19 数据库忽然断电会丢数据吗？

更新时间：2019-09-17 09:51:50

> 读书给人以快乐、给人以光彩、给人以才干。
>
> ——培根

在上一章，我讲解了 MySQL 锁的相关内容。主要谈到了全局锁、表锁、行锁以及死锁等。通过这些学习，相信我们可以理解锁的原理，并在工作中降低锁冲突的概率。这也是优化数据库必须掌握的知识点。

从本节开始，将进入一个新的 MySQL 知识大类：MySQL 事务。

1 什么是事务？

根据《高性能 MySQL》第 3 版 1.3 事务一节中定义：

> 事务就是一组原子性的 SQL 查询，或者说一个独立的工作单元。如果数据库引擎能够成功地对数据库应用该组查询的全部语句，那么就执行该组查询。如果其中有任何一条语句因为崩溃或其他原因无法执行，那么所有的语句都不会执行。也就是说，事务内的语句，要么全部执行成功，要么全部执行失败。

看上面的文字可以稍微抽象了一点，可以结合生活中的一个例子：

比如你给朋友转账 100 元，其大致过程是：从你的账户扣除 100 元，然后再到你朋友的账户中增加 100 元，试想，如果在这中间，因为网络问题或者程序问题，导致在你的账户中扣除了，但是没有在你朋友的账户中增加，那岂不是乱套了。

所以，类似这种情况，就可以把这两个步骤放到一个事务里面。要么全部成功，也就是从你的账户扣除之后，然后在你朋友账户中新增；要么全部失败，比如在中间出现问题，会回滚这中间所有的变更。大致操作步骤如下表：

步骤	对应的语句

步骤	对应的语句
开始一个事务	begin;
从你的账户扣除 100 元	update money_info set balance = balance - 100 where user_id=1;
在你朋友的账户中增加 100 元	update money_info set balance = balance + 100 where user_id=2;
事务结束	commit;

一个良好的事务处理系统，必须具备 ACID 特性：

- atomicity（原子性）：要么全执行，要么全都不执行；
- consistency（一致性）：在事务开始和完成时，数据都必须保持一致状态；
- isolation（隔离性）：事务处理过程中的中间状态对外部是不可见的；
- durability（持久性）：事务完成之后，它对于数据的修改是永久性的。

InnoDB 采用 redo log 机制来保证事务更新的一致性和持久性。什么是 redo log？下面来一起看下：

2 Redo log

Redo log 称为重做日志，用于记录事务操作变化，记录的是数据被修改之后的值。

Redo log 由两部分组成：

- 内存中的重做日志缓冲（redo log buffer）
- 重做日志文件（redo log file）

每次数据更新会先更新 redo log buffer，然后根据 innodb_flush_log_at_trx_commit 来控制 redo log buffer 更新到 redo log file 的时机。innodb_flush_log_at_trx_commit 有三个值可选：

0：事务提交时，在事务提交时，每秒触发一次 redo log buffer 写磁盘操作，并调用操作系统 fsync 刷新 IO 缓存。

1：事务提交时，InnoDB 立即将缓存中的 redo 日志写到日志文件中，并调用操作系统 fsync 刷新 IO 缓存；

2：事务提交时，InnoDB 立即将缓存中的 redo 日志写到日志文件中，但不是马上调用 fsync 刷新 IO 缓存，而是每秒只做一次磁盘 IO 缓存刷新操作。

innodb_flush_log_at_trx_commit 参数的默认值是 1，也就是每个事务提交的时候都会从 log buffer 写更新记录到日志文件，而且会刷新磁盘缓存，这完全满足事务持久化的要求，是最安全的，但是这样会有比较大的性能损失。

将参数设置为 0 时，如果数据库崩溃，最后 1秒钟的 redo log 可能会由于未及时写入磁盘文件而丢失，这种方式尽管效率最高，但是最不安全。

将参数设置为 2 时，如果数据库崩溃，由于已经执行了重做日志写入磁盘的操作，只是没有做磁盘 IO 刷新操作，因此，只要不发生操作系统奔溃，数据就不会丢失，这种方式是对性能和安全的一种折中处理。

3 Binlog

二进制日志（binlog）记录了所有的 DDL（数据定义语句）和 DML（数据操纵语句），但是不包括 select 和 show 这类操作。Binlog 有以下几个作用：

- 恢复：数据恢复时可以使用二进制日志
- 复制：通过传输二进制日志到从库，然后进行恢复，以实现主从同步
- 审计：可以通过二进制日志进行审计数据的变更操作

可以通过参数 sync_binlog 来控制累积多少个事务后才将二进制日志 fsync 到磁盘。

- sync_binlog=0，表示每次提交事务都只write，不fsync
- sync_binlog=1，表示每次提交事务都会执行fsync
- sync_binlog=N(N>1)，表示每次提交事务都write，累积N个事务后才fsync

比如要加快写入数据的速度或者机器磁盘 IO 瓶颈时，可以将 sync_binlog 设置成大于 1 的值，**但是如果设置为N(N>1)时，如果数据库崩溃，可能会丢失最近 N 个事务的 binlog**。

4 怎样确保数据库突然断电不丢数据？

通过上面的讲解，只要 innodb_flush_log_at_trx_commit 和 sync_binlog 都为 1（通常称为：双一），就能确保 MySQL 机器断电重启后，数据不丢失。

因此建议在比较重要的库，比如涉及到钱的库，设置为双一，而你的测试环境或者正式业务不那么重要的库（比如日志库）可以将 innodb_flush_log_at_trx_commit 设置为0，sync_binlog 设置成大于100 的数值，提高更新效率。

5 总结

本节讲解了什么是事务？

所谓事务：是指一组原子性的 SQL 查询，事务里的 SQL 要么全部执行成功，要么全部执行失败。

一个良好的事务处理系统，必须具备 ACID 特性： atomicity（原子性）、consistency（一致性）、 isolation（隔离性）、 durability（持久性）。

另外讲解了 Redo log 和 Binlog：

- Redo log：称为重做日志，用于记录事务操作变化，记录的是数据被修改之后的值
- Binlog：记录了所有变更操作，其作用有：恢复、复制、审计等

如果想要数据库达到最安全的状态，可以将 innodb_flush_log_at_trx_commit 和 sync_binlog 都设置为 1。

6 问题

你工作中有遇到过丢数据的场景吗？是什么原因导致的呢？

7 参考资料

《高性能 MySQL》第 3 版：1.3 事务

《MySQL 技术内幕》第 2 版：7.2 事务的实现

}

18 为什么会出现死锁？

20 MVCC怎么实现的？