重要概念

事务Transaction

InnoDB引擎，支持事务。

事务：由若干条语句组成的，指的是要做的一系列操作。

关系型数据库中支持事务，必须支持其四个属性（ACID）：

特性	描述
原子性（atomicity）	一个事务是一个不可分割的工作单位，事务中包括的所有操作要么全部做完，要么什么都不做
一致性（consistency）	事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的
隔离性（isolation）	一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰
持久性（durability）	持久性也称永久性（permanence），指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响

> 原子性，要求事务中的所有操作，不可分割，不能做了一部分操作，还剩一部分操作；
> 一致性，多个事务并行执行的结果，应该和事务排队执行的结果一致。如果事务的并行执行和多线程读写共享资源一样不可预期，就不能保证一致性。
> 隔离性，就是指多个事务访问共同的数据了，应该互不干扰。隔离性，指的是究竟在一个事务处理期间，其他事务能不能访问的问题
> 持久性，比较好理解，就是事务提交后，数据不能丢失。

MySQL隔离级别

隔离性不好，事务的操作就会互相影响，带来不同严重程度的后果。
首先看看隔离性不好，带来哪些问题：

1. 更新丢失Lost Update
   事务A和B，更新同一个数据，它们都读取了初始值100，A要减10，B要加100，A减去10后更新为90，B加100更新为200，A的更新丢失了，就像从来没有减过10一样。
2. **脏读**
   事务A和B，事务B读取到了事务A未提交的数据（这个数据可能是一个中间值，也可能事务A后来回滚事务）。事务A是否最后提交并不关心，只要读取到了这个被修改的还未提交的数据就是脏读。
3. 不可重复读Unrepeatable read
   事务A在事务执行中使用了相同查询语句，得到了不同的结果，**不能**保证同一条查询语句**重复读**相同的结果就是不可以重复读。
   例如，事务A查询了一次后，事务B**修改**了数据并提交了事务，事务A又查询了一次，发现数据不一致了。
   注意，脏读讲的是可以读到相同的数据的，但是读取的是一个未提交的数据，而不是提交的最终结果。
4. **幻读**Phantom read
   事务A中同一个查询要进行多次，事务B插入数据，导致A返回不同的结果集，如同幻觉，就是幻

读。
数据集有记录增加了，可以看做是**增加**了记录的不可重复读。

有了上述问题，数据库就必须要解决，提出了隔离级别。
隔离级别由低到高，如下表

隔离级别	描述
READ UNCOMMITTED	读取到未提交的数据
READ COMMITTED	读已经提交的数据，ORACLE默认隔离级别。也称为不可重复读
REPEATABLE READ	可以重复读，**MySQL的 默认隔离级别。**
SERIALIZABLE	可串行化。事务间完全隔离，事务不能并发，只能串行执行

隔离级别越高，串行化越高，数据库并行执行效率低；隔离级别越低，并行度越高，性能越高。
隔离级别越高，当前事务处理的中间结果对其它事务不可见程度越高。

- SERIALIZABLE，串行了，解决所有问题
- REPEATABLE READ，事务A中同一条查询语句返回同样的结果，就是可以重复读数据了。例如语句为(select * from user)。解决的办法有：
  1、对select的数据加锁，不允许其它事务删除、修改的操作
  2、第一次select的时候，对最后一次确切提交的事务的结果做快照
  解决了不可以重复读，但是仍有可能出现幻读。例如，事务A、事务B开启，事务A中增加了一条数据并提交，事务B反复查询看不到新的数据，但是可以使用update语句更新成功，再查询就看到了。这也是幻读，如同出现了幻觉。
- READ COMMITTED，在事务中，每次select可以读取到别的事务刚提交成功的新的数据。因为读到的是提交后的数据，解决了脏读，但是不能解决 不可重复读 和 幻读 的问题。因为其他事务前后修改了数据或增删了数据。
- READ UNCOMMITTED，能读取到别的事务还没有提交的数据，完全没有隔离性可言，出现了脏读，当前其他问题都可能出现。

事务语法

START TRANSACTION或BEGIN开始一个事务，START TRANSACTION是标准SQL的语法。

使用**COMMIT**提交事务后，变更成为永久变更。

ROLLBACK可以在提交事务之前，回滚变更，事务中的操作就如同没有发生过一样（原子性）。

MySQL中，默认采用自动提交，每一条查询语句都会作为一个事务提交。SET AUTOCOMMIT语句可以禁用或启用默认的autocommit模式，用于当前连接。SET AUTOCOMMIT = 0禁用自动提交事务。如果开启自动提交，如果有一个修改表的语句执行后，会立即把更新存储到磁盘。

```
-- 查询隔离级别
SELECT @@global.tx_isolation; -- 全局
show global variables like '%iso%';

SELECT @@session.autocommit;  -- 会话
SELECT @@tx_isolation;        -- 会话
show session variables like '%iso%';
show variables like '%isolation%';


-- 设置会话级或者全局隔离级别
```

```
SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL
{READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE}

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


SELECT @@autocommit;
-- 禁用自动提交
SET AUTOCOMMIT = 0
```

FOR UPDATE和锁

数据库不可避免会发生并发，有人读取，就有人写入，如果读写同一张表、同一条记录，就会有冲突。

- 读锁，共享锁，多用户同一时刻读取同一个资源，互不干扰
- 写锁，排他锁，有更高优先级，它会阻塞其它用户对该资源读写操作而使用的读锁或写锁

SELECT ... For Update会把行进行写锁定，这是排它锁。
使用主键明确记录行，那么就对存在的主键的记录使用行级锁。例如id=100
使用主键，但不能确定记录，使用表级锁，例如id <> 3。
条件中不使用主键，会使用表级锁。例如，name='tom'

这是悲观锁，我怕别人在我用的时候抢资源，先锁上，独占。悲观锁往往用在数据库的锁机制中，因为独占，所以会影响并发。所以，For Update非要使用，请一定要保证时间短，且一定利用行级锁。
乐观锁，就是心宽，认为极少冲突，只有写入才加锁。但需要检测数据冲突。

数据仓库和数据库的区别

本质上来说没有区别，都是存放数据的地方。
但是数据库关注数据的持久化、数据的关系，为业务系统提供支持，事务支持；
数据仓库存储数据的是为了分析或者发掘而设计的表结构，可以存储海量数据。

数据库存储在线交易数据OLTP（联机事务处理OLTP，On-line Transaction Processing）；数据仓库存储历史数据用于分析OLAP（联机分析处理OLAP，On-Line Analytical Processing）。

数据库支持在线业务，需要频繁增删改查；数据仓库一般囤积历史数据支持用于分析的SQL，一般不建议删改。

存储过程、触发器

存储过程（Stored Procedure），数据库系统中，一段完成特定功能的SQL语句。编写成类似函数的方式，可以传参并调用。支持流程控制语句。

触发器（Trigger），由事件触发的特殊的存储过程，例如insert数据时触发。

这两种技术，虽然是数据库高级内容，性能不错，但基本很少用了。
它们移植性差，使用时占用的服务器资源，排错、维护不方便。
最大的原因，不太建议把逻辑放在数据库中。