数据库技术第二次作业

潘亦晟
515021910384

1. 问题 1

(1) 隔离级别：TRANSACTION_SERIALIZABLE
(2) 封锁粒度：表锁
(3) 查询结果：

```
Thread Insert Created!
Thread Querry Created!
Thread Insert connected database
Thread Querry connected database
average time:90.1672
Thread Querry Done! runtime:333 ms
Thread Insert Done! runtime:11725 ms
```

调度结果：先查询后插入

(1) 隔离级别：TRANSACTION_SERIALIZABLE
(2) 封锁粒度：IX 锁
(3) 查询结果：

```
Thread Insert Created!
Thread Querry Created!
Thread Querry connected database
Thread Insert connected database
average time:90.1672
Thread Querry Done! runtime:317 ms
Thread Insert Done! runtime:11738 ms
```

调度结果：先查询后插入

(1) 隔离级别：TRANSACTION_SERIALIZABLE
(2) 封锁粒度：IS 锁
(3) 查询结果：

```
Thread Insert Created!
Thread Querry Created!
Thread Querry connected database
Thread Insert connected database
average time:90.1672
Thread Querry Done! runtime:317 ms
Thread Insert Done! runtime:11649 ms
```

调度结果：先查询后插入

(4) 隔离级别：TRANSACTION_REPEATABLE_READ

(5) 封锁粒度：表锁
(6) 查询结果：
```
Thread Insert Created!
Thread Querry Created!
Thread Insert connected database
Thread Querry connected database
average time:90.1672
Thread Querry Done! runtime:311 ms
Thread Insert Done! runtime:12383 ms
```
调度结果：先查询后插入

(4) 隔离级别：TRANSACTION_REPEATABLE_READ
(5) 封锁粒度：IX 锁
(6) 查询结果：
```
Thread Insert Created!
Thread Querry Created!
Thread Insert connected database
Thread Querry connected database
average time:90.1672
Thread Querry Done! runtime:326 ms
Thread Insert Done! runtime:12459 ms
```
调度结果：先查询后插入

(7) 隔离级别：TRANSACTION_REPEATABLE_READ
(8) 封锁粒度：IS 锁
(9) 查询结果：
```
Thread Insert Created!
Thread Querry Created!
Thread Querry connected database
Thread Insert connected database
average time:90.1672
Thread Querry Done! runtime:327 ms
Thread Insert Done! runtime:11767 ms
```
调度结果：先查询后插入

(10) 隔离级别：TRANSACTION_READ_COMMITTED
(11) 封锁粒度：表锁
(12) 查询结果：
```
Thread Insert Created!
Thread Querry Created!
Thread Querry connected database
Thread Insert connected database
```

```
average time:90.1672
Thread Querry Done! runtime:309 ms
Thread Insert Done! runtime:11694 ms
```
调度结果：先查询后插入

(13) 隔离级别：TRANSACTION_READ_COMMITTED
(14) 封锁粒度：IX 锁
(15) 查询结果：
```
Thread Insert Created!
Thread Querry Created!
Thread Insert connected database
Thread Querry connected database
average time:90.1672
Thread Querry Done! runtime:330 ms
Thread Insert Done! runtime:12757 ms
```
调度结果：先查询后插入

(16) 隔离级别：TRANSACTION_READ_COMMITTED
(17) 封锁粒度：IS 锁
(18) 查询结果：
```
Thread Insert Created!
Thread Querry Created!
Thread Insert connected database
Thread Querry connected database
average time:90.1672
Thread Querry Done! runtime:326 ms
Thread Insert Done! runtime:11934 ms
```
调度结果：先查询后插入

(19) 隔离级别：TRANSACTION_READ_UNCOMMITTED
(20) 封锁粒度：表锁
(21) 查询结果：
```
Thread Insert Created!
Thread Querry Created!
Thread Insert connected database
Thread Querry connected database
average time:90.1672
Thread Querry Done! runtime:321 ms
Thread Insert Done! runtime:12076 ms
```
调度结果：先查询后插入

(22) 隔离级别：TRANSACTION_READ_UNCOMMITTED
(23) 封锁粒度：IX 锁
(24) 查询结果：

```
Thread Insert Created!
Thread Querry Created!
Thread Insert connected database
Thread Querry connected database
average time:90.1672
Thread Querry Done! runtime:321 ms
Thread Insert Done! runtime:12076 ms
```

调度结果：先查询后插入

(25) 隔离级别：TRANSACTION_READ_UNCOMMITTED
(26) 封锁粒度：IS 锁
(27) 查询结果：

```
Thread Insert Created!
Thread Querry Created!
Thread Insert connected database
Thread Querry connected database
average time:90.1672
Thread Querry Done! runtime:321 ms
Thread Insert Done! runtime:12076 ms
```

调度结果：先查询后插入

SQLite
(28) 隔离级别：TRANSACTION_SERIALIZABLE
(29) 封锁粒度：页锁
(30) 查询结果：

```
Thread A Created!
Thread B Created!
Thread A connected database
Thread B connected database
average time: 90.1672
Thread B Done! runtime:3001
Thread A Done runtime:52016
```

调度结果：先查询后插入

结果分析（性能、正确性）：
理论上，并行性越高的
但是由于数据量的操作过小，以及 CPU 的性能不稳定等因素，我们很难使用 mysql 时不同隔离级别和不同封锁粒度下对于两个事务的并发进程运行时间无明显差异，但是对于 sqlite

而言，运行速度比 mysql 慢很多了，这可能与 sqlite 自身引擎有关。
同时我们发现，对于 sqlite 而言自身，其只支持一个默认的页锁，且在实验中发现四种隔离级别中只有 TRANSACTION_SERIALIZABLE 在运行时不会报错，因此结果仅放这一种。而对于 mysql 而言，其具有四个隔离级别，且支持表锁和两种行锁（IX 和 IS），因此一共有 12 种情况。我们发现查询结果相同，通过与插入前使用 sql 语句的查询结果，发现其调度结果全是先查询后插入。

2. 问题 2

(4) 隔离级别：TRANSACTION_SERIALIZABLE
(7) 封锁粒度：表锁
(8) 查询结果：

```
Thread Delete Created!
Thread Querry Created!
Thread Querry connected database
Thread Delete connected database
average time:94.6837
Thread Querry Done! runtime:1656 ms
Thread Delete Done! runtime:5712 ms
```

调度结果：先删除后查询

(5) 隔离级别：TRANSACTION_SERIALIZABLE
(9) 封锁粒度：IX 锁
(10) 查询结果：

```
Thread Delete Created!
Thread Querry Created!
Thread Querry connected database
Thread Delete connected database
Thread Delete Done! runtime:4269 ms
average time:94.6837
Thread Querry Done! runtime:5707 ms
```

调度结果：先删除后查询

(6) 隔离级别：TRANSACTION_SERIALIZABLE
(11) 封锁粒度：IS 锁
(12) 查询结果：

```
Thread Delete Created!
Thread Querry Created!
Thread Querry connected database
Thread Delete connected database
average time:94.7575
Thread Querry Done! runtime:1914 ms
Thread Delete Done! runtime:5622 ms
```
调度结果：先查询后删除

(31) 隔离级别：TRANSACTION_REPEATABLE_READ
(13) 封锁粒度：表锁
(14) 查询结果：
```
Thread Delete Created!
Thread Querry Created!
Thread Delete connected database
Thread Querry connected database
average time:94.7575
Thread Querry Done! runtime:1696 ms
Thread Delete Done! runtime:5186 ms
```
调度结果：先查询后删除

(32) 隔离级别：TRANSACTION_REPEATABLE_READ
(15) 封锁粒度：IX 锁
(16) 查询结果：
```
Thread Delete Created!
Thread Querry Created!
Thread Querry connected database
Thread Delete connected database
average time:94.7575
Thread Querry Done! runtime:1677 ms
Thread Delete Done! runtime:5296 ms
```
调度结果：先查询后删除

(33) 隔离级别：TRANSACTION_REPEATABLE_READ
(17) 封锁粒度：IS 锁
(18) 查询结果：
```
Thread Delete Created!
Thread Querry Created!
Thread Delete connected database
Thread Querry connected database
average time:94.7575
Thread Querry Done! runtime:1977 ms
Thread Delete Done! runtime:6116 ms
```
调度结果：先查询后删除

(34) 隔离级别：TRANSACTION_READ_COMMITTED

(19) 封锁粒度：表锁
(20) 查询结果：
```
Thread Delete Created!
Thread Querry Created!
Thread Delete connected database
Thread Querry connected database
average time:94.7575
Thread Querry Done! runtime:1698 ms
Thread Delete Done! runtime:5237 ms
```
调度结果：先查询后删除

(35) 隔离级别：TRANSACTION_READ_COMMITTED
(21) 封锁粒度：IX 锁
(22) 查询结果：
```
Thread Delete Created!
Thread Querry Created!
Thread Delete connected database
Thread Querry connected database
average time:94.6837
Thread Querry Done! runtime:1799 ms
Thread Delete Done! runtime:5204 ms
```
调度结果：先删除后查询

(36) 隔离级别：TRANSACTION_READ_COMMITTED
(23) 封锁粒度：IS 锁
(24) 查询结果：
```
Thread Delete Created!
Thread Querry Created!
Thread Delete connected database
Thread Querry connected database
average time:94.7575
Thread Querry Done! runtime:1715 ms
Thread Delete Done! runtime:5469 ms
```
调度结果：先查询后删除

(25) 隔离级别：TRANSACTION_READ_UNCOMMITTED
(26) 封锁粒度：表锁
(27) 查询结果：
```
Thread Delete Created!
Thread Querry Created!
Thread Delete connected database
Thread Querry connected database
average time:94.7575
Thread Querry Done! runtime:1926 ms
Thread Delete Done! runtime:5824 ms
```
调度结果：先查询后删除

(28) 隔离级别：TRANSACTION_READ_UNCOMMITTED
(29) 封锁粒度：IX 锁
(30) 查询结果：

```
Thread Delete Created!
Thread Querry Created!
Thread Querry connected database
Thread Delete connected database
average time:94.6837
Thread Querry Done! runtime:5397 ms
Thread Delete Done! runtime:8875 ms
```
调度结果：先删除后查询

(31) 隔离级别：TRANSACTION_READ_UNCOMMITTED
(32) 封锁粒度：IS 锁
(33) 查询结果：

```
Thread Delete Created!
Thread Querry Created!
Thread Querry connected database
Thread Delete connected database
average time:94.7575
Thread Querry Done! runtime:1947 ms
Thread Delete Done! runtime:6001 ms
```
调度结果：先查询后删除

Sqlite：
(34) 隔离级别：TRANSACTION_SERIALIZABLE
(35) 封锁粒度：页锁
(36) 查询结果：

```
Thread Delete Created!
Thread Querry Created!
Thread Querry connected database
Thread Delete connected database
average time:94.7575
Thread Querry Done! runtime:3464
Thread Delete Done runtime:23652
average time:94.7575
```

结果分析（性能、正确性）：
与第一题相同，对于 sqlite 我们同样只考虑串行化隔离级别，而对于 mysql，我们同样需要考虑四种隔离级别和三种锁的组合十二种情况。根据结果我们可以看出，其性能差异依然不大，且只有两个调度情况（先删除后查询，先查询后删除）。我们可以基于不多的数据量看出先查询后删除的性能似乎比先删除后查询的性能略微好点，但是由于 cpu 运行速度的不稳定性，也很难看出有明显差异。

3. Bonus

(1) 场景描述：我们类比于老师上课讲的飞机买票问题与作业二问题 1 进行结合考虑，来验证是否出现脏读和可重复读。策略如下：将每隔一段时间（1000ms）就读一次运行时间的平均值，一共运行十次，对比这十次结果是否有所不同，是否会出现可重复读和脏读的情况。
(2) 隔离级别：TRANSACTION_READ_UNCOMMITTED
(3) 封锁粒度：表锁
(4) 查询结果：

```
Thread Insert Created!
Thread Querry Created!
Thread Querry connected database
Thread Insert connected database
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:66.3855
average time:66.3855
average time:66.3855
Thread Querry Done! runtime:12879 ms
Thread Insert Done! runtime:11694 ms
```

(1) 隔离级别：TRANSACTION_SERIALIZABLE
(2) 封锁粒度：表锁
(3) 查询结果：

```
(4) Thread Insert Created!
(5) Thread Querry Created!
(6) Thread Querry connected database
(7) Thread Insert connected database
```

```
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
Thread Querry Done! runtime:13649 ms
Thread Insert Done! runtime:11874 ms
```

结果分析（性能、正确性）：
从上面我们可以看出，在串行化和读未提交两个隔离级别进行对比，出现了不可重复读的问题。在读未提交的隔离下，读未提交出现了同一事务下读的结果不同。

(5) 场景描述：我们类比于老师上课讲的飞机买票问题与作业二问题 1 进行结合考虑，来验证是否出现脏读和可重复读。策略如下：将每隔一段时间（1000ms）就读一次运行时间的平均值，一共运行十次，同时我们在插入事务结束时进行 rollback 操作，对比这十次结果是否有所不同，是否会出现可重复读和脏读的情况。
(6) 隔离级别：TRANSACTION_READ_UNCOMMITTED
(7) 封锁粒度：表锁
(8) 查询结果：

```
Thread Insert Created!
Thread Querry Created!
Thread Querry connected database
Thread Insert connected database
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:66.3855
average time:66.3855
average time:66.3855
Thread Querry Done! runtime:12978 ms
Thread Insert Done! runtime:11784 ms
```

(9) 隔离级别：TRANSACTION_READ_COMMITTED

(10) 封锁粒度：表锁
(11) 查询结果：

```
Thread Insert Created!
Thread Querry Created!
Thread Querry connected database
Thread Insert connected database
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
Thread Querry Done! runtime:12579 ms
Thread Insert Done! runtime:11994 ms
```

(12) 隔离级别：TRANSACTION_SERIALIZABLE
(13) 封锁粒度：表锁
(14) 查询结果：

```
Thread Insert Created!
Thread Querry Created!
Thread Querry connected database
Thread Insert connected database
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.1672
average time:90.3855
average time:90.3855
average time:90.3855
Thread Querry Done! runtime:14879 ms
Thread Insert Done! runtime:12494 ms
```

结果：我们发现出现了脏读的情况