浙江大学

本科实验报告

课程名称：	大规模信息系统构建技术导论
实验名称：	分布式 MiniSQL
姓　　名：	米博宇
学　　院：	计算机学院
系：	软件工程系
专　　业：	软件工程
学　　号：	3200102888
指导教师：	鲍凌峰

2023 年　5　月　15　日

浙江大学实验报告

实验名称：分布式 MiniSQL 客户端　　实验类型：设计实验

同组学生：李毅桐、王粤龙　　实验地点：

一、实验内容

- 设计分布式 MiniSQL 的客户端模块
- 客户端模块启动后，首先要求用户名输入，然后连接 zookeeper 服务器并创建相应的客户端节点。
- 成功创建节点后，在命令行中持续接收用户的 SQL 语句输入。
- 根据用户的 SQL 语句输入，调用业务逻辑得到对应的输出结果
- 在用户输入”quit”后，客户端结束运行。

二、功能描述

1. 语法分析

客户端支持对用户输入的 SQL 语句进行语法分析。输入的 SQL 语句有语法错误时，直接提示用户语法错误，无需调用其他方法。SQL 语句没有问题时，客户端可以判断其操作类型，并执行相应的业务逻辑。

2. 支持的 SQL 语句类型

- 创建数据表
  - 支持创建数据表，支持 MySQL 的所有数据类型
- 删除数据表
  - 支持删除存在的数据表
- 插入数据
  - 支持向存在的数据表中插入制定值的数据
- 删除数据
  - 支持从表中删除一条或多条数据
- 查询数据
  - 支持多条件查询，条件之间可以 AND，OR 等逻辑词分隔
  - 支持多表查询，笛卡尔积和 left join 等复杂 join 操作

3. 与 Master 服务器交互

客户端连接成功后，会在 zookeeper 集群中注册一个对应的节点，由 Master 服务器对应的节点持续监听该节点的变化。需要进行数据表操作时，客户端会将操作的信息（操作类型、参数等）写入对应节点的缓冲区。Master 服务器对应的节点监听到变化时，根据操作信息执行相应的数据库操作。执行完成后，将执行结果写入客户端对应的节点缓冲区，客户端节点监听到缓冲区变化后，将执行结果打印在命令行中。

三、接口说明

1. 外部接口

客户端程序从命令行中接受用户键入的 SQL 语句，在获得 SQL 语句执行结果后，将结果输出至命令行供用户查看。

2. 内部接口

本模块调用 JSQLParser 的解析函数接口获取 SQL 语句的类型并得到所有操作表的名称。调用 zookeeper 的 getData 接口向 Master 服务器获取当前数据库内所有表名，判断操作表是否存在。

四、工作原理

1. 连接 zookeeper 集群

新客户端打开后，首先要求用户输入用户名，作为在 zookeeper 集群中的唯一识别方式（如果用户名被占用则提示重新输入）。然后 DistributedClient 类的成员变量 zkClient 被初始化为一个 zookeeper 对象，名称为用户名，与 zookeeper 集群连接。

zkClient 对象监听 Master 节点。Master 节点缓冲区中存放着数据库中所有数据表和 region 服务器的对应关系。对应关系发生变化时，zkClient 重新从 Master 获取对应关系表，并存储在自身的成员变量 tableRegionMap 中。

zkClient 对象监听自身节点，自身节点存放 SQL 语句的执行结果。当 Master 或 Region 将 SQL 的执行结果写入自身节点缓冲区时，DistributedClient 将其打印到命令行中。

2. 获取用户输入

连接和初始化操作完成后，客户端开始获取 SQL 语句输入。输入规则与 MySQL 相同：用户可输入任意多行，以换行符分隔，分号作为结束标记。

3. 对 SQL 进行解析

调用 JSQL Parser 模块对输入的 SQL 语句进行解析。首先检查是否有语法错误，如有，提示用户并要求重新输入。否则，判断 SQL 语句的操作类型，并提取 SQL 语句中所有的表名。支持的操作有：CREATE_TABLE, DROP_TABLE, SELECT, DELETE, INSERT, UPDATE。

CREATE_TABLE

检查要创建的表是否存在。如果存在，则提示用户；否则将用户名和 SQL 语句写入 Master 节点的缓冲区，由 Master 节点负责完成对应操作。
对于其他操作，检查 SQL 语句的表名在 tableRegionMap 中是否都存在。如果有表不存在，提示用户；否则进行对应操作，如下所示。

DROP_TABLE

将用户名和 SQL 语句写入 Master 节点的缓冲区，由 Master 节点负责完成对应操作。

DELETE, INSERT, UPDATE

根据 tableRegionMap 查询得知需要修改操作的表所在的 Region 服务器。直接向对应服务器节点的缓冲区写入用户名和该 SQL 语句，由 Region 负责执行。

SELECT

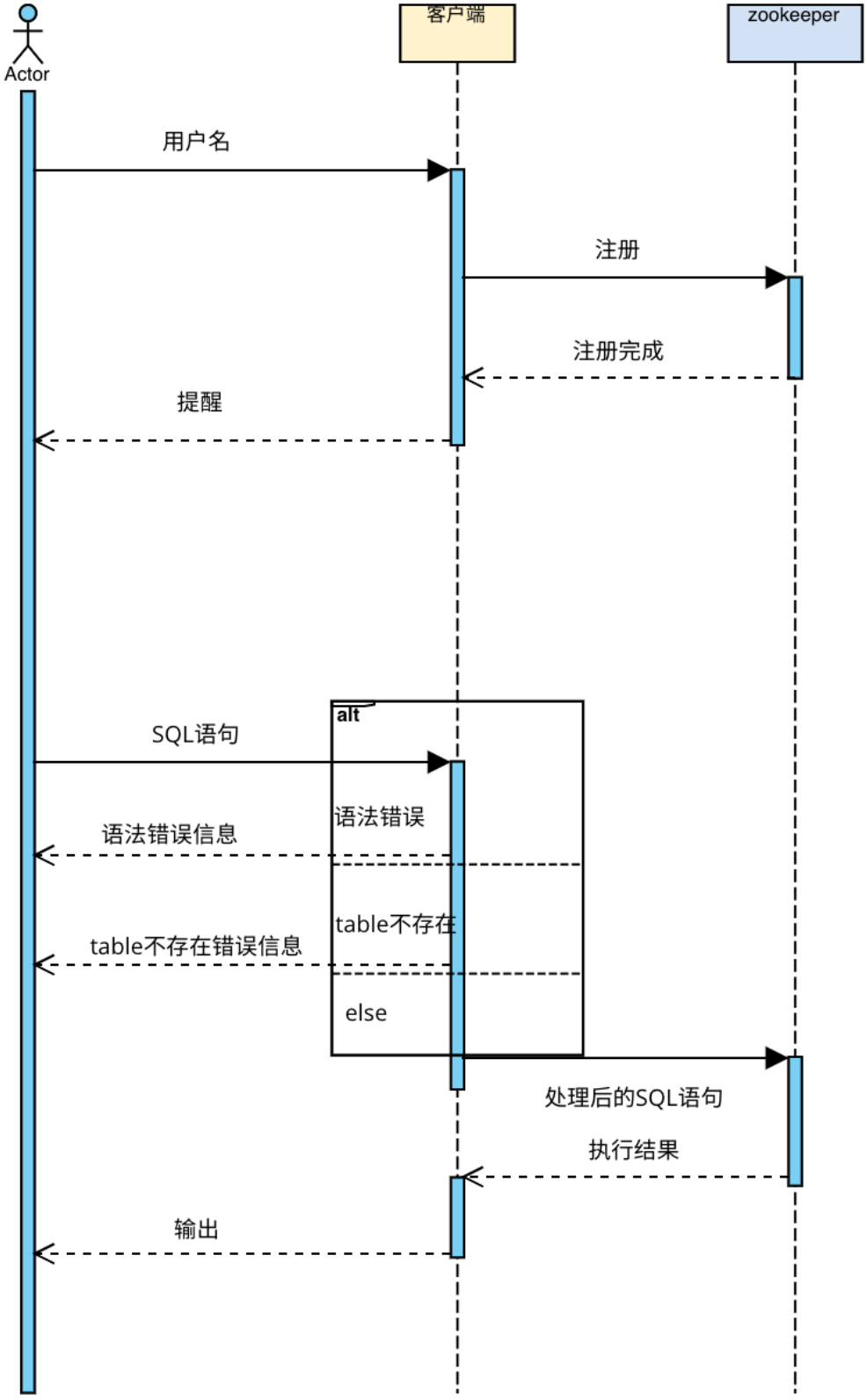
首先判断是单表查询或多表查询。
如果是单表查询：处理同上：根据 tableRegionMap 查询得知需要查询操作的表所在的 Region 服务器。直接向对应服务器节点的缓冲区写入用户名和该 SQL 语句，由 Region 负责执行。
如果是多表查询：根据 tableRegionMap 判断需要查询的表是否在同一个 region 服务器上。
如果需要查询的表都在同一个 region 服务器上。仅向对应服务器节点的缓冲区写

入用户名和该 SQL 语句，由该 Region 负责执行。否则，将需要查询的表和用户名、SQL 语句写入 Master 的缓冲区，由 Master 负责执行并返回查询结果。

4. 输出执行结果

在 SQL 语句有问题（语法错误，操作的 Table 不存在等）。客户端将直接打印相关的报错。SQL 语句执行完毕后，将向客户端节点的缓冲区中写入执行结果，监听到变化后直接打印结果。如果是 SELECT 语句，还需要逐行打印查询结果。

客户端服务的时序图如下：

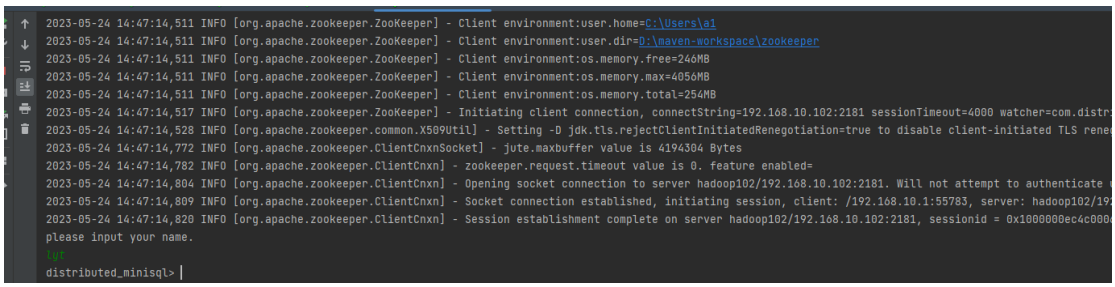

Actor
客户端
zookeeper
用户名
注册
注册完成
提醒
alt
SQL语句
语法错误
语法错误信息
table不存在
table不存在错误信息
else
处理后的SQL语句
执行结果
输出

五、 单元测试

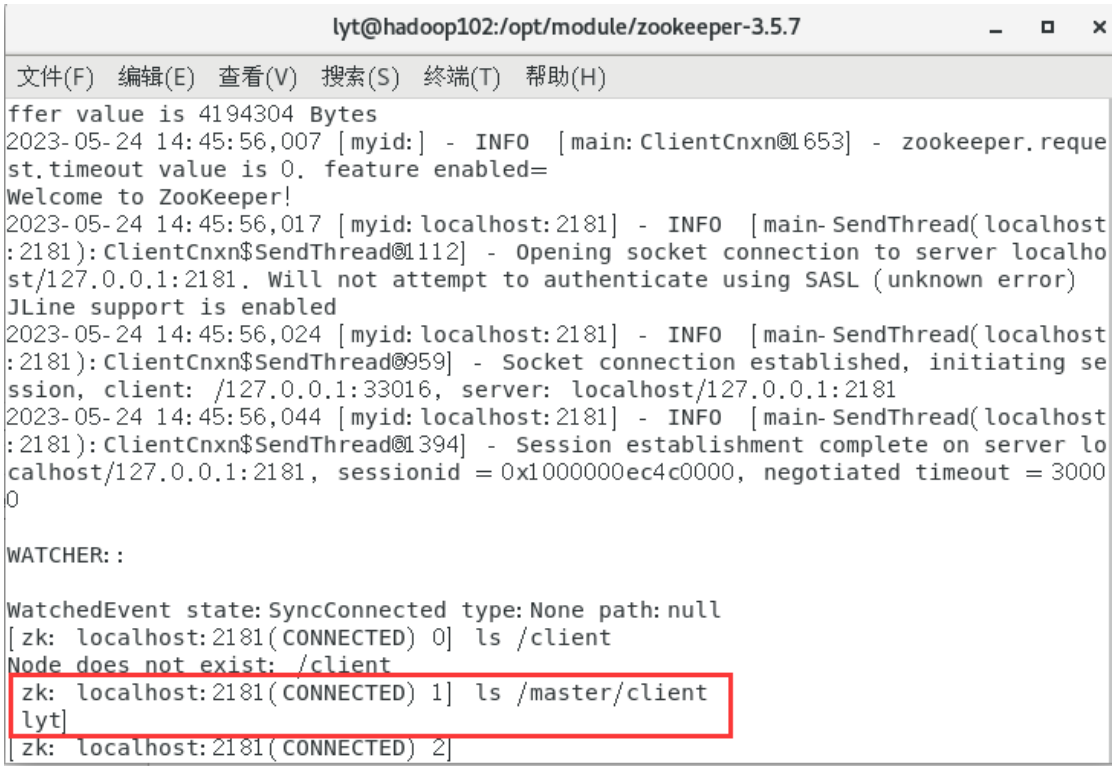
5.1 用户注册

注册新用户名

输入：打开客户端后，输入新的用户名
预期输出：客户端提示注册成功，zookeeper 中对应节点被创建
实际输出：
客户端提示：显示 sql 语句输入的提示符



Zookeeper 新增名称为用户名的节点



是否通过：通过

注册已存在的用户名

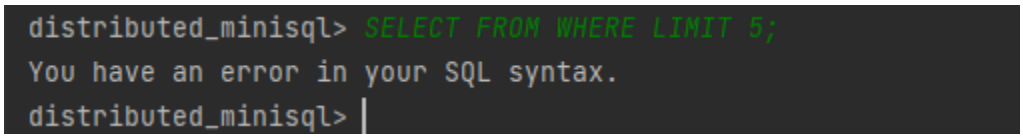
输入：打开客户端后，输入已被注册的用户名
预期输出：客户端提示用户名已存在
实际输出：
客户端提示用户名已存在

是否通过：通过

5.2 SQL 语句异常处理

SQL 语句有语法错误

输入：有语法错误的 SQL 语句
预期输出：客户端提示语法错误
实际输入：“SELECT FROM WHERE LIMIT 5;”
实际输出：
客户端提示语法错误



是否通过：通过

待创建的表已存在

输入：CRETE TABLE 操作的 SQL 语句，名称为语句中表名的数据表已存在
预期输出：客户端提示该表已存在
测试输入：“id_job”表已存在后，输入“CREATE TABLE id_job;”
实际输出：
客户端提示表已存在

```
You have an error in your SQL syntax.
distributed_minisql> show tables;
id_age
id_password
id_job
id_name
distributed_minisql> CREATE TABLE id_job;
Table id_job already exists.
```

是否通过：通过

查询/更改/删除的表不存在

输入：DROP TABLE，INSERT，UPDATE，DELETE，SELECT 操作的 SQL 语句中，名称为语句中表名的数据表之一不存在。
预期输出：客户端提示数据表不存在
测试输入：“student”表不存在时，分别输入
“DROP TABLE student;”
“INSERT INTO student VALUES (1234, ‘Alice’);”
“UPDATE student SET id=1233 WHERE id=1234;”
“DELETE FROM student WHERE id=1234;”
“SELECT * FROM student;”
“SELECT * FROM student, id_job;”
实际输出：
客户端提示“student”表不存在

```
distributed_minisql> show tables
                  -> ;
id_age
id_password
id_job
id_name
distributed_minisql> DROP TABLE student;
Table student doesn't exist.
```

```
distributed_minisql> INSERT INTO student VALUES (1234, 'Alice');
Table student doesn't exist.
distributed_minisql> UPDATE student SET id=1233 WHERE id=1234;
Table student doesn't exist.
distributed_minisql> DELETE FROM student WHERE id=1234;
Table student doesn't exist.
distributed_minisql> SELECT * FROM student;
Table student doesn't exist.
distributed_minisql> SELECT * FROM student, id_job;
Table student doesn't exist.
```

是否通过：通过

5.3 SQL 语句发送

注：本节测试经过初步检查（5.2 中异常检测）无异常的 SQL 语句能否被正确发送至期望的服务器。

建立/删除数据表

输入：CREATE, DROP 类型的 SQL 语句
预期输出：客户端将带有客户端用户名、SQL 语句的消息写入 mastersql 节点缓冲区。
实际输入：输入
“CREATE TABLE students (id INTEGER, name VARCHAR(255));”
“DROP TABLE students;”
实际输出：
每当输入一条 SQL 语句后，mastersql 节点中被写入包含用户名和 SQL 语句的信息

```
distributed_minisql> CREATE TABLE students (id INTEGER, name VARCHAR(255));
execute success!
distributed_minisql>
```

```
[zk: localhost:2181(CONNECTED) 8] get /master/masterSql
students##client/lyt##CREATE_TABLE##CREATE TABLE students (id INTEGER, name VARCHAR(255));
```

```
distributed_minisql> DROP TABLE students;
execute success!
```

```
[zk: localhost:2181(CONNECTED) 9] get /master/masterSql
students##client/lyt##DROP_TABLE##DROP TABLE students;
```

是否通过：通过

更改数据表

输入：INSERT，UPDATE，DELETE 类型的 SQL 语句
预期输出：客户端将带有客户端用户名、SQL 语句的消息写入 SQL 语句操作的数据表所在 Region 服务器对应的的 zookeeper 节点的缓冲区。
实际输入：输入
“INSERT INTO student VALUES (1234, ‘Alice’);”
“UPDATE student SET id=1233 WHERE id=1234;”
“DELETE FROM student WHERE id=1234;”
实际输出：
输入第一条 SQL 语句后，student 表所在的 Region 服务器对应的 zookeeper 节点中被写入包含用户名和 SQL 语句的信息

```
distributed_minisql> INSERT INTO student VALUES (1234, 'Alice');
execute success!1 rows of table are affected.
```

```
[zk: localhost:2181(CONNECTED) 11] get /master/region2
client/lyt##INSERT##INSERT INTO student VALUES (1234, 'Alice');
```

```
client/lyt##INSERT##INSERT INTO student VALUES (1234, 'Alice');
[zk: localhost:2181(CONNECTED) 12] get /master/region2
client/lyt##UPDATE##UPDATE student SET id=1233 WHERE id=1234;
```

```
[zk: localhost:2181(CONNECTED) 13] get /master/region2
client/lyt##DELETE##DELETE FROM student WHERE id=1234;
```

是否通过：通过

单表查询

输入：仅查询一张数据表的 SQL 语句
预期输出：客户端将带有客户端用户名、SQL 语句的消息写入 S 待查询所在 Region 服务器对应的的 zookeeper 节点的缓冲区。
实际输入： “SELECT * FROM student;”
实际输出：
student 表所在的 Region 服务器对应的 zookeeper 节点中被写入包含用户名和 SQL 语句的信息。

```
distributed_minisql> SELECT * FROM student;
Query Result:
name            id
Alice           1233
```

```
[zk: localhost:2181(CONNECTED) 14] get /master/region2
client/lyt##SELECT##SELECT * FROM student;
```

是否通过：通过

多表查询

输入：仅查询多张数据表的 SQL 语句
预期输出：如果查询的数据表均在同一个 Region 服务器上，将带有客户端用户名、SQL 语句的消息写入其所在 Region 服务器对应的的 zookeeper 节点的缓冲区。否则，将用户名和 SQL 语句的信息写入 mastersql 节点的缓冲区。
实际输入：
假设表，id_job 在 Region1 上，student, id_age 在 Region2 上，输入
“SELECT * FROM student, id_age;”
“SELECT * FROM student, id_job;”
实际输出：
输入第一条 SQL 语句时，Region2 对应的节点缓冲区写入用户名和 SQL 语句。

```
distributed_minisql> SELECT * FROM student, id_age;
Query Result:
name            id_age.id       id              age
Alice           1               1233            22
Alice           2               1233            22
```

```
[zk: localhost:2181(CONNECTED) 15] get /master/region2
client/lyt##SELECT##SELECT * FROM student, id_age;
```

输入第二条 SQL 语句时，mastersql 节点缓冲区写入用户名和 SQL 语句。

```
distributed_minisql> SELECT * FROM student, id_job;
Query Result:
id_job.id       name            id              job
1               Alice           1233            student
```

```
[zk: localhost:2181(CONNECTED) 16] get /master/masterSql
student,id_job##client/lyt##SELECT##SELECT * FROM student, id_job;
```

是否通过：通过

结果显示

输入：Region 或 Master 完成数据库操作后，将结果写入客户端对应的节点
预期输出：结果显示在命令行界面
实际输入：使用 zookeeper 命令行在客户端节点中写入“Successfully executed”
实际输出：命令行显示“Successfully executed”

```
[zk: localhost:2181(CONNECTED) 2] set /master/client/lyt "Successfully executed"
```

```
distributed_minisql> Successfully executed
```

是否通过：通过

六、开发心得

在开发中，我体会到了客户端设计的难度所在。需要仔细设计接收用户输入并显示输出的逻辑，并进行全方面的测试确保模块的可靠性。

在进行小组合作的时候，我体会到了模块之间解耦并制定统一的信息传输格式的重要性。必须完整地考虑到需要传输的各种信息，才能避免开发过程中格式的更改带来的额外工作量。

此外，在设计客户端模块的时候，也必须参与其他模块的设计并熟悉相关的函数，调用时才能做到得心应手。

总之，这个项目让我了解并掌握了 zookeeper 的使用和编程，并增强了系统设计能力和沟通合作能力。