MySQL1 前沿

并发控制

> 无论何时，只要有多个查询需要同一时刻修改数据，都会产生并发控制的问题。这里讨论两个层面的并发控制：服务器层与存储引擎层。

读写锁

> 背景：某个用户正在读取邮箱，同时另外一个用户试图删除编号为25的邮件，会出现什么?

经典办法：并发控制；可以通过实现一个由两种类型的锁组成的锁系统来解决问题。这两种锁通常被称为 共享锁（share lock） 和 排他锁 （exclusive lock ），也叫 读锁 （read lock） 和 写锁 （write lock）

读锁：读锁是共享的，或者说相互不阻塞的，多个用户在同一时刻读取同一个资源，而不干扰。
写锁：写锁是排他的，一个写锁会阻塞其他的写锁和读锁。

锁粒度

> 一种提高共享资源并发性的方式就是让锁定对象更有选择性。

- 表锁（table lock）

  - 最基本的锁策略，开销最小
  - 锁定整张表

- 行级锁

  - 可以最大程度支持并发处理
  - 资源开销大

死锁

> 死锁是指两个或者多个事务在同一资源三相互占用，并请求锁定对方的资源，然后导致恶性循环。

```
START TRANSCATION
UPDATE checking SET balance = balance - 200.00 WHERE customer_id = 10233276;
UPDATE saving SET balance = balance + 200.00 WHERE custom_id = 10233276;
COMMIT
```

```
START TRANSCATION
UPDATE saving SET balance = balance + 200.00 WHERE custom_id = 10233276;
UPDATE checking SET balance = balance - 200.00 WHERE customer_id = 10233276;
COMMIT
```

当上面两个语句同时运行第一个UPDATE语句的时候，在第二行会出现死锁现象。大多数情况下需要重新执行死锁回滚的事务即可

事务

```
START TRANSCATION
SELECT balance FROM checking WHERE customer_id = 10233276;
UPDATE checking SET balance = balance - 200.00 WHERE customer_id = 10233276;
UPDATE saving SET balance = balance + 200.00 WHERE custom_id = 10233276;
COMMIT
```

> 事务就是一组原子性的SQL查询，或者说一个独立的工作单元，如上代码。事务必须遵守严格的ACID测试。

ACID

> 一个运行良好的事务处理系统，必须具备这些标准特征

- 原子性（atomicity）

  一个事务必须被视为一个不可分隔的最小单元，整个事务中的所有操作要么全部提交要么回滚。

- 一致性（consistency）

  数据库总是从一个一致性状态转到另一个一致性状态

- 隔离性

  一个事务所作的修改在最终提交前，对其他事务是不可见的。

- 持久性

  一旦事务提交，所作的修改就会永久保存到数据库中，即使系统崩溃，修改的数据也不会丢失。

隔离级别

> 在SQL标准中定义了四种隔离级别。

- READ UNCOMMITTED (未提交读)

  事务中的修改，即使没有提交，对其他事务也是可见的。事务可以读取未提交的数据，这个叫脏读（`Dirty Read`）。一般不用

- READ COMMITTED (提交读)

  大多数数据库的默认离别等级。一个事务开始时，只能“看见”已经提交的事务所做的修改。（一个事务从开始直到提交之前，所做的任何修改对其他事务都是不可见的）

- REPEATABLE READ (可重复读)

> 解决了脏读问题，保证了同一个事务中多次读取同样记录的结果是一致的，但是无法解决幻读（Phantom Read） 问题，指当

- SERIALIZABLE (可串行化)

> 最高级别的隔离，对每一行数据都加上锁，但是消费高

隔离等级	脏读	可重复读	幻读
未授权读取	存在	不可以	存在
授权读取	不存在	不可以	存在
可重复读取	不存在	可以	存在
串行化	不存在	可以	不存在

事务型存储引擎：

> MySQL提供了两种事务型的存储模型：InnoDB 和 NDB Cluster。还有其他第三方引擎。

自动提交 （AUTOCOMMIT)

> 默认采用该模式，如果不是显式地开始一个事务，那么每个查询都被当成一个事务执行提交操作。可以通过设置AUTOCOMMIT变量来启动或者禁止自动提交模式

```
SET AUTOCOMMIT = 1;  1或者ON表示启用， 0或者OFF表示禁用
```

设置隔离级别

```
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
```