World Scientific Research Journal (WSRJ) ISSN: 2472-3703
www.wsr-j.org

# Spatial Data Management System Based on MongoDB

Xuetao Liu, Jin Cheng, Rui Chen, Yan Zhang

Information Research Center of Military Science, Beijing, 100042, China.

---

*Abstract: Traditional database has been difficult to meet the needs of increasingly huge data storage and management. To solve this problem, this paper uses MongoDB as a database, and uses Python language platform to combine with GDAL spatial data conversion library to realize the storage of spatial data. Thus, the MongoDB database is operated, and the visualization tool MongoVUE is used to realize the management of spatial data. MongoDB is well-received for its advantages of easy scalability, feature richness, high performance and easy management. This article verifies the performance advantages of MongoDB.*
*Keywords: storage and management; Mongo DB database; GDAL.*

---

## 1. INTRODUCTION

In today's era, the sharing of information and data between different industries, different departments, and different applications has become more and more common. The amount of data and data types are increasing. Therefore, management and storage are challenging topics. The scalability and storage capacity of traditional databases have been difficult to meet the current diversified needs of big data storage. The MongoDB database uses an open source distributed approach that uses documents for storage, distributed storage of massive amounts of data, and the use of MapReduce for parallel accelerated computing. At the same time, in terms of query language, MongoDB also has a very powerful function. For a single table query of a relational database, most functions can be implemented. Therefore, you can use this method for building a cloud platform.

## 2. MONGODB TABLE STRUCTURE DESIGN

MongoDB is a database for document storage. When data is stored, various attribute data are saved in the form of documents. Spatial data has more complex attribute data, and it has certain difficulty in storage and presentation. Therefore, a table structure for storing spatial data needs to be specially designed to more effectively store the attribute information of spatial data and the relationship between them. For example, by comparing the organization and storage of spatial data in the Geo Database in Arc Catalog, combined with the characteristics of MongoDB data, the spatial data storage format is designed as shown in Table 1:

Table 1. Spatial Data Storage Format

| Number | Name | Type | Indexing |
|---|---|---|---|
| 1 | _id | ObjectId | Y |
| 2 | RefName | String | Y |
| 3 | Angle | Int | N |
| 4 | LyrOn | Int | N |
| 5 | EntColor | Int | N |
| 6 | LyrHandle | String | N |
| 7 | EntLineWt | Int | N |

The storage format in Table 1 is mainly used to store spatial data attribute data. The attribute information of the spatial data is mainly represented by a number, a field name, a field type, an index, and a comment. In addition, an index item is set therein to quickly access specific information in the database table. It can make the query statement corresponding to the table execute faster. For example, the _id field represents the encoding of an object in the collection, which is allocated by the database for the object. Setting an index for the _id field can more accurately and quickly query the desired object and improve management efficiency.

In order to display the data structure stored in the MongoDB database more clearly and intuitively, as shown in Figure 1:

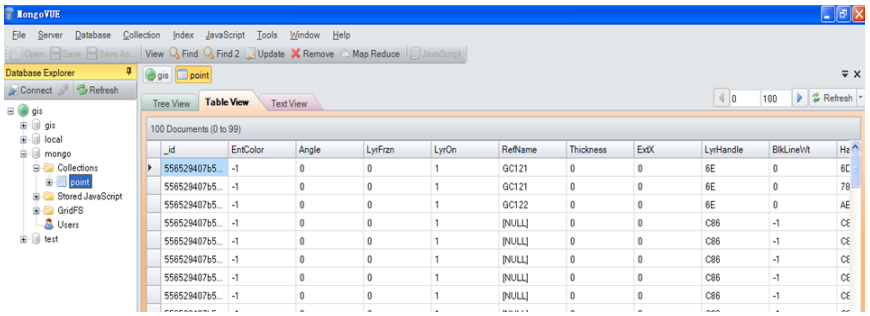


Figure 1. Data Storage Structure

## 3. RELATED SOFTWARE RELATIONSHIP DESIGN

The spatial data management system designed in this paper combines multiple development environments and software, including GDAL class libraries, Python language development environment, MongoDB, and MongoVUE. These major softwares need to be applied and integrated in an orderly manner to realize a spatial data management system. The design and use flow of each software is shown in Figure 2:

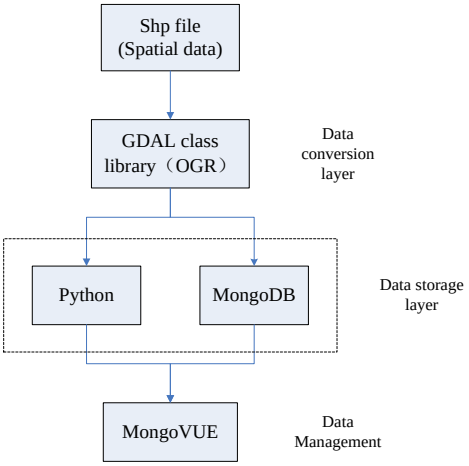


Figure 2. Software Relationship Design

The above diagram can clearly see the relationship between the software and the role it provides:

Shapefiles are used to obtain spatial data sources. Shapefiles is a vector data format provided by ESRI. There is no topology information. A group of files form a Shapefiles. There are three necessary basic files, namely: attribute file (.dbf), index Files (.shx) and coordinate files (.shp).

The data conversion layer mainly corresponds to the GDAL class library, in which the article mainly adopts a branch OGR library. OGR is an open source code library, which is mainly used to process and read GIS vector data, and OGR can process and read many popular vector data. This includes ESRI shapefiles. After obtaining the vector space data, it is converted in the data conversion layer through the Geometry data model in the OGR library. The main spatial analysis functions such as getDimension, getCoordinateDimension, transformTo, and getSpatialReference are mainly used.

The data storage layer uses the Python language and MongoDB to store the Shapefile spatial data files in the MongoDB database. In this layer, the Python language occupies an important part. First you need to import the GDAL, MongoDB extensions for Python, and then compile the converted files in the conversion layer in the environment, connect to the MongoDB database interface, and finally store the data in MongoDB. In. Implement the storage part of the spatial data.

The data management layer is mainly implemented through MongoVUE and Javascript scripts. MongoVUE can connect with MongoDB and can access spatial data stored in MongoDB. In addition, basic operations such as querying, inserting, and updating spatial data are performed through Javascript scripts.

## 4. SPATIAL DATA STORAGE IMPLEMENTATION

The MongoDB database is document-oriented. The data structure of MongoDB is very loose and close to json's bson format. It is stored as a key-value pair in a collection. At the same time, it is also possible to store relatively complex type data. Converting the data format in the shapefile file to the bson format is more complicated. Therefore, entering the spatial data

storage into the MongoDB database is an important part of the entire spatial data management system. Figure 3 shows the implementation of spatial data storage:

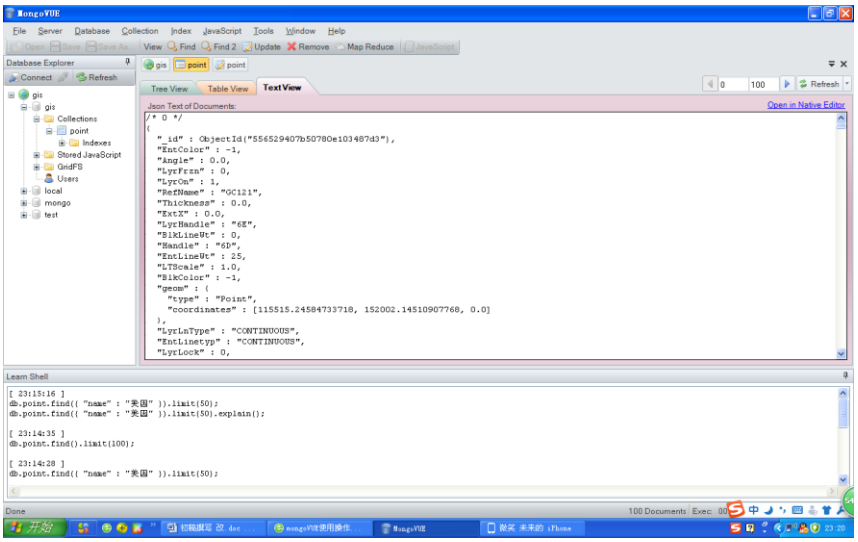

Figure 3. Data Storage Implementation

Spatial data storage is to convert the shapefile file format through GDAL, compile it in the Python environment, and finally save the spatial data into MongoDB. Import the Python environment extension module first, and then compile:

```
import os
import sys
import json
import pymongo
from pymongo.connection import Connection
from progressbar import ProgressBar
from osgeo import ogr
```

Figure 4. Import Module

The role of the above statement is to import the various modules that need to be used, the import needs to use the json format, and the pymongo, progressbar, and ogr modules are important modules for implementing the function, and need to be installed in the Python environment in advance. Connection is the connection between pymongo and the environment. This is the first step. You also need to create and define the required functions.

def shp2mongodb(shape_path, mongodb_server, mongodb_port, mongodb_db, mongodb_collection, append, query_filter):

"""Convert a shapefile to a mongodb collection"""

print 'Converting a shapefile to a mongodb collection'

driver = ogr.GetDriverByName('ESRI Shapefile')

print 'Opening the shapefile %s…' % shape_path

ds = driver.Open(shape_path, 0)

if ds is None:

print 'Can not open', ds

sys. exit (1)
lyr = ds. GetLayer ()
totfeats = lyr.GetFeatureCount()
lyr. SetAttributeFilter (query_ filter)
The above statement is to create and define the functions in mongodb database, including path, server, port number, database and data set. After the creation is complete, call GetDriverByName in the OGR library to obtain the shapefile and determine whether it is correct. Next, use GetLayer and other functions to obtain the data layer.
Print 'Starting to load %s of %s features in shapefile %s to MongoDB…' % (lyr.GetFeatureCount(), totfeats, lyr.GetName())
print 'Opening MongoDB connection to server %s:%i…' % (mongodb_server, mongodb_port)
connection = Connection(mongodb_server, mongodb_port)
print 'Getting database %s' % mongodb_db
db = connection[mongodb_db]
print 'Getting the collection %s' % mongodb_collection
collection = db[mongodb_collection]
if append == False:
print 'Removing features from the collection…'
collection.remove ({})
print 'Starting loading features…'
The above statement is mainly to connect to the MongoDB database to obtain the database and data set. If the data is not formatted correctly in the if statement, the data will be moved out of the data set.

```
pbar = ProgressBar(maxval=lyr.GetFeatureCount()).start()
k=0
# iterate the features and access its attributes (including geometry) to store them in MongoDb
feat = lyr.GetNextFeature()
while feat:
        mongofeat = {}
        geom = feat.GetGeometryRef()
        mongogeom = geom.ExportToJson()
        # store the geometry data with json format
        mongofeat['geom'] = json.loads(mongogeom)
        # iterate the feature's  fields to get its values and store them in MongoDb
        feat_defn = lyr.GetLayerDefn()
        for i in range(feat_defn.GetFieldCount()):
                value = feat.GetField(i)
                if isinstance(value, str):
                        value = unicode(value, "utf-8")
                field = feat.GetFieldDefnRef(i)
                fieldname = field.GetName()
                mongofeat[fieldname] = value
        # insert the feature in the collection
        collection.insert(mongofeat)
        feat.Destroy()
        feat = lyr.GetNextFeature()
        k = k + 1
        pbar.update(k)
pbar.finish()
print '%s features loaded in MongoDb from shapefile.' % lyr.GetFeatureCount()
```

Figure 5. Data Import to MongoDB

After the completion of the function creation, database connection and judgment is completed, the code in the figure above starts to import data into the database. The GetGeometryRef function obtains the corresponding geometric relation and information, and uses the json module to traverse the data and convert the data of Layer, Field, Name, etc. into the value in the json format.

## 5. MONGODB MANAGEMENT FUNCTION

MongoVUE is a MongoDB visual operation software that can manage large-scale data more easily. It can import\export data sets, monitor server performance, and visually manage various data.

Basic data management can be easily accomplished through the combination of MongoVUE and Javascript scripts. The following will introduce the simple management of the MongoDB database.

### 5.1 Query Data in the Database

If you want to query an object in a database collection, you need to use the find method. The object to be queried is represented in the form of json. code show as below:

```
> db.Account.find()
{ "_id" : ObjectId("4df08553188e444d001a763a"), "Angle" : 1, "RefName" : "libing", "Pass
word" : "1", "Age" : 26, "Email" : "libing@126.com", "RegisterDate" : "2011-06-09 16:31:25
" }
{ "_id" : ObjectId("4df08586188e444d001a763b"), "Angle" : 2, "RefName" : "lb", "Password
" : "1", "Age" : 25, "Email" : "libing@163.com", "RegisterDate" : "2011-06-09 16:36:95" }
> db.Account.findOne()
{"_id": ObjectId ("4ded95c3b7780a774a099b7c"),
"UserName": "libing",
"Password": "1",
"Email": "libing@126.cn",
"RegisterDate": "2011-06-07 11:06:25"}
```

In the code, db represents the database to be queried, Account is the name of the Collection dataset, and find is the query data. Among them, .find () is to query all records in the corresponding database. The above code indicates that there are only two records in this database. And .findOne() just queries a record. The result is shown in the figure below:

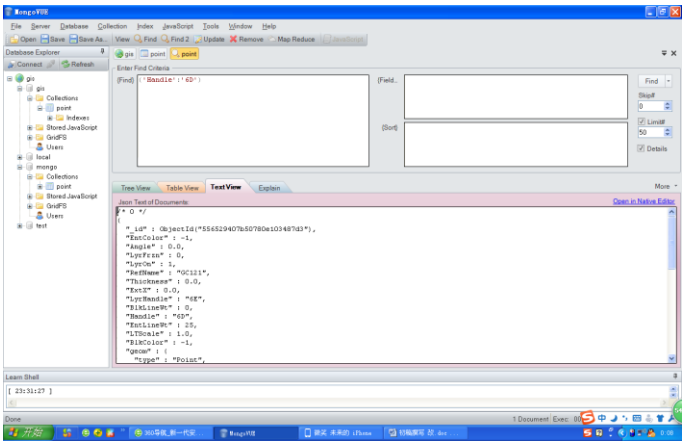

Figure 6. Query Data

### 5.2 Insert Data in the Database

If you want to insert an object in a collection in the database, you need to use the save or insert method to pass the inserted object as a parameter to the save/insert method in the form of json. The key code is as follows:

> db.Account.insert({AccountID:2,UserName:"lb",Password:"1",Age:25,Email:"libing@163.com",RegisterDate:"2017-06-09 16:36:95"})

According to the above code can be seen, automatically create a collection user, all objects in the user collection can be detected by db.user.find (); if you do not specify the value of _id insert, then the database will be added for the insertion object The _id field and its value. The results are shown below:

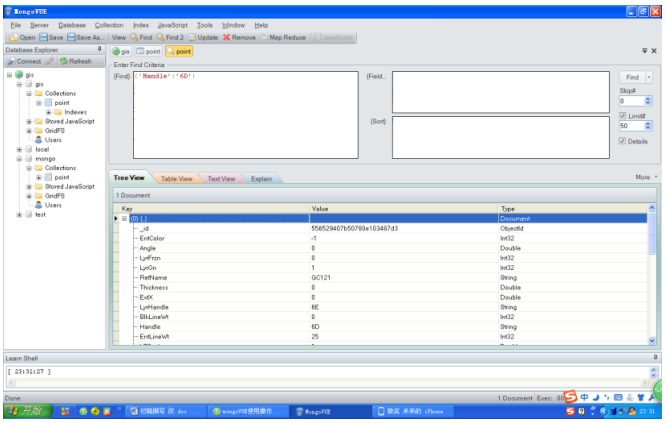

Figure 7. Insert Data

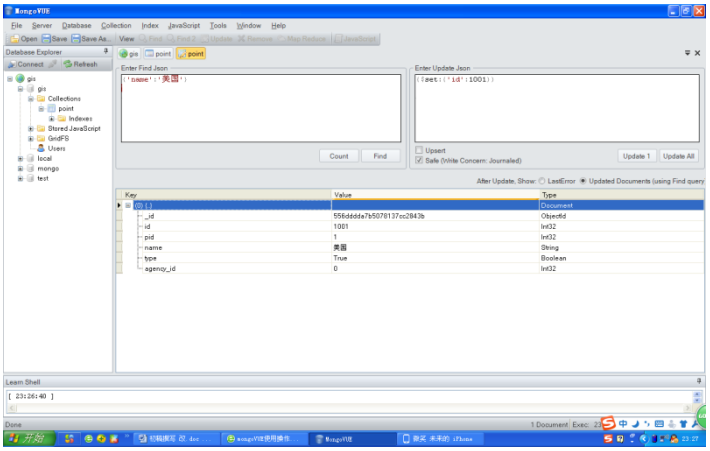

Figure 8. Updating the data

### 5.3 Update Data

Now to modify the newly inserted object, such as changing the value of age to 10, the code is as follows:

> db.Account.update({"AccountID":1},{"$set":{"Age":27,"Email":"libingql "}})

> db.Account.find({"AccountID":1})

{ "AccountID" : 1, "Age" : 27, "Email" : "libingql@163.com", "Password" : "1", "RegisterDate" : "2011-06-09 16:31:25", "UserName" : "libing", "_id" : ObjectId("4df0818e444d001a763a") }

For the database mongodb, you can use the update method to complete this operation. The query condition is represented by the first parameter. The field to be modified and the modified value are represented by the second parameter. The result is as follows:

## 6. SUMMARY

This paper builds and develops an experimental system, completes the storage of spatial data, allows the spatial data management and monitoring to be realized, and the performance advantages of MongoDB have also been verified. This paper describes the various data storage methods, data table design and server deployment of the experimental system, and lists the code blocks of the various types of extraction algorithms required by the system.

## REFERENCES

[1] Lei Delong, Guo Diansheng, Chen Chongcheng, Wu Jianwei, Wu Xiaozhu. The storage and processing system of vector space data cloud based on MongoDB [J]. Earth Information Science, 2014, 4(16):507-516.

[2] Zhang En, Zhang Guangdi, Lan Lei. Massive spatial data storage and parallelism based on MongoDB [J]. Geospatial Information, 2014, 1(12):46-48.

[3] Wang Guanglei. MongoDB database application research and program optimization [J]. Information Technology, 2016, 20: 93-96.

[4] Zhang Lu, Gan Quan, Liu Jianchuan. Research on Data Storage Model of Geographic Information Based on MongoDB [J]. Surveying and Mapping, 2014, 37(4): 147-151.