1. 长事务是基于怎样的背景提出的？

一般事务大多不是交互式，并且持续时间短。但是当事务涉及人机交互的数据库系统时，会出现很大不同，就出现了长事务。

1. 长事务有那些重要特性？
2. 持续时间长。（这点很好理解，主要是因为人加入系统后从机器的观点来看人的响应速度很慢，持续时间很长。这点可以用GITHUB中潘涛处理版本控制来说明）
3. 未提交数据的曝光。（比较简单的一个例子是word中写的文字被用户看到了，但是这些文字还未被提交。另外，几个人合作的一项设计中，用户之间可能在提交之前交换数据也是一个例子）
4. 子任务。（用户会把事务分成一组子任务，用户会终止其中的某个，但是不终止整个事务）
5. 可恢复性。（系统崩溃使长事务终止不能接受，事务必须被恢复到崩溃之前不久的一个状态。注意这里的数据有大量未保存或者说未提交的。例子是写文档时突然死机）
6. 性能。（普通事务的性能体现在吞吐量，而有人参与的长事务的性能好坏体现在反应时间。IPHONE和nokia的区别）

由于这些特性的存在，之前提到的一些技术必须改动以适应交互式长事务。

1. 只允许可串行化调度已经不切实际。并发控制协议会对长事务产生负面影响。
2. 两阶段封锁会造成等待时间过长。因为持有锁的时间过长。
3. 基于图的协议：易发生长时间锁等待。
4. 基于时间戳的协议。需要终止，如果一个长事务终止是用户不可接受的。
5. 有效性确认。同上。

强制要求事务的原子性或者导致长时间等待的可能性增大，或者导致发生级联回滚。

1. 并发控制的修正：并非所有保持数据库一致性的调度都是可串行化的。
2. 解决：嵌套事务和多级事务。

将一个长事务看成一组相关的子事务。提高并行度，并且可以不回滚整个长事务就能处理由于事务中止，系统崩溃等造成的子事务失败。

多级事务和嵌套事务的概念。

1. 补偿事务：将未提交的数据暴露出去就会导致级联回滚，可使用补偿事务。（这个大概意思就是逆操作，有点像展硕的恢复）
2. 实现问题：举例：减少保证大数据项可恢复性的开销。