DataLoader 模块设计

1. 任务概述

DataLoader 向上提供 MiniSQL 的 Client 端和 Master 以及 RegionServer 交换数据的 API 的封装。

2. 工作流程

DataLoader 同其它类之间的合作流程：

- 当执行 createTable 或者 dropTable 时：

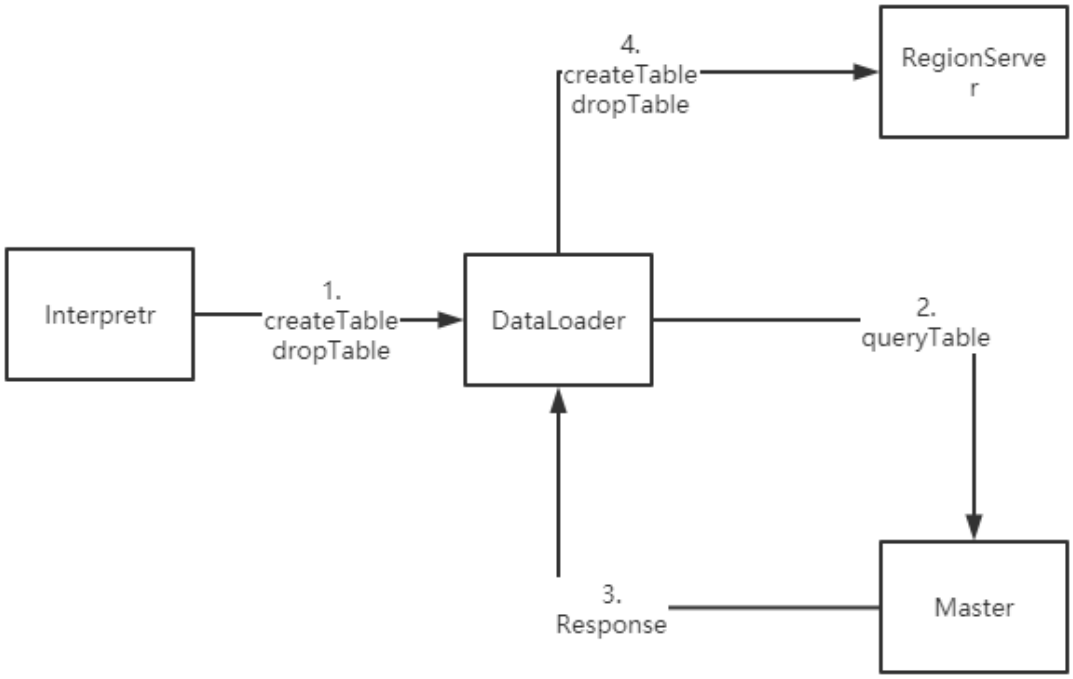

图 1 DataLoader 合作图-createTable&dropTable

- 当执行 createIndex 或者 dropIndex 时：

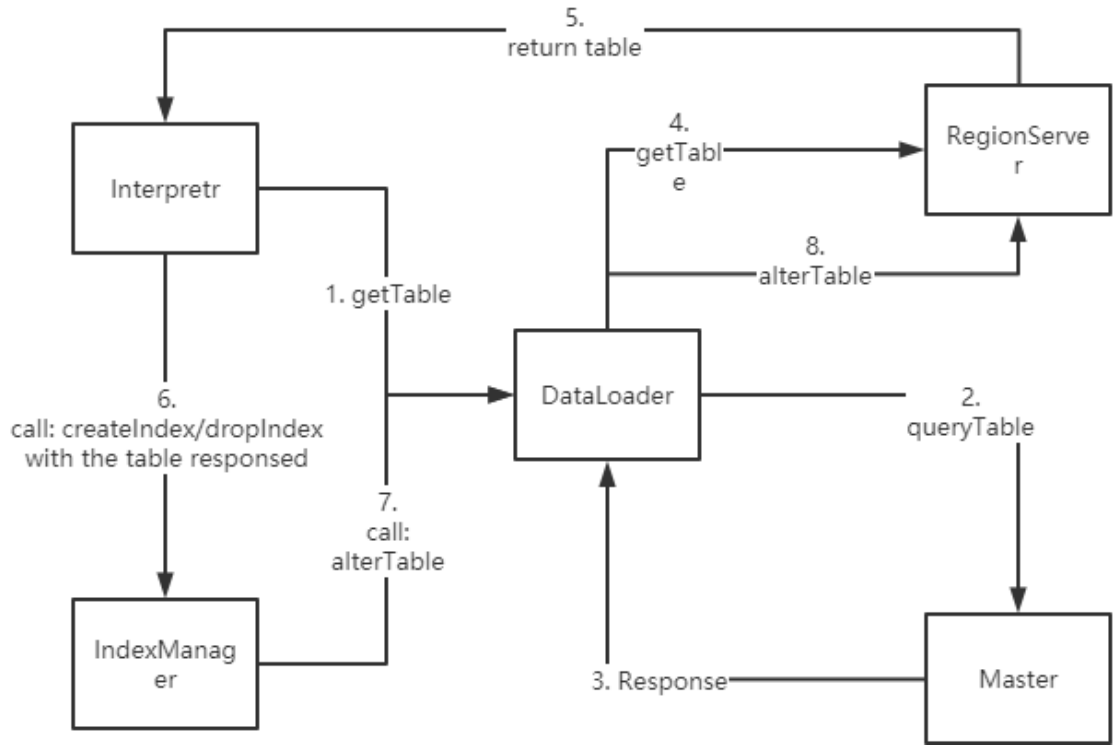

图 2 DataLoader 合作图-createIndex&dropIndex

- 当执行 insert, delete, update 时：

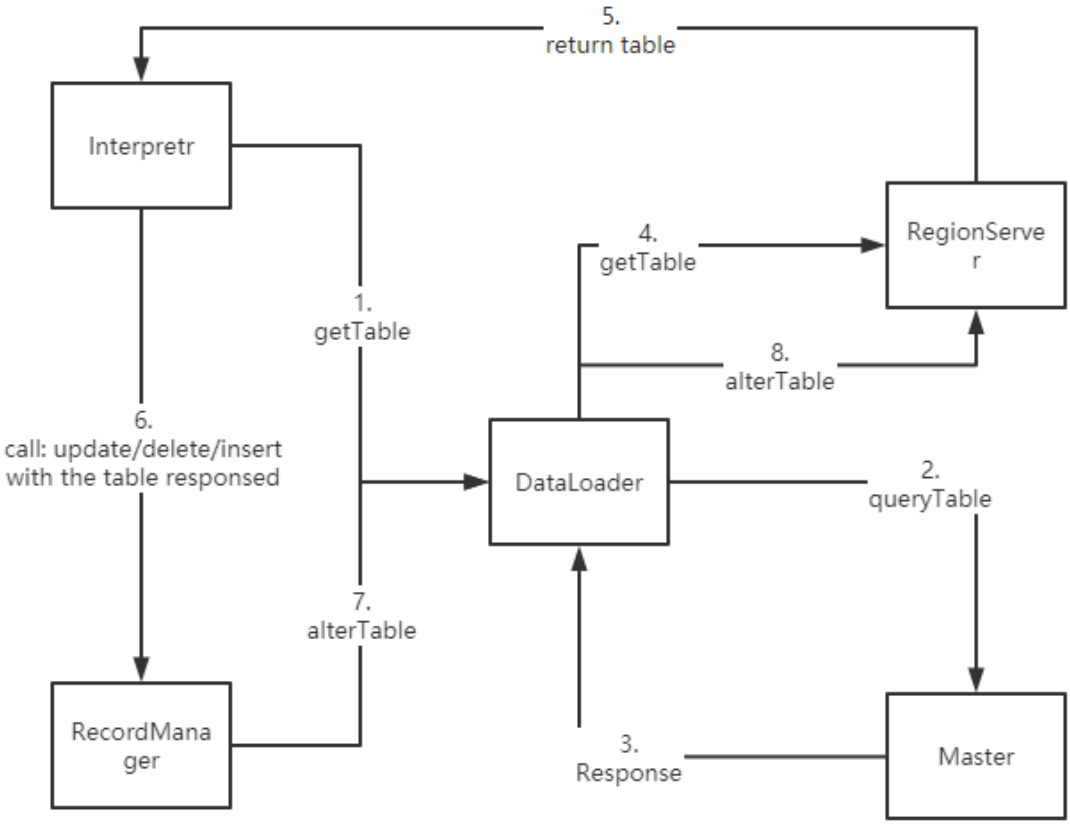

图 3 DataLoader 合作图-insert/delete/update

- 当执行 select 时：

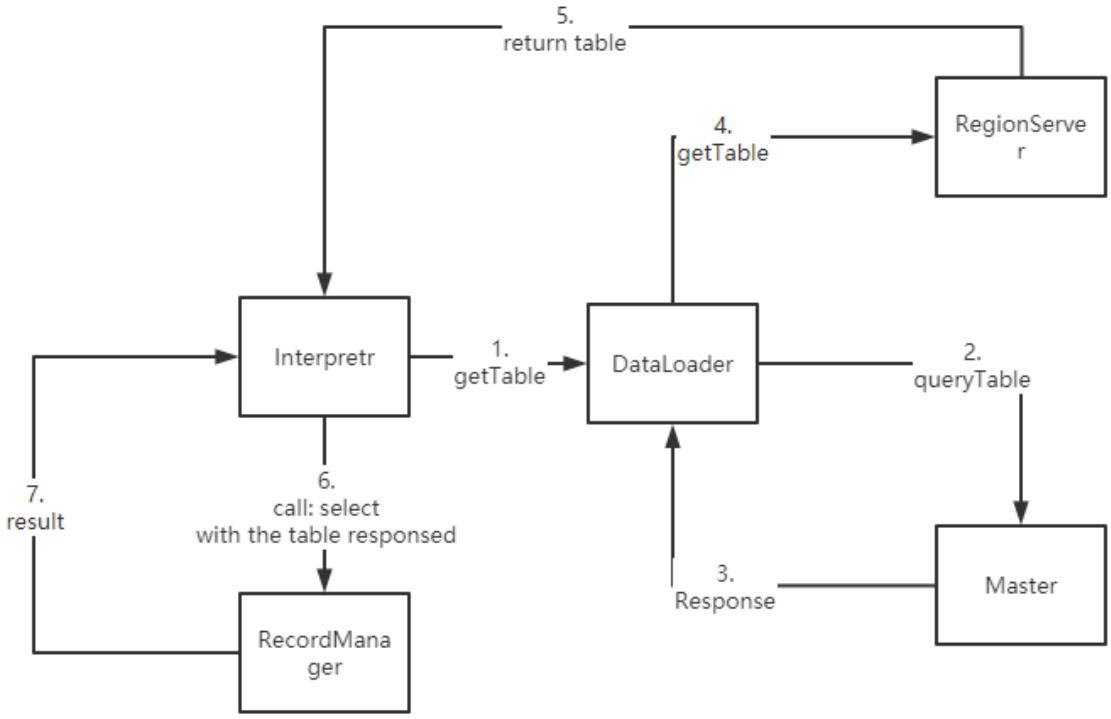

图 4 DataLoader 合作图-select

3. 功能描述

本模块服务于 Interpreter, RecordManager, IndexManager，向上提供四个接口：

- createTable
  接受 Interpreter 的参数之后，首先会向 Master 通过 RPC 请求 RegionServer 位置，即这张表应该被创建在哪个 RegionServer 上。

  如果此表存在，则 Master 会拒绝创建表格。否则将返回 RegionServer 的 IP 地址。

  再调用 RegionServer 的 RPC，完成表格创建。

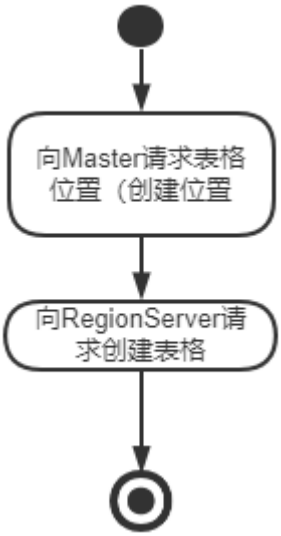

图 5 DataLoader 流程图-createTable

- dropTable
  接受 Interpreter 的参数（表格名称）之后，首先会向 Master 通过 RPC 请求 RegionServer 位置，即这张表现在在哪个 RegionServer 上，随后 Master 将返回 RegionServer 的 IP 地址。

  再调用 RegionServer 的 RPC，完成表格删除操作。

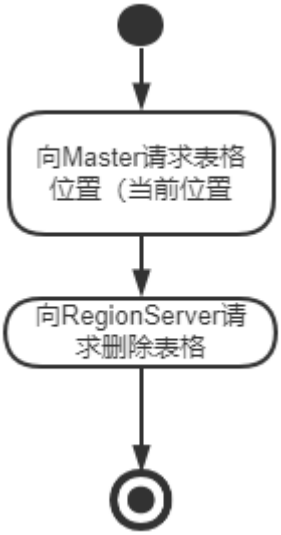

图 6 DataLoader 流程图-dropTable

- getTable
  接受 Interpreter 的参数（表格名称）之后，首先查询 DataLoader 内部的缓存，看是否有保留此表格的位置信息。

  如果在缓存中找到了位置信息，则会直接向缓存中的 RegionServer 发起 RPC 调用，请求获取表格。如果请求失败，则会重新向 Master 请求表格的位置（即缓存失效，一个可能的原因是之前那一台服务器失效了），这也会使得这张表格的缓存更新。

如果缓存中没有表格位置的记录，则会向 Master 请求表格位置，之后向 RegionServer 请求表格的数据，并且将这张表格的位置信息加入缓存。

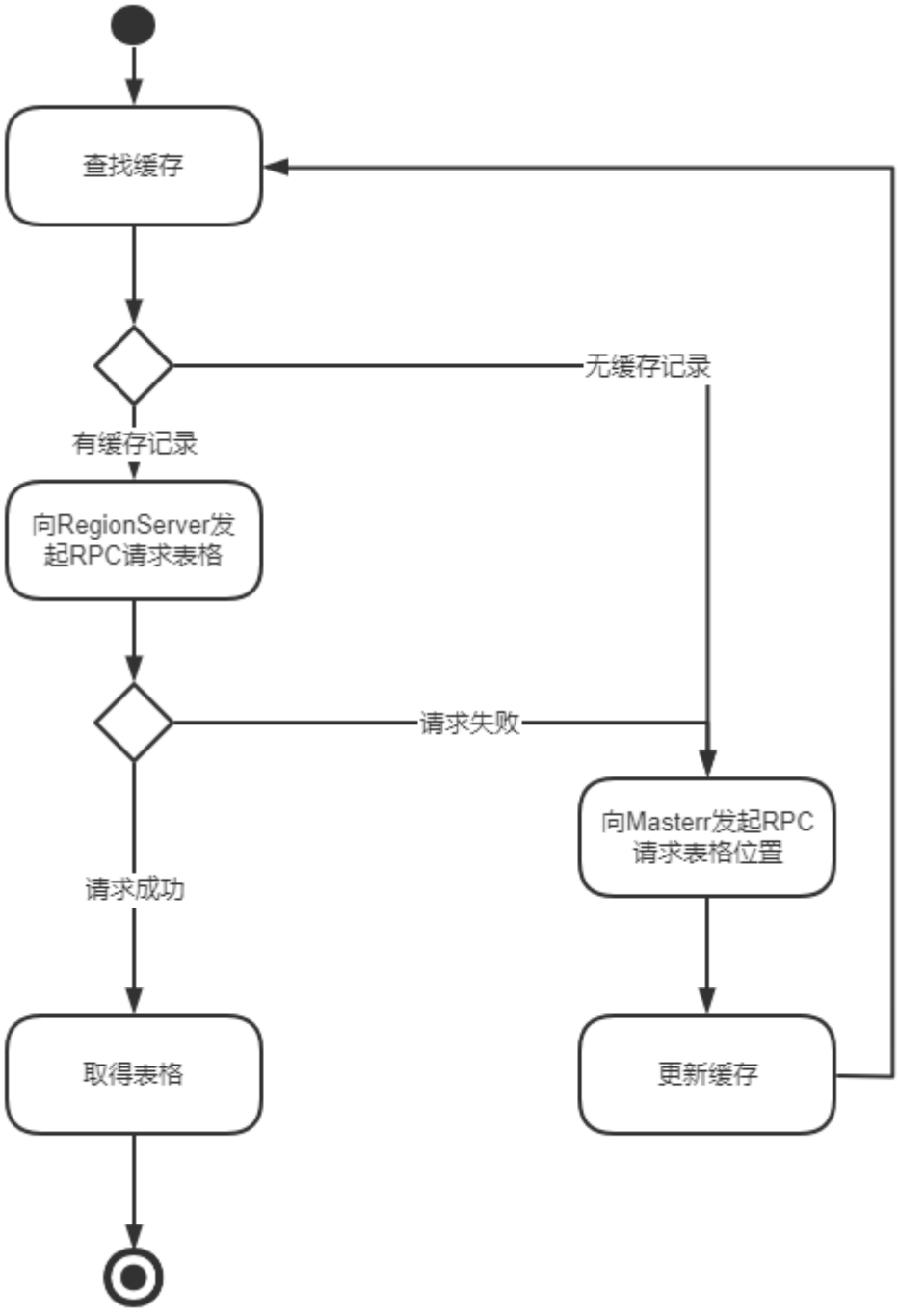

图 7 DataLoader 流程图-getTable

- alterTable

接受 Interpreter 的参数（表格名称）之后，首先查询 DataLoader 内部的缓存，看是否有保留此表格的位置信息。

如果在缓存中找到了位置信息，则会直接向缓存中的 RegionServer 发起 RPC 调用，请求更新表格。如果请求失败，则会重新向 Master 请求表格的位置（即缓存失效，一个可能的原因是之前那一台服务器失效了），这也会使得这张表格的缓存更新。

如果缓存中没有表格位置的记录，则会向 Master 请求表格位置，之后向 RegionServer 请求更新表格的数据，并且将这张表格的位置信息加入缓存。

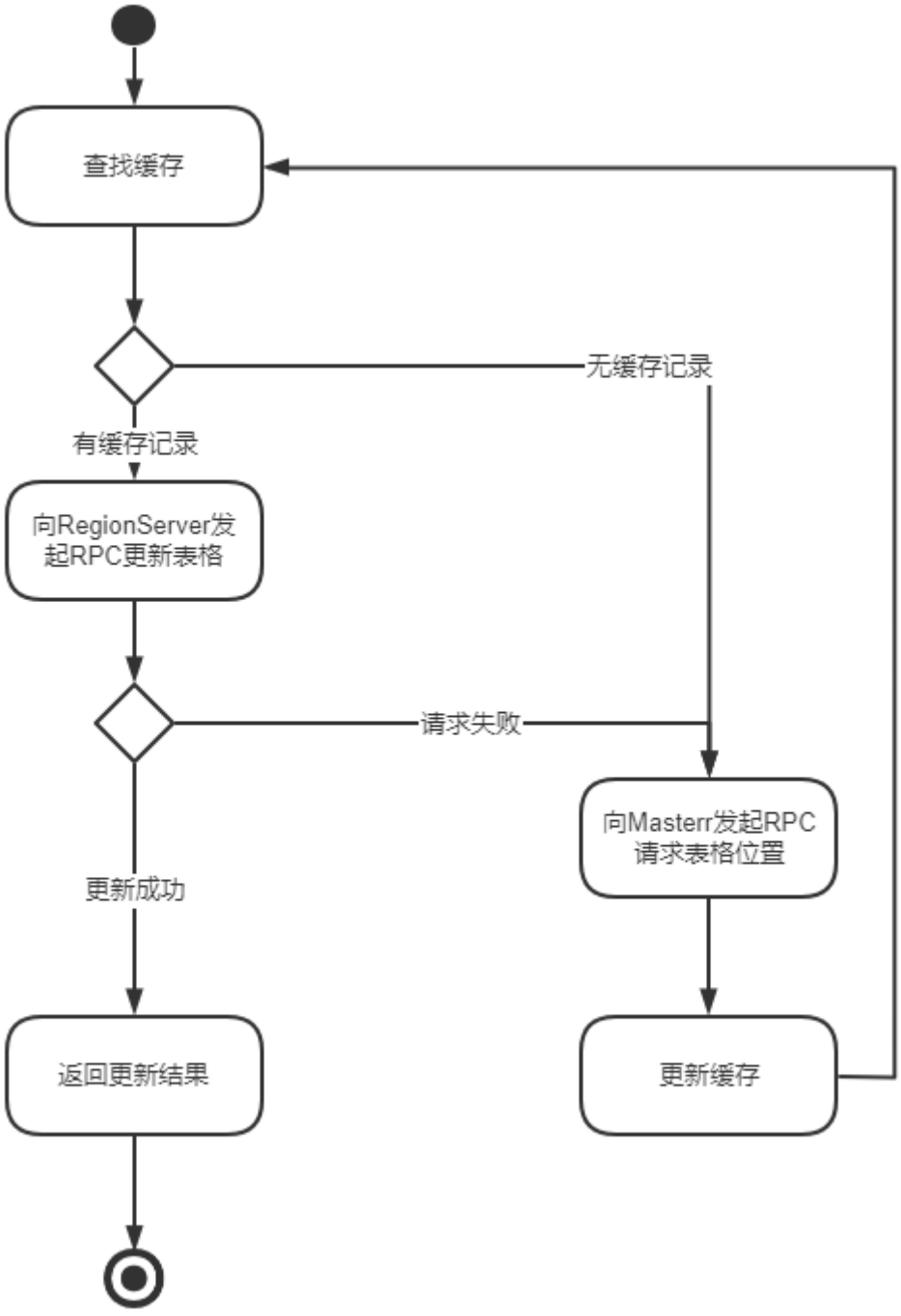

图 8 DataLoader 流程图-alterTable