

**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： | 王粤龙 |
| 学 院： | 计算机学院 |
| 系： | 软件工程系 |
| 专 业： | 软件工程 |
| 学 号： | 3200105512 |
| 指导教师： | 鲍凌峰 |

2023年 5 月 5 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、米博宇 实验地点：

## 一、实验内容

* 设计分布式MiniSQL的region服务端模块
* region服务端模块启动后，首先连接数据库，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，持续监听父节点中的信息。父节点中的信息发生改变时，根据父节点中信息对数据库执行相应操作，并将结果返回。
* 管理和维护Region发生的变化，如果副机失效，主机应从Master请求新的副机，并将其中的数据拷贝过去，如主机失效，副机成为新的主机，请求新的副机，并将其中的数据拷贝过去

## 二、功能描述

### 1. 与数据库交互

region服务端数据库与数据库产生实际交互，当接受到来自客户端或者master服务端的请求时，region判断请求类型，并调用DbOperation中的相关接口对数据库进行增删改查

### 副本维护

客户端或者master服务端的每个会对数据库产生修改的操作会被主机和副机同时监听到并且执行，从而保证主副机的数据一致性

### 容错容灾

主副机时刻监听父节点下的节点个数，如果主机监听到节点个数小于2，则说明副机挂掉，向Master申请新的副机，并完成数据库的拷贝，如果副机监听到节点个数小于2，则说明主机挂掉，副机成为新的主机，并向Master申请新的副机

### 三、接口说明

本模块集成为DistributedServer类，为客户端线程类Worker Runnable提供main函数接口，新建一个客户端线程时即新建一个DistributedServer对象并对其进行初始化操作。

本模块调用getSqlStatement接口监听发送给自身的sql请求。调用keepTwoServer接口保证主副机一方挂掉后能唤起新的主副机

## 工作原理

### Main

在main方法中，创建了一个DistributeServer对象，设置了数据库和服务器的相关参数，然后依次执行以下步骤：

* 连接数据库：调用connectDatabase方法连接到MySQL数据库。
* 连接ZooKeeper集群：调用connectZookeeper方法连接到ZooKeeper，并在指定的区域节点下注册一个临时节点。
* 获取SQL语句并执行：调用getSqlStatement方法，监听自己的父节点（区域节点），获取并执行SQL语句。
* 监听自身节点：调用getSelfData方法，监听自身节点的变化。
* 保持服务器在线：调用business方法，使服务器一直在线。
* 其他操作：调用其他自定义方法，例如keepTwoServers和sendCopyRequest。

### getSelfData

getSelfData方法中，通过zkClient.getData方法监听自身节点的变化。当节点变化时，会调用process方法进行处理。

* 首先通过zkClient.exists方法检查节点是否存在，如果存在则获取节点数据，否则输出"1"。
* 将节点数据转换为字符串，并根据字符串内容执行相应的操作。如果字符串为"need copy db"，则调用sendDbData方法发送数据库数据；否则调用copyDb方法复制数据库。

### copyDb

copyDb方法用于复制主服务器的数据库表格。

* 首先将字符串按照特定的分隔符分割成多个部分，包括表格名、创建表格的SQL语句和查询结果。
* 使用Statement对象执行删除原有表格的操作。
* 使用Statement对象依次执行创建表格的SQL语句。

解析查询结果的字符串，将每一行解析为Map对象，并将数据插入到对应的表格中。

### sendDbData

* 获取服务器上的所有数据库表格名称。
* 构建一个字符串，将表格名称以"#"分隔，并在开头添加"SELECT##"标识符。
* 依次执行每个表格的"SHOW CREATE TABLE"语句，将结果添加到字符串中，同样以"#"分隔。
* 执行每个表格的"SELECT \* FROM"语句，将查询结果添加到字符串中。
* 使用ZooKeeper客户端将构建的字符串数据设置到指定节点下，以便其他服务器可以获取并复制这些数据。

### sendCopyRequest

* 检查当前服务器是否为主服务器。如果是主服务器，则不执行任何操作。
* 如果当前服务器是副本服务器，则使用ZooKeeper客户端将一个标识为"need copy db"的数据设置到指定节点下。
* 这样，主服务器将根据该请求将数据库数据复制到副本服务器。

### keepTwoServers

* 使用ZooKeeper客户端获取当前服务器所在区域（region）下的所有子节点。
* 如果子节点数量少于2个，表示当前服务器是最后一个在线的服务器，需要唤醒一个新的服务器。
* 调用evokeServer方法来执行唤醒操作。

### evokeServer

* 如果当前服务器不是主服务器：
  + 使用ZooKeeper客户端删除当前服务器的节点，并将服务器名称更改为带有".1"后缀的名称。
  + 使用ZooKeeper客户端创建一个临时节点，表示它已经成为了新的主服务器。
  + 通过循环检查主服务器是否已经复制了数据库数据，如果复制完成，则退出循环。
  + 使用ZooKeeper客户端设置一个特定节点的数据，用于通知其他服务器触发唤醒操作。

### getSqlStatement

* 使用ZooKeeper客户端监听指定节点的数据变化。
* 当数据发生变化时，触发Watcher中的process方法。
* 在process方法中，解析从节点获取到的数据，数据由三部分以"##"分隔：
  + 目标节点的名称
  + SQL操作的类型（如SELECT、SHOW、UPDATE）
  + 具体的SQL语句
* 根据不同的操作类型，执行相应的操作，例如执行SELECT语句并将结果存入目标节点，执行SHOW语句并将结果与SELECT语句的结果合并后存入目标节点，执行UPDATE语句并将结果存入目标节点。
* 这样，其他服务器可以监听并获取这些数据，并根据数据进行相应的操作。

## 开发心得

通过这一次项目，我获得了宝贵的经验和心得。处理数据库相关操作时，对于数据的传输和复制，使用ZooKeeper作为分布式协调服务是一种有效的方式。ZooKeeper提供了节点管理、数据同步和事件监听等功能，方便实现分布式系统中的数据一致性和协作。

通过使用ZooKeeper的节点操作，可以方便地管理服务器的上下线、角色切换等任务。例如，在保持两个服务器在线的场景中，我们通过检查子节点数量并采取相应的措施来保证集群的可用性。这种动态管理机制有助于实现高可用和容错性。

然后，在开发过程中，代码的可维护性和可扩展性很重要。使用清晰的命名和注释，结构化的代码组织，以及合适的设计模式，可以提高代码的可读性和可维护性。另外，考虑到未来需求的变化，代码应具备良好的扩展性，能够方便地添加新的功能或适应更复杂的场景。

通过这一次项目，我深刻认识到了分布式系统和数据库管理的重要性，以及ZooKeeper作为分布式协调服务的优势。同时，我也加深了对代码设计和可扩展性的理解。这些经验将对我的未来开发工作产生积极的影响，并帮助我更好地应对类似的挑战。