# 精选留言(1087)

置顶

## 1.Redo log 和Bin log分别记录的内容

> 我可以认为redo log 记录的是这个行在这个页更新之后的状态，binlog 记录的是sql吗?

作者回复:

```
Redo log不是记录数据页“更新之后的状态”，而是记录这个页 “做了什么改动”。

Binlog有两种模式：
    1.statement 格式的话是记sql语句，
    2.row格式会记录行的内容，记两条，更新前和更新后都有记录。
（谢谢你提这个问题，为了不打断文章思路，这个点没在正文写，但是又是很重要的点😄😄）
```

## 2.Binlog和Redolog 二阶提交

Bin log 用于记录了完整的逻辑记录，所有的逻辑记录在 bin log 里都能找到，所以在备份恢复时，是以 bin log 为基础，通过其记录的完整逻辑操作，备份出一个和原库完整的数据。

在两阶段提交时，若 redo log 写入成功，bin log 写入失败，则后续通过 bin log 恢复时，恢复的数据将会缺失一部分。(如 redo log 执行了 update t set status = 1，此时原库的数据 status 已更新为 1，而 bin log 写入失败，没有记录这一操作，后续备份恢复时，其 status = 0，导致数据不一致）。

若先写入 bin log，当 bin log 写入成功，而 redo log 写入失败时，原库中的 status 仍然是 0 ，但是当通过 bin log 恢复时，其记录的操作是 set status = 1，也会导致数据不一致。

其核心就是， redo log 记录的，即使异常重启，都会刷新到磁盘，而 bin log 记录的， 则主要用于备份。

我可以这样理解吗？还有就是如何保证 redo log 和 bin log 操作的一致性啊?

作者回复:

```
几乎全对，除了这个“两阶段提交时，若redo log写入成功，但binlog写入失败…”这句话。

实际上，因为是两阶段提交，这时候redolog只是完成了prepare，而binlog又失败，那么事务本身会回滚，所以这个库里面status的值是0。

如果通过binlog 恢复出一个库，status值也是0。

这样不算丢失，这样是合理的结果。

两阶段就是保证一致性用的。
你不用担心日志写错，那样就是bug了…
```

## 3.binlog还不能去掉

老师,今天MYSQL第二讲中提到binlog和redo log, 我感觉binlog很多余，按理是不是只要redo log就够了?[费解]
您讲的时候说redo log是InnoDB的要求，因为以plugin的形式加入到MySQL中，此时binlog作为Server层的日志已然存在，所以便有了两者共存的现状。但我觉得这并不能解释我们在只用InonoDB引擎的时候还保留Binlog这种设计的原因.

作者回复:

> binlog还不能去掉。
>
> 一个原因是，redolog只有InnoDB有，别的引擎没有。
>
> 另一个原因是，redolog是循环写的，不持久保存，binlog的“归档”这个功能，redolog是不具备的。

## 4.redo log被写满 被迫刷盘

老师您好，有一个问题，如果在非常极端的情况下，redo log被写满，而redo log涉及的事务均未提交，此时又有新的事务进来时，就要擦除redo log，这就意味着被修改的的脏页此时要被迫被flush到磁盘了，因为用来保证事务持久性的redo log就要消失了。但如若真的执行了这样的操作，数据就在被commit之前被持久化到磁盘中了。当真的遇到这样的恶劣情况时，mysql会如何处理呢，会直接报错吗？还是有什么应对的方法和策略呢?

作者回复: 👍,

> 会想到这么细致的场景
>
> 这些数据在内存中是无效其他事务读不到的（读到了也放弃），同样的，即使写进磁盘，也没关系，再次读到内存以后，还是原来的逻辑

## 5.误删binlog

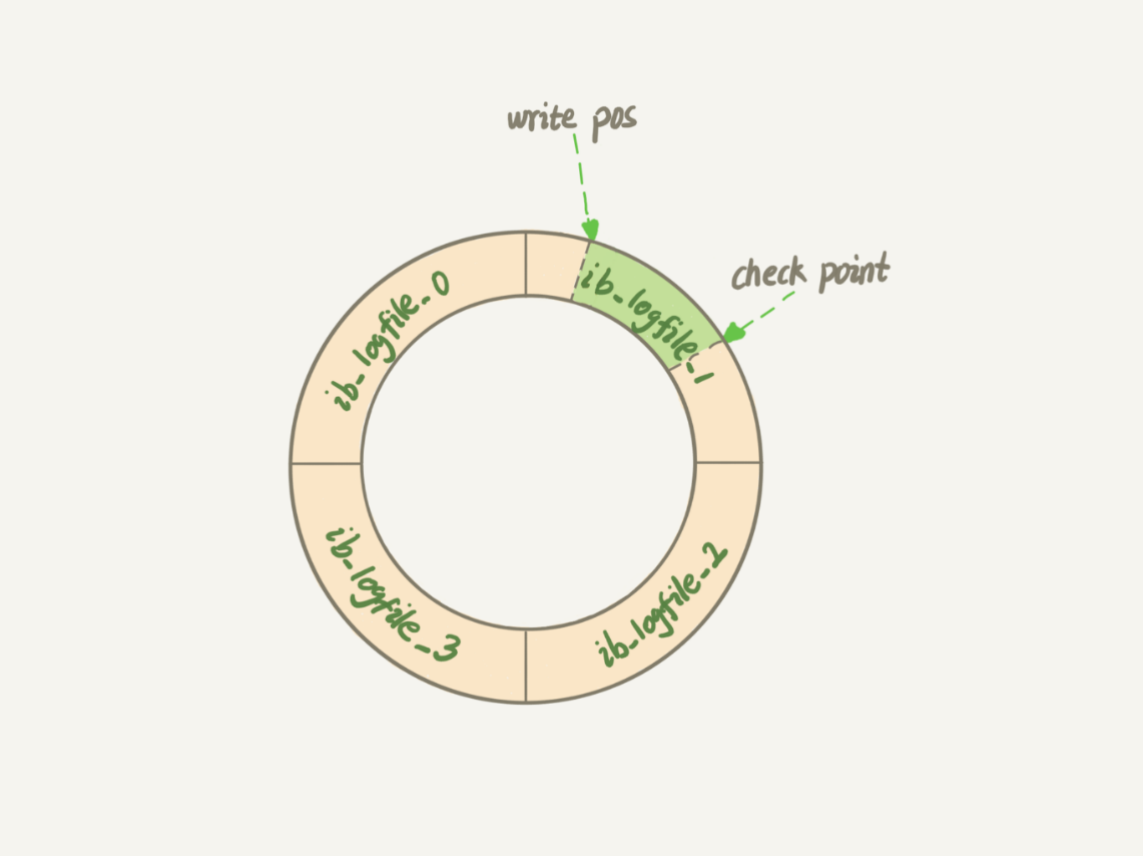
老师您好，我之前是做运维的，通过binlog恢复误操作的数据，但是实际上，我们会后知后觉，误删除一段时间了，才发现误删除，此时，我把之前误删除的binlog导入，再把误删除之后binlog导入，会出现问题，比如主键冲突，而且binlog导数据，不同模式下时间也有不同，但是一般都是row模式，时间还是很久，有没什么方式，时间短且数据一致性强的方式

作者回复:

> 其实恢复数据只能恢复到误删之前到一刻，
> 误删之后的，不能只靠binlog来做，因为业务逻辑可能因为误删操作的行为，插入了逻辑错误的语句，
>
> 所以之后的，跟业务一起，从业务快速补数据的。只靠binlog补出来的往往不完整

## 6.write pos和checkout

有个问题请教老师，既然write pos和checkout都是往后推移并循环的，而且当write pos赶上checkout的时候要停下来，将checkout往后推进，那么是不是意味着write pos的位置始终在checkout后面，最多在一起，而这和老师画的图有些出入，不知道我的理解是不是有些错误，请老师指教。




作者回复:

> 因为是“循环”的，图中这个状态下，write_pos 往前写，写到3号文件末尾，就回到0号继续写，这样你再理解看看“追”的状态。
>
> 刚好借你这个问题，说明一下，文中“write pos和checkpoint之间的是'粉板'上还空着的部分，可以用来记录新的操作。”
>
> 这句话，说的“空着的部分”，就是write pos 到3号文件末尾，再加上0号文件开头到checkpoint 的部分。

## 7. bin log 和 redo log 二阶提交场景数据保存

我再来说下自己的理解 。
1 prepare阶段 2 写binlog 3 commit
当在2之前崩溃时
重启恢复：后发现没有commit，回滚。备份恢复：没有binlog 。
一致
**当在3之前崩溃**
**重启恢复：虽没有commit，但满足prepare和binlog完整，所以重启后会自动commit。备份：有binlog. 一致**

作者回复:

👍，get 完成
说的非常对

老师，我想问下如果提交事务的时候正好重启那么redo log和binlog会怎么处理？此时redo log处于prepare阶段，如果不接受这条log，但是binlog已经接受，还是说binlog会去检查redo log的状态，状态为prepare的不会恢复？
作者回复：好问题👍表示中间那段你都听明白了👍

Binlog如果已经接受，那么redolog是prepare，binlog已经完整了对吧，这时候崩溃恢复过程会认可这个事务，提交掉。（你可以分析下这种情况下，是否符合我们要达到的“用binlog恢复的库跟原库逻辑相同”这个要求）

## 8.Rodolog也要IO操作耗费性能，为啥它就性能高

老师，我这想请教两个问题：
1.写redo日志也是写io（我理解也是外部存储）。同样耗费性能。怎么能做到优化呢
2.数据库只有redo commit 之后才会真正提交到数据库吗

作者回复:

1. Redolog是顺序写，不用去“找位置”，并且可以组提交，还有别的一些优化，收益最大是是这两个因素；

2.是这样，正常执行是要commit 才算完，但是崩溃恢复过程的话，可以接受“redolog prepare 并且binlog完整” 的情况

请教老师，redo log是为了快速响应SQL充当了粉板，这里有两个疑问
1.redo log本身也是文件，记录文件的过程其实也是写磁盘，那和文中提到的离线写磁盘操作有何区别？
2.响应一次SQL我理解是要同时操作两个日志文件？也就是写磁盘两次？

作者回复：你的理解是对的。
1. 写redo log是顺序写，不用去“找位置”，而更新数据需要找位置

2. 其实是3次（redolog两次 binlog 1次）。不过在并发更新的时候会合并写

## 9.刷脏会导致select查询变慢

> 刷脏会导致select查询变慢。
>
> 先分析下redo log再哪些场景会刷到磁盘。
>
> 场景1：redo log写满了，此时MySQL会停止所有更新操作，把脏页刷到磁盘
> 场景2：系统内存不足，需要将脏页淘汰，此时会把脏页刷到磁盘
> 场景3：系统空闲时，MySQL定期将脏页刷到磁盘

先说下阅读的收获：

1、更新的流程先写redo日志，写完后更新内存，到这里操作就直接返回了。后续的流程是生成此操作的binlog，然后写到磁盘
2、redo日志是存储引擎实现的，记录的在某个数据页做了什么修改，固定大小，默认为4GB，可以循环写，解决了每次更新操作写磁盘、查找记录、然后更新整个过程效率低下的问题，redo日志将磁盘的随机写变成了顺序写，这个机制是WAL，先写日志再刷磁盘。一句话，redo日志保证了事务ACID的特性

3、binlog日志Server层实现的，记录的是语句的原始逻辑，比如给ID=1的行的状态设置成1，追加写，上个文件写完回切换成下一个文件，类似滚动日志
4、两阶段提交，是为了解决redo log和binlog不一致问题的，这里的不一致是指redo log写成功binlog写失败了，由于恢复是根据binlog恢复的，这样恢复的时候就会少一条更新操作，导致和线上库的数据不一致。具体而言，两阶段是指prepare阶段和commit阶段，写完redo log进入prepare阶段，写完binlog进入commit阶段。
然后说下由redo log联想到之前遇到的一个问题：一个普通的select查询超过30ms，经过和DBA的联合排查，确认是由于MySQL“刷脏”导致的。
所谓刷脏就是由于内存页和磁盘数据不一致导致了该内存页是“脏页”，将内存页数据刷到磁盘的操作称为“刷脏”。刷脏是为了避免产生“脏页”，主要是因为MySQL更新先写redo log再定期批量刷到磁盘的，这就导致内存页的数据和磁盘数据不一致，为了搞清楚为什么“刷脏”会导致慢查，我们先分析下redo log再哪些场景会刷到磁盘。
场景1：redo log写满了，此时MySQL会停止所有更新操作，把脏页刷到磁盘
场景2：系统内存不足，需要将脏页淘汰，此时会把脏页刷到磁盘
场景3：系统空闲时，MySQL定期将脏页刷到磁盘

可以想到，在场景1和2都会导致慢查的产生，根据文章提到的，redo log是可以循环写的，那么即使写满了应该也不会停止所有更新操作吧，其实是会的，文中有句话“粉板写满了，掌柜只能停下手中的活，把粉板的一部分赊账记录更新到账本中，把这些记录从粉板删除，为粉板腾出新的空间”，这就意味着写满后是会阻塞一段时间的。

那么问题来了，innodb存储引擎的刷脏策略是怎么样的呢？通常而言会有两种策略：全量（sharp checkpoint）和部分（fuzzy checkpoint）。全量刷脏发生在关闭数据库时，部分刷脏发生在运行时。部分刷脏又分为定期刷脏、最近最少使用刷脏、异步/同步刷脏、脏页过多刷脏。暂时先写到这，后面打算写文详细介绍。

作者回复：