

《时空数据处理与组织》

课程期末作业报告

课程名称： 时空数据处理与组织

学 院： 遥感信息工程学院

专业年级： 2020 级空间信息与数字技术

学 号： 2020302131201

姓 名： 常耀文

成 绩：

指导教师： 周松涛

2022 年 5 月 15 日

目录

- 《时空数据处理与组织》 1
- 课程期末作业报告 1
 - (一) 课程设计要求与数据 3
 - 1. 课程设计要求 3
 - 2. 课程设计数据 5
 - 3. 对于课程设计数据的处理 6
 - 4. 对于课程设计数据与课程设计使用数据库功用的思考 8
 - (1) 采用带有全世界县级行政区的 gdam 数据; 8
 - (2) Postgresql 数据库 8
 - (3) MongoDB 数据库; 9
-  10
- 图四 (MongoDB 数据库) 10
- (二) 课程设计的主要操作步骤 10
 - 1. 实习的具体操作与关键部分截图展示 10
- (三) 课程设计心得与感悟 35

（一）课程设计要求与数据

1. 课程设计要求

（1）在同一台机器上，分别安装 postgis 和 mongodb 服务器程序，并能正确启动，然后导入 gdam 数据：

- ①说明安装过程，并创建 test 数据库，用自己名字的全拼作为导入数据的表名（postgis）和集合名（mongodb）；
- ②导入方法不限，但需要说明从原始数据导入到两个数据库的整个过程；
- ③postgis 导入过程可直接建立空间索引，mongodb 先不要建立索引；
- ④报告中应在本部分反映出实验的软硬件环境，例如设备的 CPU，内存，操作系统版本等

（2）性能及索引功能测试：

- ①分别在 postgis 及 mongodb 上，检索西经 60 度的经线穿过的所有国家。注意，数据是精确到每个国家的县级数据的，因此对查询结果需要进行综合过滤，只给出国家的名字，并统计检索的时间。多做几次检索（5 次以上），时间取平均值，然后比较 postgis（带空间索引）和 mongodb（无空间索引）的检索效率；
- ②将 mongodb 中的数据建立空间索引，然后重复①中的检索，并统计检索的平均时间，与①中的时间做比较；

③所有时间的统计信息最终放在一张表格当中进行统计，统一做性能比较分析；

④所有操作可命令行执行，也可用程序实现，但都要给出截图并详细说明。

(3) 利用程序，基于 postgis 数据库现如下功能并进行测试：

①实现一个函数，输入任意一个国家，统计出这个国家的县级行政区的数量（即一个国家数据中包含的要素数量），并统计这个国家中县级行政区面积最大的三个，打印出名字和面积值；

②实现一个函数，输入任意一个国家，统计出这个国家在地域上邻接的其它国家（需要做数据的综合，因为只有县级数据）；

③实现一个数据处理函数，函数接受三个参数：国家名，像素分辨率，输出目录。输入任意一个国家，将这个国家的数据按指定像素分辨率进行渲染（像素分辨率即一个像素代表的实际地理范围的大小，宽和高等分辨率），然后按 256x256 像素大小的索引格网，对指定国家的渲染后栅格进行切分，切成多个正方形的影像数据并写入到输出目录中（这个过程也称为切片）。需要考虑对应国家的最大外接矩形边界，然后按比例生成图片，每个小的行政区域随机指定一种颜色，所有县级边界用黑线隔开。输出切片影像数据的命名采用索引方式，例如，最左上角的切片文件名为 1-1，其下的为 2-1，右侧的为 1-2，以此类推。如果输出数据格式为 jpg 等不带有坐标信息的栅格数据，则根据每个切片文件的数据坐标参数，生成对应的坐标配准文件，例如，jpg 文件要输出对应的 jgw 文件，最终在 qgis 中打开全部切片数据，综合显示并截图；






④每个函数单独实现，并且分别用不同国家测试至少 2 次，③中的分辨率参数用 0.1 度和 0.2 度分别测试。所有实现方法和步骤，在报告中要详细说明。

(4) 利用程序，基于 **mongodb** 数据库实现如下功能并测试：

- ①实现一个函数，给定一个经纬度坐标及编码长度，则生成一个 geohash 编码；
- ②实现一个函数，给定一个 mongodb 的县级行政区名字作为查询条件，即可查询出某个县级行政区域，然后，将这个县级行政区域的几何边界用 8 位 geohash 编码数据组织成为一大段文本进行描述（相邻顶点之间不留空格，连续存放编码）；
- ③测试程序调用上述两个功能函数，并用“湖北省鄂州市鄂城市”的数据进行测试，打印出结果（注意：用属性从省级定位到县级行政区需要三级，本题中，相当于找 湖北省->鄂州市->鄂城市）。

2. 课程设计数据

(1) gadm 数据, 要求是带有全世界县级行政区的数据

 gadm404.cpg	CPG 文件	1 KB	否	1 KB	0%	2022/2/20 11:27
 gadm404.dbf	DBF 文件	10,265 KB	否	731,883 KB	99%	2022/2/20 11:35
 gadm404.prj	PRJ 文件	1 KB	否	1 KB	17%	2022/2/20 11:27
 gadm404.shp	SHP 文件	1,087,502 KB	否	2,002,054 KB	46%	2022/2/20 11:35
 gadm404.shx	SHX 文件	1,724 KB	否	2,726 KB	37%	2022/2/20 11:35

图一（gadm404. json 用于 postgis）

gadm404.json 5,468,808,... 950,732,0... JSON 文件 2022/5/8 15:14 15267413

图二 (gadm404. json 用于 mongodb)

3. 对于课程设计数据的处理

(1) 在 mongodb 中处理数据时需要对数据进行处理, 去掉 json 文件的首尾, 然后才可以按照正常操作进行相关的数据操作, 与之前的作业情况相同, 在操作此题时, 群里的同学已经将处理过的 json 文件进行了相关的处理, 去掉了头尾, 而且数据中存在大于 16m 的数据, 与同学讨论后发现大于 16m 的数据是在后续步骤中西经 60 度的南极洲, 由于南极洲不是一个国家, 所以没有导入到数据库中, 在实现相关操作的时候跳过了导入 16m 的数据。

(2) 对于 shp 类型数据的理解:

①介绍

Shapefile (shp) 是一种空间数据开放格式, 用于描述几何体对象: 点、折线与多边形。

②存储方法

Shapefile 文件指的是一种文件存储的方法, 实际上该种文件格式是由多个文件组成的。其中, 要组成一个 Shapefile, 有

三个文件是必不可少的，它们分别是".shp"（图形格式），".shx"（图形索引格式）与 ".dbf"（属性数据格式）文件，表示同一数据的一组文件其文件名前缀应该相同。例如，存储一个关于湖的几何与属性数据，就必须有 lake.shp, lake.shx 与 lake.dbf 三个文件。而其中“真正”的 Shapefile 的后缀为 shp，然而仅有这个文件数据是不完整的，必须要把其他两个附带才能构成一组完整的地理数据。除了这三个必须的文件以外，还有八个可选的文件，使用它们可以增强空间数据的表达能力。此外，所有的文件都必须位于同一个目录之中。

（3）对于 json 类型数据的理解：

①介绍

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于 JavaScript 的一个子集。JSON 采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯（包括 C, C++, C#, Java, JavaScript, Perl, Python 等）。这些特性使 JSON 成为理想的数据交换语言。易于人阅读和编写，同时也易于机器解析和生成。

4. 对于课程设计数据与课程设计使用数据库功用的思考

(1) 采用带有全世界县级行政区的 gdam 数据;

- 带有全世界县级行政区的 gdam 数据是本次实习的基础，通过带有县级行政区的 gdam 数据，可以完成后续课程设计的多种操作，比如查询特定经纬线穿过的国家，邻近的国家以及对于行政区进行渲染等操作。

(2) Postgresql 数据库

- PostgreSQL 是一种特性非常齐全的自由软件的对象-关系型数据库管理系统 (ORDBMS)，是以加州大学计算机系开发的 POSTGRES，4.2 版本为基础的对象关系型数据库管理系统。POSTGRES 的许多领先概念只是在比较迟的时候才出现在商业网站数据库中。PostgreSQL 支持大部分的 SQL 标准并且提供了很多其他现代特性，如复杂查询、外键、触发器、视图、事务完整性、多版本并发控制等。同样，PostgreSQL 也可以用许多方法扩展，例如通过增加新的数据类型、函数、操作符、聚集函数、索引方法、过程语言等。另外，因为许可证的灵活，任何人都可以以任何目的免费使用、修改和分发 PostgreSQL。在 PostgreSQL 中已经定义了一些基本的集合实体类型，这些类型包括：点 (POINT)、线 (LINE)、线段 (LSEG)、方形 (BOX)、多边形 (POLYGON) 和圆 (CIRCLE)

等;另外, PostgreSQL 定义了一系列的函数和操作符来实现几何类型的操作和运算;同时, PostgreSQL 引入空间数据索引 R-tree。尽管在 PostgreSQL 提供了上述几项支持空间数据的特性,但其提供的空间特性很难达到 GIS 的要求,主要表现为缺乏复杂的空间类型;没有提供空间分析;没有提供投影变换功能。为了使得 PostgreSQL 更好的提供空间信息服务, PostGIS 应运而生。在本次课程设计中我们主要使用 PostGIS 数据库。其中 postgresql 还有其可视化工具 pgAdmin4, 可以很好地显示数据库的连接情况。



图三 (postgresql 数据库)

(3) MongoDB 数据库;

- MongoDB 是一个基于分布式文件存储 [1] 的数据库。由 C++ 语言编写。旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。MongoDB 是一个介于关系数据库和非关系数据库之间的产品,是非关系数据库当中功能最丰富,最像关系数据库的。它支持的数据结构非常松散,是类似 json 的 bson 格式,因此可以存储比较复杂的数据类型。Mongo 最大的特点是它支持的查询语言非常强大,其语法有点类似于面向对象的查询语言,几乎可以实

现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。



图四（MongoDB 数据库）

（二）课程设计的主要操作步骤

1. 实习的具体操作与关键部分截图展示

由于本次实习对于编程语言没有要求，而经过之前的课程作业后，发现 Python 在数据库管理与数据库操作方面具有强大的能力，因此此次编程采用 Python 实现。具体实习的要求与步骤如下所示。

（1）在同一台机器上，分别安装 postgis 和 mongodb 服务器程序，并能正确启动，然后导入 gdam 数据

① 根据老师所给的链接，从 <https://gadm.org/> 网站上下载了带有全世界的县级行政区的 gdam.shp 文件，并且在之前作业的基础上已经安装了 postgresql 与 postgis 和 mongodb 数据库。数据库的安装过程如下（由于 mongodb 在之前做作业时是安装在 windows 环境下的，而 postgresql 是在 linux 环境下的，而本次实验是在 windows 环境下完成的，postgresql 下载安装的过程只保

留了第一次作业的截图，windows 下载安装没有截图，所以这里展示一下 linux 下 postgresql 的安装过程，mongodb 只显示开启服务）：

```
root@hoping-virtual-machine:/home/hoping/桌面# sudo apt-get install postgresql
postgresql-client
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
将会同时安装下列软件：
 libasn1-8-heimdall libgssapi3-heimdall libhcrypto4-heimdall
 libheimbase1-heimdall libheimntlm0-heimdall libhx509-5-heimdall libicu60
 libkrb5-26-heimdall libldap-2.4-2 libpq5 libreadline7 libroken18-heimdall
 libssl1.1 libtinfo5 libwind0-heimdall postgresql-10 postgresql-client-10
 postgresql-client-common postgresql-common
建议安装：
 postgresql-doc locales-all postgresql-doc-10 libjson-perl
推荐安装：
 sysstat
下列【新】软件包将被安装：
 libasn1-8-heimdall libgssapi3-heimdall libhcrypto4-heimdall
 libheimbase1-heimdall libheimntlm0-heimdall libhx509-5-heimdall libicu60
 libkrb5-26-heimdall libldap-2.4-2 libpq5 libreadline7 libroken18-heimdall
 libssl1.1 libtinfo5 libwind0-heimdall postgresql postgresql-10
 postgresql-client postgresql-client-10 postgresql-client-common
 postgresql-common
```

图五（安装 postgresql 数据库）

```
root@hoping-virtual-machine:/home/hoping/桌面# sudo -i -u postgres
postgres@hoping-virtual-machine:~$ psql
psql (10.20 (Ubuntu 10.20-0ubuntu0.18.04.1))
Type "help" for help.

postgres=# \q
```

图六（测试安装是否成功）

```
hoping@hoping-virtual-machine:~/桌面$ su
密码：
root@hoping-virtual-machine:/home/hoping/桌面# sudo apt-cache search postg
resql
```

图七（查看 postgresql 对应的 postgis 版本）

```
E:\MogonDB\bin>net start MongoDB
MongoDB Server (MongoDB) 服务正在启动 .
MongoDB Server (MongoDB) 服务已经启动成功。
```

图八（启动 MongoDB 服务）

② 创建数据库，首先创建了一个 Administrator 的用户，然后使用命令 `createdb -h localhost -U Administrator -p 5432 test` 创建名为 test 的数据库。

```
C:\Users\Administrator>createdb -h localhost -U Administrator -p 5432 test
```

图九（创建名为 test 的数据库）



图十（为创建的用户赋予权限（pgadmin 中））

③ 为数据库添加 postgis 扩展，使用命令：

CREATE EXTENSION postgis;

CREATE EXTENSION postgis_raster;

使用命令后数据库支持了处理空间数据的能力

④ 利用 ogr2ogr 命令将 gadm404.shp 导入到 test 数据库中 changyaowen 表中，具体命令如下：

***ogr2ogr -f PostgreSQL -nlt MULTIPOLYGON PG: "host=localhost port=5432
user=Administrator dbname=test schemas=public" -nln changyaowen
D:/data/gadm404.shp -progress***

```
C:\Users\Administrator>ogr2ogr -f PostgreSQL -nlt MULTIPOLYGON PG:"host=localhost port=5432 user=Administrator dbname=test schemas=public" -nln changyaowen D:/data/gadm404.shp -progress
0...10...20...30...40...50...60...70...80...90...100 - done.

C:\Users\Administrator>
```

图 11（将数据导入到 test 数据库）

⑤ 在 pgAdmin4 中查看导入结果并与 QGIS 连接显示结果

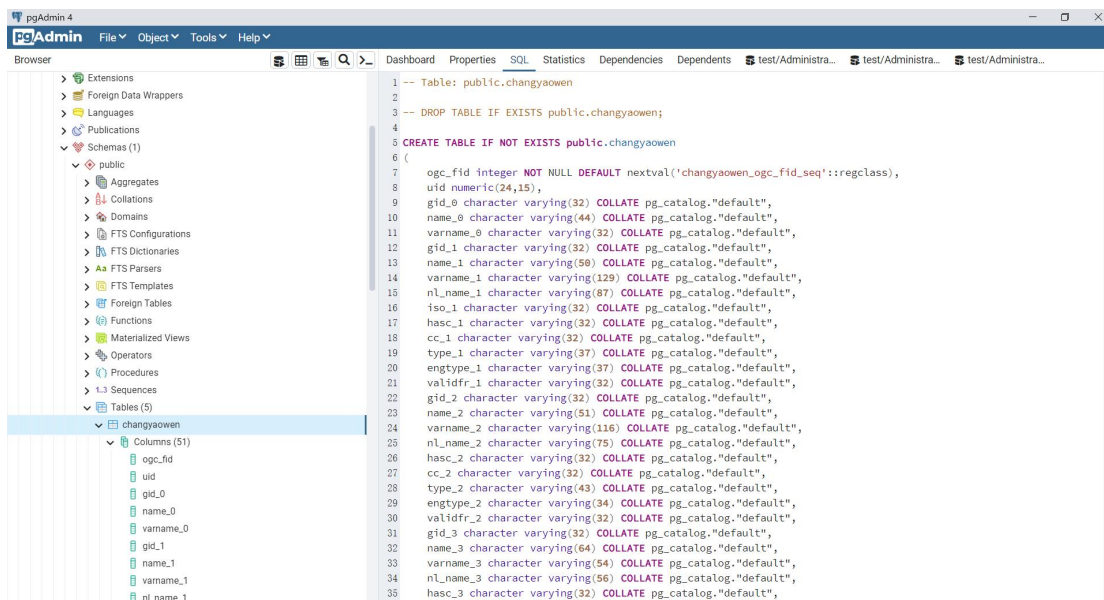


图 12（导入后的 changyaowen 表的结果）

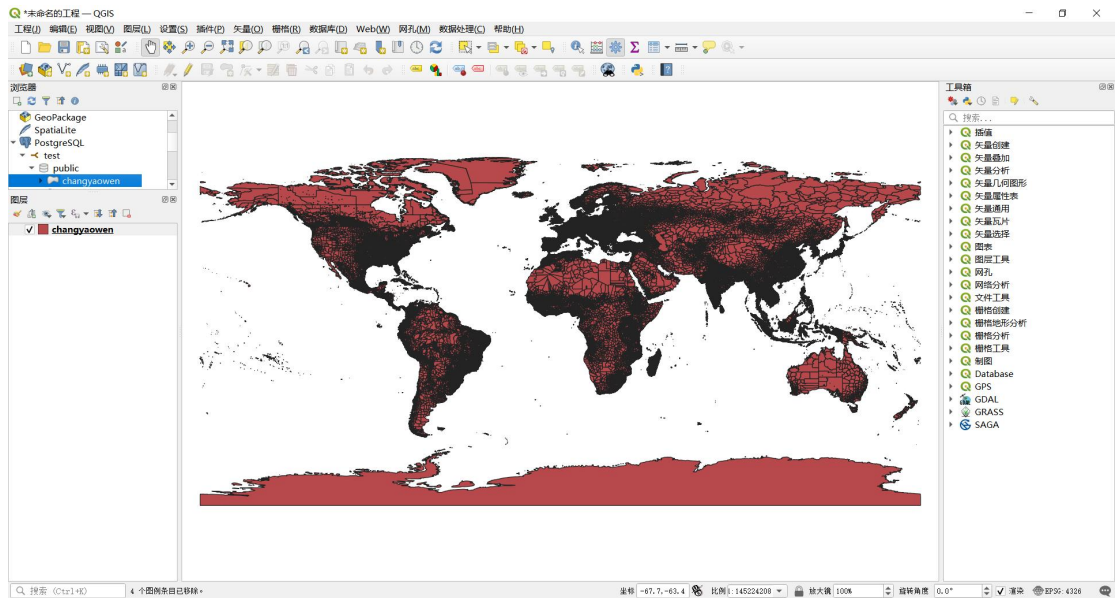


图 13（与 QGIS 连接后的结果）

⑥ 将 json 文件导入到 mongodb 数据库中，这里使用的数据是群中同学处理过的 json 文件，如果未经处理则需要使用 ogr2ogr 命令将 shp 类型文件转化为 json 类型的文件，然后删去头尾，并且考虑到单条数据的大小，跳过大于 16M 的数据，然后使用指令将数据导入到 mongodb 中，使用指令可以直接创建一个数据库以及库中的表，操作较为方便，mongodb 的数据指令如下：

```
mongoimport --db test --collection changyaowen --jsonArray --file
D:/data/gadm.json
```



```

C:\Users\Administrator>mongoimport --db test --collection changyaowen --jsonArray --file D:/data/gadm.json
connected to: mongodb://localhost/
2022-05-13T21:09:29.566+0800 [.....] test.changyaowen 69.8MB/5.01GB (1.4%)
2022-05-13T21:09:32.603+0800 [.....] test.changyaowen 121MB/5.01GB (2.4%)
2022-05-13T21:09:35.590+0800 [.....] test.changyaowen 155MB/5.01GB (3.0%)
2022-05-13T21:09:38.581+0800 [.....] test.changyaowen 209MB/5.01GB (4.1%)
2022-05-13T21:09:41.566+0800 [.....] test.changyaowen 262MB/5.01GB (5.1%)
2022-05-13T21:09:44.568+0800 [#.....] test.changyaowen 305MB/5.01GB (5.9%)
2022-05-13T21:09:47.576+0800 [#.....] test.changyaowen 343MB/5.01GB (6.7%)
2022-05-13T21:09:50.584+0800 [#.....] test.changyaowen 365MB/5.01GB (7.1%)
2022-05-13T21:09:53.580+0800 [#.....] test.changyaowen 382MB/5.01GB (7.4%)
2022-05-13T21:09:56.572+0800 [#.....] test.changyaowen 407MB/5.01GB (7.9%)
2022-05-13T21:09:59.569+0800 [#.....] test.changyaowen 427MB/5.01GB (8.3%)
2022-05-13T21:10:02.566+0800 [##.....] test.changyaowen 452MB/5.01GB (8.8%)
2022-05-13T21:10:05.566+0800 [##.....] test.changyaowen 485MB/5.01GB (9.4%)
2022-05-13T21:10:08.571+0800 [##.....] test.changyaowen 497MB/5.01GB (9.7%)
2022-05-13T21:10:11.566+0800 [##.....] test.changyaowen 518MB/5.01GB (10.1%)
2022-05-13T21:10:14.624+0800 [##.....] test.changyaowen 535MB/5.01GB (10.4%)
2022-05-13T21:10:17.567+0800 [##.....] test.changyaowen 555MB/5.01GB (10.8%)
2022-05-13T21:10:20.575+0800 [##.....] test.changyaowen 574MB/5.01GB (11.2%)
2022-05-13T21:10:23.566+0800 [##.....] test.changyaowen 602MB/5.01GB (11.7%)
2022-05-13T21:10:26.566+0800 [##.....] test.changyaowen 640MB/5.01GB (12.5%)
2022-05-13T21:10:29.571+0800 [##.....] test.changyaowen 677MB/5.01GB (13.2%)
2022-05-13T21:10:32.582+0800 [###.....] test.changyaowen 713MB/5.01GB (13.9%)
2022-05-13T21:10:35.568+0800 [###.....] test.changyaowen 764MB/5.01GB (14.9%)
2022-05-13T21:10:38.592+0800 [###.....] test.changyaowen 778MB/5.01GB (15.2%)
2022-05-13T21:10:41.588+0800 [###.....] test.changyaowen

```

图 14（将 json 文件导入 mongodb 数据）

```

2022-05-13T21:21:44.568+0800 [#####...] test.changyaowen 4.49GB/5.01GB (89.6%)
2022-05-13T21:21:47.567+0800 [#####...] test.changyaowen 4.49GB/5.01GB (89.6%)
2022-05-13T21:21:50.566+0800 [#####...] test.changyaowen 4.51GB/5.01GB (89.9%)
2022-05-13T21:21:53.566+0800 [#####...] test.changyaowen 4.51GB/5.01GB (89.9%)
2022-05-13T21:21:56.566+0800 [#####...] test.changyaowen 4.52GB/5.01GB (90.2%)
2022-05-13T21:21:59.566+0800 [#####...] test.changyaowen 4.52GB/5.01GB (90.2%)
2022-05-13T21:22:02.579+0800 [#####...] test.changyaowen 4.57GB/5.01GB (91.1%)
2022-05-13T21:22:05.591+0800 [#####...] test.changyaowen 4.61GB/5.01GB (92.0%)
2022-05-13T21:22:08.568+0800 [#####...] test.changyaowen 4.64GB/5.01GB (92.6%)
2022-05-13T21:22:11.568+0800 [#####...] test.changyaowen 4.67GB/5.01GB (93.2%)
2022-05-13T21:22:14.581+0800 [#####...] test.changyaowen 4.70GB/5.01GB (93.8%)
2022-05-13T21:22:17.588+0800 [#####...] test.changyaowen 4.75GB/5.01GB (94.7%)
2022-05-13T21:22:20.568+0800 [#####...] test.changyaowen 4.78GB/5.01GB (95.3%)
2022-05-13T21:22:23.594+0800 [#####...] test.changyaowen 4.82GB/5.01GB (96.2%)
2022-05-13T21:22:26.568+0800 [#####...] test.changyaowen 4.87GB/5.01GB (97.0%)
2022-05-13T21:22:29.569+0800 [#####...] test.changyaowen 4.91GB/5.01GB (97.9%)
2022-05-13T21:22:32.615+0800 [#####...] test.changyaowen 4.95GB/5.01GB (98.7%)
2022-05-13T21:22:35.578+0800 [#####...] test.changyaowen 4.98GB/5.01GB (99.3%)
2022-05-13T21:22:38.569+0800 [#####...] test.changyaowen 4.99GB/5.01GB (99.6%)
2022-05-13T21:22:40.457+0800 [#####...] test.changyaowen 5.01GB/5.01GB (100.0%)
2022-05-13T21:22:40.458+0800 348901 document(s) imported successfully. 0 document(s) failed to import.
C:\Users\Administrator>

```

图 15（将 json 文件导入 mongodb 数据）

可以看到有 348901 条数据成功导入到了数据库中，应该导入 348904 条数据，有 3 条未导入，因为 3 条超过了 16M，所以没有导入

⑦ 本机的软硬件环境

CPU 信息：本机的物理 CPU 为 1 个，CPU 线程数为 8，CPU 核心数为 4

```
命令提示符 - wmic
Microsoft Windows [版本 10.0.22621.1]
(c) Microsoft Corporation。保留所有权利。

C:\Users\lenovo>wmic
wmic:root\cli>cpu get Name
Name
Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz

wmic:root\cli>cpu get NumberOfCores
NumberOfCores
4

wmic:root\cli>cpu get NumberOfLogicalProcessors
NumberOfLogicalProcessors
8
```

图 16（本机 CPU 信息）

Windows 操作系统

Windows 规格

复制

^

版本

Windows 11 家庭中文版

版本

22H2

安装日期

2022/5/14

操作系统版本

22621.1

序列号

PF282JTB

体验

Windows Feature Experience Pack 1000.22632.1000.0

[Microsoft 服务协议](#)

[Microsoft 软件许可条款](#)

图 17（本机操作系统）

内存信息

```
物理内存总量: 16,252 MB
可用的物理内存: 7,821 MB
虚拟内存: 最大值: 30,076 MB
虚拟内存: 可用: 15,039 MB
虚拟内存: 使用中: 15,039 MB
```

图 18（内存信息）

最大内存支持 16G

Attributes	BankLabel	Capacity	Caption	ConfiguredClockSpeed	ConfiguredVoltage	CreationClassName	DataW
Width	Description	DeviceLocator	FormFactor	HotSwappable	InstallDate	InterleaveDataDepth	InterleavePosition
Manufacturer	MaxVoltage	MemoryType	MinVoltage	Model Name	OtherIdentifyingInfo	PartNumber	Positic
InRow	PoweredOn	Removable	Replaceable	SerialNumber	SKU	SMBIOSMemoryType	Speed
TypeDetail	Version					Status	Tag
1	BANK 0	8589934592	Physical Memory	2933		1200	Win32_PhysicalMemory
	Physical Memory	ChannelA-DIMM0	12			1	1
samsung	1500	0	1500	Physical Memory			M471A1K43DB1-CWE
			15D0A368	26		2933	Physical Memory 0
128							64
1	BANK 2	8589934592	Physical Memory	2933		1200	Win32_PhysicalMemory
	Physical Memory	ChannelB-DIMM0	12			1	2
samsung	1500	0	1500	Physical Memory			M471A1K43DB1-CWE
			15D0A33D	26		2933	Physical Memory 1
128							64

图 19（内存相关信息）

（2）性能及索引功能测试

① 在 pgadmin4 中检索西经 60 度的经线穿过的所有国家，因为只需返回国家名字，故对查询返回结果 name_0 使用 distinct 关键字，查询条件使用 st_intersect 函数，结果为：南极洲、阿根廷、玻利维亚、巴西、加拿大、福克兰群岛、格陵兰岛、圭亚那、巴拉圭、委内瑞拉十个国家或地区，使用的 sql 语句为：

```
select distinct name_0
from test.public.changyaowen
where ST_intersects(wkb_geometry,'SRID=4326;LINESTRING(-60 -90,-60
90)')
```

1	<code>select distinct name_0</code>
2	<code>from test.public.changyaowen</code>
3	<code>where ST_intersects(wkb_geometry,'SRID=4326;LINESTRING(-60 -90,-60 90)')</code>

Messages	Data Output	Explain	Notifications
<div> <div>name_0</div> <div>character varying (44)</div> <div> <div>1</div> <div>Antarctica</div> </div> <div> <div>2</div> <div>Argentina</div> </div> <div> <div>3</div> <div>Bolivia</div> </div> <div> <div>4</div> <div>Brazil</div> </div> <div> <div>5</div> <div>Canada</div> </div> <div> <div>6</div> <div>Falkland Islands</div> </div> <div> <div>7</div> <div>Greenland</div> </div> <div> <div>8</div> <div>Guyana</div> </div> <div> <div>9</div> <div>Paraguay</div> </div> <div> <div>10</div> <div>Venezuela</div> </div> </div>			

图 20（查询结果）

② Mongo 查询，使用 pymongo 库连接数据库，使用 distinct 函数筛选出西经 60° 线穿越的国家名（代码 Mongosearch）。

```

1 import pymongo
2 import time
3
4 client = pymongo.MongoClient('localhost', 27017) # 连接数据库
5 db = client.test
6 collection = db['changyaowen']
7
8 start = time.perf_counter()
9
10 result = collection.find({
11     "geometry": {"$geoIntersects":
12         {"$geometry": {"type": "LineString", "coordinates": [[-60, 90], [-60, -90]]}
13         }
14     }, {"properties.NAME_0": 1})
15
16 for tt in result:
17     print(tt)
18 end = time.perf_counter()
19 print("查询用时: ", end - start)

```

图 21（Mongo 查询）

查询结果如下图，阿根廷、玻利维亚、巴西、加拿大、福克兰群岛、格陵兰岛、圭亚那、巴拉圭、委内瑞拉九个国家或地区，和 postgresql 相比缺少南极洲，分析结果为南极洲数据超过 16M，导入时跳过：

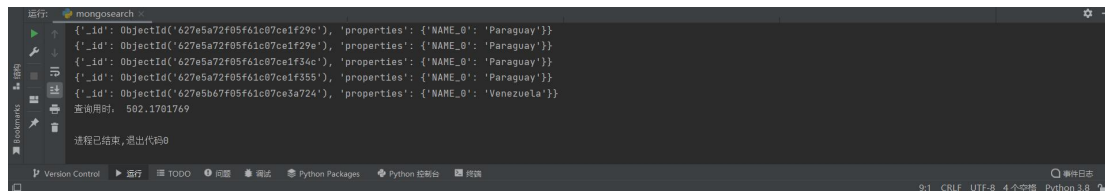


图 22（未建立索引的查询结果）

③ 查询时间的比较：

在 pgadmin 中 SELECT 语句前加 EXPLAIN ANALYZE 关键字即可获取查询时间，在 Mongodb 查询中分为有索引与无索引的情况，mongo shell 中也可以调用 explain() 函数帮助我们查看查询相关的信息，在建立索引后需要将索引删除然后再进行查询。

建立索引的命令：

```
db.changyaowen.createIndex( {"geometry": "2dsphere" } )
```

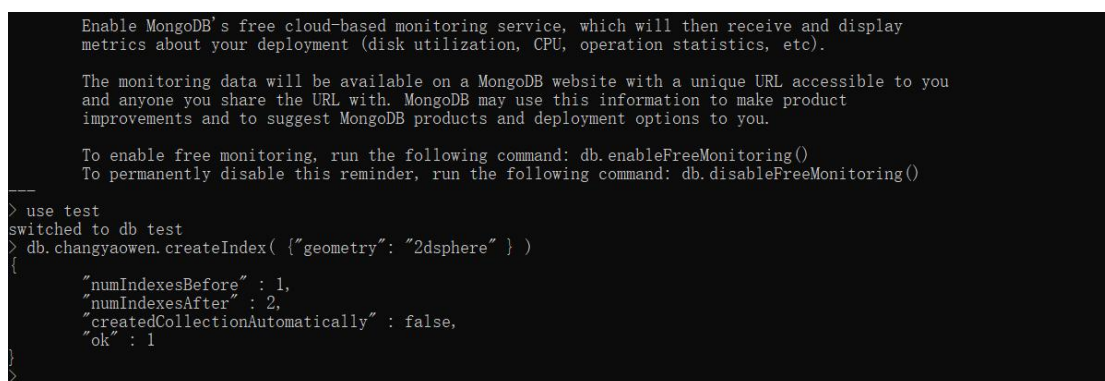


图 23（建立索引）



图 24（建立索引后的查询结果）

Postgis 查询 (s)	mongodb 不建立索引 (s)	mongodb 建立索引 (s)
0.273	494.592	2.92
0.254	438.189	2.94
0.278	484.332	2.88
0.294	507.116	2.939
0.240	463.609	2.925
0.289	502.170	2.883
0.271333333 (均值)	481.668 (均值)	2.9145 (均值)

（五次查询的时间综合）

由表可见 postgis 和 mongodb（无索引）检索效率差异显著，mongodb（无索引）平均时间为 481.668s，而 postgis 仅仅为 0.27133333s，二者相差近 1775 倍。Mongo 建立索引以后，其检索时间平均值为 2.883s，与不建索引的 481.668s 快了近 167 倍，但与 postgis 的 0.271333s 仍然有一定差距，相差近 10 倍。故我们可以得出检索效率排序：

（Postgis > Mongo 带索引 > Mongo 无索引）

(3) 利用程序，基于 postgis 数据库现如下功能并进行测试

① 实现一个函数，输入任意一个国家，统计出这个国家的县级行政区的数量（即一个国家数据中包含的要素数量），并统计这个国家中县级行政区面积最大的三个，打印出名字和面积值。因为要统计出这个国家的县级行政区的数量，以及县级行政区面积最大的三个，并打印数值，所以需要通过输入的国家名查询出该国各个县级行政区的名称与面积，并将结果输出（这里由于由大到小，所以要倒序输出）。取倒序后的前三个数据就是最大的县级行政区，使用了 psycopg2 库连接 postgresql 数据库。我使用了 st_transform() 函数，将数据的坐标系由 WGS-84 坐标系转换，把坐标的单位由度转为米，使得输出的数据更加直观，然后使用 st_area() 函数计算面积，从而获得各个县级行政区的面积从而为后续的分类提供依据。

```
1 import psycopg2 as pg
2
3
4 def Getdetail_nation(nationname):
5     # 连接数据库
6     con = pg.connect(database="test", user="Administrator", password="l123321", host="127.0.0.1", port="5432")
7     cur = con.cursor()
8     # 查询
9     sql = "select NAME_3,ST_Area(st_transform(wkb_geometry,4527))/1000000 from public.changyaowen where NAME_0='" + nationname + \'
10         "' + "ORDER BY ST_Area(st_transform(wkb_geometry,4527))/1000000 DESC" + ";"
11     cur.execute(sql)
12     result = cur.fetchall()
13
14     count = len(result)
15     print(nationname + "县级行政区的数量:" + str(count))
16     for i in range(3):
17         print(result[i][0] + " 面积为: ", result[i][1])
18     con.close()
19     return count
```

图 25（输出县级行政区代码）

以中国为测试：

```
China县级行政区的数量: 2435
Ruqiang 面积为: 226027.8152608579
Nyima 面积为: 214233.2662547603
Qieso 面积为: 169361.6146692584
进程已结束,退出代码0
```

图 26（中国结果）

以加拿大为测试：



图 27（加拿大结果）

② 实现一个函数，输入任意一个国家，统计出这个国家在地域上邻接的其它国家（需要做数据的综合，因为只有县级数据），由于两个国家的县级行政区如果在地域上有邻接关系的话，必然会有相交的部分，利用 `ST_intersects()` 函数可以获得两个相交的位置的地域，通过从两张相同的表中选取给定的国家，利用给定的国家查询相邻接的国家，相当于对数据做了综合，实现了结果的查询。操作中使用了 `psycopg2` 库连接 `postgresql` 数据库，`DISTINCT` 关键字返回不重复记录的国家名。

代码：

```
1 import psycopg2 as pg
2
3
4 def Get_nearcountry(nationname):
5     # 连接数据库
6     con = pg.connect(database="test", user="Administrator", password="l123321", host="127.0.0.1", port="5432")
7     cur = con.cursor()
8     # 查询
9     sql = "select distinct table1.NAME_0 as \
10         " from public.changyaowen table1,public.changyaowen table2" + \
11         " where table1.name_0!='' + nationname + "' " + \
12         " AND table2.name_0=''' + nationname + "'" + \
13         " AND ST_intersects(table1.wkb_geometry,table2.wkb_geometry)"
14     cur.execute(sql)
15     data = cur.fetchall()
16     con.close()
17
18     country = []
19     for tt in data:
20         country.append(tt[0])
21     return country
```

图 27（代码实现查询邻近国家）

结果：

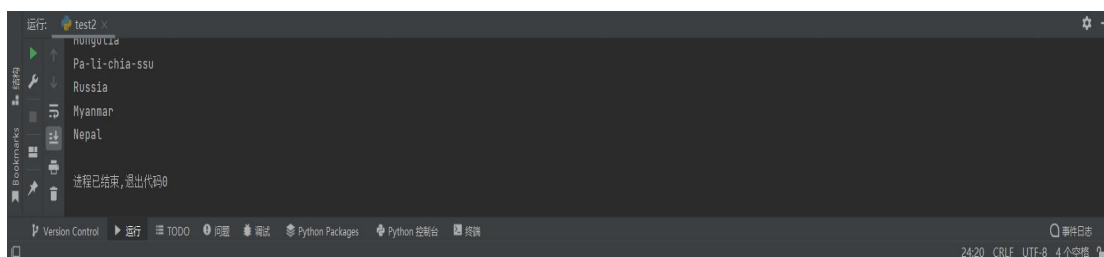


图 29（中国的邻接国家）

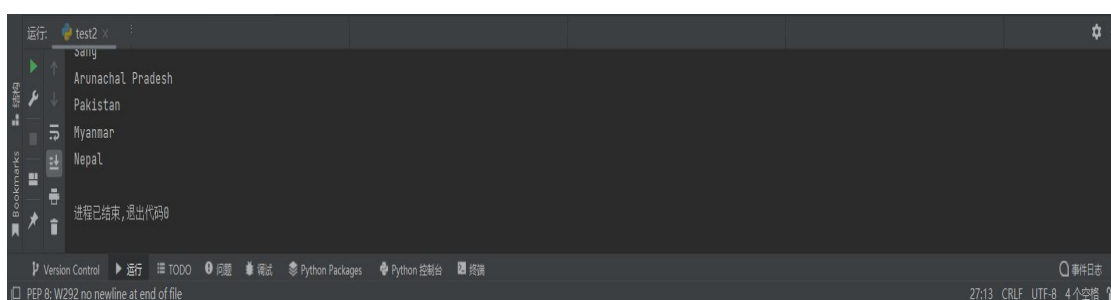


图 30（印度的邻接国家）

③ 实现一个数据处理函数，函数接受三个参数：国家名，像素分辨率，输出目录。输入任意一个国家，将这个国家的数据按指定像素分辨率进行渲染（像素分辨率即一个像素代表的实际地理范围的大小，宽和高等分辨率），然后按 256x256 像素大小的索引格网，对指定国家的渲染后栅格进行切分，切成多个正方形的影像数据并写入到输出目录中（这个过程也称为切片）。需要考虑对应国家的最大外接矩形边界，然后按比例生成图片，每个小的行政区域随机指定一种颜色，所有县级边界用黑线隔开。输出切片影像数据的命名采用索引方式，例如，最左上角的切片文件名为 1-1，其下的为 2-1，右侧的为 1-2，以此类推。如果输出数据格式为 jpg 等不带有坐标信息的栅格数据，则根据每个切片文件的数据坐标参数，生成对应的坐标配准文件，例如，jpg 文件要输出对应的 jgw 文件，最终在 qgis 中打开全部切片数据，综合显示并截图。

解决该问题需要以下的操作，获取该国的经纬度范围，通过像素分辨率以确定整张图片的大小；获取该国的全部县级行政区数据；将获取的全部县级行政区经纬度坐标转为像素坐标，并且进行图像的渲染；将原图按给定的索引格网大小进行切割，并生成坐标配准文件 .jgw，最后导入到 QGIS 中显示，少了 .jgw 会使导入

后图像重叠。注意到县级行政区的数据类型为 MUTIPOLYGON —— 多边形集，如果某个县级行政区由两块相互独立的区域组成，那么其数据存储形式应为 ((), ()) 型，而不是简单的 () 型，甚至还会存在 (((), ())) 的形式，故获取县级行政区点数据时，不能简单地将所有点数据提取出来，而应该分区域提取，否则在后续的地图绘制中，本应由若干个位置独立多边形组成的县级行政区，绘制会出现问题。我在处理的时候通过判断括号的层数，为每一个数据都加上三层括号，使得最后的结果是一个所有的数据都具有三层括号的形式，最后通过循环将括号消除，最后获取每一个点的数据，从而为渲染提供数据。

```
sql1="SELECT st_astext(st_Envelope(st_union(wkb_geometry))) \
FROM (SELECT wkb_geometry \
FROM changyaowen \
WHERE name_0='{ }')as foo;".format(nation)
cur.execute(sql1)
mt=cur.fetchall()
mt=mt[0][0]
mt=mt[9:-2]
mt=mt.split(',')
mincoor=mt[0]
maxcoor=mt[2]
mincoor=mincoor.split(' ')
maxcoor=maxcoor.split(' ')
lonmin=eval(mincoor[0])      #最小经度
latmin=eval(mincoor[1])     #最小纬度
lonmax=eval(maxcoor[0])     #最大经度
latmax=eval(maxcoor[1])     #最大纬度
origin=[lonmin,latmax]
```

图 31（获取图像的最大最小经纬度）

由于为了获得最大最小的经纬度，需要把 sql 获取的信息进行一定的提取综合，由于有括号，所以需要用到 split 函数切分数据，使得可以用 eval 获取值，但是这步操作是与同学研究数据组织方式得出的，在后续的思考中，发现有函数可以直接获得最大最小的经纬度，使用 st_xmax, st_xmin, st_ymax, st_ymin 可以直接获取想要的最大最小值经纬度，因此这一步有两种方法获得经纬度的最大最小值。


```

conn = pg.connect(database="test", user="Administrator", password="11233321", host="127.0.0.1", port="5432")
cur = conn.cursor()
# 查询
sql = "select NAME_3, st_astext(geom),st_xmin(geom),st_ymin(geom),st_xmax(geom),st_ymax(geom) from public.liuxiaoan where NAME_0='"
cur.execute(sql)
data = cur.fetchall()
conn.close()

```

图 32（获取图像的最大最小经纬度）

```

#读取国家县级行政区数据
sql2="select st_astext(wkb_geometry)\
      from changyaowen\
      where name_0='{ }'".format(nation)
county = []
cur.execute(sql2)
temp=cur.fetchall()
for item in temp:#每个县item
    temp_county=[]
    ifffloat=False
    item=item[0][12:]
    item=item.replace(' ','')
    item=eval(item)
    if type(item[0])==float:
        i=0
        if ifffloat==False :
            temp_county.append([])
            temp_county[0].append([])
            ifffloat=True
        while i<len(item):
            temp_county[0][0].append([item[i],item[i+1]])
            i+=2
    elif type(item[0][0])==float:
        if ifffloat==False :
            temp_county.append([])
            ifffloat=True
        for item0 in item:
            j=0
            temp_temp_county=[]
            while j<len(item0):
                temp_temp_county.append([item0[j],item0[j+1]])
                j+=2

```

图 33（获取县级行政区划）

这一步因为研究数据组织方式可以发现不只具有数据存储形式为 ((), ()) 型的数据，和简单的 () 型，甚至还会存在 (((), ())) 的形式，故获取县级行政区点数据时，不能简单地将所有点数据提取出来，而应该分区域提取，否则在后续的地图绘制中，本应由若干个位置独立多边形组成的县级行政区，绘制会出现问题。我在处理的时候通过判断括号的层数，利用 ifffloat 的真假值，为真值时，为数据加上一层括号，通过这样的方式为每一个数据都加上三层括号，使得最后的结果是一个所有的数据都具有三层括号的形式，保证数据的使用处理时是一个相同组织的整体，通过循环将括号消除，获取每一个点的数据。最后将所有的点储存在 country 中，获取所有的县城数据。

渲染图形，图形渲染的方法同之前的作业相类似，在渲染图形之前需要将获取图形的大小，即图形的宽高。

```
# 获取图像宽度和高度
width = int((lonmax - lonmin) // pixelcop + 1)
height = int((latmax - latmin) // pixelcop + 1)
```

图 34（通过最大最小经纬度和分辨率）

```
#绘图
img = Image.new("RGB", (width, height), "white")
draw = ImageDraw.Draw(img)
for i in county:
    col=(random.randint(0,255),random.randint(0,255),random.randint(0,255))#赋予随机颜色
    for j in i:
        for k in j:
            part = []          #像素坐标，因为一个地区可能由若干部分组成，数据组织方式较为复杂，解决套环的问题
            for f in k:
                part.append(changepixel(shp2,origin,pixelcop))
            draw.polygon(part, outline=(0,0,0), fill=col)          #渲染到图片中
#切片
```

图 35（绘制图像与之前的第二次作业有类似之处）

```
#坐标转为图像像素坐标
def changepixel(cor,corog,pixelcop):#cor为输入点，corog为原点，pixelcop为分辨率
    lon=cor[0]#输入点经度
    lat=cor[1]#输入点纬度
    lon0=corog[0]#原点经度
    lat0=corog[1]#原点纬度
    u=int((lon-lon0)//pixelcop)
    v=int((lat-lat0)//pixelcop)
    return (u,v)
```

图 36（地理坐标转换为像素坐标）

在绘制图像用到了 random, Image, ImageDraw 库，random 保证了渲染的随机性，Image 与 ImageDraw 类保证了可以定义一个空的图像，而且利用相关渲染数据可以在图像上进行绘制，这里用了四层循环也是在同学的启发下得到的，因为之前将所有的数据都组织为了（（（），（））....）形式，这样循环可以使得每次

读取获取的点的数值都能用相同的方法处理得到,从而获得每个多边形中的点的坐标,实现成功渲染,在绘制时还要进行相关的坐标转换,将地理坐标转换为像素坐标(在之前的作业中也实现过相关操作)。

```
#切片
COLUM=int(width//mx)+1
ROW=int(height//my)+1
for r in range(ROW):
    for c in range(COLUM):
        cropped = img.crop((x0+mx*c, y0+my*r, x0+mx*(c+1), y0+my*(r+1)))
        cropped.save("{}-{}-{}.jpg".format(output,r+1,c+1))
        with open("{}-{}-{}.jgw".format(output,r+1,c+1),'w') as jgw:
            jgw.write('{}\n0.00\n0.00\n-{}\n{}\n{}\n'\n'
                        .format(pixelcop,pixelcop
                                lonmin+c*mx*pixelcop,latmax-r*my*pixelcop))
```

图 37 (切片操作)

由于 jgw 的文件格式如下图:

第一种格式:

- 1、X-Scale(一个像元的大小)
- 2、旋转项
- 3、旋转项
- 4、负的Y-Scale(一个像元的大小)
- 5、转换项,即左上角X坐标
- 6、转换项,即左上角Y坐标

图 38 (jgw 文件格式)

所以在上述文件书写的时候文件格式写入时采用下列语句使得写入正确:

```
with open("{}-{}-{}.jgw".format(output,r+1,c+1),'w') as jgw:
    jgw.write('{}\n0.00\n0.00\n-{}\n{}\n{}\n'
```

图 39 (写入文件方式)

0.1
0
0
-0.1
73.55770111200007
53.56085968000019

图 40（写出的 jgw 文件）

测试结果：

中国 0.1 度

1-1.jgw	2022/5/15 10:36	JGW 文件	1 KB
1-1	2022/5/15 10:36	JPG 文件	20 KB
1-2.jgw	2022/5/15 10:36	JGW 文件	1 KB
1-2	2022/5/15 10:36	JPG 文件	28 KB
1-3.jgw	2022/5/15 10:36	JGW 文件	1 KB
1-3	2022/5/15 10:36	JPG 文件	6 KB
2-1.jgw	2022/5/15 10:36	JGW 文件	1 KB
2-1	2022/5/15 10:36	JPG 文件	2 KB
2-2.jgw	2022/5/15 10:36	JGW 文件	1 KB
2-2	2022/5/15 10:36	JPG 文件	10 KB
2-3.jgw	2022/5/15 10:36	JGW 文件	1 KB
2-3	2022/5/15 10:36	JPG 文件	1 KB

图 41(获取的文件)

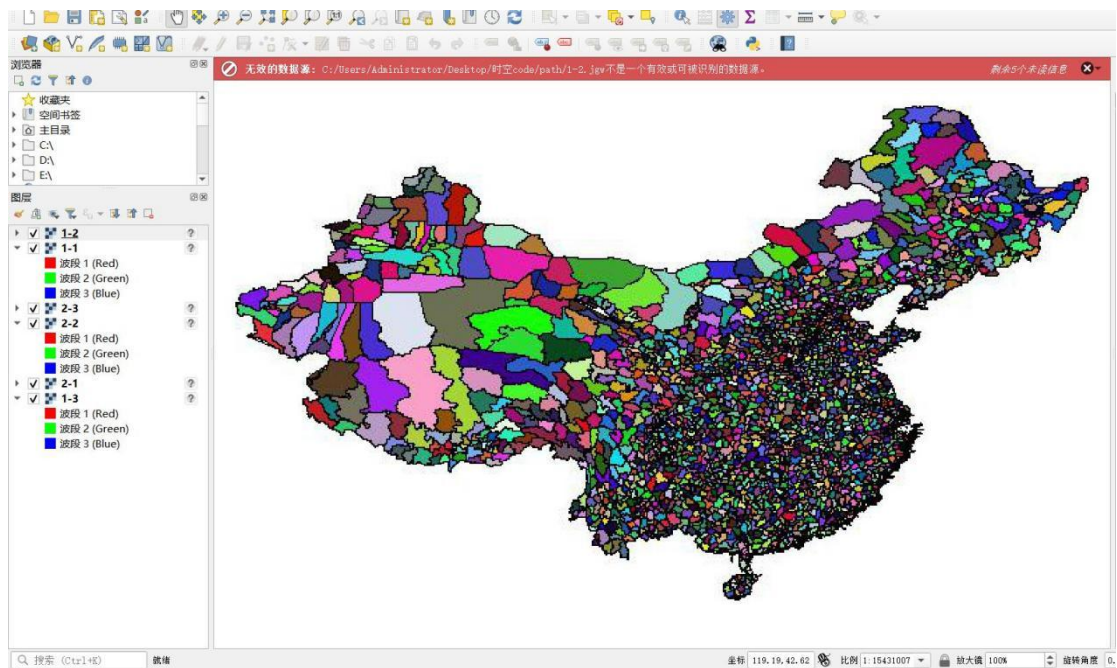


图 42 （连接 qgis 显示）

中国 0.2 度



图 43(获取的文件)

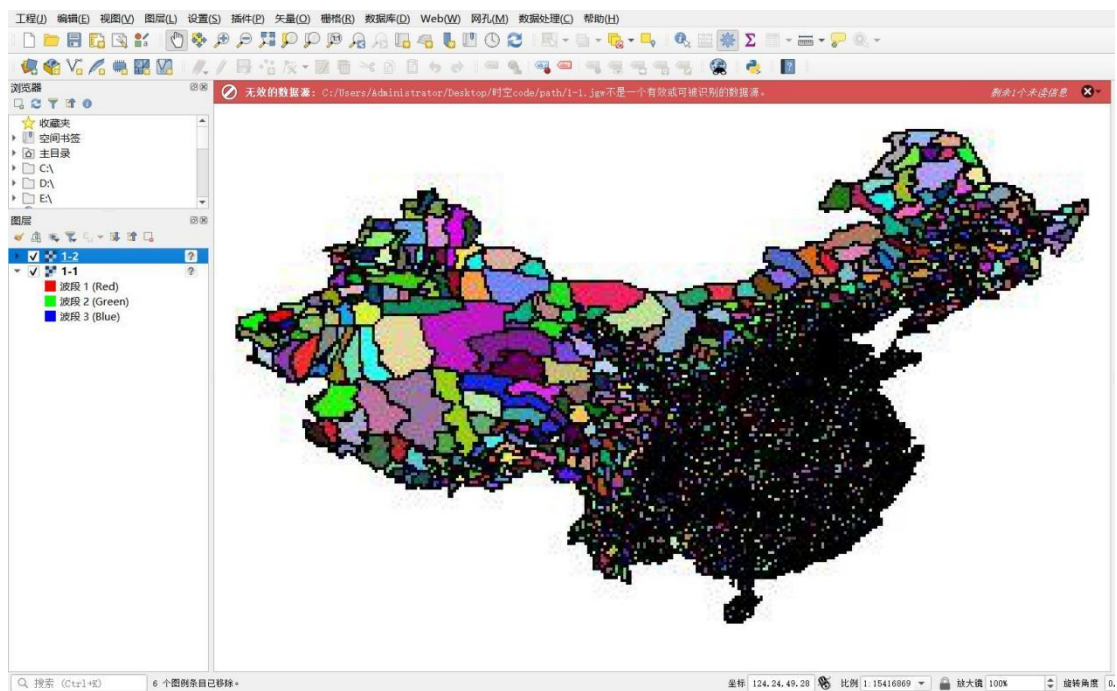


图 44 (0.2 度显示的中国)

巴西 0.1 度

此电脑 > Windows-SSD (C:) > 用户 > lenovo > 桌面 > 时空code > path2					
	名称	修改日期	类型	大小	
主文件夹					
下载	1-1.jgw	2022/5/15 10:36	JGW 文件	1 KB	
文档	1-1	2022/5/15 10:36	JPG 文件	27 KB	
图片	1-2.jgw	2022/5/15 10:36	JGW 文件	1 KB	
桌面	1-2	2022/5/15 10:36	JPG 文件	13 KB	
本地磁盘 (F:)	2-1.jgw	2022/5/15 10:36	JGW 文件	1 KB	
2020302131201-9	2-1	2022/5/15 10:36	JPG 文件	6 KB	
FileRecv	2-2.jgw	2022/5/15 10:36	JGW 文件	1 KB	
时空数据处理	2-2	2022/5/15 10:36	JPG 文件	3 KB	
OneDrive - Persona					
WPS网盘					
此电脑					
网络					

图 45(获取的文件)

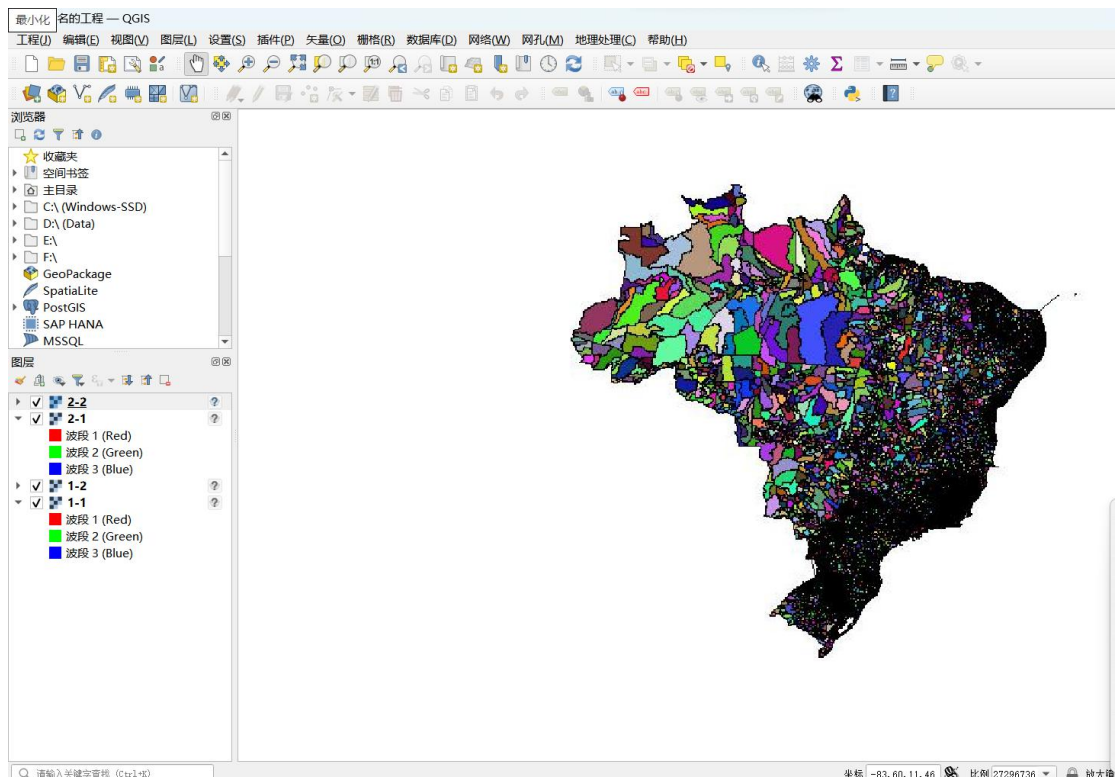


图 46(巴西 0.1 度连接 Qgis)

巴西 0.2 度

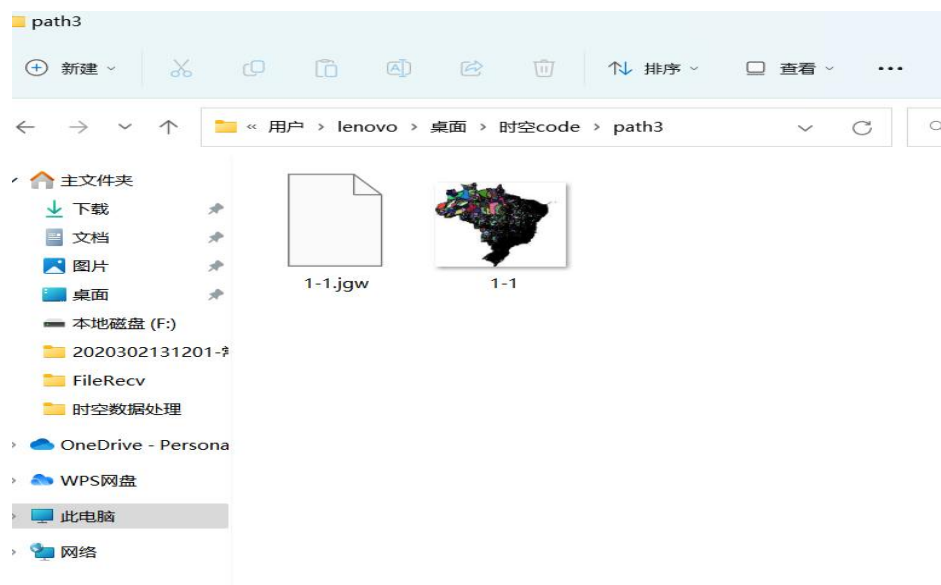


图 47(巴西 0.2 度获得的文件)

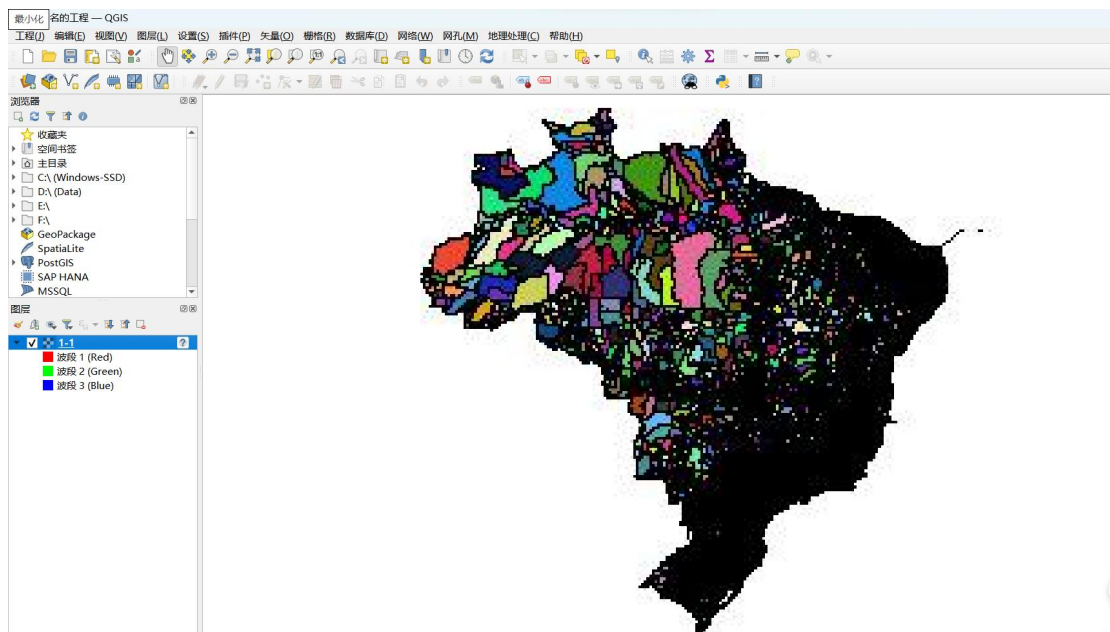


图 48 (巴西 0.2 度连接 Qgis 显示)

(4) 利用程序，基于 mongodb 数据库实现如下功能并测试

① 实现一个函数，给定一个经纬度坐标及编码长度，则生成一个 geohash 编码。首先根据编码长度判断经纬度二进制编码长度，precision 是我的编码长度，给定一个 even 作为 flag，当 even 为 true 时对经度进行编码，为 false 时对纬度进行编码，编码方法采用二分查找的方法来进行编码：

具体计算原理为：

```
while len(geohash) < precision:|
    if even:
        mid = (lon[0] + lon[1]) / 2
        if longitude > mid:
            ch |= bits[bit]
            lon = (mid, lon[1])
        else:
            lon = (lon[0], mid)
```

图 49 (对经度二进制编码)


```

else:
    mid = (lat[0] + lat[1]) / 2
    if latitude > mid:
        ch |= bits[bit]
        lat = (mid, lat[1])
    else:
        lat = (lat[0], mid)

```

图 50（对纬度二进制编码）

```

def encodehash(latitude, longitude, precision):
    lat, lon = (-90.0, 90.0), (-180.0, 180.0)
    geohash = []
    bits = [16, 8, 4, 2, 1]
    bit = 0
    ch = 0
    _base32 = "0123456789bcdefghjkmnpqrstuvwxyz"

```

图 51（编码原始数据）

```

    even = not even
    if bit < 4:
        bit += 1
    else:
        geohash += _base32[ch]
        bit = 0
        ch = 0
    return ''.join(geohash)

```

图 52（循环内部综合编码合并经纬度，将二进制编码变为 base32 编码）

由于利用了 even 这样一个 flag，使得经纬度轮流编码，且每次 base32 编码都会保存在 geohash 的数组中，当循环到达给定长度后，结束编码。

结果测试：

(58.4365, 140.6453) 按八位 geohash 编码的结果为: z4kh2qnc

(39.923201, 116.390705) 按八位 geohash 编码的结果为: wx4g0ec1

(39.928167, 116.389550) 按八位 geohash 编码的结果为: wwt5x0p5

② 实现一个函数，给定一个 mongodb 的县级行政区名字作为查询条件，即可查询出某个县级行政区域，然后，将这个县级行政区域的几何边界用 8 位 geohash 编码数据组织成为一大段文本进行描述（相邻顶点之间不留空格，连续存放编码）调用 pymongo 库连接 mongodb 后，查询该县级行政区的 geometry.coordinates 信息，遍历每个点，获取每个点的 geohash 编码后组成文本写入到 geocode 文件中

```
def hash(countyname, length=8):
    client = pymongo.MongoClient('localhost', 27017)
    db = client.test
    collection = db['changyaowen']
    #查询
    result = collection.find({
        "properties.NAME_3": countyname
    }, {"geometry": 1})

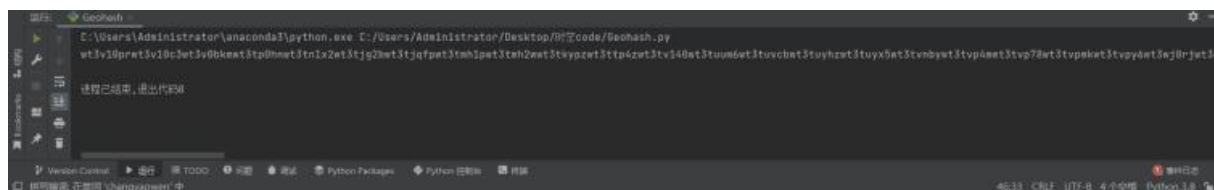
    t = []
    for tt in result:
        t.append(tt)

    #获取文本
    coordinate = t[0]['geometry']['coordinates']
    geohash = ''
    for k1 in coordinate:
        for k2 in k1:
            geohash += encodehash(k2[0], k2[1], length)
    with open("E:\期末课程设计\spatial_homework\geocode", 'w', encoding="utf-8-sig") as g:
        g.write(geohash)
    return geohash

name = "Echeng Shi"
code = hash(name)
print(code)
```

（对于相应县级区域进行编码的代码）

使用“湖北省鄂州市鄂城市”作为测试数据，得到结果为：



(结果)

```
wt3v10prwt3v10c3wt3v0bkmwt3tp0hnwt3tn1x2wt3tjg2bw3tjqfptwt3tmh1pwt3tmh2wwt3tkypzwt3ttp4zwt3tv
140wt3tuum6wt3tuvcbwt3tuyhzw3tuyx5wt3tnbywt3tvp4mwt3tvp78wt3tvpmkwt3tvp6wt3wj0rwt3wj0zrwt3
wj1wbwt3wj4n8wt3wj62nwt3wj750wt3wj7mtwt3wjebfwt3wjs78wt3wjstfwt3wjubswt3wjv5hwt3wjvm8wt3wjvre
wt3wnj9bwt3wnjefwt3wnjtfwt3wnjpywt3wnkc0wt3wnk6ewt3wn7guwt3wn7kzwt3wn7n4wt3wn6xtwt3wnd34wt
3wnd4uwt3wndjuwt3wnf0jw3wnf1wt3wnf5qwt3wnfhuwt3wnvcvwt3wnctywt3wnccq7wt3wnby8wt3wnbnwt3
tyzxqwt3tyzr3wt3tyyyywt3tyyw6wt3tyynjw3tyvvdwt3tyvmhwt3tyvj1wt3tyutmwt3tyumtw3tyunpwt3tyupjw3t
zh30wt3tzh75wt3tzhmhw3tzhnzwt3tz7bzw3tz7sbt3tz7x2wt3tze3tw3tze7swt3tzemkw3tzerjw3tzg
8jw3tztgbwt3tzu3bw3tztue8wt3tzuv2wt3tzuz3wt3vbhc4wt3vbhgw3t3vbjnw3t3vbm2wwt3vbmfw3t3vbqhmwt
3vbqw0wt3vbqzrwt3vbx2mwt3vbx0wt3y084xwt3y08u0wt3y09mswt3y09vfw3y0dnpwt3y0dw9wt3y0dztwt3y
0g1kw3y0g7tw3y0gy9wt3y15cpwt3y1h74wt3y1hgcwt3y1jk7wt3y1ju7wt3y1nm1wt3y1nxjw3y1r42wt3y1rtfw
t3y32pcwt3y389qwt3y394zwt3y39egwt3y3dj7wt3y3dw9wt3y3fbhwt3y3g1fw3y3fgpwt3y3fvkw3y3fwbwt3y3
fpdw3y618rwt3y613swt3y6157wt3y60vfw3y622qwt3y6247wt3y4rgbwt3y4rhvwt3y4qtkwt3y4qj6wt3y4mtswt
3y4krwt3y4kprwt3y46c8wt3y43brwt3y40whwt3vf5uxwt3vf5m3wt3vf1kzwt3vf0epwt3v9zzxwt3vc30wt3vc8u
mwt3vc91hwt3vc2ufwt3vc20hwt3v9nt7wt3v9jmbwt3v9hkbwt3v91skwt3v3p2uwt3v2yqywt3v2uddwt3v29zqwt
3v29xtwt3v29ptwt3v0xpuwt3v0uu6wt3v0usywt3v158nwt3v15fhw3v1hg0wt3v1jt0wt3v1jq7wt3v15yywt3v13c
5wt3v1344wt3v10pr
```

(写入文本后的结果)

(三) 课程设计心得与感悟

本次课程设计使我将上课老师所讲授的知识很好地运用,也明白了学习不只是一个人的探究,在第三个任务的第三个问题处理的时候,之前总是会遇到一些问题,在中国的第151条数据导入的时候遇到了问题,经过与同学的探讨后,发现数据的类型有着三层的括号,所以在处理的时候要把三层括号的因素考虑到,通过考虑解决了相应的问题,使我受益匪浅,这门课程教会了我如何去解决数据库有关的问题,去使用和学习对于时空数据的处理,在动手操作的过程中收获了许多之前没有体悟到的知识与方法。