飞机选座

假设一名旅客已经预定好行程，并且现在正在为每次航班选择座位。

许多航空公司的网站允许用户浏览各个航班并选择座位，然后要求用户确认其选择。而与此同时，其他旅客也可能正在选择同一架航班的座位或者更改他们所选择的座位。因此，该旅客所看到的空余座位实际上是变化的。

即使两名旅客同时选择座位，他们很可能选择不同的座位，如果是这样就不会发生真正的冲突。

如果两名旅客同时执行的座位选择事实上选择的是相同的座位，则其中一位将不会获得他所选择的座位。

不过，这种情况很容易解决，只要在更新的空余座位信息上，要求这名旅客重新执行选择即可。

通过一个时刻只允许一名旅客选择一次特定航班的座位可以保证可串行化。

然而，这样做可能会带来严重的延迟，特别是一名旅客花费很长时间来做出选择的话。

银行转账

假定转账由程序 P 实现，P 执行如下步骤：

1. 将这笔钱加到账户 2。
2. 检查账户 1 是否有足够的钱。
3. 如果没有足够的钱，将这些钱从账户 2 移走，结束。
4. 如果有足够的钱，从账户 1 减去这些钱，结束。

如果程序 P 是串行执行的，那么把钱临时放入账户 2 是无关紧要的。没人会看到这笔钱，若不能转账则减去这笔钱。

但是，假定可能有脏读。想象有三个账户：A1、A2 和 A3，分别有 100、200、300 美元。假设事务 T1 执行程序 P 从 A1 向 A2 转账 150 美元。在大约同一时刻，事务 T2 运行程序 P 从 A2 向 A3 转账 250 美元。

事件序列可能如下：

1. T2 执行步骤 1 将 250 美元加到 A3，现在 A3 有 550 美元。
2. T1 执行步骤 1 将 150 美元加到 A2，现在 A2 有 350 美元。
3. T2 执行步骤 2 的检查，发现 A2 有足够的资金（350 美元），能够从 A2 向 A3 转账 250 美元。
4. T1 执行步骤 2 的检查，发现 A1 没有足够的资金（100 美元），不能从 A1 向 A2 转账 150 美元。
5. T2 执行步骤 2b。从 A2 减去 250 美元，A2 现在有 100 美元，结束。
6. T1 执行步骤 2a。从 A2 减去 150 美元，A2 现在有 -50 美元，结束。

钱的总数没变；这三个账户仍然总共有 600 美元。但是因为 T2 在上述六个步骤中的第三步中读取了脏数据，所以导致一个账户变为负值，而这一步本来的目的是检查第一个账户是否有足够的资金。