在lazily replicated的系统保证快照隔离可能会导致事务inversions，当事务看到旧数据就会发生inversions。在centralized 数据库服务器中提供Strong 快照隔离可以避免，但是在lazy replicated系统就维护强快照隔离的成本很高。
Strong Session Snapshot isolation
One-copy serializability(1SR)是replicate data的标准的正确性标准
SI是弱于1SR的事务保证

两个无写写冲突的事务为什么在lazy replicate系统中出现问题?
是因为LR系统中每个site的事务提交顺序可能不同是吗

在所有site上以相同的顺序执行non-conflict update的事务

贡献
1.通过利用本地SI并发控制来对update事务进行排序， 从而在lazily replicated的系统保证global SI
通过capture本地并发控制的更新事务的schedule，然后用这个schedule更新其他site

lazily 同步数据库系统的主要缺点就是client不保证能看到自己的更新

SI：可能发生事务inversion，可能会有write skew
Strong SI：保证事务读最新数据库状态，不会发生事务inversions
Strong Session SI：防止事务在一个client session中inversions ，而不是在session间

Strong SI太严格了，而且其中某些Ti Tj之间的约束是没有必要的，我们用Session来表示哪些ordering constraints重要，哪些不重要
我们将事务划分到几个session中，同一个session的事务需要Strong SI，而不同的session不必要Strong SI

Ordering write
解决写冲突的一种办法是FCW

3.1synchronization overview
为了确保更新事务和相应的refresh事务start和commit在各个site都是相同的顺序必须有下面的关系

我觉得只需要第一点就行了吧，commit时间一定在start之后，第一点保证了，剩下两点肯定也成立

1. start p(T2) > commit p(T1) =》 start s(R2) > commit s(R1)
2. commit p(T2) > start p(T1) =》commit s(R2) > start s(R1)
3. commit p(T2) > commit p(T1) =》 commit s(R2) > commit s(R1)

Algorithm 3.1 Primary Update Propagation：按primary log sequence提交顺序执行所有操作

3.3 Secondary Refresh

每个secondary site都会运行一个refresh进程或者refresher，从本地更新队列中取出更新，然后应用到secondary数据库中。

Algorithm 3.2 Secondary Refresh Algorithm：从更新队列中读到start(T)就先阻塞然后提交所有的pendingQueue的东西，然后开始refresh(T)，如果从更新队列读到commitT就加到pendingQueue里，并调用Algorithm3.3执行事务，如果是读到abort就进行abort操作。这个算法确保了一个事务内的更新都是在一个refresh事务内完成的

Algorithm 3.3 Applicator Thread：就是一个应用事务的所有更新操作

3.4Concurrency and Failure

队列可能出现写写冲突，为了避免这个造成的abort，propagated record可以用数据库外部的队列存储。但是这引入另一个问题，如果secondary site 失败了，这个队列的更新和refresh状态就会被丢失失，（没看懂）

Theorem 3.1 对于每个secondary site来说，如果database state中第0个primary site和第0个secondary site是一样的话，那么后面任何的对应的site都相等

Theorem3.2 The weak SI system guarantees global weak SI

4 ENFORCING GLOBAL STRONG SESSION SI

客户端的事务请求序列组成的session

ALG Strong Session SI算法：为每个session维护一个sequence number，我的理解是这个seq代表每个session中最近commit的时间，用于控制只读事务的执行顺序

Theorem4.1每个site在本地保证强SI，ALG Strong Session SI算法结合weakSI系统的propagation和refresh mechanism，来保证全局Strong session SI