摘 要

随着因特网应用和计算机技术的飞速发展，数据库逐渐成为信息系统的核心部分并广泛应用于企业、金融机构、政府及国防等各个领域。但是传统的关系数据库在面对日益增多的大量数据时暴露了很多问题，不能对其高效、实时的处理。对系统提出的新需求，传统的关系数据库并不能很好的解决。非关系型数据库是为弥补关系数据库的不足而产生的，OrientDB数据库就是非关系型数据库的一种。目前现有的非关系数据库对大数据量数据的处理能力很强但是在在很多关键领域，关系型数据库又是非关系型数据库无法代替的。如何结合关系型数据库和非关系型数据库的优点成为当前数据库领域的关键问题。

通过对分布式理论、关系数据库理论及非关系型数据库技术的学习和研究，本文基于Java语言实现了一个分布式的关系型数据库JSQL，JSQL是一个将关系数据库技术，非关系型数据库技术和分布式技术相结合的产物。JSQL采用常用的客服端-服务器架构，分为客服端和服务器端，服务器端又分为分布式管理节点和分布式数据库节点。分布式管理节点实现了数据库服务器的负载均衡和监控管理功能。分布式数据库节点主要包含五大模块，网络模块接收前端客服端的连接请求，对客服端进行认证和授权，并对连接进行管理，并实现基于Mysql的通信协议； Sql的解析和执行模块接收客户端的SQL请求，然后解析和执行数据库存储的调用，返回执行结果；审计模块是存储和分析所有对数据库的更改情况，向分布式管理节点提供审计数据；数据库引擎模块利用了非关系型Orientdb数据库引擎，实现可靠的数据存储；分布式模块利用hazlcast实现了数据库集群，本文提出了分布式多版本并发控制方法，用来解决分布式数据一致性问题。基于以上功能模块，ＪＳＱＬ具有高可用性，可扩展性，负载均衡等特性，同时从数据库底层考虑了数据库安全审计需求，加入了数据审计图形化界面显示审计结果。

论文对系统进行了功能和性能测试。功能测试结果表明，系统在功能上符合分布式数据库的基本要求，审计系统的功能也达到本论文的要求。论文通过对性能测试结果进行分析，认为系统的性能基本达到本论文的要求。但是本系统对复杂SQL语句的支持还不是很完善，最后提出了改进的方案。

关键词：分布式数据库，关系数据库，安全审计，Mysql，非关系型数据库

ABSTRACT

With the rapid development of Internet application and computer technology, the database has gradually become the core of information system Points and widely used in enterprises, financial institutions, government and national defense and other fields. However, the traditional relational database is facing increasing When exposed to a large number of data a lot of problems, can not be efficient and real-time processing. The new needs of the system Seeking, the traditional relational database can not be solved very well. Non-relational database is to make up for the lack of relational database, the OrientDB database Is a non-relational database. At present, the existing non-relational database has great ability to handle large amounts of data But in many key areas, relational databases are non-relational databases that can not be replaced. How to combine the advantages of relational databases and non-relational databases has become the key issue in the current database area.

Through the study of distributed theory, relational database theory and non-relational database technology, This article based on the Java language to achieve a distributed relational database JSQL, JSQL is a relational database technology, The combination of non-relational database technology and distributed technology. JSQL common client-server - server architecture, divided into customer service and server-side, Server-side is divided into distributed management nodes and distributed database nodes. Distributed management nodes to achieve a database server load balancing and monitoring and management functions. Distributed database node contains five major modules, The network module accepts the connection request of the front-end application, authenticates and authorizes the client-end, and manages the connection, And realize the communication protocol based on Mysql, so easy to migrate to Mysql users to the system; Sql parsing and execution module to accept the client's SQL request, and then parse and execute the database stored call, Return the execution result The audit module is to store and analyze all changes to the database, the distributed management node to provide audit data; The database engine module makes use of the non-relational OrientDB database engine for reliable data storage; Distributed modules use hazlcast to implement a database cluster. Based on the above functional modules, JSQL has high availability, scalability, load balancing and other characteristics, while taking into account the database security audit

needs from the bottom of the database, Joined the data audit graphical interface shows the audit results.

The thesis has carried on the function and the performance test to the system. The result of function test shows that the system conforms to the basic requirements of distributed database functionally, and the function of auditing system also meets the requirements of this dissertation. Papers through the performance test results analysis, that the performance of the system basically meet the requirements of this paper. However, the system of complex SQL statement support is not perfect, and finally proposed an improved program.

Keywords: Distributed database, relational database, security audit, Mysql, Non-relational database

目 录

第一章 系统实现

按照前一章的设计，本章给出每个功能模块的具体实现。详细分析关键功能实现流程图，也包括对关键代码的分析。

1.1 代码规范和总体结构

本数据库系统采用和JAVA语言类似的 Kotlin语言开发。JAVA中的代码以包为组织单位，这样组织代码的好处是逻辑结构分明。表1-1描述了本系统所有的包和每个包的详细功能。其中每个包实现具体的功能，这样分配代码在大型工程实践中很常见。除了逻辑清晰，方便管理以外，同时也方便团队协作。 在所有包里

表 1-1 本系统所有包和各个包的作用

包	作用
io.jsql	启动或者关闭服务器
io.jsql.audit	审计监控功能
io.jsql.cache	数据库缓存
io.jsql.config	配置功能，读取外部配置文件
io.jsql.hazelcast	集群功能实现
io.jsql.mysql	实现mysql通信协议
io.jsql.netty	Netty实现高性能网络前端
io.jsql.orientstorage	嵌入式存储引擎模块
io.jsql.shutdown	关闭数据库功能
io.jsql.springaop	面向切面，实现日志等功能
io.jsql.sql	实现SQL解析模块
io.jsql.storage	存储引擎接口
io.jsql.test	测试包
io.jsql.util	所有常用的工具类

面，config包没有实现具体的功能，这个包主要包括了系统常用的配置信息。图1-2给出了config包下面每个类的具体作用，这个包主要是让用户可以配置数据库的各个方面，比如配置数据库的端口号，配置最大的连接数等等。 在本章的后面会给出其他其他功能模块的详细实现。

1.2 客服端功能实现

客服端主要实现负载均衡功能，利于JDBC连接后端的分布式数据库，利用网络连接到分布式管理节点得到分布式数据库节点的IP地址。在分布式管理节点为

表 1-2 config包下面每个类的作用

类	作用
Capabilities	处理能力标识定义
ErrorCode	所有的错误代码
Fields	字段类型及标识定义
Isolations	事务隔离级别定义
Versions	本系统的版本号，通讯包

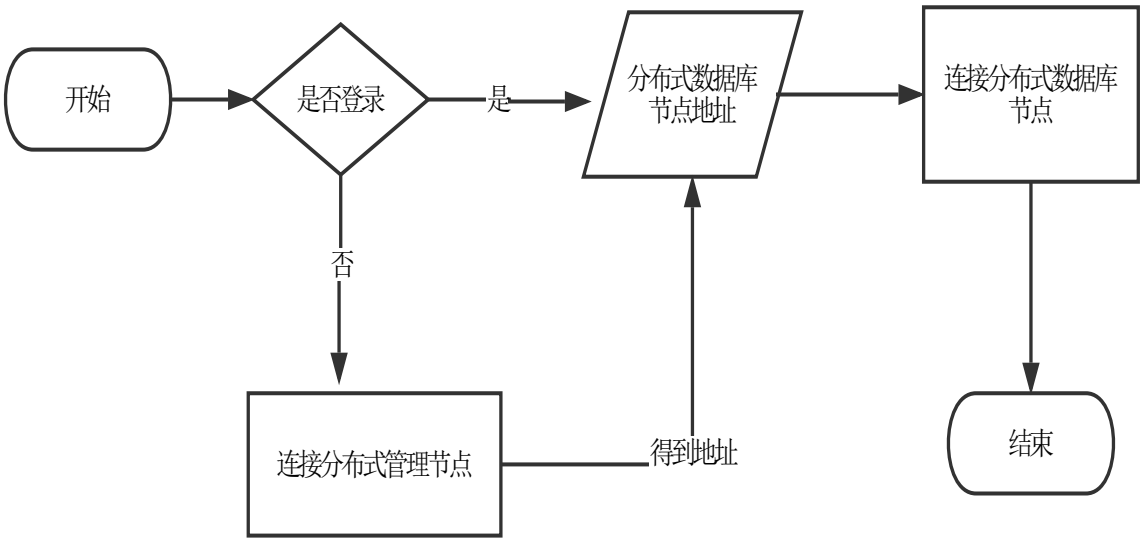

图 1-1 客服端连接功能流程图

客服端返回正确数据库地址以后，客服端就可以通过JDBC连接数据库节点。客服端连接功能流程图如图1-1所示。

首先，判断客服端有没有已经登录的会话，如果已经有已经登录的会话，那么就可以直接和对应的分布式数据库节点连接。发送数据库命令并且得到返回结果。当客服端没有连接会话的时候，客服端就首先连接分布式管理节点，分布式管理为客服端返回合适的数据节点地址，然后客服端利用这个地址再连接数据库服务器。通过这样一个步骤，分布式管理节点能利用合适的负载均衡算法为客服端选择合适的数据库节点，有利于服务器资源的均衡利用。

1.3 分布式管理节点实现

管理节点实现了负载均衡功能，实现了布式数据库节点的均衡使用。同时也为管理员提供管理和监控功能。本节主要对分布式管理节点的负载均衡的实现进行说明。分布式管理节点负载均衡算法示意图如图1-2所示。在接收到客服端的连接请求以后，分布式管理节点首先需要删除当前客服端之前的所有会话信息，同时更新响应的分布式节点的元数据。然后管理节点利用管理员设置的负载均衡算法和元数据节点信息计算出选择的节点地址。最后通过网络接口返回给客服端地址。

1.4 数据库系统实现

数据库系统功能在是分布式数据库节点中最重要的功能模块，提供数据的更删改查的功能。在源代码实现里面， SQL包下面的类主要是作为数据库系统管理类，让用户启动数据库服务器或者关闭数据库服务器。表1-3描述了SQL包下每个类的作用。 其中SpringMain类是数据库系统功能的入口，这个类主要调用其他模

表 1-3 SQL包的每个类的作用

类	功能
ShutdownMain	关闭本机服务器
SpringMain	启动类，通过spring boot启动。 ioc，aop功能

块，完成数据库系统的启动。 ShutdownMain类用来关闭数据库系统。按照客服端请求信息的处理流程，数据库功能模块主要包括网络模块，SQL解析模块和存储引擎模块，下面分别描述每个模块具体的实现。

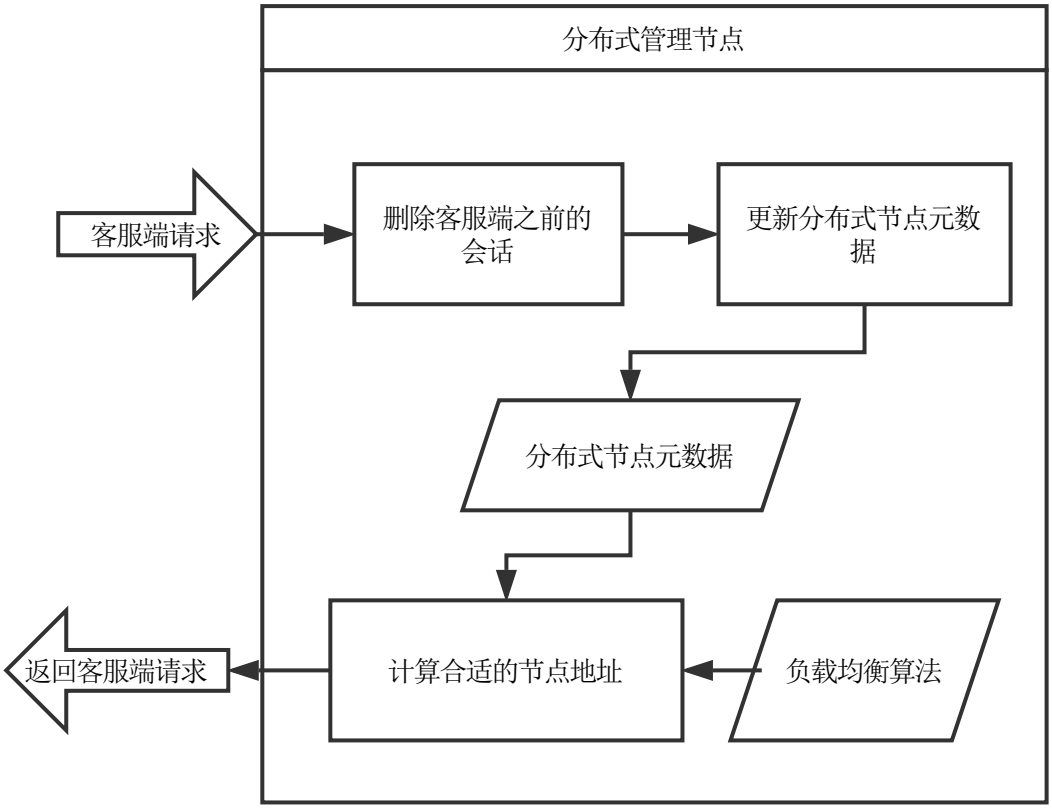

图 1-2 分布式管理节点负载均衡算法示意图

1.4.1 网络模块

在网络模块，处理网络套接字的连接和网络字节的处理。在分布式数据库节点接收到客服端的套接字连接请求以后，网络模块服务器端套接字接收请求，完成三次握手建立连接以后。交个Netty来处理。网络字节处理模块主要是用了Netty来实现的，所有和网络字节处理有关的代码都在netty包下面。表1-4描述了每个类的功能。 在接收到客服端套接字连接以后，客服端信息会依次经过下面四个类的处

表 1-4 网络模块中各个类的作用

类	作用
ByteToMysqlDecoder	接受客服端的字节消息，转化为mysql的消息格式
ByteToMysqlPacket	包字节格式转化为包对象
MysqlPacketHander	解码器，处理接受到的包对象
NettyServer	前端线程池，接受客服端的请求

理：

1.ByteToMysqlDecoder

2.ByteToMysqlPacket

3.MysqlPacketHander

4.NettyServer

具体处理流程如图1-3所示。 网络模块收到客服端的套接字连接以后需要包网络字节包装成Mysql的消息包格式，如果错误就说明客服端不是我们的客服端或者客服端发送了其他的错误。当截取到消息包格式以后，还需要解析成我们的数据结构，如果解析成功就包数据结构发送到解析模块，如果解析错误，就断开连接。

除了实现网络模块以外，我们还要实现通信协议，系统采用了和mysql一样的通信协议，在mysql包下面实现了mysql的通信协议。表1-5给出了实现通信协议所用到的所有的类。

网络服务器接受到客服端的连接以后，就要解析mysql的通信协议，把每一个消息包封装到具体的对象里面，表1-6描述了所有的mysql通信协议的封装对象。mysql协议里面有很多的包，每个协议包都需要一个类来实现，图1-4是握手包的类图， 其他包的实现大体相同，所以在这里不再重复说明。

解析到协议包以后我们就要对协议包进行处理，协议包的主要有两个阶段，一个是认证阶段，一个是命令阶段，认证阶段就是接受客服端的请求，然后检查用户的认证信息，比如用户名和密码，图1-5显示了认证流程过程。 如果认证成功以后，就建立连接会话，然后客服端就可以进行向服务器发送命令请求向分布式数据库节点处理数据。如果失败失败就返回给客服端失败原因。认证成功以后

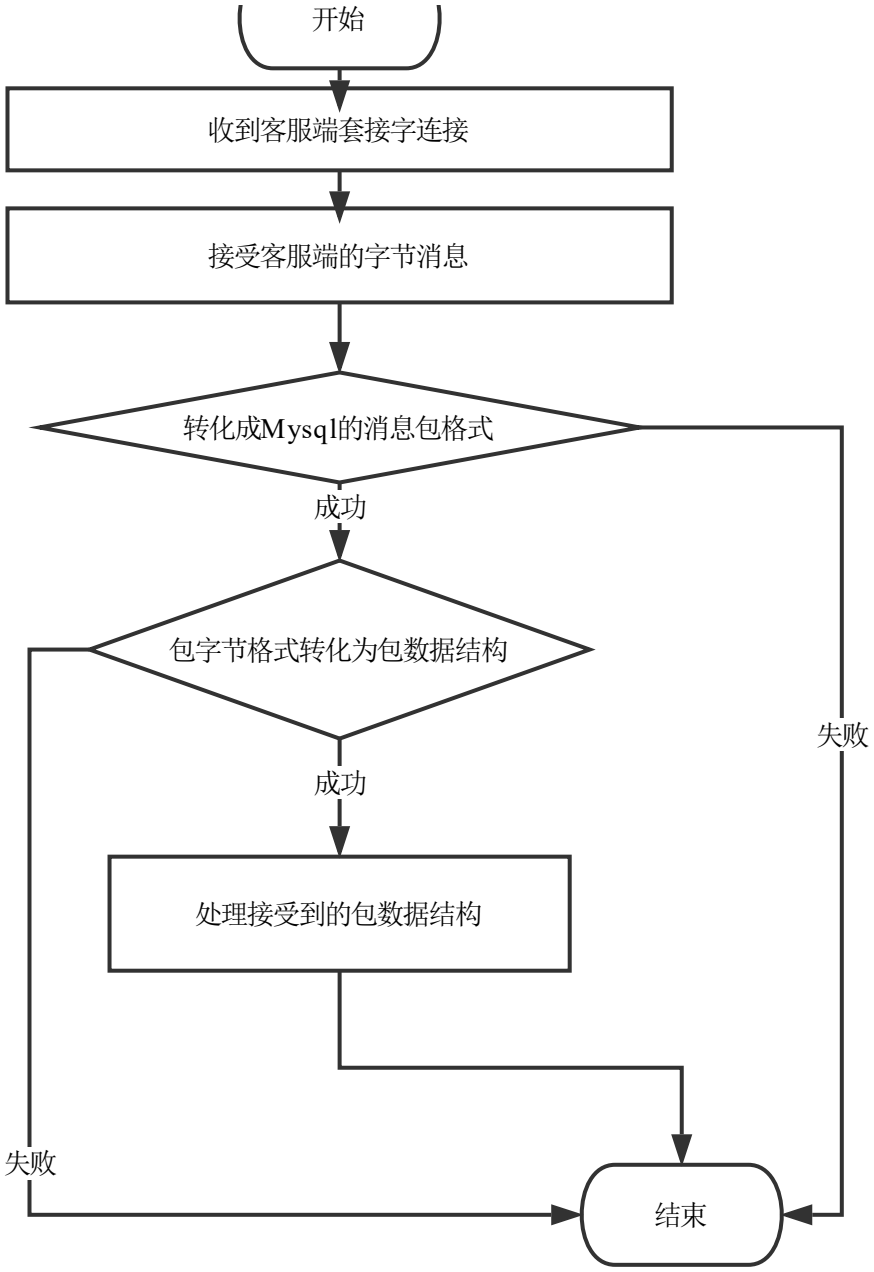
开始
收到客服端套接字连接
接受客服端的字节消息
转化成Mysql的消息包格式
成功
包字节格式转化为包数据结构
成功
处理接受到的包数据结构
失败
失败
结束

图 1-3 网络模块功能图

表 1-5 通信协议实现所用到的类

文件	类
BindValue	class BindValue
BindValueUtil	object BindValueUtil
BufferUtil	object BufferUtil
ByteUtil	object ByteUtil
CharsetUtil	object CharsetUtil
MBufferUtil	object MBufferUtil
MySQLMessage	class MySQLMessage
PacketUtil	object PacketUtil
PreparedStatement	class PreparedStatement
SecurityUtil	object SecurityUtil
StreamUtil	object StreamUtil

表 1-6 mysql所有协议包的封装对象

文件	解释
AuthPacket	From client to server during initial handshake.
BinaryPacket	class BinaryPacket : MySQLPacket
CommandPacket	From client to server , the client wants the server to do something
EOFPacket	the EOF packet contains a warning
EmptyPacket	class EmptyPacket : MySQLPacket
ErrorPacket	From server to client in response to command, if error.
ExecutePacket	class ExecutePacket : MySQLPacket
FieldPacket	part of Result Set Packets. One for each column in the result set.
HandshakePacket	From server to client during initial handshake
HandshakeV10Packet	Connection Phase The Connection Phase performs these tasks
HeartbeatPacket	From client to server when the client do heartbeat between jsql cluster
LongDataPacket	class LongDataPacket : MySQLPacket
MySQLPacket	abstract class MySQLPacket
OkPacket	From server to client ,if no error and no result set
PingPacket	class PingPacket : MySQLPacket
PreparedOkPacket	class PreparedOkPacket : MySQLPacket
QuitPacket	class QuitPacket : MySQLPacket
Reply323Packet	class Reply323Packet : MySQLPacket
RequestFilePacket	load data local infile
ResetPacket	class ResetPacket : MySQLPacket
ResultSetHeaderPacke	The Result Set Header Packet is the first of several packets
RowDataPacket	From server to client. One packet for each row in the result set

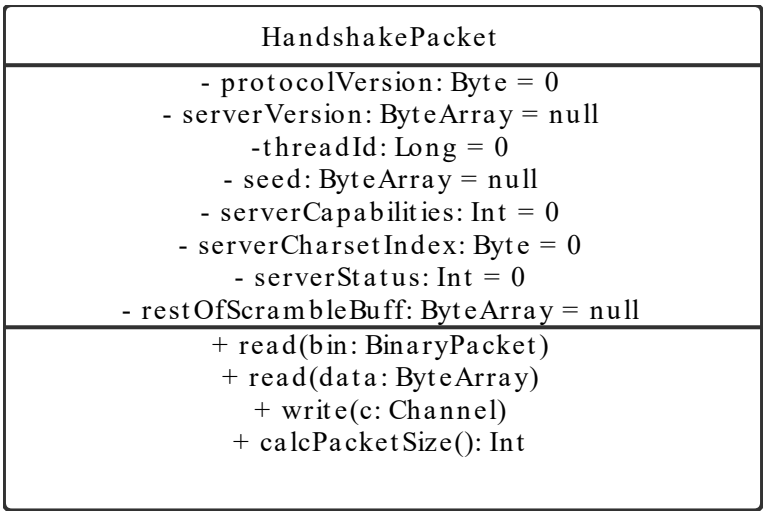

图 1-4 握手包实现类图

就是命令阶段，服务器接受客服端的命令，然后处理，图1-6显示了命令处理流程。 在接收到客服端的命令请求以后，网络模块就会调用下面模块的功能接口来对这个命令进行处理。如果命令处理过程中没有发生异常，那么就把命令处理结果发送给客服端，如果命令处理工程中发送了异常，就返回失败原因。

1.4.2 SQL解析模块

经过网络模块的处理，我们建立了客服端的连接会话，接下来我们就要对客服端发送过来的命令进行判断，如果是SQL命令，就要对这个命令进行解析，进行词法分析，语法分析和语意检查。在代码实现中，sql包下面的类主要完成对客服端连接和会话的封装和管理，表1-7显示了sql包下面每个类的具体作用。在sql解析功能模块里面系统用到了druid开源框架，用来解析sql，解析sql以后就

表 1-7 SQL前端连接模块相关的类

类	作用
MysqlSQLhander	接受前端的语句，处理用户的请求
OConnection	前端连接对象，一个客服端一个连接
ONullConnection	继承连接对象，但是不处理具体任务，用在分布式中
OconnectionPool	管理前端的连接，作为一个连接池对象

要做具体的数据处理，图1-7显示了SQL模块的处理流程图。 在sql解析流程中，如果遇到Druid框架不支持的语句，我们就要自己对这个语句进行解析。再这个工程中。任何一个步骤发送错误就返回错误信息给客服端。不然就进行发送给下一个功能模块进行具体的数据操作。 mysql当中的sql语句有很多种，每一种语句都要做不同的处理， 1-8描述了sql模块下实现的各种类型的语句。

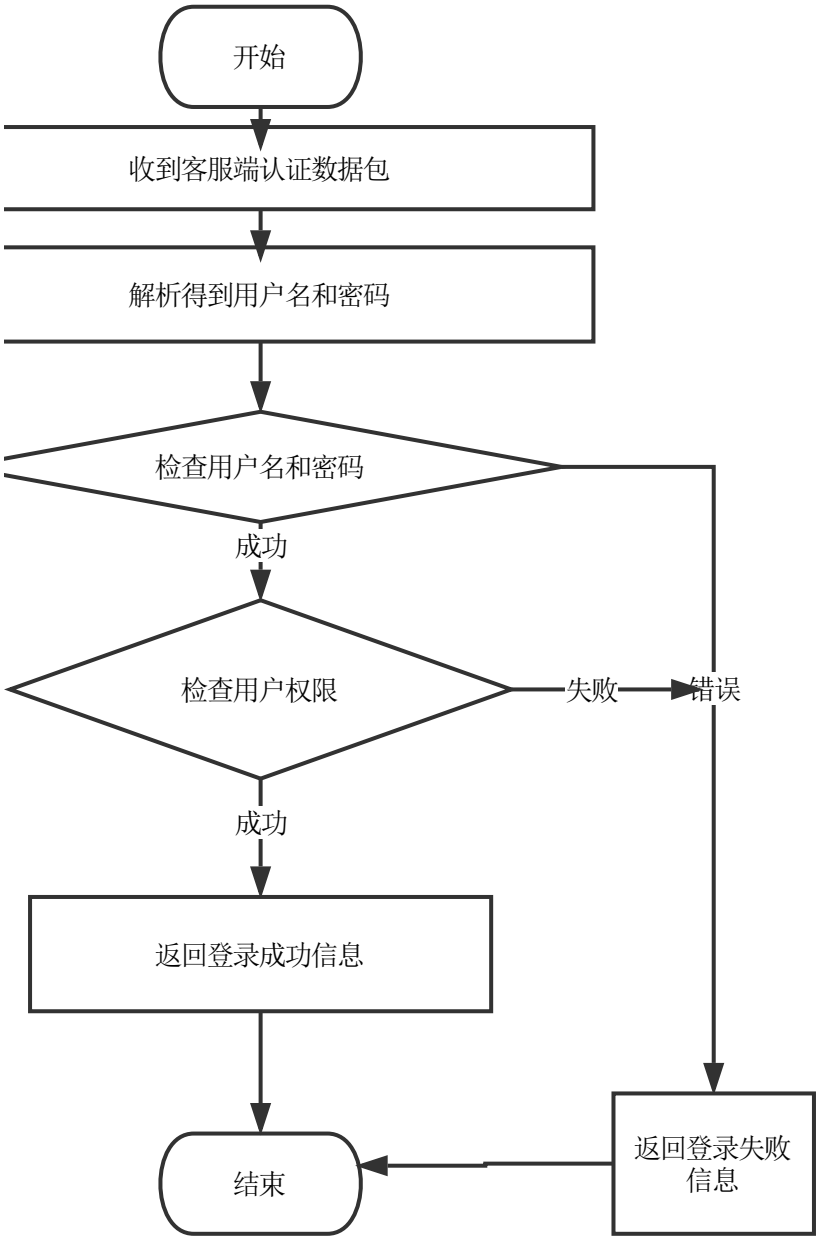
开始
收到客服端认证数据包
解析得到用户名和密码
检查用户名和密码
成功
检查用户权限
失败
错误
成功
返回登录成功信息
结束
返回登录失败信息

图 1-5 认证流程图

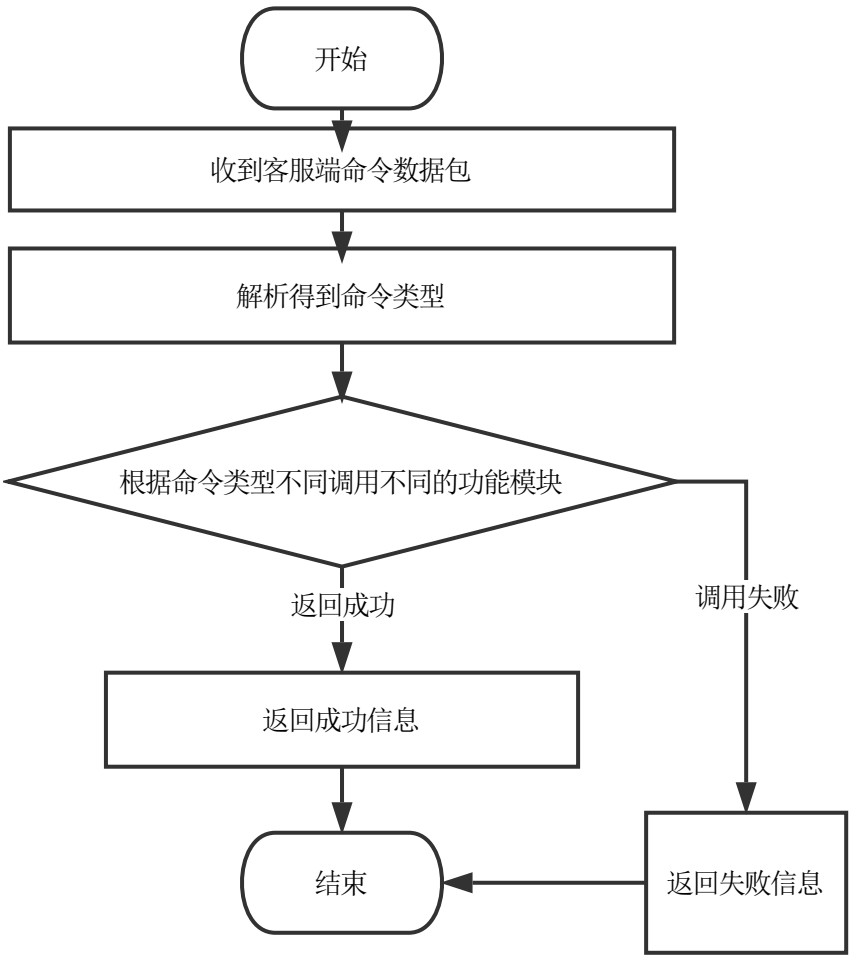

图 1-6 命令处理流程图

表 1-8 SQL模块下实现的各种类型的语句

类	作用
io.jsql.sql.handler.adminstatement	处理管理语句
io.jsql.sql.handler.componed_statement	处理组合语句
io.jsql.sql.handler.data_define	处理数据定义语句
io.jsql.sql.handler.data_mannipulation	处理数据操作数据
io.jsql.sql.handler.preparestatement	处理准备语句
io.jsql.sql.handler.replication_statement	处理复制语句
io.jsql.sql.handler.tx_and_lock	处理事务和锁控制语句
io.jsql.sql.handler.utilstatement	处理其他工具语句

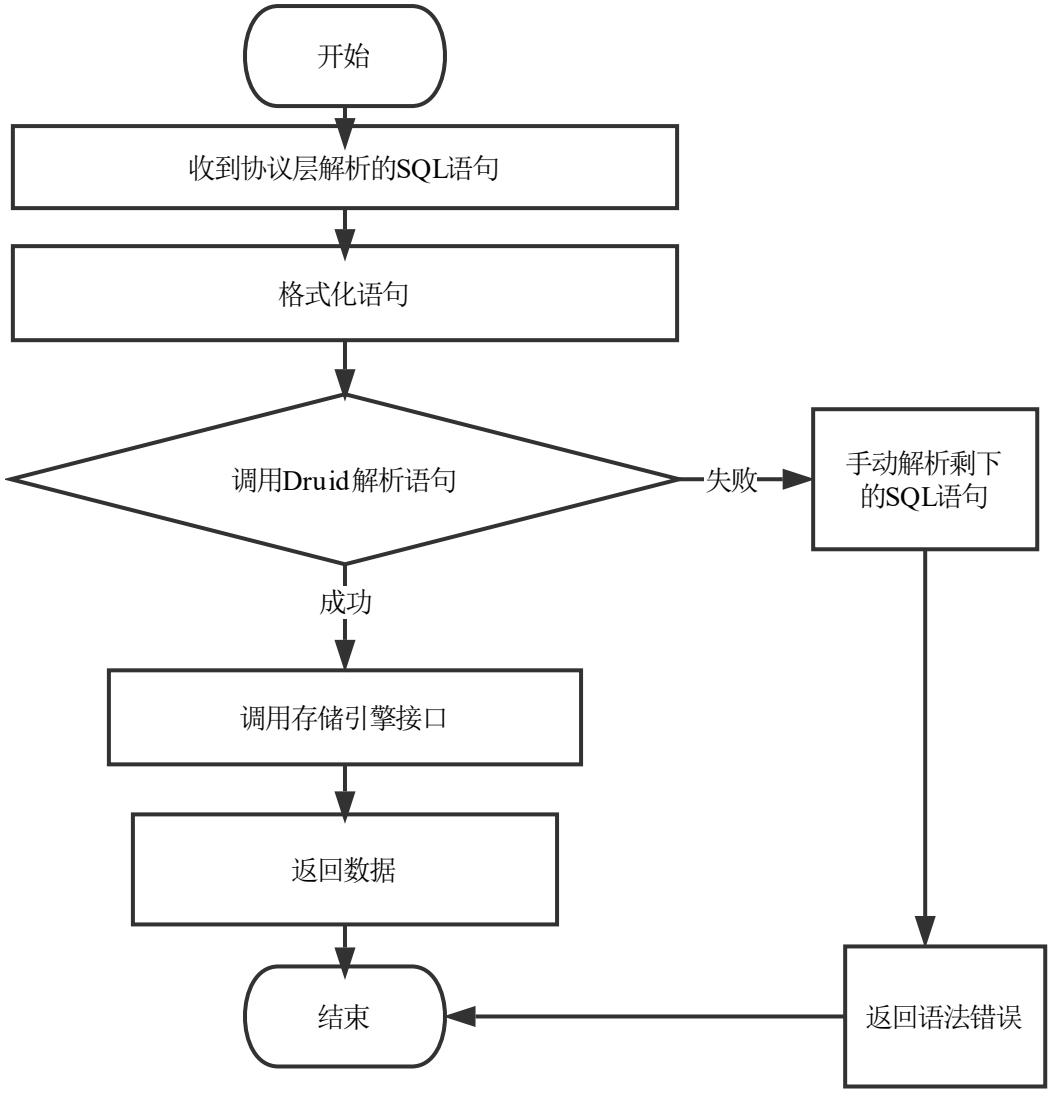

图 1-7 SQL处理流程图

每一种类型语句下面又回有很多种具体语句的实现，表1-9给出了数据操纵类型语句实现的所有相关的类和响应的功能。其他类型的语句也需要进行相同的处理，这里就不再进行详细表述。

表 1-9 数据操纵数据实现所相关的类

文件	作用
MSelectHandler	处理查找语句
Mcall	处理函数调用语句
Mdelete	处理删除语句
Mdo	处理mysql中的do语句
Mhandler	所有类的基类
Minsert	处理插入语句
MloaddataINfile	处理导入文件的语句
Mloadxml	处理导入xml的语句
Mrepelace	处理解释语句
MselectVariables	处理查找变量的语句
Msubquery	处理子查询
Mupdate	处理更新语句

1.4.3 存储引擎模块

存储引擎有关的功能主要在storage包下面实现，表1-10给出了 storage包下面各种文件的功能。 其中DB类作为底层存储引擎的接口，他的功能接口如表1-11所示。 和表有关的功能接口如表1-12所示。

表 1-10 storage包下面各种文件的作用

文件	功能
DB	接口，规定所有的存储引擎应该实现的和数据库有关的功能
Table	接口，规定所有的存储引擎应该实现的和表有关的功能

存储引擎利用非关系型数据存储，为上面模块提供操作接口。其中有关数据库操作的类接口如图1-8b所示，有关表操作的类接口如图1-9所示。类的实现主要是封装了OrientDB存储引擎的功能，为上一次提供关系操作的接口。在关系数据库中，操作接口就是SQL语言。JSQL因为选择了兼容Mysql协议，所以实现了大部分的Mysql语句功能。这样上层功能模块就可以当做Mysql一样的使用本数据库引擎。

1.5 集群架构的实现

集群功能主要是利用了开源的hazelcast框架来实现，其中的功能全在hazelcast包下面实现，表1-13给出了该包下面每个类的具体的作用。 其中最关键的代码如

表 1-11 数据库存储引擎的函数接口

接口	函数签名
close	abstract fun close(): Unit
createdbAsyn	abstract fun createdbAsyn(dbname: String): Unit
createdbSyn	abstract fun createdbSyn(dbname: String): Unit
deletedbAyn	abstract fun deletedbAyn(dbname: String): Unit
deletedbSyn	abstract fun deletedbSyn(dbname: String): Unit
exe	abstract fun exe(sql: String, db: String): Unit
exesqlNoResultAsyn	abstract fun exesqlNoResultAsyn(sql: String, dbname: String): Unit
exesqlforResult	abstract fun exesqlforResult(sql: String, dbname: String):Oresult
getallDBs	abstract fun getallDBs(): List<String>
getdb	abstract fun getdb(dbname: String): ODatabaseDocument
query	abstract fun query(sqlquery: String, dbname: String): Stream<OElement>

表 1-12 数据库引擎和表有关功能的接口

函数	函数签名
createtableSyn	abstract fun createtableSyn(dbname: String, createTableStatement
droptableSyn	abstract fun droptableSyn(dbname: String, table: String): Unit
getalltable	abstract fun getalltable(dbname: String): List<String>
gettableclass	abstract fun gettableclass(tablename: String, db: String): OClass
selectSyn	abstract fun selectSyn(oClass: OClass, dbname: String): Stream<OElement>

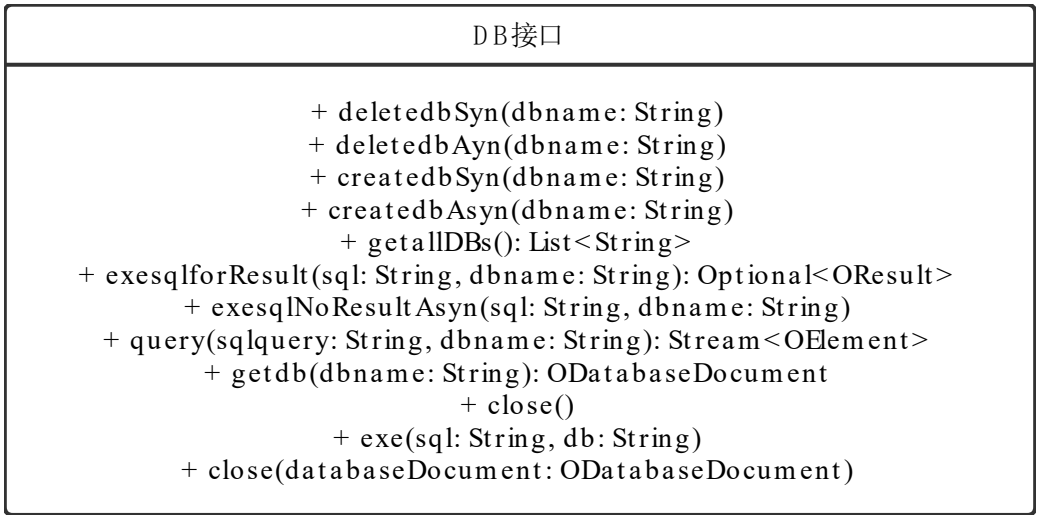

图 1-8 数据库操作接口类图

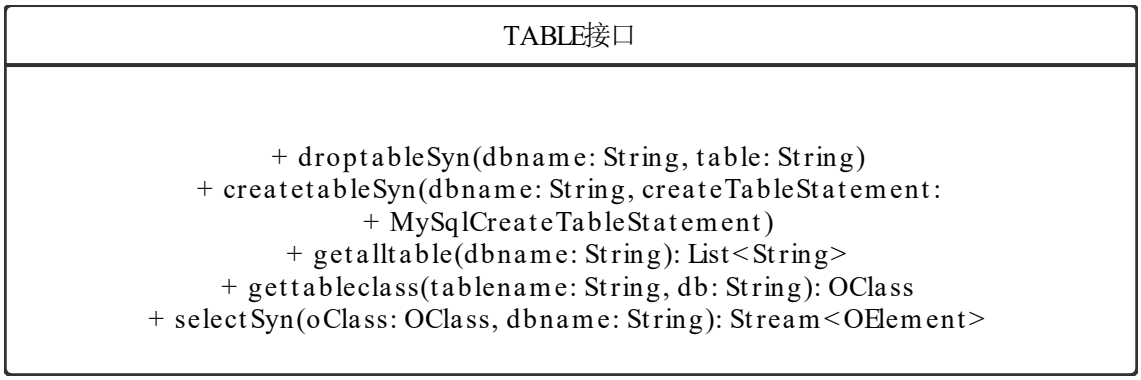

图 1-9 表操作接口类图

表 1-13 集群模块下面各个类的作用

类	作用
LogFile	本地文件系统中存储日志文件
MyHazelcast	利用Hazelcast的分布式数据结构实现集群，比如分布式队列和分布式锁
ReplicationCMD	分布式对象，在不同集群之间传输
SqlUpdateLog	当前系统LSN最大值，新的事务日志LSN将在此基础上生成

下。

```
//sql队列 复制队列命令队列。2个锁。发布
@Component
class MyHazelcast : ItemListener<SqlUpdateLog> {
    private fun exeSqlforReplication() {
        Collections.sort(localqueneReplication)
        var log: SqlUpdateLog? = localqueneReplication.poll()
        var lastlsn: Long = 0
        while (log != null) {
            locals_maxlsn = log.LSN
            logger.info("exeSqlforReplication exe sql " + log)
            oNullConnection!!.schema = log.db
            sqLhander.handle(log, oNullConnection)
            logFile!!.write(log)
            lastlsn = log.LSN
            log = localqueneReplication.poll()
        }
```

```
17          isreplicating = false
18          log = localquene.poll()
19          val remotel = iAtomic_remote_lsn!!.get()
20          while (log != null) {
21              locals_maxlsn = log.LSN
22              logger.info("exe  Sql : " + log)
23              oNullConnection!!.schema = log.db
24              sqLhander.handle(log, oNullConnection)
25              logFile!!.write(log)
26              lastlsn = log.LSN
27              if (lastlsn > remotel) {
28                  remotequene!!.offer(log)
29              }
30              log = localqueneReplication.poll()
31          }
32          if (lastlsn > remotel) {
33              iAtomic_remote_lsn!!.set(lastlsn)
34          }
35      }
36      /**
37       *  本机发出的sql语句.记录到本地logfile。
38      同时发布到其他服务器*/
39      fun exeSql(sql: String, db: String) {
40          if (isreplicating) {
41              val l = iAtomic_remote_lsn!!.get()
42              val log = SqlUpdateLog(l + 1, sql, db)
43              localquene.addLast(log)
44          } else {
45              locals_maxlsn++
46              val l = iAtomic_remote_lsn!!.addAndGet(1)
47              val log = SqlUpdateLog(l, sql, db)
48              logger.info("exe sql " + log)
49              logFile!!.write(log)
```

```
50              remotequene!!.offer(log)
51          }
52      }
53      private fun exesqlforremoteData() {
54          Collections.sort(localquene)
55          var log: SqlUpdateLog? = localquene.poll()
56          var last: Long = 0
57          while (log != null) {
58              locals_maxlsn = log.LSN
59              logger.info("exesqlforremoteData()  " + log)
60              oNullConnection!!.schema = log.db
61              sqLhander.handle(log, oNullConnection)
62              logFile!!.write(log)
63              last = log.LSN
64              log = localquene.poll()
65          }
66          val l = iAtomic_remote_lsn!!.get()
67          if (last > l) {
68              iAtomic_remote_lsn!!.set(last)
69          }
70
71      }
72
73  }
```

1.6 数据审计模块的实现

JSQL除了结合关系数据库和非关系型数据库以外，还从数据库的底层考虑数据的安全问题，对数据审计功能进行了简单的实现，在代码实现中，审计模块有关的功能全部在audit包下面实现，表1-14给出了audit包下面每个类的具体的作用。其中LoginLog和SQLlog类主要封装了两种日志类型，一种是客服端的登录记录日志，一种是客服端的命令执行日志。通过封装这两种日志记录，很容易的就可以实现审计数据的收集和发送功能。图1-10显示了审计功能演示图。 审计功能主要

表 1-14 audit包下面每个类的作用

类	作用
LoginLog	简单对象，记录登陆日志
SqlLog	简单对象，记录SQL执行日志
elasticUtil	工具类，包日志记录发送到ELK

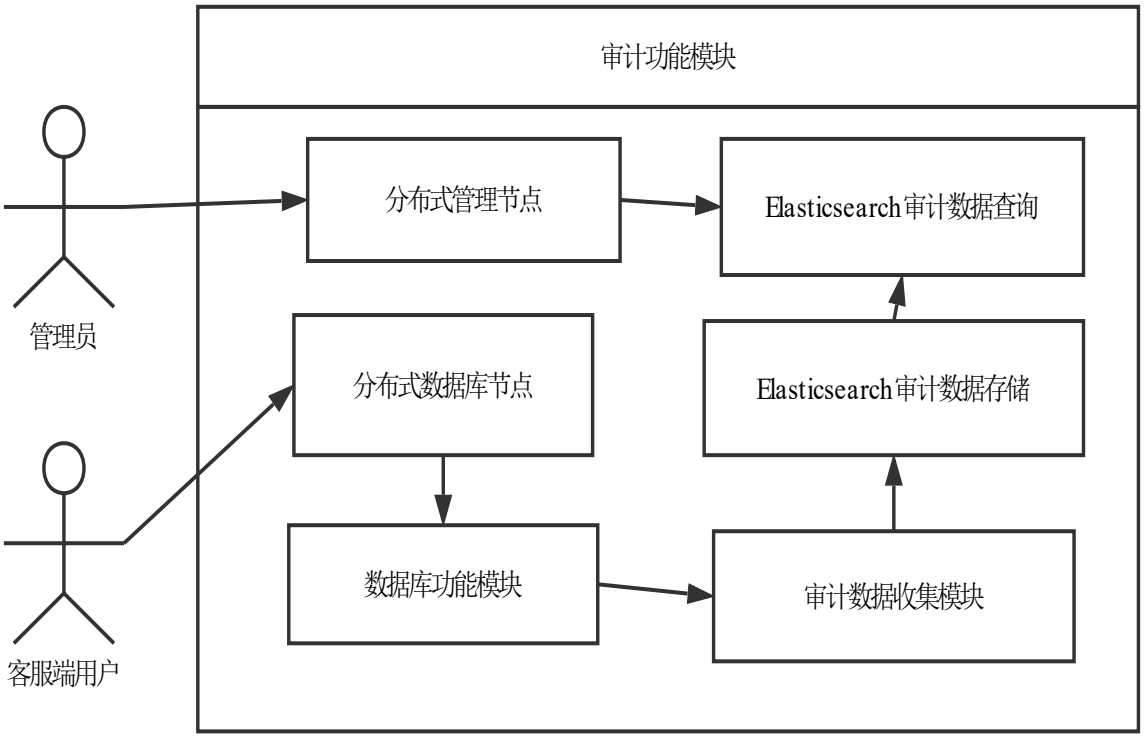

图 1-10 审计功能演示图

包括审计数据库，审计管理器和审计可视化功能模块，下面分别对每个模块进行说明

1.审计数据库

审计数据库用Elasticsearch来实现，作为审计数据库，不能让用户随意的更改，所以本系统更改了它的源代码，使得它只能增加数据和查找数据不能更改和删除数据。其中主要存储的对象如图1-14所示。

2.审计管理器

审计管理功能全部在audit下面实现，主要是存储本地的日志文件，然后发布到审计数据库。提供给前端可视化的接口。

3.审计可视化模块的实现

审计可视化模块的实现用到了grafana，主要用来监控ELK中的数据。

1.7 本章小结

本文前面一章设计了本系统的架构图和各个模块的详细功能，本章给出每个功能模块的具体实现。。

致 谢

时间过的真快，转眼间，2年多的研究生生活就快要结束了。回顾自己的研究生学习生涯，感慨万千。论文的完成，除了自己的努力以外，更离不开老师和学弟们的帮助，在论文成稿之际，衷心感谢给予自己悉心指导和热情帮助的各位老师和学弟们。

论文的完成，首先要感谢我的校内导师曹晟教授，在整个论文的的写作过程中，曹老师都给予了我很大的关心和帮助。从作者毕业论文的选题、写作一直到最终完成的过程中，曹老师都是在百忙的工作中以一贯认真负责的态度认真仔细阅读作者的论文，给予作者耐心的指导，使得论文能够顺利的完成。他严肃的科学态度，严谨的治学精神，以及精益求精的工作作风，深深地感染和激励着我。

在这一年多的时间里，我还要感谢我的学弟们。感谢你们陪我一起开发这个分布式数据库系统，我们一起学习，一个努力，才能让本系统顺利完成。任何一个系统都要靠团队合作，更何况分布式系统这个既具有挑战性的工程项目，没有你们的帮助，就不可能按时完成这个系统。对学弟们的帮助，在此表示非常的感谢。

最后，作者非常感谢负责评审论文的教师、专家和教授，感谢你们认真负责的阅读论文，感谢你们为论文提出的宝贵意见和建议。

参考文献

[1] 杨东, 谢菲. 分布式数据库技术的研究与实现[J]. 电子科学技术, 2015, 02(01):68–71

[2] 杨传辉. 大规模分布式存储系统[M]. 北京: 机械工业出版社, 2013, 56–60

[3] 马应龙. 分布式系统概念和设计[M]. 北京: 机械工业出版社, 2013, 56–60

[4] 安延文. 数据库审计系统中MySQL协议的研究和解析[D]. 河北: 华北电力大学, 2013, 50–60

[5] 贺杰. 分布式数据库中数据复制的研究和实现[D]. 南京: 东南大学, 2014, 50–60

[6] 邓蕾. 基于关联规则的数据库安全审计系统[D]. 长沙: 中南大学, 2011, 50–60

[7] 黄贵. OceanBase分布式存储引擎[N]. 华东师范大学学报, 2014年9月

[8] 李黎明. 安全数据库概述和前瞻[R]. 北京: 南京航天大学信息与技术学院, 2005年5月

[9] Wikipedia. Hard disk drive [EB/OL]. https://en.wikipedia.org/wiki/Hard_disk_drive

[10] Wikipedia. B+ tree [EB/OL]. https://en.wikipedia.org/wiki/B

[11] Wikipedia. Log-structured merge-tree [EB/OL]. https://en.wikipedia.org/wiki/Log-structured_merge-tree

[12] Wikipedia. Netty [EB/OL]. https://en.wikipedia.org/wiki/Netty

[13] Wikipedia. Elasticsearch [EB/OL]. https://en.wikipedia.org/wiki/Elasticsearch

[14] Wikipedia. Apache JMeter [EB/OL]. https://en.wikipedia.org/wiki/Apache_JMeter

[15] Wikipedia. Hazelcast [EB/OL]. https://en.wikipedia.org/wiki/Hazelcast

[16] 王智慧. 安全数据库审计子系统[D]. 上海: 复旦大学, 2011, 50–60

[17] 张瑞芳. 分布式数据库的查询优化方法设计与实现[D]. 成都: 电子科技大学, 2010, 50–60

[18] 刘蓬. NIO高性能框架的研究与应用[D]. 长沙: 湖南大学, 2013, 50–60

[19] 王威. MySQL 数据库源代码分析及存储引擎的设计[D]. 南京: 南京邮电大学, 2012, 50–60

[20] 别小凡. 分布式内存数据库的设计与实现[D]. 西安: 西安电子科技大学, 2012, 50–60

[21] 解辉. 嵌入式数据库的设计与实现[D]. 太原: 太原科技大学, 2008, 50–60

[22] 杨磊. 数据库安全审计检测系统的设计与实现[D]. 北京: 北京交通大学, 2014, 50–60

[23] 张忠能. 分布式数据库关键技术研究与应用[D]. 上海: 上海交通大学, 2015, 50–60

[24] 王琳. 数据库监控系统的设计与实现[D]. 天津: 南开大学, 2011, 50–60

[25] 谢鹏. 分布式数据库存储子系统设计与实现[D]. 成都: 电子科技大学, 2013, 50–60

[26] 张俊民. 在线审计建模与实现[D]. 河北: 华北电力大学, 2015, 50–60