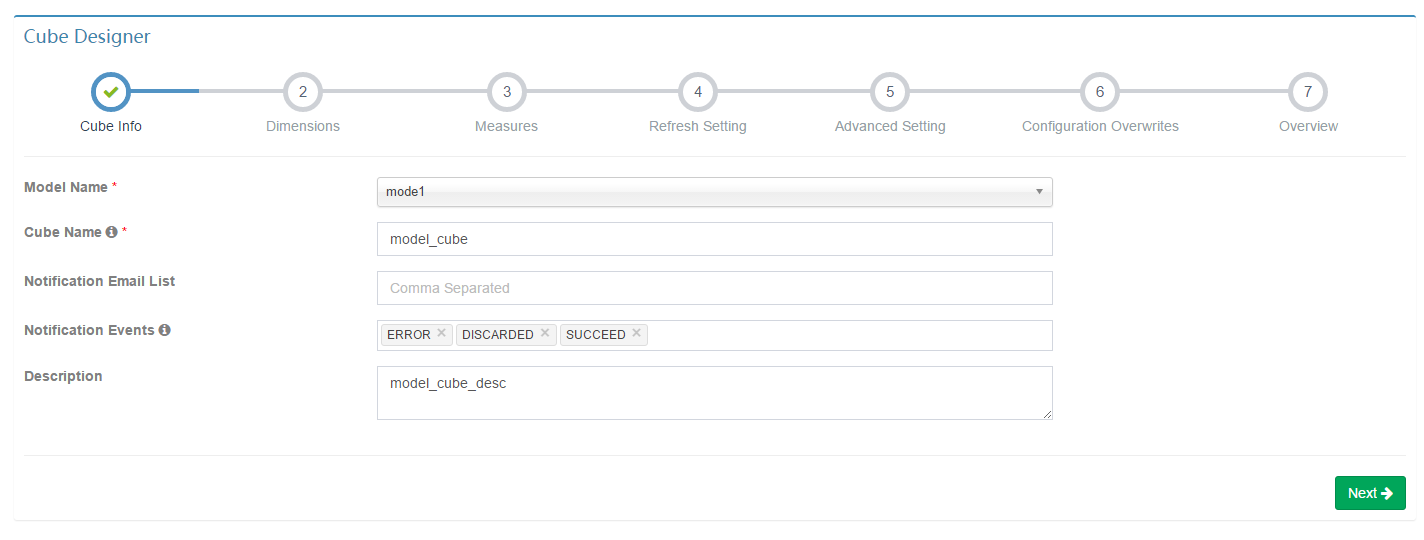
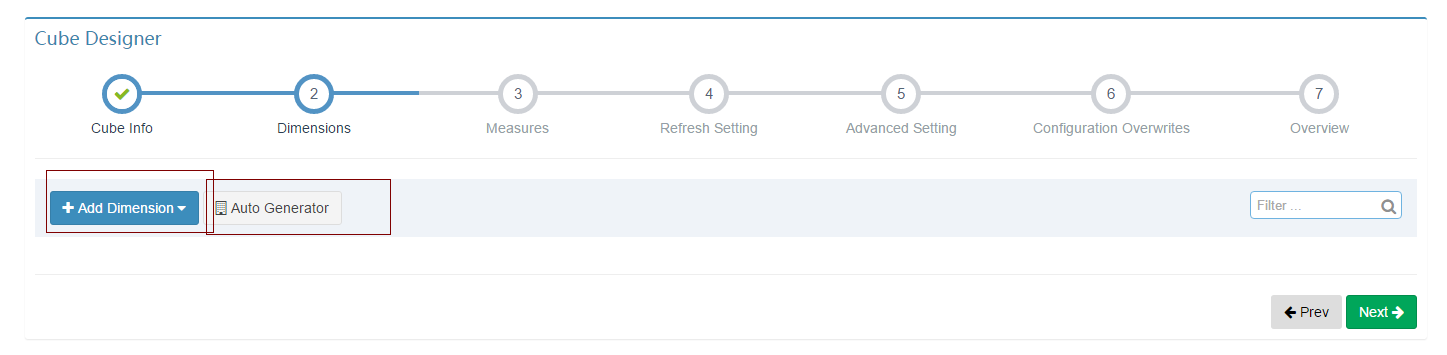
1. 原理

1.cube被model引用,而且一个cube只能对应一个model

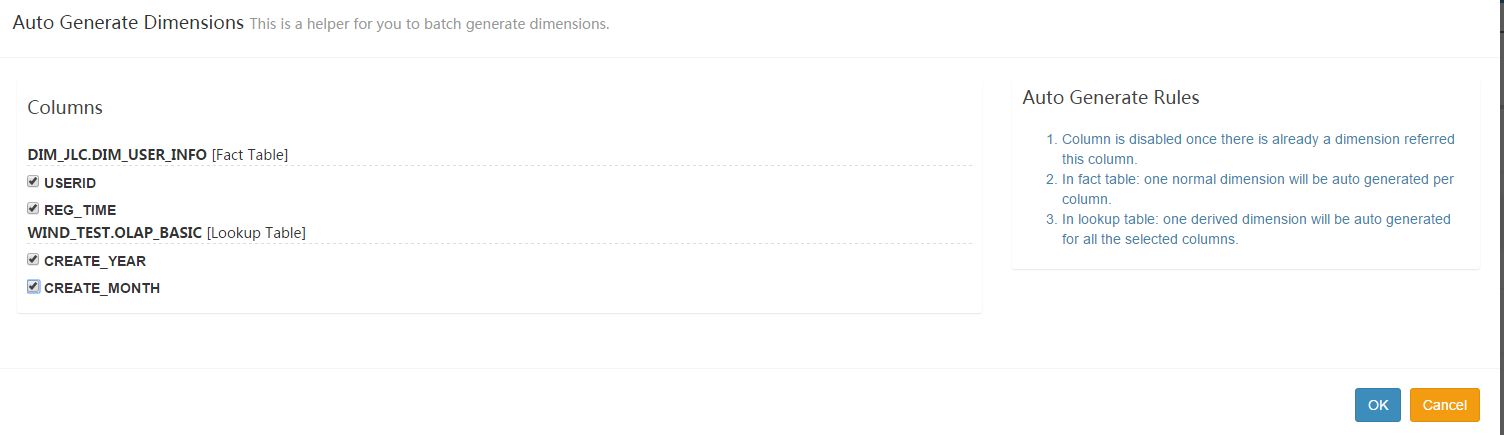
2.第一页



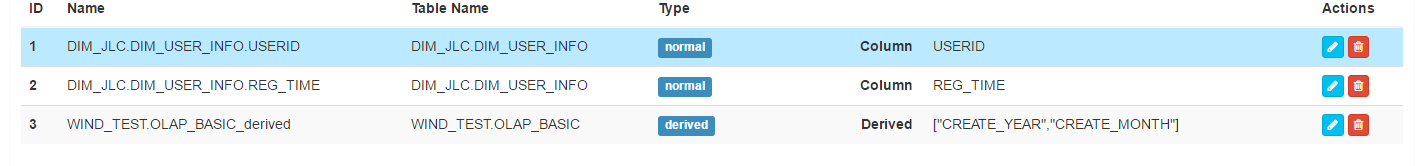
3.第二页—添加维度



选择auto generator,将该model关联的两个表所有的维度都展示出来进行选择

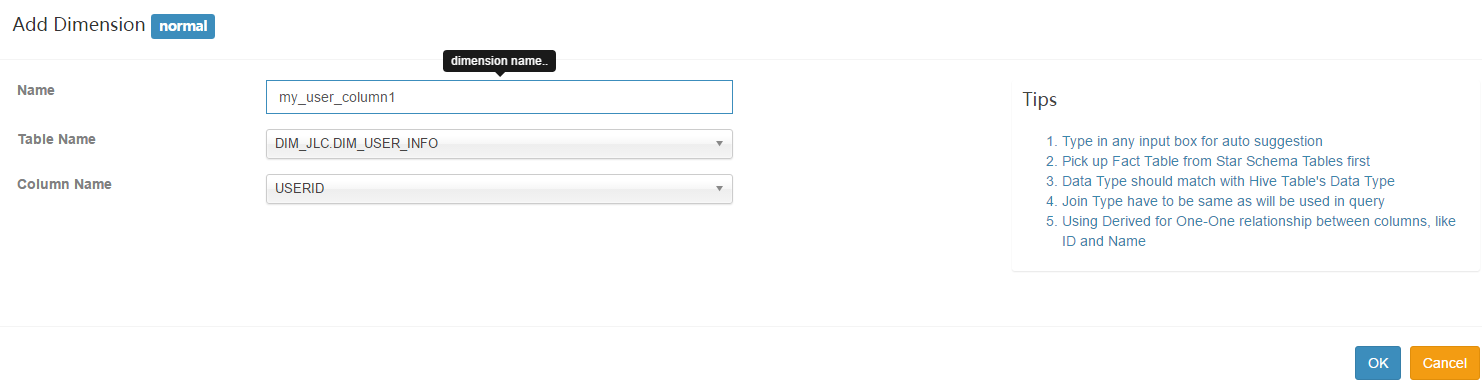


结果

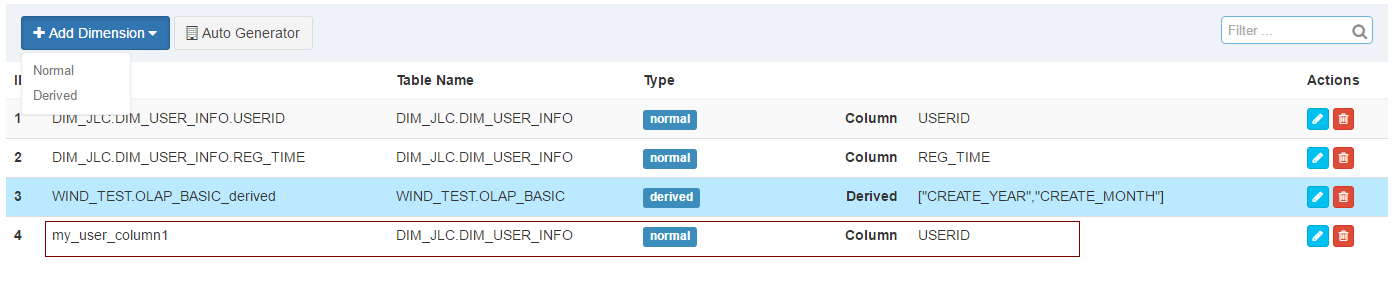


也可以手动创建

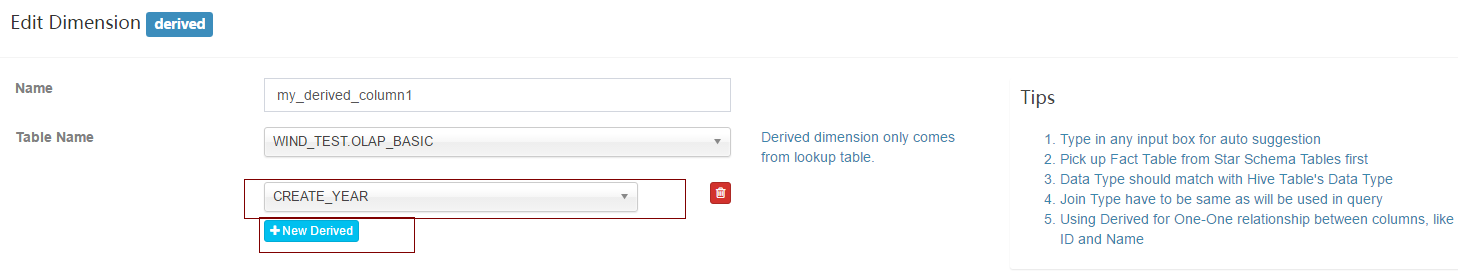
比如创建normal



结果:

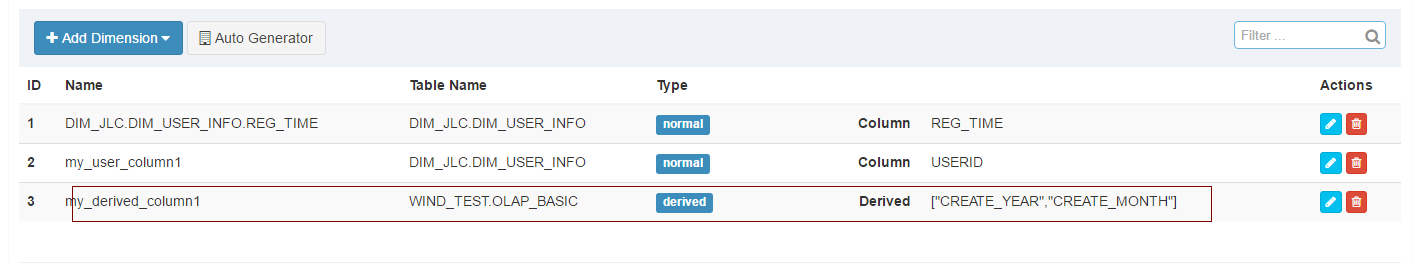


#### 创建derived

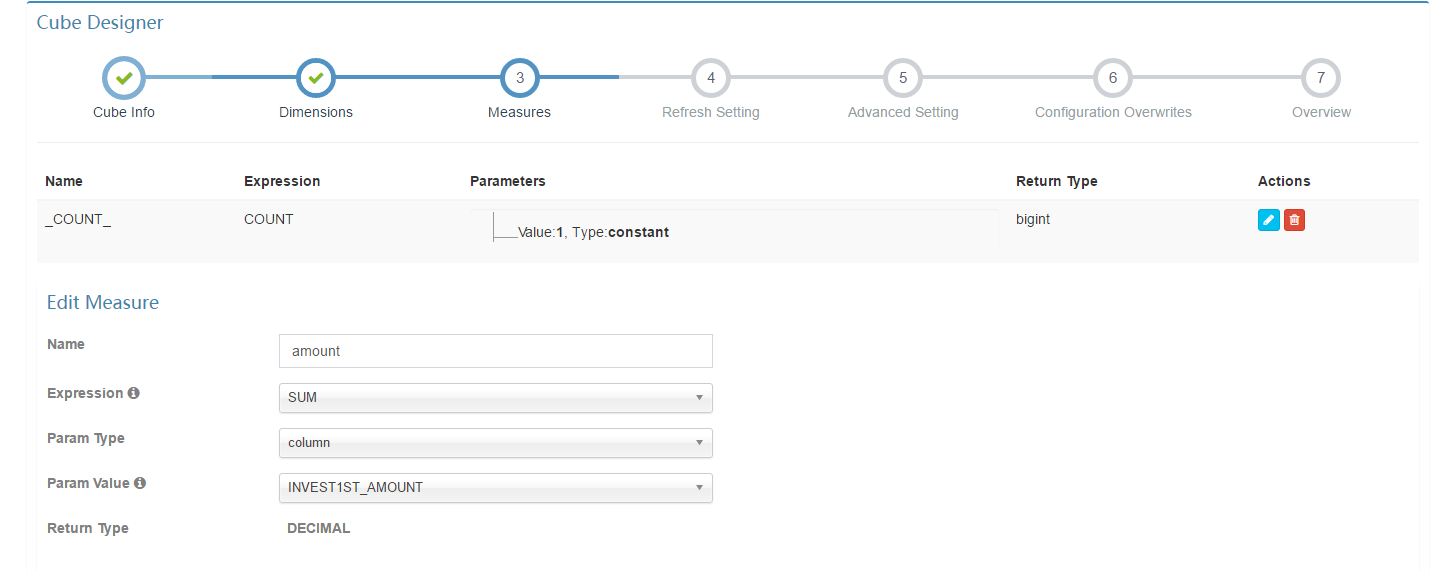


选择非事实表,则都称之为derived,选择要创建的列

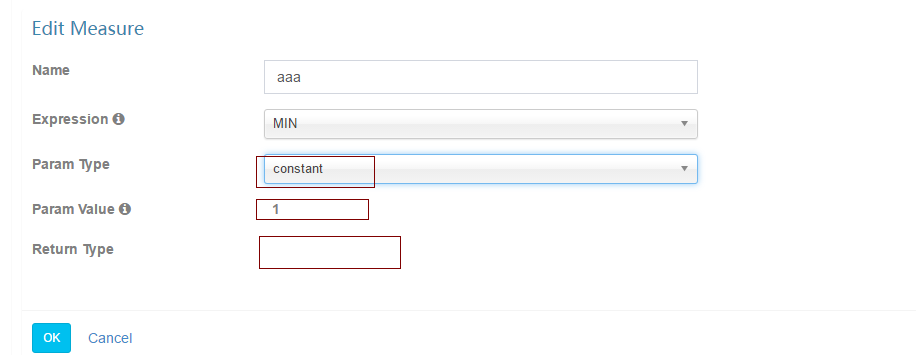
结果



4.第三页 添加度量



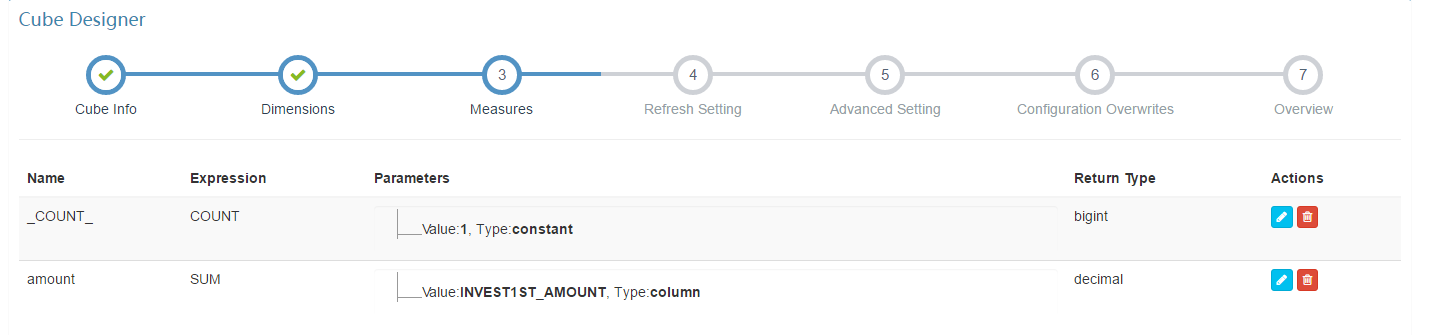
对model中度量的列进行选择,并且指定添加的度量算法



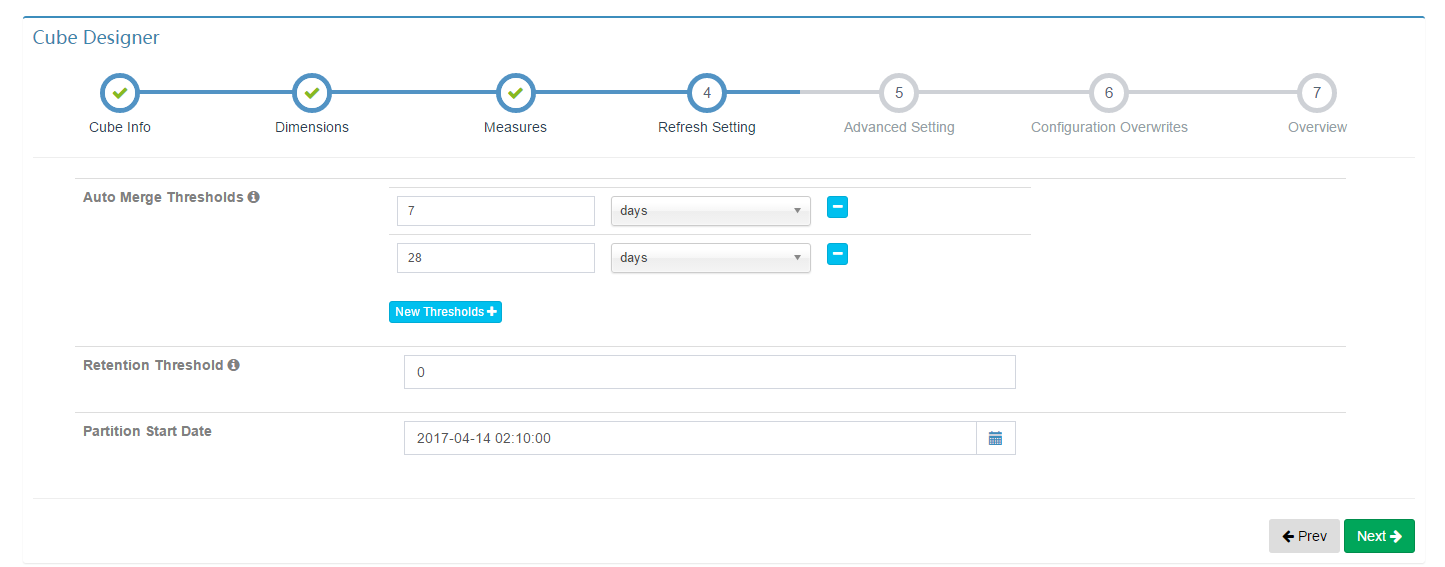
选择常量的时候,参数值就不能修改的变成1,返回类型变成空

特殊的添加方式,后续再修改的地方进行展现

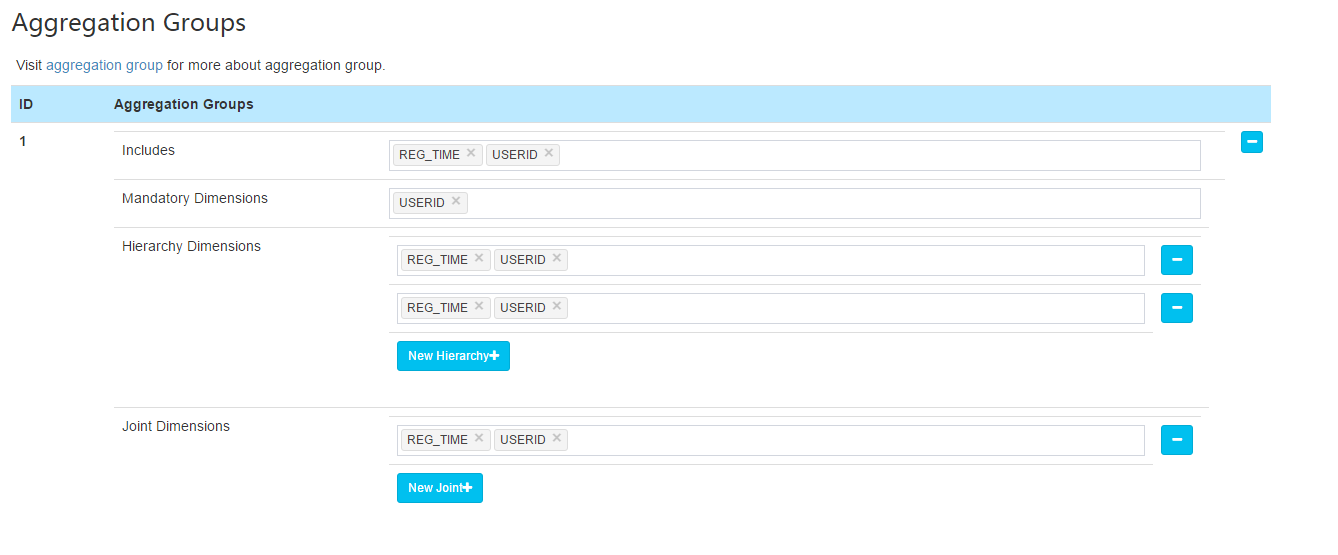
结果

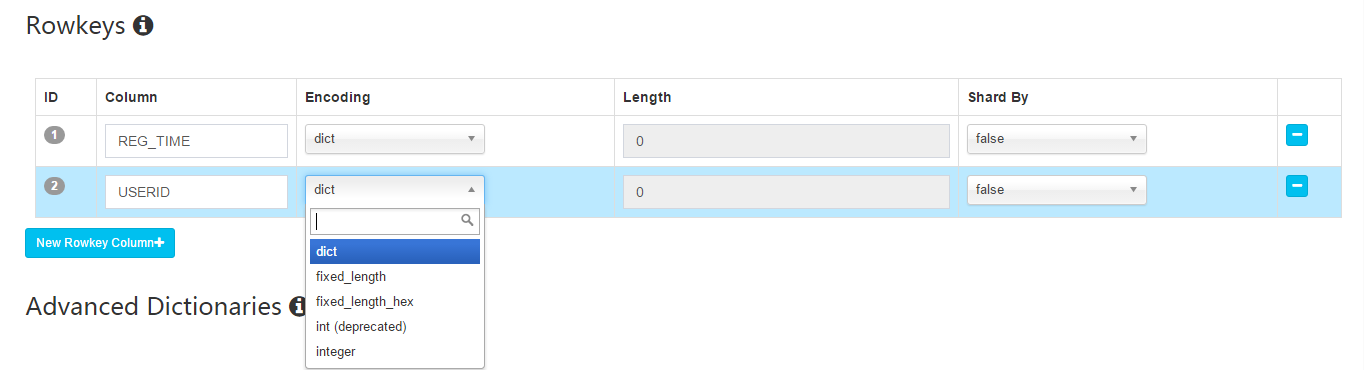


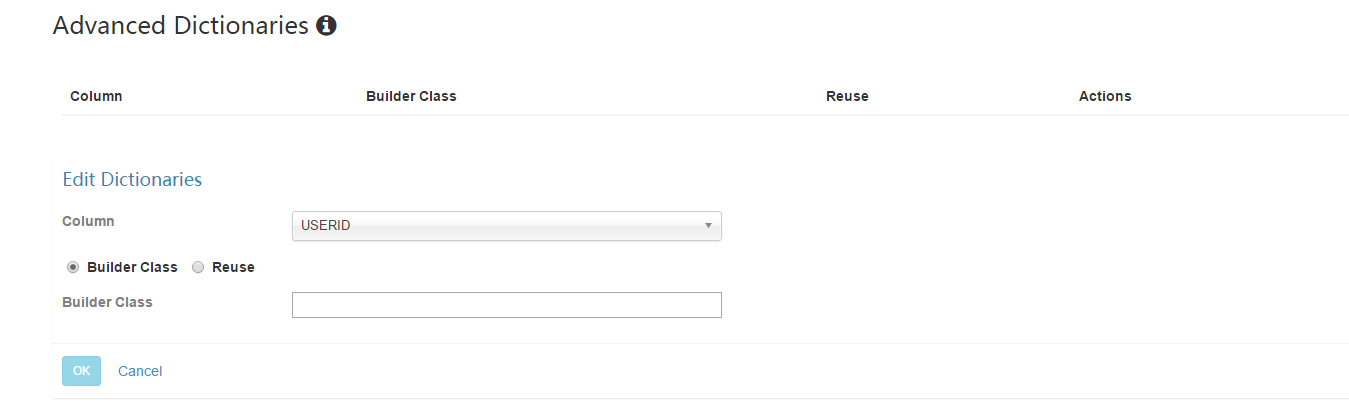
5.第4个页面



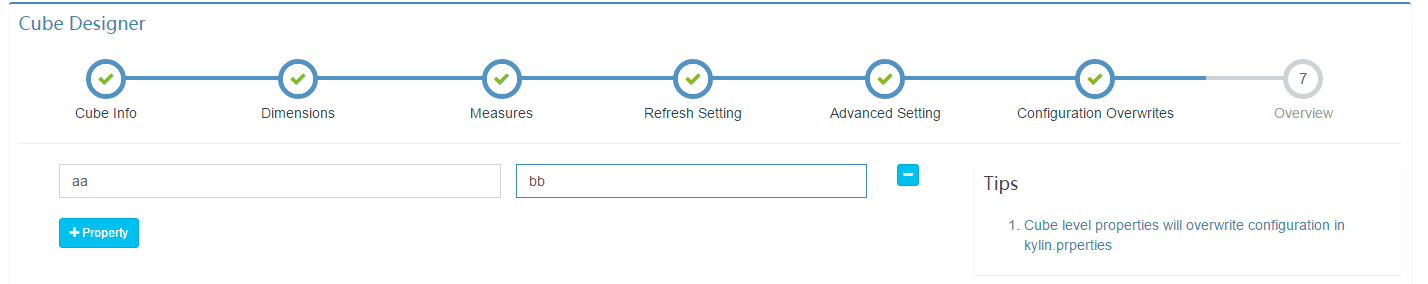
6.第5个页面





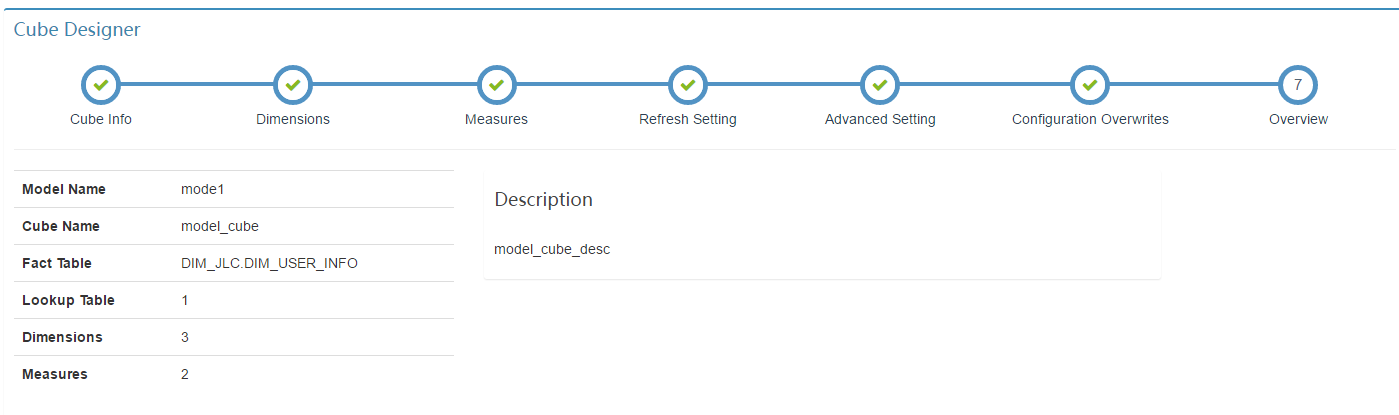


7.第6个页面



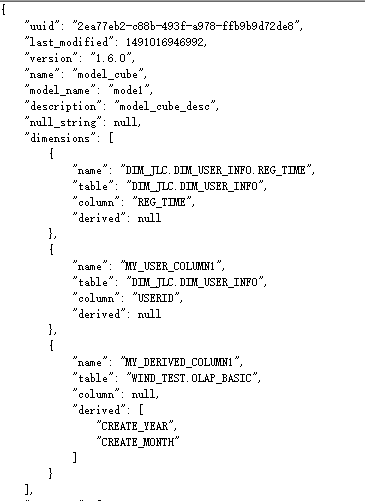
增加属性信息

8.汇总页面

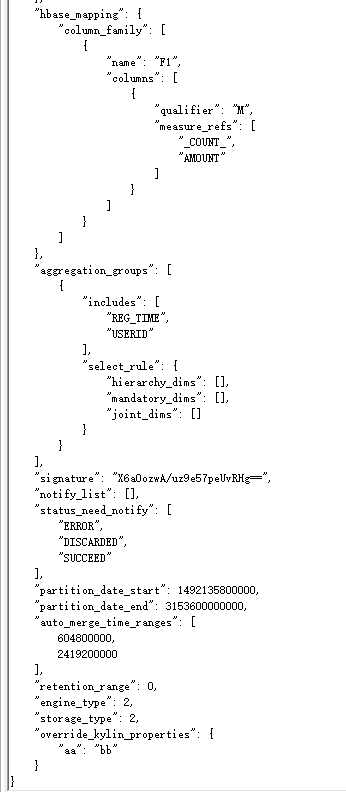


说明该cube实施表是哪个hive表,以及有多少个关联的lookup表,以及选择model中多少个维度和度量等信息

最终json







二、CubeDesc对象

1.第一个页面

添加cube的名字以及描述 在哪个model上建立cube 邮件通知事件 以及通知列表

private String name;

private String description;

private String modelName;//该cube所属model

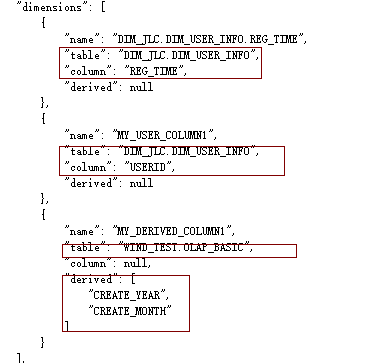
private List<String> notifyList;//邮件通知列表

private List<String> statusNeedNotify = Collections.emptyList();//什么情况下会发送邮件通知

2.第二个页面 添加维度

private List<DimensionDesc> dimensions;

DimensionDesc对象内容以及解析的json的demo



可以看到如果是事实表,则column有内容,derived没内容,反之相反.而且derived是关联的所有字段的集合

DimensionDesc对象内容:

private String name;//该维度属性名字,一般为库.表.列 或者库.表.derived,或者自定义的名字

private String table;//该维度属于哪个表

private String column;//表的具体列名字

private String[] derived;//如果是derived的时候,这个是一组列的集合,表示这个维度不是一个单独的列

初始化的时候会更新这2个字段内容

private TableDesc tableDesc;//table对应的表对象

private JoinDesc join;//找到该model中该表的join关系

private TblColRef[] columnRefs;// fact\_table中对应的列映射关系,该映射是on语法使用的列集合

3.度量页面

private List<MeasureDesc> measures;

生成的json



MeasureDesc对象属性内容:

private String name;//为该度量设置一个别名

private FunctionDesc function;//真正的度量信息,包含函数 参数(字段,常数) 返回值等信息

private String dependentMeasureRef;

依赖的具体的函数后续再关注。

4. 设置merge周期、丢弃周期、分区信息

5.第5个设置页面

private List<AggregationGroup> aggregationGroups; 设置一组聚合组,不同组内维度是可以重复设置的。

四、Cube的初始化工作

1.获取事实表对象

2. DimensionDesc的init方法,设置每一个维度对应的join对象以及维度所在表对象

3.统计cube中使用了哪些列对象集合,以及维度的列集合

TableController

五、规则

1. Mandatory 必须要有的维度列

有1个维度,就减少一半的cubeid数,因为一个维度固定,相当于2的N次方变成2的n-1次方

2．Joint 联合规则

如果两个以上的维度设置成一组joint,则任何有效的cuboid组合将是不包含这些维度中的任意维度或者全部包含这些维度的组合,换句话说,joint这些维度组合总是在一起,不能单独分开。

当cube设计师确定一些维度总是一起被查询的时候,使用joint是非常有用的.

假设有20个维度,前10个经常使用,后10个不经常时候,通过将后10个维度进行joint,因此将会将cubeid数据数量从2的20次方变成2的11次方

优化效果：将N个维度设置为联合维度，则这N个维度组合成的cuboid个数会从2的N次方减少到1。即一个joint最后就当作一个维度进行运算,因此就是1个cubeid,但是该joint由n个维度组成,即原始是2的n次方。

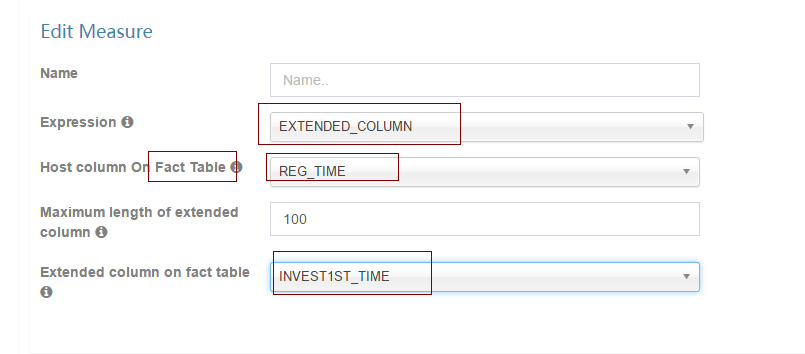
3.层次维度

使用场景：年，月，日；国家，省份，城市这类具有层次关系的维度。

优化效果：将N个维度设置为层次维度，则这N个维度组合成的cuboid个数会从2的N次方减少到N+1。

即年，月，日 、 年，月，、 年 、 无

4. EXTENDED\_COLUMN 扩展,属于度量中设置的优化内容



在OLAP分析场景中，**经常存在对某个id进行过滤，但查询结果要展示为name的情况**，比如user\_id和user\_name。这类问题通常有三种解决方式：

a. 将ID和Name都设置为维度，查询语句类似select name, count(\*) from table where id = 1 group by id,name。这种方式的问题是会导致维度增多，导致预计算结果膨胀；

b. 将id和name都设置为维度，并且将两者设置为联合。这种方式的好处是保持维度组合数不会增加，但限制了维度的其它优化，比如ID不能再被设置为强制维度或者层次维度；

c. 将ID设置为维度，Name设置为特殊的Measure，类型为Extended Column。这种方式既能保证过滤id且查询name的需求，同时也不影响id维度的进一步优化。

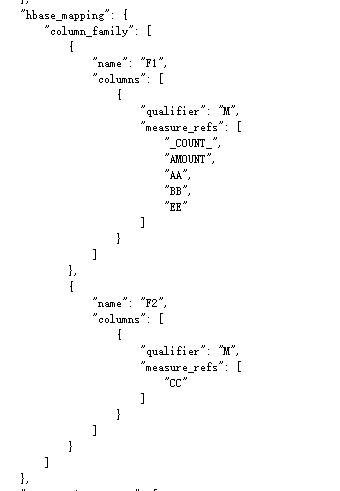
所以此类需求我们推荐使用 Extended Column。

为什么继承和joint列表中,不允许两组不同的继承或者joint关系使用同一个维度

为什么hierarchyDims和jointDims中维度不能有重合的

为什么mandatoryDims与hierarchyDims或者jointDims有重合的要发送警告

高级度量设置



"measures": [

{

"name": "\_COUNT\_",

"function": {

"expression": "COUNT",

"parameter": {

"type": "constant",

"value": "1",

"next\_parameter": null

},

"returntype": "bigint"

},

"dependent\_measure\_ref": null

},

{

"name": "AMOUNT",

"function": {

"expression": "SUM",

"parameter": {

"type": "column",

"value": "INVEST1ST\_AMOUNT",

"next\_parameter": null

},

"returntype": "decimal"

},

"dependent\_measure\_ref": null

},

{

"name": "AA",

"function": {

"expression": "EXTENDED\_COLUMN",

"parameter": {

"type": "column",

"value": "REG\_TIME",

"next\_parameter": {

"type": "column",

"value": "INVEST1ST\_TIME",

"next\_parameter": null

}

},

"returntype": "extendedcolumn(100)"

},

"dependent\_measure\_ref": null

},

{

"name": "BB",

"function": {

"expression": "TOP\_N",

"parameter": {

"type": "column",

"value": "INVEST1ST\_AMOUNT",

"next\_parameter": {

"type": "column",

"value": "INVEST1ST\_AMOUNT",

"next\_parameter": null

}

},

"returntype": "topn(100)",

"configuration": {

"topn.encoding.INVEST1ST\_AMOUNT": "dict"

}

},

"dependent\_measure\_ref": null

},

{

"name": "CC",

"function": {

"expression": "COUNT\_DISTINCT",

"parameter": {

"type": "column",

"value": "INVEST1ST\_AMOUNT",

"next\_parameter": null

},

"returntype": "hllc(14)"

},

"dependent\_measure\_ref": null

},

{

"name": "EE",

"function": {

"expression": "RAW",

"parameter": {

"type": "column",

"value": "INVEST1ST\_AMOUNT",

"next\_parameter": null

},

"returntype": "raw"

},

"dependent\_measure\_ref": null

},

{

"name": "GG",

"function": {

"expression": "COUNT\_DISTINCT",

"parameter": {

"type": "constant",

"value": "1",

"next\_parameter": null

},

"returntype": "hllc(12)"

},

"dependent\_measure\_ref": null

}

],

对应的页面

