分布式数据库项目

——大规模信息系统构建技术导论课程项目

个人设计报告——马良

指导教师：鲍凌峰

组长：马良

组员：王伊斓、罗依民

日期：2023 年 5 月 21 日

1 引言

1.1 系统目标

本项目是《大规模信息系统构建技术导论》的课程项目。项目内容为：在大二春夏学期《数据库系统》课程的基础上，结合《大规模信息系统构建技术导论》所学知识，实现一个分布式关系型简易数据库系统，实现数据分布、集群管理、分布式查询、副本管理、容错容灾、负载均衡等功能。

1.2 设计说明与任务分工

本系统由小组 3 位成员合作编写完成，使用 Java 开发，VScode 作为集成开发环境，Github 进行合作开发，每个组员都完成了目标任务，具体的分工安排如下：

表格 1 成员信息及分工

成员姓名	学号	分工职责
马良	3200105143	Master Server、Client、Region Server 之间的 Socket 通讯 Client、Master Server 代码 最终程序的集成与功能调试 视频录制
罗依民	3200105307	Region Server 代码 最终程序的集成与功能调试 视频录制
王伊斓	3200105318	Master Server 中的 Zookeeper 部分，负载均衡，FTP 操作 报告撰写、汇报 PPT 制作

2 个人负责模块设计

2.1 Client 部分

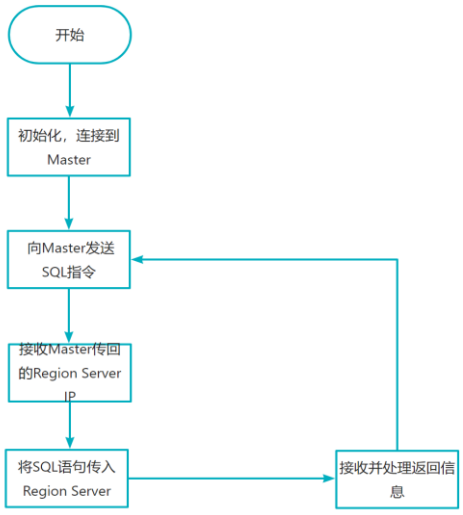
Client 模块是分布式数据库系统中直接与用户进行交互的模块，主要功能为：

1. 接收用户输入的 SQL 指令；
2. 执行 SQL 指令，获取语句执行结果，将执行结果呈现给用户。

具体执行方式：

创建数据表：向 Master Server 节点询问是否存在同名数据表，如不存在，则 Master Server 返回适合创建该表的最佳 Region Server IP 地址；

对表的查询操作：向 Master Server 节点询问数据表存在的 Region Server 地址后，与对应 Region Server 建立 Socket 通讯连接，接收由 Region Server 处理语句后发送回 Client 的消息。



2.2 Master 代码

Master Server 主服务器主要负责进行信息管理、信息接收、负载均衡、容错容灾等功能的实现。其中信息管理指的是使用 zookeeper 监听管理 Region 节点，从节点上线目录，管理结点与对应 ip、对应表信息。

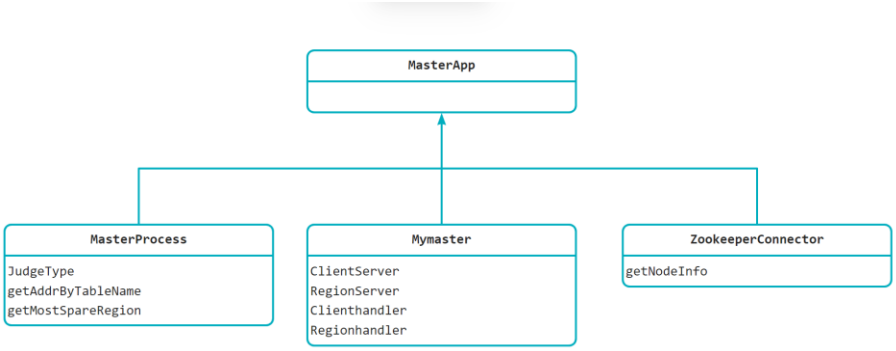
通过建立三条线程分别监听 zookeeper，客户端，从节点的变换。

对客户端监听任务为获取 sql 语句并解析其中的表名，为客户端返回操作节点，如

果为 create 类型则进行负责均衡。

对 zookeeper 监听任务为接收 region 表目录变动情况，并建立映射表

对从节点监听任务为接收从节点上线目录，获取从节点情况



代码部分：

```
public static void main(String args[]) throws IOException{
    Mymaster master = new Mymaster();        You, 上周 • 代码重构加bug修改 …

    Thread regionThread = new Thread(new Runnable(){
        @Override
        public void run(){
            master.RegionServer(params.MasterPort_Region);
        }
    });
    regionThread.start();

    Thread clientThread = new Thread(new Runnable(){
        @Override
        public void run(){
            master.CilentServer(params.MatserPort_Client);
        }
    });
    clientThread.start();

    Thread zookeeper_thread = new ZookeeperConnector(Mymaster.directory);
    zookeeper_thread.start();
}
```

2.3 解析语法树

为实现查询表名 ip，需要建立语法树解析出 sql 语句中的表名称，使用了 JSqlpaser 来建立并获取表名称。

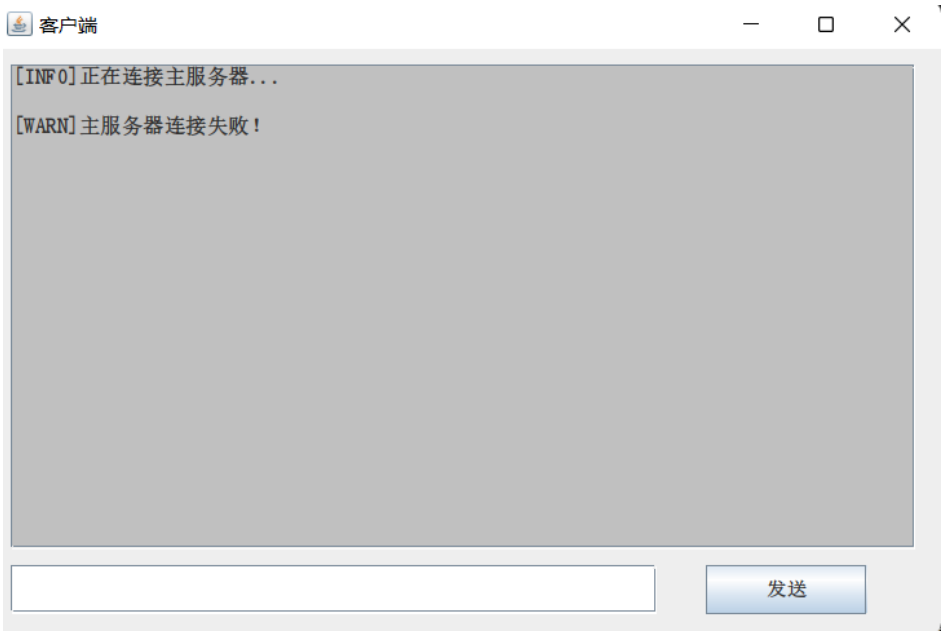
相关代码：

```
//正则表达式去除语句中的注释
public synchronized static String dSql(String sql) {
    Pattern p = Pattern.compile(regex:"(?ms)('(?:''|[^'])*')|--.*?$|/\\*.*?\\*/|#.*?$|");
    return p.matcher(sql).replaceAll(replacement:"$1");
}

//解析sql，获取表名
public synchronized static void parseSql(String sql, Set<String> tableList) {
    try {
        Statement statement = CCJSqlParserUtil.parse(sql);
        TablesNamesFinder tablesNamesFinder = new TablesNamesFinder();
        List<String> tableNameList = tablesNamesFinder.getTableList(statement);
        tableList.addAll(tableNameList);
    } catch (JSQLParserException e) {
        System.out.println(x:"解析sql出错\n");
    }
}       You, 3周前 • init
```

2.4 各部分通用窗体

为方便使用增加了窗体



相关代码如下

```
public Myclient(){
    super(title:"客户端");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(x:0, y:0, width:600, height:400);

    cc = this.getContentPane();
    setLayout(manager:null);      //设置为绝对布局
    JScrollPane jsp = new JScrollPane(jta);
    jsp.setBounds(x:10, y:10, width:560, height:300);
    jta.setBackground(Color.LIGHT_GRAY);
    jta.setFont(new Font(name:"宋体", Font.BOLD, size:13));
    jtf.setBounds(x:10, y:320, width:400, height:30);
    jb.setBounds(x:440, y:320, width:100, height:30);
    cc.add(jb);
    cc.add(jsp);
    cc.add(jtf);
    setVisible(b:true);       You, 2周前 • 搞了点ui改动 …
```

3 个人总结

本次实验中基本功能都已经实现，但部分功能模块还能够进一步完善改善，如副本管理时，仅使用了一个 FTP 服务器，而没有考虑到该 FTP 服务器出现错误运行状态后的处理方式；使用更大规模的测试数据也是测试系统性能的一个重要方法和在测试过程中所欠缺的。其次对于垂直分片也没有进行尝试。

本次实验让我对分布式的理论知识有了具体的应用，但许多模块的设计仍然较为简单，对于 java 工程的多线程和 socket 网络通讯也有了更好的实践经验，不足时在项目初期应该对于设计做更多的调整和沟通实践。