**告警合并流程图:**



**计时合并流程图：**



**合并计数流程图：**

****

**告警合并概述：**

告警合并所需：告警数据，合并规则，合并缓存，POSTGRE（主表和子表）

告警数据是由阈值匹配生成。

合并规则是进行合并功能的算法的规则，在项目启动后从POSTGRE数据库的规则表中读出，存在在本地合并缓存当中，容器为map[Key]Value。Key为string类型的字符串，由CI和KPI组合而成:CI\_KPI。Value为struct结构体，保存形式为JSON字符串。

合并缓存是判断进行“更新旧数据”还是“插入新数据”的依凭。合并缓存存放在redis中。Key为string类型的字符串由CI和KPI组合而成:CI\_KPI。value为string类型的数字值（例如：10）。

主表和子表：主表保存的是新生成的告警和合并后的更新的数据。子表保存的是每次需要合并的数据。

**告警合并流程说明：**

1. 根据告警数据获取合并规则
2. 如果获得到的规则的处理类型是合并计数countType，并获取到规则当中最大的maxcount合并次数
   1. 从合并缓存中获取合并信息（存在）,并获取到上次合并的合并次数count

2.1.1 如果count+1的值大于等于maxcount,将本次告警数据更新到数据库主表中，并也将本次的告警数据新增到子表中，同时清除该合并缓存

2.1.2 如果count+1的值小于maxcount,将本次告警数据更新到数据库主表中，并也将本次的告警数据新增到子表中，同时更新该合并缓存的count值

2.2 从合并缓存中获取合并信息（不存在）

2.2.1 将告警数据的CI\_KPI作为KEY写入合并缓存，value值为0，并将告警数据新增到主表中。

3. 如果获取到的规则的处理类型是计时合并timeType

3.1 从合并缓存中获取合并信息（存在），将本次告警数据更新到数据库主表中，并也将本次的告警 数据新增到子表中

3.2 从合并缓存中获取合并信息（不存在），将本次告警数据新增到数据库主表中，同时将数据写入 合并缓存（有效期为计时合并的有效期。过期后，该合并信息删除）

**告警合并流程伪代码：**

//获取合并规则是否存在,并获取合并规则

Value,ok := MergeRule[“CI\_KPI”]

//存在规则

If ok{

Var merge model.MergeRule

//将value的json字符串转化为merge结构体

Json.Unmarshal([]byte(value),&merge)

//如果获得到的规则的处理类型是合并计数countType

If(merge.Type == const.MERGE\_TYPE\_COUNT){

}

//如果获取到的规则的处理类型是计时合并timeType

If(merge.Type == const.MERGE\_TYPE\_TIME){

}

}