第九章作业

1.

- (1)
  - 2PC的目的是保证提交阶段的原子性操作
  - 基本过程:
    1. 准备阶段
       1. 协调者节点在本地记录Begin Commit信息到REDO日志;
       2. 协调者向所有参与者询问是否可以执行提交操作（发起投票），并等待所有参与者答复;
       3. 各参与者检查是否可以提交，或者执行子事务操作直到提交前那一刻，将 undo 和 redo 信息记入子事务日志中（但不提交事务）。如参与者认为可以提交，将Ready写入本地REDO日志
       4. 参与者回应协调者，同意提交则返回Vote Commit消息，进入就绪状态等待协调者进一步消息；如果参与者本地失败，则返回Vote Abort消息，并写入Abort到本地日志。
    2. 提交阶段
       - **若收到所有参与者的Vote Commit:**
         1. 协调者在本地记录Commit信息到REDO日志;
         2. 协调者向所有参与者节点发出Global Commit消息，进入Commit状态;
         3. 参与者收到Global Commit消息后，记录Commit消息到REDO日志，正式完成提交操作（设置了事务提交完成标志），释放事务期间占用的资源;
         4. 参与者向协调者发送Commit End消息;
         5. 如果协调者收到了所有参与者的的Commit End消息，在本地记录End Commit信息到REDO日志，完成事务。
       - **若收到某个参与者的Vote Abort，或者协调者在第一阶段的询问超时之前无法获取某些参与者的回应:**
         1. 协调者在本地记录Abort信息到REDO日志;
         2. 协调者向所有参与者节点发出Global Abort消息，进入Abort状态;
         3. 参与者收到Global Abort消息后，记录Abort消息到REDO日志，利用事务回滚机制执行回滚操作，释放事务期间占用的资源;
         4. 参与者向协调者发送Abort End消息;
         5. 协调者收到了所有参与者的Abort End消息后，事务完成，在本地记录End Abort信息到REDO日志。
- (2)

- **缺点**:
  - 同步阻塞：参与者在等待其他参与者节点的响应过程中，所有的参与者节点都是事务阻塞的；
  - 单点故障：协调者一旦发生故障，参与者会一直阻塞下去，尤其在提交阶段，参与者都处于锁定事务资源的状态中；
  - 数据不一致：第二阶段，当协调者向参与者发送commit请求后，发生了局部网络异常或在发送commit请求时协调者发生了故障，若只有部分参与者收到了commit请求，但其他部分未接到commit请求的节点无法执行提交操作，此时出现了数据不一致现象。
  - 协调者发出commit消息后宕机，而接收到该消息的参与者也同时宕机，此时无法知道事务的真实状态。
- **改进**
  - 三阶段提交、基于paxos的2PC等改进协议。

2.

- (1)
  1. canCommit: 协调者向参与者发送 commit 请求，参与者如果可以提交就返回 Vote Commit消息(参与者不执行事务操作)，否则返回Vote Abort消息；
  2. preCommit: 协调者根据参与者的回答和超时机制，确定是否可以继续事务的 preCommit，分两种情况：
     1. 收到所有参与者的Vote Commit，协调者发出Global Commit消息，进入下一阶段；
     2. 某个参与者返回Vote Abort或者等待超时，协调者发出Global Abort消息，参与者执行事务回滚。
  3. doCommit:
     - 进行真正的事务提交，在消息传递过程中存在超时机制。
     - 协调者向所有参与者发出doCommit 请求，参与者收到该消息后正式执行事务提交，并释放事务期间占用的资源。
     - 各参与者向协调者反馈完成的ack消息，协调者收到所有参与者反馈的ack消息后，即完成事务提交。
- (2)
  - **优点:**
    - 在等待超时后协调者或参与者会中断事务，相对于二阶段提交减小了阻塞范围。
    - 避免了协调者单点问题，阶段 3 中协调者出现问题时，参与者会继续提交事务。