MYSQL MVCC（多版本并发控制）

数据库默认的隔离级别：

Oracle 数据库支持 READ COMMITTED 和 SERIALIZABLE 这两种事务隔离级别
默认系统事务隔离级别是 READ COMMITTED,也就是读已提交

Mysql 默认的事务处理级别是'REPEATABLE-READ',也就是可重复读

MySQL 中的 InnoDB 存储引擎的特性有，默认隔离级别 REPEATABLE READ, 行级锁，实现了 MVCC, Consistent nonlocking read(默认读不加锁，一致性非锁定读), Insert Buffer, Adaptive Hash Index, DoubleWrite, Cluster Index。

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

- MVCC 是被 Mysql 中 事务型存储引擎 InnoDB 所支持的;
- **应对高并发事务，MVCC 比单纯的加锁更高效**;
- MVCC 只在 READ COMMITTED 和 REPEATABLE READ 两个隔离级别下工作;
- MVCC 可以使用 乐观(optimistic)锁 和 悲观(pessimistic)锁来实现;
- 各数据库中 MVCC 实现并不统一

多版本的并发控制**(MVCC)相对于传统的基于锁的并发控制主要特点是读不上锁**，这种特性对于读多写少的场景，大大提高了系统的并发度，因此大部分关系型数据库都实现了 MVCC。

1.2 MVCC 是为了解决什么问题?

大多数的 MYSQL 事务型存储引擎,如,InnoDB，Falcon 以及 PBXT 都不使用一种简单的行锁机制.事实上,他们都和 MVCC－多版本并发控制来一起使用.

大家都应该知道,锁机制可以控制并发操作,但是其系统开销较大,而 **MVCC 可以在大多数情况下代替行级锁,**使用 MVCC,能降低其系统开销.

MVCC 是通过保存数据在某个时间点的快照来实现的. 不同存储引擎的 MVCC. 不同存储引擎的 MVCC 实现是不同的,典型的有乐观并发控制和悲观并发控制.

2.MVCC 具体实现分析

下面,我们通过 InnoDB 的 MVCC 实现来分析 MVCC 使怎样进行并发控制的.
InnoDB 的 MVCC,是**通过在每行记录后面保存两个隐藏的列来实现的,这两个列，分别保存了这个行的创建时间，一个保存的是行的删除时间**。

这里**存储的并不是实际的时间值,而是系统版本号(可以理解为事务的 ID)**，每开始一个新的事务，系统版本号就会自动递增，事务开始时刻的系统版本号会作为事务的 ID.
下面看一下在 REPEATABLE READ 隔离级别下,MVCC 具体是如何操作的.

INSERT

InnoDB 为新插入的每一行保存当前系统版本号作为版本号.

SELECT

InnoDB 会根据以下两个条件检查每行记录:

a. InnoDB 只会查找版本早于当前事务版本的数据行

(也就是,行的系统版本号小于或等于事务的系统版本号)，这样可以确保事务读取的行，要么是在事务开始前已经存在的，要么是事务自身插入或者修改过的.

b. 行的删除版本要么未定义,要么大于当前事务版本号

这可以确保事务读取到的行，在事务开始之前未被删除.

只有 a,b 同时满足的记录，才能返回作为查询结果.

DELETE

InnoDB 会为删除的每一行保存当前系统的版本号(事务的 ID)作为删除标识.

UPDATE

InnoDB 执行 UPDATE，**实际上是新插入了一行记录，并保存其创建时间为当前事务的 ID，同时保存当前事务 ID 到要 UPDATE 的行的删除时间**.