OpenStreetMap Project with SQL

地图范围

本次项目所选地图是深圳市地图。 下载地址如下： https://mapzen.com/data/metro-extracts/metro/shenzhen_china/ (https://mapzen.com/data/metro-extracts/metro/shenzhen_china/) 选择深圳市地图的主要原因是： 我在深圳工作生活了多年，比较熟悉，比较容易找到那些数据是不合适。

我的地图遇到的问题

1.街道名称不统一

查看街道地址时，发现有的用的中文，有的用的拼音，有的用的英文。

2.审查标签类型时，出现异常标签

使用正则表达式查看标签类型时，发现有异常标签。

3.审查时发现部分邮政编码有问题

审查邮政编码时，发现部分邮政编码不是6位数字，而中华人民共和国邮政编码应该为6位

1.街道名称不统一

In [25]:

```
import xml.etree.cElementTree as ET
filename = 'shenzhen_china.osm'
def is_street_name(elem):                                   #构造一个匹配街道元素的函数
    return(elem.attrib['k'] == 'addr:street')               #返回元素属性为 k=" 'addr:street"，即是街道
                                                            #查找元素是否包含某个属性，也可以用iter输出

for event, elem in ET.iterparse(filename,events=("start",)): #对文件进行迭代解析 ，遍历每一行（元素），
    if elem.tag == "node" or elem.tag == "way" :            #如果元素的标签是为“node”或“way”
        for tag in elem.iter("tag"):                        #对其这些元素为“node”和“way”的行进行遍历
            if is_street_name(tag):                         #匹配街道信息所在的元素，找到含有属性为k=
                print tag.attrib['v']                       #因为街道信息在属性v的值上，因此我们打印
```

```
元朗炮仗坊 Yuen Long Pau Cheung Square
文心四路
Tsun Fu Street
San Shing Avenue
沙园了路
福华路 Fuhua road
福华路 Fuhua road
Shennan Avenue
Yankui Road
Ping Ha Road
廣福道 Kwong Fuk Road
廣福道 Kwong Fuk Road
Gaoxin S.
Castle Peak Road - Yuen Long
Ma Wang Road
Chunfeng Road
新湖路
XinSha Road
上步路
白石路
```

从上面的结果来看，街道地址描述标准不一，有中文，如南山大道、公园路等，也有英文San Hong Street、San Fung Avenue等，以及中文+英文。由于结果显示较多地址名字描述不一致的情形，因此暂时不处理地址描述问题。

2.审查标签类型时，出现异常标签

本文主要是为了审查标签类型，并尝试处理有问题的标签

如上面的代码所示，本例主要是审查标签下的k属性类型。 在此，用正则表达式(r'[=+/&<>;\'"\?%#$@\,. \t\r\n]'判断tag['k']中是否有特殊字符space.&+,#%;等 使用tag.py 编码如下：

In [8]:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import xml.etree.cElementTree as ET                    #导入解析XML文档的模块
import pprint                                          #pprint模块是得输出结果按每一行展示，是输出结果更
import re                                               #导入正则表达式模块


lower = re.compile(r'^([a-z]|_)*$') #表示仅包含小写字母且有效的标记
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$') #表示名称中有冒号的其他有效标记“：”
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]') #表示字符存在问题的标记


def key_type(element, keys):
    if element.tag == "tag":
        global error_attrib                            #设置 error_attrib 为全局变量
        k_attrib = element.attrib["k"]

        flag = 0                                       #前三个判断语句形式类似，利用flag=0作为前三个if语
        if re.search(lower, element.attrib["k"]):
            keys["lower"] += 1
            flag = 1

        if re.search(lower_colon, element.attrib["k"]):
            keys["lower_colon"] += 1
            flag = 1

        if re.search(problemchars, element.attrib["k"]):
            keys["problemchars"] += 1
            flag = 1
            error_attrib = k_attrib  # 单独为错误标签设置新变量


        if flag == 0:
            keys["other"] += 1

    return keys

def process_map(filename):
    keys = {"lower": 0, "lower_colon": 0, "problemchars": 0, "other": 0}
    for _, element in ET.iterparse(filename):
        keys = key_type(element, keys)

    return keys

if __name__ == "__main__":

    keys = process_map('shenzhen_china.osm')
    print keys

```

```
{'problemchars': 1, 'lower': 191162, 'other': 3490, 'lower_colon': 45518}
```

从运行结果可以看出，问题标签有一个，这里有个特殊字符是空格，可以将空格去掉，使用“_” 下划线将空格替换，同时只截取“：”后部分。使用函数具体如下：

In [17]:

```
print error_attrib
temp_attrib  = error_attrib.split(" ") # 将k_attrib按空格分开
k_attrib_new = "_".join(temp_attrib).split(':')[1]
print k_attrib_new
```

```
name:MT Bike
MT_Bike
```

3.审查邮政编码是否准确

In [27]:

```
import xml.etree.cElementTree as ET
filename = 'shenzhen_china.osm'

def  is_post_code(elem):                                    #构建匹配邮政编码的函数
   return(elem.attrib['k'] == "addr:postcode")              #匹配元素中含有属性为k值为 "addr:postcode"的

for event, elem in ET.iterparse(filename, events=("start",)):          #用iterparse 一次遍历所有层级的元

   if elem.tag == "node" or elem.tag == "way":                         #遍历元素时如果 遇到标签为“node
      for tag in elem.iter("tag"):                                     #遍历这些元素
         if is_post_code(tag):                                         #调用函数is_post_code，匹配属性
            post_code = tag.attrib['v']                                #获取所在元素属性为k的值；
            if len(str(post_code)) != 6:                               #判断字符串邮政编码个数是否不等
               print post_code                                         #如果不等于6，打印出出现异常的
```

```
DD109 754
DD109 754
DD117 137
DD117 137
DD117 137
DD117 137
DD117 133
DD117 133
DD117 133
DD117 137
DD117 137
DD117 137
DD117 139
DD117 136
DD117 121
DD117 121
DD117 149
DD117 149
DD91 3898
DD91 3895
```

从以上结果看出，部分邮编出现了5位和8位，均为异常邮编，在写入数据的时候，我们选取问题邮编中同时含有字符"DD"和空格的邮编进行处理。

问题邮编数据清洗

In [9]:

```
import xml.etree.cElementTree as ET
filename = 'shenzhen_china.osm'

#def update_post_code(post_code):
 #  new_post_code = post_code.replace(' ', '')    #将字符串中的空格删除
 #  return new_post_code

def  is_post_code(elem):                                    #构建匹配邮政编码的函数
   return(elem.attrib['k'] == "addr:postcode")              #匹配元素中含有属性为k值为 "addr:postcode"的

for event, elem in ET.iterparse(filename, events=("start",)):              #用iterparse 一次遍历所有层级的元

   if elem.tag == "node" or elem.tag == "way":                             #遍历元素时如果 遇到标签为“node
      for tag in elem.iter("tag"):                                         #遍历这些元素
         if is_post_code(tag):                                             #调用函数is_post_code，匹配属性
            post_code = tag.attrib['v']                                    #获取所在元素属性为v的值；
            if 'DD' in post_code:
               new_post_code = post_code.replace(' ', '')[2:]
               print new_post_code                                         #打印清洗结果
```

```
51120
51130
51131
51129
51128
51127
51126
51137
51136
51135
51134
51133
51132
1651471
1651471
1651471
1651471
913843
913876
913876
```

将数据写入csv文件

按照“案例研究"：OpenStreetMap数据[SQL]”中准备数据集的方法，将深圳市地图相关数据读入csv文件中，请查看data.py中的代码

将深圳市地图写入数据库

创建 shenzhen.db 数据库

将CSV文件导入数据表

用python代码将 nodes.csv, nodes_tags.csv, ways.csv, way_nodes.csv, ways_tags.csv这5个csv文件分别写进数

据库 shenzhen.db中，对应表格分别为 nodes, nodes_tags, ways, ways_nodes, ways_tags。

python代码见 import_nodes_csv.py 文件，以下以 nodes.csv 导入 nodes 数据表为例，代码如下：

In [11]:

```
# coding=utf-8
import csv, sqlite3

con = sqlite3.connect("shenzhen.db")  #创建数据库文件链接，如果文件不存在就会自动生存
con.text_factory = str

cur = con.cursor()

cur.execute('drop table if exists nodes')

nodes = '''create table nodes(
id Integer,
lat float,
lon float,
user Text,
uid Integer,
version Text,
changeset Integer,
timestamp Text);
'''

cur.execute(nodes)

with open('nodes.csv','rb') as fin:
    dr = csv.DictReader(fin)
    for row in dr:
        id_value = int(row['id'])
        lat_value = float(row['lat'])
        lon_value = float(row['lon'])
        user_value = str(row['user'])
        uid_value = int(row['uid'])
        version_value = str(row['version'])
        changeset_value = int(row['changeset'])
        timestamp_value = str(row['timestamp'])
        cur.execute('INSERT INTO nodes VALUES (?,?,?,?,?,?,?,?)',
            (id_value,lat_value,lon_value,user_value,uid_value,version_value,changeset_value,timestamp_

con.commit()
con.close()
```

其他csv文件的导入见相应的python文件，代码见import_nodes——tags_csv.py，import_ways_csv.py，import_ways_nodes_csv.py，import_ways_tags_csv.py

用SQL查询数据

1.查询数据库的表格

sqlite> .tables

nodes nodes_tags ways ways_nodes ways_tags

2.查询表node和ways的数量

sqlite> select count() *from nodes; 721154 sqlite> select count*() from ways； 78078

3.查询唯一用户的数量

In [12]:

```
# coding=utf-8
import csv, sqlite3
con = sqlite3.connect("shenzhen.db")
cur = con.cursor()

#提取 nodes 数据表中的独立用户数
query1 = "select uid from nodes group by uid order by uid"
cur.execute(query1)
nodes_uid = cur.fetchall()
print 'Unique uid in table nodes is:'
print len(nodes_uid)

#提取 ways 数据表中的独立用户数
query2 = "select uid from ways group by uid order by uid"
cur.execute(query2)
ways_uid = cur.fetchall()
print 'Unique uid in table ways is:'
print len(ways_uid)

#计算整个数据库中的独立用户数，需要将nodes数据表和ways数据表中的独立用户数相加并去重
for i in range(len(ways_uid)):
    if ways_uid[i] not in nodes_uid:
        nodes_uid.append(ways_uid[i])
unique_uid = len(nodes_uid)
print 'total unique users is:'
print unique_uid
```

```
Unique uid in table nodes is:
748
Unique uid in table ways is:
534
total unique users is:
832
```

关于数据集的其他想法

对于非英语国家，在地址描述上会显得比较杂乱，可以设置约束，某些字段要求用英文来描述，在增加本国语言描述的字段，方便使用。

好处：有统一的约束条件之后，可以让不同用户编辑的时候，有统一的标准，避免每个用户都按照自己的标准来编辑，导致最终清理时，出现不统一的情况。

预期的问题：统一地图上元素命名规则后，需要注意街道名称尽量采用全称，减少缩写和不规范的情况，各元素信息应当遵循规范化的表达。