基于 FME 的天地图·浙江融合数据质检程序的设计与应用

闻达

（浙江省测绘科学技术研究院，杭州 311100）

摘 要：基于FME技术，根据天地图·浙江融合设计要求，并结合天地图日常运维工作，建立一套严格的质检规则，用于检查天地图数据是否符合标准。对数据进行逻辑一致性、属性、空间关系等方面的质检，摆脱了传统人工质检机制，实现程序自动化、规范化质检。本文通过针对天地图数据中的POI和电子地图，分别设计了质检程序，解决其在数据融合过程中的问题。最终实现天地图融合数据质检程序的设计，并应用于日常数据质检工作中，提高了数据质量，提升了天地图服务质量。

关键字：FME；天地图；数据质检

1 引言

天地图是由国家、省、市（县）三级节点构成的国家地理信息公共服务平台。随着天地图应用的日益广泛与深入，采用“超链接”和“服务聚合”的方式已无法满足跨省（县）域、跨层级（国家、省、市）的地理实体与地名地址查询、专题信息挂接、综合统计分析等应用需求[1]。

随着浙江省天地图省、市（县）级节点数据融合工作的全面开展，天地图浙江的地理信息数据呈几何级增长，对于海量数据的快速更新成为浙江天地图发展的重要方向。为进一步提高天地图融合数据的更新效率，保证数据的一致性和现势性，走高效、集约的道路，使天地图更好地服务民生、社会，浙江省天地图本着统筹协调、联动更新、增量更新、快速服务的原则，于 2017 年开展了基础服务省级统一发布工作，由省级统一发布 7-20 级标准基础服务，由市县采集大比例尺数据进行省市融合、联动更新，进一步提升天地图服务水平和能力。

省市融合作为天地图·浙江数据更新的重要手段，为提升数据的精度、现势性和准确性起到了巨大的作用。但是，由于市（县）每年提交的融合成果数据量大、涉及范围广、提交时间集中，因此，需要耗费作业人员大量的时间和精力进行数据质量检查，以保证融合后省级库体的整体数据质量。现通过对天地图·浙江数据质检规则梳理，设计一套针对 POI 和电子地图的质检程序，对各市（县）融合提交的数据进行质检，保证数据的质量和精度，提高天地图数据的质检正确性，和质检的效率，为天地图·浙江提供更优质的数据服务。

2 数据质检概述

2.1 质检内容

天地图·浙江的融合数据由矢量电子地图、兴趣点 POI、地名地址和影像组成。其中，电子地图和 POI 作为天地图最重要的数据，复杂程度也最高，在天地图·浙江应用中承担着显示、检索等作用。因此，在质检程序设计时，主要实现针对电子地图和 POI 的质检。

2.2 质检规则

天地图融合的数据是从不同数据源中选取表达准确、现势性好、精度高、内容全的要素进行合并，并对合并后的结果进行几何拓扑、空间关系与逻辑一致性处理，使更新或融合后的地理信息数据在现势性、准确性、丰富性等方面达到最优[2]。

为了满足融合数据精度要求、几何表达精细度要求、现势性要求、属性取值要求、空间关系与逻辑一致性要求等，梳理了 POI 和电子地图的具体的质检规则，如下表 1、表 2 所示。

表 1 POI 质检规则

质量元素	质量子元素	检查项	具体要求
空间参考系	大地基准	坐标系统	CGCS2000
逻辑一致性	格式一致性	数据格式	为**.gdb 格式
		数据文件	数据文件无缺失、多余、数据无法读出等问题
数据完整性	数据一致性	要素遗漏	检查要素遗漏的个数及位置偏差
属性精度	属性正确性	属性值	检查属性值中不能含有异常字符
			检查分类代码的正确性
			检查名称、简称正确性
			检查显示比例、符号化字段的正确性
			检查更新时间、更新类型的正确性
			检查类型正确性
			检查电话格式正确性
		显示比例	检查比例尺赋值正确性，图面负载能相互不压盖，各级别平滑过渡
		全半角转换检查	确认名称、地址中全半角转换是否处理
逻辑一致性	概念一致性	属性项	字段中至少含有名称、简称、分类代码、显示比例、符号化字段、更新参考数据源、更新时间、更新类型且属性值不能为空
	拓扑一致性	各图层相互关系	是否存在 POI 不合理落水中、落道路上、落轨道上中的情况

表 2 电子地图质检规则

适用范围	质量元素	质量子元素	检查项	具体要求
所有图层	空间参考系	大地基准	坐标系统	CGCS2000
	表征质量	地理表达	要素关系	数据整体拓扑检查
	逻辑一致性	概念一致性	属性项	字段中至少含有唯一码、分类代码、显示比例、更新参考数据源、更新时间、更新状态、显示字段且属性值不能为空格和 NULL
	属性精度	属性正确性	属性值	各图层 FCODE、FSCALE 等属性赋值正确
	表征质量	地理表达	要素关系	各图层之间压盖关系是否正确
	数据完整性	数据一致性	要素遗漏	检查要素遗漏的个数
路网线	属性精度	属性正确性	属性必填项	道路名称采用全角符号

			检查	各级别显示时，需保证公路和城市道路，路网的连通性；不允许存在孤立道路；匝道处理检查，总体采用就低原则；公路与公路共线检查，公路与城市道路共线检查。
水系面	表征质量	几何表达	连通性	水系（面、线）及水系面与水系线间不允许断开，保持连通性
水系线	表征质量	几何表达	几何形状	有名称的水系采集水系注记线，水系注记线不超出与之对应的水系面；
		地理表达	要素关系	河流遇输水渡槽、溢洪道等可见附属设施应贯通；流遇涵洞需根据图面合理性选择贯通。
	属性精度	属性正确性	属性必填项	水系注记线各字段属性与对应水系面的字段属性值须保持一致；单线水系数据与对应水系面判重检查
辅助层	表征质量	几何表达	几何形状	检查辅助层是否从道路层截取，形状且属性与截取道路要完全一致
铁路线	表征质量	几何表达	几何形状	检查铁路的连续性
居民地面	表征质量	地理表达	要素关系	检查露天体育场有无绘出跑道且与绿地面关系是否正确
境界线	表征质量	几何表达	几何形状	检查境界线形状是否正确
政区面	表征质量	几何表达	几何形状	检查政区面形状是否正确
		地理表达	要素关系	检查政区面与政区面关系是否正确
居民地点	属性精度	属性正确性	属性项	检查同名行政村是否赋值正确；居委会、村委会填写是否统一

2.3 质检流程

融合后的 POI 和电子地图数据，在经过程序质检后，会针对每一个质检项，生成一个质检报告，质检报告中描述了错误数据的项目编码，详细的错误信息以及对应的质检项。作业人员根据质检报告，通过编码可定位到原始数据，进行数据整改。最终通过质检后，融合到省级节点中。质检流程如下图 1 所示。

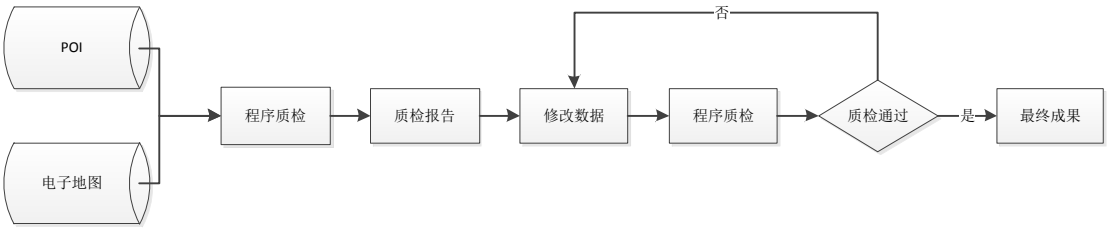

图 1 质检流程

3 基于 FME 的数据质检程序设计

3.1 FME 概述

FME(Feature Manipulate Engineering)是一个支持多元数据的提取、转换、加载、处理的强大工具，它是完整的空间ETL 解决方案。用户也可以根据自身的实际需求建立具有不同功能和应用的GIS 数据处理、转换的平台，从而大大提高在GIS数据处理过程中的工作效率，深度发掘空间地理信息数据的应用[3]。

3.2 检查方法及关键技术

根据 POI 和电子地图质检规则，基于 FME 技术，在 FME Workbench 中构建质检模板，通过 FME 转换器制定各项规则的转换流程，同时结合 ArcGIS 的 ArcPy 模块包，对数据的属性以及空间关系等做系统性检查，最后将完成的质检模板，发布到 FME Server，利用 FME Server 将各项质检规则进行整合。最终，用户可根据需要自由组合质检模板，对数据进行质检。

3.3 POI 质检程序设计

POI 数据以天地图 •浙江 POI 为本底，市县 POI 为主要参考数据源。参考数据源的格式、坐标、属性以及空间位置关系等有一定的差异，需要对参考数据进行标准化处理。融合后的 POI 要素要处理好与相关地物的位置关系，保证与道路、水系、道路等要素的相关位置正确，属性信息填写正确，并保证数据的现势性。

POI 质检程序，统一、集中对融合后的 POI 做检查，梳理其可能存在的基础信息错误，空间关系错误以及属性信息错误，并对错误数据进行定位及提供相关的错误描述。

3.3.1 基础质检规则设计

POI 的基础质检规则，包括空间参考、逻辑一致性、数据完整性。空间参考主要包括坐标系统和高程系统，现只检查坐标系统，应采用 2000 国家大地坐标系；逻辑一致性包括数据格式和数据文件的正确性，数据文件应保证无缺失、多余、数据无法读出等问题；数据完整性则需要统计具体要素的个数以及位置的偏差。

（1）空间参考

在质检过程中，通过 FME 的 CoordinateSystemExtractor 转换器获取 POI 的坐标信息并判断其是否符合要求，具体质检过程如下图 1 所示，并在输出过程中建立质检项字段，将不符合要求的数据按照项目编码进行输出，如下图 2 所示。

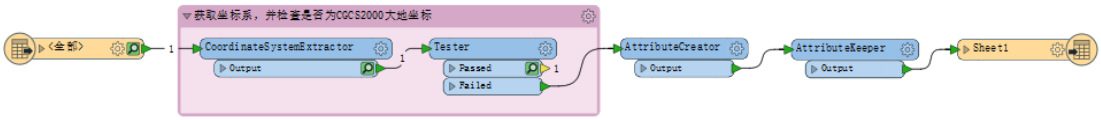

图 1 坐标系检查

图 2 创建质检项

（2）逻辑一致性

POI 的逻辑一致性指的是其存储格式应该为 GDB，且 GDB 文件中不包含数据集，有且仅有唯一的 POI 要素类，如下图 3 所示。利用 Pythoncallor 转换器中，编写 Python 代码，通过调用 ArcGIS 的 arcpy 模块编写其逻辑一致性规则，如下图 4 所示。

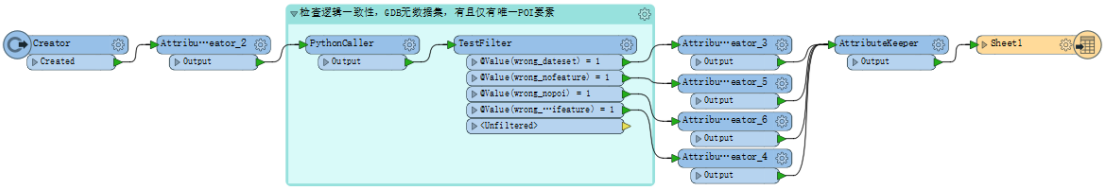

图 3 逻辑一致性检查

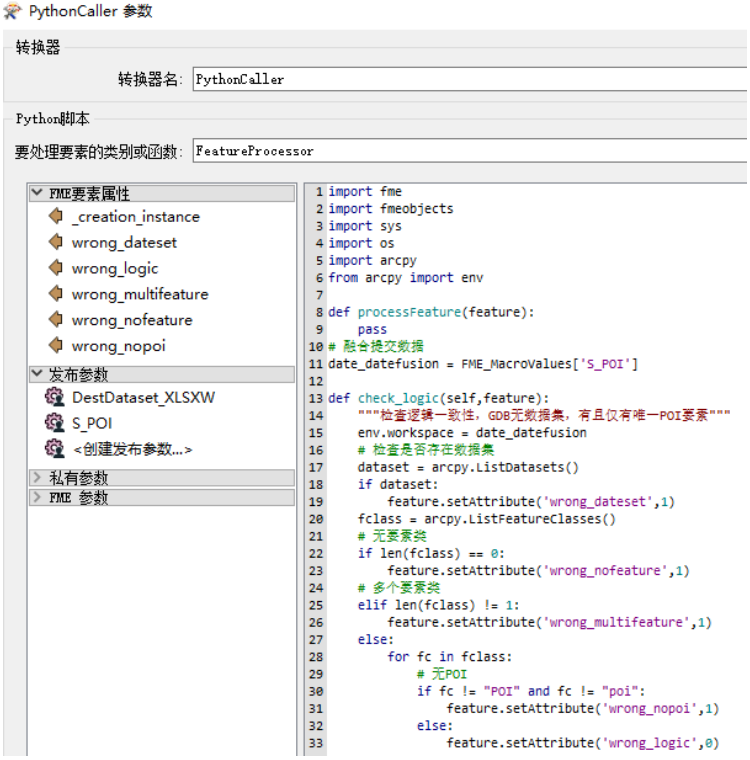

图 4 逻辑一致性检查规则

（3）数据完整性

POI 的数据完整性，指的是融合数据与原始数据进行比对，统计数据量和空间位置是否一致，如下图 5 所示。在 Pythoncallor 中编写统计数据量和空间分析的功能来实现，如下图 6 所示。

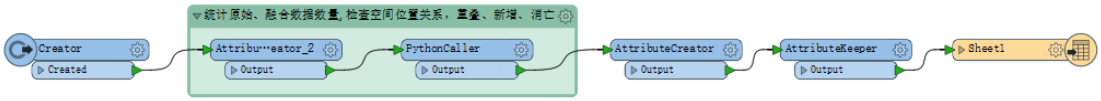

图 5 数据完整性检查

```
# 检查空间位置关系
print("Start checking spatial position...")
try:
    poi_original = date_original+"\\"+"POI"
    poi_datefusion = date_datefusion+"\\"+"POI"
    arcpy.MakeFeatureLayer_management(poi_original, "OriginalPOI_lyr")
    arcpy.SelectLayerByLocation_management("OriginalPOI_lyr", "intersect", poi_datefusion)
except Exception as e:
    print(e)
matchcount = int(arcpy.GetCount_management("OriginalPOI_lyr").getOutput(0))
print(matchcount)
feature.setAttribute('sam_num',str(matchcount))
```

图 6 检查空间位置是否一致

3. 3. 2 属性质检规则设计

属性质检规则主要是针对属性字段值的质检，包括 POI 的名称、分类代码、简称、比例等的正确性检查。其中较为重要的有以下几类：

（1）分类代码和符号化字段检查

POI 的分类代码不为空且不能超出天地图的分类代码范围，将分类代码表集成在 Pythoncallor 中做检查，且不能有异常符（空格、回车键），如下图 7 所示。符号化字段的检查，分为普通符号化字段和特殊符号化字段检查，如下图 8 所示。其中，与分类代码唯一对应的为普通符号化字段，如下图 9 所示，不能唯一对应的则为特殊符号化字段，需要再检查符号化字段与名称字段的对应关系，是否符合名称字段的关键字要求，如下图 10 所示。

图 7 分类代码检查

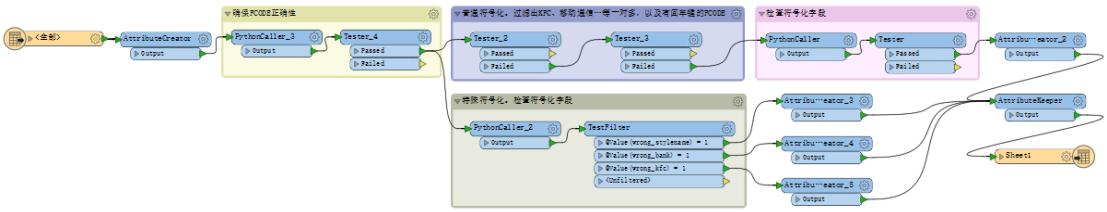

图 8 符号化字段检查

```
def check_stylename(self,feature):
    """符号化字段检查"""
    dicts = {"100201": "大楼大厦", "100202": "房产小区", "100301": "特定区域", "100302": "兴趣点通用", "110101": "机关单位",
             "110102": "机关单位", "110103": "机关单位", "110104": "机关单位", "110201": "机关单位", "110301": "公安交警",
             "110302": "机关单位", "110303": "机关单位", "110304": "公安交警", "110305": "机关单位", "110401": "机关单位",
             "110501": "机关单位", "110601": "机关单位", "120101": "宾馆酒店", "120102": "宾馆酒店", "120103": "宾馆酒店",
             "120104": "招待所", "120105": "招待所", "120201": "中餐馆", "120202": "西餐馆", "120203": "中餐馆",
             "130101": "公园", "130102": "动物园", "130103": "植物园", "130104": "场馆", "130201": "风景名胜", "130202": "公园",
             "130203": "风景名胜", "130204": "风景名胜", "130301": "烈士陵园", "130302": "风景名胜", "130303": "寺庙道观",
             "130304": "清真寺", "130305": "教堂", "130306": "其他兴趣点", "130307": "其他兴趣点", "130308": "其他兴趣点",
             "130309": "塔", "130310": "亭", "130311": "风景名胜", "140101": "场馆", "140102": "博物馆", "140103": "场馆",
             "140104": "场馆", "140105": "场馆", "140106": "场馆", "140107": "场馆", "140108": "场馆", "140109": "场馆",
             "140201": "场馆", "140202": "高尔夫球场", "140203": "企事业单位", "140204": "企事业单位", "140205": "场馆",
```

图 9 部分普通符号化字段与分类代码对应关系

```python
def check_special_stylename(self,feature):
    """检查特殊符号化字段对应的名称，包括银行、地铁、通信等"""
    # 特殊的银行、通信字段
    bank = ["中国人民银行", "工商银行", "广发银行", "光大银行", "华夏银行", "建设银行", "交通银行",
            "民生银行", "农业银行", "浦发银行", "深发银行", "兴业银行", "招商银行", "中国银行", "中信银行"]
    otherbank = ["其他银行"]
    specialbank = {"农业银行": "中国农业发展银行", "浦发银行": "上海浦东发展银行", "中信银行": "中信实业银行", "深发银行": "深圳发展银行"
                   "中国银行": "中行", "中国人民银行": "人民银行", "招商银行": "招行", "工商银行": "工行", "广发银行": "广东发展银行",
                   "建设银行": "建行"}
    KFCMcD = ["KFC", "麦当劳", "中餐馆"]
    tel = ["移动", "联通", "电信", "通信"]
    # 检查银行符号化字段
    if feature.getAttribute('FCODE') == "160501":
        if feature.getAttribute('STYLENAME').encode("utf-8") not in bank and feature.getAttribute('STYLENAME').encode("utf-8") no
            feature.setAttribute('wrong_stylename',1)

    if feature.getAttribute('STYLENAME').encode("utf-8") in bank:
        if feature.getAttribute('STYLENAME').encode("utf-8") not in feature.getAttribute('NAME').encode("utf-8"):
            if ((feature.getAttribute('STYLENAME').encode("utf-8") in specialbank.keys() and specialbank[feature.getAttribute('ST
                feature.setAttribute('wrong_bank',1)

    # 检查通信符号化字段
    if feature.getAttribute('FCODE') == "160401":
        if feature.getAttribute('STYLENAME').encode("utf-8") not in tel:
            feature.setAttribute('wrong_stylename',1)

    # 检查KFC、McDonald等符号化字段
    if feature.getAttribute('FCODE') == "120204":
        if feature.getAttribute('STYLENAME').encode("utf-8") not in KFCMcD:
            feature.setAttribute('wrong_stylename',1)
    if feature.getAttribute('STYLENAME').encode("utf-8") in KFCMcD:
        if feature.getAttribute('STYLENAME').encode("utf-8") == "KFC" and ("肯德基" not in feature.getAttribute('NAME').encode("ut
            feature.setAttribute('wrong_kfc',1)
        if feature.getAttribute('STYLENAME').encode("utf-8") == "麦当劳" and ("麦当劳" not in feature.getAttribute('NAME').encode(
            feature.setAttribute('wrong_kfc',1)
```

图 10 特殊符号化字段处理

（2）显示比例检查

POI 的显示比例根据分类代码的权重，与分类代码一一对应（如图 12），且比例尺应该在 7~21 级之间，检查其是否超限以及是否有误，如下图 11 所示。

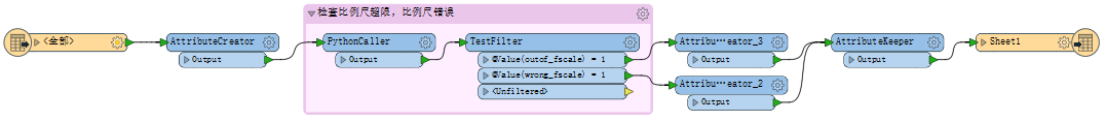

图 11 显示比例检查

```python
def check_fscale(self,feature):
    """按权重表检查fcode对应的fscale级别"""
    Fcode_Fscale = {"100201": 14, "100202": 14, "100301": 14, "100302": 14, "110101": 12, "110102": 14,
                    "110103": 14, "110104": 14, "110201": 16, "110301": 14, "110302": 16, "110303": 16,
                    "110304": 16, "110305": 16, "110401": 16, "110501": 15, "110601": 15, "120101": 13,
                    "120102": 13, "120103": 15, "120104": 16, "120105": 14, "120201": 14, "120202": 16,
                    "120203": 16, "120204": 17, "130101": 11, "130102": 14, "130103": 14, "130104": 16,
                    "130201": 9, "130202": 9, "130203": 9, "130204": 10, "130301": 16, "130302": 14,
                    "130303": 14, "130304": 16, "130305": 15, "130306": 15, "130307": 17, "130308": 16,
                    "130309": 16, "130310": 15, "130311": 16, "140101": 16, "140102": 14, "140103": 16,
                    "140104": 16, "140105": 16, "140106": 15, "140107": 15, "140108": 15, "140109": 14,
                    "140201": 13, "140202": 13, "140203": 18, "140204": 16, "140205": 16, "140206": 15,
```

图 12 部分分类代码对应的比例尺级别

（3）异常字符和空字段检查

POI 的异常字符检查主要针对名称和简称，且两者不能同时为空，如下图 13 所示。异常字符包括“？”，“.”和空格，简称可以包含空格。分类代码、显示比例、符号化字段、更新参考数据源、更新时间、更新类型等字段属性值必须有值，不能为空。

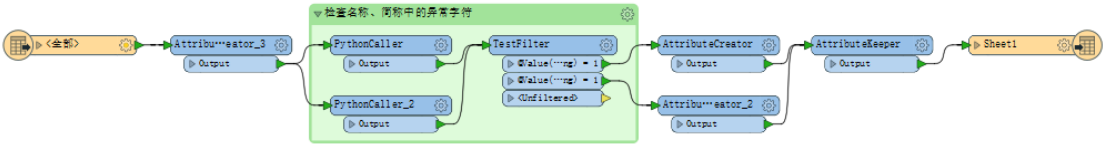

图 13 名称和简称异常字符检查

（4）全半角符号检查

POI 的全半角符号检查针对名称和地址，检查全半角转换是否处理，如下图 14 所示，Pythoncallor 中的代码如下图 15 所示。

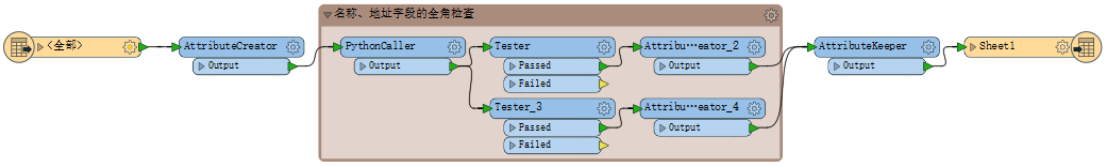

图 14 全半角符号检查

```
def full_angel_name(self,feature):
    print(feature.getAttribute('NAME'))
    a=[]
    for s in feature.getAttribute('NAME'):
        # 全角空格、全角字符
        if ord(s)==12288 or (ord(s)>= 65281 and ord(s)<= 65374):
            # print("full angle!")
            a.append(s)
            feature.setAttribute("full_angel_name",1)
            break
    if len(a) == 0:
        # 无全角字符
        feature.setAttribute("full_angel_name",0)
```

图 15 全半角符号处理

3.3.3 空间关系规则设计

POI 的空间关系规则主要涉及到图面负载以及不规则落线面。

（1）图面负载检查

图面负载，指的是对数据符号化显示的过程中，要素过于密集而产生的压盖现象，因此，需要联合检查 POI 和电子地图中的居民地点（RES_PT）。按照距离权重，在每一比例尺级别下，查找每个点对应的最近点的距离是否小于该级别下的临界距离，保证图面负载正常，各级别平滑过渡，如下图 16 所示。

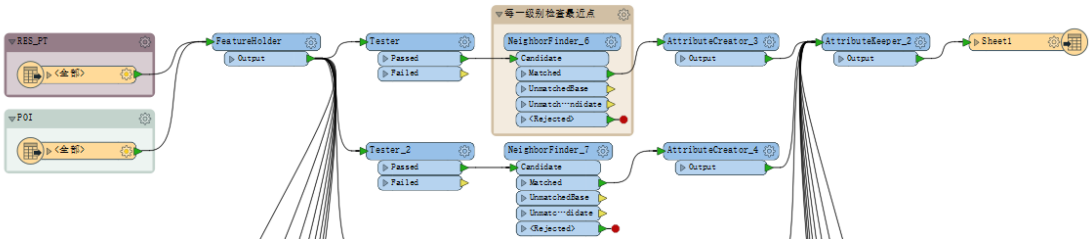

图 16 图面负载检查

（2）不规则落线面检查

POI 不规则落线、面指的是 POI 不能落在水系面，道路线和轨道线上。水系面中除了岛屿、滩涂，道路线和轨道线除了公交站、地铁站等，在一定容差范围内，不允许点落在水系面，道路线和轨道线上。通过 FME 的空间分析工具 PointOnArea，查找不规则落线、面的 POI，如下图 17 所示。

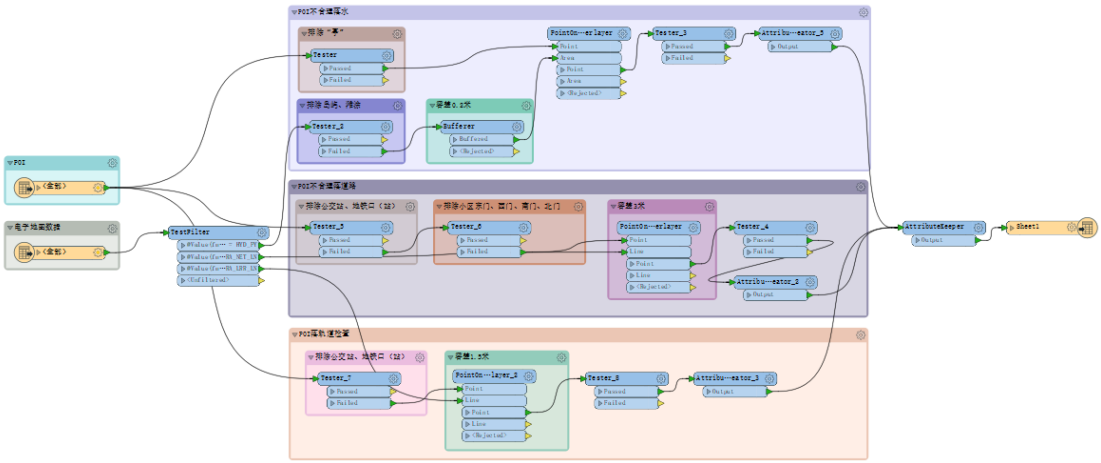

图 17 不规则落线面检查

3.4 电子地图质检程序设计

电子地图数据包括水系、植被、交通、居民地、境界等等，同样以省级节点数据为主，参考市、县级节点更新源数据对各要素类图层的属性和空间信息进行融合补充。

融合后的电子地图数据要处理各要素类图层本身以及图层之间的拓扑关系，属性信息填写正确，并保证数据的现势性。

3.4.1 基础质检规则设计

电子地图的基础质检规则，主要针对全要素类数据，包括空间坐标系、整体拓扑检查、逻辑一致性和数据完整性。其中，空间坐标系检查、逻辑一致性检查和数据完整性检查与 POI 的检查同理。

整体拓扑检查，包含了复合线面检查、自相交线面检查、线首尾相连检查、独立层线面重叠检查四个部分，是对电子地图所有的线要素和面要素做统一检查。

（1）复合线面检查

复合线面指的是，未经过打散处理的线要素和面要素。利用 FME 的 Deaggregator 转换器，判断要素是否打散处理，最终通过筛选，将未打散的数据输出，如下图 18 所示。

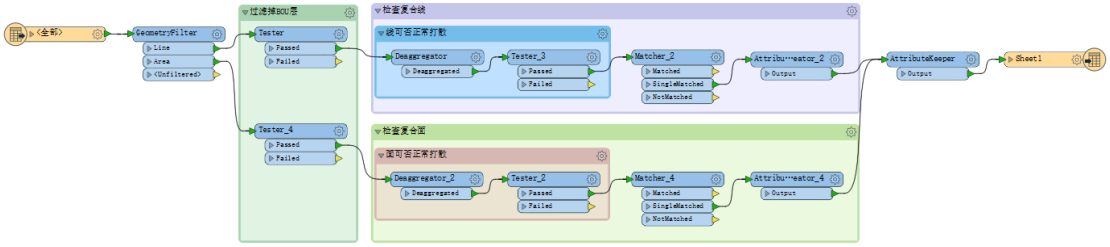

图 18 复合线面检查

（2）自相交线面检查

拓扑检查的另一个重要的检查项是自相交检查，在电子地图中，不应该存在自相交的要素。通过 FME 检查拓扑的转换器 GeometryValidator，并设置自相交选项（如图 20），判断数据是否存在自相交，如下图 19 所示。

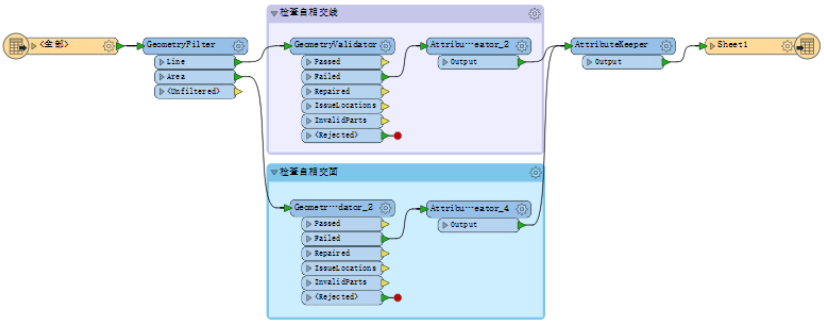

图 19 自相交线面检查

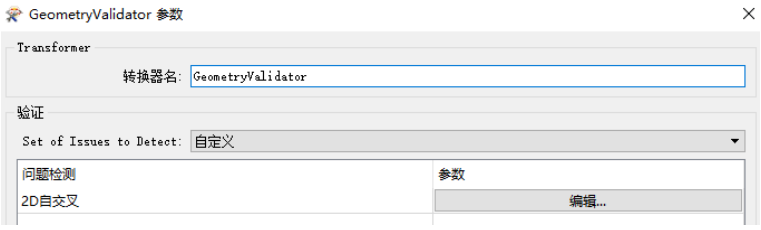

图 20 自相交选项设置

（3）线首尾相连检查

在电子地图中，线要素不允许存在首尾相连的情况。利用 FME 的 CoordinateExtractor 转换器，捕捉到线首尾两点的坐标值，通过检查两点坐标值是否完全一致来判断线的首尾是否相连，如下图 21 所示。

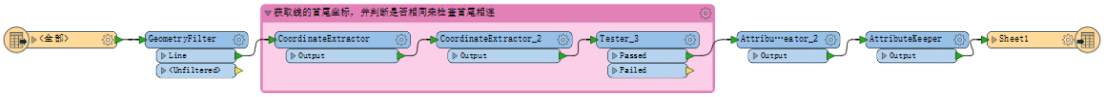

图 21 线首尾相连检查

（4）独立层线面重叠检查

独立层线面重叠指的是，在单独的线要素类图层或面要素类图层中，要素存在重叠现象。通过 pythoncallor 转换器直接调用 arcpy 的 arcpy.Intersect_analysis()函数（如图 23），来判断要素之间是否重叠相交，如下图 22 所示。

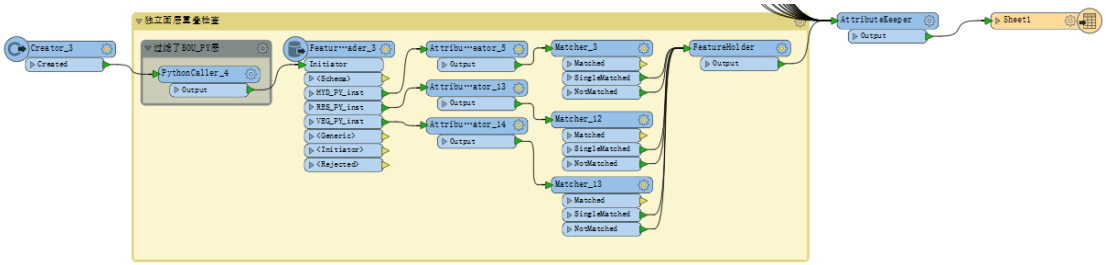

图 22 独立层线面重叠检查

```
def check_areaon_selfarea(self):
    """检查面的自我重叠"""
    fclass = ["HYD_PY", "RES_PY", "VEG_PY"]
    for fc in fclass:
        try:
            inFeatures = fc
            outFeatures = fc+"_inst"

            inputpath = os.path.join(date_datefusion, inFeatures)
            outputpath = os.path.join(date_datefusion, outFeatures)

            arcpy.Intersect_analysis(inputpath, outputpath, "ALL", "", "")
        except Exception as e:
            print(e)
```

图 23 独立层线面重叠检查实现

3.4.2 属性质检规则设计

电子地图的属性质检，是针对于电子地图的每一个要素类图层的属性进行检查，每一个要素类图层都有其独特的属性信息。除了对分类代码、显示比例、全半角符号等这些通用的，与 POI 属性项一致的检查外，还包括了道路路线编码、道路属性必填项、水系注记线属性等。

（1）道路路线编码检查

道路的路线编码，是由特定的英文大写和数字组成（如国道：G25），描述了道路的类型，在 Pythoncallor 中对检查规则进行定义（如图 25）。

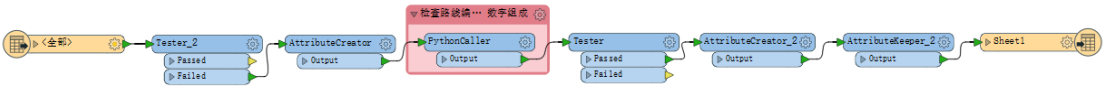

图 24 道路路线编码检查

```
def check_roadcode(self,feature):
    print(feature.getAttribute('ROADCODE'))
    a=feature.getAttribute('ROADCODE')
    b=len(a)
    if a[0] not in ["X","Y","S","Q","G","Z"] or b>5 :
        feature.setAttribute("roadcode_wrong",1)
    elif b>=2 and a[1:].isdigit() == False:
        if a[1:].isdigit() == False:
            feature.setAttribute("roadcode_wrong",1)
    else:
        feature.setAttribute("roadcode_wrong",0)
```

图 25 独立层线面重叠检查实现

（2）道路属性必填项检查

道路属性必填项包括三个方面，分别是公路与城市道路共线、公路与公路共线以及匝道处理。

公路与公路共线时，属性总体采用从高到低原则，FCODE、FNAME 及 ROADCODE 填写较高等级道路代码、名称和路线编码，DISPLAY 填写较高等级公路代码。FCODE2、FNAME2 及 ROADCODE2 填写次高等级道路代码、名称和路线编码，以此类推。利用 Tester 和 TestFilter 转换器对各属性值进行比较和分类，并结合自定义转换器建立分类代码和道路编码的权重关系，最终判断公路与公路共线时的各项属性值填写问题，如下图 26 所示。

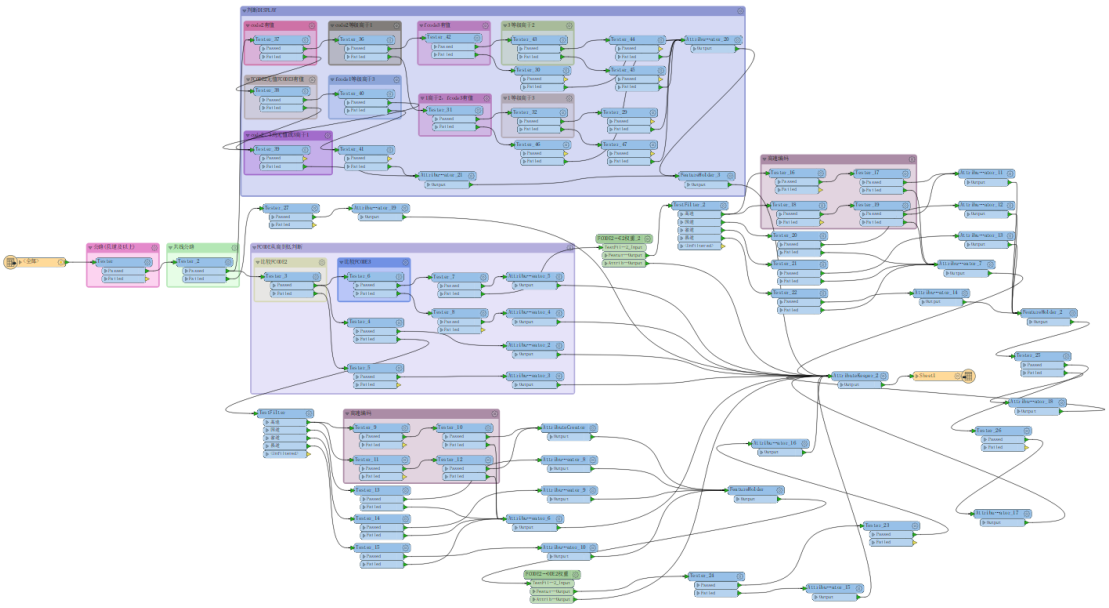

图 26 公路与公路共线检查

公路与城市道路共线，FCODE 及 FNAME 填写城市道路代码及名称，FCODE2、FNAME2 及 ROADCODE2 填写公路代码、名称和路线编码。DISPLAY 按实际填写，保证路网连通性，县道及以上填写公路代码，县道以下填写城市道路代码，如下图 27 所示。

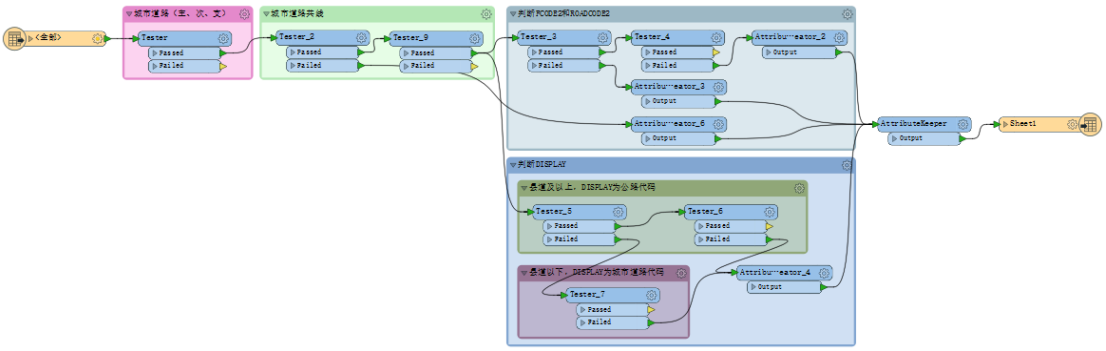

图 27 公路与城市道路共线检查

匝道处理，总体采用就低原则：即 DISPLAY 填写与匝道相连接的较低等级的 DISPLAY。

检查与匝道首或尾相连的较低等级道路 DISPLAY。若其中一端连接匝道，DISPLAY 参照匝道 DISPLAY，若两端无匝道，则做 50 米缓冲，参照缓冲内的匝道 DISPLAY，如下图 28 所示。

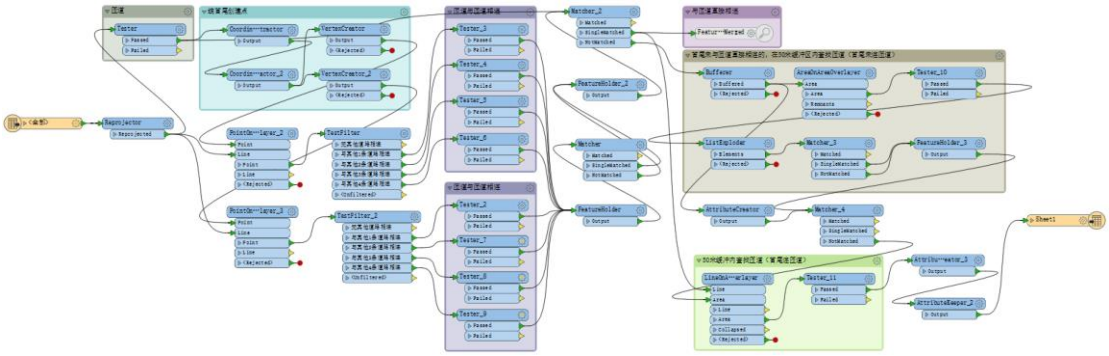

图 28 匝道检查

（3）水系注记线属性检查

水系注记线各字段属性与对应水系面的字段属性值必须保持一致，其比例尺应该大于或等于水系面的比例尺，利用 SpatialFilter 工具判断水系注记线与水系面的空间位置关系，并对名称属性进行分组检查，如图 29 所示。

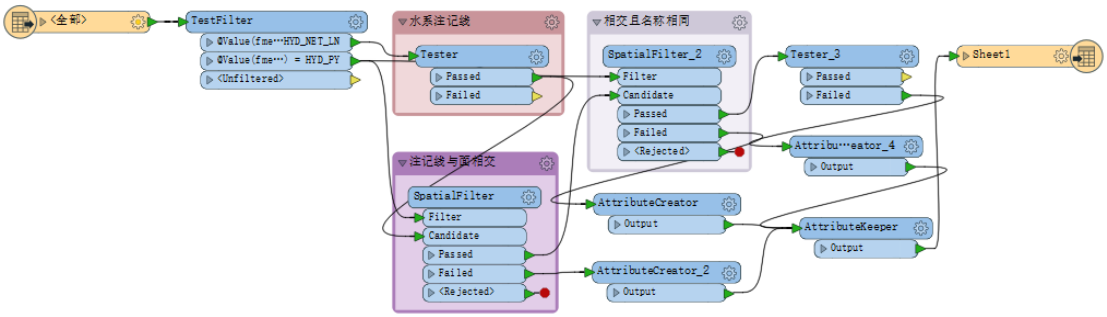

图 29 水系注记线属性检查

3.4.3 空间关系规则设计

电子地图由于其数据本身的复杂性，在空间关系规则建立时，涉及到很多方面，包括道路的连通性和孤立性，道路与桥梁的关系，水系附属设施的贯通，辅助层道路一致性，以及各线和面要素图层之间不合理的压盖关系等。在整体的拓扑检查后，对分要素类图层的拓扑检查至关重要。

（1）道路连通性和孤立性检查

道路的孤立指的是单条道路未与其他的道路有相交，而连通性检查，是在此基础上，检查每一个比例尺级别上道路的孤立性，通过设置缓冲范围并检查空间关系，最终将孤立道路筛选出，如图 30 所示。

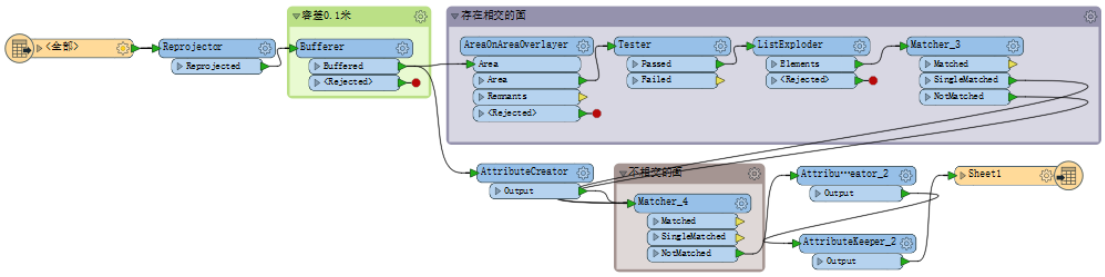

图 30 道路连通性检查

（2）桥梁与水系和道路关系检查

通常情况下，道路与水系面压盖时需要有桥梁，且桥梁不能落水。因此，需要检查桥梁和水系面的空间关系，以及判断是否与道路线重叠。筛选出桥梁线，并获取桥头桥尾点坐标，通过 PointOnArea 功能检查桥梁端点是否落在水系面内；对道路做缓冲，设置容差，检查桥梁是否在道路面内，如下图 31 所示。

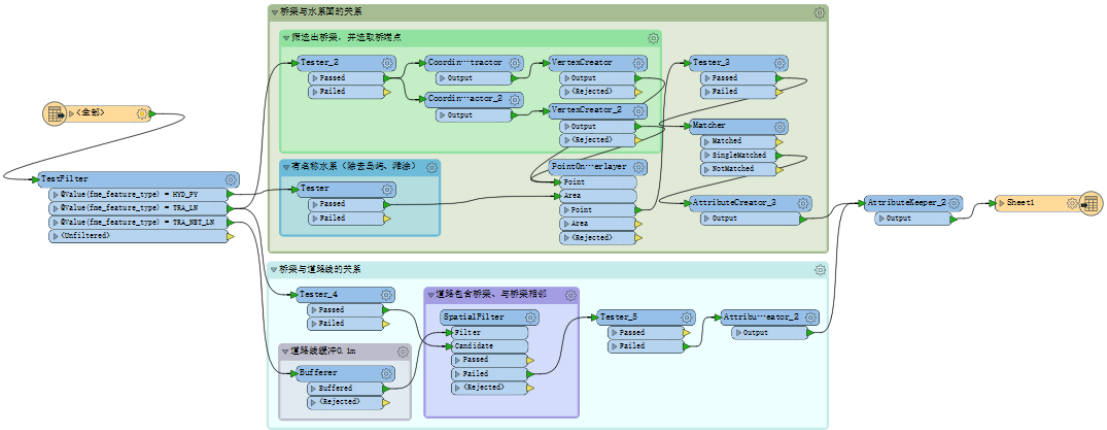

图 31 桥梁与水系道路的关系检查

（3）图层相互关系检查

图层相互关系指的是各类型数据的线面层与其他类型数据的线面层之间的拓扑关系，比如道路线与水系线和水系面的拓扑关系，植被面与居民地面之间的拓扑关系等，图层相互关系正确是电子地图正常显示的基础。现以建筑面与水系线和水系面的拓扑检查为例，来说明一般线面重叠、面面重叠的情况。

在检查线与面的关系时，通过 FME 的 LineOnArea 转换器，并设置容差来检查线与面的压盖关系。在检查面与面的关系时，使用 AreaOnArea 转换器，先除去面要素自身的重叠，再通过面面叠加，设置容差来检查面与面的压盖关系，如下图 32 所示。

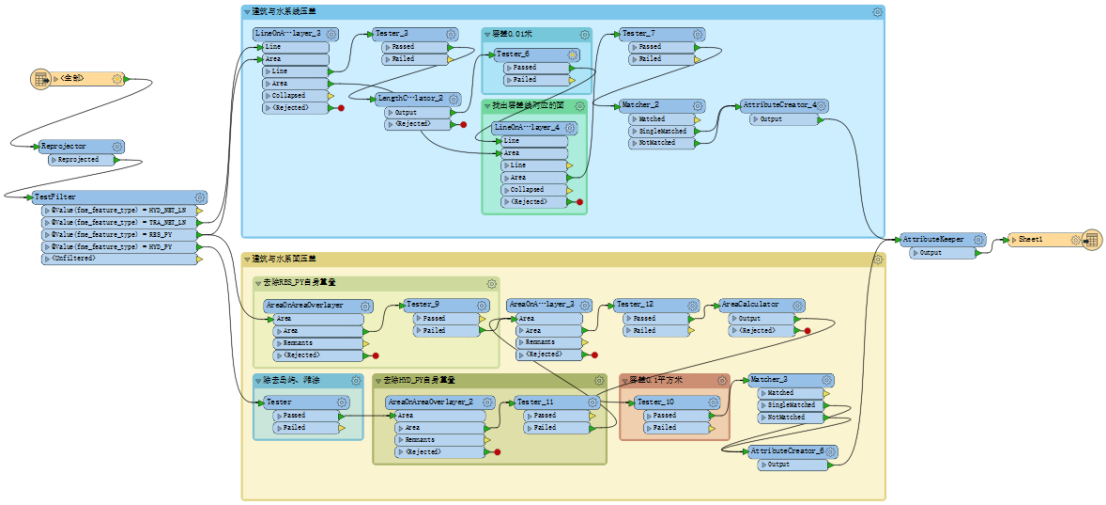

图 32 建筑面与水系线水系面压盖检查

（4）水系附属设施贯通检查

水系附属设施包括输水渡槽、溢洪道等，这类附属设施应该与水系线或水系面合理贯通。在筛选出水系附属设施以后，对水系线和面建缓冲区设置容差，与附属设施做线面相交。最后筛选出未相交的即未贯通的水系附属设施，如下图 33 所示。

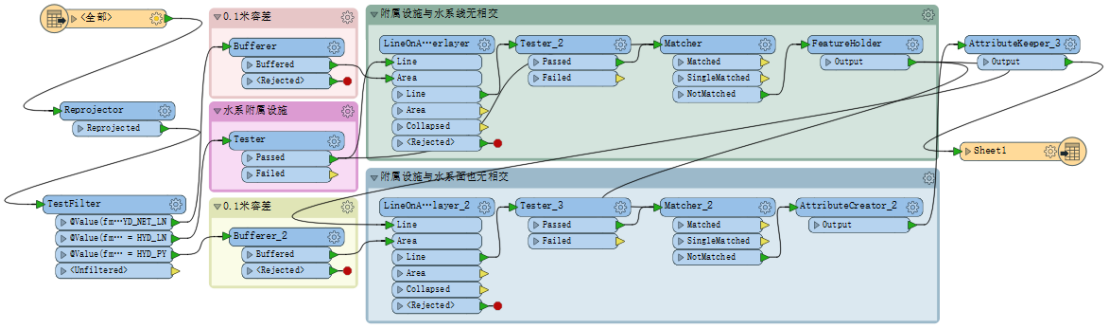

图 33 水系附属设施贯通检查

（5）辅助层的道路一致性检查

按照要求，辅助层（ASST_LN）道路必须从道路数据中截取，形状和属性与截取的道路完全一致。建立辅助层缓冲面，通过 SpatialFilter 工具做相交判断，如下图 34 所示。并对部分属性（FCODE、FSCALE、DISPLAY）分组，判断其是否一致（图 35）。

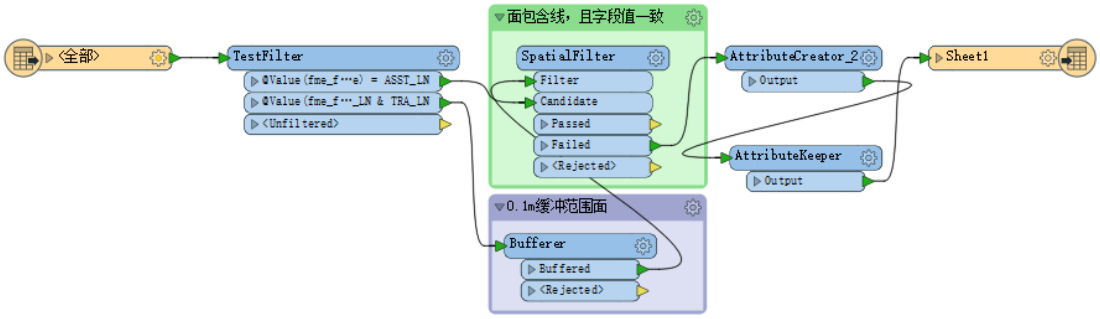

图 34 辅助层道路一致性检查

图 35 属性分组

4 质检程序应用

在所有的电子地图和 POI 的质检模板设计完成以后，将其上传至 FME Server 上，根据质检需求利用 FME Integration 对各个功能模板进行组合、配置，创建相应的质检方案，最终实现数据质检。

目前，质检程序已经在 2019 年度的天地图•浙江融合工作中全面应用，对各个参与省、市（县）融合的市（县）数据进行了质检，并反馈质检报告供他们进行数据整改，最终将整改完的数据融合到天地图•浙江省库数据中。

5 结束语

在天地图•浙江数据融合工作中，通过灵活运用FME的数据处理、分析转换器，并结合Python的二次开发，实现了对天地图•浙江的POI和矢量电子地图数据属性以及空间关系

上的质检， 节约了时间成本和人工成本，提高了检查效率，满足了成果精度要求，为提升数据质量奠定了基础，为天地图融合的有序推进提供了保障[4]。

文献参考：

[1]天地图分布式数据中心建设与数据融合示范［S］，国家测绘地理信息局

[2]邱新忠,朱俊.天地图省市数据融合若干关键技术研究及应用—以“天地图·浙江”为例[J].测绘与空间地理信息，2018,41(1):179-181.

[3]任俊儒.FME 在 GIS 数据处理中的应用[J].测绘与空间地理信息，2018,41(11):129-134

[4]侯超.基于 FME 的天地图融合数据质量检查方法研究[J].江西测绘，2017,02:2-4