**任务执行逻辑**

任务启动

删除affiliatetrackdata中datatype = 9的临时数据

重置任务的状态为初始状态

读取任务配置数据

初始化进度统计计数器

统计任务数据总数

检查数据总数

若数据总数大于0

检查任务属性match\_action值

若match\_action = ‘fix’

删除表report\_trackmatch中matchId为当前任务的matchId并且datatype=2的数据

若match\_action = ‘match’

删除表report\_trackmatch中matchId为当前任务的matchId并且datatype=3的数据

更新表report\_trackmatch中matchId为当前任务的matchId的数据的matched状态为0

若 match\_action = ‘mbs’

更新表report\_trackmatch中matchId为当前任务的matchId的数据的matched状态为0

初始化任务数据匹配状态信息

循环任务数据

执行核心任务逻辑（见下页core begin至core end标记中间部分）

若数据总数等于0

检查任务属性no\_confirm\_data是否为1， 该属性标记以系统统计的数据为准进行匹配

若no\_confirm\_data = 1

检查任务属性match\_action值

若match\_action=’match’

统计report\_trackmatch中符合任务属性设置的记录总数

更新上述记录的匹配状态

更新任务状态

重建数据报表

获得任务的起始和截止时间

将截止时间增加一天

调用报表生成过程为当前任务重建数据报表

---------- core begin -------------

判断match\_action操作类型

**若match\_action=fix，执行数据补入操作**

检查site\_id值

若site\_id大于0

检查site表中site\_id是否存在

若site\_id存在

<继续后续脚本>

若site\_id不存在

更新任务数据的状态disabled为1，标记匹配消息为“BAD DATA(No such site!)”

<中断，继续下一条任务数据>

若site\_id不大于0

更新任务数据的状态disabled为1，标记匹配消息为“BAD DATA(Need site!)”

<中断，继续下一条任务数据>

判断任务数据中affilliate\_advertise\_id是否大于0

若affilliate\_advertise\_id大于0

检查在维度表summarydimension中是否存在该affilliate\_advertise\_id

若affilliate\_advertise\_id存在

检查campaign\_id是否一致

若campaign\_id一致

从维度表中选出系列维度ID（并根据affilliate\_advertise\_id在关系表advertiseaffiliate中查出advertise\_id）

更新当前任务数据的系列维度ID值

若campaign\_id不一致

更新任务数据的状态disabled为1，标记匹配消息为“BAD DATA(Aid own a different campaign!)”

<中断，继续下一条任务数据>

若affilliate\_advertise\_id不存在

标记需要修正site表与advertise的关系

若affilliate\_advertise\_id不大于0

标记需要修正site表与advertise的关系

判断是否需要修正advertiseAffiliate表关系

如需要修正advertiseAffiliate关系

检查advertise\_id值

若advertise\_id大于0

检查advertise表中是否存在该值

若advertise\_id存在

检查关系表中是否存在当前site和advertise的关系

若存在

执行修正：按创建时间选取最古老的关系id作为advertise\_affiliate\_id

标记无需补入advertiseAffiliate关系

**[解决早期重拿广告产生多个关系但却未保存后拿ID的bug]**

若不存在

标记需要补入advertiseAffiliate关系

若advertise\_id不存在

标记需要修正advertise\_id

标记需要补入advertiseAffiliate关系

若advertise\_id不大于0

标记需要修正advertise\_id

标记需要补入advertiseAffiliate关系

判断是否需要修正advertise\_id

若需要修正advertise\_id

根据campaign\_id从advertise表中选取最古老的advertise.id，修正到当前advertise\_id

检查修正advertise\_id后在关系表上当前site是否存在其它关系

若存在

执行修正：按创建时间选取最古老的关系id作为advertise\_affiliate\_id

标记无需补入advertiseAffiliate关系

**[解决在补入关系时又可能发生重复关系的问题，该表未建唯一索引]**

若不存在

标记需要补入advertiseAffiliate关系

判断是否需要补入advertiseAffiliate关系

若需要

向关系表中补入关系记录（当前site\_id,advertise\_id）

获取最新插入的关系id，设置到当前数据的advertise\_afffiliate\_id中

若effect\_ip为空，则修正ip值为fix

创建补入数据的唯一ID值

向affiliatetrackdata表插入一条dataType=9的数据，获取其ID作为补入数据的id，（ip为随机字符）

计算默认佣金

查询佣金规则，根据规则类型和数值，计算当期数据的佣金，详情见佣金规则需求说明书

如果同时确认佣金，则计算梯度佣金

根据梯度佣金规则，详情见规则需求

**[解决在佣金确认匹配时对需要补入的数据的处理]**

向report\_trackmatch表补入新数据

更新当前任务数据信息

**若match\_action = match，执行匹配动作**

检查数据状态，预拼装查询条件@share\_sql，准备执行匹配时使用

**加入条件: 数据状态为未匹配 matched=0**

判断任务数据中affilliate\_advertise\_id是否大于0

若affilliate\_advertise\_id大于0

检查在维度表summarydimension中是否存在该affilliate\_advertise\_id

若affilliate\_advertise\_id存在

检查campaign\_id是否一致

若campaign\_id一致

从维度表中选出系列维度ID（并根据affilliate\_advertise\_id在关系表advertiseaffiliate中查出advertise\_id）

更新当前任务数据的系列维度ID值

**加入条件：advertiseAffiliateId = affilliate\_advertise\_id**

**[解决因为补入数据修正了aid后匹配不到的问题]**

若campaign\_id不一致

更新任务数据的状态disabled为1，标记匹配消息为“BAD DATA(Aid own a different campaign!)”

若affilliate\_advertise\_id不存在

<继续后续脚本>

若affilliate\_advertise\_id不大于0

<继续后续脚本>

判断任务数据中effect\_cid是否为空

若effect\_cid为空

<继续后续脚本>

若effect\_cid不为空

**加入条件：orderid = effect\_cid**

检查effect\_ip是否为空

若effect\_ip为空

加入条件：trackIp = “fix”

若effect\_ip不为空

**加入条件：trackIp = effect\_ip**

加入效果产生的时间约束

获取effect\_date的yyyy-MM-dd格式

起始时间： yyyy-MM-dd 00:00:00

截止时间： yyyy-MM-dd 23:59:59

**加入条件：trackTime >= 起始时间**

**trackTime <= 截止时间**

加入campaign约束

**加入条件：campaignId = campaign\_id**

加入排序

排序：trackTime asc

加入返回数量限制 limit 0,1 只取符合条件的最古老的一条记录

根据@share\_sql条件查询report\_trackmatch，检查是否存在记录

若存在记录

计算默认佣金和梯度佣金

更新report\_trackmatch表中数据及状态

更新当前任务数据状态

若不存在记录

检查site\_id是否存在

若存在

<继续后续脚本>

若不存在

调用效果Site分配过程(见后续说明)，指派site\_id给当前记录

**调用match\_action = fix过程逻辑，设定同时执行佣金确认**

**[解决有效点击约束导致的未记录点击被确认有效的问题]**

**若match\_action = mbs，执行按总数匹配动作，通常只适用于cpc**

根据当前effect\_date从 report\_trackmatch中选取当前campaign和effect\_type的跟踪记录，加入随机数，然后从该结果集中按随机数排序选取effect\_quatity指定的数量

更新上述数量的记录集匹配状态为已匹配，并设置匹配后佣金等于匹配前的佣金。

**------------- core end -------------------**

存储过程结构划分

执行上述逻辑的存储过程共分为5个：

proc\_match\_task 主控SQL程序

proc\_task\_dofix 执行数据补入的SQL程序

proc\_task\_domatch 执行数据匹配的SQL程序

proc\_task\_dombs 按指定的效果总数值进行匹配，以数据库中记录的数据为准

proc\_commision\_rule 根据佣金规则计算佣金的SQL程序

存储过程之间的关系：

proc\_match\_task 调用 proc\_task\_dofix 和 proc\_task\_domatch，proc\_task\_domatch在无法找到siteId时，会设置siteId为达闻的默认id，并调用proc\_task\_dofix将数据补入，在proc\_task\_dofix和proc\_task\_domatch中，都会调用佣金计算程序proc\_commision\_rule。