MySQL手动提交事务

什么是行锁？行锁和隔离等级有什么关系？

什么是行锁？

行锁（Row Lock） 是数据库管理系统（DBMS）用于控制并发操作的一种锁机制。它是指在多个事务并发执行时，**锁定数据库中的某一行数据**，从而防止其他事务对该行数据进行修改。行锁能够有效地减少不同事务之间对数据的冲突，允许多个事务并行操作不同的数据行。

- **行锁的粒度** 是数据库中对数据的锁定范围，它比 **表锁** 更加精细。表锁是锁定整个表，影响表中的所有行，而行锁只锁定一行数据，允许其他行可以并行操作。

行锁的作用：

- **保护数据一致性**：行锁可以确保多个事务同时修改不同的数据行时不会冲突，保持数据一致性。
- **提高并发性**：行锁比表锁允许更多的并发操作，因为它只锁定需要访问的特定行，而不影响其他行的操作。

行锁的工作原理

行锁通常通过以下方式实现：

1. **读取数据时加锁**：在读取数据时，数据库可以选择加锁以防止其他事务对读取的行进行修改。通常，这种锁叫做 **共享锁**（Shared Lock）。多个事务可以同时持有共享锁，但不能修改被锁定的行。
2. **修改数据时加锁**：在修改数据时，数据库会加 **排他锁**（Exclusive Lock）。排他锁只允许持有锁的事务修改数据，其他事务只能等待这个锁被释放。
3. **锁的释放**：事务在提交（`COMMIT`）或回滚（`ROLLBACK`）后释放所持有的行锁。

行锁与事务隔离级别的关系

事务的隔离级别决定了事务如何看到其他事务的更改，并且它对行锁的使用有直接的影响。不同的隔离级别会决定 **行锁的加锁方式** 以及 **是否允许并发操作**。

以下是不同隔离级别下行锁的使用情况：

1. READ UNCOMMITTED（读未提交）

- **脏读**：事务可以读取到其他事务尚未提交的修改，可能会读取到不一致的、未提交的数据。
- **行锁使用情况**：通常不会加锁（即使读取数据也不加锁），因此多个事务可以同时修改同一行数据，导致 **脏读**、**不可重复读** 等问题。

行锁情况：

- 由于不加锁，其他事务可以并发修改同一行数据，导致脏读和并发冲突。

2. READ COMMITTED（读已提交）

- **不可重复读**：事务只能读取已经提交的数据，避免了脏读，但在同一事务内的两次读取操作可能会读取到不同的数据（即数据发生了变化）。
- **行锁使用情况**：事务在读取数据时会加 **共享锁**，在修改数据时会加 **排他锁**。

行锁情况：

- 在 `READ COMMITTED` 隔离级别下，查询数据时会加 **共享锁**，但只锁定当前读取的行，锁会在读取结束后释放。对于修改操作，系统会加 **排他锁**，直到事务提交。

3. REPEATABLE READ（可重复读）

- **不可重复读**：事务在执行多次读取操作时，读取的数据是相同的，避免了不可重复读的问题，但依然存在 **幻读**（即某一事务读取数据时，其他事务可能插入新数据，导致当前事务再读取时数据不一致）。
- **行锁使用情况**：在执行查询操作时，事务会使用 **共享锁** 来防止其他事务修改数据。同时，修改操作时使用 **排他锁**。

行锁情况：

- `REPEATABLE READ` 会确保在事务生命周期内，对同一行数据的查询总是返回相同的数据，因此会锁定读取的行。
- 但是，这个级别可能会发生 **幻读**，即在一个事务内的多次查询之间，其他事务可以插入数据导致查询结果不同。

4. SERIALIZABLE（串行化）

- **串行化**：是最严格的隔离级别，完全避免脏读、不可重复读和幻读，所有事务会像串行执行一样，一个事务必须等待另一个事务结束后才能执行。
- **行锁使用情况**：事务在进行查询时会加 **共享锁**，并且会对查询的范围加 **范围锁**（Range Lock），确保没有其他事务能够插入或修改被查询的数据。

行锁情况：

- `SERIALIZABLE` 隔离级别会对所有涉及的数据行加锁，甚至对 **范围内的数据行**（即可能符合查询条件的数据行）加锁。这可以确保事务执行时不会被其他事务插入或修改相关数据。

行锁的加锁策略与隔离级别总结

隔离级别	读取未提交的数据	行锁使用情况	幻读
READ UNCOMMITTED	允许脏读	不加锁（可以并发修改数据）	存在
READ COMMITTED	不允许脏读	读取时加共享锁，更新时加排他锁	存在
REPEATABLE READ	不允许脏读	读取时加共享锁，更新时加排他锁，防止不可重复读	存在
SERIALIZABLE	不允许脏读	读取时加共享锁，更新时加排他锁，且对查询范围加锁	不存在

5. 行锁的优缺点

优点：

- **高并发性**：行锁的粒度小，可以让多个事务并发操作不同的行，提高数据库的并发性。
- **防止冲突**：在多事务同时修改数据时，行锁能确保数据的一致性，避免冲突。

缺点：

- **锁争用**：如果多个事务频繁访问同一行数据，可能会导致锁争用，性能下降。
- **死锁**：如果事务间的锁获取顺序不当，可能导致死锁，系统会进入一种互相等待的状态，必须通过回滚事务来解决。

结论

- **行锁** 是数据库锁定数据时的一种精细化锁机制，锁定单独的行而非整个表，允许多个事务并发操作不同的数据行。
- **隔离级别** 会影响行锁的加锁方式，`READ UNCOMMITTED` 隔离级别不会加锁，而 `SERIALIZABLE` 则对数据进行严格的锁定。
- 行锁的使用可以确保并发执行时的数据一致性，但在高并发的情况下，可能会导致锁争用和性能下降。