# 第5章 GridFS

**本章学习目标**

- 了解 GridFS 的概念和应用场景。
- 熟悉 GridFS 的存储结构。
- 掌握使用 Shell 操作 GridFS。
- 熟悉使用 Python 操作 GridFS。
- 了解使用 Java 操作 GridFS。

在前两章的学习中，了解到 MongoDB 使用 BSON 格式来保存二进制对象。然而，BSON 对象存在内存限制，最大不能超过 16 MB。当需要存储巨大的文件，例如视频时，可能会遇到存储瓶颈。为了解决这个问题，可以使用 GridFS 规范来将一个大文件分割成多个小文档，以保存大的文件对象，这将有效地解决以上问题。本章将详细介绍 GridFS 的相关知识。

## 5.1 认识 GridFS

### 5.1.1 GridFS 简介

GridFS 是一种将大型文件存储在 MongoDB 中的文件规范，此规范获得了所有官方驱动的支持，并且支持分布式应用。虽然 GridFS 不是 MongoDB 本身的特性，但是它可以通过开发语言驱动实现大文件在数据库中的处理。GridFS 将文件存储在单个文档中的做法已经被废弃，取而代之的是将那些超过 BSON 文件限制的对象，如图片、音频、视频等，分成多个块，然后将每个块存储为单独的文档。这种方式可以更有效地利用服务器的存储空间，并提高文件传输和查询的性能。此外，GridFS 也支持按需分割文件，将大文件拆分成多个小文件，以减少网络传输的数据量，从而提高文件传输效率。因此，读者无须了解 GridFS 规范中的细节，仅需学习在各个语言版本的驱动中有关 GridFS API 的部分或是如何使用 mongofiles 即可。

### 5.1.2 GridFS 的应用场景及优势

**1. 应用场景**

MongoDB 普通的集合不支持 16 MB 以上的文档，比直接存储在本地文件系统中更加适合的方法是将这种大文件存储在 GridFS 中。因此，MongoDB 文档的大小超过 16 MB 是使用 GridFS 的条件之一，例如用户上传大量的图片或者系统本身的文件发布等，都可以使用 GridFS。除此之外，GridFS 的其他应用场景如下。

(1) 文件的数量迅速增长,可能达到单机操作系统自身的文件系统的查询性能瓶颈,甚至超过单机硬盘的扩容范围。如果文件系统的目录对文件数量有限制,那么可以使用GridFS来存储更多数量的文件。

(2) 文件的索引、存储除文件本身以外还需要关联更多的元数据信息。如果想要访问大文件的部分信息,又不想将整个文件加载到内存中,可以使用GridFS来调用文件的某些部分,如调用文件的作者、发布时间、标签等信息,而无须将整个文件加载到内存。

(3) 文件的备份、文件系统访问的故障转移和修复。如果想要在多个系统和设施之间自动同步文件和元数据,可以使用GridFS。使用地理上分布的副本集(Geographically Distributed Replica Sets),MongoDB可以将文件和元数据自动地分发到多个mongo实例和设施上。

**2. GridFS对比传统文件系统的优势**

在一些解决方案中,使用MongoDB的GridFS存储大文件,比使用系统级别的文件系统更便利。GridFS对比传统文件系统的优势如下。

(1) 支持分布式。GridFS利用MongoDB的分布式存储机制,可以直接使用MongoDB Replication和Sharding机制,有效地保证了数据可靠性和水平扩展性。因为MongoDB将数据文件空间以2 GB为一块进行分配,所以GridFS不会产生磁盘碎片问题。基于MongoDB来存储文件数据和文件元数据,将数据库与文件系统完美地结合在一起,兼具文档数据库和文件系统的优势。

(2) 支持MapReduce。GridFS能够进行复杂管理和查询分析。

(3) 支持索引和缓存。元数据存储在MongoDB中,可以对文件和文件元数据添加索引操作,以此提高系统效率。

(4) 支持Checksum。GridFS可以为文件产生散列值,以此校验文件来检查文件的完整性。

(5) 对开发者友好。在MongoDB基础上,GridFS无须使用独立文件存储架构,这样一来代码和数据能够实现真正的分离,方便开发者管理。由此可知,在项目开发中,使用GridFS可以简化需求,减少开发成本,在云计算平台的效果尤其显著。

除了以上优势外,GridFS还可以避免用于存储用户上传内容的文件系统出现的某些问题,如数据一致性问题。

### 5.1.3 GridFS的存储结构

GridFS通过两个集合来存储文件,分别为存储文件块的集合(chunks)和存储文件元数据的集合(files)。当执行查询文件操作时,驱动程序能够根据需要重新组装文件块。GridFS为此指定了一个将文件分块的标准,来实现对GridFS存储的文件进行范围查询或者从文件的某个部分开始查询,如访问视频或音频文件的某个节点。该标准是指每个大文件都会在文件集合对象中保存一个元数据对象,一个或多个块对象则被组合保存在一个块集合中。

GridFS会将集合chunks和集合files放在一个桶(Bucket)中,并且这两个集合使用Bucket的名字作为前缀。Bucket在MongoDB中是一个概念,GridFS默认使用fs命名的Bucket存放上述两个文件集合,对应的两个集合名称是fs.chunks和fs.files,例如给桶自

定义为 baby，对应的集合则为 baby.chunks 和 baby.files。需要注意的是，不但可以定义不同的 Bucket 名称，还可以在一个数据库中定义多个 Bucket，但所有集合的名称都不得超过 MongoDB 命名空间的限制。

Bucket 是建立在数据库 Database 上的，在操作 GridFS 时，可以直接操作 Bucket。

当使用 GridFS 存储文件时，如果文件大于 chunksize（每个 chunk 大小为 256 KB）的值，GridFS 会先将文件分割成多个 chunk，最终将 chunk 的信息存储在集合 fs.chunks 的多个文档中。然后将文件信息存储在集合 fs.files 的唯一一份文档中。对于同一个大文件，集合 fs.chunks 中多个文档中的字段 file_id 对应集合 fs.files 中某一个文档的字段_id。

在读取文件时，首先根据查询条件在集合 files 中找到对应文档的字段_id；然后根据字段_id 的值在集合 chunks 中查询所有与字段 files_id 相同值的文档；最后根据字段 n 的顺序读取集合 chunks 的字段 data 数据，还原文件。GridFS 的存储过程如图 5-1 所示。

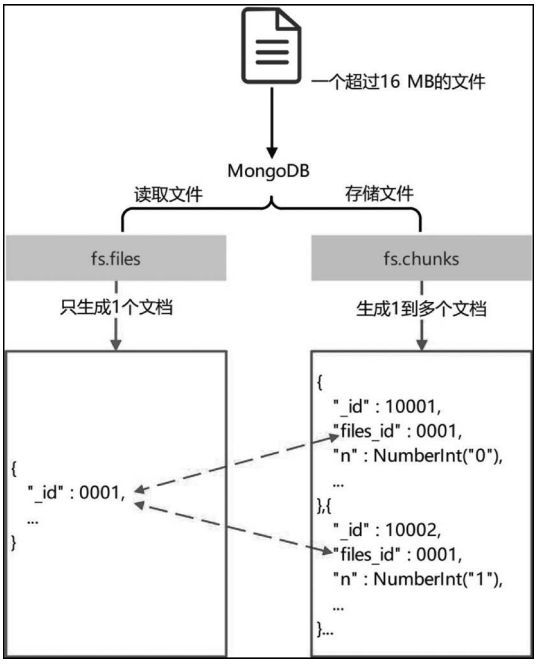


图 5-1　GridFS 的存储过程

集合 fs.files 以类 JSON 格式的文档形式存储文件的元数据，每在 GridFS 中存储一个文件，就会在集合 fs.files 中对应生成一个文档。集合 fs.files 中文档的存储内容一般如下。

```
{
    "_id": <ObjectId>,              //文档 ID,唯一标识
    "chunkSize": <num>,             //chunk 大小,256KB
    "uploadDate": <timetamp>,       //文件上传时间
    "length": <num>,                //文件长度
    "md5": <string>,                //文件 MD5 值
    "filename": <string>,           //文件名
    "contentType": <string>,        //文件的类型
    "metadata": <dataObject>        //文件自定义信息
}
```

上述代码中，<ObjectId>是一种特殊的 BSON 类型，它保证集合的唯一性。ObjectId

值的长度为 12 字节，包括 4 字节时间戳、5 字节随机数、3 字节增长量。uploadDate 指文件第一次被 GridFS 存储的日期，该值是日期类型。md5 指 MD5 算法，该算法已经被 MongoDB 驱动程序弃用，并将在未来版本中删除 MD5 算法，需要文件摘要的应用程序应该在 GridFS 之外实现它，并存储在 files. metadata 中。filename 指可读的 GridFS 文件名称。metadata 是可选的，元数据字段可以是任何数据类型，可以保存用户想要存储的任何附加信息。

集合 fs. chunks 以类 JSON 格式的文档形式存储文件的二进制数据。每在 GridFS 中存储一个文件，GridFS 就会将文件内容按照 256 KB 分成多个文件块，然后将文件块按照类 JSON 格式存放在集合 fs. chunks 中，每个文件块对应集合 fs. chunks 中的一个文档。一个存储文件会对应一到多个 chunk 文档。集合 fs. chunks 中文档的存储内容一般如下。

```
{
   "_id": <ObjectId>,          //数据块的标识,文档 ID,唯一标识
   "files_id": <ObjectId>,     //对应 fs.files 文档的 ID
   "n": <num>,                 //块的序列号,标识文件的第几个 chunk
   "data": <binary>            //数据块中装载的数据,二进制类型
}
```

上述代码中，键 n 指块的序列号，GridFS 会为所有块编号，编号从 0 开始。

为了提高检索速度，MongoDB 为 GridFS 的两个集合建立了索引：fs. files 集合使用的是 filename 与 uploadDate 字段作为唯一、复合索引；fs. chunk 集合使用的是 files_id 与 n 字段作为唯一、复合索引。

## 5.2 在不同环境下操作 GridFS

### 5.2.1 使用 Shell 操作 GridFS

MongoDB 提供了 mongofiles 工具来使用 GridFS，通过命令行实现上传、获取、查找、下载和删除 GridFS 对象中存储的文件。mongofiles 工具提供了存储在文件系统和 GridFS 中的对象之间的接口。mongofiles 命令行的语法格式如下。

```
mongofiles <options> <connection-string> <command> <filename or _id>
```

上述语句中，<options>主要配置 mongofiles 的一些读写优先级；<connection-string>是连接 mongod/mongos 的配合信息，如 host、port、安全认证等相关配置；<command>是 mongofiles 具体的文件操作，如上传(put)、下载(get)、查询(list、search)等；<filename or _id>指存储文件的名称。

使用 GridFS 时需要注意，当设置安全访问控制时，mongofiles 连接的用户需要具备 read 权限(list、earch、get 等命令)和 readWrite 权限(put、delete 等命令)。

接下来详细介绍 mongofiles 工具中<command>5 个具体的文件操作。

**1. 上传文件(put)**

put 操作用于将指定的文件从本地文件系统写入 GridFS 中，多个文件可以使用空格进行分隔。put 操作的基本语法格式如下。

```
mongofiles -d <数据库名称> -l <文件所在路径> put <存储文件名称> --prefix=<桶的名称>
```

上述语句中，d 表示数据库名称；l 表示原始文件的所在路径；put 表示存储文件名称；prefix 用于指定文件存储的桶，默认为 fs，若指定的桶不存在，则新建一个 GridFS。

除此之外，读者可通过 mongofiles --help 命令查看 mongofiles 命令的具体使用方法，常用的参数和选项如下。

（1）参数 r：表示当 GridFS 存储在桶中，文件名称相同时，会删除原有文件，存入新文件。

（2）参数 port：用于指定主机端口，用法如“--port 27017”。

（3）参数 uri：用于指定 MongoDB 的连接信息，可远程连接，用法如“--uri＝mongodb://127.0.0.1:27017”。

（4）参数 c：用于指定集合名，默认是 fs。

（5）参数 u 和参数 p：分别用于指定用户名和密码。

（6）参数 prefix：用于指定前缀，不指定则默认为 fs，用法如“--prefix＝myfs”。

接下来将通过示例演示使用 mongofiles 工具实现上传大文件，如例 5-1 所示。

**【例 5-1】** 将本地文件系统/data/下的锋云智慧使用手册（文件名：fengyunzhihui_shouce.pdf）上传到 MongoDB 数据库 mytest 中，并将文件存储在 myfs 桶中，具体如下。

```
[root@qf ~] # mongofiles -d mytest -l /data/fengyunzhihui_shouce.pdf put fengyunzhihui_
shouce.pdf --prefix=myfs
2022-12-01T15:41:23.199+0800  connected to: mongodb://localhost/
2022-12-01T15:41:23.199+0800  adding gridFile: fengyunzhihui_shouce.pdf
2022-12-01T15:41:23.434+0800  added gridFile: fengyunzhihui_shouce.pdf
```

由上述输出结果可知，已经成功将 PDF 文件上传至 MongoDB 数据库中。为了进一步验证，可以连接 MongoDB 客户端查看数据库 mytest 中的数据信息，具体如下。

```
#连接 MongoDB 数据库
[root@qf ~] # mongosh
#查看数据库
test> db
```

输出结果如下。

```
test
#查看所有数据库
test> show dbs
```

输出结果如下。

```
admin   132.00 KiB
config   84.00 KiB
local    72.00 KiB
mytest   18.94 MiB
#切换数据库 mytest
test> use mytest
```

输出结果如下。

```
switched to db mytest
#查看数据库 mytest 中的所有集合
mytest> show collections
```

输出结果如下。

```
myfs.chunks
myfs.files
#查看集合 files
mytest > db.myfs.files.find()
```

输出结果如下。

```
[
  {
    _id: ObjectId("63885aa3ffe803cfe1aff235"),
    length: Long("21134285"),
    chunkSize: 261120,
    uploadDate: ISODate("2022 - 12 - 01T07:41:23.433Z"),
    filename: 'fengyunzhihui_shouce.pdf',
    metadata: {}
  }
]
```

由上述输出结果可知，集合 myfs.files 以文档形式存储了 PDF 文件的元数据。

**2. 获取文件列表(list)**

list 操作用于列出 GridFS 存储的文件。list 操作的基本语法格式如下。

```
mongofiles -d <数据库名称> --prefix = <桶> list <存储文件名称>
```

接下来将通过示例演示如何通过 mongofiles 工具获取文件列表，如例 5-2 所示。

**【例 5-2】** 查找数据库 mytest 下 GridFS 中的文件，并且该文件的存储桶为 myfs。

```
[root@qf ~] # mongofiles -d mytest list --prefix = myfs
2022 - 12 - 01T17:32:22.275 + 0800  connected to: mongodb://localhost/
```

输出结果如下。

```
fengyunzhihui_shouce.pdf  21134285
```

由上述输出结果可知，数据库 mytest 下的 GridFS 桶中只有一个文件 fengyunzhihui_shouce.pdf。为了进行验证，读者可使用 MongoDB 可视化管理工具 Studio 3T 查看数据库 mytest 中的 GridFS Buckets，如图 5-2 所示。

**3. 查看文件(search)**

search 操作用于根据文件名搜索文件。search 操作的基本语法格式如下。

```
mongofiles -d <数据库名称> --prefix = <桶> search <存储文件名称>
```

接下来将通过示例演示如何通过 mongofiles 工具查找文件，如例 5-3 所示。

**【例 5-3】** 查找数据库 mytest 下 GridFS 中的文件 fengyunzhihui_shouce.pdf，并且该文件的存储桶为 myfs。

```
[root@qf ~] # mongofiles -d mytest --prefix = myfs search "fengyunzhihui_shouce.pdf"
```

输出结果如下。

```
2022 - 12 - 01T17:44:01.492 + 0800  connected to: mongodb://localhost/
fengyunzhihui_shouce.pdf  21134285
```

由上述输出结果可知，成功查询到文件 fengyunzhihui_shouce.pdf。

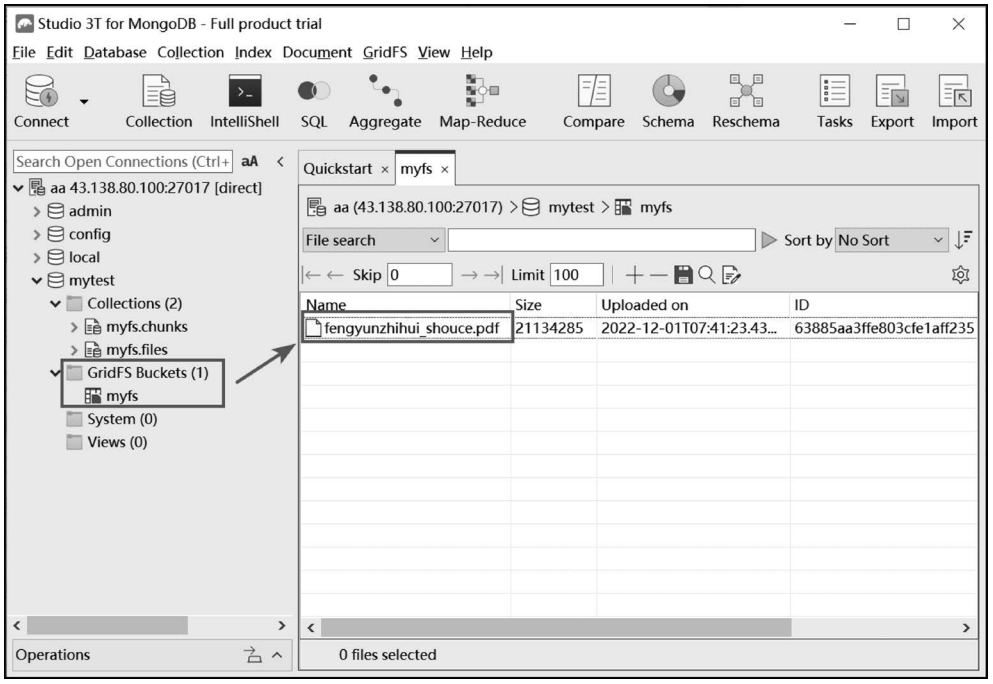


图 5-2　查看数据库 mytest 中的 GridFS Buckets

**4. 下载文件(get)**

get 操作用于将指定的文件从 GridFS 存储下载到本地文件系统,也是获取文件的方式之一。get 操作的基本语法格式如下。

```
mongofiles -d <数据库名称> --prefix=<桶> get <存储文件名称> --local=<下载到本地的文件名>
```

上述语句中,省略 prefix 选项时,则默认选择桶 fs; local 选项可修改文件名。

接下来将通过示例演示如何通过 mongofiles 工具实现下载大文件,如例 5-4 所示。

**【例 5-4】** 将 MongoDB 数据库 mytest 下的锋云智慧使用手册(文件名: fengyunzhihui_shouce.pdf)下载到本地文件系统/tmp/中,并命名为 fengyun.pdf。

```
[root@qf ~] # mongofiles -d mytest --prefix=myfs get "fengyunzhihui_shouce.pdf" --local=/tmp/fengyun.pdf
```

输出结果如下。

```
2022-12-01T15:55:07.004+0800  connected to: mongodb://localhost/
2022-12-01T15:55:07.052+0800  finished writing to /tmp/fengyun.pdf
```

由上述输出结果可知,已成功将文件下载到本地目录/tmp/fengyun.pdf 中。为了进一步验证,查看本地文件系统/tmp/下是否存在 fengyun.pdf 文件,具体如下。

```
[root@qf ~] # ls /tmp/
fengyun.pdf
……省略其他文件……
```

**5. 删除文件(delete)**

delete 操作用于将指定的文件从 GridFS 中删除。delete 操作的基本语法格式如下。

```
mongofiles -d <数据库名称> --prefix=<桶> delete <存储文件名称>
```

接下来将通过示例演示如何通过 mongofiles 工具实现删除文件，如例 5-5 所示。

**【例 5-5】** 将 MongoDB 数据库 mytest 下的锋云智慧使用手册(文件名：fengyunzhihui_shouce.pdf)删除，具体如下。

```
[root@qf ~] # mongofiles -d mytest --prefix=myfs delete "fengyunzhihui_shouce.pdf"
```

输出结果如下。

```
2022-12-01T17:48:57.779+0800  connected to: mongodb://localhost/
2022-12-01T17:48:57.794+0800  successfully deleted all instances of 'fengyunzhihui_
shouce.pdf' from GridFS
#查找 fengyunzhihui_shouce.pdf 文件
[root@qf ~] # mongofiles -d mytest --prefix=myfs search "fengyunzhihui_shouce.pdf"
```

输出结果如下。

```
2022-12-01T17:49:00.957+0800  connected to: mongodb://localhost/
```

由上述输出结果可知，已成功将 fengyunzhihui_shouce.pdf 文件删除。

## 5.2.2 使用 Python 操作 GridFS

在第 4 章中讲解了使用 Python 操作 MongoDB，学习了数据的增、删、改、查。本节继续学习使用 Python 编程语言操作 GridFS，实现文件的上传、下载等操作。

**1. 上传文件**

首先，通过示例讲解如何上传文件，如例 5-6 所示。

**【例 5-6】** 将本地的新闻报道文件(文件名：NewsReport.pdf)上传至 GridFS。

```
1   import pymongo
2   from gridfs import GridFS
3   from GridFS_test import MongoGridFS
4   class MongoGridFS(object):
5       UploadCache = "uploadcache"
6       dbURL = "mongodb://43.138.80.100:27017"
7       def upLoadFile(self, file_coll, file_name, data_link):
8           client = pymongo.MongoClient(self.dbURL)
9           db = client["store"]
10          filter_condition = {"filename": file_name, "url": data_link, 'version': 2}
11          gridfs_col = GridFS(db, collection=file_coll)
12          file_ = "0"
13          query = {"filename": ""}
14          query["filename"] = file_name
15          if gridfs_col.exists(query):
16              print('已经存在该文件')
17          else:
18              with open(file_name, 'rb') as file_r:
19                  file_data = file_r.read()
20                  file_ = gridfs_col.put(data=file_data, **filter_condition)  # 上传到 gridfs
21                  print(file_)
22          return file_
23      if __name__ == '__main__':
24          a = MongoGridFS("")
25          file = a.upLoadFile("py", "NewsReport.pdf", "")      # 上传 NewsReport.pdf 文件
26          print("file")
```

上述代码中：第 6 行用于连接 MongoDB 所在的主机和端口；第 7 行中的 file_coll 表示集合名，file_name 表示文件名，data_link 表示文件链接；第 9 行中的 store 表示要操作的数据库；第 15～20 行使用 if 判断语句，若文件不存在，则上传该文件，否则返回“已经存在该文件”提示语；第 23～26 行，调用 a. upLoadFile()方法上传 NewsReport. pdf 文件，并返回该文件的 ID。

运行上述代码，实现上传文件返回文件 ID，查看控制台的输出结果，如图 5-3 所示。


图 5-3　上传文件返回文件 ID

为了验证文件是否上传成功，可以通过 MongoDB 可视化管理工具 Studio 3T 查看数据库 mytest 中的 GridFS Buckets，如图 5-4 所示。

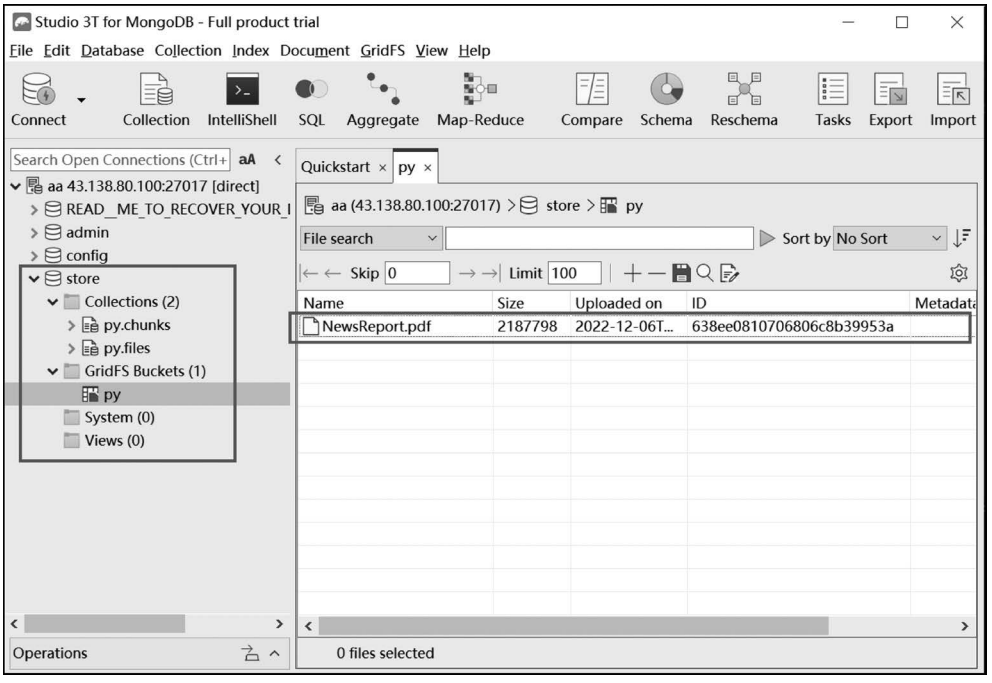


图 5-4　查看数据库 mytest 中的 GridFS Buckets

**2. 下载文件**

下面通过示例讲解如何下载文件，如例 5-7 所示。

**【例 5-7】** 将 GridFS 的新闻报道文件（文件名：NewsReport. pdf）下载至本地，并命名为 News1. pdf。

```
1   import pymongo
2   from gridfs import GridFS
3   class MongoGridFS(object):
4       UploadCache = "uploadcache"
5       dbURL = "mongodb://43.138.80.100:27017"
6       # 按文件名获取文档
7       def downLoadFile(self, file_coll, file_name, out_name, ver = -1):
8           client = pymongo.MongoClient(self.dbURL)
9           db = client["store"]
10          gridfs_col = GridFS(db, collection = file_coll)
11          file_data = gridfs_col.get_version(filename = file_name, version = ver).read()
12          with open(out_name, 'wb') as file_w:
13              file_w.write(file_data)
14  if __name__ == '__main__':
15      a = MongoGridFS()
16      a.downLoadFile("py", "NewsReport.pdf", "News1.pdf")   # 按文件名 NewsReport.pdf 下载
                                                              # 保存到 News1.pdf
```

上述代码中：第 7 行定义 downLoadFile()方法，其中 file_coll 表示集合名，file_name 表示文件名，out_name 表示下载后的文件名，ver 表示版本号，默认－1 表示最近一次的记录；第 16 行 a 类的实例化对象调用 downLoadFile()方法，将 py 桶里的 NewsReport.pdf 文件下载到本地，并重命名为 News1.pdf。查看当前 Python 项目包文件，如图 5-5 所示。

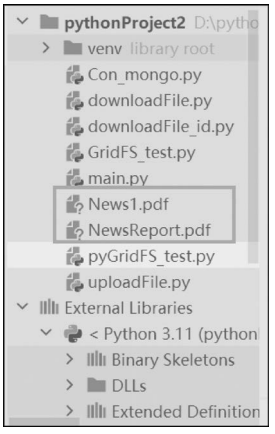


图 5-5　查看当前 Python 项目包文件

例 5-7 成功实现指定文件名下载文件，除此之外，还可以指定文件 ID 下载文件，如例 5-8 所示。

**【例 5-8】** 将 GridFS 的新闻报道文件(文件名：NewsReport.pdf)通过指定文件 ID 下载至本地，并命名为 News2.pdf。

```
1   import pymongo
2   from gridfs import GridFS
3   from bson.objectid import ObjectId
4   class MongoGridFS(object):
5       UploadCache = "uploadcache"
6       dbURL = "mongodb://43.138.80.100:27017"
7       # 按文件_id 获取文档
8       def downLoadFilebyID(self, file_coll, _id, out_name):
9           client = pymongo.MongoClient(self.dbURL)
10          db = client["store"]
11          gridfs_col = GridFS(db, collection = file_coll)
12          O_Id = ObjectId(_id)
13          gf = gridfs_col.get(file_id = O_Id)
14          file_data = gf.read()
15          with open(out_name, 'wb') as file_w:
16              file_w.write(file_data)
17          return gf.filename
18  if __name__ == '__main__':
19      a = MongoGridFS()
20      name = a.downLoadFilebyID("py", '638ee0810706806c8b39953a', "News2.pdf")
                                                          # 按 files_id 下载文件保存到 b.pdf
21      print(name)
```

上述代码中：第 8 行定义 downLoadFilebyID()方法，其中 file_coll 表示集合名，_id 表示文件 ID，out_name 表示下载后的文件名；第 20 行 a 类的实例化对象调用 downLoadFilebyID()方法，将 py 桶里的 NewsReport.pdf 文件下载到本地，并重命名为 News2.pdf；最后第 21 行输出文件名。

运行上述代码，指定文件 ID 返回文件名，查看控制台的输出结果，如图 5-6 所示。


图 5-6　指定文件 ID 返回文件名

查看当前 Python 项目包文件，如图 5-7 所示。


图 5-7　查看已下载的文件

如图 5-7 所示，已经成功下载并命名文件为 News2.pdf。

### 5.2.3　使用 Studio 3T 操作 GridFS

本节将讲解如何使用 MongoDB 可视化管理工具 Studio 3T 操作 GridFS。

**1. 上传文件**

首先打开 Studio 3T 软件，创建一个数据库(qianfengDB)，如图 5-8 所示。

如图 5-8 所示，数据库 qianfengDB 是空的。选中并右击 qianfengDB→Add GridFS Bucket 选项，弹出填写 Bucket Name 的页面，如图 5-9 所示。

自定义桶的名称为 qf，单击 OK 按钮，则成功创建以 qf 为前缀的 chunks 和 files 集合，如图 5-10 所示。

如图 5-10 所示，以 qf 命名的 GridFS Buckets 中没有任何文件。右击 Name 字段下的空白处，在显示的选项中选择 Add File 选项即可添加文件，如图 5-11 所示。

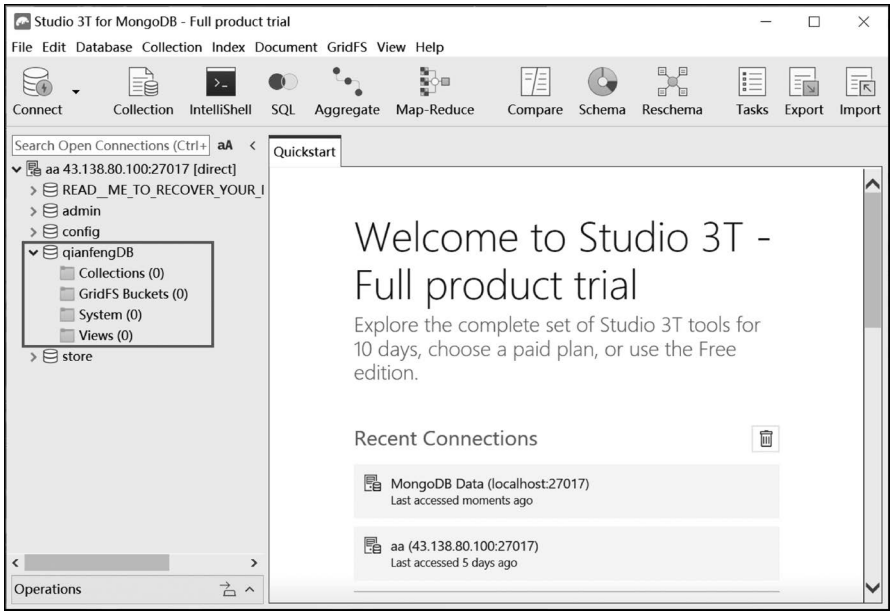


图 5-8　创建数据库 qianfengDB

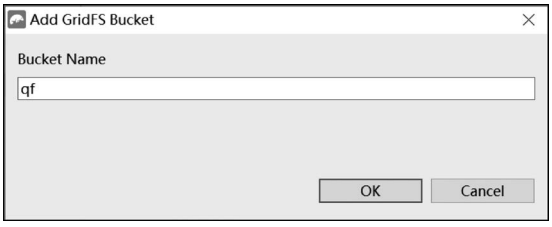


图 5-9　填写 Bucket Name

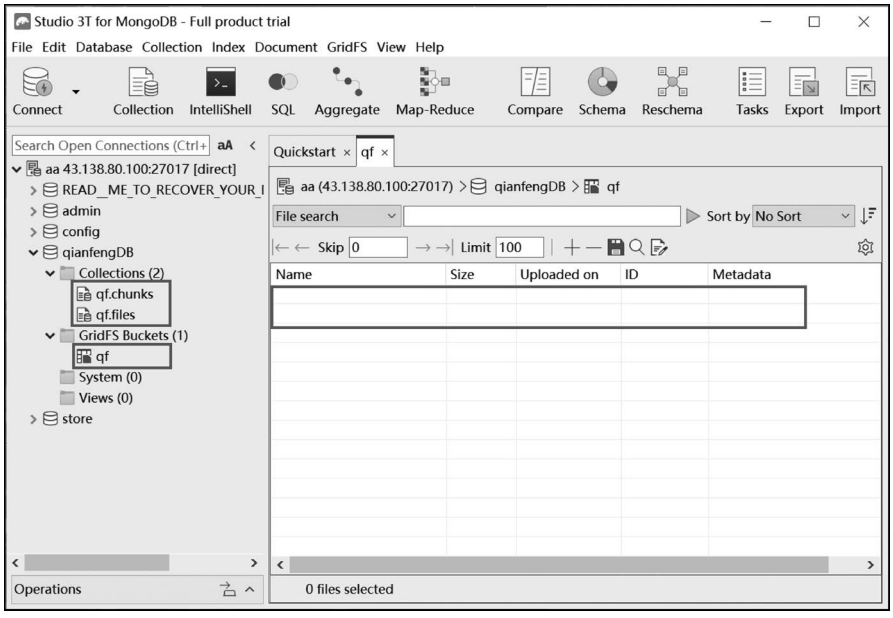


图 5-10　查看 qf 中的文件

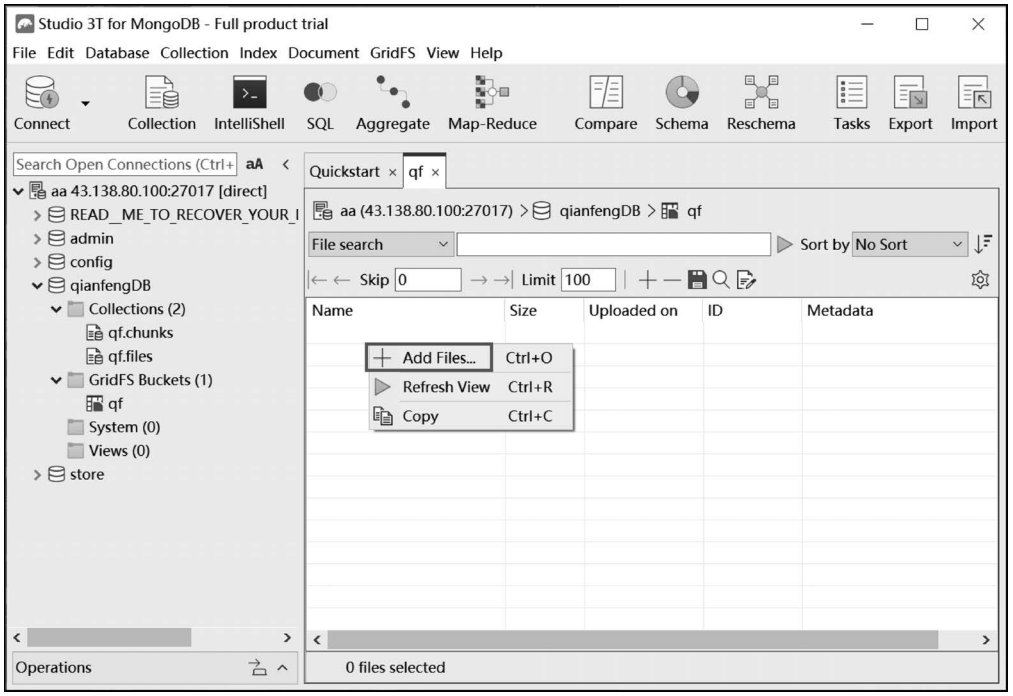


图 5-11　添加文件

单击 Add File 选项，弹出本地文件系统，选择要上传的文件，如图 5-12 所示。

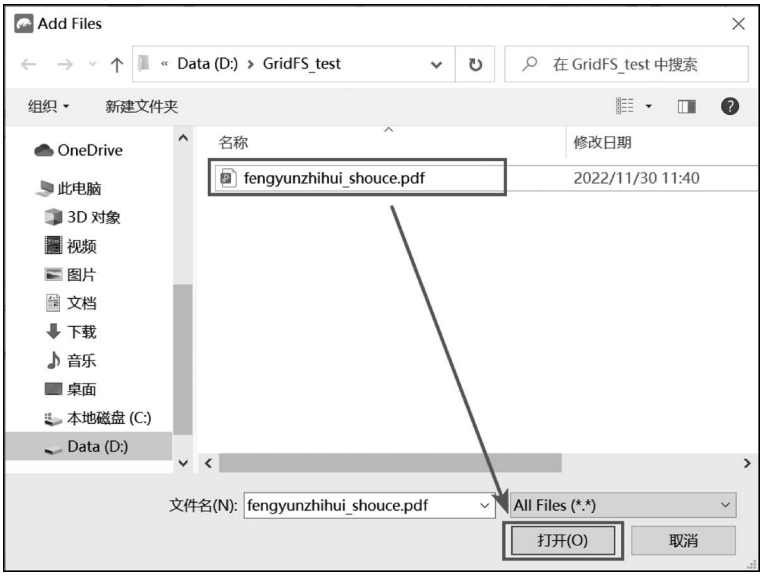


图 5-12　选择要上传的文件

单击“打开”按钮，即可上传该文件。文件上传成功的效果如图 5-13 所示。

**2. 下载文件**

为了不影响实验结果，先将已经上传的 fengyunzhihui_shouce. pdf 修改为 fengyun. pdf。给文件重命名，如图 5-14 所示。

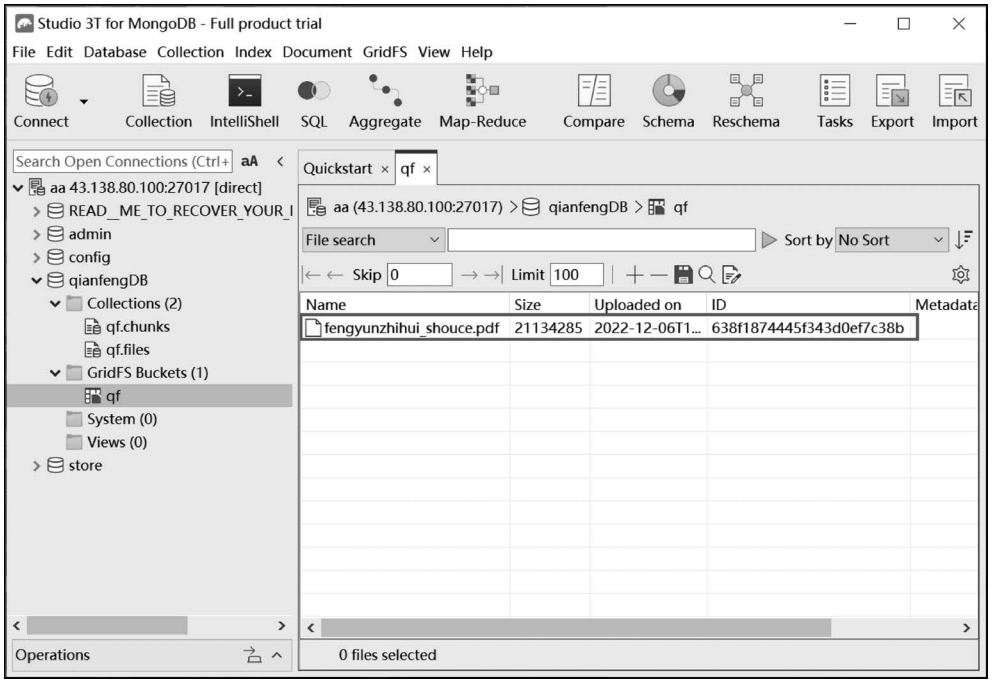


图 5-13　文件上传成功

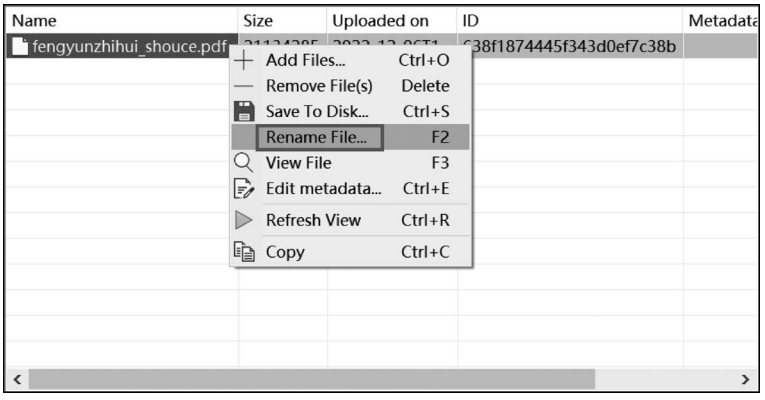


图 5-14　给文件重命名

选中并右击 fengyunzhihui_shouce.pdf 文件，单击 Rename File 选项，在编辑栏里输入 fengyun.pdf，即可修改成功。

选中并右击 fengyunzhihui_shouce.pdf 文件，单击 Save To Disk 选项，在弹出的本地文件系统中选择目标目录，如图 5-15 所示。

单击“选择文件夹”按钮即可完成下载。查看下载的文件，如图 5-16 所示。

**3. 删除文件**

选中并右击要删除的目标文件，如图 5-17 所示。

单击 Remove File(s)选项，将会弹出确认删除的窗口，单击 Remove files 按钮即可删除该文件，如图 5-18 所示。


图 5-15　选择目标目录


图 5-16　查看下载的文件

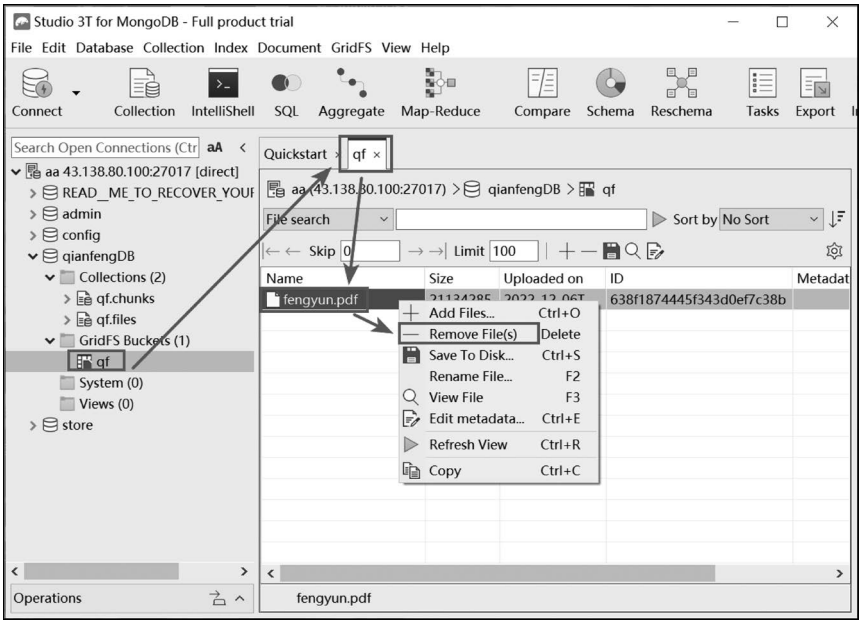


图 5-17　删除目标文件

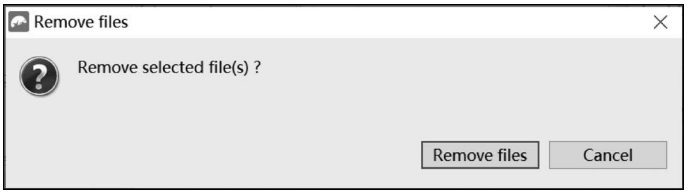


图 5-18　确认删除

## 5.3　本 章 小 结

本章首先介绍了 GridFS 的概念、应用场景，重点介绍了 GridFS 的存储结构；然后讲解了如何在不同开发环境下操作 GridFS，其中详细讲解了如何使用 Shell 操作 GridFS；最后讲解了使用 Python 和可视化管理工具操作 GridFS。“工欲善其事，必先利其器”，希望读者在自己的专业领域深刻学习好开发语言，进一步掌握 GridFS 的相关操作。

## 5.4　习　　题

**1. 填空题**

(1) GridFS 是一种将大型文件存储在 MongoDB 中的_____，获得了所有官方驱动的支持，并且支持_____。

(2) GridFS 不是 MongoDB 本身的特性，也无法通过 MongoDB 获取实现它的代码，而是通过_____实现大文件在数据库中的_____。

(3) GridFS 对比传统文件系统的优势有_____、_____、_____、_____、_____。

(4) GridFS 通过两个集合来存储文件，分别为存储文件块的_____和存储文件元数据的_____。

(5) Bucket(桶)在 MongoDB 中是一个概念，GridFS 默认使用 fs 命名的 Bucket 存放两个文件集合，对应的两个集合名称则是_____和_____。

**2. 简答题**

(1) 简述 GridFS 的几种应用场景。

(2) 简述 GridFS 的存储过程。

**3. 操作题**

请在 Linux 平台下完成 MongoDB 数据库的安装，并准备两个大于 16 MB 的文件。

① 使用 mongofiles 将文件上传至 GridFS。

② 使用 mongofiles 查看所有文件。

③ 使用 mongofiles 下载文件。

④ 使用 mongofiles 删除文件。