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](#_Toc14823)

[1.实习目的 1](#_Toc23502)

[2.实习要求 1](#_Toc11683)

[3.实习安排 1](#_Toc16756)

[二、实习内容与过程 2](#_Toc6727)

[1.环境配置和数据准备 2](#_Toc622)

[2.空间数据库概念设计 3](#_Toc9157)

[3.空间数据库逻辑设计 4](#_Toc23879)

[3.1道路对象关系表 4](#_Toc5983)

[3.2超市对象关系表 5](#_Toc14254)

[3.3医院对象关系表 5](#_Toc9135)

[3.4建筑物对象关系表 5](#_Toc3797)

[3.5运动场对象关系表 6](#_Toc15894)

[3.6交叉关系表 6](#_Toc9131)

[3.7距离量算关系表 6](#_Toc2193)

[4.数据库建立 7](#_Toc2522)

[5.数据表建立 7](#_Toc25412)

[6.数据插入 8](#_Toc14251)

[6.1 sql语言插入 8](#_Toc28467)

[6.2 python语言批量插入 9](#_Toc26191)

[6.3创建查询关系表 12](#_Toc13324)

[7.空间查询操作 12](#_Toc19734)

[7.1查询每栋建筑物的边界（STEnvelope） 12](#_Toc26855)

[7.2计算每条道路的长度(STLength) 14](#_Toc22418)

[7.3为每条道路构建缓冲区（STBuffer，缓冲距离为20m） 14](#_Toc8300)

[7.4每条道路构建缓冲区后会影响到哪些建筑物？ 15](#_Toc13763)

[7.5查询道路穿越建筑物的部分(STIntersection) 15](#_Toc23229)

[7.6找出距离“4号教学楼”最近的超市和最近的医院（STDistance） 16](#_Toc24257)

[8.在ArcGIS中加载数据库 17](#_Toc21522)

[9.实习成果展示 18](#_Toc8565)

[三、实习中遇到的问题及解决方案 19](#_Toc23797)

[四、实习体会与心得 20](#_Toc31846)

[\*\*\*\*\*\* 20](#_Toc12657)

[五、附件说明 20](#_Toc31910)

[1.地理数据 20](#_Toc11962)

[2.代码数据 20](#_Toc9562)

[3.空间数据库实习报告.pdf 21](#_Toc789)

[4.空间数据库实习报告.word 21](#_Toc11395)

# 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。

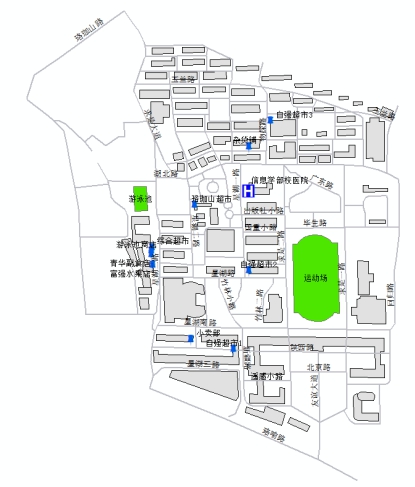


图 1 实习数据图

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：

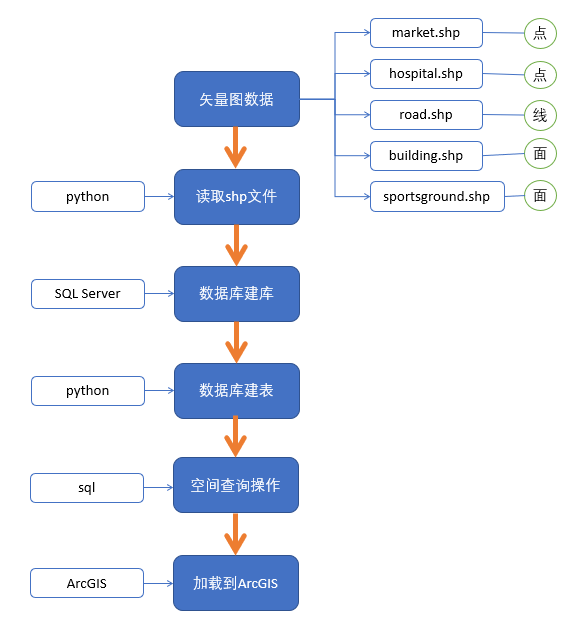


图 2 流程图

## 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。

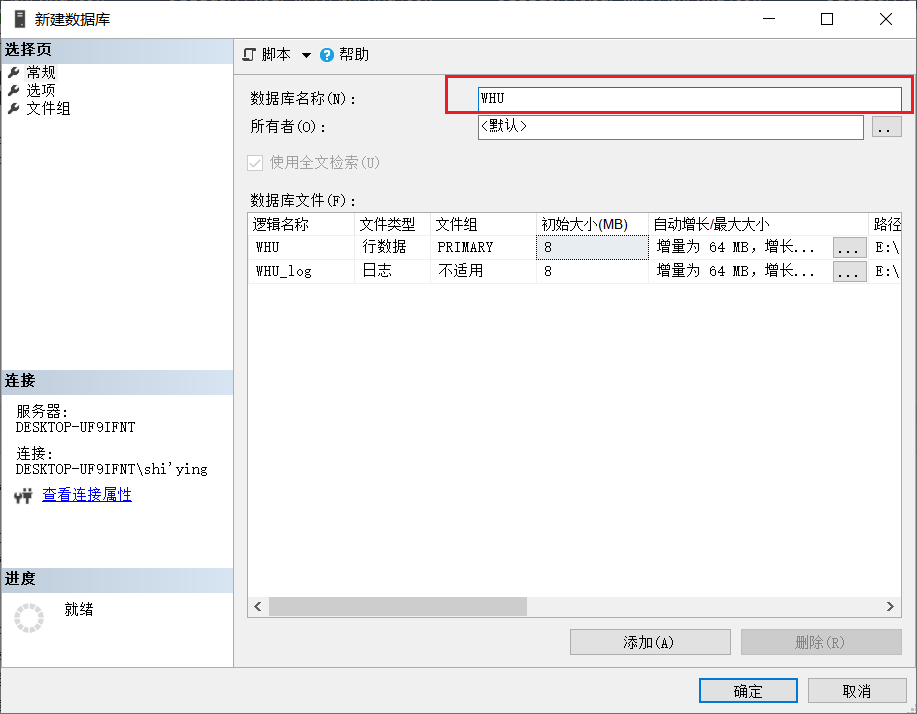


图 4 新建数据库WHU

## 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**

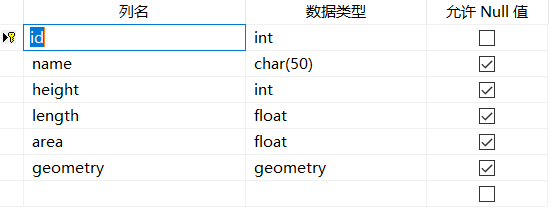


图 5 操作界面建表

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。

1. **sql语言建表**

以market表为例：

1. **CREATE** **TABLE** market
2. (id **int** **PRIMARY** **KEY**,
3. **name** **char**(50),
4. x **float**,
5. y **float**,
6. geometry geometry
7. );

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。

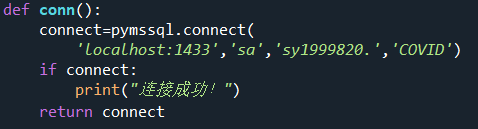


图 6 数据库连接函数

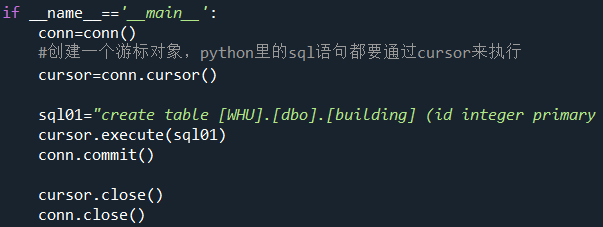


图 7 数据库连接主函数和sql语言

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：

1. **INSERT** **INTO** **table**
2. **VALUES**(X,X,…)；

以超市的表为例，展示插入数据的具体sql语句：

1. **INSERT** **INTO** market
2. **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。

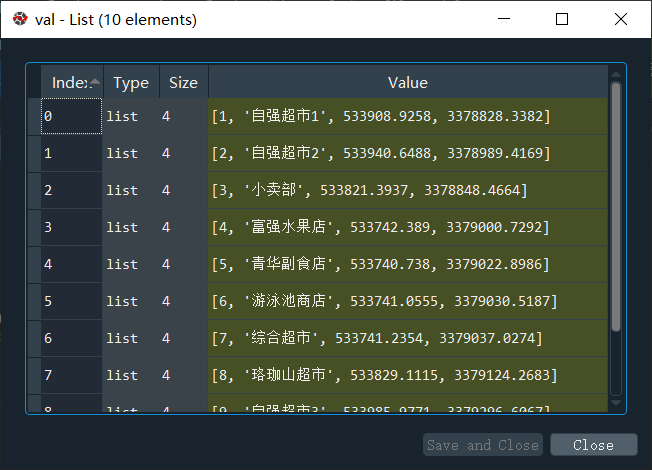


图 8 属性数据

1. **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。

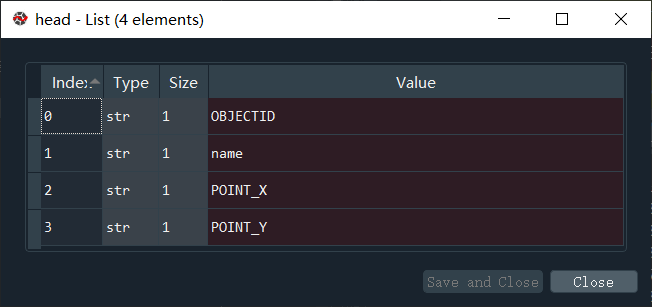


图 9 获取字段名

1. **创建表并输入数据**
2. **def** market\_sql():
3. market="market.shp"
4. val,typ,head=read\_shp(market)
6. sql01="create table [WHU].[dbo].[market] (id integer primary key,name char(50),x float,y float,geometry geometry)"
7. cursor.execute(sql01)
8. conn.commit()
10. **for** index **in** val:
11. 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])
12. cursor.execute(sql02)
13. conn.commit()
14. **查看数据表**

结果：

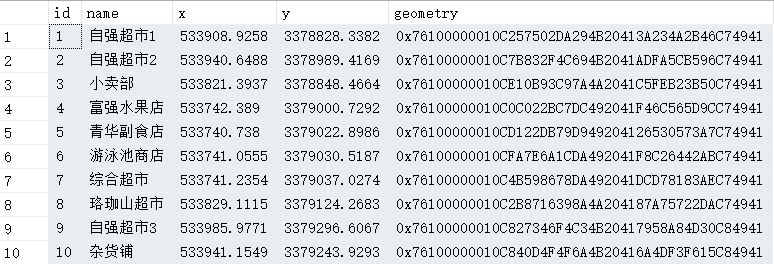


图 10 market查询结果

空间结果：

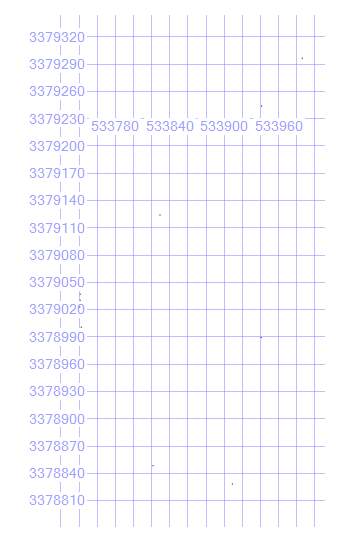


图 11 market空间查询结果

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）
2. **SELECT** \* **INTO** [WHU].[dbo].hospital\_dis\_id **FROM**
3. (**SELECT** b.id building,h.id hospital,b.geometry.STCentroid().STDistance(h.geometry) distance
4. **FROM** [WHU].[dbo].building b,[WHU].[dbo].hospital h
5. )s
6. 距离量测关系表（超市）
7. **SELECT** \* **INTO** [WHU].[dbo].market\_dis\_id **FROM**
8. (**SELECT** b.id building,m.id market,b.geometry.STCentroid().STDistance(m.geometry) distance
9. **FROM** [WHU].[dbo].building b,[WHU].[dbo].market m
10. )s2
11. 交叉关系表（道路）
12. **SELECT** \* **INTO** [WHU].[dbo].road\_cross\_id **FROM**
13. (**SELECT** b.id building,r.id road,b.geometry.STIntersection(r.geometry) intersection
14. **FROM** [WHU].[dbo].building b,[WHU].[dbo].road r
15. )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 |  |  |

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 |  |  |

### 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：

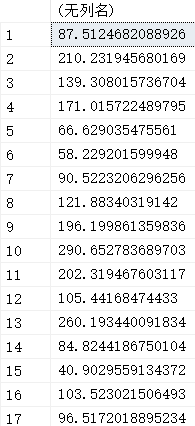


图 12 长度查询结果

### 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 |  |  |

### 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 |  |  |

### 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).STAsText()  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 |  | |

### 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：
2. **SELECT** m.**name**
3. **FROM** (**SELECT** geometry **FROM** building **WHERE** **name**='4号楼') b,market m
4. **WHERE** b.geometry.STCentroid().STDistance(m.geometry)<
5. ALL(
6. **SELECT** b.geometry.STCentroid().STDistance(m2.geometry)
7. **FROM** market m2
8. **WHERE** m.**name**<>m2.**name**
9. )

查询结果：

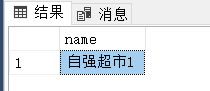


图 13 距离“4号教学楼”最近的超市

1. 找出距离“4号教学楼”最近的医院，sql语言：
2. **SELECT** h.**name**
3. **FROM** (**SELECT** geometry **FROM** building **WHERE** **name**='4号楼') b,hospital h
4. **WHERE** b.geometry.STCentroid().STDistance(h.geometry)<
5. ALL(
6. **SELECT** b.geometry.STCentroid().STDistance(h2.geometry)
7. **FROM** hospital h2
8. **WHERE** h.**name**<>h2.**name**
9. )

查询结果：

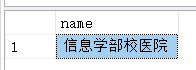


图 14 距离“4号教学楼”最近的医院

## 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。

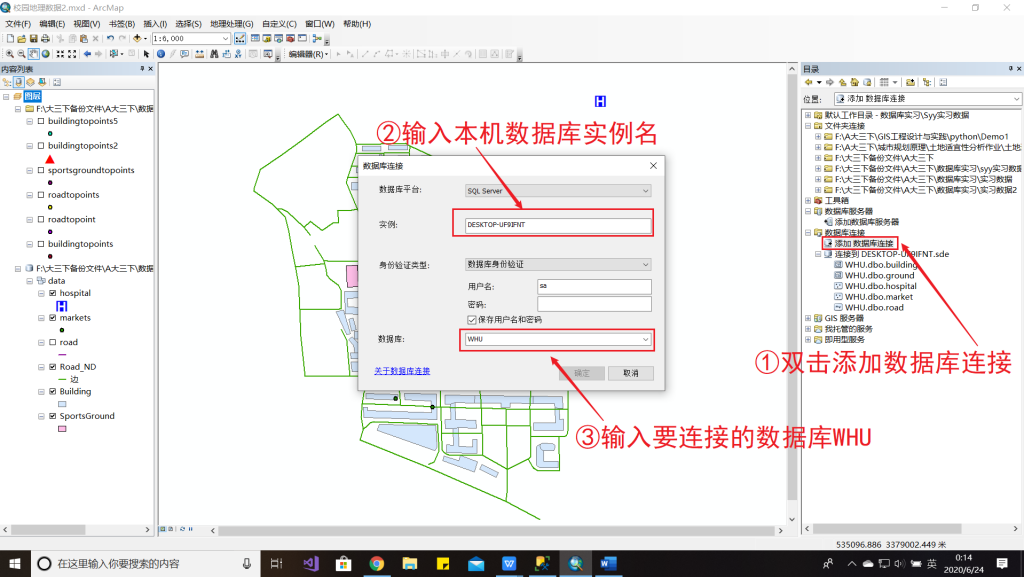


图 15 导入数据库

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。

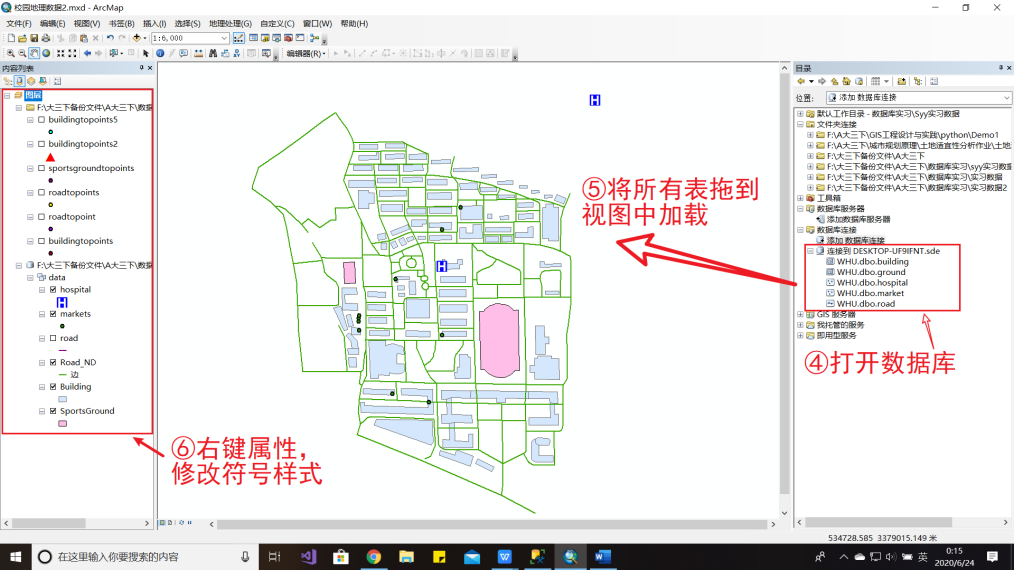


图 16 插入数据并可视化

## 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：

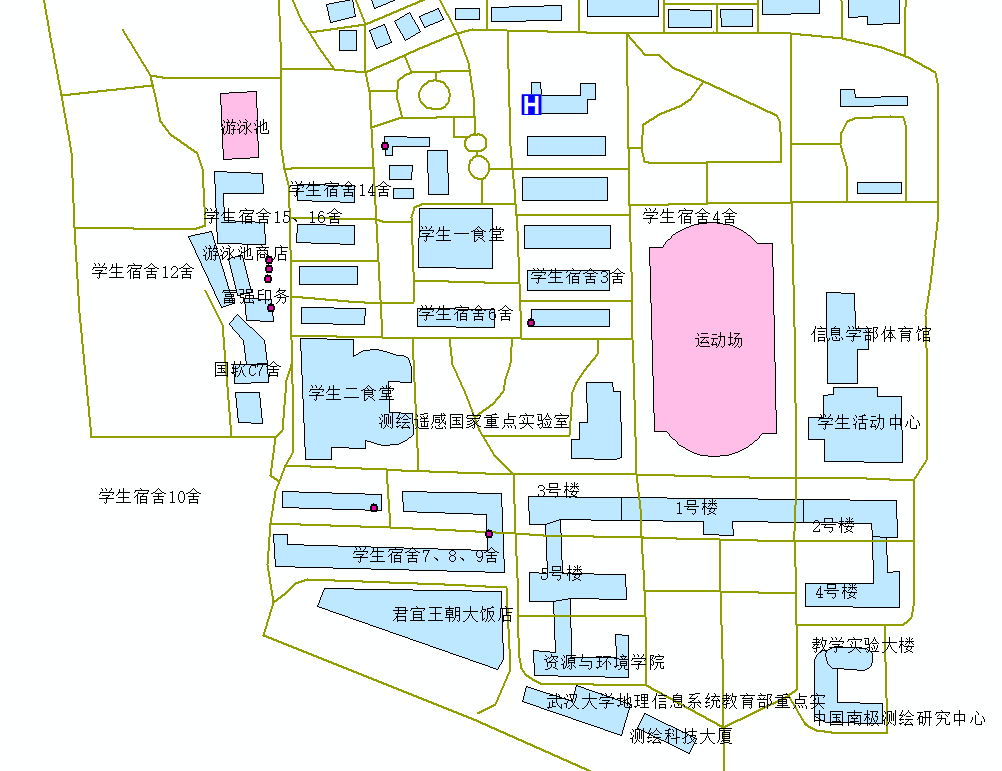


图 17 最终成果展示图（含标注）

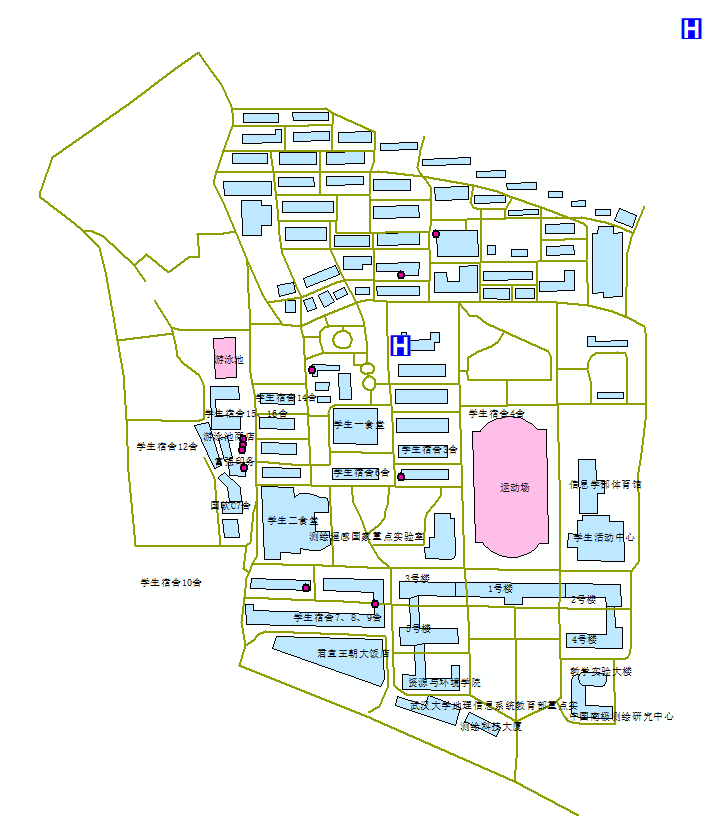


图 18 最终成果展示图（全图）

# 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** | | | |