链滴

MySQL 事务隔离级别详解

作者：Ethan

原文链接：https://ld246.com/article/1544507264790

来源网站：链滴

一、四大特性

原子性

事务必须是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。通常，与某个事关联的操作具有共同的目标，并且是相互依赖的。如果系统只执行这些操作的一个子集，则可能会破事务的总体目标。原子性消除了系统处理操作子集的可能性。

一致性

事务在完成时，必须使所有的数据都保持一致状态。在相关数据库中，所有规则都必须应用于事的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构（如 B 树索引或双向链表）都须是正确的。某些维护一致性的责任由应用程序开发人员承担，他们必须确保应用程序已强制所有已的完整性约束。例如，当开发用于转帐的应用程序时，应避免在转帐过程中任意移动小数点。

隔离性

由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。事务查看数据时数据所处的状，要么是另一并发事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看中间状的数据。这称为可串行性，因为它能够重新装载起始数据，并且重播一系列事务，以使数据结束时的态与原始事务执行的状态相同。当事务可序列化时将获得最高的隔离级别。在此级别上，从一组可并执行的事务获得的结果与通过连续运行每个事务所获得的结果相同。由于高度隔离会限制可并行执行事务数，所以一些应用程序降低隔离级别以换取更大的吞吐量。

持久性

事务完成之后，它对于系统的影响是永久性的。该修改即使出现致命的系统故障也将一直保持。

二、四种隔离级别

Read Uncommitted（读取未提交内容）

在该隔离级别，所有事务都可以看到其他未提交事务的执行结果。本隔离级别很少用于实际应用，因

它的性能也不比其他级别好多少。读取未提交的数据，也被称之为脏读（Dirty Read）。

Read Committed（读取提交内容）

这是大多数数据库系统的默认隔离级别（但不是MySQL默认的）。它满足了隔离的简单定义：一个务只能看见已经提交事务所做的改变。这种隔离级别会造成所谓的不可重复读（Nonrepeatable Rea），因为同一事务的其他实例在该实例处理其间可能会有新的commit，所以同一select可能返回不结果。

Repeatable Read（可重读）

这是MySQL的默认事务隔离级别，它确保同一事务的多个实例在并发读取数据时，会看到同样的数行。不过理论上，这会导致另一个棘手的问题：幻读 （Phantom Read）。简单的说，幻读指当用户取某一范围的数据行时，另一个事务又在该范围内插入了新行，当用户再读取该范围的数据行时，会现有新的“幻影” 行。InnoDB和Falcon存储引擎通过多版本并发控制（MVCC，Multiversion Concrrency Control）机制解决了该问题。

Serializable（可串行化）

这是最高的隔离级别，它通过强制事务排序，使之不可能相互冲突，从而解决幻读问题。简言之，它在每个读的数据行上加上共享锁。在这个级别，可能导致大量的超时现象和锁竞争。

> 这四种隔离级别采取不同的锁类型来实现，若读取的是同一个数据的话，就容易发生问题。例如：

1. 脏读(Drity Read)：某个事务已更新一份数据，另一个事务在此时读取了同一份数据，由于某些原，前一个RollBack了操作，则后一个事务所读取的数据就会是不正确的。

2. 不可重复读(Non-repeatable read):在一个事务的两次查询之中数据不一致，这可能是两次查询过中间插入了一个事务更新的原有的数据。

3. 幻读(Phantom Read):在一个事务的两次查询中数据笔数不一致，例如有一个事务查询了几列(Row数据，而另一个事务却在此时插入了新的几列数据，先前的事务在接下来的查询中，就会发现有几列据是它先前所没有的。

在MySQL中，**实现了这四种隔离级别**，分别有可能产生问题如下所示：

隔离级别	脏读	不可重复读	幻读
读未提交（Read uncommitted）	V	V	V
读已提交（Read committed）	X	V	V
可重复读（Repeatable read）	X	X	V
可串行化（Serializable）	X	X	X

三、锁

InnoDB引擎的锁机制：**InnoDB支持事务，支持行锁和表锁用的比较多，Myisam不支持事务，只支表锁。**

-

共享锁（又称读锁）、排它锁（又称写锁）

1. 共享锁（S）：允许一个事务去读一行，阻止其他事务获得相同数据集的排他锁。

2. 排他锁（X)：允许获得排他锁的事务更新数据，阻止其他事务取得相同数据集的共享读锁和排他写锁。

3. 意向共享锁（IS）：事务打算给数据行加行共享锁，事务在给一个数据行加共享锁前必须先取得该的IS锁。

4. 意向排他锁（IX）：事务打算给数据行加行排他锁，事务在给一个数据行加排他锁前必须先取得该的IX锁。

共享锁和排他锁都是**行锁**，意向锁都是**表锁**，应用中我们只会使用到共享锁和排他锁，意向锁是**mysq内部使用的，不需要用户干预。**

1. 1） 对于UPDATE、DELETE和INSERT语句，InnoDB会自动给涉及数据集加排他锁（X)；

2） 对于普通SELECT语句，InnoDB不会加任何锁，事务可以通过以下语句显示给记录集加共享锁排他锁。

3） MyISAM在执行查询语句(SELECT)前，会自动给涉及的所有表加读锁，在执行更新操作(UPDAT、DELETE、INSERT等)前，会自动给涉及的表加写锁。

4） InnoDB行锁是通过给索引上的索引项加锁来实现的，因此InnoDB这种行锁实现特点意味着：有通过索引条件检索数据，InnoDB才使用行级锁，否则，InnoDB将使用表锁！

共享锁（S）：SELECT * FROM table_name WHERE ... LOCK IN SHARE MODE。

排他锁（X)：SELECT * FROM table_name WHERE ... FOR UPDATE。

对于锁定行记录后需要进行更新操作的应用，应该使用Select...For update 方式，获取排它锁。（共享锁，在读了之后再写会阻塞，会导致死锁）

-

乐观锁、悲观锁

悲观锁

悲观锁，正如其名，它指的是对数据被外界（包括本系统当前的其他事务，以及来自外部系统的事务理）修改持保守态度，因此，在整个数据处理过程中，将数据处于锁定状态。悲观锁的实现，往往依数据库提供的锁机制（也只有数据库层提供的锁机制才能真正保证数据访问的排他性，否则，即使在系统中实现了加锁机制，也无法保证外部系统不会修改数据）

1）使用悲观锁，我们必须关闭mysql数据库的自动提交属性，采用手动提交事务的方式，因为MySQ默认使用autocommit模式，也就是说，当你执行一个更新操作后，MySQL会立刻将结果进行提交。

2）需要注意的是，在事务中，只有SELECT ... FOR UPDATE 或LOCK IN SHARE MODE 同一笔数据会等待其它事务结束后才执行，一般SELECT ... 则不受此影响。对于UPDATE、DELETE和INSERT语，InnoDB会自动给涉及数据集加排他锁（X)。

3）补充：MySQL select...for update的Row Lock与Table Lock

使用select...for update会把数据给锁住，不过我们需要注意一些锁的级别，MySQL InnoDB默认Row Level Lock，所以只有「明确」地指定主键（或有索引的地方），MySQL 才会执行Row lock (只锁被选取数据)，否则MySQL 将会执行Table Lock (将整个数据表单给锁住)。

乐观锁

乐观锁（ Optimistic Locking ） 相对悲观锁而言，乐观锁假设认为数据一般情况下不会造成冲突，以在数据进行提交更新的时候，才会正式对数据的冲突与否进行检测，如果发现冲突了，则让返回用错误的信息，让用户决定如何去做（**一般是回滚事务**）。那么我们如何实现乐观锁呢，一般来说有以2种方式：

1）.使用数据版本（Version）记录机制实现，这是乐观锁最常用的一种实现方式。何谓数据版本？为数据增加一个版本标识，一般是通过为数据库表增加一个数字类型的 “version” 字段来实现。当取数据时，将version字段的值一同读出，数据每更新一次，对此version值加一。当我们提交更新的候，判断数据库表对应记录的当前版本信息与第一次取出来的version值进行比对，如果数据库表当版本号与第一次取出来的version值相等，则予以更新，否则认为是过期数据。

2）.乐观锁定的第二种实现方式和第一种差不多，同样是在需要乐观锁控制的table中增加一个字段，称无所谓，字段类型使用时间戳（timestamp），和上面的version类似，也是在更新提交的时候检查前数据库中数据的时间戳和自己更新前取到的时间戳进行对比，如果一致则OK，否则就是版本冲突。

转载：https://www.cnblogs.com/protected/p/6526857.html