# 智慧金融大数据分析

## 需求分析

1. 致力于金融大数据的数据治理，展示更加准确的数据
2. 实时数据监控，分析个人行为数据，
3. 实现用户画像
4. 风控
5. 反欺诈等

## 2.、概要设计

一、简介：

该项目主要分为三个部分，**智慧金融大数据分析**，**大数据计算**，**大数据展示**。

1、智慧金融大数据分析：使用python爬取用户数据分析用户数画像。

2、智慧金融大数据计算：

使用flink实现放款金额的实时数据监控。

使用ETL实现数据报表的展示。

使用深度学习分析用户风险行为，反欺诈等

3、智慧金融大数据展示：使用web技术展示数据分析和计算的结果

### 二、技术架构及其应用场景：

1、Python：用来爬取用户信息和同行信息（风控数据，是否在其他金融贷款平台的黑名单中），使用python在之后的数据仓库中进行数据分析，数据的清洗和导入，使用机器学习实现风控系统对用户风险的估值和评分。

2、Mongodb：存储非结构化数据，使用mongodb存储用户的**通话记录**和**短信数据**，存储用户的行为日志

3、Redis：主要存储短信验证码和页面数据缓存

4、Mysql：结构化数据的源头,用户基本信息，交易流水，等等。

5、Hadoop:存储数据和分布式计算

7、Zookeeper：多个服务器配置的管理

8、Kafka：流式处理（用户数据实时监控，贷款金额监控）和（用户行为数据分析）日志的分析

9、Hive：数据的分析和查询

10、Hbase：分布式数据库存储

11、Flume：收集日志的工具

12、Sqoop:用于在关系数据库，数据仓库和hadoop之间注意数据

13、Spark：基于内存的分布式计算框架，可以用流计算

14、Flink：流计算

15、Elasticsearch：分词，全局搜索

16、Vue：数据分析后的显示

17、Springcloud：分布式web框架

18、数据分层

db：结构化数据（mysql），半结构化数据（redis,mongodb），

ods: 全称是 Operational Data Store，操作数据存储.“面向主题的”，数据运营层，也叫ODS层，是最接近数据源中数据的一层，数据源中的数据，经过抽取、洗净、传输，也就说传说中的 ETL 之后，装入本层。本层的数据，总体上大多是按照源头业务系统的分类方式而分类的。但是，这一层面的数据却不等同于原始数据。在源数据装入这一层时，要进行诸如去噪(例如有一条数据中人的年龄是 300 岁，这种属于异常数据，就需要提前做一些处理)、去重(例如在个人资料表中，同一 ID 却有两条重复数据，在接入的时候需要做一步去重)、字段命名规范等一系列操作。一个用于存储当前需要加载的数据，一个用于存储处理完后的历史数据。历史数据一般保存3-6个月后需要清除，以节省空间。

dw：是数据仓库的主体.在这里，从 ODS 层中获得的数据按照主题建立各种数据模型。DW层的数据应该是一致的、准确的、干净的数据，即对源系统数据进行了清洗（去除了杂质）后的数据。这一层的数据一般是遵循数据库第三范式的，其数据粒度通常和ODS的粒度相同。在PDW层会保存BI系统中所有的历史数据，例如保存10年的数据。

数据仓库层是我们在做数据仓库时要核心设计的一层，在这里，**从 ODS 层中获得的数据按照主题建立各种数据模型**。DW层又细分为 DWD（Data Warehouse Detail）层、DWM（Data WareHouse Middle）层和DWS（Data WareHouse Servce）层。

1、数据明细层：DWD（Data Warehouse Detail）

**该层一般保持和ODS层一样的数据粒度**，并且提供一定的**数据质量**保证。同时，为了提高数据明细层的易用性，该层会采用一些**维度退化手法**，将维度退化至事实表中，减少事实表和维表的关联。

另外，在该层也会做一部分的数据聚合，**将相同主题的数据汇集到一张表中**，提高数据的可用性，后文会举例说明。

2、数据中间层：DWM（Data WareHouse Middle）

该层会在DWD层的数据基础上，对数据做**轻度的聚合操作**，**生成一系列的中间表，提升公共指标的复用性**，减少重复加工。

直观来讲，就是对**通用的**核心维度进行聚合操作，算出相应的统计指标。

3、数据服务层：DWS（Data WareHouse Servce）

又称数据集市或宽表。**按照业务划分**，如流量、订单、用户等，**生成字段比较多的宽表**，**用于提供后续的业务查询，OLAP分析，数据分发等。**

一般来讲，该层的数据表会相对比较少，一张表会涵盖比较多的业务内容，由于其字段较多，因此一般也会称该层的表为宽表。

在实际计算中，如果直接从DWD或者ODS计算出宽表的统计指标，会存在计算量太大并且维度太少的问题，**因此一般的做法是，在DWM层先计算出多个小的中间表，然后再拼接成一张DWS的宽表。由于宽和窄的界限不易界定，也可以去掉DWM这一层，只留DWS层，将所有的数据在放在DWS亦可。**

DM层：为数据集市层，这层数据是面向主题来组织数据的，通常是星形或雪花结构的数据。从数据粒度来说，这层的数据是轻度汇总级的数据，已经不存在明细数据了。从数据的时间跨度来说，通常是PDW层的一部分，主要的目的是为了满足用户分析的需求，而从分析的角度来说，用户通常只需要分析近几年（如近三年的数据）的即可。从数据的广度来说，仍然覆盖了所有业务数据。

app: 这一层是提供为数据产品使用的结果数据

分层好处：

1、清晰数据结构：每一个数据分层都有它的作用域和职责，在使用表的时候能更方便地定位和理解

2、减少重复开发：规范数据分层，开发一些通用的中间层数据，能够减少极大的重复计算

3、统一数据口径：通过数据分层，提供统一的数据出口，统一对外输出的数据口径

4、复杂问题简单化：将复杂的任务分解成多个步骤来完成，每一层只处理单一的步骤，比较简单和容易理解。当数据出现问题之后，不用修复所有的数据，只需要从有问题的步骤开始修复。

5、屏蔽原始数据的异常：不必改一次业务就需要重新接入数据。

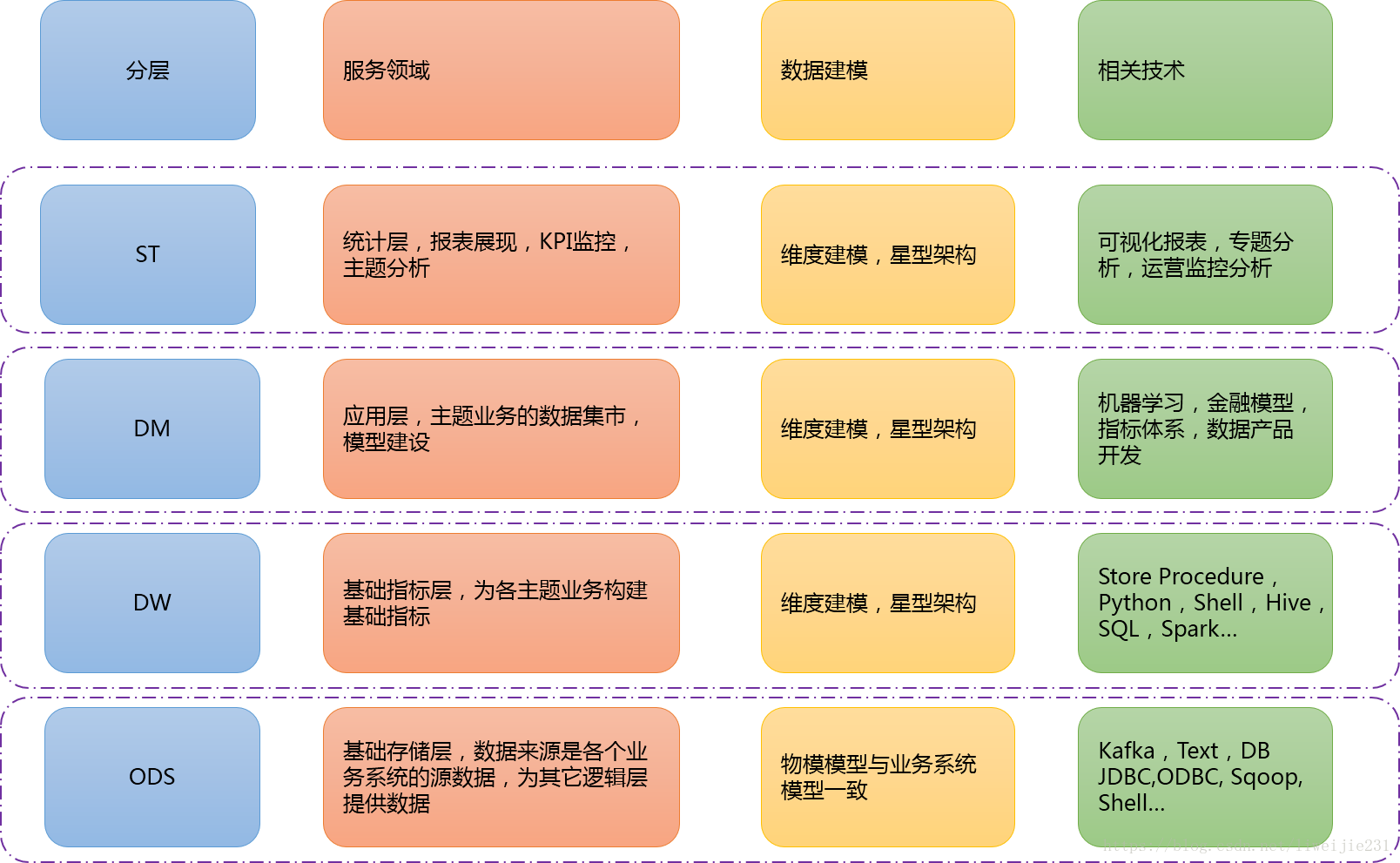


图1.1

数据建模：

很多人一听到数据建模，就觉得高不可攀，觉得是很高深难以理解的东西，其实简单来说，数据建模就是搞清楚每个表都有哪些字段、表之间有什么联系，然后根据需要添加字段或度量值、建立关系的过程。

字段值、字段类型、表、表之间的关系，都是数据模型的一部分，在PowerBI中，建立的度量值同样是模型的一部分。数据建模不难理解，也并不代表数据建模就很简单，当表比较少并且结构简单的时候，数据建模确实不难，但当表的数据达到上百个，关系错综复杂的时候，想建立好一个模型并不是那么简单的，你需要掌握的不仅是建模知识，更要深入了解业务之间的内在联系，并通过建立关系的方式，将业务的内在联系可视化。

详细https://blog.csdn.net/weixin\_45399233/article/details/100551818（来源网上的博客）

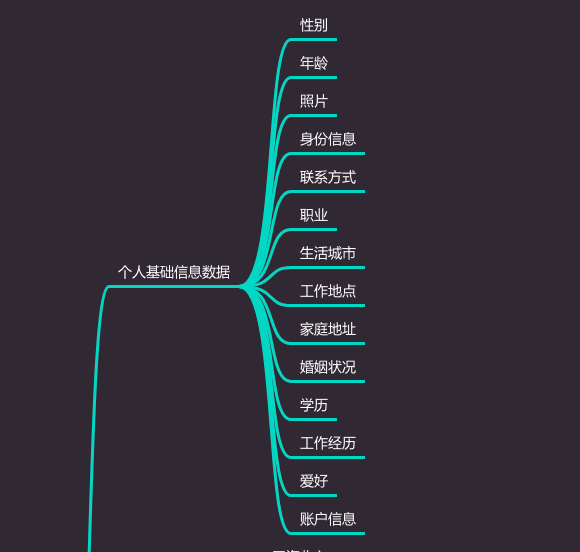
### 模块设计

#### 1、人物画像

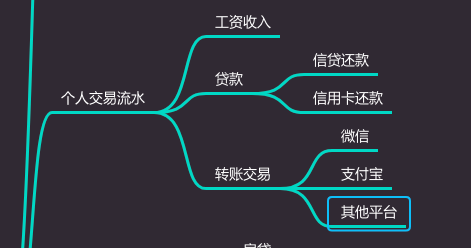
简介：通过爬虫获取户个人信息和交易流水返回的json字符串（由于涉及到个人隐私一些重要数据可以采取模拟的方式）

，用于当做风控分析的参数。其中包括

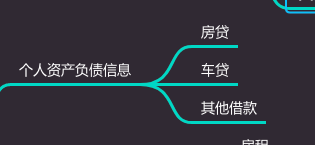
1. 个人基础数据：



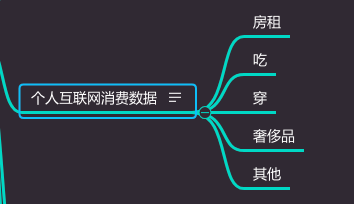
二、个人交易流水



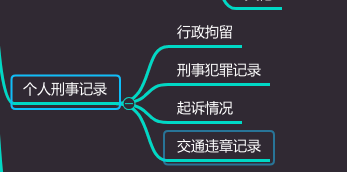
三、个人资产信息和负债



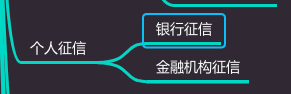
四、个人互联网消费数据



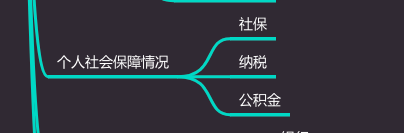
五、个人刑事记录



六、征信



七、社会保障情况



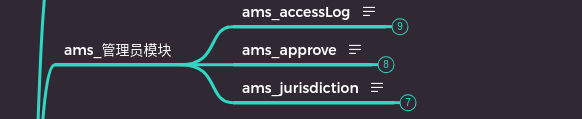
八、三方催款记录



#### 2、贷款系统模块（因为是企业的内部数据所以可以采用模拟的方式或者爬虫爬取的方式。意在模拟企业真实的数据环境,这里使用原来系统里面的模块稍加优化）

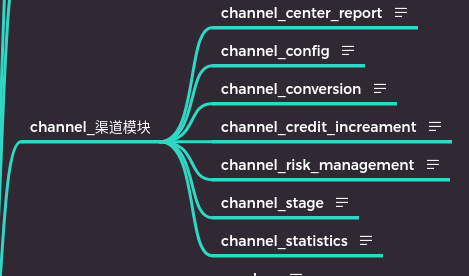
一、管理员模块

简介：用来管理管理员信息和后台请求日志，登录日志等



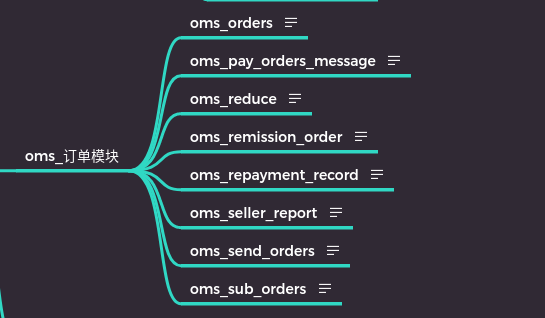
二、渠道模块

简介：渠道的意思是从哪个通道来的，比如我投放了一个广告在哔哩哔哩那么这个哔哩哔哩就是可以称为一个渠道来源。



三、订单模块

简介：用来存储用户的订单信息，包括订单的逾期，续借，还款，减免等



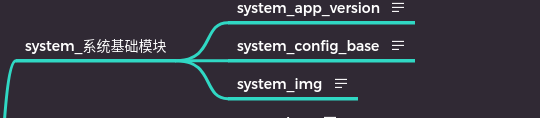
四、风控对接模块

简介：传入用户画像数据分析用户的风险情况，风控分越高代表个人信息用度越高。这边的老系统用的是三方接口对接返回的数据，这个模块也是我们接下来着重开发的模块。



五、系统基础模块

简介：用来存储系统配置和图片等



六、用户模块

简介：用户模块的数据是重点来源



七、催收模块

简介:主要管理催收公司的数据，这个模块在本项目不重要

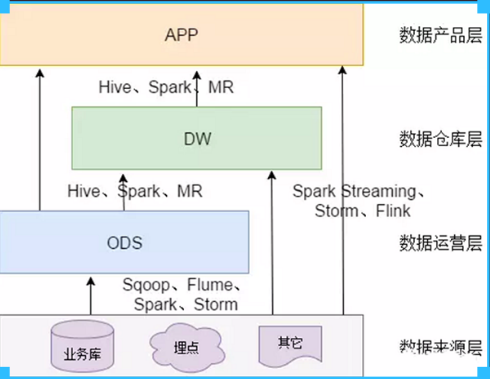
#### 3、风控模块

简介：传入人物画像分析得出用户的风控分。

### 四、数据结构和数据库设计

#### 1、简介

看了上面的ETL数据仓库的设计思路，为了更快的查找数据和实现大数据量的存储，不难得出本项目的数据分层主要分为五层。



#### 第一层db：结构化和半结构化数据库

作为现有的一套业务实体系统（网贷app）的原始数据，分散在各个数据库和消息中间中（mysql，mongodb，redis,elastisSearch，kafka）

在这些数据中我们要抽取一些有用的数据（去重，去掉无用数据）存储到下一层ods层保存3到六个月作为历史数据

eleasticSeach：

Mongodb：

1. 存储用户个人画像json
2. 运营商返回的json数据

Redis：

1. 短信验证码
2. 页面缓存数据

Kafka;

1、订单数据

mysql数据库设计：（由于这个数据库是我之前公司做的业务改编过来，只需留下有用表）

（设计完模块后补上）

#### 第二层db：ods层

简介：该层主要存储3-6月的临时数据

使用的中间件：

Hadoop：hdfs的数据清洗

Zookeeper：集群的配置管理

Hbase：使用mapReduce写入hbase进行数据的存储

Sqoop：从hive中导出数据到mysql

Flume：日志采集

#### 第三层db：dw层（数据仓库）

简介：该层主要存储所有的历史数据，DW层的数据应该是一致的、准确的、干净的数据，即对源系统数据进行了清洗（去除了杂质）后的数据。数据仓库层是我们在做数据仓库时要核心设计的一层，在这里，**从 ODS 层中获得的数据按照主题建立各种数据模型**。DW层又细分为 DWD（Data Warehouse Detail）层、DWM（Data WareHouse Middle）层和DWS（Data WareHouse Servce）层。

1. 数据明细层：DWD（Data Warehouse Detail）

hbase

1. 数据中间层：DWM（Data WareHouse Middle）

hive

3、数据服务层：DWS（Data WareHouse Servce）

clickhouse

使用的消息中间件：

Hadoop：hdfs的数据清洗

Zookeeper：集群的配置管理

Hive：从ods层中hbase中抽取数据进行清洗写入hive进行分析

ClickHouse：作为数据服务层的数据存储

数据建模：

（设计完模块后补上）

#### 5、第四层db：DM层(数据集市)

消息中间件：hive

#### 第五层db：App层（数据应用层）

数据库:

Mysql:

Redis:

Flink: