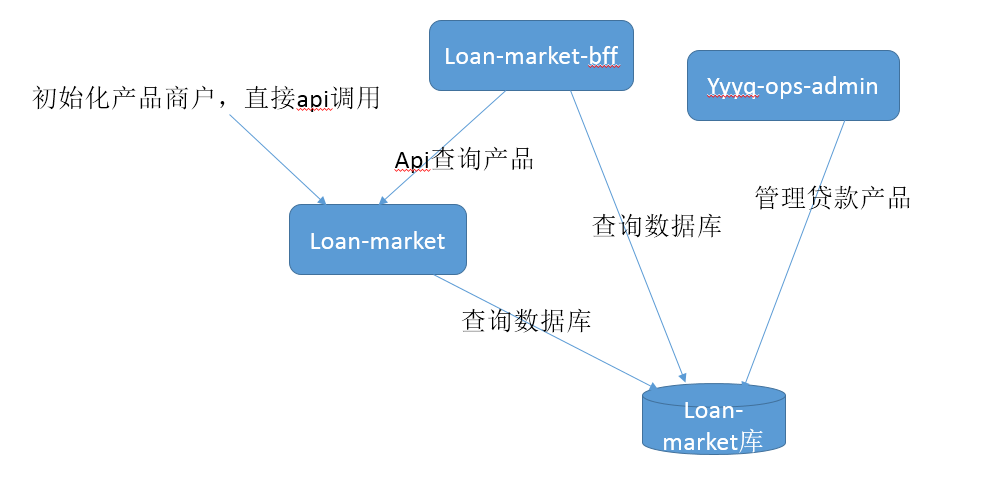
产品服务

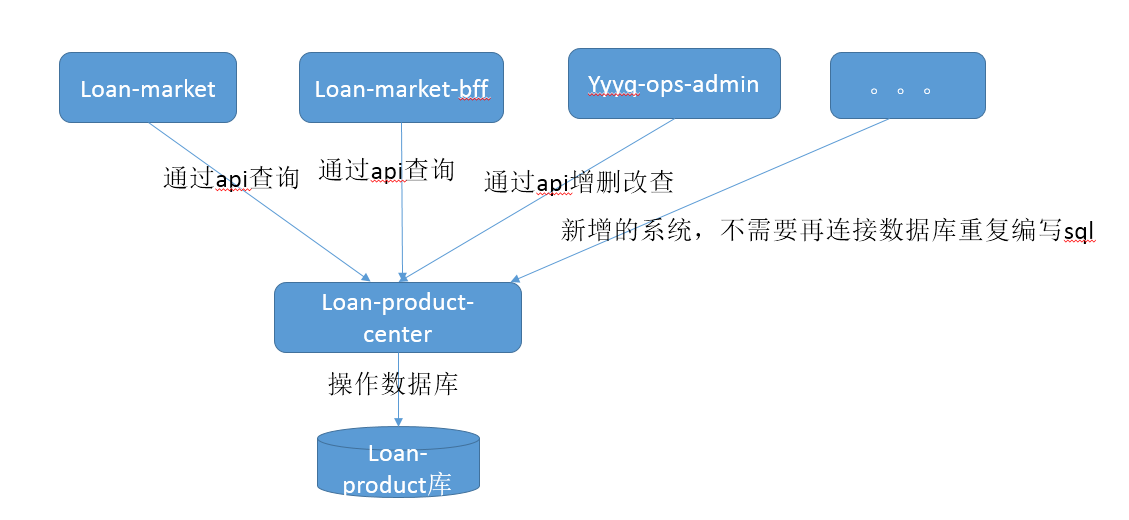
**一、贷超产品相关功能的现状。**



贷超相关功能散落在各系统中，调用关系混乱，而且都有直接操作数据库，管理不容易。

一些问题，比如：1、产品相关的一些枚举，各系统都有维护，但是又不一致。2、产品相关的功能分布在各处，如果要修改，到处都得检查，如果有的地方没有修改，要么逻辑有问题，要么程序直接报错。3、功能没有专人维护，开发人员自己需要了自己写一个功能，很混乱。

**二、贷款产品抽取后的情况：**



**产品服务独立出来好处：**

1. 如果贷款产品更改，只需要调整loan-product-center工程，对外提供的接口不需要修改，减小改动量。
2. 现在的贷款产品都是贷超产品，后续如果需要增加其他类型贷款产品，也可以放在一起，来集中管理。

贷款产品抽取成单独的服务，实现对贷款产品集中管理，对外提供服务。算是比较底层的服务，对外提供服务的接口设计尽量通过领域模型设计的思想来设计，其他系统都直接调用产品服务的api就好了。

贷款产品域主要包括贷款基本信息、贷款的标签（属性标签和推荐标签），申请贷款需要的材料等数据。贷款产品服务抽取，主要是将贷款产品相关的功能集中在一起，统一对外提供贷款产品服务api。

**三、贷款产品抽取涉及的范围：**

涉及的系统：yyyq-ops-admin, loan-market,loan-market-bff

**yyyq-ops-admin系统**



1、分页查询贷款产品/loan\_products

2、分页查询上架贷款产品/loan\_products/active

3、查看所有上架产品/loan\_products/activeall

4、查询推荐产品列表/loan\_products/recommend 查询所有上架的产品

5、更新上架产品的排序顺序/loan\_products/sort

6、添加贷款产品/products

7、更新贷款产品/products

8、更新贷款产品状态/products/{productNo}

9、确认下架贷款产品/products/doUpdateProductActive/{productNo}

注：这个功能会检查产品是否有关联分类，是否有关联推广位。需要loanmarket，channel- promotion提供相应的接口查询

10、根据产品编号查询产品/products/{productNo}

11、获取所有产品所需材料/products/application\_materials

12、查询产品标签类型/tags/types

13、添加产品标签/tags

14、更新产品标签/tags

15、根据id查询产品标签/tags/{id}

16、查询所有产品标签/tags

17、查询所有推荐标签/tags/recommendTags

19、分页查询属性标签/tags/attributeTags

20、删除产品标签/tags/{id}

**loan-market-bff系统**：

com.yinghuo.yyyq.app.domain.product.service.LoanProductDomainService

1、根据申请材料编号查询贷款产品

2、根据贷款名称查询

3、根据产品编号查询

4、获取产品所需材料列表/loan\_products/application\_materials

5、根据条件以及排序加载所有产品详情列表/loan\_products

6、根据产品编号加载产品详情/loan\_products/{productNo}

7、查询分类关联的产品，查询贷款申请材料，查询标签，最终是通过产品编码来查询的贷款产品。

**loan-market系统：**

1、根据产品编号查询产品详情/loan\_products/query\_by\_product\_no

2、查询上线产品列表/loan\_products/products

3、贷超发布管理服务，这两个功能是直接调用接口来使用的。

4、初始化产品商户/merchant\_init

5、初始化指定产品商户/merchant\_init/{productNo} 这两个最终会调用update产品商户，tag信息，申请材料信息

6、推荐产品查询

7、联合登录有查询产品

8、商户订单查询贷款产品，创建订单是检查贷款产品是否是api合作

1. 还款查询贷款产品

**四、涉及的数据库表：**

Loanmarket\_loan\_product,

loanmarket\_product\_tag,

loanmarket\_product\_tag\_rel,

loanmarket\_product\_application\_material\_rel,

loanmarket\_application\_material

**五、分步骤进行。**

第一步先将yyyq-ops-admin, loan-market,loan-market-bff系统中与产品相关的功能复制到product-center提供服务，相同的功能合并在一起。

第二步修改yyyq-ops-admin, loan-market,loan-market-bff系统，将涉及到产品的查询数据库的代码改成调用product-center提供的服务来实现相同的功能。

**六、待优化的地方，只是一些拙见：**

1、如果产品服务的qps很高，数据库有压力，可以加入缓存，将经常访问到的产品缓存起来。

2、产品服务应该分配专人负责，不能谁都可以加功能改表。大家都来改，没有统一的规划，代码没有统一的风格，也会很乱，不好维护，出了问题也不好排查。不过一个系统要有个backup。

3、如果产品服务变得很重要，高可用方面：mysql做高可用集群，如果缓存，也需要做缓存集群。如果放入缓存的数据比较大的话，还得做缓存优化设计，来减少内存消耗。

4、如果产品数据量大，可以做水平分表。不过不太可能。

其实这些都是情况而定。做缓存，高可用，都会增加系统的复杂度，势必也会增加成本。

如果服务没有那么多访问量，就没必要用缓存。

如果服务挂一会儿没关系，也可以不用做高可用。

**埋点数据可视化**。

业务主要流程

埋点数据写入kafka，然后从kafka消费，转换数据存入influxdb，然后用grafana可视化。

主要解决的问题是防止数据丢失。主要情况就是influxdb出现了问题的情况，数据不丢失。主要是靠kafka消费手动提交，成功存入influxdb后，再提交。当数据存入influxdb成功，需要重新消费数据，尝试重新写入influxdb。在这里遇到了一个问题，没有提交，还是有数据丢失。解决办法是，kafka的消费端如果消费不成功，需要初始化，重新消费。

这里比较关键的组件，kafka，延伸一下，就是rocketmq。

**网关系统**

主要是ZuulFilter