# Go语言学习笔记

如何保证在不停机的情况下保证迁移数据的一致性？

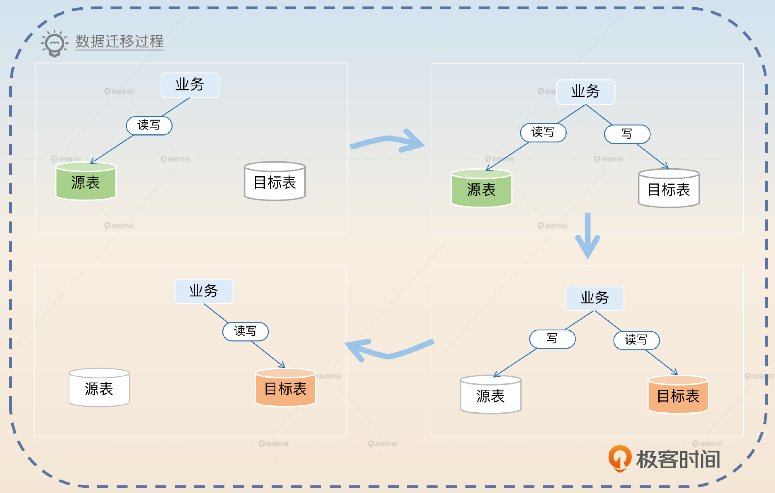
## 导入

我进公司的时候，刚好遇上单库拆分分库分表。我主要负责的事情就是设计一个数据迁移方案，把数据从单库迁移到分库分表上。

## 解决方案总述

1. 创建目标表。
2. 用源表的数据初始化目标表。
3. 执行一次校验，并且修复数据，此时用源表数据修复目标表数据。
4. 业务代码开启双写，此时读源表，并且先写源表，数据以源表为准。
5. 开启增量校验和数据修复，保持一段时间。
6. 切换双写顺序，此时读目标表，并且先写目标表，数据以目标表为准。
7. 继续保持增量校验和数据修复。
8. 切换为目标表单写，读写都只操作目标表。

如果不考虑数据校验，那么整个数据迁移过程是这样的。



## 第1步 初始化目标表数据

1. 在创建了一个目标表之后，第一步是先尝试初始化目标数据
   1. 第一个问题就是如何拿到源表数据，可以有两种方式：
   2. 第一种，使用源表的历史备份
   3. 第二种，使用备份工具导出源表数据
   4. 最终选择第二种，因为源表的历史备份时间比较久了
   5. 使用的是 mysqldump 工具。mysqldump 是一个开源的逻辑备份工具，优点是使用简单，能够直接导出整个数据库。缺点则是导出和导入的速度都比较慢，尤其是在数据量非常大的情况下。所以我针对 mysqldump 做了一些优化，来提高导出和导入的性能。
      1. 加快导出速度能做的事情并不多
         1. 主要就是开启 extended-insert 选项，将多行合并为一个 INSERT 语句
      2. 加快导入速度就可以做比较多的事情
         1. 关闭唯一性检查和外键检查，源表已经保证了这两项，所以目标表并不需要检查。
         2. 关闭 binlog，毕竟导入数据用不着 binlog。
         3. 调整 redo log 的刷盘时机，把 innodb\_flush\_log\_at\_trx\_commit 设置为 0。

## 第一次校验与修复

xxx

1st

#### xxx

1. xxx
2. xxxxzx
   1. xxx
      1. xx
   2. xx
      1. xxx
      2. xx
      3. xx
         1. xxx
   3. xx
3. xxx
4. xxxx

##### xxx

xxx xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx

xxx