三分钟图解事务隔离级别，看一遍就懂

程序员小六

“锁" 是数据库系统区别于文件系统的一个关键特性，其对象是事务，用来锁定的是数据库中的对象，如表、页、行等。锁确实提高了并发性，但是却不可避免地存在一些潜在的并发一致性问题。

不过好在锁只会带来四种问题（丢失更新、脏读、不可重复读、幻读），如果可以防止这四种情况的发生，那将不会产生并发异常。为此，ISO 和 ANIS SQL 标准制定了四种事务隔离级别标准，用来对应地解决锁带来的几种问题。

锁带来的四种并发一致性问题

丢失更新 Last To Modify

丢失更新非常好理解，简单来说其就是一个事务的更新操作会被另一个事务的更新操作所覆盖，从而导致数据的不一致。

举个例子：

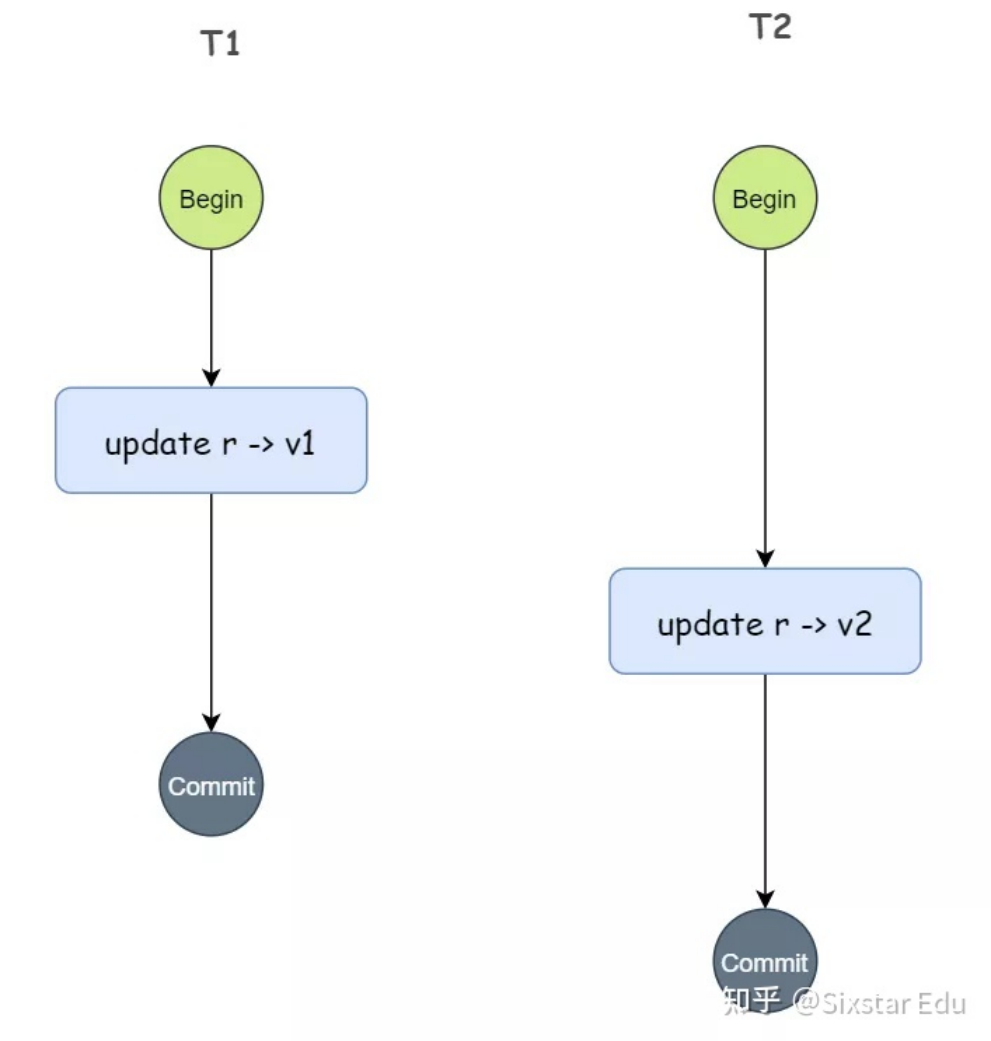
1）事务 T1 将行记录 r 更新为 v1，但是事务 T1 并未提交

2）与此同时，事务 T2 将行记录 r 更新为 v2，事务 T2 未提交

3）事务 T1 提交

4）事务 T2 提交

如下图所示，显然，事务 T1 丢失了自己的修改。



但是，事实上，这种情况准确来讲并不会发生。

因为我们说过对于行进行更新操作的时候，需要对行或其他粗粒度级别的对象加锁，因此当事务 T1 修改行 r 但是没提交的时候，事务 T2 对行 r 进行更新操作的时候是会被阻塞住的，直到事务 T1 提交释放锁。

所以，从数据库层面来讲，数据库本身是可以帮助我们阻止丢失更新问题的发生的。

不过，在真实的开发环境中，我们还经常会遇到逻辑意义上的丢失更新。举个例子：

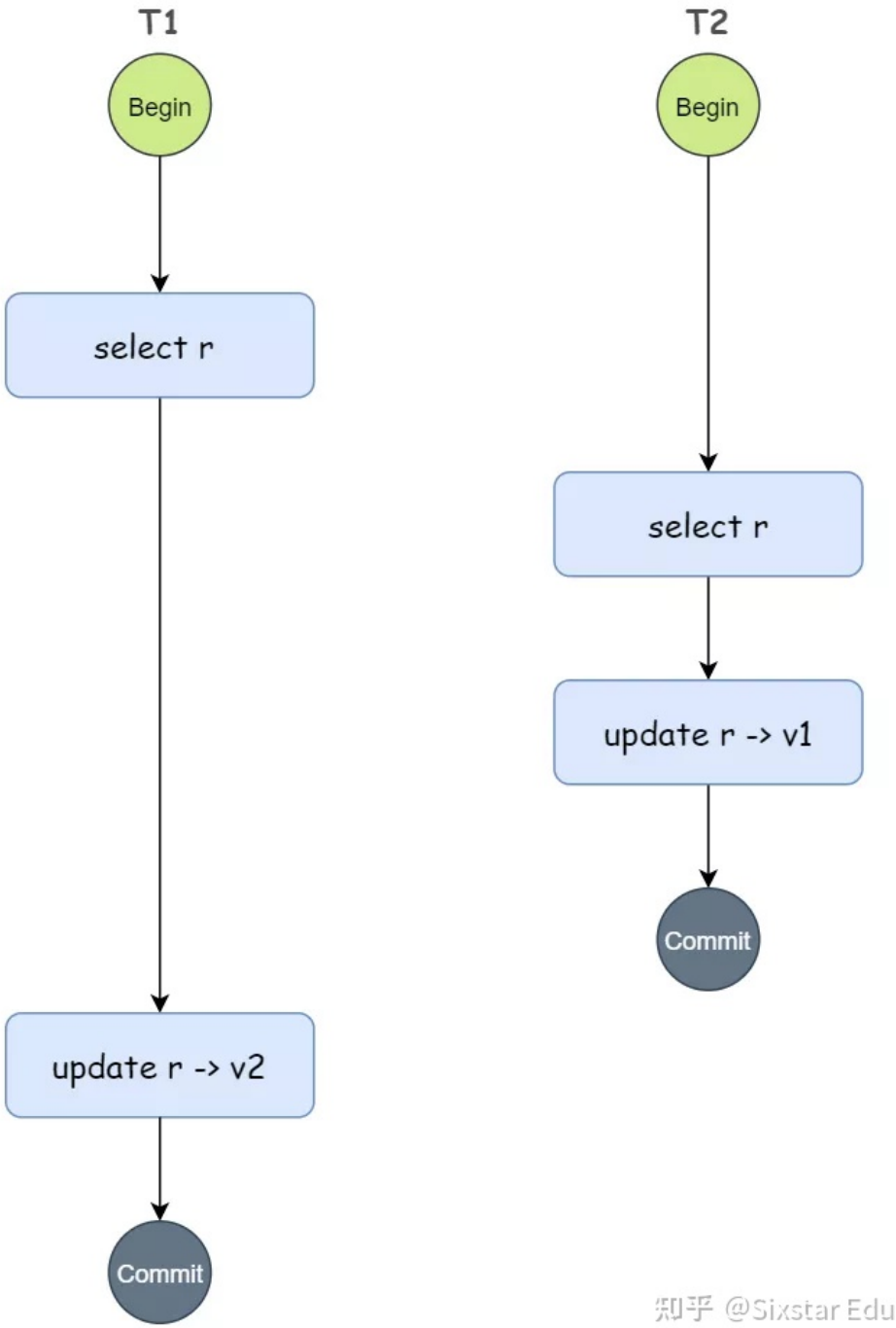
1）事务 T1 查询一行数据 r，放入本地内存，并显示给一个用户 User1

2）事务 T2 也查询该行数据，并将取得的数据显示给另一个用户 User2

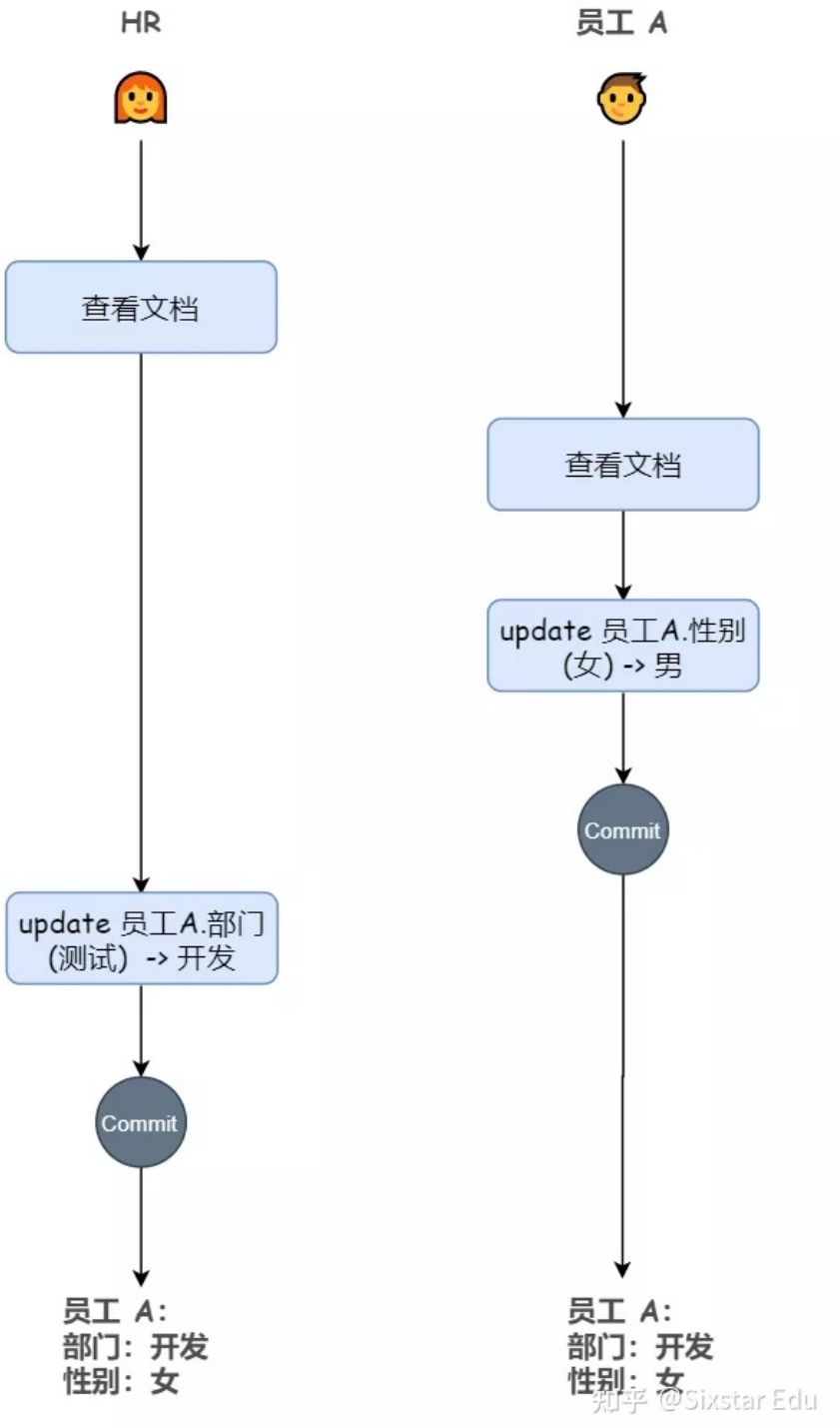
3）User1 修改了行记录 r 为 v1，更新数据库并提交

4）User2 修改了行记录 r 为 v2，更新数据库并提交

显然，最终这行记录的值是 v2，User1 的更新操作被 User2 覆盖掉了，丢失了他的修改。



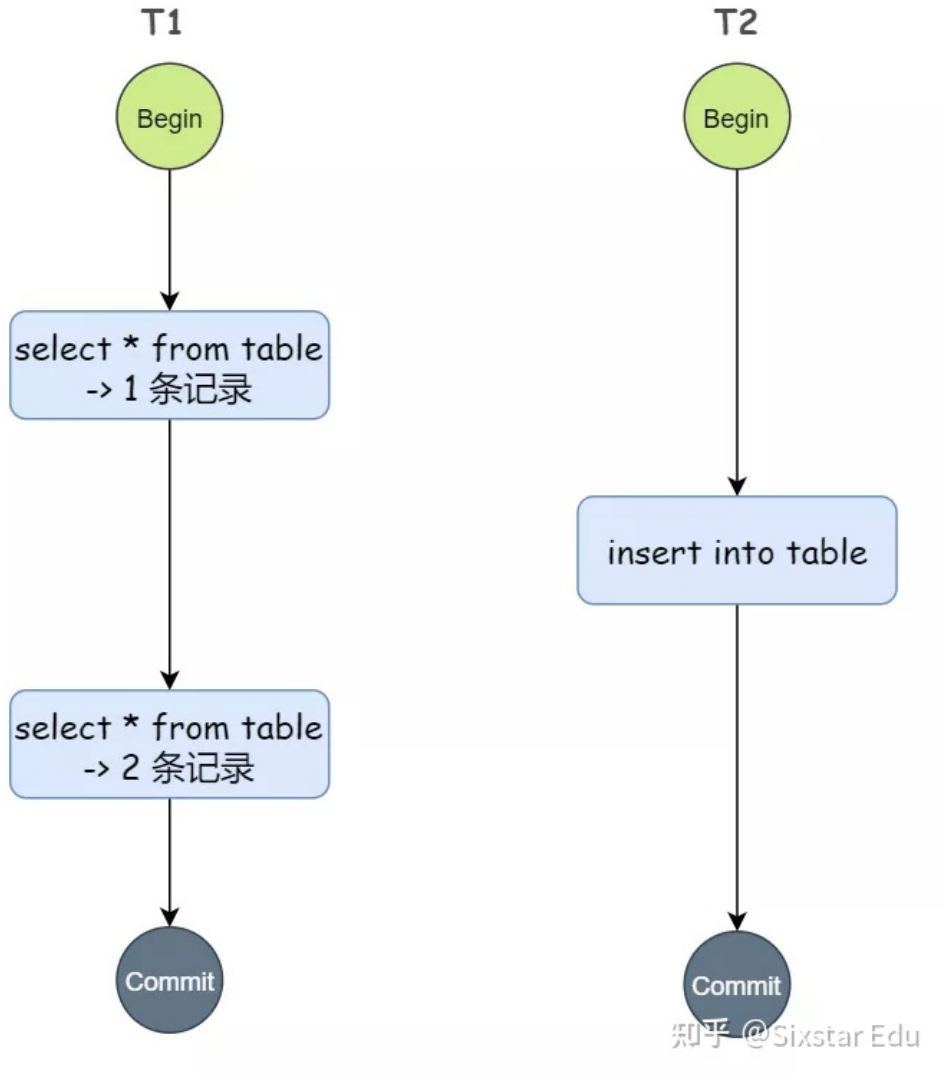
可能还是云里雾里，我来举个更现实点的例子吧，一个部门共同查看一个在线文档，员工 A 发现自己的性别信息有误，于是将其从 “女” 改成了 “男”，就在这时，HR 也发现了员工 A 的部门信息有误，于是将其从 ” 测试 “ 改成了 ” 开发 “，然后，员工 A 和 HR 同时点了提交，但是 HR 的网络稍微慢一点，再次刷新，员工 A 就会发现，擦，我的性别怎么还是 ” 女 “?



脏读 Dirty Read

所谓脏读，就是说一个事务读到了另外一个事务中的 "脏数据"，脏数据就是指事务未提交的数据

如下图所示，在事务并没有提交的前提下，事务 T1 中的两次 SELECT 操作取得了不同的结果：

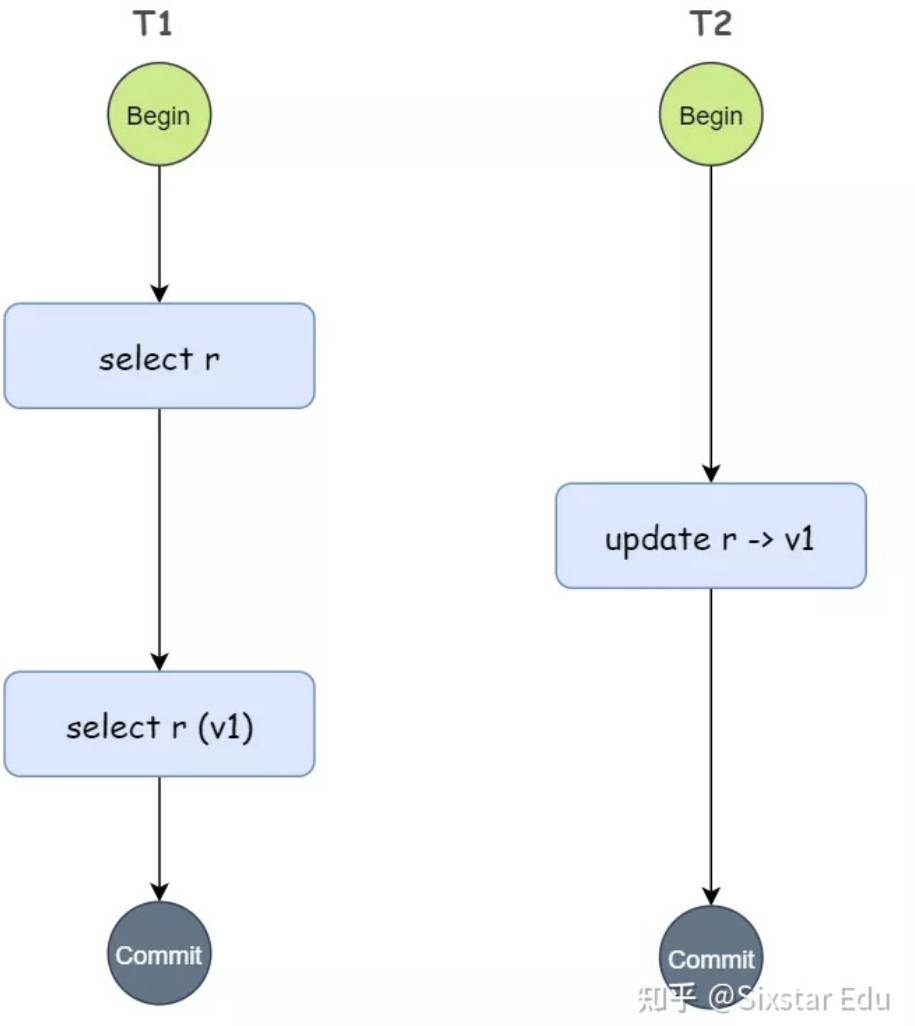


注意，如果想要再现脏读这种情况，需要把隔离级别调整在 Read UnCommitted（读取未提交）。所以事实上脏读这种情况基本不会发生，因为现在大部分数据库的隔离级别都至少设置成 READ COMMITTED

不可重复读 Unrepeatableread

不可重复读是指在一个事务内多次读取同一数据集合。在这个事务还没有结束时，另外一个事务也访问该同一数据集合，并做了一些修改操作。因此，在第一个事务中的两次读数据之间，由于第二个事务的修改，那么第一个事务两次读到的数据可能是不一样的。

举个例子：事务 T1 读取一行数据 r，T2 将该行数据修改成了 v1。如果 T1 再次读取这行数据，此时读取的结果和第一次读取的结果是不同的

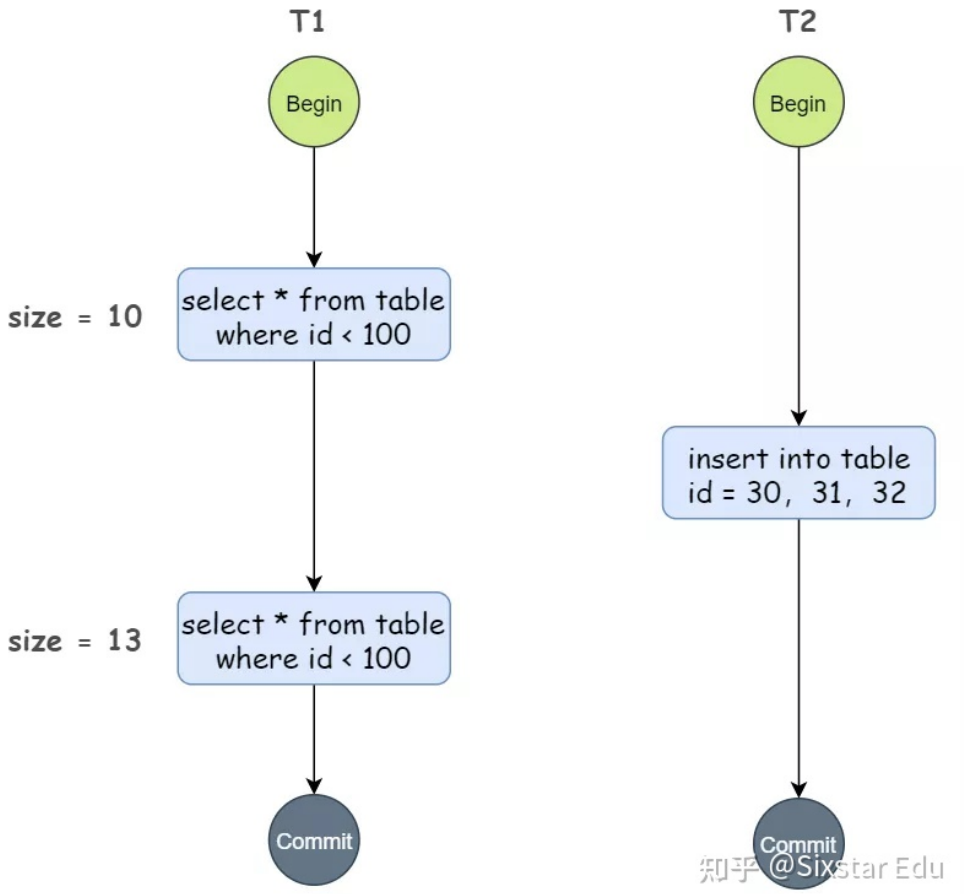


不可重复读和脏读的区别是：脏读是读到未提交的数据，而不可重复读读到的却是已经提交的数据，但是其违反了事务一致性的要求。

幻读 Phantom Read

幻读本质上是属于不可重复读的一种情况，区别在于，不可重复读主要是针对数据的更新（即事务的两次读取结果值不一样），而幻读主要是针对数据的增加或减少（即事务的两次读取结果返回的数量不一样）

举个例子：事务 T1 读取某个范围的数据，事务 T2 在这个范围内插入了一些新的数据，然后 T1 再次读取这个范围的数据，此时读取的结果比第一次读取的结果返回的记录数要多



四种事务隔离级别标准

SQL 标准定义了四种越来越严格的事务隔离级别，用来解决我们上述所说的四种事务的并发一致性问题。

1）READ UNCOMMITTED 读取未提交：事务中的修改，即使没有提交，对其它事务也是可见的。

上面提到过，数据库本身其实已经具备阻止丢失更新的能力，也就是说，即使是最低的隔离级别也可以阻止丢失更新问题。所以：

- 这个隔离级别可以阻止 丢失更新

2）READ COMMITTED 读取已提交：一个事务只能读取已经提交的事务所做的修改。换句话说，一个事务所做的修改在提交之前对其它事务是不可见的。

- 这个隔离级别可以阻止 丢失更新 + 脏读

3）REPEATABLE READ 可重复读（InnoDB 存储引擎默认的隔离级别）：保证在同一个事务中多次读取同一数据的结果是一样的

- 这个隔离级别可以阻止 丢失更新 + 脏读 + 不可重复读

4）SERIALIZABL 可串行化：强制事务串行执行（需要使用锁机制来实现），这样多个事务互不干扰，不会出现并发一致性问题。

- 这个隔离级别可以阻止 丢失更新 + 脏读 + 不可重复读 + 幻读

	丢失更新	脏读	不可重复读	幻读
READ UNCOMMITTED	✅	❌	❌	❌
READ COMMITTED	✅	✅	❌	❌
REPEATABLE READ	✅	✅	✅	❌
SERIALIZABL	✅	✅	✅	✅

可以看到四种隔离级别能阻止的并发一致性问题越来越多，但并不代表越高的隔离级别就越好，因为事务隔离级别越高，数据库付出的性能代价也就相应地越大。

另外，多提一嘴，InnoDB 存储引擎在 REPEATABLE READ 事务隔离级别下，使用 Next-Key Lock 锁的算法避免了大部分幻读场景的产生。

> *声明：本文于网络整理，版权归原作者所有，如来源信息有误或侵犯权益，请联系我们删除或授权事宜。

发布于 2021-11-05 13:53

图解 分布式事务