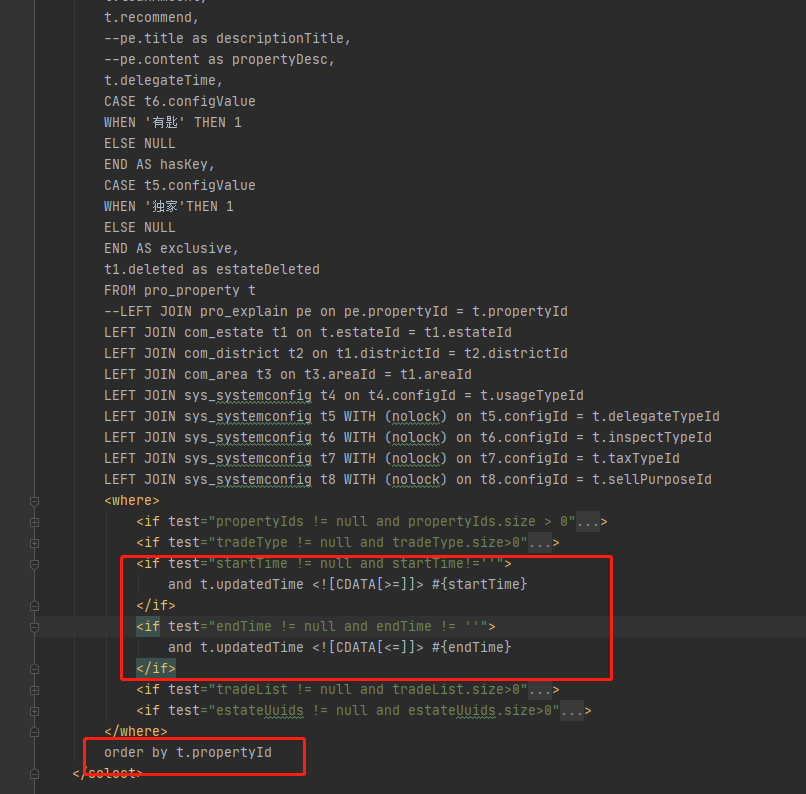
# # 分页漏数据BUG

## 2019-Q3~2020.Q3

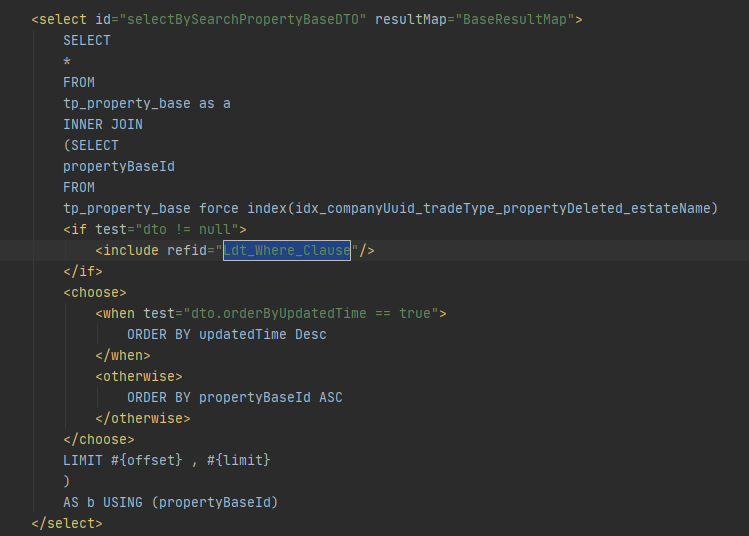
### V10分页推送到营销库漏数据+深度分页慢

问题分析：V10同步到营销库，采用按照更新时间范围搜索的方式，会导致分页不稳定，期望落在第二页的数据可能会因为第一页数据被更新，被挤到第一页的现象。



### 幸福里分页拉取营销库漏数据+深度分页慢

问题分析：营销库提供openapi给第三方，如下述代码，采用Limit offset,limit会存在深度遍历问题和漏行问题（如果使用更新时间起始值）。



### 时间段分页查询的问题演示

假设数据库中有10条数据，T10时刻更新时间在T1…T10之间，现在搜索条件为[T1,T10],每页为2条；在翻到第二页的时候

**1.T10时刻，拉取第一页**

SQL：Select ID from TABLE where T between T1,T10 limit 0,2;

结果：ID1、ID2

**2.T12时刻，拉取第二页，但是ID2被更新，时间为T12**

SQL：Select ID from TABLE where T between T1,T10 limit 2,2;

结果：ID4、ID5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **自增ID升序** | **更新时间**  **（T10时刻）** | **预期所属页数**  **（T10时刻）** | **更新时间**  **（T12时刻）** | **内存所属页数**  **（T12时刻）** |
| ID1 | T1 | 1 | T1 | 1（T10时刻到内存） |
| ID2 | T2 | 1 | T12 | 1（T10时刻到内存） |
| ID3 | T3 | 2 | T3 | 被遗漏，不会再次被捕捉 |
| ID4 | T4 | 2 | T4 | 2（T12时刻到内存） |
| ID5 | T5 | 3 | T5 | 2（T12时刻到内存） |
| ID6 | T6 | 3 | T6 |  |
| ID7 | T7 | 4 | T7 |  |
| ID8 | T8 | 4 | T8 |  |
| ID9 | T9 | 5 | T9 |  |
| ID10 | T10 | 5 |  |  |

针对上述问题，一般有两种解决办法：

快照：将某个时间段的内容固化，能够避免分页问题，这种很显然

滚动拉取：从for循环变成while循环，只带一个limit参数。这样拉取SQL变为

**1.T10时刻，拉取第一页**

SQL：Select ID from TABLE where T between T1,T10 limit 2;

结果：ID1、ID2

**2.T12时刻，拉取第二页，但是ID2被更新，时间为T12**

SQL：Select ID from TABLE where T between T1,T10 limit 2 and ID>ID2;

结果：ID3,ID4

**3.T15时刻，拉取第三页，但是ID5被更新，时间为T15**

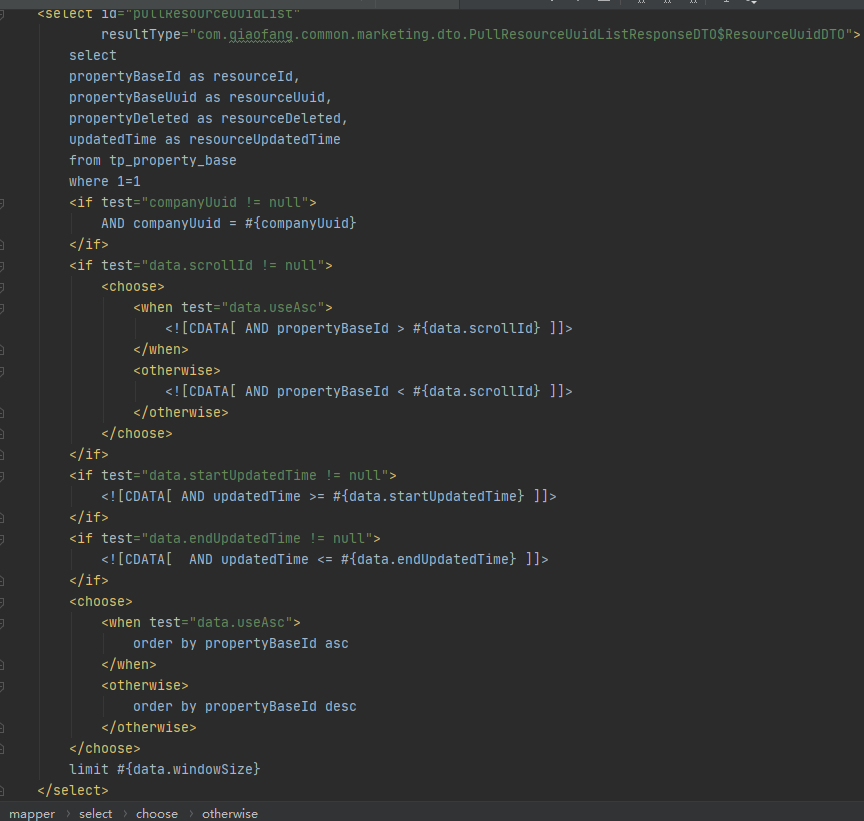
SQL：Select ID from TABLE where T between T1,T10 limit 2 and ID>ID4;

结果：ID6,ID7

虽然ID5不能立马拉取到，但是在未来的时间区间【T11,T20】被拉取到

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **自增ID升序** | **更新时间**  **（T10时刻）** | **预期所属页数**  **（T10时刻）** | **更新时间**  **（T12时刻）** | **所属页数**  **（T12时刻）** | **更新时间**  **（T15时刻）** | **内存所属页数**  **（T15时刻）** |
| ID1 | T1 | 1 | T1 | 1 T10时刻到内存） | T1 | 1（T10时刻到内存） |
| ID2 | T2 | 1 | T12 | 1 T10时刻到内存） | T12 | 1 T10时刻到内存） |
| ID3 | T3 | 2 | T3 | 2 (T12时刻到内存） | T3 | 2 (T12时刻到内存） |
| ID4 | T4 | 2 | T4 | 2 (T12时刻到内存） | T4 | 2 (T12时刻到内存） |
| ID5 | T5 | 3 | T5 |  | T15 | 被跳过，但后面仍然能捕捉到 |
| ID6 | T6 | 3 | T6 |  | T6 | 3 (T15时刻到内存） |
| ID7 | T7 | 4 | T7 |  | T7 | 3(T15时刻到内存） |
| ID8 | T8 | 4 | T8 |  |  |  |
| ID9 | T9 | 5 | T9 |  |  |  |
| ID10 | T10 | 5 |  |  |  |  |

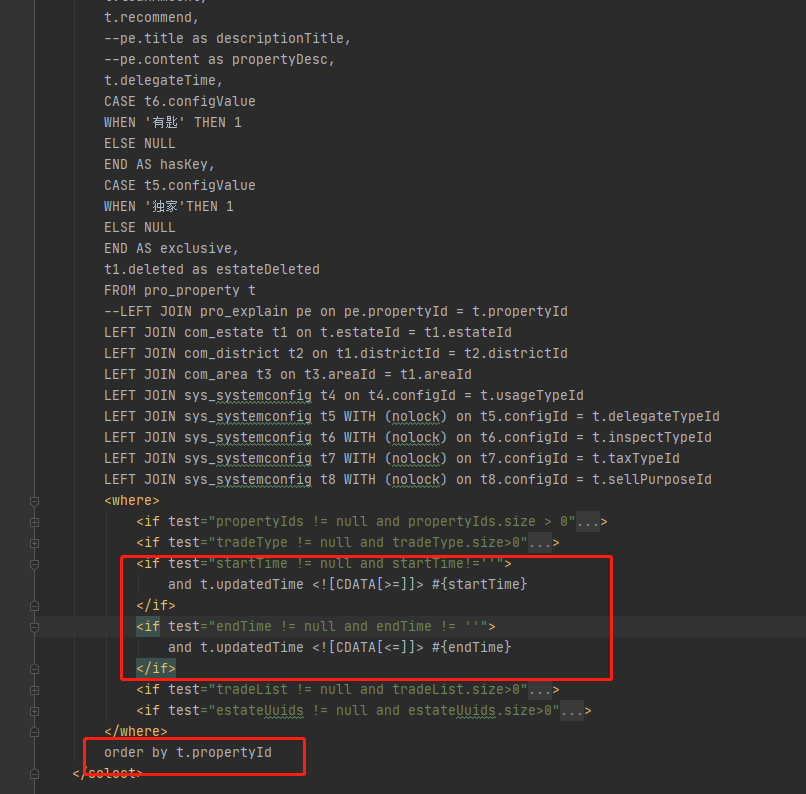
### 使用窗口滚动方式拉取数据



### 页面上第一页和第二页数据一样

现象：V10推送房源，分页遍历不能拿到所有数据。

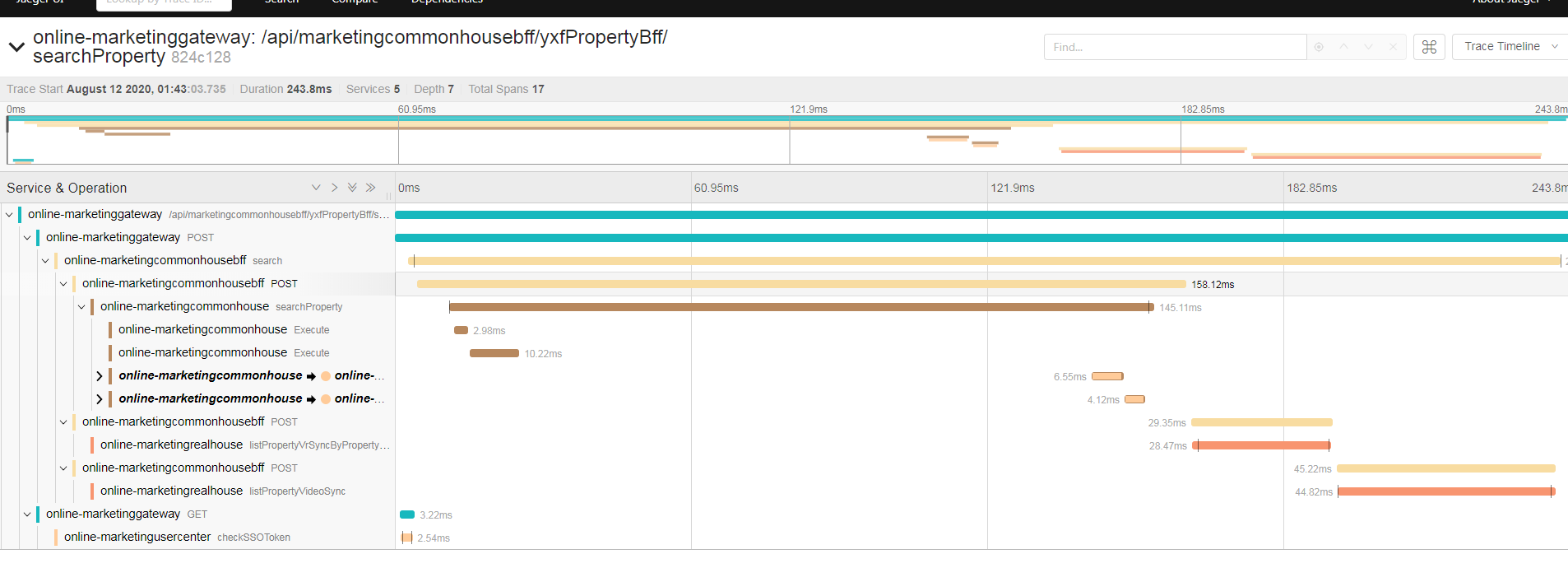
分析：没有加排序条件，下面红色标示出来的（后面已增加）

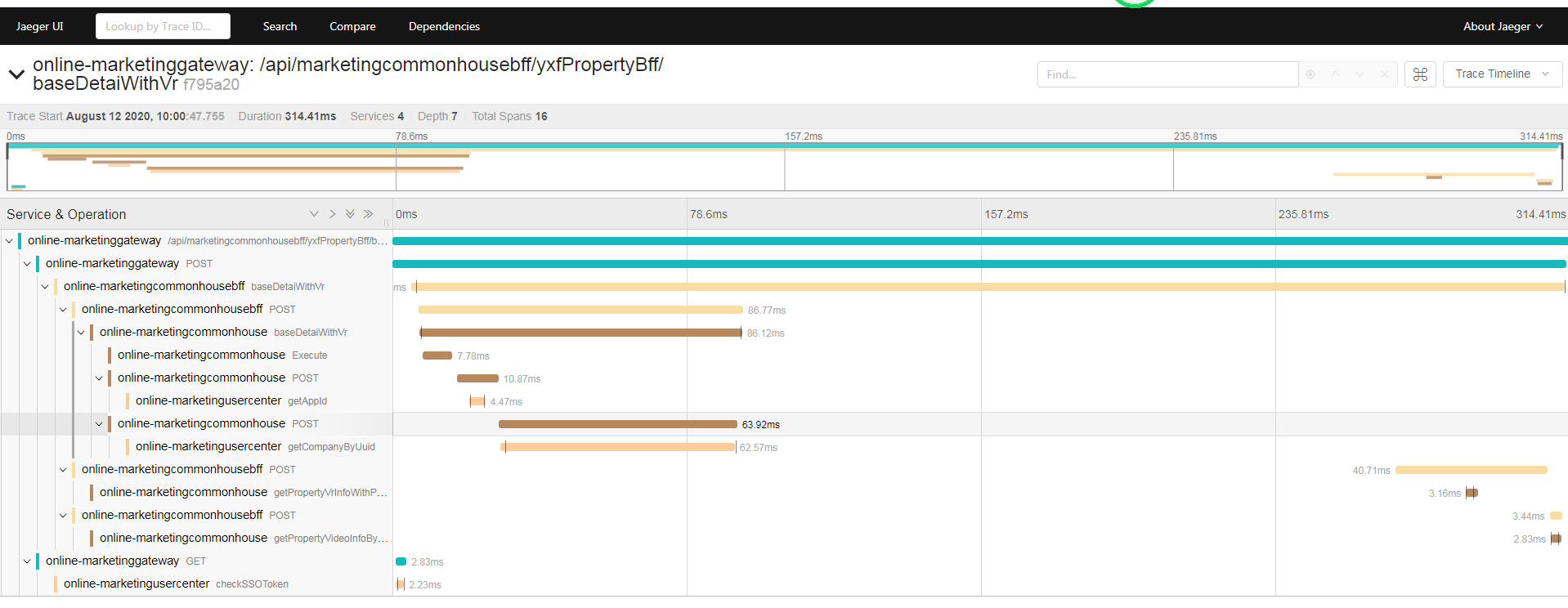


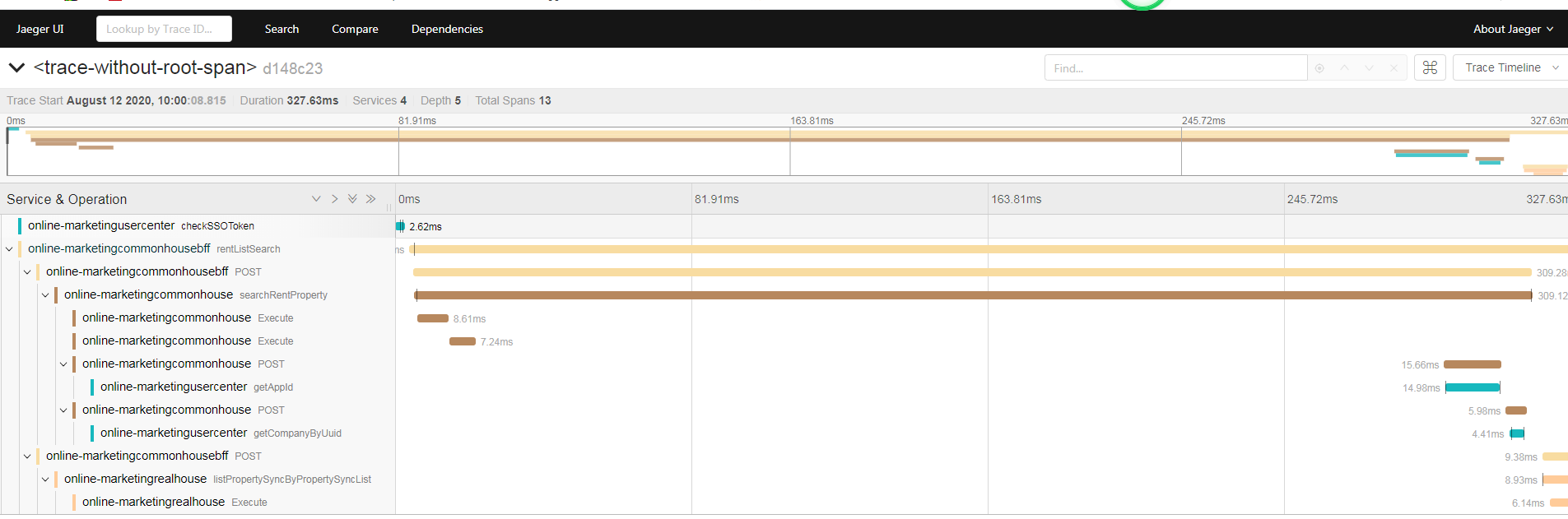
问题分析：没有使用唯一排序，数据库自动优化，返回了理论上和搜索条件一致的数据，所以需要增加唯一排序条件。

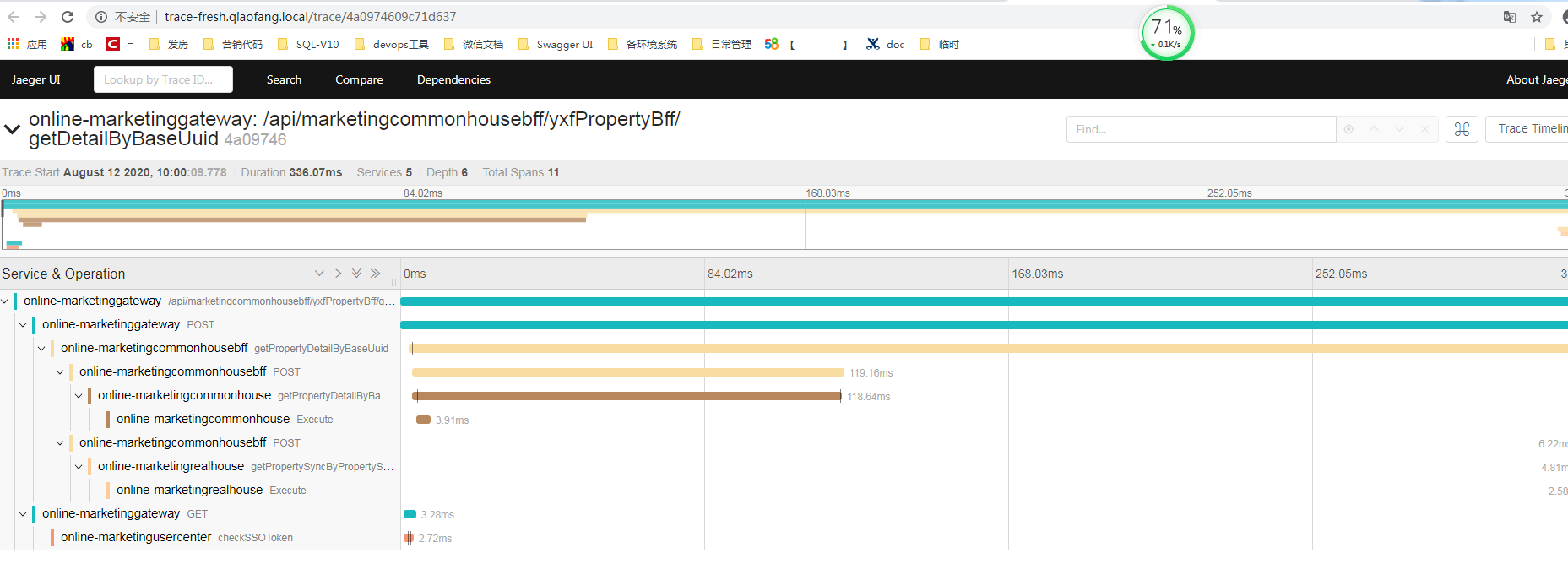
# #大搜索页面的性能bug

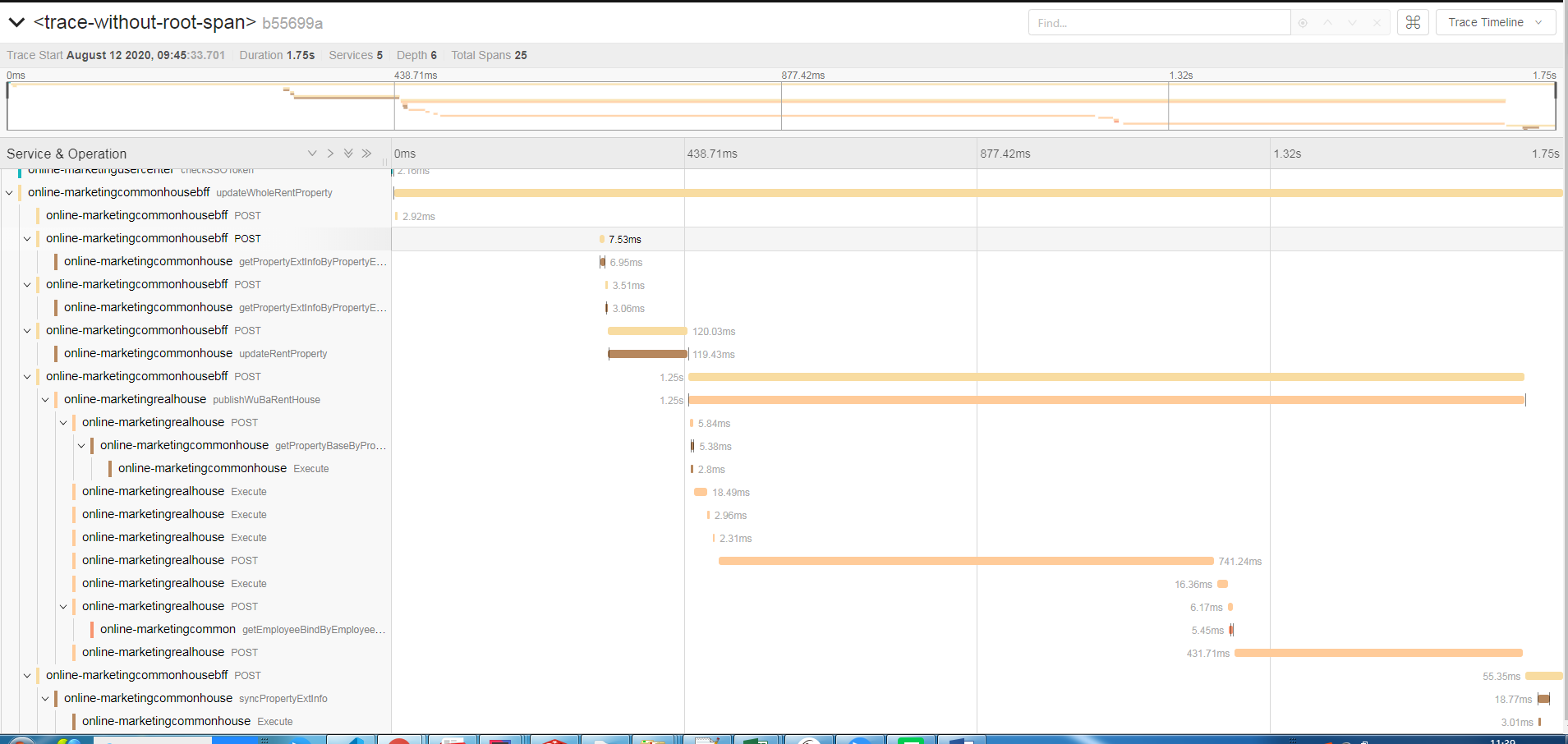
## 营销房源搜索页面优化











问题：一个页面需要四个表（基本信息表——100个字段、营销信息表——50个字段、标签表——小于10个字段、发布记录表——10个字段）的数据，搜索条件涉及到四个表，如何优化？

最开始的代码：

优化后的代码：



### 避免不必要的left join

Count 语句，有时候只涉及一个表的搜索条件，就可以利用mybatis来动态生成，避免left join。

### 避免大表join查询

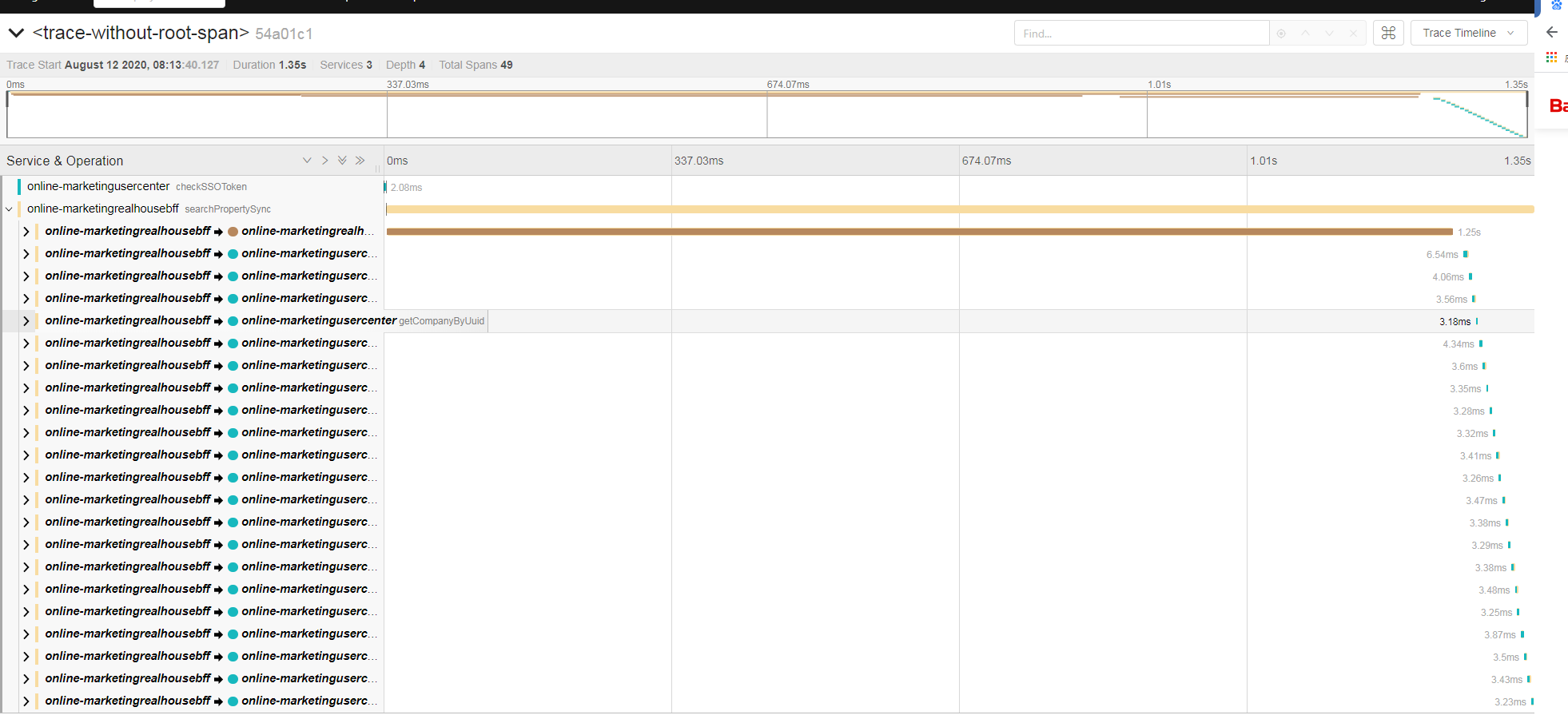
四个表join，可以对于不常用的搜索条件，进行in 或者 not in 语句替换。我们优化的时候，有时候需要用join来替代in 语句；有时候需要用in语句来替代join语句，主要看搜索字段的频率和数据限制。

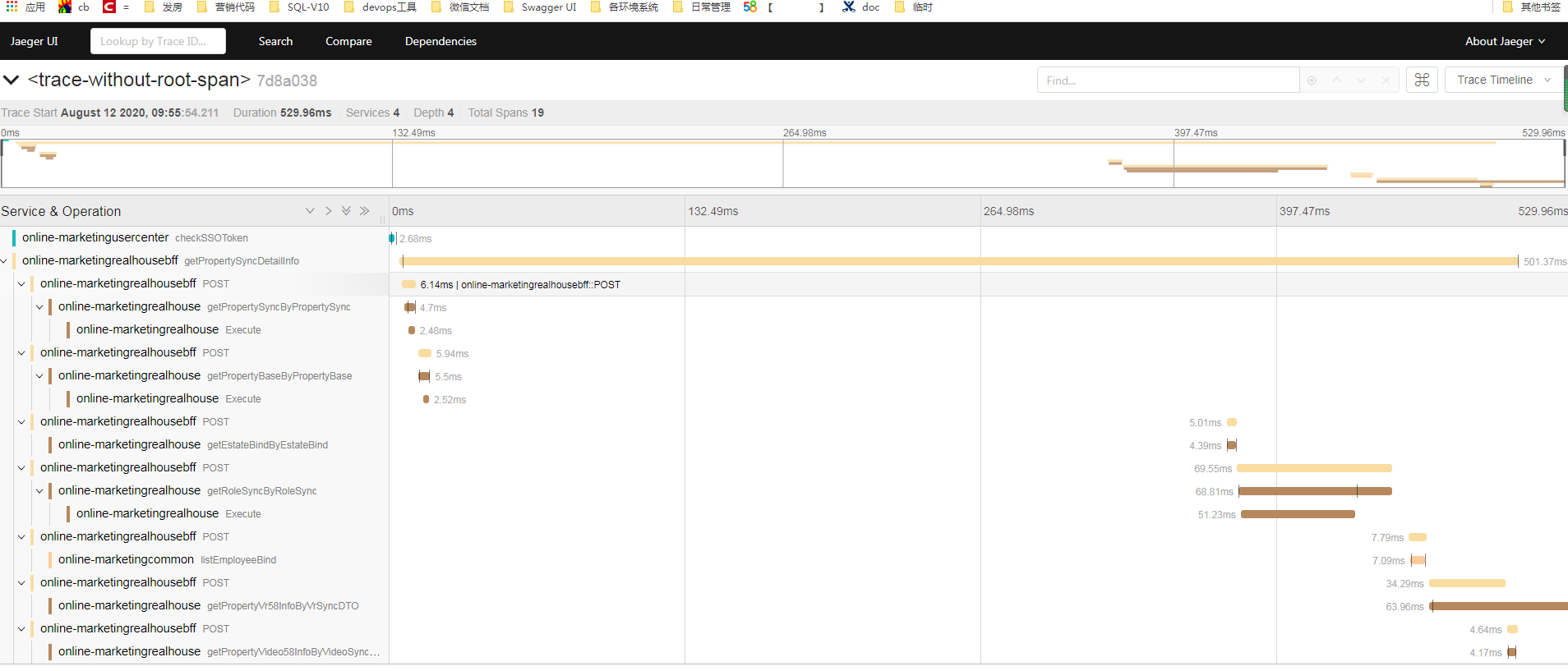
### 避免select \*

页面展示多少字段，就搜索多少字段，不要select \*，可能表的字段越来越多，性能就会越来越低。

## 真房搜索页面优化

### 重复的公司uuid遍历





## 清理冗余数据

### 清理realhouse数据库中非发房和真房的数据

Commonhouse和realhouse的表存在较多数据冗余。可以清理出100G的数据和索引。

### 清理MICROSHOP数据库中非发房和真房的数据

Microshop冗余数据清除。

# #API必填项校验常见BUG

### 营销库室厅卫数据库必填

实际上外部系统 室厅卫不一定必填，导致数据同步问题多

### 58发房多个字段必填校验

Bug1——58发房 朝向不能为空

Bug2——58发房 装修情况不能为空

Bug3——58发房 房屋类型不能为空

Bug4——58发房 单元号不能为空

同步校验解决办法：的字段：如楼栋单元房号等规则，58有同步校验，交由58去校验即可；后期58变更了校验规则，我方可不用维护；





异步校验解决办法：如图片大小校验，需要在UI上面提前反馈给客户。

缺点：我们增加了校验，维护成本较大；

优点：客户体验校验，避免数据推送后再有异步校验，表面产生不必要的性能问题。

### 来电通白名单数据，手机号非必填

来电通白名单，遇到state=1的，可能手机号，姓名为空，需要放开校验

修改前：

修改后：



### @Valid在多层的bean结构中下层未设置



# #缓存一致性BUG

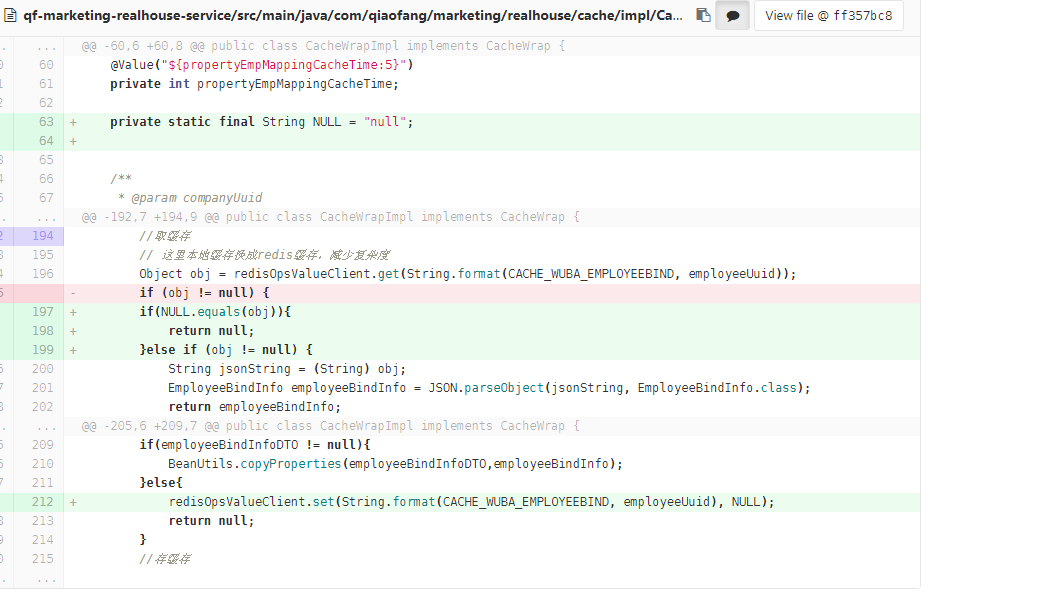
### 双应用都写一个缓存

Common写缓存为1D

Realhouse写缓存为空 2小时，非空无期限

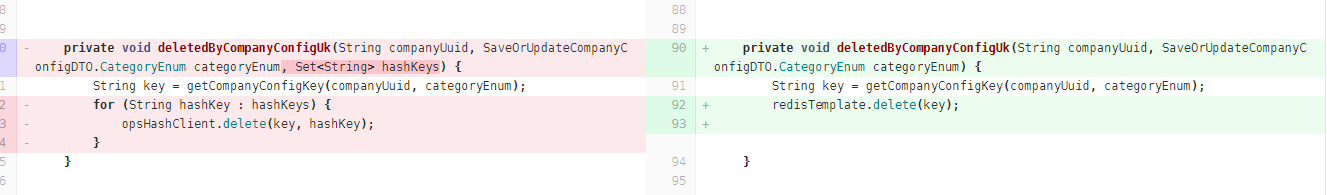
### 数据库无数据无法缓存，导致一直缓存击穿

需要设置空缓存，避免数据库击穿



### 缓存map结构，但清空按照map-entry清空

多配置时候，使用map结构，会导致缓存和数据库不一致。



# #关系表BUG

### 非1对1关系不能按照一个字段来处理数据

1.真房非1对1问题 companyUuid和clientId的映射关系有三种：1对1，多对1，1对多。因此必须是companyUuid+clientId才能作为唯一索引。如果补偿逻辑，从companyUuid开始遍历，就会导致1对多的场景出现重复补偿；

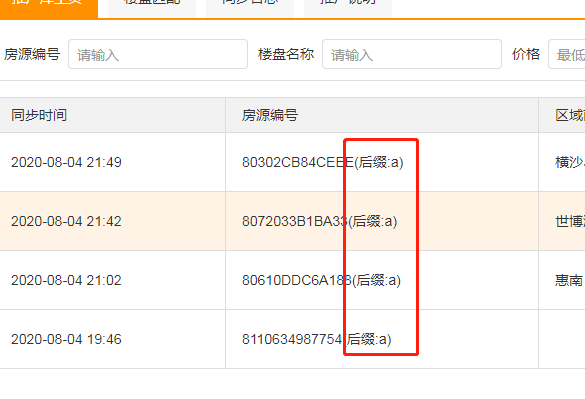
# ##功能的闭环

## 2020-Q3

### 真房多对一尾缀展示

业务诉求：真房多对一房源编号增加了尾缀，但是在真房系统不展示，仅在58展示。而且真房没有一个地方展示出这种尾缀，客户和客服无法排查问题，只能报bug。青岛科威拆分一房网就遇到大量这个问题。

解决办法：直接将多对一的公司的后缀展示出来，方便客服和用户排查问题。



### Saas和真房权限保持一致

业务诉求：sass中的一个员工看到真房同步状态，但是点进去却看不到，也没有任何提示。

解决办法：真房敏感字段已不展示，可以高于SAAS的列表展示房源权限，具体的详情页依然依赖SAAS，保证敏感信息不泄露；另外营销库目前也是这个逻辑，保持产品线的一致。



点击 真房同步



优化后—>

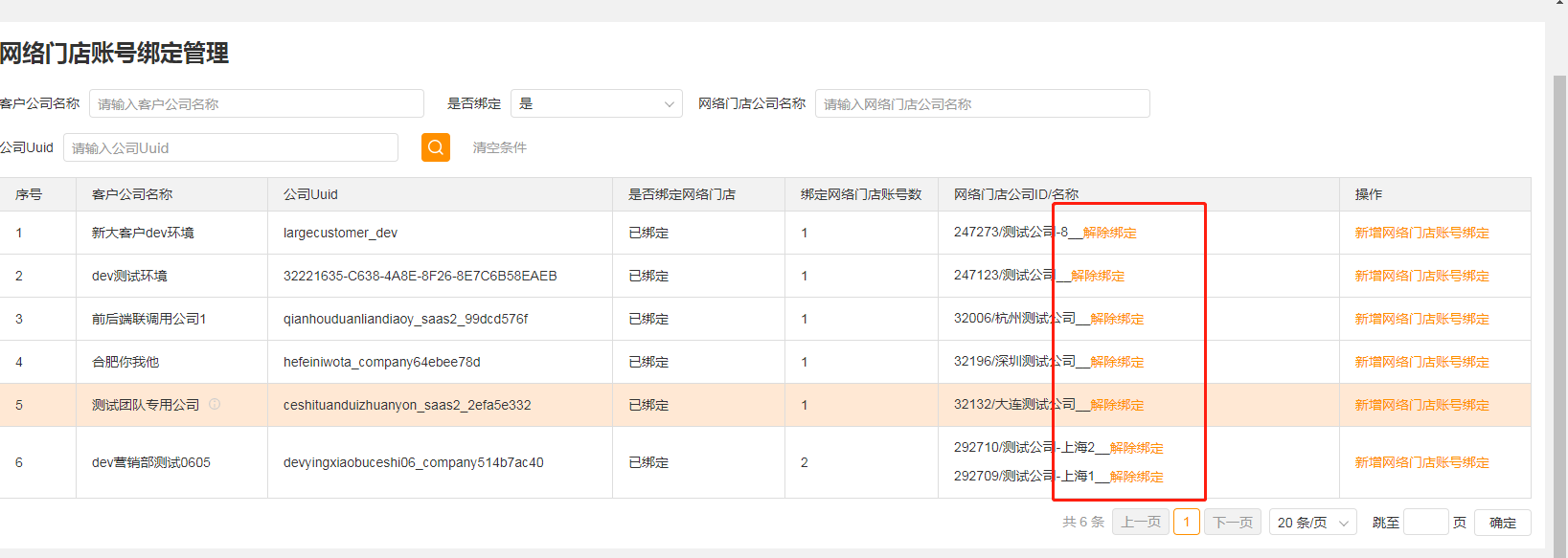
放开展示，SAAS可见，营销就可见。点会SAAS详情，依然由SAAS控制权限。

### 增加网络门店解绑功能

业务诉求：中介竞争压力大，兼并，改主体并不少见。

解决办法：网络门店绑定，有新增，没有删除，不能完全满足客户需求，增加解绑功能。

（红框为新加部分）



解绑的操作说明：

1.处理历史数据（真房、发房、来电通等） @客户

此公司有一套发布房源（\*\*）未下架，可通过设置保护房下架。

2.解绑网络门店 @bill

下架完成后，bill在营销工具后台帮解绑网络门店。



3.重新绑定网络门店 @客户



进入此页面，客户可按照此引导重新绑定授权码

### 展示网络门店部门

业务诉求：非1对1公司较多，账号管理较为麻烦，有时候客户换了门店，不知道自己需不需要重新绑定（如：发房公司错误，第二家58网络门店无房源）。

解决办法：增加部门或门店名称展示，方便客户自查问题。

个人用户视图（红框为新加部分）：



Manager用户视图（红框为新加部分）：

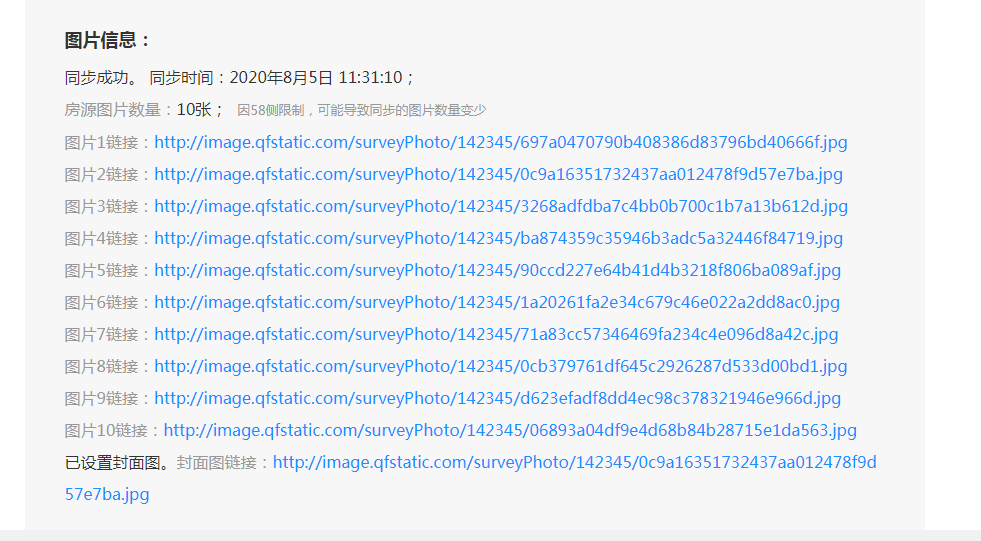


### 真房展示图片的像素信息

业务诉求：图片不过滤发送58，客户很难自查问题；客服也难查问题，FAQ也不管用。

解决办法：增加图片像素，大小，张数提示，方便用户自查问题。

修改前🡪



修改后🡪



### 房源非经纪人可视权限问题

分析：我们未与58组织结构有很好的对标，此功能咱不适合，故取消房源同步到58的网络门店部门ID字段同步。

### 发房-楼盘解绑后不需要再同步更新房源

# #减少数据同步的数据量和并发度

## 2019-Q3~2020-Q3 kafka队列

### Openapi到营销库的房源、图片、视频、VR走kafka队列

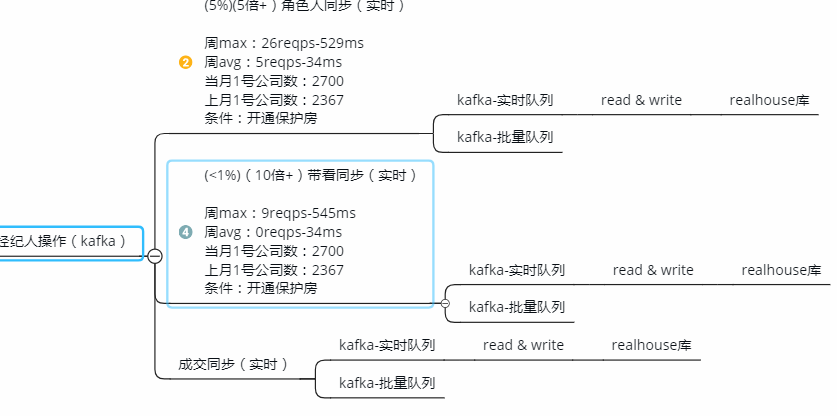
以公司uuid+房源uuid为key，保证顺序性

使用kafka来控制拉取数和并发数

### Openapi到真方库的角色人、成交、带看走kafka队列

使用公司uuid作为key，保证批量处理同一公司

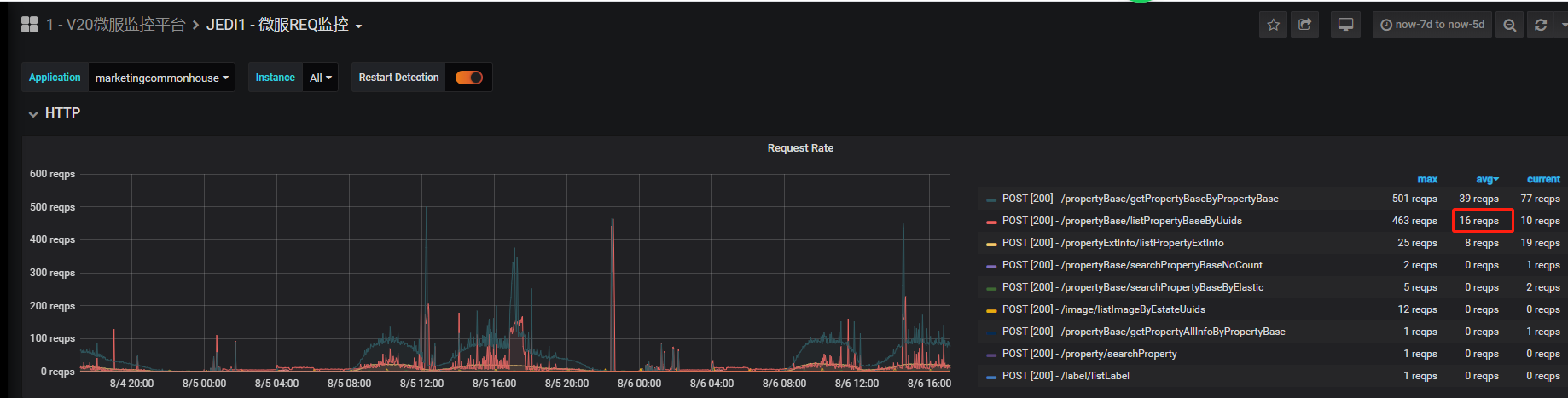
使用kafka控制并发数和流量



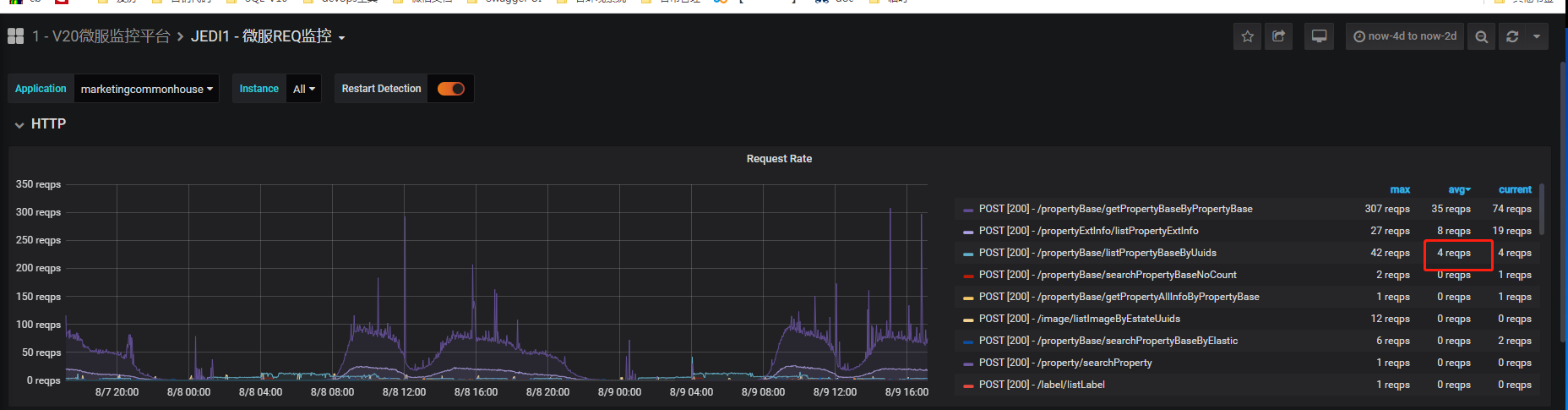
### 营销库到真方库的房源、图片、视频、VR走kafka队列（08-07 2:00）

因为kafka队列的数据库被多个group共享（atool、58真房发房），减少id反查。

上线前两天：



上线后两天：

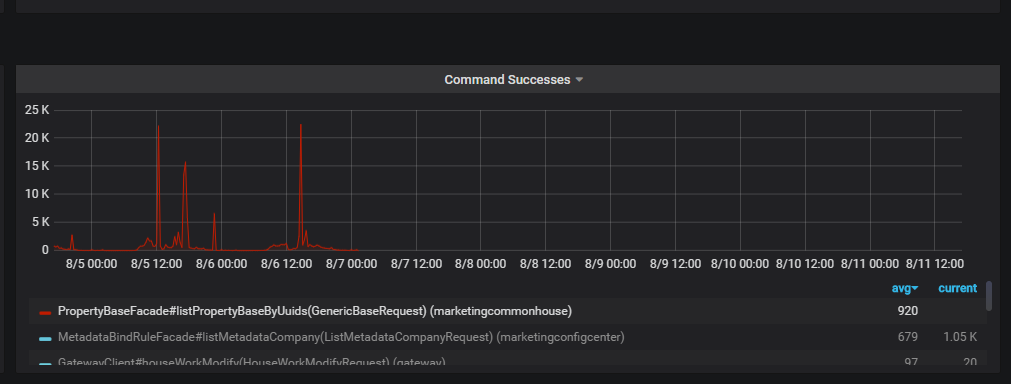


**该接口平均减少3/4的QPS；最大QPS减少了90%。**

最近7天：



Realhouse最近7天：



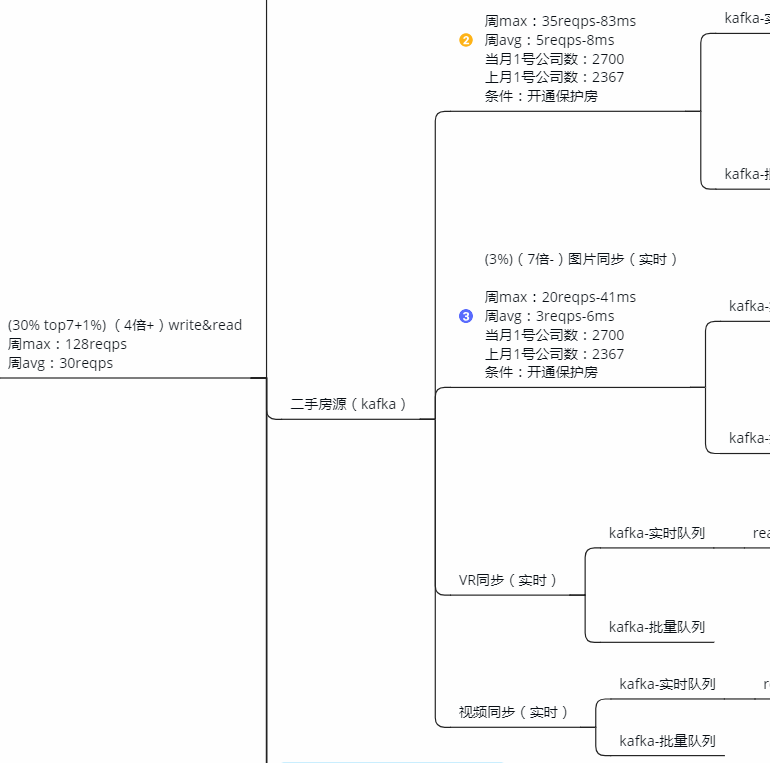
Realhouse慢SQL：



Realhouse服务健康度：



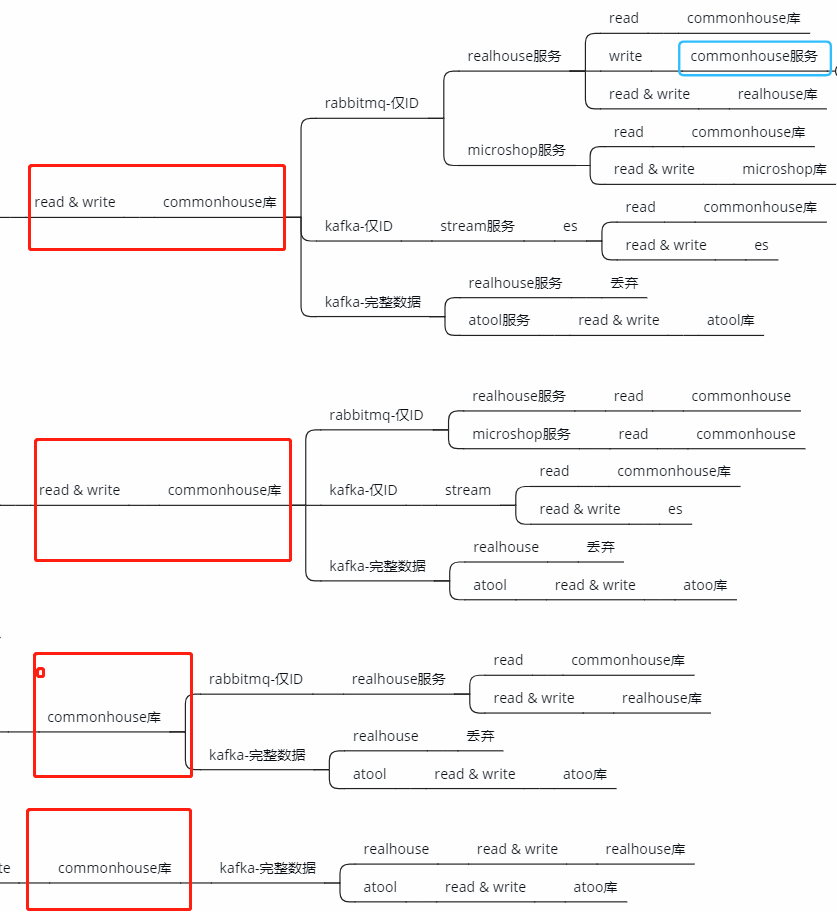




## 2020-Q1~Q3 有变更

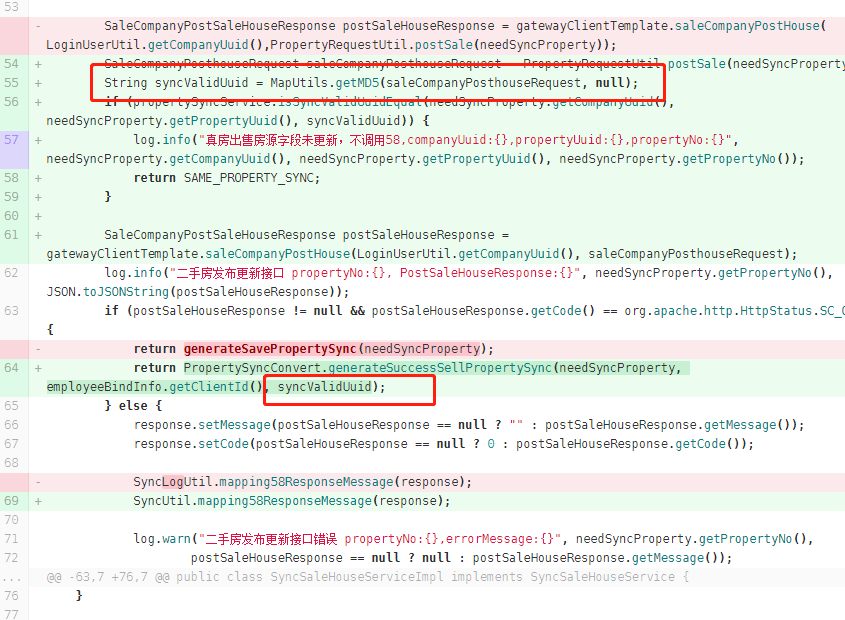
### 营销库全字段校验不重复才同步数据到真房、发房、微门店

控制红色区域，如果发现数据重复，就不往下游同步，极大减少消息的翻倍



### 真方库发送58字段全字段校验不重复才同步数据到58

同步给58的几十个字段的MD5值发生变更才重新同步58，减少同步58的流量，避免被限流



### 发房编辑营销信息发生了变更才同步数据到58

如果经纪人没有变更信息，确实没必要触发下游的逻辑

### 发房发送58字段全字段校验不重复才同步数据到58

同步给58的几十个字段的MD5值发生变更才重新同步58，减少同步58的流量，避免被限流

## 2020-Q3 不全量

### 微门店不开通（或到期）不进行全量房源自动同步维护

背景：

1.营销房源库的房源会复制到真方库和微门店库，数据会复制三份；

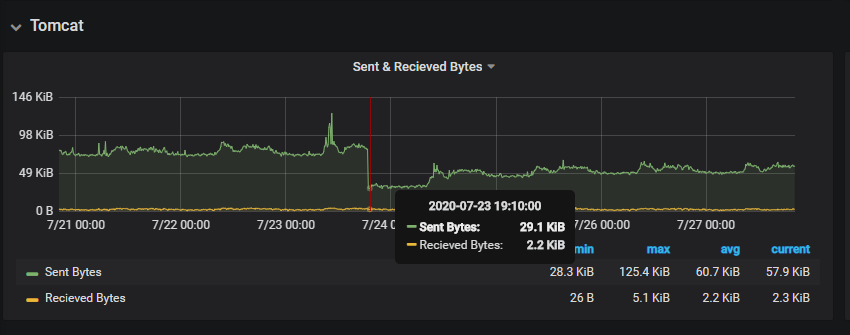
2.最近产品技术一起讨论，有优化营销库性能的需要，但考虑成本，需要优先选取性价比高的部分优先优化；

3.微门店不再新开公司（微门店已停止新开和续费），目前仅剩100家左右开通公司（微门店开通且未过期），而开通营销库的有接近3000家公司。分析认为可不再对非有效微门店进行房源基本信息和房源图片的维护，以减少对营销房源库（commonhouse）的反查压力。

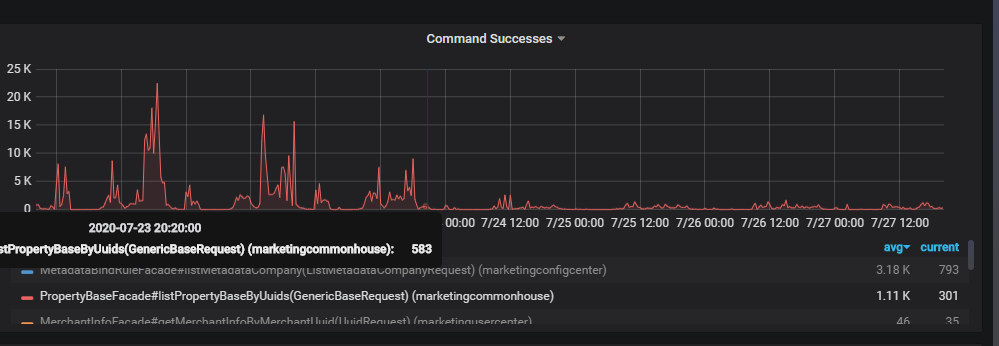
效果：

如下图，07-24上线后

1.微门店访问外部接口流量下降（75K-->50K）



2.微门店调用营销库反查房源图片接口大幅下降(1.5K平均到0.3K）



### 58发房只更新主动发布过的房源的相关信息，不全量同步到真方库

业务诉求：系统压力大，业务诉求：系统压力大；

解决办法：

1.发布房源的时候，补齐房源信息，图片信息，vr信息，视频信息

2.后台在房源信息，图片信息，vr信息，视频信息变更时，先检查是否有发布记录，有就同步，没有跳过。

### 58真房不开通不进行全量房源自动同步维护

业务诉求：系统压力大，冗余数据过多；

解决办法：

1．真房开通公司百分比不足10%，不需要全部几千家公司的房源维护，只同步必要的公司；

2．新开或重开真房，从营销库初始化全量同步一遍。

# #保证数据同步的准确性和成功率

## 2019-Q4 定时补偿

### 真房房源同步、下架、角色人、成交、带看定时补偿

### 发房房源下架补偿

下架补偿数据，看上架到几家门店，就尝试下架几家，不再判断是否一对一

### 真房、发房VR同步定时补偿

补偿数据，看同步到几家门店，就尝试补偿几家，不再判断是否一对一

### 真房、发房视频同步、视频解绑定时补偿

补偿数据，看同步到几家门店，就尝试补偿几家，不再判断是否一对一

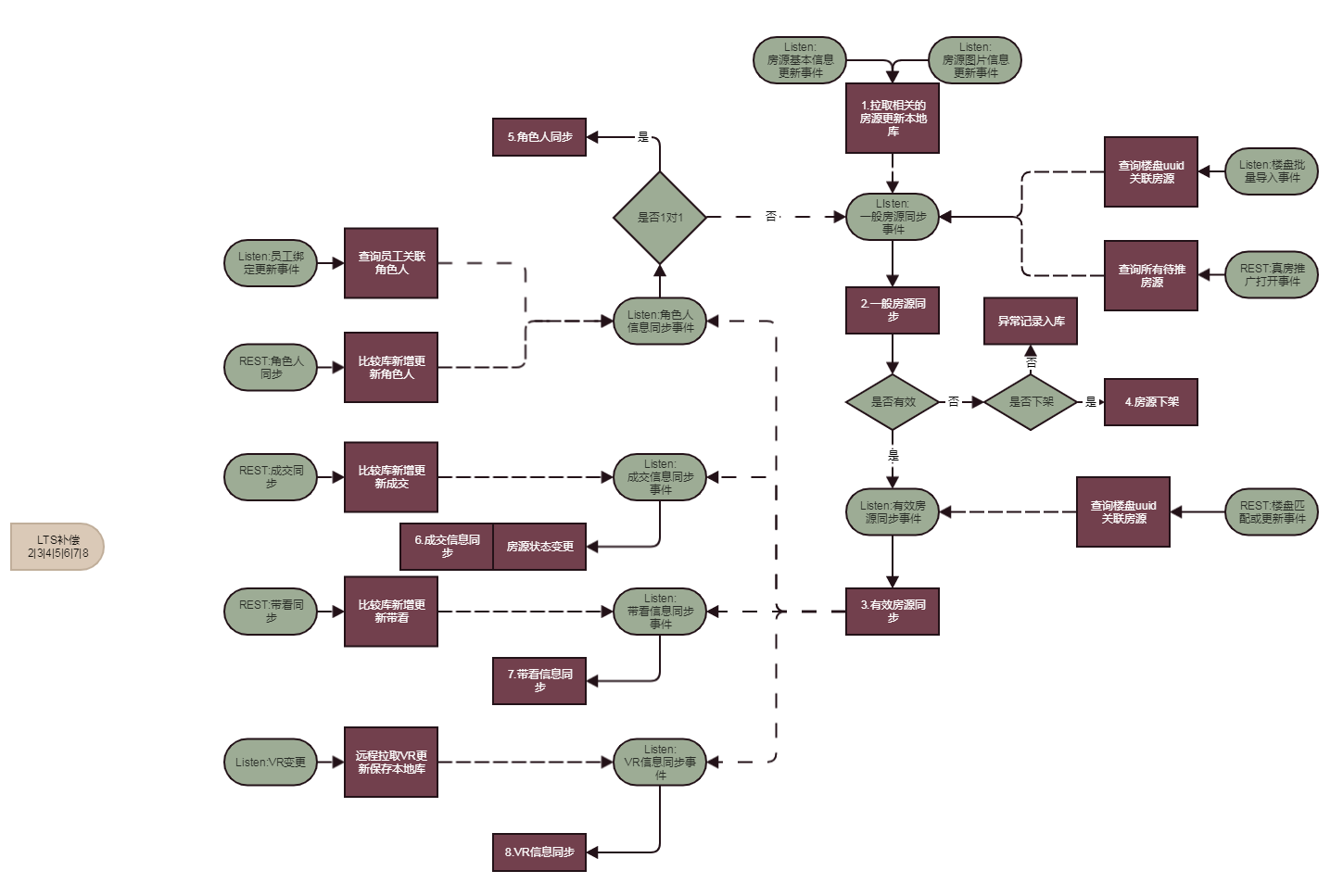
## 2020-Q1~Q3 依赖关系

### 营销库——图片、视频、VR依赖房源同步

图片、视频、vr消息到来时，检查房源同步关系，如果查不到，放到等待队列，进行重试。

### 真房——房源、图片、角色人、成交、带看有依赖同步

依赖同步的问题，可以使用消息机制来优化。消息机制需要先分析哪些是充分条件，哪些是必要条件，遇到充分条件，可同步；遇到必要条件，可尝试同步。



### 发房——房源、图片、营销信息三种消息组装使用

每种消息到来时候，需要先检查是否有房源同步成功记录，如果有，就拉取其他两种消息的数据，进行组装，同步58.

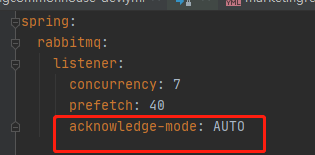
## 2020-Q2 updatedTime

### 拆分实时队列和定时队列，需要校验消息的正确性

多个队列来的数据，可能会出现后生成的数据先来，需要进行时间戳比较判断

## 中间件rabbitmq的事务

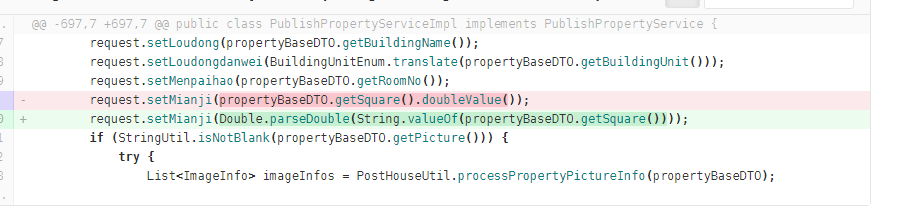
Rabbitmq结合spring的需要配置为下述形式



Rabbitmq本身提供事务和无事务两种功能，spring进行异常进事务的补充。

# 数据类型转换常见bug

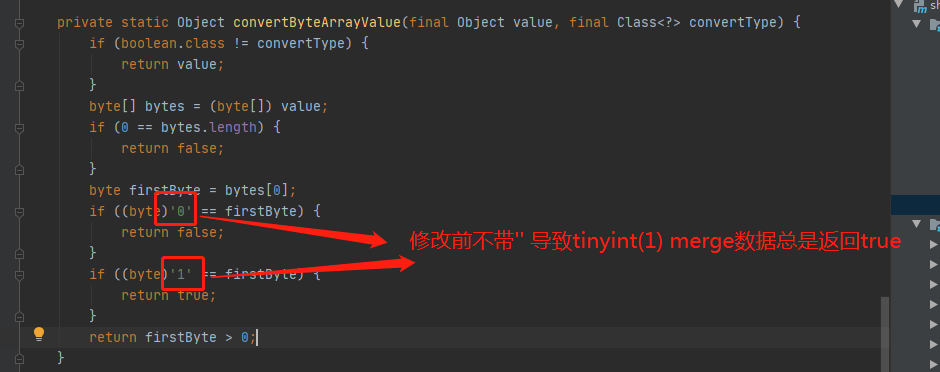
### Float转为double的小数点增加问题



### BeanUtils.copyProperty的target中遇到非包装类型字段可能报错

Target的属性中如果有非包装类型，就要求source中有此字段，否则报错

### Byte类型转换（sharing-jdbc bug）

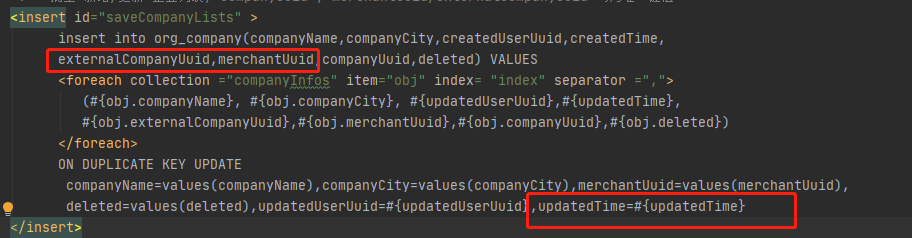


# duplicate key 问题

### V10迁移V20公司Duplicate key导致只更新updatedTime

背景：数据库中的唯一key为merchantUuid-externalCompanyUuid；现在需要将V10公司迁移到V20，如何迁移？

|  |  |  |
| --- | --- | --- |
| **merchantUuid** | **externalCompanyUuid** | **UpdatedTime** |
| V10-merchantUuid | Company1 | 2019-08-02 |
| V20-merchantUuid | Company1 | 2020-08-02 |



修改前：

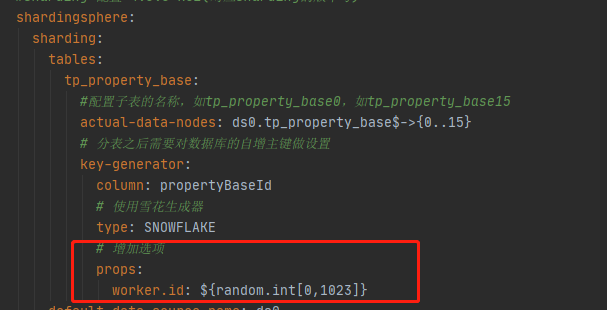


修改后：



### Sharding-jdbc自增主键冲突

调整雪花生成器的策略



### 房源同步到营销库，companyUuid+propertyUuid冲突

因为kafka消息的key是companyUuid+propertyUuid，但是每批次拉到的数据可能是属于不同公司的，后面的逻辑，必要时要进行group by操作

# 集合操作BUG

## Collectors操作 问题

### Collectors. groupingBy 分组键为空报错

#### 设置区域商圈缓存，没有判断区域为空的情况，导致空指针异常



#### 外部传来的companyUuid为空，导致报错



### Collectors. groupingBy 分组后的各list进行集合操作不会影响原list

在子组vrInfos0进行remove操作，不会影响到vrInfos的size。





## Collections类操作问题

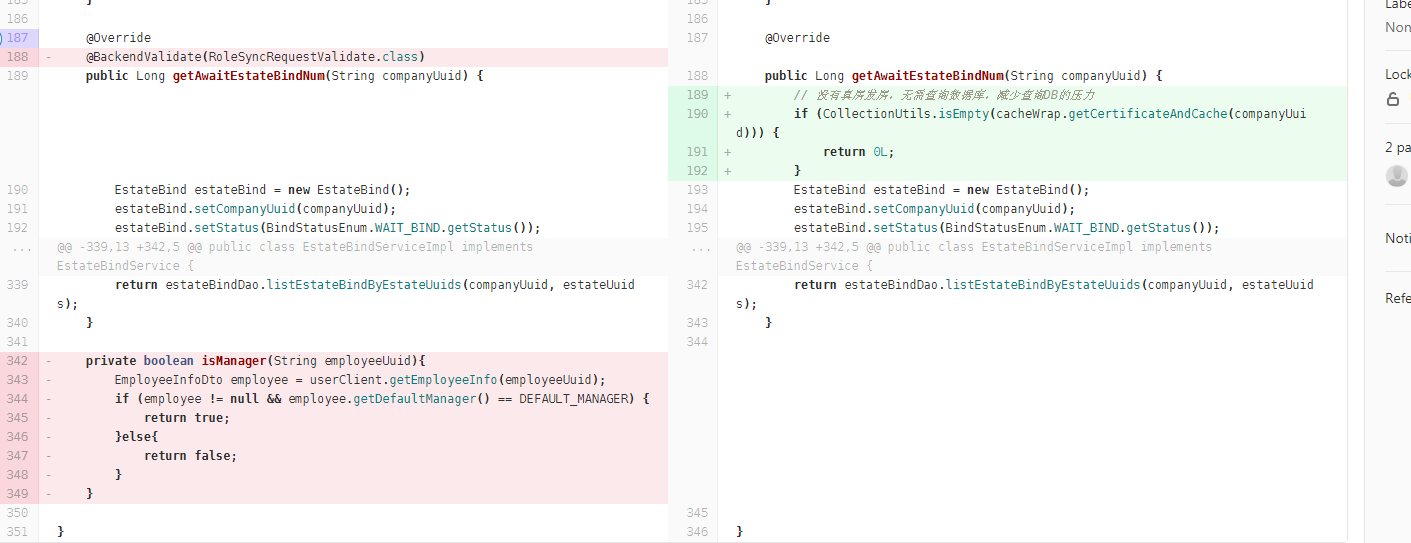
# MD5算法常见错误

### 错误的补0



# 减少高频接口不必要的 数据库查询

### 查询楼盘绑定

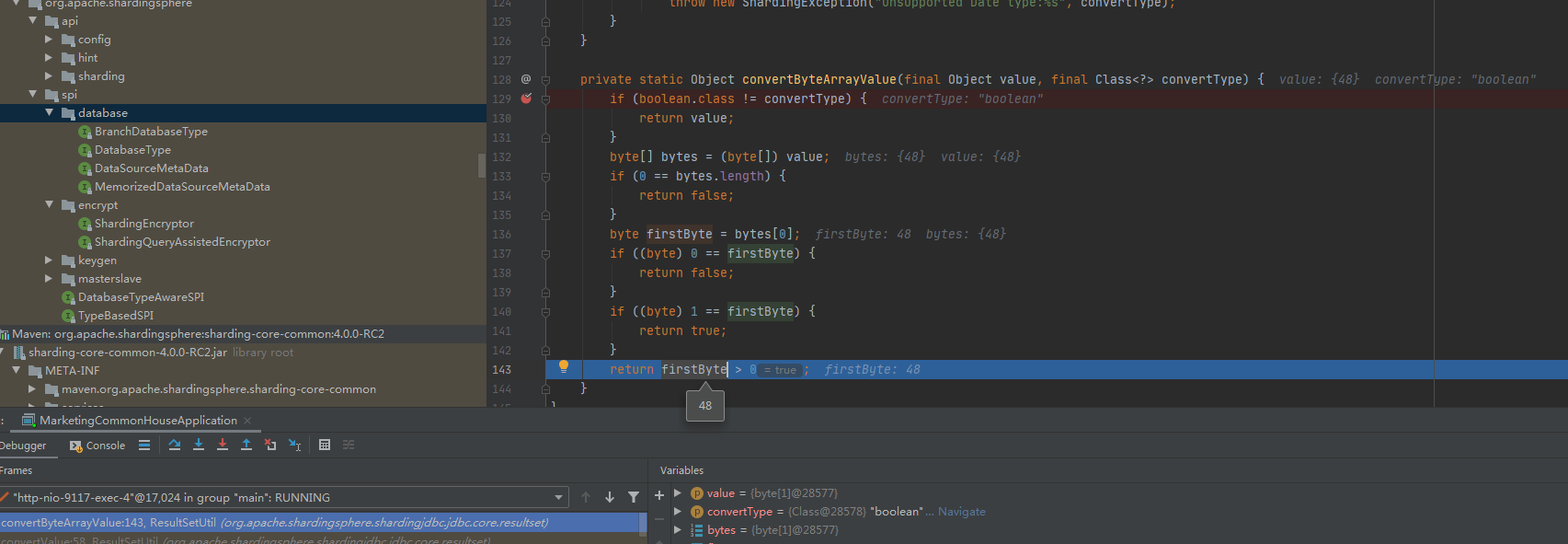


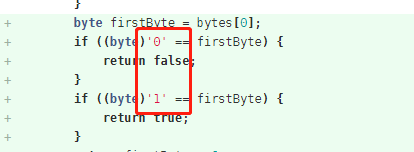
# Tidb特殊优化

### 使用tinyint(1)替代bit(1)

# Sharding-jdbc的bug

### Sharding的mergeResultSet提取布尔值错误





# Spring-Kafka

### 丢失数据可能

Kafka在手动提交时候，发生异常，会进行消息重试；但spring-kafka实现机制不同，不会立即重试，继续往后消费。而是下次重启spring-kafka时候，重新扫描offset，找到上次未消费的offset。如果下次重启之前，消息已经从磁盘消失，将不能继续消费。

springboot 整合 Kafka 消费时，消费者服务器每次重启，消费偏移量都从零开始消费。

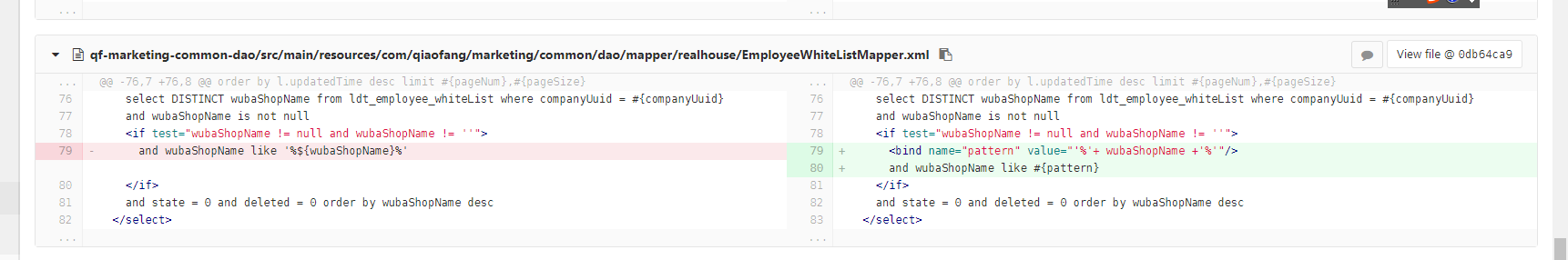
# Spring-Rabbitmq

### Auto模式

Rabbitmq提供两种事务模式，手动事务和无事务，spring-rabbitmq进行封装，增加了auto模式，auto模式可以在发生未捕获的异常时候，进行消息重试。

# 其他错误

### SQL注入错误



### Redis大value性能问题

阿里云会进行redis流量的拦截，虽然本身性能并不高，300KB一次请求，积累100次，就是30M。能用内存的就不要使用缓存，可以在系统启动时候自动加载

### int（19）

Mysql的int(19)还是只能存int(11)，这是假象。Insert on duplicate key 会导致数据的自增不停增加，最终可能有一天导致自增主键超过int类型。

# 自动化

### 58接口权限自动开通

系统启动时候，进行compareAndSet，来初始化接口权限，避免每次修改配置文件后，还要做一系列操作。

### 系统启动时自动加载配置

多节点刻在系统启动时加载配置，每次重启就生效，最大程度提升性能，如usercenter这种核心服务。

# 灰度

### 发房虚拟门店灰度开关

使用灰度开关，可以配合自动化程序，实现定向的灰度，如果灰度时间结束，开关可以取消。如真房优品开关，虚拟发房开关、新角色人开关、新楼盘匹配开关，发房禁止编辑营销价格。

# 第三方

### SDK提高单元测试效率

### SDK就是知识库

### SDK的日志同样重要

避免不成熟的第三方问题太多，扯皮太多