微博系统开发计划见：《微博系统开发计划》

开发分工简单描述：

1、前台的框架由孙奥完成（目前已经完成）

2、后台的框架由王辉完成（吴剑协助）

3、爬虫穆德宝完成

4、后台管理系统放到目前的OA系统中完成

5、其它各模块每个模块的开发周期大约是2周完成，平均每人2个模块

开发过程和方法：

1、首先由王贺提供模块的模板（html和css）

2、开发人员根据前台开发规范开始写自己的模块。

3、当需要和后台交互的时候则实现一个Action，每个交互是一个Action。

4、当需要Action里面需要涉及到数据存储和访问的则有王辉统一处理。

5、基于Action的日志系统和权限系统是自动的，不用实现代码，只需注册和使用（授权）。

6、和后台的所有交互都是通过前台提供的AJ完成。

7、所有的页面是基于模板输出，由后台TemplateServlet提供，每个模块需要根据自己的权限控制来实现具体的IModuleTemplate的html方法返回合适的HTML内容。

数据库设计方案

1、数据库结构已经建好，具体不写了，每个字段都有说明。

服务器地址：192.168.2.13

数据库名称：xk\_weibo

2、数据库表命名按照（模块）\_（业务表）

3、目前所有的数据都放在RDBM里面，表后缀为$的为零时设计，将来可能存放在其它地方。具体参考下面的《微博关系存储方案demo》

4、采样数据表标记为\*，跟踪表是业务采样都以\_Track结尾。

5、所有的日志都记录192.168.2.13的xk\_weibo\_logs的xk\_logs表。

6、数据库表设计中有任何遗漏都可补充，但必须按照上面的约束修改和添加。

# 我的应用（APP\_\*）

## 应用

我的应用是一个官方的应用，需要让腾讯和新浪都审核通过，并且是一个稳定的可维护的应用，此应用目前在新浪上是【讯库舆情】应用。

同时是要一些爬虫应用，来获得一些API数据。

讯库应用为：讯库舆情

网站名称：实时舆情

网站域名：mudb.xunku.org

ICP备案信息：京ICP备11002306号-1

来源链接：http://mudb.xunku.org

网站分类：新闻

面向终端：个人电脑

网站简介：实时舆情

网站介绍：通过定制关键词信息，打造专属于自己的舆情监测平台。

App Key：3699809923

App Secret：eaf67a6525cf39a0fc443d03d9bc24d2

爬虫应用为：XXX的博客

【应用表】如下：Base\_Apps

|  |  |  |
| --- | --- | --- |
| ID |  |  |
| AppName | 应用的名称 |  |
| Url | 应用对应的URL |  |
| AppKey | 申请应用时分配的appkey | 新浪：client\_id  腾讯：client\_id |
| AppSecret | 申请引用时分配的appsecret | 新浪：app\_secret  腾讯：app\_secret |
| Manager | 应用的管理者 | 字符串即可，标识这个应用由谁来管理（创建）维护的 |
| Type | 申请这个应用的类型 | 1、Normal 这个应用是给我的微薄使用的普通应用  2、Crawler 这个应用是给爬虫使用的应用 |
| CallbackUrl | 回调地址 | 腾讯：redirect\_uri  新浪：redirect\_uri |
| platform | 平台 | 1、腾讯  2、新浪  3、人民 |

## 影子帐号

影子帐号是一些内部帐号，他们主要功能：

1、关注一些特别重要的人，授权给Crawler应用，这样可以获得这些特别重要人的动态信息。（【帐号监测】）

2、调用API获得其他有用的信息，数据采集。

在帐号监测中影子帐号的功能非常重要，影子帐号的基本信息和微博博主一样，并且他们是系统级别的授权帐号（相对企业级别）。

【影子帐号表】Base\_App\_Shadows

|  |  |  |
| --- | --- | --- |
| ID |  |  |
| uid | 帐号id | 参考博主数据结构描述 |
| platform | 平台 | 1、腾讯  2、新浪  3、人民 |
| Token | 授权票据 |  |
| AppID | 给哪个App授权 | 关联上面的App表 |
| Expired | 何时过期 |  |
| password | 帐号密码 |  |
| owner | 帐号所有者 |  |
| uids | 该帐号关注的帐号列表 | 【帐号监测】时使用，可动态变化的 |
| Times | 次数/小时 | 影子帐号使用App调用API了多少次，累加值，小时按自然小时算。 |

影子帐号是可以自动授权的。

【影子帐号调用表】Base\_App\_Shadows\_Log

|  |  |  |
| --- | --- | --- |
| 字段 | 描述 | 备注 |
| Id | 唯一标识 |  |
| time | 什么时候调用的 |  |
| sid | 影子帐号系统ID | Base\_App\_Shadows的主键 |

该表记录了每个影子帐号的调用日志。以方便统计和控制。

理论上来说每个影子帐号关注一个帐号这样可以保证数据是干净的，否则需要区分各种被关注人的动态。

讯库舆情应用

官微、微博系统授权帐号

应用管理器

XXX博客

授权

授权

影子帐号

应用管理器功能（AppManager.class）：

1、创建应用：应用本身是由外部人员创建并且申请审核通过的，这里是只创建一个已经审核通过的应用，将这个应用添加到系统中。

2、创建帐号：同上。

3、帐号授权：某个帐号给某个应用授权。

4、帐号续期：快到期的帐号需要续期，具体是通过某个用户调用API时检查到期时间，如果已经到期或者还有1天到期，则启动帐号授权对帐号进行授权。

**应用管理器是调用API的唯一出口！**

# 微博博主

博主是具体一个微博的帐号，包括微博帐号的基本信息，粉丝数，微博数，关注数等信息。

## 数据结构

|  |  |  |
| --- | --- | --- |
| 字段名 | 描述 | 备注 |
| Id | 系统唯一标识 | 微博系统内唯一标识 |
| ~~from~~ | ~~这条信息从哪来~~ | ~~1、API~~  ~~2、Crawler爬虫~~ |
| uid | 微博帐号ID，字符串类型 | 新浪：user.id 长度 int64 最多19个字符  腾讯：data.openid 长度32个字符  平台微博帐号标识，如果保存到key-value数据库中platform+uid是key，其他内容是value。  进一步简化，platform追加到原始id的第一位。所以每个uid或者tid的第一位是标识这个用户或者微博属于哪个平台。 |
| platform | 平台类型 | 1、腾讯  2、新浪  3、人民 |
| displayName | 显示名 | 新浪：user.name  腾讯：data.name |
| nickName | 昵称 | 新浪：user.screen\_name  腾讯：data.nick |
| city | 城市ID | 新浪：user.city  腾讯：data.city\_code  如果是抓取的，这个值需要通过下面的字段算出来。  保存的是讯库的City编号，提供转换方法来将不同平台的转换到讯库的City。  getCity(platform,code) |
| location | 所在地 | 新浪：user.location  腾讯：data.location |
| Description | 个人描述 | 新浪：user.description  腾讯：data.introduction |
| Url | 微博地址 | 新浪：user.Url  腾讯：无 |
| Gender | 性别 | 新浪：user.gender  腾讯：data.sex |
| regtime | 注册时间 | 新浪：user.created\_at  腾讯：data.regtime |
| followers | 粉丝数 | 新浪：user.followers\_count  腾讯：data.fansnum |
| friends | 关注数 | 新浪：user.friends\_count  腾讯：data. idolnum  腾讯里面叫收听的人数 |
| weibos | 微博数 | 新浪：user.statuses\_count  腾讯：data. tweetnum |
| favourites | 收藏数 | 新浪：user. favourites\_count  腾讯：data. favnum |
| verified | 是否是认证用户 | 新浪：user. verified  腾讯：data. verifyinfo |
| bi\_followers | 互粉数 | 新浪：user. bi\_followers\_count  腾讯：data. mutual\_fans\_num |
| ~~birth~~ | ~~生日~~ | ~~新浪：无~~  ~~腾讯：data. birth\_day+data. birth\_month+data. birth\_year~~ |
| tags | 用户的标签 | 如果取了就保存一下，否则为空，这个是比较消耗API的  新浪：只提供批量API  http://open.weibo.com/wiki/2/tags/tags\_batch  data.tags  腾讯：data.tag  多个标签用|分割 |
| monitor | 是否监测 | True为需要监测  False 为不需要监测  需要监测是指这个微博帐号在【帐号监测】里，否则为一般的帐号。 |

如果是爬虫提供的数据，除了city需要计算获得，其他信息均为空。

微博的URL腾讯没有？

博主的生日新浪未提供。（已删除）

## 存储方式

微博博主采用分布式的内存存储方式，目前暂定redis存储，可做分布式集群。

所有的微博帐号信息都保存在这个分布式集群上，持久化到SQL Server里。

具体方案如下：

Sql Server单表存储微博帐号信息。后台服务启动时将所有的帐号信息装载到redis里面。装载到redis里的目的有两个：

1、提供高速的内存访问和并发访问，根据uid能快速定位到用户的基本信息。

2、跨库时依然可以有效的提供数据。

粉丝关系也保存在SQL里面然后装载到redis里面，讯库的粉丝关系是**曾经是粉丝关系**，也就是说只要以前是粉丝，那现在也还是粉丝，用户主动移除这种关系，不能体现到存储上，除非有全量粉丝表。

SQL 博主粉丝

SQL 博主库

SQL 博主关注

redis存储

更新队列

服务接口

读取、分析接口

更新接口

SQL数据库:

xk\_weibo\_users： 讯库微博用户数据库。

基本信息表：

Users：微博用户基本信息表。【SQL博主库】

关系表，基于基本信息表。

Following：关注表，这里面保存的是谁关注了谁。【SQL博主关注】

Followed：粉丝表，这里面保存的是谁是谁的粉丝。【SQL博主粉丝】

## 服务接口

### 读取分析接口

博主服务接口描述：

1、根据uid获得微博基本信息（实际上是通过uid + platform）

user get(uid, platform)

2、给定一组uid，获得这组uid里面男的有多少，女的有多少

sex getsex(uids, platform)

sex{f:100,m:100,u:2}男的100，女的100，未知的2个。

这个计算是基于redis的。

3、给定一组uid，获得这组uid里面有多少vip

vip getvip(uids, platform)

vip{v:100}这组用户里面有100个是vip

4、获得粉丝里粉丝数最多的TOPN

user[] getfans(uid, topn, platform);

获得排好序的大粉丝列表

5、获得粉丝的地域分布

region[] getregion(uid, platform);

region{r1:100,r2:200} r1区域有100个粉丝，r2区域有200个粉丝。

### 更新接口

1、更新博主信息

void update(user, platform)

首先跟新redis的内容，其次是将更新内容发送到【消息队列】等待更新SQL数据库。

服务提供方式为REST。

redis 部署目前采用单机部署，单点故障！

# 微博

微博是指的各平台具体的一条微博信息，微博系统基本上有两个渠道获得微博：

1、通过讯库搜索

讯库搜索是由任务（舆情）、事件模块整理的关键字由爬虫爬取，然后再提供搜索接口给微博系统，然后搜下来的。

2、通过API调用

直接调用API获得各平台提供的微博数据。

## 数据结构

微博数据采用业务相关的公用设计原则，对于两个平台都有的属性进行合并，并且只取和分析业务相关的数据。

微博tid编码方案：

每个tid的首位都添加一个平台标识后保存。所有读出数据时第一位就是平台标识，后面是具体平台的微博id。

|  |  |  |
| --- | --- | --- |
| 字段名 | 描述 | 备注 |
| id | 自增长的ID | 无意义 |
| platform | 平台信息 | 腾讯：1  新浪：2  人民：3 |
| tid | 微博ID | 腾讯：info.id 微博唯一id  新浪：status.mid 微博mid  爬虫：网页上得  这两个id都可以参与URL的计算获得微博的地址  腾讯：1+info.id  新浪：2+status.mid  人民：3+rm.id |
| type | 微博类型 | 1-原创发表  2-转发  3-评论  腾讯：info.type 有7种类型，其中评论是7.其他的类型不做处理  新浪：无此字段 |
| text | 微博的内容 | 腾讯：info.text  新浪：status.text |
| created | 微博创建时间 | 腾讯：info.timestamp  新浪：status.created\_at |
| form | 微博的来源 | 腾讯：info.from+info.fromurl  新浪：status.source |
| uid | 微博的作者 | 腾讯：info.openid  新浪：status.user.id  新浪返回整个用户信息，返回的用户信息应该进入消息队列去更新用户表 |
| reposts | 转发数 | 腾讯：info.count  新浪：status.reposts\_count |
| comments | 评论数 | 腾讯：info.mcount  新浪：status.comments\_count |
| attitudes | 表态数 | 腾讯：无  新浪：status.attitudes\_count |
| source | 源微博 | 腾讯：info.type ==2时info.source为转发源  新浪：status.retweeted\_status有时为转发，没有时为原创  在自定义存储里这里保存的是tid,根据type和tid可以在当前表里面查到原微博 |
| contentid | 聚类标识 | 这个字段是线下计算的结果，目前为空，内容相似的微博的contentid是一样的，微博暂不支持聚类，因为文本太短了，目前无法聚类。 |
| collected | 收集的时间 | 入库时间，插入记录的时间 |
| Images | 微博关联的图片列表 | 存储一组图片URL |

# 我的微薄

我的微薄帐号是指在讯库微博系统中已经授权的微博帐号，一般是指官微和企业帐号。

我的微薄是指通过我的微薄帐号发布的所有微博信息，包括通过讯库微博系统发布的和其他方式发布的（其它方式包括使用帐号在其它应用中发布的，比如微博客户端）。

讯库微博系统对应的App是【讯库舆情】专门为这些官微使用，我的微薄帐号都给这个App授权。

授权过程参考【我的应用】。

## 我的帐号

所有微博帐号的数据结构按照博主的数据结构定义，所以我的微薄帐号也是如此定义。

我的微薄帐号有授权信息和授权过期时间以及当前的授权状态等信息，这些信息单独存储为微博帐号授权信息。

微博帐号授权信息表【My\_Account\_Auths】

|  |  |  |
| --- | --- | --- |
| ID |  |  |
| uid | 帐号id | 参考博主数据结构描述 |
| platform | 平台 |  |
| Token | 授权票据 |  |
| AppID | 给哪个App授权 |  |
| Expired | 何时过期 |  |
| password | 帐号密码 | 如果用户同意讯库的条款，并且希望达到自动授权的目的，则需要用户提供用户名帐号的密码，当授权将要过期的时候会自动去获取新的授权。  如果有密码则认为是允许自动授权的 |

新浪微博获得Token方法：

https://api.weibo.com/oauth2/authorize?client\_id=123050457758183&redirect\_uri=http://www.example.com/response&response\_type=code

同意授权后会重定向到http://www.example.com/response&code=CODE

这里的CODE就是获得Token的凭据。

调用https://api.weibo.com/oauth2/access\_token

代入参数，其中有代入的CODE来获得Access\_Token，到此获得了Token，则完成这个微博帐号对这个应用的授权。

腾讯微博获得Token方法：

参考http://wiki.open.t.qq.com/index.php/OAuth2.0%E9%89%B4%E6%9D%83

## 我的微薄

帐号管理方式为：企业管理员创建微博帐号，分配帐号给操作员。所以帐号是企业相关的。

【企业微博帐号表】用来保存某企业有哪些微博帐号

|  |  |  |
| --- | --- | --- |
| 字段 | 描述 | 备注 |
| ID |  |  |
| CustomID | 客户ID |  |
| tid | 微博帐号ID | 这个tid遵循微博博主tid的编码规则 |
| creator | 创建人ID | 这个微博是谁创建的 |

在创建我的微薄同时需要给这个帐号授权，授权信息保存在上面的My\_Account\_Auths表里，授权成功后，获得帐号的基本信息通过【微博博主】-【服务接口】-【更新接口】更新到微博博主数据库中。

由我的帐号发布的所有的微博信息，微博数据结构参考【微博】的【数据结构】，所有的微博结构都如上定义，所以我的微薄也是这么定义的。

我的微薄数据量分析：按我的微薄每天发布M条计算，三个平台，N个帐号则每天每客户产生M\*N\*3条微博数据。

M=（30-50），N=10（2-10），一个月30天发布微博的量是：100\*10\*30\*3=9万，一年大约100万。

我的微薄信息暂且保存在一个数据库的一个表里。表结构按微博结构保存。

【My\_Weibos】结构参考【微博】的【数据结构】。

和我相关的所有微博：

1、我评论的微博（要保存原始微博）

where uid=我 and type=3

2、我发布的微博

where uid =我 and type = 1

3、我转发的微博（要保存原始微博）

where uid=我 and type = 2

4、评论我的微博（要保存评论微博）

a.先找到source是我发的微博

b.过滤type=3

select \* from weibos a inner join weibos b on a.tid = b.source

where a.uid = 我 and where type = 3

5、@我的微博，

a. where text like '@我%'

b. where type=4????

c. 建立一个@我的关系表，保存<uid, tid(@我的微博ID)>的关系

每次有新微博入库，如果是@我的，则增加【我的微博表】追加【@我的关系表】。

我的微薄发布渠道有两种：

1、通过讯库微博系统发布

这种发布方式可以在讯库微博系统调用API发送成功之后记录发布时间和发布内容等信息。通过系统发布的微博如果发送成功则直接将微博内容注入到我的微薄列表里。

2、通过其他方式发布

这种发布的，可以通过爬虫或者API获得。获得之后将内容注入到微博列表里。

注入方式有两种：

1、统一注入方式

意思是，不论是哪种渠道发送的微博，最后都通过API获得微博信息，然后将信息注入到微博列表里。

2、直接入库的方式

意思是，当通过其他方式发布时使用API获得微博信息，然后注入列表。当通过微博系统发布时，如果通过API提交到新浪则认为是发布成功的，然后将微博信息直接注入微博列表中，这里省去一次API的调用。

新浪API：https://api.weibo.com/2/statuses/user\_timeline.json

最多返回最新的2000条，对于我的微博来说我觉得是够用了。

腾讯API：https://open.t.qq.com/api/statuses/broadcast\_timeline

这个貌似是没限制。

爬虫API：不支持。

**注入的过程不是一个插入的过程，注入微博列表是将微博信息插入微博列表之前，通过一个消息队列中间件来处理微博的内容，根据业务逻辑消息队列会调用对应的业务组件来处理微博信息，业务组件都处理完了以后（或以前）将微博信息插入到微博列表中。**

\*单表存储可能数据膨胀厉害，需考虑分库存储，如果按客户分库太碎，可以考虑使用分区表存储。

我的微博里保存了下面四种类别的微博：

别人发的微博，@我的

我发的微博

别人转发我的微博

我转发的微博

别人的评论，评论了我的微博

我发的评论，评论了别人的微博

## @到我的

所有提及到我的微博，微博存储结构不变，所有提到我的微博是别人@到我的，这里保存的是@我的微博的列表信息。

~~【My\_Mentions】~~

|  |  |  |
| --- | --- | --- |
| ~~字段名~~ | ~~描述~~ | ~~备注~~ |
| ~~ID~~ |  |  |
| ~~uid~~ | ~~我的微博ID~~ |  |
| ~~atuid~~ | ~~谁at我了~~ | ~~@我那个人的博主ID~~ |
| ~~...其他微博字段~~ |  |  |

只保存关系，微博的具体信息保存在my\_weibos表里。

新浪API：https://api.weibo.com/2/statuses/mentions.json

腾讯API：https://open.t.qq.com/api/statuses/mentions\_timeline

爬虫API：不支持。

~~这里保存的是具体的微博信息，日后分析使用。~~

## 我的评论

我收到和发出的评论，评论本身认为也是微博，所以评论的数据结构同微博，类型为3。

我的评论数据保存在【My\_weibos】里面。

新浪API：https://api.weibo.com/2/comments/timeline.json

腾讯API：https://open.t.qq.com/api/statuses/user\_timeline

根据类型type来获得我的评论。

引用：拉取类型（需填写十进制数字）0x1：原创发表，0x2：转载，0x8：回复，0x10：空回，0x20：提及，0x40：评论。如需拉取多个类型请使用|，如(0x1|0x2)得到3，则type=3即可，填零表示拉取所有类型。

爬虫API：不支持

## 发布管理

微博未提交到API之前由【发布管理】管理。提交到API之后由【我的微博】管理。

提交

要审？

审核通过

发送

结束

### 提交

提交对应系统中的【定时发送】和【立即发送】，讯库微博系统的所有微博提交均通过统一的接口提交，接口描述如下：

void submit(time, text, accounts, images)

参数说明：

time：何时发布，当前提交的微博计划何时发布？如果是立刻发布则这里传0，否则传时间的long值。

text：发布的内容，如果有表情则按照规则[表情]插入到内容中，如果是话题则按#话题内容#插入到内容中。

accounts：用哪些微博帐号提交微博内容，是一个微博帐号的数组。帐号为系统绑定的官微帐号，并且已经授权给Normal的应用，发布的时候都通过这个应用发布微博。

images：图片，和这个微博关联的图片路径，当用API提交的时候将使用这些图片，这是一个图片数组。

提交结果放到到待发送表里，【待发送表】定义如下：

|  |  |  |
| --- | --- | --- |
| 字段名 | 描述 | 备注 |
| ID |  |  |
| UserID | 谁提交的 | 这个帐号和一个微博帐号绑定了，如果帐号未绑定则不能发送。 |
| uid | 用哪个微博发送 |  |
| Text | 提交的内容 |  |
| submit | 提交的时间 |  |
| sent | 发送时间 | 如果是0则立即发送，否则是按照指定时间的long格式发送 |
| images | 图片列表 | 直接保存的是图片路径的集合，不做子表直接保存在这个字段里，用|分割所有的图片，如果需要修改上传的图片内容，正确的流程是：先按照列表Load图片给用户，当用户修改完成之后再把修改字段直接Update这个字段即可 |
| Approved | 审核状态 | 0、不需要审核，可直接发送的  1、需要审核，待审核的，不能发送  2、需要审核，审核通过的，可以发送的  3、需要审核，审核失败的，不能发送的 |
| Auditor | 审核人 | 如果需要审核，该单据是被谁审核的，默认为空 |
| Send | 是否已经发布 |  |

### 审核

具有审核权限的用户可以审核本企业内部提交的微博。审核分配在【系统管理】里面。

审核微博服务描述如下：

void AuditWeibo(id, approved)

参数说明：

id为待发送列表中的id，标识要审核哪条微博。

approved为审核状态，审核状态为枚举类型，参考【待发送列表】的approved字段。

不论审核状态如何，调用完该函数，【待发送列表】的审核人将变成当前的操作员。

如果审核通过了则把需要立即发布的微博【发送】出去。

定时发送的则进入【发送队列】等待发送。

### 发送

调用该接口可以发送微博到指定的平台

void send(weibo)

参数说明：

weibo：为【待发送列表】结构。

首先通过accountid检查绑定的帐号是属于哪个平台，然后调用指定平台的API将微博内容发布出去。

## 数据分析

针对官微的一些数据分析。

### 发布统计

发布统计是统计我的微薄帐号发布情况的统计信息。

存储：发布统计表

|  |  |  |
| --- | --- | --- |
| ID |  |  |
| Sampling | 采样时间 | 采样时间应该控制到每小时一次，这样该表既可以满足按小时统计也可以满足按天统计 |
| uid | 哪个微博号 |  |
| platform | 是哪个平台发布的 | 如果该统计信息保存在K-V里，则platform和uid是key |
| weibos | 当前的微博数 | 记录的是每小时的信息，所以如果要计算上一个小时发了多少微博就是这个小时的微博减去上个小时的就是过去一小时发布微博的数量 |
| followers | 当前的粉丝数 |  |
| friends | 当前的关注数 |  |

针对每用户每平台每天生成记录24条（每小时生成一条），对于N个用户3个平台每天生成24\*N\*3=72\*N，如果N为1000，则每天产生72000条记录，一个月是2160000。

\*如果数据量太大，该表可能太大导致性能问题。

发布统计逻辑如下：采样时调用API获得当前用户的基本信息，其中包括发微博数、粉丝数、关注数，这些信息更新的到上面的表里面。获取数据时只获取weibos的数据即可，另外两个字段followers和friends是业务扩展字段，以后可能会用。

新浪API：https://api.weibo.com/2/users/show.json

腾讯API：https://open.t.qq.com/api/user/info

爬虫API：只需要获得三个数字即可，这三个数字在页面上是可以抓到的。

### @我统计

由于调用API都获得的是当时@我的微薄信息，所以需要存储历史的记录来作为统计数据。

|  |  |  |
| --- | --- | --- |
| 字段名 | 描述 | 备注 |
| ID |  |  |
| uid | 微博帐号ID |  |
| platform | 平台ID | Uid和platform为组合关键字 |
| Sampling | 采样时间 | 按照需求采样时间最小单位为每天采样一次。 |
| Value | 采样时间点@我的人有多少 | 只记录数据，不记录谁@我了  腾讯：data.totalnum  data.timestamp为服务器时间戳  新浪：result.total\_number |
| lastid | 最后一条微博的ID | 上次取到的最后一条id，再取的时候需要穿这个值过去，更新之后需要把这个值更新成新的lastid  腾讯：lastid  新浪：since\_id |

新浪API：https://api.weibo.com/2/statuses/mentions.json

腾讯API：https://open.t.qq.com/api/statuses/mentions\_timeline

### 趋势分析

该帐号的趋势是指，过去一段时间，每个时间点（每小时）该帐号转发和评论了多少次。

需要支持查询，支持指定时间范围的趋势查询。

趋势分析可以通过【我的微博】来统计，统计规则为：首先【我的微博】是有转发和评论类型的，可以根据该类型筛选我转发了多少，评论了多少。同时这些转发和评论上都有发布时间，根据发布时间找到最小时间和最大时间，然后再中间时间每个有数据的点标记出来，然后将所有标记出来的点连线就是全部时间的趋势图。

## 粉丝分析

粉丝分析需要知道我的粉丝数量以及全量的粉丝列表才能分析，目前只有商业API才可以获得全量粉丝列表，下面的分析是针对普通API设计的如何获得尽量全的粉丝列表。

商业API：

可获得全量粉丝列表，一切都不是问题了。。。

如果用户同意支付商业API的费用，可以使用商业API。

普通API：

新浪：https://api.weibo.com/2/friendships/followers.json

最多返回5000个粉丝。

腾讯：https://open.t.qq.com/api/friends/user\_fanslist

mode参数为0是拉取新粉丝，只拉取1000个新粉丝。

mode参数为1是拉取10000个粉丝，未排序的。

所有的粉丝也是一个微博博主，存储结构参考【微博博主】，粉丝的数据可以通过【博主服务】获得。

每次调用API获得粉丝列表后，都要调用博主服务的update方法来更新粉丝的基本信息。

**下列的分析均基于粉丝采样数据！**

### 认证比例

我的粉丝里面有多少认证的粉丝？

getvip -参考服务接口

### 性别比例

我的粉丝里面有多少女粉丝？

微博博主服务接口提供！

### 粉丝人气

我的这些粉丝里面他们的粉丝数都是多少？这里要的是一个范围？下面的粉丝最多用户要的是粉丝数的TOP-N。

微博博主服务接口提供！

### 用户热门标签

我的粉丝里所有的用户标签里最多的几个标签是什么？

首先获得我的所有粉丝的列表fans。

其次通过列表得到所有用户的标签tags。

对标签进行聚合，按照聚合数排序显示。

### 地域分布

我的粉丝都分布在哪？

微博博主服务接口提供！

### 粉丝最多用户

我的粉丝里哪些粉丝的粉丝数最多？

微博博主服务接口提供！

### 粉丝互动排行

我和我的粉丝互动次数最多的列表？

首先通过@我的【My\_mention】表查询出来所有@我的博主（字段为atuid），然后做分组得到atuid的次数信息[{uid:123,count:100},{uid:234,count:200}]，表示uid123用户@我了100次，uid234的用户@我了200次。

接着将这个列表里的uid取出来，去我的粉丝表里面过滤，得到我的粉丝集合。

最后按count排序即可。

# 微博任务（事件）系统

任务和事件的关系：任务是一个持久的监测项，事件是一个突发检测项。二者无本质区别，所以从设计上来说，这两种应用属于同一个设计范畴。

本质上任务和事件都是对关键词的管理以及围绕关键词相关内容的对象。它们都对应一组关键词，所以事件和任务其实就是**关键词包**。

下文提到的任务如无特别说明均指任务和事件。

## 数据结构

数据结构是否可自定义？

a.如果可自定义，系统复杂度将大大提升，并且在异构的两个数据集上查询也可能带来困难，结构变更则是自动的，查询需要适配不同结构。

b.不可自定义，系统定义一个结构，然后所有的数据均按照这个结构存储，如果需要发生结构变化，则需要一个变更方案来为所有的已有数据做结构变更。

综上所述使用b方案，微博数据结构由系统定义，结构定义参考【微博-数据结构】，数据表名定义为Task\_data。

数据注入：指将微博数据注入到Task\_DATA表里面的过程。这个过程是需要判断是否重复的，如果Task\_data里面有这条微博则更新相关的信息，否则插入新的微博信息。

如果已经有了，更新的策略是如果转发数（评论数）大于原始转发数则更新转发数，否则不更新。

**所有的分析都是平台相关的，所以所有的业务数据里都包含一个Platform字段，未写出的默认就添加上。Platform的意义和Task\_data一致。**

业务数据服务

元数据描述

数据模型

基础数据访问

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

Data

## 微博舆情分析

微博舆情是任务舆情，也是任务系统的一部分，任务由系统管理的任务管理创建并提交。普通的操作员进入系统只是查看任务的数据，不能修改任务的内容。所以任务是一个相对稳定的关键词包。

### 舆情分组

每个客户可以创建一组自己的舆情项，这些舆情项可以分组显示，目前每个客户最多可以创建层级为2个分组，第二组分组下显示的是具体的任务项集合。

分组表为标准的树形设计结构，【任务分组表】如下：

|  |  |  |
| --- | --- | --- |
| 字段 | 描述 | 备注 |
| ID | 分组ID |  |
| Name | 分组名称 |  |
| PID | 改组的父节点 | 理论上这是一个无层级限制的分组结构，实现2层的结构则需要由程序来控制。 |
| CustomID | 客户ID | 标识这个分组是哪个客户的 |

舆情描述保存在【舆情描述表】中，如下图所示：

|  |  |  |
| --- | --- | --- |
| ID | 任务唯一标识 |  |
| GroupID | 任务分组ID |  |
| Name | 任务名称 |  |
| Keywords | 任务包含的关键字集合 | 每个关键词之间用,分开 |
| Creator | 创建者 | 这个任务是谁创建的，这应该是一个客户的管理员 |
| CustomID | 客户编号 | 这是一个冗余字段，用来记录但钱的任务属于哪个客户，实际上客户信息可以从Creator查到 |
| CreateTime | 创建时间 | 这个任务的创建时间，也是记录插入的时间 |
| LastUpdate | 最后一次更新时间 | 这个任务的修改时间，无论修改什么内容都需要更新这个字段，来记录任务描述信息最后一次是什么时候更新的 |
| StoreAddress | 任务数据存储地址 | 这是一个描述符，用来描述当前的任务数据保存在什么地方？描述符结构如下定义。 |

【任务存储描述符】是指任务当前任务的具体数据保存在什么地方的描述信息，描述信息如下：

{

type:存储类型/SQL/Mongdb/bdb/或者其他什么DB,

connectionInfo:连接信息，如果是SQL是连接串，如果是其他数据库则是链接地址和端口,

mainDB:主数据库,

task\_data:主表,

info\_data:信息描述表

}

这个字段是一个非常重要的字段，一旦确定不能再修改，并且在UI上也没有地方可以修改。

为了保证建立的数据存储可以被双向找到，也就是说通过task可以找到存储，通过存储也可以找到Task，需要在存储结构上有一个info\_data表，该表用来描述这个存储的任务基本信息。如此即使StoreAddress错误了或者不存在也可以反相查找到。

任务调用存储的方案为：

1、如果StoreAddress有值则获取具体的读取信息（连接信息），尝试连接存储体

2、连接到存储体之后然后验证info\_data里面的数据描述和本任务是否一致？

3、如果一致则可以进行操作，否则不能操作。

存储微博基本信息，主表结（task\_data）构同：【微博】的【数据结构】

## 微博舆情业务

### 过滤

时间过滤：where created between startTime and endTime

转发/原创过滤：

where type =1 --原创

where type=2 --转发

平台过滤

where platform =1 --腾讯

where platform =2 --新浪

where platform =3 --人民

### 排序

时间排序（默认）：order by created

相似排序：order by contentid

转发排序：order by reposts

评论排序：order by comments

### 相似列表

暂不实现。

### 舆情收集

选择微博进行后续处理，选择的微博保存到微博选取数据表中，表结构如下：

【微博信息收集表】 Task\_Collections

|  |  |  |
| --- | --- | --- |
| 字段名 | 描述 | 备注 |
| ID | 主键 | 无意义 |
| TaskID | 任务ID |  |
| tid | 微博ID | 可以到Task对应的存储里找到 |
| Status | 处理状态 | 1、待处理  2、已处理  3、已忽略  4、已放弃 |
| Platform | 平台类型 | 关联时使用 |
| GroupID | 舆情分组ID | 标示这条舆情在收集的时候放在哪个分组里，收集分组（分类）ID |

【收集分类信息表】 Task\_Collection\_Groups：

|  |  |  |
| --- | --- | --- |
| 字段 | 描述 | 备注 |
| ID | 分组ID |  |
| Name | 分类名称 |  |
|  |  |  |

处理时的转发和评论则调用平台的API操作。

~~微博详情为相似详情。打开相似，操作同上。~~

### 舆情处理

舆情处理是处理舆情收集中的舆情，是操作自己收集的舆情所以不涉及权限。

操作的数据还是上面的【微博信息收集表】。

转发：

1、获得当前微博的所属的平台

2、获得当前用户绑定的微博平台

3、如果有对应平台则可以转发并且提示选择多个帐号转发，如果没有则提示不能转发

评论：

1、获得当前微博的所属的平台

2、获得当前用户绑定的微博平台

3、如果有对应平台则可以转发并且提示选择多个帐号评论，如果没有则提示不能评论

忽略：

就是不想要的微博，将status直接设置为3即可。

### 微博详情

暂不实现！

## 事件分析

是一个事件列表，展示的是每个事件中包含的微博具体数量。

【事件表】

|  |  |  |
| --- | --- | --- |
| ID |  |  |
| Name | 事件名称 |  |
| startTime | 开始监测时间 |  |
| endTime | 结束监测时间 | 开始时间和结束时间的时间间隔最多为1个月（30天） |
| Keywords | 包含关键字集合 | 用,分割 |
| Notkeywords | 不包含关键字集合 | 用,分割 |
| location | 地点 | 发布人的注册地点，这个值是个过滤条件。 |
| StoreAddress | 存储位置描述符 | 该描述符意义同微博舆情 |

事件分析存储同【微博舆情】

## 事件分析业务

### 趋势分析

任务趋势表：记录任务每天或者每小时产生微博数量的采样数据表，用来跟踪任务里微博数量变化。

微博表里保存有created字段用来记录微博创建的时间。也可以通过该字段来统计。

1、查询Task\_data表获得所有的微博信息

2、根据微博的created字段来将所有的微博分组

3、按照分组显示数量即可

### 直发转发分析

微博表里保存有type字段用来标识该微博是否是转发。

查询Task\_data表获得type=2的记录总数就是转发数，其它类型也可查询。

type列为索引列。

### 内容关键词图

聚合任务的所有微博，抽取关键词，显示前25个关键字。这里内容是计算得来的，数据库只保存一个计算的快照。

|  |  |  |
| --- | --- | --- |
| 字段名 | 描述 | 备注 |
| ID | 自增长 |  |
| Key | 关键词 |  |
| Count | 关键词出现的次数 |  |
| Lastupdated | 最后一次更新时间 | 这个时间可以做Cache同步时使用和作为JOB更新策略的条件之一 |

此数据属于线下计算数据。

### 关键观点

获得当前任务里面转发数最多的几个微博。

SELECT \* FROM TASK\_DATA ORDER BY REPOSTS

如果数据量特别大，做不了ORDER BY，则使用关键观点表来解决

1、首次注入数据时则生成一个根据Reposts排序的结果集（TOP-100）DATA。

2、下次注入时，则将注入的这批数据里面所有Reposts数大于DATA里面Reposts最小那一条的tid按顺序插入到DATA里面，把100外的删除即可。

每次直接读取DATA表即可。

|  |  |  |
| --- | --- | --- |
| 字段名 | 描述 | 备注 |
| ID |  |  |
| tid | 微博ID |  |
| reposts | 转发数 | 这个字段是个排序字段，该表里该字段是排好序的 |

### 性别分布

1、查询Task\_data表获得所有的微博信息

2、通过微博信息获得所有的微博博主信息（排重）

3、通过博主列表调用博主服务接口获得有多少男女博主。

### 用户类型

同性别分析

### 来源分布

1、查询Task\_data获得所有微博信息。

2、获得所有的博主信息（排重）

3、调用博主服务接口获得这些人的地域分布信息。

### 注册时间分布

1、查询Task\_data获得所有微博信息。

2、获得博主信息。

3、按注册时间分组，分组规则为（半年以内、半年到一年，一年到两年，两年以上的）

实现的时候需要建立一个枚举值【时间分布枚举】。

### 地域分布

地域分布是指发微博的这个人注册时候使用的地域，不是微博在哪发的。

地域信息项包括：省（**province**）、城市（city）、所在地（location）

1、首先获得当前任务所有的博主信息。

2、根据博主信息去【博主服务接口】获得这些博主的地域分布列表。

分布列表是：

[{

province:1,

city:[code:1,val:1000]

}]

省份1的城市1有1000个粉丝。

### 关键用户分析

按照博主的粉丝数取TOP-N

假如任务里产生了N条微博，这N条微博是由M个博主发的（M<=N），博主按粉丝数排序后取得TOP-N个博主。

关键用户表

|  |  |  |
| --- | --- | --- |
| 字段名 | 描述 | 备注 |
| ID |  |  |
| uid | 用户ID |  |
| Follows | 粉丝数 |  |

首次注入获得任务里面所有的博主信息，获得粉丝数最大的TOP1000的用户插入关键用户表里。

每次注入时，则获得博主信息，博主的粉丝数大于这1000个用户的插入到对应的位置将粉丝量最小的淘汰了，保证此表一直只有1000条记录。

当需要获得关键用户时则直接读取这个表的TOP-N即可。

## 业务（事件）消息队列

JOB运行任务，每当任务产生新的微博时，将微博信息插入任务数据主表，同时将数据插入任务消息队列，队列负责将数据做相应的业务处理。

该消息队列可以理解为一个消息管道，所有的微博通过该消息管道时将会触发管道上的业务事件处理，做相应的业务处理。

固定业务逻辑有：更新博主信息。

博主信息是保存在redis数据库缓存在内存中的，如果有博主信息发生变化则需要跟新redis同时将更新博主基本信息的持久化存储，则将这个更新业务添加到队列中，由队列慢慢提交完成。

目前关键用户分析就使用该队列。

# 帐号监测

帐号监测方法：讯库系统首先应该有很多自己的帐号，参考【我的应用】-->【影子帐号】，这些帐号关注需要监测的帐号。

监测一个微博帐号的基本情况的变化，微博帐号基本信息的存储结构参考微博博主数据结构。【监测帐号表】记录所有待监测的帐号信息。

|  |  |  |
| --- | --- | --- |
| 字段名 | 描述 | 备注 |
| ID |  |  |
| UserID | 哪个用户要监测 | 这是讯库微博系统的用户ID，根据该ID可以区分哪些用户监测了哪些微博帐号 |
| uid | 要监测的帐号uid | 这是平台的微博帐号 |
| platform | 平台信息 | 平台信息 |
| status | 监测状态 | 1、停止  2、暂停  3、运行  4、已删除 |
| Original | 原创率 | 基于API的采样数据  这里只存数字，不存采样数据 |
|  |  |  |
| ~~fans~~ | ~~粉丝数~~ | ~~基于API~~ |
| ~~statuses~~ | ~~微博数~~ | ~~基于API~~ |
| 平均每日转发 |  | 暂不支持 |
| 平均每日评论 |  | 暂不支持 |

如果要显示监测帐号列表，则通过该表直接通过UserID过滤获得即可。

当API调用次数过多时，列表中的微博数、粉丝数通过【博主服务接口】获得。

微博原创率是基于采样数据。

动态检查用户是否存在需要调用API，讯库可以提供一个讯库的微博博主提示内容。

**平均每日转发、平均每日评论**

## 实时监测

首先应该有个【影子帐号】关注被关注的帐号。

帐号监测表

|  |  |  |
| --- | --- | --- |
| 字段名 | 描述 | 备注 |
| ID |  |  |
| uid | 监测帐号的ID |  |
| owner | 监测所属人 | 可以理解为创建人，是谁打算监测这个帐号 |

两种监测方法：

1、基于采样数据的监测

采样详细信息，数据不全面，但是计算简单，爬虫压力大，需要得到微博列表、转发列表、评论列表，太难无法实现。

采样数字，只采样被监测用户每天发微博的数量。转发和评论无法得到。

2、基于API的数据监测

新增微博数：可通过API获得。

新增粉丝数：可通过API获得。

新增转发数：商业API提供。

新增回复数：商业API提供。

商业API提供的当未购买商业API时，次数据为0，无从获得。

【用户实时分析表】 account\_realtime

|  |  |  |
| --- | --- | --- |
| 字段 | 描述 | 备注 |
| ID | 无意义 |  |
| uid | 博主ID |  |
| followers | 粉丝数 |  |
| Weibos | 发微博的数 |  |
| Friends | 关注数 |  |
| Reposts | 总转发数 | 商业API提供 |
| Comments | 总回复数 | 商业API提供 |
| Sampling | 采样时间 | 采样时间为每小时一次，所以采样时间是一个连续的时间，由于取数据的方式不同，这里要尽量保证是每个小时末采样数据。 |

### 数据分布图

维度

微博数：{time:采样时间,weibos:数量}

微博数列表需要采样数据，再点击详细信息时需要穿透显示。

下面三个维度不支持穿透，因为无法得到关注时间，没有评论列表和转发列表。

粉丝数：{time:采样时间,followers:数量}

评论数：{time:采样时间,comments:数量}

转发数：{time:采样时间,reposts:数量}

根据上述字段可获得数组，根据time和[维度]可画图。

## 趋势分析

趋势分析是分析转发和评论的，所以基于【用户实时分析表】可得到具体时间点的具体转发和评论数，不支持穿透。

## 粉丝分析

新浪API：https://api.weibo.com/2/friendships/followers.json

腾讯API：https://open.t.qq.com/api/friends/user\_fanslist

爬虫API：无

粉丝分析基于采样数据分析，所以粉丝分析是基于博主信息库分析的，博主信息库中应该有标识哪些博主是需要监测的？需要监测的应该给予更大的刷新机会和频率。因此粉丝分析的所有逻辑均按照【事件分析业务】来分析。

流程如下：

1、首次监测帐号时如果博主库里没有，则新建一条，同时更新消息队列。

2、如果有，则更新monitor字段，并且调用API更新帐号信息。

3、通过API获得采样数据更新博主的粉丝列表【粉丝关系库】，更新消息队列。

4、如果API超过调用次数，则跳过3直接显示历史数据。更新消息队列（消息队列记录需要更新这个博主的信息，但是由于没有API调用需要延迟调用，如果有同样的请求过来，则后者覆盖前者，当有API调用的时候则队列里对于一个博主最多只有一次API的调用）。

## 博文分析

新浪API：https://api.weibo.com/2/statuses/user\_timeline.json

最多返回2000条

腾讯API：https://open.t.qq.com/api/statuses/user\_timeline

爬虫API：不支持

【用户博文表】Account\_Weibos，该表结构同【微博数据结构】，基于采样数据的分析。

发布时间分布：按照Account\_weibos的created过滤即可

@提到的：按照Account\_weibo的text过滤@即可？？？

转发对象：按照Account\_weibo的text过滤//即可？？？

热门微博：按照Account\_weibo的reposts+comments过滤即可？？？

热门微博可代入【传播分析】（微博分析）界面。

新浪API：批量获取指定微博的转发数评论数

http://open.weibo.com/wiki/2/statuses/count

腾讯API：转播数或点评数

https://open.t.qq.com/api/t/re\_count

跟新【用户博文表】

## 微博详情

显示微博信息列表，实时界面穿透过来的。显示监测用户发布的微博采样数据。

# 传播分析

传播分析就是微博分析，可分析的微博有：新浪、腾讯、人民。新浪和腾讯用过API可以是实时的，人民是非实时的（系统有个预估时间，队列等待时间）。

新浪专门的接口目前只支持新浪的服务。所有的传播分析只能分析原创，不能分析转发和评论（也许以后可以）。

## 分析列表

保存用户最近几次的分析列表，显示微博的基本信息。

【微博分析列表】

|  |  |  |
| --- | --- | --- |
| 字段 | 名称 | 备注 |
| tid | 微博编号 | 这个微博编号的第一位是标识微博的平台，后面几位是实际平台对应的微博编号，调用API可以通过这个获得 |
| UserID | 谁要分析这个微博 |  |
| Url | 分析微博的地址 |  |
| ~~PublishTime~~ | ~~发布时间~~ |  |
| uid | 发微博的人 | 这个人应该在博主列表里，如果没有则先添加到博主列表里 |
| Text | 微博的内容 | 发布微博的内容 |
| reposts | 转发数 |  |
| comments | 评论数 |  |
| monitorTime | 监控时间 | 是一个Long类型的 |
| Platform | 平台信息 | 这个字段恶意从URL里面获得，冗余设计 |

## 趋势分析

分析转发和评论的趋势。全部基于采样数据分析。

新浪API：

转发：http://open.weibo.com/wiki/2/statuses/repost\_timeline 获得指定微博的转发列表

最多返回2000条。

评论：http://open.weibo.com/wiki/2/comments/show 获得指定微博的评论列表

最多返回2000条。

腾讯API：

转发：https://open.t.qq.com/api/t/re\_list

评论：https://open.t.qq.com/api/t/re\_list 获得指定微博的转发或者评论列表，根据flag来切换。flag=0转播列表。flag=1评论列表。2转播和评论的全拿。

爬虫API：无

【微博转发评论表】数据结构同【微博数据结构】

|  |  |  |
| --- | --- | --- |
| 字段 | 描述 | 备注 |
| tid | 微博id | 描述同上 |
| ... |  |  |

获得转发：where type =2

获得评论：where type =3

表里面的数据来源为用户输入了分析，然后点击分析后产生，过程如下：

1、用户通过输入或者其他方式定向到传播分析页面，载入URL。

2、根据URL判断是要分析哪个平台的微博。

3、如果是新浪或者腾讯则调用相应的API获得最近的2000条转发和评论。

4、更新微博转发评论表

4.1 如果API里面的tid在库里有数据则更新信息，否则就插入

4.2 更新过程是在消息队列里完成

5、根据上面的条件来筛选

## 层级分析

层级分析主要是分析微博的转发层级，首先要得到微博的所有转发微博，由于接口限制只能获得2000的转发微博。这些微博保存在上面的【微博转发评论】表里。

分析层级流程如下：

1、首先获得所有的转发微博，where type =2

2、取出所有的微博的text部分

3、计算有多少个//，每个//代表一层

找出//最多的微博就是该微博转发的最大层数。

带来转发数：是通过分析//后面一样的有多少，//后面一样的说明是他带来的转发，然后计算数量。

~~按粉丝排序，是通过查看该用户有多少粉丝来排序的，粉丝数可以通过【博主服务接口】获得~~。

~~按时间排序~~

## 关键用户分析

关键用户有两种：

1、转发者中粉丝最多的用户称之为关键用户。

2、转发者中转发最多的用户称之为关键用户。

已传播是指被转发人的粉丝转发过的转发微博，按照粉丝和转发过滤和排序。

未传播是指未被转发人的粉丝转发过的转发微博，按照认证用户筛选，粉丝数排序。

实现流程：

1、先得到转发列表，然后得到所有的转发人。

2、这些转发人和转发人直接是否有粉丝关系？如果有则说明有传播，如果没有则说明没传播。得到一个传播列表A和未传播列表B。

3、A列表所有的用户按粉丝排或者按二次转发排序（二次转发数计算方法往下看）。

4、B列表的按照粉丝数排序，认证过滤即可。

二次转发数计算方法：

有转发用户列表A，有转发用户a，如何找到a的二次转发数？

1、现在所有的转发里面找到//a的微博。

2、再数//a在第二层的有多少就是a的二次转发数。

3、遍历转发用户列表得到每个用户的二次转发数，然后排序。

## 转发用户分析

由于微博转发和评论是基于采样数据，所以转发用户也只能基于采样数据。

实现流程如下：

1、找到所有的转发微博对应的博主。这个在调用API的时候已经得到过了，并且已经记录到redis里面同时写到消息队列里面更新本地存储。

2、找到博主后（目前也最多就2000个转发博主），拿这2000个博主去调用【微博博主服务接口】获得TOP-N排行以及其他的分析数据。

如果是全量转发，则可以分析出来全量粉丝以及全量的TOP-N。

# 协同办公

## 任务通知

任务通知是消息通知的一种方式。添加一个任务通知，按优先级分别发送给指定人员。

点删除按钮按消息ID删除消息。

消息表(Office\_Messages)

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段描述 | 备注 |
| Id | 自增长列ID | 无实际含义 |
| Type | 消息类型 | 普通消息：0  预警消息：1  微博消息：2 |
| Message | 消息内容 |  |
| Rank | 消息优先级 | 低：0  中：1  高：2 |
| CreatedTime | 创建时间 | 发送时间 |

消息发送表(Office\_MessageReceivers)

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段描述 | 备注 |
| ID | 自增长ID | 无实际含义 |
| MsgID | 消息编号 | 关联消息表 |
| Receiver | 接收人 | 接收消息的人 |
| IsHandle | 是否处理 | 该状态的意义是根据Message的Type状态决定的 |
| Status | 消息状态 | 未读：0  已读：1 |

## 考核管理

### 微博内容统计

对每个微博账号发布的微博受关注程度进行统计，显示每条微博各个机构转发的次数。

1. 按照时间范围和账号进行搜索显示
2. 显示统计列表，列表内容包括：微博内容，发布时间，该微博转发数，该微博评论数和转发的机构数（该微博被该机构转发了多少条） #需要提供微博列表和机构列表

#select \*from 微博表 where a.uid= AccountID where b.accountID=? and between ?and ?

select count(ID) from微博表 a join 机构表 b on a.uid=b. AccountID where b.accountID=? and between ?and ? #转发的机构数

select \* from 微博表 a join 微博表 b on a.id=b.source where a.uid=? and type=2

1. 通过点击转发数来查看微博转发的详细列表，点击评论数查看微博评论的详细列表和点击转发机构查看微博转发的详细列表。

转发详细列表：select \* from 机构 a join (select \* from 微博表 a join 微博表 b on a.id=b.source where a.uid=? and type=2

) b on a.account=b.uid

### 微博发布统计

对客户下的系统用户提交的微博，及采纳情况的统计。#提交就是发布微博，采纳是发布的微博被审核通过。

1. 通过时间范围搜索来获取提交统计和采纳统计。并导出word，excel，PDF等
2. 通过点击提交数查看提交的微博列表和采纳的微博列表。

### 微博处理统计

对客户下的系统用户一共处理的微博数进行统计，微博处理包括转发和评论。

1. 按照时间范围和微博账号进行搜索显示，并导出word，excel，PDF等
2. 通过点击转发和评论查看其详细列表

### 网评员统计

对客户下的网评员进行统计。通过当前登录的操作员获取当前的客户，用客户编号和时间段来搜索网评员的信息的统计列表，列表内容包括：网评员、网评员的微博账号的个数和总转评到我的个数（转发和评论我的微博之和），并导出word，excel，PDF文件。此外还有添加网评员和删除网评员。

网评员统计(Office\_Navies$)

|  |  |  |
| --- | --- | --- |
| 字段 | 字段描述 | 备注 |
| ID | 自增长列id | 无意义 |
| Name | 网评员名称 | 也叫水军 |
| Platform | 微博平台 | 新浪微博：0  腾讯微博：1  人民微博：2 |
| Account | 微博账号 | 微博账号 |
| IsEnable | 是否可用 | 默认都可用，如果不可用可以将此值设为0 |
| CustomID | 客户编号 | 客户编号 |

## 机构管理

1. 对下属组织机构的微博账号的添加及查看。添加按微博账号分别添加到不同的公共平台，查看分别按 “新浪”，“腾讯”，“人民”平台中的组织机构微博账号信息显示。搜索按日期范围来搜在此范围内发微博的数量和@我的微博的数量，然后按照搜索结果导出。

|  |  |  |
| --- | --- | --- |
| 字段 | 字段描述 | 备注 |
| ID | 自增长列id | 无意义 |
| CustomID | 客户编号 | 关联客户 |
| Name | 机构平台名称 | 微博账号昵称 |
| Platform | 微博平台 | 新浪微博：0  腾讯微博：1  人民微博：2 |
| AccountID | 微博账号 | 微博账号 |
| IsEnable | 是否可用 | 默认都可用，如果不可用可以将此值设为0 |

# 推送服务

对推送任务进行新建和管理。

1. 通过当前登录的操作员显示推送任务列表，列表内容包括：序号、推送名称、类型（日常监测和事件监测）、状态、删除。
2. 点击“停止推送”，该任务将会停止发送，按钮状态变为“开始发送”。
3. 点击“删除”，删除该推送任务。

* 点击“新建”，弹出新建导航界面，填写任务名称，任务类型（日常监测，事件监测）选择日常监测，下边控件刷新。添加任务（列表内容为已经设置好的舆情展示中的监测任务）
* 选择事件监测，控件刷新，添加任务，弹出任务列表，勾选要进行发送的任务。

1. 从通讯录中选择推送发送的人员。如果没有该人员就直接输入Email。
2. 选择推送的方式，包括：邮件。
3. 对推送任务的周期进行设置，填写任务开始时间。
4. 点击“完成”完成推送任务的设置，返回列表。

|  |  |  |
| --- | --- | --- |
| 字段 | 字段描述 | 备注 |
| ID | 自增长列id | 无意义 |
| Name | 推送任务名称 | 推送任务名称 |
| Type | 推送类型 | 0：日常监测  1：事件监测 |
| UserID | 用户编号 | 当前登录的用户 |
| IsSendMail | 是否发送邮件 | 是：true；否：false |
| FirstRunTime | 首次执行的时间 | 首次执行的时间 |
| LastExecuteTime | 最后一次执行的时间 | 最后一次执行的时间 |
| TopN | 每栏目发送显示的条数 | 每次最多多少条 |
| PeriodType | 周期类型 | 0：分钟、1：小时、2：天  3：周、4：月、5：年 |
| PeriodCount | 周期数 | 周期数 |
| IsStop | 是否停止 | 是：true；否：false |

# 系统设置

## 账号设置

对新浪、腾讯、人民网微博账号进行绑定。账号授权流程：通过APP应用获取code，然后用code通过回调获取token，把token保存下来授权成功。

1. 显示已绑定的账号列表，列表包括序号、微博、微博登陆账号、微博名称、微博状态（是否有效）、剩余天数（从现在到截止日期的相隔天数）、延长授权、自动授权和删除；
2. 延长授权：点击延长授权，弹出新浪微博绑定界面，进行重新授权。
3. 自动授权：点击自动授权，填写账号密码，在授权到期后使用客户微博账号自动授权。
4. 删除：点击删除，取消绑定并将该账号绑定信息删除。
5. 添加：点击按钮弹出界面，选择微博平台，根据所选平台，进入到相应平台的授权界面，进行授权，授权后返回列表。

#微博帐号授权信息表

|  |  |  |
| --- | --- | --- |
| ID |  |  |
| uid | 帐号id | 参考博主数据结构描述 |
| platform | 平台 |  |
| Token | 授权票据 |  |
| AppKey | 给哪个App授权 |  |
| Expired | 何时过期 |  |
| password | 帐号密码 | 如果用户同意讯库的条款，并且希望达到自动授权的目的，则需要用户提供用户名帐号的密码，当授权将要过期的时候会自动去获取新的授权。 |

## 用户维护

通过添加、删除、修改操作对客户下的用户进行维护。

1. 显示用户列表，字段包括：姓名，邮箱，电话，登录账号，角色（关联角色表）。
2. 支持增加，修改，删除；点击增加按钮，进行员工的添加，角色下拉列表从角色列表中获取。

用户表

|  |  |  |
| --- | --- | --- |
| 字段 | 字段描述 | 备注 |
| ID | 自增长列id | 无意义 |
| CustomID | 客户编号 | 关联客户 |
| UserName | 用户名称 | 用户名称 |
| NickName | 昵称 | 昵称 |
| Token | 用户密码的票据 | 用户密码的票据 |
| Email | 邮箱 | 邮箱 |
| Tele | 电话 | 电话 |
| IsAdmin | 是否是管理员 | 是：true；否：false |

客户角色表

|  |  |  |
| --- | --- | --- |
| 字段 | 字段描述 | 备注 |
| ID | 自增长列id | 无意义 |
| Name | 角色名称 | 角色名称 |
| CustomID | 客户编号 | 关联客户 |

## 审核设置

设置发布微博需要进行审核的账号，设置的账号在新建发布时，需要审核后才能发布，没有设置的账号，新建微博发布可直接发布。

1. 显示审核列表，字段包括：需审核的微博账号，平台，审核人，需审核的操作，修改和删除。
2. 点击“增加”按钮，选择平台类型，选择“需审核的微博账号”（从账号绑定中获取），选择“审核人”（从用户列表中获取）
3. 点击完成，返回刷新列表。
4. 支持修改和删除，修改界面同增加页面。

|  |  |  |
| --- | --- | --- |
| 字段 | 字段描述 | 备注 |
| ID | 自增长列id | 无意义 |
| AccountID | 微博账号 |  |
| Type | 平台 |  |
| UseID | 用户 |  |
| Option | 审核操作 |  |

## 角色权限维护

在客户下对角色和角色的权限进行维护。该角色列表通过客户编号调用滕工提供的服务获取的

1. 左侧为角色列表，显示所有的角色。

* 新建账号默认有admin角色，拥有所有权限。Admin不可修改和删除。
* 可以点击+号增加角色；
* 支持对角色名称的修改；
* 支持对角色的删除，删除时给出提示“删除角色，所属该角色的用户将无法登录，请确认是否删除”，删除角色对应该角色的用户无法登录，用户列表中用户角色为“空”。

1. 右侧为权限列表，显示所有的权限。切换不同角色显示不同权限。

* 权限只显示末级导航菜单的权限；
* 权限分为“编辑”，“浏览”，“导出”
* 浏览时，勾选框状态为灰，不可编辑，点击编辑时，勾选框变为可编辑状态；点击保存，进行权限保存。

角色关联权限表

|  |  |  |
| --- | --- | --- |
| 字段 | 字段描述 | 备注 |
| ID | 自增长列id | 无意义 |
| RoleID | 角色编号 | 关联角色表 |
| PermissionID | 权限编号 | 关联权限表 |

## 任务设置

1. 分为“一级分类”，“二级分类”，“新建任务”三个选项
2. 新建一级分类和二级分类，一级分类支持左右位置的互换，二级分类（组）支持修改和删除；
3. 选择二级分类后新建任务，鼠标移动到任务名称上时，在下方显示“修改”和“删除”按钮，点击按钮进行相应的操作。
4. 新建任务，填写任务名称，选一级分类，二级分类，设置表达式和垃圾词的过滤。

任务组

|  |  |  |
| --- | --- | --- |
| 字段 | 字段描述 | 备注 |
| ID | 自增长列id | 无意义 |
| GroupName | 组名 | 分类的名称 |
| ParentID | 父类的编号 | 父类的编号 |
| CustomID | 客户编号 | 关联客户 |
| Sort | 排序 | 支持分类移动 |

任务

|  |  |  |
| --- | --- | --- |
| 字段 | 字段描述 | 备注 |
| ID | 自增长列id | 无意义 |
| GroupID | 分组的编号 | 关联组 |
| Name | 任务名称 | 任务名称 |
| Keywords | 关键词 | 搜索关键词 |
| Creator | 谁创建的 | 冗余字段 |
| CustomID | 客户编号 | 客户编号 |
| StoreAddress | 数据源 | 数据源的存储地址 |

# 权限系统

## 产品权限

1、Action是具体的功能，如果Action需要授权则在Permission里面添加一个功能权限项，然后把这个功能权限项和Action关联起来即可。

2、Menu是一个菜单，在系统里面保存到Base\_Module里面，同样如果需要权限控制，则需要创建一个权限项，然后做M和P之间的关联即可。

3、UI权限，在系统里保存到Base\_UIElements里面，如果有UI需要控制权限，则创建一个对应的权限项，然后做U和P之间的关联即可。

4、为了方便用户授权，UI可以和Action之间有关系，当用户为功能授权之后，则使用该功能的所有UI都具有这个权限。所以需要一个Action和UI的关系表描述某个Action关联了多少UI。

### 权限系统提供API如下:

1、验证权限

参数：用户ID，权限项ID

返回值：允许/不允许/未设置

2、获得当前客户的可用权限列表

参数：客户ID

返回值：客户可使用的分组权限表{g1:[p1,p2,p3],g2:[p4,p5]...}

## 客户权限管理

客户权限管理是指客户管理员在本客户内部创建的权限系统，客户权限管理和产品权限管理公用一套权限项，客户通过API获得当前客户的权限项列表，来给本企业内部的角色分配权限。

## 权限验证

需要验证权限的业务逻辑在处理当前的业务逻辑时都要验证当前用户是否允许该操作，如果允许则继续否则终止。

首先验证的是客户权限，通过之后验证产品权限，通过之后处理业务逻辑，否则显示异常。

# 调用API策略

根本的限制为：每服务器总数限制 N次/小时

测试用户保证及时性全部调用API。

重要客户保证及时性全部调用API。

一般客户则使用剩下的API，和本地缓存的数据。

**API调用走统一的消息管道完成，管道负责控制流量和频率，如果调用失败则管道自动调用历史数据返回上次的历史结果。历史结果是从数据库表里面直接返回的。**

注册多个帐号和授权多个帐号可以拥有多个消息管道，这样可以提高并发数量。

# JOB引擎

创建任务产生JOB。

# 数据库表设计

业务数据和微博数据分离的策略。业务数据保存在一个数据库里面，通过Service直接访问。

微博数据则提供一个微博服务的封装。该封装提供对外的API来和外部交互。

1、插入/更新博主信息

2、插入/更新微博信息

3、查询博主和微博的信息

微博系统数据存储

微博基础数据存储

任务数据存储

## Account\_\*：和微博帐号相关的表

微博用户关系描述

假如有两个用户W1和W2，W1关注W2，则W2就是W1的朋友，W1就是W2的粉丝。在实际应用中，想知道W1的粉丝有谁，W1的朋友有谁。如果设计一张表T1来保存用户关系则可以描述为{User ID, Follow ID}，User ID表示我的帐号，Follow ID表示被关注的帐号，获得W1朋友是只需要User ID=W1即可，

Account\_BaseInfo$：微博帐号基本信息表，采样数据。\*

Account\_BaseInfo\_Track：微博帐号跟踪表，记录某些帐号的数据变化（粉丝数、微博数、关注数）。

Account\_Followers\_Track：微博帐号粉丝跟踪表，记录某个帐号的粉丝的粉丝数量、微博数量、关注数量。\*

Account\_Posts$：微博列表，用于传播分析，记录具体的微博信息。\*

Account\_Posts\_Track：微博跟踪表，用来记录某个微博的变化情况，方便趋势统计。

## Base\_\*：基础数据表

Base\_Actions：业务行为（Action）描述表。该表用来注册开发过程中使用的Action，每个开发人员和后台交互都通过Action完成，所以开发人员只需要写自己的Action类并且将其注册到系统里（就是写到这张表里面），微博系统运行的时候工厂方法会根据这张表来创建具体的Action。Action是有具有权限的，可以加入权限系统中。

Base\_Approved：业务审批表，未完成。

Base\_Customs：客户信息表，记录客户基本信息。

Base\_Logs：业务日志描述表，默认每个Action都会按照ActionName产生日志描述信息，其它日志描述信息需要自己创建系统才会自动记录。

Base\_Modules：系统模块基本信息。

Base\_Organizations：机构信息。

Base\_Permissions：基本权限信息表。

Base\_Profiles：个人配置信息表。K-V结构。

Base\_Roles：角色信息表。

Base\_Source：来源信息表，比如来自Iphone或者Note3或者网页，这是一个枚举。

Base\_User\_Approved：用户审批表。

Base\_User\_Roles：用户和角色关联表。

Base\_Users：用户基本信息表。

Base\_Weibo\_Accounts：官微或者要绑定的微博信息表。

## Disseminate\_\*：传播

Disseminate\_Posts：要进行传播分析的微博信息，具体的某个微博。

Disseminate\_Posts\_Track：微博跟踪表，跟踪指定微博的信息变化采样数据。

Disseminate\_Reposts$：微博回复列表\*。这个表可能很大。

## Event\_\*：事件

Event\_Events：事件基本信息表。

Event\_Weibo\_Track：事件对应的微博的跟踪表。

## Home\_\*：首页

Home\_Hots：首页热点信息表

Home\_Tasks\_Track：首页的舆情统计跟踪表，统计某个客户下所有的任务产生的舆情数量。

Home\_WeiboInfos：首页的微博基本信息表，显示当前客户下绑定的微博的基本信息，这个数据是个定时刷新的数据。

策略如下：根据服务器的配置（主要是API的限制，来计算本次调用的是API还是爬虫，按照企业特点和服务器资源计算每天每企业有几次调用API，当这个次数耗尽则走API），如果调用的是API则立刻返回新的数据并回写到基本信息表里面。如果是调用爬虫则显示上次的数据并产生一个爬虫任务爬虫回调更新，在用爬虫的时候这个值可能是一个快照值，会有延迟。

## My\_\*：我的微博

My\_Comments：我的回复表。保存当前用户的回复信息，记录当前用户用哪个微博帐号回复了什么信息。

My\_Follows\_Track：我的粉丝跟踪表。保存指定微博帐号的粉丝，以及这个粉丝的粉丝数量\*。

My\_Posts：我的微博表。通过微博系统发布的微博。

## Office\_\*：办公自动化

Office\_MessageReceivers：消息接收者信息

Office\_Messages：消息信息

Office\_Navies$：水军信心（网评员）

## Task\_\*：舆情

Task\_Collections：收集信息表

Task\_Groups：任务分组信息表

Task\_Tasks：任务定义信息表

微博关系存储方案(demo)

# 微博/博主存储

微博发布以后内容不会变化，可变化的内容由跟踪表跟踪变化。

## 微博内容存储（Store）

内容保存到讯库文件系统，支持定位（URL索引）。

支持关键字/表达式查询（由讯库支持）。

业务存储模型来实现具体的业务需求（微博系统实现）。

## 爬虫任务（Crawler）

定义（CralwerJob）：爬虫任务就是一组关键词，爬虫根据关键词去微博系统中爬取数据并建立讯库索引，并且调用微博系统的服务将爬取内容写入微博系统。

参数定义：

{

jobID:"作业编号，该编号由调度程序产生",

type:"keywords/url"

value:"关键字列表/URL",

depth:"深度",

callbackUrl:"回调的URL，执行成功后，把结果POST到callbackUrl"

result:"json"

}

返回结果：(JSON/XML)

{

err:0,//0无错,非0为错误码（自定义）

result:{...}

}

如果是同步调用则直接返回【返回结果】，如果是异步则调用callbackUrl（采用异步的方式）。

回写管道：爬虫回调的异步方法提供一个筛选管道的机制，可根据业务逻辑设置筛选条件分流回写数据。

interface IFilter{

void filter(string result, IDataStore store);

}

表达式过滤器：

class TaskFilter{

var task = gettask();

void filter(result,store){

var query = task.getQuery();

if(query.test(result)){

store.set(result);

}

}

}

回调可以考虑NodeJS可以解决大吞吐量的问题。

## 跟踪器（Tracker）

负责产生爬虫任务（一组关键词），并且提供关键词的排重（过滤重复的关键字）。

Task\_Tracker：任务跟踪器

Account\_Tracker：帐号跟踪器

Event\_Tracker：事件跟踪器

Weibo\_Tracker：微博跟踪器

...

Track\_Manager：跟踪器管理器

Job.Run(Track\_manager)

# 关系存储

## 业务关系分析

操作员：对应微博系统的帐号。该帐号是系统的内部帐号。

帐号：微博平台帐号，例如：腾讯微博帐号，新浪微博帐号。简称帐号。

### 微博帐号

1、授权帐号

通过系统注册的帐号受系统管理的帐号为授权帐号。

2、未授权帐号

除了授权帐号以外的帐号都是非授权帐号。

3、粉丝关系

关注了指定帐号的帐号。

4、好友（关注）关系

#### 存储模式

Radis存储粉丝列表和关注列表。

URL->(粉丝URL列表)

URL->(关注 URL列表)

微博系统不保存微博帐号的基本信息，需要保存业务数据，保存如下结构（**payload**）：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| URL | Follows | Friends | Weibos | FoAddress | FrAddress | WbAddress | ... |

URL：帐号的URL地址

Follows：粉丝数

Friends：关注数

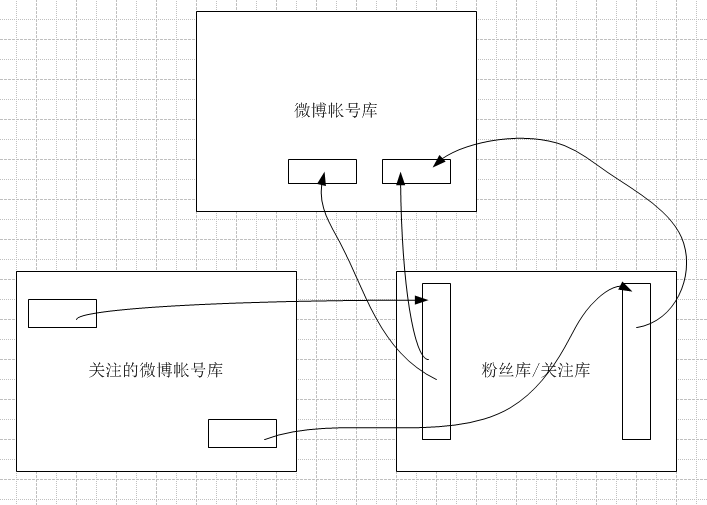
Weibos：发布的微博数

FoAddress：粉丝列表的地址，下图中的箭头

FrAddress：粉丝列表的地址，下图中的箭头

WbAddress：发送微博列表的地址，下图中的箭头

...：其他信息（情感信息）



微博帐号库：该库保存了所有系统打过交道的帐号，这个帐号库是各平台帐号库的一个采样集合。保存的是轻量级的帐号信息，参考payload。该库中的微博帐号几乎不更新或者更新频率很低。

关注的微博帐号库：微博系统需要监测的微博帐号库，包括授权帐号和非授权帐号的payload信息。

粉丝/关注库：专门用来存储关注微博帐号和微博库中的帐号的粉丝关系，Key就是关注帐号的ID，Value对应的微博帐号库里面的帐号。这里面保存的Value是已经按照粉丝数排好序的。

该方案可通过关注帐号迅速获得粉丝列表

### 微博

1、通过系统发布的我的微博

2、通过其他方式发布的我的微博（API获得）

3、跟踪微博转发、微博评论

URL->(转发URL列表)

URL->(评论URL列表)

URL->(情感URL列表)//赞

# 快速索引

原则： 关系数据库中保存元数据描述以及数字跟踪信息。

跟踪表都保存了完整的数字信息，采样数据则由爬虫和API完成。

URL是关联关系型和非关系型的KEY。

# 具体业务分析

## 微博舆情

CreateTask：创建舆情

TaskTracker.AddTask(Task)：添加到跟踪器开始跟踪任务

TaskTracker.GenCrawler()：产生爬虫任务

TaskTracker.Callback(json)：爬虫回调

基于任务的舆情系统，首先需要定义任务，根据任务产生爬虫任务，并且交给调度程序来调用爬虫，并且由爬虫回调爬取结果。

所以该部分信息是非实时（no-realtime）的（如果需要实时的可以调用商业的API接口来实现）。

admin create task -> task build Job -> job center -> Crawler run Job

operator view task -> invoke xunku webservice -> get fake realtime data from xunku-> show viewer

### 业务要求

筛选要求：

1、原创：讯库无法直接实现，需单独实现。

2、转发：同上。

3、平台（新浪、腾讯、人民），讯库的SourceID可直接获得，讯库支持。

排序要求：

1、按时间排序，讯库在时间字段上建立索引，讯库支持。

2、按转发排序，讯库无法完成，需单独实现。

3、按相似排序，同上。

4、按评论排序，同上。

响应时间：

1、调用商业API可立即获得数据。

2、调用爬虫需等待若干时间获得数据。（设计上要尽量缩短若干时间）

### 设计实现

在不考虑商业API的情况下，每个任务需要一个TaskDB用来保存这个Task相关的信息。

TaskDB是一个离线数据库，数据来源如下：

1、由爬虫回调函数完成数据的填充。

2、由JOB调度从讯库搜索来填充数据。

3、其它填充方法，API（如果有商业API则优先由API获得数据，商业API由用户自己购买）

~~TaskDB的特征是具有一类关键字的微博集合，该数据库由关键字来管理，所以该数据库是一个基于关键字的数据库~~。

列式存储（列可扩展）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| URL | Reposts | Comments | Similarities | PublishDate |

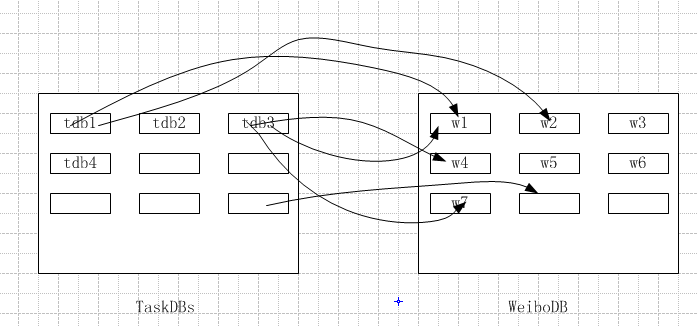
关键字会发生变化如何处理：

~~任务设置为有限多个关键字，关键字之间有逻辑关系，逻辑关系为AND OR NOT三种集合关系。例如：关键字组W1 AND W2 AND W3 OR W4 NOT W5。~~

1、重建任务库，抛弃一个DB，立刻重建一个并从讯库里面搜索数据填充库，重建库搜索讯库最近3个月的数据（已最快为目的）

2、追加任务库，历史数据保留

每个TaskDB都是任务到WeiboDB的索引管理器，每个TaskDB管理一组微博。



WeiboDB保存的是微博的基本信息集合，由讯库提供。

任务创建流程：

1、管理员创建任务

2、任务提交到任务管理器

3、管理器创建任务存储器

4、管理器创建任务作业（JOB）

5、提交任务作业到爬虫

6、等待爬虫回调任务管理器

7、回调后将数据写入任务存储器

任务使用流程：

1、打开任务视图

2、根据排序需求查询任务存储获得URL列表

3、Build查询结果集

4、返回结果集

## 事件监控

业务索引槽：根据每个业务需求，建立对应的索引槽满足业务排序的需求，索引槽都是排好续的。

索引槽存储：内存存储/硬盘存储？DUMP？