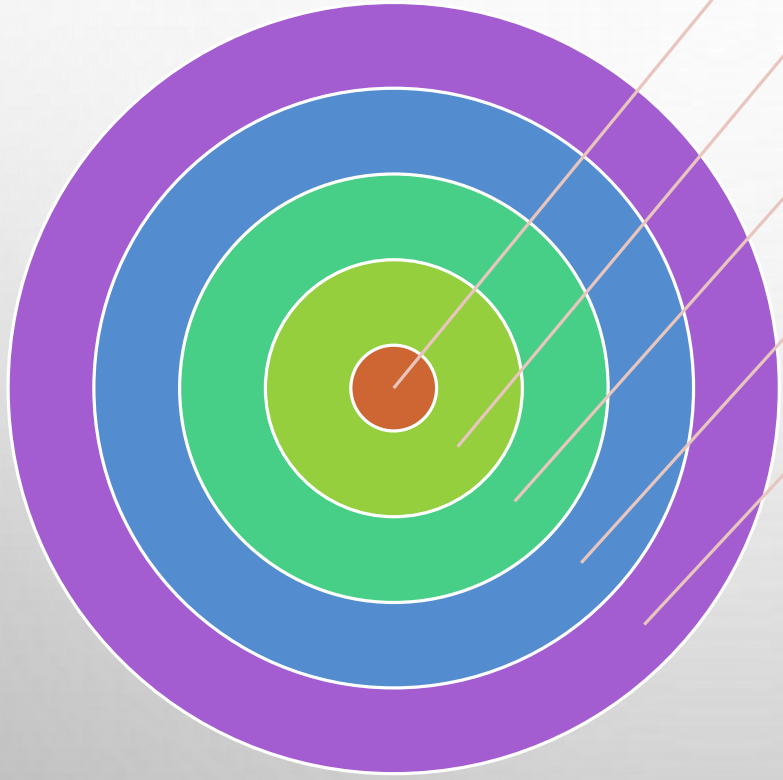
GDAL/OGR处理空间数据

虾神卢

OGR体系结构



Geometry 几何对象

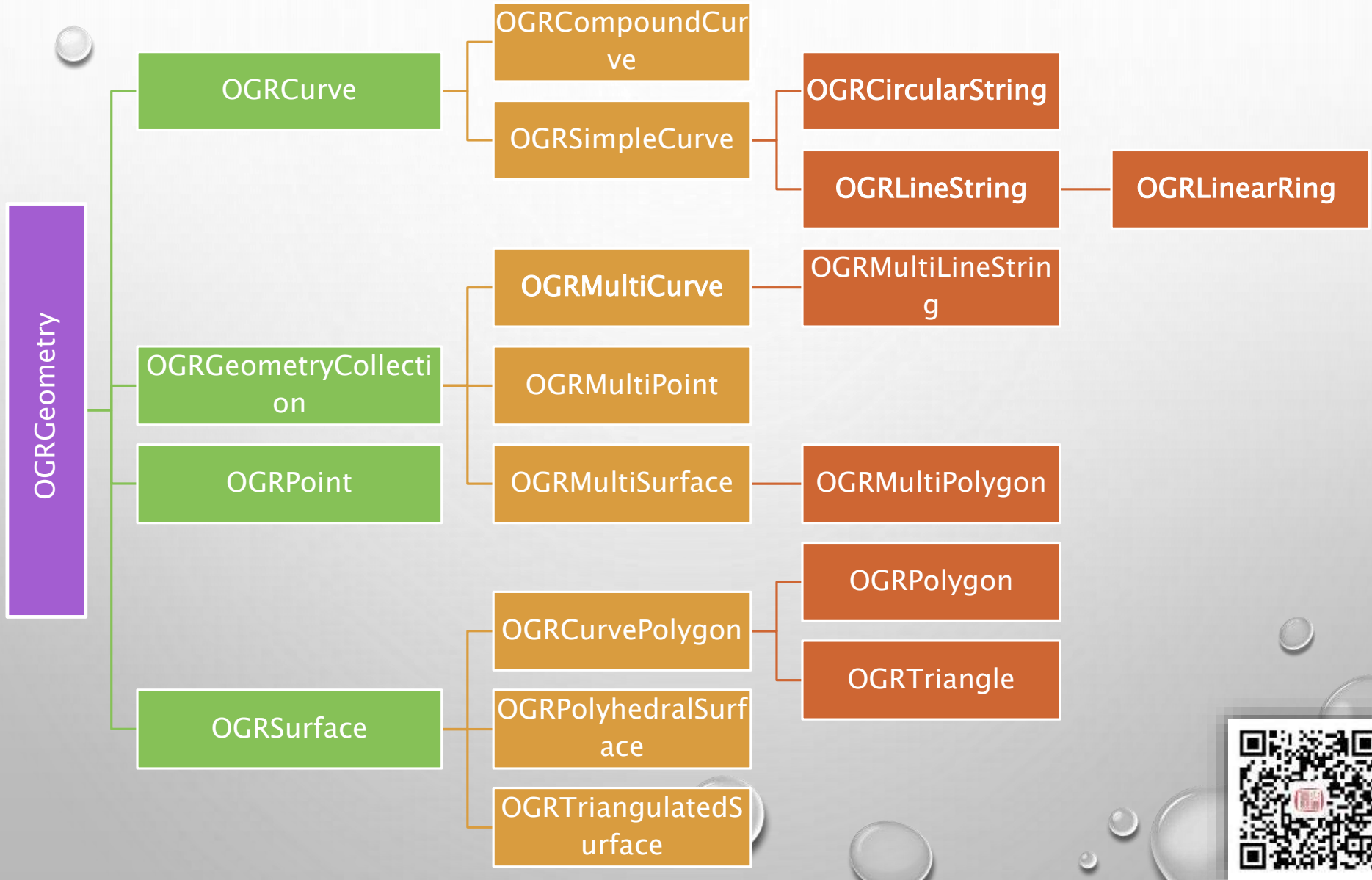
Feature 要素

Layer 图层

Data Source 数据源

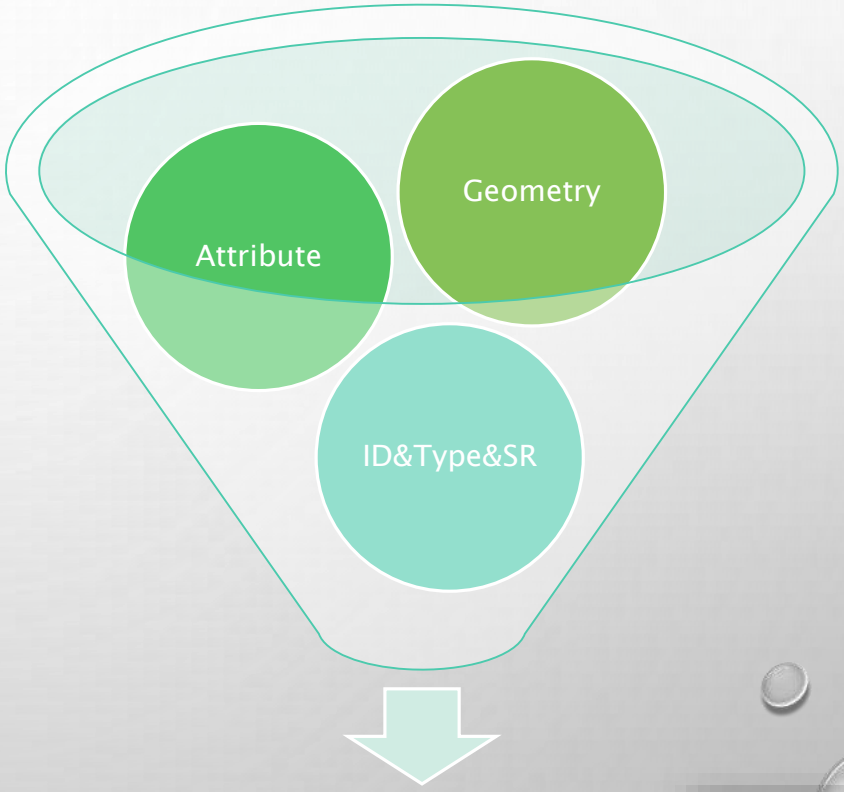
Drivers驱动格式

OGR的空间对象结构


OGRGeometry
OGRCurve
OGRCompoundCurve
OGRSimpleCurve
OGRCircularString
OGRLineString
OGRLinearRing
OGRGeometryCollection
OGRMultiCurve
OGRMultiLineString
OGRMultiPoint
OGRPoint
OGRMultiSurface
OGRMultiPolygon
OGRSurface
OGRCurvePolygon
OGRPolygon
OGRTriangle
OGRPolyhedralSurface
OGRTriangulatedSurface

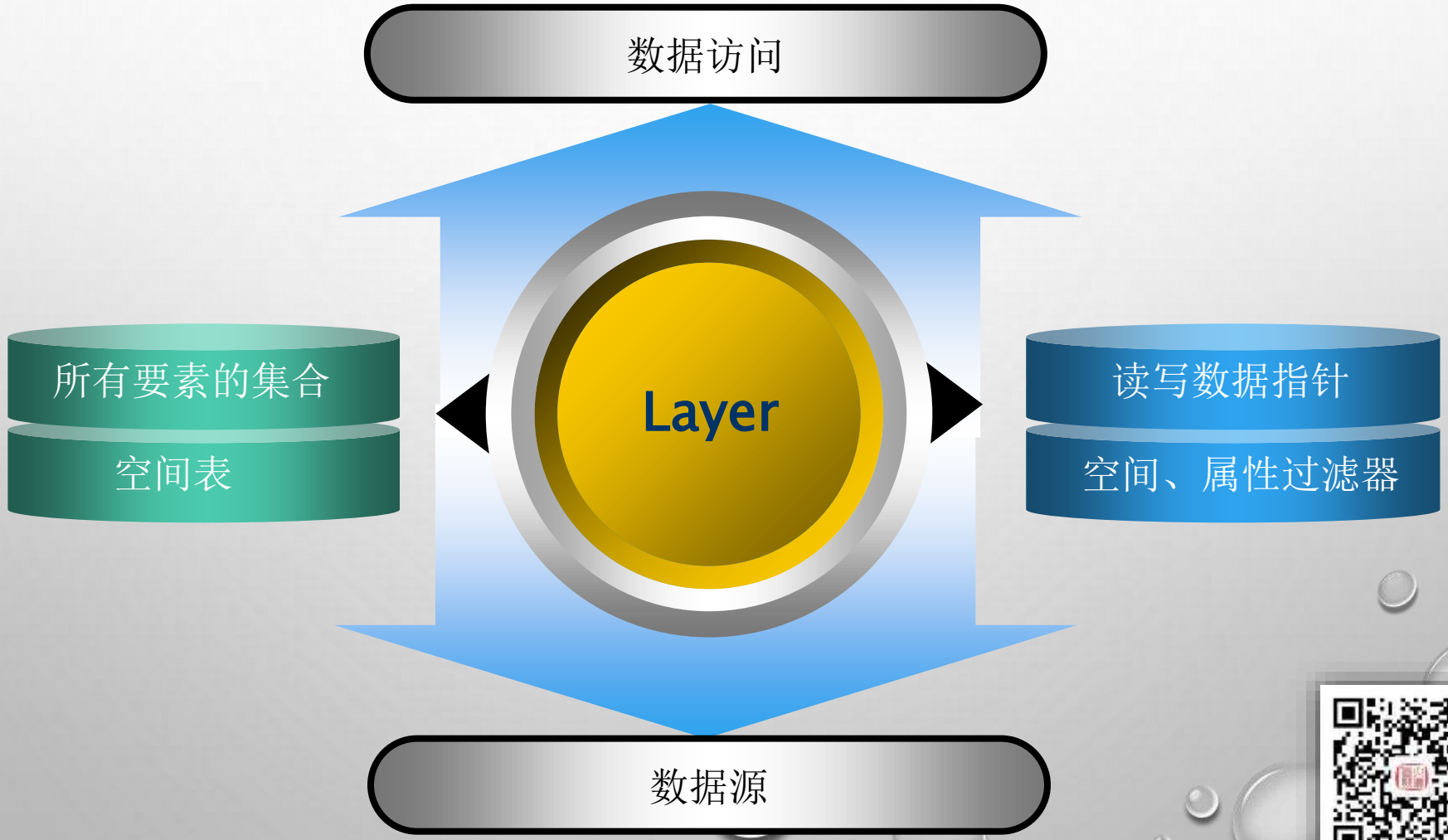
FEATRUE/FEATURE DEFINITION

- 要素及要素定义表示了一组完整的地理对象的描述，包括了如下部分：
  - 几何信息：GEOMETRY
  - 属性信息：ATTRIBUTE
  - 要素唯一编码 ： FID
  - 要素类型：TYPE
  - 空间参考：SPATIAL REFERENCE



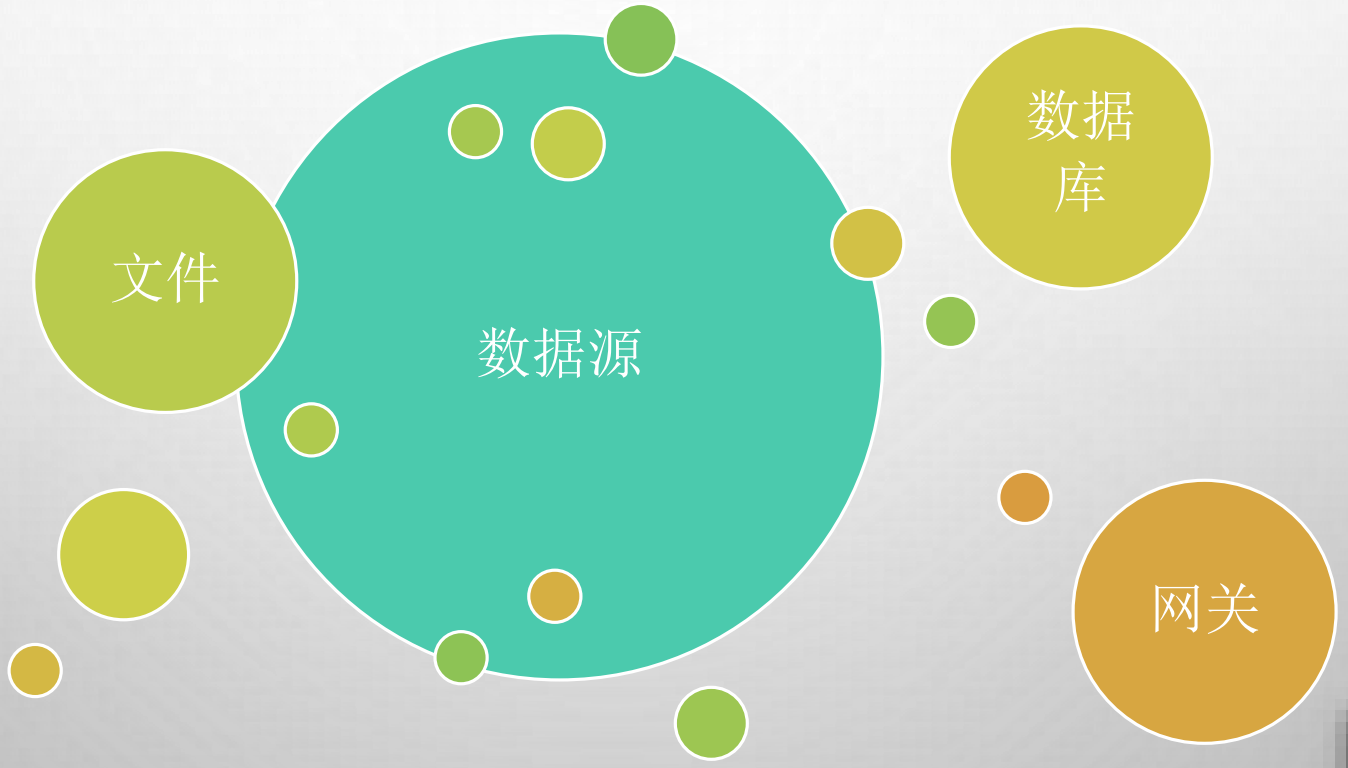
LAYER ： 图层

- 图层是OGR架构里面非常重要的一个概念，起到承上启下的作用。



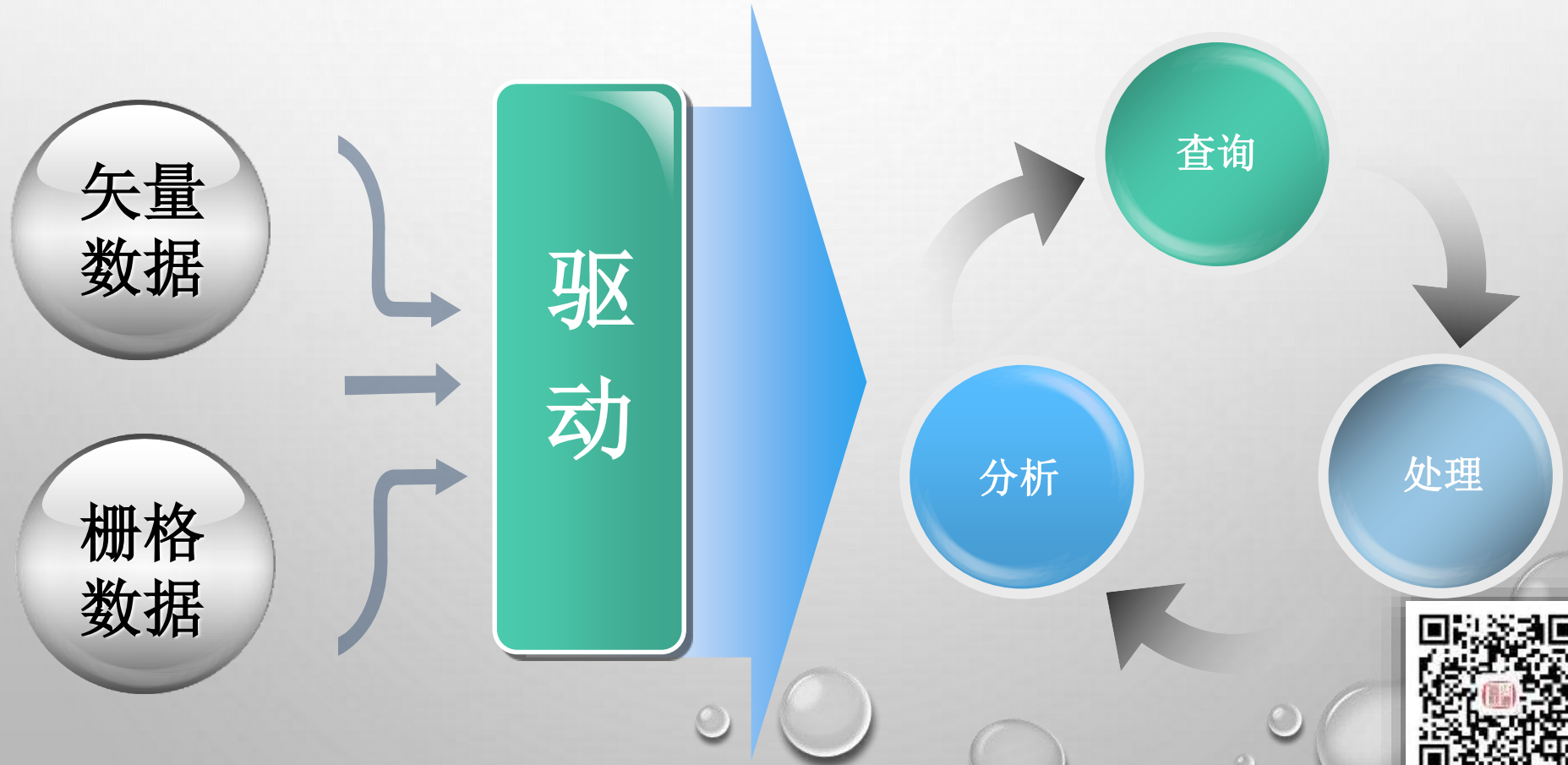
DATA SOURCE

- 数据源是一系列LAYER的组合，通常可以是一个文件（或者一组文件），数据库、网关等。

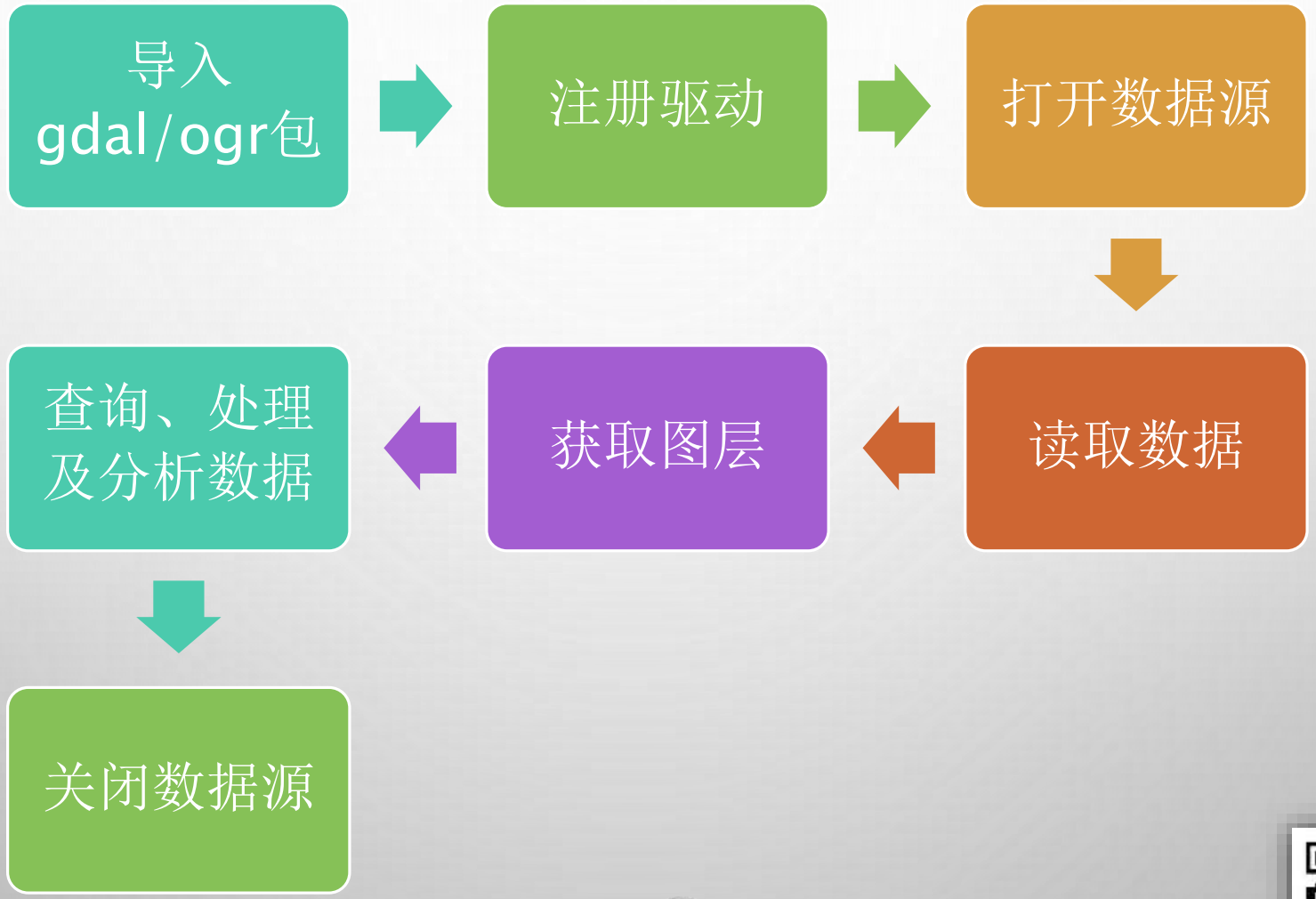


DRIVERS

- 驱动是OGR支持的一种文件格式的实现，在使用任何类型的文件之前，都需要对驱动进行注册，然后通过注册成功的驱动接口，来打开数据源。



GDAL/ORG PYTHON API使用流程

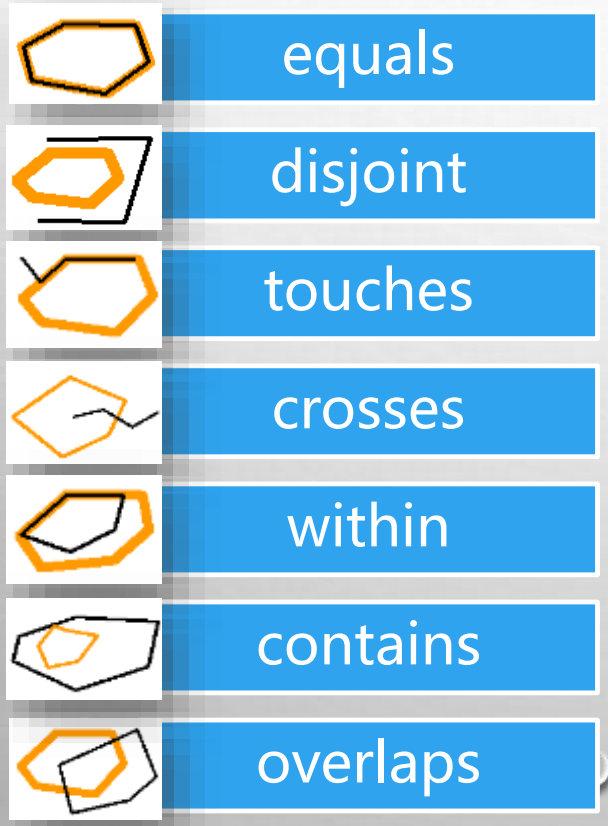


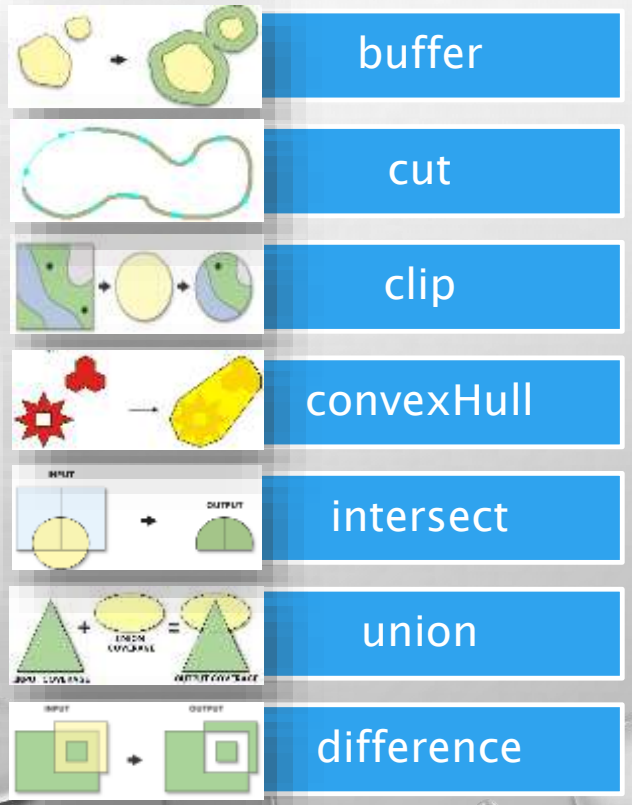
GEOJSON、WKT、WKB

- GEOJSON是一种对各种地理数据结构进行编码的格式，基于JAVASCRIPT对象表示法的地理空间信息数据交换格式。
  - GEOJSON对象可以表示几何、特征或者特征集合。其支持下面几何类型：点、线、面、多点、多线、多面和几何集合。
  - GEOJSON里的特征包含一个几何对象和其他属性，特征集合表示一系列特征。
- WKT和WKB是OPENGIS的说明书中定义了两个表述空间对象的标准方式
  - WKT（THE WELL-KNOWN TEXT）：文本模式描述信息
  - WKB（THE WELL-KNOWN BINARY）：二进制模式描述信息
  - 这两种形式都包括对象的类型信息和形成对象的坐标信息。

GDAL的空间对象操作

- GDAL/OGR的空间对象操作，主要依托于GEOS库
- GEOS（GEOMETRY ENGINE OPEN SOURCE：开源几何对象引擎），是一套C编写的，用于操作几何对象的开源算法包。
- 实现的也是OGC的那套标准的空间分析算法，包括但不限于如下：





GDAL/ORG的PYTHON调用

注册驱动和打开数据源

- 驱动常用方法：
  - OGR.GETDRIVERCOUNT()：获取当前系统可用驱动的数量
  - OGR.GETDRIVER(INDEX)：按照索引获取驱动
  - OGR.GETDRIVERBYNAME(DRIVERNAME)：按照名称获取驱动

- 打开数据源：
  - DRIVER.OPEN(数据源连接字符串，模式）
  - 连接字符串：
    - 文件模式用URL
    - 数据库模式用数据库连接字符串
  - 模式：
    - 0：只读模式
    - 1：更新模式

矢量数据描述

```
print("图层描述 :{0}".format(layer.GetDescription()))
print("图层范围 :{0}".format(layer.GetExtent()))
print("要素数量 :{0}".format(layer.GetFeatureCount()))
print("元数据描述 :{0}".format(layer.GetMetadata()))
print("空间参考 :{0}".format(layer.GetSpatialRef()))
```

```
图层描述 :北京_point
图层范围 :(115.37294, 117.36857, 39.41652, 41.07743)
要素数量 :128554
元数据描述 :{'DBF_DATE_LAST_UPDATE': '2009-05-19'}
空间参考 :GEOGCS["Geographic Coordinate System",
    DATUM["WGS84",
        SPHEROID["WGS84",6378137.0,298.257223560493]],
    PRIMEM["Greenwich",0.0],
    UNIT["degree",0.0174532925199433],
    AUTHORITY["EPSG","4326"]]
```

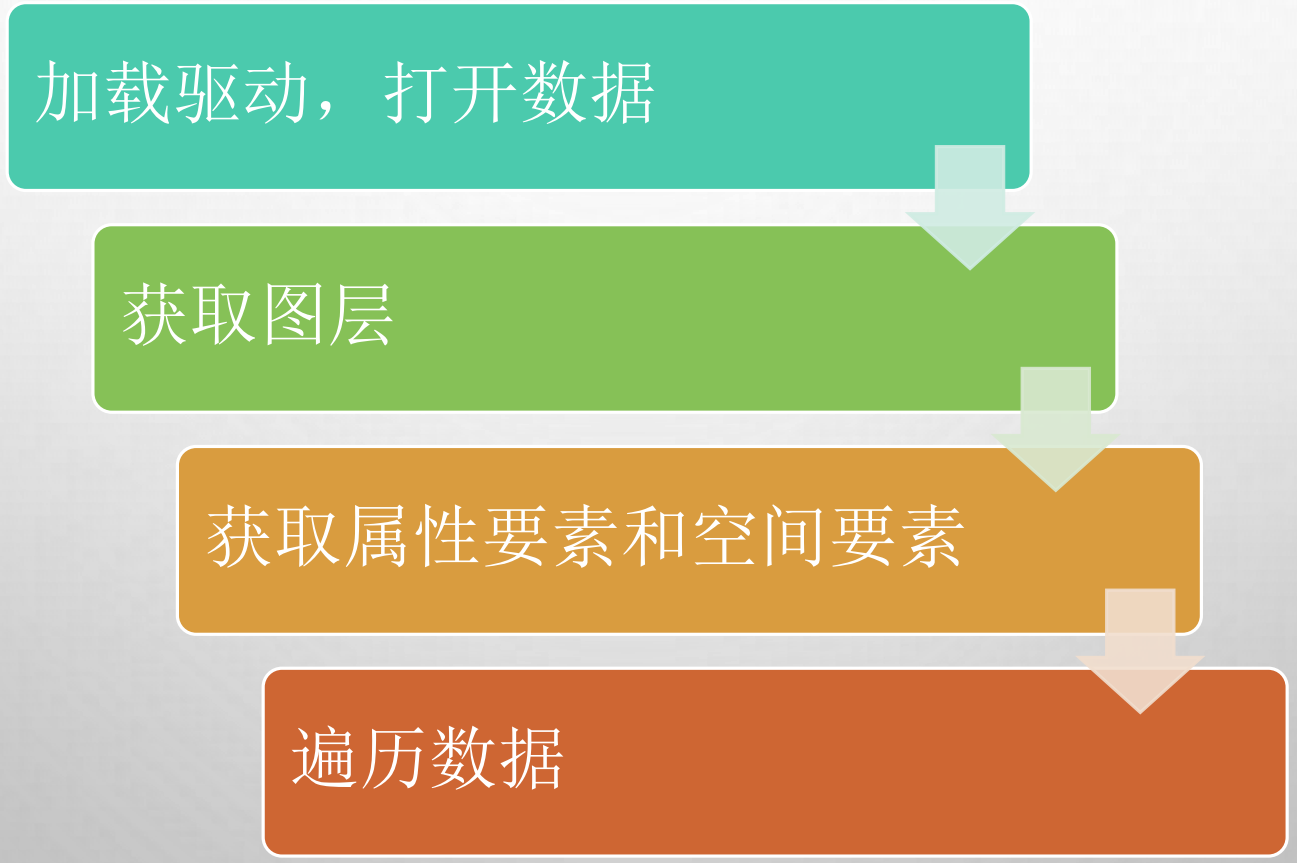
字段描述

```
layerDefinition = layer.GetLayerDefn()
for i in range(layerDefinition.GetFieldCount()):
    fieldName =  layerDefinition.GetFieldDefn(i).GetName()
    fieldTypeCode = layerDefinition.GetFieldDefn(i).GetType()
    fieldType = layerDefinition.GetFieldDefn(i).GetFieldTypeName(fieldT
    fieldWidth = layerDefinition.GetFieldDefn(i).GetWidth()
    GetPrecision = layerDefinition.GetFieldDefn(i).GetPrecision()
    print("字段名：{0}  字段类型：{1}  字段长度：{2}  \
          字段精度：{3}".format(fieldName,fieldTypeCode,
                              fieldType,fieldWidth,GetPrecision))
```

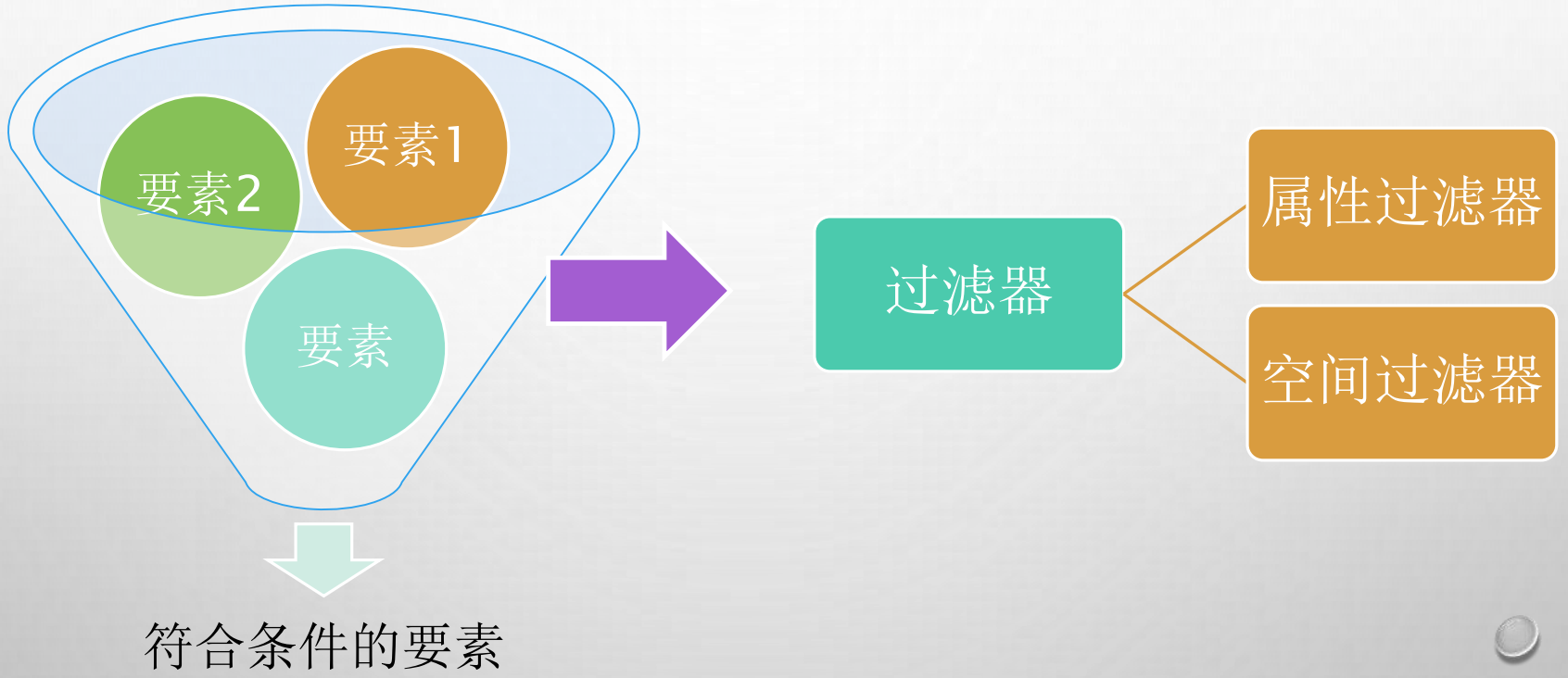
```
字段名：NAME 字段类型：4 字段长度：String 字段精度：65
字段名：LAYER 字段类型：4 字段长度：String 字段精度：21
字段名：MARINE 字段类型：4 字段长度：String 字段精度：1
字段名：RegionName 字段类型：4 字段长度：String 字段精度：6
字段名：DataLevel 字段类型：0 字段长度：Integer 字段精度：1
字段名：MP_TYPE 字段类型：4 字段长度：String 字段精度：6
字段名：Phone 字段类型：4 字段长度：String 字段精度：12
字段名：StreetDesc 字段类型：4 字段长度：String 字段精度：46
字段名：HighwayIdx 字段类型：0 字段长度：Integer 字段精度：2
字段名：ZipIdx 字段类型：0 字段长度：Integer 字段精度：1
字段名：City 字段类型：4 字段长度：String 字段精度：1
```

数据遍历

- OGR的数据遍历流程如下：

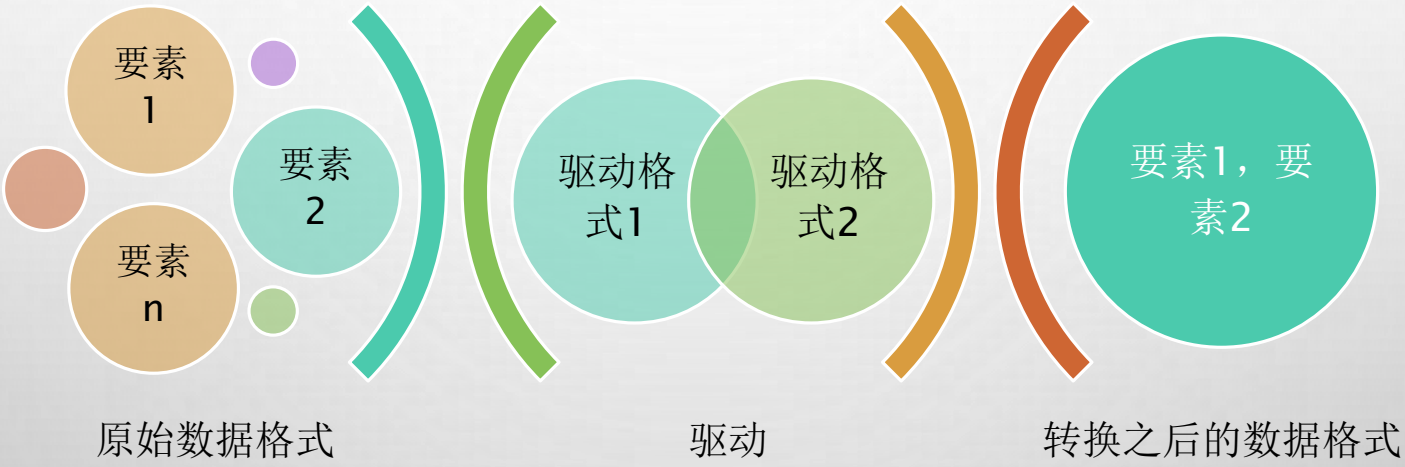


空间、属性过滤器



数据的转换与写入

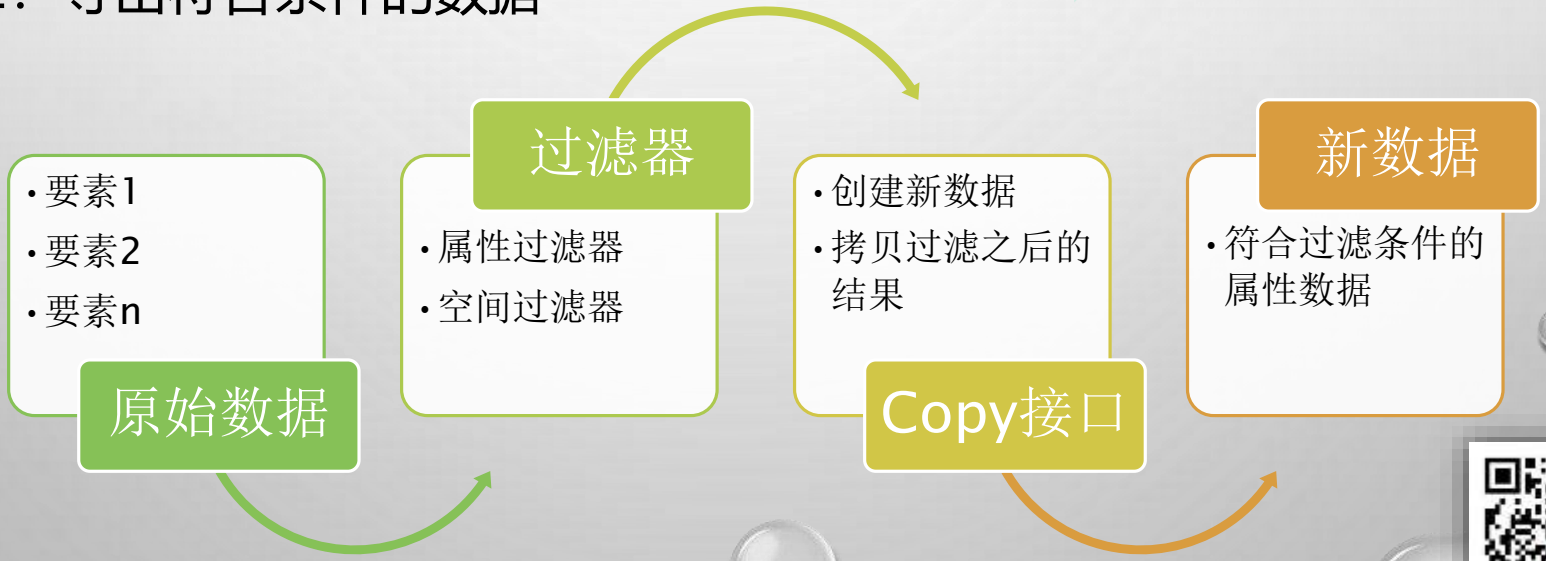
- 只要能够加载数据驱动，那么OGR能够在不同都是数据之间进行直接转换。



数据的COPY与导出

情况1：直接copy生成副本数据

情况2：导出符合条件的数据



数据分析与处理：包络多边形

数据分析与处理：获取中心点