

**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： |  |
| 学 院： | 计算机学院 |
| 系： | 软件工程系 |
| 专 业： | 软件工程 |
| 学 号： |  |
| 指导教师： | 鲍凌峰 |

2023年 5 月 5 日

目录

[1引言 2](#_Toc135824241)

[1.1 系统目标 2](#_Toc135824242)

[1.2 时间安排 2](#_Toc135824243)

[2 设计分工 3](#_Toc135824244)

[3 总体设计 3](#_Toc135824245)

[3.1 技术字典 3](#_Toc135824246)

[3.2 总体架构设计 4](#_Toc135824247)

[3.3 业务流程设计 5](#_Toc135824248)

[3.4 技术重点设计 7](#_Toc135824249)

[4. 模块详细设计 8](#_Toc135824250)

[4.1 Client模块 8](#_Toc135824251)

[4.2 Master模块 8](#_Toc135824252)

[4.3 Server模块 8](#_Toc135824253)

[4.4 其他模块 8](#_Toc135824254)

[5. 功能运行测试 8](#_Toc135824255)

[5.1 分布式模块测试 8](#_Toc135824256)

[5.2 数据库基本功能测试 9](#_Toc135824257)

[5.3 负载均衡与主从复制 11](#_Toc135824258)

[5.4 容错容灾 13](#_Toc135824259)

[5.5 master数据表缓存测试 15](#_Toc135824260)

**浙江大学实验报告**

实验名称： 分布式MiniSQL 实验类型： 设计实验

同组学生： 米博宇、李毅桐、王粤龙 实验地点：

## 1引言

本实验是浙江大学《大规模信息系统构建技术导论》课程项目。实验要求开发一个分布式的MiniSQL数据库。本组选择使用Java语言实现，使用zookeeper接口实现分布式服务器的部分管理功能。

### 1.1 系统目标

本应用的目标是设计并实现一个分布式数据库引擎MiniSQL，允许用户通过命令行界面输入SQL语句进行数据表的新建、删除，记录的更新、删除、查询。相比MiniSQL，还需要实现多表join操作、数据分布、集群管理和分布式查询，并且系统需要具有容错容灾、副本复制、负载均衡功能。

### 1.2 时间安排

2023.03.22 – 2023.04.08项目前期准备

2023.04.09 – 2023.04.14 项目整体架构设计、模块分工

2023.04.15 – 2023.04.23 各模块详细设计

2023.04.24 – 2023.05.03 zookeeper相关的操作函数完成，各模块逻辑完成，接口和API设计完成

2023.05.04 – 2023.05.12 各模块最终完成

2023.05.13 – 2023.05.19 项目整合和集成测试

2023.05.20 – 2023.05.23 项目文档编写、录制视频并提交

## 2 设计分工

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 学号 | 分工 | 备注 |
| 李毅桐 | 3200105851 | 实现Master，设计收发包的格式。负责使用zookeeper实现消息传递 | 组长 |
| 米博宇 | 3200102888 | 系统详细设计、客户端主要函数的编写和实现Db Operations | 组员 |
| 王粤龙 | 3200105512 | 实现region server部分,制作ppt | 组员 |

## 3 总体设计

### 3.1 技术字典

Zookeeper：

ZooKeeper是一个分布式的，开放源码的分布式应用程序协同服务。ZooKeeper的设计目标是将那些复杂且容易出错的分布式一致性服务封装起来，构成一个高效可靠的原语集，并以一系列简单易用的接口提供给用户使用。ZooKeeper可以用于发布/订阅、负载均衡、命令服务、分布式协调/通知、集群管理、Master选举、分布式锁和分布式队列等功能。

Master：

Master是专门用于管理region的服务器，负责管理和维护表的分区信息（或者分布信息)等元数据信息、维护Region服务器列表、分配Region、实现不同Region服务器之间的负载均衡管理用户对表的增加、删除、修改、查询等操作、对发生故障失效的Region服务器上的Region进行迁移等功能

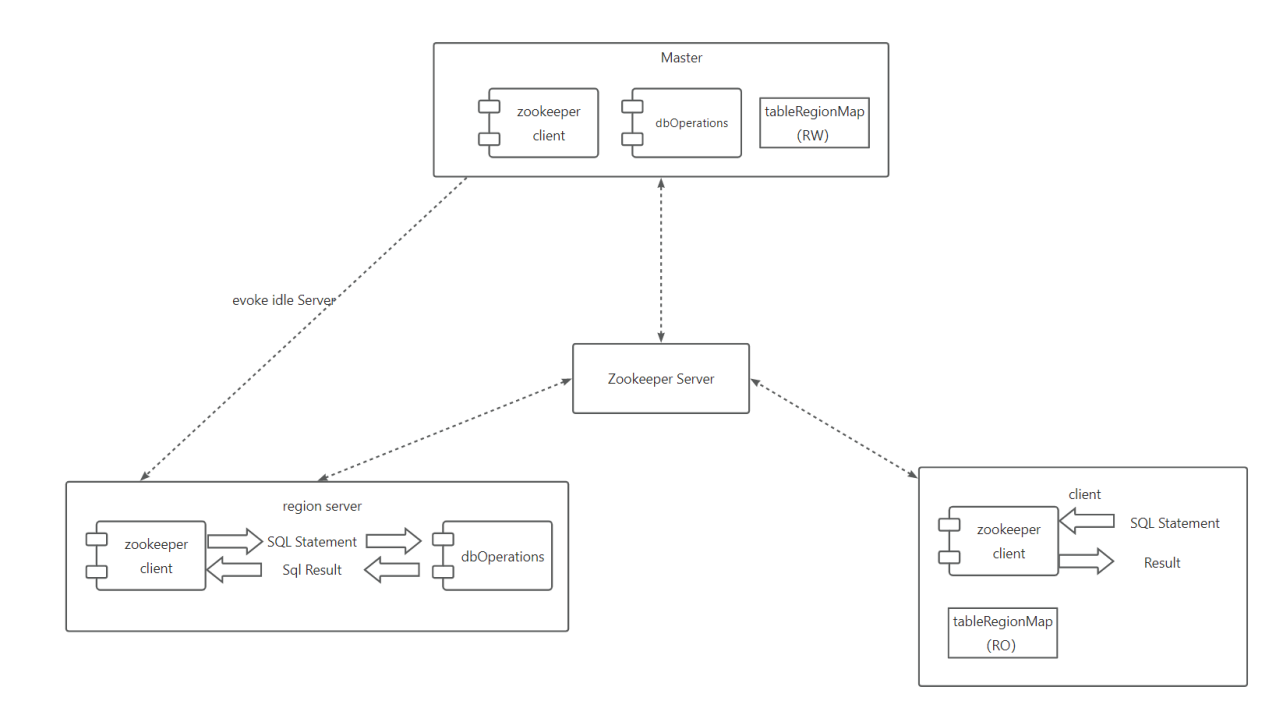
Region：

本项目的一个region逻辑上表示一个数据库，它由一个主服务器和一个副服务器组成，主服务器和副服务器都会连接一个直接的数据库。同时服务器数据库的数据和主服务器的一致。

### 3.2 总体架构设计

本程序的主要功能有：

1. 用户输入SQL语句，系统对其逻辑进行解析，返回SQL语句执行结果
2. Master根据各个region表的数量，决定下一个表应该建在哪一个region实现负载均衡
3. Master 当某个region的服务器下线时，Master监听获得目标region，唤醒一个闲置的服务器到该region，同时复制该region的数据库，实现容错容灾。



由上述架构图可以看出，本应用整体分为Client，Master，region Server，Zookeeper Server四个部分。其中Zookeeper是开着Zookeeper集群服务的一台或者多台物理机。Client，Master和region Server基本上都是通过Zookeeper集群来实现消息的传递。这样Client只需要知道Zookeeper集群的IP地址就可以实现与其他数据库进行交互。

* Client与用户进行直接交互。用户输入SQL语句，Client通过修改特定节点的值将SQL语句传递给Server或者Master，通过监听特定节点收到SQL语句执行结果，最终将结果打印给用户。
* Server的功能是监听特定节点，将得到的SQL语句交给数据库进行执行，返回SQL语句执行结果并写入目标节点。同时Server辅助Master实现容错容灾。
* Master的功能是对各个Region的table进行管理，实现负载均衡；监听服务器上下线，实现容错容灾的功能。建立临时数据库，辅助复杂查询。

#### 3.2.1 Master 模块

由于Master拥有tableRegionMap的读写权利，所以对应创建表和删除表这两个比较特殊的SQL操作，需要由Client先发给Master，然后Master再发给特定的region server执行。

同时当涉及到多个region多个表格的复杂查询时，需要由Master作为中介进行复杂查询，然后将结果返回给Client。

当监听到某个region的服务器下线时，需要由master手动唤起唤醒一个闲置的服务器到该region。

#### 3.2.2 region server 模块

存储拥有本地数据库，监听对应的zookeeper节点，执行来自master和client的sql语句，并返回对应的结果。同时当同一region的服务器下线时，需要给Master发送消息，请求Master唤醒新的服务器。

#### 3.3.3 client模块

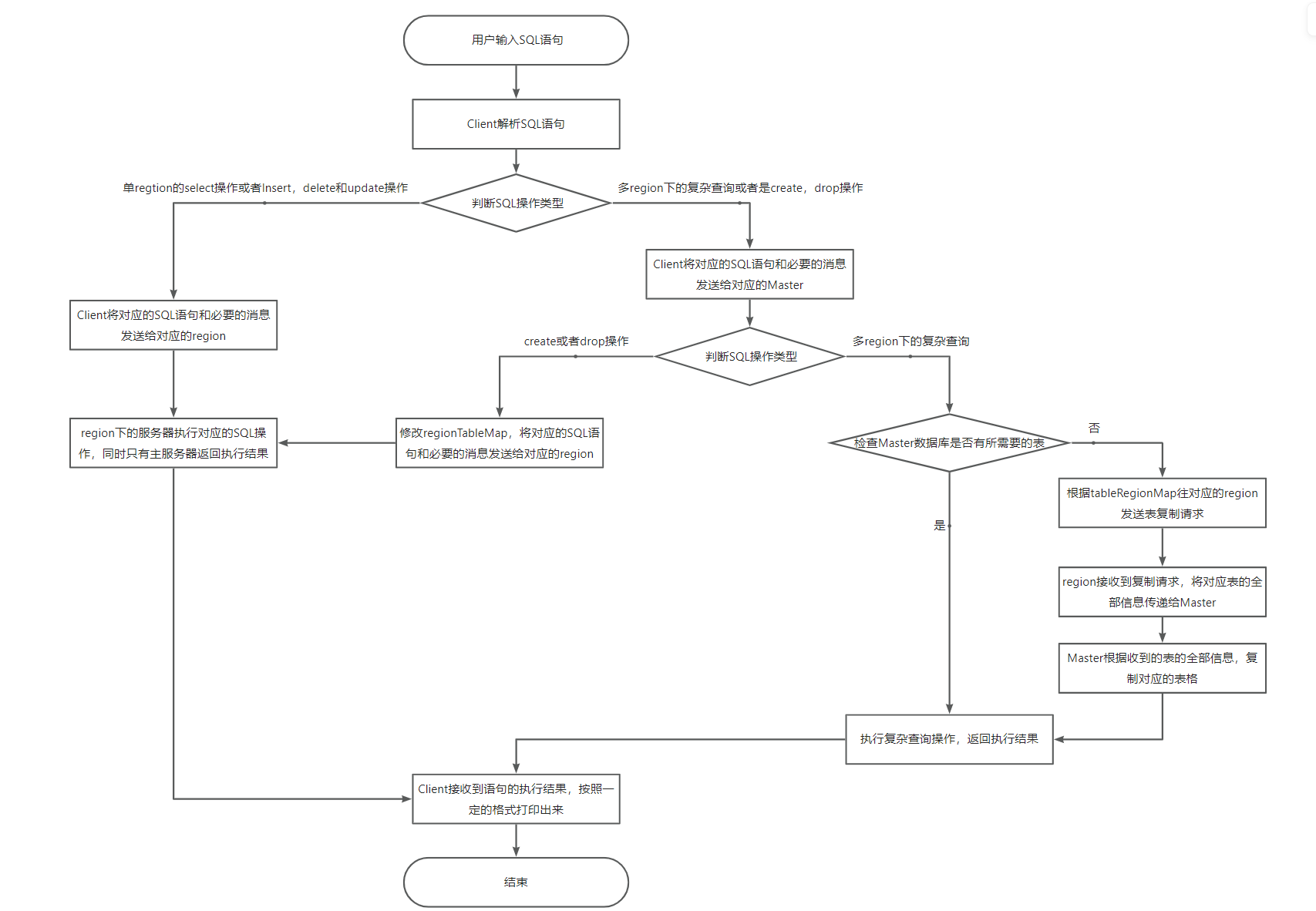
用户直接操作此模块，输入SQL语句，同时client根据读取到的tableRegionMap将SQL语句传递到对应的节点。同时简单执行结果的节点，以获得SQL语句的执行结果，并将其打印给用户。

#### 3.3.4 dboperations模块

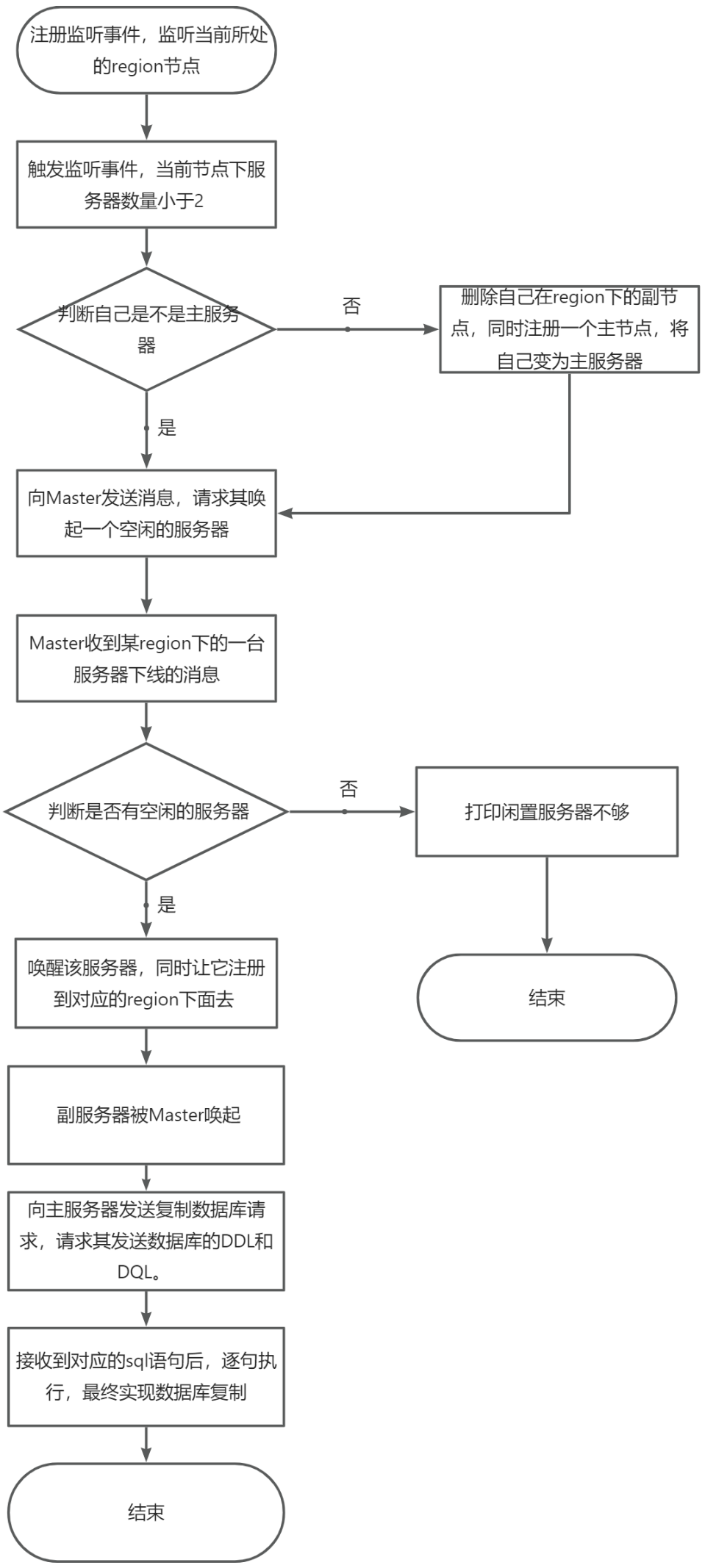
集成封装了一些常用的SQL操作，方便其他模块的使用。

### 3.3 业务流程设计

一次SQL语句的执行流程



容错容灾的实现方法：



### 3.4 技术重点设计

#### 3.4.1 机器情况

* 上述架构中每个模块，其实可以看作一个进程，也就是说所有的模块实际上可以在同一台物理机上，只需要开多个进程来进行模拟即可（这是一种解决方案）
* 由于至少需要两个region，每个region需要一台主服务器和一台副服务器，所以一共需要4台region server。同时需要一台物理机做为Master, 一台物理机做为Client。

#### 3.4.2 数据通信

* 由于zookeeper的监听机制，所以使用zookeeper来作为数据通信的桥梁，这样可以隐藏其他服务器的IP地址，无论是客户端还是服务器，只需要知道zookeeper集群的一个IP地址，就可以与其他服务器进行通信。当要发送消息就，发送方就往接收方监听的节点写入数据。
* 虽然zookeeper每个节点的数据上限只有1MB，但是可以通过修改zookeeper的配置文件来改变存储数据的大小

#### 3.4.3 副本维护与容错容灾

* 采用主从复制策略，选择一定量的机器作为副本机器，分别对应其他的主机。
* 在用户进行读写操作的时候，主机和副本机同时进行读写操作。
* 当某台服务器下线时，若其为副服务器，删除自己在region下的副节点，同时注册一个主节点，将自己变为主服务器，然后向master发送唤醒闲置服务器的请求，若是主服务器，则直接向master发送唤醒闲置服务器的请求。
* 空闲副本机刚刚上线时，会复制主服务器的数据库内容。

#### 3.4.5负载均衡

* 采用桶装均衡策略
* 假设在系统中各个region被访问的频率取决于它们容纳的表格数量
* 每次新增表格的时候，新增的表格都会被安排到当前拥有表格数量最小的服务器上。

## 4. 模块详细设计

### 4.1 Client模块

<客户端.docx>

### 4.2 Master模块

<Master模块和region服务端容错容灾模块.docx>

### 4.3 Server模块

<Master模块和region服务端容错容灾模块.docx>

<region服务端.docx>

### 4.4 其他模块

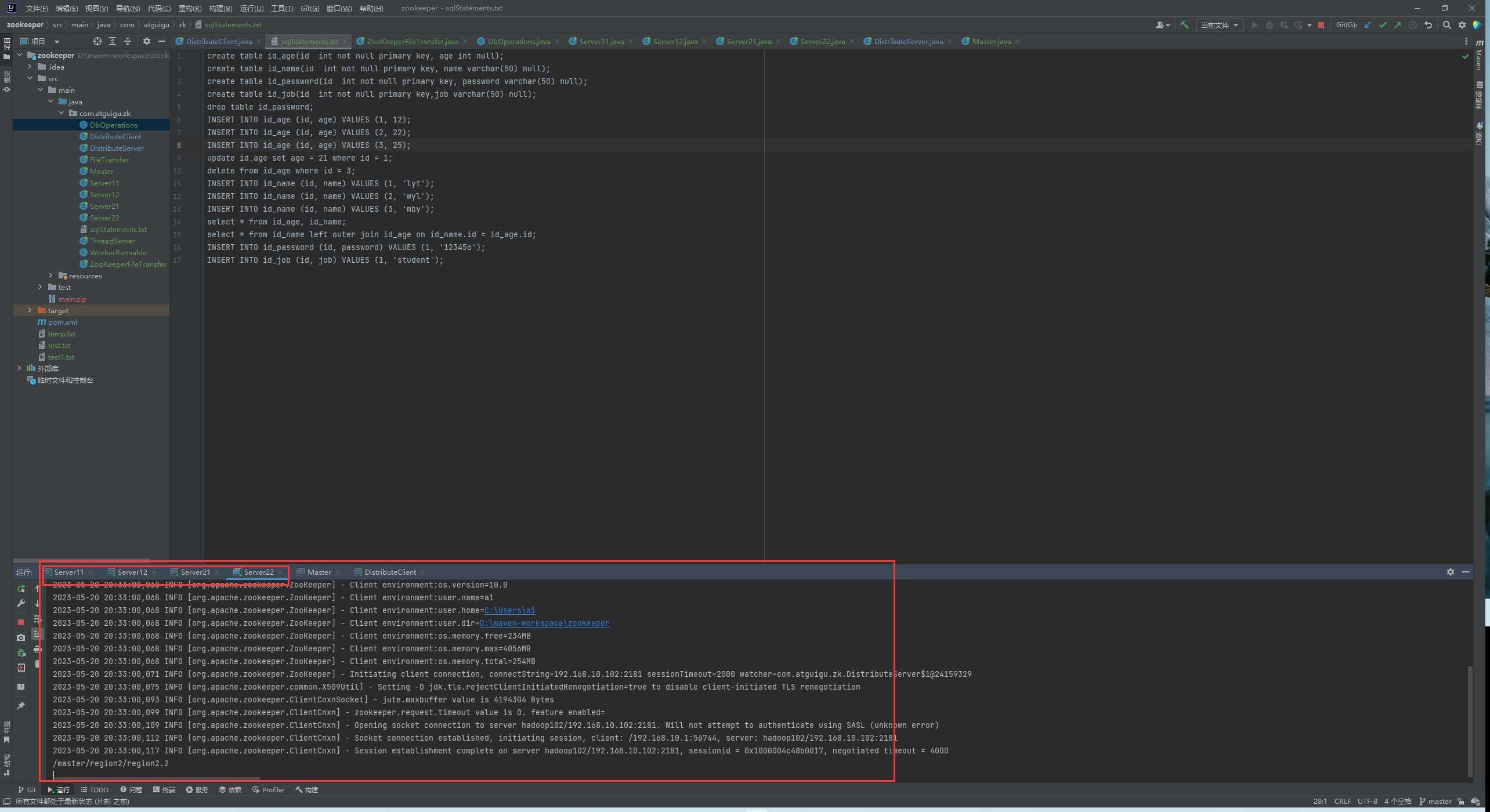
<其他模块.docx>

## 5. 功能运行测试

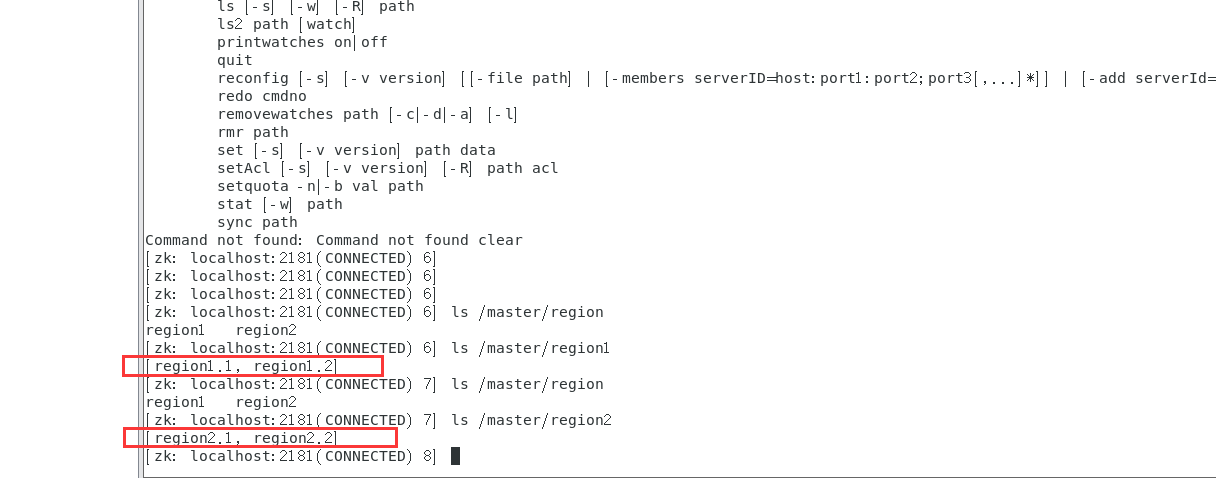
### 5.1 分布式模块测试

首先，由于我们组机器数量不够， 在测试时用进程来代替 Region Server，通过输入不同的region和serverName以及连接不同的数据库来构造。创建后在zookeeper集群下对应的region节点下注册一个自己的server节点。

打开4个server



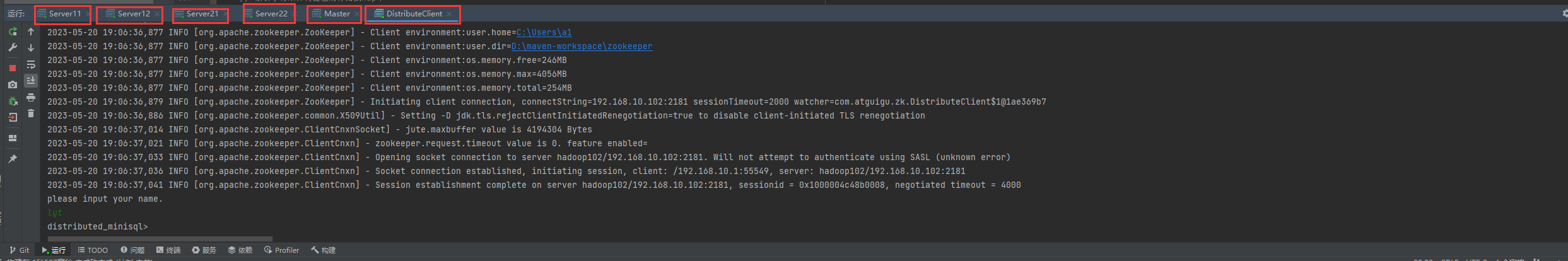
在zookeeper集群下注册对应的节点



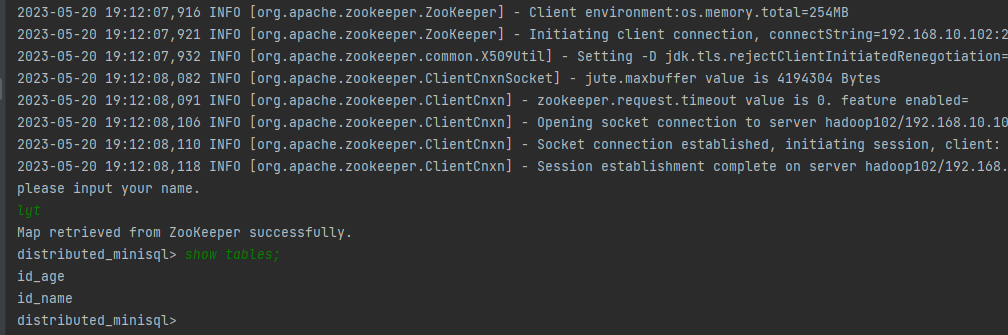
其中region1.1和region1.2是region下的服务器，region1.1是主服务器，同理region1.2是region的主服务器。他们各连接db1.1，db1.2，db2.1，db2.2。

### 5.2 数据库基本功能测试

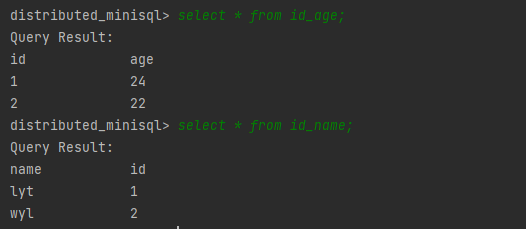
先打开zookeeper集群，然后依次打开Server11，Server12，Server21， Server22，Master，然后打开client，输入自己的客户名。



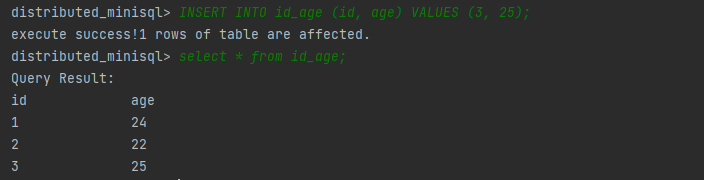
查询当前数据库有哪些表格。



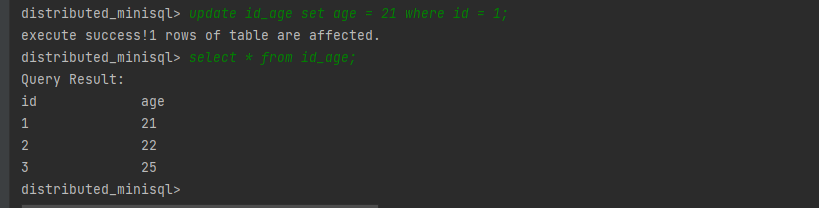
进行简单的查询



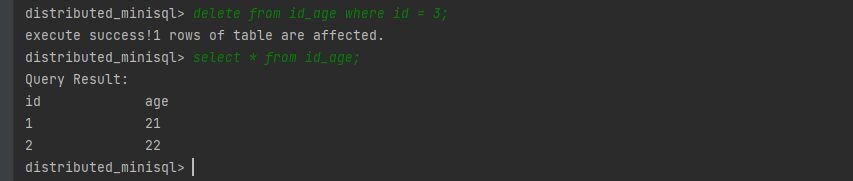
Insert 操作



Update操作

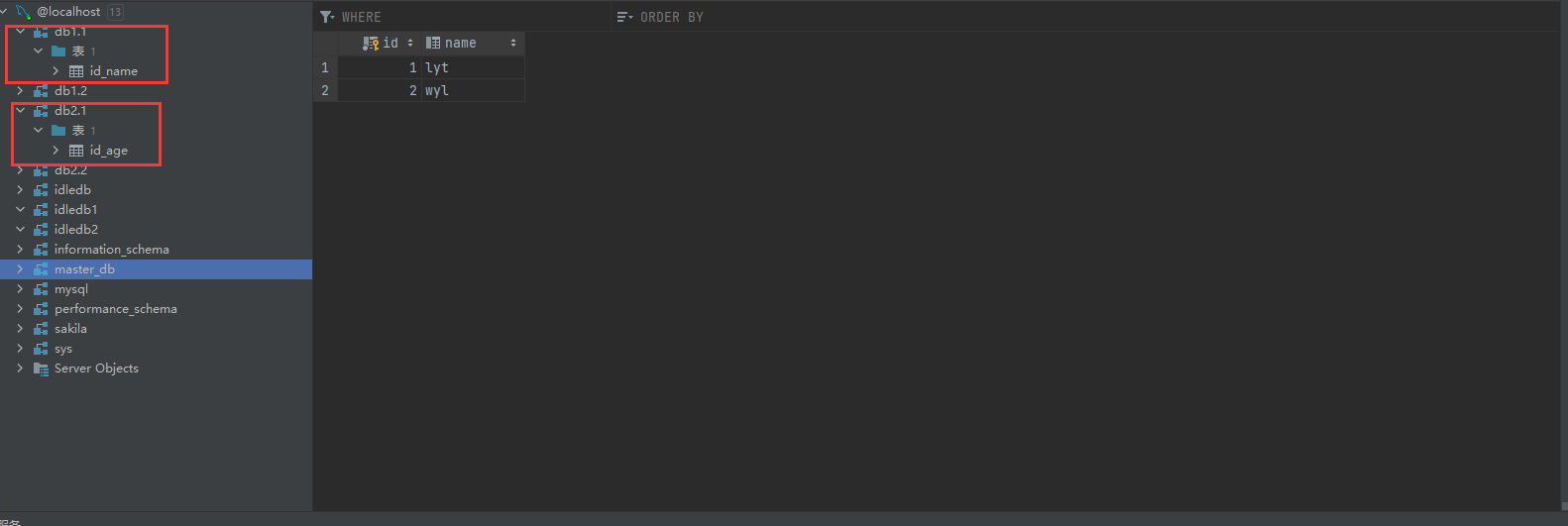


Delete操作

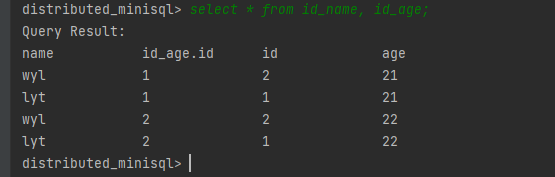


复杂查询

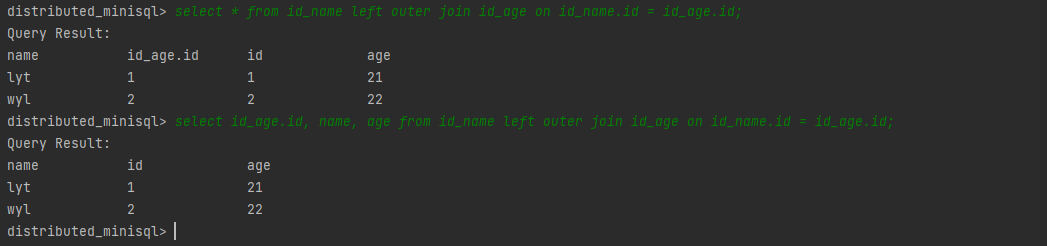
通过DataGrip可以看到id\_name和id\_age是存储在两个不同的数据库的，不能直接进行在一个数据库上进行查询。



最简单的复杂查询（两个表做笛卡尔积）。当有两个相同的列名时，前一个回

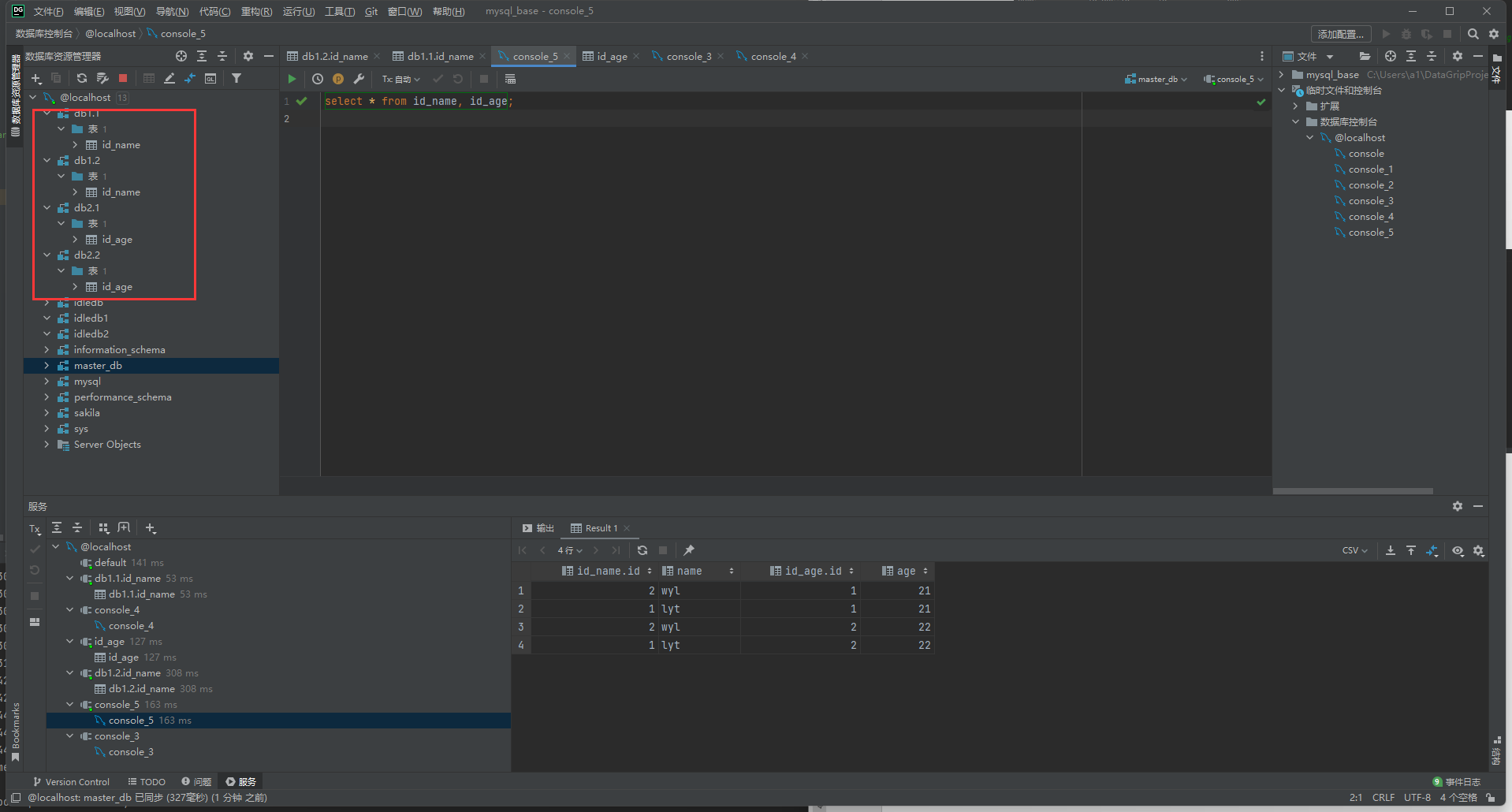


Join查询：

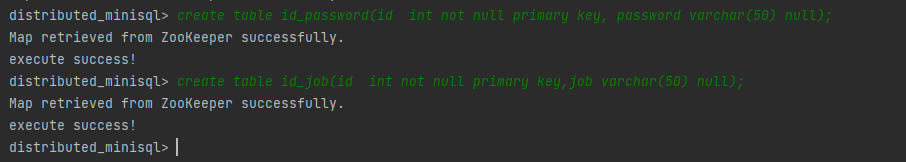


### 5.3 负载均衡与主从复制

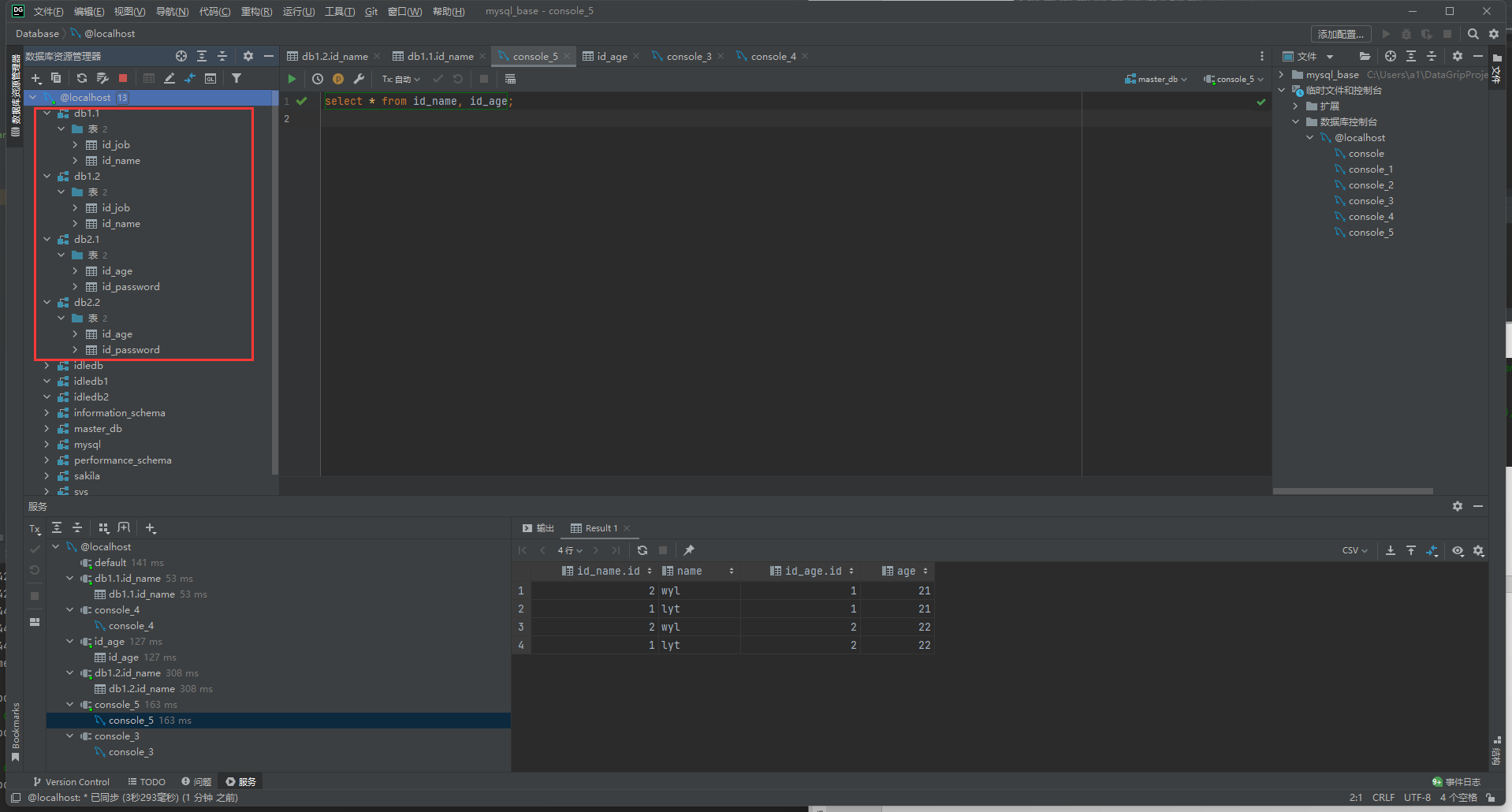
此时，region1和region2都只有一张表



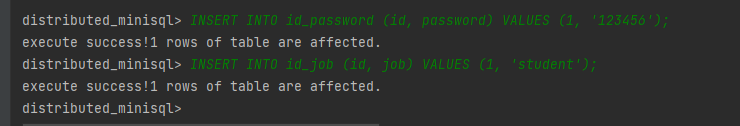
执行两次建表操作



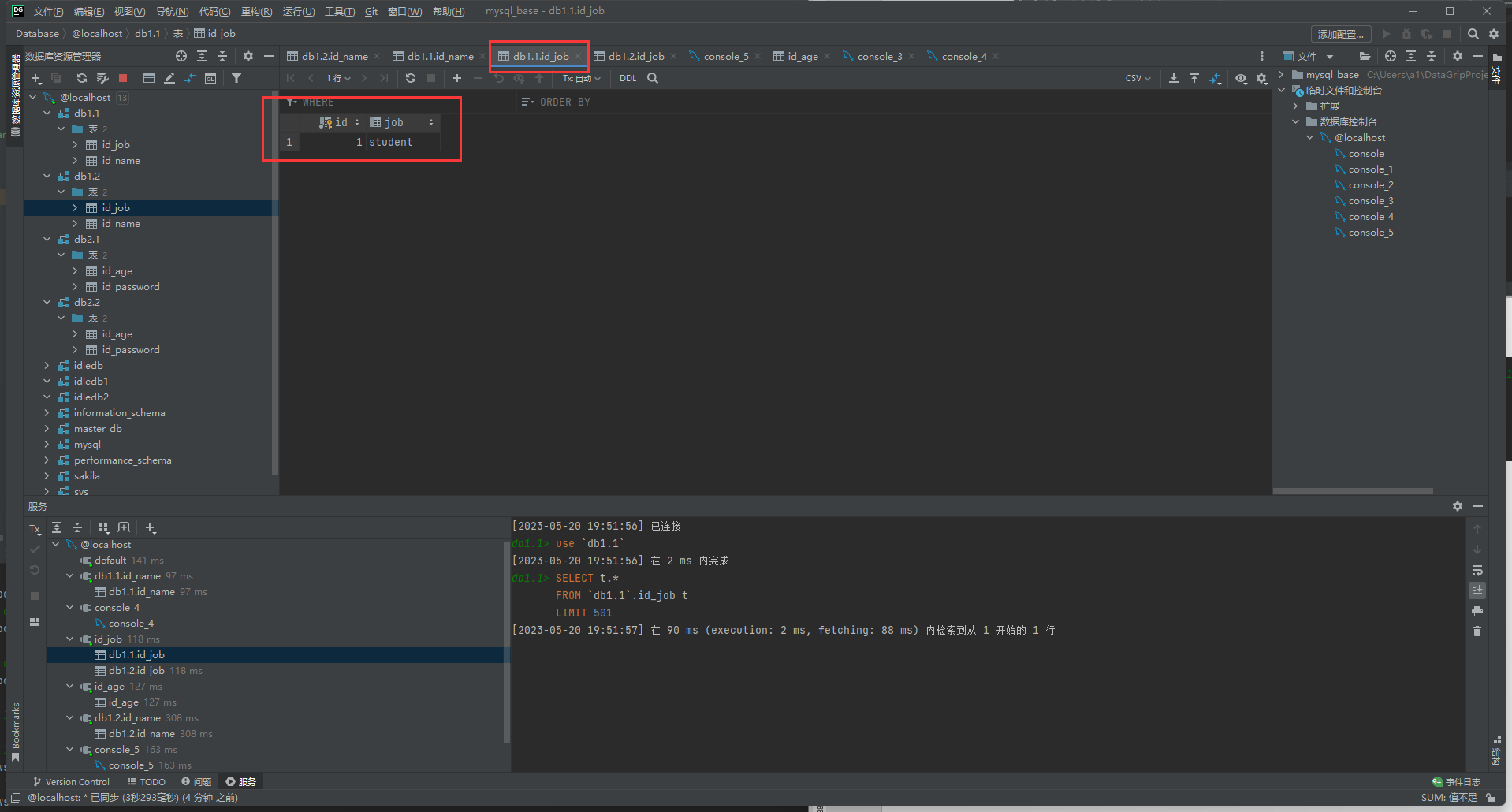
region1和region2都各建了一张表

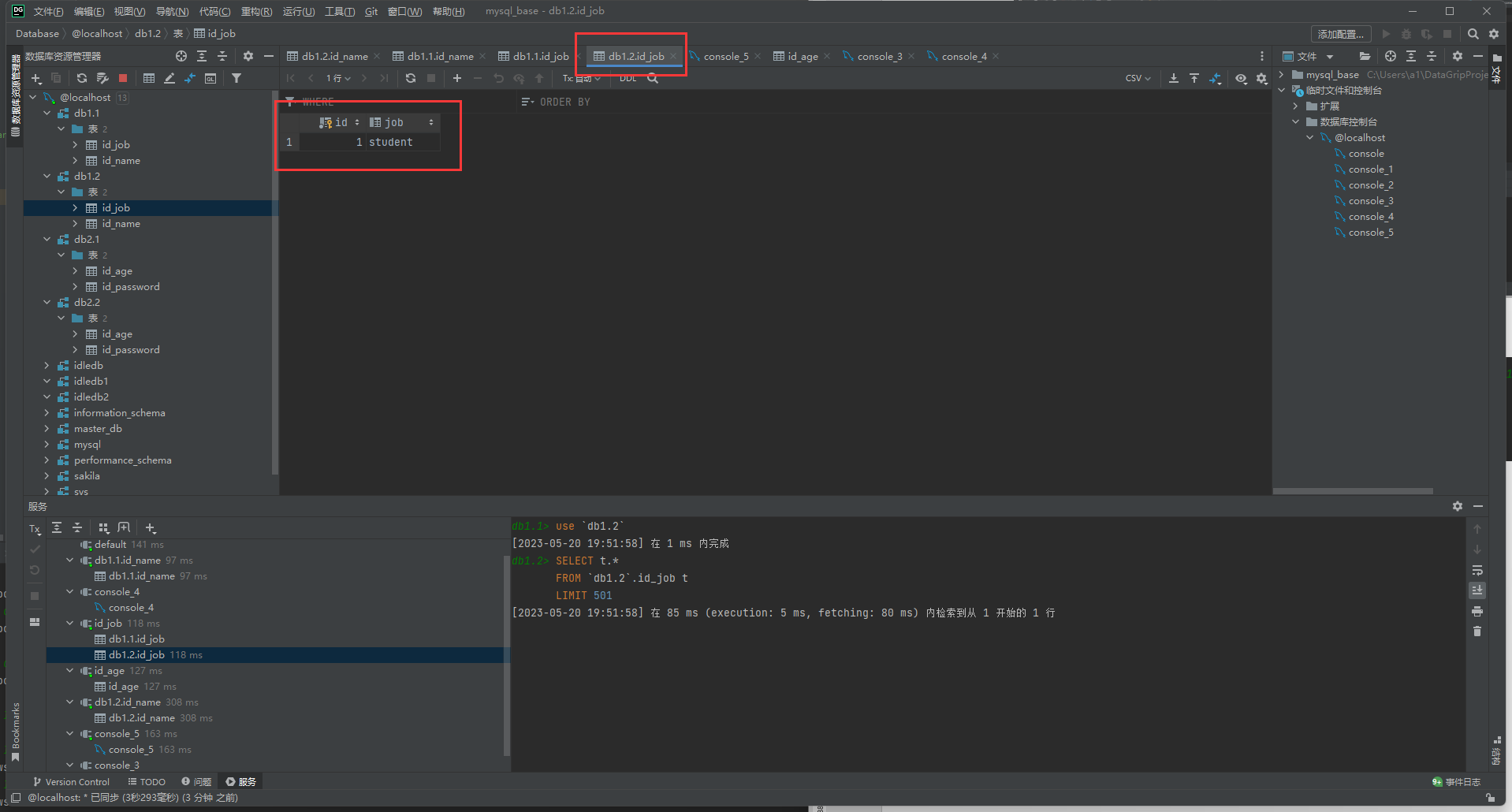


对新表进行插入操作



同一region下两个服务器都插入了相同的数据

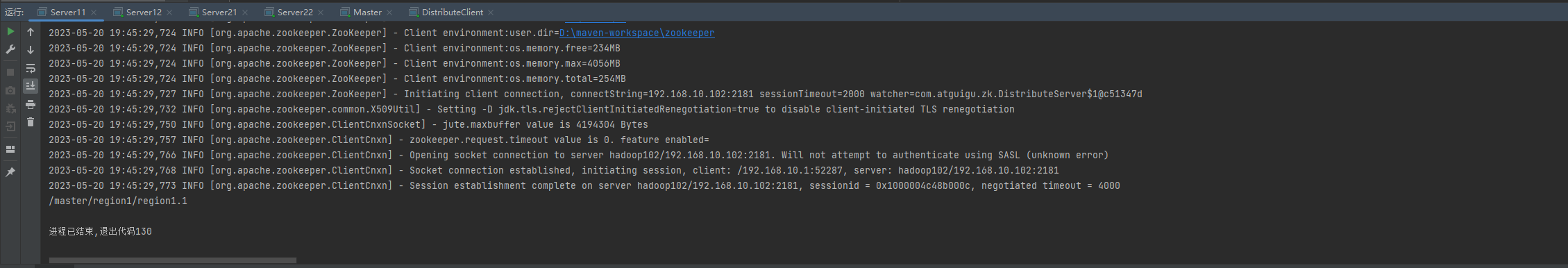




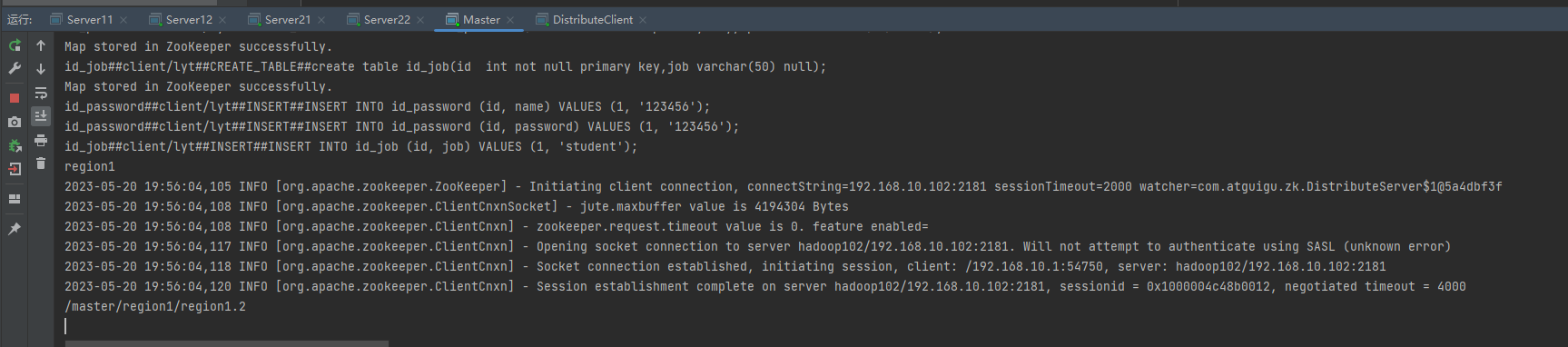
### 5.4 容错容灾

模拟服务器挂掉，master是否回唤起闲置的服务器，同时复制同一region下另外一台服务器的数据库。

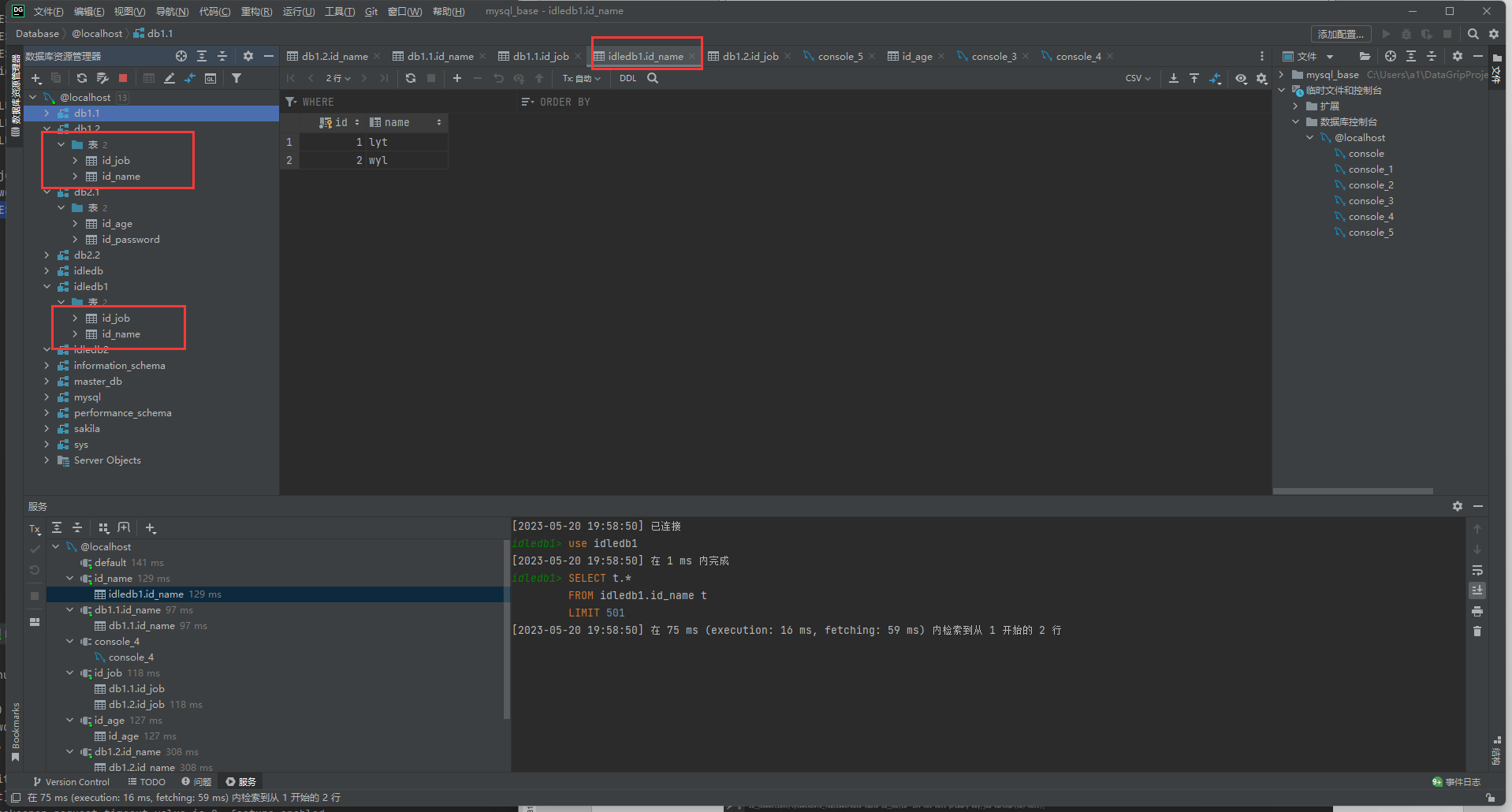
手动挂掉server11



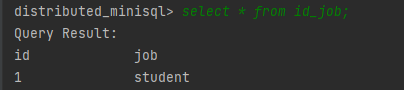
此时server12代替server11成为主服务器，同时master唤醒一个新的服务器，其连接的是闲置数据库idledb1

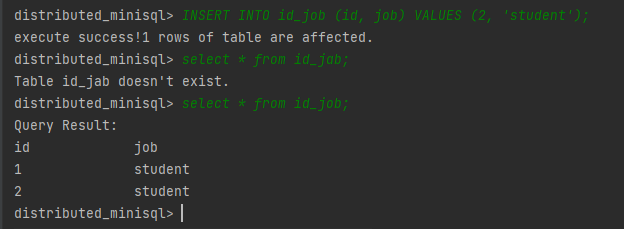


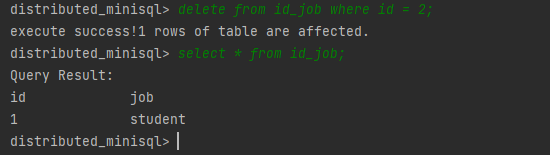
此时idledb1也成功复制了对应的数据库



同时基本是数据库功能也正常



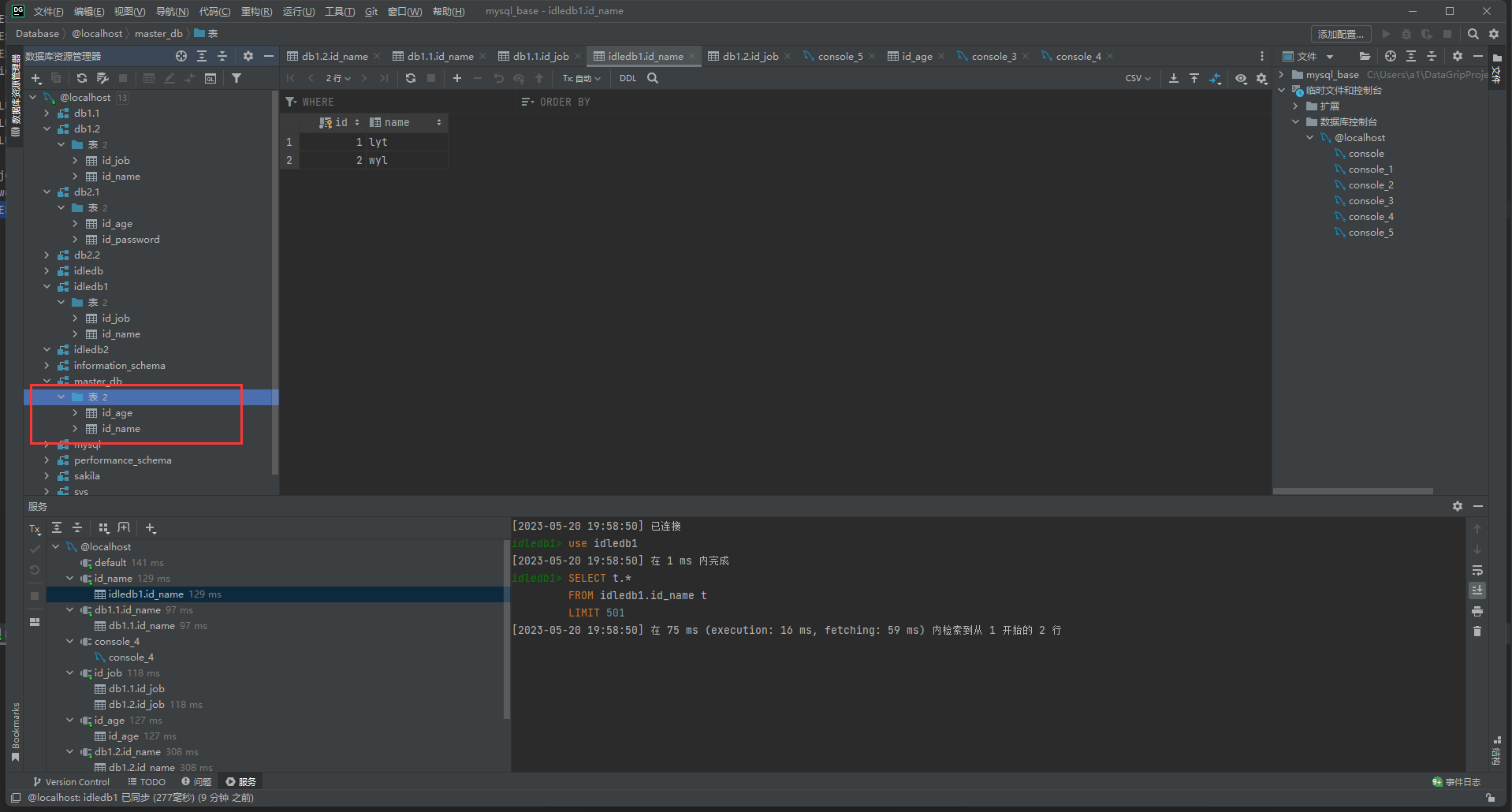




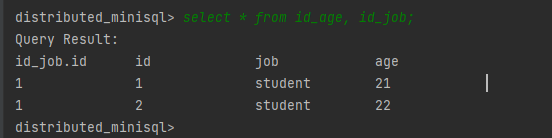
### 5.5 master数据表缓存测试

由于实现复杂查询时，其需要从不同的region克隆对应的表格到master的数据库，为了防止分布式数据库退化为普通数据库，所以设置了master的数据库存储表格上限。为了方便测试，设置的数据表上限为3.当表格超过数量限制时，采用LRU算法来删除表格。

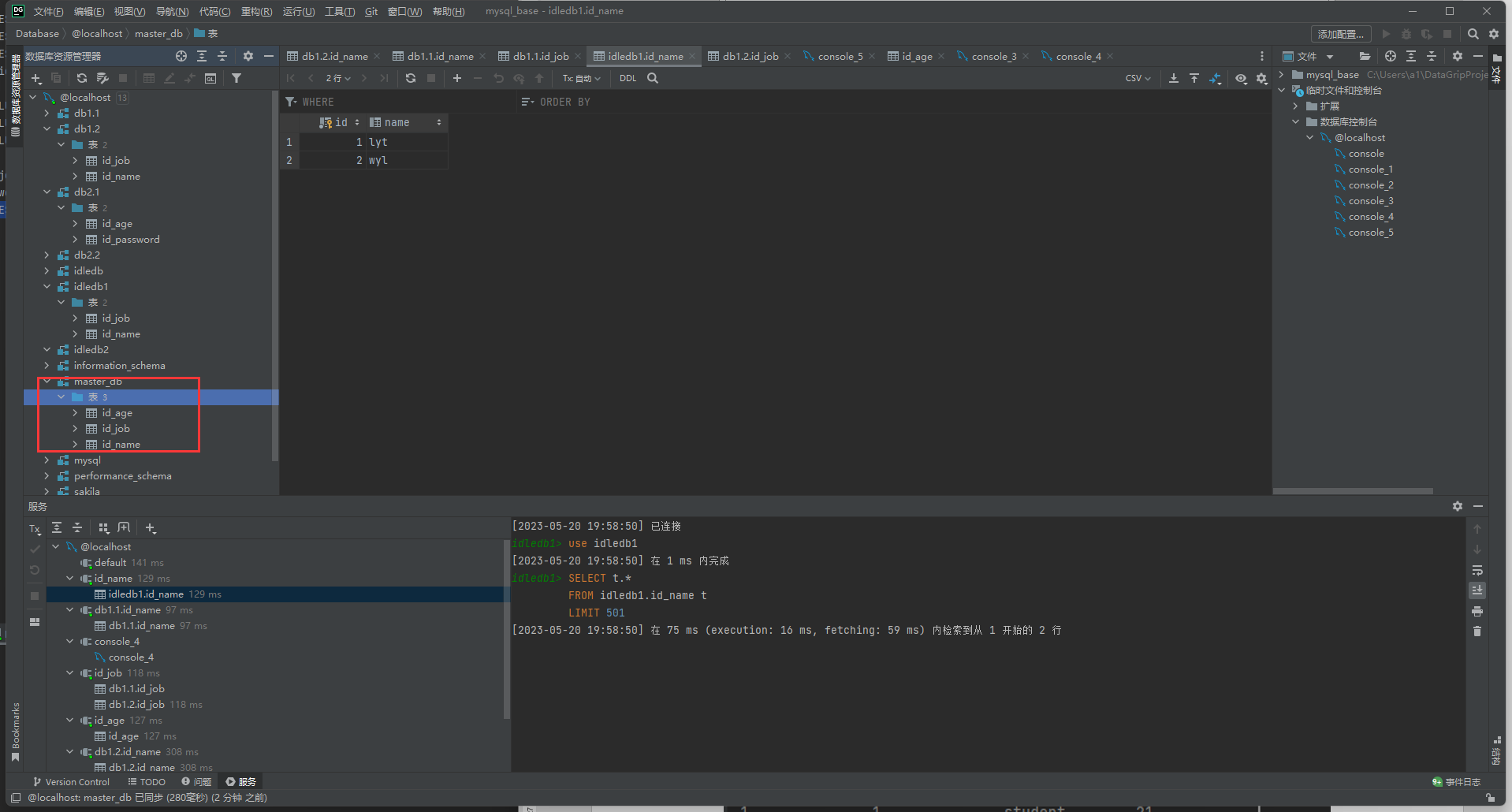
此时master表的数量为



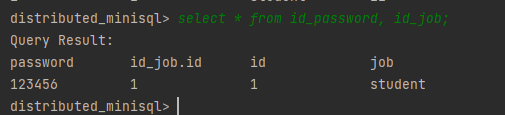
执行一次不同region下的复杂查询后

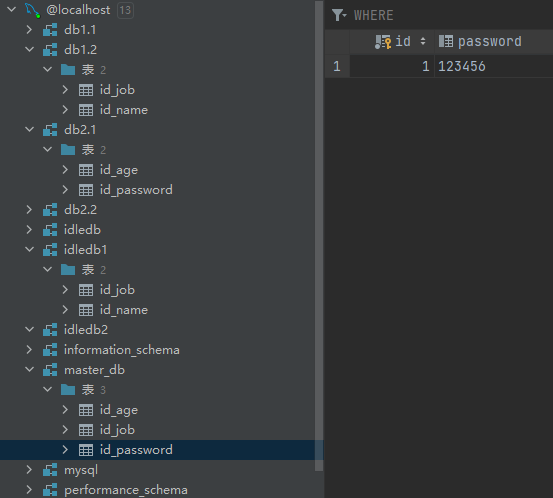


表格数量为3



再执行执行一次不同region下的复杂查询后，表格数量超过3，此时最久为被使用的表为id\_name，故自动删除此表





至此，全部模块功能全部测试完毕，测试结果符合预期。