# 关于二开对总账核算维度的指导意见

|  |  |  |  |
| --- | --- | --- | --- |
| **Ver. No** | **日期** | **编制/修订** | **审核** |
| V1.0 | 2015-08-07 | 刘晓辉 | 张荣华 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 1 编写目的

我们在给客户处理问题时，发现有不少2开的客户对总账核算维度的处理有问题，所以就有了下面的篇幅。我们假设读者对基础资料这一块完全不了解，如果了解可以略过第2节。

## 2 必备的基础知识

### 2.1 基础资料控制类型

在K3/Cloud里，基础资料控制类型分为3种，即私有型、分配型和共享型。如下图：



图2.1基础资料控制类型

私有型，顾名思义私有，只适用于当前或组织，与外界隔离；

共享型，即共享，适用于所有，没有组织隔离的概念。如：省、市、区。

其它相关的知识可以参考产品论坛，搜索相关关键字。下面主要介绍分配型相关的知识。

### 2.2 基础资料控制策略

我们先看基础资料控制策略，如下图所示。既然可以分配，那么就有源和目标。源即创建组织，目标是分配目标组织，即使用组织。

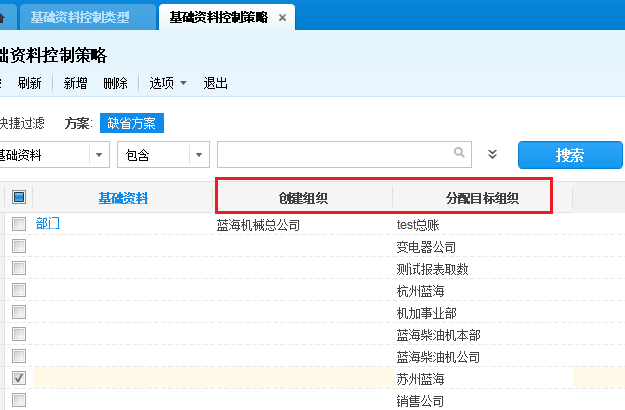


图2.2 基础资料控制策略

具体客户基础资料如何分配，如下图所示。

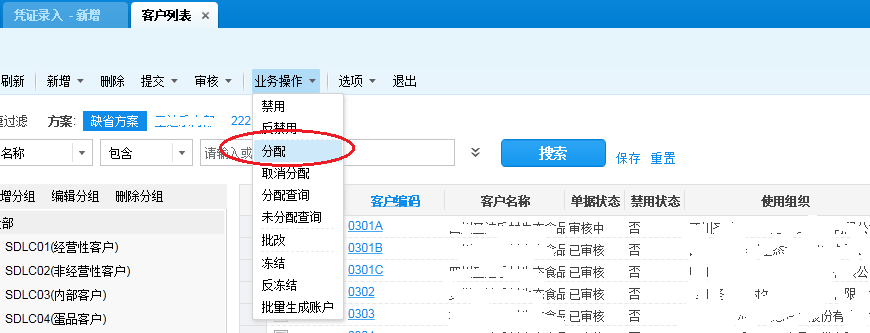


图2.3 客户列表

## 3 核算维度相关知识

我们先看一下总账凭证录入核算维度入口，如下图所示。



图3.1 凭证录入

凭证一旦录入客户、部门信息之后，科目余额表等相关的报表根据这些维度就可以找到想要的数据。作为开发人员，我们如何快速判断“客户”基础资料控制策略类型呢？用下面的SQL即可。红框中的那列值（FSTRATEGYTYPE）为 “2” 即为分配型。

--查看基础资料[客户]所属类型

select \* from T\_META\_BASEDATATYPE where fnumber='BD\_Customer'

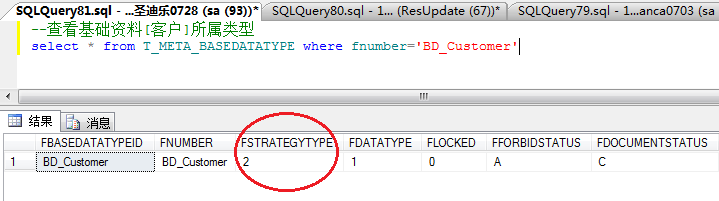


图3.2 查看基础资料类型

所有的核算维度从下面的表可以知晓，如下图所示。

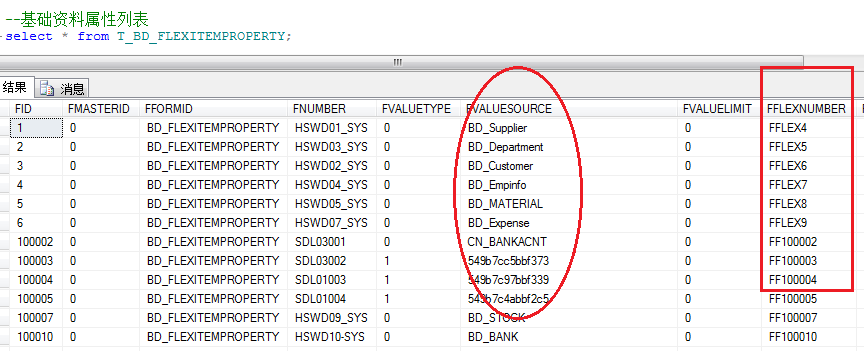


图3.3 查看核算维度属性表

具体核算维度明细从下面的表知晓（俗称横表），如下图所示。

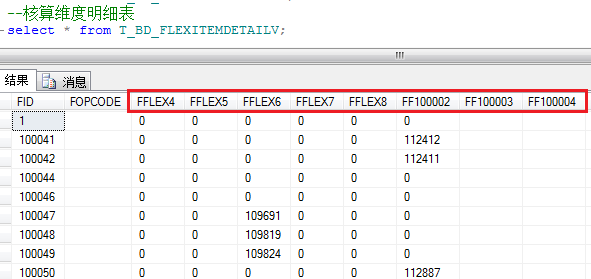


图3.4 查看核算维度明细表

该表与上面核算维度属性表形成行列关系。当增加某一维度，属性表增加一行，横表自动增加一列；同样，删除的时候，属性表删除一行，横表会删除对应列（无维度数据方可删除），如下图所示。

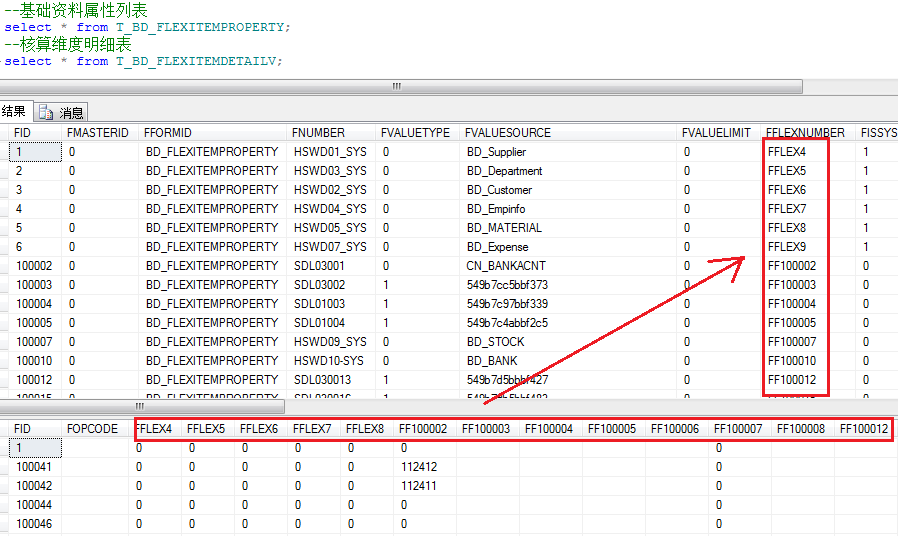


图3.5 查看核算维度属性和明细表之间行列关系

当了解核算维度相关的表结构后，我们来看下核算维度明细表（横表）中存放的数据。基础资料如果是私有或共享型的，那么这个ID值肯定是唯一的，存放于横表中。如：列FFLEX4（假设它属于私有或共享基础资料），这一列对应的值应该都是这个唯一主键。如果FFLEX4是分配型的呢？不一定。我们先来看下基础资料供应商表结构。

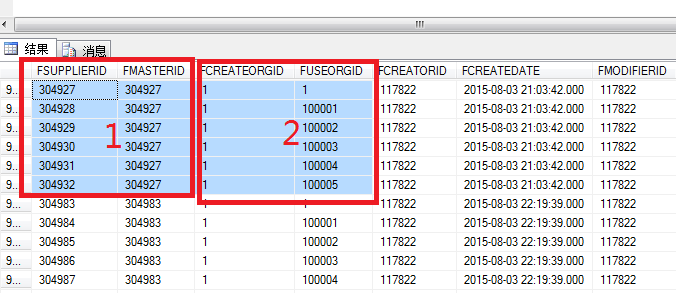


图3.6 查看供应商基础资料表

里面有2组数据需要我们注意。

第一组：FSUPPLIERID(主键)、FMASTERID（可以理解为主ID）

第二组：FCREATEORGGID（创建组织ID）、FUSEORGID（使用组织ID）

从图中，我们不难观察到，当我们首次创建一个“供应商”基础资料时，FSUPPLIERID= FMASTERID，FCREATEORGGID= FUSEORGID，它们的值两两相等；而分配给其它组织，FMASTERID和FCREATEORGGID不变，而FSUPPLIERID变了（因为是主键必须变），FUSEORGID（使用组织）变了，这个为什么变应该不难理解。

那么，我们核算维度明细表（横表）里的值为什么要放FMASTERID，而不是FSUPPLIERID（主键ID）？

我们来个假设，如果是存放FSUPPLIERID（主键ID），那么我们凭证录入选择维度的时候就可以有多个相同的名称、编码等一样的信息（**此处务必理解：前提是使用多组织，因为分配，所以F8能选到分配过来的相同名称、相同编码的基础资料信息**），选完后系统自动保存到核算维度明细表（横表）中，然后我们去查询科目余额表。你会发现相同的核算维度编码/名称没有合并。为什么没有合并？因为核算维度的ID不一样，虽然编码名称一样。

凭证分录表（T\_GL\_VOUCHERENTRY）中的FDETAILID和科目余额表（T\_GL\_BALANCE）中的FDETAILID均来源于核算维度明细表中的FID。核算维度明细表如果存放的是基础资料的主键ID，那么分配过来的就不能合并成一个FID，体现在科目余额表中必然有多个不一样的FID，但背后的基础资料信息是一样的，只不过是分配的而已。这样的结果显然不是科目余额表想要的。

## 4 处理办法

**为了让核算维度不出问题，我们2开必须要按照下面两种方法选其一执行，否则帐表的数据将不对。**

### 4.1 F8事件过滤时加上筛选条件

我们可以重写该事件，最终的目的还是取FMASTERID的值，详细代码如下：

public override void BeforeF7Select(BeforeF7SelectEventArgs e)

{

//判断是否选择核算维度，否则单据上一切F7都会调用这里

//这里的值”$$FDETAILID“要换上2开单据上的维度的ID，如何查看？调试进来即可查看，通常是表单的ID加上FDETAILID

if (!e.FieldKey.StartsWith("$$FDETAILID")) return;

base.BeforeF7Select(e);

BaseDataField baseDataField = e.BaseDataField as BaseDataField;

//获取相关使用组织

List<long> lstOrg = GetOrgList(baseDataField);

if (lstOrg != null && lstOrg.Count > 0)

{

//获取基础资料类型

int baseDataControlType = OrganizationServiceHelper.GetBaseDataType(this.Context, baseDataField.LookUpObject.FormId);

//如果是分配型，前面讲到的是 2

if (baseDataControlType == 2)

{

//是否组织隔离（多组织才用）

((ListShowParameter)e.DynamicFormShowParameter).IsIsolationOrg = false;

//获取相关的表、主键、组织等名称

string pkFieldName = baseDataField.LookUpObject.PkFieldName;

string masterIdFieldName = baseDataField.LookUpObject.MasterIdFieldName;

string tableName = baseDataField.LookUpObject.TableName;

string userFieldName = baseDataField.LookUpObject.OrgFieldName;

//增加过滤条件，进行数据筛选 （已审核 FDOCUMENTSTATUS='C'、非禁用 FFORBIDSTATUS='A'）

//这里如果是单组织 pkFieldName = masterIdFieldName 就可以了

e.ListFilterParameter.Filter = e.ListFilterParameter.Filter.JoinFilterString(

string.Format("{0}={1} AND {1} IN (SELECT DISTINCT {1} FROM {2} WHERE FDOCUMENTSTATUS='C' AND FFORBIDSTATUS='A' AND {3} IN({4}))",

pkFieldName, masterIdFieldName, tableName, userFieldName, string.Join(",", lstOrg)));

}

}

}

当然，我们还可以反编译组件Kingdee.K3.FIN.GL.Business.PlugIn，查看总账凭证录入选核算维度时重写类ShowFlexGLCustomPageEdit的实现方法。

### 4.2 直接维护核算维度明细表

假如上面的过滤因为某种原因无法实现，那么我们可以直接维护这张表。在插入这张表时，如果基础资料是私有、共享型的，插入便可；如果是分配型的，取到FMASTERID插入；如果第一次无法取到FMASTERID，那么后面再做一步更新的动作。另外，插入前需要判断维度是否已经存在（可以通过FCALCOL来判断唯一性，FCALCOL是个计算列，各维度列组合成字符串连接，字段为空的加上空格），因为该表有联合主键约束PK\_BD\_FLEXITEMDETAILV。

还要注意的是，核算维度明细表主键FID可以从种子表Z\_BD\_FLEXITEMDETAILV读取。Sql Server 的读取方式为：先插入一条任意记录然后删掉，获取当前的起始数；Oracle 直接读取该表的Sequence，切不可自己单独处理。

## 5 二开实例解析

某伙伴进行了凭证摊销的二次开发，界面如下图所示。

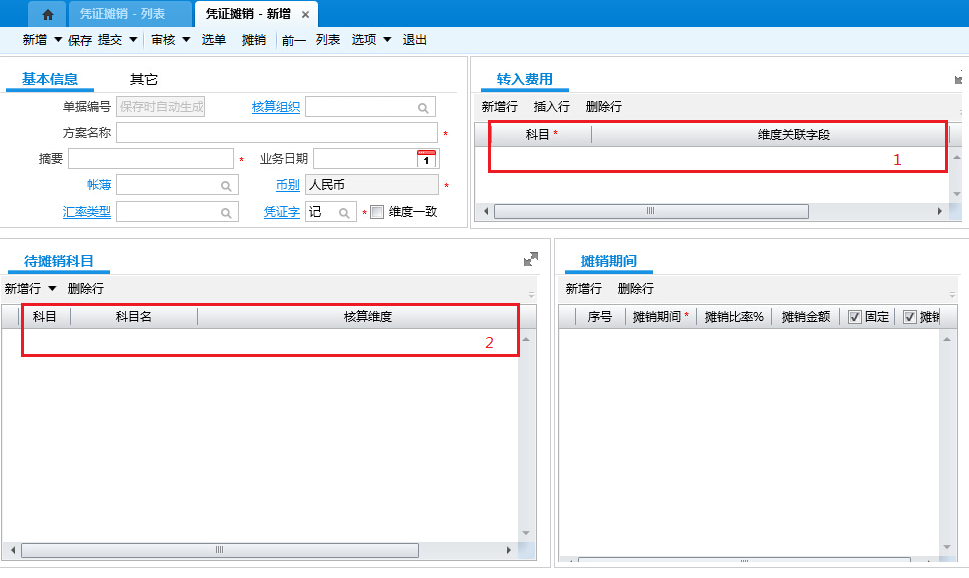


图5.1 凭证摊销界面

界面上有两处根据科目选核算维度，类似于总账凭证录入。从客户的插件代码来看，前面的两种处理办法都没有实现。结果就是每月需要我们总部修复维度数据，多痛苦的一件事情。所以，作为2开的人员，我们必须了解这里面的实现原理，只有真正的理解，才不会出错。测试的时候，也必须充分。

节选部分代码。



图5.2 代码片段

代码片段如下：

/\*\* Kingdee注释-------------------------------------------------------------------------------------1

\* 代码从这里开始直接插入到凭证分录表，先不管代码写的多不规范。我们看【Kingdee注释2】代码，取核算维度的值，string FDETAILID = dsZr.Tables[0].Rows[i]["F\_LIN\_FLEXZR"].ToString();这里的维度FDETAILID是借助界面弹性域中的维度控件自动产生的维度值，这个值如果没有处理，默认是取基础资料的主键，而非FMASTERID。如果F8事件过滤时没有处理，这样写还不够，还应该去维护核算维度明细表，此时核算维度明细表的FID等于这里的FDETAILID，由此作为条件，从核算维度明细表取到具体的基础资料的主键，然后将之修改成FMASTERID。

\* \*/

string sqlTmpEntry = @"INSERT INTO T\_GL\_VOUCHERENTRY

([FVOUCHERID],[FENTRYID],[FEXPLANATION],[FACCOUNTID],[FDETAILID]

,[FAMOUNTFOR],[FAMOUNT],[FCURRENCYID],[FEXCHANGERATETYPE]

,[FEXCHANGERATE],[FDC],[FSETTLETYPEID],[FSETTLENO]

,[FDEBIT],[FCREDIT],[FCASHFLOWITEM],[FISMULTICOLLECT],F\_LIN\_SourceBillNo,F\_LIN\_SourceId,F\_LIN\_SourceEntryId,fentryseq) VALUES ('" + FVOUCHERID + "',{0},'" + dsHead.Tables[0].Rows[0]["F\_LIN\_ZHAIYAO"].ToString() + "',{1},{2}" + " ,{3},{4}," + dsHead.Tables[0].Rows[0]["FCURRENCYID"].ToString() + ",'" + dsHead.Tables[0].Rows[0]["FEXCHANGERATETYPE"].ToString() + "'" + ",1,{5},0,''" + ",{6},{7},0,1,'" + dsHead.Tables[0].Rows[0]["FBILLNO"].ToString() + "'," + dsHead.Tables[0].Rows[0]["FID"].ToString() + ",{8},{9});";

#region 转入

int iZr = dsZr.Tables[0].Rows.Count;

for (int i = 0; i < iZr; i++)

{

string FENTRYID = GetId("Z\_GL\_VOUCHERENTRY");

string FACCOUNTID = dsZr.Tables[0].Rows[i]["F\_LIN\_KEMUZR"].ToString();

double dZrBl = Convert.ToDouble(dsZr.Tables[0].Rows[i]["F\_LIN\_RATEZR"]);

string FDETAILID = dsZr.Tables[0].Rows[i]["F\_LIN\_FLEXZR"].ToString(); //Kingdee注释2

//FAMOUNTFOR原币金额=金额\*比率。累计反写

double FAMOUNTFOR = TxJe \* dZrBl / 100;

……

以下附件是插件的完整代码，双击打开。正确的代码见本篇第4节处理办法。

