MySQL日志系统：持久性和一致性是如何实现的？

原创 T Ti 笔记 2025年02月28日 17:37 广东

本文以更新语句为例，介绍MySQL是如何通过日志系统确保更新语句执行的**持久性**和**一致性**。核心内容包含**Redo Log 、 Binlog和2PC**。

一、SQL执行过程回顾

在分析日志系统是如何实现**持久性**和**一致性**之前，可以回顾下SQL语句的执行过程。

▲ 深入解析：一条查询语句在 MySQL 中的执行之旅

更新语句的核心流程

1. **连接器**：验证权限，建立连接。
2. **分析器**：解析语法，识别操作类型（如 UPDATE ）。
3. **优化器**：选择最优执行计划（如走哪个索引）。
4. **执行器**：
   - 调用存储引擎接口查找目标数据行（若不在缓冲池，需从磁盘加载）。
   - **执行修改**，触发日志系统的关键动作！

二、两大日志系统：Redo Log 与 Binlog

1. Redo Log（重做日志）

InnoDB引擎特有的日志，主要用于保证事务的持久性和数据的一致性。当数据发生变化时，InnoDB会先记录到redo log中，然后再将数据写入磁盘。这样，即使系统在数据还未完全写入磁盘时就发生了崩溃，也能通过redo log将数据恢复到最近一次一致的状态。它是一种物理日志，记录的是数据页级别的修改操作，比如在某个数据页上做了什么具体的更改。

- **角色**：InnoDB引擎的“应急小本本”，保证崩溃恢复时不丢数据。
- **特点**：
  - 物理日志，记录数据页的修改（如“在表空间xx页偏移yy处写zzz”）。
  - 循环写入，空间固定（如4个1GB文件）。
- **WAL技术（Write-Ahead Logging）**：
  先写日志，再写磁盘。修改数据时，先写redo log到内存缓冲区，后续再异步刷盘。

2. Binlog（归档日志）

MySQL Server层的日志，所有存储引擎都可以使用。它的主要作用是记录数据库的操作语句，以便进行数据的复制和恢复。例如，在主从复制架构中，主服务器会将binlog发送给从服务器，从服务器根据binlog中的语句进行相应的操作，从而实现数据的同步。它是一种逻辑日志，记录的是SQL语句的原始逻辑，比如对某一行数据进行了更新操作，包括更新的字段、更新前后的值等信息。

- **角色**：Server层的“操作记录仪”，用于主从复制、数据恢复。
- **特点**：
  - 逻辑日志，记录语句的原始逻辑（如“给ID=2的账户余额+100”）。
  - 追加写入，文件可无限扩展（需定期归档）。

3. 为什么有两套日志系统？

最初，MySQL自带的引擎是MyISAM，它没有crash-safe的能力，也就是不支持事务的崩溃恢复。当时MySQL引入了binlog日志，但其主要作用仅仅是用于归档，以便于数据的备份和恢复，而无法提供事务的持久性和一致性保障。

后来，InnoDB作为另一个公司开发的存储引擎以插件形式引入MySQL。InnoDB需要实现crash-safe能力，因此它自带了一套redo log日志系统，用于记录数据页的修改等物理操作，以保证即使在系统崩溃的情况下，也能够根据这些日志恢复到一致的状态

三、两阶段提交（2PC）

为确保redo log和binlog的**逻辑一致性**，MySQL采用两阶段提交：

1. **Prepare阶段**：
   - 写redo log，标记为 prepare 状态。
2. **Commit阶段**：
   - 写binlog，提交事务。
   - 将redo log标记为 commit 状态。

为何需要两阶段提交？
如果先写Redo Log再写Binlog，那么在宕机重启后，由于Redo Log中已经记录了该事务的日

志，但Binlog中没有，就会导致主库节点根据Redo Log恢复的数据与从库节点通过Binlog恢复的数据不一致，造成数据不一致的问题。

如果先写Binlog再写Redo Log，那么在宕机重启后，由于Redo Log中没有该事务的日志记录，因此无法通过Redo Log进行数据恢复。而Binlog中虽然有该事务的日志，但由于缺乏Redo Log的支持，也无法完成完整的数据恢复过程，从而导致数据丢失。

四、崩溃恢复过程

数据库重启时，若发现存在 `prepare` 状态的redo log：

1. 检查对应的binlog是否完整。
   - 若binlog完整：提交事务（redo log -> commit）。
   - 若binlog不完整：回滚事务。
2. 通过redo log重放未刷盘的数据页修改。

五、更新语句执行过程举例

以执行下面的更新语句为例

```
UPDATE accounts SET balance=balance+100 WHERE id=2;
```

1. 执行器通过索引找到id=2的行（可能在内存或磁盘）。
2. 修改内存中的数据页，记录redo log（prepare）。
3. 记录binlog：“将id=2的余额+100”。
4. 提交事务，redo log标记为commit。
5. 后续由InnoDB择机将数据页刷盘。

总结

- **Redo Log**：InnoDB引擎特有日志系统，确保崩溃恢复的**持久性**（物理日志）。
- **Binlog**：MySQL Server层日志系统，实现数据归档和主从复制的**一致性**（逻辑日志）。
- **两阶段提交**：两大日志协作的“安全锁”。

往期推荐

▲ 组合模式：构建树形结构的高效设计

▲ 桥接模式：解耦抽象与实现的优雅设计

▲ deepseek本地部署C盘空间占用优化

END

别忘了

点赞、分享、爱心

↓↓↓

MySQL 17 数据库 17

MySQL · 目录

上一篇

深入解析：一条查询语句在 MySQL 中的执行之旅

下一篇

MySQL事务隔离机制是如何实现的？