

开发手册

Author:

Creation Date: 2015年05月07日

Last Updated: 2015年06月01日

Document Ref:

Version: 1.0

1. **Title, Subject, Last Updated Date, Reference Number**, **and** **Version** are marked by a Word Bookmark so that they can be easily reproduced in the header and footer of documents. When you change any of these values, be careful not to accidentally delete the bookmark. **You can make bookmarks visible by selecting Tools->Options…View and checking the Bookmarks option in the Show region.**
2. To add additional approval lines, press [Tab] from the last cell in the table above.
3. You can delete any elements of this cover page that you do not need for your document. For example, Copy Number is only required if this is a controlled document and you need to track each copy that you distribute.



## 文档控制

**更改记录**

| 日期 | 章节 | | 作者 | 版本 | 更改参考 |
| --- | --- | --- | --- | --- | --- |
|  |  | | |  |  |
| 2015/05/07 | 1.环境准备 | 王亦阳 | | 1.0 | 无前版本 |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |
|  |  |  | |  |  |

4

内容目录

[文档控制 2](#_Toc421518509)

[1. 环境准备 4](#_Toc421518510)

[1.1. 下载软件 4](#_Toc421518511)

[1.2. Kettle安装配置及使用 4](#_Toc421518512)

[1.3. DbVisualizer安装 5](#_Toc421518513)

[2. Kettle应用 10](#_Toc421518514)

[2.1. 设置参数 10](#_Toc421518515)

[2.2. 文件推送 13](#_Toc421518516)

[2.3. 生成sequence主键 16](#_Toc421518517)

[3. Hive应用 19](#_Toc421518518)

[3.1. DML、DDL 19](#_Toc421518519)

## 环境准备

本章节重点介绍Windows系统下环境的安装和配置。

### 下载软件

【请从相关网站上下载所需软件】

### Kettle安装配置及使用

WINDOWS环境：

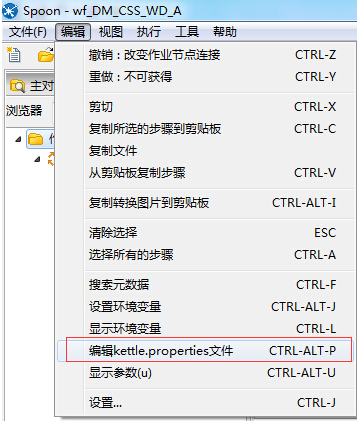
Kettle为绿色版的无需安装，解压即可使用，为了使安装和说明更为简练，我们在根目录下创建一个名为hadoop\_dev的目录（如在C盘的根目录下C:\ hadoop\_dev），然后将后面所有的软件都装在这个目录下。

#### 软件安装

在data-integration\lib目录下增加ojdbc6.jar（连接oracle）和phoenix-4.3.0-client.jar（连接phoenix）。如需要连接其它数据库，需要增加相应的JAR包。

#### 配置

定义全局变量，在kettle.properties中配置全局变量



增加以下变量：

KETTLE\_ETL\_HOME：ETL程序全局目录。**（自定义路径，需要修改）**

KETTLE\_ETL\_PUBLIC：ETL公共程序存放目录。



ORACLE数据库链接变量按照下图按需创建：



Phoenix连接变量如下：



注：后续如有其它需求可以按需增加。

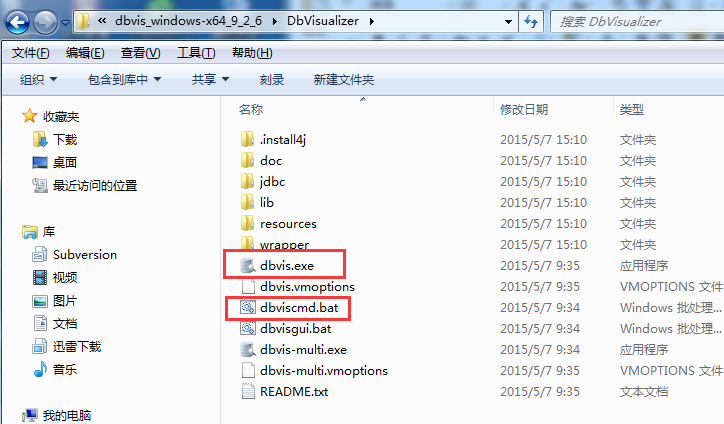
目前已经将kettle.properties文件配置好，进入到C:\Users\用户名\.kettle文件夹中，替换原有kettle.properties。

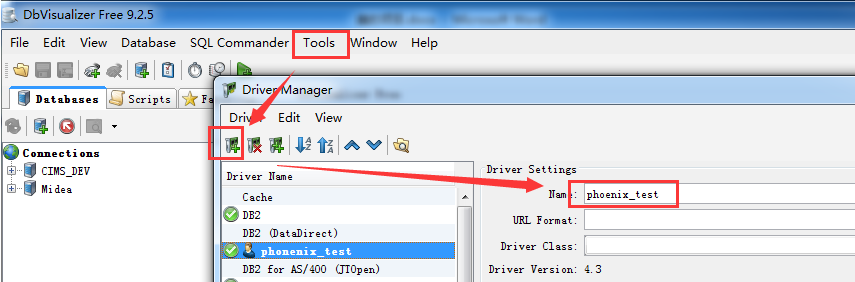
### DbVisualizer安装

为了方便操作Hbase，引入DbVisualizer图形化工具。

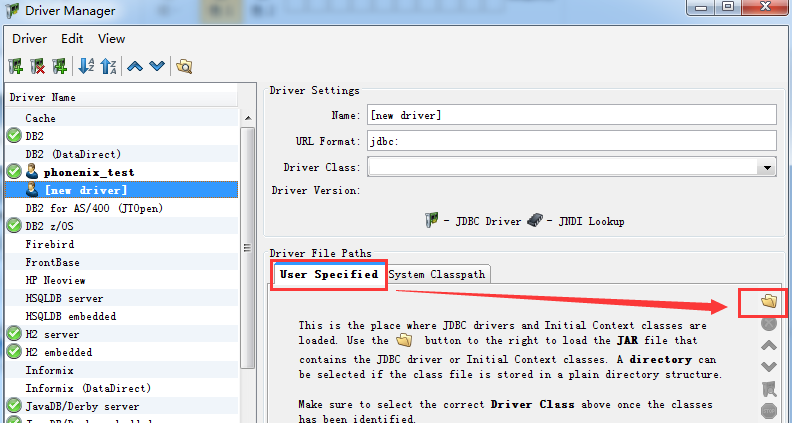
#### 驱动管理

进入到DbVisualizer文件夹中双击dbvis.exe，如不能打开，点击dbviscmd.bat启动软件：

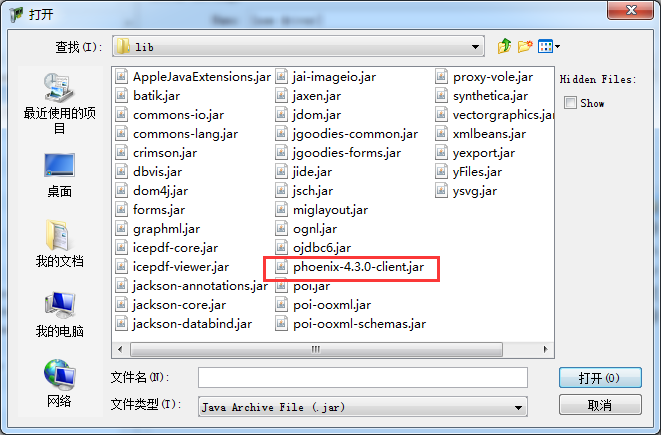
进入软件后点击tools->Driver Manager，点击图示的新建按钮，给Driver Manager命名：



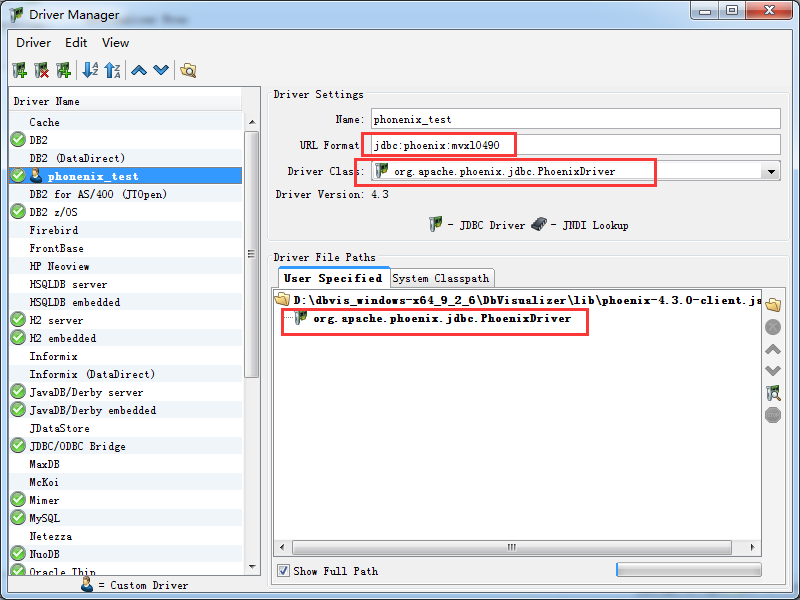
点击User Specified下的“文件夹”图标。



选择phoenix-4.3.0-client.jar，等待加载结束。



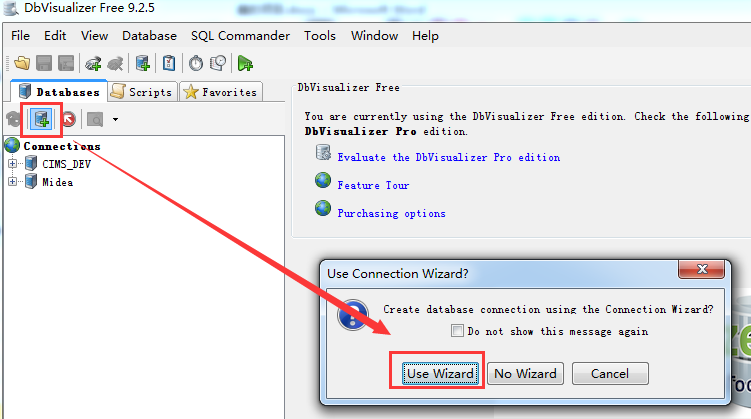
加载成功后效果如下：



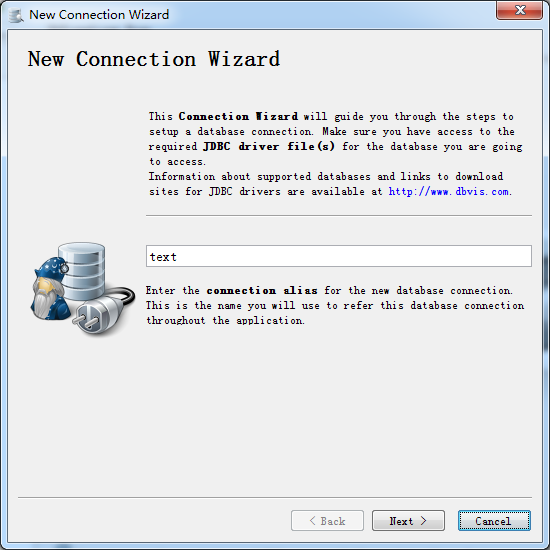
这时候Driver Class中可以选择驱动信息了，然后添加URL Format信息jdbc：phoenix：mvxl0490（说明：mvxl0490是域名，其对应IP及服务器信息需要在host文件中预先配置）

#### 数据库连接

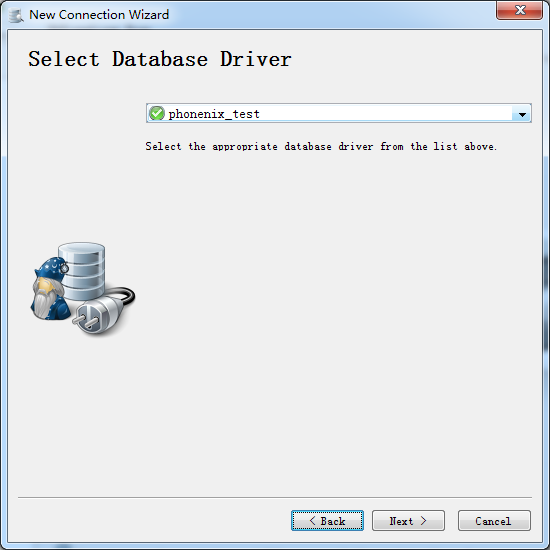
回到主页面，点击创建数据库按钮，如下图所示：



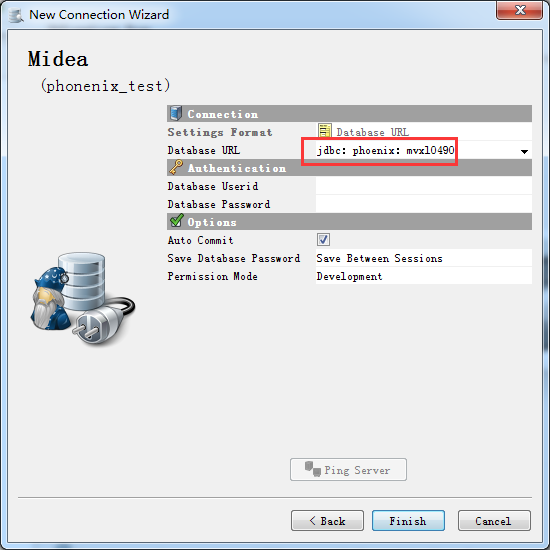
添加一个名称：



下一步选择我们刚才创建的driver manager



填写Database URL ，内容与刚刚创建的Driver Manager中的Driver Class内容相同：



如填写正确可查看到Hbase数据库：

## Kettle应用

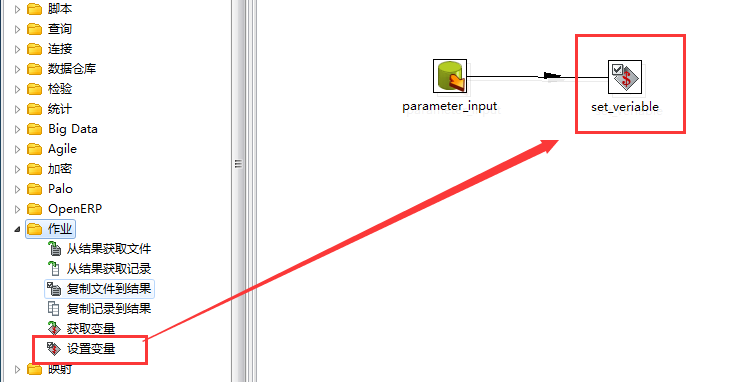
本章介绍了Kettle的一些在项目中的实际应用

### 设置参数

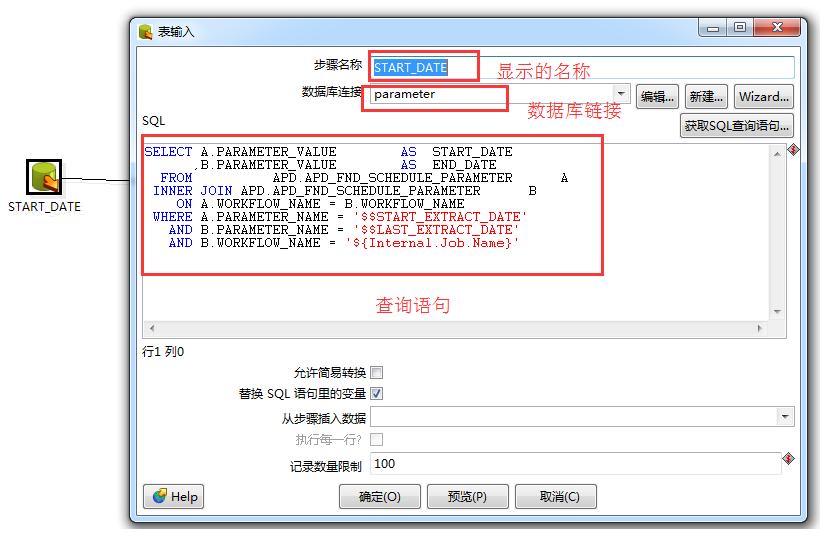
在执行job时我们可能需要一些参数，这些参数我们是通过设置的参数来获取的，如下是我们如何通过从数据库获取一些变量信息，并将这些值设置为我们的参数：

#### 设置参数

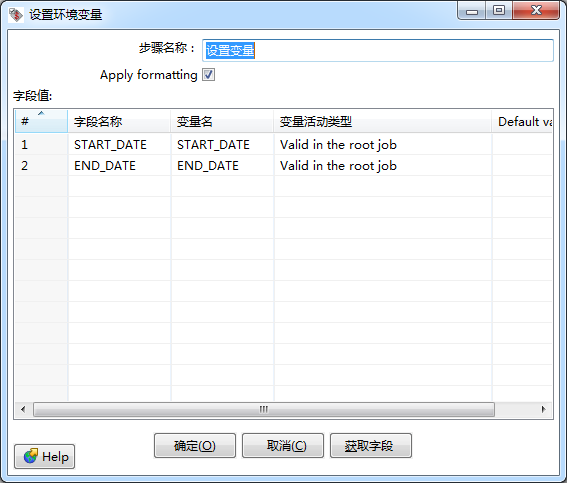
首先我们新建一个转换，拖拽“表输入”和“设置变量”组件，并形成下图所示转换过程。



在“表输入”中，根据下图所示填写。

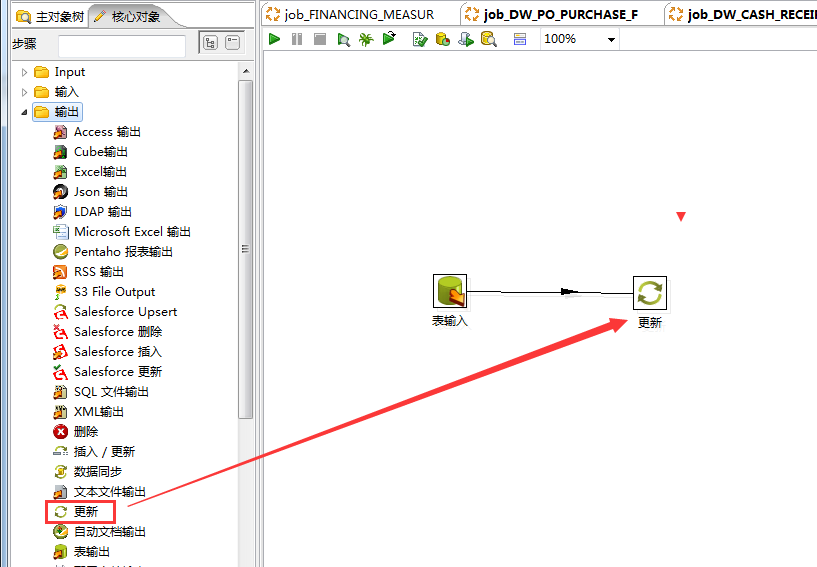


在“设置环境变量”中，主要根据我们的查询条件来设置对应参数信息：

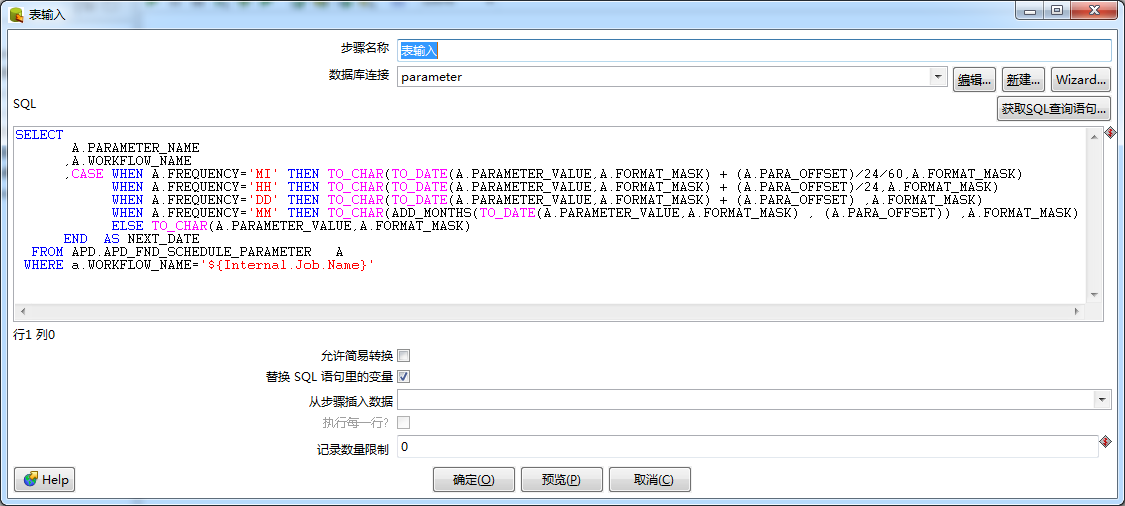


#### 更新参数

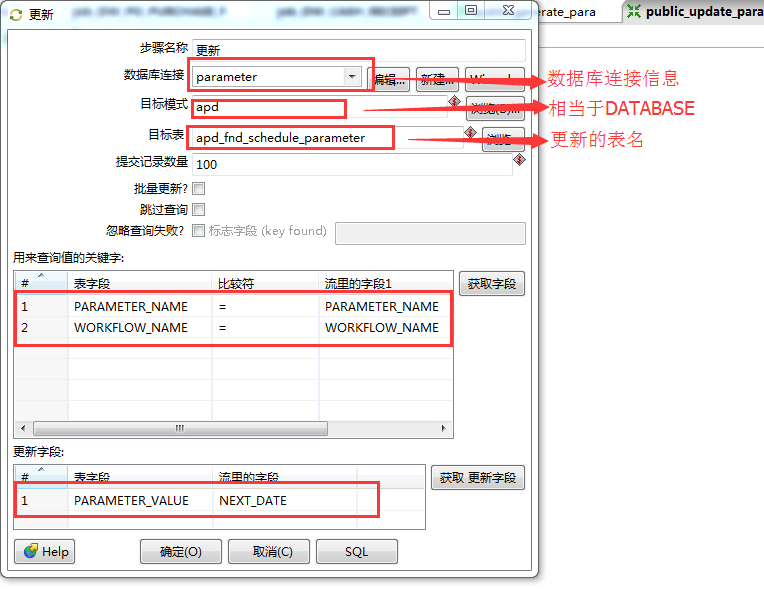
新建一个转换，拖拽如下组件：



同设置参数过程填写“表输入”，将更新字段的sql语句写入：

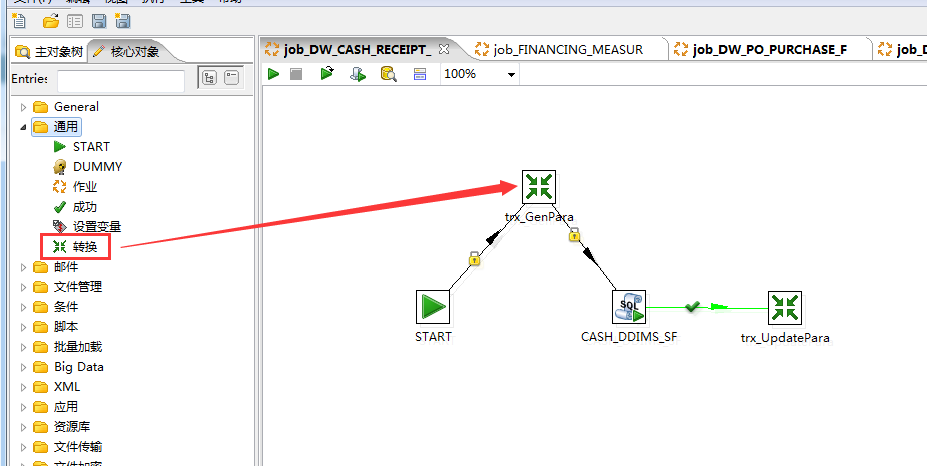


在“更新”中，安装下图填写相应信息：

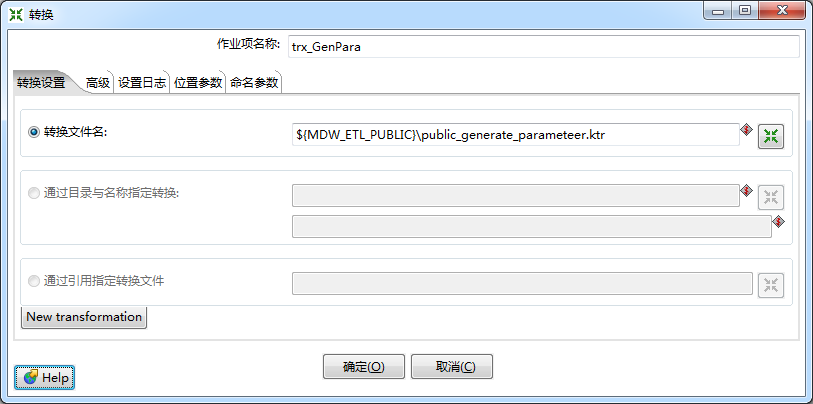


#### 参数使用

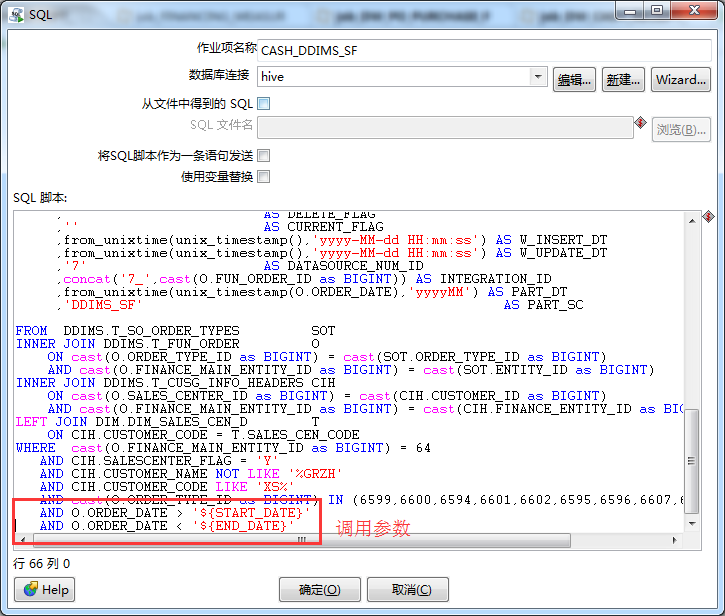
在一个job中，我们来使用上述流程设置的参数，拖拽如下组件：



在trx\_GenPara设置需要调用的“获取参数”的转换，配置相应的文件路径即可。



然后在我们的SQL脚本中调用参数：



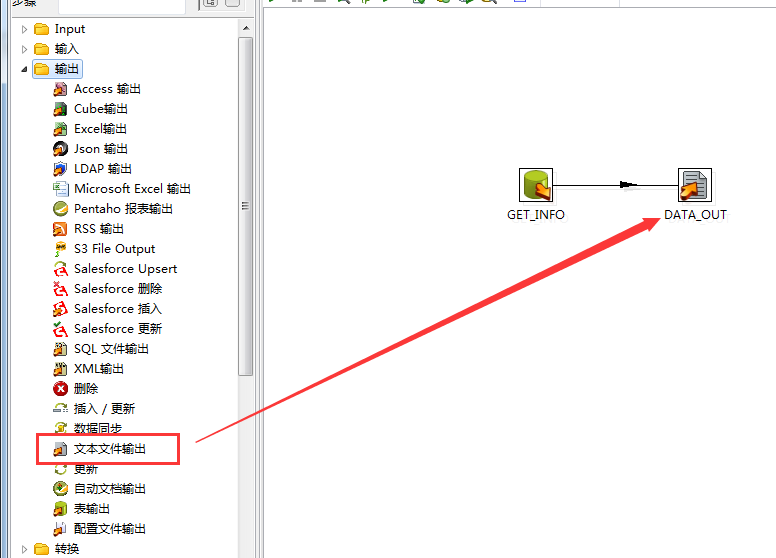
最后按照获取参数的过程，配置“更新参数”。

### 文件推送

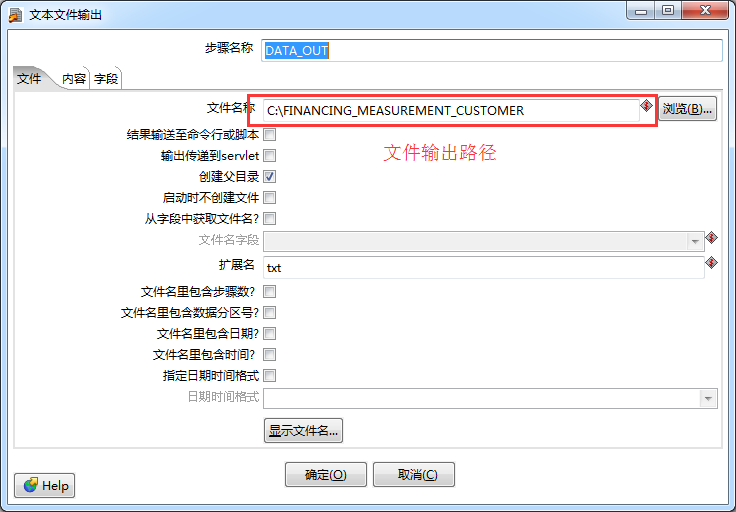
业务场景：通过查询数据库sql语句，将获取的结果生成文档，推送到服务器。

#### 文档生成

首先我们新建一个转换，拖拽“表输入”和“文本文件输出”组件，并形成下图所示转换过程。



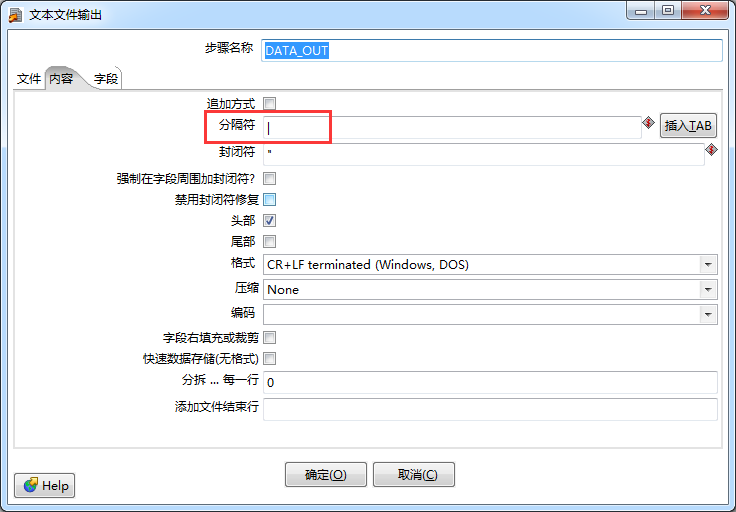
表输入的过程不再复述，在“文本文件输出”中，在“文件”中设置文件的保存位置：



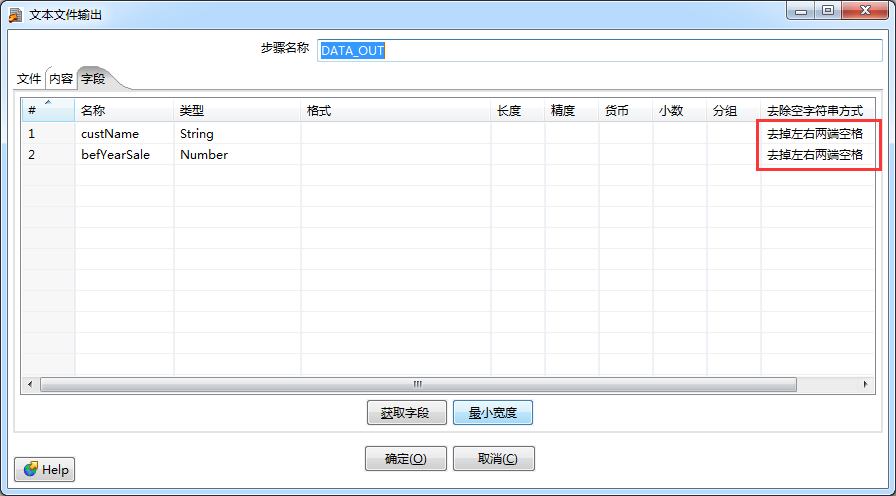
**注：**文件名称可直接填写FTP路径，这样生成的文件会直接推送服务器，如:

[ftp://**用户名**:**密码**@**10.16.14.102**/**文件夹**/](ftp://用户名:密码@10.16.14.102/文件夹/)**文件名（不含后缀）**

在“内容”中，设置我们的输出文件字段的“分割符”：

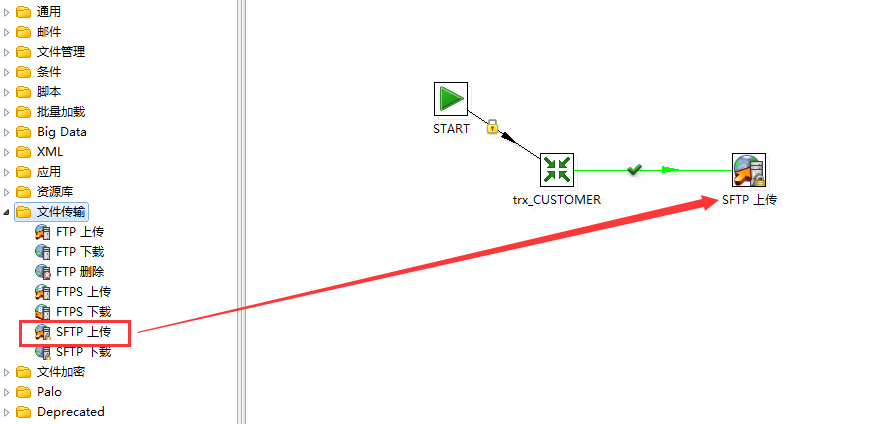


在字段中设置我们文件输对应的字段信息，我们也可以点击“获取字段”来自动生成字段：

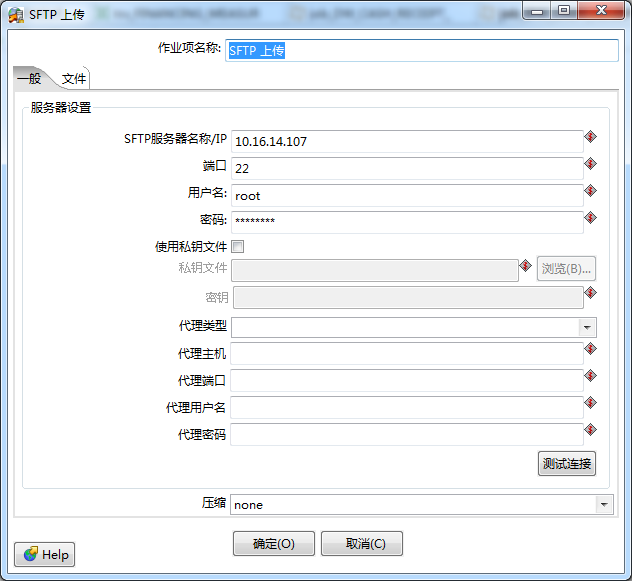


#### 文件推送

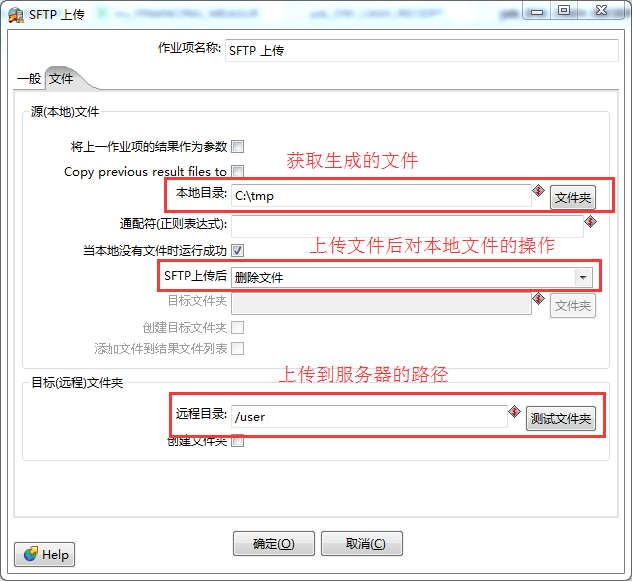
首先我们新建一个转换，拖拽如下组件，并形成下图所示过程。



转换中调用我们刚刚设置的“文档生成”转换，过程不再复述，“SFTP上传”中，填写服务器的连接信息，可点击“测试链接”进行查看链接是否成功。



在“文件”中，配置如下信息：



**注：**

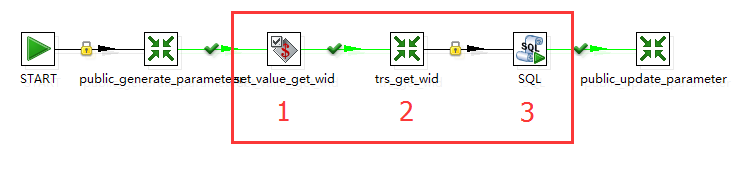
1. 如上传的文件名相同，最新的的文件会覆盖原有文件，不会生成新文件并重命名，若想保留历史文件可在文件名后添加日期信息等参数来区分文件。
2. 获取生成的文件传入的是文件目录，会将当前目录下的文件全部获取，不能通过“本地目录”获取指定的上传文件。

### 生成sequence主键

业务场景：维表新增条目，添加主键（不改变原有主键值，继续生成sequence主键）。

#### 整体流程

首先我们设置要添加主键的表的相应信息，再根据当前主键值进行主键生成。其流程如下：

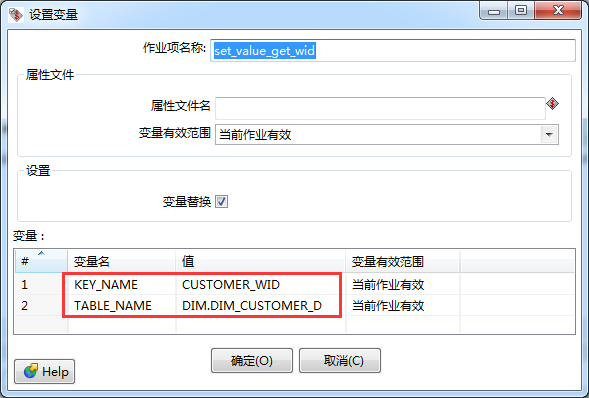


说明：

1. 设置要生成主键表的TABLE\_NAME、KEY\_NAME写入变量
2. 获取该表的主键序列起始值
3. 生成自增主键

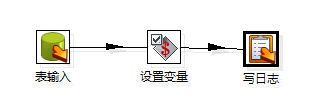
#### 设置参数

在步骤1中，我们设置要生成主键表的TABLE\_NAME、KEY\_NAME写入变量：

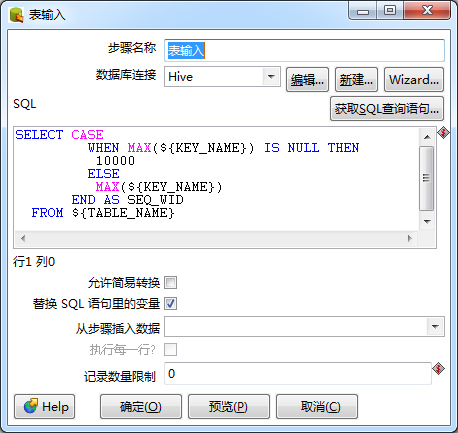


#### 设置主键序列起始值

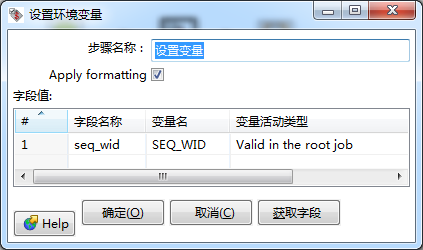
首先我们通过表输入查找到当前表中主键最大值，然后将该值写入变量中，最后的日志是为了测试中在执行过程中可以观测到获取的主键值情况。其流程如下：



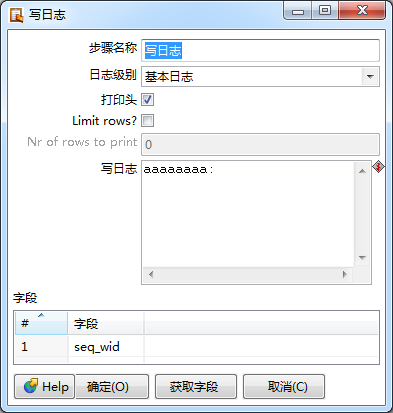
1. 首先，在表输入中通过上一步骤传递的TABLE\_NAME查询当前表的主键最大值，若空则设置一个默认值：



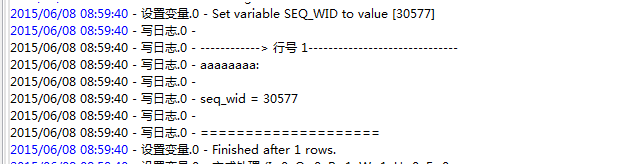
1. 将获取的值写入到变量中



1. 日志输出该主键值

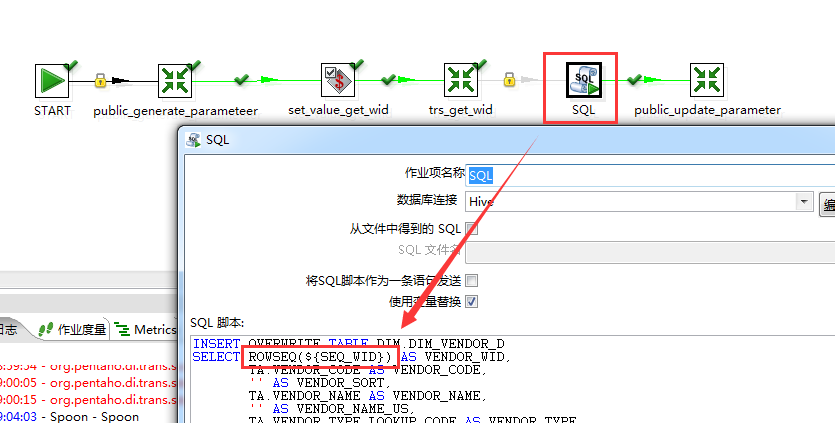


输出效果：



#### 生成自增主键

通过如下的调用方法（“ROWSEQ(${SEQ\_WID})”）来生成自增主键：



## Hive应用

本章介绍了Hive数据库的一些基础操作。

### DML、DDL

#### ODS数据类型对应关系

1. Oracle -> Hive 数据类型对应关系

|  |  |  |
| --- | --- | --- |
| **Oracle -> HIVE 对应关系** | |  |
| **Oracle** | **Hive** | **Impala** |
| NUMBER | BIGINT | BIGINT |
| LONG |  |  |
| NUMBER(x,y) | DOUBLE | DOUBLE |
| FLOAT |  |  |
| DECIMAL | DECIMAL | DECIMAL |
| NUMBER(x,y) |  |  |
| VARCHAR | STRING | STRING |
| CHAR |  |  |
| VARCHAR | VARCHAR | VARCHAR |
| CHAR |  |  |
| DATETIME |  |  |
| DATE |  |  |
| TIMESTAMP | TIMESTAMP | TIMESTAMP |
| DATE |  |  |
| DATATIME |  |  |
| DATE | DATE | DATE |

**注:**

1. Oracle DateTime类型数据抽取到STG\_XXX 表中采用VARCHAR存储’2015-06-02 12:21:32′ 数据;STG\_XXX 转 XXX 过程中，新增TIMESTAMP 数据类型字段，存一份从VARCHAR 转换过来的日期TimeStamp
2. Oracle CLOB类型字段不抽取到STG\_XXX 表中
3. Oracle NUMBER字段默认转换为Hive DOUBLE，若默认Hive Double满足不了要求，手动改为Hive Decimal
4. Hive 表中所有WID 统一使用BIGINT

默认类型转换:

NUMBER -> DOUBLE

VARCHAR -> STRING

CHAR -> STRING

DATETIME -> STRING

#### 建表

在hive中建表

CREATE TABLE IF NOT EXISTS DW.DW\_CASH\_RECEIPT\_F(

PERIOD\_WID BIGINT,

MANAGE\_ORG\_WID BIGINT,

DIRECT\_WID BIGINT,

…

GL\_DATE STRING,

DUE\_DATE STRING,

CASH\_RECEIPT\_TYPE STRING,

…

)

row format delimited fields terminated by '\036' stored as parquet;

说明：

1. ID字段等特殊情况使用BIGINT类型，其他数字类型我们采用DOUBLE类型，可能出现的问题有：**获取的ID末尾会出“.0”，对于原系统NUMBER(12,2)类型的数据，小数位的精度无法控制。**
2. 日期类型的数据在获取时采用STRING类型，若使用DATE类型可能会出现空值等问题，也可采用VARCHER(17)来控制日期类型的数据取值范围
3. 其他数据类型都采用STRING
4. 为该表创建分割符 row format delimited fields terminated by '\036'
5. 使用parquet格式进行文件存储

#### 数据删除

hive执行

DROP TABLE table\_name;

DROP TABLE IF EXISTS table\_name;

**要想删除表数据执行：**

删除指定数据：

insert overwrite table t\_table1 select \* from t\_table1 where XXXX;

清空表：

insert overwrite table t\_table1 select \* from t\_table1 where 1=0;

#### 删除表

注意在hive中进行删表，若在建表时是在root用户下建表的，需要在root用户下删除表，若用其他用户删表则会出现**表已删除**，**但数据没有删除**的情况，这样在重新建立相同表时，在进行数据抽取会造成**数据重复**。

#### ODS系统抽数

在ODS系统抽数时，我们在本项目中是从ORACLE数据库向HIVE数据库进行抽数的，其sqoop抽数脚本如下：

1 sqoop import \

2 --hive-import \

3 --connect jdbc:oracle:thin:@//10.16.17.124:1528/mdccsio \

4 --username BI\_ODS \

5 --password biods \

6 --table SINOCC.ITEM \

7 --hive-table CCS\_ITEM2 \

8 --delete-target-dir \

9 --null-string ‘\\N’ \

10 --null-non-string ‘\\N’ \

11 --fields-terminated-by “\0x1E” \

12 --hive-drop-import-delims \

13 --split-by PRICE\_LINE\_ID

14 --num-mappers 4

15 --where "LAST\_UPDATE\_DATE >= to\_date('${START\_DATE}', 'yyyy-mm-dd') AND LAST\_UPDATE\_DATE < to\_date('${END\_DATE}', 'yyyy-mm-dd')"

16 --query 'select \* from person where $CONDITIONS'

说明：

1. **该脚本是在root用户下执行的，故在重新导数时，要在root用户下drop table，避免数据重复问题。**
2. table指数据来源表
3. hive-table指目标表
4. delete-target-dir指删除抽数过程中所创建的临时文件
5. 在抽数过程中对于空值的处理9、10行，能用 is null 来判断
6. 11行指定对于自行建表的分割规则（注：如使用自动建表，可使用”--create-hive-table \ --split-by 某一字段 \”，来进行分割）
7. sqoop默认启动4个map来跑数，我们可以通过14行来修改此值，当我们指定大于1的map跑数时，要注意原系统表是否有主键，若没有主键要选定一个ID字段来做分割条件（13行为分割条件），当指定map数为1时，也可不指定“split by”
8. 15行为做增量时添加的筛选条件
9. 16行为执行视图抽数时使用的语法，用来替换6行，抽取视图，需指定“where $CONDITIONS”参数

#### DW模型

ODS抽数结束后我们将根据DW模型建表，并根据业务逻辑进行数据读取，其建表规则同上，只是在建表的同时给其添加分区：

partitioned by(PART\_DT STRING,PART\_SC STRING)

说明:

创建分区：partitioned by(PART\_DT STRING,PART\_SC STRING)指定通过PART\_DT（期间范围）和PART\_SC（系统来源）两个字段来进行分区。

建表完成后使用下面语句进行插入数据:

set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

INSERT OVERWRITE TABLE table\_name

PARTITION (**PART\_DT, PART\_SC**)

SELECT column1,

column2,

**…**

coloum1 AS **PART\_DT,**

coloum2 AS **PART\_SC**

FROM table\_name

说明：

1. 常用的转换函数：

指定日期格式：from\_unixtime(unix\_timestamp(TACRH.CASH\_RECEIPT\_DATE),'yyyyMMdd')

获取系统时间：

from\_unixtime(unix\_timestamp(),'yyyy-MM-dd HH:mm:ss')

字符串连接函数：

CONCAT('45','\_',TACRH.CASH\_RECEIPT\_ID)

类型转换函数：

cast(PLL.PO\_LINE\_ID as BIGINT)

1. 分区说明：

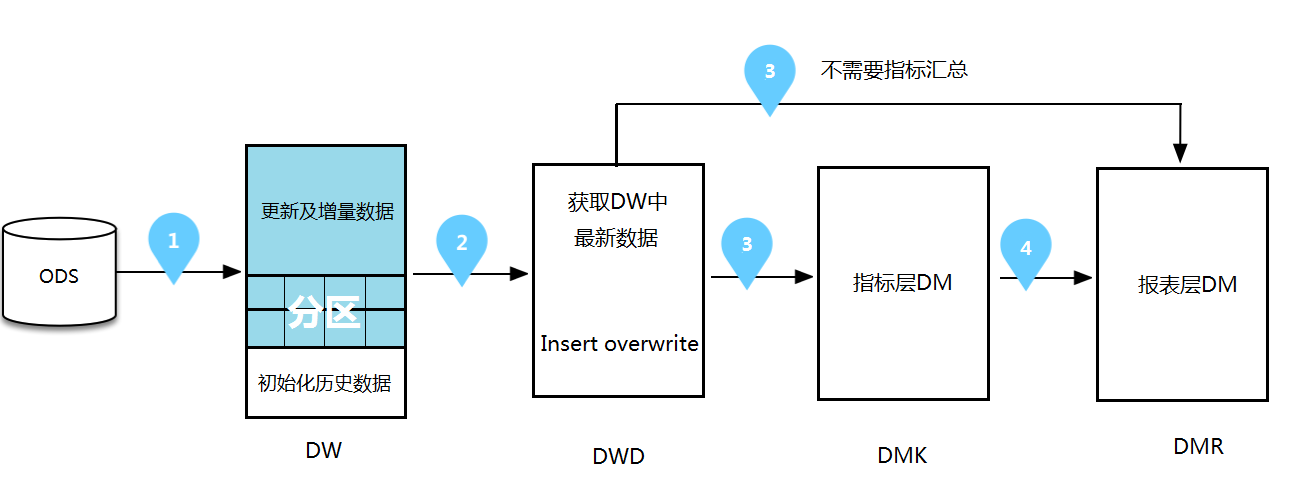
设置分区参数：

set hive.exec.dynamic.partition=true; (可通过这个语句查看：set hive.exec.dynamic.partition.?)

set hive.exec.dynamic.partition.mode=nonstrict;

由于使用INSERT OVERWRITE进行数据插入，在每次插入数据时都会根据SELECT中的分区字段进行分区覆盖，覆盖的分区为，当前分区字段对应的分区。我们设置coloum1、coloum2所对应的值为分区范围。

#### DW模型



1. ODS🡪DW

根据DW模型建表，并建立分区，分区字段为PART\_SC，PART\_DT（PART\_SC = 'source\_code',PART\_DT = '${START\_DATE}'），数据类型为STRING类型，PART\_SC 格式为**yyyy-MM-dd**。

按照下面的SQL脚本来写DW：

set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

INSERT

OVERWRITE TABLE **DW.XXX**

PARTITION (**PART\_SC**, **PART\_DT**)

SELECT

from\_unixtime(unix\_timestamp('${START\_DATE}', 'yyyy-MM-dd'), **'yyyyMMdd'**) AS PERIOD\_WID

,SOURCE\_CODE AS **PART\_SC**

,from\_unixtime(unix\_timestamp('${START\_DATE}', 'yyyy-MM-dd'), **'yyyy-MM-dd'**) AS **PART\_DT**

FROM ODS表

WHERE **主表更新日期字段 >= '${START\_DATE}'**

1. DWD🡪DMK

建立与DW层相同的DW模型按照下面的SQL脚本来写DW：

INSERT

OVERWRITE TABLE **DWD.XXX**

SELECT **a.\***

FROM DW.XXX **a**,

(SELECT 唯一键, MAX(PERIOD\_WID) MPERIOD\_WID

FROM DW.XXX

GROUP BY 唯一键) **b**

WHERE **a.唯一键 = b. 唯一键**

AND **a.PERIOD\_WID = b.MPERIOD\_WID**

1. DMK🡪DMR