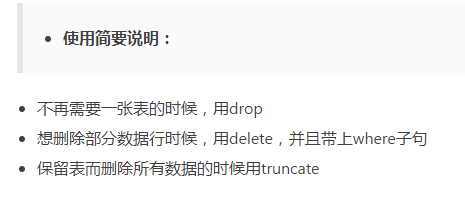
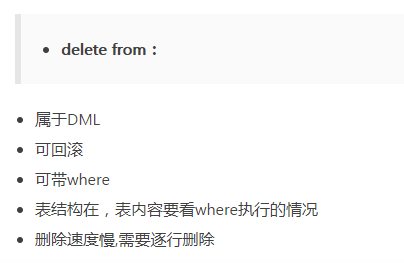
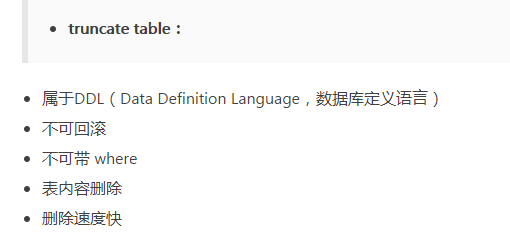
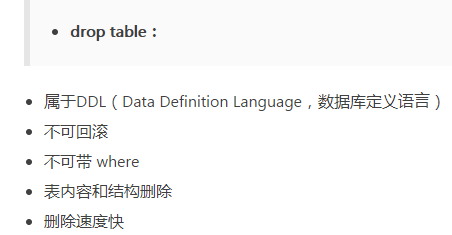
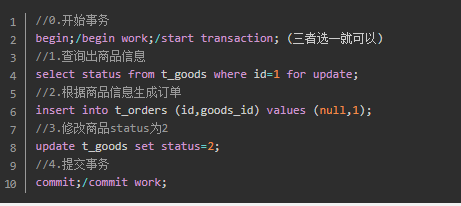
事务

1. 事务简单来说就是一个**Session中所进行的所有操作，要么同时成功，要么同时失败**。
2. 作为单个逻辑工作单元执行的一系列操作，满足以下四个特征（ACID）：
3. 原子性（Atomicity）：事务作为一个整体，要么全部执行，要么全部不执行； 一致性（Consistency）：保证数据库状态从一个一致状态转变为另一个一致状态；隔离性（Isolation）：多个事务并发执行时，一个事务的执行不应影响其他事务的执行；持久性（Durability）：一个事务一旦提交，对数据库的新修改应该永久保存。 例如：用户A向用户B转账
4. 事务并发问题有哪几种？ 答：①丢失更新：一个事务的更新覆盖了另一个事务的更新；②脏读：一个事务读取了另一个事务未提交的数据；③不可重复读：不可重复读的重点是修改，同样条件下两次读取结果不同，也就是说，被读取的数据可以被其它事务修改；④幻读：重点在于新增或者删除，同样条件下两次独处的记录数据不一样。
5. 事务的隔离级别有哪几种？ 答：隔离级别决定了一个session中的事务可能对另一个session中的事务的影响。ANSI标准定义了四个隔离级别。①读未提交（READ UNCOMMITTED）： 最低级别的隔离，通常又称为dirty read，它允许一个事务读取另一个事务还没commit的数据，这样可能会提高性能，但也会导致脏数据。②读已提交（READ COMMITTED）： 在一个事务中只允许对其它事务已经commit的记录可见，该隔离级别不能避免不可重复读问题。③**可重复读（REPEATABLE READ）：在一个事物开始后，其它事务对数据库的修改在本事务中不可见，直到本事务commit或rollback。但是，其它事务的insert/delete操作对该事务是可见的，也就是该隔离几倍并不能避免幻读问题。在一个事务中重复select的结果一样，除非本事务中update数据库**。④序列化（SERIALIZABLE）：最高级别的隔离，只允许事务串行执行。
6. 什么是视图？ 视图是一种**虚拟的表**，具有和物理表相同的功能。可以对视图**进行增删改查操作**，视图通常**是有一个表或者多个表的行或列的子集**。对视图的修改**不影响基本表**。它使得我们获取数据更容易，相比多表查询。 如下两种情景一般会使用到视图：① **不希望访问者获取整个表的数据**，只暴露部分字段给访问者，所以就建立一个虚表，也就是视图。 ② **查询的数据来源在不同的表**，而查询者希望以统一的方式查询，这样也可以建立视图，把多个表查询结果联合起来，查询者只需要直接从视图中获取数据，不必考虑数据来源不同表所带来的差异。 注意：视图是在数据库中创建的，而不是用代码创建的。
7. drop delete truncate的区别？ drop直接删除表；truncate删除表中数据，再插入时自增长id又从1开始； delete删除表中数据，可以加where字句。



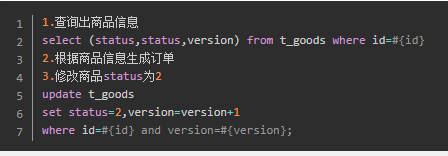
1. **数据库中的乐观锁和悲观锁？** 数据库管理系统（DBMS）中的**并发控制**的任务是确保在多个事务同时存取数据库中同一数据时不破坏事务的隔离性和统一性以及数据库的统一性。 **乐观并发控制（乐观锁）和悲观并发控制（悲观锁）是并发控制主要采用的技术手段**。
2. **悲观锁：假定会发生并发冲突，屏蔽一切可能违反数据完整性的操作**。 实现方式： 悲观锁是一种利用数据库内部机制提供的锁的方式，也就是**对更新的数据加锁**，这样在并发期间一旦有一个事务持有了数据库记录的锁，其它的线程将不能对数据进行更新了。 **MySQL InnoDB中使用悲观锁**： 要使用悲观锁，我们必须关闭mysql数据库的自动提交属性，因为MySQL默认使用autocommit模式，也就是说，当你执行一个更新操作后，MySQL会立刻将结果进行提交。 set autocommit = 0;



上面的查询语句中，我们使用了 select…for update 的方式，这样就**通过开启排他锁的方式实现了悲观锁**。此时在t\_goods表中，id为1的 那条**数据就被我们锁定了**，其它的事务必须等本次事务提交之后才能执行。这样我们可以保证当前的数据不会被其它事务修改。使用 select…for update 会把数据给锁住，不过我们需要注意一些**锁的级别**，MySQL InnoDB默认行级锁。行级锁都是基于索引的，如果一条SQL语句用不到索引是不会使用行级锁的，会使用表级锁把整张表锁住。

**优点与不足：** 悲观并发控制实际上是“**先取锁再访问**”的保守策略，为数据处理的安全提供了保证。但是在效率方面，处理加锁的机制会让数据库产生额外的开销，还有增加产生死锁的机会；另外，在只读型事务处理中由于不会产生冲突，也没必要使用锁，这样做只能增加系统负载；还有会降低了并行性，一个事务如果锁定了某行数据，其他事务就必须等待该事务处理完才可以处理那行数据。

1. **乐观锁：假设不会发生并发冲突，只在提交操作时检查是否违反数据完整性。** 实现方式：乐观锁是一种不会阻塞其他线程并发的控制，它不会使用数据库的锁进行实现，它的设计里面由于不阻塞其他线程，所以并不会引起线程频繁挂起和恢复，这样便能够提高并发能力，所以也有人把它称为**非阻塞锁**。一般的实现乐观锁的方式就是**记录数据版本**。 **使用版本号实现乐观锁： 使用版本号时，可以在数据初始化时指定一个版本号，每次对数据的更新操作都对版本号执行+1操作。并判断当前版本号是不是该数据的最新的版本号。**



优点和不足： 乐观并发控制相信事务之间的数据竞争(data race)的概率是比较小的，因此尽可能直接做下去，直到提交的时候才去锁定，所以不会产生任何锁和死锁。但如果直接简单这么做，还是有可能会遇到不可预期的结果，例如两个事务都读取了数据库的某一行，经过修改以后写回数据库，这时就遇到了问题。

1. **超键、候选键、主键、外键分别是什么？** 
   1. 超键： 在关系中能唯一标识元组的属性集称为关系模式的超键。一个属性可以为作为一个超键，多个属性组合在一起也可以作为一个超键。超键包含候选键和主键。
   2. 候选键： 是最小超键，即没有冗余元素的超键。
   3. 主键： 数据库表中对储存数据对象**予以唯一和完整标识的数据列或属性的组合**。一个数据列**只能有一个主键**，且主键的**取值不能缺失**，即不能为空值（Null）。
   4. 外键： 在一个表中存在另一个表中的主键称为此表的外键。

**候选码和主码：**

例子：邮寄地址（城市名，街道名，邮政编码，单位名，收件人）

它有**两个候选键:**{城市名，街道名} 和 {街道名，邮政编码} ；如果我选取{城市名，街道名}作为唯一标识实体的属性，那么{城市名，街道名} 就是**主码(主键)**。

1. **SQL约束有哪几种？** 
   1. NOT NULL ： 用于控制字段内容一定不能为空（NULL）。
   2. UNIQUE ： 用于控制字段内容不能重复，一个表中可以有多个UNIQUE约束
   3. PRIMARY KEY : 也是用于控制字段内容不能重复，但它在一个表中只能出现一次
   4. FOREIGN KEY : 用于预防破坏表之间连接的动作，也能防止非法数据插入外键列，因为它必须是它指向的那个表中的值之一。
   5. CHECK ： 用于控制字段值得范围