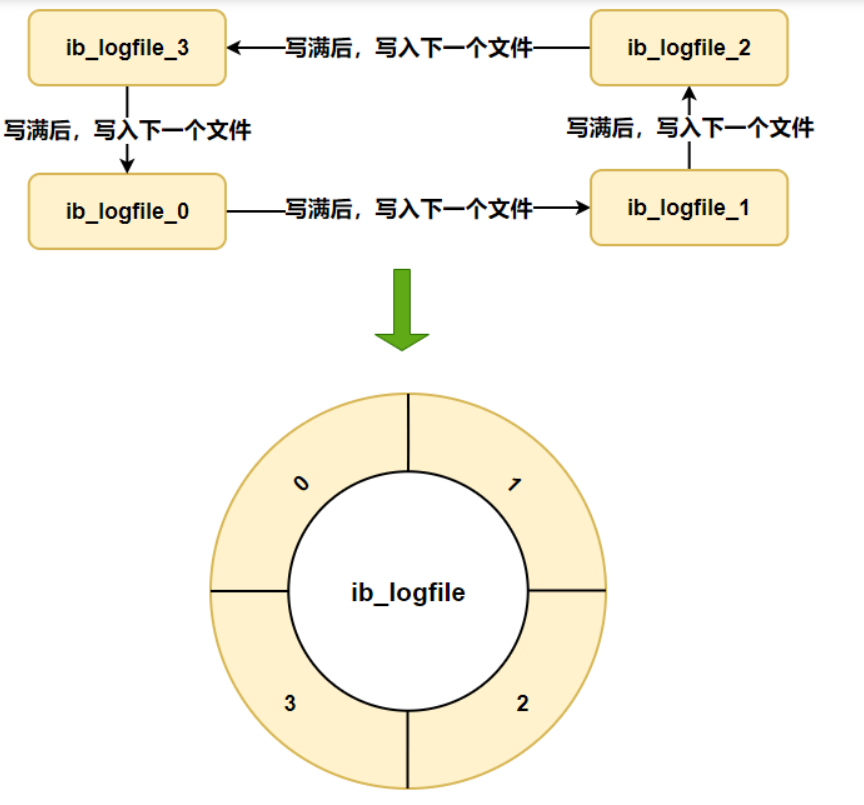
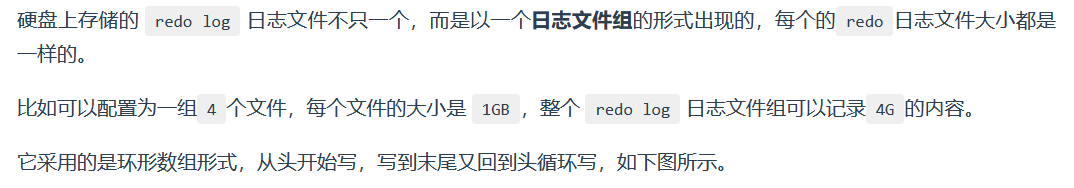
Redolog作用：

在DML操作时，是先写入缓存中的，比如bufferpool，redolog的目的就是防止buffer pool还没写回磁盘时宕机造成的数据不一致。数据库宕机一段时间，恢复后redolog中的数据就是还没有更新的，需要更新到磁盘。

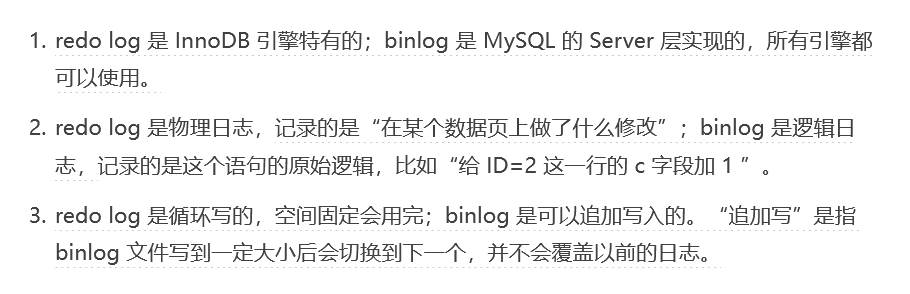
磁盘上的Redolog是循环写入的，写满了怎么办？

Redolog写满了会将之前的数据擦除，将bufferpool中对应的数据写回磁盘。

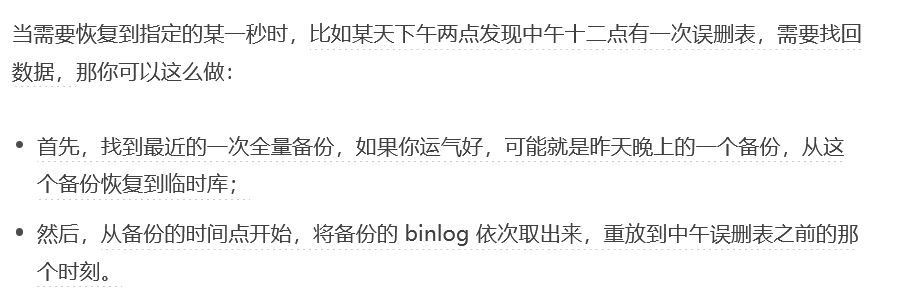
日志文件组：



两个日志的区别：



怎样让数据库恢复到半个月内任意一秒的状态？



全局锁：

Flush tables with read lock (FTWRL) 加全局读锁。DML、DDL语句会被阻塞。

解锁：Unlock tables、客户端断开。

可以用来备份库，mysqldump -uroot -p010802 fache > fache.sql;

也可以不用锁来备份，通过MVCC拿到一致性视图来备份，当 mysqldump 使用参数–single-transaction 的时候，导数据之前就会启动一个事务，来确保拿到一致性视图。

但是不支持MVCC的引擎就不能这样，MyISAM连事务都不支持，肯定不支持MVCC。

因此single-transaction 方法只适用于所有的表使用事务引擎的库。

缺点：业务停摆、从库加锁不能同步主库的binlog造成主从延迟。

set global readonly=true也可以锁库，但是FTWRL 命令之后由于客户端发生异常断开，那么 MySQL 会自动释放这个全局锁，整个库回到可以正常更新的状态。而将整个库设置为 readonly 之后，如果客户端发生异常，则数据库就会一直保持 readonly 状态，这样会导致整个库长时间处于不可写状态，风险较高。

表级锁：表锁、元数据锁、意向锁

表锁：

lock tables xxx read/write;

解锁：unlock tables、客户端断开。

线程A给表1加锁：

加读锁：线程A和其他线程对表1能读不能写。线程A释放锁前对其他表不能读也不能写。

加写锁：线程A能对表1读写，其他线程对表1不能读写。线程A释放锁前对其他表不能读也不能写。

元数据锁：DDL加写锁，DQL、DML加读锁。读锁在语句开始执行时加上，事务提交后才释放。

这就造成一个问题：如果写锁被一个长事务(MDL在事务结束后才释放锁)阻塞，那后面的读锁都会被阻塞，线程会爆满。所以如果要在一个表上做DDL语句可以选择先暂停长事务。

意向锁：

行锁：

两阶段锁协议：自动加锁，在事务结束时释放。

如果你的事务中需要锁多个行，要把最可能造成锁冲突、最可能影响并发度的锁尽量往后放。

间隙锁：

跟间隙锁存在冲突关系的，是“往这个间隙中插入一个记录”这个操作。间隙锁之间都不存在冲突关系。

具体可以看极客时间。

临键锁：

加锁时是分为两步，先加间隙锁再加行锁，间隙锁和间隙锁之间是不互斥的，间隙锁只和插入操作有冲突，但行锁之间有冲突。

出现死锁后，有两种策略来解决：

等待锁超时。默认超时等待时间为50s，可通过innodb\_lock\_wait\_timeout参数来设置，具体等待时间需要根据自己的实际业务来设置。

死锁检测。设置参数innodb\_deadlock\_detect = on，开启后，系统会自动检测死锁的事务并回滚某一个事务，让其他可以事务继续执行。

Undolog清除时机：

每个事务通过readview都能找到自己当前能看到的最晚的版本，如果unlog记录没有事务会用到(能看到的最晚版本)则会删除。

长事务：

1. 两阶段锁协议，锁到事务结束才会释放(包括元数据锁)，容易造成锁冲突。
2. Unlog会保留长事务能看到的记录导致unlog很长。

Innodb和myisam区别：

事务、外键、锁、聚簇索引、崩溃恢复

哈希索引：

优点：等只查询快

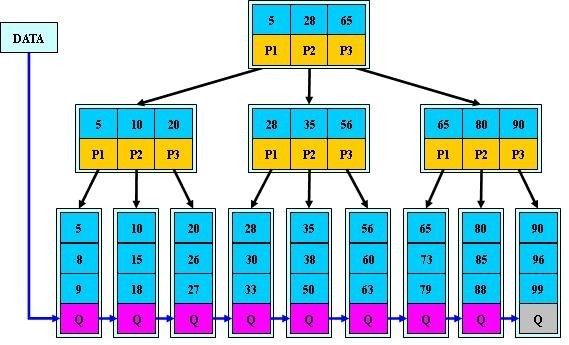
缺点：范围查询慢

有序数组：

优点：等值查询快(二分)、范围查询快

缺点：插入删除慢适合静态存储引擎

B+树结构：



注意：

每个结点中元素个数和分支个数相同，而B树的节点个数比分支个数少1.

为什么自增字段做主键：

二级索引占用空间少。

插入删除时防止页分裂、页合并。

缓冲池：<https://juejin.cn/post/6844903875271475213>

写缓冲(change buffer以前的版本叫insert buffer以前只支持insert)：

Buffer pool的作用是查询或修改数据时将磁盘中的页保存到pool中，下次增删改查直接在内存中操作。针对的是读写操作。

Change buffer的作用是当buffer pool中没有你要的页时，将增删改操作记录在change buffer中，当buffer pool中有数据时再merge。针对的是写操作。

Change buffer只适用于修改非唯一索引，因为唯一索引需要判断数据是否唯一，这需要将数据页读到内存。所以尽量使用非唯一索引。

Change buffer适用于写多读少的场景。因为如果读多的话，刚把修改操作写进change buffer就要读，而读取就需要将页调入内存。

Redo log和change buffer：

Redo log主要为了减少写磁盘的次数，change buffer主要为了减少读磁盘的次数。