[转]MySQL日志——Undo | Redo

本文是介绍MySQL数据库InnoDB存储引擎重做日志漫游

00 – Undo Log

Undo Log 是为了实现事务的原子性，在MySQL数据库InnoDB存储引擎中，还用Undo Log来实现多版本并发控制(简称：MVCC)。

- 事务的原子性**(Atomicity)**

事务中的所有操作，要么全部完成，要么不做任何操作，不能只做部分操作。如果在执行的过程中发生了错误，要回滚(Rollback)到事务开始前的状态，就像这个事务从来没有执行过。

- 原理

Undo Log的原理很简单，为了满足事务的原子性，在操作任何数据之前，首先将数据备份到一个地方（这个存储数据备份的地方称为Undo Log）。然后进行数据的修改。如果出现了错误或者用户执行了ROLLBACK语句，系统可以利用Undo Log中的备份将数据恢复到事务开始之前的状态。

除了可以保证事务的原子性，Undo Log也可以用来辅助完成事务的持久化。

- 事务的持久性**(Durability)**

事务一旦完成，该事务对数据库所做的所有修改都会持久的保存到数据库中。为了保证持久性，数据库系统会将修改后的数据完全的记录到持久的存储上。

- 用**Undo Log**实现原子性和持久化的事务的简化过程

假设有A、B两个数据，值分别为1,2。
A.事务开始.
B.记录A=1到undo log.
C.修改A=3.
D.记录B=2到undo log.
E.修改B=4.
F.将undo log写到磁盘。
G.将数据写到磁盘。
H.事务提交

这里有一个隐含的前提条件：‘数据都是先读到内存中，然后修改内存中的数据，最后将数据写回磁盘’。

之所以能同时保证原子性和持久化，是因为以下特点：
A. 更新数据前记录Undo log。
B. 为了保证持久性，必须将数据在事务提交前写到磁盘。只要事务成功提交，数据必然已经持久化。
C. Undo log必须先于数据持久化到磁盘。如果在G,H之间系统崩溃，undo log是完整的，可以用来回滚事务。
D. 如果在A-F之间系统崩溃,因为数据没有持久化到磁盘。所以磁盘上的数据还是保持在事务开始前的状态。

缺陷：每个事务提交前将数据和Undo Log写入磁盘，这样会导致大量的磁盘IO，因此性能很低。

如果能够将数据缓存一段时间，就能减少IO提高性能。但是这样就会丧失事务的持久性。因此引入了另外一种机制来实现持久化，即Redo Log.

01 – Redo Log

- 原理

和Undo Log相反，Redo Log记录的是新数据的备份。在事务提交前，只要将Redo Log持久化即可，不需要将数据持久化。当系统崩溃时，虽然数据没有持久化，但是Redo Log已经持久化。系统可以根据Redo Log的内容，将所有数据恢复到最新的状态。

- **Undo + Redo**事务的简化过程

假设有A、B两个数据，值分别为1,2.
A.事务开始.
B.记录A=1到undo log.
C.修改A=3.
D.记录A=3到redo log.
E.记录B=2到undo log.
F.修改B=4.
G.记录B=4到redo log.
H.将redo log写入磁盘。
I.事务提交

- **Undo + Redo**事务的特点

A. 为了保证持久性，必须在事务提交前将Redo Log持久化。
B. 数据不需要在事务提交前写入磁盘，而是缓存在内存中。
C. Redo Log 保证事务的持久性。
D. Undo Log 保证事务的原子性。
E. 有一个隐含的特点，数据必须要晚于redo log写入持久存储。

- **IO**性能

Undo + Redo的设计主要考虑的是提升IO性能。虽说通过缓存数据，减少了写数据的IO.
但是却引入了新的IO，即写Redo Log的IO。如果Redo Log的IO性能不好，就不能起到提高性能的目的。
为了保证Redo Log能够有比较好的IO性能，InnoDB 的 Redo Log的设计有以下几个特点：

A. 尽量保持Redo Log存储在一段连续的空间上。因此在系统第一次启动时就会将日志文件的空间完全分配。以顺序追加的方式记录Redo Log,通过顺序IO来改善性能。
B. 批量写入日志。日志并不是直接写入文件，而是先写入redo log buffer.当需要将日志刷新到磁盘时

(如事务提交),将许多日志一起写入磁盘.
C. 并发的事务共享Redo Log的存储空间，它们的Redo Log按语句的执行顺序，依次交替的记录在一起，以减少日志占用的空间。例如,Redo Log中的记录内容可能是这样的：
记录1: <trx1, insert …>
记录2: <trx2, update …>
记录3: <trx1, delete …>
记录4: <trx3, update …>
记录5: <trx2, insert …>
D. 因为C的原因,当一个事务将Redo Log写入磁盘时，也会将其他未提交的事务的日志写入磁盘。
E. Redo Log上只进行顺序追加的操作，当一个事务需要回滚时，它的Redo Log记录也不会从Redo Log中删除掉。

02 – 恢复(Recovery)

- 恢复策略
前面说到未提交的事务和回滚了的事务也会记录Redo Log，因此在进行恢复时,这些事务要进行特殊的的处理.有2中不同的恢复策略：

A. 进行恢复时，只重做已经提交了的事务。
B. 进行恢复时，重做所有事务包括未提交的事务和回滚了的事务。然后通过Undo Log回滚那些未提交的事务。

- InnoDB存储引擎的恢复机制
MySQL数据库InnoDB存储引擎使用了B策略, InnoDB存储引擎中的恢复机制有几个特点：

A. 在重做Redo Log时，并不关心事务性。 恢复时，没有BEGIN，也没有COMMIT,ROLLBACK的行为。也不关心每个日志是哪个事务的。尽管事务ID等事务相关的内容会记入Redo Log，这些内容只是被当作要操作的数据的一部分。
B. 使用B策略就必须要将Undo Log持久化，而且必须要在写Redo Log之前将对应的Undo Log写入磁盘。Undo和Redo Log的这种关联，使得持久化变得复杂起来。为了降低复杂度，InnoDB将Undo Log看作数据，因此记录Undo Log的操作也会记录到redo log中。这样undo log就可以象数据一样缓存起来，而不用在redo log之前写入磁盘了。
包含Undo Log操作的Redo Log，看起来是这样的：
记录1: <trx1, **Undo log insert** <undo_insert …>>
记录2: <trx1, insert …>
记录3: <trx2, **Undo log insert** <undo_update …>>
记录4: <trx2, update …>
记录5: <trx3, **Undo log insert** <undo_delete …>>
记录6: <trx3, delete …>
C. 到这里，还有一个问题没有弄清楚。既然Redo没有事务性，那岂不是会重新执行被回滚了的事务？确实是这样。同时Innodb也会将事务回滚时的操作也记录到redo log中。回滚操作本质上也是对数据进行修改，因此回滚时对数据的操作也会记录到Redo Log中。
一个回滚了的事务的Redo Log，看起来是这样的：
记录1: <trx1, Undo log insert <undo_insert …>>
记录2: <trx1, **insert A**…>
记录3: <trx1, Undo log insert <undo_update …>>
记录4: <trx1, **update B**…>
记录5: <trx1, Undo log insert <undo_delete …>>
记录6: <trx1, **delete C**…>
记录7: <trx1, **insert C**>
记录8: <trx1, **update B** to old value>
记录9: <trx1, **delete A**>
一个被回滚了的事务在恢复时的操作就是先redo再undo，因此不会破坏数据的一致性.

- InnoDB存储引擎中相关的函数
Redo: recv_recovery_from_checkpoint_start()
Undo: recv_recovery_rollback_active()
Undo Log的Redo Log: trx_undof_page_add_undo_rec_log()

转载至:http://www.mysqlops.com/2012/04/06/innodb-log1.html

_Boz
关注 - 21
粉丝 - 135
+加关注

1 1

« 上一篇：[笔记]MySQL 配置优化
» 下一篇：[转]MySQL索引详解(1)

posted @ 2013-03-18 17:46 _Boz 阅读(7373) 评论(0) 编辑 收藏

最新**IT**新闻:
· 滴滴加速出租车网约车融合 合作出租车企业超150家
· 特朗普当选总统后，硅谷却哭成一片
· 大众点评张涛：以往成功的经验，可能下一步就成为了你失败的根源
· 川普赢了美国大选，来VR里体验一下他的生活吧
· 苹果开始卖翻新iPhone了，最多便宜120美元
» 更多新闻...

最新知识库文章:
· 循序渐进地代码重构
· 技术的正宗与野路子
· 陈皓：什么是工程师文化？
· 没那么难，谈CSS的设计模式
· 程序猿媳妇儿注意事项
» 更多知识库文章...

--shawnye

5. Re:[原] KVM虚拟机网络闪断分析

@felix021哈哈，谢谢，我们认识，以前在segmentfault讨论过Python字节码优化问题...

--_Boz

阅读排行榜

评论排行榜

推荐排行榜

4AI?Z:cp1z?_RJQle1]Gs;P!T)RHroW|