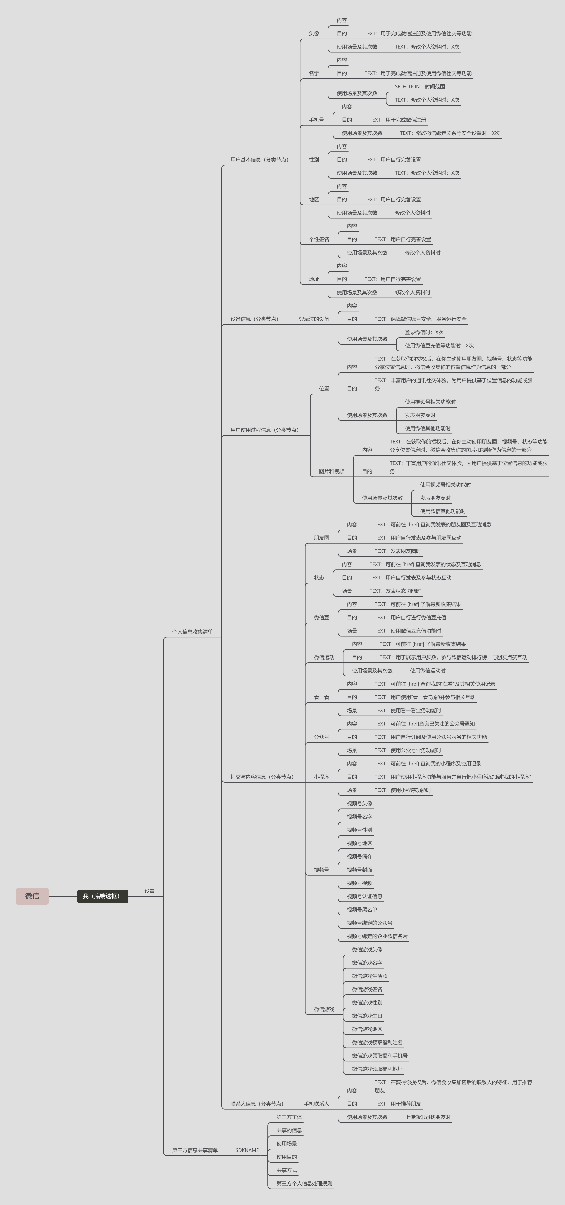
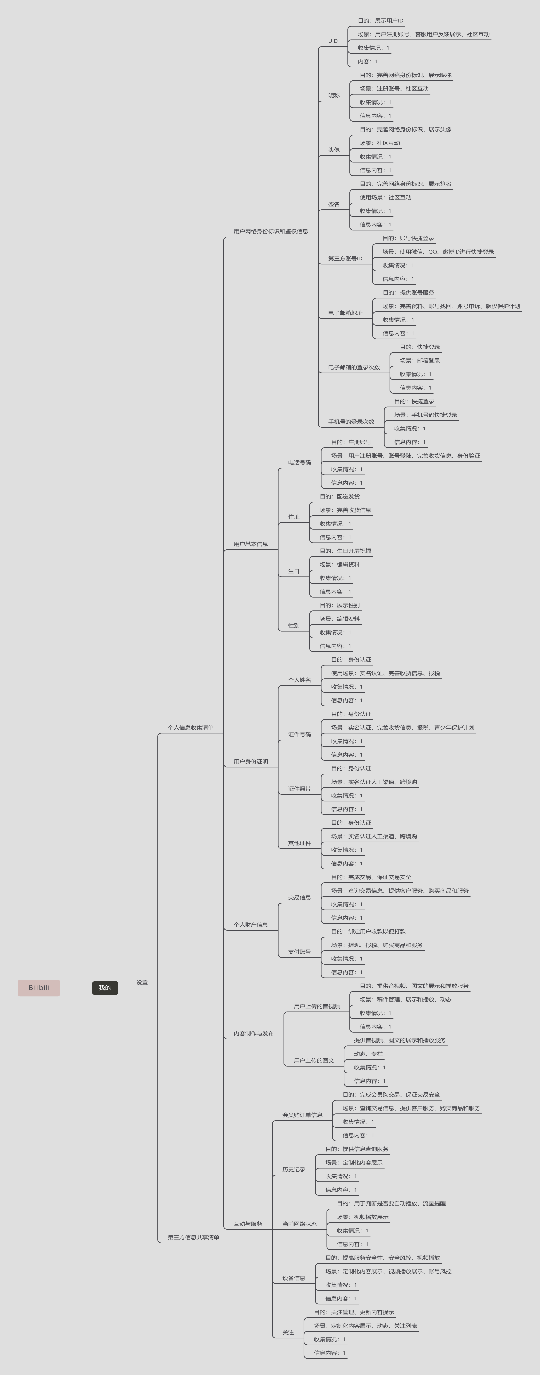
# 2023-3-29

## 需求分析

## 逻辑结构

### UML图



### 解释

#### 实体解释

* 软件
* 个人信息收集清单

各个软件的个人信息手机清单Button内，体系复杂且多样，并且带有多种结构，如：Button跳转+表结构展示 / Label展示分类+Button跳转+label显示内容 / etc.

由于结构多样且跳转复杂，且有少量app带有深层路径，因此采用可扩展性强的树状结构存储，并且将展示分类的设为分类节点，将跳转的button设为不带内容的中间节点，将能够跳转后显示内容的button/表格型数据设为带内容的叶节点，以此存储每个app个人信息收集清单的数据。

* 第三方信息共享清单

每个app的第三方信息共享清单均是由跳转button进入页面，表格呈现清单内容，无层次结构，因此采用常规存储方式。

第三方信息共享类型每个app设定不同，大体分为SDK、关联商、合作商。

#### 字段解释

- Software:

- **Software\_ID**：作为唯一标识软件的主键

- Software\_Name：软件名称

- Company\_Name：公司名称

- Software\_Version：软件版本号

- Software\_Trace：软件到达双清单的路径

- PersonalList:

- **PLNode\_ID**：唯一标识个人信息清单节点的主键

- Software\_ID：标识软件的外键

- PLNode\_Name：节点名称

- PLNode\_Type：节点类型，有三种（分类节点0，无内容节点1，有内容叶节点2）

- PLNode\_Relation

- **Parent\_ID**：树形结构父节点ID

- **Child\_ID**：树形结构子节点ID

- PLNode\_Attributions（有内容叶节点的具体信息）

- **PLNode\_ID**：唯一标识个人信息清单节点的主键

- PLNode\_Name：节点名称

- PLNode\_Purpose：收集目的

- PlNode\_UseCase：收集场景

- PLNode\_Collect\_Situation：收集情况是否展示

- PLNode\_Content：收集内容是否展示

- ThirdList

- **TLNode\_ID**：唯一标识第三方共享清单的主键

- Software\_ID：标识软件的外键

- TLNode\_Type：节点类型（SDK，合作商，关联商，etc.）

- TLNode\_OS：（iOS，Android）

- TLNode\_Name：第三方共享信息工具名称

- TLNode\_Company：第三方公司名称

- TLNode\_InformationContent：共享信息内容

- TLNode\_Purpose：收集目的

- TLNode\_UseCase：收集场景

- TLNode\_Way：收集方式

- TLNode\_Detailed：隐私政策细节

## 数据库实施

### 可拓展性探求

针对53款app的双清单人工考虑是否满足数据库的设计模式。



### 数据填入

对于数据库数据填充，我们采用的方式是用幕布人工填入半结构化文本（兼有思维导图的功能）；导出为opml，使用python脚本解析并填入数据库。

## 附1

use demo;

create table software (

     Software\_ID int ,

     Software\_Name nvarchar(20),

     Company\_Name nvarchar(60),

     Software\_Version nvarchar(20),

     Software\_Trace nvarchar(40),

     Primary key(Software\_ID)

     );

*-- 0:没有内容的节点 1:有内容的叶节点*

 create table PersonalList (

    PLNode\_ID int,

    Software\_ID int,

    PLNode\_Name nvarchar(20),

    PLNode\_Type tinyint,

    Primary key(PLNode\_ID),

*foreign key*(Software\_ID) *references* software(Software\_ID),

*Check* ( PLNode\_Type in (0, 1) )

 );

create table PLNode\_Relation (

    Parent\_ID int,

    Child\_ID int,

    Primary key(Parent\_ID, Child\_ID),

*foreign key*(Parent\_ID) *references* personallist(PLNode\_ID),

*foreign key*(Child\_ID) *references* personallist(PLNode\_ID)

 );

*-- 0：无 1：有*

create table PLNode\_Attributions (

    PLNode\_ID int,

    PLNode\_Purpose nvarchar(100),

    PlNode\_UseCase nvarchar(100),

    PLNode\_Collect\_Situation tinyint,

    PLNode\_Content tinyint,

    Primary key(PLNode\_ID),

*foreign key*(PLNode\_ID) *references* personallist(PLNode\_ID),

*check* ( PLNode\_Collect\_Situation in (0, 1)),

*check* ( PLNode\_Content in (0, 1) )

 );

*-- 没定好*

create table ThirdList (

    TLNode\_ID int,

    Software\_ID int,

    TLNode\_Type nvarchar(20),

    TLNode\_OS nvarchar(20),

    TLNode\_Name nvarchar(100),

    TLNode\_Company nvarchar(100),

    TLNode\_InformationContent nvarchar(300),

    TLNode\_Purpose nvarchar(100),

    TLNode\_UseCase nvarchar(100),

    TLNode\_Way nvarchar(100),

    TLNode\_Detailed nvarchar(300),

    primary key(TLNode\_ID),

*foreign key*(Software\_ID) *references* software(Software\_ID)

 );