TotalLINK

产品手册

上海朝识智能科技有限公司

2020 年 8 月

专家功能

目　录

文档控制

■ 主要内容

本文介绍专家基本功能点，实施人员可根据需要学习。

■ 更改记录

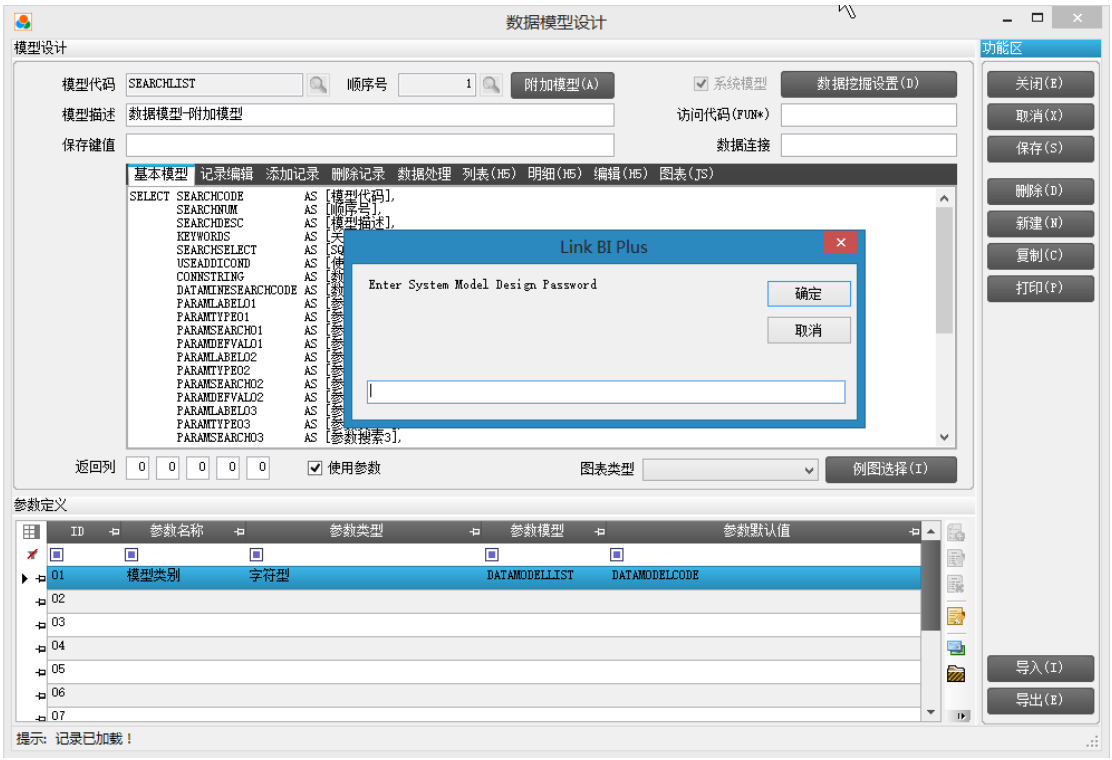
日期	版本	作者	备注
2017-05	1.0	Randy	初始发布
2020-08-31	2.0	Jozey	整理
2020-09-18	2.1	Jozey	ROW_COLOR颜色设置

■ 支持版本

非特殊说明的功能，默认前后版本都支持

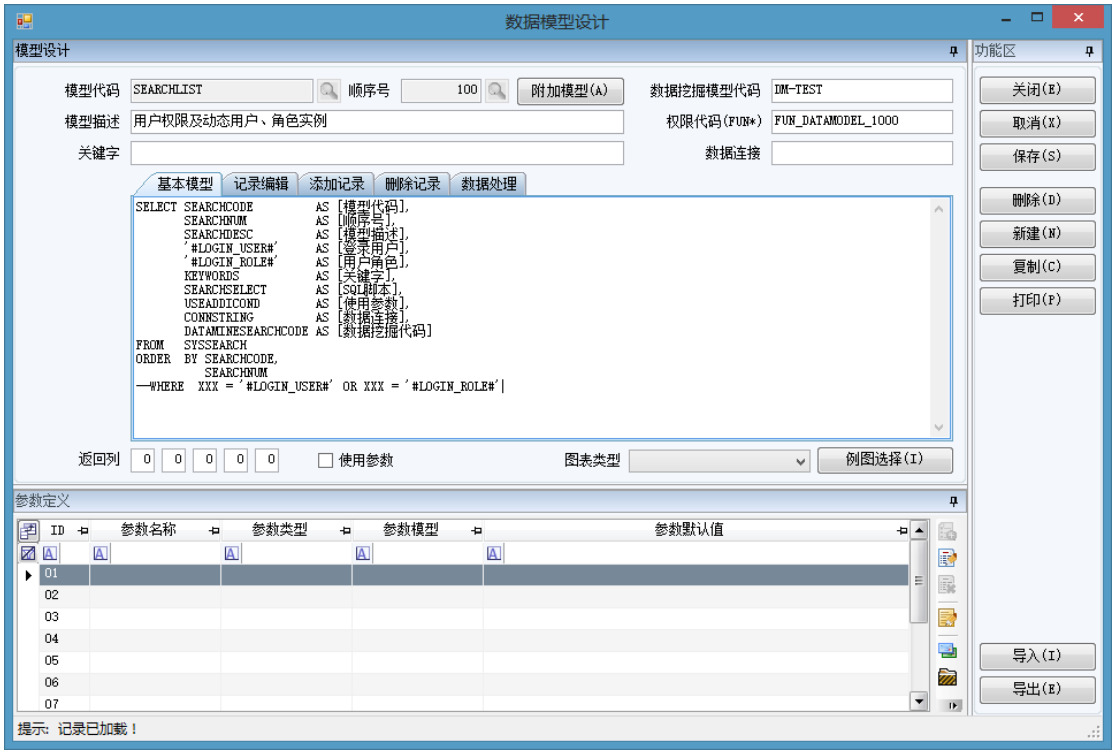
仅支持T20版本及以后版本的功能点
▪ ROW_COLOR 颜色设置

1 系统模型的修改



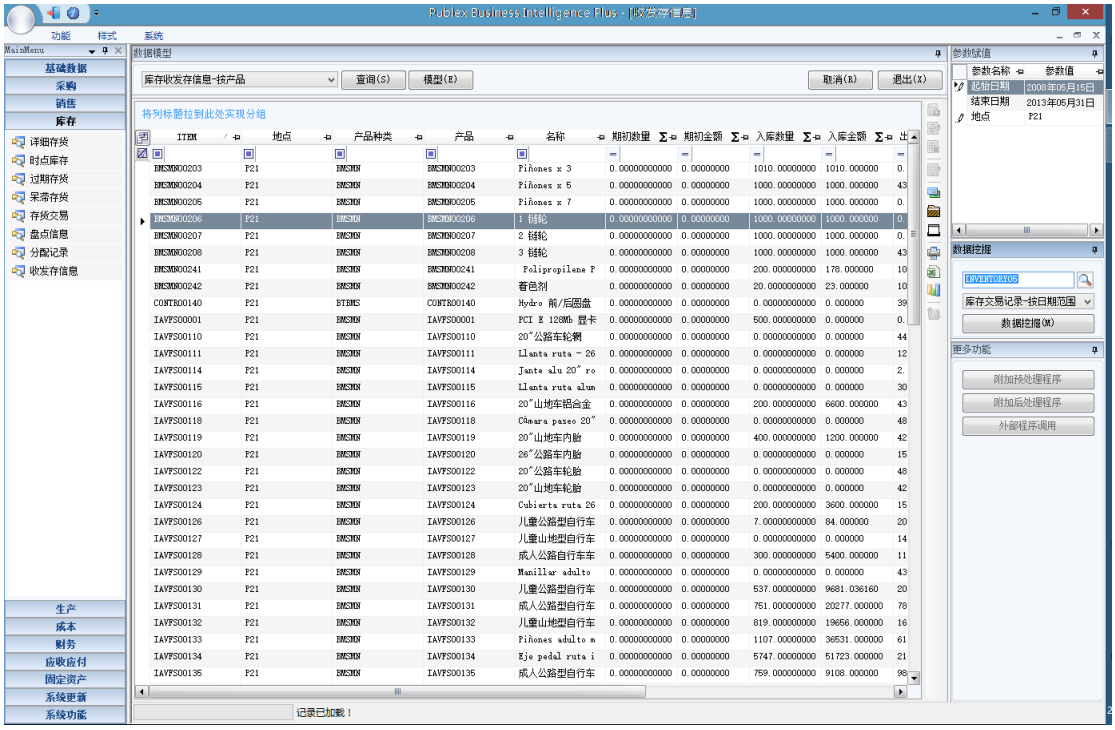
- 系统中有些模型是为 Total Link 系统本身的功能而设计的，这些模型内容的修改需要谨慎
- 系统中，对这些模型有“系统模型”标记
- 当需要“修改、删除或者导入覆盖”这些模型时，需要输入“口令”，“口令”不正确的时候不允许修改

2 模型导入与导出



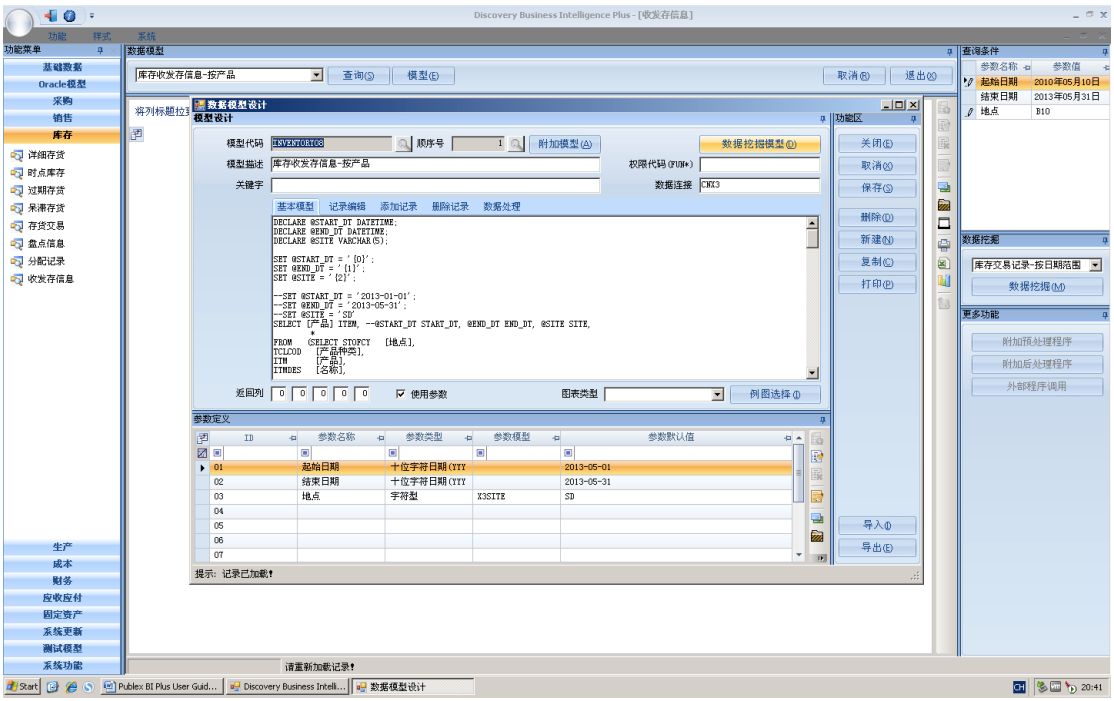
- 系统模型可以导出为 XML 格式，也可以将 XML 文件格式保存的数据模型导入系统
- 这样大大方便了模型数据的交换功能

3 数据挖掘



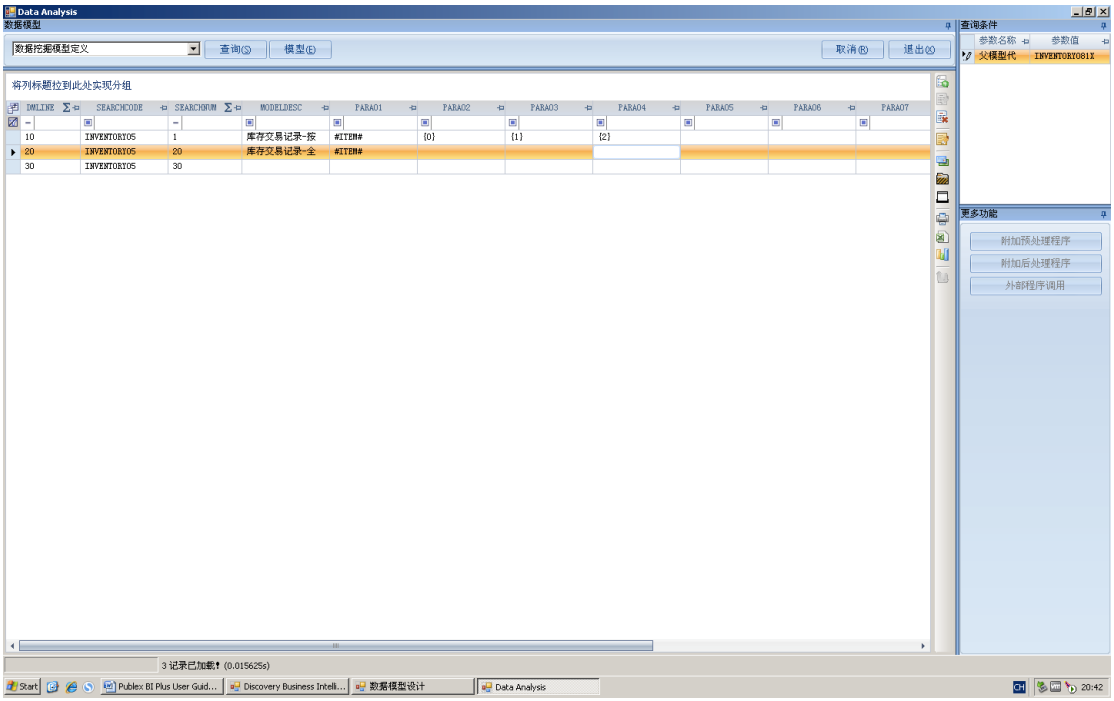
- 系统支持复杂的数据挖掘功能
- 当需要定义数据挖掘时，可以使用模型的数据挖掘功能定义当前模型的数据挖掘子模型
- 当模型具有数据挖掘功能时，用户可以选择相应的功能挖掘明细数据（见上图）
- 数据挖掘模型支持单一参数或多参数传递，因此可以定义复杂的数据挖掘功能

4 数据挖掘模型的定义



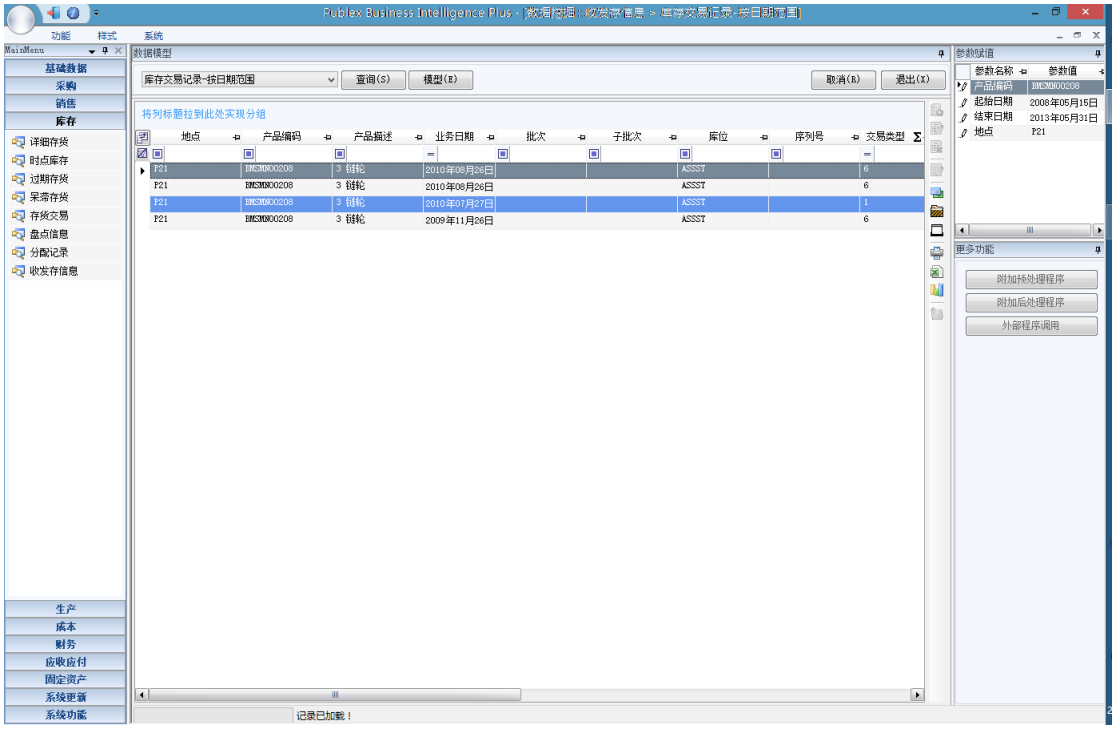
- 如上图所示，通过“数据挖掘模型”定义功能可以为当前的模型定义数据挖掘功能；

5 数据挖掘模型的参数传递



- 定义数据挖掘模型时，只要指定挖掘模型的代码和参数即可
- 数据挖掘模型的参数可以是多张来源：
  - 使用当前查询出的表格中的数据，通过“#数据列标题#”的方式引用
  - 使用主模型的参数，作为子模型的参数，通过“{0}、{1}、{2}……”的方式引用
  - 使用当前登录的用户或角色，通过全局变量引用，包括#LOGIN_USER#/#LOGIN_ROLE#等
  - 具体全局变量的使用可参见后文相关内容

6 多参数模型数据挖掘设计举例



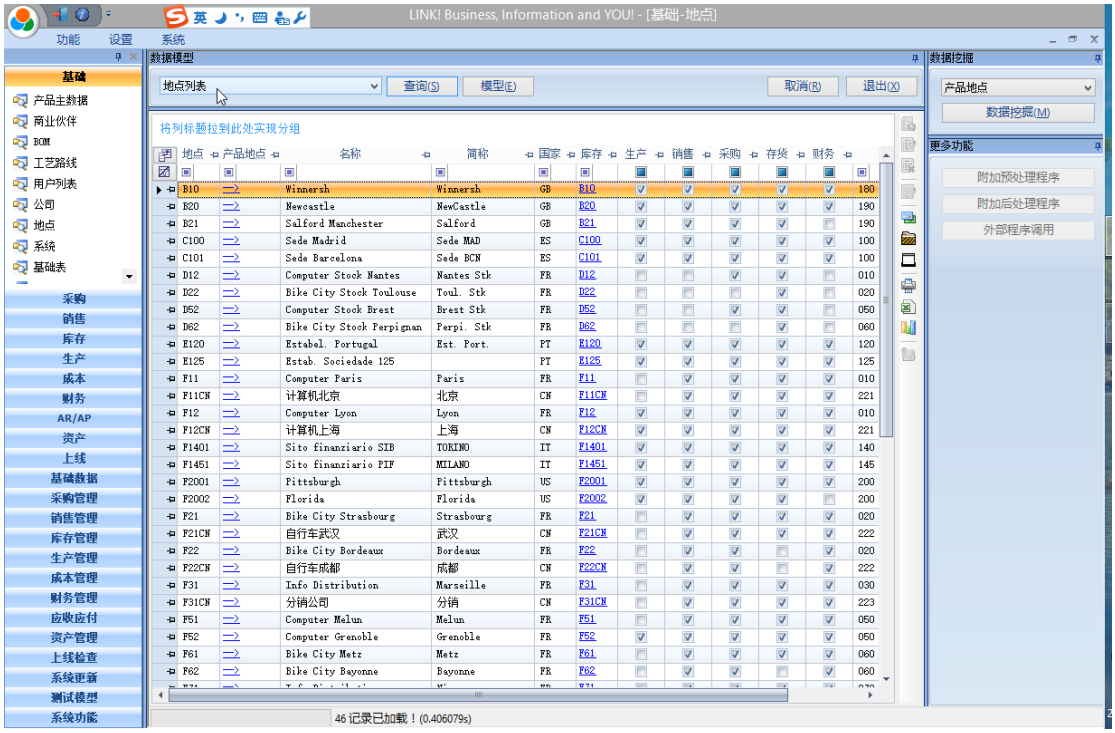
- 当使用数据挖掘功能时，系统支持可以向“数据挖掘子模型”传递多个参数
- 本例中，子模型的脚本如下：

```
DECLARE @ITEM VARCHAR(30);
DECLARE @START_DT DATETIME;
DECLARE @END_DT DATETIME;

SET @ITEM ='{0}';
--SET @ITEM = '8736801249';
SET @START_DT ='{1}';
SET @END_DT ='{2}';

SELECT STOFCY_0 [地点],
       J.ITMREF_0 [产品编码],
       I.ITMDES1_0 [产品描述],
       IPTDAT_0 [业务日期],
       LOT_0 [批次],
       SLO_0 [子批次],
       LOC_0 [库位],
       SERNUM_0 [序列号],
       TRSTYP_0 [交易类型],
       L704.LANMES_0 [交易说明],
       VCRTYP_0 [单据类型],
       L701.LANMES_0 [单据说明],
       VCRNUM_0 [单据号],
       VCRLIN_0 [单据行],
       QTYSTU_0 [存货数量],
       J.STU_0 [存货单位],
```

```
       QTYPCU_0 [包装数量],
       J.PCU_0 [包装单位]
FROM   STOJOU J
       INNER JOIN ITMMASTER I
              ON I.ITMREF_0 = J.ITMREF_0
       LEFT OUTER JOIN APLSTD L704
                    ON L704.LANCHP_0 = '704'
                       AND L704.LAN_0 = 'CHI'
                       AND L704.LANNUM_0 = J.TRSTYP_0
       LEFT OUTER JOIN APLSTD L701
                    ON L701.LANCHP_0 = '701'
                       AND L701.LAN_0 = 'CHI'
                       AND L701.LANNUM_0 = J.VCRTYP_0
WHERE  J.ITMREF_0 = @ITEM
       AND J.QTYSTU_0 <> 0
       AND J.IPTDAT_0 BETWEEN @START_DT AND @END_DT
       AND J.STOFCY_0 = '{3}'
```

- 在这个模型中，共用到 4 个参数，分别为：
  - 产品编码
  - 起始日期
  - 结束日期
  - 地点
- 四个参数分别对应
  - 主模型的列 ITEM
  - 参数 1（起始日期）
  - 参数 2（结束日期）
  - 参数 3（地点）。
- 因此数据挖掘定义中，参数设定为
  - #ITEM#
  - {0}
  - {1}
  - {2}
- 通过上述设置，在执行数据挖掘时，系统就会自动将相关参数传递给子模型，可以顺利的根据条件获取数据挖掘的分析数据

7 模型中的数据挖掘列



- 如上图所示，当为模型定义了数据挖掘之后，还可以在数据表行中定义数据挖掘列

- 上例中
  - 产品地点一列以“→”标记，点击该信息可以数据挖掘到对应地点下的产品地点信息
  - 库存列也是一个数据挖掘列，点击可以查看该地点下的库存数据

- 上述数据挖掘列是以“LINKSUBxx_XXXX”标识的

- 模型的脚本如下：

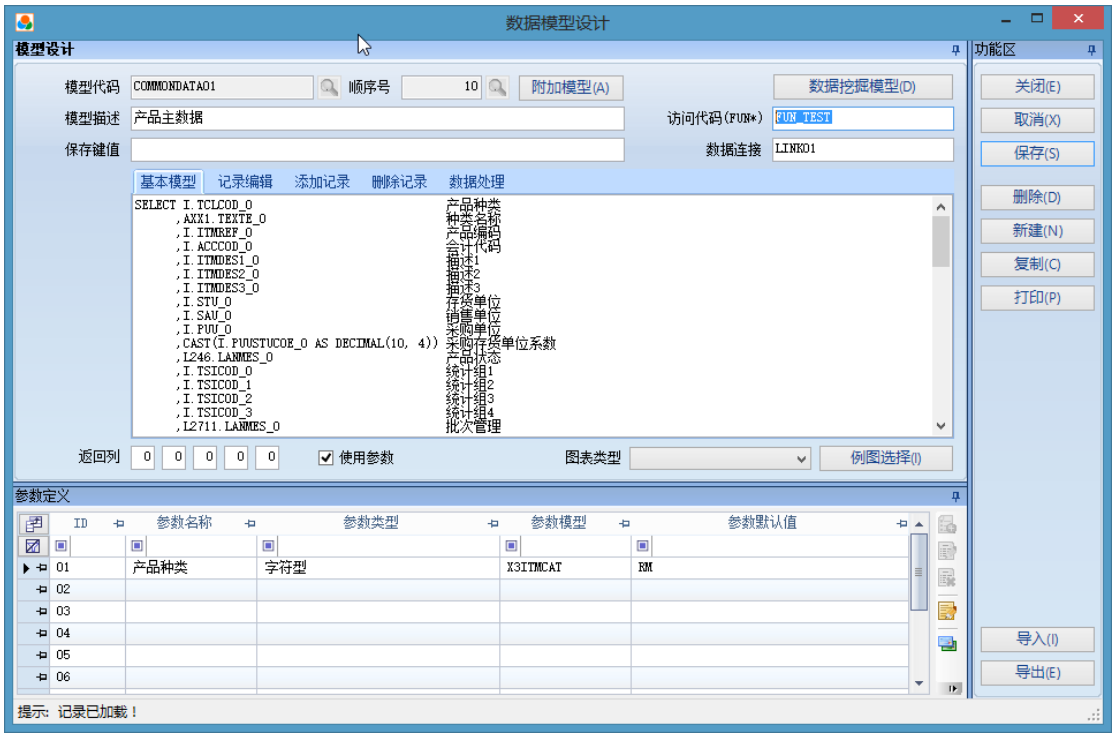
```
SELECT FCY_0 [地点],
       '-->' LINKSUB10_产品地点,
       FCYNAM_0 [名称],
       FCYSHO_0 [简称],
       CRY_0 [国家],
       FCY_0 LINKSUB20_库存,
       CAST (( CASE MFGFLG_0
                  WHEN 2 THEN 1
                  ELSE 0
                END ) AS BIT) AS [生产],
       CAST (( CASE SALFLG_0
                  WHEN 2 THEN 1
                  ELSE 0
                END ) AS BIT) AS [销售],
       CAST (( CASE PURFLG_0
```

```
                WHEN 2 THEN 1
                ELSE 0
              END ) AS BIT) AS [采购],
       CAST (( CASE WRHFLG_0
                WHEN 2 THEN 1
                ELSE 0
              END ) AS BIT) AS [存货],
       CAST (( CASE FINFLG_0
                WHEN 2 THEN 1
                ELSE 0
              END ) AS BIT) AS [财务],
       LEGCPY_0 [公司]
FROM   FACILITY
ORDER  BY FCY_0;
```

8 练习-设计数据挖掘模型

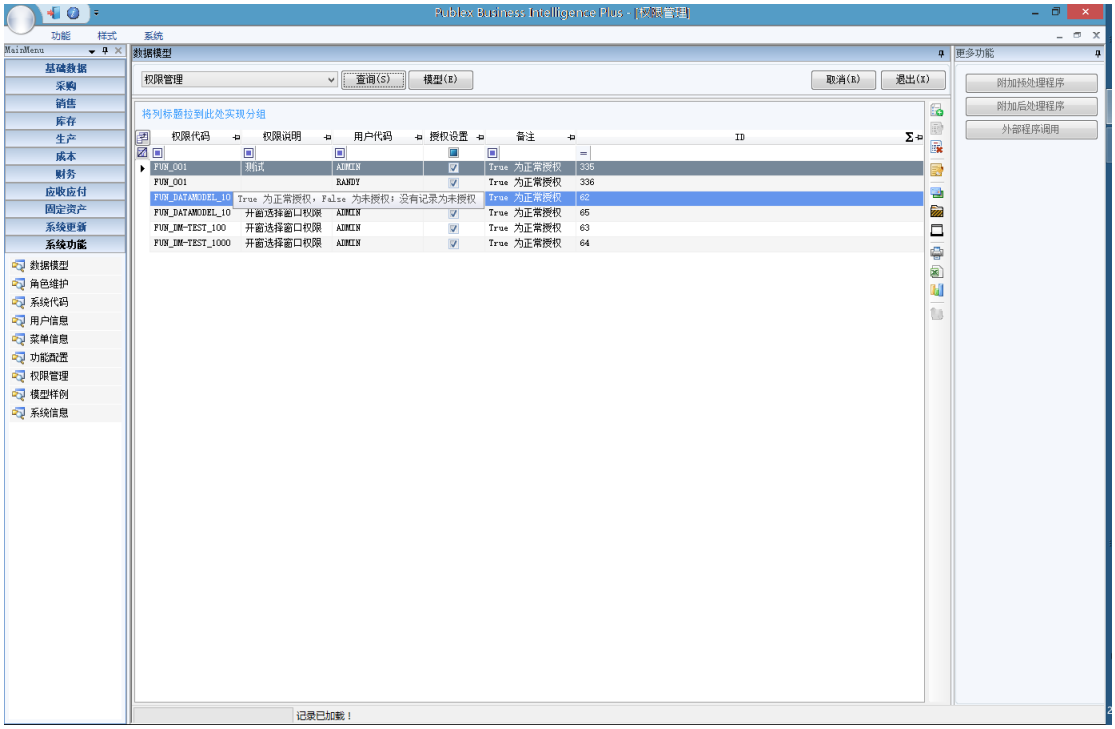
- 请参照上述说明，请设计一个多参数的数据挖掘模型
- 该数据挖掘模型会传递
  - 产品编码、开始日期、结束日期、地点等四个参数
- 基于主数据模型的数据，进行数据挖掘操作练习

9 权限代码控制数据访问



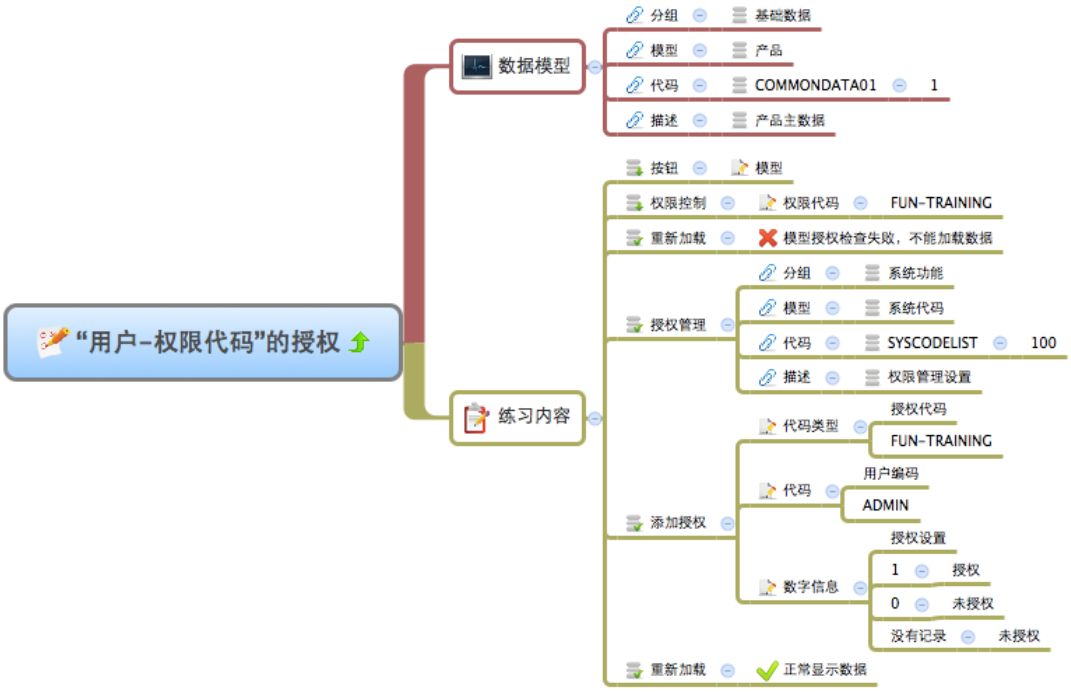
- 如图所示，为了限制用户对模型的访问，可以使用访问代码
- 访问代码的格式要求以“FUN_xxx”的格式设置

10 设置访问代码允许用户使用



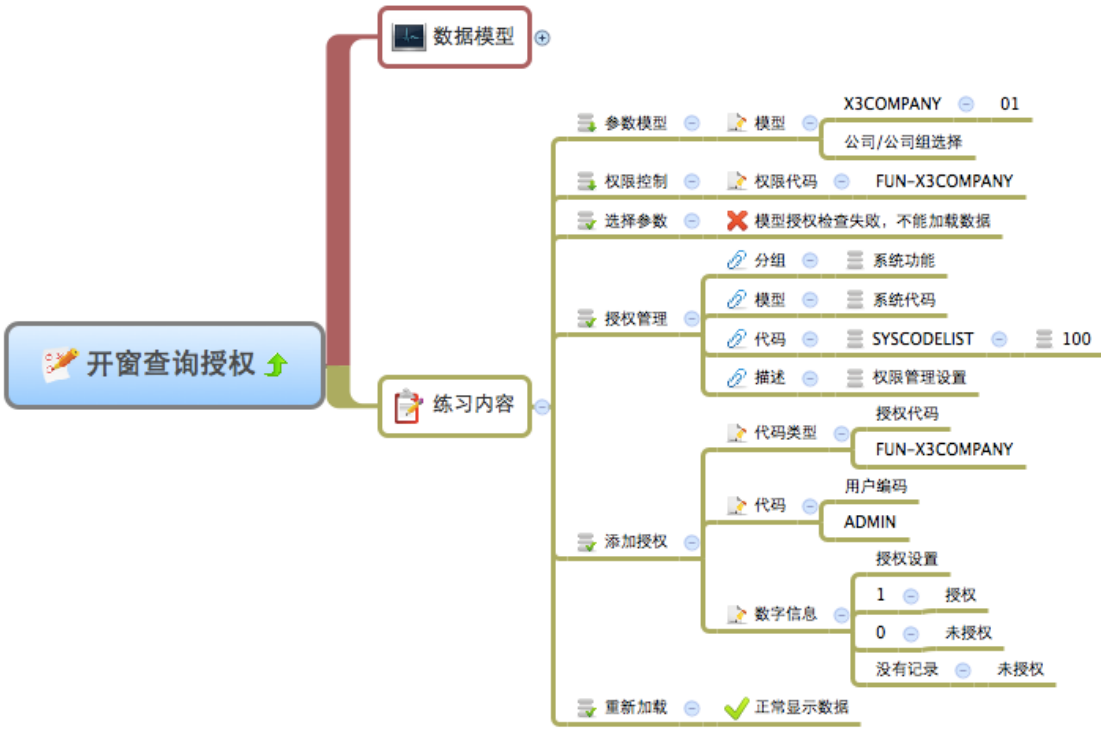
- 为了允许用户访问模型，需要将访问代码赋值给用户
- 仅允许访问的用户可以在列表中看到模型，其他用户不能看到此模型
- 没有设置访问代码的模型，允许所有用户访问

11 练习-通过权限代码控制模型的访问



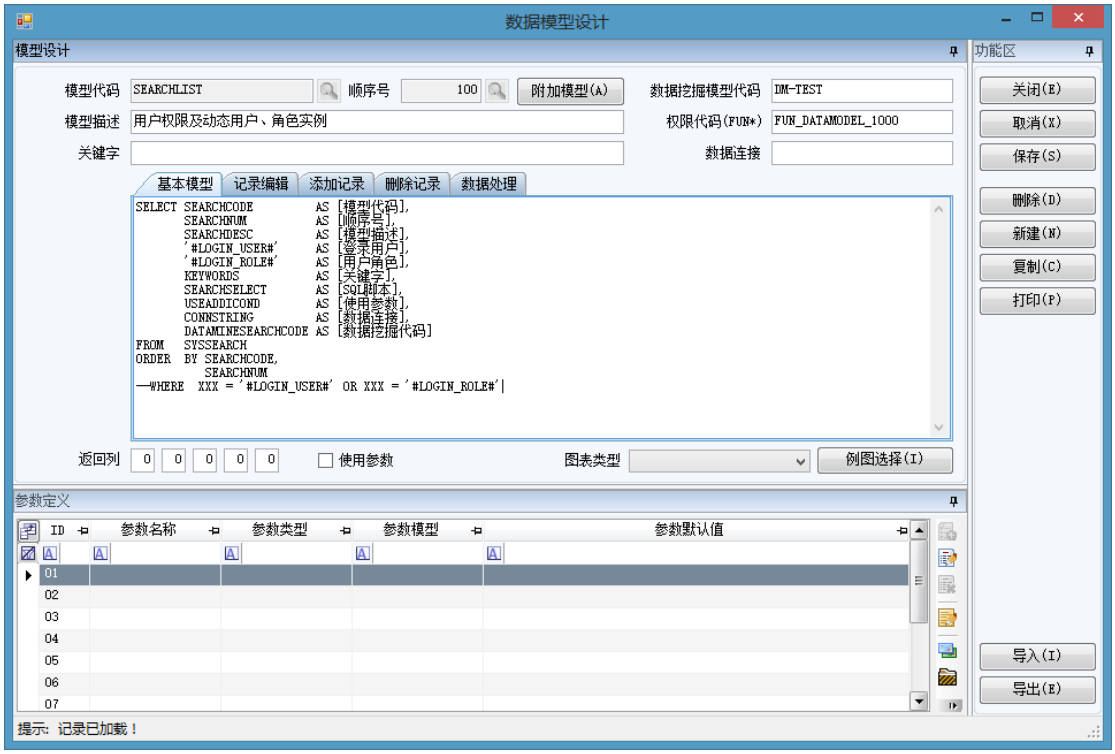
- 参照上述图示，为模型设置权限代码，并测试不同不同用户是否可以正常访问数据

12 开窗选择的授权



- 当需要对开窗选择的结果进行授权时，可以采用对模型授权同样的模式
- 请参照上述图示完成开窗选择的授权

13 动态角色及用户控制



- 在模型设计时，可以使用动态的用户角色及用户代码（参考下文：使用全局变量）

- 这样不同用户执行查询时，相当于使用了动态的参数，可以方便设计灵活的权限控制查询分析功能

14 使用全局变量

- 灵活运用特殊的全局变量可以使得模型的编制更加简单
- 常用：
- #LOGIN_USER#——当前登录的用户代码
- #LOGIN_ROLE#——当前登录的用户对应的角色代码
- #LOGIN_ID#——当前登录用户对应的 ID
- #LOGIN_COMPUTER#——当前登录的电脑名称
- #LINKSERVER#——当前访问的服务器基本地址
- #LINKCUSCODE#——服务器指定的客户代码
- #CURR_MODELCODE#——当前执行的模型的模型代码
- #CURR_MODELNUM#——当前执行的模型的模型编号
- #ATTACH_MODELCODE#——当前执行的模型的附加模型代码
- #LINKLOCATION#——当前地理位置信息（字符串）
- #LINKLOCATIONJSON#——当前地理位置信息（JSON 格式数据）
- #LINKID_RESULT#——为统一的 GUID 号码，每次执行附加模型的动作，相当于 NEWID()
- #LINKID_ACTION#——为统一的 GUID 号码，每次执行附加模型的动作，相当于 NEWID()
- #LINKREPORTPATH#——ActiveReport 报表路径，默认为“C:\TotalLink\Reports”；使用方式举例：“reportFile":"#LINKREPORTPATH#\\daily.rdlx",具体使用见文档 LINK27_报表设计应用.pdf
- #LINKCONN.LINK01#--替代数据源连接语句

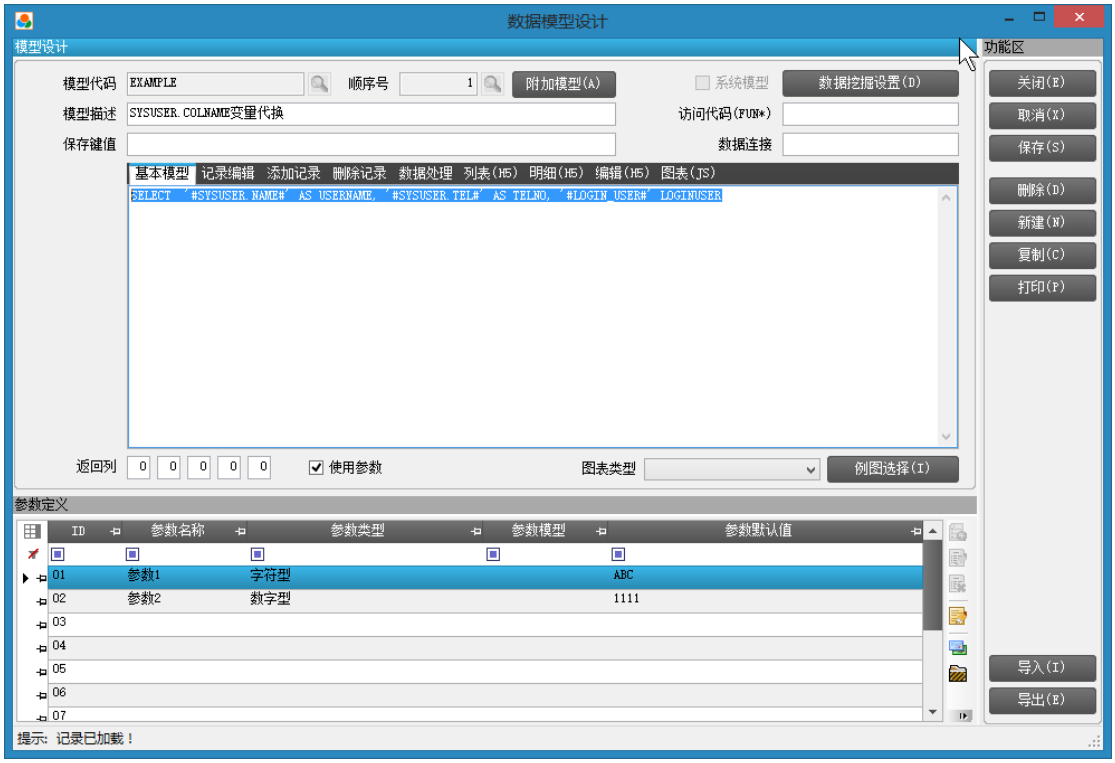
- 与时间有关：
- #LINK_TODAY#——当前日期（yyyy-mm-dd）
- #LINK_YESTERDAY#——昨天（yyyy-mm-dd）
- #LINK_TOMORROW#——明天（yyyy-mm-dd）
- #LINK_THISYEAR#——今年（yyyy）
- #LINK_THISMONTH#——当月（mm）
- #LINK_THISYEARMONTH#——当前年月（yyyy-mm）
- #LINK_THISYEARMONTH6#——当前年月（yyyymm）
- #LINK_THISYEARMONTH#-01——今年当前月份的第一天（yyyy-mm-01）
- #LINK_FIRSTDAY_THISMONTH#——当前月份的第一天（yyyy-mm-01）
- #LINK_LASTDAY_THISMONTH#——当前月份的最后一天（yyyy-mm-xx）
- #LINK_LASTYEAR#——去年（yyyy）
- #LINK_LASTMONTH#——上个月（mm）

- #LINK_LASTYEARMONTH#——当前月份减去一年（yyyy-mm）
- #LINK_LASTYEARMONTH6#——当前月份减去一年（yyyymm）
- #LINK_FIRSTDAY_LASTMONTH#——上个月的第一天（yyyy-mm-01）
- #LINK_LASTDAY_ LASTMONTH#——上个月最后一天（yyyy-mm-xx）
- #LINK_TIMETICKS#——时间戳
- #LINK_TIMEHM#——当前时分（hh:mm）
- #LINK_TIMEHMS#——当前时分秒（hh:mm:ss）
- #LINK_FULLTIME#——当前时间（yyyy-MM-dd hh:mm:ss.ffff）

- 除了上述列举的固定的全局变量以外，系统允许使用 SYSUSER 表中的任意列的值作为模型脚本中可以替代的变量。

- 为了适应更加复杂的从多个不同的业务系统中交叉获取数据的情况，系统允许通过特定的系统模型交换数据，作为可以替代的变量自动处理模型脚本。

15 使用 SYSUSER.COLNAME 实现变量代换



- 如图所示，在脚本中允许使用类似#SYSUSER.NAME#的方式引用 SYSUSER 表中的各列的值，在执行脚本的时候，系统自动用 SYSUSER 这个表的对应列的值替代这些特殊的变量。

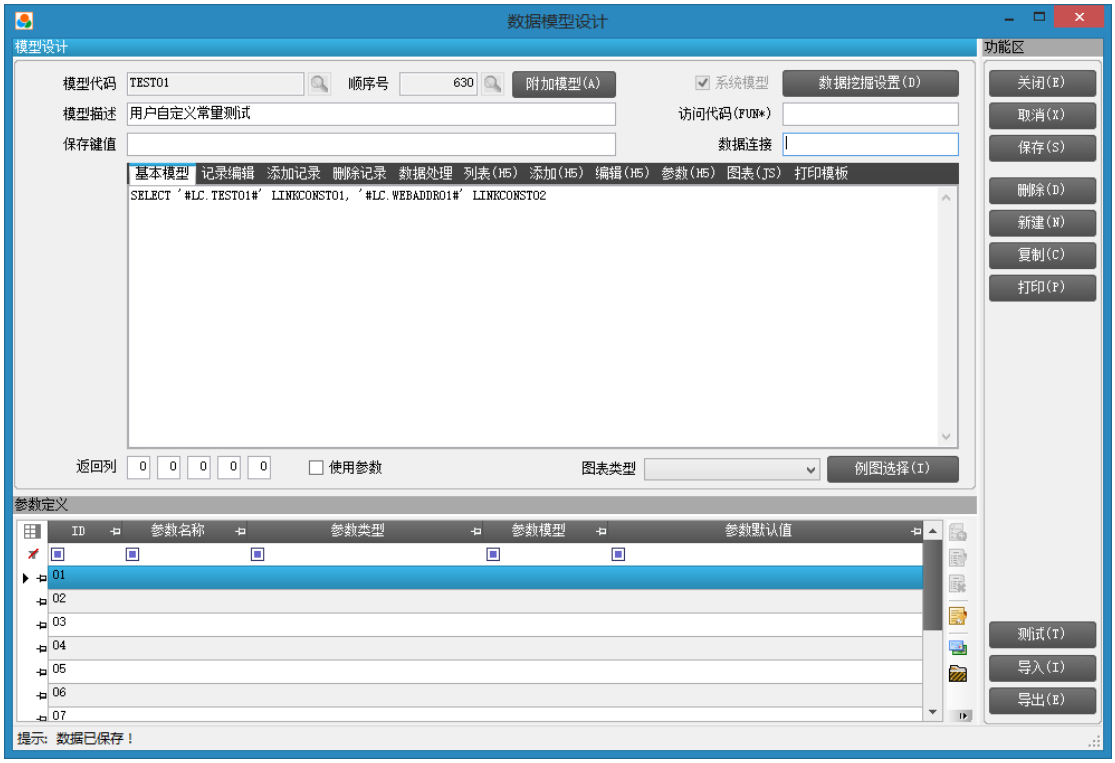
- 比如，对应如下脚本：

```
 SELECT  '#SYSUSER.NAME#' AS USERNAME,
         '#SYSUSER.TEL#' AS TELNO,
'#LOGIN_USER#' LOGINUSER
```

- 系统执行的结果如下图所示。

- 请注意，在使用这种类型的“全局变量”代换时，模型脚本中的字段名称需要使用大写字母表示。

16 使用用户自定义常量



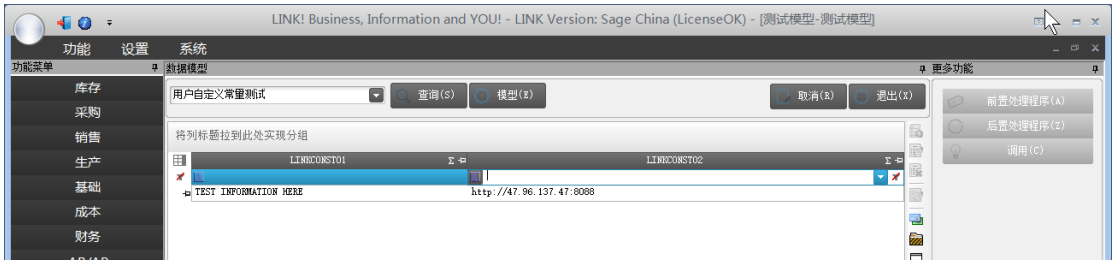
- 如图所示，在模型的脚本中，允许用户使用自定义常量的值替换脚本中的特殊标记

- 如上图所示，脚本如下

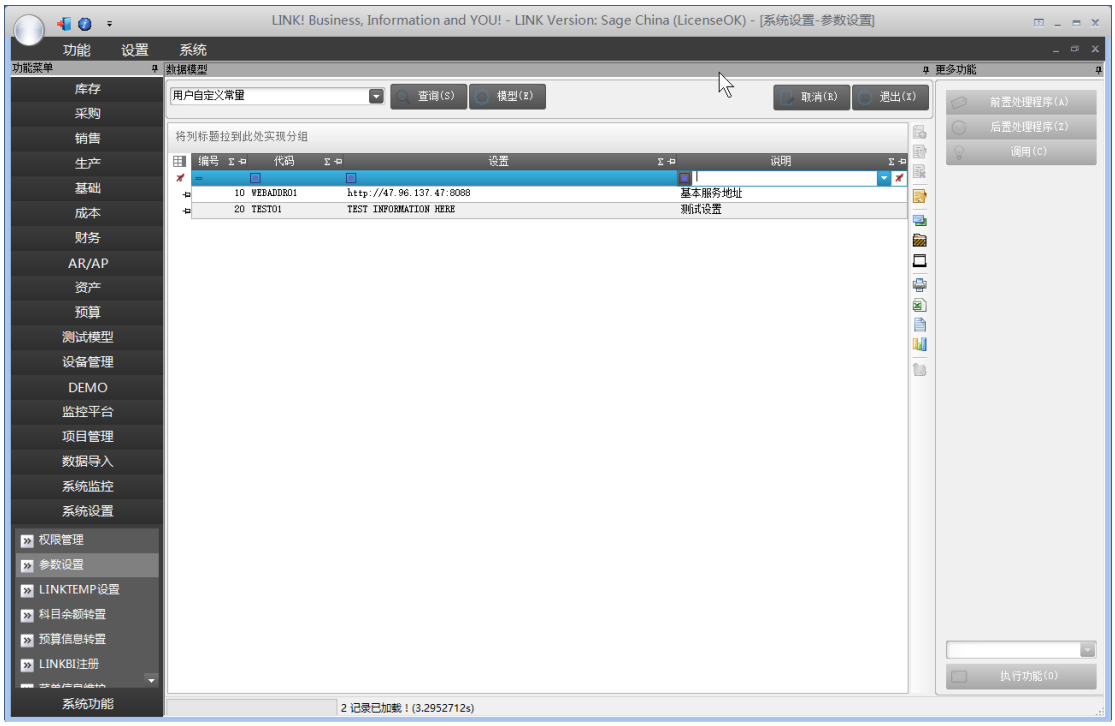
```
SELECT '#LC.TEST01#' LINKCONST01, '#LC.WEBADDR01#' LINKCONST02
```

- 在模型脚本中，引用用户自定义常量时，采用如下格式
  - 以 #LC. 开头，以 # 结尾
  - 中间引用用户自定义常量的名称

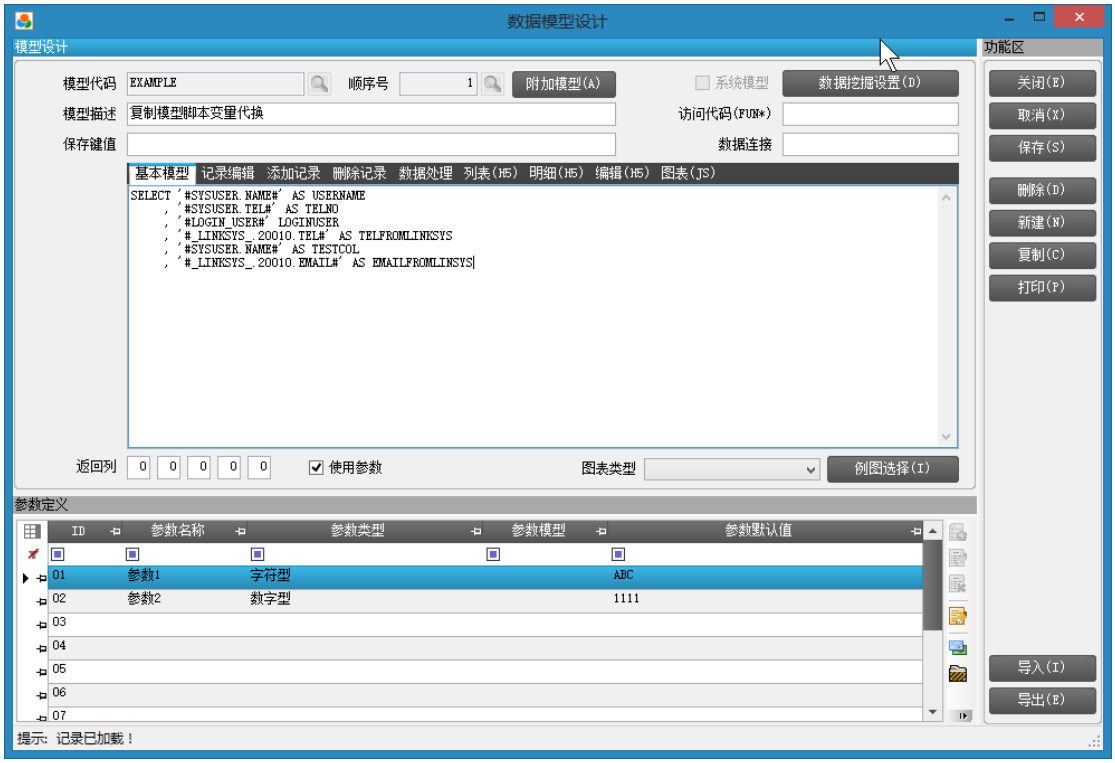
- 上述脚本中红色标记部分是用户自定义的常量
  - 系统在执行改脚本时，会自动根据设置替换为常量对应的值
  - 上述模型执行的结果如下图所示



- 自定义常量及值的设定通过如下功能实现



17 使用系统模型交换数据



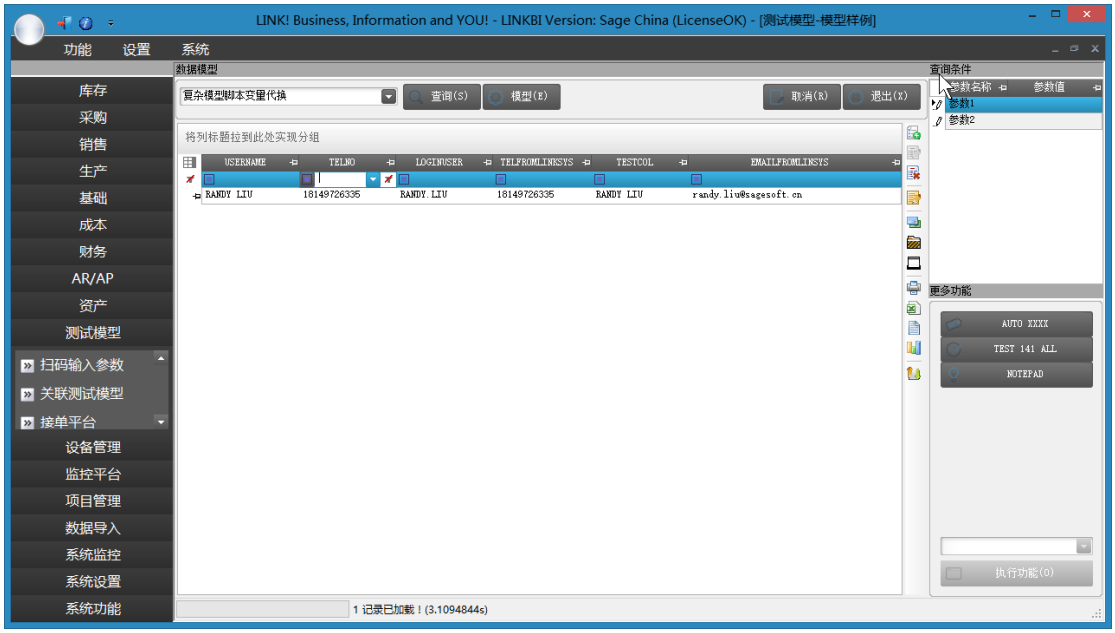
- 如图所示，我们定义模型的脚本如下：

```
SELECT '#SYSUSER.NAME#' AS USERNAME
     , '#SYSUSER.TEL#' AS TELNO
     , '#LOGIN_USER#' LOGINUSER
     , '#_LINKSYS_.20010.TEL#' AS TELFROMLINKSYS
     , '#SYSUSER.NAME#' AS TESTCOL
     , '#_LINKSYS_.20010.EMAIL#' AS EMAILFROMLINSYS
```

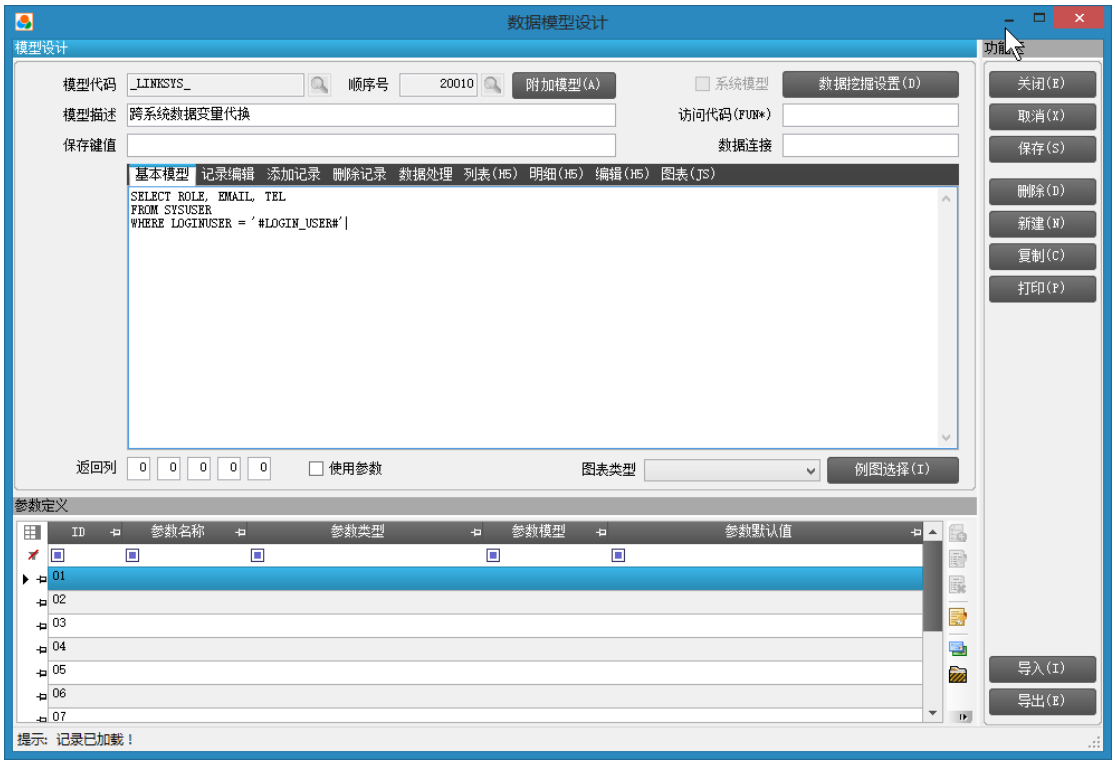
- 在该模型中，我们使用了一些特殊的内容
  - #_LINKSYS_.20010.TEL#
    - 表示来自模型“_LINKSYS_/20010”的 TEL 列的数据
  - #_LINKSYS_.20010.EMAIL#
    - 表示来自模型“_LINKSYS_/20010”的 EMAIL 列的数据

- 在模型执行过程中，系统会自动执行模型“_LINKSYS_/20010”，并获取第一行的数据，将对应列的“值”替换脚本的内容。

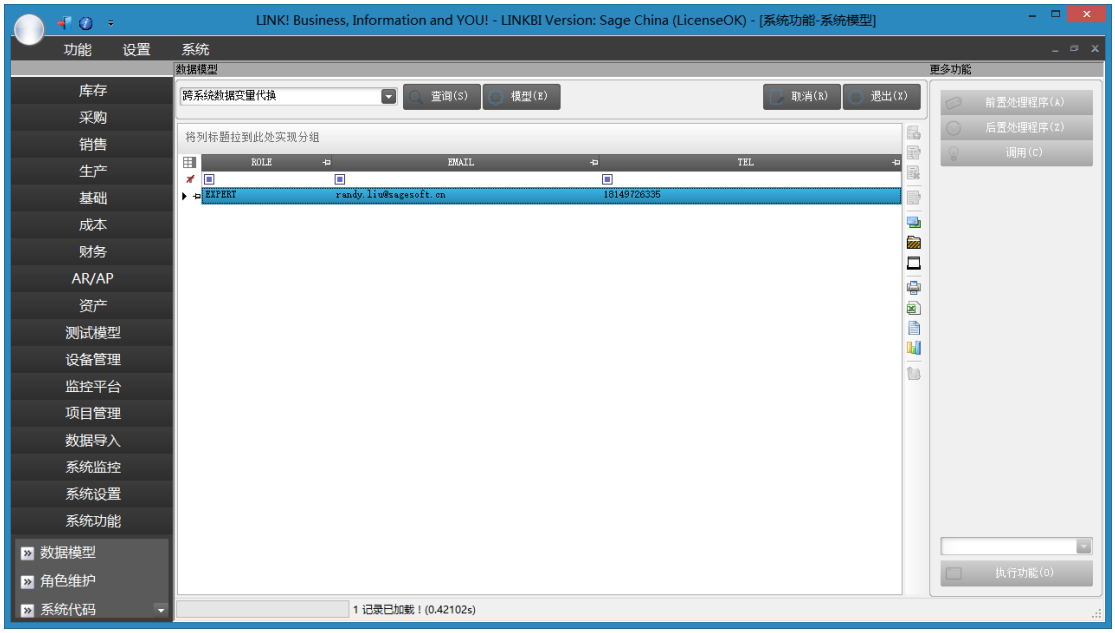
- 该模型的执行结果如下图所示。



- 可以看到，脚本中的内容被真实的数据替换了。
- 为了实现这样的结果，需要首先定义一个用于信息交换的模型，如下图所示。



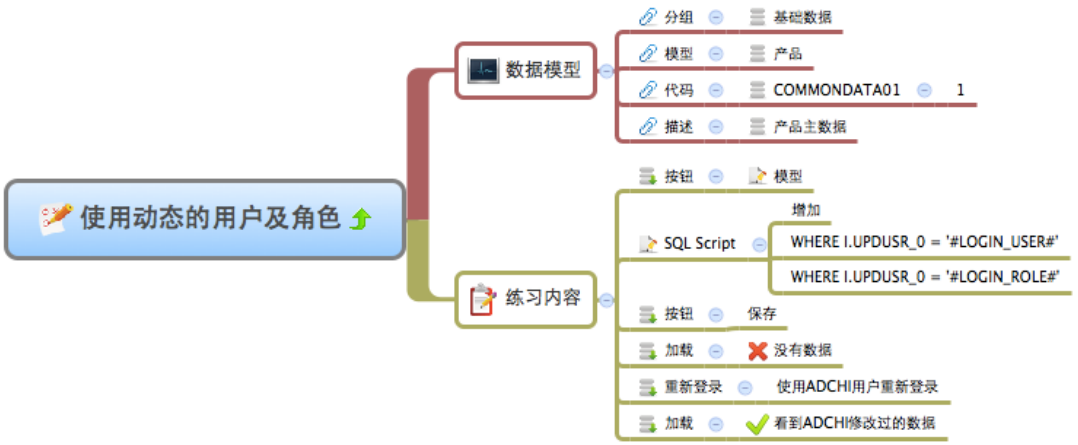
- 测试执行该模型，可以得到如下结果



- 该执行结果中的“内容”，即 TEL、EMAIL 两列的具体值，会自动替换掉“测试模型”中的内容。

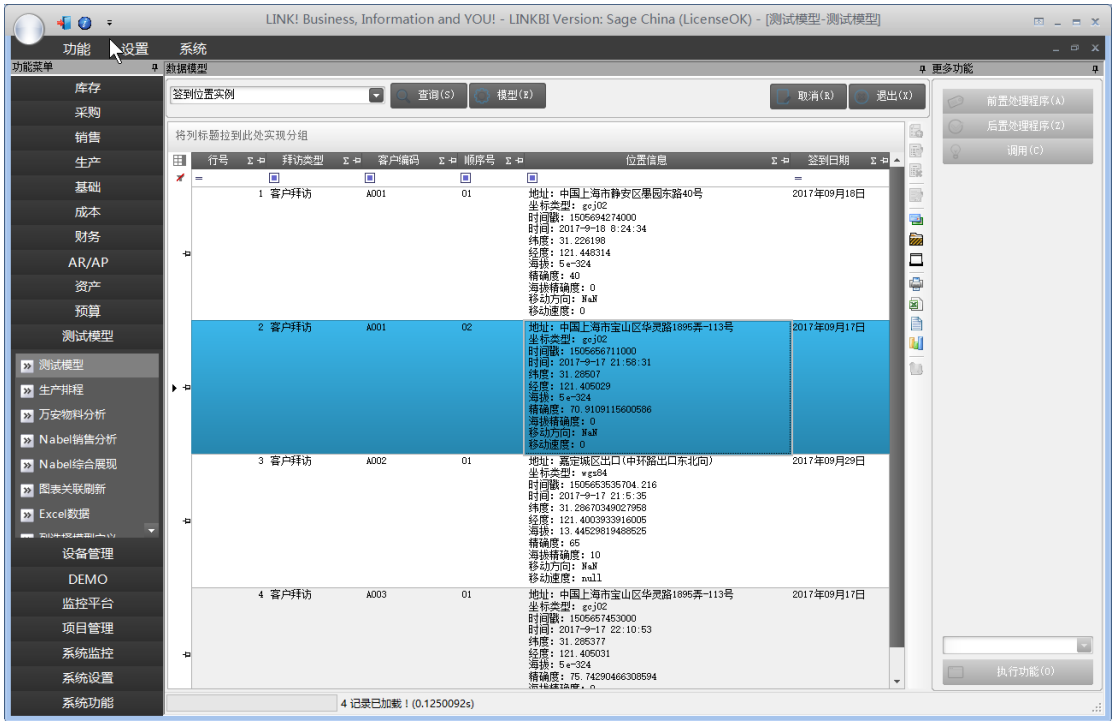
- 注意：
  - 通过这种方式进行变量代换，只能使用代码为“_LINKSYS_”的一系列特殊模型
  - 如果该模型查询得到多行数据，使用第一行数据进行替换
  - 如果该模型查询无法得到数据，系统提示
    - “DATAMODEL.NUM.COLNAME.NODATA”
  - 该模型不限制只是使用 LINK 系统数据库的数据，根据链接代码可以来自任何一个业务系统

18 练习-使用动态角色或用户代码

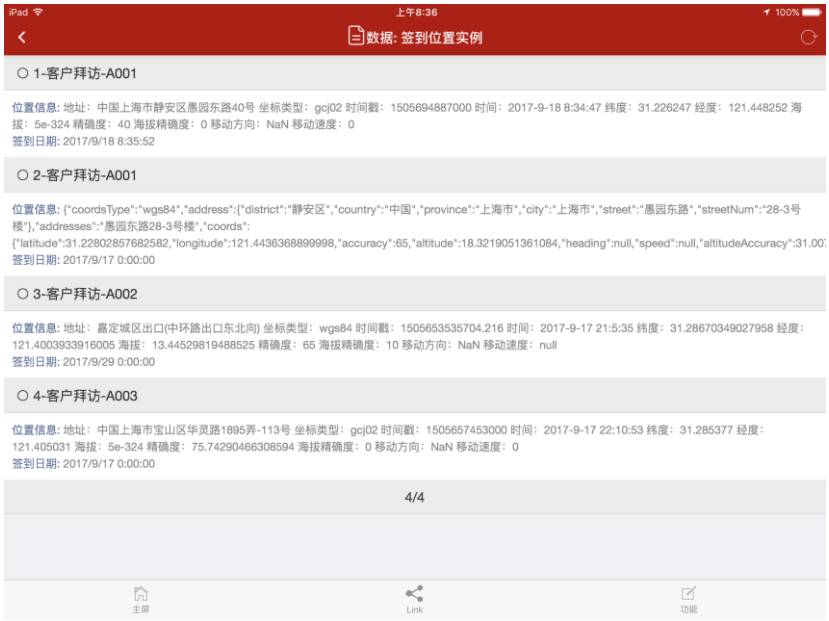


- 参照上述图示，设计一个包含动态用户（#LOGIN_USER#）或动态角色（#LOGIN_ROLE#）的模型，用户执行时会自动将登陆的用户名或角色替换模型中的脚本内容
- 从而控制用户可以访问的数据

19 使用移动设备的地理位置信息

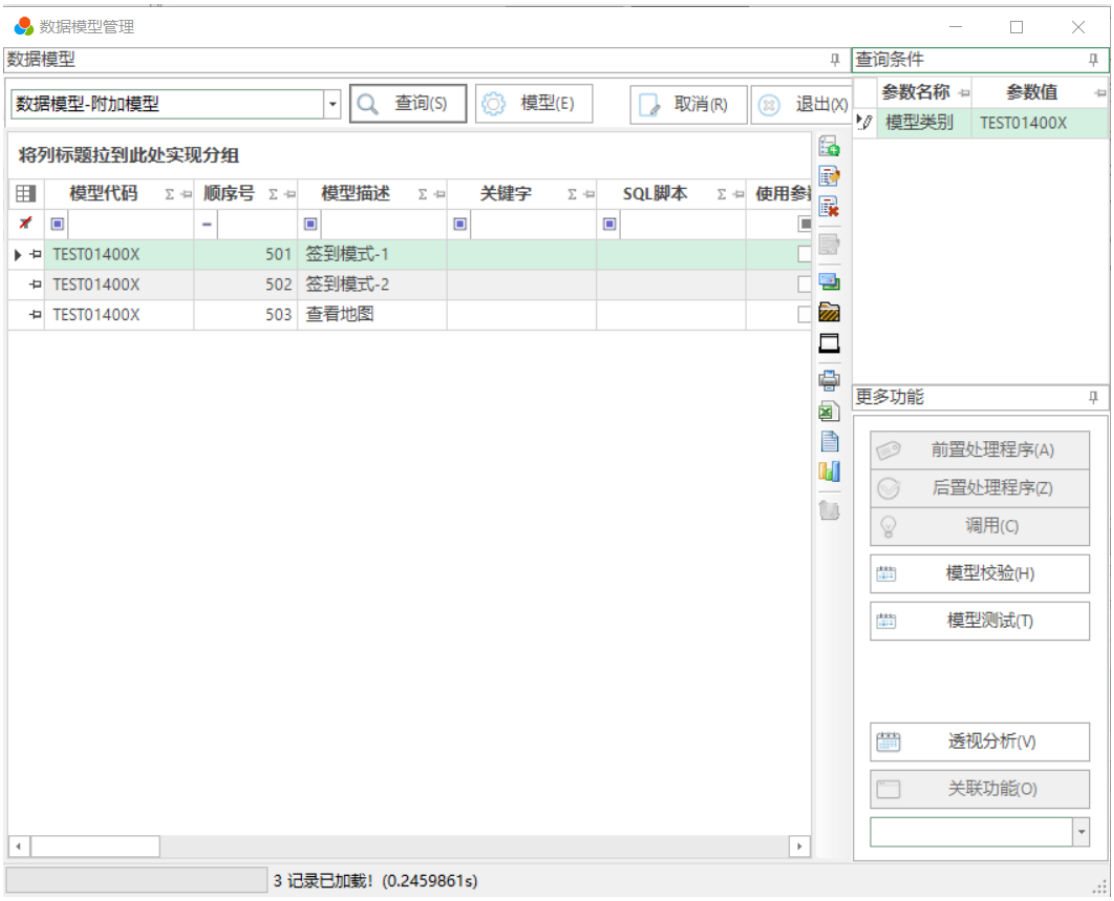


- 随着移动应用的普及，发生业务时所在的地理位置成为一种重要的信息数据
- 结合业务发生的时间，如果同时保留地理位置信息，业务就对应了完整的“时空”信息，这对某些业务是很重要的
- 在 total link 系统的模型数据处理时，可以通过在脚本中嵌入“#LINKLOCATION#或 #LINKLOCATIONJSON#”以自动获取设备所在的地理位置信息



- 如图所示，在信息中的“地理位置信息”是通过移动设备自动获取的

- 为此，我们在模型设计时，设计了三个附加处理模型，分别是：
  - 签到模式-1（获取详细的地理位置信息内容）
  - 签到模式-2（获取以 JSON 格式保存的地理位置信息数据）
  - 查看地图（定位地理位置在地图上）



- 签到模式-1 和签到模式-2 这两个模型的内容分别是：

```
        UPDATE LINKPOSITION SET LOCATION = '#LINKLOCATION#', CHECKDATE = GETDATE() WHERE ID = #行号#
UPDATE LINKPOSITION SET LOCATION = '#LINKLOCATIONJSON#' CHECKDATE = GETDATE() WHERE ID = #行号#
```

- 在执行相关模型处理的时候，系统可以根据模型中的“全局变量”分别进行位置信息处理
  - #LINKLOCATION#
    - 这里的全局变量，会被当前地理位置的详细描述替代，通常类似于下面的内容

      地址：中国上海市静安区愚园东路 40 号
      坐标类型：gcj02
      时间戳：1505694887000
      时间：2017-9-18 8:34:47
      纬度：31.226247
      经度：121.448252
      海拔：5e-324
      精确度：40
      海拔精确度：0
      移动方向：NaN

移动速度：0

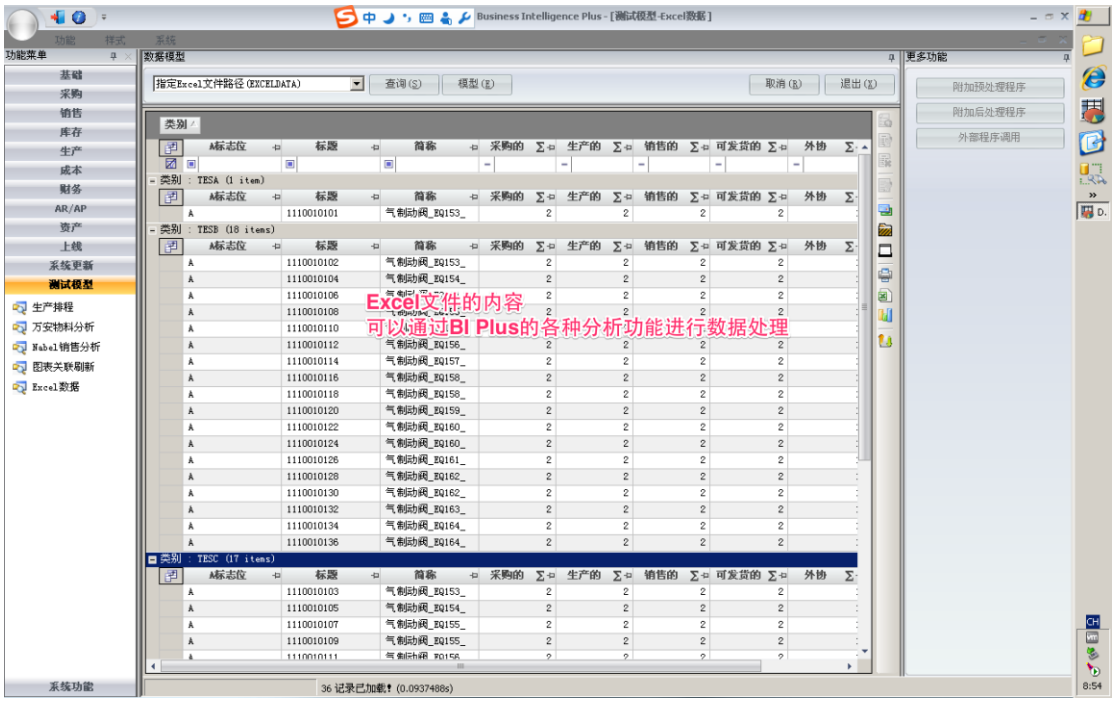
- #LINKLOCATIONJSON#
  - 这里的全局变量，会被描述当前地理位置的 JSON 格式的数据替代，通常类似于下面的内容

```
{
    "coordsType": "wgs84",
    "address": {
        "district": "静安区",
        "country": "中国",
        "province": "上海市",
        "city": "上海市",
        "street": "愚园东路",
        "streetNum": "28-3 号楼"
    },
    "addresses": "愚园东路 28-3 号楼",
    "coords": {
        "latitude": 31.22802857682582,
        "longitude": 121.4436368899998,
        "accuracy": 65,
        "altitude": 18.3219051361084,
        "heading": null,
        "speed": null,
        "altitudeAccuracy": 31.00787056715399
    },
    "timestamp": 1505694955665.752
}
```

- 在获取相关的“地理信息”之后，用户可以根据需要对其中的具体内容进行处理

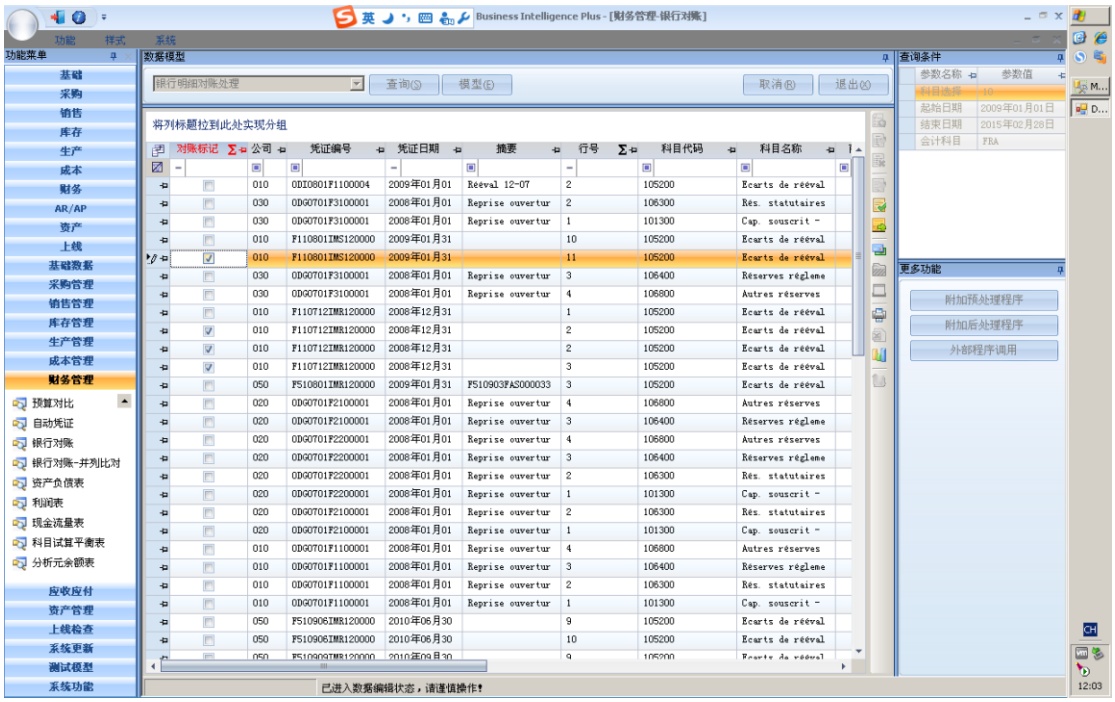
- 查看地图模型通过在数据连接处添加“LINKMAP”调用当前设备地理位置信息显示在地图上

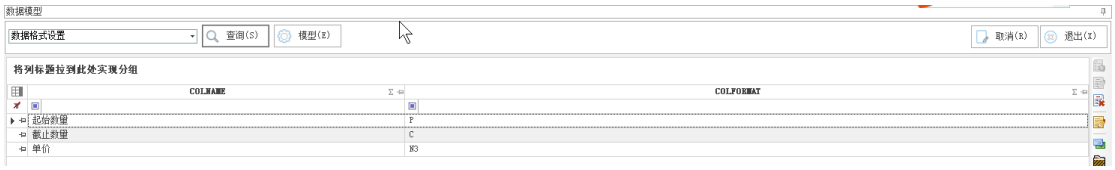
20 Excel 数据访问



- 通过简单的系统配置，即可实现访问本地 Excel 文件中的数据，结合模型设计即可方便实现利用 Excel 中的数据更新系统数据的功能
- 对于 Excel 文件的访问，系统提供三种模式
  - EXCELDATA 模式
  - EXCELFILE 模式
  - ODBC 模式
- 详细内容请参考《数据源配置》部分的内容

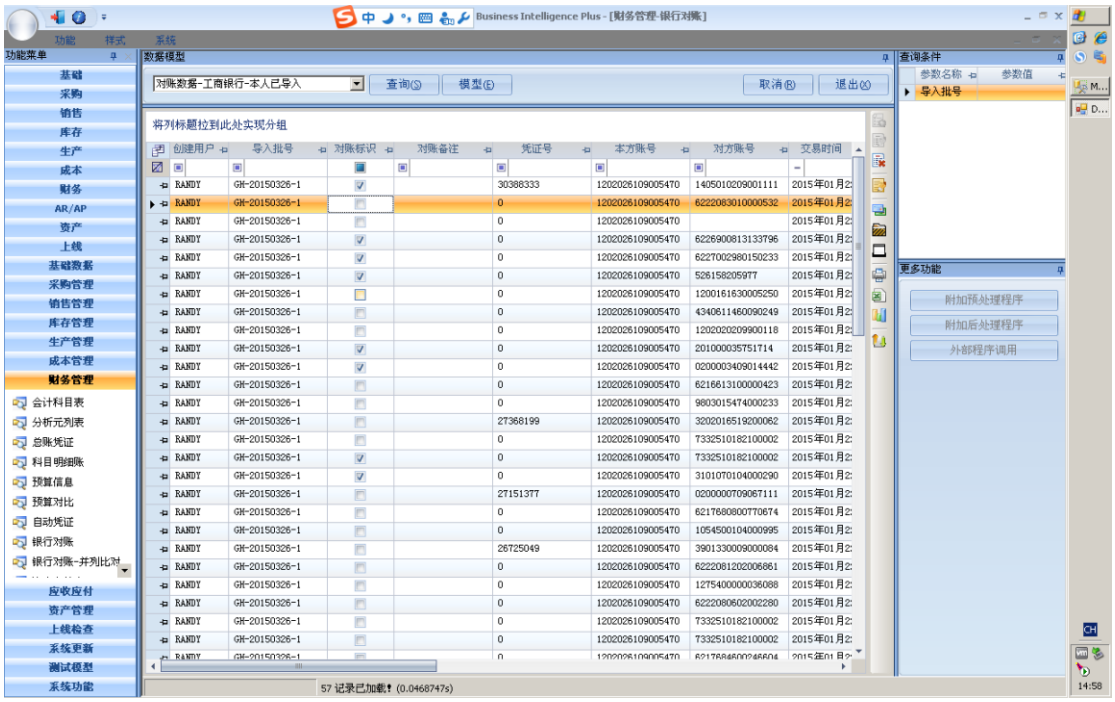
21 列格式显示格式



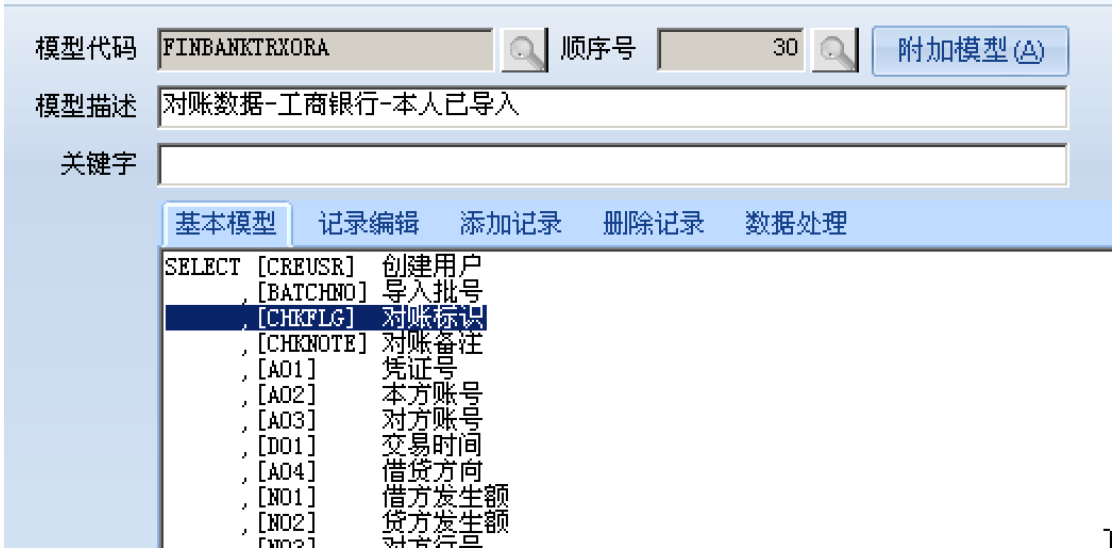


- 通常情况下，系统根据获取的数据，或自动判断列的格式，并以相应的表内控件显示，比如字符型、数字型、布尔型、日期型等等
- 以常用的 SQL Server 和 Oracle 数据库为例，布尔型的数据处理会略有不同

22 SQL Server 模式下的布尔型数据列



- 对于常用的 SQL Server 数据库类型，当数据列为“bit”时，系统自动以“CheckBox”显示内容
- 比如上图中的“对账标识”列

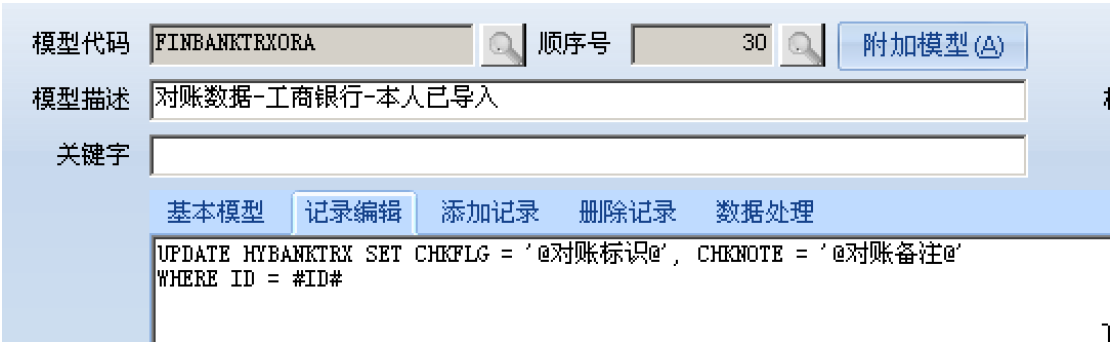


- 当需要对该列进行编辑时，可以使用下面的语法

```
UPDATE HYBANKTRX
SET    CHKFLG = '@对账标识@',
       CHKNOTE = '@对账备注@'
```

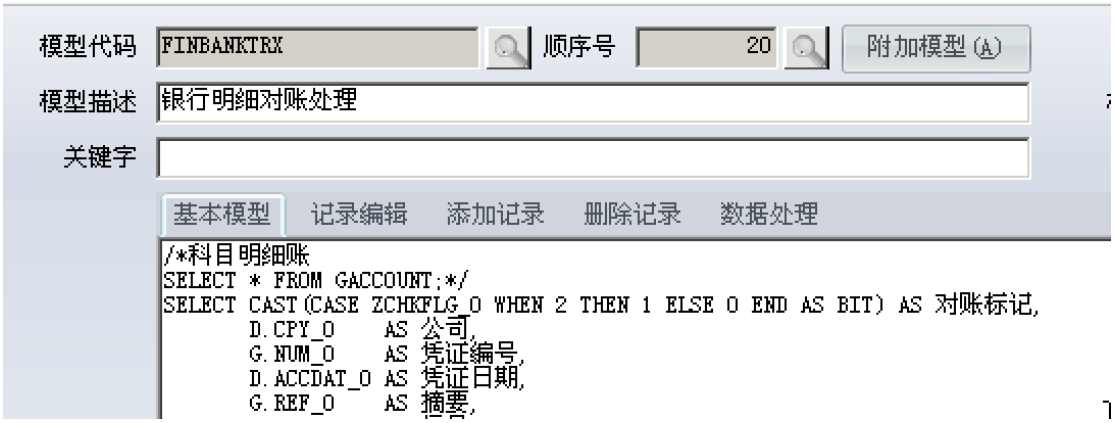
```
WHERE  ID = #ID#
```

- 如下图所示



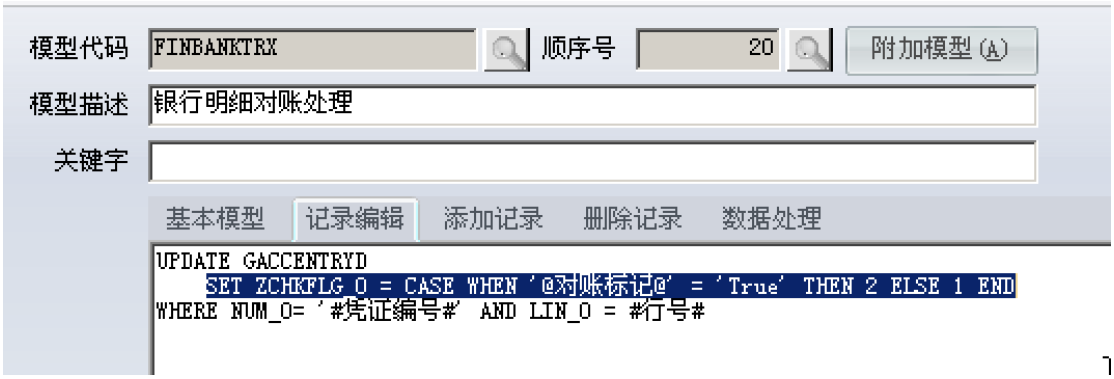
- 对于 Sage ERP X3 的本地菜单 M1 类型的数据，可以采用下面的模式处理

```
CAST(CASE ZCHKFLG_0 WHEN 2 THEN 1 ELSE 0 END AS BIT) AS 对账标记,
```

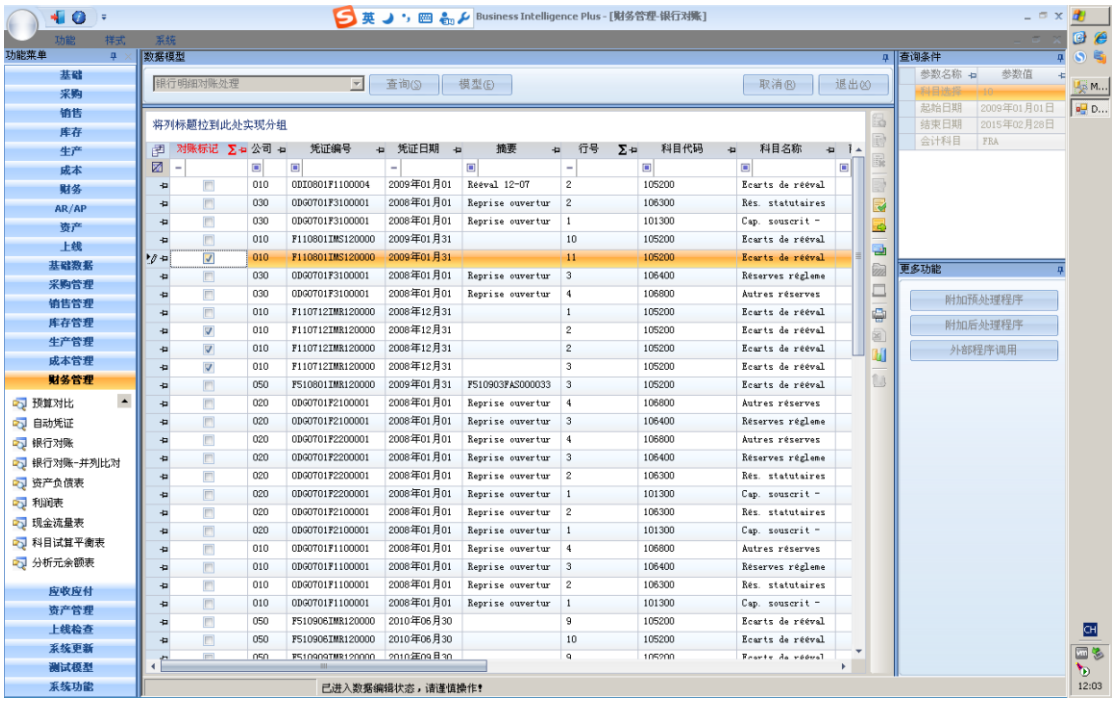


- 编辑代码的格式如下

```
SET ZCHKFLG_0 = CASE WHEN '@对账标记@' = 'True' THEN 2 ELSE 1 END
```



23 Oracle 模式下的布尔型数据列



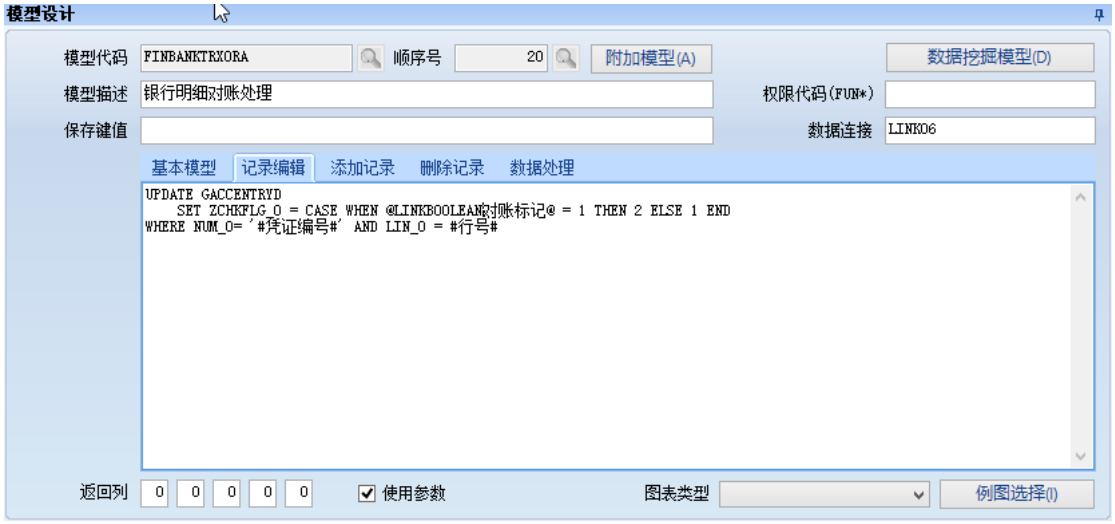
- 由于 Oracle 没有布尔型数据，可以使用数字型数据表示“是/否”
- 为了能够使得系统将其识别为布尔型数据，需要将列标题设置为“LINKBOOLEANxxx”格式
- 比如上图中的“对账标记”的列标题为“LINKBOOLEAN 对账标记”

- 对于 Sage ERP X3 的本地菜单类型（M1），可以参照上面的语法实现

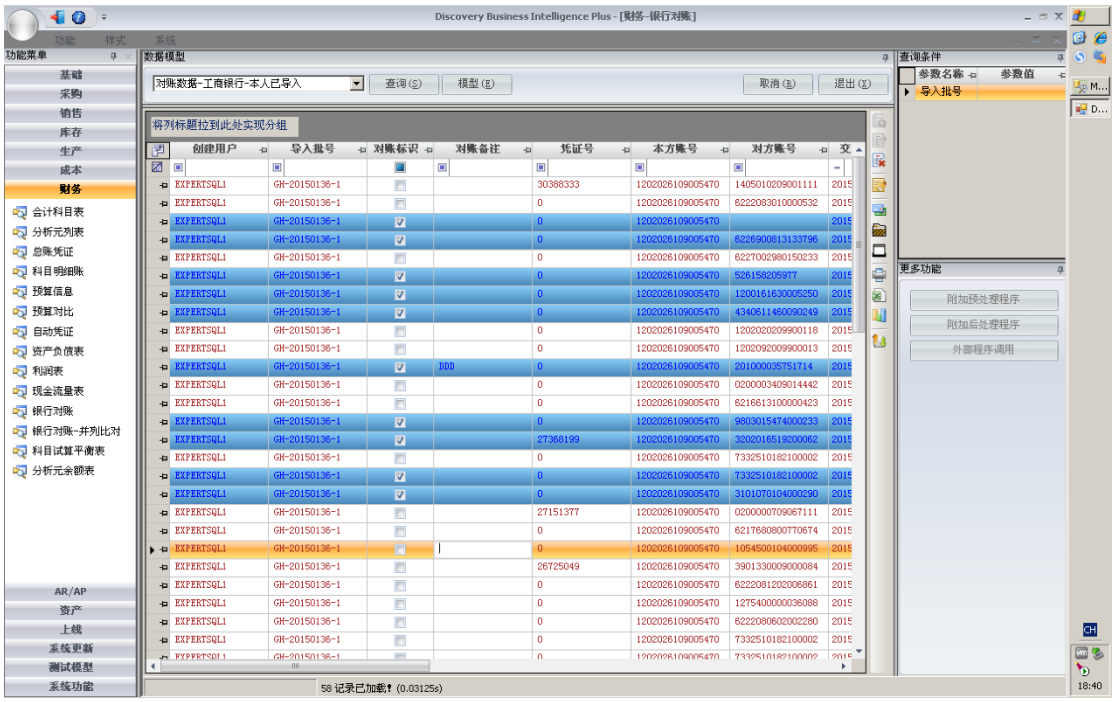
```
SELECT DECODE(D.ZCHKFLG_0, 2, 1, 0) LINKBOOLEAN 对账标记,
```

- 如果需要编辑数据，可以采用下面的语法

```
UPDATE GACCENTRYD
    SET ZCHKFLG_0 = CASE WHEN @LINKBOOLEAN 对账标记@ = 1 THEN 2 ELSE 1 END
WHERE NUM_0= '#凭证编号#' AND LIN_0 = #行号#
```

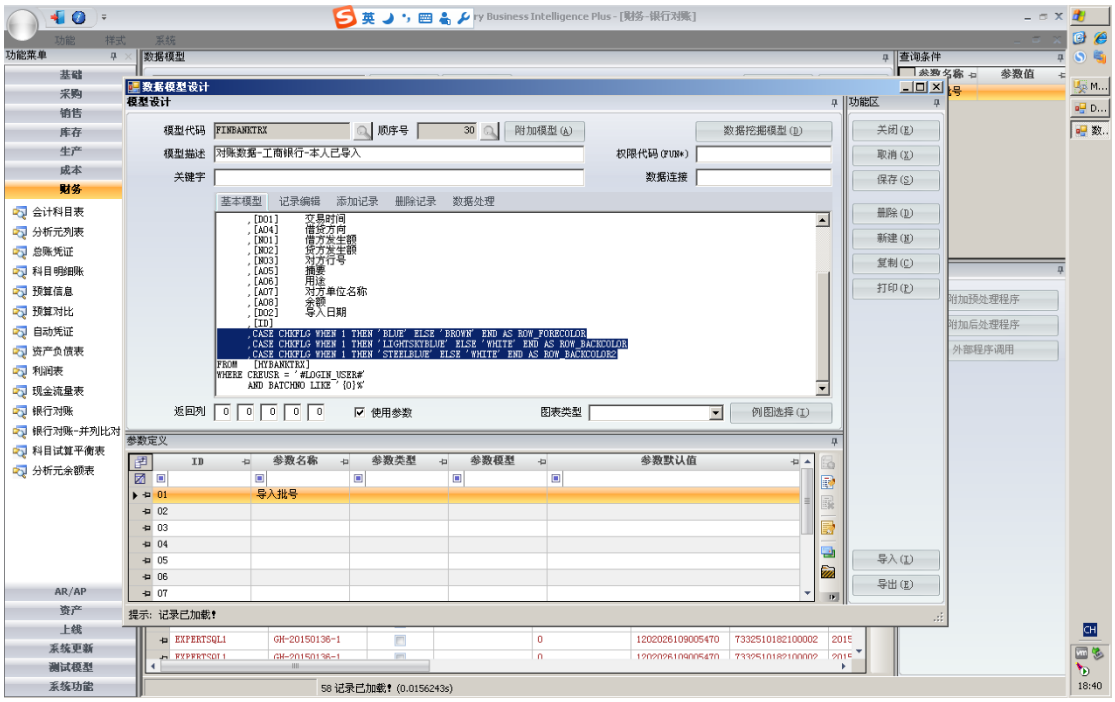


24 行颜色控制



- 根据需要，用户可以非常方便地控制行颜色显示
- 这个特性可以用于突出显示某些特殊的数据

25 行颜色实现



- 为了能够控制不同的数据行以不同的颜色显示，仅需要在数据模型中包含以下各列即可
  - ROW_FORECOLOR：设定的文本颜色
  - ROW_BACKCOLOR：设定的背景色
  - ROW_BACKCOLOR2：设定的第二种背景色
  - ROW_COLOR:同时设定文本颜色和背景颜色

- 背景色设置说明
  - 当仅指定一种背景色时，行的背景以设定颜色显示
  - 当仅指定第二种背景色，或者指定两种背景色时，系统自动将背景设置为从背景色一到背景色二的渐变颜色
  - ROW_FORCECOLOR、ROW_BACKCOLOR、ROW_BACKCOLOR2、ROW_COLOR 值为 NULL,''不改变原来的行颜色
  - 若语句中 ELSE 为’’或者不写 ELSE，系统默认颜色为原来的颜色

- 实例代码
  - 比如为了实现不同对账状态的数据以不同的颜色显示，可以在模型中加入类似如下代码
  - SQL Server 的脚本模型示例

```
,CASE CHKFLG WHEN 1 THEN 'BLUE' ELSE 'BROWN' END AS ROW_FORECOLOR
,CASE CHKFLG WHEN 1 THEN 'LIGHTSKYBLUE' ELSE 'WHITE' END AS ROW_BACKCOLOR
```

```
,CASE CHKFLG WHEN 1 THEN 'STEELBLUE' ELSE 'WHITE' END AS ROW_BACKCOLOR2
```

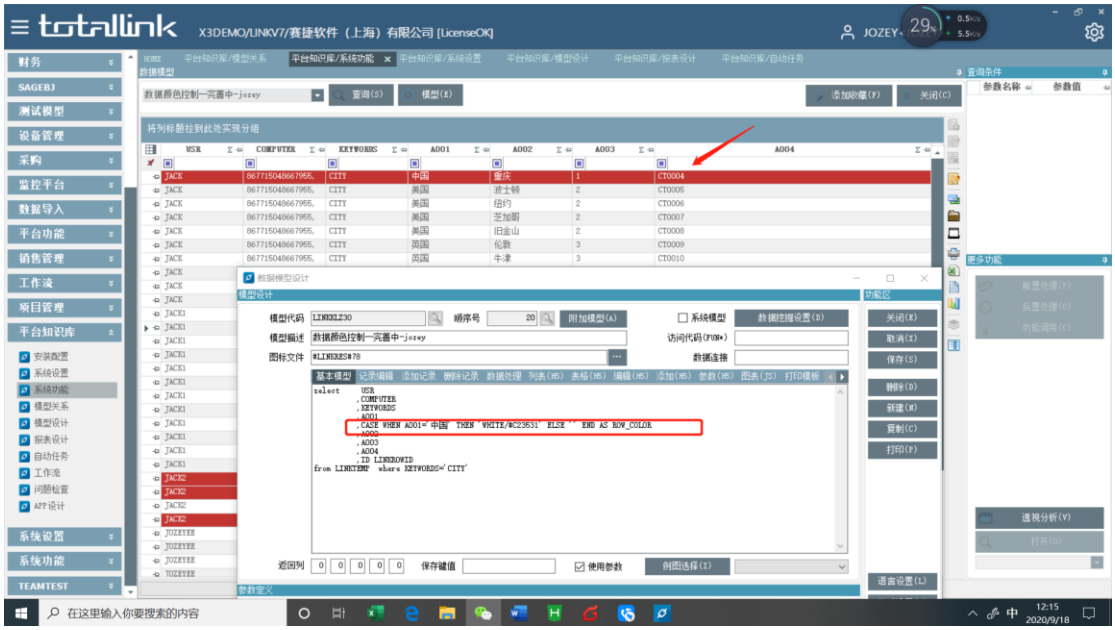
- ORACLE 脚本模型示例（当判断固定的值时可以使用 DECODE 函数）

```
DECODE(D.ZCHKFLG_0, 2, 'WHITE', 'LIGHTYELLOW') ROW_BACKCOLOR,
DECODE(D.ZCHKFLG_0, 2, 'LIGHTBLUE', 'GREY') ROW_BACKCOLOR2,
```

- 系统并不限制只有两种颜色，可以根据需要在不同的条件下设置多种不同的颜色

下面演示 ROW_COLOR 使用流程

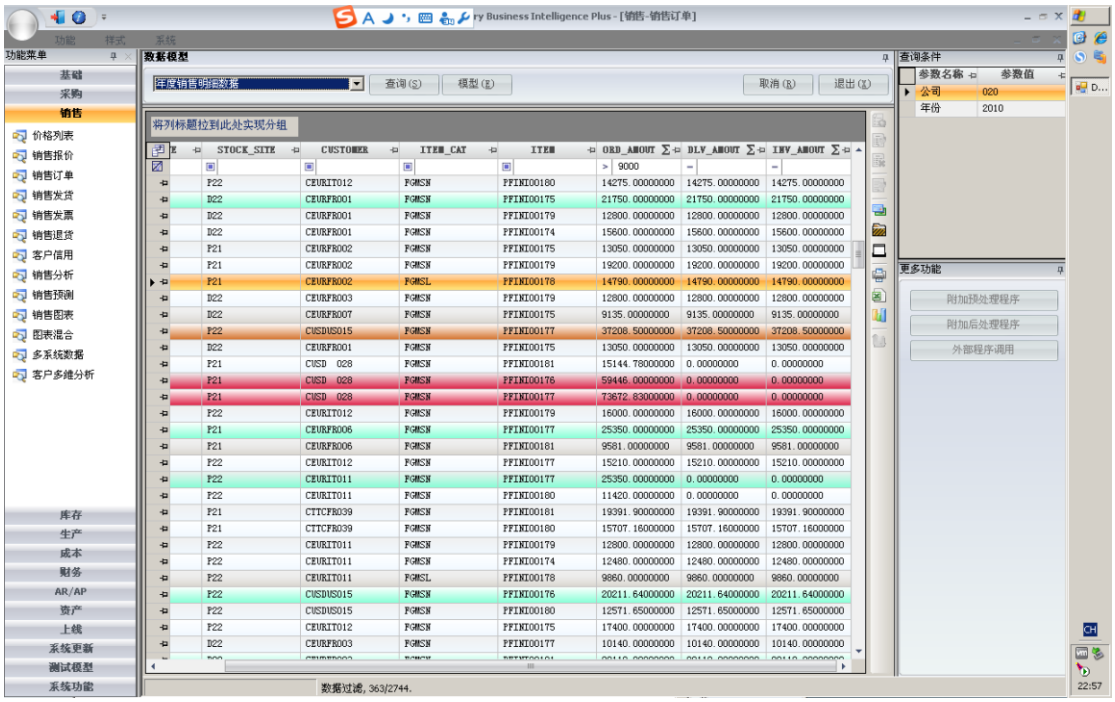
- 如图，在模型中加入 ROW_COLOR 的代码，文字颜色变为白色，背景颜色变成红色



参考代码：

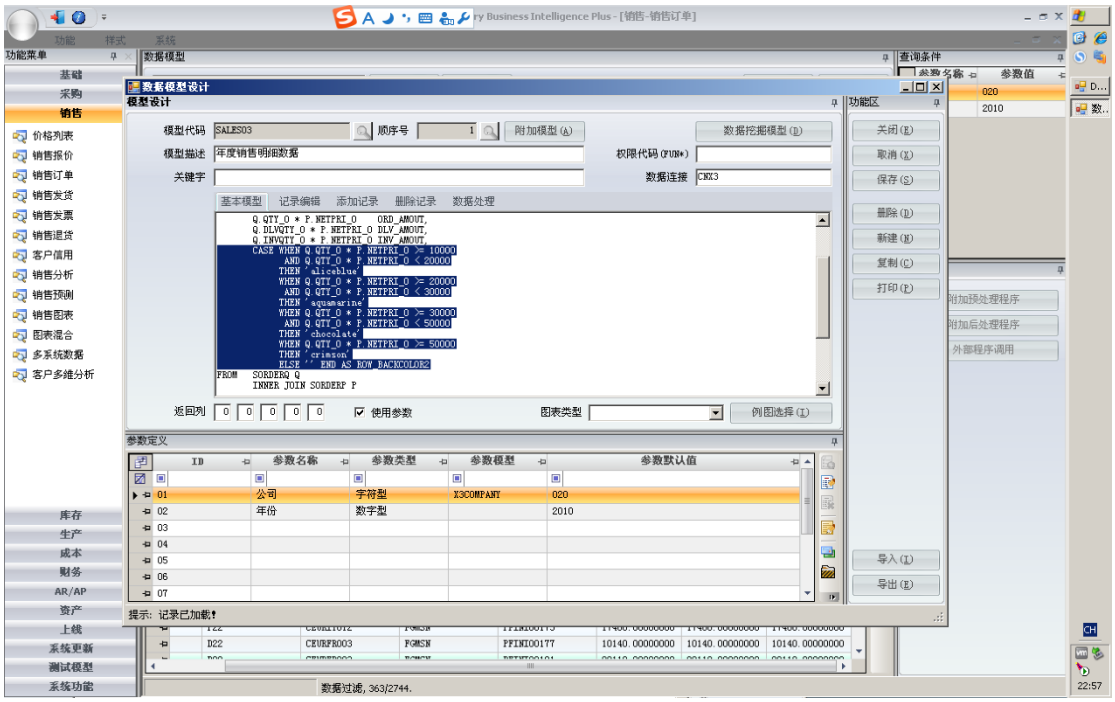
```
select USR
       ,COMPUTER
       ,KEYWORDS
       ,A001
       ,CASE WHEN A001='中国' THEN 'WHITE/#C23531' ELSE '' END AS ROW_COLOR
       ,A002
       ,A003
       ,A004
          ,ID LINKROWID
from LINKTEMP  where KEYWORDS='CITY'
```

26 多种条件颜色格式



- 如图所示，可以按照订单的总金额级别分别以不同的颜色显示

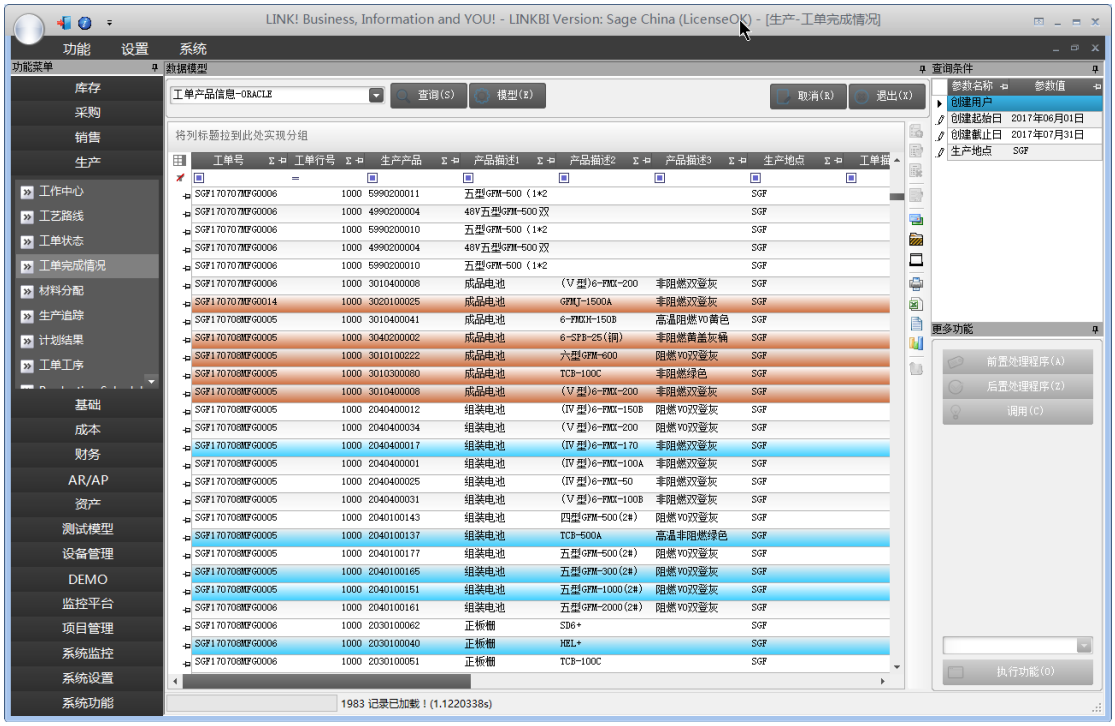
27 多级别颜色的实现



- 要实现多种级别的条件下显示不同的颜色，仅需要适当调整模型即可
- SQL Server 模型的脚本可以参考如下代码

```
CASE
  WHEN Q.QTY_0 * P.NETPRI_0 >= 10000
       AND Q.QTY_0 * P.NETPRI_0 < 20000 THEN 'aliceblue'
  WHEN Q.QTY_0 * P.NETPRI_0 >= 20000
       AND Q.QTY_0 * P.NETPRI_0 < 30000 THEN 'aquamarine'
  WHEN Q.QTY_0 * P.NETPRI_0 >= 30000
       AND Q.QTY_0 * P.NETPRI_0 < 50000 THEN 'chocolate'
  WHEN Q.QTY_0 * P.NETPRI_0 >= 50000 THEN 'crimson'
  ELSE ''
END AS ROW_BACKCOLOR2
```

28 HTML 颜色值

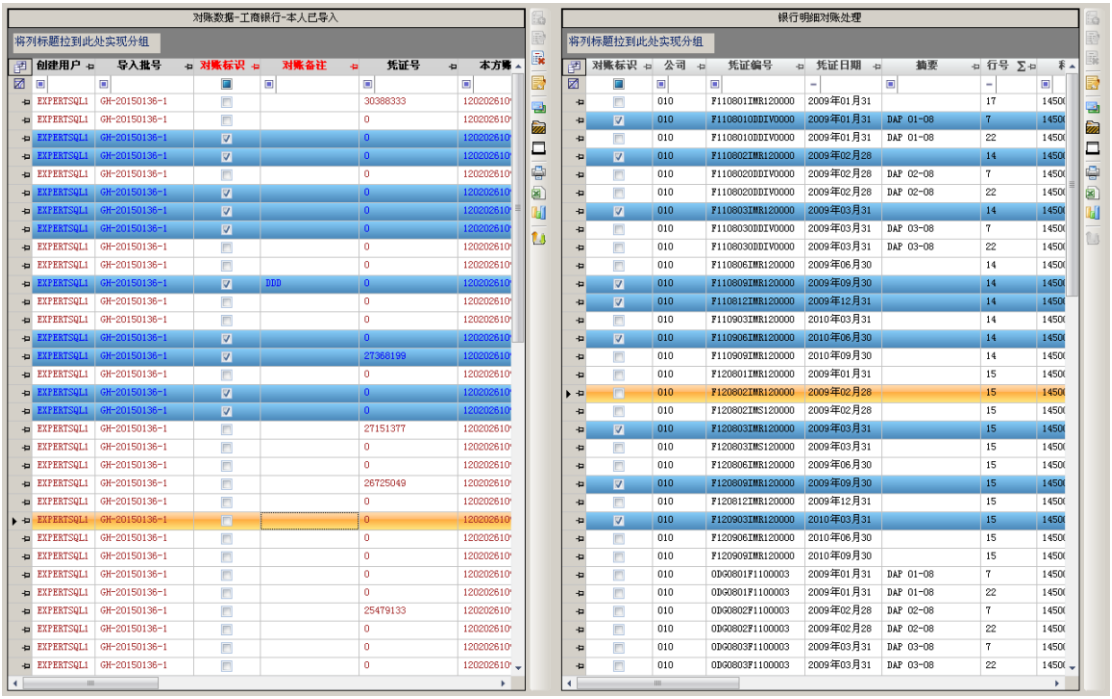


- 如图所示，系统支持以 HTML 颜色值定义的模式控制行颜色，即以“#XXXXXX”的值指定颜色

- 上图的对应颜色控制的脚本如下

```
      , CASE
          WHEN I.RMNEXTQTY_0 > 0
               AND I.CPLQTY_0 > 0 THEN '#33CCFF'
          WHEN I.RMNEXTQTY_0 > 0 THEN '#CC6633'
          ELSE 'WHITE'
        END AS ROW_BACKCOLOR2
```

29 多表模式下的颜色显示



- 对于模型的颜色设置，在多表模式下同样有效
- 上图中“银行对账”时原始银行数据和系统中的对账数据分列左右，并且分别以不同的颜色显示显示

30 颜色列表

- 对于行颜色的取值，系统遵循 HTML COLOR 命名规范
- 具体的颜色名称参考下表

颜色命名对照表

aliceblue	antiquewhite	aqua	aquamarine
azure	beige	bisque	black
blanchedalmond	blue	blueviolet	brown
burlywood	cadetblue	chartreuse	chocolate
coral	cornflowerblue	cornsilk	crimson
cyan	darkblue	darkcyan	darkgoldenrod
darkgray	darkgreen	darkkhaki	darkmagenta
darkolivegreen	darkorange	darkorchid	darkred
darksalmon	darkseagreen	darkslateblue	darkslategray
darkturquoise	darkviolet	deeppink	deepskyblue
dimgray	dodgerblue	firebrick	floralwhite
forestgreen	fuchsia	gainsboro	ghostwhite
gold	goldenrod	gray	green
greenyellow	honeydew	hotpink	indianred
indigo	ivory	khaki	lavender
lavenderblush	lawngreen	lemonchiffon	lightblue
lightcoral	lightcyan	lightgoldenrodyellow	lightgreen
lightgrey	lightpink	lightsalmon	lightseagreen

lightskyblue	lightslategray	lightsteelblue	lightyellow
lime	limegreen	linen	magenta
maroon	mediumaquamarine	mediumblue	mediumorchid
mediumpurple	mediumseagreen	mediumslateblue	mediumspringgreen
mediumturquoise	mediumvioletred	midnightblue	mintcream
mistyrose	moccasin	navajowhite	navy
oldlace	olive	olivedrab	orange
orangered	orchid	palegoldenrod	palegreen
paleturquoise	palevioletred	papayawhip	peachpuff
peru	pink	plum	powderblue
purple	red	rosybrown	royalblue
saddlebrown	salmon	sandybrown	seagreen
seashell	sienna	silver	skyblue
slateblue	slategray	snow	springgreen
steelblue	tan	teal	thistle
tomato	turquoise	violet	wheat
white	whitesmoke	yellow	yellowgreen

31 命令行启动

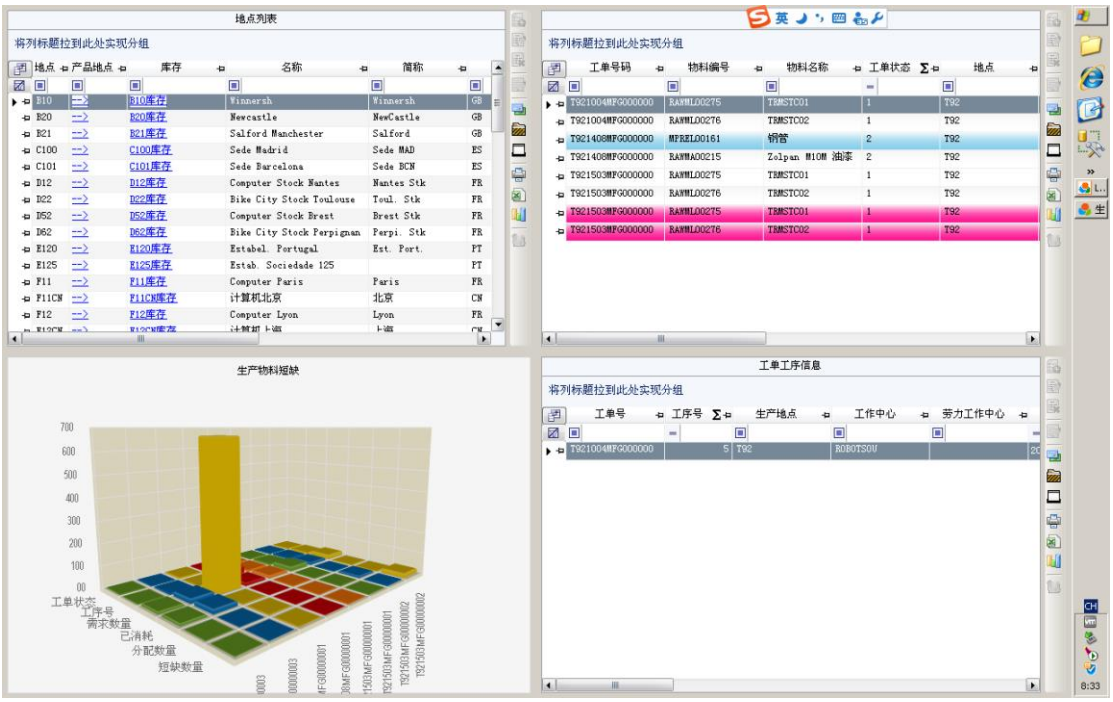
系统支持带参数的命令行启动。

命令行启动参数的语法如下：

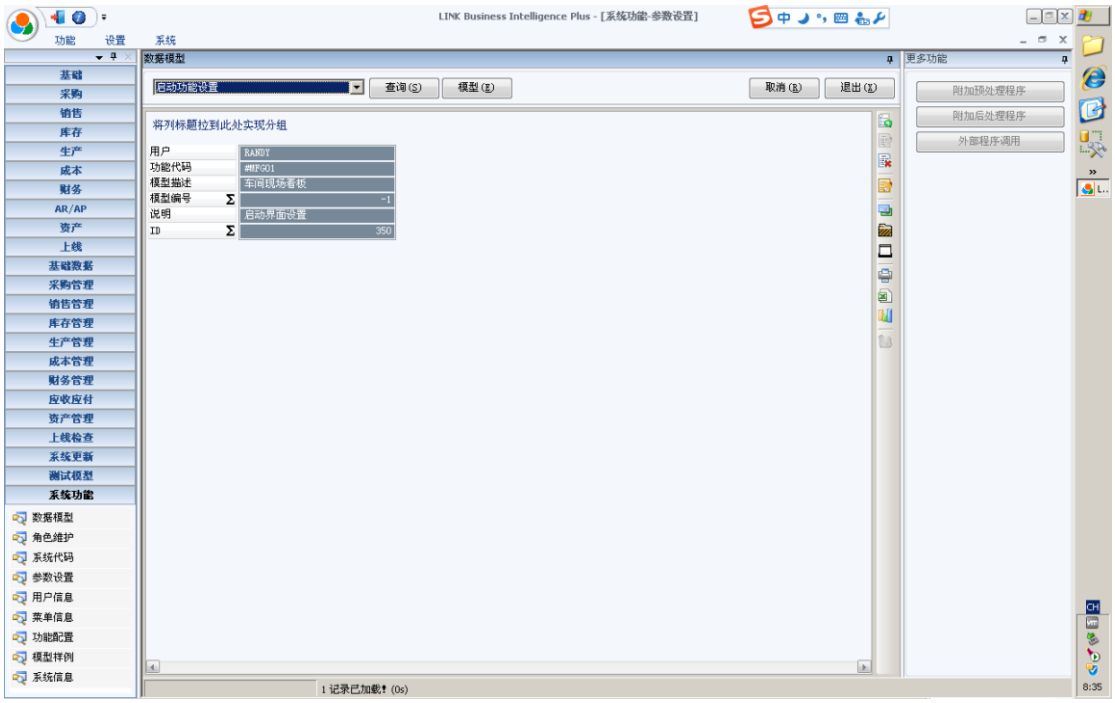
```
LINKBI.EXE user password -start menuno datamodelnum
```

- 启动参数说明如下：
  - LINKBI.EXE/LINKBI，应用程序
  - user，登录用户
  - password，登录密码
  - -start，表示后续参数为自动启动的功能
  - menuno，自动启动的菜单项
  - datamodelnum，自动启动的功能编号
  - 对于#xxx 或者@xxx 形式的图表内容，可以不填写 datamodelnum 参数，或者使用 0 作为参数均可
- 当设置了-start menuno datamodelnum 参数序列时，登录即启动对应功能
- 当未设置-start menuno datamodelnum 参数序列时，系统会检查用户参数中设置的 startpage 参数设置，并启动相应的功能（说明见下一节）。这种方式的优先级低于-start menuno datamodelnum 参数序列。

32 自动启动功能



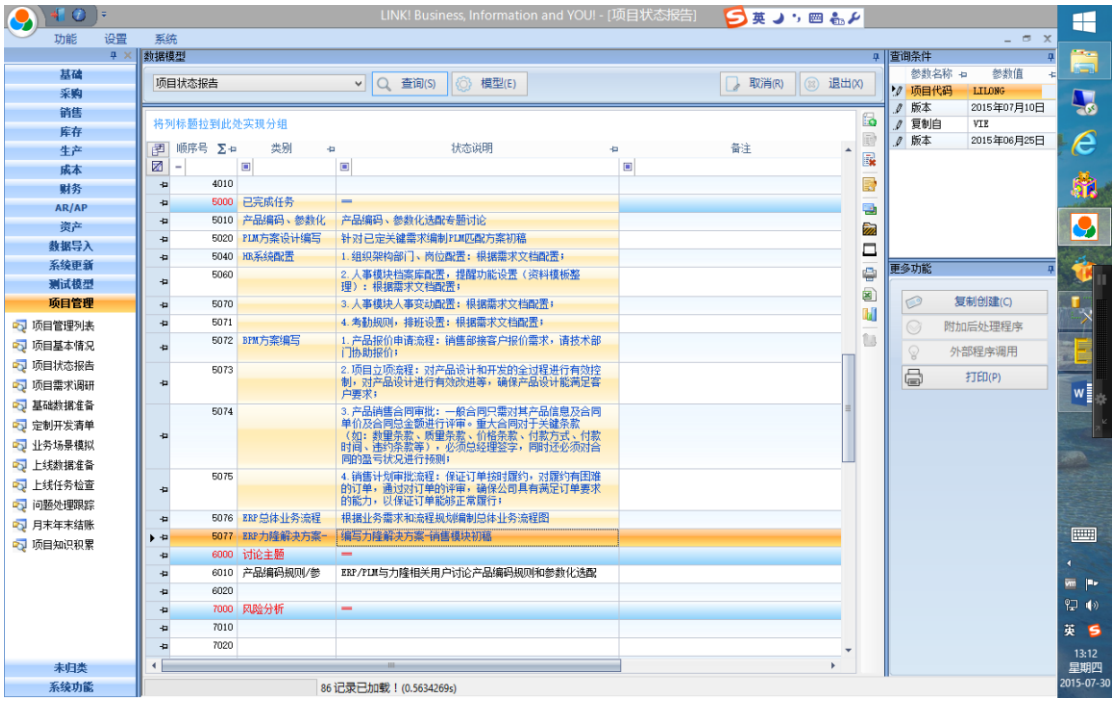
- 如上图所示，我们可以设定某个用户启动的时候自动启动某个特定的功能
- 这个自动启动的功能可以是组合图表，也可以是某个具体的数据分析功能



- 如上图所示，可以使用“系统功能-参数设置-自动启动设置”为用户设置一个

自动启动的功能

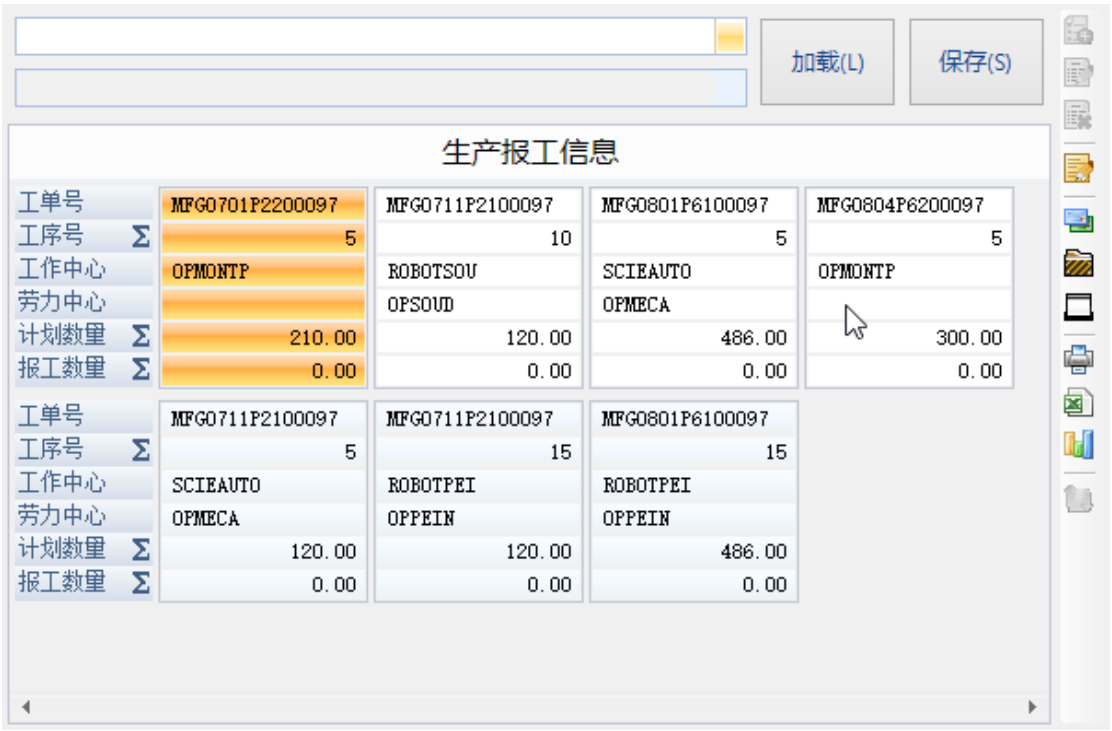
33 多单元格内容的复制



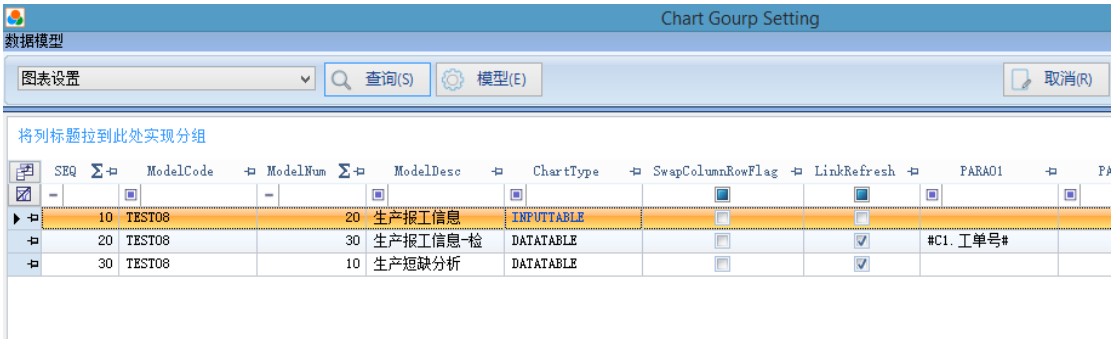
- 如图所示，系统允许选择多个单元格的内容
- 在选中部分内容的情况下，可以使用快捷键（Ctrl+C、Ctrl+V、Ctrl+X）等进行复制、粘帖、剪切等操作。
- 复制内容可以粘帖到 Excel 表中，或者从 Excel 中复制输入粘帖到系统中。

34 参数输入表格

- 如图所示，在设计多图/表数据展现时，允许通过扫描/输入的方式直接输入参数
- 输入参数后，按回车键即可加载数据模式对应的数据
- 左上角的输入框，分别对应模型的参数 1、参数 2，也可以通过选择填入参数



- 上述功能实现，需要在展现模型定义中使用 InputTable 类型



35 多参数关联及参数使用技巧

- 当某个模型需要使用多个参数时，各个参数之间可以相互影响，即后选择的参数在进行参数选择时，可以根据其他参数值影响选择的结果
- 比如，上图中
  - 模型使用三个参数
  - 其中为参数 2 的模型中可以引用已经存在的三个参数（包括参数 2 本身）
- 这样，就可以利用前面输入的参数影响后续选择的内容
- 同样也可以使用已经输入的内容影响选择的内容
  - 比如，参数 2 的条件可以表示为

    WHERE ITMREF_0 LIKE '{1}%'
  - 就可以按照用户输入的内容选择相关的产品

36 文件名使用参数

发送邮件附件、文件导出等功能的文件名中可以引用模型中的参数。

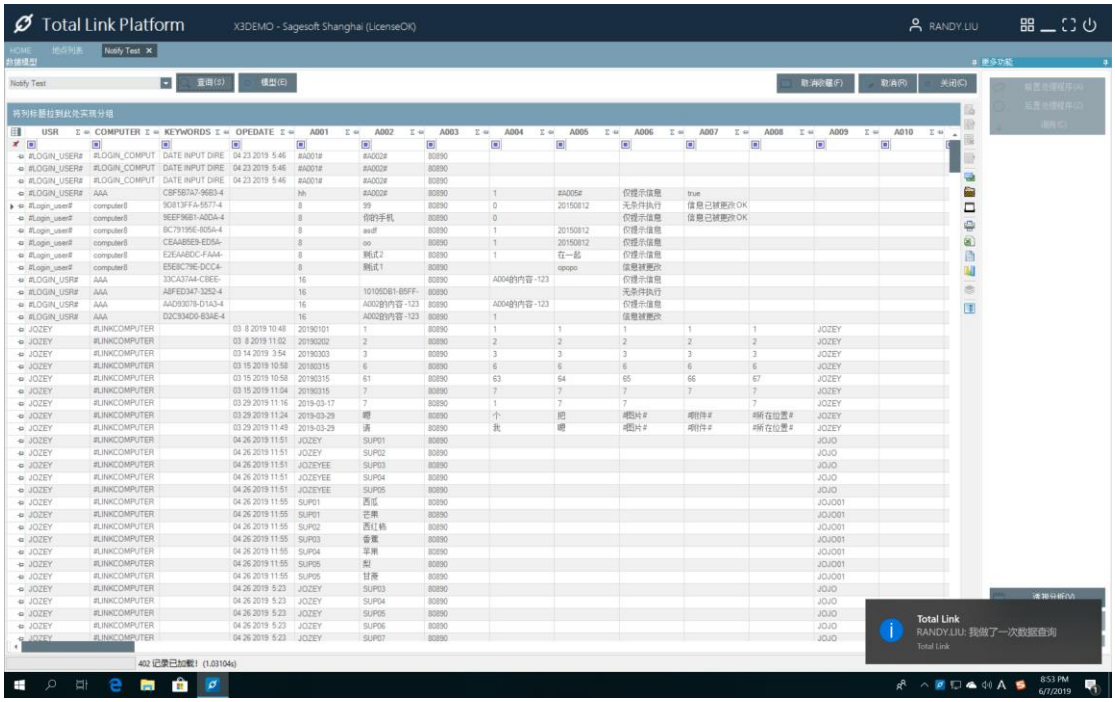
- 如下图，在批次任务执行中的导出模型，数据保存为文件时，可以用参数命名：

```
CALLFUNCTION~FILEEXPORT~
    [{{
        "fileType": "DATA.EXCEL",
        "fileName": "TEST-{0}-{1}",
        "dmWithParams":
        {{
            "dmCode": "TEST01",
            "dmNum": 660,
            "Para": [
                "AAA",
                "BBB",
                "{2}"
            ]
        }}
    }},
    {{
        "fileType": "DATA.TEXT",
        "fileName": "CompanyList",
        "dmWithParams":
        {{
            "dmCode": "TRAINING",
            "dmNum": 70,
            "Para": []
        }}
    }}]
```

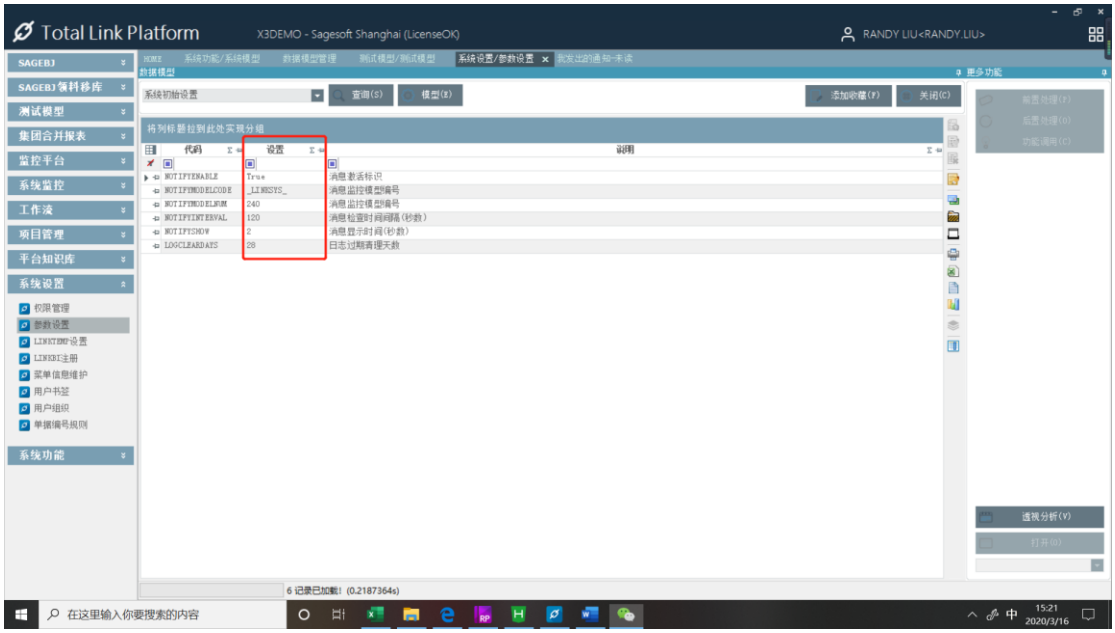
需要注意的是，由于系统设定，只要模型中使用了参数，除了参数本身，所有包含花括号“{}”的地方都改为“{{}}”

37 PC 端消息通知-Notify

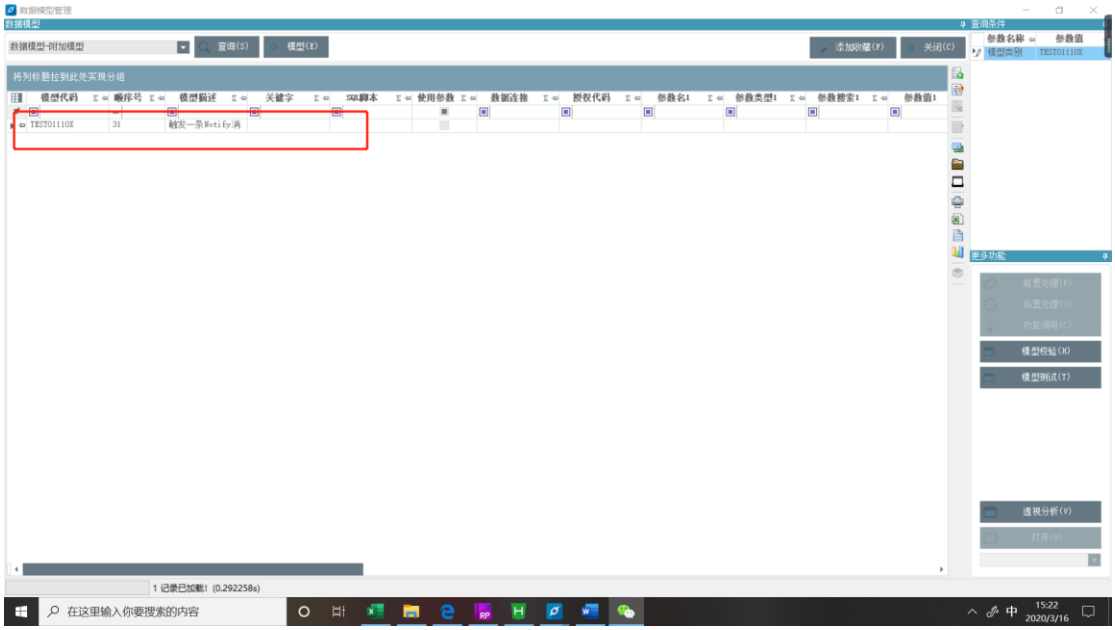
在参数设置->系统初始设置 菜单设置通知参数，且执行系统操作的同时往 LINKNOTIFYPOOL 表插入一条记录，可实现 TotalLINK 客户端消息通知的功能。下图为在执行查询动作时触发客户端通知的案例。

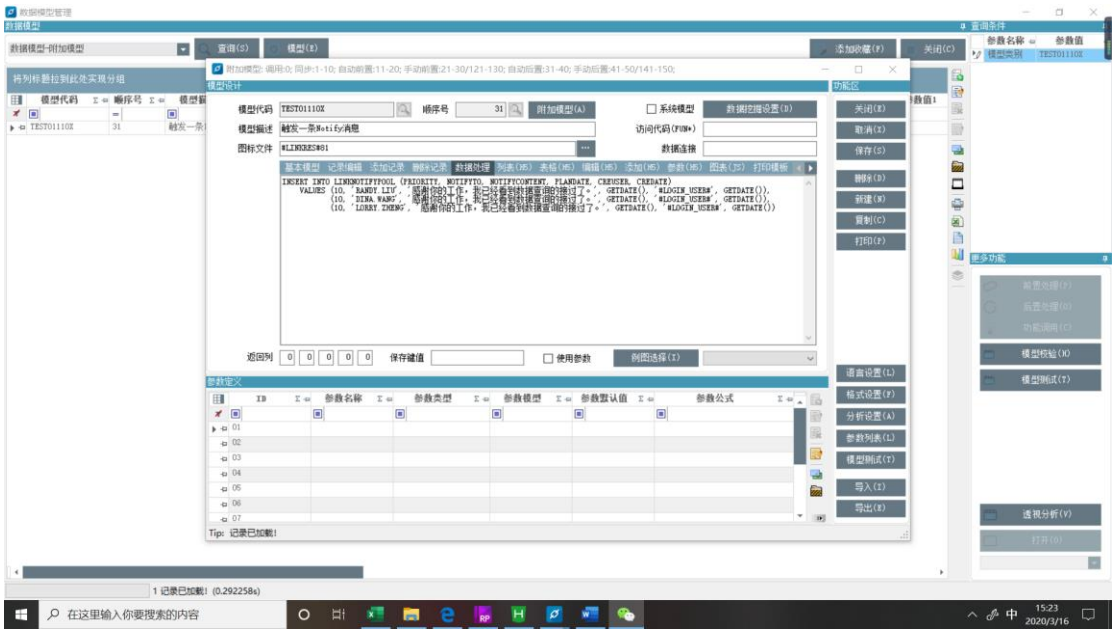


- 在“参数设置->系统初始设置”设置通知参数，NOTIFYENABLE 默认为 false,即消息通知默认关闭，如果需要用到该功能，填 True 即可。



- 基本模型背后添加 31 模型

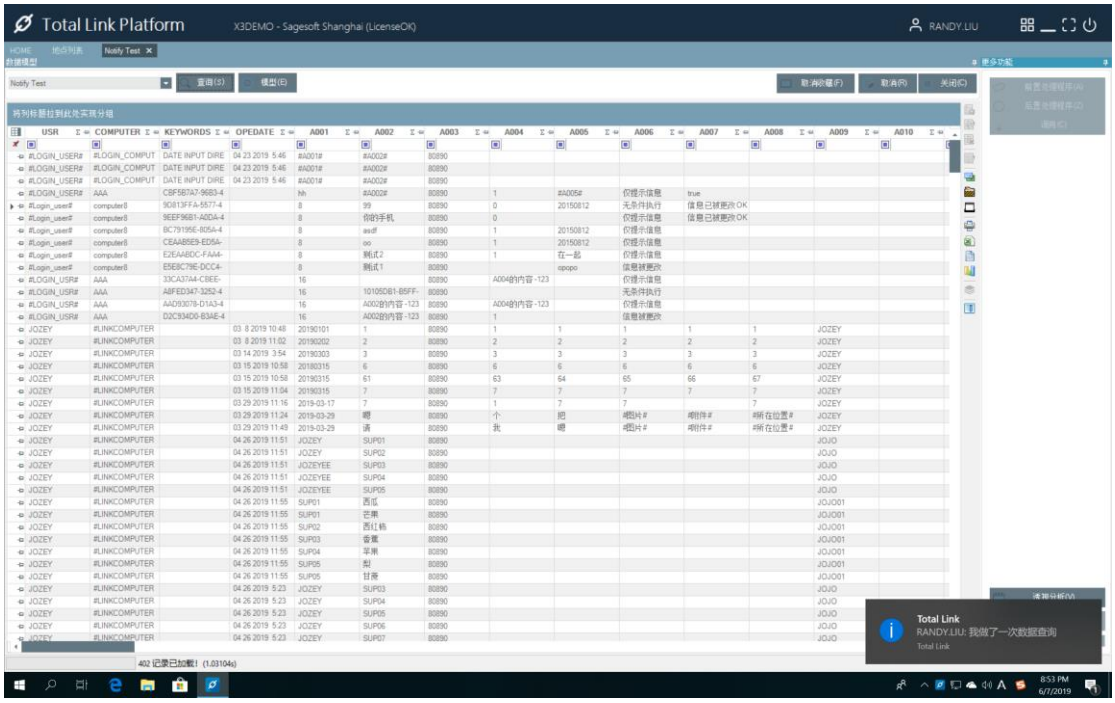




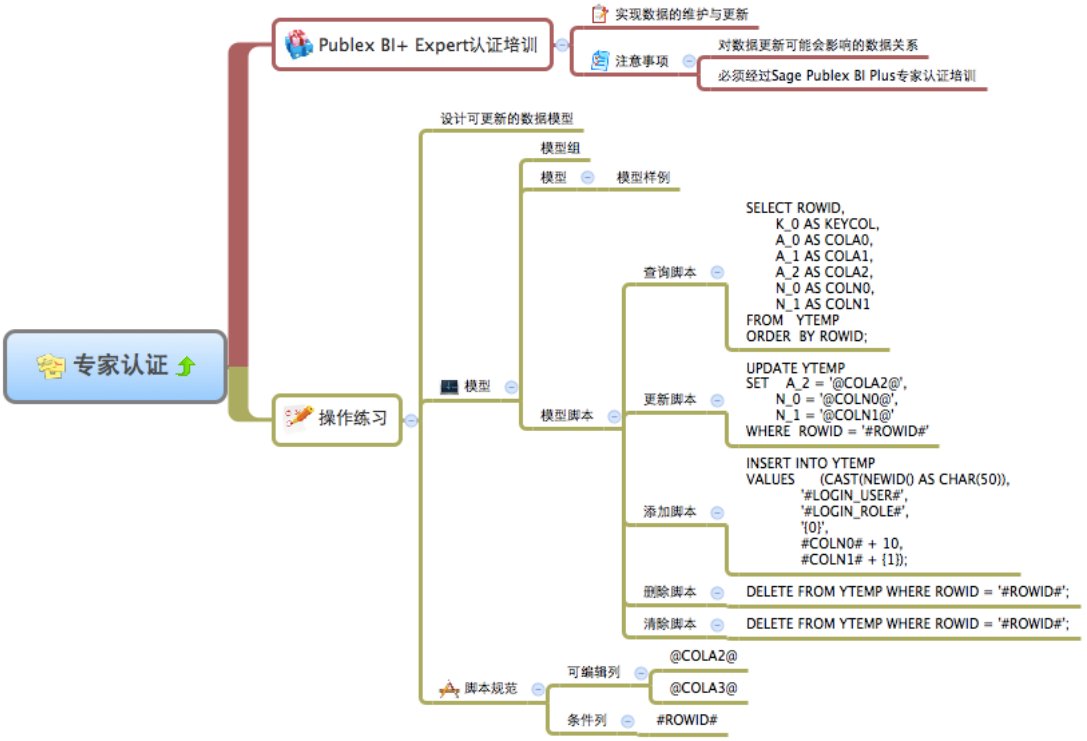
参考代码：

```
INSERT INTO LINKNOTIFYPOOL (PRIORITY, NOTIFYTO, NOTIFYCONTENT, PLANDATE, CREUSER, CREDATE)
    VALUES (10, 'RANDY.LIU', '感谢你的工作，我已经看到数据查询的接过了。', GETDATE(), '#LOGIN_USER#', GETDATE()),
          (10, 'DINA.WANG', '感谢你的工作，我已经看到数据查询的接过了。', GETDATE(), '#LOGIN_USER#', GETDATE()),
          (10, 'LORRY.ZHENG', '感谢你的工作，我已经看到数据查询的接过了。', GETDATE(), '#LOGIN_USER#', GETDATE())
```

- 效果：当查询该模型时，RANDY.LIU 、DINA.WANG、LORRY.ZHENG 用户 PC 端 2 分钟后会提示消息，消息持续两秒后消失

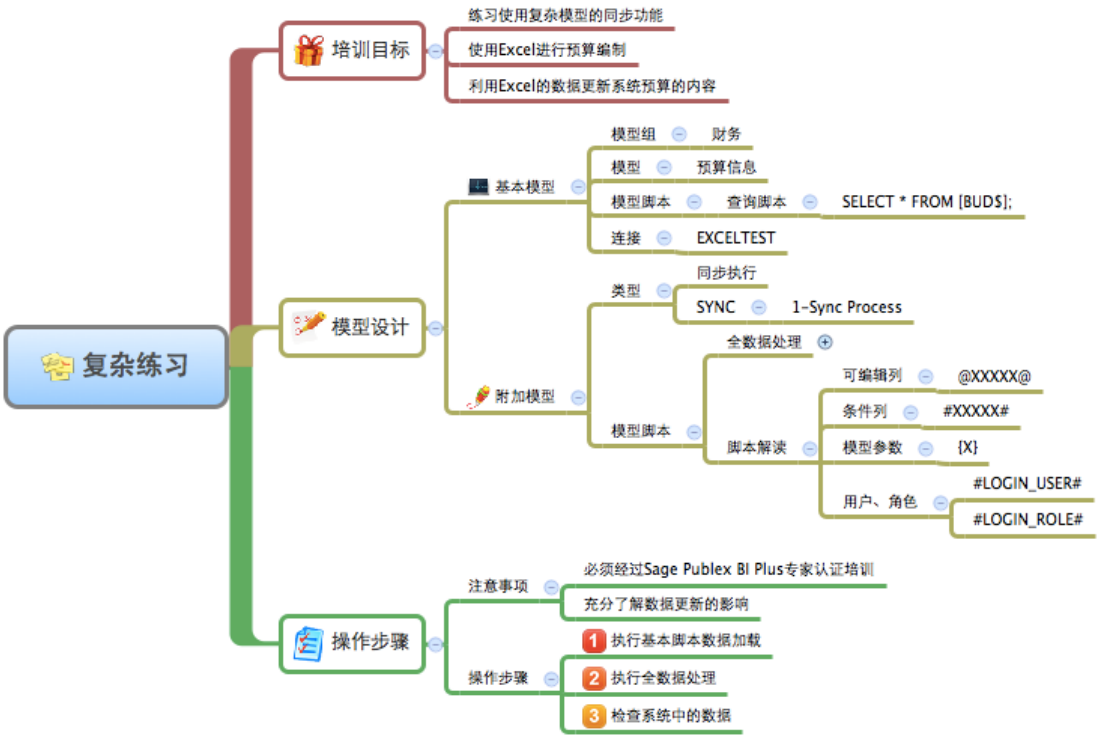


38 专家认证综合练习一



- 上图的练习内容包含了较复杂的内容，包括
  - 使用动态用户及角色
  - 实现可以编辑的数据模型
  - 实现可以删除记录的数据模型
  - 实现对多数据记录处理的数据模型
- 相关内容，请参加专家开发培训

39 专家认证综合练习二



- 上述练习包含了较复杂的内容，包括：
  - 定义 Excel 文件访问
  - 定义同步执行脚本（不同的脚本可以同时在不同的系统中运行）
  - 定义复杂的可以编辑列等
- 相关内容，请参加专家开发培训