6 RECOVERY MANAGER

6.1 实验概述

Recovery Manager 负责管理和维护数据恢复的过程，包括：

- 日志结构的定义
- 检查点CheckPoint的定义
- 执行Redo、Undo等操作，处理插入、删除、更新，事务的开始、提交、回滚等日志，将数据库恢复到宕机之前的状态

出于实现复杂度的考虑，同时为了避免各模块耦合太强，前面模块的问题导致后面模块完全无法完成，同组成员的工作之间影响过深，我们将Recovery Manager模块单独拆了出来。另外为了减少重复的内容，我们不重复实现日志的序列化和反序列化操作，实现一个纯内存的数据恢复模块即可。

6.2 数据恢复

数据恢复是一个很复杂的过程，需要涉及系统的多个模块。以InnoDB为例，在其恢复过程中需要redo log、binlog、undo log等参与，这里把InnoDB的恢复过程主要划分为两个阶段：第一阶段主要依赖于redo log的恢复，而第二阶段需要binlog和undo log的共同参与。

第一阶段，数据库启动后，InnoDB会通过redo log找到最近一次checkpoint的位置，然后根据checkpoint相对应的LSN开始，获取需要重做的日志，接着解析日志并且保存到一个哈希表中，最后通过遍历哈希表中的redo log信息，读取相关页进行恢复。

在该阶段中，所有被记录到redo log但是没有完成数据刷盘的记录都被重新落盘。然而，InnoDB单靠redo log的恢复是不够的，因为数据库在任何时候都可能发生宕机，需要保证重启数据库时都能恢复到一致性的状态。这个一致性的状态是指此时所有事务要么处于提交，要么处于未开始的状态，不应该有事务处于执行了一半的状态。所以我们可以通过undo log在数据库重启时把正在提交的事务完成提交，活跃的事务回滚，保证了事务的原子性。此外，只有redo log还不能解决主从数据不一致等问题。

第二阶段，根据undo中的信息构造所有未提交事务链表，最后通过上面两部分协调判断事务是否需要提交还是回滚。InnoDB使用了多版本并发控制(MVCC)以满足事务的隔离性，简单的说就是不同活跃事务的数据互相是不可见的，否则一个事务将会看到另一个事务正在修改的数据。InnoDB借助undo log记录的历史版本数据，来恢复出对于一个事务可见的数据，满足其读取数据的请求。

在我们的实验中，日志在内存中以`LogRec`的形式表现，定义于`src/include/recovery/log_rec.h`。出于实现复杂度的考虑，我们将Recovery Manager模块独立出来，不考虑日志的落盘，用一个unordered_map简易的模拟一个KV Database，并直接在内存中定义一个能够用于插入、删除、更新，事务的开始、提交、回滚的日志结构。`CheckPoint`检查点应包含当前数据库一个完整的状态，该结构已帮大家实现好了。`RecoveryManager`则包含`UndoPhase`和`RedoPhase`两个函数，代表Redo和Undo两个阶段。

在本节中，你需要实现以下结构和函数：

- `Init()`： `RecoveryManager`的初始化函数
- `RedoPhase()`:从`CheckPoint`开始，根据不同日志的类型对`KvDatabase`和活跃事务列表作出修改
- `UndoPhase()`：Undo阶段，对每个未完成的活跃事务进行回滚

- `struct LogRec`:内存日志结构。这里可以不考虑消耗内存空间的优化，实现一种能够用于所有类型日志的日志结构。
- `CreateInsertLog()`：创建一条插入日志
- `CreateDeleteLog()`：创建一条删除日志
- `CreateUpdateLog()`：创建一条更新日志
- `CreateBeginLog()`：创建一条事务开始日志
- `CreateCommitLog()`：创建一条事务提交日志
- `CreateAbortLog()`：创建一条事务回滚日志

实现完成后，你的代码需要通过`recovery_manager_test.cpp`中的测试用例。

6.3 模块相关代码

- `src/include/recovery/log_rec.h`
- `src/include/recovery/recovery_manager.h`
- `test/recovery/recovery_manager_test.cpp`

6.4 思考题

本模块中，为了简化实验难度，我们将Recovery Manager模块独立出来。如果不独立出来，真正做到数据库在任何时候断电都能恢复，同时支持事务的回滚，Recovery Manager应该怎样设计呢？此外，CheckPoint机制应该怎样设计呢？

注：如果完成了本模块，请在实验报告里完成思考题。思考题占本模块30%的分数，请尽量回答的详细些，比如具体到涉及哪些模块、哪些函数的改动，大致怎样改动。有能力、有时间的同学也可以挑战一下直接在代码上更改。

6.5 诚信守则

1. 请勿从其它组或在网络上找到的其它来源中复制源代码，一经发现抄袭，成绩为`0`；
2. 请勿将代码发布到公共Github存储库上