面试复盘：MySQL InnoDB 事务隔离级别与 MVCC 分析/为什么可重复读的死锁概率高?

Asthenian 2025-03-30 51 阅读6分钟 关注

告别排队，Trae订阅上线，首月Pro仅$3，开发无忧

_

最近一次面试中，面试官让我分析 MySQL InnoDB 的四种事务隔离级别，以及相关的 MVCC（多版本并发控制）和 ReadView 机制。还要求顺带讲讲不同隔离级别解决了哪些问题，以及为什么从 Repeatable Read 切换到 Read Committed 可以降低死锁概率。以下是我的复盘和总结。

一、MySQL InnoDB 的四种事务隔离级别

MySQL InnoDB 支持 SQL 标准定义的四种事务隔离级别，从低到高分别是：

1. **Read Uncommitted（读未提交）**
   - 事务可以读取其他事务尚未提交的数据（即“脏数据”）。
   - 这是隔离级别最低的一种，几乎不提供隔离性，但在实际生产中很少使用。
2. **Read Committed（读已提交）**
   - 事务只能读取其他事务已经提交的数据，解决了“脏读”（Dirty Read）问题。
   - 但仍存在“不可重复读”（Non-repeatable Read）问题，即同一事务内多次读取同一数据时，可能因为其他事务的提交而返回不同的结果。
3. **Repeatable Read（可重复读）**
   - 保证同一事务内多次读取同一数据的结果一致，解决了不可重复读问题。

- 这是 InnoDB 的默认隔离级别。但在理论上仍可能出现“幻读”（Phantom Read），不过 InnoDB 通过 MVCC 和间隙锁（Gap Lock）在大多数场景下避免了幻读。

4. **Serializable（串行化）**

- 最高隔离级别，通过强制事务串行执行，避免所有并发问题（如脏读、不可重复读、幻读）。
- 但性能开销较大，适用于对数据一致性要求极高的场景。

二、MVCC 和 ReadView 的作用

1. MVCC（多版本并发控制）

MVCC 是 InnoDB 实现事务隔离的核心机制。它通过为每行记录维护多个版本（基于事务 ID 和回滚指针），允许多个事务并发访问数据，而无需加锁阻塞。MVCC 的关键点包括：

- **事务 ID（trx_id）**：每开启一个事务，InnoDB 会分配一个唯一的事务 ID。
- **回滚段（Undo Log）**：记录数据的旧版本，用于回滚和多版本读取。
- **版本链**：通过回滚指针（roll_pointer）串联起数据的多个历史版本。

MVCC 的好处是读写分离，读操作无需等待写操作完成，从而提升并发性能。

2. ReadView

ReadView 是 MVCC 的实现基础，用于决定事务能看到哪些数据版本。ReadView 包含以下关键信息：

- **活跃事务列表**：记录当前未提交的事务 ID 集合。
- **up_limit_id**：活跃事务列表中最小的事务 ID，表示“最早未提交的事务”。
- **low_limit_id**：当前系统最大事务 ID + 1，表示“未来事务的起点”。
- **创建时间（trx_id）**：ReadView 生成时的事务 ID。

数据可见性规则（以下是对规则的详细解释）：
当一个事务通过 ReadView 判断某行数据的版本（由 `trx_id` 表示）是否可见时，会按以下规则进行：

- **如果数据的 `trx_id < up_limit_id`**：
  说明这个数据版本是在所有当前活跃事务开始之前就已经提交的，肯定是可见的。例如，`up_limit_id = 100`，而数据 `trx_id = 90`，表示这个数据在事务 100 开始前已提交，对当前事务可见。
- **如果数据的 `trx_id` 在活跃事务列表中**：
  说明这个数据版本是由一个尚未提交的事务创建的，因此不可见。例如，活跃事务列表中有事务 105，而数据 `trx_id = 105`，表示数据还未提交，不能被当前事务看到。
- **如果数据的 `trx_id >= low_limit_id`**：
  说明这个数据版本是在 ReadView 创建之后才由新事务生成的（即“未来的数据”），因此不可见。例如，`low_limit_id = 110`，而数据 `trx_id = 115`，表示数据是在当前事务视图生成后才提交的，对当前事务不可见。

此外，如果 `trx_id >= up_limit_id` 但不在活跃事务列表中，且 `< low_limit_id`，则表示数据是在 ReadView 创建前提交的，也可见。

在 **Read Committed** 中，每次 SELECT 都会生成一个新的 ReadView，因此能看到最新的已提交数据。而在 **Repeatable Read** 中，整个事务只生成一次 ReadView，保证了数据的一致性。

三、事务隔离级别解决的问题

1. **Read Uncommitted**
   - 问题：脏读（读取未提交数据，可能被回滚）。
   - 未解决：不可重复读、幻读。
2. **Read Committed**
   - 解决：脏读。
   - 未解决：不可重复读（同一事务内数据可能变化）、幻读。
3. **Repeatable Read**
   - 解决：脏读、不可重复读。
   - 部分解决：幻读（通过 MVCC 和间隙锁在大多数情况下避免，但特定场景仍可能发生）。
4. **Serializable**

- 解决：脏读、不可重复读、幻读。
- 代价：并发性能显著下降。

四、为什么从 Repeatable Read 切换到 Read Committed 能降低死锁概率?

1. Repeatable Read 的锁机制

在 Repeatable Read 下，InnoDB 使用 MVCC 结合间隙锁（Gap Lock）和下一键锁（Next-Key Lock）来防止幻读。例如：

- 当执行 `SELECT ... FOR UPDATE` 或更新操作时，会锁住索引范围（包括间隙）。
- 如果多个事务同时尝试操作相邻的数据范围，可能导致锁冲突，进而引发死锁。

例如，事务 A 更新 `id=5` 的记录并锁住 `(3, 5]`，事务 B 更新 `id=4` 并试图锁住 `(1, 4]`，两个事务互相等待对方的锁释放，形成死锁。

2. Read Committed 的锁机制

在 Read Committed 下：

- 不使用间隙锁，仅对当前记录加行锁。
- 每次读取都会生成新的 ReadView，数据视图更“实时”，但牺牲了重复读的保证。
- 由于锁的范围更小（仅限单行，不涉及间隙），锁冲突和死锁的概率显著降低。

3. 为什么死锁概率降低?

- **锁粒度减小**：Repeatable Read 的间隙锁范围较大，容易与其他事务的锁重叠；而 Read Committed 只锁单行，冲突概率低。
- **锁持有时间缩短**：Read Committed 不需要维持整个事务的视图一致性，锁释放更快，减少了锁竞争窗口。

4. 权衡

从 Repeatable Read 切换到 Read Committed 虽然降低了死锁概率，但会导致不可重复读问题。如果业务对一致性要求不高（如实时统计场景），这种切换是合理的优化选择。

五、总结

这次面试让我深刻理解了 InnoDB 事务隔离级别的实现原理。MVCC 和 ReadView 是解决并发读写问题的核心，而不同隔离级别则是对一致性和性能的权衡。Repeatable Read 到 Read Committed 的切换本质上是牺牲一致性换取更高的并发性和更低的死锁风险。复盘下来，我对这些知识点的掌握更扎实了，也提醒自己在实际开发中要根据业务需求选择合适的隔离级别。

标签： 后端

本文收录于以下专栏

MYSQL面试 专栏目录

MYSQL面试

12 订阅 · 66 篇文章

订阅

上一篇 如何排查InnoDB的MySQL服...

下一篇 SQL执行顺序与ON vs WHERE...

评论 0

登录 / 注册 即可发布评论！

暂无评论数据

目录

收起

相关推荐

2.动手实践整洁架构 - 分层架构存在哪些问题
462阅读 · 5点赞

Spring 依赖注入方式及原理
103阅读 · 1点赞

深入SpringBoot启动流程：自动配置与Bean生命周期核心解析
94阅读 · 1点赞

开启多线程异步执行踩得坑
379阅读 · 0点赞

为你推荐

一篇文章图解MVCC!
是fancy呀 3年前 2.0k 6 评论 数据库

为什么说MVCC无法彻底解决幻读的问题?
Asthenian 1月前 117 5 评论 后端

如何修改 MySQL 的数据库隔离级别：命令global、session/my.cnf中修改

Asthenian 2月前 111 点赞 评论 后端

MySQL 8.0 MVCC 源码解析

程序员囧辉 4年前 2.6k 38 14 面试

【MySQL】一文看懂MySQL所有常见问题

Henry游戏开发 1年前 321 4 评论 后端 数据库 MySQL

数据库面试经验分享：MVCC与MySQL锁机制的深度剖析

Asthenian 2月前 115 点赞 评论 后端

深入理解数据库中的 MVCC

Asthenian 28天前 126 点赞 评论 后端

RR有幻读问题吗？MVCC能否解决幻读?

Java中文社群 1年前 1.9k 15 3 后端 面试 Java

什么是MVCC机制?

AmbitionsZoe 3年前 1.4k 5 1 后端 MVC

美团面试官：可重复读隔离级别实现原理是什么？（一文搞懂MVCC机制）

看点代码再上班 1年前 1.8k 25 评论 后端 数据库 MySQL

面试官：事务隔离级别和锁有什么关系（上）｜8月更文挑战

切图老司机 3年前 532 9 评论 数据库 程序员

面试必问：3分钟掌握mysql-mvcc底层原理

想打游戏的程序猿 10月前 3.9k 22 评论 后端 MySQL

全方位解析 MySQL 及相关面试题二（收藏点赞系列）

coderxdh 3年前 333 1 评论 MySQL 面试

MySQL 如何解决幻读（MVCC 原理分析）

LBXX 3年前 6.7k 34 6 MySQL

MySQL并发知识（面试高频）

MrBetter 1年前 971 9 1 MySQL