浙江大学

# 大规模信息系统构建技术导论

## 分布式 MiniSQL 系统设计报告

2022-2023 学年　春夏学期

组员信息（第一行请写组长信息）

| 学号 | 姓名 |
| --- | --- |
| **3200104091** | **沈轩喆** |
| 3200103483 | 孙宇桐 |
| 3200105252 | 李湘 |
| 3200105109 | 孔郁杰 |
| 3200105355 | 徐鑫 |

2023 年 4 月 12 日

# 目　　录

# 1 引言

## 1.1 设计目的

本项目是基于《数据库系统》课程学习的数据库基本知识和《大规模信息系统构建技术导论》课程学习的分布式系统与大规模软件系统构建的知识，所完成的分布式关系型 miniSQL 项目。

本项目设计并实现一个分布式的关系型 SQL 引擎，除实现数据库各类基本操作（增删改查、索引）、SQL 语句执行等基本功能外，包含 Zookeeper 集群、客户端、主从节点等多个模块，具有数据分区、均衡负载，、客户端缓存、副本管理、容错容灾等功能。

本系统使用 Java 语言进行项目构建，使用 Github 进行版本管理和协作开发，由小组五位成员共同完成。

## 1.2 设计说明

本程序采用 Java 程序设计语言，在 IntelliJ IDEA 平台下编辑、编译与调试。具体程序由 5 人组成的小组开发而成。小组成员的具体分工如表 1 所示：

**表 1 各成员分工表**

| 成员姓名 | 学号 | 分工 |
|---|---|---|
| 沈轩喆 | 3200104091 | 负责 client 模块的设计，将用户的输入输出进行封装并处理 |
| 孙宇桐 | 3200103483 | 负责主节点模块 minisql 的设计，并形成完整的 API 文档 |
| 李湘 | 3200105252 | 负责 region server 模块的设计并进行不同 server 之间调度算法的设计 |
| 孔郁杰 | 3200105109 | 负责 master 模块的设计，监听用户请求并进行容错容灾管理 |
| 徐鑫 | 3200105355 | 负责搭建系统框架，后期数据库的维护，以及各模块的功能调试 |

# 2 总体设计

## 2.1 功能模块设计

系统的总体架构设计如下图所示：

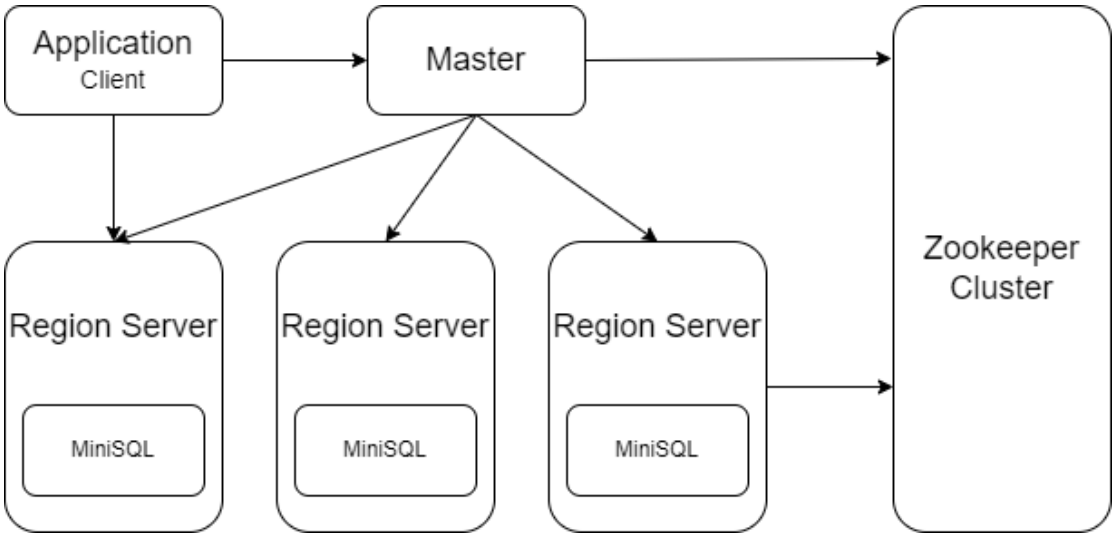


图 2.1 总体架构设计图

系统整体架构分为 Zookeeper Cluster、Master、Region Server 和 Application 四个模块。其中，Region Server 底层由 MiniSQL 提供服务，Application 由 Client 提供服务。

### 2.1.1 Zookeeper Cluster

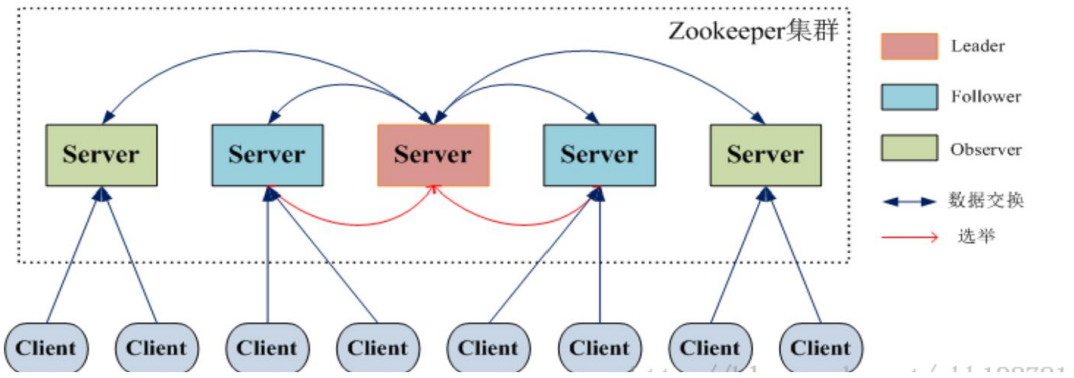


图 2.1.1 Zookeeper Cluster 设计图

Zookeeper Cluster 是一套分布式锁管理系统。它可以保证分布式系统的顺序一致性、原子性、单一镜像、可靠性以及实时性等特性，实现数据发布 / 订阅、负载均衡、命名服务、分布式协调 / 通知、集群管理、Master 选举、分布式锁、分布式队列等功能。

在本系统中，Zookeeper Cluster 主要负责两部分的任务:

(1) Region Server 管理。Master 和 Region Server 监控 Zookeeper 中的目录；知道 Region 集群中有哪些服务器； 当 Region Server 崩溃时，通过 Zookeeper 可以通知 Master，Master 做出适当的调整（容错容灾）。当 Master 崩溃时，在 Region Server 中选举出一个成为新的 Master。

(2) 小数据存储。存储了 Region Server 的元数据，用于进行 Region Server 的调度管理。

### 2.1.2 Master

在本系统中，Master 的主要功能有：

(1) 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；

(2) 维护 Region 服务器列表；

(3) 分配 Region（Region 直接对应一个 Table，不进行切分，分裂或合并）；

(4) 实现不同 Region Server 之间的负载均衡；

(5) 管理用户对表的增加、删除、修改、查询等操作；

(6) 对发生故障失效的 Region Server 上的 Region 进行迁移。

### 2.1.3 Region Server

在本系统中，Region Server 的主要功能有：

(1) Region Server 负责存储和维护分配给自己的 Region，处理来自客户端的读写请求；

(2) Region Server 利用 MiniSQL 来管理 Region，负责 MiniSQL 的启动和管理，和 Client 的通信；

(3) 有缓存机制，记录最近的若干次查询，可以在一定程度上提高效率。

### 2.1.4 Client

客户端并不直接从 Master 主服务器上读取数据，而是在获得 Region 的存储位置信息后，直接从 Region 服务器上读取数据。

客户端可以不依赖 Master，可以通过 Zookeeper 来获得 Region 位置信息（需要设计一套定位机制）或者从 Master 中获得，大多数客户端甚至从来不和 Master 通信，这种设计方式使得 Master 负载很小。

为减轻 Master 负担，在客户端可以有缓存，保存 Table 定位信息。

## 2.2 工作流程图

### 2.2.1 总体流程图

本系统分为 Client、Master 以及 Region 三个子模块，其中，Client 端用来处理用户的输入，并且将处理完成的输入提交给 Master。Master 利用 Zookeeper 等协助工具负责集群管理，以及系统的副本管理、容错容灾等内容。Region 模块则服从于 Master 的调度，执行对应的 SQL 语句。

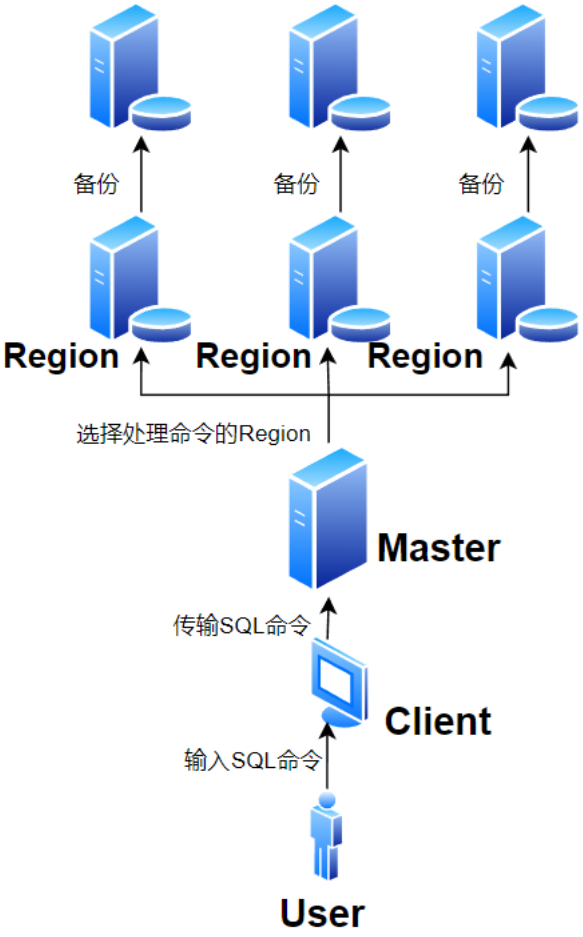


**图 2.2.1 总体流程图**

### 2.2.2 Client 部分

Client 对上向用户负责，处理用户的输入并在检查无误后对语句进行预处理，并将预处理结果发送给 Master。同时，Client 要负责回显 Master 的数据给用户，并对处理过程中的状态进行监控和播报。具体的流程图如下

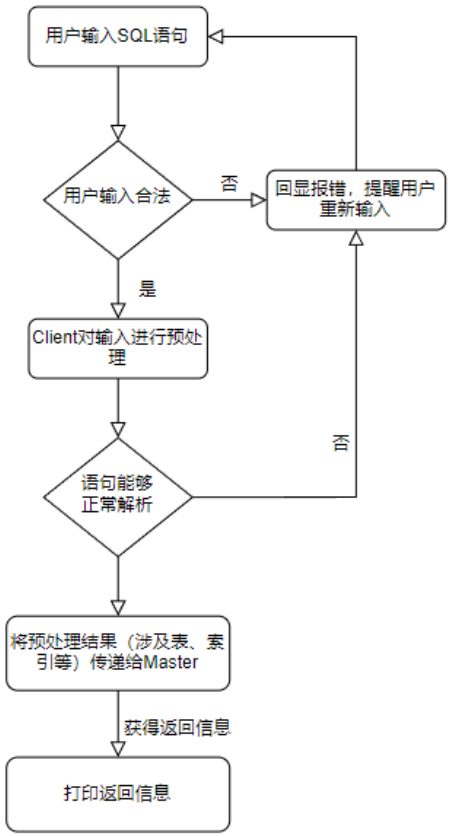


**图 2.2.2 Client 流程图**

### 2.2.3 Master 部分

Master 模块主要负责接受 Client 端的 SQL 请求，根据所管理的所有 Region 的状态表以及 Client 端预处理的字段信息来决定处理该请求的 Region，并将任务派放给对应的 Region。同时，Master 要管理所有 Region 的状态信息，并且决定在 Region 出现故障时候的对应操作（防控防灾），来协调整个分布式系统正常运行。

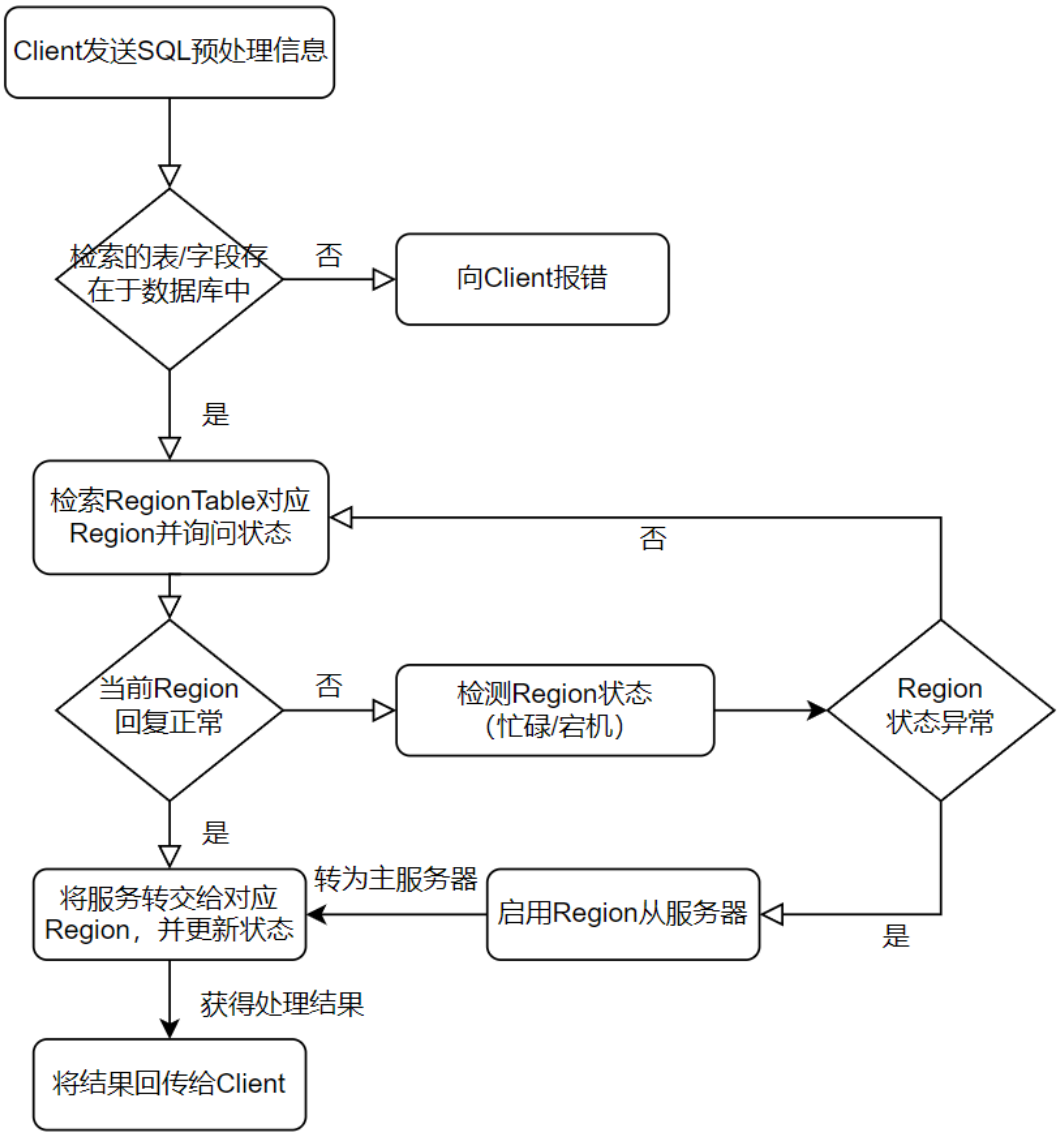


**图 2.2.3 Master 流程图**

### 2.2.4 Region 部分

Region 模块主要负责接收 Master 的调度并执行对应的 SQL 语句，同时 Region 需要定期向 Master 汇报自己的状态以及数据库情况，对于 SQL 语句的插入/删除等操作，对数据库进行修改，对于查询等操作，向 Master 汇报查询结果。同时，主 Region 在更新数据后要将日志同步给从 Region，来保持数据一致性。

考虑到 Region 直接与 miniSQL 接轨，Region 服务器应当在对 Master 的指令进行解析后将其重新封装为适配 miniSQL 的语句并通过 miniSQL 连接数据库进行 SQL 操作，当 miniSQL 出现查询报错时也应当将结果即时向上汇报。

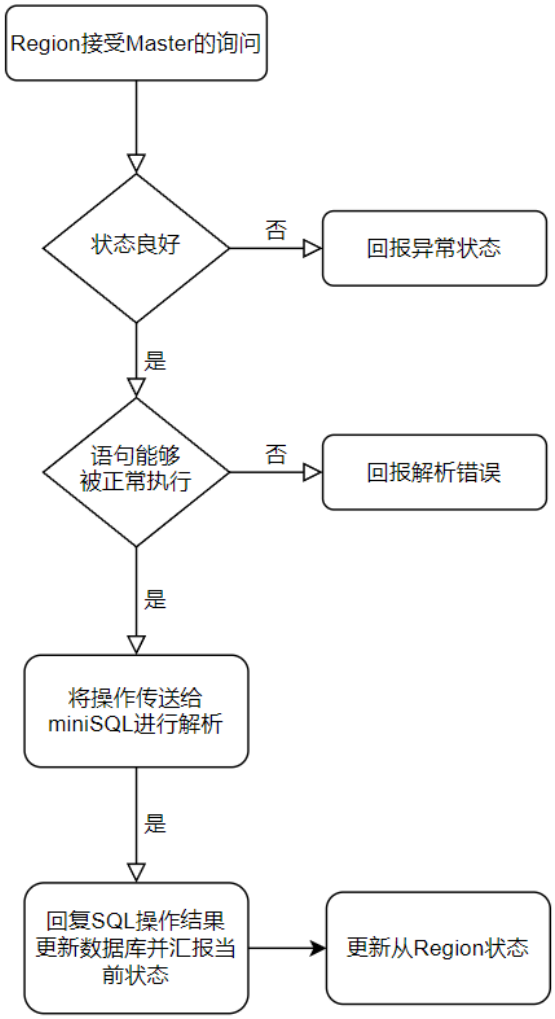


**图 2.2.4 Region 流程图**

### 2.2.5 防控防灾部分

考虑到分布式系统中可能存在数据不一致、服务器宕机等现象，Master 需要对此事件进行相应的应急管理，来使得服务可以正常运行。为了保证数据一致性，假设我们的集群中一共拥有 2k+1 台服务器，其中有 1 台 Master 服务器，并有 k 台主 Region 服务器和 k 台副 Region 服务器，针对不同的服务器，需要有不同的操作来适配，具体流程如下：

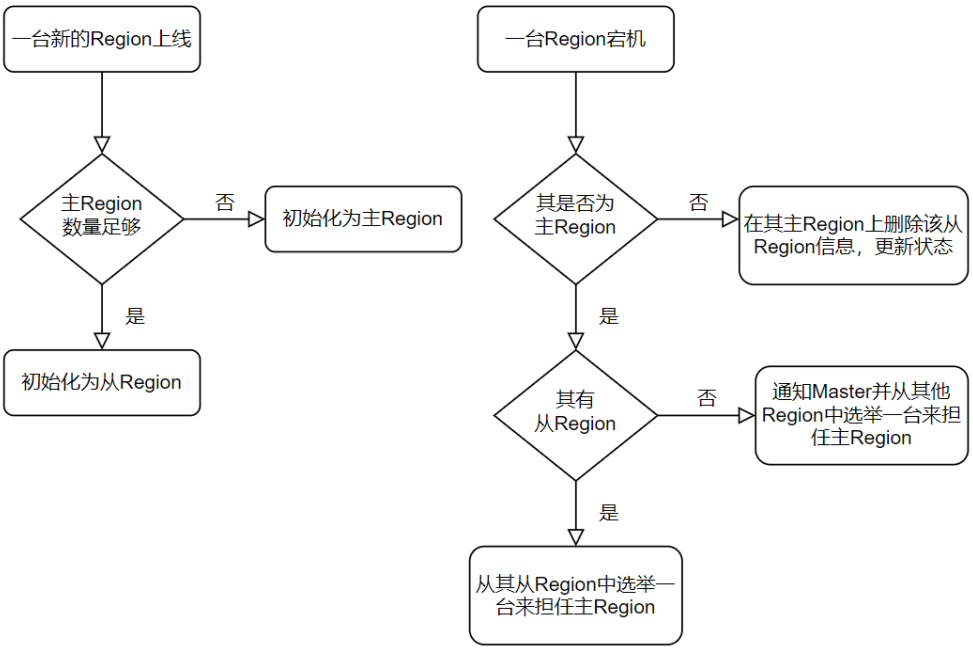


**图 2. 2. 5 防控防灾处理流程图**

# 3 进度安排

表 3 进度安排表

| 时间 | 任务 |
| --- | --- |
| **2023.03.31 - 2023.04.08** | 项目成员会议，项目整体设计架构初步完成 |
| **2023.04.09 - 2023.04.12** | 编写设计报告，对整体架构进行完善和责任明确 |
| **2023.04.13 - 2023.04.19** | 完成和完善项目各个模块详细设计 |
| **2023.04.20 - 2023.04.29** | 各个模块基本架构完成 |
| **2023.04.30 - 2023.05.06** | ZooKeeper 模块基本完成，理解所用 MiniSQL 各部分功能并进行改造，完成各项目间接口和 API 定义 |
| **2023.05.07 - 2023.05.13** | MiniSQL 各部分改造完成,Region,Server,Manager 三个模块基本完成 |
| **2023.05.14 - 2023.05.20** | 完成项目整体集成和修改，进行系统测试 |
| **2023.05.21 - 2023.05.27** | 项目文档编写，录制视频，提交作业 |

# 4 总结

本次的课程项目总体来说规模和难度较大，涉及的知识面广，细节设计较复杂，需要五位小组成员通力合作、互相协调配合，确保各模块能够成功运行并协调拓展成为一个具备分布式存储、数据分区、均衡负载、副本管理和容错容灾等完备功能的分布式数据库系统。

目前该项目暂时进行到初步的系统模块功能设计，开始进行基本的系统框架搭建，后续的具体实现还有比较大的工作量，需要我们对于相关知识自主学习。之后，我们会在初步的系统设计基础上，进一步完善模块功能，对项目进行若干次功能迭代，最终实现一个稳定的、符合设计预期、功能完备的分布式数据库系统。