<http://blog.csdn.net/zdwzzu2006/article/details/5947062>

版权声明：本文为博主原创文章，未经博主允许不得转载。

        事务（Transaction）是并发控制的基本单位。所谓的事务，它是一个操作序列，这些操作要么都执行，要么都不执行，它是一个不可分割的工作单位。例如，银行转账工作：从一个账号扣款并使另一个账号增款，这两个操作要么都执行，要么都不执行。所以，应该把它们看成一个事务。事务是数据库维护数据一致性的单位，在每个事务结束时，都能保持数据一致性。

       针对上面的描述可以看出，事务的提出主要是为了解决并发情况下保持数据一致性的问题。

       事务具有以下4个基本特征。

●   Atomic（原子性）：事务中包含的操作被看做一个逻辑单元，这个逻辑单元中的操作要么全部成功，要么全部失败。

●   Consistency（一致性）：只有合法的数据可以被写入数据库，否则事务应该将其回滚到最初状态。

●   Isolation（隔离性）：事务允许多个用户对同一个数据进行并发访问，而不破坏数据的正确性和完整性。同时，并行事务的修改必须与其他并行事务的修改相互独立。

●   Durability（持久性）：事务结束后，事务处理的结果必须能够得到固化。

**2.事务的语句**　开始事物：BEGIN TRANSACTION  
　提交事物：COMMIT TRANSACTION  
　回滚事务：ROLLBACK TRANSACTION  
**3.事务的4个属性**     ①原子性(Atomicity)：事务中的所有元素作为一个整体提交或回滚，事务的个元素是不可分的，事务是一个完整操作。  
  ②一致性(Consistemcy)：事物完成时，数据必须是一致的，也就是说，和事物开始之前，数据存储中的数据处于一致状态。保证数据的无损。  
  ③隔离性(Isolation)：对数据进行修改的多个事务是彼此隔离的。这表明事务必须是独立的，不应该以任何方式以来于或影响其他事务。  
  ④持久性(Durability)：事务完成之后，它对于系统的影响是永久的，该修改即使出现系统故障也将一直保留，真实的修改了数据库  
**4.事务的保存点**     SAVE TRANSACTION 保存点名称 --自定义保存点的名称和位置  
     ROLLBACK TRANSACTION 保存点名称 --回滚到自定义的保存点

 其他高手的一些补充：

 事务的标准定义： 指作为单个逻辑工作单元执行的一系列操作，而这些逻辑工作单元需要具有原子性，  一致性，隔离性和持久性四个属性，统称为ACID特性。

所谓事务是用户定义的一个数据库操作序列,这些操作要么全做要么全不做,是一个不可分割的工作单位。例如,在关系数据库中,一个事务可以是一条SQL语句、一组SQL语句或整个程序。   
事务和程序是两个概念。一般地讲,一个程序中包含多个事务。  
事务的开始与结束可以由用户显式控制。如果用户没有显式地定义事务,则由DBMS按缺省规定自动划分事   
务。在SQL语言中,定义事务的语句有三条：   
BEGIN TRANSACTION   
COMMIT   
ROLLBACK

同生共死。。  
显示事务被用begin transaction 与 end transaction 标识起来，其中的 update 与 delete 语句或者全部执行或者全部不执行。。 如：  
begin transaction T1  
update student  
set name='Tank'  
where id=2006010  
delete from student  
where id=2006011  
commit  
简单地说，事务是一种机制，用以维护数据库的完整性。

其实现形式就是将普通的SQL语句嵌入到Begin Tran...Commit Tran 中（或完整形式 Begin Transaction...Commit Transaction)，当然，必要时还可以使用RollBack Tran 回滚事务，即撤销操作。

利用事务机制，对数据库的操作要么全部执行，要么全部不执行，保证数据库的一致性。需要使用事务的SQL语句通常是更新和删除操作等。

end transaction T1

关于savepoint

用户在事务（transaction）内可以声明（declare）被称为保存点（savepoint）  
的标记。保存点将一个大事务划分为较小的片断。

用户可以使用保存点（savepoint）在事务（transaction）内的任意位置作标  
记。之后用户在对事务进行回滚操作（rolling back）时，就可以选择从当前  
执行位置回滚到事务内的任意一个保存点。例如用户可以在一系列复杂的更  
新（update）操作之间插入保存点，如果执行过程中一个语句出现错误，用  
户 可以回滚到错误之前的某个保存点，而不必重新提交所有的语句。

在开发应用程序时也同样可以使用保存点（savepoint）。如果一个过程  
（procedure）内包含多个函数（function），用户可以在每个函数的开始位置  
创建一个保存点。当一个函数失败时， 就很容易将数据恢复到函数执行之前  
的状态，回滚（roll back）后可以修改参数重新调用函数，或执行相关的错误  
处理。

当事务（transaction）被回滚（rollback）到某个保存点（savepoint）后，  
Oracle将释放由被回滚语句使用的锁。其他等待被锁资源的事务就可以继续  
执行。需要更新（update）被锁数据行的事务也可以继续执行。

将事务（transaction）回滚（roll back）到某个保存点（savepoint）的过程如  
下：  
1. Oracle 回滚指定保存点之后的语句  
2. Oracle 保留指定的保存点，但其后创建的保存点都将被清除  
3. Oracle 释放此保存点后获得的表级锁（table lock）与行级锁（row  
lock），但之前的数据锁依然保留。

被部分回滚的事务（transaction）依然处于活动状态，可以继续执行。

一个事务（transaction）在等待其他事务的过程中，进行回滚（roll back）到  
某个保存点（savepoint）的操作不会释放行级锁（row lock）。为了避免事务  
因为不能获得锁而被挂起，应在执行 UPDATE 或 DELETE 操作前使用 FOR  
UPDATE ... NOWAIT 语句。（以上内容讲述的是回滚保存点之前所获得的  
锁。而在保存点之后获得的行级锁是会被释放的，同时保存点之后执行的  
[**SQL**](javascript:;) 语句也会被完全回滚）。