读未提交：

可以读到一个事务没有提交的数据：当事务被回滚的情况发生，就会出现脏读的情况

读提交（read commited，RC）：

这里的读已提交是事务一开始读就不会读到别的事务没有提交的数据，之后如果有其他事务回滚了，那对当前事务的数据没有任何影响，一直都是对的，也就避免了脏读

但是其他事务提交了修改，第二次就会查询到修改的数据，导致前后不一致，也即是出现了不可重复读

读已提交的实现机制就是：若有事务对数据进行UPDATE操作，那么读操作事务要等待这个更新操作事务提交之后才能读取数据（加写锁），也就是解决脏读问题

序列化：即串行读取，很安全但是效率不高

可重复读（repeated read，RR）：mysql默认数据库隔离级别，

普通的查询使用快照读的机制（在第一次创建ReadView后，这个ReadView就会一直维持到事务结束），因此一个事务前后两次查询的结果是一致的，避免了脏读和不可重复读

特别的读已提交与这里不同点就是每次sql语句都会产生新的ReadView，也就是两次查询之间如果有事务提交了，两次查询会看到不同的数据

这里解决脏读的机制就是快照读（MVCC机制-一种解决读-写冲突的无锁并发控制），这种机制避免了使用写锁而导致其他事务不能读的情况，大大提更了并发能力

虽然解决了不可重读问题，但是会出现幻读即在两次读取的前后有数据新增/删除，导致两次出现了不一样的结果

MVCC机制，

多版本并发控制，一种解决读-写冲突的无锁并发控制

每一行都有两个隐藏列，创建版本号和回滚指针，多个 并发的事务同时操作某行，不同的事务对改行的update操作会产生多个版本，然后通过回滚指针组成undo log链，而mvcc机制的快照读正是通过事务id和创建版本号从而实现的快照读，参考链接（https://www.jianshu.com/p/fbec6d1fa16c，https://blog.csdn.net/qq_40194399/article/details/120863119）

特别的：这里多个事务的并发update，在第一个事务提交前，其他事务都会被阻塞，直到前面的事务被提交，这里的undo log链主要是为了回滚和进行当前事务的读

幻读产生的原因

行锁只能锁定已经有的数据，不能锁定刚新加的数据

可重复读的原理（）

给数据库每一行加入系统版本号，每次新增删除修改都会更新这个系统版本号，进行读取的事务也会有版本号，这时候只需要去读版本号小于等于当前版本号的数据即可

其中利用到了快照读，以及当前读，前后读到的是历史数据，后者正在修改的数据可能会被其他事务修改，其实都没有解决幻读问题，最好的方式就只能是串行事务

快照读

针对普通的select语句

执行方式是：生成readView，直接利用MVCC机制来读取，并不会对记录加锁

补充：他是基于多版本并发的MVCC机制，因此快照读读到的数据不一定是最新的数据，也可能是历史的 版本

当前读

针对加锁的语句，通过加锁的方式，使得当前读读过的数据不能被新增修改或者删除

1、select ... lock in share mode 当前读，加读锁，也叫共享锁

2、select ... for update 当前读，加写锁，又叫排它锁

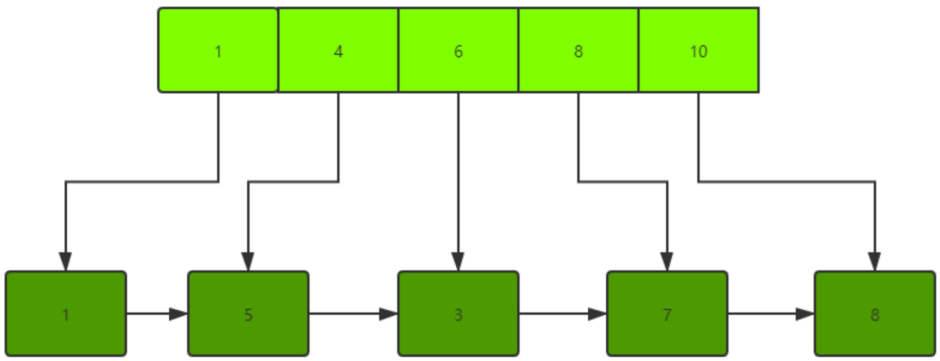
3、innodb里面的update（排它锁），insert（排它锁），delete（排它锁）都会自动给设计的语句加锁

4、串行化的事务的隔离级别

实现方式：next-key（行记录锁+间隙锁）即临键锁，是前后开闭区间

间隙锁

2 索引B结构



begin; select * from z where b = 6 for update;

这条sql语句之后看看我们 需要做什么才能保证不发生幻读。

1不能插入b为6的数据

2不能删除b为6的数据

3不能修改b为6的数据

4不能把别的数据修改为b为6

突然一看挺复杂的，这个锁要怎么加呢，mysql大牛灵机一动，给叶子节点5的next指针加锁，给叶子节点3加行锁，给叶子节点3的next指针加锁。如下图所示

如上：间隙锁要锁的就是可能导致的上述4中情况，主要宗旨就是阻止产生，删除或修改符合查询条件的情况

参考链接：https://www.cnblogs.com/phyger/p/14377651.html

数据库的两段式提交

参考链接：https://blog.csdn.net/qq_33591903/article/details/122030252