLObjectTypeList>
<!--
=====
-->
<!-- FILTER CAPABILITIES SECTION -->
<!--
=====
-->
<ogc:Filter_Capabilities>
  <ogc:Spatial_Capabilities>
    <ogc:GeometryOperands>
      <ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>
      <ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
      <ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
      <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
      ...
    </ogc:GeometryOperands>
    <ogc:SpatialOperators>
      <ogc:SpatialOperator name="BBOX" />
      <ogc:SpatialOperator name="Equals" />
      <ogc:SpatialOperator name="Disjoint" />
      <ogc:SpatialOperator name="Intersects" />
      <ogc:SpatialOperator name="Touches" />
      <ogc:SpatialOperator name="Crosses" />
      <ogc:SpatialOperator name="Within" />
      <ogc:SpatialOperator name="Contains" />
      <ogc:SpatialOperator name="Overlaps" />
      <ogc:SpatialOperator name="Beyond" />
    </ogc:SpatialOperators>
  </ogc:Spatial_Capabilities>

```

```

<ogc:Scalar_Capabilities>
  <ogc:LogicalOperators />
  <ogc:ComparisonOperators>
    <ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
    <ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>
    <ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>
    <ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
    <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
    <ogc:ComparisonOperator>NotEqualTo</ogc:ComparisonOperator>
    <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
    <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
    <ogc:ComparisonOperator>NullCheck</ogc:ComparisonOperator>
  </ogc:ComparisonOperators>
  <ogc:ArithmeticOperators>
    <ogc:SimpleArithmetic />
    <ogc:Functions>
      <ogc:FunctionNames>
        <ogc:FunctionName nArgs= "1">MIN</ogc:FunctionName>
        <ogc:FunctionName nArgs= "1">MAX</ogc:FunctionName>
        <ogc:FunctionName nArgs= "1">SIN</ogc:FunctionName>
        <ogc:FunctionName nArgs= "1">COS</ogc:FunctionName>
        <ogc:FunctionName nArgs= "1">TAN</ogc:FunctionName>
      </ogc:FunctionNames>
    </ogc:Functions>
  </ogc:ArithmeticOperators>
</ogc:Scalar_Capabilities>
<ogc:Id_Capabilities>
  <ogc:EID />
  <ogc:FID />
</ogc:Id_Capabilities>
</ogc:Filter_Capabilities>
</wfs:WFS_Capabilities>

```

• DescribeFeatureType 操作

1.KVP 格式请求

DescribeFeatureType 操作需要以下的参数：

参数	是否必须	默认值
VERSION	是	1.1.0
SERVICE	是	WFS
REQUEST=DescribeFeatureType	是	
TYPENAME		
OUTPUTFORMAT		text/xml; subtype=gml/3.1.1

以下是一个 WFS 使用 KVP 格式的 DescribeFeatureType 操作示例：

```
http://www.someserver.com/wfs?  
SERVICE=WFS&  
VERSION=1.1.0&  
REQUEST=DescribeFeatureType&  
TYPENAME=TreesA_1M,BuiltUpA_1M
```

2.XML 格式请求

以下是一个 WFS 使用 XML 格式的 DescribeFeatureType 操作示例：

```
<?xml version="1.0"?>  
<DescribeFeatureType version="1.1.0" service="WFS"  
  outputFormat="text/xml; subtype=gml/3.1.1" xmlns="http://www.opengis.net/wfs"  
  xmlns:myns="http://www.myserver.com/myns" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">  
  <TypeName>myns:Person</TypeName>  
</DescribeFeatureType>
```

3. 响应示例

以下是一个 WFS 的 DescribeFeatureType 操作的响应示例：

```
<?xml version="1.0"?>  
<wfs:FeatureCollection xmlns="http://www.someserver.com/myns"  
  xmlns:myns="http://www.someserver.com/myns" xmlns:wfs="http://www.opengis.net/wfs"  
  xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd  
  http://www.someserver.com/myns ex10.xsd">  
  <gml:boundedBy>  
    <gml:Envelope>  
      <gml:coord>  
        <gml:X>10</gml:X>  
        <gml:Y>10</gml:Y>  
      </gml:coord>  
      <gml:coord>  
        <gml:X>20</gml:X>  
        <gml:Y>20</gml:Y>  
      </gml:coord>  
    </gml:Envelope>  
  </gml:boundedBy>  
  <gml:featureMember>
```

```

<Person>
  <myns:lastName>Smith</myns:lastName>
  <myns:firstName>Fred</myns:firstName>
  <myns:age>35</myns:age>
  <myns:sex>Male</myns:sex>
  <myns:location>
    <gml:Point>
      <gml:pos>15 15</gml:pos>
    </gml:Point>
  </myns:location>
  <myns:mailAddress>
    <myns:Address>
      <myns:streetName>Main St.</myns:streetName>
      <myns:streetNumber>5</myns:streetNumber>
      <myns:city>SomeCity</myns:city>
      <myns:province>Someprovince</myns:province>
      <myns:postalCode>X1X 1X1</myns:postalCode>
      <myns:country>Canada</myns:country>
    </myns:Address>
  </myns:mailAddress>
</Person>
</gml:featureMember>
</wfs:FeatureCollection>

```

• GetFeature 操作

1. KVP 格式请求

GetFeature 操作需要以下的参数：

参数	是否必须	默认值
VERSION	是	1.1.0
SERVICE	是	WFS
REQUEST=GetFeature	是	
TYPENAME	是	
OUTPUTFORMAT		text/xml; subtype=gml/3.1.1
BBOX		
FILTER		
SORTBY		
MAXFEATURES		
PROPERTYNAME		
SRSNAME		
FEATUREID		

EXPIRY		
RESULTTYPE		results
FEATUREVERSION		

以下是一个 WFS 使用 KVP 格式的 GetFeature 操作示例：

```
http://www.someserver.com/wfs ?
SERVICE=WFS&
VERSION=1.1.0&
REQUEST=GetFeature&
PROPERTYNAME=InWaterA_1M/wkbGeom,InWaterA_1M/tileId&
TYPENAME=InWaterA_1M&
FILTER= <Filter> <Within> <PropertyName>InWaterA_1M/wkbGeom<PropertyName>
      <gml:Envelope> <gml:lowerCorner> 10,10</gml:lowerCorner>
      <gml:upperCorner> 20 20</gml:upperCorner> </gml:Envelope> </Within> </Filter>
```

2.XML 格式请求

以下是一个 WFS 使用 XML 格式的 GetFeature 操作示例：

```
<?xml version="1.0"?>
<GetFeature version="1.1.0" service="WFS" handle="Query01"
  xmlns="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:gml="http://www.opengis.net/gml" xmlns:myns="http://www.someserver.com/myns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
  <Query typeName="myns:Hydrography">
    <wfs:PropertyName>myns:geoTemp</wfs:PropertyName>
    <wfs:PropertyName>myns:depth</wfs:PropertyName>
    <ogc:Filter>
      <ogc:Not>
        <ogc:Disjoint>
          <ogc:PropertyName>myns:geoTemp</ogc:PropertyName>
          <gml:Envelope srsName="EPSG:63266405">
            <gml:lowerCorner>
              -57.9118 46.2023
            <gml:lowerCorner>
              <gml:upperCorner>-46.6873 51.8145</gml:upperCorner>
            </gml:Envelope>
          </ogc:Disjoint>
        </ogc:Not>
      </ogc:Filter>
    </Query>
  </GetFeature>
```

3. 响应示例

以下是一个 WFS 的 GetFeature 操作的响应示例：

```
<?xml version="1.0"?>
<wfs:FeatureCollection xmlns="http://www.someserver.com/myns"
  xmlns:wfs="http://www.opengis.net/wfs" xmlns:gml="http://www.opengis.net/gml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.someserver.com/myns Hydrography.xsd
http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
  <gml:boundedBy>
    <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
      <gml:lowerCorner>10 10</gml:lowerCorner>
      <gml:upperCorner>20 20</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <gml:featureMember>
    <HydrographyHydrography gml:id="HydrographyHydrography.450">
      <geoTemp>
        <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
          <gml:pos>10 10</gml:pos>
        </gml:Point>
      </geoTemp>
      <depth>565</depth>
    </HydrographyHydrography>
  </gml:featureMember>
  <gml:featureMember>
    <HydrographyHydrography gml:id="HydrographyHydrography.450">
      <geoTemp>
        <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#63266405">
          <gml:pos>10 11</gml:pos>
        </gml:Point>
      </geoTemp>
      <depth>566</depth>
    </HydrographyHydrography>
  </gml:featureMember>
</wfs:FeatureCollection>
```

• Transaction 操作

1. KVP 格式请求

Transaction 操作使用 KVP 格式请求目前只支持 Delete(Insert 和 Update

必须通过 XML 格式请求发送)。Transaction 操作需要以下的参数：

参数	是否必须	默认值
VERSION	是	1.1.0
SERVICE	是	WFS
REQUEST=Transaction	是	
OPERATION=Delete	是	
TYPENAME	是	
RELEASEACTION		
FILTER		
BBOX		
FEATUREID		

以下是一个 WFS 使用 KVP 格式的 Transaction 操作示例：

```
http://www.someserver.com/wfs?
SERVICE=WFS&
VERSION=1.1.0&
REQUEST=Transaction&
OPERATION=Delete&
TYPENAME=InWaterA_1M,BuiltUpA_1M&
FILTER=(  
<Filter>  
<Within>  
<PropertyName>InWaterA_1M/wkbGeom<PropertyName>  
<gml:Envelope>  
<gml:lowerCorner>10 10</gml:lowerCorner>  
<gml:upperCorner>20 20</gml:upperCorner>  
</gml:Envelope>  
</Within>  
</Filter>)(  
<Filter>  
<Within>  
<PropertyName>BuiltUpA_1M/wkbGeom  
<PropertyName>  
<gml:Envelope>  
<gml:lowerCorner>10 10</gml:lowerCorner>  
<gml:upperCorner>20,20</gml:upperCorner>  
</gml:Envelope>  
</Within>  
</Filter>)
```

2.XML 格式请求

以下是一个 WFS 使用 XML 格式的 Transaction 操作示例：

```
<?xml version="1.0"?>
<wfs:Transaction version="1.1.0" service="WFS"
  handle="Transaction 01" xmlns="http://www.someserver.com/myns"
  xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.someserver.com/myns
http://www.someserver.com/wfs/cwwfs.cgi?
request=DESCRIBEFEATURETYPE&
typename=ELEVP_1M,RoadL_1M,BuiltUpA_1M
http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
<!--
```

Create a new instance of feature type RoadL_1M which has complex properties segment and roadType.

```
-->
<wfs:Insert handle= "ComplexInsert">
  <RoadL_1M>
    <name>Highway 401</name>
    <segment>
      <designation>SEG_A41</designation>
      <geometry>
        <gml:LineString gid= "e3"
          srsName= "http://www.opengis.net/gml/srs/epsg.xml#63266405">
          <gml:posList>...</gml:posList>
        </gml:LineString>
      </geometry>
    </segment>
    <roadType>
      <surfaceType>Asphalt</surfaceType>
      <nLanes>12</nLanes>
      <grade>15</grade>
    </roadType>
  </RoadL_1M>
</wfs:Insert>
<!--
  Update the designation of a particular range of segments which are now
  being collapsed into a single segment. The The filter uses an XPath
  expression to reference the designation property
-->
<wfs:Update typeName= "RoadL_1M">
  <wfs:Property>
    <wfs:Name>RoadL_1M/segment/designation</wfs:Name>
    <wfs:Value>SEG_A60</wfs:Value>
  </wfs:Property>
  <ogc:Filter>
    <ogc:PropertyIsBetween>
      <ogc:PropertyName>RoadL_1M/segment/designation</ogc:PropertyName>
      <ogc:LowerBoundary>
        <ogc:Literal>SEG_A60</ogc:Literal>
      </ogc:LowerBoundary>
      <ogc:UpperBoundary>
        <ogc:Literal>SEG_A69</ogc:Literal>
      </ogc:UpperBoundary>
    </ogc:PropertyIsBetween>
  </ogc:Filter>
</wfs:Update>
<!-- Delete the feature instance BuiltUpA_1M.1013. -->
<wfs>Delete typeName= "BuiltUpA_1M">
  <ogc:Filter>
    <ogc:GmlObjectId gml:id= "BuiltUpA_1M.1013" />
  </ogc:Filter>
</wfs>Delete>
</wfs:Transaction>
```


3. 响应示例

以下是一个 WFS 的 Transaction 操作的响应示例：

```
<?xml version="1.0"?>
<wfs:TransactionResponse version="1.1.0"
  xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wfs ../wfs/1.1.0/WFS.xsd">
  <wfs:TransactionSummary>
    <wfs:totalInserted>1</wfs:totalInserted>
    <wfs:totalUpdated>1</wfs:totalUpdated>
    <wfs:totalDeleted>1</wfs:totalDeleted>
  </wfs:TransactionSummary>
  <wfs:InsertResults>
    <wfs:Feature handle="ComplexInsert">
      <ogc:FeatureId fid="RoadL_1M.1553"/>
    </wfs:Feature>
    <wfs:Feature handle="Statement 2">
      <ogc:FeatureId fid="RoadL_1M.509876"/>
    </wfs:Feature>
    <wfs:Feature handle="Statement 2">
      <ogc:FeatureId fid="BuiltUpA_1M.509877"/>
    </wfs:Feature>
  </wfs:InsertResults>
</wfs:TransactionResponse>
```

• ArcGIS Server 对 WFS 的支持

ArcGIS Server 10 中支持的 WFS 版本为最新的 1.1.0。在 ArcGIS Server 中，只需简单地勾选的 Capabilities 选项卡中可以选择支持 WFS，如图 19。如果勾选了 Transaction 复选框，那么通过该 WFS 还可以进行数据更新（注意，需要是 SDE 数据源）。

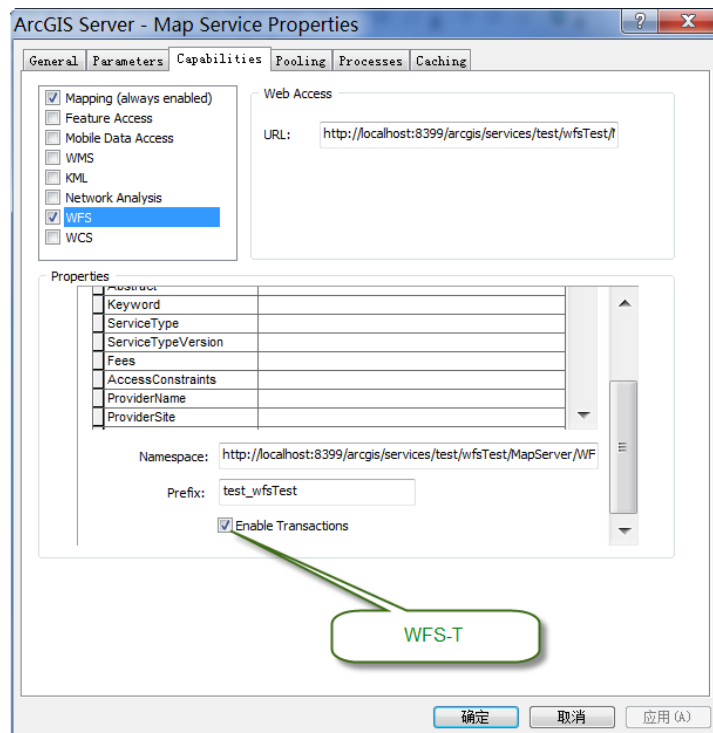


图 19 ArcGIS Server 发布 WFS 服务

现在我们在浏览器中执行一个 GetFeature 操作，查询名为 “Beijing” 的对象：

```
http://localhost:8399/arcgis/services/test/wfsTest/MapServer/WFSServer?
request=getfeature&
typename=test_wfsTest:sde_cities&
filter=(
  <ogc:Filter>
    <ogc:PropertyIsEqualTo>
      <ogc:PropertyName>CITY_NAME</ogc:PropertyName>
      <ogc:Literal>Beijing</ogc:Literal>
    </ogc:PropertyIsEqualTo>
  </ogc:Filter>
)
```

这个操作返回如下的结果：

```
<wfs:FeatureCollection
  xsi:schemaLocation= 'http://localhost:8399/arcgis/services/test/wfsTest/MapServer/WFSServer
http://localhost:8399/arcgis/services/test/wfsTest/MapServer/WFSServer?request=DescribeFeatureType%26version
=1.1.0%26typename=sde_cities http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd'
  xmlns:test_wfsTest= 'http://localhost:8399/arcgis/services/test/wfsTest/MapServer/WFSServer'
  xmlns:gml= 'http://www.opengis.net/gml' xmlns:wfs= 'http://www.opengis.net/wfs'
  xmlns:xlink= 'http://www.w3.org/1999/xlink' xmlns:xsi= 'http://www.w3.org/2001/XMLSchema-instance'>
  <gml:boundedBy>
    <gml:Envelope srsName= 'urn:ogc:def:crs:EPSG:6.9:4326'>
      <gml:lowerCorner>-86.0026169900000007
```

```

        -176.15156363599999</gml:lowerCorner>
        <gml:upperCorner>102.93161888100001
        179.22188769499999</gml:upperCorner>
    </gml:Envelope>
</gml:boundedBy>
<gml:featureMember>
    <test_wfsTest:sde_cities gml:id='F3_474'>
        <test_wfsTest:OBJECTID>474</test_wfsTest:OBJECTID>
        <test_wfsTest:CITY_NAME>Beijing</test_wfsTest:CITY_NAME>
        <test_wfsTest:GMI_ADMIN>CHN-BJN</test_wfsTest:GMI_ADMIN>
        <test_wfsTest:ADMIN_NAME>Beijing</test_wfsTest:ADMIN_NAME>
        <test_wfsTest:FIPS_CNTRY>CH</test_wfsTest:FIPS_CNTRY>
        <test_wfsTest:CNTRY_NAME>China</test_wfsTest:CNTRY_NAME>
        <test_wfsTest:STATUS>National and provincial capital</test_wfsTest:STATUS>
        <test_wfsTest:POP_RANK>1</test_wfsTest:POP_RANK>
        <test_wfsTest:POP_CLASS>5,000,000 and greater</test_wfsTest:POP_CLASS>
        <test_wfsTest:PORT_ID>0</test_wfsTest:PORT_ID>
        <test_wfsTest:LABEL_FLAG>1</test_wfsTest:LABEL_FLAG>
        <test_wfsTest:NEAR_FID>40</test_wfsTest:NEAR_FID>
        <test_wfsTest:NEAR_DIST>185.74736243999999</test_wfsTest:NEAR_DIST>
        <test_wfsTest:Shape>
            <gml:Point>
                <gml:pos>39.906189088000019 116.38803663600004</gml:pos>
            </gml:Point>
        </test_wfsTest:Shape>
    </test_wfsTest:sde_cities>
</gml:featureMember>
</wfs:FeatureCollection>

```

接着，我想删掉这个对象，就要使用 Transaction 操作。在 ArcGIS Server 中，执行删除之前还需要 lockId 属性，因此还需要通过 GetFeatureWithLock 操作获取一个锁：

```

http://localhost:8399/arcgis/services/test/wfsTest/MapServer/WFSServer?
request=getfeaturewithlock&
typename=test_wfsTest:sde_cities&
filter=(
  <ogc:Filter>
    <ogc:PropertyIsEqualTo>
      <ogc:PropertyName>CITY_NAME</ogc:PropertyName>
      <ogc:Literal>Beijing</ogc:Literal>
    </ogc:PropertyIsEqualTo>
  </ogc:Filter>
)

```

这个操作返回 lockId 值为

"{CEBC222E-00AD-49F5-A0E9-9F4CB98EE07E}"，接下来，我们通过一个

POST 请求对这个对象进行删除，请求体内容如下：

```
<wfs:Transaction version="1.1.0" service="WFS"
  xmlns="http://www.someserver.com/myns"
  xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <wfs:LockId>{CEBC222E-00AD-49F5-A0E9-9F4CB98EE07E}</wfs:LockId>
  <wfs:Delete typeName="test_wfsTest:sde_cities">
    <ogc:Filter>
      <ogc:GmlObjectId gml:id="F3_474"/>
    </ogc:Filter>
  </wfs:Delete>
</wfs:Transaction>
```

删除成功后将会返回如下的结果：

```
<wfs:TransactionResponse version="1.1.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:wfs="http://www.opengis.net/wfs">
  <wfs:TransactionSummary>
    <wfs:totalDeleted>1</wfs:totalDeleted>
  </wfs:TransactionSummary>
</wfs:TransactionResponse>
```

我们也可以在其它的客户端中直接加载这个 WFS 服务，比如使用 uDig：

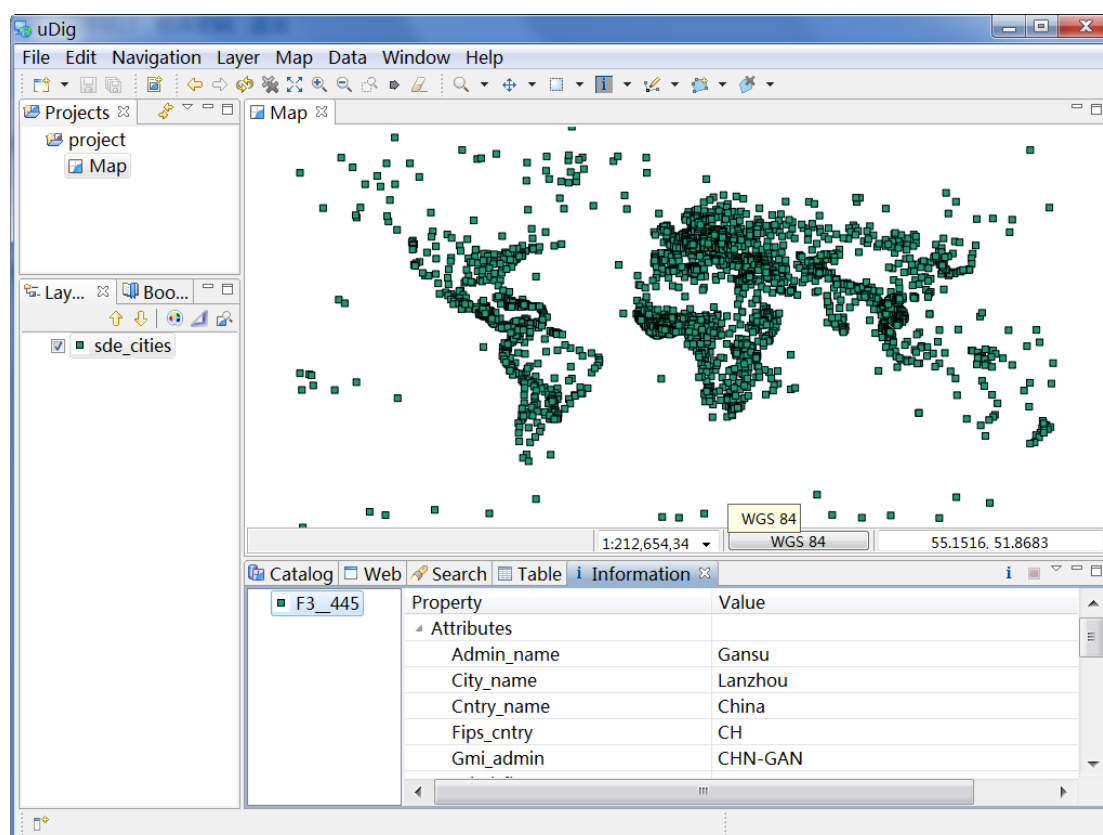


图 20 uDig 中加载 ArcGIS Server 发布的 WFS 服务

VIII. WMS-地图 Web 服务

• 概述

WMS (OpenGIS® Web Map Service) 当前版本是 1.3.0。WMS 标准定义了一些操作 ,这些操作允许用户在分布式的环境下通过 HTTP 对空间数据进行出图等操作。

• WMS 种类与操作

比起 WFS , WMS 的操作要简单的多 :

1. GetCapabilities (获取服务中的要素类及支持的操作)
2. GetMap (获取地图)
3. GetFeatureInfo (根据地图上的像素点获取更详细的要素信息 , 类似 Identify 功能)

同样 , 这些操作并不是必须全部实现 , 而是实现全部或部分。根据所支持的

操作不同 , WMS 可以分为 2 类 :

1. Basic WMS (就是最常被提及的 WMS , 必须支持 GetCapabilities/ GetMap 操作)
2. Queryable WFS (必须在 Basic WMS 基础上加上 GetFeatureInfo 操作)

关于服务涉及的基本元素 , 可以参考前面的章节 : 《服务涉及的基本元素》。

WMS 中只规定了 KVP 格式的请求 , 因此下面介绍一下 WMS 的请求和响应内容。

• GetCapabilities 操作

GetCapabilities 操作需要以下的参数 :

参数	是否必须
VERSION	
SERVICE=WMS	是
REQUEST=GetCapabilities	是
FORMAT	
UPDATESEQUENCE	

以下是一个 WMS 的 GetCapabilities 操作示例：

```
http://www.someserver.com/wms?
VERSION=1.3.0&
SERVICE=WMS&
REQUEST=GetCapabilities
```

这样的请求返回的响应示例：

• GetMap 操作

GetMap 操作需要以下的参数：

参数	是否必须
VERSION	是
REQUEST=GetMap	是
LAYERS	是
STYLES	是
CRS	是
BBOX	是
WIDTH	是
HEIGHT	是
FORMAT	是
TRANSPARENT	
BGCOLOR	
EXCEPTIONS	
TIME	
ELEVATION	

以下是一个 WMS 的 GetMap 操作示例：

```
http://www.someserver.com/wms?
VERSION=1.3.0&REQUEST=GetMap&
CRS=CRS:84&BBOX=-97.105,24.913,-78.794,36.358&
WIDTH=560&HEIGHT=350&LAYERS=BUILTUPA_1M,COASTL_1M,POLBNDL_1M&
```

STYLES=0xFF8080,0X101040,BLACK&FORMAT=image/png&BGCOLOR=0xFFFFFF&
TRANSPARENT=TRUE&EXCEPTIONS=INIMAGE

该请求的响应就是一张图片。

• GetFeatureInfo 操作

GetFeatureInfo 操作需要以下的参数：

参数	是否必须
VERSION	是
REQUEST=GetFeatureInfo	是
GetMap 请求参数	是
QUERY_LAYERS	是
INFO_FORMAT	是
FEATURE_COUNT	
I	是
J	是
EXCEPTIONS	

以下是一个 WMS 的 GetFeatureInfo 操作示例：

```
http://www.someserver.com/wms?  
request=GetFeatureInfo&  
VERSION=1.3.0&FORMAT=image/jpeg&  
BBOX=-4.0325,-2.9078,40.5189,41.6436&  
QUERY_LAYERS=Countries,Cities&  
INFO_FORMAT=text/html  
HEIGHT=400&WIDTH=400&  
STYLES=&  
I=290&J=246&
```

• ArcGIS Server 对 WMS 的支持

ArcGIS Server 10 中支持的 WMS 版本为最新的 1.3.0。在 ArcGIS Server 中，只需简单地勾选的 Capabilities 选项卡中可以选择支持 WMS，如图 21。

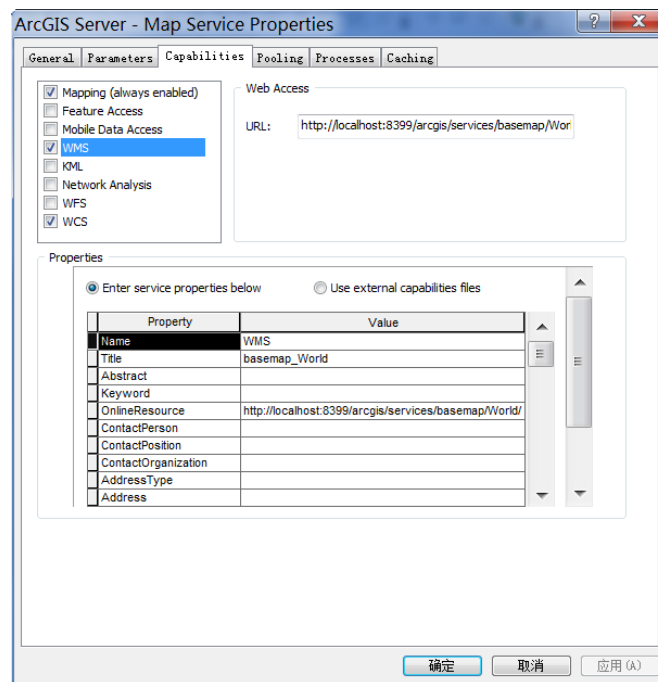
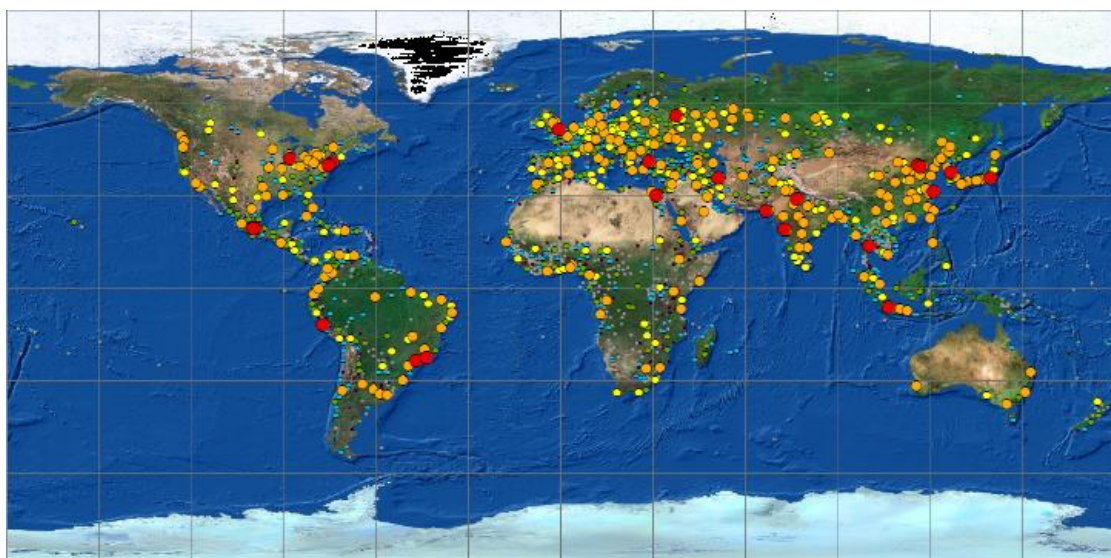


图 21 ArcGIS Server 发布 WMS 服务

现在我们通过浏览器直接发送请求获取一张动态地图：

```
http://localhost:8399/arcgis/services/basemap/World/MapServer/WMServer?
VERSION=1.3.0&REQUEST=GetMap&
CRS=CRS:84&BBOX=-180,-90,180,90&
WIDTH=720&HEIGHT=360&LAYERS=0,1,2&
STYLES=,,&FORMAT=image/png&
TRANSPARENT=TRUE
```

这个请求返回一张这样的图片，这和 ArcMap 中的配置是完全一致的：



然后，我们模拟某个用户在图片上进行了一个点击，想看看点击处的点对象

都有什么属性值，这时会有一个这样的请求发送到 WMS 服务上：

```
http://localhost:8399/arcgis/services/basemap/World/MapServer/WMServer?
  VERSION=1.3.0&REQUEST=GetFeatureInfo&
  CRS=CRS:84&BBOX=-180,-90,180,90&
  WIDTH=720&HEIGHT=360&
  INFO_FORMAT=text/xml&
  QUERY_LAYERS=2&
  I=593&J=100
```

这个请求返回如下的结果，很显然，刚才在地图上的北京附近进行了点击，

服务返回回来 “Beijing” 这个要素的所有属性：

```
<?xml version= "1.0"?>
<FeatureInfoResponse xmlns:esri_wms= "http://www.esri.com/wms"
  xmlns= "http://www.esri.com/wms">
  <FIELDS ObjectID= "2165" Shape= "Null" CITY_NAME= "Beijing"
    GMI_ADMIN= "CHN-BJN" ADMIN_NAME= "Beijing" FIPS_CNTRY= "CH" CNTRY_NAME= "China"
    STATUS= "National and provincial capital" POP_RANK= "1"
    POP_CLASS= "5,000,000 and greater" PORT_ID= "0" LABEL_FLAG= "1"> </FIELDS>
</FeatureInfoResponse>
```

IX. WCS-栅格 Web 服务

• 概述

WCS (OpenGIS® Web Coverage Service) 当前版本是 1.1.2。WCS 标准定义了一些操作，这些操作允许用户访问 “Coverage” 数据，如卫星影像、数字高程数据等，也就是栅格数据。

• WCS 的操作

WCS 包括以下 3 个操作：

1. GetCapabilities (获取服务的元信息)
2. DescribeCoverage (获取 Coverage 的描述信息)
3. GetCoverage (获取 Coverage)

关于服务涉及的基本元素，可以参考前面的章节：《服务涉及的基本元素》。

• GetCapabilities 操作

1.KVP 格式请求

GetCapabilities 操作需要以下的参数：

参数	是否必须	默认值
SERVICE	是	WCS
REQUEST=GetCapabilities	是	
ACCEPTVERSIONS		
SECTIONS		
UPDATESEQUENCE		
ACCEPTFORMATS		

以下是一个 WCS 使用 KVP 格式的 GetCapabilities 操作示例：

```
http://hostname:port/path?  
service=WCS&  
request=GetCapabilities
```

2.XML 格式请求

以下是一个 WCS 使用 XML 格式的 GetCapabilities 操作示例：

```
<?xml version="1.0" encoding="UTF-8"?>  
<GetCapabilities xmlns="http://www.opengis.net/wcs/1.1"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.opengis.net/wcs/1.1 ../wcsGetCapabilities.xsd"  
  service="WCS" />
```

3. 响应示例

以下是一个 WCS 的 GetCapabilities 操作的响应示例：

```
<?xml version="1.0" encoding="UTF-8"?>  
<wcs:Capabilities xmlns="http://www.opengis.net/wcs/1.1"  
  xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.opengis.net/wcs/1.1 ../wcsDescribeCoverage.xsd  
http://www.opengis.net/ows/1.1 ../ows/1.1.0/owsAll.xsd">  
  <ows:ServiceIdentification>  
    <ows:Title>Web Coverage Service</ows:Title>  
    <ows:Abstract>WCS</ows:Abstract>  
    <ows:ServiceType>WCS</ows:ServiceType>  
    <ows:AccessConstraints>NONE</ows:AccessConstraints>  
  </ows:ServiceIdentification>  
  <ows:OperationsMetadata>  
    <ows:Operation name="GetCapabilities">  
      ...  
    </ows:Operation>  
    <ows:Operation name="DescribeCoverage">  
      ...  
    </ows:Operation>  
    <ows:Operation name="GetCoverage">  
      ...  
    </ows:Operation>  
    <ows:Constraint name="PostEncoding">  
      <ows:AllowedValues>
```

```

        <ows:Value>XML</ows:Value>
      </ows:AllowedValues>
    </ows:Constraint>
  </ows:OperationsMetadata>
  <wcs:Contents>
    ...
  </wcs:Contents>
</wcs:Capabilities>

```

• DescribeCoverage 操作

1.KVP 格式请求

DescribeCoverage 操作需要以下的参数：

参数	是否必须	默认值
VERSION	是	1.1.2
SERVICE	是	WCS
REQUEST=DescribeCoverage	是	
IDENTIFIERS	是	

以下是一个 WCS 使用 KVP 格式的 DescribeCoverage 操作示例：

```

http://server_address/path/script?
service=WCS &
request=DescribeCoverage&
version=1.1.2 &
identifiers=Cov1,Cov2,Cov3

```

2.XML 格式请求

以下是一个 WCS 使用 XML 格式的 DescribeCoverage 操作示例：

```

<?xml version="1.0" encoding="UTF-8"?>
<DescribeCoverage xmlns="http://www.opengis.net/wcs/1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wcs/1.1 ../wcsDescribeCoverage.xsd"
  service="WCS" version="1.1.2">
  <Identifier>Cov1</Identifier>
  <Identifier>Cov2</Identifier>

```

```
<Identifier>Cov3</Identifier>
</DescribeCoverage>
```

3. 响应示例

以下是一个 WCS 的 DescribeCoverage 操作的响应示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<CoverageDescriptions xmlns="http://www.opengis.net/wcs/1.1"
  xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wcs/1.1 ../wcsDescribeCoverage.xsd
http://www.opengis.net/ows/1.1 ../ows/1.1.0/owsAll.xsd">
  <CoverageDescription>
    <ows:Title>TBD</ows:Title>
    <ows:Abstract>TBD</ows:Abstract>
    <Identifier>TBD</Identifier>
    <Domain>
      <SpatialDomain>
        <ows:BoundingBox>
          <ows:LowerCorner>-30.00 -30.00</ows:LowerCorner>
          <ows:UpperCorner>30.00 30.00</ows:UpperCorner>
        </ows:BoundingBox>
      </SpatialDomain>
    </Domain>
    <Range>
      <Field>
        <ows:Title>TBD</ows:Title>
        <ows:Abstract>TBD</ows:Abstract>
        <Identifier>TBD</Identifier>
        <Definition>
          <ows:AnyValue />
        </Definition>
        <InterpolationMethods>
          <InterpolationMethod>linear</InterpolationMethod>
          <Default>cubic</Default>
        </InterpolationMethods>
      </Field>
    </Range>
    <SupportedCRS>urn:ogc:def:crs:EPSG::XXXX
    </SupportedCRS>
    <SupportedCRS>urn:ogc:def:crs:EPSG::YYYY
    </SupportedCRS>
    <SupportedFormat>text/xml</SupportedFormat>
  </CoverageDescription>
</CoverageDescriptions>
```

• GetCoverage 操作

1.KVP 格式请求

GetCoverage 操作需要以下的参数：

参数	是否必须	默认值
VERSION	是	1.1.2
SERVICE	是	WCS
REQUEST=GetCoverage	是	
IDENTIFIER	是	
BOUNDINGBOX	是	
FORMAT	是	
TIMESEQUENCE		
RANGESUBSET		
STORE		
GRIDBASECRS		
GRIDTYPE		
GRIDCS		
GRIDORIGIN		
GRIDOFFSETS		

以下是一个 WCS 使用 KVP 格式的 GetCoverage 操作示例：

```
http://my.service.org/path/script?  
service=WCS&  
version=1.1.2&  
request=GetCoverage&  
identifier=Cov123&  
BoundingBox=-71,47,-66,51,urn:ogc:def:crs:OGC:2:84&  
format=image/netcdf
```

2.XML 格式请求

以下是一个 WCS 使用 XML 格式的 GetCoverage 操作示例：

```
<?xml version="1.0" encoding="UTF-8"?>  
<GetCoverage xmlns="http://www.opengis.net/wcs/1.1"  
  xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.opengis.net/wcs/1.1 ../wcsGetCoverage.xsd"
```

```

service= "WCS" version= "1.1.2">
<ows:Identifier>Cov123</ows:Identifier>
<DomainSubset>
  <ows:BoundingBox crs= "urn:ogc:def:crs:OGC:2:84">
    <ows:LowerCorner>-71 47</ows:LowerCorner>
    <ows:UpperCorner>-66 51</ows:UpperCorner>
  </ows:BoundingBox>
</DomainSubset>
<Output format= "image/netcdf" />
</GetCoverage>

```

3. 响应示例

以下是一个 WCS 的 GetCoverage 操作的响应示例：

```

<?xml version= "1.0" encoding= "UTF-8"?>
<Coverages xmlns= "http://www.opengis.net/wcs/1.1" xmlns:ows= "http://www.opengis.net/ows"
  xmlns:xlink= "http://www.w3.org/1999/xlink" xmlns:xsi= "http://www.w3.org/2001/XMLSchema-instance"
  xmlns:schemaLocation= "http://www.opengis.net/wcs/1.1 ../owsCoverages.xsd
  http://www.opengis.net/ows/1.1 ../ows/1.1.0/owsAll.xsd">
  <Coverage>
    <ows:Title>TBD</ows:Title>
    <ows:Abstract>Coverage created from GetCoverage operation
      request to a WCS</ows:Abstract>
    <Identifier>TBD</Identifier>
    <Reference xlink:href= "http://my.server.com/coverage/image.tiff"
      xlink:role= "urn:ogc:def:role:WCS:1.1:coverage" />
    <Reference xlink:href= "http://my.server.com/coverage/metadata.xml"
      xlink:role= "urn:ogc:def:role:WCS:1.1:metadata" />
  </Coverage>
</Coverages>

```

• ArcGIS Server 对 WCS 的支持

ArcGIS Server 10 中支持的 WCS 版本为 1.0.0¹。在 ArcGIS Server 中，只需简单地勾选的 Capabilities 选项卡中可以选择支持 WCS，如图 22。

¹ 并非最新版本，因此下面例子的参数和前面叙述的 1.1.2 版本不同。

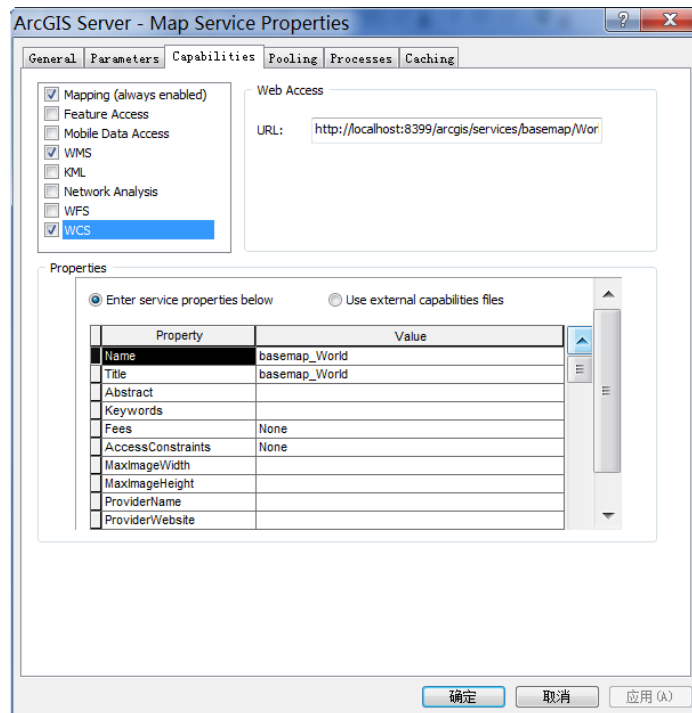
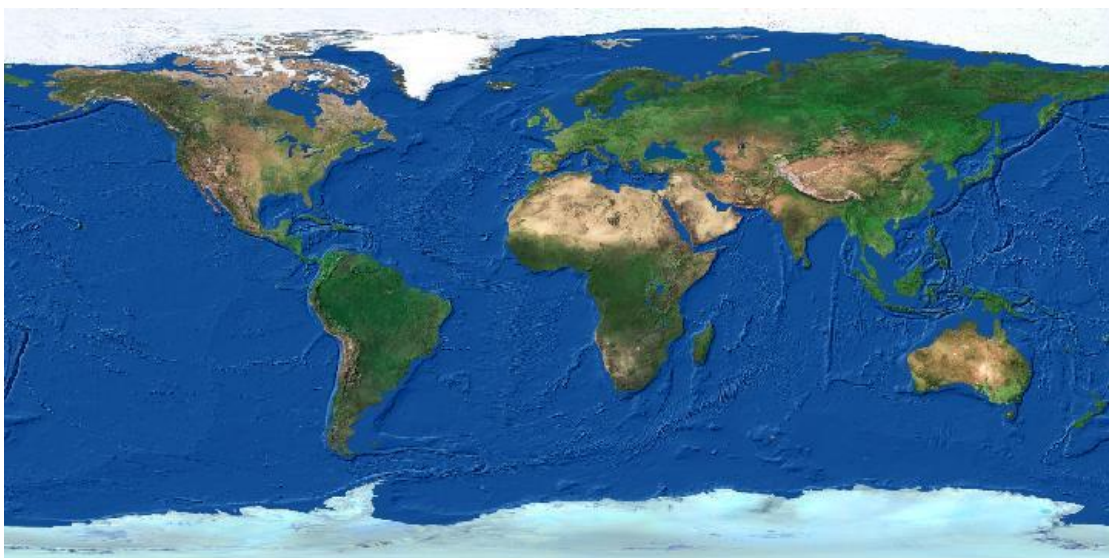


图 22 ArcGIS Server 发布 WCS 服务

现在我们通过浏览器直接发送请求获取栅格数据 返回格式选择 GeoTIFF：

```
http://localhost:8399/arcgis/services/basemap/World/MapServer/WCSTServer?
version=1.0.0&request=GetCoverage&service=WCS&
crs=EPSG:4326&bbox=-180,-90,180,90&
width=720&height=360&
format=geotiff&
coverage=1
```

这个请求将会返回如下的结果：



X. WMTS-切片地图 Web 服务

- 概述

WMTS (OpenGIS® Web Map Tile Service) 当前版本是 1.0.0。WMTS 标准定义了一些操作，这些操作允许用户访问切片地图。WMTS 可能是 OGC 首个支持 RESTful 访问的服务标准。

- WMTS 的原理和操作

WMTS 的切片坐标系统及其组织方式可参考图 23：

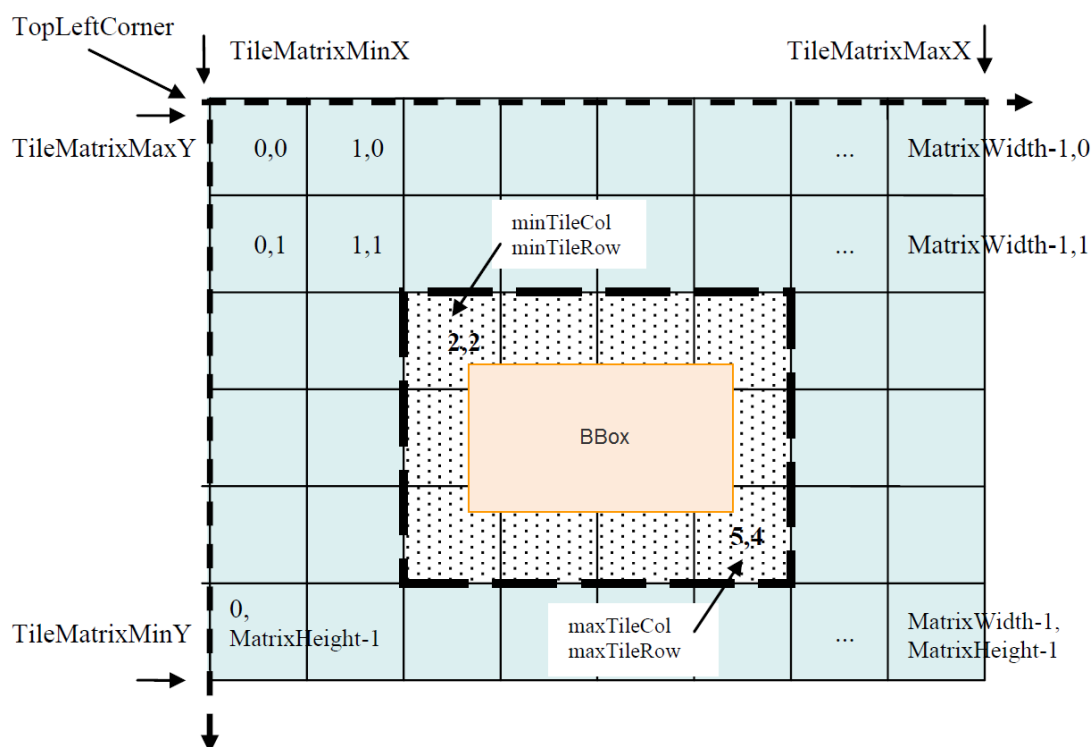


图 23 WMTS 的坐标系统和组织方式

如果知道一个请求的边界范围，可以根据以下的算法获取切片的序号：

```
// 计算切片序号范围
epsilon = 1e-61 //一个不影响坐标精度的小数值
tileMinCol = floor((bBoxMinX - tileMatrixMinX) / tileSpanX + epsilon)
tileMaxCol = floor((bBoxMaxX - tileMatrixMinX) / tileSpanX - epsilon)
tileMinRow = floor((tileMatrixMaxY - bBoxMaxY) / tileSpanY + epsilon)
tileMaxRow = floor((tileMatrixMaxY - bBoxMinY) / tileSpanY - epsilon)
// 避免超出范围
if (tileMinCol < 0) tileMinCol = 0
if (tileMaxCol >= matrixWidth) tileMaxCol = matrixWidth-1
if (tileMinRow < 0) tileMinRow = 0
if (tileMaxRow >= matrixHeight) tileMaxRow = matrixHeight-1
```

在一个 WMTS 服务中包括以下 3 个操作：

1. GetCapabilities (获取服务的元信息)
2. GetTile (获取切片)
3. GetFeatureInfo (可选，获取点选的要害信息)

可以看到这些操作和 WMS 的操作非常的异曲同工。

• GetCapabilities 操作

1.KVP 格式请求

GetCapabilities 操作需要以下的参数：

参数	是否必须	默认值
SERVICE	是	WMTS
REQUEST=GetCapabilities	是	

以下是一个 WMTS 使用 KVP 格式的 GetCapabilities 操作示例：

```
http://www.maps.bob/maps.cgi?
service=WMTS&
request=GetCapabilities
```

¹ 标准文档如是说，但我觉得这个地方应该取一负值。

2.SOAP 格式请求

以下是一个 WMTS 使用 SOAP 格式的 GetCapabilities 操作示例：

```
<?xml version= "1.0" encoding= "UTF-8"?>
<soap:Envelope xmlns:soap= "http://www.w3.org/2003/05/soap-envelope">
  <soap:Body>
    <GetCapabilities service= "WMTS"
      xmlns= "http://www.opengis.net/ows/1.1">
      <AcceptVersions>
        <Version>1.0.0</Version>
      </AcceptVersions>
      <AcceptFormats>
        <OutputFormat>application/xml</OutputFormat>
      </AcceptFormats>
    </GetCapabilities>
  </soap:Body>
</soap:Envelope>
```

3.RESTful 格式请求

GetCapabilities 返回的就是服务的元信息 (ServiceMetadata), 因此 RESTful 格式请求就是一个指向元信息的资源地址 , 下面是一个 WMTS 使用 RESTful 格式的 GetCapabilities 操作示例：

```
http://www.maps.bob/1.0.0/WMTSCapabilities.xml
```

• GetTile 操作

1.KVP 格式请求

GetTile 操作需要以下的参数：

参数	是否必须	默认值
SERVICE	是	WMTS
REQUEST=GetTile	是	

VERSION	是	1.0.0
LAYER	是	
STYLE	是	
FORMAT	是	
TILEMATRIXSET	是	
TILEMATRIX	是	
TILEROW	是	
TILECOL	是	
Sample dimensions 参数		

以下是一个 WMTS 使用 KVP 格式的 GetTile 操作示例：

```
http://www.maps.bob/maps.cgi?
service=WMTS&
request=GetTile&
version=1.0.0&
layer=etopo2&
style=default&
format=image/png&
TileMatrixSet=WholeWorld_CRS_84&
TileMatrix=10m&
TileRow=1&
TileCol=3
```

2.SOAP 格式请求

以下是一个 WMTS 使用 SOAP 格式的 GetTile 操作示例：

```
<?xml version= "1.0" encoding= "UTF-8"?>
<soap:Envelope xmlns:soap= "http://www.w3.org/2003/05/soap-envelope">
  <soap:Body>
    <GetTile service= "WMTS" version= "1.0.0"
      xmlns= "http://www.opengis.net/wmts/1.0">
      <Layer>etopo2</Layer>
      <Style>default</Style>
      <Format>image/png</Format>
      <TileMatrixSet> WholeWorld_CRS_84</TileMatrixSet>
      <TileMatrix> 10m</TileMatrix>
      <TileRow>1</TileRow>
      <TileCol>3</TileCol>
    </GetTile>
  </soap:Body>
</soap:Envelope>
```

3.RESTful 格式请求

以下是一个 WMTS 使用 RESTful 格式的 GetTile 操作示例：

```
http://www.maps.bob/etopo2/default/WholeWorld_CRS_84/10m/1/3.png
```

- **GetFeatureInfo 操作**

1.KVP 格式请求

GetFeatureInfo 操作需要以下的参数：

参数	是否必须	默认值
SERVICE	是	WMTS
REQUEST=GetFeatureInfo	是	
VERSION	是	1.0.0
GetTile 请求参数		
I	是	
J	是	
INFOFORMAT	是	

以下是一个 WMTS 使用 KVP 格式的 GetFeatureInfo 操作示例：

```
http://www.maps.bob/maps.cgi?  
service=WMTS&  
request=GetFeatureInfo&  
version=1.0.0&  
layer=coastlines&  
style=default&  
format=image/png&  
TileMatrixSet=WholeWorld_CRS_84&TileMatrix=10m&TileRow=1&TileCol=3&  
J=86&I=132&  
InfoFormat=application/gml+xml; version=3.1
```

2.SOAP 格式请求

以下是一个 WMTS 使用 SOAP 格式的 GetFeatureInfo 操作示例：

```

<?xml version= "1.0" encoding= "UTF-8"?>
<soap:Envelope xmlns:soap= "http://www.w3.org/2003/05/soap-envelope">
  <soap:Body>
    <GetFeatureInfo service= "WMTS" version= "1.0.0"
      xmlns= "http://www.opengis.net/wmts/1.0">
      <GetTile service= "WMTS" version= "1.0.0"
        xmlns= "http://www.opengis.net/wmts/1.0">
        <Layer>etopo2</Layer>
        <Style>default</Style>
        <Format>image/png</Format>
        <TileMatrixSet> WholeWorld_CRS_84</TileMatrixSet>
        <TileMatrix>10m</TileMatrix>
        <TileRow>1</TileRow>
        <TileCol>3</TileCol>
      </GetTile>
      <J>86</J>
      <I>132</I>
      <InfoFormat>application/gml+xml; version=3.1</InfoFormat>
    </GetFeatureInfo>
  </soap:Body>
</soap:Envelope>

```

3.RESTful 格式请求

以下是一个 WMTS 使用 RESTful 格式的 GetFeatureInfo 操作示例：

```
http://www.maps.bob/etopo2/ default/WholeWorld_CRS_84/10m/1/3/86/132.xml
```


XI. 附录：ArcGIS 支持的 OGC 标准列表

参考：《[Esri® Supported Open Geospatial Consortium, Inc.®, and ISO/TC](#)

[211 Standards](#)》

OGC 标准	可提供	可使用
Web Mapping Service (WMS) 1.1.1	ArcIMS 9.2, 9.3, 10 ArcGIS Server 9.2, 9.3, 10	GIS Portal Toolkit 3.1, 9.3 ArcGIS Desktop 9.2, 9.3, 10 ArcGIS Explorer ArcGlobe 9.2, 9.3, 10 ArcGIS .NET and Java ADF 9.2, 9.3, 10 ArcGIS Server Geoportal Extension 9.3.1, 10
Web Mapping Service (WMS) 1.3	ArcIMS 9.2, 9.3, 10 ArcGIS Server 9.2, 9.3, 10	GIS Portal Toolkit 3.1, 9.3 ArcGIS Desktop 9.2, 9.3, 10 ArcGIS .NET and Java ADF 9.2, 9.3, 10 ArcGIS Server Geoportal Extension 9.3.1, 10
Styled Layer Descriptor (SLD) 1.0	ArcIMS 9.2, 9.3, 10 ArcGIS Server 9.3, 10	GIS Portal Toolkit 3.1, 9.3 ArcGIS Desktop 9.3, 10
Web Feature Service (WFS) 1.0	ArcIMS 9.2, 9.3, 10 ArcGIS Server 9.3, 10	ArcGIS Desktop 9.2, 9.3, 9.3.1, 10 ArcGIS Data Interoperability 9.2, 9.3, 9.3.1, 10 GIS Portal Toolkit 9.3 ArcGIS Server Geoportal Extension 9.3.1, 10
Web Feature Service (WFS) 1.1	ArcIMS 9.3, 10 ArcGIS Server 9.3, 10	ArcGIS Desktop 9.3, 10 ArcGIS Data Interoperability 9.3, 10 GIS Portal Toolkit 3.1, 9.3 ArcGIS Server Geoportal Extension 9.3.1, 10
Web Feature Service (WFS-T) 1.1	ArcGIS Server 9.3, 10	
Filter Encoding 1.0	ArcIMS 9.2, 9.3, 10	ArcGIS Desktop 9.2, 9.3, 10 ArcGIS Data Interoperability 9.2, 9.3, 10 GIS Portal Toolkit 3.1, 9.3 ArcGIS Server Geoportal Extension

		9.3.1, 10
Filter Encoding 1.1	ArcGIS Server 9.3, 10	ArcGIS Desktop 9.3, 10 ArcGIS Data Interoperability 9.3, 10 GIS Portal Toolkit 3.1, 9.3 ArcGIS Server Geoportal Extension 9.3.1, 10
Web Coverage Service (WCS) 1.0, 1.1, 1.1.1	ArcGIS Server 9.3, 10	GIS Portal Toolkit 3.1, 9.3 ArcGIS Desktop 9.3, 10 ArcGIS Server Geoportal Extension 9.3.1, 10
Catalog Services 1.0—Z39.50	ArcIMS 9.2, 9.3, 10 GIS Portal Toolkit 3.1, 9.3 ArcGIS Server Geoportal Extension 9.3.1, 10	GIS Portal Toolkit 3.1, 9.3 ArcGIS Server Geoportal Extension 9.3.1, 10
Web Catalog Service (CSW) 2.0.1	ArcIMS 9.2, 9.3, 10 GIS Portal Toolkit 3.1, 9.3	GIS Portal Toolkit 3.1, 9.3 GIS Portal Toolbar for ArcGIS 9.3 CSW Client for ArcGIS 9.3.1, 10 ArcGIS Server Geoportal Extension 9.3.1, 10
Web Catalog Service (CSW) 2.0.2	ArcIMS 9.3, 10 GIS Portal Toolkit 3.1, 9.3 ArcGIS Server Geoportal Extension 9.3.1, 10	GIS Portal Toolkit 3.1, 9.3 GIS Portal Toolbar for ArcGIS 9.3 CSW Client for ArcGIS 9.3.1, 10
Simple Features (SF) 1.1	ArcSDE ArcGIS	ArcGIS
Geography Markup Language (GML) 2.x	ArcIMS 9.3 WFS Connector ArcIMS 10 WFS Connector	ArcGIS Data Interoperability 9.3, 10
Geography Markup Language (GML) 3.1.x	ArcGIS Data Interoperability 9.2, 9.3, 10	ArcGIS Data Interoperability 9.2, 9.3, 10
OGC KML 2.2	ArcGIS Server 9.3, 10 ArcGIS Desktop 9.3, 10 GIS Portal Toolkit 9.3 ArcGIS Server Geoportal Extension 9.3.1, 10	ArcGIS Explorer ArcGlobe 9.2, 9.3, 10 ArcGIS Data Interoperability Extension 9.3, 10