# 摘 要

随着互联网技术的快速发展，基于互联网技术的地理信息系统(GIS)的应用也开始越来越广泛。如何设计更好的WebGIS已经成为各大互联网公司的重要研究课题，本文基于Web设计并实现了一个能够管理瓦片数据操作的工具。

本文在设计瓦片管理工具之前，首先需要给瓦片数据一个合适的存储环境。由于地理空间产生的瓦片数据随着分辨率层级的增加呈指数上升，所以分布式的存储方式对于瓦片的存储性价比最高。本文首先利用“瓦片打包存储”的方法，将数万张瓦片数据进行打包，存储在HDFS的文件目录中。为了方便查询瓦片，在存储瓦片数据时，还将每一张瓦片对应的瓦片索引存储在HBase数据表中。另外本文还将瓦片所属的金字塔元数据保存在Phoenix数据表中方便查询。

在此基础之上，就可以设计瓦片数据管理工具，针对瓦片的操作，管理工具设计了7大功能，包括管理员登陆及保存登陆状态、瓦片出入库、瓦片编辑(查询+删除)、瓦片目录浏览及预览。管理员使用特定的账号和密码登入系统(两者均为“admin”)；瓦片入库是将指定的一张瓦片存储在HDFS中，为了不浪费空间之后的瓦片采用追加的方式存入之前的文件中；瓦片出库是将指定的瓦片信息通过JDBC方式存入Mysql中；瓦片删除通过一个标志位isDelete字段来实现，不会真正的删除瓦片；瓦片目录浏览是将HDFS的目录结构以HTML语言的方式展示在客户端页面。

本文采用的瓦片数据样例的个数为数万张，瓦片数量不多，但是瓦片数据的存储和瓦片管理工具的功能已经经过测试，可以到达预期的要求。

关键词： 瓦片数据 分布式存储 WebGIS Hadoop

# **ABSTRACT**

With the rapid development of Internet technology, the application of Geographic Information System (GIS) based on Internet technology has begun to become more and more widespread. How to design a better WebGIS has become an important research topic for major Internet companies. This paper designs and implements a tool that can manage tile data operations based on Web.

Before designing tile management tools. First, this article needs to provide a suitable storage environment for tile data. The tile data produced by geospatial index increase exponentially with the increase of resolution level, the distributed storage method is most cost-effective for storing tiles. This article first uses the "tile packing and storage" method to pack tens of thousands of tile data and store it in the HDFS file directory. In order to facilitate the query of the tiles, when the tile data is stored, the tile index corresponding to each tile is also stored in the HBase data table. In addition, the pyramid metadata that the tile belongs to is saved in the Phoenix data table for easy query.

Based on this, you can design a tile data management tool. For the operation of the tile, the management tool has designed 7 major functions, including administrator login and save the login status, tile out of the warehouse and enter the warehouse, tile editing (Query + delete ), tile directory browsing and preview. The administrator uses a specific account and password to log in to the system (both "admin"); The tiles are stored in the HDFS in a specified tile, in order not to waste space, tiles are added to the previous file, Tile out of storage is to store the specified tile information in Mysql through JDBC; Tile deletion through a flag bit “isDelete” field to achieve, will not really delete tiles; Tile catalog browsing is to display the directory structure of HDFS in the form of HTML language on the client page.

The number of tile data samples used in this paper is tens of thousands, and the number of tiles is not large. However, the function of storing tile data and tile management tools has been tested and can reach the expected requirements.

**Keywords: tile data distributed storage WebGIS Hadoop**

# 目 录

**[第一章 绪 论](#_Toc22985)** [1](#_Toc22985)

[1.1 项目背景 1](#_Toc1465)

[1.2国内外研究现状 1](#_Toc3950)

[1.2.1 国外GIS发展情况 1](#_Toc2229)

[1.2.2国内GIS发展情况 2](#_Toc23772)

[1.3论文主要工作 3](#_Toc22246)

[1.4章节安排 3](#_Toc8696)

**[第二章 相关技术分析](#_Toc32542)** [5](#_Toc32542)

[2.1瓦片数据存储相关技术 5](#_Toc11978)

[2.1.1 Hadoop分析 5](#_Toc984)

[2.1.2 HDFS分析 6](#_Toc27690)

[2.1.3 HBase分析 7](#_Toc432)

[2.1.4 Phoenix分析 9](#_Toc1757)

[2.2 基于web数据管理工具相关技术 9](#_Toc21125)

[2.2.1 JSP分析 9](#_Toc6323)

[2.2.2 Servlet分析 10](#_Toc23215)

[2.2.3 Tomcat服务器分析 11](#_Toc26455)

[2.3本章小结 12](#_Toc22776)

**[第三章 详细设计](#_Toc20218)** [13](#_Toc20218)

[3.1瓦片数据存储方案实现 13](#_Toc5937)

[3.1.1 HDFS存储瓦片数据打包文件 13](#_Toc24323)

[3.1.2 HBASE存储瓦片索引数据 16](#_Toc2203)

[3.1.3 Phoenix存储金字塔元数据 19](#_Toc8536)

[3.2 基于web的瓦片数据管理工具的实现 20](#_Toc3202)

[3.2.1管理工具的界面设计 20](#_Toc2320)

[3.2.2功能实现 21](#_Toc32437)

[3.3本章小结 26](#_Toc11142)

**[第四章 功能测试](#_Toc19654)** [27](#_Toc19654)

[4.1瓦片数据的存储功能测试 27](#_Toc16159)

[4.1.1写瓦片数据和索引 27](#_Toc15922)

[4.1.2读取瓦片数据 30](#_Toc30033)

[4.1.3 存储瓦片金字塔元数据 31](#_Toc20192)

[4.2数据管理工具的功能测试 31](#_Toc12699)

[4.2.1管理员登陆及保存登陆状态 32](#_Toc13987)

[4.2.2瓦片出入库 32](#_Toc28053)

[4.2.3瓦片编辑(删除+查询) 34](#_Toc19859)

[4.2.4瓦片目录浏览及预览 36](#_Toc23113)

[4.3本章小结 37](#_Toc15493)

**[第五章 结束语](#_Toc4456)** [39](#_Toc4456)

**[致谢](#_Toc8679)** [41](#_Toc8679)

**[参考文献](#_Toc32032)** [43](#_Toc32032)

# 第一章 绪 论

## 1.1 项目背景

GIS是地理信息系统(Geoon System)的缩写，它是涉及到地理、计算机和GPS定位系统的一项综合性系统。通过计算机系统结合GPS定位对地球空间中任意的地理位置分布进行数据采集、存储和分析[[[1]](#endnote-0)]。

如今，随着地理信息技术、云计算技术研究的不断深入，GIS得到了迅猛发展。传统的GIS系统主要以桌面应用系统为主，而随着互联网技术的快速发展，基于互联网技术的地理信息系统--WebGIS就应运而生。并且在很多领域都得到了广泛的应用。

最近几年，云计算技术的发展逐渐引起了社会上广泛的关注，并渐渐被各大研究部门和互联网公司用来解决数据存储、分析等问题。这种现象使得云计算的技术更加成熟，越来越多的学科也逐渐开始采用这种技术手段来解决数据存储和分析所带来的问题。这其中也包括了GIS领域，还有很多著名的公司，比如国内的BAT这三家公司，还有如今发展势头迅猛的华为等，国外的Google、微软和IBM等国际互联网巨头公司也开始这一技术的研究。因为研究云计算技术的企业众多，并且行业没有统一的标准和规范，这些互联网公司相继都推出了自己公司的云计算平台，例如Abiquo公司推出的AbiCloud平台[[[2]](#endnote-1)]、加利福尼亚大学的Eucalyptus项目和当前最流行的，属于Apache基金会的开源云计算平台Hadoop。Hadoop由于其在性能、经济以及存储上的优点，被很多的公司所使用，得到了广泛的发展。

在GIS领域，由于地理空间的多样性导致GIS产生的数据也十分庞大，相应的也就要有能处理海量数据的平台，云计算技术则很好的满足这一条件，所以云计算在GIS领域中广泛应用开来。

## 1.2国内外研究现状

### 1.2.1 国外GIS发展情况

GIS在国外最先发展起来，所以国外的GIS技术相比较国内要成熟许多，在产品的成熟度上要比国内高一点。比如，美国环境系统研究所的ArcGIS[[[3]](#endnote-2)]、美国MapInfo Corporation公司的MapInfo、美国Calper公司开发的Maptitude以及1982年由Autodesk(欧特克)开发设计的AutoCAD Map等。

MapInfo是美国MapInfo Corporation公司[[[4]](#endnote-3)]开发的基于桌面的GIS软件，它是Mapping和information的共同缩写。Mapping表示地图的含义，即包含地图空间影像等一系列与地图相关的属性对象；information表示数据，可以衡量特定的地理空间地图中能够量化的属性数据。MapInfo根据地理学科相关概念、集成多种数据库，开发出的一款可以被很多行业所使用的软件系统。其中MapInfo Professional更是可以在Windows平台上分析相关的地图数据信息。

Autodesk公司[[[5]](#endnote-4)]在1982年推出的AutoCAD Map将CAD(计算机辅助设计)和GIS相结合，并实现了用户本身可以去管理地理空间中所产生的数据。而且Map 3D可以将数据和照片相结合进行渲染，并且可以实时勘测现场的各种地形结构进行更新数据。除此之外，Map工具可以将地图空间数据、影像地图等相关数据导出。在错误处理方面，Map可以自动消除在绘图方面的错误，提高数据采集、分析过程中的正确率。AutoCAD Map将地理空间数据分析功能结合到CAD的流程制作过程之中，可以用CAD的相关技术去处理影响数据。

Autodesk公司除了上面说的AutoCAD Map，还有MapGuide EnterPrise。它是一款企业级GIS平台，用户可以在浏览器中通过Web服务访问到相关数据，是一个基于互联网的地理信息系统(WebGIS)。它提供了一个可以供用户交互的可视化界面，用户在这个界面可以清晰直观的看到空间数据。另外它还链接了很多的数据库，包括Mysql和SQL Server以及上面提到的Map 3D。

### 1.2.2国内GIS发展情况

自动国外的WebGIS迅猛发展以后，国内公司不甘落后，也开始相继研究属于自己本国的GIS产品。到目前为止，国内GIS发展势头相对不错。主要有SuperMap、MapGIS和GeoBeans以及WebGIS On Hadoop[[[6]](#endnote-5)]。

SuperMap是北京一家以GIS著称的制图公司制作的GIS平台，SuperMap是中国主流的GIS软件之一，它通过独立化各个模块，减少了各个模块之间的耦合性。此外SuperMap支持很多主流的OS。可以在不同的操作系统之间进行移植。更重要的是它采用FastDFS文件系统来保存地图数据。

MapGIS 和Autodesk相仿，同样支持三维建模。它将GIS和摇动感知相结合，不可以分析地表上层，更是可以对空中以及地表下层的地理影像数据进行采集和分析,MapGIS同样可以实现跨平台GIS的开发，并在可以存储栅格地图、图片数据等。

GeoBeans是由中国科学院遥感应用研究所和北京中遥地网信息技术有限公司开发的地理信息系统[[[7]](#endnote-6)]。GeoBeans采用了JAVA构件的模型，可以移植在任意的平台上，另外该系统更是可以应用于军事领域，在视野上用户可以不断缩小，从世界地图一直缩小至亚洲、中国、北京并直至60厘米分辨率的层次。最重要的是它采用分布式资源查询，可以实现面向多种架构的地理信息系统的部署。

WebGIS On Hadoop是中科院开发设计的GIS系统，它将收集到的地图影像，光栅数据以及瓦片数据以打包的方式存在分布式文件系统HDFS中，本次设计所采用的方法就是借鉴此处的设计思想，通过数据+索引的方式，将海量数据存储在分布式系统中。

## 1.3论文主要工作

论文主要包括两个方面的内容，一是存储瓦片数据、瓦片索引数据以及金字塔元数据，并在存储瓦片数据时对比了打包文件存储和瓦片作为独立文件存储的优劣，并且在存储瓦片索引数据时采用了四叉树编码模型进行瓦片存储；二是设计并实现瓦片数据管理工具。论文主要工作有以下几点：

(1)研究国内外GIS产品的发展现状。

(2)存储MerpdfJpg072801金字塔下的瓦片数据，并分析并讨论了不用的存储方法之间的异同，最终采用瓦片打包文件的方式存储瓦片数据

(3)存储MerpdfJpg072801金字塔中瓦片索引Value值，并采用四叉树编码实现瓦片的近邻存储。

(4)设计实现瓦片数据管理工具，实现瓦片出入库、编辑、目录浏览及预览等功能。

(5)功能测试，对管理员登陆、cookies保存登陆状态、瓦片操作功能进行测试，检查是否与预期功能一致。

## 1.4章节安排

第一章首先介绍了项目的研究背景，还有GIS在国内外的研究现状，最后说明了论文的主要工作和每一章的安排。

第二章详细介绍了项目开发中所用的一些重要的技术，包括Hadoop、HBase、HDFS以及Phoenix等，还有在开发Web工程中用到的Tomcat服务器、Jsp和Servlet技术。

第三章首先介绍了瓦片数据在HDFS中的存储，瓦片索引在HBase中的存储以及金字塔元数据在Phoenix中的存储；之后在此基础上介绍如何开发能够管理这些瓦片数据的管理工具，从瓦片管理工具的每一个功能入手，详细介绍了功能的作用和实现。

第四章在第三章设计的基础上，对每一个开发阶段中设计的功能进行了测试，测试包括正确情况下的效果图还有错误情况下的处理机制。

第五章对本次毕设所做的工作进行了总结，点明了自己在开发过程中的所做的成果和不足之处。

# 

# 第二章 相关技术分析

## 2.1瓦片数据存储相关技术

### 2.1.1 Hadoop分析

Hadoop是Apache基金会收购的一个开源分布式基础架构[[[8]](#endnote-7)]，用户可以很容易的在开发文档的帮助下搭建一个Hadoop的开发环境。Hadoop有两个最重要的模块，一个是HDFS，它可以解决大量数据的贮存问题；另外一个是MapReduce，通过它可以从大量数据中提取有用的信息，是分析数据的基石。Hadoop从实际出发实现了从一台计算机处理数据到多台计算机并发处理数据的转变。

Hadoop集群搭建一共给出了三种模式，本次设计采用其中的伪分布模式。在这种模式下HDFS中的各项组件都运行在一台虚拟机中，从理论上实现了集群到单个机器的转化。其他的两种模式为单机模式和完全分布模式，这两种均不适用于本次设计，单机模式下不会运行HDFS进程，不适合用来开发设计，但是可以用来调试MapReduce程序。完全分布模式需要多台计算机搭建集群，但是在开发时，没有多余的PC机去实现这个操作。作为一个分布式系统架构，Hadoop具有以下几个优点：

1. 扩容能力

Hadoop可以稳定的存储和处理千兆字节(PB),并且支持实时的往集群中添加和删除节点，以便增强Hadoop的存储和运算能力。

1. 成本低

Hadoop集群可以通过普通的计算机组成去存储和处理数据，并且Hadoop还是一项开源的项目，所以搭建Hadoop集群不会浪费大量经济。

1. 高效率

Hadoop集群通过将海量的数据进行分发，分给集群中的不同计算机，可以让数据所在的节点并行的处理它们，从而使得大数据的计算变的十分节省时间。

1. 可靠性

Hadoop在集群内部会自动维护数据的多份副本数量。当Hadoop集群中某个节点由于某种未知原因突然无法继续工作，HDFS将会把重新分配任务。而且如果MapReduce的计算任务失败，也会重新开始分配任务。



图2.1 Hadoop的生态圈

Hadoop以分布式文件系统HDFS和MapReduce为核心，但是在Hadoop的生态圈中还有其他许多的项目，包括Ambari、Zookeeper、HBase和YARN等。如图2.1所示，图中的每一个模块都是Hadoop不可或缺的一部分。Ambari可以用来监控Hadoop集群；HBase提供非关系型的数据库接口等。这些模块使得Hadoop在互联网行业是十分受欢迎。

### 2.1.2 HDFS分析

HDFS是Hadoop中的核心模块，是解决海量数据存储的基础，它是一个分布式文件系统[[[9]](#endnote-8)]。与传统的文件系统相比，具有很多的优点：

1. 高存储

HDFS作为分布式文件系统，存储的单个文件的容量可以达到千兆字节，并且文件数据的总容量可以达到PB级别。而且随着Hadoop集群中的节点数量的增多而增加。

1. 通透性

HDFS的读写操作实际上是通过网络来访问其中的文件，但是在程序和使用者的角度来看，就像在访问本地的磁盘。

1. 容错性

HDFS在集群内部会自动维护数据的多份副本数量，当Hadoop集群中某个节点由于某种未知原因突然无法继续工作，HDFS会将该节点上的数据通过它们的副本所在的节点进行创建，并执行在运行正常的DataNode上。

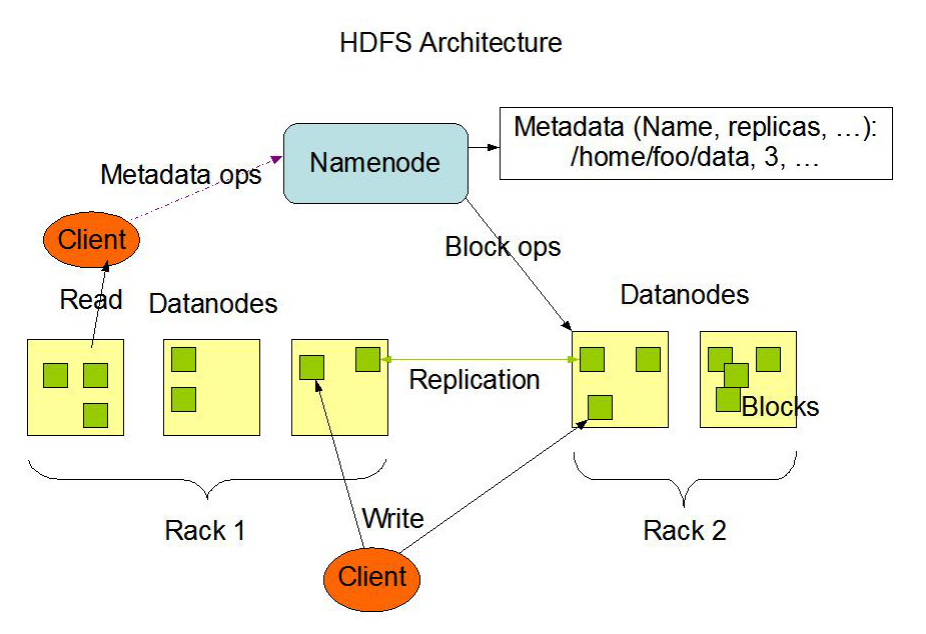


图2.2 HDFS 结构图

图2.2展示了HDFS中各个模块之间的协同工作流程以及所包含的组件。HDFS所含三大组件，NameNode是HDFS的核心，掌管整个文件系统中的目录信息，包括目录中所含文件的信息，除此之外，它还需要响应Client的请求信息，对Client提出的处理请求给与相应的答复。Secondary NameNode是NameNode的助手，它会在固定的时间更新NameNode保存的元数据信息，不让其数据丢失。DataNode是存储数据的基础，HDFS中的文件都是存储在Datanode中，并且是通过Block的形式存储，本次设计采用的Block大小为128M，文件在存储时大于这个大小的将被分为2块，小于这个大小的文件也不会完全占用这128M。除此之外，HDFS为了容错，会自动把保存在DataNode的数据备份至其他的节点出防止由于该节点的死亡导致数据丢失。

### 2.1.3 HBase分析

HBase是Hadoop集群中的数据库[[[10]](#endnote-9)]，由于HDFS的存在，使得HBase也可以存储海量数据。它将HBase中的数据表存在HDFS中，之后用MapReduce分析这些数据，Zookeeper和NameNode一样是HBase的核心组件。

表2.1 HBase表数据存储模型

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RowKey | Timestamp | Column Family | | | Column Family | |
| Column1 | Column2 | Column3 | Column1 | Column2 |
| Rk\_key1 | T1 | **...** |  |  | **...** |  |
| Rk\_key2 | T2 |  | **...** |  |  |  |
| Rk\_key2 | T3 |  |  | **...** |  | **...** |
| **...** | **...** |  |  |  |  |  |

表2.1表示HBase数据表的存储模型，一共由三部分组成，包括行键(RowKey)、列族(Column Family)和时间戳(TimeStamp)。其中RowKey行键是数据表中的主键，不能为空，通过主键可以直接定位到唯一的记录。HBase中通过RowKey和Column确定唯一的存储单元。在创建表时可以指定Version，即同一个存储单元可以有好多个数据，但是不同的数据的时间戳不同。



图2.3 HBase结构图

如图2.3表示HBase的各个模块之间的关系已经模块的工作原理，其中最重要的是Zookeeper和Master，Master是负责给Region Server分配Region的，假如当有一个Region发生故障，Master会马上给这个Server重新分配一个。如果Master发生故障的话，Zookeeper就会起作用，它的ME机制会保证当前有且只有一个Master会起作用。

### 2.1.4 Phoenix分析

Phoenix也是Apache的项目之一，但是并不是Apache开发的，而是由一个叫saleforce的公司开发的[[[11]](#endnote-10)]。传统的HBase不支持数据库的SQL语句查询。为了解决这个问题，在HBase上构建了一个能够用于SQL查询数据表的数据库，这就是Phoenix。

Phoenix是用Java编写的项目，它将传统的SQL语句转换为可以操作HBase数据库的专用语句，去操作HBase数据库创建表，插入数据等。用Phoenix查询的效率十分高。

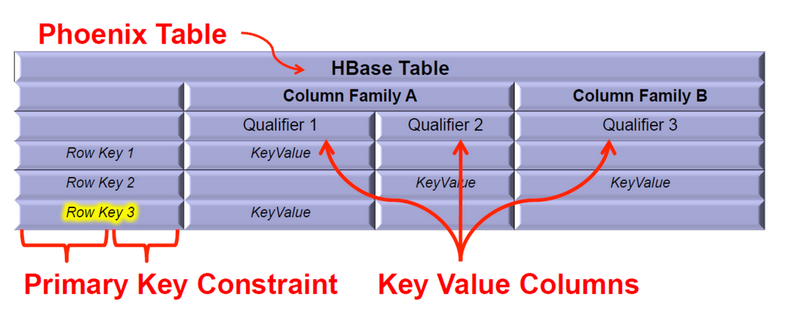


图2.4 Phoenix与HBase的映射关系

图2.4说明了Phoenix和HBase数据库之间是如何映射的，在Phoenix中的数据表与HBase表的关系就是图2.4所示一样，将Column Family去掉，把所有的Column都当成一个字段去存储。

## 2.2 基于web数据管理工具相关技术

### 2.2.1 JSP分析

JSP技术是Web工程中的最重要的技术，服务端的页面就是用JSP来完成的，动态网页的开发离不开它的支持。通过JSP的规范方法，可以在传统的纯HTML语言的静态页面中嵌入Java程序，可以简化页面中的程序代码量[[[12]](#endnote-11)]。

在Web服务器中通过引擎来编译执行JSP页面中的代码，并且在执行之后转化为HTML页面响应给浏览器。

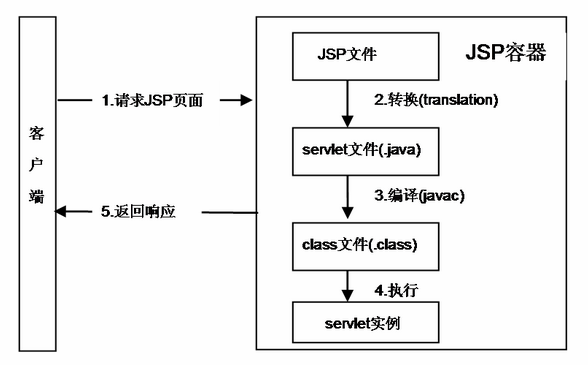


图2.5 JSP页面流程

JSP的规范包括JSP标签的语法结构，该标签以“<%”开始，在这个标签内部可以书写正常的Java代码来实现基本的业务逻辑。JSP的元素种类较少，可以分为下面三种：

1. 基本元素：JSP的基本元素里面可以写Java代码，开始结束标签的不同可以写定义变量，内容注释等信息。
2. 指令元素：指令元素是告诉JSP引擎在编译时需要做的事情，常用的指令元素有include指令，它可以将指定页面内容直接加载进来；page指令可以导入相应的Java库等。
3. 动作元素：属于服务器端的JSP元素，它用来标记并控制Servlet引擎的行为，主要有include动作和forward动作。

以上三种就是JSP的元素，通过这三种元素，我们可以在JSP页面开发自己所需的各项功能。

### 2.2.2 Servlet分析

Servlet本质是一个Java程序，但是和传统的Java代码不一样，它有着自己的规范写法[[[13]](#endnote-12)]，不需要传统意义上的Main函数，在Servlet代码中必须重新它所继承的父类HttpServlet中的doGet方法或者doPost方法中的任意一个。

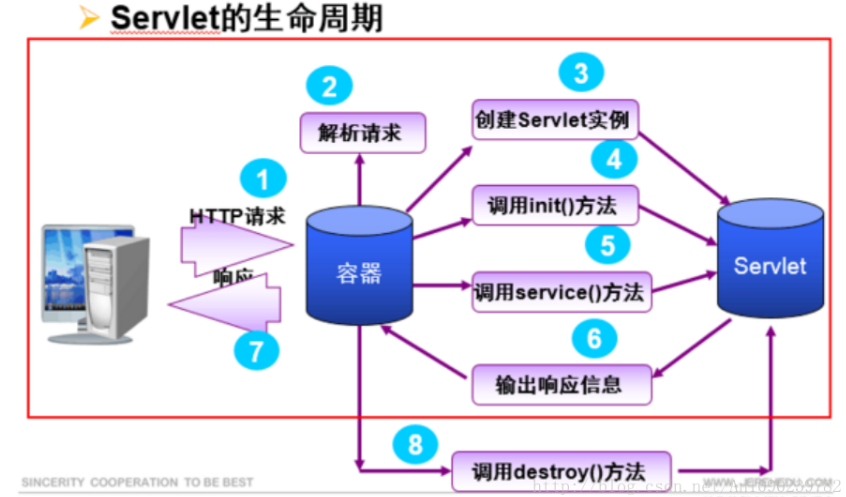


图2.6 Servlet生命周期

图2.6表示Servlet的工作原理，从图中可以知道，当client通过浏览器发出一个HTTP请求给Web服务器时，Web服务器根据配置文件中对应属性直接定位到相应的Servlet类中，在加载这个类之前，判断这个类是否已经被装载过，如果没有则创建Servlet实例并执行初始化方法，并执行里面的Post或者Get方法处理HTTP请求并返回结果。最后在服务器被停止或重启之前，调用destroy()方法释放资源。

### 2.2.3 Tomcat服务器分析

Tomcat是Web服务器的一种，由于其简单容易，操作清晰并且开发的特性，被很多人所喜爱，尤其是开发人员。作为一个轻量级服务器，可以简化很多开发人员的操作，配置也相对简单，默认端口采用8080端口，可以在配置文件中进行修改。也是Apache基金会的项目之一[[[14]](#endnote-13)]。

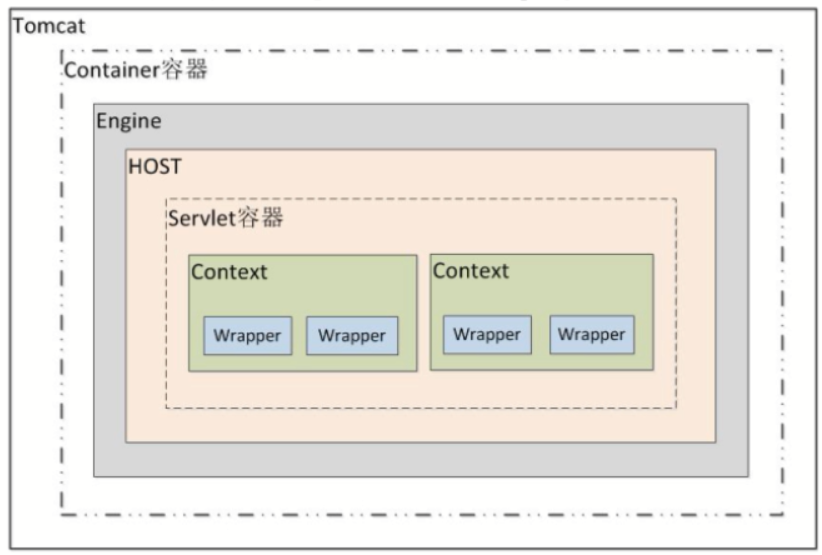


图2.7 Tomcat容器等级

Tomcat采用嵌套容器的方式，一共有四种容器等级，如图2.7。包括COntainer容器、Engine引擎容器、HOST容器和Context容器。其中Context容器对应与我们的Java Web工程项目。里面所包含的Wrapper是项目中的包括的所有具体Servlet。

## 2.3本章小结

本章主要对开发中用到的技术进行了详细的介绍，包括云计算平台Hadoop及其它生态圈中的相关组件，包括存储数据的HBase数据库技术，还有分布式文件系统HDFS的介绍以及构建在HBase之上的SQL层Phoenix等。还有开发Web工程中需要用的Jsp技术和Servlet技术，最后介绍了运行Web工程所需要的Tomcat服务器。

在介绍这些相关技术时，从技术的原理入手，对其的工作流程以及各个模块之间的工作机制进行了分析和介绍，并且还扩展了与其相关的有关技术，比如在介绍Hadoop平台时，还介绍了Hadoop生态圈中的一些技术。

本章通过对以上技术的详细介绍，为分布式瓦片数据管理系统的设计与实现打下了坚实的基础。

# 第三章 详细设计

## 3.1瓦片数据存储方案实现

### 3.1.1 HDFS存储瓦片数据打包文件

HDFS作为分布式文件系统，最简单的存储方法是把每一张瓦片地图数据存成一个独立的小文件，并根据其元数据信息对其命名，比如“MerpdfJpg#2#3#6.png”，其中“MerpdfJpg”是瓦片数据所在的金字塔，“2#3#6”表示瓦片数据的级别、行号及列号，“png”表示瓦片数据图像格式。如图3.1所示：



图3.1 瓦片作为独立文件的存储方式

这种方法的优点在于逻辑十分简单易懂，用户不需要做任何工作，只需要把所有的瓦片数据按要求放好，之后的工作都交给HDFS去做，包括存储文件的各种信息。

然而这种存储方式的弊端也很明显。一旦瓦片数据越来越多，用户想要定位到其中一个瓦片文件的时间将会非常耗时。另外，在大多数的文件系统中，当用户查询一个文件时，文件系统会获取到文件名，之后开始从第一个开始按照顺序线性往下查询，一旦文件数量增多，将花费很长时间来找到这个文件。

此外，文件系统在存储文件时并不是存储文件的大小，而是根据其文件系统的块大小来定，一般是其整数倍。比如一个文件系统的块大小为4KB，当一个文件大于4KB，比如说这个文件为6KB时，存储这个文件总共需要2个块，这样就浪费了很多的空间。并且文件越小，浪费的空间就越大。而且瓦片数据的大小一般都只有一千多K。由此可见，将瓦片作为独立的文件进行存储是不可取的。

在GIS中，最常用的方式是“打包文件存储”[[[15]](#endnote-14)]，也是此次设计所采用的方式，这种方案是说先将多个瓦片这种小文件打包成一个大文件，最后在把打包后的大文件存储在HDFS上。在本次设计中，由于HDFS的BLock Size为128M。故而将瓦片数据进行打包后的文件大小也是128M，这样就不会浪费文件系统的空间。

在打包瓦片数据的过程中，也并不仅仅是直接将瓦片往一个文件中存储，为了之后读取方便，添加了很多的对瓦片的描述字段，如图3.2所示，其中魔数是为了区分两张瓦片的位置所加的。瓦片打包文件中不考虑瓦片添加的顺序，所以如果单独添加瓦片可以直接在瓦片打包文件的末尾进行添加。



图3.2 瓦片打包文件结构

在HDFS中，为了以后方便的存储不同金字塔以及不同金字塔版本的瓦片数据，首先根目录下有一个/tilePyramid目录用来存储不同的金字塔，如瓦片金字塔1，瓦片金字塔2，在瓦片金字塔目录下存储的是该瓦片金字塔的所有版本，在瓦片金字塔版本的目录下就是存储瓦片数据的打包文件。图3.3描述了瓦片金字塔的存储目录结构[[[16]](#endnote-15)]。

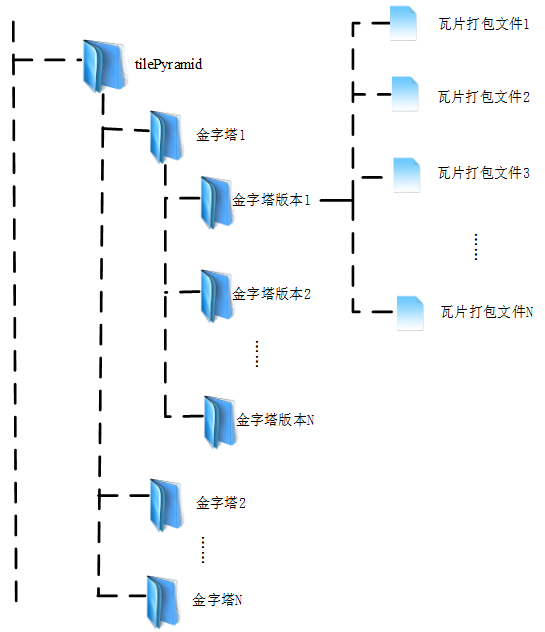


图3.3瓦片存储目录结构

从图3.3中可以看出瓦片存储目录的层级结构，所有的瓦片金子塔都存储在一个tilePyramid的根目录中，每个金字塔目录下面还有各自的金字塔版本号，在金字塔版本号目录的下方才是瓦片所在的打包文件。

在HDFS中实际存储瓦片数据时，所有瓦片金字塔都存储在/tilePyramid/目录下，如图3.4所示，在瓦片金字塔目录下的是所有瓦片金塔的版本，以MerpdfJpg072801金字塔为例，在/tilePyramid/MerpdfJpg072801/下面有一个当前金字塔的版本，在该目录下就是对应的瓦片数据的打包文件，打包文件的命名采用系统的当前时间来定，可以防止文件重名。



图3.4 瓦片金字塔目录结构

图3.4中的两个目录结构分别表示两个不同的金字塔，Geoglobal0803和MerpdfJpg072801，每一个金字塔的对应的目录下方存储着该金字塔的所有版本，在本次设计中，统一制定金字塔下的版本号为：MerpdfJpg072801@20180331144716000，它是由“金字塔名称+@+时间”构成的，在金字塔版本的目录下为瓦片打包文件。打包文件的大小为128M时，表示文件已经写满，需要重新创建一个新的文件。本次设计中的2万张瓦片数据一共打包两个文件。

### 3.1.2 HBASE存储瓦片索引数据

仅仅在HDFS中存储瓦片数据，客户端是无法访问的，因为用户无法找到想要的瓦片所对应的位置。所以，为了知道每张瓦片的精确位置，将其索引信息存储在HBase数据表中，存储模型如表3.1所示：

表3.1 HBase存储数据的模型

|  |  |  |  |
| --- | --- | --- | --- |
| Row key | TimeStamp | Column Family | |
| Column1 | Column2 |
| rk\_001 | T1 | Contents1 | Contents2 |
| rk\_002 | T2 | Contents3 | Contents4 |
| ... | ... | ... | ... |

本次设计中主键为瓦片的索引行键，由瓦片金字塔名称、层级和行列号得出的四叉树编码组成，如表3.2所示。瓦片金字塔名称为瓦片所处的金字塔名。层级为瓦片所处的层级（或称级别）。行列号得出的四叉树编码是通过计算得出。

表3.2 瓦片索引key格式

|  |  |  |
| --- | --- | --- |
| 瓦片金字塔 | 层级 | 行列号得出的四叉树编码 |

瓦片索引值存储瓦片在打包文件中的位置信息，它由四个字段值组成，包括瓦片所在文件名、偏移量、瓦片大小和瓦片删除标志位，如表3.3所示：

表3.3 瓦片索引Value格式

|  |  |  |  |
| --- | --- | --- | --- |
| 瓦片所在文件名 | 偏移量 | 瓦片大小 | 删除标志位 |

瓦片所在文件名这个字段可以在用户想要定位瓦片时，直接获取瓦片所在的位置，之后通过偏移量定位到指定瓦片的准确位置。瓦片大小为在打包文件中瓦片的实际存储大小。这个字段可以直接比较两张瓦片在物理存储上是否邻近而不用通过对打包文件进行操作，另外在HDFS上读取瓦片数据时，可以通过这个字段计算出瓦片图像数据的大小进而直接读取瓦片数据，不必先读取瓦片的大小在读取瓦片数据，可以减少瓦片的读取时间。删除标志位是一个标志位，用来表示该瓦片是否已经被删除，当客户端删除一张瓦片时，将该标志位设置为true表示瓦片已经被删除，默认情况下这个字段为false。

瓦片索引表的名称由：瓦片金字塔名称+“\_TileIndex组成。不同的字段通过“#”分隔。

在表3.2中提到将行列号采用四叉树编码进行存储，这是因为HBase数据表中的数据是按照字典序进行排序的。而在查找瓦片的时候，更多的时候并不仅仅只是访问一张瓦片，而是同时访问多张瓦片，并且多张瓦片有一个共同特点，那就是在空间上是邻近的。这就要求在设计的时候，在这种情况下，能够将空间上邻近的瓦片在存储空间上也邻近，那么在读取瓦片时，将会节省大量的时间。利用HBase的这个特点，将瓦片的编号利用四叉树进行编码就可以解决这个问题。



图3.5 瓦片数据

图3.5是16张瓦片，按照一般的瓦片编号，则是以瓦片所在的行和列为该张瓦片的标识，那么图中阴影部分的四张瓦片的编码分别为(0,0)、(0,1)、(1,0)、(1,1)，但是这四张瓦片在空间上是近邻的，但是如果按照这种方式编码瓦片，则瓦片索引的存储空间中该四张瓦片并没有相邻。采用四叉树的编码方式如图3.6所示：

图3.6 瓦片的四叉树编码方式

四叉树编码方式就是按照图3.6的方法对每张瓦片进行编码，空间上近邻的瓦片采用“C”型方式进行编码，坐标为(0,0)的瓦片标号为0，坐标为(0,1)的瓦片标号为1，坐标为(1,0)的瓦片标号为2，坐标为(1,1)的瓦片标号为3。当分辨率增加时，只需要将现有的区域分割为四个区域，按照上述的方式进行编码即可。那么每一层的瓦片标识为父索引标识加上子索引标识。



图3.7 四叉树编码的树状结构

图3.7中所示为四叉树的树状结构，每一个区域可以分为4个子区域(0,1,2,3),将子区域在分为四个子区域(00,01,02,03),该模型和金字塔瓦片模型相对应。每一张瓦片对应一个区域，不同的分辨率对应不同的层级，将层级加上索引标识即可以表示一张瓦片。例如，图3.5所示的阴影部分如果采用四叉树编码的方式，可以将该四张瓦片编码为3\_010,3\_011,3\_012,3\_013，这样将这些值作为瓦片索引的行键进行存储，在存储空间上也就相近了。

### 3.1.3 Phoenix存储金字塔元数据

传统的HBase不支持数据库的SQL语句查询。为了解决这个问题，在HBase上构建了一个能够用于SQL查询数据表的数据库，这就是Phoenix。

Phoenix是用Java编写的项目，它将传统的SQL语句转换为可以操作HBase数据库的专用语句，去操作HBase数据库创建表，插入数据等。本次设计采用Phoenix存储金字塔元数据，利用JDBC连接Phoenix，通过提供的API接口将XML中的元数据字段信息存储在Phoenix的数据表中。

数据表的名称为Metadata，各项字段名如表3.4所示:

表3.4 金字塔元数据字段名

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 含义 |
| name | varchar | 瓦片金字塔的名称 |
| dataType | varchar | 金字塔数据类型 |
| schemaType | varchar | 投影类型 |
| tileFormat | varchar | 瓦片压缩格式 |
| tileSize | integer | 瓦片大小 |
| version | varchar | 金字塔版本号 |
| updateTime | varchar | 金字塔更新时间 |

字段name是金字塔的名称。

字段dataType的数据类型总共有四种，包括影像瓦片(DATA\_RASTER\_IMAGE)、DEM瓦片(DATA\_RASTER\_DEM)、电子地图瓦片(DATA\_RASTER\_MAP)和矢量数据瓦片(DATA\_RASTER\_VECTOR)，本次的瓦片数据均属于影像瓦片类型。

字段schemaType为投影类型，典型的投影类型为墨卡托投影(Mercator)和地理坐标投影(Geodetic)，本次的投影类型为(SCHEMA\_GEODETIC)。

字段tileFormat是瓦片的压缩格式，瓦片数据本质是一张张的图像数据，所以该字段的取值常为“JPG”和“PNG”，本次为PNG格式的瓦片数据。

字段tileSize为不是每张瓦片的大小，而是瓦片打包文件的大小，本次采用和HDFS的BlockSize一样的值128M。

字段Version为金字塔的版本号，由金字塔的名称和版本号组成，版本号的形式为系统的17位毫秒级时间，例如本次MerpdfJpg金字塔的版本为MerpdfJpg072801@20180331144716000。

字段updateTime为金字塔的更新时间，将最后一次修改金字塔的时间以字符串的形式存储至数据库中。

## 3.2 基于web的瓦片数据管理工具的实现

### 3.2.1管理工具的界面设计

瓦片数据管理工具是方便开发人员管理瓦片数据的一个后台管理系统，通过一个简单的web浏览器，开发人员可以远程对瓦片数据进行增删改查的基本操作。针对这个特性，本次开发的管理工具共有两个主界面：

1. 管理员登陆界面
2. 瓦片操作管理界面

管理员登陆界面要求客户端提供一个管理员账号和密码，两者均正确才可以进入瓦片操作管理界面，防止一些其他非相关人员进行误操作。界面的设计采用div+css的样式，通过对div属性的设置，使内部的控件更漂亮，数据的接收采用form表单的形式，通过action属性，将这个页面的数据提交给指定的servlet类，在form表单下方有一个CheckBox复选框，询问客户端是否需要保存登陆状态，在勾选该框的前提下，客户端下次可以不用输入账号和密码，直接点击登陆即可，界面的效果如图3.5所示：

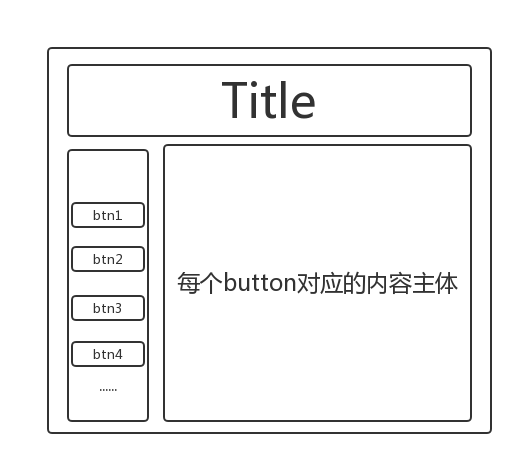
 

图3.8 管理员登陆界面 图3.9 瓦片操作管理模型图

瓦片操作管理界面同样采用div+css的样式，利用div将浏览器分为三个盒子，模型效果如图3.6所示，上方的div盒子为管理工具的标题栏，左侧的div盒子为瓦片菜单栏，里面为实现瓦片操作的基本按钮，在右侧的div盒子中利用ifame标签实现不同button按钮之间内容主体的实时切换。最终的瓦片操作管理界面图如3.7所示：



图3.10 瓦片数据管理工具界面

### 3.2.2功能实现

瓦片数据管理系统是为了开发人员方便管理瓦片数据的。基于这一目的，本次设计的瓦片管理工具总共实现了以下七个功能：

1. 管理员登陆

在功能通过一个form表单接受数据，当客户端填写数据之后，通过提交按钮将数据以属性method中的方式发送到属性action填写的地址处，该处地址指向的是一个servlet的类，这个类通过继承HttpServlet重写doPost或者doGet方法，在方法里面通过jsp内置对象request获取用户的账号和密码，并进行逻辑判断，如果账号密码均填写正确，则跳转至瓦片操作管理界面，否则通过内置对象response将错误消息告知登陆页面，在login.jsp中通过JavaScript获取response传来的错误信息并提示给客户端。

1. 保存管理员登陆状态

在管理员登陆界面可以勾选是否保存登陆状态，默认情况是保存。当用户不修改默认状态时，LoginServlet会首先判断管理员账号和密码是否正确，如果均正确，servlet会创建cookies保存账号和密码，保存时间为1天。如果用户不想保存登陆状态，则将之前保存的cookies时长设置为0，表示马上销毁这两个cookies。在Login.jsp中获取所有cookies表单，获取指定账号密码，显示在控件中。

1. 瓦片的入库

瓦片入库的意思是说将一个指定的瓦片数据放入HDFS的指定位置，并将瓦片索引存储至HBase的数据表中。瓦片入库的思想和上述3.1.1节中瓦片数据的存储是一致的，将瓦片数据打包成一个文件，存储在HDFS的目录中，由于现在的操作是以瓦片的个体，如何直接打包成文件，所以可以通过追加的方式来完成瓦片数据的打包工作，即第一张瓦片入库的时候创建一个文件，之后其他的瓦片入库的时候直接将瓦片的数据(包括哨兵元素、瓦片实际地址、瓦片大小和瓦片图像数据)追加在这个文件的末尾，当文件的大小达到128M时，关闭文件的输出流，重新创建一个文件，如此循环。瓦片入库的流程图如3.8所示：

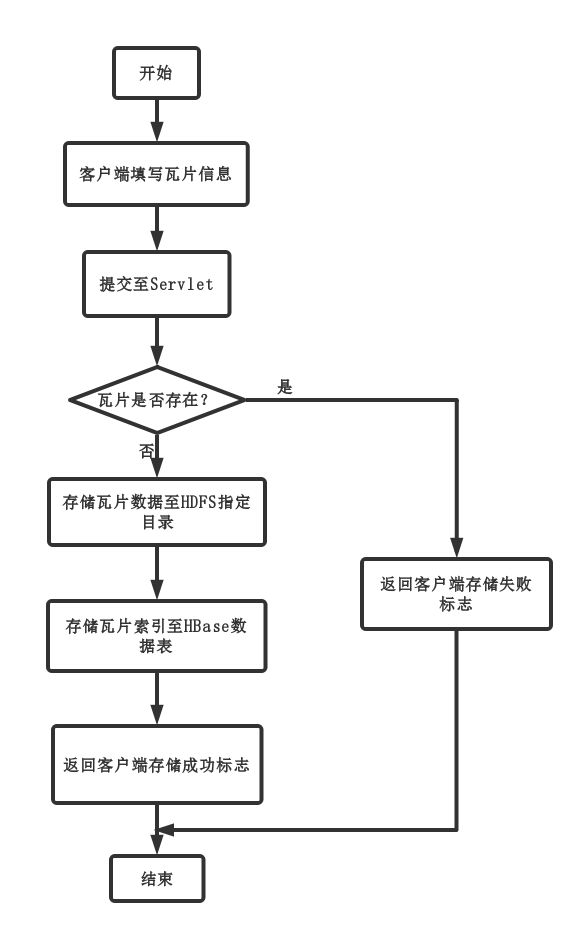


图3.11 瓦片入库流程图

1. 瓦片的出库

瓦片出库的含义和入库正好相反，在客户端的form表单中填写需要出库的瓦片信息，包括瓦片所处的金字塔以及瓦片的名称，之后将表单数据提交给servlet处理，servlet将金字塔名和瓦片行列号得到的四叉树编码封装成瓦片索引key，在HBase数据表中根据RowKey进行查询，得到瓦片索引Value，根据瓦片索引Value中的四个字段，在瓦片打包文件中读出相对应的瓦片数据，并将瓦片数据写进Mysql中瓦片数据出库的流程图如3.9所示：

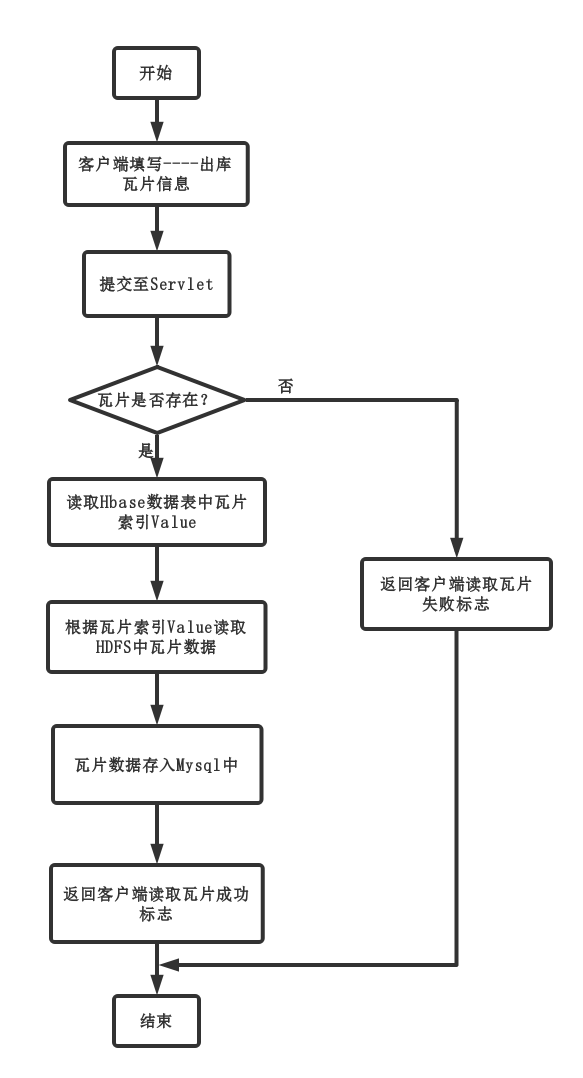


图3.12 瓦片出库流程图

随着瓦片数据出库越来越多，将瓦片数据存入Mysql中方便管理取出的瓦片，首先在Mysql中创建db\_webtile的数据库，在数据库中创建名为tile\_info的数据表，表中各个字段如表3.5所示：

表3.5 Mysql字段表

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 含义 |
| tileName | varchar(20) | 瓦片名称 |
| tileSize | integer | 瓦片大小 |
| tilePyramid | varchar(30) | 瓦片金字塔 |
| tileFileName | varchar(20) | 瓦片所处文件名 |
| tileImage | blob | 瓦片图像数据 |

tileName为瓦片的名称，在Mysql的数据表中为主键，不能为空(not null)，瓦片的名称例如：0#0#0.png,分别有瓦片所处层级、行列号组成。

tileSize为瓦片的大小，单位为字节Byte。

tilePyramid为瓦片所处的金字塔名称，在本次设计中共有两种金字塔，Geoglobal0803和MerpdfJpg072801,样例中的所有瓦片均属于以上两种瓦片。

tileFileName为瓦片所处的文件名，文件名为瓦片打包文件的名称，由瓦片索引Value中可以获得。

tileImage为瓦片的图像数据，由于计算机保存图像数据的方式是将其转换为二进制，所以该字段的类型为二进制类型(blob),该类型将图像保存为二进制的01矩阵。

1. 对HDFS中的瓦片进行编辑

瓦片的编辑一般分为增删改查四个功能，但是由于瓦片增加和瓦片入库的功能相似，瓦片的修改可以看出是瓦片的删除加瓦片的入库一起完成，即先将需要修改的瓦片数据删除，之后重新添加瓦片数据，所以瓦片的编辑下面只有两个功能：瓦片删除和瓦片查询。

瓦片删除采用设立标志位的方法，在HBase的数据表中的瓦片索引Value中添加一个isDelete字段，表示该瓦片数据是否已经被删除，true表示瓦片已经不存在，false表示瓦片数据没有被删除。通过这种方式可以简化瓦片的操作步骤，在瓦片的复原过程中，只需要修改一个字段并将其重新写入HBase表中即可，大大节省了时间。

瓦片查询在思想上和瓦片出库一致，只是瓦片查询不需要保存瓦片的信息，只需要将查询的瓦片数据信息显示在客户端的网页中，方便用户查看。具体步骤同瓦片数据出库。

1. HDFS文件系统的目录浏览

这个功能的含义是在客户端的页面中展示HDFS存储瓦片的目录结构，让开发人员清晰条理的看到瓦片的存储位置。

该功能实现的基本步骤分为三点：

1. 保存HDFS中每个目录结点。
2. 将目录结点的父节点和子节点记录下来，生成结点树
3. 遍历结点树，生成Html语言，显示在客户端。

首先新建一个文件结点FileNode，在FileNode中保存其自己的id和父结点id，还有nodeName，通过遍历HDFS的目录结构，生成所有目录下文件名和文件的FileNode，将其加入List<FileNode>列表中。之后新建一个文件树FileTree，在其中将List列表中的所有FileNode结点通过递归的方式生成HTML语言的目录树，显示在客户端网页中。 例如：在HDFS中根目录为tilePyramid下，记录tilePyramid的id为0，父节点为null，nodeName为tilePyramid，将这三条属性保存为FileNode结点，以此类推，递归遍历所有结点，生成List<FileNode>结点列表，在FileTree中循环List列表，生成以<ul>和<li>标签的HTMl语言，最后显示在界面上，如图3.12所示：

1. 瓦片图像数据的预览

瓦片图像预览功能可以看到指定的瓦片图片，通过指定瓦片金字塔和瓦片名称远程获取瓦片图像的二进制数据矩阵，之后将二进制数据通过response对象传给tile\_image.jsp页面，在页面中通过outputStream将图像数据输出至网页中显示，如果指定的瓦片数据不存在则给出错误提示。

## 3.3本章小结

本章从设计者的角度出发，详细的介绍了瓦片数据在HDFS文件系统中的存储，瓦片索引Value在HBase表中的存储以及金字塔元数据在Phoenix中的存储。并且对比了瓦片打包文件存储和瓦片作为独立文件存储这两种方式的不同以及优缺点。

之后为了方便管理这些瓦片数据，开发设计了基于web的瓦片数据管理工具，在3.2小节中着重介绍了数据管理工具所需要的功能以及前端的界面布局和后端的代码逻辑实现。

# 第四章 功能测试

## 4.1瓦片数据的存储功能测试

瓦片数据的样例一共有两个，即共有两种瓦片金字塔，分别为Geoglobal0803和MerpdfJpg072801。在测试的时候，瓦片数据存储功能测试采用MerpdfJpg072801，实现瓦片数据和瓦片索引的入库。在数据管理工具功能测试时采用Geoglobal0803，实现瓦片操作的功能检测。

在测试之前，首先将两种瓦片金字塔放在指定目录下：F:\功能测试：如图4.1所示。其中，MerpdfJpg072801放在瓦片数据存储功能测试目录下，Geoglobal0803放在瓦片数据管理工具功能测试目录下。



图4.1 瓦片金字塔目录结构

每个瓦片金字塔下层目录结构均是按照瓦片所处的分辨率层级放置，以MerpdfJpg072801金字塔为例，该金字塔下层有一个金字塔版本号MerpdfJpg072801@20180331144716000表示当前金字塔下只有一个版本，在该金字塔版本目录下，将瓦片数据分为了15个层级表示共有15级分辨率，每一级下面都有若干瓦片数据，相同行号的在同一层级目录，以“层级#行号#列号.jpg”的格式命名，如图4.2所示为瓦片的命名，该张瓦片表示在14层分辨率下第6653行的12517列。



图4.2 瓦片图像数据

### 4.1.1写瓦片数据和索引

1. 写瓦片数据测试

在MerpdfJpg072801中共有19818张瓦片数据，将这些瓦片数据打包成文件存在HDFS中的目录中，运行程序后观察HDFS的目录如图4.3所示：



图4.3 HDFS目录内容

从图4.3中可以看出19818张瓦片数据被打包成了两个文件，每一个文件名都是采用当前系统的17位毫秒级时间来命名的，打包文件的大小为128M，和HDFS的Block Size大小相同，从图中倒数第二行可以看出。观察目录结构可以看出瓦片打包文件的绝对路径在瓦片所属金字塔下的金字塔版本号下级目录中。与最初设计的预计要求相符。

1. 写瓦片索引值测试

瓦片索引分为两个部分:瓦片索引key(行键)和瓦片索引Value。瓦片索引key采用“瓦片金字塔#瓦片层级、行列号的四叉树编码”的形式存储；瓦片索引Value由“瓦片所处文件#瓦片开始偏移量#瓦片大小#瓦片是否删除标志”四个字段组成。两个部分组成瓦片的一条记录存储在HBase数据表中，功能测试结果如图4.4所示：

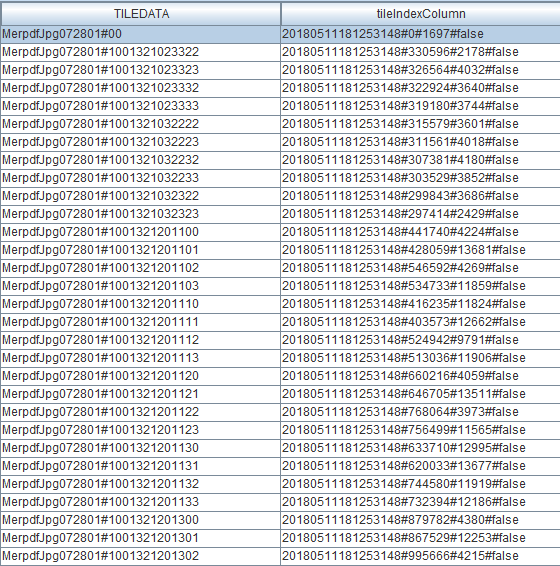


图4.4 瓦片索引存储表数据集

为了方便查看HBase数据库中的数据集，采用Squirrel SQL Client通过Phoenix连接HBase。在Phoenix中通过映射HBase中的表，使HBase中的数据表可以在Phoenix中显示，进而可以在Squirrel客户端使用SQL语言查看HBase表中数据集。例如，以本次存储瓦片索引数据的表MerpdfJpg072801\_tileIndex为例，首先在Phoenix中使用SQL语句创建表格，数据表名称与MerpdfJpg072801\_tileIndex一致，在填写字段时primary key会自动与HBase中表名一致的表的RowKey对应，如本例中Phoenix字段TILEDATA与HBase中RowKey对应，其余字段一一对应。每一张瓦片数据对应一条瓦片索引记录，即共有19818条瓦片索引数据集，图4.4列出30条记录，HBase采用字典序的排列方式排列数据集，例如第一条记录集图4.5所示，TILEDATA为MerpdfJpg072801#00，可以得知该张瓦片所在的金字塔为MerpdfJpg072801，“#”为不同字段之间的分隔符，“00”表示瓦片所在位置的四叉树编码。tileIndexColumn为HBase中的列名，其中“20180511181253148”表示瓦片数据在HDFS上的文件名，“0”表示该瓦片在瓦片打包文件中的起始位置，“1697”表示为瓦片数据在打包文件中一共占的字节数，“false”表示该瓦片没有被删除。



图4.5 瓦片索引第一条记录

综上所述：从瓦片数量上来看，19818张瓦片数据和瓦片索引全部存储成功；从瓦片存储内容来看，不管是瓦片数据的目录结构和打包文件还是瓦片索引的RowKey字段和瓦片索引Value字段均存储正确。所以写瓦片数据的功能已经可以达到预测目标。

### 4.1.2读取瓦片数据

在验证完写瓦片数据和索引之后，为了保证写入的瓦片数据正确，需要对HDFS中已经存在的瓦片数据进行读取操作。由于在读取瓦片数据时，首先是根据瓦片索引Value的字段去定位瓦片在打包文件中的位置。所以在验证读取瓦片数据时，将瓦片数据和瓦片索引值的功能测试统一测试。

首先指定MerpdfJpg072801中的一张瓦片名称，通过瓦片索引表和瓦片打包文件获取指定信息的瓦片数据，并与之前提供的瓦片进行对比。例如，在MerpdfJpg072801中选取一张名为14#6653#13522.JPG的瓦片图像如图4.6所示。



图4.6 测试的瓦片图像

首先通过瓦片名称14#6653#13522.JPG可知该瓦片在第14层级，并且处在第6653行的13522列的位置，由此可以生成瓦片位置对应的四叉树编码14013210233232212，其中前两位数字14表示瓦片数据所在的层级，后面的位数为层级数+1，表示瓦片在每一层级所处的位置(0-3)。将瓦片金字塔和四叉树编码组成瓦片索引Rowkey(tileIndexRowKey)为“MerpdfJpg072801#14013210233232212”，根据瓦片索引RowKey在HBase数据库的MerpdfJpg072801\_tileIndex表中查询该条记录，如图4.7所示：



图4.7 指定瓦片索引RowKey查询记录结果

由图4.7可以得知，该瓦片对应的瓦片索引记录存在，并且该瓦片的详细数据在HDFS瓦片打包文件名为20180511181253148中，并且瓦片数据在该文件62253587字节开始的位置处，共占用9054字节，且瓦片没有被删除。根据这些或许到的信息，在HDFS的指定位置进行查找，获取指定信息下瓦片数据的所有信息，并将瓦片图像的二进制矩阵重新写入本地目录中：F:\功能测试\瓦片数据存储功能测试\读瓦片数据测试\，查询到的瓦片数据如图4.8所示：

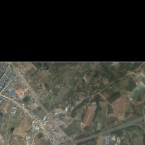


图4.8 测试结果的瓦片图像

比较图4.7和图4.8可知，两张瓦片相似率极高，可以认定为是同一张瓦片，即从HDFS中获取的瓦片是之前存进去的指定瓦片的图像数据，进而可以验证读瓦片数据和瓦片索引值的正确性。

### 4.1.3 存储瓦片金字塔元数据

以MerpdfJpg金字塔为例，将金字塔所有的字段信息都保存在XML文件中，通过读取XML中的文件内容，将其转换为NodeList列表，之后遍历循环NodeList中的每一个Node结点，把Node中的所有属性都存入Phoenix数据表中。结果如图4.9所示，共有7个字段信息已经全部存入数据库中。



图4.9 MerpdfJpg金字塔元数据信息

## 4.2数据管理工具的功能测试

瓦片数据管理工具总共有七大功能，并且在测试的过程中采用Geoglobal0803金字塔下的瓦片数据，在Geoglobal0803金字塔下有一个对应的金字塔版本号Geoglobal0803@20180331144716000，在该金字塔版本号目录下有6个瓦片层级，共有2731个瓦片数据，数据管理工具可以实现将瓦片数据进行个体单位上的存储，即指定在HDFS上存储瓦片数据和索引信息，详细测试如下：

### 4.2.1管理员登陆及保存登陆状态

瓦片数据管理后台需要指定的用户才能登陆，当管理员账号和密码均正确时才能跳转至瓦片操作管理界面。并且如果客户端选择保存登陆状态按钮，则在一天之内再次登陆界面则不需要输入账号密码。当输入错误的账号和密码时，客户端收到servlet的错误信息，提示密码错误如图4.10所示，输入正确则进入操作页面。当再一次打开登陆页面是，如果用cookies保存过用户名和密码，则进入登陆页面如图4.11所示，用户名和密码均自动填写至相应控件处。

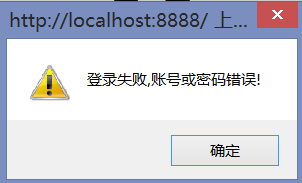
 

图4.10 登陆失败界面 图4.11 使用cookies保存登陆信息

### 4.2.2瓦片出入库

(1)瓦片入库测试

瓦片入库将一张指定的瓦片放入HDFS的目录下，并保存该瓦片数据的索引信息在HBase数据库中。如果瓦片已经存在则返回错误信息给浏览器，成功返回成功信息。例如：在Geoglobal0803中选取一张“5#12#38.JPG”的瓦片数据，将这张瓦片数据按照写瓦片数据和索引的步骤进行存储。由于存储的瓦片只有一张，为了不浪费HDFS的空间，依旧采用瓦片打包文件的方式存储瓦片数据。只需要在存储瓦片数据时判断是否有打包文件没有存满，如果有则打开该文件继续往文件最后追加内容即可，否则以系统当前时间创建一个新的打包文件去存储瓦片数据。以“5#12#38.JPG”这张瓦片为例，在客户端中填写该瓦片的基本信息如图4.12所示：



图4.12 瓦片入库信息

需要填写瓦片所处的金字塔名称、瓦片的名称以及瓦片的存储路径，“5#12#38.JPG”这张瓦片的金字塔名称为“Geoglobal0803”，存储路径采用测试的瓦片存储目录：F:\功能测试\瓦片数据管理工具功能测试\Geoglobal0803\Geoglobal0803@20180331144716000\5\12。之后点击提交，如果存储成功，服务器端会利用response内置对象的请求重定向给客户端浏览器发送一个error参数，其参数值为no，表示瓦片已经存储成功，成功的效果图如4.13所示。如果存储失败，即瓦片已经存在，则web服务器使用相同的方法给客户端发送error=yes的参数，失败的效果图如图4.14所示：

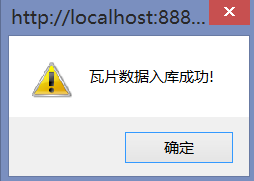
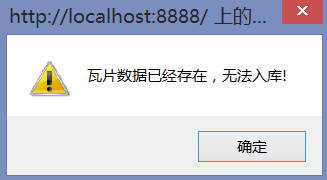
 

图4.13 瓦片成功入库图 图4.14 瓦片失败入库图

当瓦片成功入库时，在HDFS的/tilePyramid目录下会多出一个Geoglobal0803的子目录，在其子目录下增加一个对应瓦片所在金字塔版本号的目录，在目录下即可以看到成功上传的瓦片打包文件如图4.15所示：



图4.15 测试瓦片在HDFS的目录结构图

(2)瓦片出库测试

瓦片出库功能测试，客户端通过指定想要出库的瓦片数据信息，将查询获得的瓦片数据信息写入Mysql中。例如，以刚刚写入HDFS的瓦片数据“5#12#38.JPG”为例，将指定的金字塔信息和瓦片名称填入客户端浏览器的Form表单中，存储成功后可以看到客户端提示，如图4.16所示。之后查看Mysql服务器的db\_webtile数据库，查看tile\_info数据表，可以得到如图4.17所示的数据：

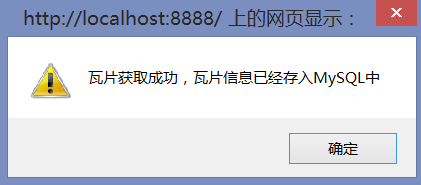


图4.16 测试瓦片出库成功信息

图4.17 测试瓦片在Mysql信息

从图4.17可以看出，tile\_info数据表中只有测试的这一条数据，一共包含5个字段。每一个字段都是描述“5#12#38.JPG”这张瓦片数据的，将tileSize和tileFileName两个字段对比图4.15中的相应信息可以看到是一致的，由此可以得到瓦片出库功能的正确性。

### 4.2.3瓦片编辑(删除+查询)

1. 瓦片删除测试

从本章3.2.2的(5)可以知道瓦片删除采用一个标志字段isDelete，该字段存储在瓦片索引Value的第四个字段，当其值为false时，表示瓦片没有被删除，反之已经被删除。以“5#12#38.JPG”瓦片为了，首先查看该瓦片在索引表中的记录，如图4.18所示：



图4.18 删除前瓦片索引记录

从图4.18可以看到目前瓦片tileIndexColumn列中的第四个字段值为false，表示该张瓦片没有被删除，当执行完瓦片的删除操作后，继续查看瓦片在索引表中的记录，如图4.19所示：



图4.19 删除后瓦片索引记录

从图4.19中可以看到先前的false值已经变为了true，表示该瓦片已经从逻辑意义上被删除了。当删除的瓦片不存在或者已经被删除了(tileIndexColumn的字段值为true)时，客户端提示相应的错误信息和上述错误信息一致。

1. 瓦片查询测试

瓦片查询的功能测试和瓦片出库一致，唯一不同点是瓦片查询只需要将描述瓦片的信息显示在客户端浏览器中，不需要存储在Mysql中，更加直观。所以测试过程不在此叙述，测试成功的结果图如4.20所示：



图4.20 瓦片查询信息

当瓦片查询成功时，客户端浏览器页面显示图4.20的内容，用户可以方便的查看当前瓦片的内容信息以及图像数据。

### 4.2.4瓦片目录浏览及预览

1. 瓦片目录浏览测试

瓦片目录浏览可以显示当前HDFS中的所有目录结构，以HTML语言的形式呈现在客户端，通过递归遍历HDFS的目录结构，对每一个目录结点保存其父结点ID、自己ID及目录名称，将所有的目录结点保存在列表中，之后生成HTML语言，如图4.21所示。与其对应的Web页面如图4.22所示：

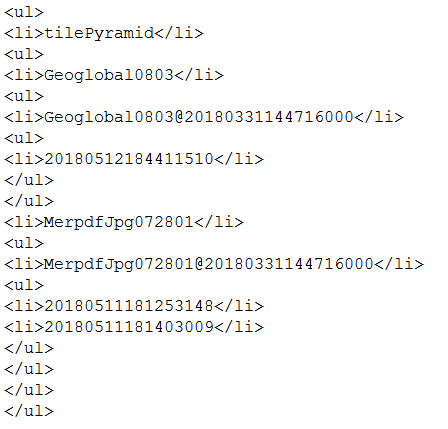


图4.21 目录结构的HTML语言

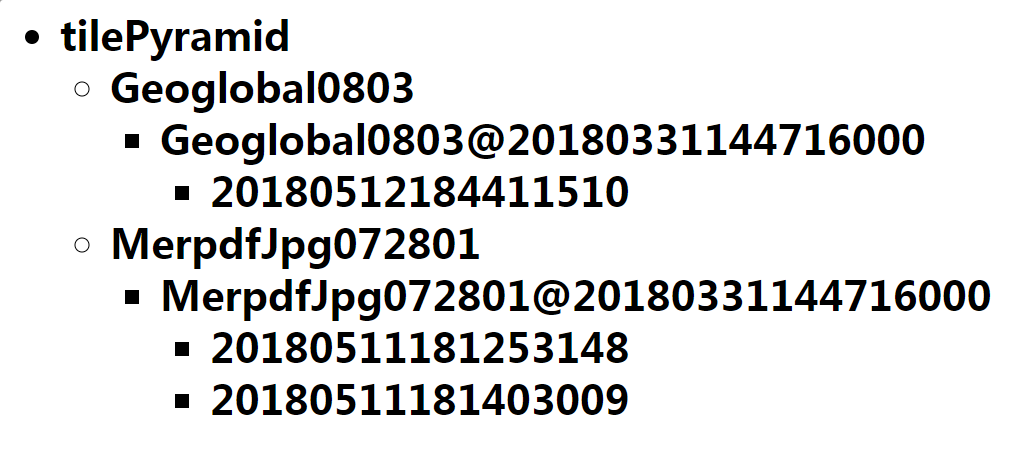


图4.22 HDFS目录结构页面

从图4.22可以看出，HDFS的根目录下有一个tilePyramid的文件夹,它有两个子文件夹对应两个瓦片金字塔Geoglobal0803和MerpdfJpg072801，在每个金字塔目录下都有一个对应的金字塔版本号目录，在版本号的目录下是所有瓦片的打包文件。

1. 瓦片预览测试

瓦片预览与瓦片查询功能相近，可以根据指定信息查看对应的瓦片图像，直接将图像以二进制数据的形式写入Web页面中，在客户端可以看到瓦片图像。测试与瓦片查询测试相同。

## 4.3本章小结

本章在开发设计的基础上，对设计好的每一个功能模块进行测试。在测试阶段，综合考虑所有可能出现的情况，包括错误情况下的处理和正确情况下的逻辑都进行了针对性测试。另外在叙述阶段，采用图文并茂的形式，对每一个功能的测试都做了详细的描述。

# 

# 第五章 结束语

本文主要介绍了两个方面的内容，一是存储瓦片数据、瓦片索引数据以及金字塔元数据，并在存储瓦片数据时对比了打包文件存储和瓦片作为独立文件存储的优劣，并且在存储瓦片索引数据时采用了四叉树编码模型进行瓦片存储；二是设计并实现瓦片数据管理工具。本文所做的工作如下几点：

(1)存储MerpdfJpg072801金字塔下的瓦片数据，并分析并讨论了不用的存储方法之间的异同，最终采用瓦片打包文件的方式存储瓦片数据

(2)存储MerpdfJpg072801金字塔中瓦片索引Value值，并采用四叉树编码实现瓦片的近邻存储。

(3)设计实现瓦片数据管理工具，实现瓦片出入库、编辑、目录浏览及预览等功能。

(4)功能测试，对管理员登陆、cookies保存登陆状态、瓦片操作功能进行测试，检查是否与预期功能一致。

由于时间和个人能力的缘故，虽然已经实现了毕设所要求的的基本功能，但是在很多的方面功能还是相对不足，有很多地方还有瑕疵，没有从长远的角度考虑问题。比如存储瓦片数据时没有考虑在写入瓦片时如果用户正好要读取瓦片这种情况；另外在瓦片管理工具中的功能也不全面，无法实时获取Hadoop集群中瓦片数据的信息，以及一些HDFS中节点的使用率等，另外在功能上没有实现批量处理瓦片这一情况，所开发的功能都是针对单张瓦片数据的。在以后的开发过程中，会尽量考虑所有情况的处理，保证系统的稳定性。

# 

# 致谢

转眼间，大学生涯已经接近尾声。马上就要步入研究生的门槛了，在此十分感谢在大学四年中帮助我的同学和朋友，让我这四年过得精彩。

毕业设计马上就要完成了，在这里我首先要感觉李龙海老师，我毕设论文的完成离不开李老师的细心指导。我在大三的专业课上认识了李老师，李老师生动的讲课方式，渊博的知识都非常令我钦佩。在之后的3年的研究生生活中，也希望李老师能够带我一起做项目，学知识。再一次向李老师表示衷心的感谢。

另外我还要感谢邱峰学长和朱耀耀学长，它们是我实验室的师兄。在毕设的开始阶段，在我遇到困难和重重问题无法继续下去的时候，是邱峰师兄告诉我如何开始一步一步的着手这个项目以及这个项目的原理。在他们的鼓励和指导下，我才调整好心态，潜下心来慢慢学习相关技术和理论知识。在中期的系统设计中，同样也是朱耀耀师兄给我讲解相关的Web知识，告诉我如何开发一个Web工程项目，并在我遇到问题的时候耐心的给我讲解，知道我学会为止。

最后，我还要感谢我的父母和大学辅导老师。在大学期间，是父母给我交学费，关心我平时的生活，挣钱供我上学。是辅导员叫我在社会中做人的基本素质，给我树立正确的人生价值观。没有父母的养育和老师的栽培就没有现在的我，最后在一次向他们表达我的感激之情。

# 参考文献

1. [] 搜狗百科，地理信息科学，<https://baike.sogou.com/v154914609.htm?fromTitle=GIS> [↑](#endnote-ref-0)
2. [] 云逸88，云计算平台，<https://wenku.baidu.com/view/a128663083c4bb4cf7ecd18f.html> [↑](#endnote-ref-1)
3. [] ArcGIS，ArcGIS平台，<https://www.arcgis.com/features/index.html> [↑](#endnote-ref-2)
4. [] WIKIPEDIA，MapInfo Corporation，<https://en.wikipedia.org/wiki/MapInfo_Corporation> [↑](#endnote-ref-3)
5. [] AUTODESK，Autodesk 3D,<https://www.autodesk.com/> [↑](#endnote-ref-4)
6. [] 二元空间，Implementing WebGIS on Hadoop-A case study of improving small file IO performance on HDFS，<https://wenku.baidu.com/view/96541b61ccbff121dc36833d.html> [↑](#endnote-ref-5)
7. [] GeoBeans，一体化网络地理信息系统，<http://www.geobeans.cn/> [↑](#endnote-ref-6)
8. [] Tom, White. Hadoop权威指南[M]. 北京:清华大学出版社, 2017年 [↑](#endnote-ref-7)
9. [] 文艾, 王磊. 高可用性的HDFS--Hadoop分布式文件系统深度实践[M]. 北京:清华大学出版社, 2015年 [↑](#endnote-ref-8)
10. [] Lars, George. HBase权威指南[M]. 北京:人民邮电出版社, 2012年 [↑](#endnote-ref-9)
11. [] LBD. Phoenix综述(史上最全Phoenix中文文档). <https://www.cnblogs.com/linbingdong/p/5832112.html> [↑](#endnote-ref-10)
12. [] Maneesh, Sahn. Java Server Pages编程实例[M]. 北京:机械工业出版社, 2002年 [↑](#endnote-ref-11)
13. [] 林信良. JSP&Servlet学习笔记[M]. 北京:清华大学出版社, 2012年 [↑](#endnote-ref-12)
14. [] Budi, Kurniawan Paul, Deck. 深入剖析Tomcat[M]. 北京:机械工业出版社, 2011年 [↑](#endnote-ref-13)
15. [] 邱晨健，栅格地图分布式存储方案设计与性能分析，西安电子科技大学，2014年 [↑](#endnote-ref-14)
16. [] 每春辉，基于Hadoop的多版本栅格地图存储于管理系统，西安电子科技大学，2014年 [↑](#endnote-ref-15)