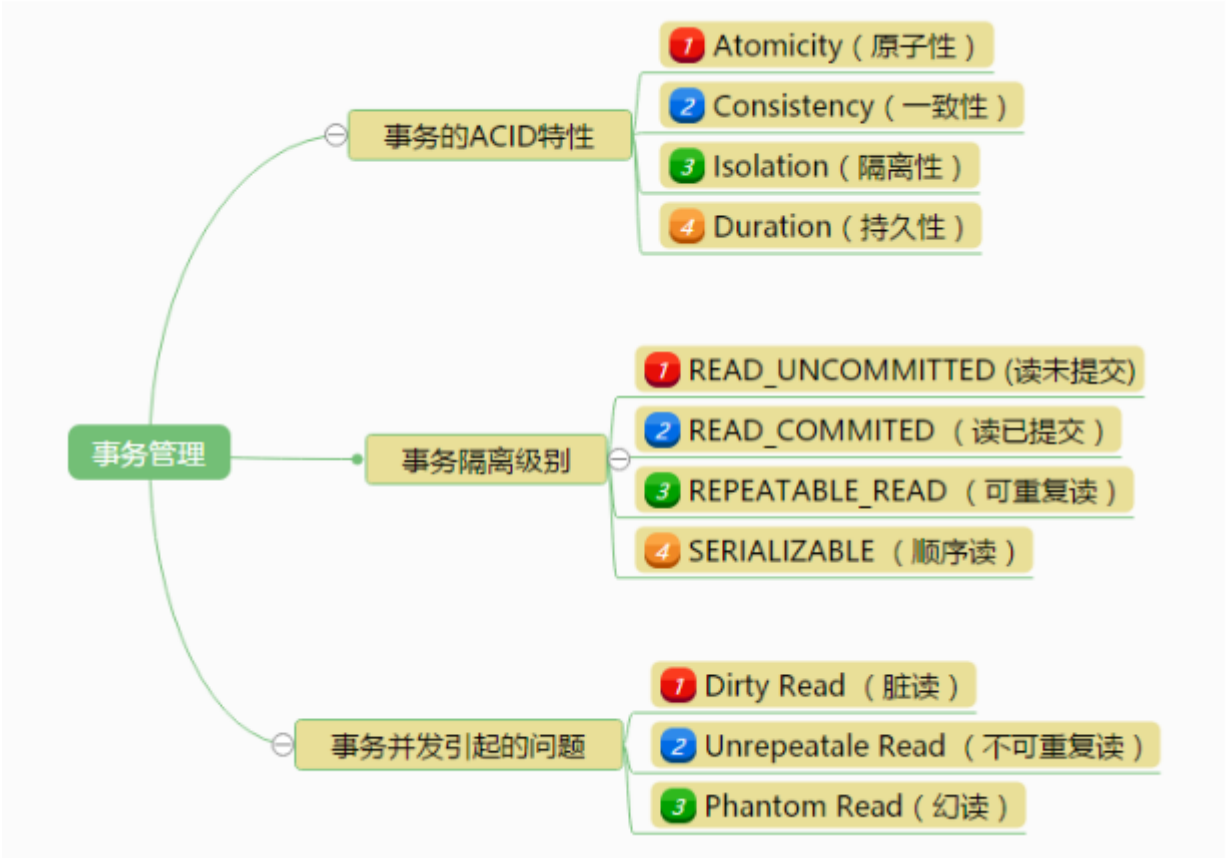
第一章 事务

事务（Transaction）是由一系列对系统中数据进行访问与更新的操作所组成的一个程序执行逻辑单元。 (1) 事务的语法 (2) 事务的特性 (3) 事务并发问题 (4) 事务隔离级别 (5) 不同隔离级别的锁的情况(了解) (6) 隐式提交(了解)



1.1 事务的语法

1. start transaction; begin;
2. commit; 使得当前的修改确认
3. rollback; 使得当前的修改被放弃

1.2 事务的ACID特性

1. 原子性（Atomicity）
   事务的原子性是指事务必须是一个原子的操作序列单元。事务中包含的各项操作在一次执行过程中，只允许出现两种状态之一。
   （1）全部执行成功
   （2）全部执行失败
   事务开始后所有操作，要么全部做完，要么全部不做，不可能停滞在中间环节。事务执行过程中出错，会回滚到事务开始前的状态，所有的操作就像没有发生一样。也就是说事务是一个不可分割的整体，就像化学中学过的原子，是物质构成的基本单位。
2. 一致性（Consistency）
   事务的一致性是指事务的执行不能破坏数据库数据的完整性和一致性，一个事务在执行之前和执行之

后，数据库都必须处以一致性状态。
比如：如果从A账户转账到B账户，不可能因为A账户扣了钱，而B账户没有加钱。

3. 隔离性（Isolation）
事务的隔离性是指在并发环境中，并发的事务是互相隔离的。也就是说，不同的事务并发操作相同的数据时，每个事务都有各自完整的数据空间。
一个事务内部的操作及使用的数据对其它并发事务是隔离的，并发执行的各个事务是不能互相干扰的。
隔离性分4个级别，下面会介绍。
4. 持久性（Duration）
事务的持久性是指事务一旦提交后，数据库中的数据必须被永久的保存下来。即使服务器系统崩溃或服务器宕机等故障。只要数据库重新启动，那么一定能够将其恢复到事务成功结束后的状态

1.3 事务的并发问题

脏读：读取到了没有提交的数据, 事务A读取了事务B更新的数据，然后B回滚操作，那么A读取到的 数据是脏数据。 不可重复读：同一条命令返回不同的结果集（更新）.事务 A 多次读取同一数据，事务 B 在事务A 多次读取的过程中，对数据做了更新并提交，导致事务A多次读取同一数据时，结果不一致。 幻读：重复查询的过程中，数据就发生了量的变化（insert， delete）。

```
create table users(
id int auto_increment primary key,
name varchar(10),
age int,
account int
)

insert into users values(null,'张三',25,10000),(null,'李李四',20,100),(null,'王五',23,0);
```

1.4 事务隔离级别

事务隔离级别	脏 读	不可重复读	幻 读
读未提交（READ_UNCOMMITTED）	允许	允许	允许
读已提交（READ_COMMITTED）	禁止	允许	允许
可重复读（REPEATABLE_READ）	禁止	禁止	可能会
顺序读（SERIALIZABLE）	禁止	禁止	禁止

4种事务隔离级别从上往下，级别越高，并发性越差，安全性就越来越高。 一般数据默认级别是 读以提交或可重复读

查看当前会话中事务的隔离级别

```
mysql> select @@tx_isolation;
+-----------------+
| @@tx_isolation |
+-----------------+
| REPEATABLE-READ |
+-----------------+
1 row in set, 1 warning (0.93 sec)
```

设置当前会话中的事务隔离级别 mysql> set session transaction isolation level read uncommitted; Query OK, 0 rows affected (0.00 sec)

1. 读未提交（READ_UNCOMMITTED）
   读未提交，该隔离级别允许脏读取，其隔离级别是最低的。换句话说，如果一个事务正在处理理某一数据，并对其进行了更新，但同时尚未完成事务，因此还没有提交事务；而以此同时，允许另一个事务也能够访问该数据。
   脏读示例：

在事务A和事务B同时执行时可能会出现如下场景：

时间	事务A（存款）	事务B（取款）
T1	开始事务	——
T2	——	开始事务
T3	——	查询余额（1000元）
T4	——	取出1000元（余额0元）
T5	查询余额（0元）	——
T6	——	撤销事务（余额恢复1000元）
T7	存入500元（余额500元）	——
T8	提交事务	——

余额应该为1500元才对。请看T5时间点，事务A此时查询的余额为0，这个数据就是脏数据，他是事务B 造成的，很明显是事务没有进行隔离造成的。

2. 读已提交（READ_COMMITTED）
   读已提交是不同的事务执行的时候只能获取到已经提交的数据。 这样就不会出现上面的脏读的情况了。
   但是在同一个事务中执行同一个读取,结果不一致
   不可重复读示例
   可是解决了脏读问题，但是还是解决不了可重复读问题。

时间	事务A（存款）	事务B（取款）
T1	开始事务	——
T2	——	开始事务
T3	——	查询余额（1000元）
T4	查询余额（1000元）	——
T5	——	取出1000元（余额0元）
T6	——	提交事务
T7	查询余额（0元）	——
T8	提交事务	——

事务A其实除了了查询两次以外，其它什什么事情都没做，结果钱就从1000变成0了了，这就是不不可重复读的 问题。

3. 可重复读（REPEATABLE_READ）
   可重复读就是保证在事务处理理过程中，多次读取同一个数据时，该数据的值和事务开始时刻是一致的。
   因此该事务级别限制了不可重复读和脏读，但是有可能出现幻读的数据。
   幻读
   幻读就是指同样的事务操作，在前后两个时间段内执行对同一个数据项的读取，可能出现不一致的结果。
   诡异的更新事件

时间	事务A	事务B
T1	开始事务	——
T2	查询当前所有数据	开始事务
T3	——	插入一条数据
T4	查询当前所有数据	提交事务
T5	进行范围修改	——
T6	查询当前所有数据	——
T7	提交事务	——

4. 顺序读（SERIALIZABLE）
   顺序读是最严格的事务隔离级别。它要求所有的事务排队顺序执行行行，即事务只能一个接一个地处理，不能并发。

1.5 不同的隔离级别的锁的情况(了解)

1. 读未提交（RU）：有行级的锁，没有间隙锁。它与RC的区别是能够查询到未提交的数据。

2. 读已提交（RC）：有行级的锁，没有间隙锁，读不到没有提交的数据。
3. 可重复读（RR）：有行级的锁，也有间隙锁，每次读取的数据都是一样的，并且没有幻读的情况。
4. 序列化（S）：有行级锁，也有间隙锁，读表的时候，就已经上锁了

1.6 隐式提交(了解)

DQL:查询语句句 DML:写操作(添加,删除,修改) DDL:定义语句句(建库,建表,修改表,索引操作,存储过程,视图) DCL:控制语言言(给用用户授权,或删除授权) DDL（Data Define Language）：都是隐式提交。 隐式提交：执行行行这种语句句相当于执行行行commit; DDL

https://dev.mysql.com/doc/refman/5.7/en/implicit-commit.html