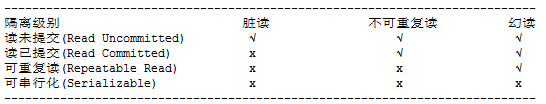
**MySql事务隔离级别**

***数据库有四种隔离级别:***

1. 读取未提交的内容(Read Uncommitted):顾名思义，就是读取其他事务未提交的操作，比如A事务读取了B事务未提交的操作，如果B事务回滚了，那么A所读取的数据就是有误的，这个亦称之为脏读(Dirty Read)。这个隔离级别也不会比其他的隔离级别性能好多少，所以一般不用。
2. 读取提交的内容(Read Committed):这个是大部分数据库的隔离级别，但是不是MySql数据库的隔离级别，这个要注意。这种也称之为不可重复读(Nonrepeatable Read),因为同一事务在处理数据时，数据可能被多次修改，所以条件一样的select可能会返回不同的结果。
3. 可重复读(Repeatable Read):这个是MySql默认的数据库隔离级别，由此可看出MySql对自己的性能有多自信。这个可以解决脏读和不可重复读的问题，不过他会有幻读的情况，比如A事务查询一个范围内的数据记录，这个时候另一个线程往范围里面insert了一条数据，在A事务里面其他操作再查询同样的范围时，会出现多一条幻影行，这个就称之为幻读。
4. 可串行化(Serializable):解决了上述的问题，是最高的事务隔离级别。在读的数据行上加上共享锁，这个会引起锁竞争，会有一定的性能问题。



***延伸:***

查看数据库隔离级别:select @@tx\_isolation;

设置数据库隔离级别:set tx\_isolation='read-uncommitted';

开始事务:start transaction;

回滚操作:rollback;

提交事务操作:commit;

注意：目前设置的数据库隔离级别，都是针对当前命令会话的。

***实验准备***

1. 创建数据库:test
2. 创建数据库表:创建一张tx表

use test;

CREATE TABLE `tx` (

id int(11) not null,

num int(11) not null

) ENGINE=InnoDB AUTO\_INCREMENT=19 DEFAULT CHARSET=utf8 COMMENT='test';

1. 插入3条数据

insert into `tx`(`id`,`num`) values(1, 1);

insert into `tx`(`id`,`num`) values(2, 2);

insert into `tx`(`id`,`num`) values(3, 3);

***开始实验:***

***1.设置A终端隔离级别为 【读取未提交】***

set tx\_isolation='read-uncommitted';

验证一下当前session的isolation

select @@tx\_isolation;

A的命令终端，开启一个查询事务

start transaction;

select \* from tx;

打开一个B的命令终端,开启一个更新的事务

start transaction;

update tx set num=4 where id=1;

用A命令终端继续在之前的事务里面查询

select \* from tx;

!!A终端查询到B未提交的事务数据!!

如果B执行了rollback，那么A读取到的是脏数据。

---这就是 读取未提交的数据，亦称之为 脏读。

***2. A数据库隔离级别为 【读取已提交】***

set tx\_isolation='read-committed';

这个还是跟刚才的实验一样，可以看到当B终端未之星commit时，A终端看到的还是之前的数据。所以解决了脏读的问题

但是不能解决不可重复读的问题。A终端事务多次读取B终端提交的数据时，看到的结果是不一样的。

***3. 将A终端的事务隔离界别设置为【可重复读】***

set tx\_isolation='repeatable-read';

select @@tx\_isolation;

A命令终端开启查询事务

start transaction;

select \* from tx;

B命令终端开启一个更新事务，注意一定要在事务里面，不能直接执行更新命令

start transaction;

update tx set num=4 where id=1;

commit;

A命令终端当前事务继续查询

select \* from tx;

可以看到，虽然B终端的更新事务执行了commite操作，但是A命令终端看到的还是B更新前的数据。

所以可重复读可以解决不可重复读的问题。但是他不能解决幻读的问题。

***4.剩下那个是解决幻读的，没必要做实验了，实验的结果已大致能想象***

就是同一个A终端在一个事务里面，同样的范围查询，得出的数据条数永远是一样的，无论B终端有没有insert数据。