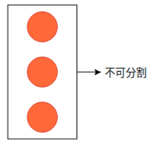
ACID中的AD

相关概念

原子性(Atomicity)

- 解释
  - 原子性是指在一个事务中的所有操作要么全部执行，要么全部不执行，不会出现部分执行的情况
  - 当事务中的任何一条语句执行出错时，已经执行的SQL语句必须全部撤销，并使得数据库状态与执行事务之前的数据库状态保持一致
- 示例
  - 转账操作应该是解释原子性的一个最佳的例子了。A给B转账100，A账户扣除100，B账户增加100，这两个操作必须作为一个不可分割的原子单位进行执行

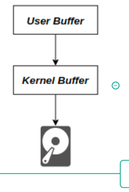


持久性(Durability)

- 解释
  - 持久性是指事务一旦被提交，其结果就应该是永久性的。即使在事务提交后数据库发生断电宕机，下次重启时对数据进行正确的恢复
  - 但是，若是因外部因素导致的数据库宕机，则不在持久性保证的范围内，例如磁盘永久损坏
  - **持久性保证的是高可靠性(High Reliability)，而不是高可用性(High Availability)。高可用性通常由数据副本来进行保证**
- 示例
  - 同样以转账操作为例，当转账事务成功提交后，若数据库发生内部宕机(机房断电、进程Crash)，当数据库正确重启时，能够保证数据该转账数据不会丢失

Linux文件I/O

- 在Linux中，使用系统调用open()打开文件，并获取文件描述符后，紧接着调用write()系统调用将数据写入磁盘，write()调用成功返回后，数据可能并未写入至磁盘，而仅是写入至了内核缓冲区中



- Linux内核会寻找合适的时机将内核缓冲区的数据刷新至磁盘中，通常由系统级别的线程执行
- 当数据位于内核缓冲区并且未同步至磁盘，若此时系统发生宕机，则会永久地丢失这部分数据
- 所以，为了避免数据丢失，通常会在wirte()调用成功返回后，调用fsync()或者fdatasync()将数据强制地从内核缓冲区刷新至磁盘，这个过程的性能完全由磁盘性能决定

redo log

- 在MySQL InnoDB存储引擎中，事务特性的原子性和持久性均由redo log(重做日志)实现

基本概念

- InnoDB重做日志是一种物理日志，即详细的记录了对某一数据页的具体修改
- InnoDB通过Force Log At Commit机制来实现持久性。即当事务提交时，首先要将所有的日志写入磁盘，此时事务已提交成功。后续InnoDB线程定时地将数据持久化进.ibd核心存储文件中
  - 这里的所有日志包括redo log，undo log以及binlog

组成

重做日志缓冲(redo log buffer)

- 位于内存，为易失数据，其目的在于提供写入性能
- 重做日志缓冲由log block组成，本质上类似于一个数组，并且是一个循环使用的数组，最大大小通常为2GB

- log block
  - 大小固定为512字节，由日志数据+日志块头以及日志块尾组成

重做日志文件(redo log file)

- 位于磁盘，为持久文件，设计该文件的目的就是为了实现事务的原子性和持久性

重做日志缓冲持久化的时机

- 重做日志首先位于用户空间的重做日志缓冲区中，需要在恰当的时机将其持久化
- 时机
  - **当事务提交时**
  - 当log buffer有一半的空间被使用时
- 方式
  - InnoDB存储引擎在打开日志文件时，并未指定O_DIRECT位掩码，所以，在调用write()系统调用时，数据将首先写入内核缓冲区中
  - 再由前面所提到的Linux文件I/O方式，则必须调用fsync()或者是fdatasync()将内核缓冲区的数据刷新至磁盘，否则就失去了持久化特性的意义了
- 所以，当批量地插入数据时，尽量在一个事务中完成(一次fsync代价)，而不是一条插入语句一个事务(多次fsync代价)

两阶段提交

重做日志与binlog

- **redo log的目的在于实现事务的原子性以及持久性，属于存储引擎范围内的日志记录，并且直接记录对物理页的物理操作**
- **binlog记录的是每一个事务所执行的所有SQL语句，如果binlog_format的格式为Row的话，还会进一步地记录行修改前和修改后的数据内容，属于数据库范围内的日志记录，并且记录的是事务的逻辑日志**
- 当一个事务成功提交时，重做日志与binlog必须要能够保持一致，即向这两个文件内写入数据时也应该是一个原子过程

redo log与binlog的两阶段提交

1. redo log持久化至磁盘，并将状态更新为commit
2. binlog持久化至磁盘
3. 提交事务，并将状态更新为commit

两阶段提交在Crash的情况

- binlog有记录，redo log状态为prepare
  - 在写入binlog之后事务崩溃，需对数据进行恢复
- binlog有记录，redo log状态为commit
  - 此时事务已正常完成，无需进行恢复
- binlog无记录，redo log状态为prepare
  - 事务在中间崩溃，事务直接回滚
- binlog无记录，redo log状态为commit
  - 在binlog持久化时崩溃，事务直接回滚