3、亿级大表单库如何平滑迁移到分库分表?

一、业务背景

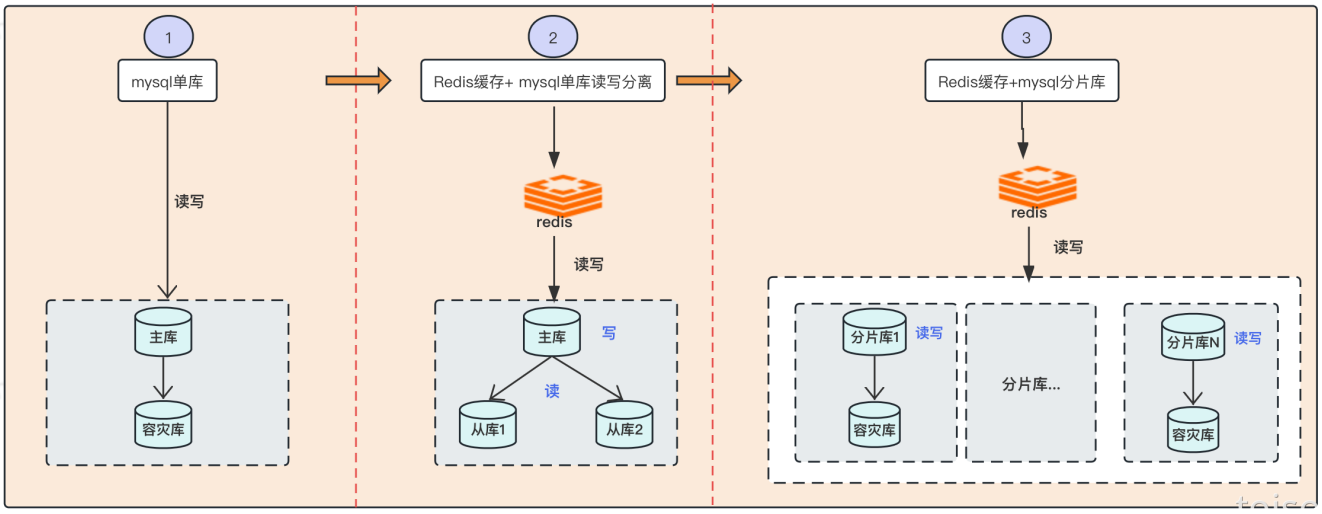
1、前言

产品初期，一般的系统没什么用户量，随着线上渠道不断的推广，业务发展越来越快，注册用户也越来越多，数据开始呈现爆发式增长，比如我们平时使用比较多的几个系统：淘宝、拼多多、微信、美团等，都是这样的发展规律。我有幸在21年独立负责一个偏C端的预付费产品：基于亿级用户的动态折扣系统，经过短短2年的发展，折扣订单的数据量从日均2w增长到100w+，1个月从60w增到到3000w+，而

且还在持续增长中。系统数据库的读写压力越来越大，业务高峰期，数据库CPU频繁告警。为了保障数据库稳定提供读写能力，在业务侧，我们说服业务方限制客户查询数据范围（只提供3个月内的查询记录），在技术侧，我们不断优化和调整系统数据存储方案，从传统的单数据库--》redis + 数据库读写分离架构--》redis + 分片数据库架构，通过不断的演进，逐步的提升数据库处理数据的能力，为公司未来几年的快速发展，提供了有力稳定性保障。

2、数据存储演进过程

动态折扣系统从2021年5月份上线以后，系统底层数据存储架构共经历3个阶段，如下图所示：



2.1、mysql单库

2021年5月，系统上线后，系统处于推广初期，用户量比较少，QPS较低(高峰期不到QPS不到100)，研发为了功能快速上线和节省存储成本，采用mysql单库设计，另外为了保证mysql高可用，防止单点故障，设计同城容灾库备用，其中mysql单库基于云部署(方便快速扩容)，硬件配置(CPU/内存/磁盘)：16C/32G/512G。

2.2、Redis缓存+mysql读写分离

2022年12月，系统支持折扣的维度越来越多，产品与运营开始向全网各个地区推广，系统数据量开始稳步提升，使用地区从5个增长到10个，扩张一倍，核心表折扣曝光表数据单日增长10w，增长势头很猛，为了应对系统压力，我们进行第1轮架构优化，具体优化点如下：

1） 核心查询高并发接口增加redis数据缓存。当数据有变更操作，数据更新到redis，减少数据回表操作。

2）数据读写分离。数据库横向扩展，增加2个从库：硬件配置(CPU/内存/磁盘)：16C/32G/512G，主库负责写，从库负责读，通过不同的账号配置进行权限隔离，集成动态数据源实现读写切换。

2.3、Redis缓存+mysql分片库

2023年初，运营推广力度进一步扩大，覆盖地区扩大到全国45个大区，核心表折扣曝光表数据单日暴涨到100w+，1个月涨到3000w+，数据量达到这个级别后，我们很难保证mysql的查询和写入性能，平时某些正常的SQL语句在并发稍微高一点的时候也会不时出现慢查询，慢查询会引起接口查询超时、操作无响应、页面白屏等异常现象。如果碰到双十一等重大节假日，投放流量增加，系统会迎来更大的访问压力，所以我们在第1轮优化的基础上再进行第2轮架构优化，具体优化点如下：

1）排查和优化慢SQL。通过慢SQL监控工具，梳理相关慢SQL语句，分迭代排期优化。思路：优化索引，尽可能避免表的关联查询，调整为单表查询，数据的加工处理放在逻辑层，数据库只做存储和简单查询。

2）分库分表。从单库切换为128个分片库，可支撑12亿数据。按照目前日增100w规划，1年增长数据量=100w*30*12=3.6亿，可以支持未来4年发展。另外我们的折扣系统业务是面向C端用户，分片库按用户userId hash取模分片，数据比较均匀，避免数据倾斜造成单库压力过大。

3）冷热数据分离。大部分面向C端用户的系统，都没有查询3个月之前数据的习惯，基于这个特点，我们对核心表折扣曝光表数据进行冷热分离，3个月之前的冷数据同步到BDP离线库，供运营侧报表查询，mysql分片库只保留3个月热数据，数据量不至于太大，数据库读写性能稳定运行。

二、痛点

我们针对第3阶段的数据库分库内容展开讲，由于系统不是一开始就进行的分片，因此需要将数据从单库迁移到分片库，按照以往的开发经验，如果是只迁移一张日志表会简单很多，因为它只记录数据的新增，但是对于频繁更新的表，进行分库迁移还要兼顾历史数据的处理，实现不停服的数据迁移，会带来很多技术挑战，具体如下：

- 如何保证不影响业务，平滑过渡(包括异常回滚)
- 分布式数据库如何选择
- 历史数据如何迁移
- 实时数据如何同步
- 数据最终一致性如何保证

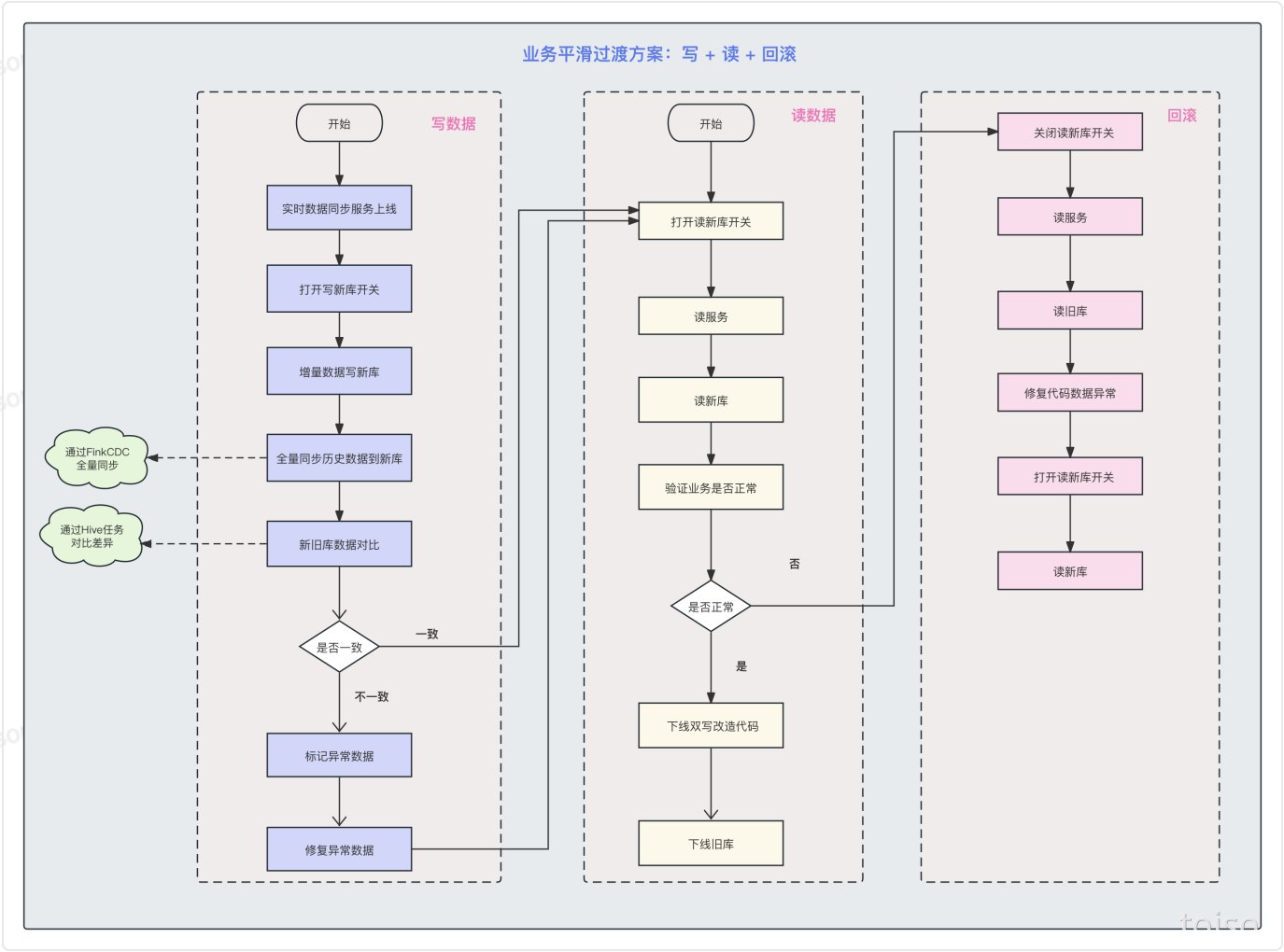
我们这里迁移的场景是动态折扣系统核心表折扣曝光表数据，表更新频繁，而且是属于线上创收业务，系统SLA至少保证99.995%，不能停服务。

三、解决方案

1、如何保证不影响业务，平滑过渡(包括异常回滚)

1）线上正常业务采用库双写方案，完成切换后，下线旧库，平滑过渡。

2）旧库下线后，离线报表调整到读新库，保障业务不受影响。



2、分片库中间件技术选型

市面上分布式数据库中间件众多，集团不同中心之间引入中间件也有差别，主要有：shardingsphere、mycat，TiDB，它们都支持mysql数据库。考虑到公司部分中心在使用TiDB有一些大坑，风险不可控，暂时不考虑引入，shardingsphere与mycat综合比较后，我们选择shardingsphere，对会员userId进行哈希取模分片，128个分片库。

中间件	成本	性能	稳定性	活跃度
shardingsphere	低	高	高	高
mycat	高	中	高	基本不维护

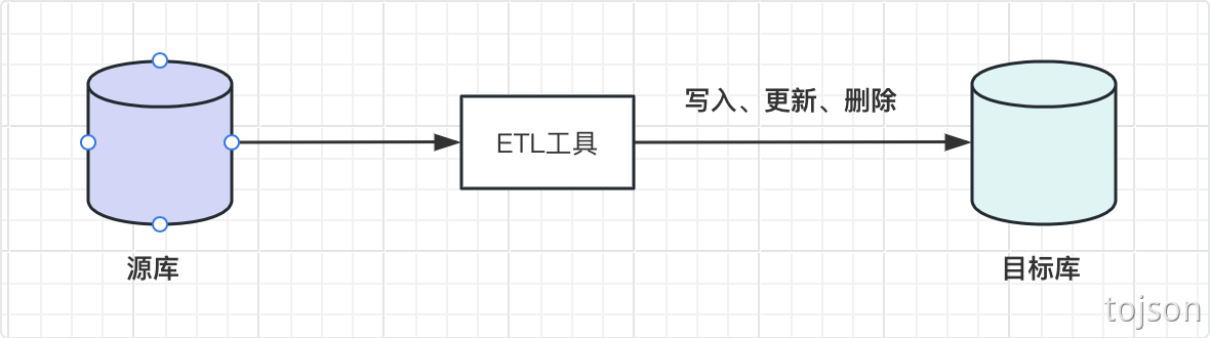
TiDB	高	高	低	高

3、历史数据如何迁移

数据迁移有很多的方案，市面上迁移的工具也非常多，针对不同的场景我们选择迁移的方案也不一样。

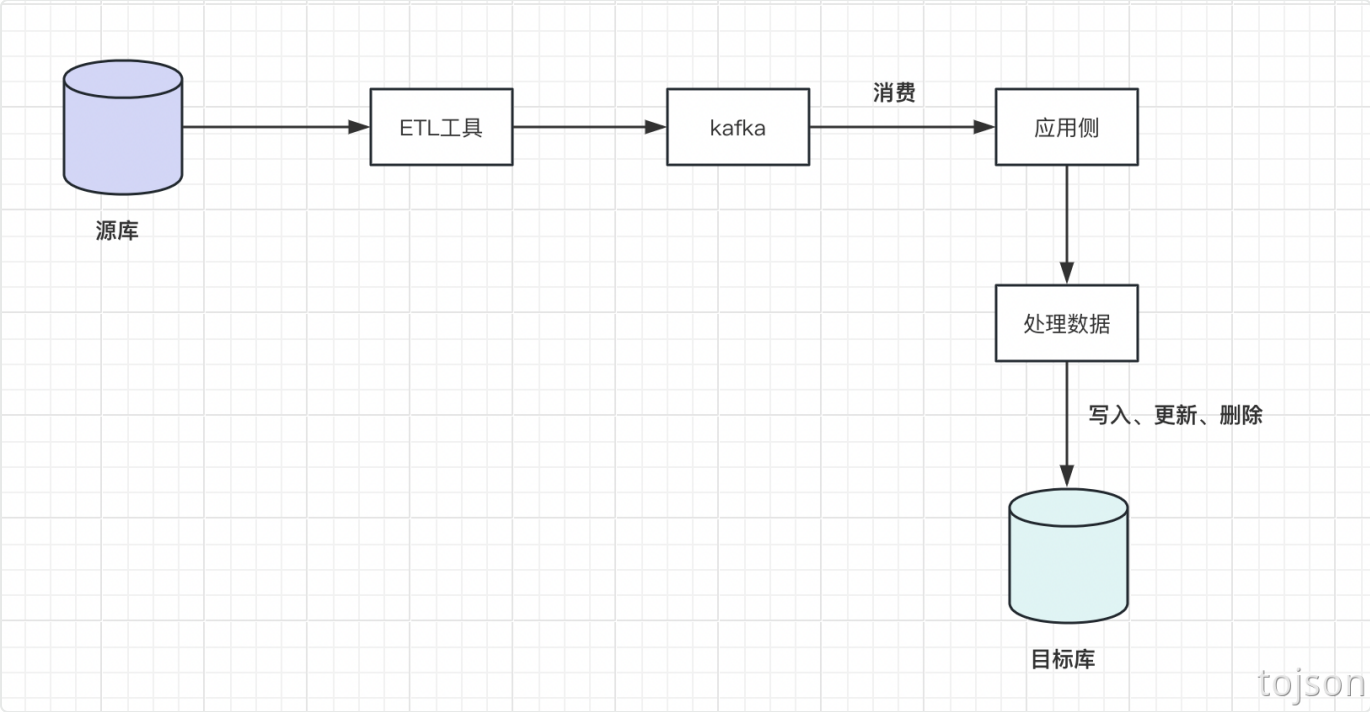
3.1、单库--》单库迁移

适合源库与目标库的表字段可以一对一映射，无需额外的开发工作量，这个时候代码层不用改造。ETL工具有很多种：Flink CDC、Canal、DataX，目前比较主流的是Flink CDC，可以全量同步也可以增量同步。



3.2、单库--》分片库迁移&分片库--》分片库迁移

适合源库与目标库的表字段无法一对一映射，这个时候代码层需要改造，应用侧消费数据后，对数据进行加工处理，写入目标库。这次讲的动态折扣系统核心表折扣曝光表迁移属于这个场景，因为历史数据量大，耗时旧，为了减少对目标库的冲击，建议在晚上12点以后低峰期操作，另外kafka处理数据的线程不要开的太大。

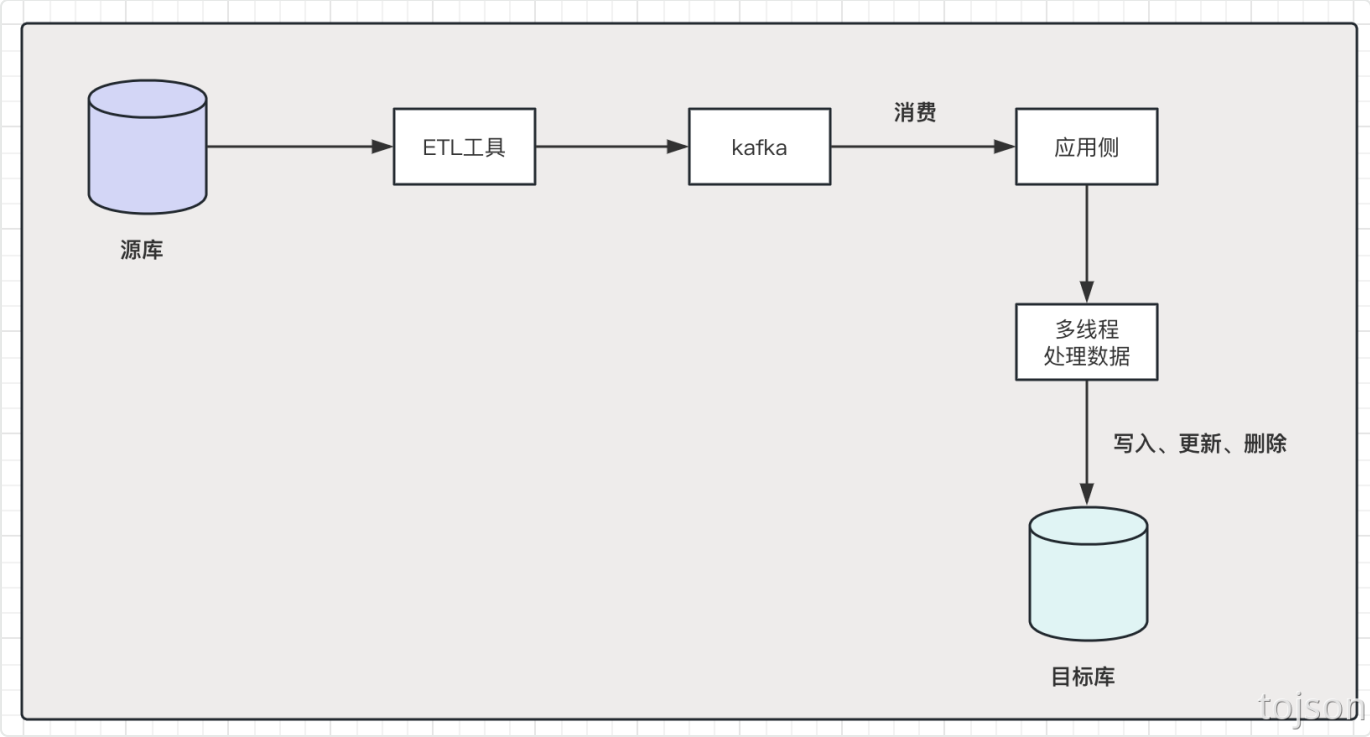


4、实时数据如何同步

目前有3种方案可选择：1) 监听mysql binlog日志 2）通过拦截mybatis的sql拦截器，获取变更语句，发送kafka变更消息 3）新旧库双写。**方案1无侵入性，比较可靠，优先考虑，如果没有开放binlog，考虑方案3。**

4.1、方案1–监听mysql Binlog日志（推荐）

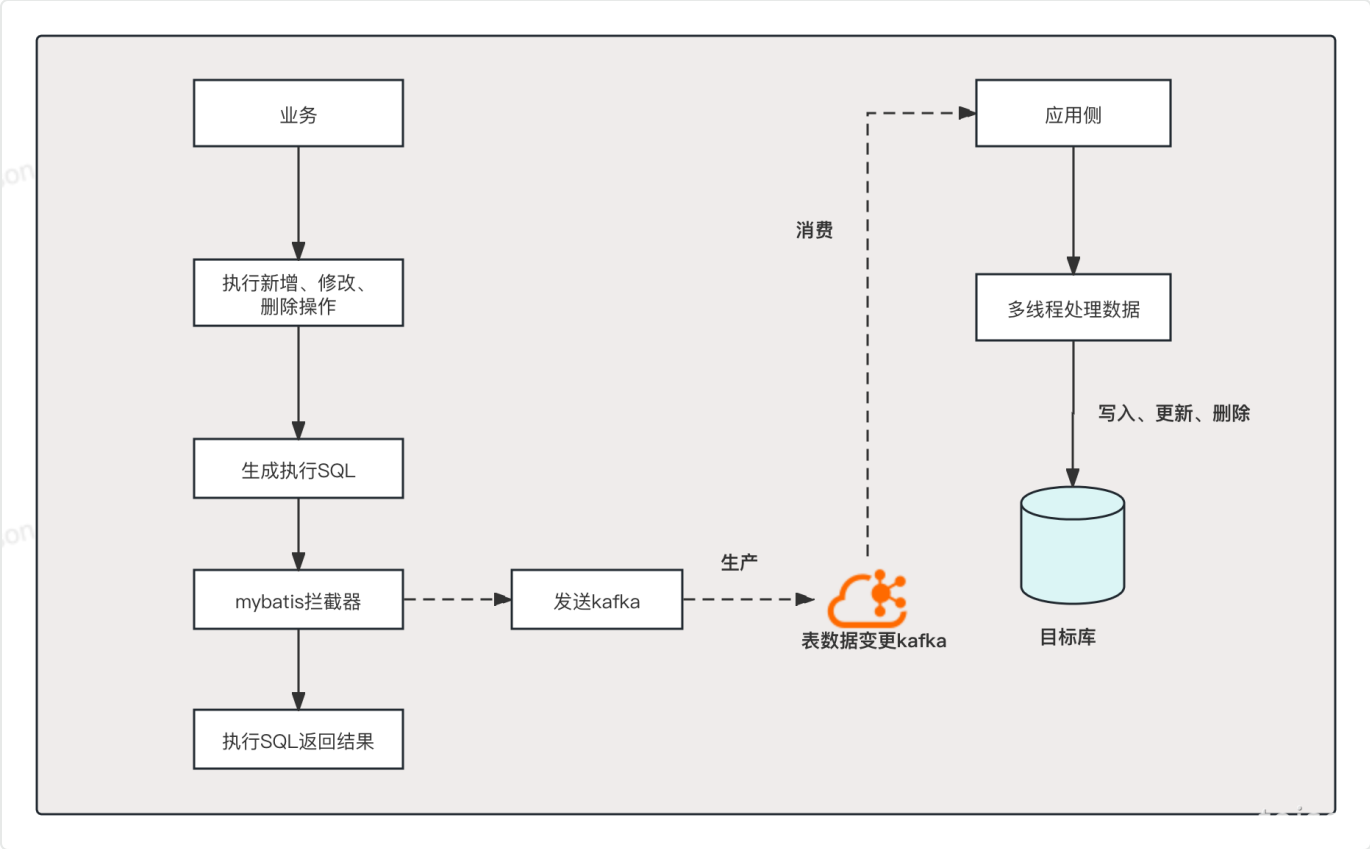
通过Flink CDC或Canal监听binlog数据，写入kafka，应用侧消费完成后，更新新库。



4.2、方案2-自定义mybatis sql拦截器，获取变更语句，发送kafka变更消息

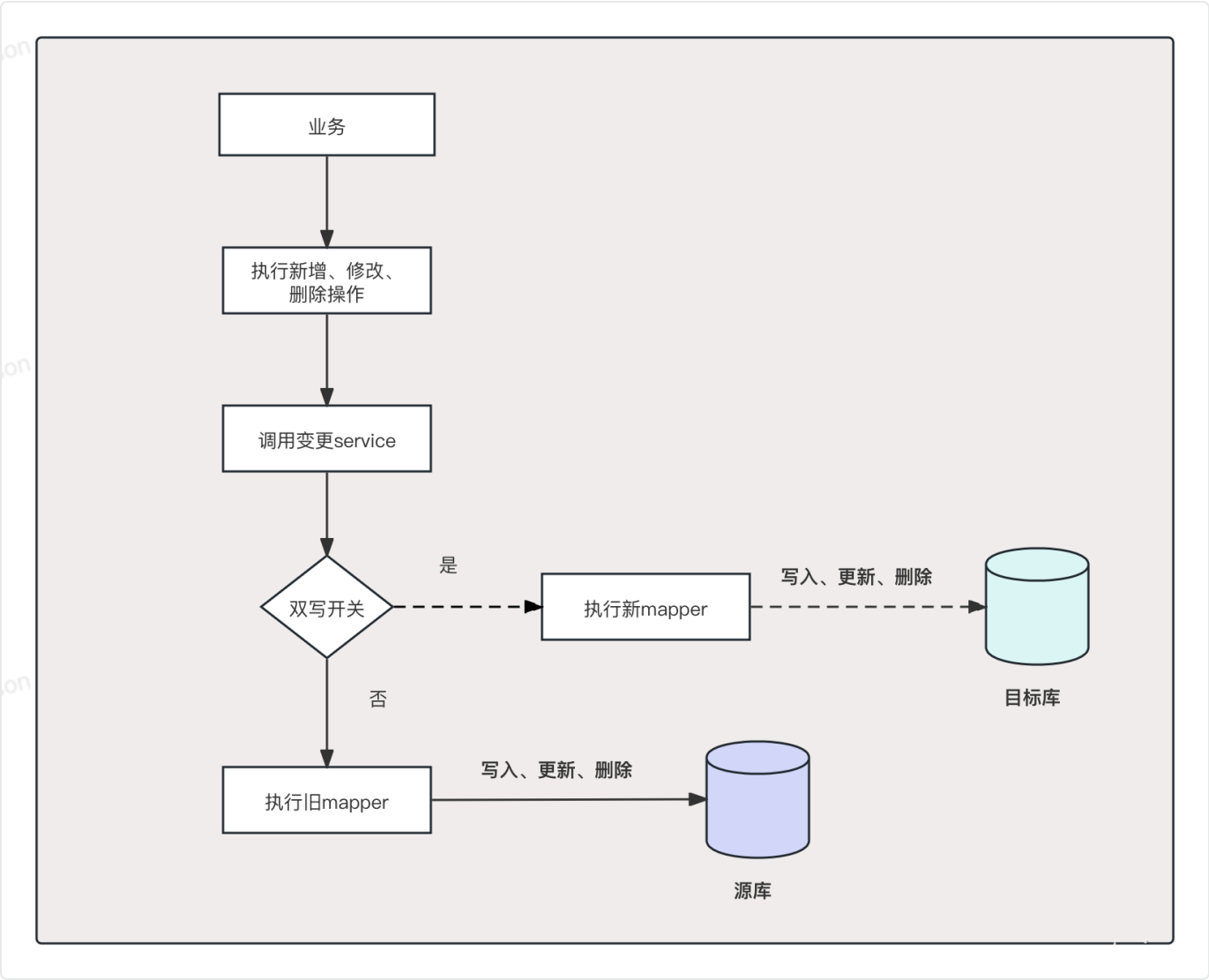
优点：对于数据库的增、删、改，能实时监听统一处理。

弊端：改写程序代码，自定义一个mybatis拦截器，拦截所有的插入、更新、删除语句，对原有的系统有侵入性，还会产生一定的性能损耗。此外，如果执行过程中，方法事务回滚，kafka无法回滚。



4.3、方案3–新旧库双写

通过修改业务代码，通过双写开关控制双写时机，打开开关后，变更完旧库再变更新库。



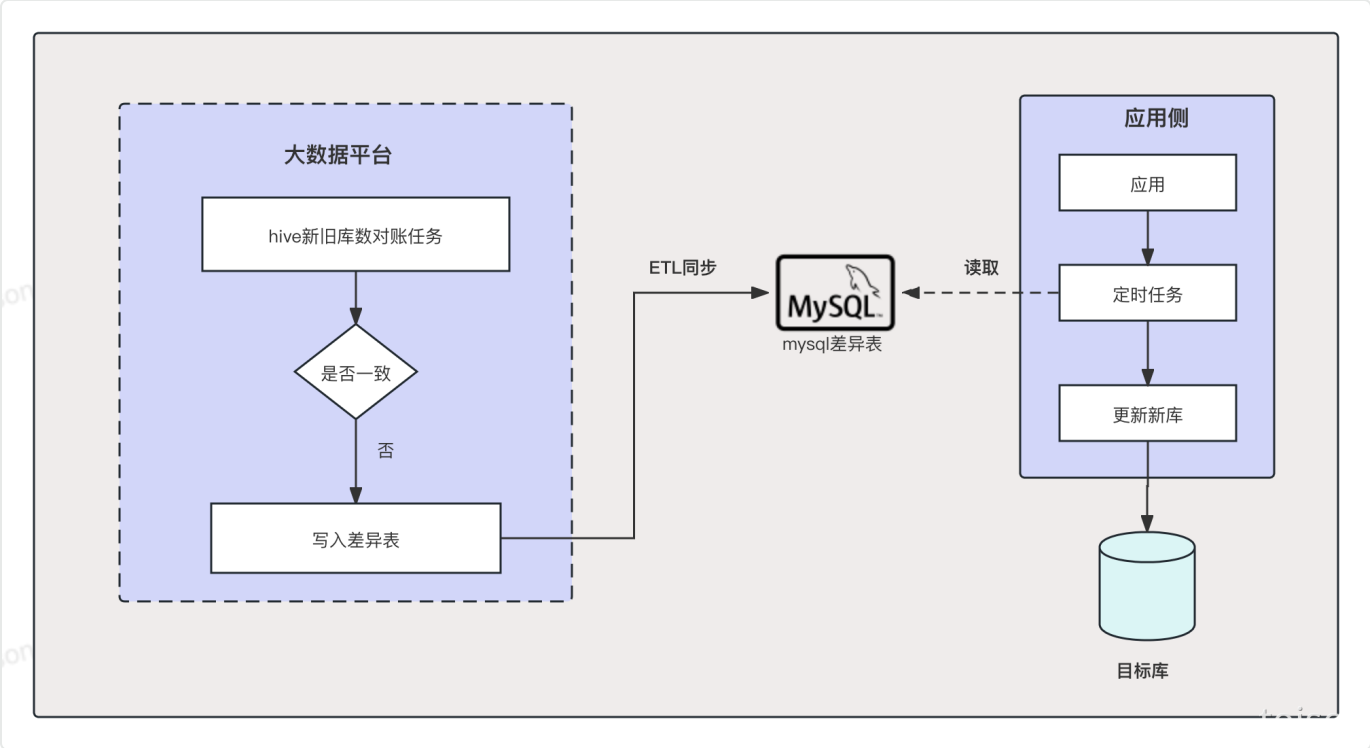
5、数据一致性如何保证

通过对账程序自动对账，同时配置人工告警，防止对账失败，人工介入。具体步骤如下：

1）考虑到数据量很大，我们通过大数据平台hive，创建对账任务，每天下半夜的凌晨1点对账新库、旧库表数据，不一致，写入异常数据到差异表。同时配置人工告警，通知相关的研发人员。

2）通过ETL同步差异表数据到mysql库差异表（一般差异表数据量比较小）。

3）应用侧通过定时任务定时读取mysql库差异表，更新新库。



四、落地实现

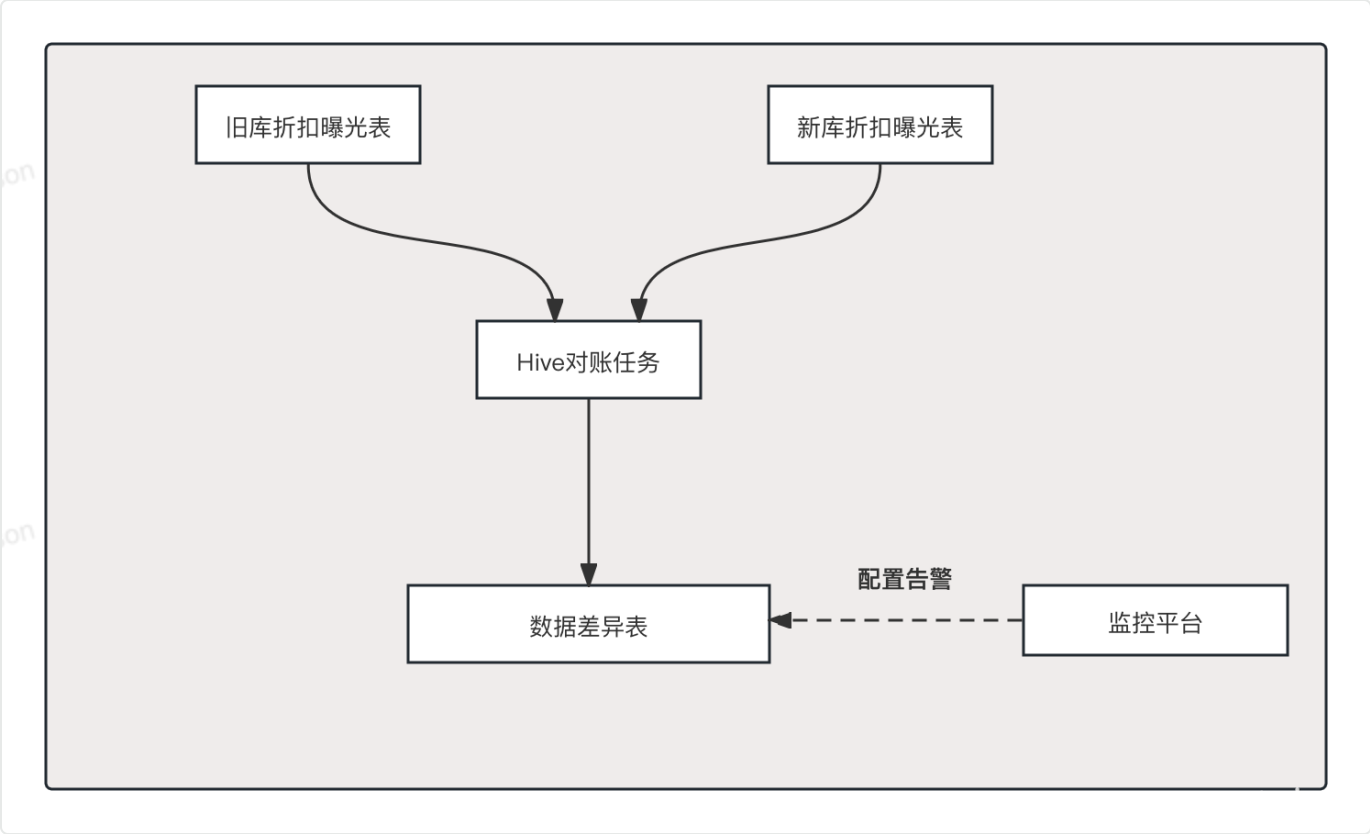
1、通过ETL工具同步旧库数据到kafka

通过ETL工具Flink CDC同步动态折扣**旧库**折扣曝光表全量或增量数据到kafka。

2、实时数据同步服务消费kafka数据后入新库

完成数据同步服务搭建，多线程消费步骤1中的kafka数据，按要求处理完数据后，写入新库。

3、创建Hive对账任务，比对新旧库，异常数据写入大数据差异表，同步在监控平台配置监控告警。



4、同步大数据差异表数据到mysql数据差异表

5、应用侧创建定时任务，定时读取mysql库差异表数据，写入新库

写入新库的时候，根据主键id将新库数据更新同旧库一致。

6、持续自动对比数据，发现不一致，人工介入，及时定位原因修复上线，直到数据最终一致。

7、打开开新库开关，启用新库，下线旧库和相关旧代码。

五、经验总结

经过2个迭代的努力，完成动态折扣系统折扣曝光表从单库到分片库的平滑迁移，线上业务0感知，0客诉，从结果上看，还是比较成功的，整个迁移过程中有一些优秀可复用的经验供大家参考：

1、旧库数据写入到新库要保证两边id一致，否则涉及以id作为条件的sql语句会有问题。如果旧库是采用数据库自增主键，那就需要返回id插入到新库。或者直接采用第三方生成主键也行。

2、数据对比因为是拉取全量数据，数据源最好使用容灾库，避免对生产库造成影响。

3、数据对比周期尽量长一点，以便暴露隐藏问题。

部分内容参考：http://wed.xjx100.cn/news/295408.html?action=onClick