幻读和间隙锁（Gap Lock）

幻读

幻读与事务隔离级别有关，在 READ UNCOMMITTED，READ COMMITTED，REPEATABLE READ 这三个事务隔离级别下，若没有其它限制手段，都有可能出现幻读

从官方文档上来看，幻读就是在同一事务中两次 select 返回了不同的数据集合，比如第一次 select 返回了 10 行，第二次同样的 select 却返回了 11 行，而该行数据并不由当前事务所插入

快照读与当前读

InnoDB 通过 undo log 实现的 MVCC 来实现快照读，通俗的来说就是比较数据的事务版本号。快照读又称之为一致性非锁定读，读取的数据并不一定是最新的，但是可以保证在同一个事务内对同一行数据进行多次读取的结果是一样的，以及在范围读取时不会多出几行数据

当前读表示每一次的读取都是数据库中最新的数据，比如 update、insert 以及 delete，都是当前读，即首先获取 X Lock，然后再进行修改、插入和删除动作

一定要注意的是，幻读一定发生在多个事务都执行了"当前读"，包括 insert/update/delete，单一的快照读（select）可通过 MVCC 解决幻读问题

解决幻读

幻读产生的原因就在于另一个事务插入了满足当前事务搜索条件的数据，并且当前事务执行了"当前读"。也就是说，行锁只能锁住某一行或者多行，但是插入数据这个动作，要更新的是数据和数据之间的间隙。因此，InnoDB 为了解决幻读问题，不得不引入新的锁机制，也就是间隙锁（Gap Lock）

间隙锁本质上锁定的是一个范围，当事务 A 对 (X, Y)这一区间添加间隙锁时，事务 B 将无法在该范围内插入数据，必须等待 A 结束

间隙锁

间隙锁和行锁一起形成 Next-Key Lock，专门用于解决 InnoDB RR 隔离级别下的幻读问题。正因为 Next-Key Lock 的存在，锁住的数据行要比 RC 隔离级别下更多，其并发性能会略微降低

Next-Key Lock是前开后闭的区间，Gap Lock是全开区间

间隙锁的冲突关系与 X Lock 的冲突关系是不一样的，间隙锁和间隙锁之间是不会有冲突的，和间隙锁存在冲突关系的是 "往这个间隙中插入一条记录" 这个操作

死锁问题

session A	session B
begin; select * from t where id=9 for update;	
	begin; select * from t where id=9 for update;
	insert into t values(9,9,9); (blocked)
insert into t values(9,9,9); (ERROR 1213 (40001): Deadlock found)	

两个 session 进入互相等待状态，形成死锁。当然，InnoDB 的死锁检测马上就发现了这对死锁关系，让 session A 的 insert 语句报错返回了