文章描述了两个方面的内容，数据库[事务](http://blog.nosqlfan.com/tags/%e4%ba%8b%e5%8a%a1)的约束条件，以及分布式系统中事务性保证所用到的基础的[两阶段提交](http://blog.nosqlfan.com/tags/%e4%b8%a4%e9%98%b6%e6%ae%b5%e6%8f%90%e4%ba%a4)策略。对理解分布式系统的一致性模型有一定的帮助。

* 1 事务的ACID

事务是保证数据库从一个一致性的状态永久地变成另外一个一致性状态的根本，其中，ACID是事务的基本特性。

A是Atomicity，原子性。一个事务往往涉及到许多的子操作，原子性则保证这些子操作要么都做，要么都不做，而不至于出现事务的部分操作成功，而另外一部分操作没有成功。如果事务在执行的过程中发生错误，那么数据库将回滚到事务发生之前的状态。比如银行的转账服务，这个事务的最终结果一定是：某个账户的余额增加了x，而另外一个账户的余额减少了x，或者两个账户的余额未发生变化。而不会出现其他情况。

C是Consistency，一致性。一致性是指事务发生前和发生以后，都不会破坏数据库的约束关系，保证了数据库元素的正确性、有效性和完整性。这种约束关系可以是数据库内部的约束，比如数据库元素的值必须在一定的范围内，也可以是应用带来的约束，比如转账以后银行账户的余额不能为负数。

I是Isolation，隔离性。一个事务的操作在未提交以前，是不会被并行发生的其他事务访问到的。也就是说，数据库操作不会看到某个事务的中间操作结果，比如转账过程中，用户是不能查询到一个账户余额减少了，而另外一个账户余额未发生变化的情况。

D是Durability，持久性。事务完成以后，它对数据库的影响是永久性的，即使在数据库系统发生宕机或者其他故障的情况下，这种影响也会得到保持。

* 2 两阶段提交

在分布式系统中，事务往往包含有多个参与者的活动，单个参与者上的活动是能够保证原子性的，而多个参与者之间原子性的保证则需要通过两阶段提交来实现，两阶段提交是分布式事务实现的关键。

很明显，两阶段提交保证了分布式事务的原子性，这些子事务要么都做，要么都不做。而数据库的一致性是由数据库的完整性约束实现的，持久性则是通过commit日志来实现的，不是由两阶段提交来保证的。至于两阶段提交如何保证隔离性，可以参考Large-scale Incremental Processing Using Distributed Transactions and Notifications中两阶段提交的具体实现。

两阶段提交的过程涉及到协调者和参与者。协调者可以看做成事务的发起者，同时也是事务的一个参与者。对于一个分布式事务来说，一个事务是涉及到多个参与者的。具体的两阶段提交的过程如下：

第一阶段：

首先，协调者在自身节点的日志中写入一条的日志记录，然后所有参与者发送消息prepare T，询问这些参与者（包括自身），是否能够提交这个事务；

参与者在接受到这个prepare T 消息以后，会根据自身的情况，进行事务的预处理，如果参与者能够提交该事务，则会将日志写入磁盘，并返回给协调者一个ready T信息，同时自身进入预提交状态状态；如果不能提交该事务，则记录日志，并返回一个not commit T信息给协调者，同时撤销在自身上所做的数据库改；

参与者能够推迟发送响应的时间，但最终还是需要发送的。

第二阶段：

协调者会收集所有参与者的意见，如果收到参与者发来的not commit T信息，则标识着该事务不能提交，协调者会将Abort T 记录到日志中，并向所有参与者发送一个Abort T 信息，让所有参与者撤销在自身上所有的预操作；

如果协调者收到所有参与者发来prepare T信息，那么协调者会将Commit T日志写入磁盘，并向所有参与者发送一个Commit T信息，提交该事务。若协调者迟迟未收到某个参与者发来的信息，则认为该参与者发送了一个VOTE\_ABORT信息，从而取消该事务的执行。

参与者接收到协调者发来的Abort T信息以后，参与者会终止提交，并将Abort T 记录到日志中；如果参与者收到的是Commit T信息，则会将事务进行提交，并写入记录

一般情况下，两阶段提交机制都能较好的运行，当在事务进行过程中，有参与者宕机时，他重启以后，可以通过询问其他参与者或者协调者，从而知道这个事务到底提交了没有。当然，这一切的前提都是各个参与者在进行每一步操作时，都会事先写入日志。

唯一一个两阶段提交不能解决的困境是：当协调者在发出commit T消息后宕机了，而唯一收到这条命令的一个参与者也宕机了，这个时候这个事务就处于一个未知的状态，没有人知道这个事务到底是提交了还是未提交，从而需要数据库管理员的介入，防止数据库进入一个不一致的状态。当然，如果有一个前提是：所有节点或者网络的异常最终都会恢复，那么这个问题就不存在了，协调者和参与者最终会重启，其他节点也最终也会收到commit T的信息。

* 3 日志

数据库日志保证了事务执行的原子性和持久性，日志类型可以分为redo log，undo log，undo/redo log。关于这几种日志形式的具体介绍，可以参照