# SQL\_HEPPER使用说明书

## 总体结构

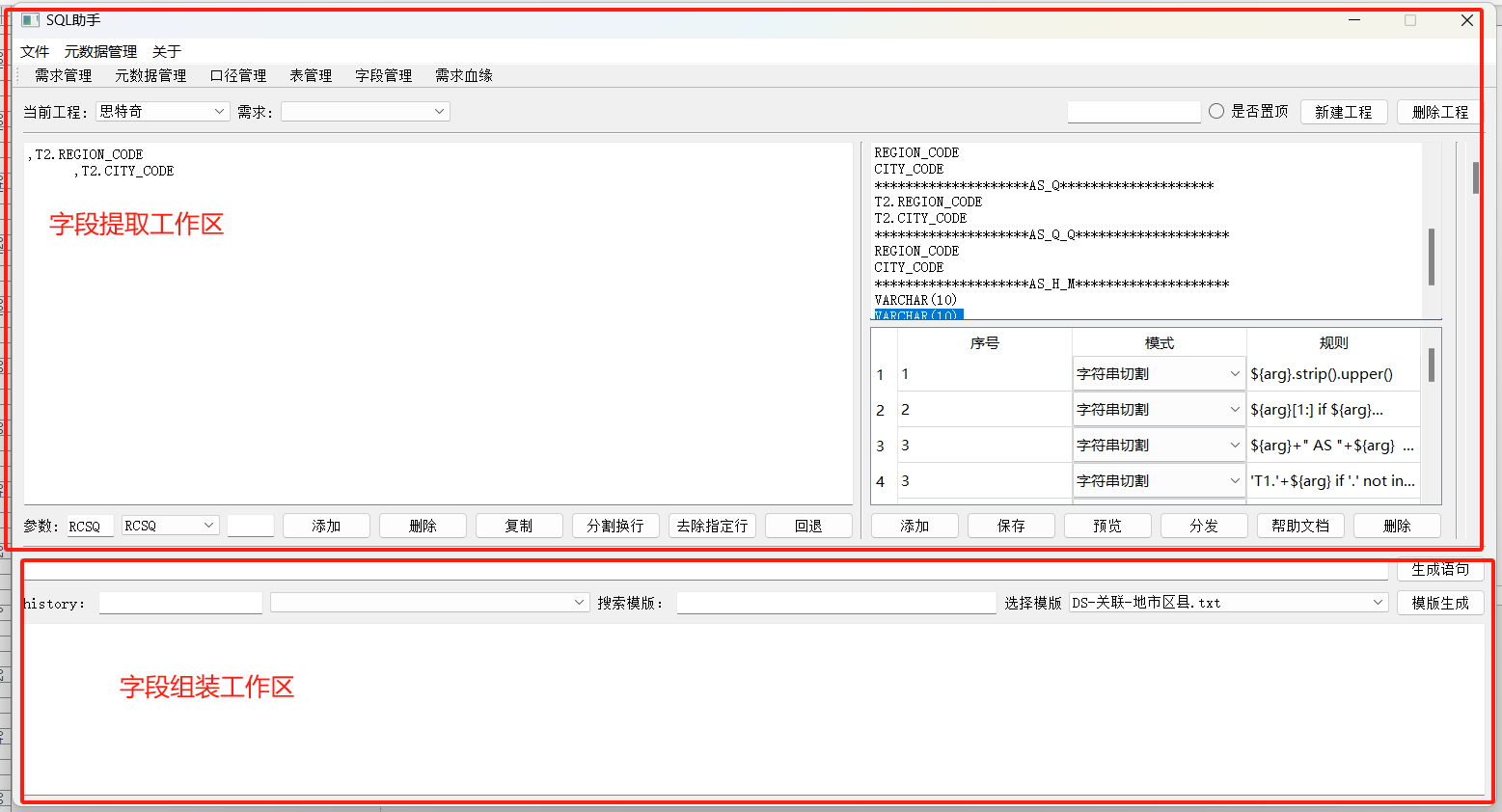


## 模块介绍

### SQL\_HEPPER核心主件

SQL\_HEPPER核心主件是一个批量化的文本生成工具，可以根据用户提供的原始文本，基于自定义规则，一键生成目标文本；可以分为字段提取阶段和组装阶段两个环节，具体工作流程图如下：





#### 2.1.1字段提取阶段

字段提取阶段：根据用户提供的原始文本与编辑的文本处理规则，分别从原始文本的每行提取出目标字段，这里将每个变量提取的逻辑称为一个变量。

##### 变量

**变量预处理：**

* 分隔换行：根据指定分隔符，把原始文本分为多行
* 去除指定行：保留原始文本中包含指定字符串的行

**变量加工：**

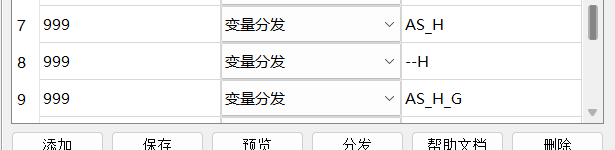
一个变量由1个或多个算子组成，算子可以叠加使用，下游算子的输入是上游算子的输出，根据序号排序算子的上下游关系，序号小的先执行。

目前支持的算子如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **算子名称** | **算子功能介绍** | **算子写法** | **算子特性** |
| 常规 | 不进行任何操作 | 啥也不用写 | 1进1出 |
| 正则表达式 | 以正则表达式方式从文本中提取目标文本 | 例子： reg=\'(.\*?)\';index=1 实现了从文本中提取’’中间文本的功能。  reg后跟正则表达式，index跟索引号，reg和index之间以;分隔 | 1进1出 |
| 字符串切割 | 通过简单的python语法，完成对目标字段的提取 | 例子1： ${arg}.split('--')[1].strip() 实现了从文本中提取 --后字符的功能  ${arg} 是算子中一个的内置变量，可以把他当成python中的一个str对象，python中str对象支持的方法，${arg}都支持（有特定需求可以网上直接搜,如python实现文本大小写转换等，把特定变量换成${arg}就可以直接用）。  例子2： ${arg}[1:] if ${arg}[0]==',' else ${arg} 实现如果第一个字符是‘,’则删除该‘,’，否则保持原样功能，字符串切割也支持if写法，可以个性化的根据每行文本的具体内容，写特定的语法。 | 1进1出 |
| 轮询 | 根据传入参数，扩充目标文本行数 | 例子：传入一个整数，可以是正数也可以是负数，分别代表对原始文本不同的扩充方法 | 1进n出(n是传入的整数) |
| 去除换行 | 将原始文本由多行转化为一行，可以指定分隔符 | 例子：, 将原始文本转化为1行，通过‘,’分隔行。(多用于SQL的IN写法中美化格式) | 多进1出 |
| GROUP BY | 根据原始文本行数，生成Gbase Group by写法中的1,2,3,4,5,6 | 无 | 多进1出 |
| 行分隔 | 根据参数选择原始文本中的特定行 | 例子： n 输出原始文本中第n+1行（从0开始算）  -n 输出原始文本中倒数第n行  n:m 输出原始文本中底n+1~第m行的数据（可以省略n或m,如:m） | x进y出,y<x |
| 元数据 | 给定表和字段，查询字段元数据信息 | 例子：  TB:PMID.TB\_MID\_PAR\_USER\_DAY K1  TB:PDATA.TB\_PDT\_SERV\_ATTR\_INFO\_DAY K2  TB:PCDE.TB\_CDE\_SERV\_ATTR\_CODE T1  K1.USER\_ID  K1.CUST\_ID  K1.ACCT\_ID  K1.PHONE\_NO  K1.MAIN\_SERV\_ID  K2.SVC\_ID  从特定表中查询特定字段的类型和注释，输出  K1|USER\_ID|DECIMAL(14,0)|用户ID  K1|CUST\_ID|DECIMAL(14,0)|客户ID  K1|ACCT\_ID|DECIMAL(14,0)|帐户ID  K1|PHONE\_NO|VARCHAR(15)|手机号码  K1|MAIN\_SERV\_ID|VARCHAR(20)|主体服务ID  K2|SVC\_ID|VARCHAR(40)|服务标识 | 1进1出 |
| 自定义函数 | 传入一个python自定义函数，按行处理原始文本 | 例子：  def fun1(a):  if a.endswith('\_DATE'):  return 'DATE'  if a.endswith('\_PAY'):  return 'DECIMAL(18,4)'  if a.endswith('\_FEE'):  return 'DECIMAL(18,4)'  if a.endswith('\_CODE'):  return 'VARCHAR(10)'  if a.endswith('\_RATE'):  return 'DECIMAL(5,4)'  if a.endswith('\_NAME'):  return 'VARCHAR(500)'  return 'VARCHAR(50)'  实现：根据字段名称，一键生成字段类型  函数名称随意，传入参数只能有一个，参数其实就是原始文本每一行的内容,输出参数只能是文本类型 | 1进1出 |
| 超级自定义函数 | 传入一个python自定义函数，直接处理整个原始文本 | 例子：  def fun2(a):  tmp = [f"'{i}'" for i in a]  res = ','.join(tmp)  return [res,]  实现：加引号+换行，场景 A,B,C -> ‘A’,’B’,’C’  函数名称随意，传入参数只能有一个，参数其实是原始文本按行分隔的数组，输出参数只能是数组 | - |

##### 变量分发

算子分发是指将一个变量处理的结果，分发给其他变量去处理，避免同一个文本被贴来贴去。变量分发写法如下，传入参数是要分发给的变量名称。

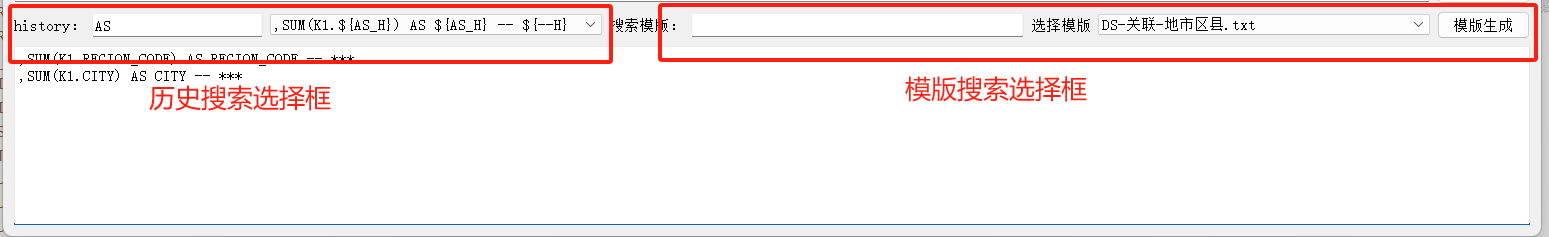


#### 字段组装阶段

字段组装阶段是指将字段提取阶段生成的变量，按照特定的格式组装起来，写法如下：

,**SUM**(K1.${AS\_H}) AS ${AS\_H} *-- ${--H}*

其中${} 包含的都是变量，其他的所见即所得。字段组装支持复用历史写过的拼装语句，通过历史搜索选择框实现；也支持模版功能（模版主要是实现多段拼装语句），通过模版搜索选择框实现。



模版文件已txt格式存储在特定位置，想要新加模版，在特定位置创建txt文件就行，点一下‘帮助文档’按钮，会自动把模版文件加载进来，模版文件写法如下（例子）：

CREATE TABLE **STQ\_REPORT**.${JBYJ\_5}(

    ,${JBYJ\_O\_1} ${JBYJ\_O\_2}

)

;

COMMENT ON TABLE   STQ\_REPORT.${JBYJ\_5} IS  '${JBYJ\_9}';

COMMENT ON COLUMN  STQ\_REPORT.${JBYJ\_5}.${JBYJ\_O\_1} IS '${JBYJ\_8}';

*---------------*

DELETE FROM STQ\_REPORT.${JBYJ\_5} WHERE DEAL\_DATE=DATE '${$}${<}TX\_DATE\_FORMAT${>}';

DELETE FROM STQ\_REPORT.${JBYJ\_5} WHERE DEAL\_DATE= '${$}${<}TX\_MONTH${>}';

ALTER TABLE STQ\_REPORT.${JBYJ\_5} ENABLE ROW MOVEMENT;

ALTER TABLE STQ\_REPORT.${JBYJ\_5} SHRINK SPACE;

### 需求管理

需求管理工具辅助管理需求开发过程中的全部代码和文档，其中代码支持版本管理，并提供高效的检索功能，主要界面如下：



#### 快速检索功能

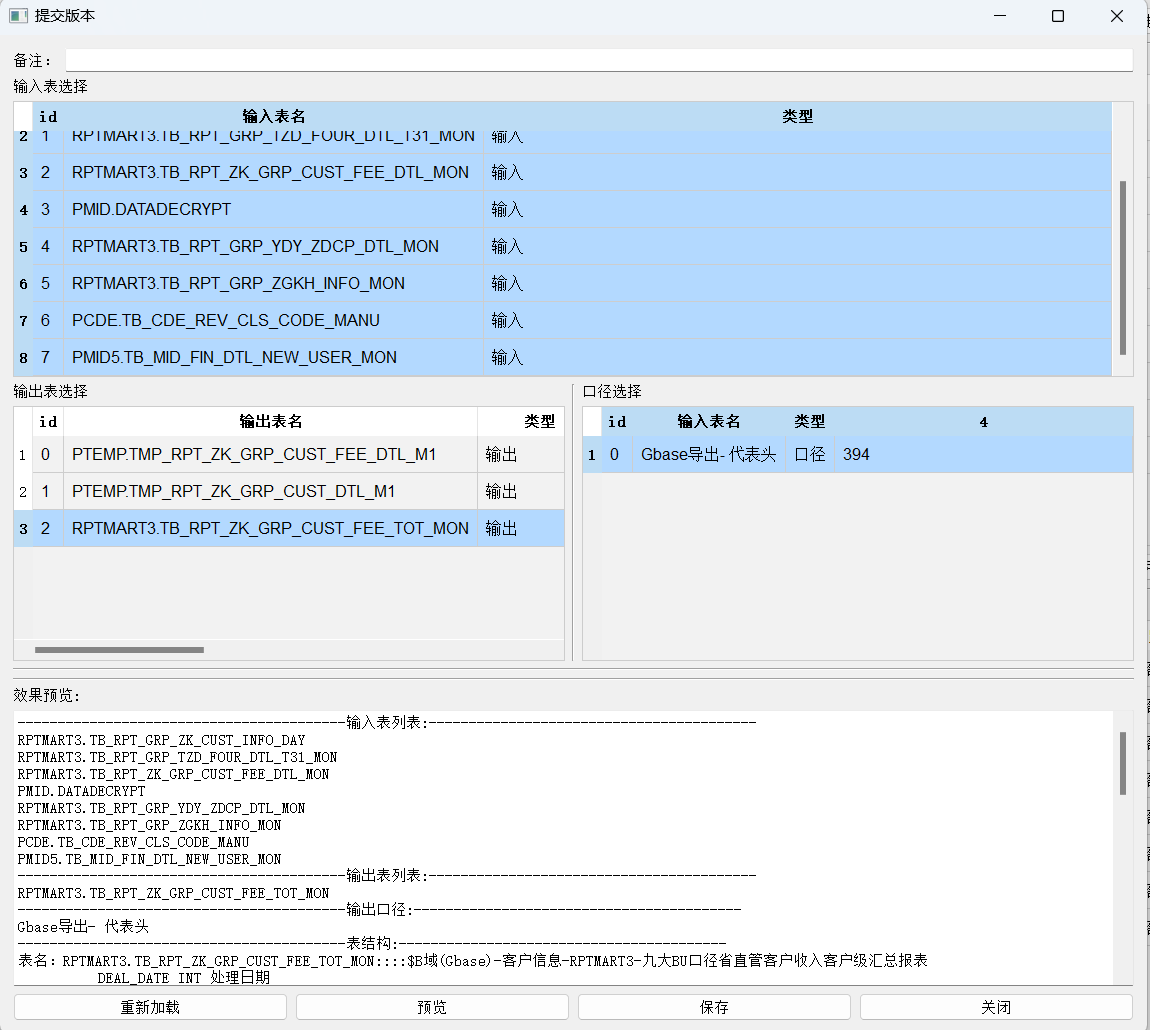
需求、文档、代码支持“与”、“或”检索，例子如下（以检索需求为例，文档代码同）：

工信部|BU|欠费&物联网

搜索需求名称包含“工信部”，或者包含“BU”，或者包含“物联网”且包含“欠费”的需求。

#### 版本管理/版本提交

版本管理仅支持对SQL代码的版本管理，选择SQL代码文件后点击“提交版本”按钮，弹出提交版本界面，点击保存后，可以在通过点击“显示历史版本”查看保存的历史版本（历史版本为只读模式，不可编辑）。提交版本不仅可以实现对代码版本的管理，也同步完成了对代码血缘、需求血缘、口径等元数据的录入。



#### 超级检索

超级检索实现了在多个需求之间根据关键字对代码的搜索（正则表达式），多用于一下想不起SQL在哪个需求里写的了的场景。点击需求管理界面上的“超级检索”，输入关键字完成搜索。

### 元数据管理

#### 表管理

表管理中导入了目前GBASE中常用库的大部分表结构信息，支持根据表名（英文）、表描述、表备注（结合自己开发实际做的笔记）对表进行检索，可以便捷的查询到表是哪个需求产生的，哪些需求中用到了这个表。



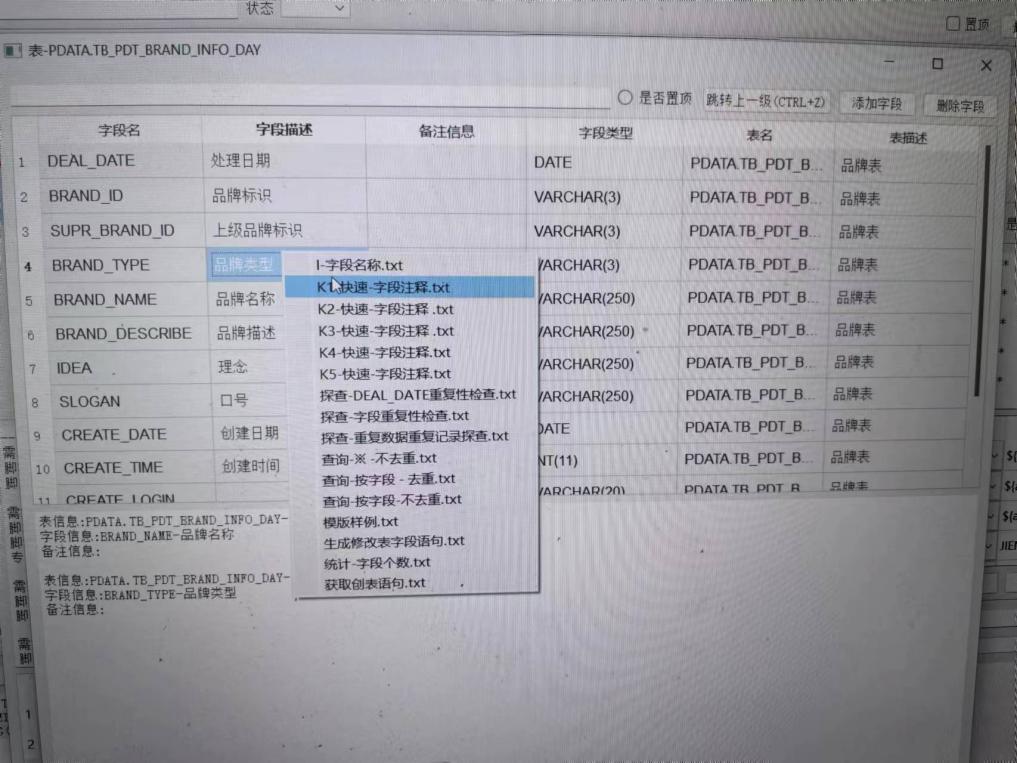
#### 字段管理

字段管理展示了表的元数据信息，支持根据自己理解对每个字段添加备注。



##### 快速模版

字段管理支持基于元数据的快速代码生成，选择待选字段，右键，选择对应模版（模版支持自定义写法），会在下方展示界面生成对应的语句（并将内容直接放在剪切板中，直接Ctrl+V就可以贴在自己代码中），效果如下：





**自定义模版编辑**

自定义模版是txt格式，放在指定位置，写好模版后直接放在特定位置，便可在字段管理中右键使用。

自定义模版例子：

-- 关键字介绍

-- ZDMC:选择的字段名称

-- BM：字段对应表名

-- TX\_DATE：业务日期 YYYYMMDD

-- TX\_DATE\_FORMAT：业务日期 YYYY-MM-DD

-- TX\_MONTH:业务日期YYYYMM

-- SELECT\_ZDMC : 选择了的字段名称,逗号分隔

-- SELECT\_ZDMC\_RANGE : 根据选择字段个数返回1,2,3,4,5，一般用于GROUP BY

-- SELECT\_ZDMC\_MS : 选择了的字段名+注释,TN.CN AS CN -- 注释

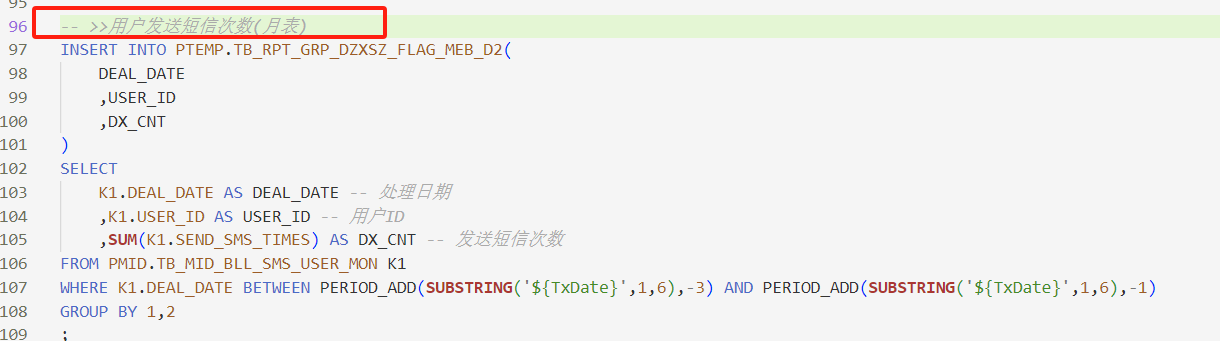
K5.${SELECT\_ZDMC\_MS}

#### 口径管理

口径管理用于记录开发过程中的可能会复用的口径，通过简单的标记后，可以在后续开发中十分快捷的查询到。

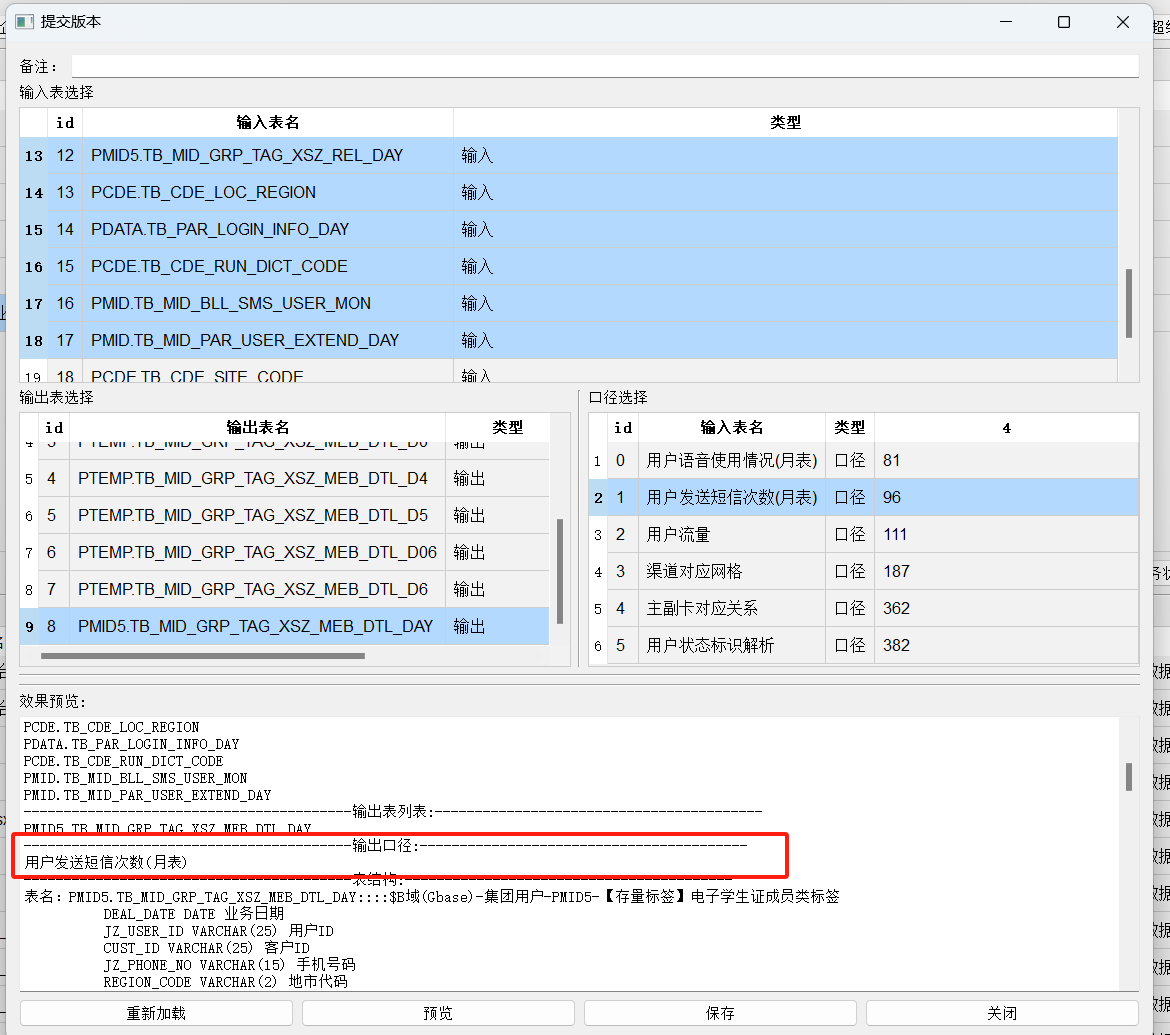
##### 口径管理标记方法

如图，通过“-- >>”在代码中进行标记



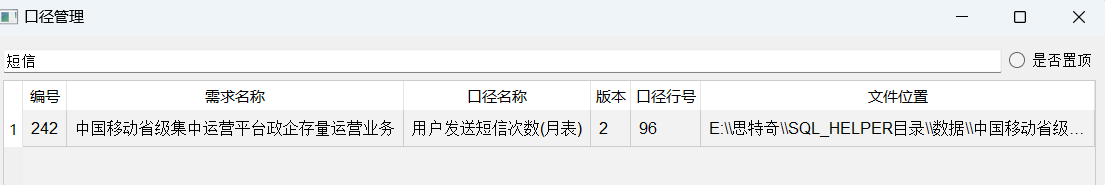
##### 口径管理提交方法

在需求管理提交版本的时候，提交口径。



##### 口径管理查询方法

在口径管理中，输入关键字查询口径。



#### 需求血缘

需求血缘用于记录该需求的对应SQL代码用了哪些表，产出了哪些表。



## 安装配置

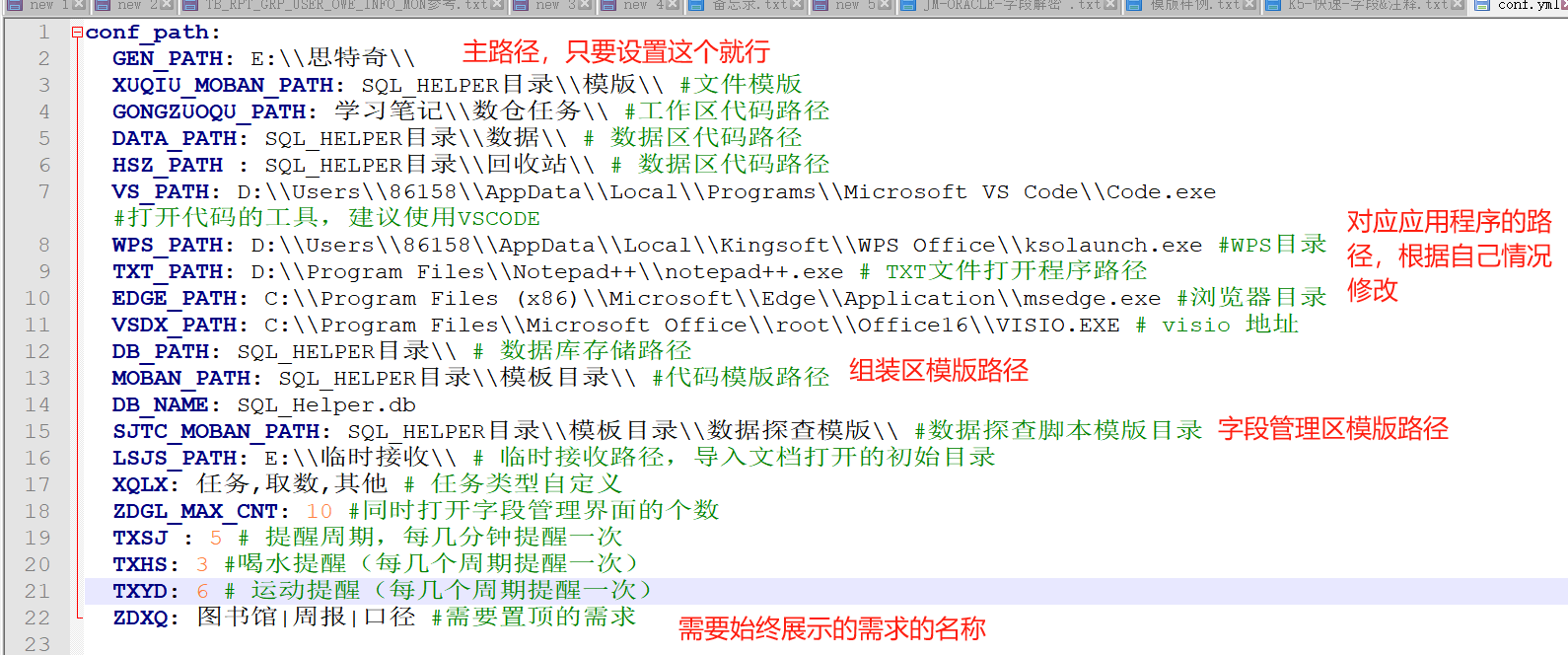
**步骤一：解压发布版SQL\_助手.zip,解压后会有三个文件**



* SQL\_HELLPER\_R.exe ：应用程序
* conf.yml ： 配置文件
* 初始化文件：程序运行所需的模版、数据库信息

**步骤二：将“SQL\_HELLPER\_R.exe”和配置文件“conf.yml”放在同一个目录下**

修改配置文件，根据自己需要，修改配置文件的路径。



**应用配置与打开文件的对应关系：**

|  |  |
| --- | --- |
| **配置路径** | **打开的文件** |
| WPS\_PATH | 打开文档文件，包括word,excel,ppt |
| TXT\_PATH | 后缀是txt的文件 |
| EDGE\_PATH | pdf文件 |
| VSDX\_PATH | visio文件 |
| VS\_PATH | SQL文件 |

注：推荐使用VSCOD和WPS这些都做了适配，其他的可以修改成对应路径试一下，看可不可以。

程序界面

