2024-5-30

叶沐阳

3210104095

个人报告

大规模信息系统构建技术导论

目录

[1 个人开发过程与主要贡献 1](#_Toc168000713)

[2 个人模块设计 2](#_Toc168000714)

[2.1 Master模块 2](#_Toc168000715)

[2.2 Region模块 4](#_Toc168000716)

[2.3 Client模块 6](#_Toc168000717)

[2.4 Util模块 6](#_Toc168000718)

[3 心得体会 8](#_Toc168000719)

# 1 个人开发过程与主要贡献

在本次大规模分布式minisql项目中，我担任了组长，在项目初完成了整体的项目框架搭建工作，将项目分成成了3个模块，敲定了使用sqlite数据库作为region内数据库，并在代码中实现集群管理、副本管理、分布式查询和部分容错容灾。

具体来讲，我在项目初时简要尝试了使用docker容器实现分布式等方式，最终选择了在本地利用端口和不同的数据库文件结合不同的终端来完成分布式节点的搭建，并完善了三个模块的框架，基本实现了没有分布式情况下client-master-region的rpc消息联通，也就是独立完成了单节点的minisql，以便之后组员们合作完成分布式设计部分。

随后，经过小组成员的一致讨论，我们决定使用主从服务器对进行副本管理，并且进行了分工。在之后的分布式工作中，我主要完成了集群搭建和副本管理工作。集群搭建上，我设计region节点通过etcd租约来维持心跳，master可以通过etcd集群来监控不同节点的上线、中断情况，并按照不同的情况简单写好了容错容灾的四个接口。在副本管理上，我主要完成了master和region的属性设计和部分对接函数。

在完成自己的模块后，我还参与了项目后期的整合与debug，并撰写了部分报告，完成了验收工作。

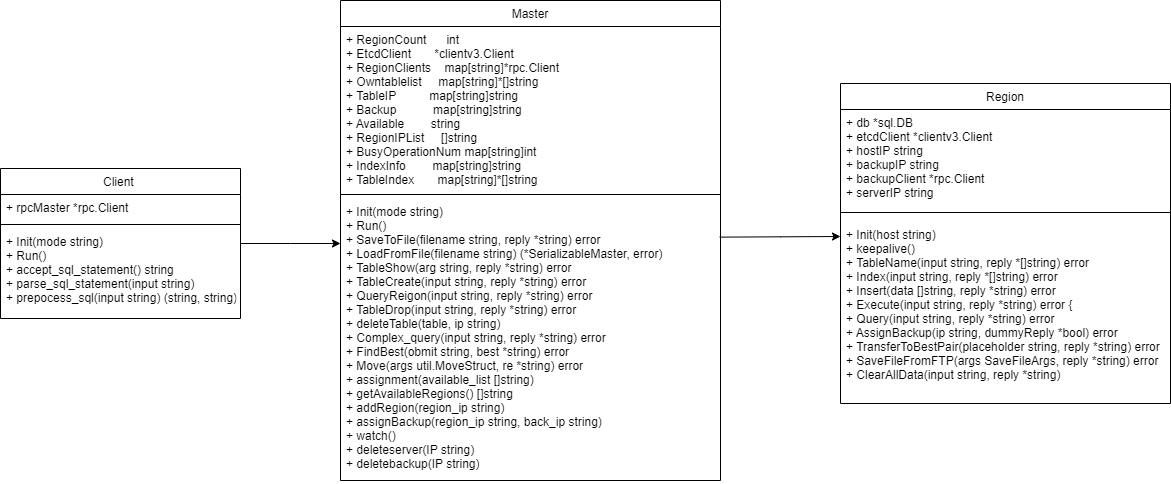
我的分工如下：

|  |  |  |
| --- | --- | --- |
| 成员姓名 | 学号 | 分工职责 |
| 3210104095 | 叶沐阳 | 总体框架搭建，集群管理、副本管理、分布式查询，部分容错容灾 |

# 2 个人模块设计

## 2.1 Master模块

Master的设计如下图所示:



在Master模块中，我完成了以下函数：

**func (master \*Master) Init(mode string)：**

该函数用于初始化 master 系统，包括读取数据库文件、初始化变量和缓存表以及加载数据。通过这个函数，master 系统能够在启动时准备好所有必要的数据和配置。

**func (master \*Master) Run()：**

该函数负责初始化 etcd 集群并开启 RPC 服务进程。通过这个函数，master 系统能够与 etcd 进行通信，并提供远程过程调用服务，以便其他模块进行交互。

**func (master \*Master) InitTableIP()：**

该函数用于同步本地数据库文件中的 table 信息。通过这个函数，master 可以确保本地缓存中的 table 信息与数据库中的数据一致。

**func (m \*Master) TableShow(arg string, reply \*string) error：**

该函数用于直接查看 owntablelist 中的所有 region 的 table，并以表格形式返回所有 table 及其所属 region 的 IP 地址。

**func (master \*Master) TableCreate(input string, reply \*string) error：**

该函数用于在数据库中创建 table。通过调用这个函数，可以在数据库中添加新的 table。

**func (master \*Master) assignment(available\_list []string)：**

该函数记录了 etcd 的相关代码，监视在 util 中的 region\_ip 的 etcd 变化。通过调用这个函数，可以监控 region 的 IP 地址变化，并进行相应的处理。

**func (master \*Master) getAvailableRegions()[]string：**

该函数用于获取当前待用的 region。通过调用这个函数，可以获取当前可用的 region 列表。

**func (master \*Master) addRegion (region\_ip string)：**

该函数用于初始化和后续加入 region 的连接。通过调用这个函数，可以将新的 region 加入到系统中，并建立连接。

**func (master \*Master) watch()：**

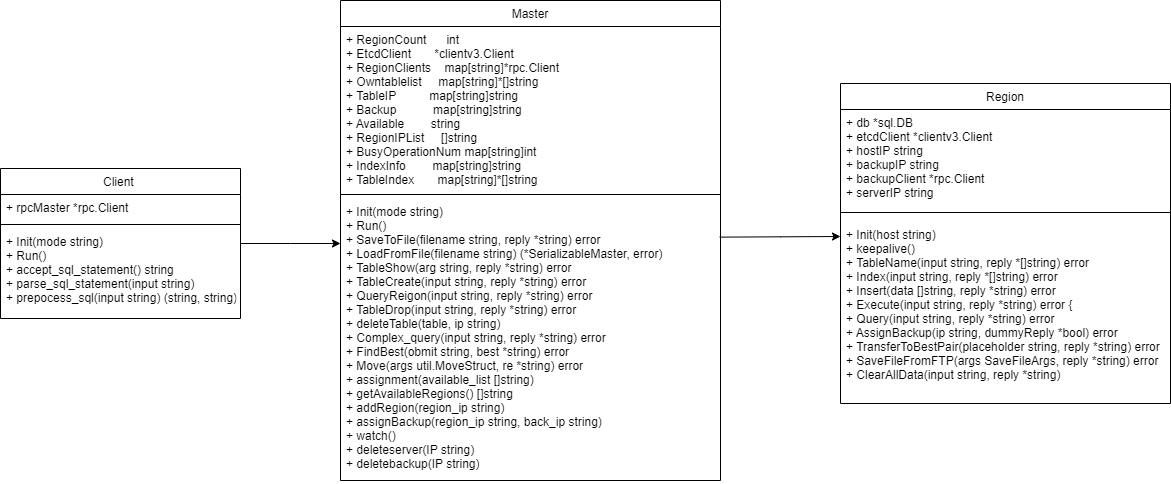
该函数用于监听 etcd 键值对存储系统的各种更新（键值对的增添与删除），并进行相应的操作。通过调用这个函数，可以实时监控 etcd 中的数据变化，并进行处理。

**func (master \*Master) watch()：**

该函数用于监听 etcd 键值对存储系统的各种更新（键值对的增添与删除），并进行相应的操作。通过调用这个函数，可以实时监控 etcd 中的数据变化，并进行处理。

## 2.2 Region模块

Region的类图如下：



在Region模块中，我主要实现了region的初始化，以及基础的查询、执行sql语句的函数。具体来讲，我完成了以下函数：

**func (region \*Region) Init()：**

该函数用于实现 region 的初始化，包括连接数据库文件和注册 RPC 服务。通过调用这个函数，region 服务器能够准备好所有必要的资源和服务，以便处理来自 Master 和 Client 的请求。

**func (region \*Region) keepalive()：**

该函数用于在 etcd 中保持活跃。通过调用这个函数，region 服务器可以定期向 etcd 发送心跳信号，确保自己在集群中的活跃状态，从而避免被误认为已下线。

**func (region \*Region) TableName(input string, reply \*[]string) error：**

该函数用于返回当前 region 管理的 table 的名字。通过调用这个函数，可以获取该 region 服务器上所有 table 的名称列表，便于管理和查询。

**func (region \*Region) Execute(input string, reply \*string) error：**

该函数用于处理非查询类的 SQL 语句。通过调用这个函数，可以执行插入、更新、删除等操作，并返回执行结果。

**func (region \*Region) Query(input string, reply \*string) error：**

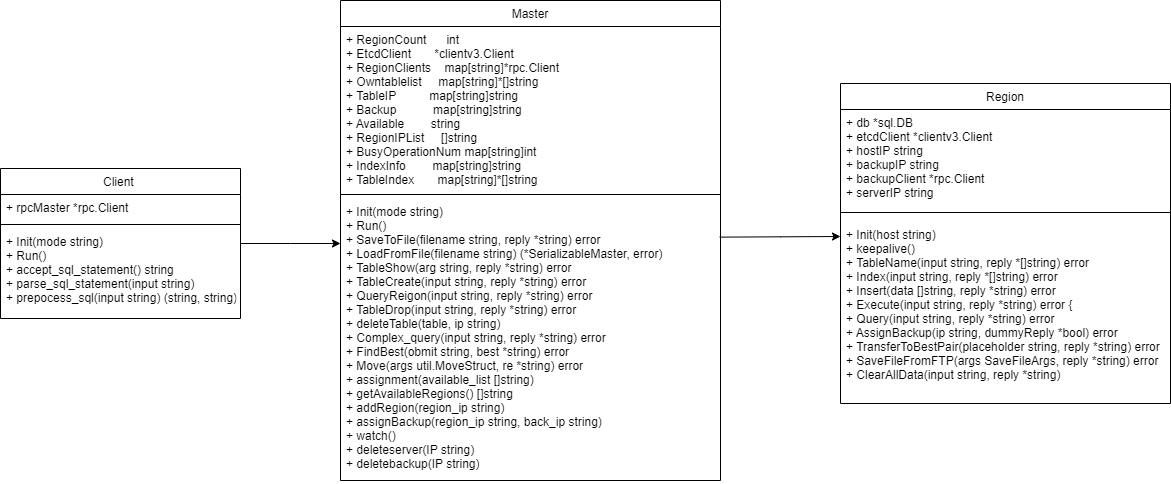
该函数用于处理查询类的 SQL 语句。通过调用这个函数，可以执行查询操作，并返回查询结果。

**func (region \*Region) AssignBackup(ip string, dummyReply \*bool) error：**

该函数用于给主 region 分配副本 region。通过调用这个函数，可以指定一个副本 region，以实现数据的高可用性和容错机制。

## 2.3 Client模块

Client的类图如下：



我实现了Client模块的大部分功能，并解析了sql语句，分类讨论了不同语句的处理方法；把从region返回结果按照表格的形式可视化输出。具体来讲，实现了以下函数：

**func (client \*Client) Init(mode string)**

该函数初始化了Client类，连接了master终端

**func (client \*Client) Run()**

该函数从命令行反复读取sql语句，直到遇到分号，简单处理了接收到的sql命令，区分是查询语句还是执行语句，通过RPC交给对应的master、region函数进行处理。

**func (client \*Client) parse\_sql\_statement(input string)：**

该函数用于解析sql语句，分类讨论查询语句和执行语句，提取table名称，询问master获取region的ip地址，区分table和index语句进行处理。

# 3 心得体会

在本项目的框架搭建过程中，我深入学习了一个分布式数据库的整体框架。本项目实现分布式的方法虽然比较简单，使用的是主从服务器进行备份的形式，虽然想起来很简单，但是在真正代码实现的时候才发现有很多的细节需要去考虑。

比如在容错容灾函数设计时，我们发现在不同的节点间转移数据时设计到大量数据变量同时变化，由于master中保存了很多关键数据，设计了很多变量，所以转移一个表格有很多变量也需要跟着改变，难免出现错漏，而出错后检查又需要花费大量时间，这告诉我们写代码的时候需要注意层次，例如在节点间转移表格这样的函数，应该按照一个顺序对master变量进行修改，不然会出现级联反应，从一个小变量开始导致整个函数报错。

在编程技术上，我在本项目中首次接触了Go语言，第一次使用了sqlite数据库、etcd集群和rpc消息通讯。Go语言在并发处理和性能上都表现优异，本次项目中我使用了很多golang的库，也算是偏向工业界的一个项目；同时，由于我负责了最初的框架搭建，各个类模块的计，方法的管理，互相之间接口的设计，都很大的提升了我的编程能力。

在团队合作上，我们每个成员都有自己主要负责的内容，由于分布式系统的紧密性、三个模块互相联通，因此大家沟通紧密，很好的提高了我们的团队沟通能力。