1. 默认用户列表

正在活动期内的活动，每个活动的所有歌曲随机打乱，生成1000份数据，放到redis集群，key为 default\_${actid}\_1 default\_${actid}\_2，default\_${actid}\_1000.values类型为redis的list类型

此外还要存一个各个活动的整体统计信息，

Key为default\_${actid} field = size 表示每个活动的歌曲数，field 为count表示每个活动保存的随机份数，目前是1000

1. 热门列表：

具体功能说明参考需求文档，

目前在redis的保存格式是：

Key 是hot\_${actid} value 是20个歌曲id

1. 用户动态和大V列表的功能参考需求文档（开发：杜焕明），

在redis的存储：

用户动态：key： activity+活动ID 如activity111

Value ：activityid|userid|songid, activityid|userid|songid

大V：key ：he\_activity\_user\_bigv

Value：用逗号分隔的歌曲id

1. he\_activity\_match\_songs表分数的计算(暂时还有小问题)：
2. 计算规则，具体统计方法（**每项占多少分是mis设置的**）：



单曲满分为100分，歌曲播放次数占20分，歌曲收藏占30分，歌曲被投票占50分；A歌曲试听量为1000次试听项排名第一，B歌曲试听量为800次，则A歌曲在试听项上的得分为20分，B歌曲试听项得分为20\*ln800/ln1000=19.4分。

1. 首先要根据真实的数据进行计算得分，试听量取自统计日志，点赞数和投票数取自投票的api接口。

日志的格式如下：

TRACE 3289786231 20160307 14:54:49 /home/bae/zhangzilong/lua\_he/ui/song.lua:50 url[/hevoice/activity/song/play?song\_id=10181] ref[] ip[192.168.217.11] msg[log\_song\_play] baiduid[] log\_source[] cost[0] cuid[] log[111\_10156\_1234\_2]

其中msg是log\_song\_play是试听日志，最后一个字段是统计信息。

格式是 activityid\_songid\_userid\_userlevel，

投票和点赞的接口参考：

http://wiki.taihenw.com/pages/viewpage.action?pageId=3736304

1. 先按照得分排序，得分相同按照投票量排序，投票量相同按照点赞量排序，点赞量相同按照试听量排序，试听量相同按照上传时间排序（这个目前可以通过he\_activity\_match\_songs的id判断出来，id小的上传时间早）
2. 如果存在运营手工干预的歌曲（he\_activity\_match\_songs 表的promotion 是1），处理规则如下，每个活动会有一个晋级的人数，这个从mis设置，具体设置见上图
3. 手工干预的歌曲必须保证每次计算的时候都晋级（即排名在每个活动的那个设置的晋级人数之内），且排名从最后开始计算，顺序不保证。比如某活动有100个晋级人数。人工干预的有五首歌，要保证每次计算的时候这五个人工干预的人排名在96到100之间，顺序不用保证，但是如果这五个人中有某一个人已经到了97名（在96-100之间），这个人的分数则不用再处理，排名保证在96-100之间即可。如果这五个人其中有人已经排在了90名，（不在96-100名之间）那么这个人的排名和分数都不用改。
4. 对于人工干预的歌曲，系统会按照新的排名，并参考其前后的歌曲的分数，计算出一个虚拟的分数、试听量、点赞数、投票数，对于这些歌曲的虚拟分数、点赞、试听、投票数，不保证其虚拟得分能通过虚拟的点赞、试听、投票数计算得出，虚拟的量不一定符合计算规则
5. 最后将每个活动的真实的试听量、虚拟的试听、点赞、投票、排名、得分入库，保存在数据里。