课程编号:20193024311　　课程性质:必修

空间数据库

课程设计与实习报告

学院: _____

专业: _____

地点: _____

班级: _____

组号: _____

姓名: _____

学号: _____

教师: _____

2020年6月22日　至　2020年6月26日

目录

一、实习目的与要求.....1
1.实习目的.....1
2.实习要求.....1
3.实习安排.....1
二、实习内容与过程.....2
1.环境配置和数据准备.....2
2.空间数据库概念设计.....3
3.空间数据库逻辑设计.....4
3.1 道路对象关系表.....4
3.2 超市对象关系表.....5
3.3 医院对象关系表.....5
3.4 建筑物对象关系表.....5
3.5 运动场对象关系表.....6
3.6 交叉关系表.....6
3.7 距离量算关系表.....6
4.数据库建立.....7
5.数据表建立.....7
6.数据插入.....8
6.1 sql 语言插入.....8
6.2 python 语言批量插入.....9
6.3 创建查询关系表.....12
7.空间查询操作.....12
7.1 查询每栋建筑物的边界（STEnvelope）.....12
7.2 计算每条道路的长度(STLength).....14
7.3 为每条道路构建缓冲区（STBuffer，缓冲距离为 20m）.....14
7.4 每条道路构建缓冲区后会影响到哪些建筑物？.....15
7.5 查询道路穿越建筑物的部分(STIntersection).....15
7.6 找出距离“4 号教学楼”最近的超市和最近的医院（STDistance）16

8.在 ArcGIS 中加载数据库.....17
9.实习成果展示.....18
三、实习中遇到的问题及解决方案.....19
四、实习体会与心得.....20
******.....20
五、附件说明.....20
1.地理数据.....20
2.代码数据.....20
3.空间数据库实习报告.pdf.....21
4.空间数据库实习报告.word.....21

一、实习目的与要求

1.实习目的

通过实习让学生深入的理解空间数据库的概念和原理，掌握《空间数据库原理》的相关内容。更为重要的是达到以下的目的：

1) 通过对主流空间数据库管理软件（如 SQLServer）的实践操作，加深对空间数据库基本原理的理解及领会，最终能够熟练运用一种空间数据库库管理软件；
2) 掌握地理空间数据库管理方法；
3) 掌握空间数据库设计方法；
4) 掌握空间数据库建库方法。
5) 掌握空间数据插入、删除、修改和空间函数的操作方法；
6) 掌握空间数据库在 GIS 软件中的加载与应用。

2.实习要求

1) 在实习时，仔细阅读实习文档中的有关概念和方法。
2) 学生固定使用指定区域的计算机。
3) 使用指定的数据，完成指定实习任务，做好实习过程记录。
4) 实习中应能灵活应用实习文档中方法，处理实习过程中出现的问题。
5) 写出整个实习过程和所遇到的问题，并给出解决方法。
6) 成果整理、写实习报告（包括实习过程、遇到的问题、解决的方法及实习成果分析）。

3.实习安排

1) 实习时间: 2020 年 6 月 22 日——6 月 26 日
2) 实习地点: 线上
3) 实习安排

序号	时间安排	实习内容	备注
1	6 月 22 日	实习动员、数据准备、环境准备	
2	6 月 22 日—6 月 23 日	空间数据库设计	
3	6 月 24 日	空间数据库建库	
4	6 月 25 日	空间数据操作	
5	6 月 26 日下午	提交实习报告；实习报告必须通过老师审核合格后才能够提交，统一审核时间为 6 月 26 日下午。	线上提交

二、实习内容与过程

1.环境配置和数据准备

本次实习采用的是 SQLServer 数据库，其操作平台为 Microsoft SQL Server Management Studio；地理数据可视化采用的是 ArcGIS 软件；其中还用到了 python 语言，其编译器为 Spyder。

实习的地图数据为武汉大学测绘校区的 ArcGIS 矢量化测图数据，包括点、线、面数据。

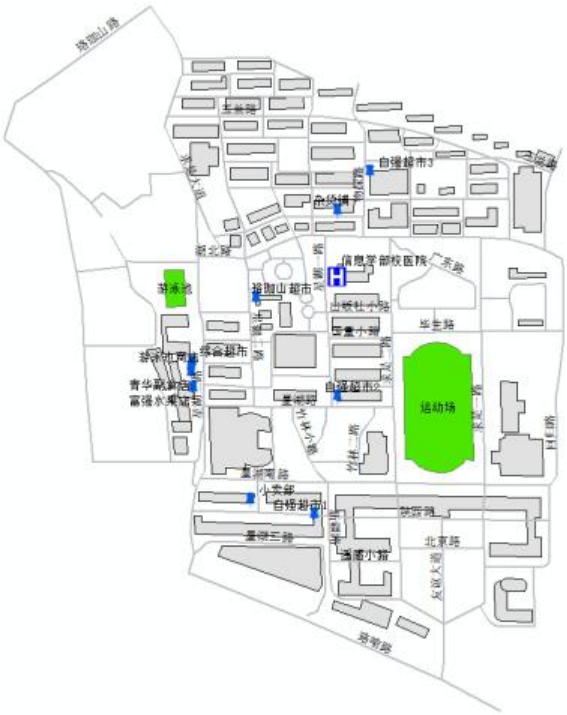


图 1 实习数据图

本次实习的测绘校区矢量图由五类地物组成：超市、医院、道路、建筑物和运动场。这五类地物的几何属性不完全一样，超市和医院是点对象，道路是线对象，建筑物和运动场是面对象。不同的几何类型能帮助我们更好地认识地图矢量数据的处理和显示，通过学习不同类型数据的读取和导入，让我们对 ArcGIS 数据的结构了解更加透彻。

根据实习目的和要求，现将整个实习的步骤用流程图表示：

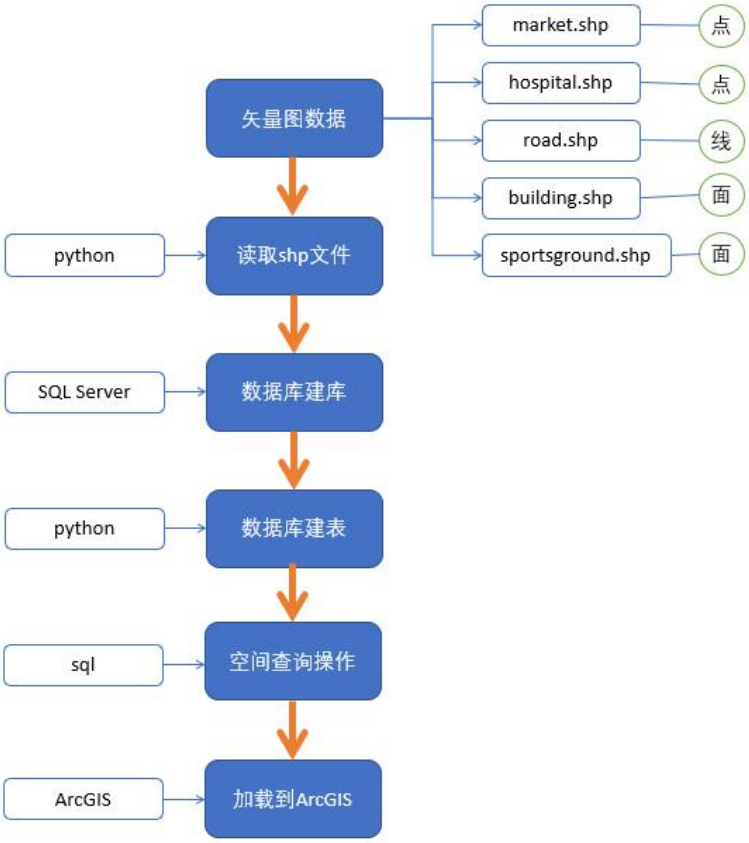


图 2 流程图

2.空间数据库概念设计

本次的空间数据库主要由超市、医院、道路、建筑物、运动场五类空间实体组成，每类实体之间都有相交、距离量算等关系，这些关系还包含相对应的属性。

空间数据库概念设计的 ER 图如下图所示。

图 3 空间数据库的拓展 ER 图

3.空间数据库逻辑设计

根据空间数据库的 ER 图，可以建立其逻辑设计的关系数据表，一张二维表即代表一个关系。

3.1 道路对象关系表

表格 1 道路对象关系表

字段名	字段类型	备注
id	integer	primary key
name	char(50)	
type	char(50)	
length	float	
geometry	geometry	

3.2 超市对象关系表

表格 2 超市对象关系表

字段名	字段类型	备注
id	integer	primary key
name	char(50)	
x	float	
y	float	
geometry	geometry	

3.3 医院对象关系表

表格 3 医院对象关系表

字段名	字段类型	备注
id	integer	primary key
name	char(50)	
x	float	
y	float	
geometry	geometry	

3.4 建筑物对象关系表

表格 4 建筑物对象关系表

字段名	字段类型	备注
id	integer	primary key
name	char(50)	
height	float	
length	float	
area	float	
geometry	geometry	

3.5 运动场对象关系表

表格 5 运动场对象关系表

字段名	字段类型	备注
id	integer	primary key
name	char(50)	
type	integer	
length	float	
area	float	
geometry	geometry	

3.6 交叉关系表

表格 6 交叉关系表

字段名	字段类型	备注
建筑物 id	integer	foreign key
道路 id	integer	foreign key
交叉部分 intersection	linestring	

3.7 距离量算关系表

表格 7 距离量算关系表（超市）

字段名	字段类型	备注
建筑物 id	integer	foreign key
超市 id	integer	foreign key
距离 distance	float	

表格 7 距离量算关系表（医院）

字段名	字段类型	备注
建筑物 id	integer	foreign key
医院 id	integer	foreign key
距离 distance	float	

4.数据库建立

打开 Microsoft SQL Server Management Studio，在左侧菜单栏中右键新建数据库，并命名为 WHU，即建立了一个名为 WHU 的数据库。

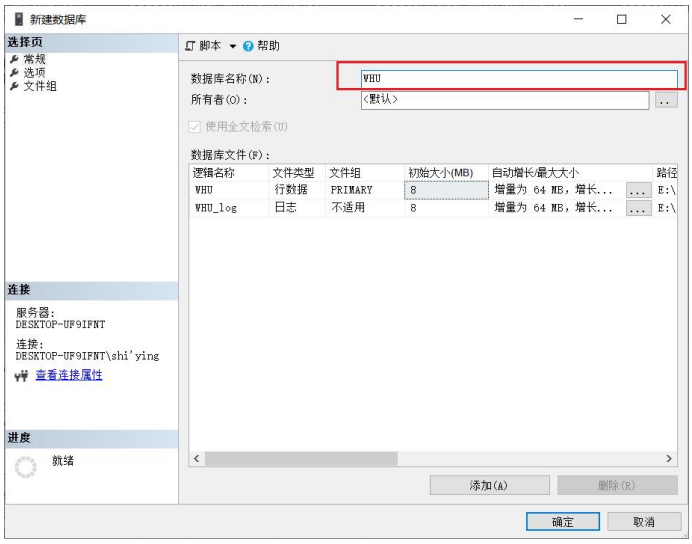


图 4 新建数据库 WHU

5.数据表建立

数据表的建立分两种方式，一种是在 SQL Server Management Studio 通过界面操作进行建表；另一种方法是 sql 语言建表。

1) 操作界面建表

列名	数据类型	允许 Null 值
id	int	☐
name	char(50)	☑
height	int	☑
length	float	☑
area	float	☑
geometry	geometry	☑
		☐

图 5 操作界面建表

在 SQL Server Management Studio 中右键“表”，选择新建表，可以设计表的字段名和字段类型，如图 5 所示。

2) sql 语言建表

以 market 表为例：

```sql
CREATE TABLE market
 (id int PRIMARY KEY,
    name char(50),
    x float,
    y float,
    geometry geometry
   );
```

由以上 sql 语句，便可建立如 3.2 所示的 market 表。

本次实习我采用的是 sql 语言建表的方式，并且是在 python 中引用 pymssql 库进行和数据库的连接并建立数据表。

打开 Spyder，编写 Python 代码进行数据库连接和建表工作。

```python
def conn():
    connect=pymssql.connect(
        'localhost:1433','sa','sy1999820.','COVID')
    if connect:
        print("连接成功！")
    return connect
```

图 6 数据库连接函数

```python
if __name__=='__main__':
    conn=conn()
    #创建一个游标对象，python里的sql语句都要通过cursor来执行
    cursor=conn.cursor()

    sql01="create table [WHU].[dbo].[building] (id integer primary
    cursor.execute(sql01)
    conn.commit()

    cursor.close()
    conn.close()
```

图 7 数据库连接主函数和 sql 语言

通过图 6 和图 7 所示的代码，可以实现数据库的连接，通过 cursor.execute(sql) 能够将 python 中的 sql 语句放到 SQLServer 中去运行。

6.数据插入

6.1 sql 语言插入

使用 INSERT INTO 语句将数据输入到对应的数据表中，INSERT INTO 语句基本语法为：

```sql
INSERT INTO table
VALUES(X,X,…);
```

以超市的表为例，展示插入数据的具体 sql 语句：

```
INSERT INTO market
VALUES(1,'自强超市1',533908.9258,3378828.3382,geometry::STGeomFromText('POINT(533908.9258 3378828.3382)',4214))
```

在本次实习中，共有五种空间实体，且包含了点、线、面三种矢量类型。因此，在使用 geometry::STGeomFromText()的过程中会有一些区别，其中超市和医院是点类型、道路是线类型、建筑物和运动场是面类型。三种类型的构建方法如下。

插入点：`STGeomFromText('POINT(X Y)',SRID);`

插入线：`STGeomFromText('LINESTRING(X1 Y1,X2 Y2,…,Xn Yn)',SRID);`

插入面：`STGeomFromText('POLYGON((X1 Y1,X2 Y2,…,Xn Yn))',SRID);`

其中 SRID 是是由欧洲石油测绘组(European Petroleum Survey Group, EPSG)的标准定义的空间引用标识符，它对应于基于特定椭圆体的空间引用系统，可用于平面球体映射或圆球映射。本次实习使用的数据的椭球是北京 54 椭球，所以将所有空间数据的 SRID 设定为 4214。

6.2 python 语言批量插入

上述 sql 语言简单介绍了矢量数据插入表的基本语法，但是本次实习中，点线面的数量庞大，不宜一条一条地输入数据。因此我采用了 python 语言读取 shp 文件并将数据批量插入相应的表格中。

针对点、线、面三种表，我采用了三种不同的方式去读取 shp 文件，但是其核心思想是一致的。首先通过 shapefile.Reader(path)函数读取 shp 文件，然后使用 ShapeRecords()函数获得具体的对象记录，其中每条记录都是该 shp 文件的属性表的一行值，因此可以通过循环将其取出。而针对每一列的标题，使用 fields 函数将其取出，根据需要进行存放。

以 market 为例进行全过程的解释：

1) 获取 ShapeRecords

通过 shapefile 库的函数，将属性数据读取进来后进行观察，发现它们全部都是属性表里对应的值，不包含列名。

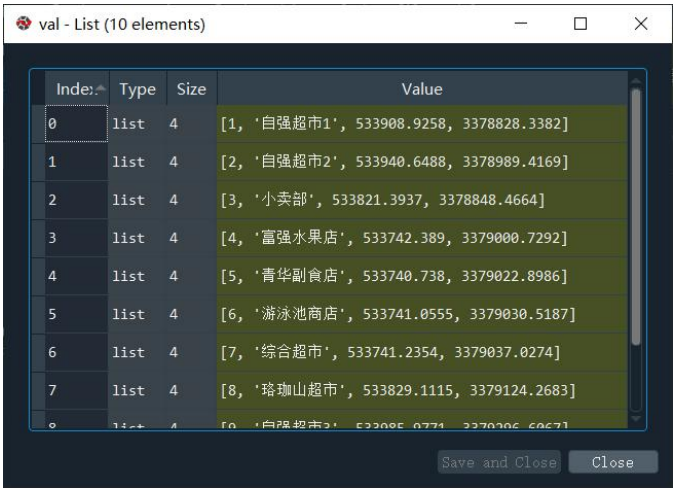


图 8 属性数据

2) 获取 fields

通过 shp.fields 函数获取 shp 文件的列名，第一列是没有用的，需要 pop 掉。剩余的根据需要，在创建表时使用对应的字段名。

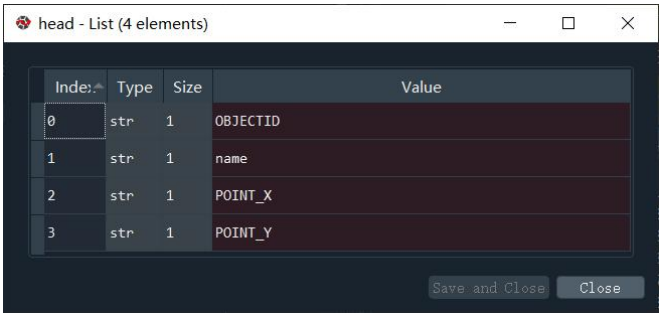


图 9 获取字段名

3) 创建表并输入数据

```
def market_sql():
    market="market.shp"
    val,typ,head=read_shp(market)

    sql01="create table [WHU].[dbo].[market] (id integer primary key,name char(50),x float,y float,geometry geometry)"
    cursor.execute(sql01)
    conn.commit()

    for index in val:
        sql02="--insert into [WHU].[dbo].[market] (id,name,x,y,geometry) values({},'{}',{},{},geometry::STGeomFromText('POINT({} {})',4214))".format(index[0],index[1],index[2],index[3],index[2],index[3])
        cursor.execute(sql02)
        conn.commit()
```

4) 查看数据表

结果：

	id	name	x	y	geometry
1	1	自强超市1	533908.9258	3378828.3382	0x76100000010C257502DA294B20413A234A2B46C74941
2	2	自强超市2	533940.6488	3378989.4169	0x76100000010C7B832F4C694B2041ADFA5CB596C74941
3	3	小卖部	533821.3937	3378848.4664	0x76100000010CE10B93C97A4A2041C5FEB23B50C74941
4	4	富强水果店	533742.389	3379000.7292	0x76100000010C0C022BC7DC492041F46C565D9CC74941
5	5	青华副食店	533740.738	3379022.8986	0x76100000010CD122DB79D949204126530573A7C74941
6	6	游泳池商店	533741.0555	3379030.5187	0x76100000010CFA7E6A1CDA492041F8C26442ABC74941
7	7	综合超市	533741.2354	3379037.0274	0x76100000010C4B598678DA492041DCD78183AEC74941
8	8	珞珈山超市	533829.1115	3379124.2683	0x76100000010C2B8716398A4A204187A75722DAC74941
9	9	自强超市3	533985.9771	3379296.6067	0x76100000010C827346F4C34B20417958A84D30C84941
10	10	杂货铺	533941.1549	3379243.9293	0x76100000010C840D4F4F6A4B20416A4DF3F615C84941

图 10 market 查询结果

空间结果：

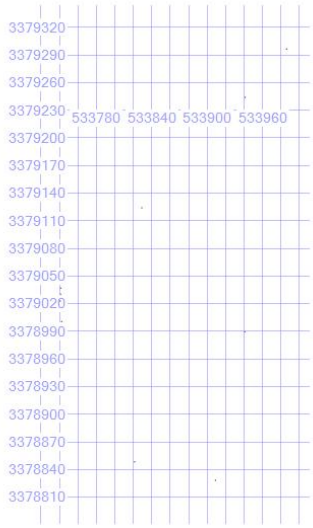


图 11 market 空间查询结果

需要注意的是，一开始我在读取点数据的时候，有 x，y 坐标，我在建 POINT 的时候用的是 x，y 坐标值。其中还进行了一些处理，使 x，y 的坐标列表和 sql 语法一致。但是在处理线数据和面数据时，我发现 shp.shapes()函数可以直接提取所有线要素或面要素，使得在插入值的时候，不需要对坐标值进行处理，大大简化了代码量，也让整个插入值的过程更加严谨和简洁。这说明我在实习的过程中，并没有一成不变，而是一边写代码一边优化代码，让数据处理过程更加的简洁和优美。

6.3 创建查询关系表

由 ER 图可知，建筑物和超市、医院之间存在距离量测关系表，建筑物和道路之间存在相交关系表，因此，在 SQL Server 中使用 sql 语言，将 building 表和其他表的空间关系写进新的表里。

1) 距离量测关系表（医院）

```
SELECT * INTO [WHU].[dbo].hospital_dis_id FROM
(SELECT b.id building,h.id hospital,b.geometry.STCentroid().STDistance(h.geometry) distance
FROM [WHU].[dbo].building b,[WHU].[dbo].hospital h
)s
```

2) 距离量测关系表（超市）

```
SELECT * INTO [WHU].[dbo].market_dis_id FROM
(SELECT b.id building,m.id market,b.geometry.STCentroid().STDistance(m.geometry) distance
FROM [WHU].[dbo].building b,[WHU].[dbo].market m
)s2
```

3) 交叉关系表（道路）

```
SELECT * INTO [WHU].[dbo].road_cross_id FROM
(SELECT b.id building,r.id road,b.geometry.STIntersection(r.geometry) intersection
FROM [WHU].[dbo].building b,[WHU].[dbo].road r
)s3
```

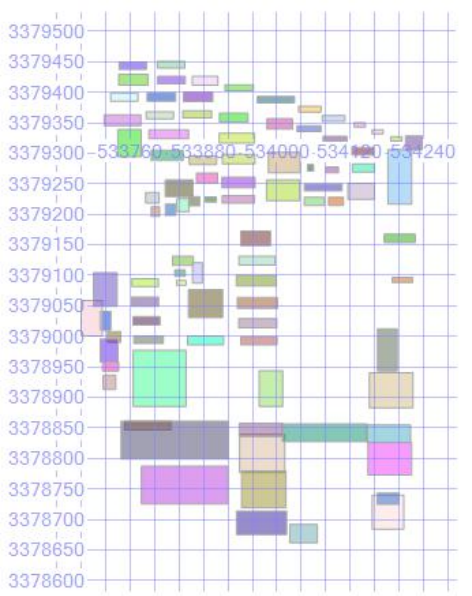
7.空间查询操作

7.1 查询每栋建筑物的边界（STEnvelope）

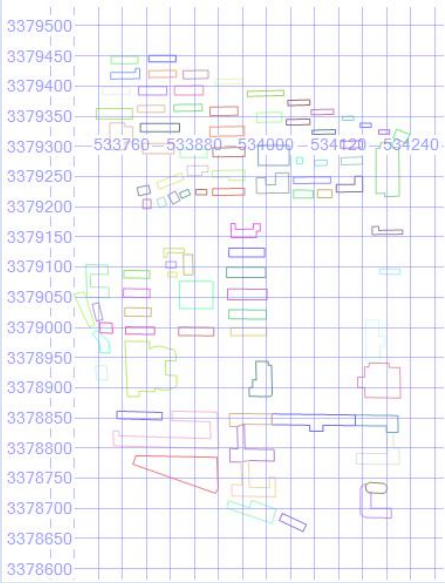
查询面状地物的边界有两种查询语言，分别是 STEnvelope() 和 STBoundary()。这两种语句的不同之处在于 STEnvelope()的返回值是面状的（最小外接矩形），而 STBoundary()的返回值是线状的边界。

下面使用这两种语言进行查询，查询结果由“空间结果”和“文本显示”两种形式展示。

使用 STEnvelope()查询面状最小外接矩形：

	空间结果型	文本型
sql 语句	SELECT geometry.STEnvelope() FROM building	SELECT geometry.STEnvelope().STAsText() FROM building
查询结果		(无列名) 1 POLYGON ((534021.10340000037 3378661.7920999993... 2 POLYGON ((533750.74100000039 3378846.8539000005... 3 POLYGON ((533933.72609999962 3378675.4175000004... 4 POLYGON ((533777.65869999956 3378725.4168, 5339... 5 POLYGON ((533744.51300000027 3378799.1982000005... 6 POLYGON ((533938.76599999983 3378777.1671999991... 7 POLYGON ((534151.6409 3378883.0699000005, 53422... 8 POLYGON ((534164.90979999956 3378943.4060999993... 9 POLYGON ((533970.89599999972 3378885.4589000009... 10 POLYGON ((533764.39360000007 3378885.2949, 5338... 11 POLYGON ((533715.0513000004 3378913.3129999992,... 12 POLYGON ((533714.04150000028 3378943.6121999994... 13 POLYGON ((533722.20390000008 3378990.4656000007... 14 POLYGON ((533710.42669999972 3378958.2182, 5337... 15 POLYGON ((533709.91289999988 3379010.3861999996... 16 POLYGON ((533679.24089999963 3379001.0793999992... 17 POLYGON ((533699.3355 3379049.4602000006, 53373... 18 POLYGON ((533765.53440000024 3378988.7029999997... 19 POLYGON ((533764.07919999957 3379018.4685999993... 20 POLYGON ((533761.83019999973 3379049.5571999997... 21 POLYGON ((533762.35940000042 3379081.5899, 5338... 22 POLYGON ((533853.67030000035 3378986.2825000007... 23 POLYGON ((533854.8574000001 3379031.2216, 53391...

使用 STBoundary()查询外围线状的边界：

	空间结果型	文本型
sql 语句	SELECT geometry.STBoundary() FROM building	SELECT geometry.STBoundary().STAsText() FROM building
查询结果		(无列名) 1 LINESTRING (534060.7910000002 3378661.792099999... 2 LINESTRING (533826.14740000013 3378846.85390000... 3 LINESTRING (534008.97030000016 3378675.41750000... 4 LINESTRING (533914.81900000013 3378725.4168, 53... 5 LINESTRING (533920.19660000037 3378799.19820000... 6 LINESTRING (533957.33980000019 3378777.16719999... 7 LINESTRING (534223.07859999966 3378883.06990000... 8 LINESTRING (534188.8547 3378943.4060999993, 534... 9 LINESTRING (534004.76269999985 3378885.45890000... 10 LINESTRING (533788.04739999957 3378885.2949, 53... 11 LINESTRING (533735.90050000045 3378913.31299999... 12 LINESTRING (533714.83519999962 3378943.61219999... 13 LINESTRING (533743.89979999978 3378990.46560000... 14 LINESTRING (533723.43439999968 3378958.2182, 53... 15 LINESTRING (533716.93219999969 3379010.38619999... 16 LINESTRING (533704.74679999985 3379001.07939999... 17 LINESTRING (533737.96470000036 3379049.46020000... 18 LINESTRING (533813.8208999997 3378988.702999999... 19 LINESTRING (533808.00009999983 3379018.46859999... 20 LINESTRING (533805.61880000029 3379049.55719999... 21 LINESTRING (533805.55399999954 3379081.5899, 53... 22 LINESTRING (533912.3021 3378986.2825000007, 533... 23 LINESTRING (533910.8049999997 3379031.2216, 533... 24 LINESTRING (533940.64879999962 3378986.59029999...

7.2 计算每条道路的长度(STLength)

查询语言：

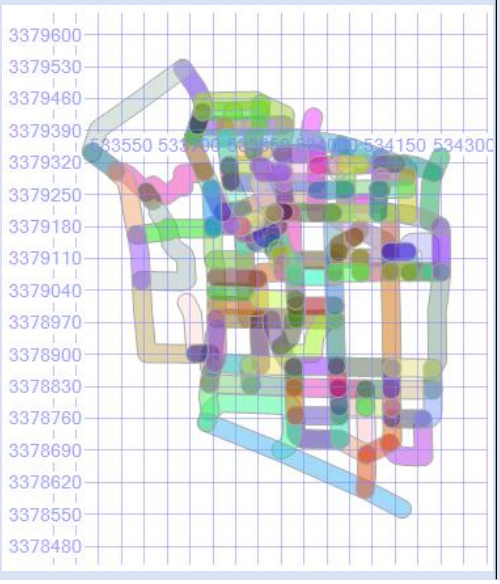
```
SELECT geometry.STLength()
From road
```

查询结果：

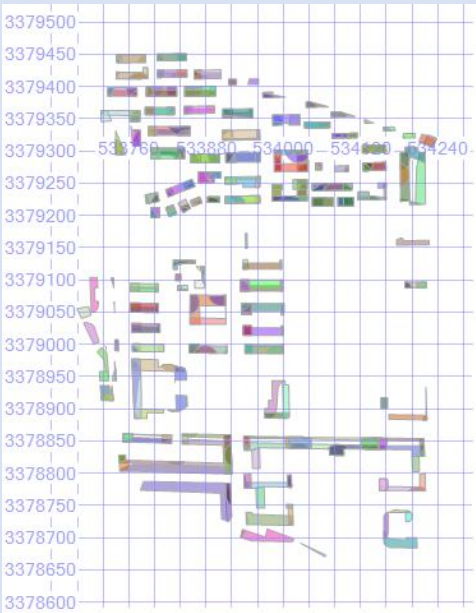
	(无列名)
1	87.5124682088926
2	210.231945680169
3	139.308015736704
4	171.015722489795
5	66.629035475561
6	58.229201599948
7	90.5223206296256
8	121.88340319142
9	196.199861359836
10	290.652783689703
11	202.319467603117
12	105.44168474433
13	260.193440091834
14	84.8244186750104
15	40.9029559134372
16	103.523021506493
17	96.5172018895234

图 12 长度查询结果

7.3 为每条道路构建缓冲区（STBuffer，缓冲距离为 20m）

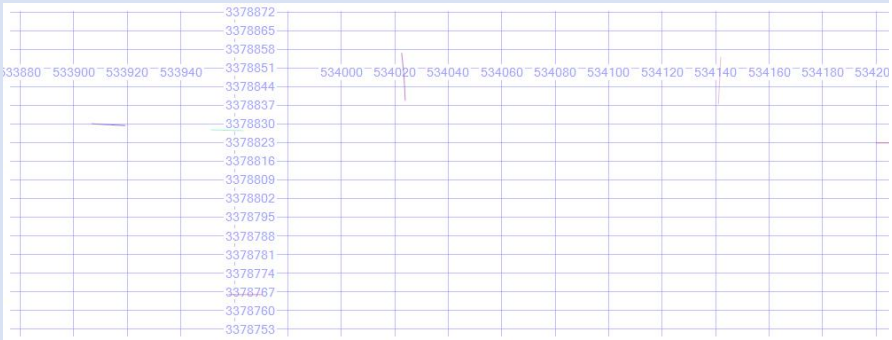
	空间结果型	文本型
sql语句	SELECT geometry.STBuffer(20) FROM road	SELECT geometry.STBuffer(20).STAsText() FROM road
查询结果		(无列名) 1 POLYGON ((534014.55101280613 3379102.6996167535... 2 POLYGON ((534142.6260404021 3378851.1033553197,... 3 POLYGON ((534028.39738429652 3378693.8347047027... 4 POLYGON ((534199.24170805968 3378656.8463119995... 5 POLYGON ((534209.42970227054 3378738.9202622557... 6 POLYGON ((534084.86064995523 3378764.7901006131... 7 POLYGON ((534232.030551172 3378802.83068367, 53... 8 POLYGON ((534142.59170890611 3378851.4593018102... 9 POLYGON ((533757.57271604589 3378959.9307000157... 10 POLYGON ((533899.30350223964 3378672.2627064027... 11 POLYGON ((533827.00425758108 3378958.6395000275... 12 POLYGON ((533926.2029393689 3379086.7219062094,... 13 POLYGON ((534017.78288217389 3378956.765234563,... 14 POLYGON ((533727.37388045772 3379433.8222129047... 15 POLYGON ((533749.98754163773 3379162.3869000385... 16 POLYGON ((534086.85987979139 3378661.2909146645... 17 POLYGON ((533930.61005784431 3378746.0639003911... 18 POLYGON ((533832.99188071548 3378813.4130002079... 19 POLYGON ((533854.6904167321 3378856.92400724, 5...

7.4 每条道路构建缓冲区后会影响到哪些建筑物？

	空间结果型	文本型
sql 语句	SELECT b.geometry.STIntersection(r.geometry.STBuffer(20)) FROM building b,road r WHERE b.geometry.STIntersects(r.geometry.STBuffer(20))=1	SELECT b.geometry.STIntersection(r.geometry.STBuffer(20)).STAsText() FROM building b,road r WHERE b.geometry.STIntersects(r.geometry.STBuffer(20))=1
查询结果		(无列名) 1 POLYGON ((534064.95227140374 3378670.6083516791... 2 POLYGON ((533826.14740000013 3378846.8539000005... 3 POLYGON ((533826.14740000013 3378846.8539000005... 4 POLYGON ((533826.14740000013 3378846.8539000005... 5 POLYGON ((533826.47930232808 3378857.980838052,... 6 POLYGON ((533826.48488275846 3378858.1679205084... 7 POLYGON ((533751.98451817362 3378849.0658120029... 8 POLYGON ((533758.17455762 3378848.881193738, 53... 9 POLYGON ((533764.16248729988 3378848.7026034081... 10 POLYGON ((533751.95189321111 3378849.066785044,... 11 POLYGON ((533753.07851507945 3378860.9904043656... 12 POLYGON ((534014.56525893707 3378694.0954522621... 13 POLYGON ((533935.6518119711 3378700.3530666996,... 14 POLYGON ((534015.824602088 3378698.2995852176, ... 15 POLYGON ((534015.82350288564 3378698.2959156912... 16 POLYGON ((533914.81900000013 3378725.4168, 5339... 17 POLYGON ((533914.81900000013 3378725.4168, 5339... 18 POLYGON ((533918.8621744893 3378749.8809959362,... 19 POLYGON ((533918.86234678573 3378749.87632503, ... 20 POLYGON ((533920.19660000037 3378799.1982000005... 21 MULTIPOLYGON (((533852.21707279037 3378845.4846...

7.5 查询道路穿越建筑物的部分(STIntersection)

	空间结果型	文本型
sql 语句	SELECT b.geometry.STIntersection(r.geometry) FROM building b,road r WHERE b.geometry.STIntersects(r. geometry)=1	SELECT b.geometry.STIntersection(r.geometry).S TAsText() FROM building b,road r WHERE b.geometry.STIntersects(r.geometry)=1

空间结果型 查询结果	
文本型 查询结果	

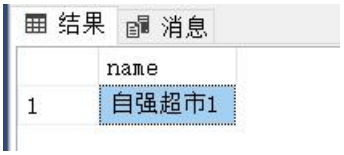
	(无列名)
1	LINESTRING (533906.74541687779 3378830.084748779, 533919.1427432627 3378829.4448742103)
2	LINESTRING (533951.3434461077 3378827.823534546, 533963.24734273506 3378827.543813807)
3	LINESTRING (534200.02393197757 3378822.9342217366, 534211.04169051431 3378822.8984870482)
4	LINESTRING (534141.81712612708 3378855.1039086804, 534140.9785029731 3378837.4927784274)
5	LINESTRING (534022.54004994163 3378856.5981672173, 534023.38399999961 3378848.2433, 534023.8867591999 3378
6	LINESTRING (533970.40569678659 3378766.0717760292, 533957.79839049466 3378766.0692811417)

7.6 找出距离“4 号教学楼”最近的超市和最近的医院（STDistance）

1) 找出距离“4 号教学楼”最近的超市，sql 语言：

```
SELECT m.name
FROM (SELECT geometry FROM building WHERE name='4 号楼') b,market m
WHERE b.geometry.STCentroid().STDistance(m.geometry)<
ALL(
SELECT b.geometry.STCentroid().STDistance(m2.geometry)
FROM market m2
WHERE m.name<>m2.name
)
```

查询结果：



	name
1	自强超市1

图 13 距离“4 号教学楼”最近的超市

2) 找出距离“4 号教学楼”最近的医院，sql 语言：

```
SELECT h.name
FROM (SELECT geometry FROM building WHERE name='4 号楼') b,hospital h
WHERE b.geometry.STCentroid().STDistance(h.geometry)<
ALL(
SELECT b.geometry.STCentroid().STDistance(h2.geometry)
FROM hospital h2
WHERE h.name<>h2.name
)
```

查询结果：

	name
1	信息学部校医院

图 14 距离“4 号教学楼”最近的医院

8.在 ArcGIS 中加载数据库

首先在 ArcGIS 中新建一个空的 mxd，然后双击添加数据库连接，输入本机数据库实例名 DESKTOP-UF9IFNT，选择身份验证为数据库身份验证或本机验证，选择数据库为 WHU。点击确定，就能将所有的表导入。

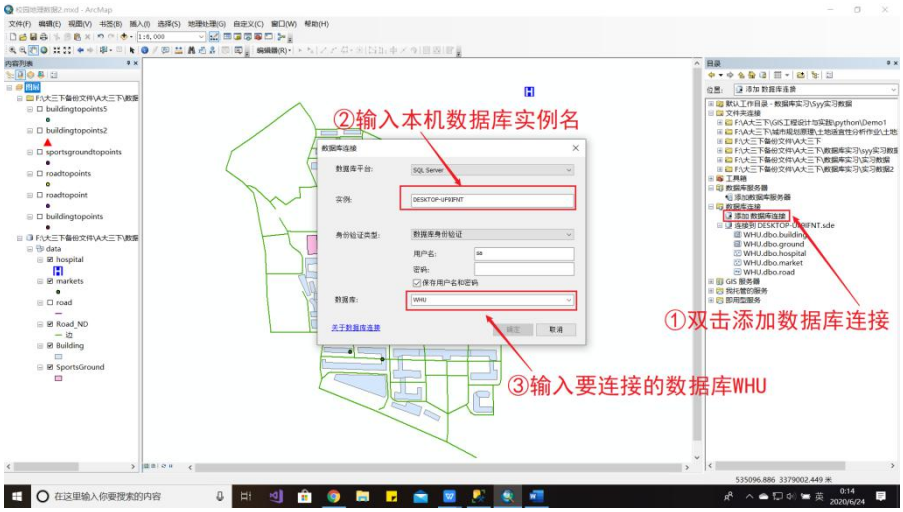


图 15 导入数据库

将所有的表拖到视图中加载，为了让地图更加美观，通过右键属性修改符号样式。

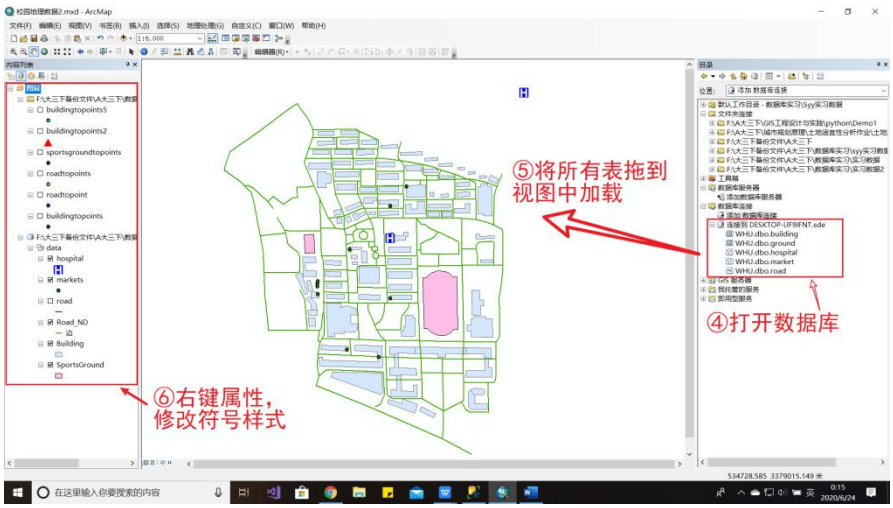


图 16 插入数据并可视化

9.实习成果展示

最终，我们可以在 ArcGIS 中右键“building”和“ground”，点击“标注要素”，将建筑物和运动场的名称标注在地图上，最终成果如下图所示：

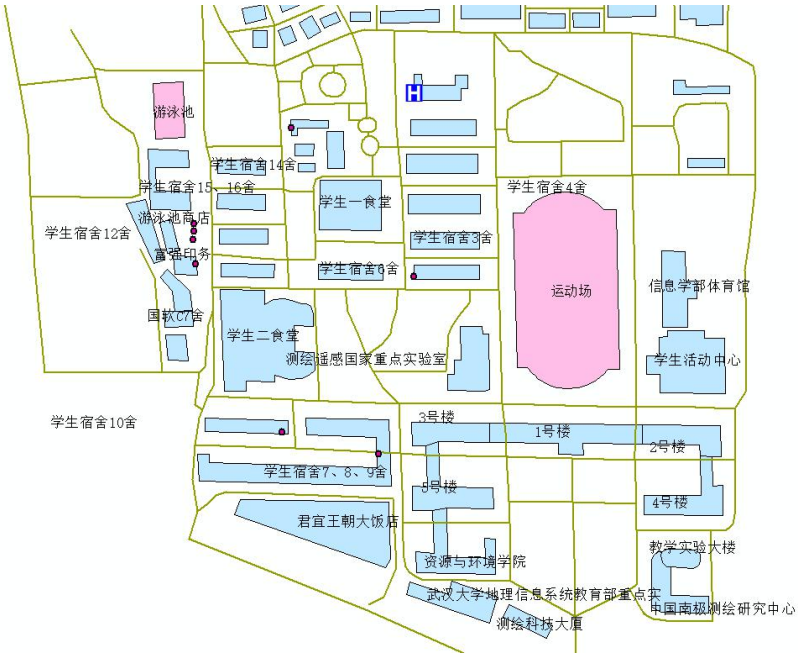


图 17 最终成果展示图（含标注）

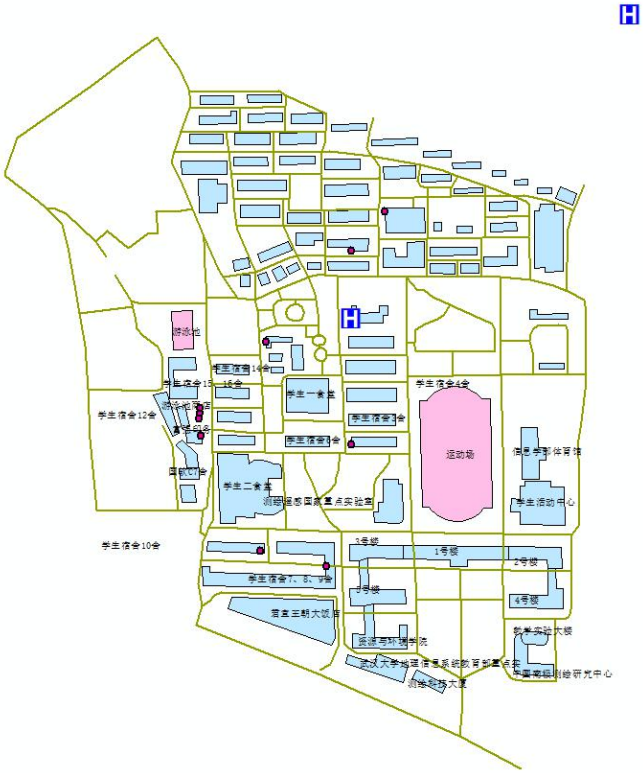


图 18 最终成果展示图（全图）

三、实习中遇到的问题及解决方案

在本次实习的过程中，主要运用了 Microsoft SQL Server Management Studio 和 ArcGIS，语言方面使用的是 python 和 sql 语言。作为 D 方向的学生，大三下学期陆续接触了很多数据库和 ArcGIS 方面的操作，因此实习方面的基础操作并没有很大的难度，主要难点在于数据表的概念设计、逻辑设计以及数据库建表。

在数据库的概念设计和逻辑设计之初，我首先在 ArcGIS 中加载了本次实习的数据底图，然后一个个打开属性表观察表的结构和列名。我发现，超市和医院是点数据，它们都有 XY 坐标这两列；道路是线类型，它没有具体的 XY 坐标，但是有长度这一属性，以及道路类型；建筑物和运动场是面类型，它也没有 XY 坐标，但是有长度和面积这些属性。因此，在进行概念设计的时候，需要明确哪些字段是需要的，哪些是可以去掉的，哪些实体之间有怎样的关联，这些都是要去注意的。

在弄清楚五张表的字段之后，我在草稿纸上画了一张简单的概念设计图，然后便开始进行读取 shp 文件的编程了。我通过百度，查阅了一些博客，了解了 python 读取 shp 文件主要依赖的是 shapefile 库，且有 shapes()、ShapeRecords()、fields 等函数能帮助我们快速地将 shp 文件中的几何对象、属性值记录和字段名提取出来，大大简化了代码量。

第二个难点就在于将 shp 文件读取并转化成 sql 语言应用到数据库中。一开始我对超市和医院的点数据进行的读取，由于文件编码问题，直接读取老师所给的 shp 文件会报错，所以要自己手动将这些 shp 文件再导出一次。一开始我并不知道可以直接用 shapes()函数接口，所以我选择的是读取 XY 坐标并插入到一个 list 列表里，在写 insert 语句是将 list 赋值给 values。然而，对于点数据这种做法还行得通，但是对于线数据和面数据，它们没有 XY 坐标，这种做法不可行。我本来是想按照老师的思路，读取 buildingtopoints4 这个文件的，里面有面数据所有点的坐标和 name，但是我感觉一个个读点进来很麻烦，而且后续还要写到 POLYGON 里，那些堆叠的括号很容易出错。所以，我转念一想，既然 building 是面数据，而且 building 的属性里有一列是 SHAPE（面），因为我充分有理由怀疑这个 SHAPE 里面其实暗藏了几何坐标信息，只是我们看不到罢了。于是我继续百度，最后发现了 shapes.points 这个函数，能够直接把矢量数据的几何数据

坐标直接读取出来，而且是列表的形式，方便使用。

在学会了 shapes.points 这个函数的使用之后，我直接就把线数据和面数据导出成 shp，然后直接读取，没有用到 roadtopoints、buildingtopoints 这些点坐标表，也让程序更加简洁和方便。

因为之前了解过 python 语言连接数据库并写入的操作，所以我并没有把读取的数据写到 csv 或 txt 里再去数据库调用，我直接用 pymssql 库的 connect 函数连上了数据库，并用 consor.execute(sql)把 sql 放到数据库里去执行，这样我的整个建表操作都是用 python 完成的，建表和插值都是通过代码直接完成，不需要在数据库进行任何操作（除了一开始的建库）。

在将所有的表导入之后，我首先根据实际情况修改了我的 ER 图，并绘制了相应的逻辑设计表，有三张表是根据已有的表生成的，所以我又写了 sql 代码将 building 表和其他的表一起查询，生成距离量测关系表和交叉关系表。最后，将所有的表导入 ArcGIS 中进行加载和可视化，就完成了整个实习的任务。

四、实习体会与心得

******

五、附件说明

在上交的文件夹中，共有 2 个文件夹：

1.地理数据

含“校园地理数据最终版.mxd”，是将数据库中建的表加载到 ArcGIS 后呈现的最终结果。

2.代码数据

其中 read_shp.py 是读取文件的代码，其中已经将 path 改成相对路径，老师可以直接运行程序（前提是建好库）。空间查询语言.sql 对应空间查询操作中的所有解答。关系查询 id 表.sql 是距离量测表（医院、超市）和交叉关系表（道路）建立过程对应的代码。

3.空间数据库实习报告.pdf

实习报告 pdf 版。

4.空间数据库实习报告.word

实习报告 word 版。

综合评语：

平时成绩		所占比例	30%
报告成绩		所占比例	40%
考核成绩		所占比例	30%
总评成绩			
指导教师： 年　月　　日			