# 数据库设计

## 1 数据库说明

数据库的设计是综合数据爬取和数据统计的结果，可以让两方面都比较方便的使用。

### 1.1 数据任务的派发

添加一个新任务时，会将这个任务存放到tasklist表，这个表以用户uid为主键。里面存放粉丝数据所在的表、爬取间隔时间、任务创建时间之类的。

为了对评论、转发、@数据的增量爬取，还有一个任务表。里面存放的是每条微博评论、转发的爬取数量、爬取状态。方便下次爬取时，判断上次爬了多少数据。

### 1.2 评论、转发、@的数据

对于评论、转发、@的数据：我们要统计的数据中对评论、转发、@的数据大部分都是以天和自然周为单位进行统计的。所以可以考虑将评论、转发、@原始数据以周的方式分表处理。这样有利于存储，后期统计也比较好，统计天的数据只用查找那一天所在的周，统计周的数据也只用查那一张表就可以，统计月的数据就要查4张表。

爬取是一小时一次。数据是增量添加，为保证数据完整性，一条微博的评论、转发都在一张表里。不管它的评论、转发是否已经查过了周限制。

对于表空间的大小，假如我们检测1000个账号，每天每个账号发微薄平均3篇，总共评论、转发200次。那么一周的原始数据表单表里将有7\*1000\*3\*200=420W数据。对于数据库的压力来说不是问题。

### 1.3 粉丝数据

对于粉丝数据：粉丝最开始只能爬取5000个，以后就是每个小时的增量爬取。对于粉丝表来说一些粉丝增量比较慢的账号，表里面的数据会比较少。但是也不能将所有被监测用户的粉丝数据都放在同一张表里。所以做了分表处理。

分表策略：爬取用户基本信息的时候查看他微博创建时间、微博现有粉丝数。然后计算下它到现在粉丝增长的比例。判断它将近1年后数据增长空间。这个值只做参考意义。将增长比例比较小的用户的粉丝放在一张数据表里。程序控制，自动建表，保证每张表的数据小于1000W。这样后期统计查询数据比较快。

### 1.4 被监测用户的详细信息

这个数据是为了看用户每天的粉丝变化，每天基本信息的变化。可以将所有被监测用户的基本信息单独放在一张表，对每个用户，一天增加一条信息。

假如有1000个被监测用户，每天也只会多1000条数据。对于统计近1个月来用户粉丝的变化，基本信息的变化，查询速度也很快。

## 2 数据库表结构

### 2.1 任务表

原始任务表一个被监测用户一行数据。如果后期有需要在前端显示的内容，可以扩充这个表里的字段。现在这些字段是现在需要用到的。

CREATE TABLE `tasklist` (

`uid` bigint(20) DEFAULT NULL COMMENT '用户id',

`screen\_name` VARCHAR(60) DEFAULT NULL,

`uid\_created\_at` datetime DEFAULT NULL COMMENT '微博账号创建时间',

`finishtype` tinyint(4) DEFAULT NULL COMMENT '完成任务类型 0为未开始 1为开始未完成 2为已完成 3为任务出错',

`taskerr` varchar(500) DEFAULT NULL COMMENT '任务出错信息',

`follow\_table\_name` varchar(100) DEFAULT NULL COMMENT '粉丝原始数据存的表名',

`create\_at` datetime DEFAULT NULL COMMENT '任务创建时间',

`lastupdatetime` datetime DEFAULT NULL COMMENT '最后更新时间',

PRIMARY KEY (`uid`),

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

评论转发任务表。里面存放的是所有被监测用户的微博、@品牌账号的微博。

CREATE TABLE `weibolist` (

`weiboid` bigint(20) NOT NULL DEFAULT '0' COMMENT '微博id',

`uid` bigint(20) DEFAULT NULL COMMENT '发微薄用户id',

`tasktype` tinyint(1) DEFAULT 0 COMMENT '任务类型，0为品牌用户的微博，1为@品牌账号的微博',

`respond\_time` datetime DEFAULT NULL COMMENT '品牌回应时间只有当上面类型为1时才可能有这个值',

`hashtag` VARCHAR(100) DEFAULT NULL ,

`profile\_image\_url` varchar(600) DEFAULT NULL,

`verified\_type` int(11) DEFAULT NULL,

`source` varchar(50) DEFAULT NULL,

`weibo\_created\_at` datetime DEFAULT NULL,

`text` varchar(1000) DEFAULT NULL,

`repost\_count` int(11) DEFAULT '0' COMMENT '转发数',

`repost\_finishnum` int(11) DEFAULT '0' COMMENT '完成任务数量',

`repost\_finishtype` tinyint(4) DEFAULT NULL COMMENT '完成任务类型 0为未开始 1为开始未完成 2为已完成 3为任务出错',

`comment\_count` int(11) DEFAULT '0' COMMENT '评论数',

`comment\_finishnum` int(11) DEFAULT '0',

`comment\_finishtype` tinyint(4) DEFAULT NULL,

`taskerr` varchar(500) DEFAULT NULL COMMENT '任务出错信息',

`repost\_table\_name` varchar(100) DEFAULT NULL COMMENT '转发原始数据存的表名',

`comment\_table\_name` varchar(100) DEFAULT NULL COMMENT '评论原始数据存的表名',

`direct\_table\_name` varchar(100) DEFAULT NULL COMMENT '@原始数据存的表名',

`create\_at` datetime DEFAULT NULL COMMENT '创建时间',

`lastupdatetime` datetime DEFAULT NULL COMMENT '最后更新时间',

PRIMARY KEY (`weiboid`),

KEY `uid` (`uid`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

### 2.2 原始数据表

评论、转发、@数据的格式都差不多，可以在下面的基础上添加或删减字段。

CREATE TABLE `weibodata` (

`id` bigint(20) NOT NULL COMMENT '转发ID',

`src\_uid` bigint(20) DEFAULT NULL COMMENT '发帖人ID',

`src\_id` bigint(20) DEFAULT NULL COMMENT '被转发微博ID',

`uid` bigint(20) NOT NULL COMMENT '用户ID',

`screen\_name` varchar(100) DEFAULT NULL COMMENT '转发人名称',

`source` varchar(50) DEFAULT NULL COMMENT '发微薄工具',

`province` int(11) DEFAULT NULL COMMENT '用户省份',

`location` varchar(150) DEFAULT NULL COMMENT '用户地址',

`gender` varchar(5) DEFAULT NULL COMMENT '性别',

`verified` tinyint(1) DEFAULT NULL COMMENT '是否是微博认证用户',

`verified\_type` int(11) DEFAULT NULL COMMENT '认证类型',

`reposts\_count` int(11) DEFAULT NULL COMMENT '转发数',

`comments\_count` int(11) DEFAULT NULL COMMENT '评论数',

`text` varchar(1000) DEFAULT NULL COMMENT '转发内容',

`statuses\_count` int(11) DEFAULT NULL COMMENT '用户微博数',

`friends\_count` int(11) DEFAULT NULL COMMENT '用户关注数',

`followers\_count` int(11) DEFAULT NULL COMMENT '用户粉丝数',

`favourites\_count` int(11) DEFAULT NULL COMMENT '用户收藏数',

`tags` varchar(500) DEFAULT NULL COMMENT '用户tag',

`created\_at` datetime DEFAULT NULL COMMENT '转发时间',

`raw\_data` text COMMENT '原始数据',

`lastupdatetime` datetime DEFAULT NULL COMMENT '最后一次更新时间',

`profile\_image\_url` varchar(600) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY ` src\_uid ` (`src\_uid`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='微博原始数据表';

粉丝原始数据表

CREATE TABLE `weibofollowers` (

`uid` bigint(20) NOT NULL COMMENT '用户ID',

`src\_uid` bigint(20) NOT NULL COMMENT '被关注人的用户ID',

`screen\_name` varchar(100) DEFAULT NULL COMMENT '用户名称',

`source` varchar(50) DEFAULT NULL COMMENT '发微薄工具',

`province` int(11) DEFAULT NULL COMMENT '用户省份',

`location` varchar(150) DEFAULT NULL COMMENT '用户地址',

`gender` varchar(5) DEFAULT NULL COMMENT '性别',

`url` varchar(100) DEFAULT NULL COMMENT '用户微博地址',

`weihao` varchar(50) DEFAULT NULL COMMENT '用户的微号',

`verified` tinyint(1) DEFAULT NULL COMMENT '是否是微博认证用户',

`verified\_type` int(11) DEFAULT NULL COMMENT '认证类型',

`created\_at` datetime DEFAULT NULL COMMENT '用户创建（注册）时间',

`description` varchar(255) DEFAULT NULL COMMENT '描述',

`statuses\_count` int(11) DEFAULT NULL COMMENT '微博数',

`friends\_count` int(11) DEFAULT NULL COMMENT '关注数',

`followers\_count` int(11) DEFAULT NULL COMMENT '粉丝数',

`favourites\_count` int(11) DEFAULT NULL COMMENT '收藏数',

`tags` varchar(500) DEFAULT NULL COMMENT '用户tag',

`raw\_data` text COMMENT '原始数据',

`lastupdatetime` datetime DEFAULT NULL COMMENT '最后一次更新时间',

PRIMARY KEY (`uid`,`src\_uid`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='粉丝原始数据表';

用户每天原始信息表

### 2.3 统计结果表

# 任务流程

## 1用户添加URL

### 1.1说明

用户提交URL后，后台判断是否是新浪微博用户的URL，如果是就继续后续操作，根据url提取出用户的ID或者screen\_name获取用户信息然后存入数据库。如果不是，会提示用户输入URL错误。如果用户输入的URL是以前添加过的的，会提示用户该用户以前就检测过。

### 1.2 流程图



## 2 后台调度

### 2.1 说明

爬取目前订为1个小时爬取一次。可以使用linux的crontab执行shell脚本1个小时调用一次调度程序。让调度程序进行统一调度，爬取需要爬取的数据。

调度程序先从数据库中取出所有需要爬取的用户数据。这里需要注意上一次没有爬取完成的任务。对于上一次没有爬取完成的任务可以做下统计是因为什么原因没有爬取完成。然后将这些任务分发给10个不同的进程进行爬取。

下面有2种分发思路。

一种是将所有要爬取的用户数据分为10等分然后分给10个进程进行爬取。

另一种是每次分发一个任务给一个进程，累计10个之后停止分发。当有任务完成时，调度程序继续分发一个任务给一个进程。直到所有任务完成。

分发的方法也有2种考虑。

一种是使用webserver请求一个页面，页面函数里写上需要爬取的方法。

另一种就是直接执行代码。

### 2.2 流程图



## 3 数据爬取

### 3.1 说明

所有要爬取的信息有如下几种：

1. 粉丝

2. 粉丝的标签

3. 微博详细内容

4. 微博的评论

5. 微博的转发

6. 微博@的数据

7. 被监测用户每次的详细信息

以上要爬取的内容我们可以分成3个类。

第一类：粉丝和粉丝的标签作为一个整体，他们有依赖关系，要先爬取粉丝，才能后续的爬取粉丝的标签。

第二类：微博内容和微博的评论和转发。先获取微博内容然后获取单挑微博的评论和转发。

第三类： 微博@数据，这个是单独的一个部分.

这3类爬取任务对应3个不同的爬取模块。调度时候可以分开调度，也可以一个用户做成一个串行任务。这个可以实际应用中做不同的考虑。

爬取评论和转发的量比较多，如果消耗的时间特别长的话，可以考虑使用多线程。

### 3.2总体流程图



### 3.3 分部流程图

爬取转发、评论



爬取粉丝、粉丝tag



爬取@品牌账号数据



### 3.4爬取所需的资源

根据在SAE近2个月来爬取的评论转发数据，API请求基本是130-160条数据/S。一次API请求的数据为200条。所以每次API请求为1.25秒左右。放宽条件取1秒进行计算。

API请求限制为4W/小时 我们开10个进程进行爬取，不间断的爬取1个小时调用API次数为1\*10\*60\*60=36000次，在4W次以内。

存储方面可以看到SAE上的原始数据表

weibocomments      515,869   MyISAM   utf8\_general\_ci        2.2 GB

weiboreposts  1,009,524         MyISAM   utf8\_general\_ci        2.3 GB

评论的数据 每一条4.2KB左右 转发的数据 每一条2.2KB左右

## 4 数据统计

数据统计是爬取完成后的操作，有分天任务，周任务，月任务。下面给出了需要统计的数据，做了分类，以及需要用到的数据。

### 4.1 fans统计

fans统计可以分为三类：

1. 每天粉丝增长、每周粉丝增长、粉丝的增长率。这类是粉丝的基本信息可以直接查看用户详细信息表得到粉丝每天的变化。

2. 不同类型的新粉丝比例、粉丝范围比例、TOP10省份比例、性别、年龄段、粉丝tag。这类数据是查找粉丝原始数据表可以得到的。一个用户的所有新增粉丝详细数据都存在一张表中。只需要取出该用户的天、周、月数据就可以得到上面的统计结果。

3. 活跃粉丝和互动粉丝。 这个类数据是API请求商业接口得到。

4. 粉丝活跃时间比例。这个是查看每天、每小时有多少粉丝和品牌账号互动（转发、评论）。

### 4.2总体行为统计

总体行为可以分成下面几种：

1. 每天发微博总数、转发总数、评论总数、@被监测用户的微博总数。这类数据直接查询原始数据表就可以得到。这类数据得到了每天的数据后，每周，每月的数据累加就行了。

2. 曝光总数。这个是用户授权后，用商业接口得到的数据。

3. 7天和30天的ER值。ER是 (转发+评论+@)/微博数/粉丝数。

### 4.3 微博统计

1. 互动TOP5的微博URL、ER、RT、CT。互动值是通过ER判断的，ER是(转发数+评论数)/粉丝数。在微博任务表中有转发、评论数值，这个数据每次爬取的时候会更新，所以每次计算结果可能不同。分天、周统计，可以给时间点显示统计最新数据。

2. TOP3的hashtag。这个是在微博任务表中找出所有带有hashtag的微博，计算出ER值。一个hashtag的ER值是所有带这个hashtag标签的ER的平均值。

3. TOP3的影响者的url、转发数、评论数、@数。根据转发数，评论数以及直接@数的总数来排名。在评论、转发、@原始表中可以找到，然后统计。

### 4.3 @数据统计

1. @品牌账号的微博数：这个数据上面总体行为里面也有。

2. 品牌的响应问题的数量：查看@原始数据表，看是否有品牌账号的评论。这个可以在爬取的时候就判断，然后更新评论、转发任务表里的字段。让它知道这条微博品牌有回应。

3. 品牌响应的平均时间：将有品牌评论的时间提取出来，和发帖时间对比。

## 5 前端数据展示