关于属性系统的计算公式

- **计算尽可能通用：Buff可能影响的基础属性数据、状态等等，就基础属性数据而言，我们希望公式的统一，因此类似LifeData内的血量数据将转移至属性中处理；**
- **Buff失效后可回退到生效前状态：加法因子和乘法因子单独存储记录；**

最初属性计算公式：

```
float value = (attr.BaseValue + attr.AddValue) * (1 + attr.MultiValue);
```

结果 = （基础值 + 加法因子）（1 + 乘法因子）

加法因子 = 加法因子1 + 加法因子2 + ... + 加法因子n;

乘法因子 = 乘法因子1 + 乘法因子2 + ... + 乘法因子n;

任何一个BUFF的失效，最多也是对 加法因子 和 乘法因子 的 对应的某一项归 0；

公式引发的问题：溢出

场景1：某个Buff 一直加血，加法因子一直累积。
attr.Value = 100;
attr.AddValue = 1000;
此时我们需要扣血 -50 ， 希望在 attr.Value = 100 的基础上扣血
然后通过公式得到：
*float value = (100 + 1000 - 50) * (1 + 0);*
value = 1050;
然后通过最大最小值clamp，value = 100，显然扣血失败，这是一个溢出问题导致的BUG。

场景2：某个Buff 改变移速。
attr.Value = 100;
attr.AddValue = 1000;
*float value = (100 + 1000) * (1 + 0);*
若移速最大值为1000，那么value 最后结果被clamp后，value就是1000，满足。
当buff失效时，加法因子的某一项归0。
*float value = (100 + 0) * （1 + 0）;*
value = 100;
移速恢复到Buff生效前，整个过程正常。

这个差异也在于血量的buff效果，一般在失效时不需要考虑重置回去。

最后得出结论：

```
float value = (attr.Value + attr.AddValue) * (1 + attr.MultiValue);
attr.AddValue = 0;
attr.MultiValue = 0;
attr.Value = value;
```

类似血量的Buff 是不需要考虑失效时重置的，所以得出以下分类

```
    //判断基础属性是否是可重置类型的基础属性
    if (self.CheckAttrRecoveryState(attrType))
    {
        //可重置属性的计算
        return self.CalculateRecoveryAttr(self.attrDic[attrType]);
    }
    else
    {
        //不可重置属性的计算
        return self.CalculateClampAttr(self.attrDic[attrType]);
    }
```

修改公式后，还需要把部分数据中的数据转移。例如LifeData。

在修改属性系统之前，HpValue是存储在LifeData中的，统一计算公式后，LifeData中的 UpdateHp 中的计算，将采用通用属性计算公式。

属性的PB结构

- 当前值
- 基础值
- 加法因子
- 乘法因子
- 最大值属性ID
- 最小值属性ID

```
//属性
message Attr
{
    required uint32 id                                  = 1;// attribute_id 配置表ID  Attribute.xls  CharacterAttribute的cfg_id
    optional float  value                               = 3; // 当前值
    optional float  base_value                          = 4; // 基础值
    optional float  add_value                           = 5; // 加法系数
    optional float  multi_value                         = 6; // 乘法系数
    optional uint32 min_link                            = 7;
    optional uint32 max_link                            = 8;
}
```

最大值、最小值属性ID也许不需要，如果需求上，不存在对最大值、最小值修改的Buff，例如：某个buff 修改血量的最大值，然后需要在buff 失效后，最大值恢复，这种buff就不应当存在。

Buff系统设计

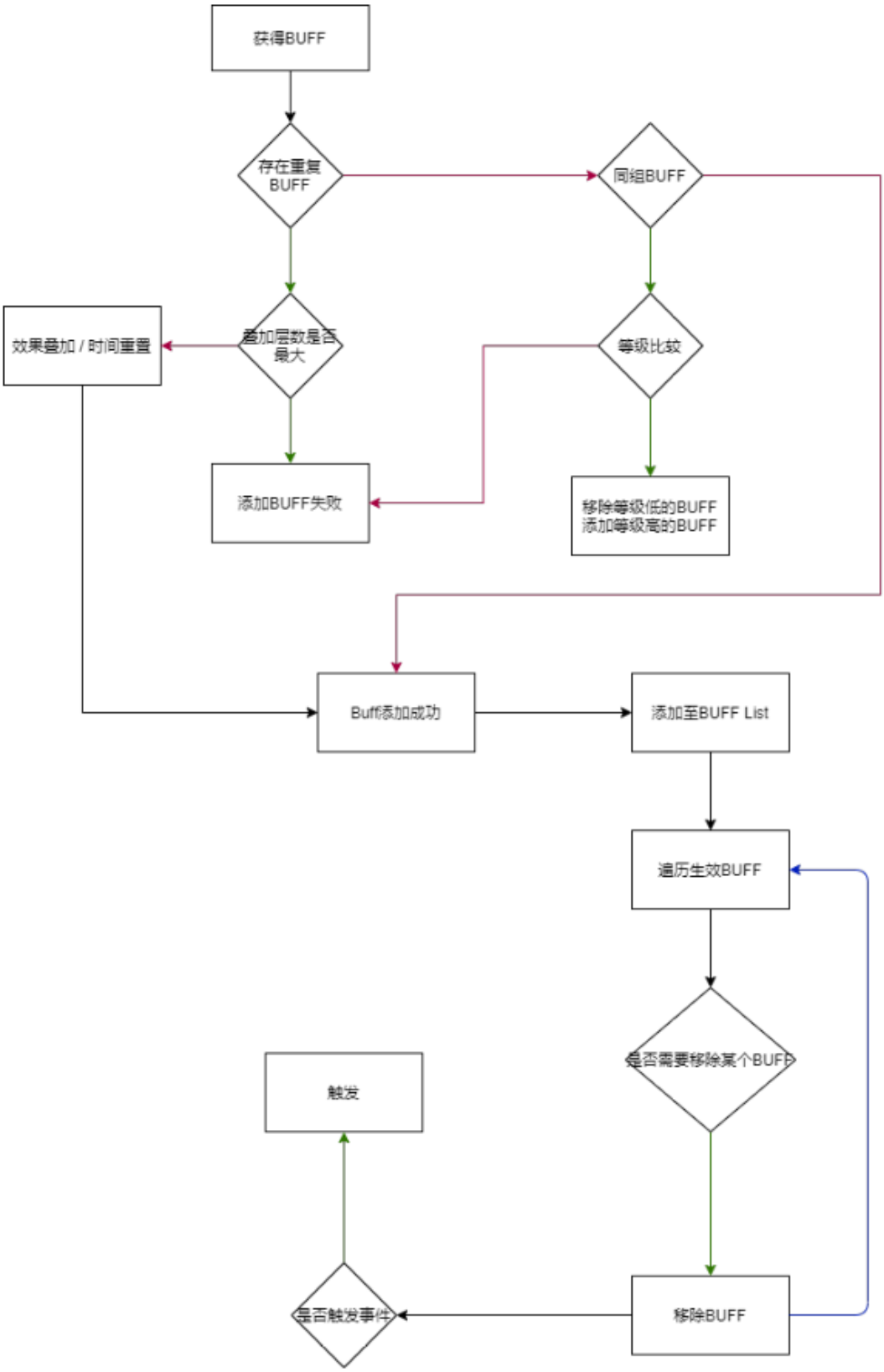
每个buff可以包含多个buff效果。buff效果是指实际生效的内容，例如:加血、加移速、改状态等等。而一个Buff可以同时包含加血、加移速。

- 指定生效EntityType
- 指定过滤EntityType
- buff所处Group
- buff所处等级
- 可叠加层数
- 叠加后的处理方式
- 关联buff图标、特效等资源

required	required	required		optional	optional	optional	optional	optional	optional	optional	optional	required	optional
uint32	string	string		string	string	uint32	uint32	uint32	uint32	string	uint32	float	uint32
cfg_id	name	desc		for_entity_list	without_entity_lis	group	level	max_count	reset_type	buff_icon	effect_id	duration	great_type
状态ID	buff名称	buff描述	备注	指定作用的Entity类型	指定无法作用的Entity类型	组ID	等级（默认0）	最大层数（默认1）	叠加类型（1=重置时间且执行，默认=不重置时间，2=重置时间且不执行）	buff图标	特效ID	持续时间(-1表示无限时间)	Buff类型(1=增益；默认=减益)

添加buff 的流程

- 需要比较是否已存在相同Buff
- 比较是否存在相同Group的Buff
- 比较Buff等级
- 叠加的处理方式



BuffEffect的结构

实际的buff效果，buff只有包含它，才有实际意义

	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE
		required	optional	optional	optional	optional	required		optional	optional	optiona	optiona	optiona	optional	optio	optiona	optional
		uint32	uint32	float	float	uint32	float		uint32	uint32	uint32	float	float	string	uint3	uint32	uint32
		effect1_trigg	effect1_tr	effect1_tr	effect1_tr	effect1_tr	effect1_trigger_rate		effect1_fu	effect1_fu	effect1_	effect1_	effect1_	effect1_f	effect	effect1	effect1_reset
	效果1-触发类型备注	效果1-触发类型（选择备注复制公式）	效果1-触发参数1	效果1-触发参数2	效果1-触发参数3	效果1-触发参数4	效果1-触发概率	效果1-效果类型备注	效果1-效果类型（选择备注后自动生成，复制）	效果1-效果参数1	效果1-效果参数2	效果1-效果参数3	效果1-效果参数4	效果1-效果参数5	效果1-特效ID	效果1-音效ID	移除时是否重置效果(默认不重置;1=效果失效重置;2=buff移除时重
3	立即	2					1	状态修改	13	11	1						2

触发类型

很多buff是被动类型或延迟类型的，并不是立即生效，所以需要管理起来
不同的触发类型，在不同的时机下才会生效真正的效果。

A	B	C	D	E	F	G	H
序号	触发条件	触发参数1（整型）	触发参数3（浮点型）	触发参数4（浮点型）	触发参数2（整型）	描述	优先级
1	延时	时间：T秒	-	-	-	BUFF添加到玩家BUFF队列中，延迟T秒生效，如果T＜持续BUFF持续时间，则不会生效	1期完成
2	立即	-	-	-	-	当buff生效时触发	1期完成
3	持续	时间：T秒	-	-	-	buff开始后生效T秒	1期完成
4	周期	时间：T秒	-	-	-	在生命周期内每X秒触发1次	1期完成
5	移除时	-	-	-	-	当buff移除时触发	1期完成
6	攻击时	-	-	-	-	角色发动攻击时触发效果	待规划
7	攻击命中时	-	-	-	-	发出攻击且敌方受到该攻击伤害时触发效果	1期完成
8	受击时	-	-	-	-	角色受到伤害时触发效果	1期完成
9	属性变化	属性ID	百分比	-	-	buff生效期间，角色属性类型损失A%时触发	待规划
10	属性对比	属性1ID	百分比P1	百分比P2	属性2ID	buff生效期间，当属性1值*P1＜属性2值*P2时触发	1期完成
11	昼夜		时间1	时间2		在游戏时一天0-24h内的指定时间段[t1,t2]	1期完成
	.....						

效果类型

根据不同的效果类型采用不同的处理方式

- 基础属性修改
- 状态修改
- 移动锁

序号	状态效果	参数1(int)	参数2(int)	参数3(float)	参数4(float)	参数5(string)	说明	备注	优先级
1	属性变化	属性ID	参照属性ID	±固定值V	±百分比P		修正后的属性 = 修正前的属性 + 参照属性ID * P + V	对目标造成属性变化（+或*）	1期完成
2	攻击	攻击属性ID		值		目标类型		以指定攻击类型的X攻击力对目标造成伤害（非真实伤害）	待规划
3	真实伤害	攻击属性ID						对目标造成角色指定类型攻击力X%的真实伤害	待规划
4	属性变化效率	属性ID			±百分比P	增/减/全部	修正后的恢复量 = 修正前的恢复量 * P	对指定的属性，增/减的情况下，吸收P%	1期完成
5	禁体力恢复							体力不可自然恢复	待规划
6	眩晕							角色无法做出行为	待规划
7	禁止驱散							角色无法驱散buff	待规划
8	触发	buffID	概率A					有A%的概率触发其他buff	待规划
9	光环	buffID	Search表ID	半径A	概率B	目标类型		在角色圆形半径A范围内，有B%的概率对目标附加一个buff	1期完成
10	驱散	组ID	概率A					有A%的概率驱散1层属于组ID的buff	待规划
11	元素影响	X元素ID	范围A			目标类型		在范围A内对所有注定目标中类型发出一次X元素判定	待规划
12	状态改变	X状态ID	概率A					有A%的概率使目标状态为X状态直到buff移除	待规划
13	状态修改	状态ID	状态值					将状态ID设定指定的状态值	1期完成
14	奔跑锁	1						1 锁住，无法奔跑 0 解锁	1期完成

移除类型

- 生命周期移除
- 属性条件触发移除

序号	驱散条件	参数1（整型）	参数2（整型）	参数3（浮点）	参数4（浮点）	描述	备注
1	生命周期					buff在生命周期到达后失效（N=-1表示永久生效）	1期完成
2	累积层数	N				buff叠加到N层时移除	待规划
3	属性对比	属性ID1	属性ID2	乘数N1	乘数N2	属性ID1*N1＜=属性ID2*N2时移除	1期完成
4	受击时					目标受击时移除	待规划
5	攻击时					目标发起攻击时移除	待规划
	.....						

buff 的Pb 结构

记录buff所需的时间戳，层数、效果列表等等

```
message BuffItem
{
    required uint32 id                                      = 1;//buff id
    optional float step_time                                = 2;//间隔时间
    optional float keep_live_time                           = 3;//生命时间
    optional float add_time                                 = 4;//buff 添加时间
    optional float start_time                               = 5;//buff 开始时间
    optional float last_execute_time                        = 6;//上一次生效时间
    optional float delay_time                               = 7;//延迟生效时间
    optional uint32 buff_count                              = 8;//buff 层数
    optional bool is_delete                                 = 9;//是否需要删除
    repeated BuffEffectItem buff_effect_merge_list          = 10;//效果列表
    optional float delete_time                              = 11;//被删除的时间
}
```

buffEffect 的 Pb 结构

记录buff效果所需的时间戳，层数等等

```
message BuffItem
{
    required uint32 id                                      = 1;//buff id
    optional float step_time                                = 2;//间隔时间
    optional float keep_live_time                           = 3;//生命时间
    optional float add_time                                 = 4;//buff 添加时间
    optional float start_time                               = 5;//buff 开始时间
    optional float last_execute_time                        = 6;//上一次生效时间
    optional float delay_time                               = 7;//延迟生效时间
    optional uint32 buff_count                              = 8;//buff 层数
    optional bool is_delete                                 = 9;//是否需要删除
    repeated BuffEffectItem buff_effect_merge_list          = 10;//效果列表
    optional float delete_time                              = 11;//被删除的时间
}
```