链滴

关于 Mysql 四种隔离级别中 Lock 和 MVCC 的关系

作者：Zcoin

原文链接：https://ld246.com/article/1603546902776

来源网站：链滴

读写锁

共享锁（share lock） | 读锁（read lock）

读锁是共享的，或者说是相互不阻塞的。多个客户在同一时刻可以同时读取同一个资源，而互不干扰

SELECT ... LOCK IN SHARE MODE

排他锁（exclusive lock） | 写锁（write locl）

写锁则是具有排他性，也就是说一个写锁会阻塞其他的写锁和读锁，只有这样，才能确保在给定的时里，只有一个用户能执行写入，并防止其他用户读取正在写入的同一资源

SELECT ... FOR UPDATE

锁粒度

表锁（table lock）

对整张表进行控制

行级锁（row lock）

对一行数据进行控制

多版本并发控制（MVVC）

大多数事物行存储引擎实现的都不是简单的行级锁。基于提神并发性能的考虑，它们一般都同时实现多版本并发控制。

MVCC的实现，是通过保存数据在某一时间点的快照来实现。也就是说，不管需要执行多长时间，每事物看到的数据都是一致的。根据事物开始的时间不同，每个事物对同一张表，同时刻看到的数据可是不一致的。

Msql InnoDB的MVCC，是通过在每行记录后面保存两个隐藏的列来实现。这两个列，一个保存列行创建时间，一个保存行的过期时间（删除时间）。当然存储的并不是实际的时间值，而是系统版本号每开始一个新的事物，系统版本号都会自动递增。事物开始时刻的系统版本号会作为事物的版本号，来查询到每行记录的版本号进行比较。

SELECT

InnoDB会根据以下两个条件检查每行记录：

1. InnoDB只查找版本早于当前事物版本的数据行（也就是，行的系统版本号小于或等于事物的系统本号），这样可以确保事物读取的行，要么是在事物开始前已存在的，要么是事物自身插入或者修改的。

2. 行的删除版本要么未定义，要么大于当前事物版本号。这可以确保事物读取到的行，在事物开始之未被删除。

只有符合上述两个条件的记录，才能返回作为查询结果

INSERT

- InnoDB为新插入的每一行保存当前系统版本号作为行版本号

DELETE

- InnoDB为删除的每一行保存当前系统版本号作为行删除标识

UPDATE

- InnoDB为插入一行新记录，保存当前系统版本号作为行版本号，同时保存当前系统版本号到原来行作为行删除标识

总结

1. 保存这两个额外系统版本号，使大多数读操作都可以不用加锁。这样设计使得读数据操作很简单，能很好，并且也能够保证只会读取到符合标准的行，不足之处是每行都要额外的储存空间，需要做更的行检查工作，以及额外的维护工作。

2. MVCC只在REPEATABLE READ 和 READ COMMITTED两个隔离级别下工作。其他两个隔离级别和MVCC不兼容，因为READ UNCOMMITTED 总是读取最新的数据行，而不是符合当前事物版本的据行。而SERIALIZABLE则会对所有读取的行都加锁

隔离级别

READ UNCOMMITTED（未提交读）

事物可以读取未提交的数据（脏读）

READ COMMITTED（提交读）

只能看见已经提交的事物所做的修改（不可重复读）

REPEATABLE READ（可重复度）（默认隔离级别）

当前事物在读取某个范围内的记录时，另外一个事物又在该范围内插入了新的记录，当前事物再次查时，会产生幻行（幻读）

SERIALIZABLE（可串行化）

在读取的每一行数据上都加锁（排他锁）

隔离级别	脏读	不可重复读	幻读	锁读
READ UNCOMMITTED	YES	YES	YES	

O

READ COMMITTED	NO	YES	YES
REPEATABLE READ	NO	NO	YES
SERIALIZABLE	NO	NO	NO

PS：

乐观锁

乐观锁大多是基于数据版本记录机制实现，一般是给数据库表增加一个"version"字段。读取数据时，此版本号一同读出，之后更新时，对此版本号加一。此时将提交数据的版本数据与数据库表对应记录当前版本信息进行比对，如果提交的数据版本号大于数据库表当前版本号，则予以更新，否则认为是期数据。

悲观锁

悲观锁依靠数据库提供的锁机制实现。MySQL中的共享锁和排它锁都是悲观锁。数据库的增删改操默认都会加排他锁，而查询不会加任何锁。

补充

当前读、快照读，record lock(记录锁)、gap lock(间隙锁)、next-key lock

本来只有 SERIALIZABLE 隔离级别才可以解决幻读问题，而实际上由于快照读的特性使可重复读也解了幻读问题。

当前读是因为innodb默认为它加入了间隙锁，防止在事务期间对相关数据集插入记录，从而避免出幻读。

在RR级别下，快照读是通过MVVC(多版本控制)和undo log来实现的，当前读是通过加record lock(录锁)和gap lock(间隙锁)来实现的。如果需要实时显示数据，还是需要通过手动加锁来实现。这个时会使用next-key技术来实现。

在mysql中，提供了两种事务隔离技术，第一个是mvcc，第二个是next-key技术。这个在使用不同语句的时候可以动态选择。不加lock inshare mode之类的快照读就使用mvcc。否则 当前读使用next key。mvcc的优势是不加锁，并发性高。缺点是不是实时数据。next-key的优势是获取实时数据，但需要加锁。

Record lock

单条索引记录上加锁，record lock锁住的永远是索引，而非记录本身，即使该表上没有任何索引，那innodb会在后台创建一个隐藏的聚集主键索引，那么锁住的就是这个隐藏的聚集主键索引。所以说一条sql没有走任何索引时，那么将会在每一条聚集索引后面加X锁，这个类似于表锁，但原理上和表应该是完全不同的。

Gap Lock

在索引记录之间的间隙中加锁，或者是在某一条索引记录之前或者之后加锁，并不包括该索引记录本

。gap lock的机制主要是解决可重复读模式下的幻读问题。

Next-Key Lock

行锁和间隙锁组合起来就叫Next-Key Lock。