13.4. 应用级别的数据完整性检查

对于使用读已提交事务的数据完整性强制业务规则非常困难，因为对每一个语句数据视图都在变化，并且如果一个写冲突发生即使一个单一语句也不能把它自己限制到该语句的快照。

虽然一个可重复读事务在其执行期间有一个稳定的数据视图，在使用MVCC快照进行数据一致性检查时也有一个小问题，它涉及到被称为*读/写冲突*的东西。如果一个事务写数据并且一个并发事务尝试读相同的数据（不管是在写之前还是之后），它不能看到其他事务的工作。读取事务看起来是第一个执行的，不管哪个是第一个启动或者哪个是第一个提交。如果就到此为止，则没有问题，但是如果读取者也写入被一个并发事务读取的数据，现在有一个事务好像是已经在前面提到的任何一个事务之前运行。如果看起来最后执行的事务实际上第一个提交，在这些事务的执行顺序图中很容易出现一个环。当这样一个环出现时，完整性检查在没有任何帮助的情况下将不会正确地工作。

正如第 13.2.3 节中提到的，可序列化事务仅仅是可重复读事务增加了对读/写冲突的危险模式的非阻塞监控。当检测到一个可能导致表面的执行顺序中产生环的模式，涉及到的一个事务将被回滚来打破该环。

13.4.1. 用可序列化事务来强制一致性

如果可序列化事务隔离级别被用于所有需要一个一致数据视图的写入和读取，不需要其他的工作来保证一致性。在PostgreSQL中，来自于其他环境的被编写成使用可序列化事务来保证一致性的软件应该“只工作”在这一点上。

当使用这种技术时，如果应用软件通过一个框架来自动重试由于序列化错误而回滚的事务，它将避免为应用程序员带来不必要的负担。把`default_transaction_isolation`设置为`serializable`可能是个好主意。通过触发器中的事务隔离级别检查来采取某些动作来保证没有其他事务隔离级别被使用（由于疏忽或者为了破坏完整性检查）也是明智的。

性能建议见第 13.2.3 节。

警告

这个级别的使用可序列化事务的完整性保护还没有扩展到热备份模式（第 27.4 节）。由于这个原因，那些使用热备份的系统可能想要在主数据库上使用可重复读和显式锁定。

13.4.2. 使用显式锁定强制一致性

当可以使用非可序列化写时，要保证一行的当前有效性并保护它不受并发更新的影响，我们必须使用SELECT FOR UPDATE、SELECT FOR SHARE或一个合适的LOCK TABLE 语句（SELECT FOR UPDATE和SELECT FOR SHARE锁只针对并发更新返回行，而LOCK TABLE会锁住整个表）。当从其他环境移植应用到PostgreSQL时需要考虑这些。

关于这些来自其他环境的转换还需要注意的是SELECT FOR UPDATE不保证一个并发事务将不会更新或删除一个被选中的行。要在PostgreSQL中这样做，你必须真正地更新该行，即便没有值需要被改变。SELECT FOR UPDATE ***临时阻塞***其他事务，让它们不能获取该相同的锁或者执行一个会影响被锁定行的UPDATE或DELETE，但是一旦正持有该所锁的事务提交或回滚，一个被阻塞的事务将继续执行冲突操作，除非当锁被持有时一个该行的实际UPDATE被执行。

在非可序列化MVCC环境下，全局有效性检查需要一些额外的考虑。例如，一个银行应用可能会希望检查一个表中的所有扣款总和等于另外一个表中的收款总和，同时两个表还会被更新。比较两个连续的在读已提交模式下不会可靠工作的SELECT sum(...)命令，因为第二个查询很可能会包含没有被第一个查询考虑的事务提交的结果。在一个单一的可重复读事务里进行两个求和则给出在可串行化事务开始之前提交的所有事务产生的准确结果 — 但有人可能会合理地置疑在结果被递交的时候，它们是否仍然相关。 如果可重复读事务本身在尝试做一致性检查之前应用了某些变更，那么检查的有用性就更加值得讨论了， 因为现在它包含了一些（但不是全部）事务开始后的变化。 在这种情况下，一个小心的人可能希望锁住所有需要检查的表，这样才能获得一个无可置疑的当前现状的图像。 一个SHARE模式（或者更高）的锁保证在被锁定表中除了当前事务所作的更改之外，没有未提交的更改。

还要注意如果某人正在依赖显式锁定来避免并发更改，那么他应该使用读已提交模式，或者是在可重复读模式里在执行命令之前小心地获取锁。 在可重复读事务里获取的锁保证了不会有其它修改该表的事务正在运行，但是如果事务看到的快照在获取锁之前， 那么它可能早于表中一些现在已经提交的更改。 一个可重复读事务的快照实际上是在它的第一个查询或者数据修改命令（SELECT、INSERT、UPDATE或DELETE）开始的时候冻结的，因此我们可以在快照冻结之前显式地获取锁。

上一页 上一级 下一页
13.3. 显式锁定 起始页 13.5. 提醒