TotalLINK

产品手册

上海朝识智能科技有限公司

2020 年 8 月

拓展应用

目　录

拓展应用.....2
1 拓展应用说明.....5
2 附加模型设计.....6
2.1 功能调用.....6
2.2 同步模型.....7
2.3 自动预处理模型.....7
2.4 手动预处理模型.....7
2.5 手动预处理外部程序调用.....7
2.6 自动后处理模型.....8
2.7 手动后处理模型.....8
2.8 手动后处理外部程序调用.....9
2.9 附加模型的执行顺序.....9
2.10 其它附加模型.....9
3 关联菜单执行程序.....11
4 附加功能程序.....12
5 附件功能模型实例-调用 RDL.....13
6 多层级数据的附加模型处理.....14
6.1 多级数据模型的主模型设计.....15
6.2 多级数据模型的子模型设计.....15
6.3 关系模型设计.....17
7 多层级数据模型实例.....19
7.1 主模型设计.....20
7.2 附加多层数据模型.....20
8 多数据源合并数据的附加模型处理.....22
8.1 多数据源合并数据的主模型设计.....23
8.2 多数据源合并数据的子模型设计.....24
9 MULTIBLOCKDATA 合并多个数据源.....27
10 附加模型中执行存储过程.....29

11 智能设备控制.....30
12 智能设备控制实例.....32
13 系统邮件发送功能设置.....34
14 模型的默认发送邮件设置.....36
15 邮件发送功能.....38
16 文件导出设置.....39
17 短信平台.....40
18 短信平台的配置.....41
19 短信处理模型.....42
20 附加模型功能.....44
21 多数据源操作.....45
22 临时数据保存设置.....46
23 提示用户保存临时数据.....47
24 创建临时数据表 LINKTEMP.....48
25 巧用临时数据处理.....49
26 使用临时表模型数据设计实例一.....50
27 使用临时表模型模型设计实例二.....55
28 图表的相关关联.....59
29 图表关联参数设置约定.....61
30 银行对账模型配置.....62
31 数据编辑、添加、删除等复杂模型.....64
32 利用添加行添加记录--LINKADDNEWROWS.....66
33 编辑模式下单元格的移动.....68
34 为可编辑列定义选择模型.....69
35 编辑列选择模型实现.....70
36 列选择模型返回多列值.....71
37 数据透视分析.....73
38 自动产生参数列表.....75
39 Function List.....77

文档控制

■ 主要内容

本文整合 LINK 功能列及 LINK 特殊标识，方便开发者统一学习。

■ 更改记录

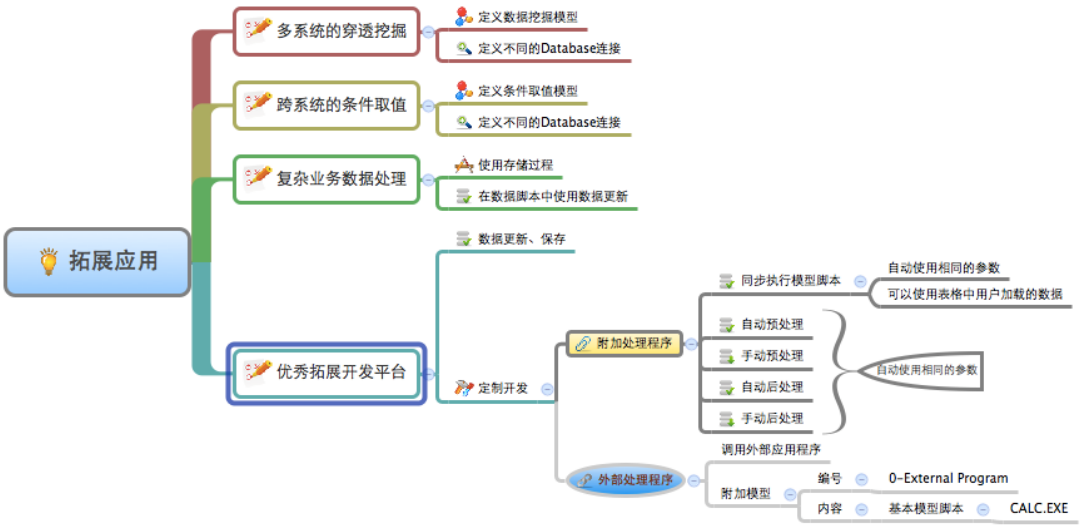
日期	版本	作者	备注
2017-05	1.0	Randy	初始发布
2020-06-05	1.1	Jozey	添加自动生成参数列表
2020-08-28	2.0	Jozey	jozey改版整理
2020-10-14	3.0	Liz	新增functionlist功能

■ 支持版本

非特殊说明的功能，默认前后版本都支持

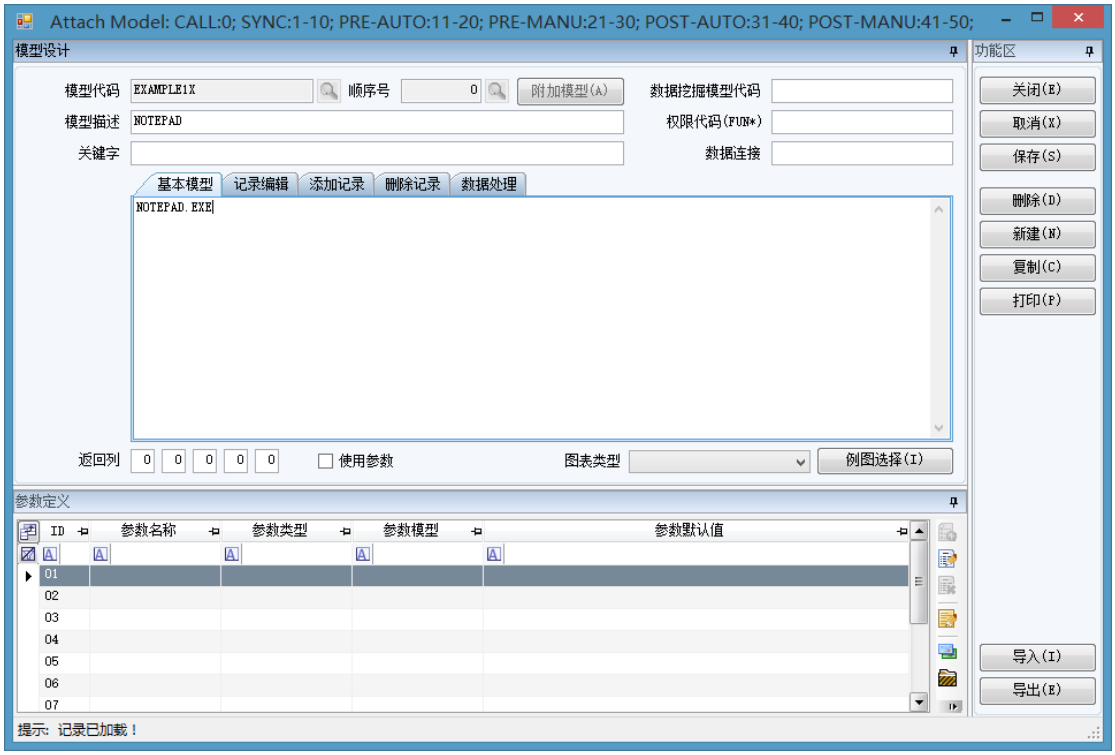
序号	T20更新功能点	T20版本号
1	MULTIBLOCKDATA 合并多个数据源	V 8.2.1.2

1 拓展应用说明



- 作为一款优秀的数据查询及应用开发平台，系统具有复杂的拓展功能
  - 可以实现复杂的多系统穿透查询
  - 可以实现跨系统的条件取值
  - 可以执行复杂的数据处理
  - 可以通过定义复杂的“附加模型组”实现更加复杂的应用
- 系统的此类特性，大大拓展了系统的应用范围
- 有关拓展功能的使用，请参加系统专家开发培训

2 附加模型设计



- 通过“附加模型”设计，可以实现更加强大的数据处理
- 对于每个数据分析模型，在此我们称之为“主模型”，都可以定义多个（组）附加模型
- 附加模型在主数据执行的不同时刻按顺序执行，或者可以手动执行
- 附加处理模型只执行“数据处理”模型中的脚本
- 非常重要的是，附加模型可以在“主模型”使用不同的连接，利用此功能可以同时在不同的业务系统中执行复杂的功能

2.1 功能调用

- 附加模型编号为“0”的模型为功能调用模型
- 当存在功能调用模型时，系统将此模型的基本模型脚本作为“操作系统”的基本命令调用；
- 比如“NOTEPAD.EXE”脚本调用的结果就是打开“记事本”应用程序
- 用户使用此功能可以任意调用 WINDOWS 的应用程序

2.2 同步模型

- 附加模型编号为“1-10”的模型为自动同步处理脚本
- 根据需要，当用户在主模型中操作时，可以同步触发模型编号为“1-10”的模型，比如同步执行编辑保存、删除、添加及全数据处理操作
- 同步模型与主模型使用相同的参数
- 同步模型处理时可以使用当前表格中的数据作为参数
- 注意：同步处理模型不处理程序仅处理当前过滤处理的有效数据行，不处理分组中的数据等特殊的行记录

2.3 自动预处理模型

- 附加模型编号为“11-20”的模型为自动预处理脚本模型
- 执行主数据查询脚本前，系统自动执行自动预处理模型中的脚本
- 自动预处理模型与主模型使用相同的参数
- 注意：自动预处理模型使用当前的行记录

2.4 手动预处理模型

- 附加模型编号为“21-30/121-130”的模型为手动预处理脚本模型
- 当存在此组模型时，系统会激活“手动预处理”功能按钮
- 此功能按钮的标题为编号为 21/121 的模型的描述
- 执行主数据查询脚本前，用于可以手动触发“此组模型”中的脚本
- 手动预处理模型与主模型使用相同的参数
- 手动预处理模型处理时可以使用当前表格中的数据作为参数
- 21-30 模型可以使用当前行数据作为参数，执行一次
- 121-130 模型对当前过滤出的所有行的数据逐行执行多次
- 注意：手动预处理模型会处理当前未被条件过滤掉的所有行，即使是多层次分组的子行也会遍历处理

2.5 手动预处理外部程序调用

- 附加模型编号为“61-70/161-170”的模型为预处理时调用外部程序的模型

- 当存在此组模型时，系统会激活“手动预处理”功能按钮
- 此功能按钮的标题为编号为 61/161 的模型的描述（如果同时存在 21/121 组模型，则按钮标题使用 21/121 组的标题）
- 执行主数据查询脚本前，用于可以手动触发“此组模型”中的脚本，用于执行外部程序调用
- 手动预处理模型与主模型使用相同的参数
- 手动预处理模型处理时可以使用当前表格中的数据作为参数
- 61-70 模型只处理当前选中的行的数据
- 161-170 模型可以处理当前过滤出的所有行的数据
- 注意：手动预处理模型会处理当前未被条件过滤掉的所有行，即使是多层次分组的子行也会遍历处理
- 注意：对于外部程序调用，系统不进行自动处理，如果需要与外部系统交换数据，可以采用自动预处理模型将数据保存，供外部程序调用

2.6 自动后处理模型

- 附加模型编号为“31-40”的模型为自动后处理脚本模型
- 执行主数据查询脚本后，系统自动执行自动后处理模型中的脚本
- 自动后处理模型与主模型使用相同的参数
- 注意：自动后处理模型使用当前的行记录

2.7 手动后处理模型

- 附加模型编号为“41-50/141-150”的模型为手动后处理脚本模型
- 当存在此组模型时，系统会激活“手动后处理”功能按钮
- 此功能按钮的标题为编号为 41 或 141 的模型的描述
- 执行主数据查询脚本后，用于可以手动触发“此组模型”中的脚本
- 手动后处理模型与主模型使用相同的参数
- 手动后模型处理时可以使用当前表格中的数据作为参数
- 41-50 模型只处理当前选中的行的数据
- 141-150 模型可以处理当前过滤出的所有行的数据
- 注意：手动后处理模型会处理当前未被条件过滤掉的所有行，即使是多层次分组的

子行也会遍历处理

2.8 手动后处理外部程序调用

- 附加模型编号为“81-90/181-190”的模型为后处理时调用外部程序的模型
- 当存在此组模型时，系统会激活“手动后处理”功能按钮
- 此功能按钮的标题为编号为 81/181 的模型的描述（如果同时存在 41/141 组模型，则按钮标题使用 41/141 组的标题）
- 执行主数据查询脚本前，用于可以手动触发“此组模型”中的脚本，用于执行外部程序调用
- 手动预处理模型与主模型使用相同的参数
- 手动预处理模型处理时可以使用当前表格中的数据作为参数
- 81-90 模型只处理当前选中的行的数据
- 181-190 模型可以处理当前过滤出的所有行的数据
- 注意：手动预处理模型会处理当前未被条件过滤掉的所有行，即使是多层次分组的子行也会遍历处理
- 注意：对于外部程序调用，系统不进行自动处理，如果需要与外部系统交换数据，可以采用自动后处理模型将数据保存，供外部程序调用

2.9 附加模型的执行顺序

- 模型执行顺序
  - 手动预处理（如果有，并且用户点击操作）
  - 自动预处理（如果有）
  - 主数据查询模型
  - 自动后处理（如果有）
  - 手动后处理（如果有，并且用户点击操作）
- 当使用附件模型的时候，请从起始编号开始编写附加模型
- 如果存在多个同一类的附加模型，请保持编号连续
- 有关附件模型设计等功能请参加“专家开发”功能培训

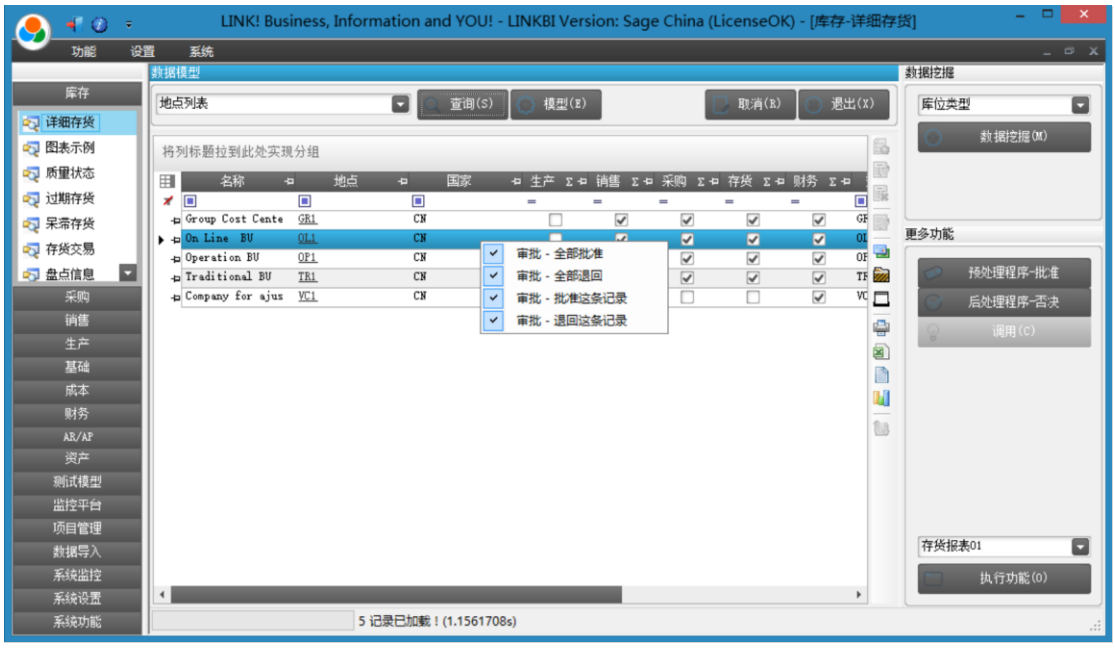
2.10 其它附加模型

- 附加模型 51-60：自动前置调用功能，执行 All 脚本，可以以当前单行的数据作为参数

- 附加模型 71-80：自动后置调用功能，执行 All 脚本，可以以当前单行的数据作为参数
- 附加模型 201-220：通过 GanttView/CanlendarView 进行数据处理时的附加模型
- 附加模型 501-520：执行时传回的是当前模型的参数。数据行上的右键菜单，执行 All 脚本，对应移动端的长按菜单，系统限制最多可以定义 20 个右键菜单功能
- 附加模型 601-650：附加的功能，比如以 URL 定义的报表打印功能调用等等，通过 CONNCODE 区分（LINKURL/LINKREPORT)
- 附加模型 1000：数据处理的条件，在该附加模型中定义各类数据处理脚本的条件
- 附加模型 1001-1020：多层级数据，在主模型中以--MULTILEVELDATA 表示
- 附加模型 1021：该模型描述多层级数据的各表之间的关系，以 0.KEY01=1.KEY01 方式描述关系，多列关联时需要组合成为一个 KEY
- 附加模型 1501-1520：右键或长按菜单功能中，如果用到标签扫描解析，使用该模型 AttachDmCode/AttachDmNum+1000
- 附加模型 1521：输入模型参数时，如果用到标签扫描解析，使用该附加模型 globleDmCode + globleDmNum + 'X1521’
- 附加模型 1522：数据行编辑或者添加时，如果用到标签扫描解析，使用该附加模型 globleDmCode + globleDmNum + 'X1522'
- 附加模型+2000：通知类附加模型。执行按钮类附加模型后返回查询的内容。

3 关联菜单执行程序

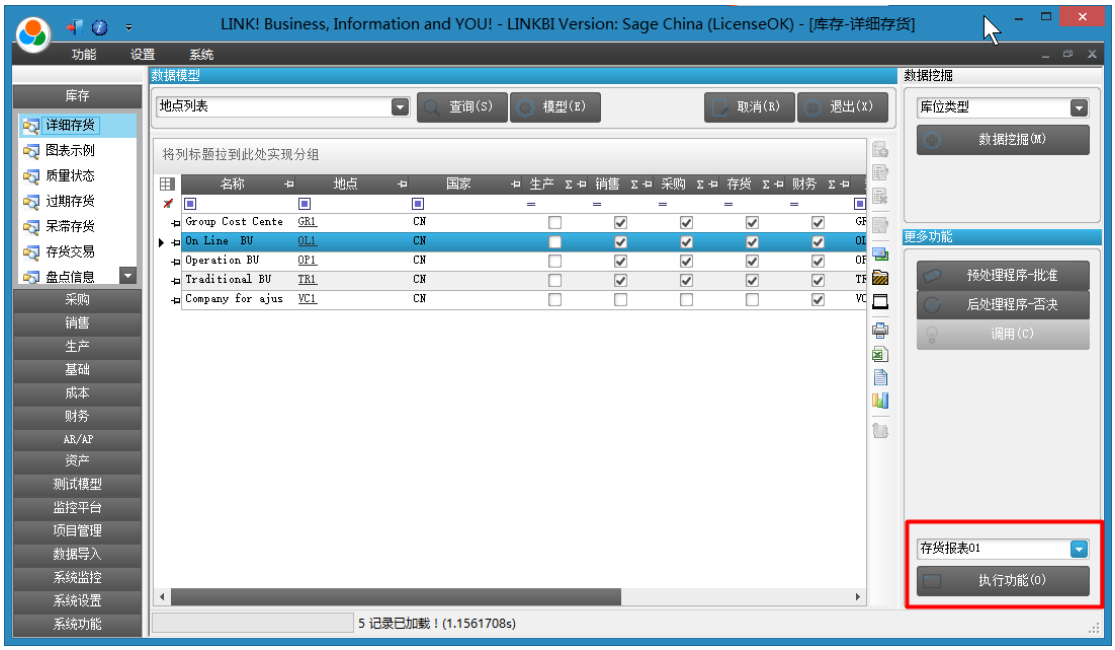
通过附加模型处理关联菜单功能。



- 附加模型编号为“501-520”的模型为关联菜单调用的程序模型
- 当存在这一组模型的时候系统允许用户执行相应模型中的“数据处理脚本”
- 在 PC 端，该模型的功能以数据行上的右键菜单出现
- 关联菜单执行之后，允许自动刷新数据
  - 如果需要实现此功能，只需要在模型的第一行用下列特殊标记即可 --LINKREFRESH
  - 有此特殊标记的模型，在 PC 端的关联菜单功能上以“√”标记
  - 有此特殊标记的模型，在移动端的“长按”菜单上以刷新图表标记
- 关联菜单处理模型与主模型使用相同的参数
  - 需要注意的时，当关联模型需要使用额外的附加参数(?)的时候，首先需要为附加模型设置与主模型一致的参数
  - 在执行附加模型功能的时候，系统首先将与主模型对应的参数赋值
- 关联菜单处理模型处理时可以使用当前表格中的当前行的数据作为参数

4 附加功能程序

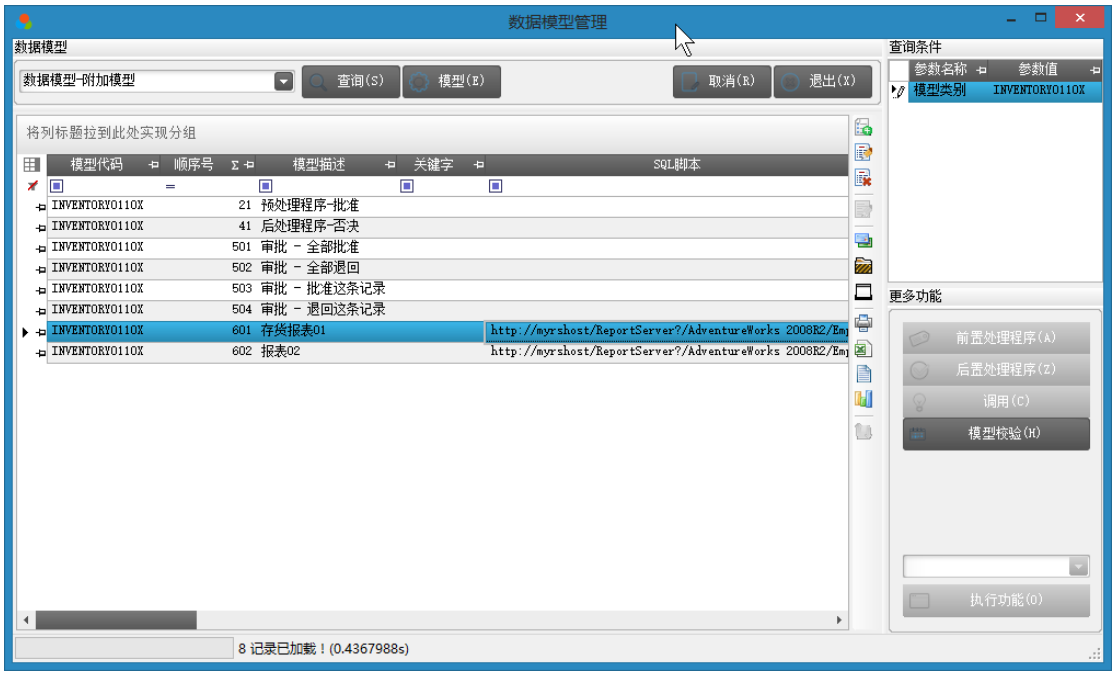
通过附加模型处理额外的功能程序。



- 附加模型编号为“601-620”的模型为关联菜单调用的程序模型
- 当存在这一组模型的时候系统允许用户执行相应模型中的“基本模型脚本”
  - 该功能可以用来使用模型参数、行数据等组合出各种不同的操作系统可以执行的功能
  - 比如，以 URL 方式调用 Microsoft SQL Server Reporting Services 报表，并自动利用当前模型的主参数、行记录作为参数调用报表打印等
- 本组模型的当前版本仅 PC 端功能可用

5 附件功能模型实例-调用 RDL

系统的模型的附加功能定义允许执行各种不同的功能。并且，在执行这些功能的时候，创建的功能“命令行”或者“URL”中可以使用模型的主参数及表格中的数据等。因此，使用此功能可以配置出复杂的系统应用。比如，可以根据模型创建一组报表打印功能，以 URL 的模式打开 Microsoft SQL Server Reporting Services 中的 RDL 报表。



- 如图所示，定义了模型 601 和 602，分别用于利用 Reporting Services 打印 RDL 报表
- 模型的参考脚本如下：

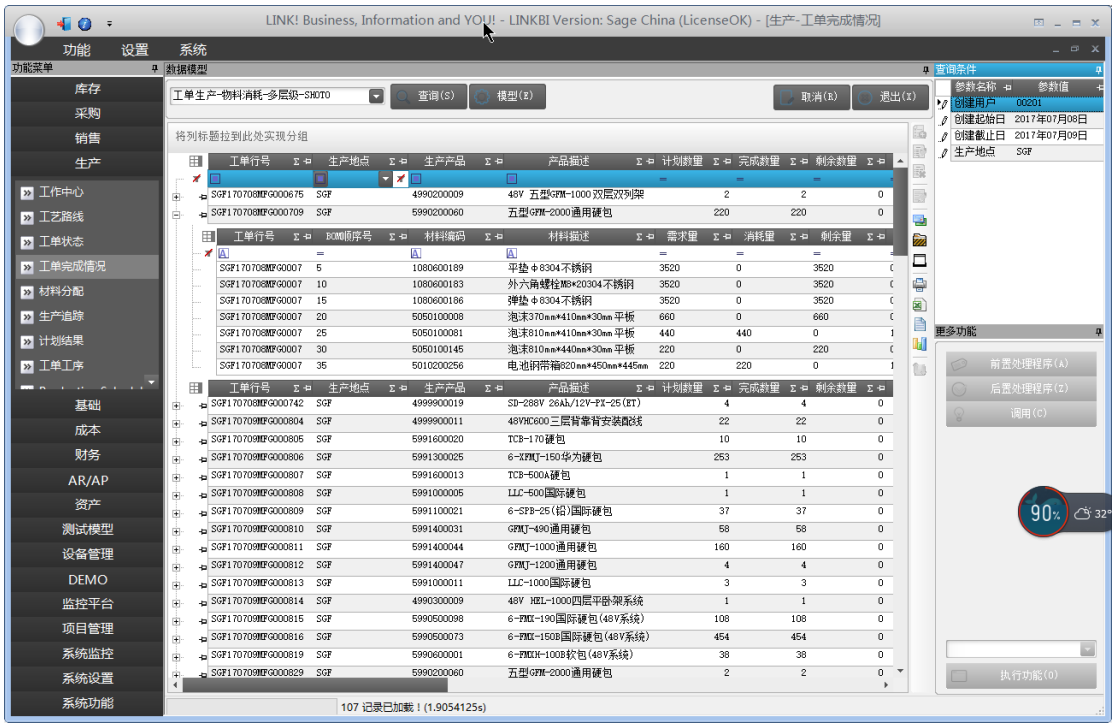
```
http://myrshost/ReportServer?/AdventureWorks 2008R2/Employee_Sales_Summary_2008R2&ReportMonth={0}&ReportYear={1}&Para01=#名称#
```

- 在实际执行时，经过脚本参数替换，得到实际需要启动的命令行如下：

```
http://myrshost/ReportServer?/AdventureWorks%202008R2/Employee_Sales_Summary_2008R2&ReportMonth=&ReportYear=&Para01=Group%20Cost%20Center
```

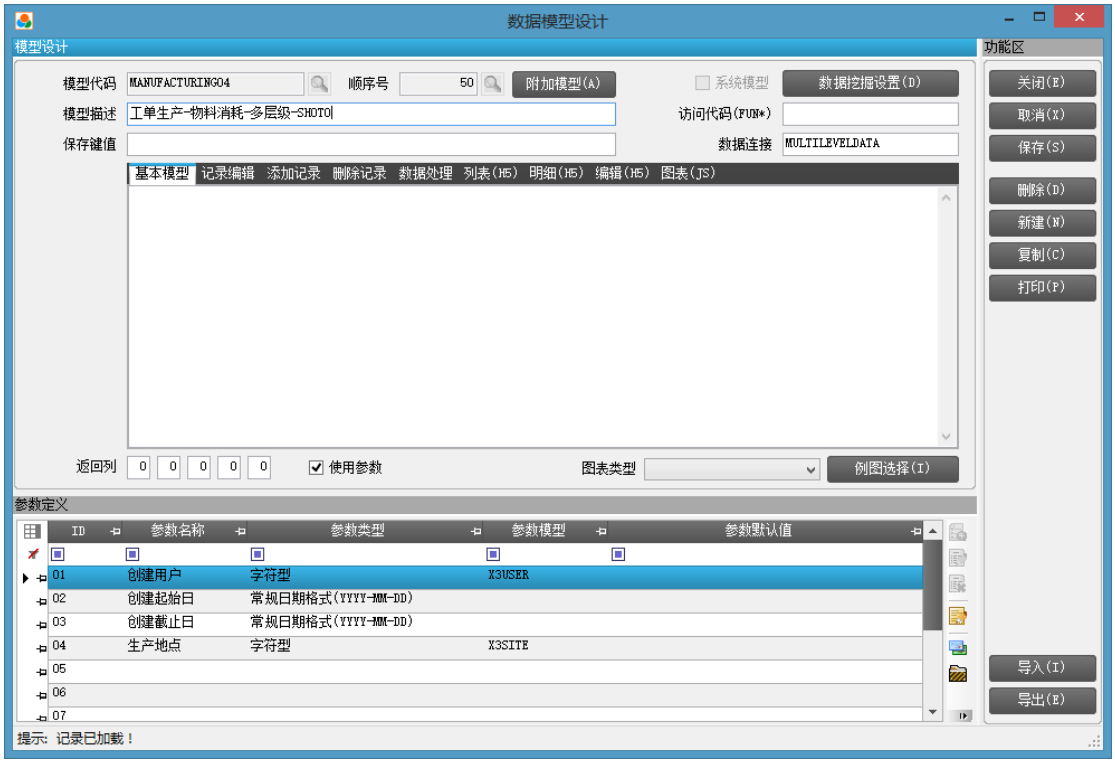
- 在模型脚本中，可以使用模型主参数或数据集中的列数据作为参数
- 系统可以自动根据当前选中的行，生成相应的功能命令
- 在执行数据查询的时候，系统可以自动判断是否有该组模型的定义，当存在该组功能模型的时候，自动激活相应的功能，用户根据需要进行操作即可

6 多层级数据的附加模型处理



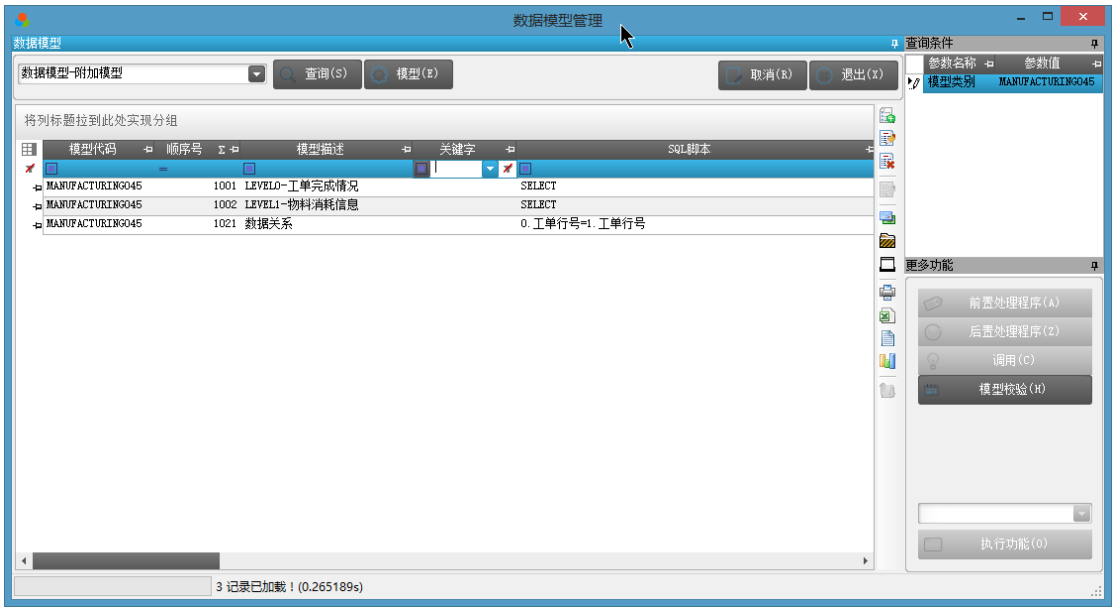
- 如上图所示，可以通过模型定义在系统中显示复杂的多级关联数据。

6.1 多级数据模型的主模型设计

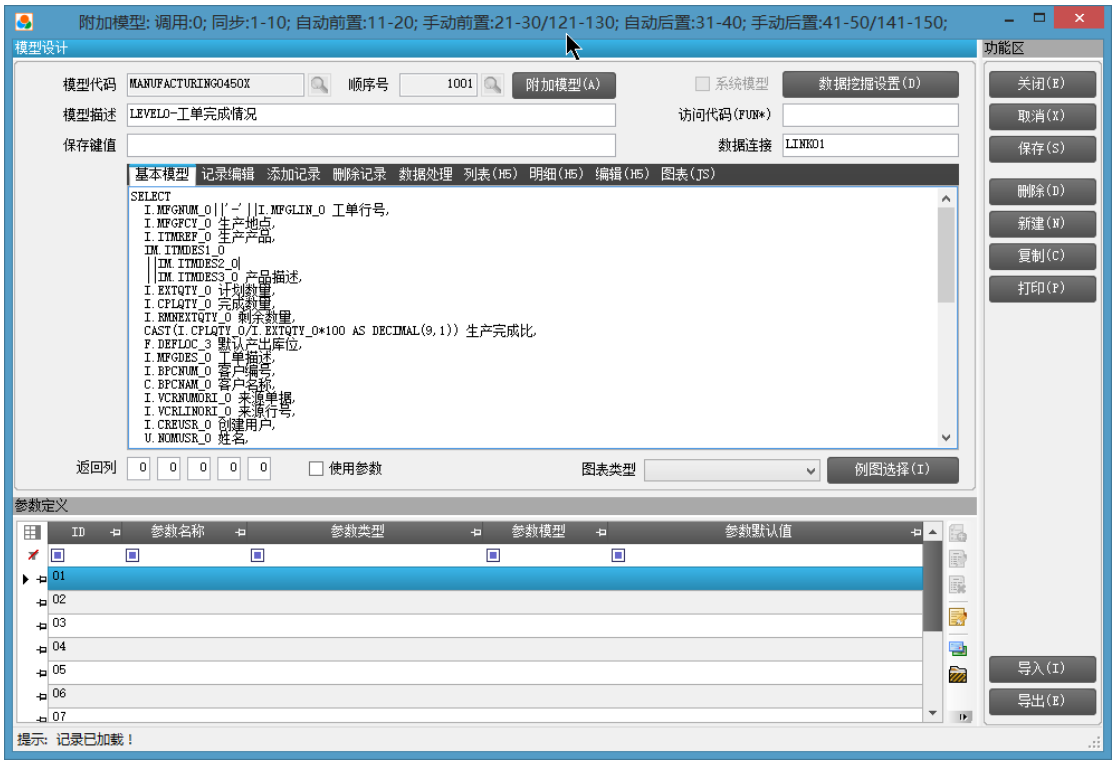


- 处理多级数据的主模型用 MULTILEVELDATA 关键字，作为数据连接代码
- 当系统需要对该模型进行处理的时候，会自动查找对应多级数据的各子模型
- 处理各个子模型的时候，系统会自动传递主模型的参数

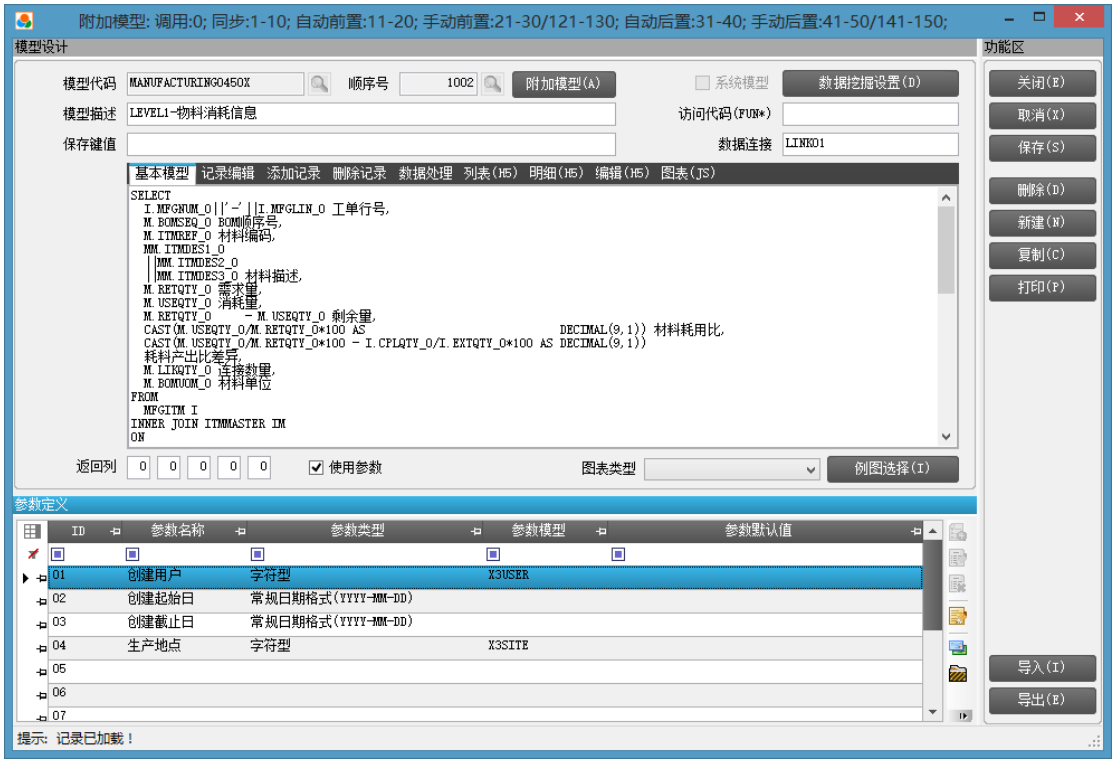
6.2 多级数据模型的子模型设计



- 处理多级关联数据需要用到附加模型的 1001-1020 和 1021 号模型，其中：
  - 1001-1020，按照常规模型进行设计，可以对应不同的数据源
  - 其中的每个模型分别对应最终 Dataset 中的一个数据表
  - 数据表编号顺序从 0 开始依次递增，即 0-1001；1-1002；3-1003……
  - 1021 模型用户描述数据表之间的关系，比如 0.工单行号=1.工单行号

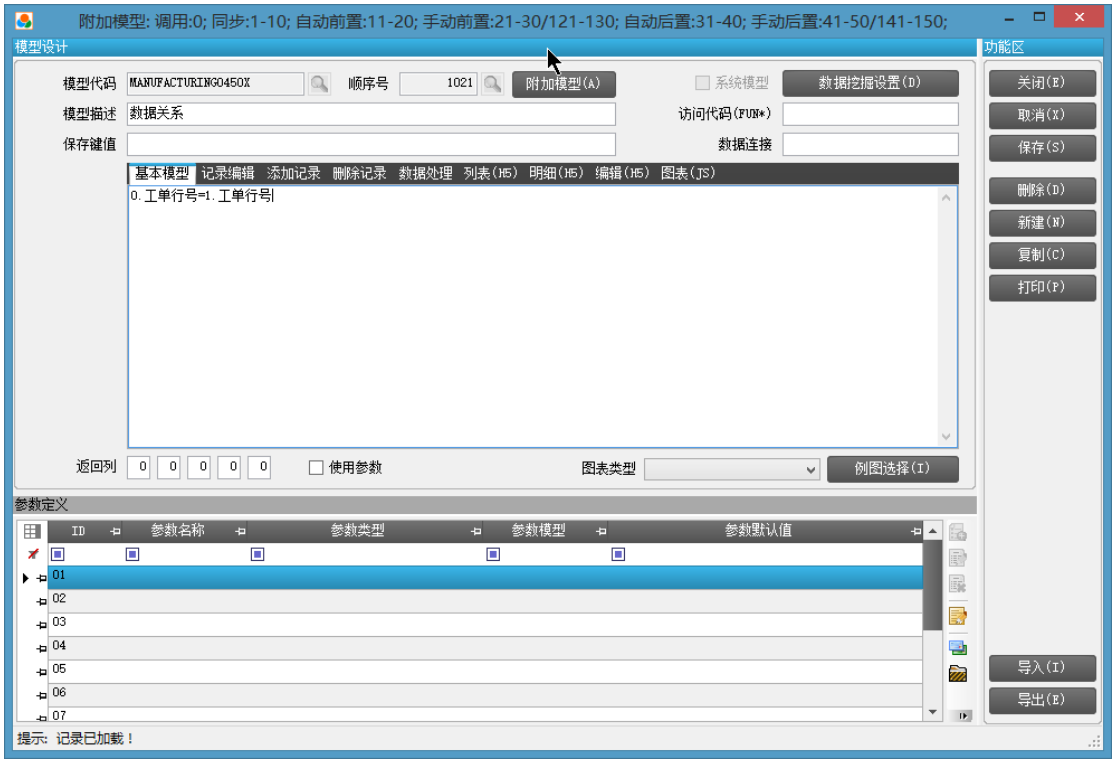


- 比如，上述模型（编号 1001）用于产生第一层数据（Table0）



- 该模型（编号 1002）用于产生第二层数据（Table1）

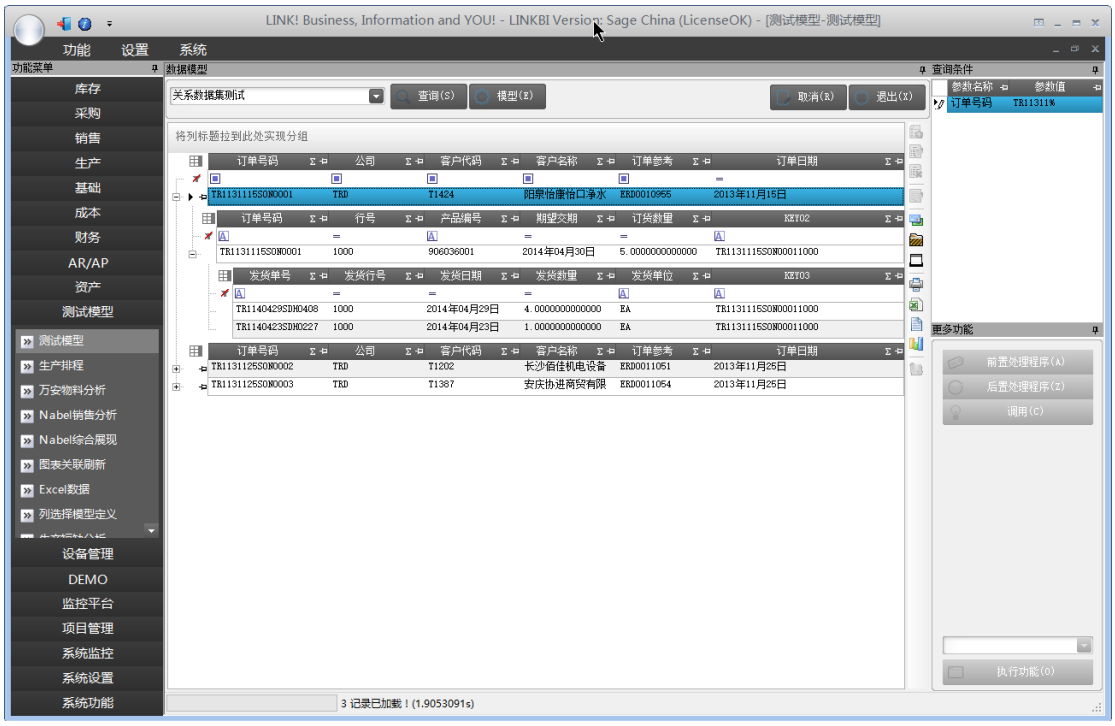
6.3 关系模型设计



- 如上图所示，该模型定义 1001-1020 模型对应的各层数据之间的关系

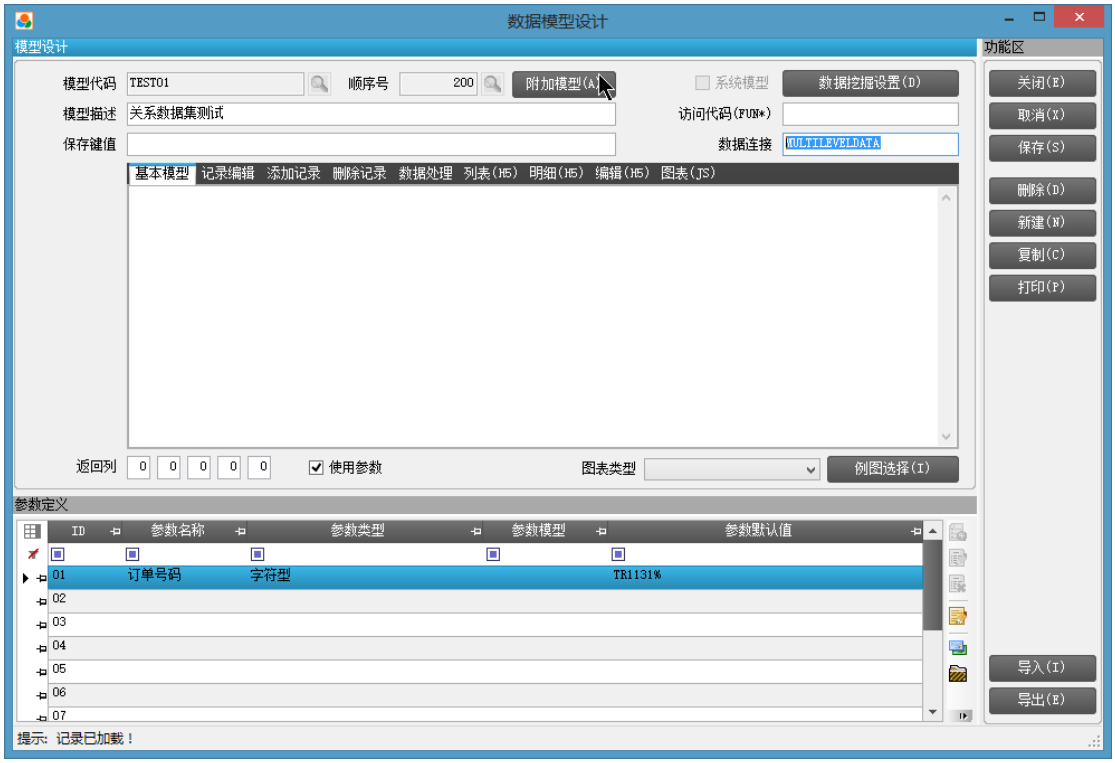
- 本例中：0.工单行号=1.工单行号，表示
  - 表 0 中的“工单行号”列作为主表关键字
  - 表 1 中的“工单行号”作为子表的外键与主表关联

7 多层级数据模型实例

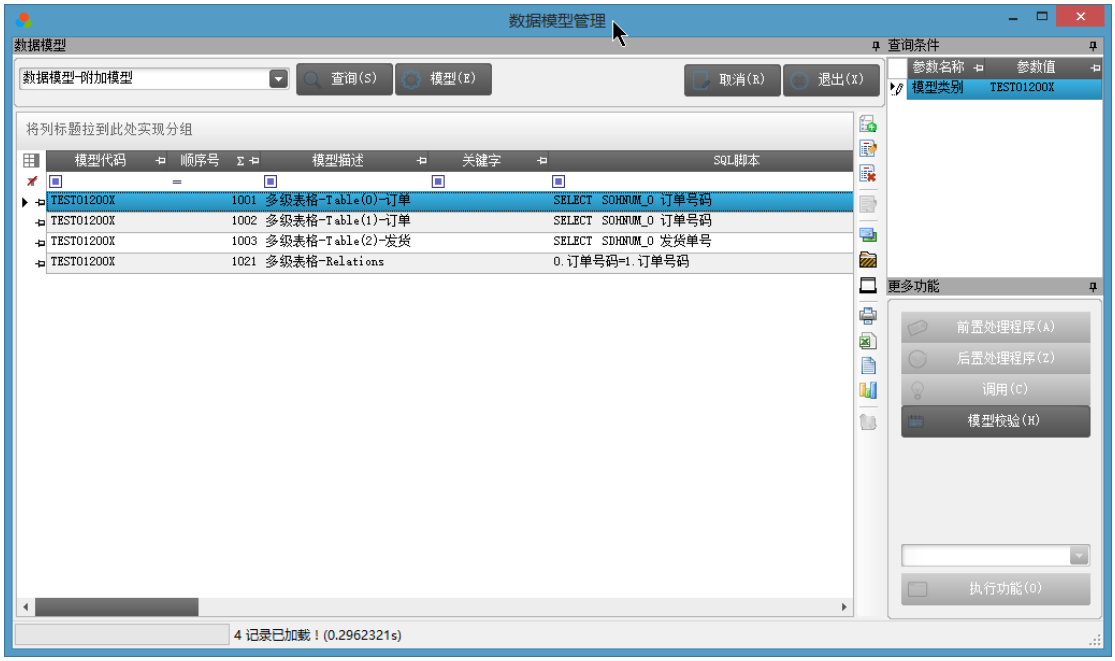


- 如图所示，是一个三层数据的实例，分别是：
  - 1001，订单信息
  - 1002，订单行信息
  - 1003，订单行发货记录
  - 1021，数据关系描述

7.1 主模型设计



7.2 附加多层数据模型



- 模型 1001

```
SELECT SOHNUM_0 订单号码
```

```
        , CPY_0  公司
        , BPCORD_0  客户代码
        , BPCNAM_0  客户名称
        , CUSORDREF_0  订单参考
        , ORDDAT_0  订单日期
    FROM    SORDER
    WHERE   SOHNUM_0 LIKE '{0}'
```

- 模型 1002

```
    SELECT SOHNUM_0  订单号码
        , SOPLIN_0  行号
        , ITMREF_0  产品编号
        , EXTDLVDAT_0  期望交期
        , QTY_0  订货数量
        , SOHNUM_0 + CAST(SOPLIN_0 AS VARCHAR(10)) KEY02
    FROM    SORDERQ
    WHERE   SOHNUM_0 LIKE '{0}'
```

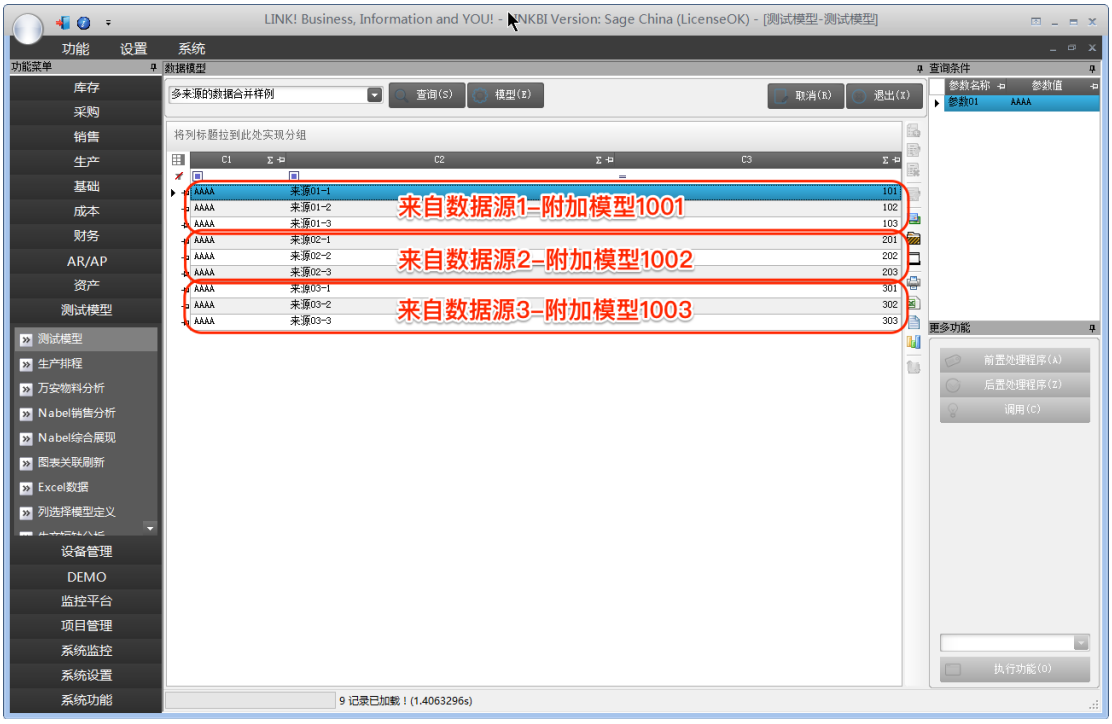
- 模型 1003

```
    SELECT SDHNUM_0  发货单号
        , SDDLIN_0  发货行号
        , SHIDAT_0  发货日期
        , QTY_0  发货数量
        , SAU_0  发货单位
        , SOHNUM_0 + CAST(SOPLIN_0 AS VARCHAR(10)) KEY03
    FROM    SDELIVERYD
    WHERE   SOHNUM_0 LIKE '{0}'
```

- 模型 1021

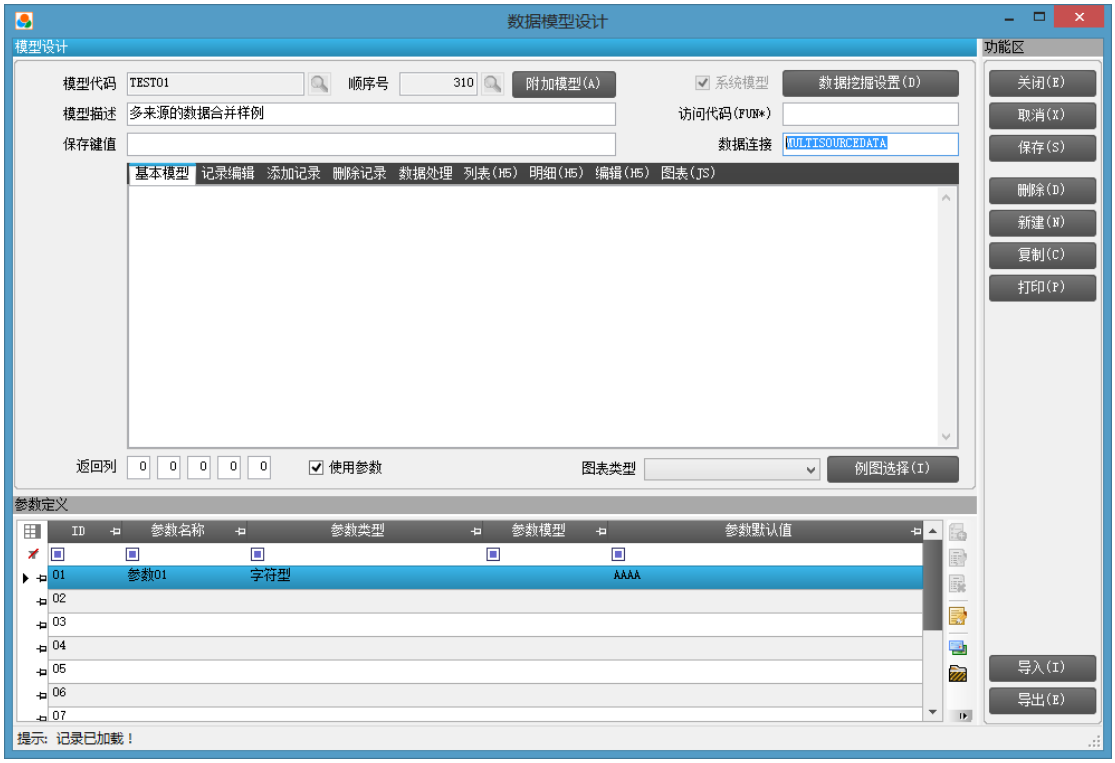
```
    0.订单号码=1.订单号码

    1.KEY02=2.KEY03
```

8 多数据源合并数据的附加模型处理



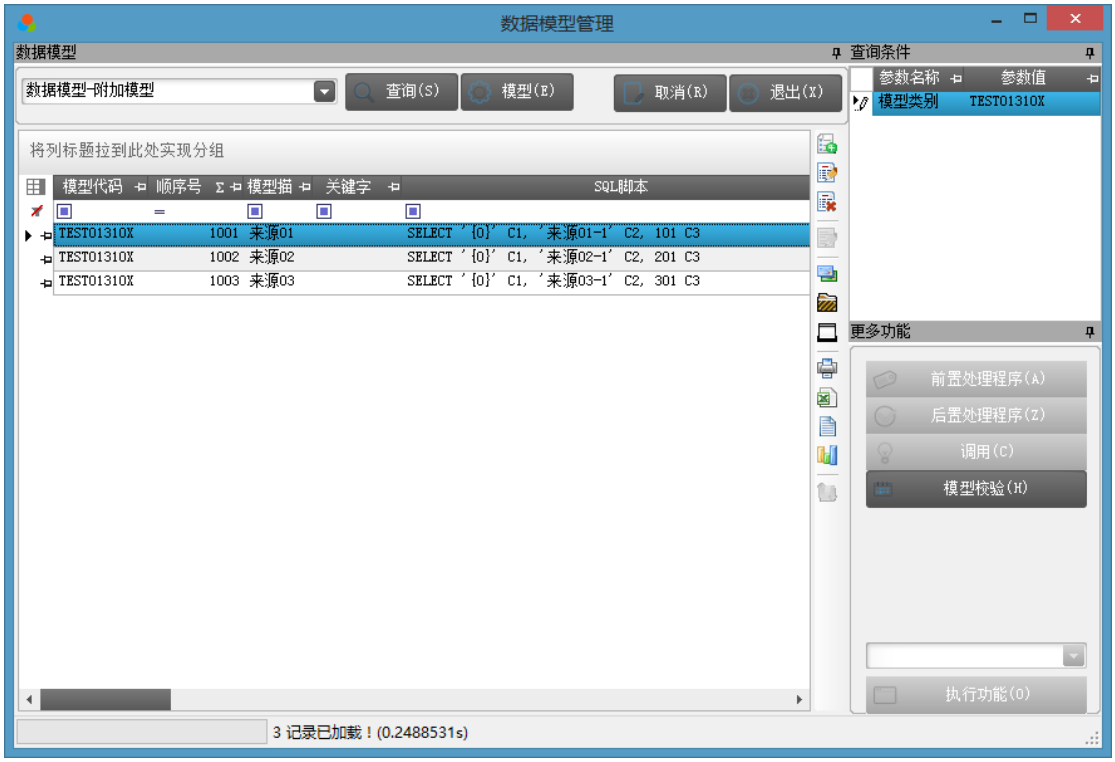
- 如上图所示，可以通过模型定义在系统中显示复杂的多数据源合并的数据。
- 该模式要求，来自多个数据源的数据必须保持相同的数据结构。

8.1 多数据源合并数据的主模型设计

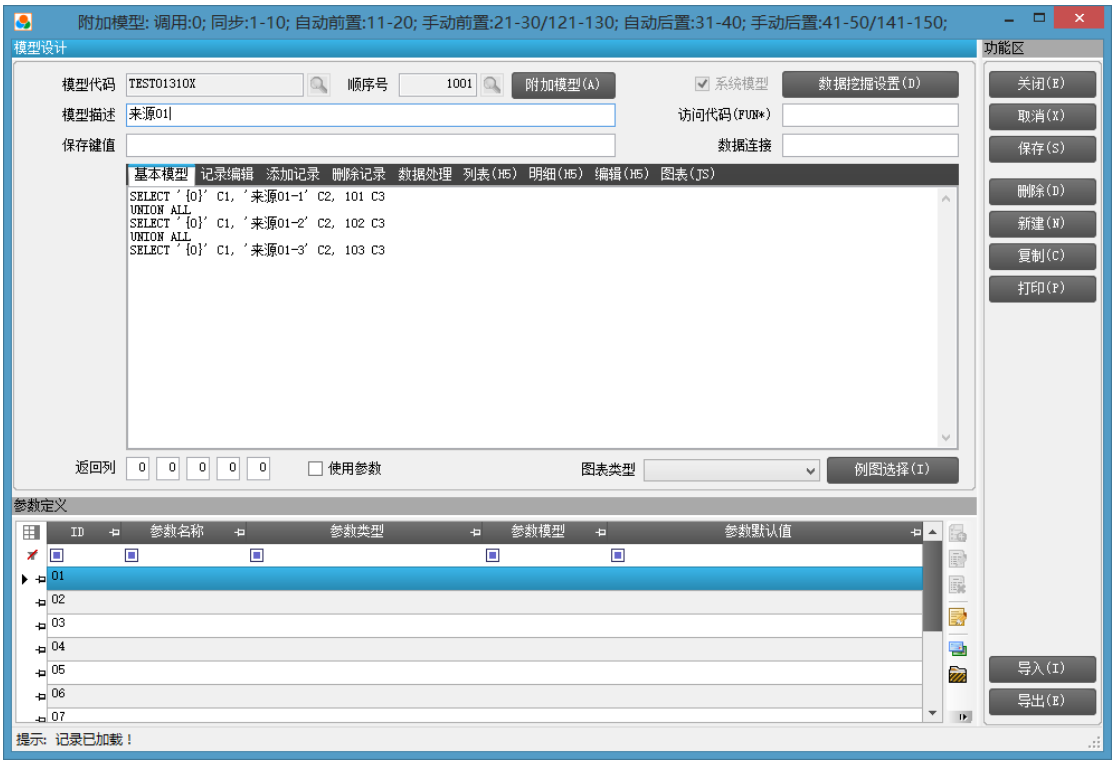


- 处理多数据源合并数据的主模型用 MULTISOURCEDATA 关键字，作为数据连接代码
- 当系统需要对该模型进行处理的时候，会自动查找对应多级数据的各子模型
- 处理各个子模型的时候，系统会自动传递主模型的参数

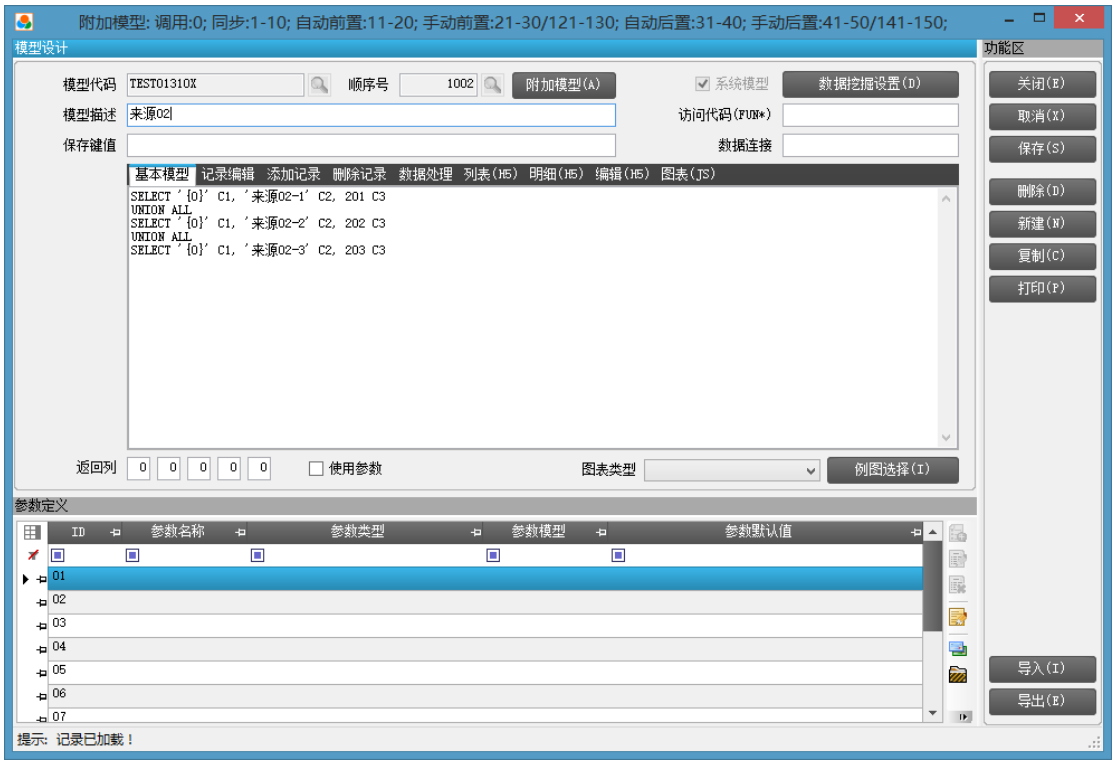
8.2 多数据源合并数据的子模型设计



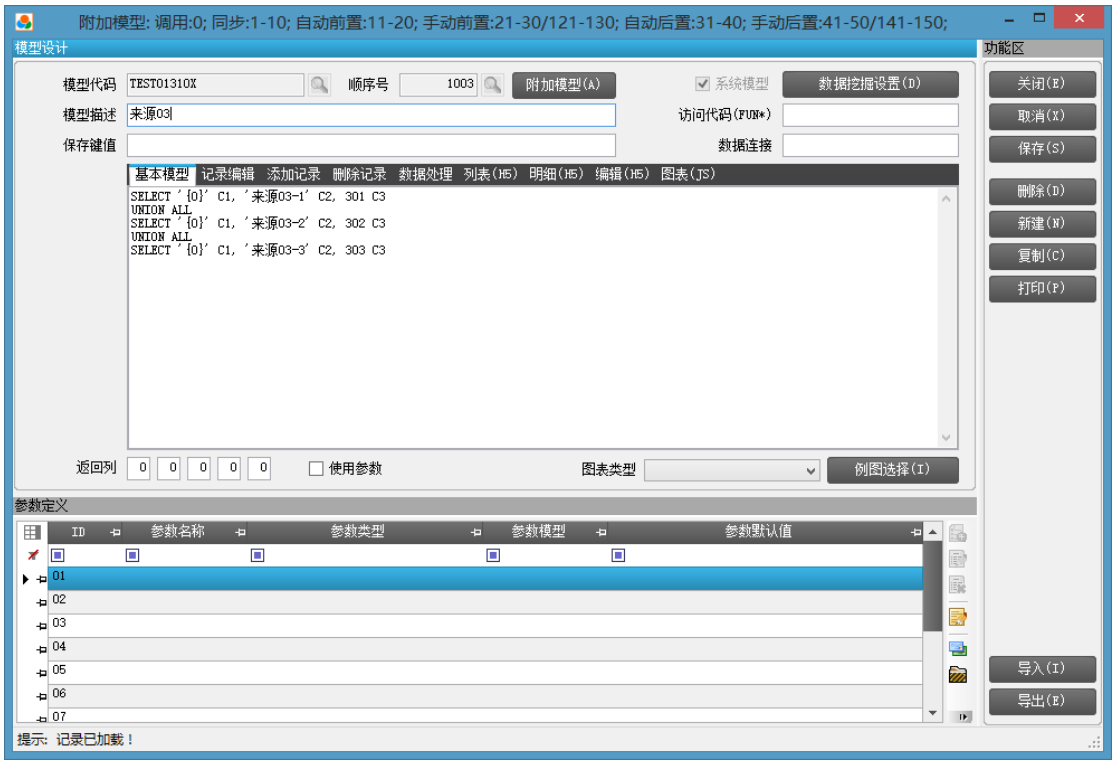
- 处理多数据源合并数据需要用到附加模型的 1001-1020 和 1021 号模型，其中：
  - 1001-1020，按照常规模型进行设计，可以对应不同的数据源
  - 其中的每个模型分别对应最终 Dataset 中的数据表的一部分内容



- 比如，上述模型（编号 1001）用于产生第一部分数据



- 该模型（编号 1002）用于产生第二部分数据

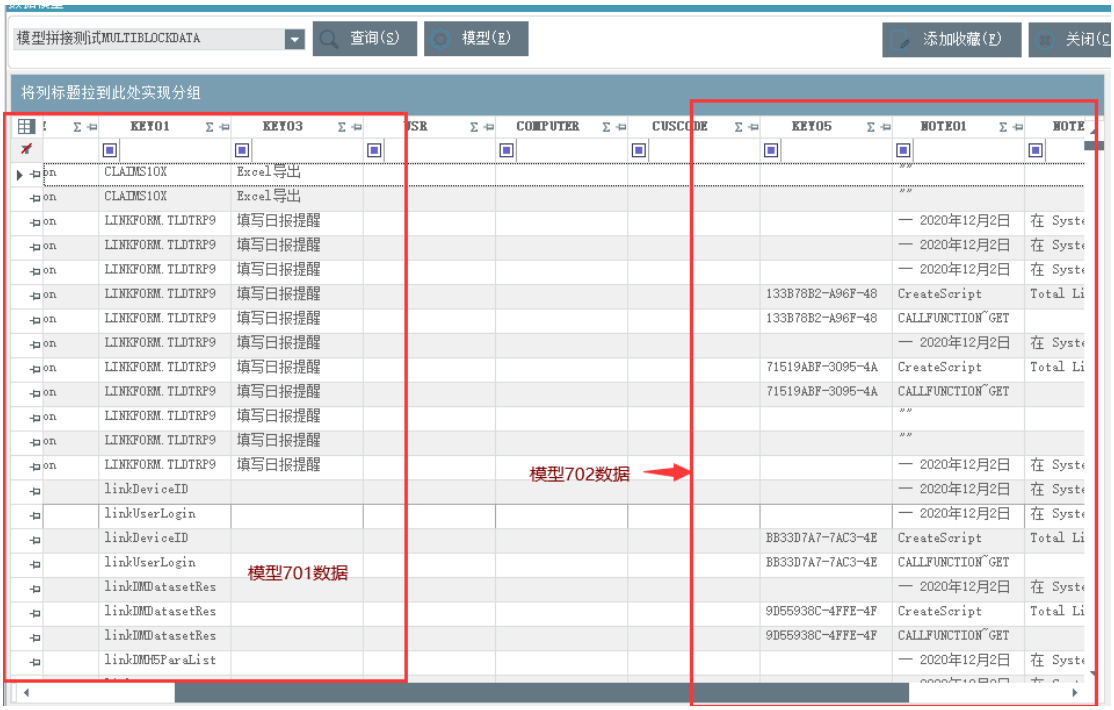


- 如图，该模型（编号 1003）用于产生第三部分数据

- 当前，系统允许来自 20 个不同数据源的数据进行拼接（模型 1001-1020）
- 模型执行之后，来自不同子模型的数据自动“合并”在一起显示。

9 MULTIBLOCKDATA 合并多个数据源

效果如下：



实现步骤如下：

- 如图所示，通过 JSON 可以将多个模型查询结果合并在一个模型中
  - 主模型参考代码如下：

```
[{{
  "dm":{{
```

```
            "dmCode": "WERNER01120X",                \\模型代码
            "dmNum": 701,                            \\模型顺序号
            "Para": ["","",""]          \\模型参数
             }},
            "rowData":{{"STARTTIME":"{0}","ENDTIME":"{1}","USER":"{2}"}},
            \\与 701 模型的交互参数，将 value 值（{0}，{1}，{2}）传至 701 模型，在 701 模
            型中用 key 值（STARTTIME，ENDTIME，USR）代替
             "colRemove": ["KEY05","note01","NOTE03"]
            \\隐藏字段，可以将一些不必要的参数隐藏
            }},{{
                "dm":{{                  \\添加模型
                    "dmCode": "WERNER01120X",          \\模型代码
                    "dmNum": 702,                      \\模型顺序号
                    "Para": ["{0}","{1}","{2}"]              \\模型参数
                }},
                "colRemove": []
}}]
```

701 模型脚本如下：

```
            SELECT TOP 100 [LOGDATE]
                          ,[LOGTYPE]
                          ,[KEY01]
                          ,[KEY03]
                          ,[KEY05]
                          ,[NOTE01]
                          ,[NOTE03]
            FROM LINKLOG
            WHERE CONVERT(VARCHAR(112),LOGDATE,23) BETWEEN '#STARTTIME#' AND
            '#ENDTIME#'
            AND
            USR LIKE '%#USER#%'
            --#STARTTIME#、#ENDTIME#、#USER#：与主模型中 rowData 交互的参数
```

702 模型脚本如下：

```
SELECT  TOP 1000 [USR]
           ,[COMPUTER]
           ,[CUSCODE]
          ,[KEY05]
           ,[NOTE01]
           ,[NOTE02]
           ,[NOTE03]
FROM LINKLOG
WHERE CONVERT(VARCHAR(112),LOGDATE,23) BETWEEN '{0}' AND '{1}'
AND
USR LIKE '%{2}%'
```

10 附加模型中执行存储过程

根据业务需要，有时某些情况下需要在服务器端执行存储过程，以便高效地完成某些数据处理任务。这时，可以使用下面的语法执行存储过程。

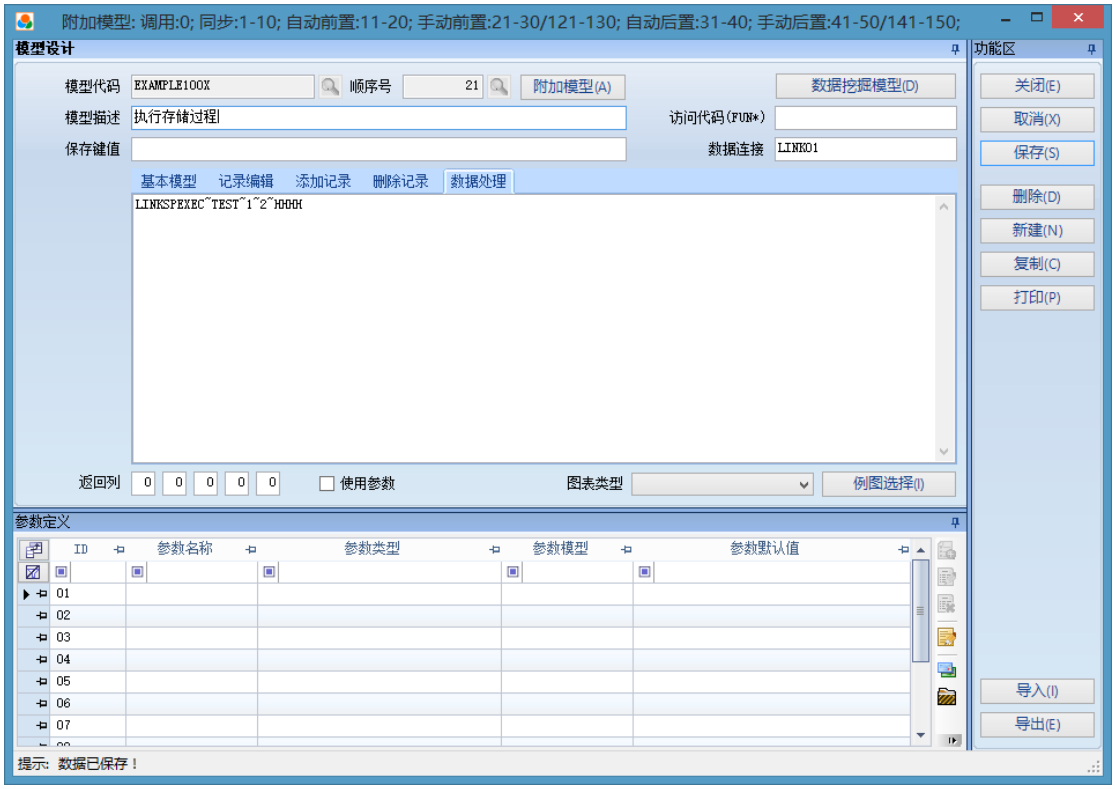
```
LINKSPEXEC~SPNAME~PARA1~PARA2~PARA3~PARA4...
```

上述表达式中，

- LINKSPEXEC 是系统定义关键字，用于标明这是一段存储过程执行的脚本
- 后面 SPNAME 部分是存储过程的名称
- 接下来是存储过程的参数，系统支持调用的存储过程可以有 0-25 个参数

说明：

- 当前版本，存储过程不支持返回数据集或者数据项。
- 如果需要返回数据集等，可以在调用存储过程执行结束后，通过常规的模型脚本访问数据。
- 存储过程不支持自动将数据保存到 LINKTEMP 临时表中，如果需要保存数据，请在存储过程中编写脚本处理。



- 如图所示，可以编写一个编号为 21 的附加模型，执行一个数据端的存储过程

11 智能设备控制

- TOTAL LINK 系统可以通过设定预处理及后处理程序控制各类智能设备
- 控制智能设备时，使用与外部调用程序使用相同的附加模型
  - 附加模型编号为“61-70/160-170”的模型为预处理时调用外部程序的模型
  - 附加模型编号为“81-90/180-190”的模型为后处理时调用外部程序的模型
- 如果预处理或后处理程序的调用语法符合以“!SETDEVICE!”开始时，系统会自动处理为对应的智能设备控制
- 控制语法如下：

```
LINKDEVICE~DEVICETYPE~NAME~PASSWORD~VALUE~MESSAGE
```

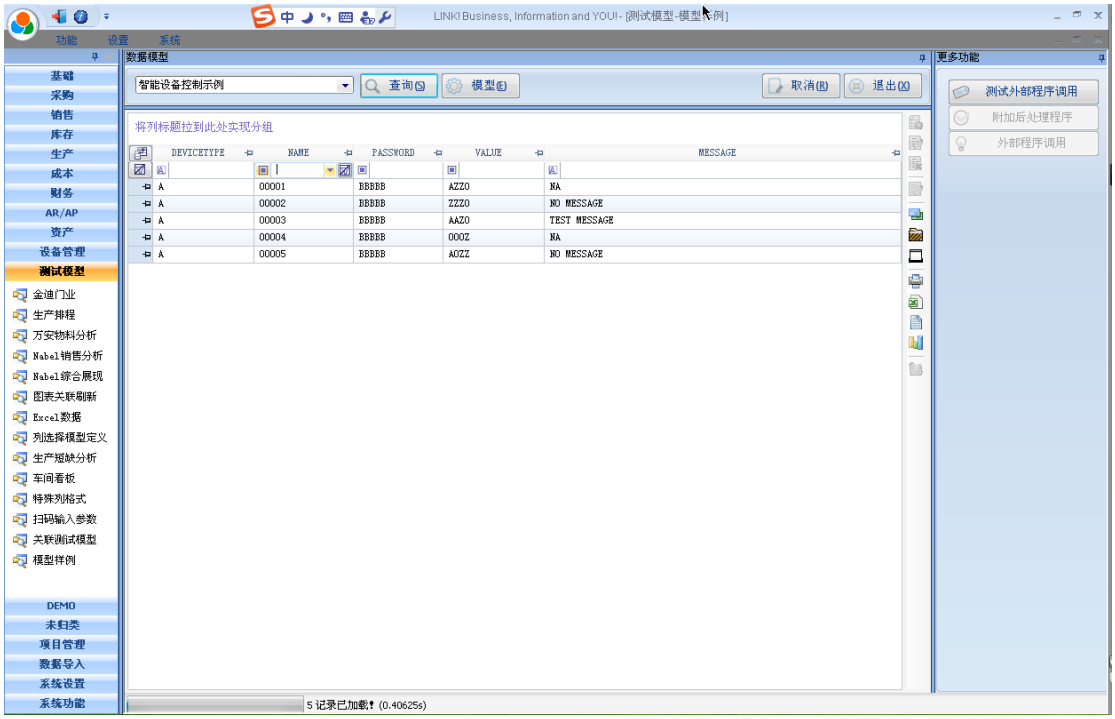
- 参数说明
  - DEVICETYPE: 设备类型，用于控制不同的智能设备
  - NAME: 智能设备的登录名
  - PASSWORD: 智能设备的登录密码
  - VALUE: 传递给智能参数的控制值
    - 比如，某智能设备有 4 路继电器开关
    - 分别以 A 表示关闭、Z 表示打开、0 表示保持原来的开关状态
    - VALUE 设置为 AZ00 时，即表示关闭 1、打开 2、保持 3/4 的状态不变
  - MESSAGE: 传递到智能设备的信息（不需要传递信息时，可以用 NA 作为参数值）

- 举例如下：

```
LINKDEVICE~DEVA~00001~ABCDE~A000~NA
LINKDEVICE~DEVA~00002~ABCDE~AZ00~NA
LINKDEVICE~DEVA~00003~ABCDE~ZZA0~NA
LINKDEVICE~DEVA~00004~ABCDE~ZZZ0~NA
```

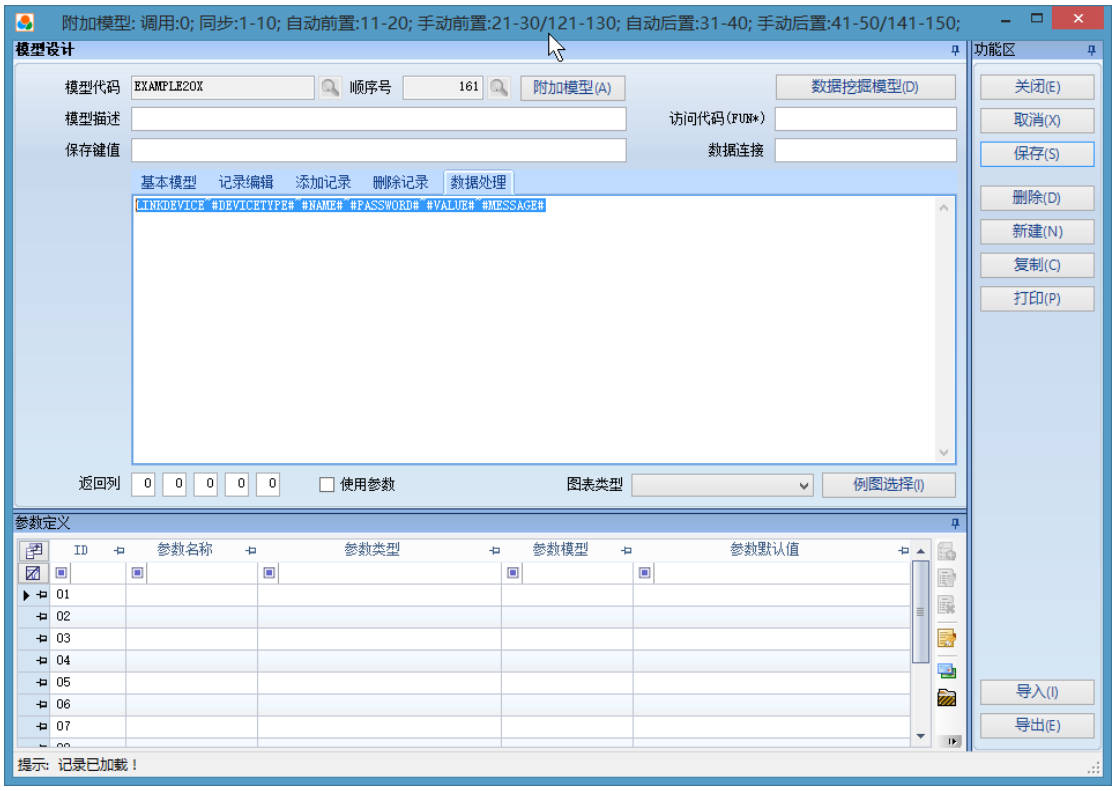
- 可以使用此功能控制各类不同的智能设备
- 比如，设备警示灯及各种电子标签等

12 智能设备控制实例



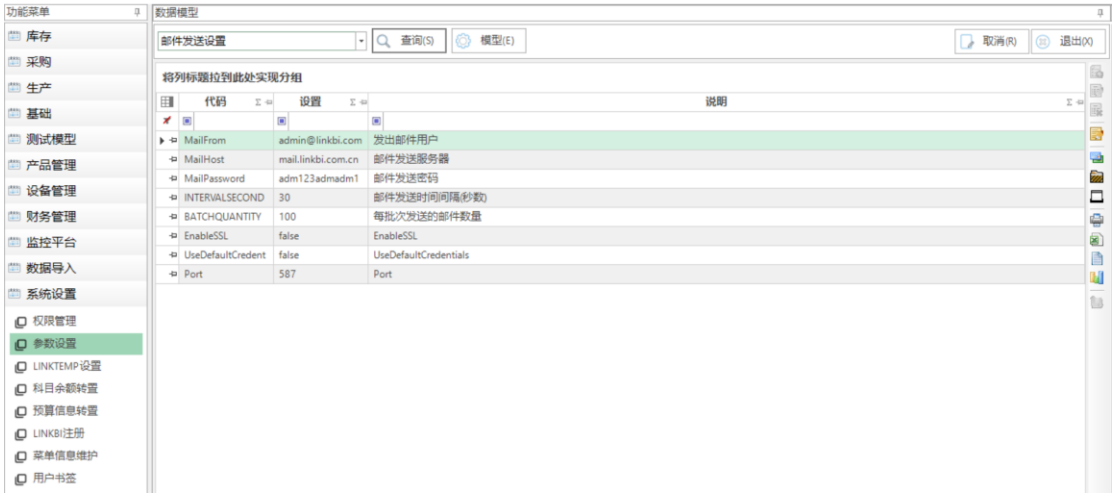
- 如图所示，是一个智能控制的模型实例

- 附加模型 161 如下图所示



- 对于主模型的数据，当执行模型时，即可根据参数设定控制对应的智能设备

13 系统邮件发送功能设置



- 请按照上图所示的内容设置邮件发送的基本信息
- 其中：
  - MailFrom，指发出邮件的地址
  - MailHost，发出邮件的发件服务器地址
  - MailPassword，发件服务器的密码
- 系统升级说明
  - 执行如下脚本增加邮件参数

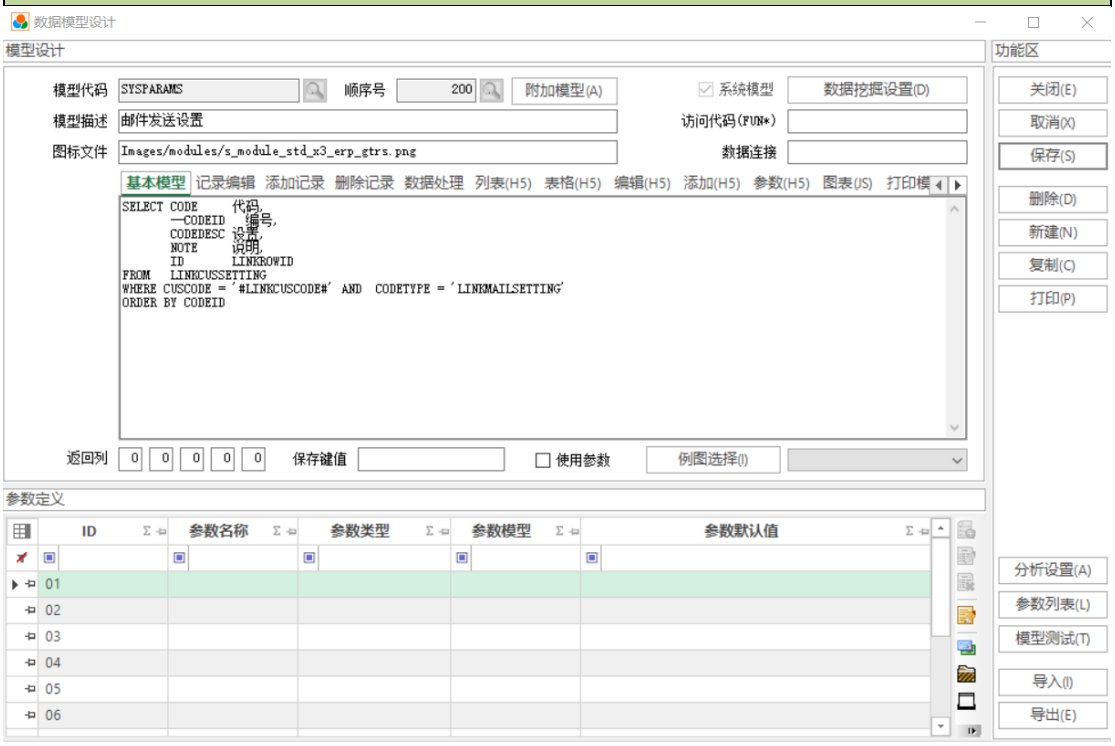
```
INSERT INTO SysCode (CodeType, Code, CodeID, CodeDesc, Note) VALUES ('LINKMAILSETTING', 'MailFrom', 10, '', '发出邮件用户');
INSERT INTO SysCode (CodeType, Code, CodeID, CodeDesc, Note) VALUES ('LINKMAILSETTING', 'MailHost', 20, '', '邮件发送服务器');
INSERT INTO SysCode (CodeType, Code, CodeID, CodeDesc, Note) VALUES ('LINKMAILSETTING', 'MailPassword', 30, '123456', '邮件发送密码');
```

  - 在系统中增加模型 SYSPARAMS-70
  - 基本模型脚本

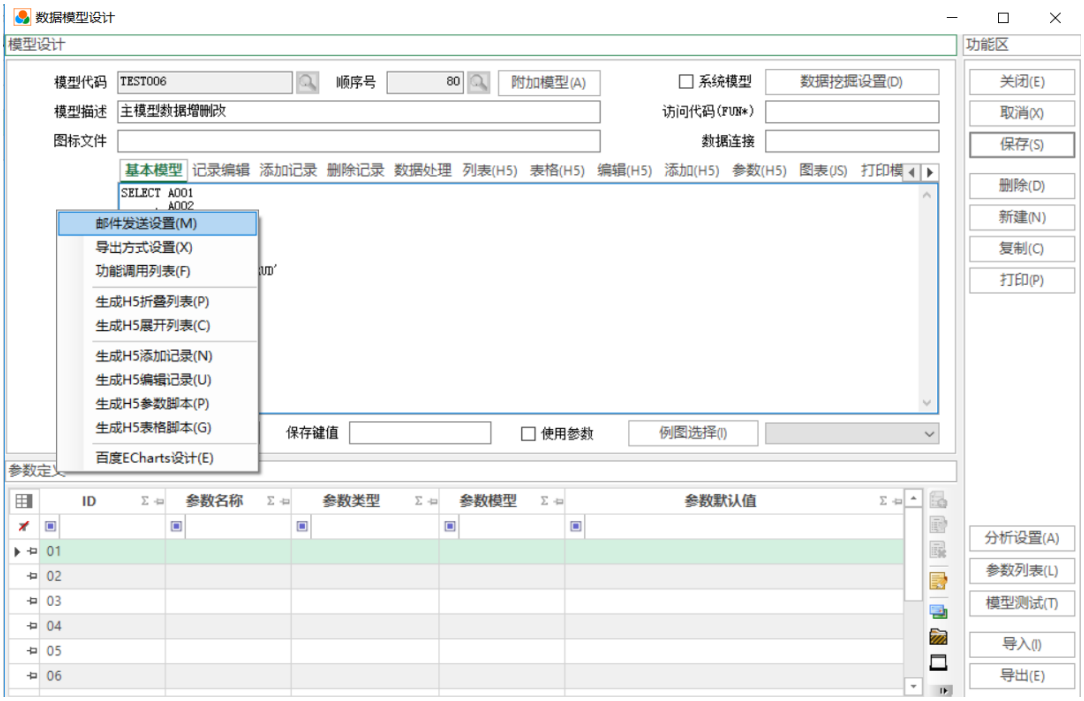
```
SELECT CODE 代码,
--CODEID 编号,
CODEDESC 设置,
NOTE 说明,
ID LINKROWID
FROM LINKCUSSETTING
WHERE CUSCODE = '#LINKCUSCODE#' AND CODETYPE = 'LINKMAILSETTING'
ORDER BY CODEID
```

■ 模型编辑脚本

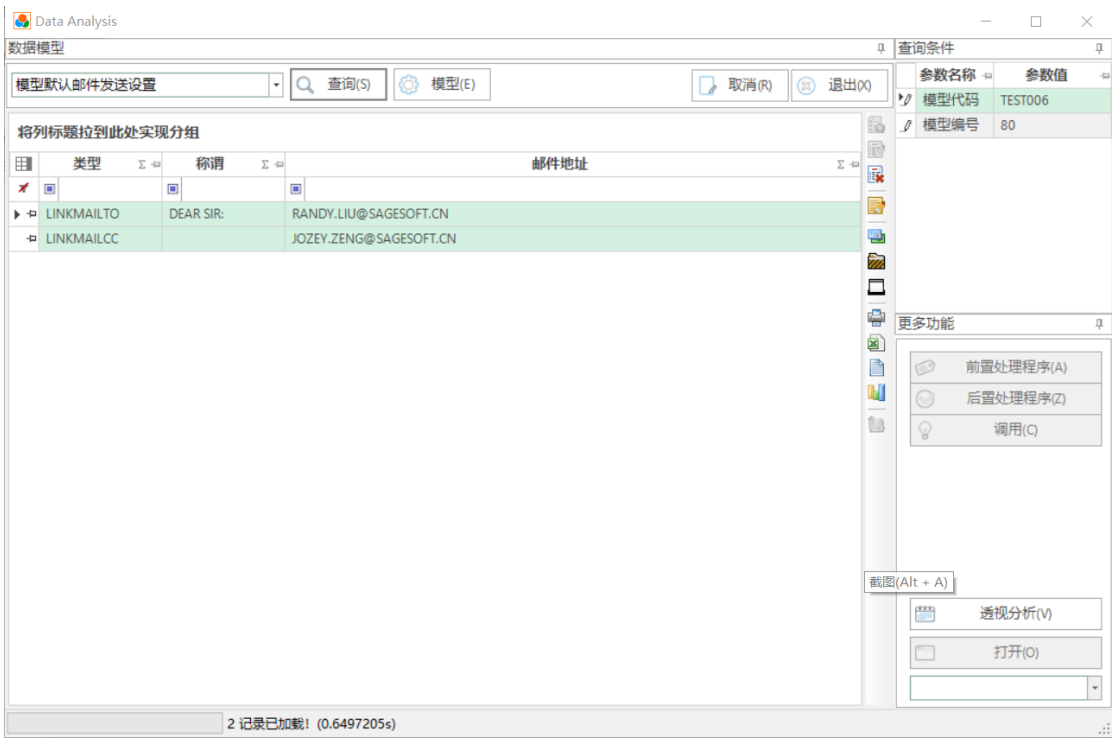
```
UPDATE LINKCUSSETTING

SET       CODEDESC = '@设置@',
NOTE = '@说明@'
WHERE CUSCODE = '#LINKCUSCODE#' AND    ID = #LINKROWID#
```



14 模型的默认发送邮件设置



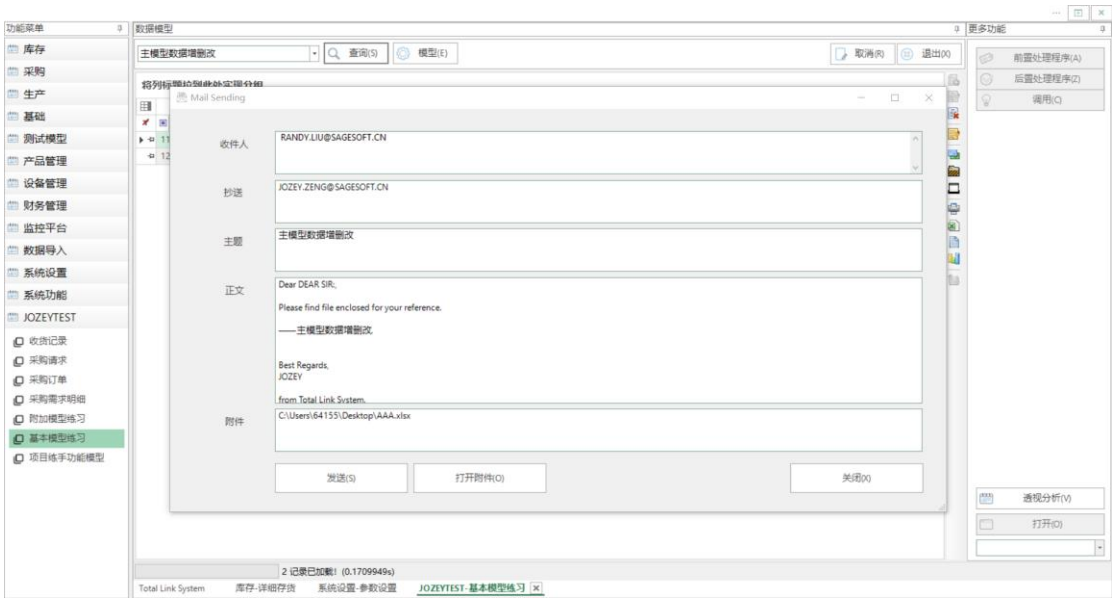
- 如上图所示，在模型定义功能上右键菜单，可以显示“邮件发送设置”



- 按照上图格式可以设置该模型的默认邮件发送设置
- 可以为模型设置默认的收件人地址列表以及抄送人地址列表

- 类型 LINKMAILTO 指的是收件人信息，可填写收件人称谓，收件人邮件地址列表，多个收件人地址用“;”分割
- 类型 LINKMAILCC 指的是抄送人信息，抄送人邮件地址列表中多个收件人地址用“;”分割

15 邮件发送功能

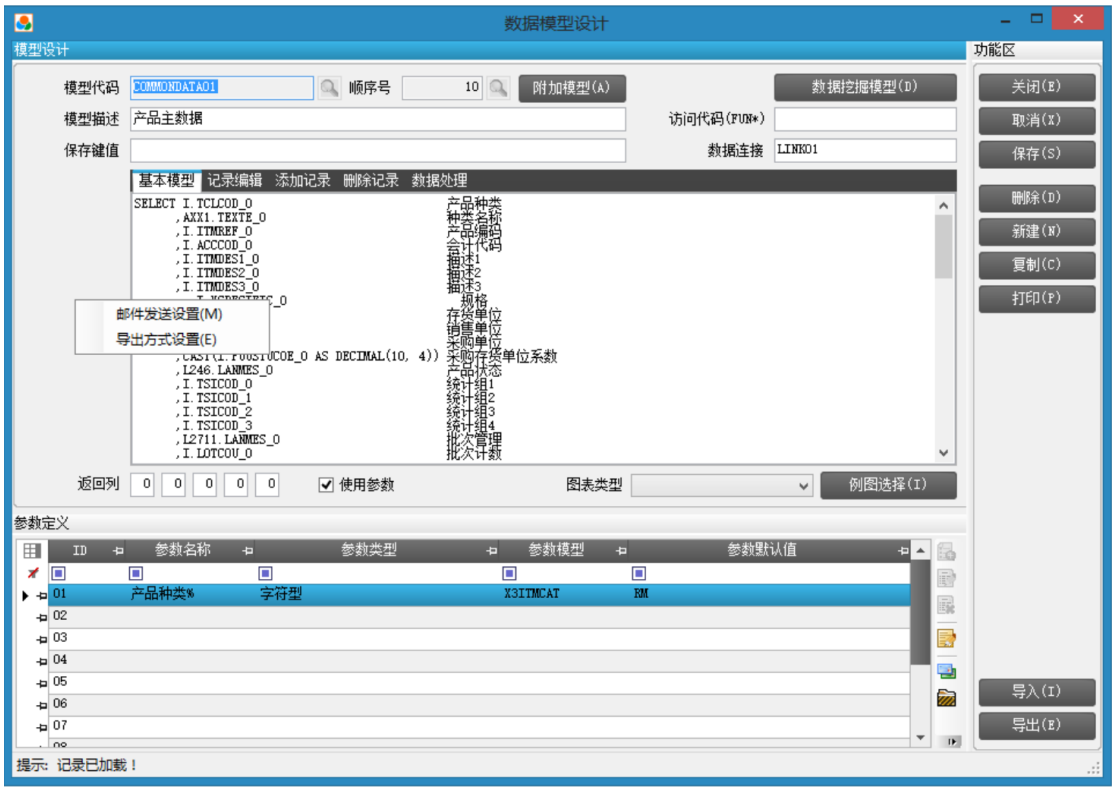


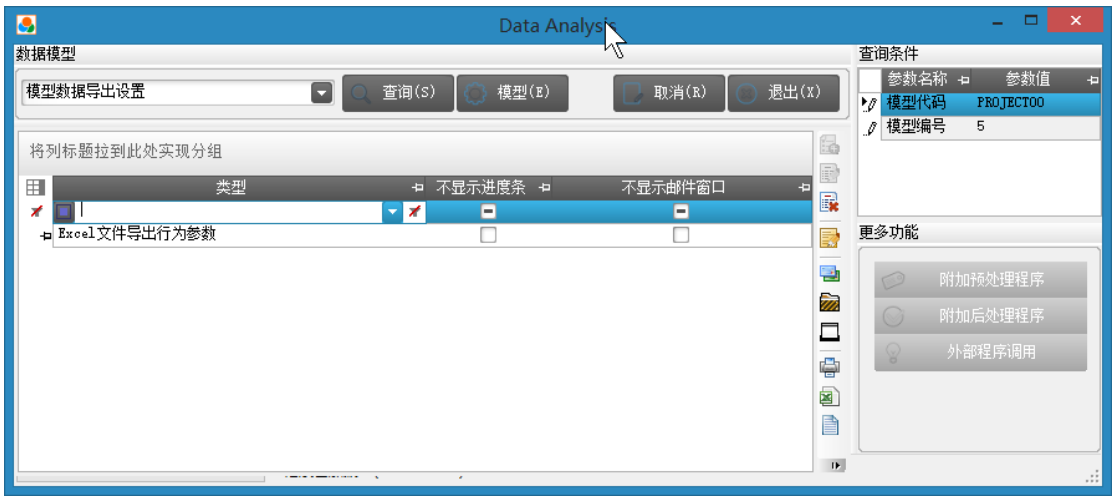
- 数据查询的结果通过 Excel 导出之后会自动加载邮件发送功能
- 邮件发送的收件人、抄送人、主题、正文、附件等内容均是根据模型的设置自动生成，点击发送即可完成提交邮件发送任务

16 文件导出设置

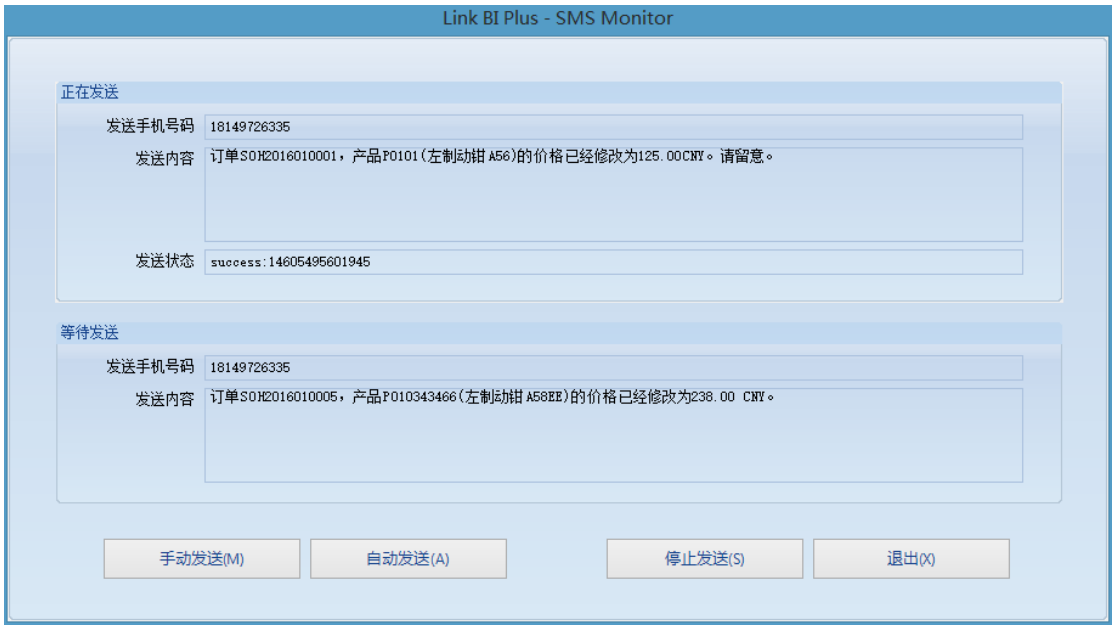
针对模型的 Excel 导出功能，默认会出现数据导出进度条，用来指示已经导出的行数量。这种导出模式下，导出的速度会稍慢。根据模型设置，可以关闭导出进度条。

另外，数据导出完毕，系统默认会打开邮件发送窗口，直接以导出的文件作为附件发送邮件。根据需要，也可以关闭邮件发送窗口。



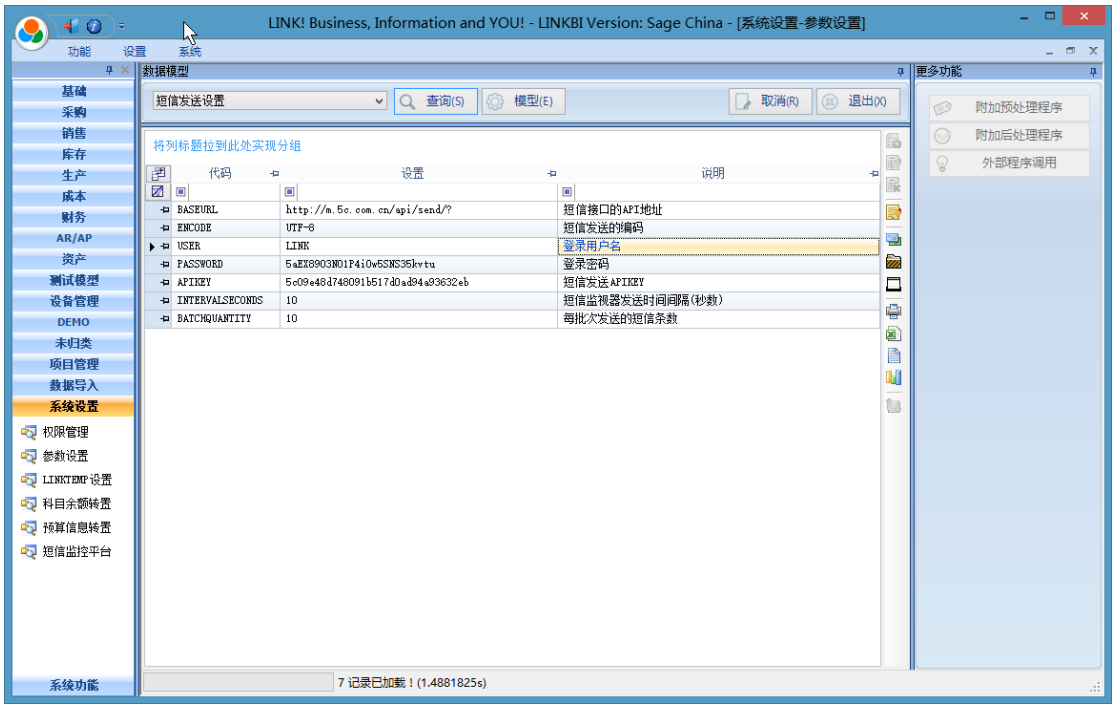


17 短信平台



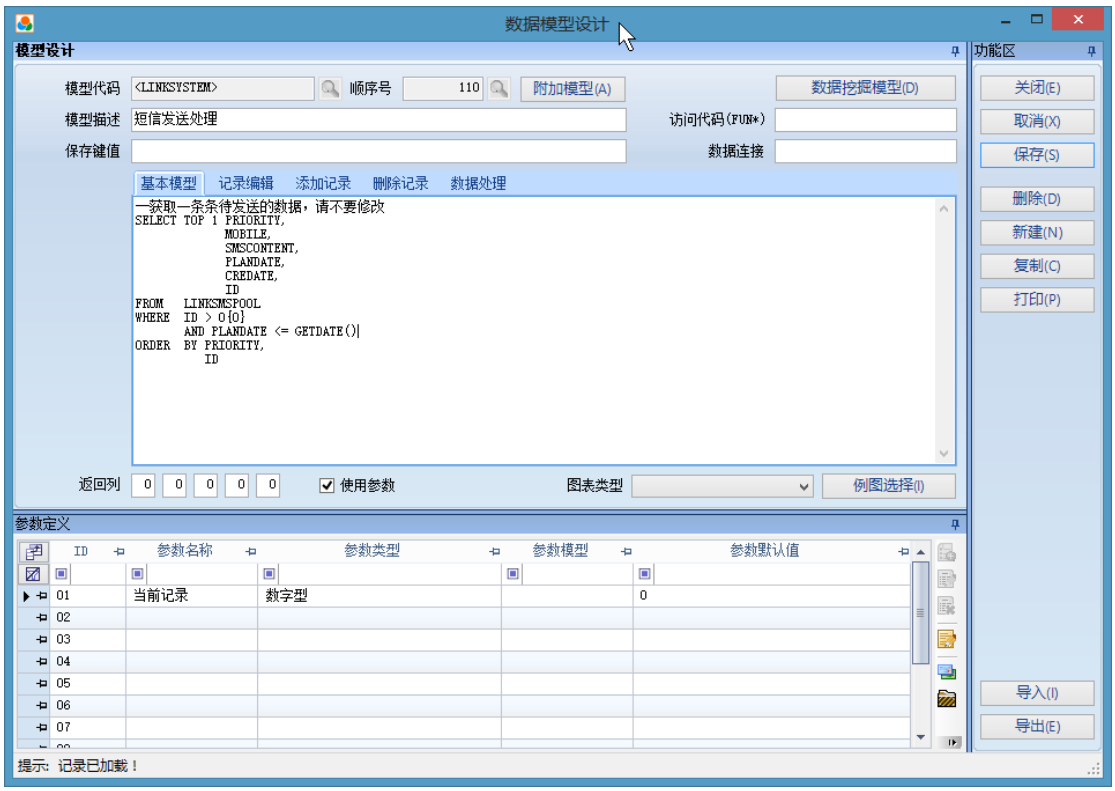
- 如图所示，系统具备完善的短信发送平台，可以实现对短信缓冲区的检查并发送短信。
- 短信发送不需要增加额外的通信设备，只需要注册账号进行相应的配置即可。
- 可以通过短信平台发送信息系统的各种内容
  - 订单通知
  - 会议通知
  - 发货通知
  - 款项收到信息
  - 质检不良信息
  - 注册验证码
- 所有短信的内容可以方便定制，通过 TotalLINK 系统的模型配置及参数配置即可实现各种信息系统的短信通知功能

18 短信平台的配置



- 如图所示，对于短信平台需要配置的信息包括：
  - BASEURL，短信接口平台的地址
  - ENCODE，短信编码
  - USER，短信发送平台注册的用户名
  - PASSWORD，短信发送平台密码（需注意密码规则）
  - APIKEY，短信发送平台的 APIKEY 值
  - INTERVALSECONDS，短信发送平台的间歇时间间隔（秒数）
  - BATCHQUANTITY，每个发送时间间隔间作为一批发送，这个数字表示发送条数
- 短信内容，根据需要填写到短信缓冲区即可
- 每次发送后，会将短信的内容转移到历史存储区
- 可以为短信设置发送时间，到达设定发送时间的短信才会发送
- 可以为短信设置发送优先级，优先级高的内容优先发送（可以插队处理）

19 短信处理模型



- 具体短信的缓冲区及历史存储区处理，参照系统模型 <LINKSYSTEM>, 110 模型的内容
- 该模型包含了获取短信内容的脚本及短信转移处理的脚本
- 短信数据读取脚本

--获取一条条待发送的数据，系统模型，请谨慎修改

```
SELECT TOP 1 PRIORITY,
             MOBILE,
             SMSCONTENT,
             PLANDATE,
             CREDATE,
             ID
FROM     LINKSMSPOOL
WHERE    ID > 0{0}
         AND PLANDATE <= GETDATE()
ORDER    BY PRIORITY,
            ID
```

- 短信插入到历史存储区脚本

--处理一条数据到历史表中，系统模型，该语句与删除在同一个事务中处理

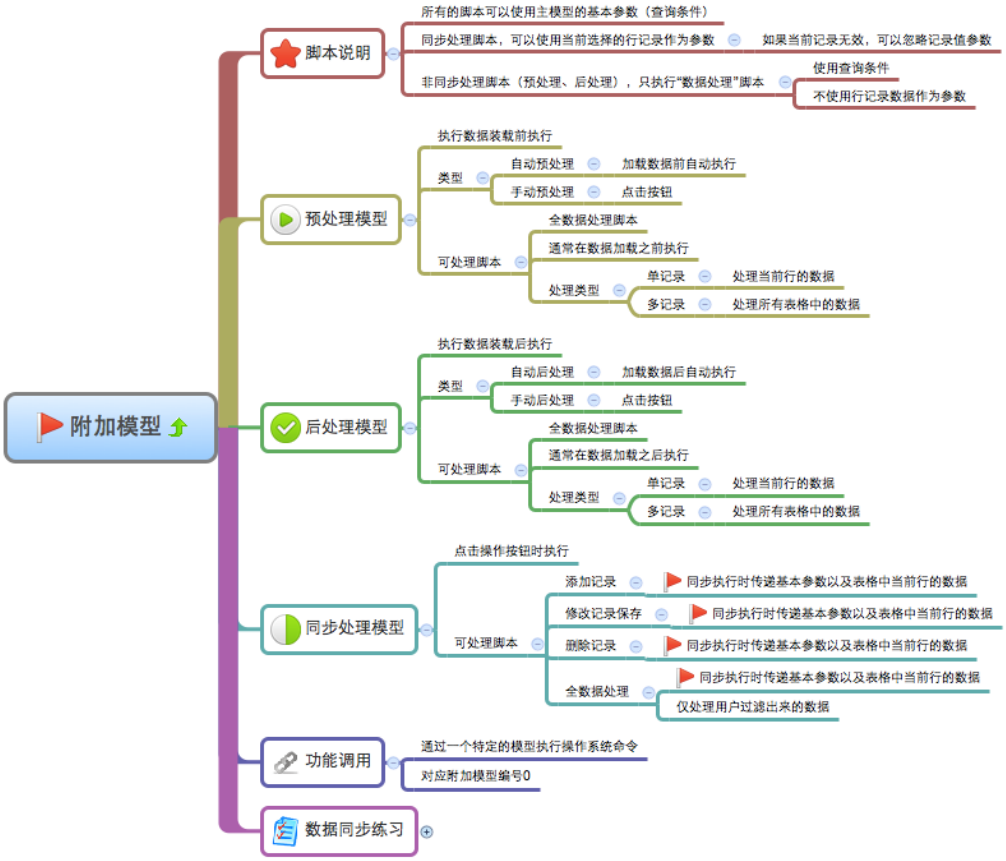
```
INSERT INTO LINKSMSHIS
                    (PRIORITY,
                     MOBILE,
                     SMSCONTENT,
                     PLANDATE,
                     CREDATE,
                     SENDRESULT,
                     ORIID)
SELECT PRIORITY,
          MOBILE,
          SMSCONTENT,
          PLANDATE,
          CREDATE,
          '{1}',
          ID
FROM     LINKSMSPOOL

WHERE    ID = {0}
```

- 短信从缓冲区清除脚本

  --删除一条记录，此脚本与添加到临时表的脚本使用同一个事务处理

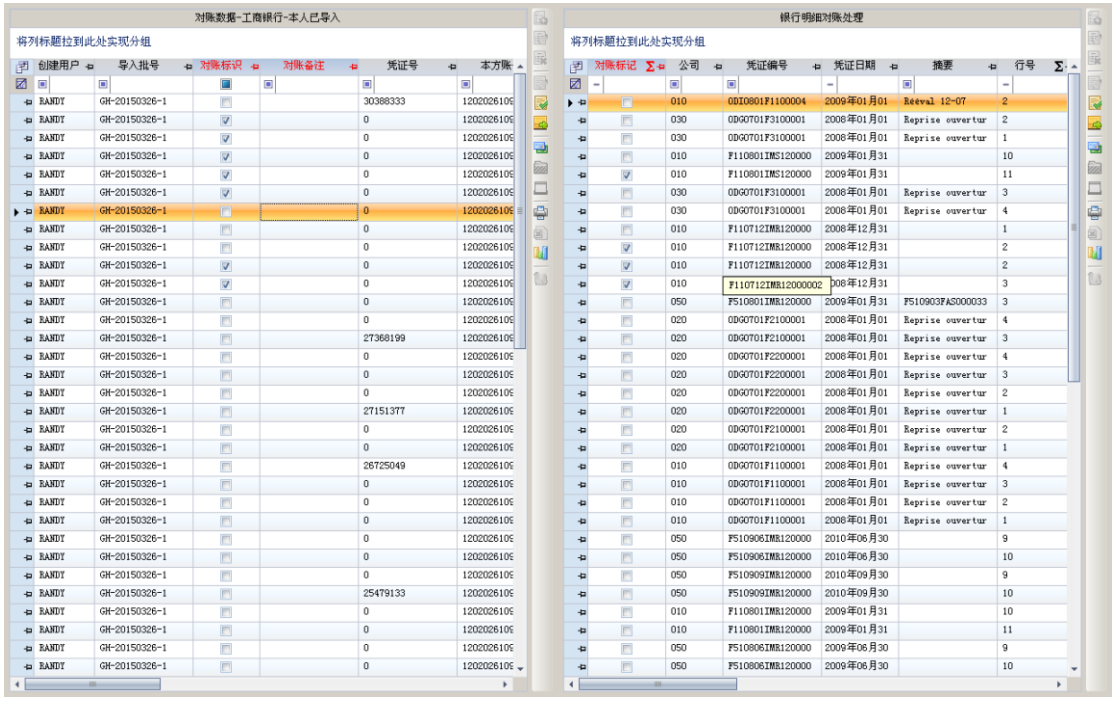
```
DELETE FROM LINKSMSPOOL
WHERE    ID = {0}
```

20 附加模型功能



- 有关附加模型的脚本说明及可以使用的参数、数据等见上图的简要说明

21 多数据源操作

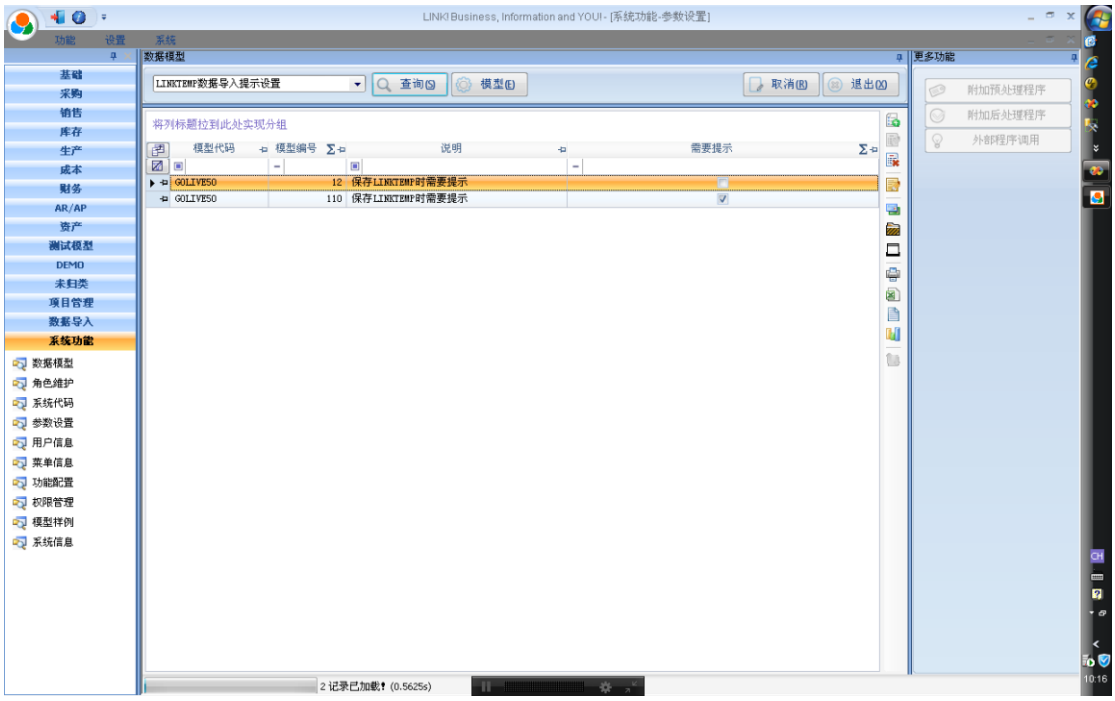


- 系统允许多数据源同时操作
  - 比如通过简单设置即可实现“银行对账”功能
  - 在进行银行对账时，可以方便的一侧显示原始银行记录，一侧显示系统的相关会计凭证
  - 两侧的内容可以分别编辑，进行方便的对账操作

22 临时数据保存设置

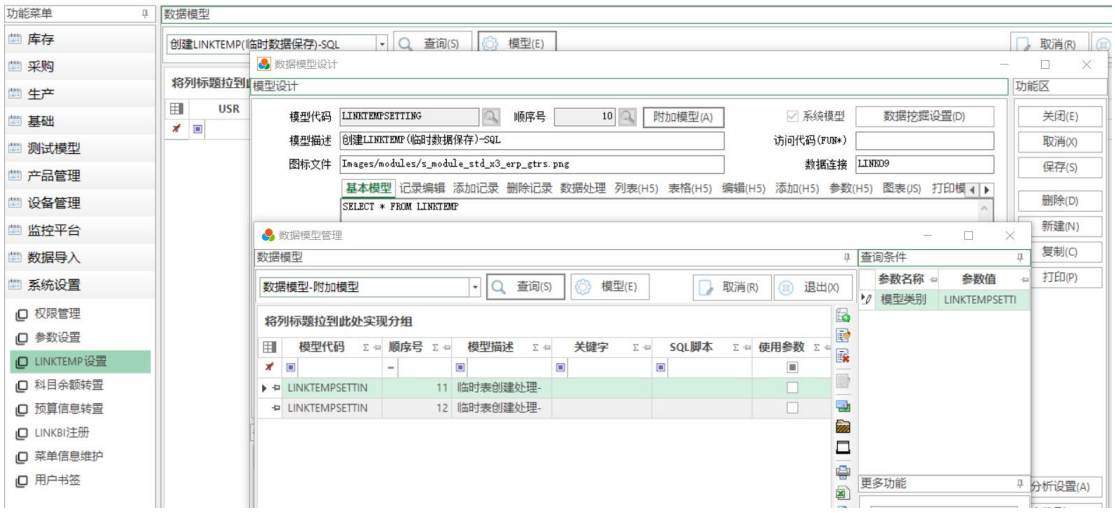
- 系统预留了 LINK09(TEMP)作为临时数据保存的链接
- 如果某个模型的“保存键值”非空白的时候，会自动执行保存操作
- 系统自动将模型查询的内容根据“保存键值”的设置保存的 LINK09(TEMP)对应链接的 LINKTEMP 中
- 每次自动保存前系统会根据“登录用户、登录电脑、保存键值、模型代码、模型号码”清除上次已经保存的内容
- 如果用户需要保留临时保存到 LINKTEMP 的内容，可以通过后处理模型转移到其他表文件中

23 提示用户保存临时数据



- 对于设置了保存键值的模型，系统会自动根据需要将查询的内容保存到 LINKTEMP 表中（对应 LINK09(TEMP)链接）
- 系统默认不提示用户，而直接根据模型的保存键值设置完成自动保存
- 如果需要提示用户保存信息，可以通过“参数设置-LINKTEMP 数据导入提示设置”功能设置模型需要显示提示保存

24 创建临时数据表 LINKTEMP

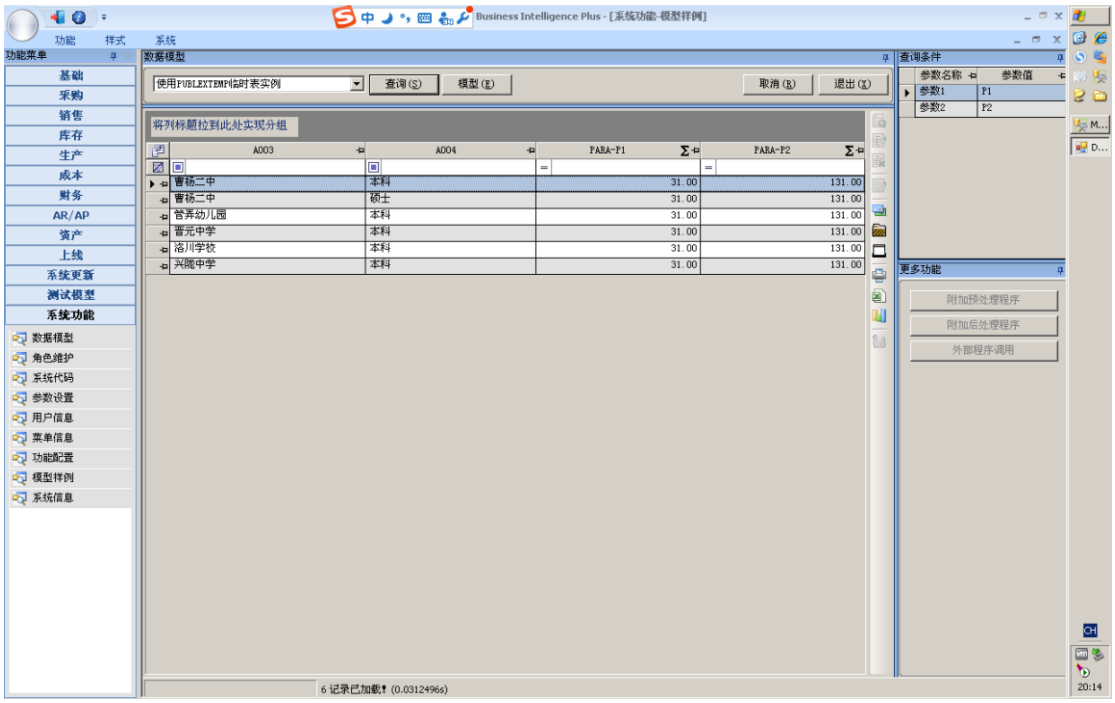


- 在使用临时表数据保存功能前，需要在 LINK09(TEMP)数据链接对应的数据库系统中创建 LINKTEMP 表
- 如图所示，可以根据数据库系统分别选择创建 SQL 或 ORACLE 系统的 LINKTEMP 表
- 执行“系统设置-LINKTEMP 设置-创建 LINKTEMP(临时保存数据)-SQL/ORACLE”的数据查询操作
- 系统会在 LINK09(TEMP)对应的数据库系统中创建 LINKTEMP 表，用户临时数据保存

25 巧用临时数据处理

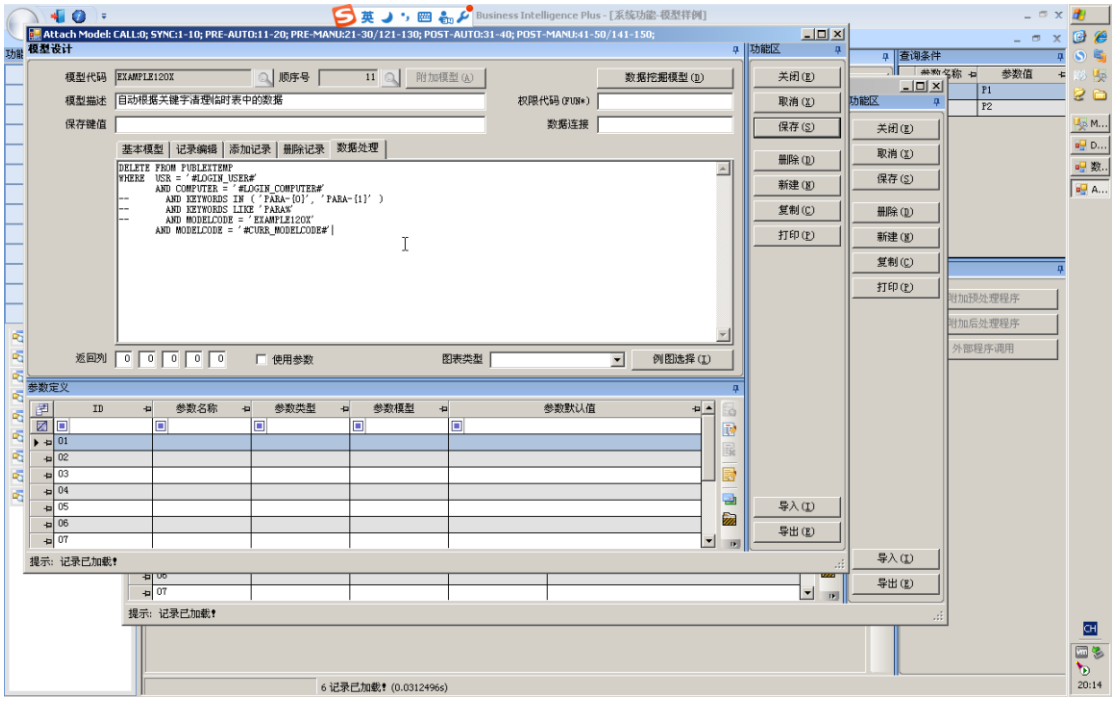
- 在设计数据互联模型时，巧妙地使用临时数据有时可以大大简化模型的复杂程度
- 在使用临时数据处理时，基本的模式可以参考下面的思路
  - 设计主数据模型，可以最终从临时表中获取数据
  - 模型的执行过程如下：
    - 执行附加自动预处理模型 11，根据条件删除临时表记录
    - 系统会自动删除上次同样的用户、电脑、保存键值、模型代码、模型号码的记录
    - 执行附加自动预处理模型 12，从 X3 数据库中查询一组数据并保存到临时表中(使用第一组参数)
    - 执行附加自动预处理模型 13，从 X3 数据库中查询一组数据并保存到临时表中(使用第二组参数)
    - 主模型加载执行，加载临时表中的数据到用户界面
    - 执行附加自动后处理模型 31，清除临时表中的数据
    - 如果不通过后处理模型清除，系统在下次加载数据时会自动清除 LINKTEMP 对应的数据
    - 如果需要永久保留数据，需要使用后处理脚本将数据转移到其他的表中

26 使用临时表模型数据设计实例一



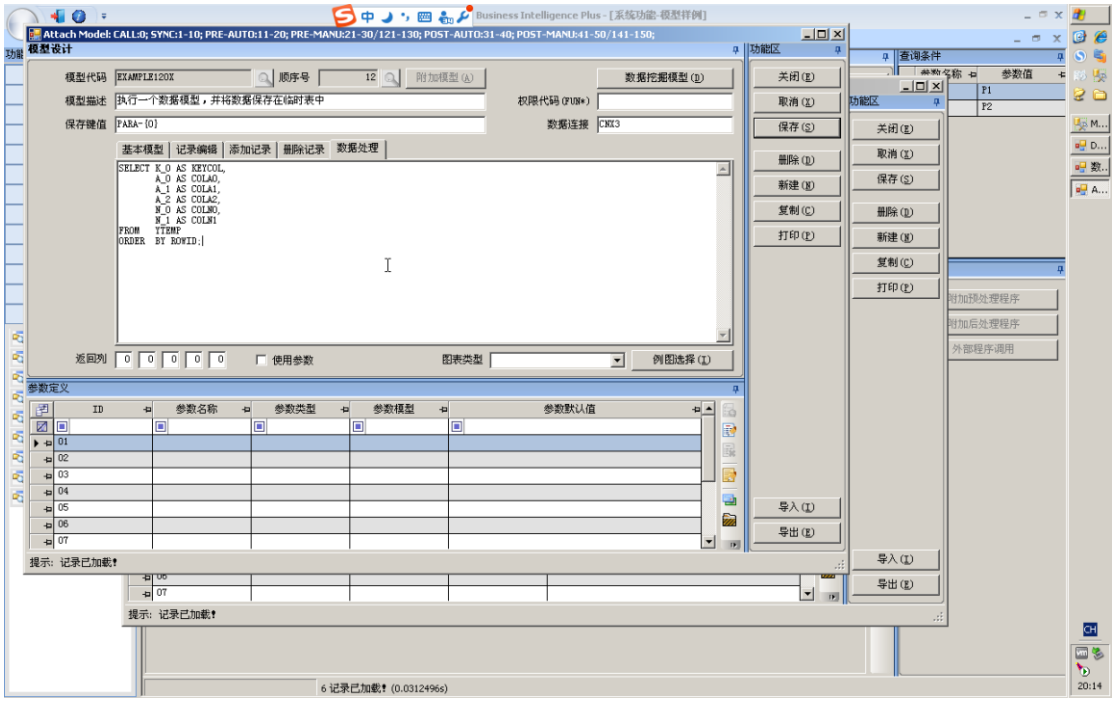
- 本例说明在将数据展现给用户前，先利用临时表进行保留数据的基本模式
- 本模型定义两个参数 P1/P2，在附加的自动预处理模型中分别使用不同的参数执行查询并将结果保存下来，然后利用转置模型将数据展现给用户
- 上图为主模型的展现结果
- 本模型的脚本如下：

```
SELECT A003,
       A004,
       [PARA-{0}],
       [PARA-{1}]
FROM   (SELECT KEYWORDS,
               A003,
               A004,
               CAST('0' + A005 AS DECIMAL(18, 2)) A005
        FROM   PUBLEXTEMP
        WHERE  MODELCODE = '#ATTACH_MODELCODE#') TEMP
       PIVOT ( SUM(A005)
             FOR KEYWORDS IN ([PARA-{0}],
                              [PARA-{1}]) )PVT
ORDER  BY A003,
          A004
```



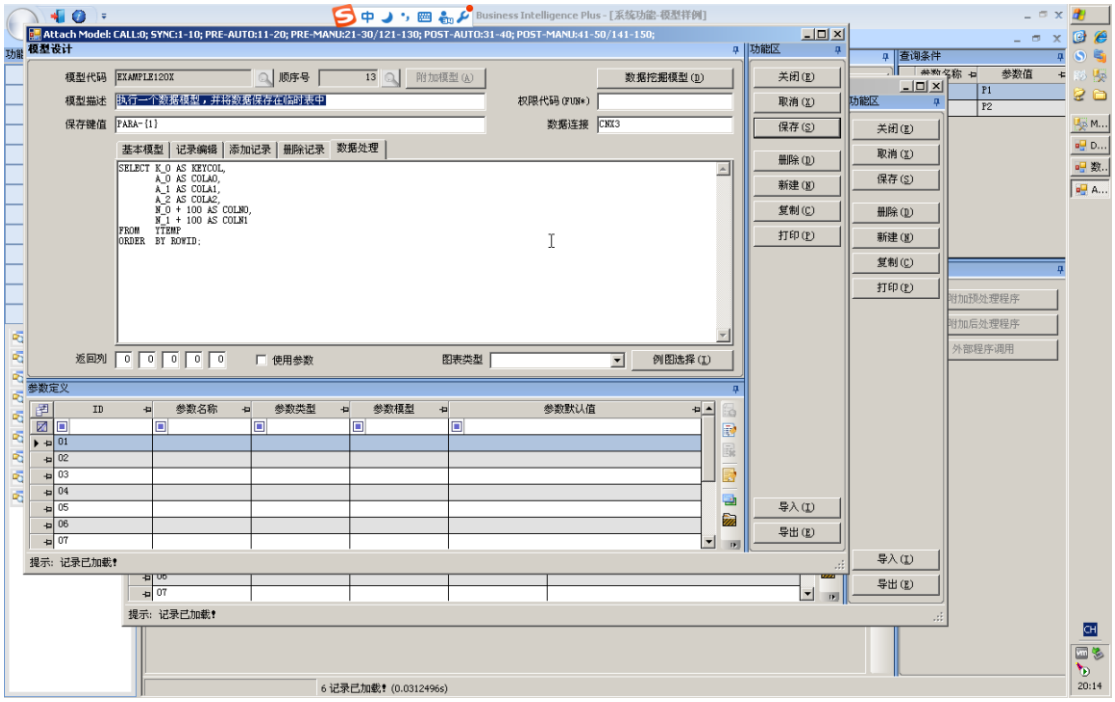
- 定义第一个自动预处理模型
- 用户清除对应的临时数据
- 脚本如下：

```
DELETE FROM PUBLEXTEMP
WHERE  USR = '#LOGIN_USER#'
       AND COMPUTER = '#LOGIN_COMPUTER#'
       AND MODELCODE = '#CURR_MODELCODE#'
```



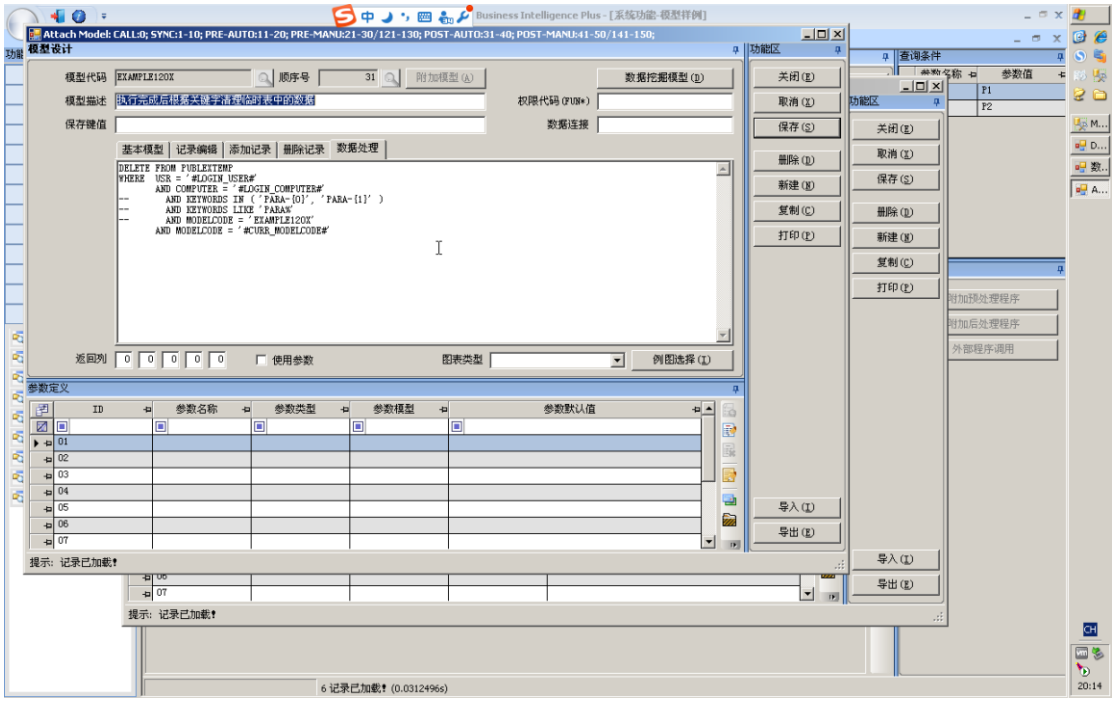
- 定义第二个自动预处理模型
- 从系统中查询数据
- 保存键值：PARA-{0}，表示保存键值（KEYWORDS）为 PARA-加第一个参数的记录
- 脚本如下：

```
SELECT K_0 AS KEYCOL,
       A_0 AS COLA0,
       A_1 AS COLA1,
       A_2 AS COLA2,
       N_0 AS COLN0,
       N_1 AS COLN1
FROM   YTEMP
ORDER  BY ROWID;
```



- 定义第三个自动预处理模型
- 从系统中查询数据
- 保存键值：PARA-{1}，表示保存键值（KEYWORDS）为 PARA-加第二个参数的记录
- 脚本如下：

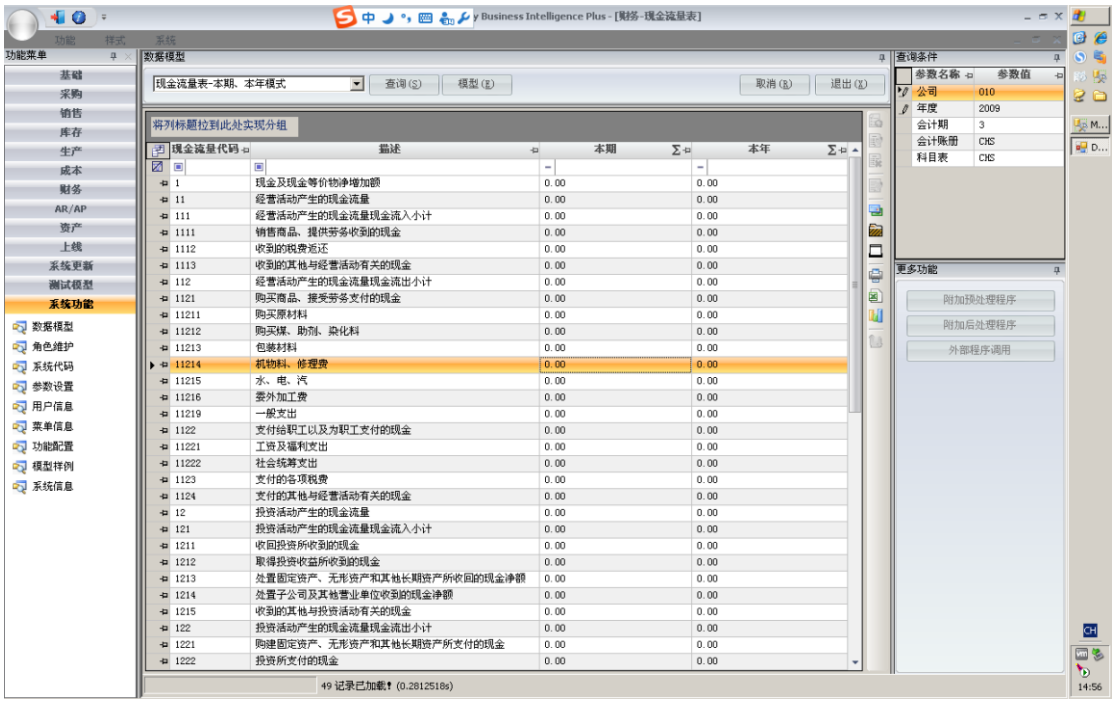
```
SELECT K_0 AS KEYCOL,
       A_0 AS COLA0,
       A_1 AS COLA1,
       A_2 AS COLA2,
       N_0 + 100 AS COLN0,
       N_1 + 100 AS COLN1
FROM   YTEMP
ORDER  BY ROWID;
```



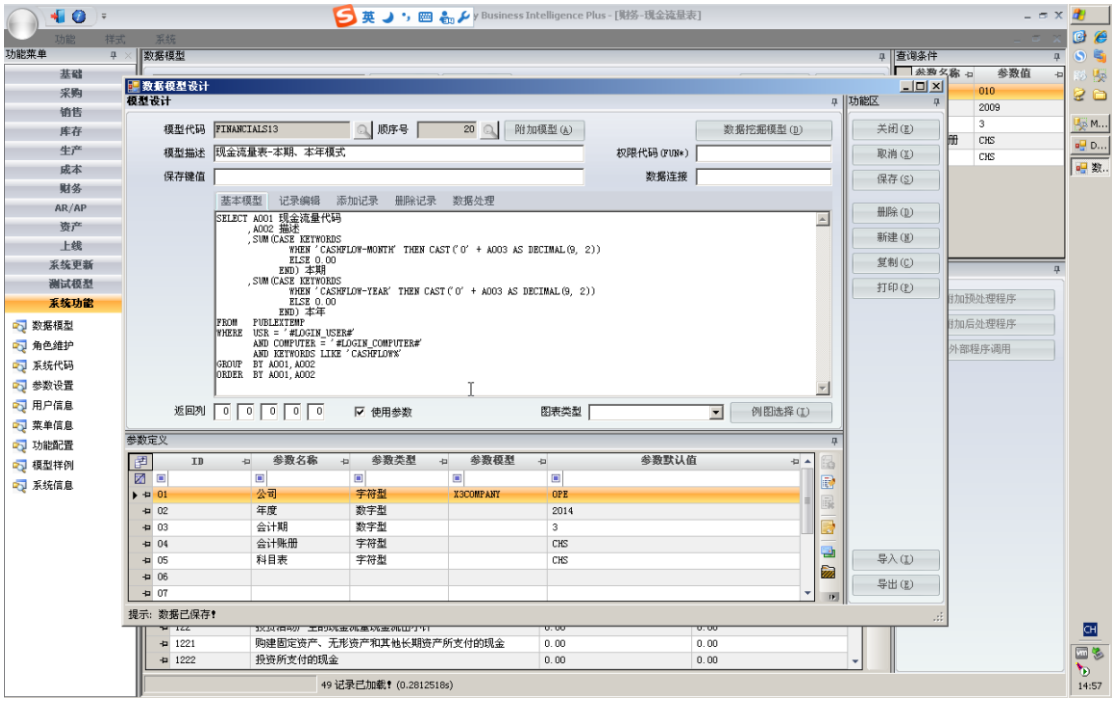
- 定义第四个附加模型，此模型的编号为 31，表示是一个自动后处理模型
- 用于数据展现之后清除相关数据
- 模型脚本如下：

```
DELETE FROM PUBLEXTEMP
WHERE  USR = '#LOGIN_USER#'
       AND COMPUTER = '#LOGIN_COMPUTER#'
       AND MODELCODE = '#CURR_MODELCODE#'
```

27 使用临时表模型模型设计实例二



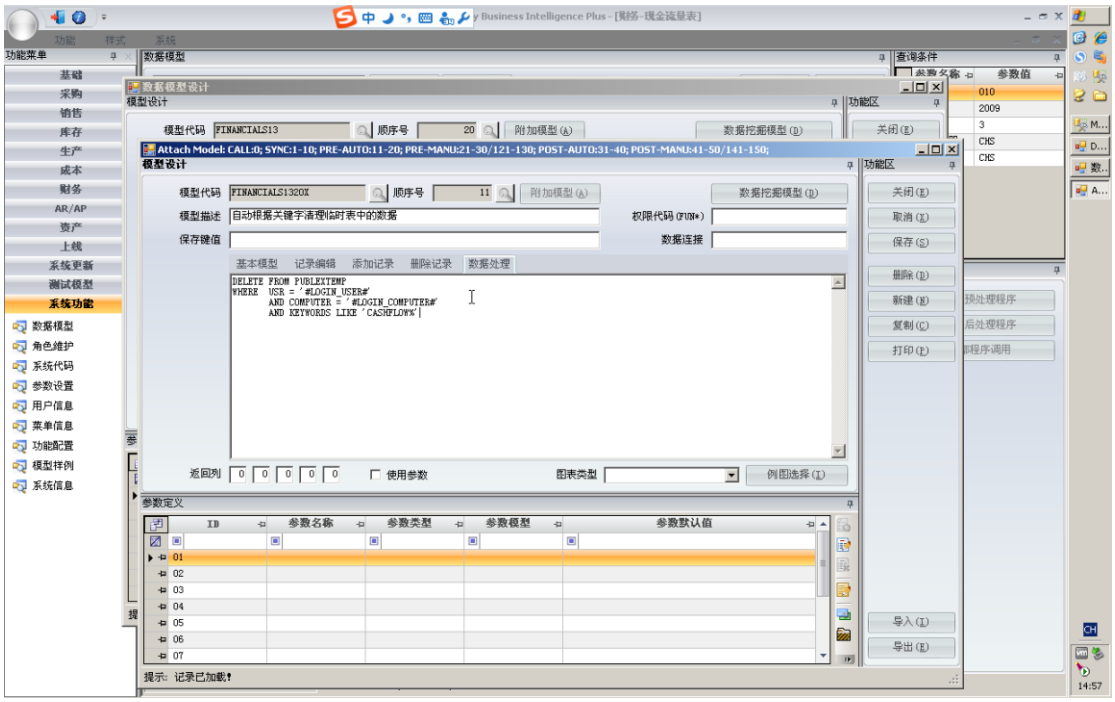
- 上图中设计了一个包含本期和本年数据的现金流量表
- 从 X3 ERP 系统获取数据时计算本期和本年的现金流量的逻辑是一致的，因此我们设计一个取数模型，然后使用不同的参数用于实现方便获取本期和本年数据



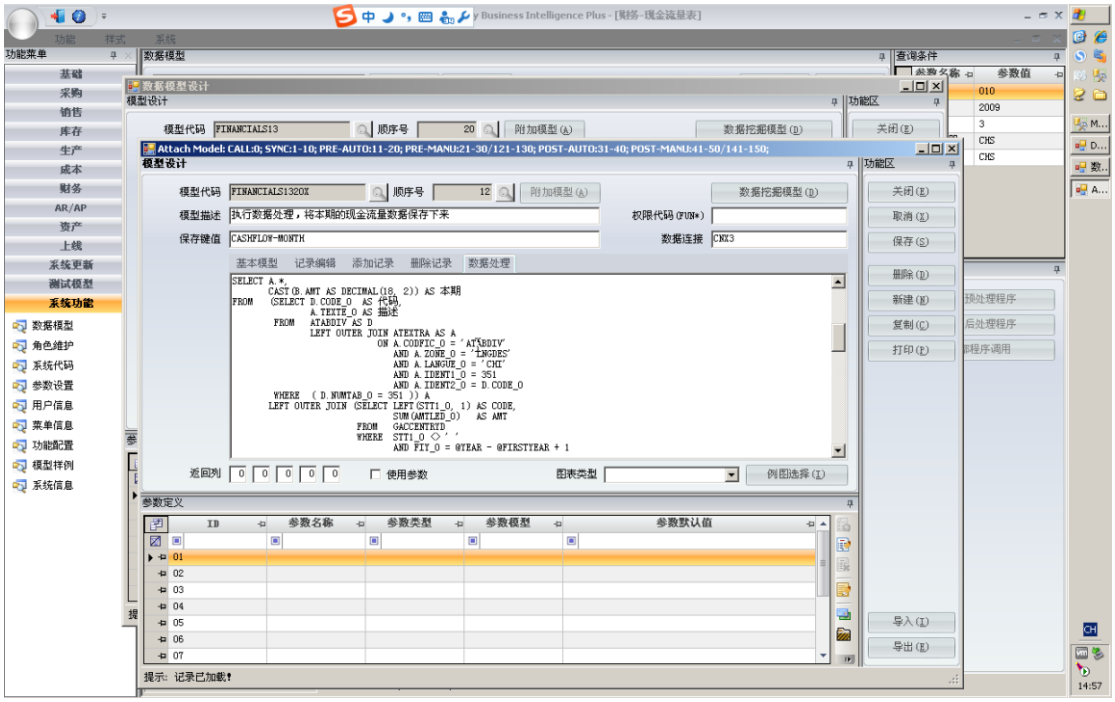
- 这里是数据主模型，也就是最终加载前面已经执行完成的结果给用户

- 数据模型如下：

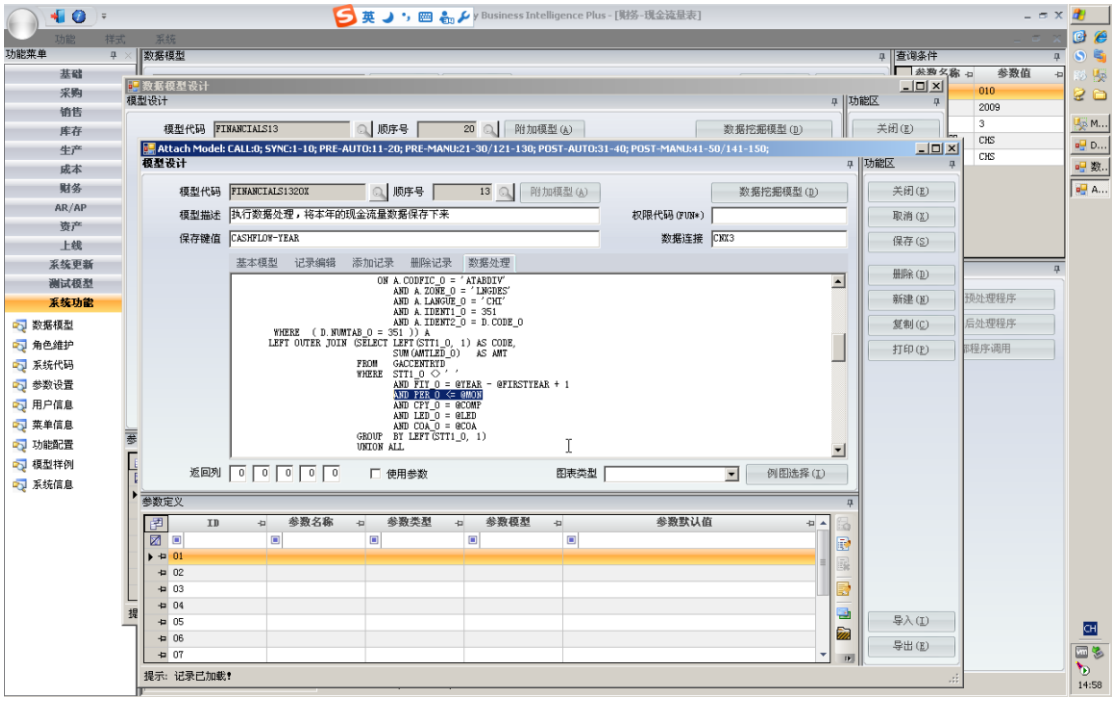
```
SELECT A001 代码,
       A002 描述,
       [本期],
       [本年]
FROM   (SELECT A001,
               A002,
               KEYWORDS,
               CAST('0' + A003 AS DECIMAL(9, 2)) A003
        FROM   PUBLEXTEMP
        WHERE  MODELCODE = '#ATTACH_MODELCODE#') TEMP
       PIVOT ( SUM(A003)
             FOR KEYWORDS IN ([本期],
                              [本年]) )PVT
ORDER  BY A001
```



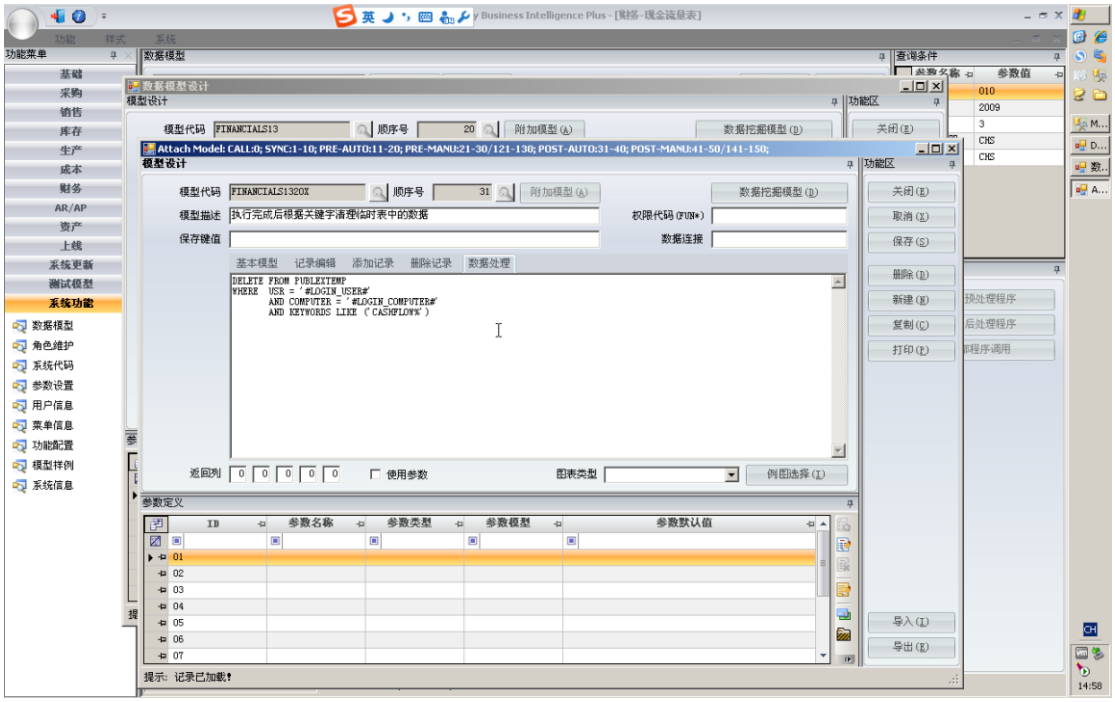
- 进行模型计算前，清除当前用户的历史遗留临时数据（如果有）



- 使用 PERIOD = @MON 作为参数执行，获取本期数据并自动保存

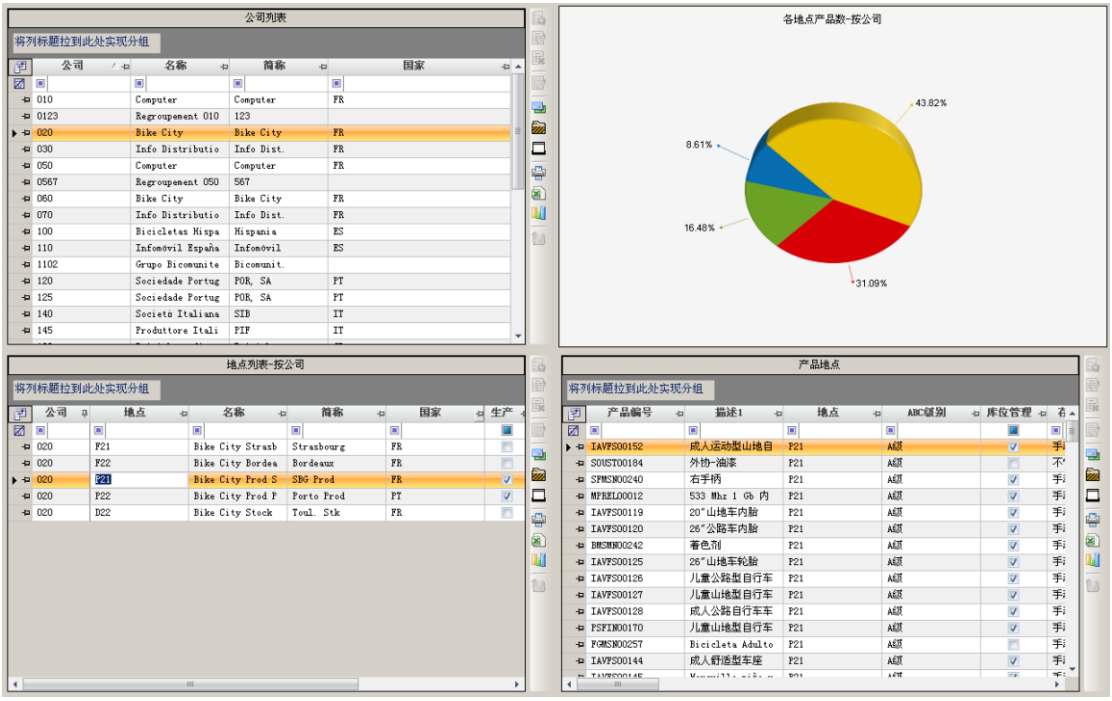


- 使用 PERIOD <= @MON 作为参数执行，获取本年数据并自动保存
- 此时，数据执行完成，用户可以看到将本期和本年加载在一起的数据



- 数据加载完成后，根据条件清理临时表的数据

28 图表的相关关联



- 某些情况下，在同一屏幕中展现的各项内容是需要有相互关联的。以数据互联系统的术语来说，就是一个模型的参数依赖与另外一个模型的数据。

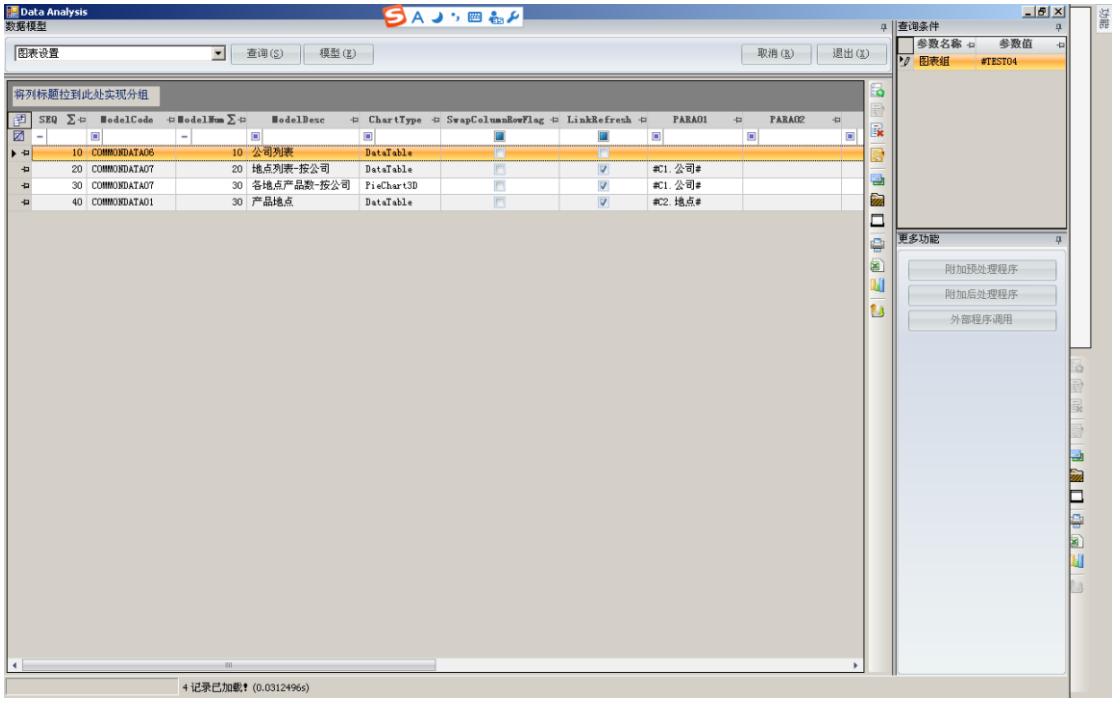
- 比如上图中，四个图表的内容分别是：
  - 公司列表

  内容独立（该模型没有参数）
  - 对应公司的地点列表

  当选择的公司改变时，该表格显示的是对应公司的所有地点信息
  - 对应公司的各个地点的产品记录数（饼图）

  当选择的公司改变时，用饼图显示该公司所有地点的“产品-地点”的记录数
  - 对应地点的“产品-地点”数据

  改变公司、地点时，该表格显示的时对应的产品-地点数据



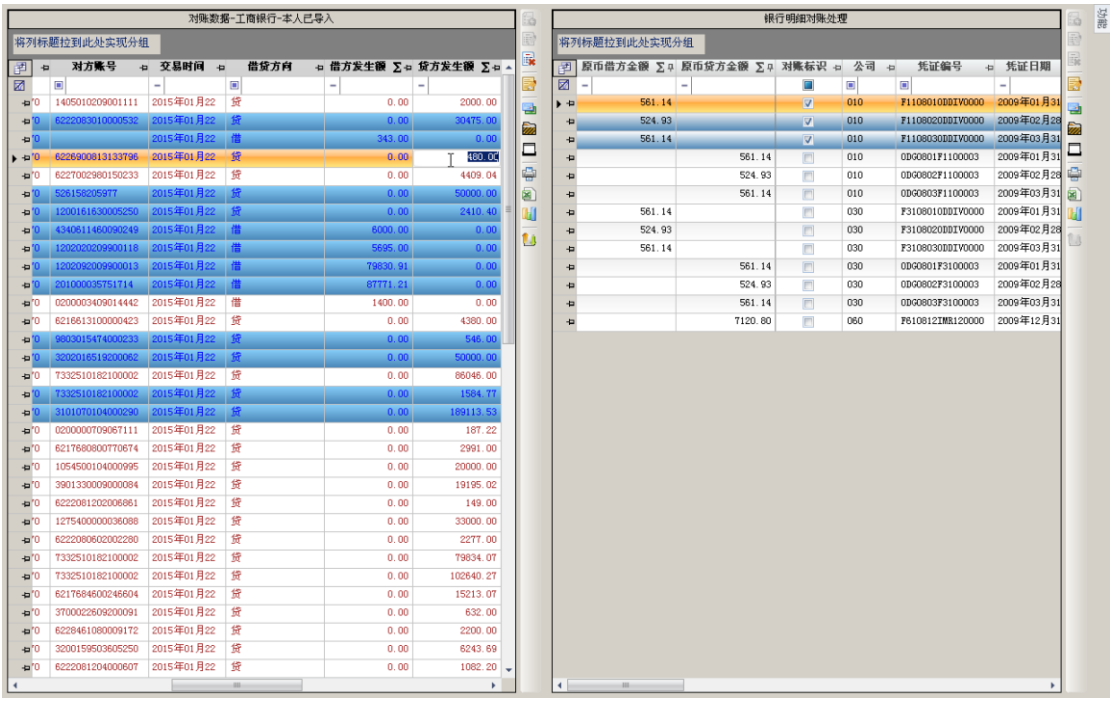
- 上图说明了为了实现上述功能的简单设置
  - 将 4 个简单的模型组合起来
  - 第一个模型显示公司信息
  - 第二个模型显示公司的地点信息，这里设定此模型可以自动根据关联数据刷新，其参数来自第一个表格的公司字段，因此设置参数为“#C1.公司#”
  - 第三个模型显示公司的各个地点产品记录数，同样设定此模型可以自动根据关联数据刷新，其参数来自第一个表格的公司字段，因此设置参数为“#C1.公司#”
  - 第四个模型显示对应地点的“产品-地点”数据，设定此模型可以自动根据关联数据刷新，其参数来自第二个表格的地点字段，因此设置参数为“#C2.地点#”

29 图表关联参数设置约定

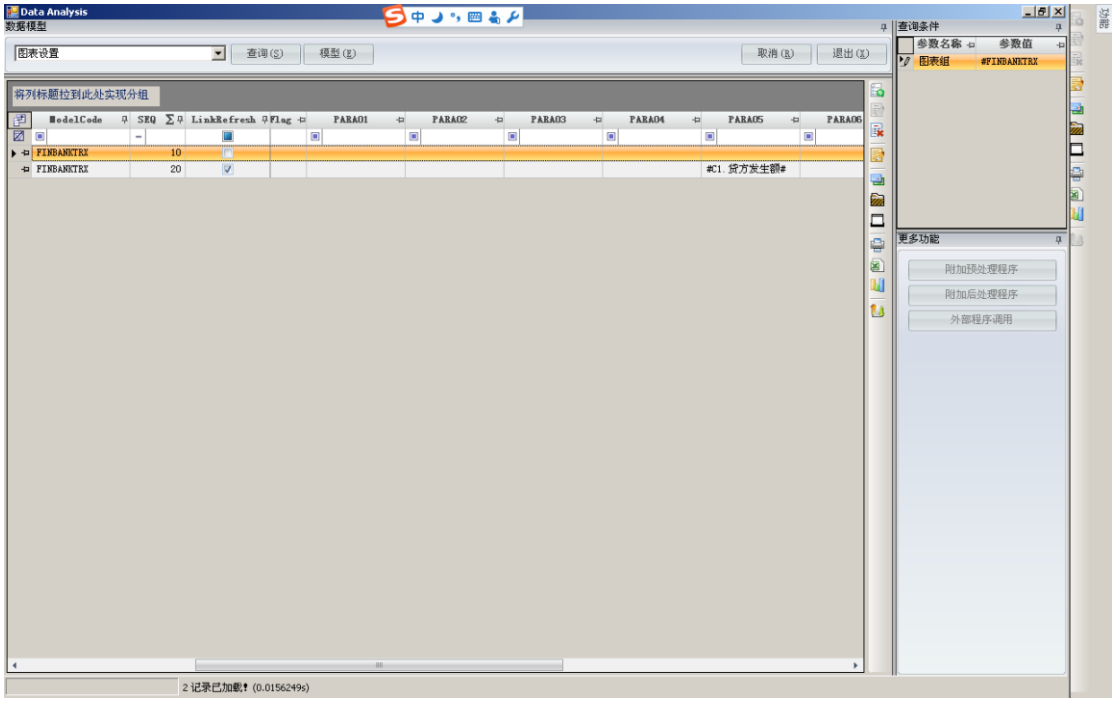
- 当需要设置不同的图表数据相互关联时，按照类似“C1.公司”的方式设置参数的传递
- 其中 C1-C8 为对应的表格的名字
- “.”后面的为表格中数据的字段名

30 银行对账模型配置

- 利用多个关联，可以快速配置出不同的各类复杂的应用。
- 实际上，各个模型可以不仅仅来自同一个数据源，并且各个模型都可以实现“编辑、保存、添加、删除”等功能。
- 比如，参照下面的例子，可以快速实现银行对账功能。

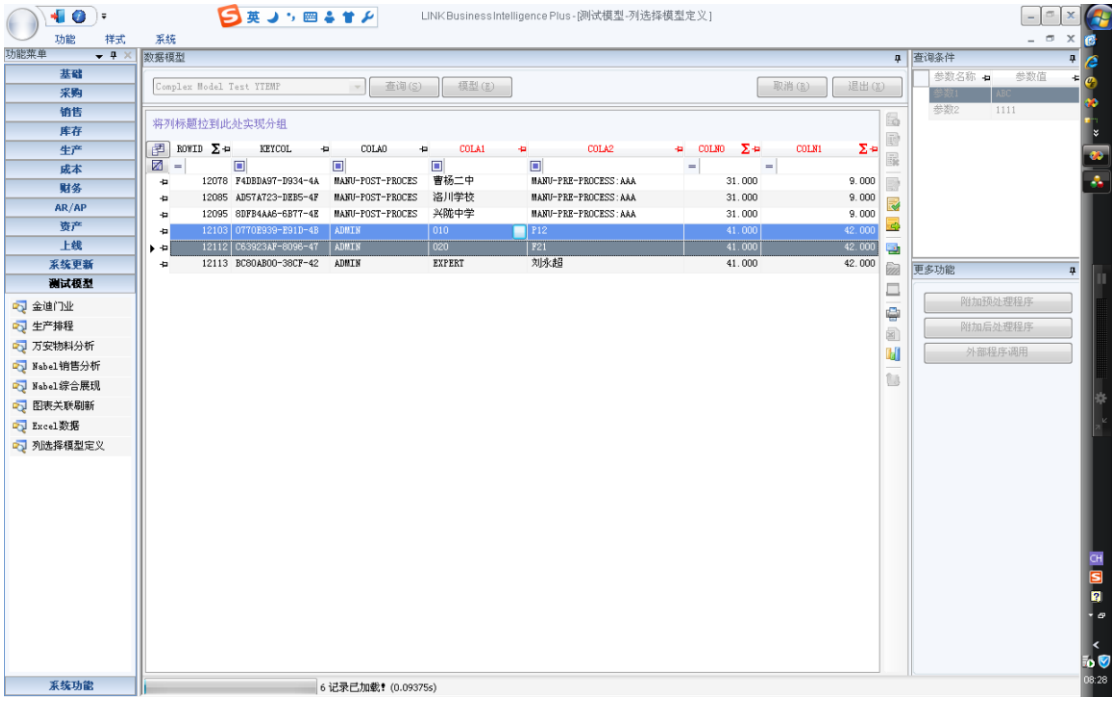


- 在这个例子中，左侧列表显示了来自银行的以 Excel 保存的银行交易数据
- 右侧列表显示了 ERP 系统中的对应银行的交易数据
- 分别定义独立的编辑、保存功能，用户可以快速设置每一条记录的对账标记
- 利用系统的本身技术特点，用户可以非常方便的实现：
  - 两边都按照金额排序，快速判断是否可以对账
  - 两边都可以设置数据检索条件，将已经对账的内容过滤掉，不再显示
- 利用本节描述的功能，还可以为第二个模型设置一个特殊的条件，比如金额与第一个表格的交易金额一致
- 这样，当第一个表格中选择了一条记录后，后面一个表格的数据可以跟随改变，以便为用户提供更加方便的操作。


Data Analysis
数据模型
图表设置
查询(S)
模型(E)
取消(B)
退出(X)
将列标题拉到此处实现分组
ModelCode
SEQ
LinkRefresh
Flag
PARA01
PARA02
PARA03
PARA04
PARA05
PARA06
FINBANKTRX
10
FINBANKTRX
20
#C1. 贷方发生额#
查询条件
参数名称
参数值
图表组
#FINBANKTRX
更多功能
附加预处理程序
附加后处理程序
外部程序调用
2 记录已加载! (0.0156249s)

31 数据编辑、添加、删除等复杂模型

通过对模型的简单设置，可以实现复杂的编辑功能。



- 如上图所示，在系统中定义了临时表 YTEMP
- 通过模型可以查询出相应的数据，并且 COLA1/COLA2/COLN1 等列的内容是允许用户编辑并保存的。
- 此功能可以通过设置脚本的编辑模型实现
  - 基本模型脚本

```
SELECT ROWID
      ,K_0 AS KEYCOL
      ,A_0 AS COLA0
      ,A_1 AS COLA1
      ,A_2 AS COLA2
      ,N_0 AS COLN0
      ,N_1 AS COLN1
FROM   YTEMP

ORDER  BY ROWID;
```

  - 数据编辑模型脚本

```
UPDATE YTEMP
SET    A_1 = N'@COLA1@',
       A_2 = N'@COLA2@',
```

```
            N_0 = N'@COLNO@',
            N_1 = '@COLN1@'
WHERE   ROWID = '#ROWID#'
```

- 记录添加模型脚本

```
INSERT INTO YTEMP
VALUES      (CAST(NEWID() AS CHAR(50)),
             '#LOGIN_USER#',
             '#LOGIN_ROLE#',
             N'{0}',
             #COLNO# + 10,

#COLN1# + 0{1});
```

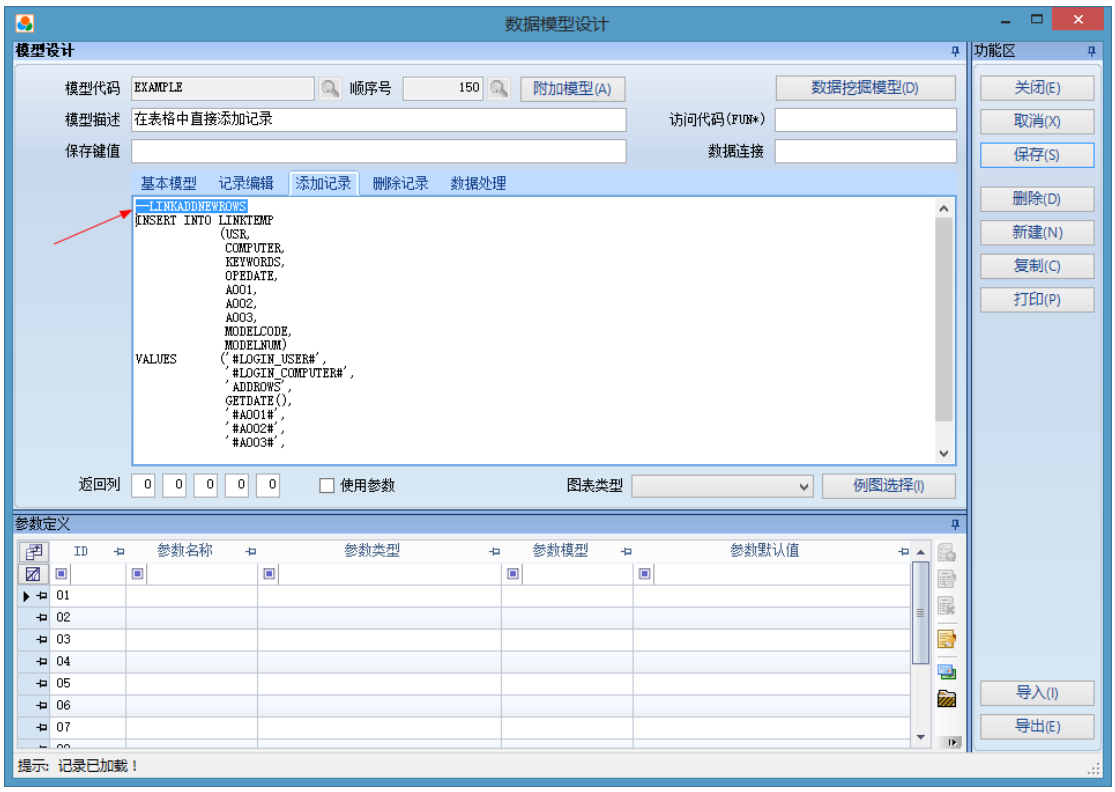
- 单行记录删除脚本

```
DELETE FROM YTEMP
WHERE   ROWID = '#ROWID#';
```

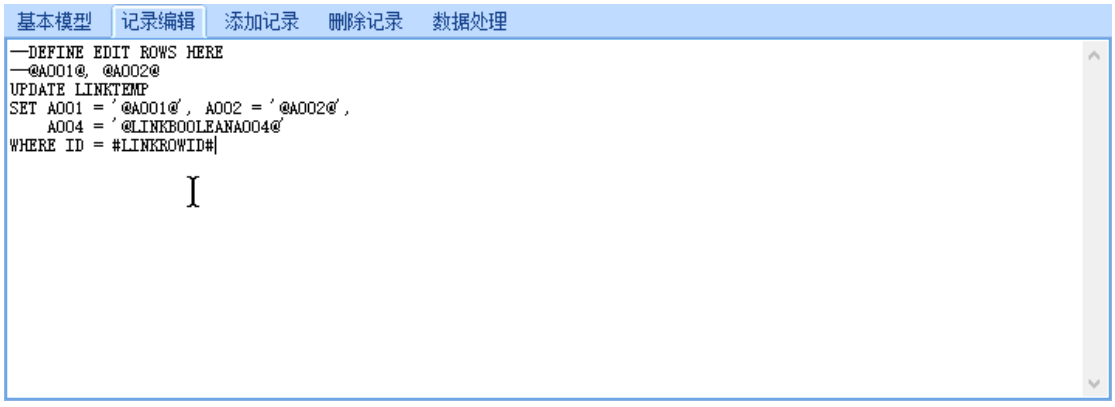
- 多行记录删除脚本

```
DELETE FROM YTEMP
WHERE   ROWID = '#ROWID#';
```

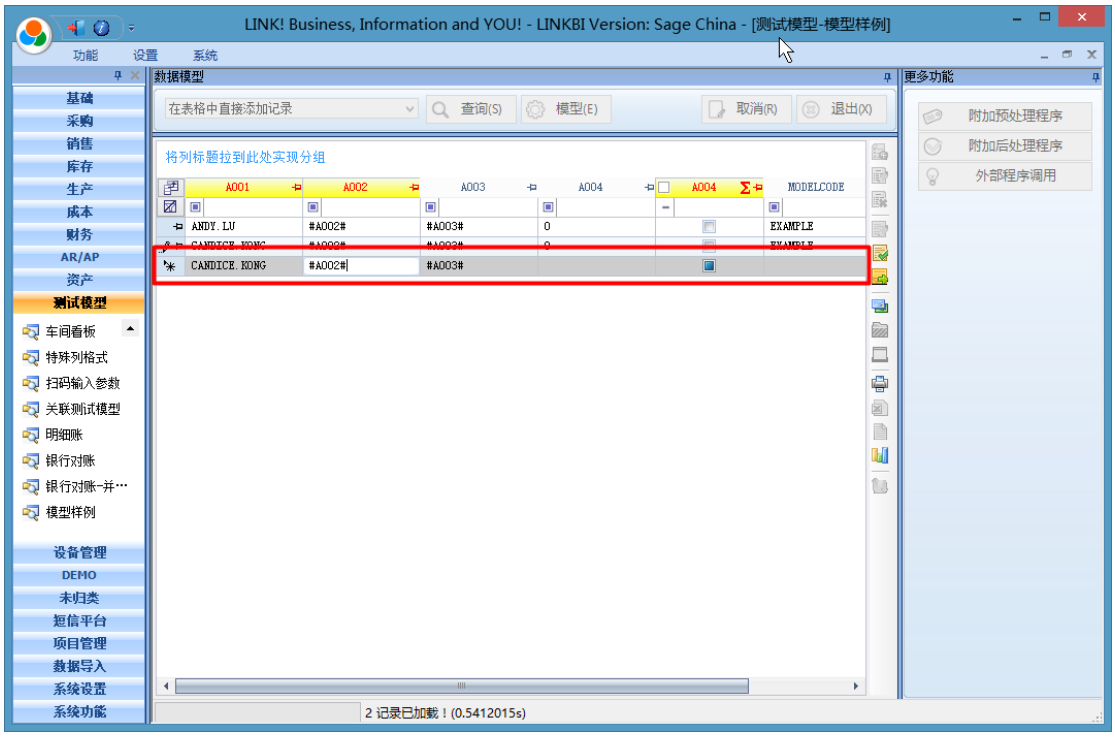
32 利用添加行添加记录--LINKADDNEWROWS



- 当模型的添加记录脚本的第一行以特殊的标记开始时，系统任务该模型在编辑时可以开启行添加功能
- 这个特殊的标记是：--LINKADDNEWROWS
- 开启此功能后，在数据的最后以后允许用户输入新的记录
- 可以编辑的列仍然由记录编辑模型定义



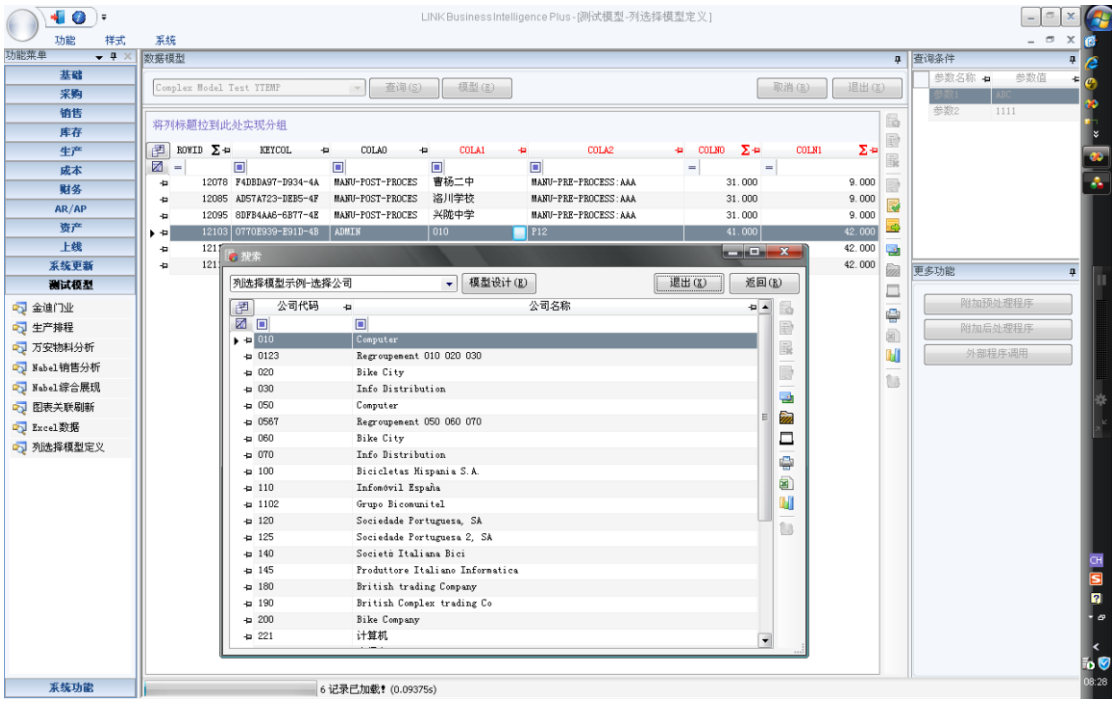
- 表格中的数据编辑时，会自动获得上一行的数据内容，并且允许用户编辑
- 保存数据时，系统会自动判断是添加的新行记录还是已经存在的记录，并对其进行分别处理
- 如下图所示



33 编辑模式下单元格的移动

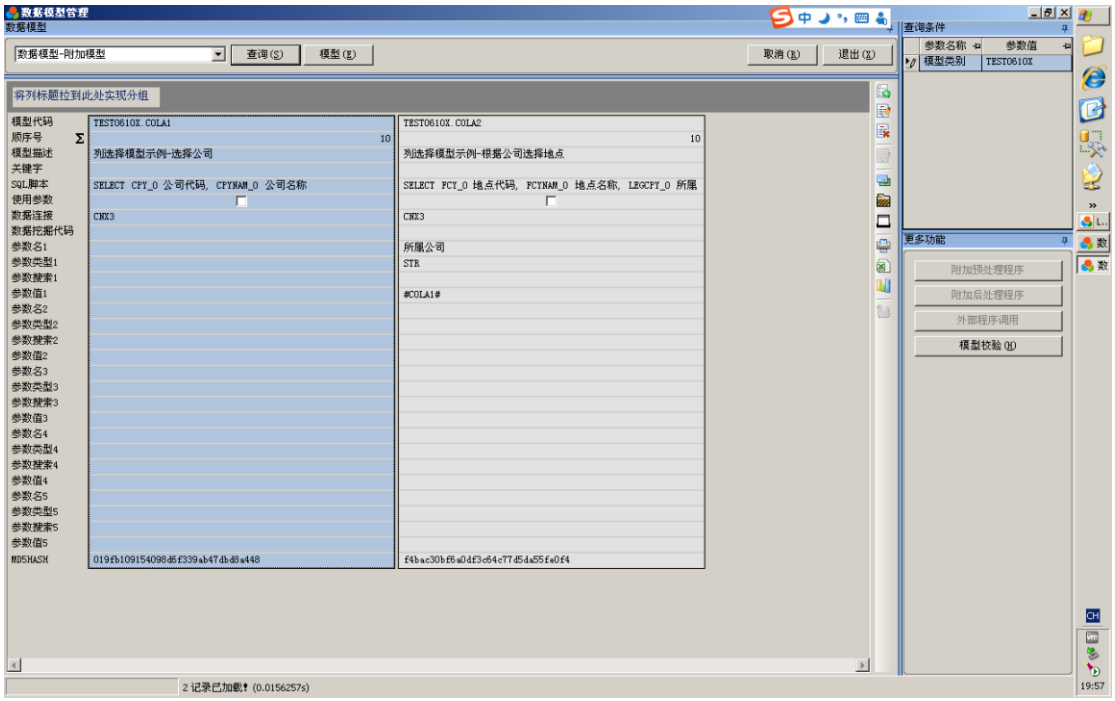
- 通常情况下，使用“上下左右”箭头可以实现在表格中的各个单元格数据之间的对应方向的移动。
- 在编辑模式下，如果进入某一个单元格中，移动被控制在单元格内部。
- 如果按住 SHIFT 键，使用“上下左右”可以在单元格之间移动。

34 为可编辑列定义选择模型



- 如上图所示，为该模型的两列分别定义“开窗选择”的模型
- COLA1——从公司列表中选择公司代码
- COLA2——从地点列表中选择地点代码，同时要求选择时只显示所属公司为 COLA1 的地点

35 编辑列选择模型实现



- 根据上一节的要求，为该模型定义列选择的附加模型
- 附加模型的代码格式为
  - “模型代码”+”模型编号”+”X.列标题”
- 附加列选择模型 1——TEST0610X.COLA1

```
SELECT CPY_0     公司代码
      ,CPYNAM_0  公司名称
FROM   COMPANY
ORDER  BY CPY_0
```

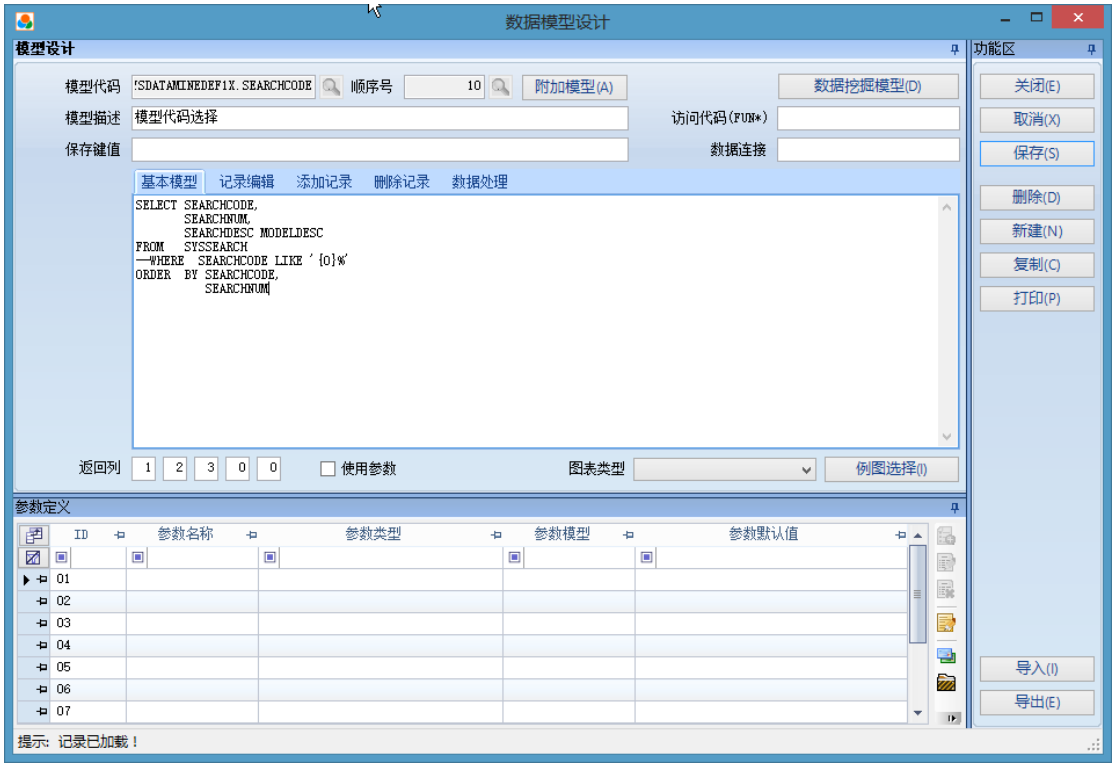
- 附加列选择模型 2——TEST0610X.COLA2

```
SELECT FCY_0     地点代码
      ,FCYNAM_0  地点名称
      ,LEGCPY_0  所属公司
FROM   FACILITY
WHERE  LEGCPY_0 = '{0}'
```

  - 该模型定义了一个参数，用于确定地点列表对应某个所属公司
- 为了能够使得选择地点时能够对应 COLA1 中选择的公司，只需要为模型的参数设置为#COLA1#即可

36 列选择模型返回多列值

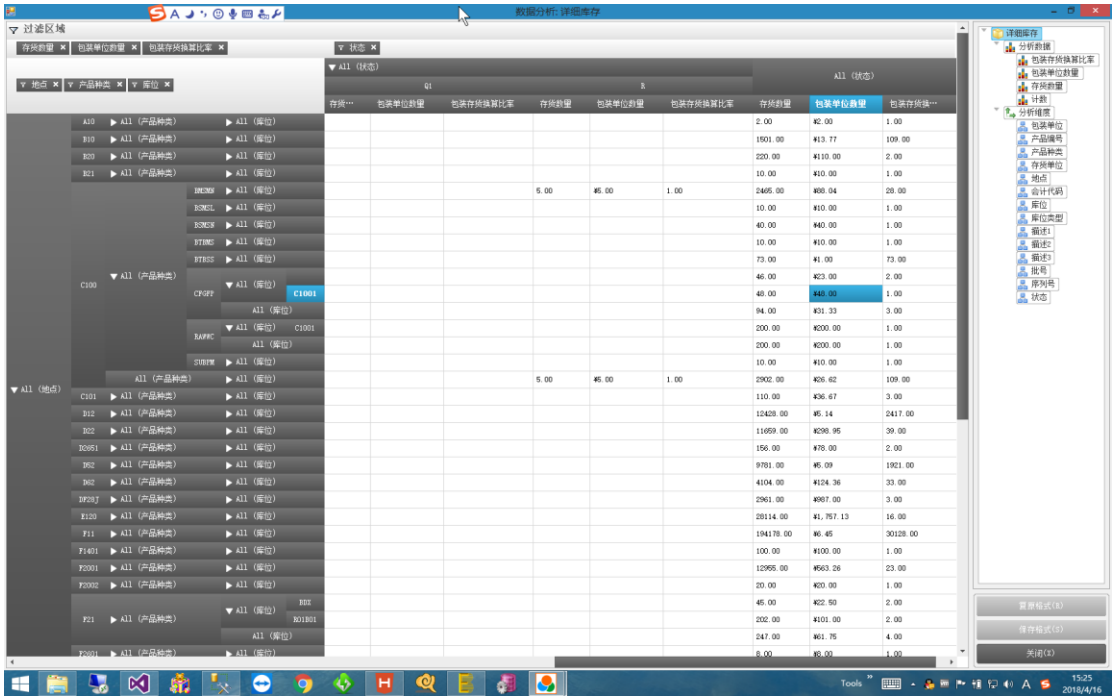
- 如上图所示，通过 SEARCHCODE 列的选择模型可以选择“需要进行数据挖掘”的“模型代码”。
- 在附加的“选择模型”中，列的标题为：
  - SEARCHCODE
  - SEARCHNUM
  - MODELDESC
- 这三列与数据表格中的列是对应的。
- 那么，按照下面的模型设计，即可在选择某一行数据的时候自动填写三列的值。



- 如图的模型设计中，包含了三列数据：
  - SEARCHCODE
  - SEARCHNUM
  - MODELDESC
- 并且，返回列选项中标明，需要返回 1/2/3 三列的内容。
- 这样，返回的数据行中的数据就会自动填写表格。
  - 第 1 个返回值填充当前列
  - 第 2-5 个返回值，按照列的标题填写对应的列

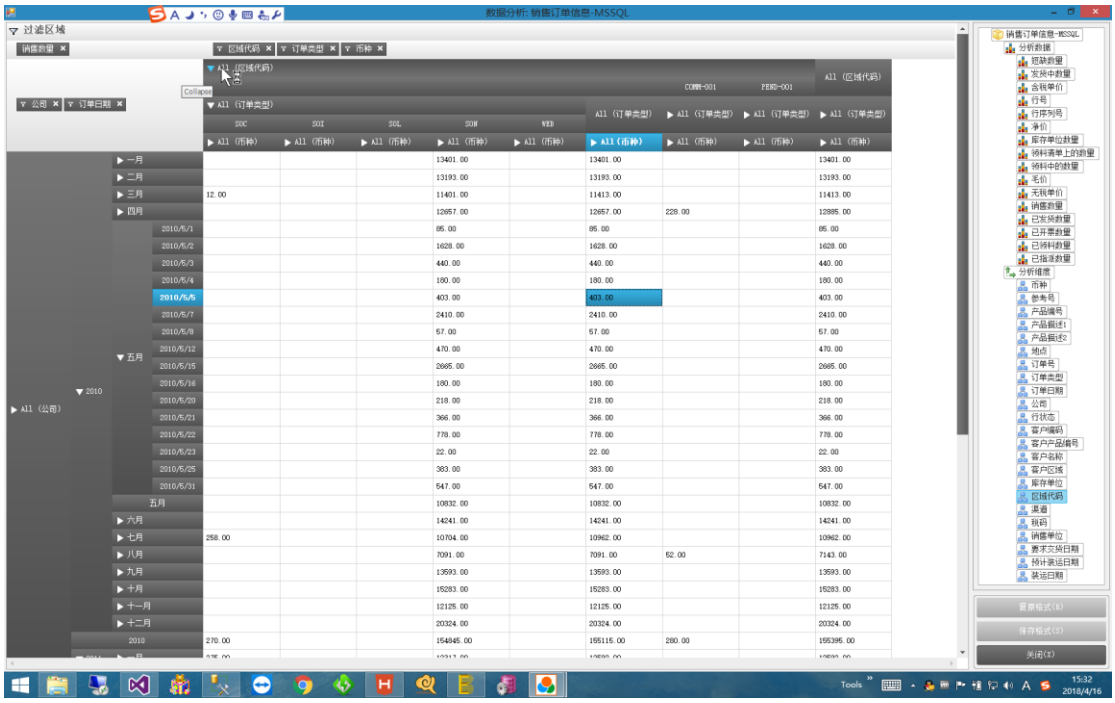
37 数据透视分析

- 如图所示，模型获取的数据，可以执行更加复杂的数据分析。

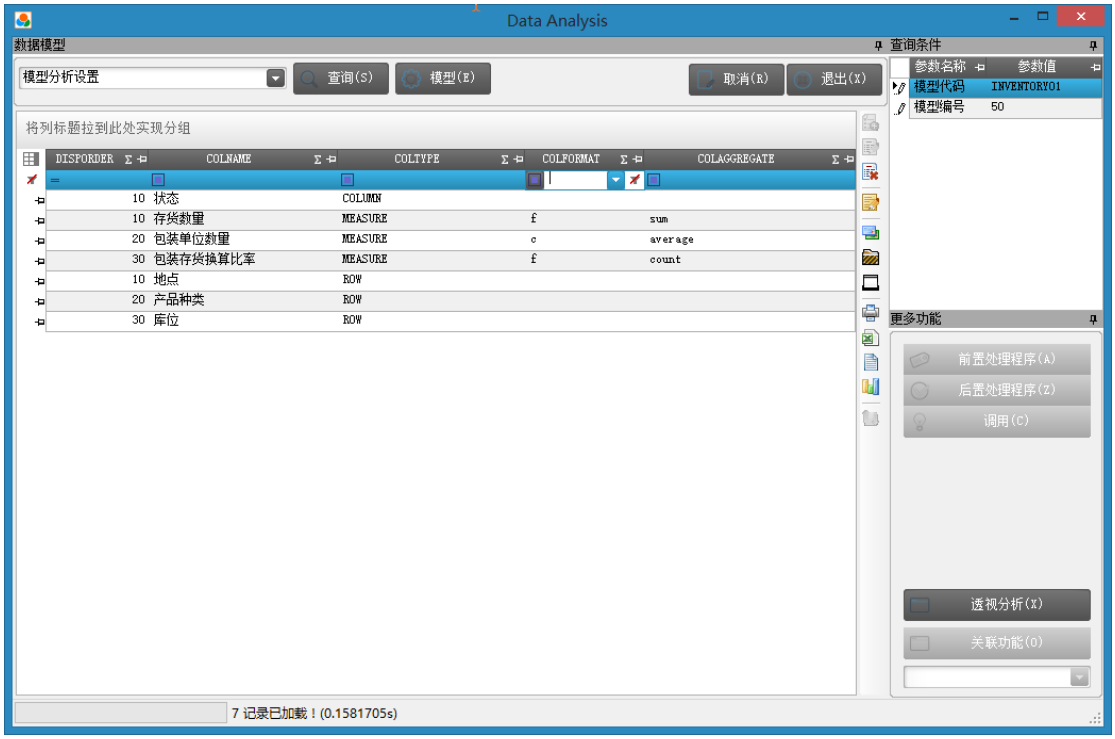


- 在进行数据透视分析的时候，用户可以方便选择行、列的维度，并对数据进行分析
- 默认情况下，对于模型的数据，按照下列逻辑区分维度与分析数据
  - 字符型、日期型信息可以作为分析维度
  - 数字型信息可以作为分析数据（默认进行汇总分析）

- 对于日期型信息，系统会自动按照“年、月、日”三级展开

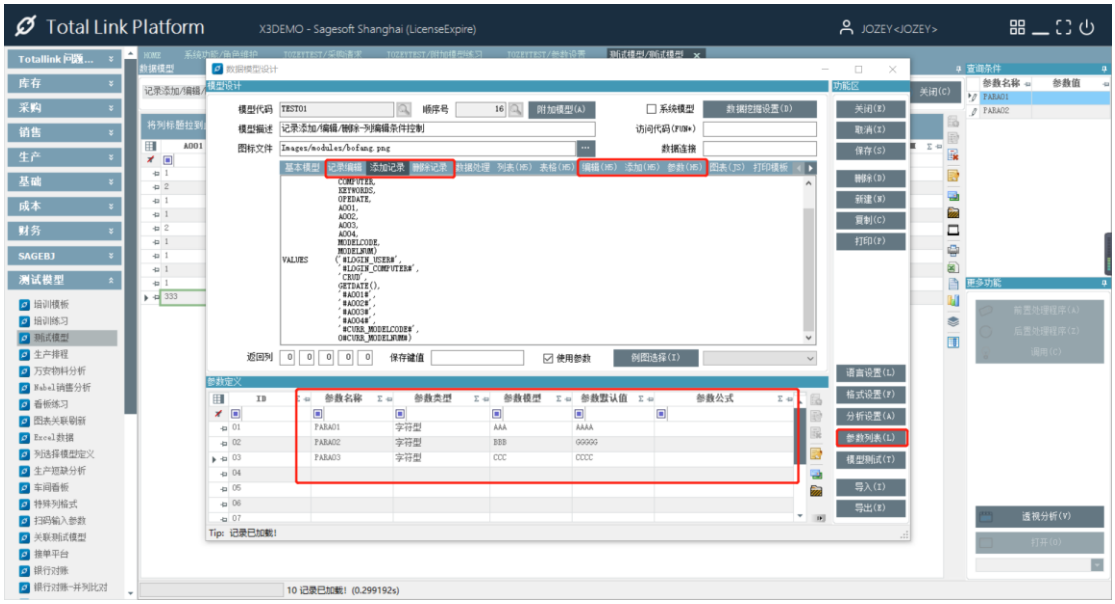


- 默认情况下，系统自动根据模型中的数据，创建分析维度和分析数据列表
- 用户可以根据需要自行拖拽信息到相应的区域中完成数据分析
- 如果需要设置默认的行列分析维度，可参照如下功能，在模型的分析设置中，指定用户需要的行、列、分析数据及分析方式

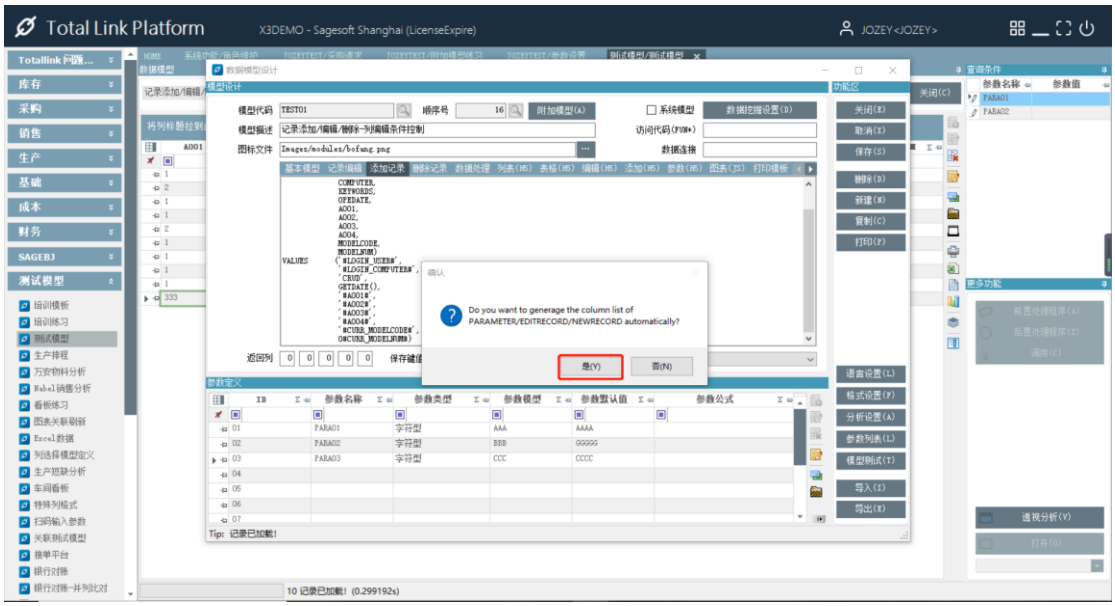


38 自动产生参数列表

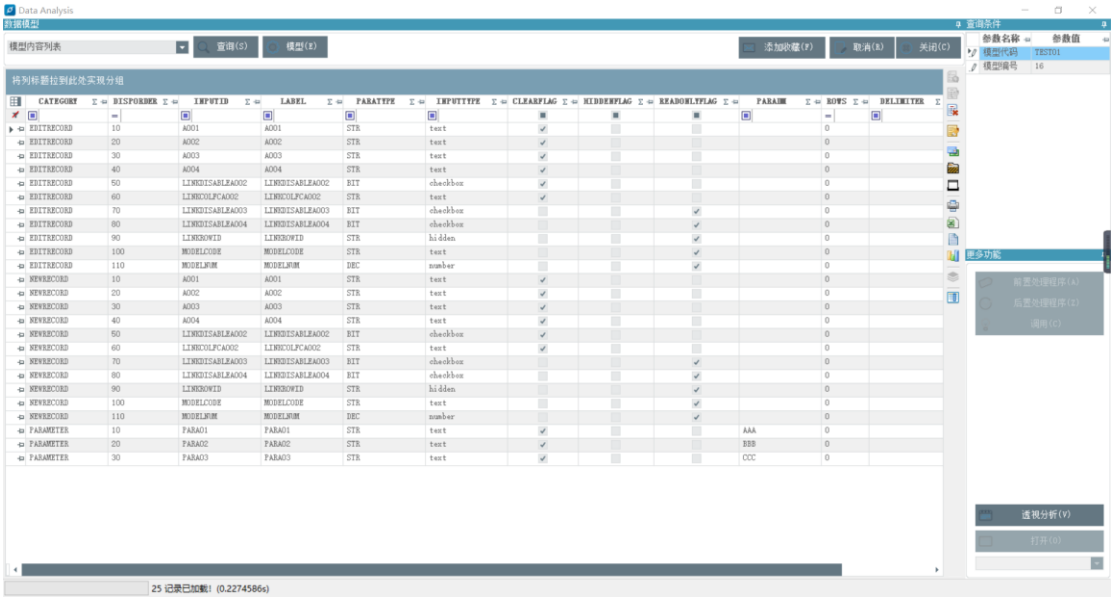
- TotalLINK 可以根据添加编辑模型以及参数定义列表里的代码自动生成参数列表



- 点击“参数列表”，选“是”即可自动产生参数列表



- 生成的参数列表结果如下


Data Analysis
数据模型
查询条件
模型内容列表
查询(S)
模型(E)
添加收藏(F)
取消(K)
关闭(C)
参数名称
参数值
模型代码
TEST01
模型编号
16
将列标题拉到此处实现分组
CATEGORY
DISPORDER
INPUTID
LABEL
PARATYPE
INPUTTYPE
CLEARFLAG
HIDDENFLAG
READONLYFLAG
PARAM
ROWS
DELIMITER
EDITRECORD
NEWRECORD
PARAMETER
更多功能
前置处理程序(A)
后置处理程序(Z)
调用(C)
透视分析(V)
打开(O)
25 记录已加载! (0.2274586s)

39 Function List

当项目中出现一个功能中需要调用两个 CallFunction 时，我们可以使用 FunctionList 的方式来实现。例如，点击一个按钮，将报表文件进行邮件发送，有两步动作，导出报表文件，进行邮件发送，具体操作如下：