1 开发规范

1.1 数据开发规范

1.1.1 hive 数据目录规范

1.1.1.1 表分区键命名

月分区键： month

日分区键： day

创建分区键的时候分区键不能和表字段名相同。

1.1.1.2 hdfs 存放位置

表数据文件在 hdfs 存放位置： /user/hive/warehouse/ 模式名 .db/ 表名 / 分区名 /[子分区名] 。

示例：

/user/hive/warehouse/ods.db|dwd.db|dwa.db|dwf.db/dwa_s_d_use_mb_voice_comm_d/month=201405/day=31

1.1.1.3 字段分隔符

字段分割符： 表的字段分割符默认使用 '\001' ，接口层表的字段分割符仍采用源文件中的字段分割符。

空值处理：为了保证导出数据的正确性，空值统一用 SERDEPROPERTIES('serialization.null.format' = '') 进行处理。

1.1.1.4 分区表结构

创建分区表分为 2 种，一种是单分区， 也就是说在表文件夹目录下只有一级文件夹目录。另外一种是多分区，表文件夹下出现多文件夹嵌套模式。

示例：

单分区表

```
create table dwa_v_d_wap_mb_url_label
( month_id   string          COMMENT '账期月 ',
day_id       string          COMMENT '账期日 ',
device_number string         COMMENT '手机号码  ',
prov_id      string          COMMENT '省份代码  ',
url_1        string          COMMENT 'URL标签 ',
url_2        string          COMMENT 'URL标签 ',
url_3        string          COMMENT 'URL标签 '
)
partitioned by  (month string)
row format delimited fields terminated by '\001' ;
```

多分区表

```
create table dwa_v_d_wap_mb_url_label
( month_id   string          COMMENT '账期月 ',
day_id       string          COMMENT '账期日 ',
device_number string         COMMENT '手机号码  ',
prov_id      string          COMMENT '省份代码  ',
url_1        string          COMMENT 'URL标签 ' ,
url_2        string          COMMENT 'URL标签 ',
url_3        string          COMMENT 'URL标签 '
)
partitioned by  (month string,day string)
row format delimited fields terminated by '\001' ;
```

1.1.2 hbase数据目录规范

1.1.2.1 hdfs 存放位置

表数据文件在 hdfs 存放位置： /hbase / 表名

示例：

/hbase/test

1.1.2.2 表结构

Hbase 使用表作为顶级结构来存储数据，写数据到 hbase,就是写数据到表。一般我们创建的表只有一个列族。

示例：

Create 'table','cf'

创建表 table，列族为 cf

1.2 脚本开发规范

1.2.1 脚本目录规范

1.2.1.1 目录规范说明

1. 开发用户

开发用户为 Hadoop 开发人员所使用的用户，具备自身开发脚本的新建、修改、删除、执行。

为了保障 Hadoop 系统安全，避免误操作删除 Hadoop 系统文件，建议创建开发用户，与 Hadoop 管理用户在 Hadoop 系统操作权限上进行多租户隔离。

开发用户的创建， 参考 《大数据技术实施规范——系统运维规范》 开发用户的设置。

2. 应用目录说明

按照数据源区分应用数据目录。

序号	命名	说明
1	gn	Gn 上网行为分析

2	bi	经分数据仓库迁移
3	sg	信令数据（ Signaling ）

应用子目录说明：每个应用目录下包含 target 、src、sh 三个子目录，子目录详细说明如下：

1) 导出文件目录 ：target

2) 导入文件目录： src

3) shell 脚本目录： sh

3. shell 脚本目录说明

1) shell 脚本目录： 存放通用 shell 脚本，通用脚本调用 hive shell 加工脚本。

/home/[hadev]/app/[gn,bi,s g,…]/sh

2) 加工脚本目录 ：存放 shell 脚本，例如 Hive Shell 加工脚本

/home/[hadev]/app/[gn,bi,s g,…]/sh/sql

3) 日志文件目录 ：存放 Shell 脚本执行过程日志、报错信息、 MapReduce输出信息等，便于 Job 调试、错误捕捉、执行情况监控。

/home/[hadev]/app/[gn,bi,s g,…]/sh/log

4) 配置文件目录 ：存放脚本中调用的环境变量，例如目录配置、 ORACLE连接相关变量等。

/home/[hadev]/app/[gn,bi,s g,…]/sh/ param

4. 加工脚本目录说明：按照数据模型规范，进行加工脚本目录划分。例如：

/home/[hadev]/app/[gn,bi,s g,…]/sh/sql/[ods,dwd,dwa,dwf, …]/，ods、dwd、dwa、dwf 分别代表数据模型规范中不同的层次。

1.2.1.2 目录结构体系

```
/home/

|-------hadev/            ## 开发用户

        |-------app/          ## 应用开发目录
```

```
|-------gn/            ## 应用数据库目录,      GN  表示  GN  详单数据库

|-------bi/            ## 应用数据库目录,      bi  表示经分库

        |-------target /          ## 导出文件加工目录

        |-------src /             ## 导入文件加工目录

    |-------sh/          ## 应用  shell  脚本执行目录

        |-------sql/         ##hive-SQL     脚本文件目录，通用脚本存放目录

                |-------ods/          ods  层加工脚本目录

                |-------dwd/          dwd   层加工脚本目录

                |-------dwa/          dwa   层加工脚本目录

                |-------dwf/          dwf   层加工脚本目录

        |-------log/         ## 日志文件目录

            |-------param/           ## 参数文件目录
```

1.2.2 脚本文件命名规范

文件名规范：数据库类型　_文件名命名参考数据模型规范中表实体命名。

文件扩展名规范：扩展名为　　sh。

文件名命名举例：

hi_ods_d_imei_al_gprs.sh　　，表示 ods 层日表 imei_al_gprs　 的 Hive 处理脚本。

数据库类型说明：

序号	数据库类型	对应数据库
1	hi	Hive
2	hb	HBase
3	st	Strom
4	or	Oracle

5	pi	Pig
6	gp	GreenPlum
7	my	MySql
8	mr	MapReduce
9	sl	Spark-sql
10	ss	Spark-streaming

1.2.3 脚本运行、调试、编辑

1）脚本运行环境

在 liunx　操作系统上运行，通过命令方式调用脚本。

2）调用脚本和终止脚本

可以用多种方式调用脚本如下：

Sh 脚本名

./ 脚本名

Source　脚本名

/xxxx/xxx/　脚本名　-----　脚本的全路径

终止脚本

通过 ps 命令查询到正在执行脚本的进程，用　kill　命令杀掉执行中的脚本。

ps　– ef|grep　脚本名；

kill　脚本的进程

3）编辑工具

使用 linux　系统自带的　vi 编辑，　UE等。

1.2.4 脚本注释说明

脚本中需要有基本的注释信息方便以后维护，　如参数、执行示例、　脚本存放位置、　日志文件地址、创建人、创建日期等信息。

脚本注释示例：

```
##############################################################################
#说明：通用  hive 过程调用  shell。调用  HQL 文件，判断前置过程、添加日志
#参数  1        说明：过程名称        示例： UI.P_UI_L_USER_USE_M
#参数  2        说明：日 / 月账期       格式： YYYYMMDD/YYYYMM     示例： 201404
#执行示例：  sh /home/hadoop/shell/runtask.sh src.p_src_m_dev_user_all       20140531
#脚本存放地址：  xxxxxxxxxxxxxxxxxxxxxxx
#日志文件地址：  xxxxxxxxxxxxxxxxxxxxxxx
#创建人：  xxxxxxxxx
#创建日期：  xxxxx
#最新修改日期：  xxxxxxxxx
#修改人、修改时间、修改内容：      xxxxxxxxxxxxxxxxx
#修改人、修改时间、修改内容：      xxxxxxxxxxxxxxxxx
##############################################################################
```

1.2.5 脚本变量、参数命名规范

1）脚本变量命名

变量名： v_开头

过程名： p_开头

2）参数命名

账期参数： v_day_id（YYYYMMDD）

#输入参数账期

v_day_id=$1

v_month=`echo $1 | cut -c 1-6` ---------YYYYMM

v_day=`echo $1 | cut -c 7-8   ----------DD

1.2.6 脚本环境变量配置文件规范

配置文件是脚本中用到的环境变量, oracle 连接 ,公共脚本地址等都放到配置文件用方便以后管理。

1） 配置文件存放路径及其命名

配置文件路径及命名： /home/[hadev]/app/[gn,bi]/sh/ param/hdp_shell.config

2） 配置文件中示例内容如下

PUBLIC_PATH:/home/hadev/app/bi/sh/param/public/

DAY_LOG_PATH:/home/[hadev]/app/[gn,bi]/sh/log/daily/logs/

MON_LOG_PATH: /home/[hadev]/app/[gn,bi]/sh/log/monthly/logs/

SHELL_LANGUAGE:American_America.ZHS16GBK

ORA_CLIENT_PATH:/u01/oracle/app/

ORA_LINK:hdp/hdp131@ora131

REPLACE_VAR:VVV_ACCT_DATE

3） 配置文件编写规范

配置文件中以 冒号 做分割符，第一列是变量名需要大写。

1.2.7 脚本执行前置条件配置表规范

脚本执行前置条件判断配置表放在 oracle 数据库表中（可根据实际情况选用其他数据库存储，例如 MySQL）。

配置表结构如下：

```
create table PROCEDURE_INFO
(
  user_name VARCHAR2(20),
  procname  VARCHAR2(100),
  proc_desc VARCHAR2(200),
  condition VARCHAR2(4000),
  expected  VARCHAR2(10),
  flag      VARCHAR2(1)
);
  -- Add comments to the columns
comment on column PROCEDURE_INFO.user_name
```

```
 is '数据库用户名   ';
comment on column PROCEDURE_INFO.procname
 is '过程名称  ';
comment on column PROCEDURE_INFO.proc_desc
 is '过程说明  ';
comment on column PROCEDURE_INFO.condition
 is '过程前置条件    sql';
comment on column PROCEDURE_INFO.expected
 is '过程前置判断数量    ';
comment on column PROCEDURE_INFO.flag
 is 'D：日过程     M：月过程  ';
```

配置表示例内容，如下图：

```
select * from src.PROCEDURE_INFO t
```

	USER_NAME	PROCNAME	PROC_DESC	CONDITION	EXPECTED	FLAG
1	VAAS	P_MS_INFO_USER_INFO_FLOW_M	用户流量统计	SELECT COUNT(*) from SRC.T_EXECPROC_HIVE_LOG T	3	M
2	SRC	P_DWA_M_ADSL_INFO_DAY	宽带用户信息表	SELECT COUNT(DISTINCT T.PROCNAME) FROM SRC.T_EXECPROC_HIVE_LO	8	D
3	SRC	P_DWA_M_NEW_USER_DAY_ZB	新增用户清单	SELECT COUNT(DISTINCT T.PROCNAME) FROM SRC.T_EXECPROC_HIVE_LO	1	D
4	SRC	P_DWA_M_USER_PACKAGE_D	用户订购附属包明细	SELECT COUNT(DISTINCT T.PROCNAME) FROM SRC.T_EXECPROC_HIVE_LO	2	D
5	SRC	P_DWA_V_D_USE_MB_USER	用户资料日表	SELECT COUNT(DISTINCT T.PROCNAME) FROM SRC.T_EXECPROC_HIVE_LO	2	D
6	SRC	P_DWD_O_B_TRADE_DAY	受理渠道汇总	SELECT COUNT(DISTINCT T.PROCNAME) FROM SRC.T_EXECPROC_HIVE_LO	1	D
7	SRC	P_FU_T_USER_MMS_D	用户使用彩信日表	SELECT COUNT(DISTINCT T.PROCNAME) FROM SRC.T_EXECPROC_HIVE_LO	2	D
8	SRC	P_FU_T_USER_SALE	合约到期日表	SELECT COUNT(DISTINCT T.PROCNAME) FROM (select * fr	2	D
9	SRC	P_FU_T_USER_SMS_D	用户使用短信日表	SELECT COUNT(DISTINCT T.PROCNAME) from SRC.T_EXECPROC_HIVE_LO	2	D
10	VAAS	P_MS_SUM_REGION_USER	覆盖用户数报表	SELECT COUNT(DISTINCT T.PROCNAME) from SRC.T_EXECPROC_HIVE_LO	7	D

前置条件判断脚本

```
#####根据过程名称获得配置信息
v_sqlset="set echo off \n set head off \n set pagesize 0 \n set linesize 2000 \n set heading off \n
set trimspool on \n set feedback off \n set term off \n"
#####过程前置条件
v_sqlstr=`echo  -e  $v_sqlset"SELECTREPLACE(t.Condition,'$v_replace_var',   "'$v_acct_day"')   AS
Condition FROM src.Procedure_Info t WHERE UPPER(t.Procname) = UPPER('$v_procname') AND
UPPER(t.User_Name) = UPPER('$v_user');"|sqlplus -s $ORA_HBDW`;
###过程前置判断数量
v_expected=`echo  -e  $v_sqlset"SELECT  t.Expected  FROM  src.Procedure_Info  t  WHERE
UPPER(t.Procname)  =  UPPER('$v_procname')  AND  UPPER(t.User_Name)  =
UPPER('$v_user');"|sqlplus -s $ORA_HBDW`;
```

```
###执行前置语句并获得结果

v_actual=`echo -e $v_sqlset "$v_sqlstr;"|sqlplus -s $ORA_HBDW`

echo "v_sqlstr:"$v_sqlstr

echo "v_expected:"$v_expected

echo "v_actual:"$v_actual



#####前置判断

if [ $v_expected == $v_actual ]; then

调用  HIVE sql脚本

fi
```

1.2.8 Shell开发说明

Shell 开发包括公用 shell 脚本和 sql shell 脚本两部分。公用 shell 脚本采用统一模版，用于处理前置条件判断、日志记录操作等，sql shell 脚本由开发人员自己开发，负责处理独立的业务逻辑和数据加工。

脚本书写说明如下：

1）shell 解释器：以“ #! /bin/bash ”开头，指明使用哪种解释器。

2）设置字符集处理，支持中文

```
LANG=zh_CN.UTF-8

export NLS_LANG=american_america.AL32UTF8
```

3）设置 MapReduce 会话内存占用大小，默认为 2GB 大小。

```
set mapreduce.map.memory.mb=2048;

set mapreduce.reduce.memory.mb=4096;
```

4）打开执行用户

```
use dwd;
```

5）hive -e 语句规范

```
hive -e ''

##需要执行的 SQL

quit;
```

```
" 2>&1 |tee -a ${v_logfile} >>/dev/null
```

1.3 Hive 开发规范

1.3.1 日志表结构

日志表用于记录过程调用记录和执行结果，在 oracle 数据库中创建日志表。

日志表结构如下：

```
create table T_EXECPROC_HIVE_LOG
(
  logdate        VARCHAR2(10),
  pkg_name       VARCHAR2(100),
  procname       VARCHAR2(100),
  svc_id         VARCHAR2(100),
  begin_date     DATE,
  end_date       DATE,
  flag           VARCHAR2(200),
  duration       NUMBER,
  rowcount       NUMBER,
  note           VARCHAR2(4000),
  table_name     VARCHAR2(100),
  classname      VARCHAR2(100),
  funcname       VARCHAR2(100),
  custommsg      VARCHAR2(1024),
  exceptionmsg   VARCHAR2(1024),
  exceptioncause VARCHAR2(1024)
);
  -- Add comments to the columns
comment on column T_EXECPROC_HIVE_LOG.logdate
  is '账期日期  ';
comment on column T_EXECPROC_HIVE_LOG.pkg_name
  is '数据库用户  ';
comment on column T_EXECPROC_HIVE_LOG.procname
  is '过程名称  ';
comment on column T_EXECPROC_HIVE_LOG.begin_date
  is '过程执行开始时间    ';
comment on column T_EXECPROC_HIVE_LOG.end_date
```

```
  is '过程执行结束时间    ';
comment on column T_EXECPROC_HIVE_LOG.flag
  is '过程执行结果   ';
comment on column T_EXECPROC_HIVE_LOG.duration
  is '执行时长（秒）    ';
comment on column T_EXECPROC_HIVE_LOG.rowcount
  is '条数 ';
comment on column T_EXECPROC_HIVE_LOG.note
  is '备注 ';
```

T_EXECPROC_HIVE_LOG_HISTORY 是备份日志表，与日志表表结构完全相同。

1.3.2 日志表过程

Hive 数据库日志表及其存储过程在 oracle 数据库上创建，通过 shell 来调用存储过程。

1） 插入日志存储过程

```
CREATE OR REPLACE PROCEDURE P_INSERT_HIVE_LOG(V_LOGDATE VARCHAR2,
                                              V_PKG VARCHAR2,
                                              V_PROCNAME VARCHAR2,
                                              V_TELE_TYPE VARCHAR2,
                                              V_DATE DATE,
                                              V_TABLE_NAME VARCHAR2 DEFAULT '')
IS
    /*@
    **************************************************************
    *名称  --%@NAME:P_INSERT_HIVE_LOG
    *功能描述  --%@COMMENT:在 HIVE上的过程  插入日志过程
    *执行周期  --%@PERIOD日/月
    *参数  --%@PARAM:P_NGATHER_DATE日期,格式 YYYYMMDD
    *参数  --%@PARAM:P_NOUT_STATUS 过程运行结束成功与否标志
    *创建人  --%@CREATOR: xxxxxx
    *创建时间  --%@CREATED_TIME:xxxxxxxxx
    *备注  --%@REMARK插入日志过程
```

```
        * 来源表    --%@FROM:

        * 目标表    --%@TO:T_EXECPROC_LOG

        ************************************************************

        @*/


BEGIN


        --备份到历史表中

        INSERT INTO T_EXECPROC_HIVE_LOG_HISTORY

                SELECT T.*, SYSDATE

                  FROM T_EXECPROC_HIVE_LOG t

                 WHERE LOGDATE = V_LOGDATE

                   AND PKG_NAME = UPPER(V_PKG)

                   AND PROCNAME = UPPER(V_PROCNAME);

        COMMIT;

        DELETE T_EXECPROC_HIVE_LOG

          WHERE LOGDATE = V_LOGDATE

            AND PKG_NAME = UPPER(V_PKG)

            AND PROCNAME = UPPER(V_PROCNAME);

        COMMIT;

        INSERT INTO T_EXECPROC_HIVE_LOG

        VALUES

                (V_LOGDATE,

                  UPPER(V_PKG),

                  UPPER(V_PROCNAME),

                 V_TELE_TYPE,

                 V_DATE,

                 '',

                 'RUNNING',
```

```
                '',
                '',
                '',
                UPPER(V_TABLE_NAME),
                '',
                '',
                '',
                '');
        COMMIT;
END;
```

2） 更新日志存储过程

```
CREATE OR REPLACE PROCEDURE P_UPDATE_HIVE_LOG(V_ACCT_MONTH VARCHAR2,
                                              V_PKG          VARCHAR2,
                                              V_PROCNAME     VARCHAR2,
                                              V_RETINFO      VARCHAR2,
                                              V_RETCODE      VARCHAR2,
                                              V_DATE         DATE,
                                              V_COUNT        NUMBER) IS
/*@
**************************************************************
*名称   --%@NAME:P_UPDATE_HIVE_LOG
*功能描述   --%@COMMENT: 在 hive 上跑的过程，更新日志过程
*执行周期   --%@PERIOD日/月
*参数   --%@PARAM:P_NGATHER_DATE日期,格式  YYYYMMDD
*参数   --%@PARAM:P_NOUT_STATUS过程运行结束成功与否标志
*创建人   --%@CREATOR: xxxxx
*创建时间   --%@CREATED_TIME:xxxxxx
```

```
* 备注   --%@REMARK插入日志过程

* 来源表   --%@FROM:

* 目标表   --%@TO:T_EXECPROC_LOG

**********************************************************************

@*/

V_TIME          VARCHAR2(500);

    BEGIN_DATE DATE;
BEGIN

    DBMS_OUTPUT.PUT_LINE(BEGIN_DATE);

    /*END_DATE := SYSDATE;*/

    SELECT TO_CHAR(CEIL(3600 * 24 * TO_NUMBER(SYSDATE - BEGIN_DATE)))

      INTO V_TIME

      FROM T_EXECPROC_HIVE_LOG

     WHERE LOGDATE = V_ACCT_MONTH

       AND PKG_NAME = UPPER(V_PKG)

    AND PROCNAME = UPPER(V_PROCNAME);


UPDATE T_EXECPROC_HIVE_LOG

    SET END_DATE = V_DATE,

        DURATION = V_TIME,

        NOTE         = V_RETINFO,

        FLAG         = V_RETCODE,

        ROWCOUNT = V_COUNT

  WHERE LOGDATE = V_ACCT_MONTH

      AND PKG_NAME = UPPER(V_PKG)

      AND PROCNAME = UPPER(V_PROCNAME);

    COMMIT;

END;
```

1.3.3 日志过程调用脚本

通过 shell 来调用 hive 日志的过程，日志脚本为插入日志脚本和跟新日志脚本

1） 日志脚本的命名

插入日志脚本命名： p_pub_insert_log.sh

更新日志脚本命名： p_pub_update_log.sh

2） 插入日志脚本示例

```
#!/bin/bash


export NLS_LANG=American_America.ZHS16GBK
export ORACLE_HOME=/oracle/product/OraHome
CONNECT_ORA="SRC/SRC123456@HBDW"


v_Acct_Date=$1
v_Pkg=$2
v_Procname=$3
v_Retinfo=$4
v_Retcode=$5
v_Rowline=$6


v_Sql="
EXECUTEDECLAREBEGIN src.P_INSERT_HIVE_LOG('$v_Acct_Date','$v_Pkg',  '$v_Procname','',
SYSDATE,''); END;"


echo $v_Sql


echo "$v_Sql"|sqlplus -s $CONNECT_ORA;
```

更新脚本示例

```
#!/bin/bash


export NLS_LANG=American_America.ZHS16GBK
export ORACLE_HOME=/oracle/product/OraHome
CONNECT_ORA="SRC/SRC123456@HBDW"


v_Acct_Date=$1
```

```
v_Pkg=$2
v_Procname=$3
v_Retinfo=$4
v_Retcode=$5
v_Rowline=$6



v_Sql="EXECUTE DECLARE BEGIN src.P_UPDATE_HIVE_LOG('$v_Acct_Date', '$v_Pkg',
'$v_Procname', '$v_Retinfo', '$v_Retcode', SYSDATE, $v_Rowline); END;"

echo $v_Sql

echo "$v_Sql"|sqlplus -s $CONNECT_ORA;
```

1.3.4 Hive Shell脚本开发规范

1） Hive 数据加工公用脚本， exec_task_main.sh

公用脚本传 2 个参数：过程名和账期。

```
#! /bin/bash
#############################################################################
#说明：通用 hive 过程调用 shell。调用 HQL 文件，判断前置过程、添加日志
#参数 1 说明：过程名称 示例： UI.P_UI_L_USER_USE_M
#参数 2 说明：日 /月账期 格式： YYYYMMDD/YYYYMM 示例： 201404
#执行示例： sh /home/hadoop/shell/runtask.sh src.p_src_m_dev_user_all 20140531
#脚本存放地址： /src/marketing/daily/p_ui_l_user_gp_flow_info.sh
#日志文件地址： /src/marketing/daily/logs/p_ui_l_user_gp_flow_info_xxxxxxxx.log
#创建人： xxxx
#创建日期： xxx
#最新修改日期： xxxxx
#修改人、修改时间、修改内容： xxxx
#修改人、修改时间、修改内容： xxxxxx
#############################################################################
```

```
#检查参数个数
if [ "$#" -ne "2" ]; then
        echo "Please check parameter: $0 <v_user_procname> <v_acct_day>      "
        exit 1
fi



#输入参数账期
export v_acct_day=$2
export v_month=`echo $2 | cut -c 1-6` #   本年月
export v_day=`echo $2 | cut -c 7-8`        #本日
export v_year=`echo $2 | cut -c 1-4`      #本年


v_mid_day1=$v_month"01"
export v_last_one_mon=`date      -d "-1 month $v_mid_day1" +%Y%m` #  上月
export v_last_two_mon=`date       -d "-2 month $v_mid_day1" +%Y%m` # 上上月
export v_last_three_mon=`date      -d "-3 month $v_mid_day1" +%Y%m` #  上上上月



#echo "v_last_one_mon:"$v_last_one_mon
#echo "v_last_two_mon:"$v_last_two_mon
#echo "v_last_three_mon:"$v_last_three_mon



v_month_2=`echo $2 | cut -c 5-6`
export v_last_day=$v_month`cal $v_month_2 $v_year |xargs|awk '{print $NF}'` #      该月最后一天


len=`expr length $v_acct_day`
```

```
if [ $len -eq 8 ];then
        export v_yest_date=`date      -d "-1 day $v_acct_day" +%Y%m%d` #昨日
        export v_yest_mon=`echo $v_yest_date | cut -c 1-6`       #昨日  年月
        export v_yest_day=`echo $v_yest_date | cut -c 7-8`      #昨日   日
fi


v_mid_day2=$v_month"15"
export v_next_mon=`date       -d "1 month $v_mid_day2" +%Y%m` # 下月

v_cur_day=`date -d yesterday +%Y%m%d` #取当前日账期     有些月过程需要用到最新的日数据



#echo "本月最后一天  :"$v_last_day
#echo "昨日 :"$v_yest_date


#过程名
v_user_procname=$1

#用户名
typeset -u v_user
v_user=`echo $v_user_procname |cut -d '.' -f1` #   取得库名

#过程名称
typeset -u v_procname
v_procname=`echo $v_user_procname |cut -d '.' -f2` #   删除点号左边的，取到过程名


echo "v_user:"$v_user
echo "v_procname:"$v_procname
```

```
#设置环境变量
export NLS_LANG=`awk -F: '{if($1 =="SHELL_LANGUAGE") print $2}'
/home/hadoop/shell/hdp_shell.config`
export ORACLE_HOME=`awk -F: '{if($1 =="ORA_CLIENT_PATH") print $2}'
/home/hadoop/shell/hdp_shell.config`

#ORACLE的连接
ORA_HBDW=`awk -F: '{if($1 =="ORA_LINK") print $2}' /home/hadoop/shell/hdp_shell.config`

#前置语句中应该替换的变量
v_replace_var=`awk -F: '{if($1 =="REPLACE_VAR") print $2}'
/home/hadoop/shell/hdp_shell.config`

#公共脚本地址
v_shell_path=`awk -F: '{if($1 =="PUBLIC_PATH") print $2}' /home/hadoop/shell/hdp_shell.config`

#日、月日志文件地址
v_day_log_path=`awk -F: '{if($1 =="DAY_LOG_PATH") print $2}'
/home/hadoop/shell/hdp_shell.config`
v_mon_log_path=`awk -F: '{if($1 =="MON_LOG_PATH") print $2}'
/home/hadoop/shell/hdp_shell.config`

#记录日志
v_pkg=$v_user
cd $v_shell_path
#echo "./p_pub_insert_log.sh $v_acct_day $v_pkg $v_procname"
```

```
echo `./p_pub_insert_log.sh $v_acct_day $v_pkg $v_procname` >>/dev/null



#日志文件

  v_log="       cat /home/hadoop/shell/log/runtask_${1}_${2}.log"



#根据过程名称获得配置信息

v_sqlset="set echo off \n set head off \n set pagesize 0 \n set linesize 2000 \n set heading off \n

set trimspool on \n set feedback off \n set term off \n"

#v_pkg=`echo -e $v_sqlset"SELECT t.User_Name FROM src.Procedure_Info t WHERE

UPPER(t.Procname) = UPPER('$v_procname') AND UPPER(t.User_Name) =

UPPER('$v_user');"|sqlplus -s $ORA_HBDW`;

v_sqlstr=`echo -e $v_sqlset"SELECT REPLACE(t.Condition, '$v_replace_var', "'$v_acct_day'") AS

Condition FROM src.Procedure_Info t WHERE UPPER(t.Procname) = UPPER('$v_procname') AND

UPPER(t.User_Name) = UPPER('$v_user');"|sqlplus -s $ORA_HBDW`;

v_expected=`echo -e $v_sqlset"SELECT t.Expected FROM src.Procedure_Info t WHERE

UPPER(t.Procname) = UPPER('$v_procname') AND UPPER(t.User_Name) =

UPPER('$v_user');"|sqlplus -s $ORA_HBDW`;



if    [ -z $v_expected    ]

then

       v_retcode="FAIL"

       v_retinfo="$v_user_procname not in src.Procedure_Info."

       v_rowline=0

       echo "v_retinfo"$v_retinfo

       cd $v_shell_path

       echo `./p_pub_update_log.sh $v_acct_day $v_pkg $v_procname """"$v_retinfo         $v_log""""

$v_retcode $v_rowline` >>/dev/null
```

```
    exit 1
fi



#执行前置语句并获得结果
v_actual=`echo -e $v_sqlset "$v_sqlstr;"|sqlplus -s $ORA_HBDW`



echo "v_sqlstr:"$v_sqlstr
echo "v_expected:"$v_expected
echo "v_actual:"$v_actual



#根据账期长度进入不同的日志文件目录
v_length=`expr length $v_acct_day`
if [ $v_length -eq 8 ]; then
    v_path=$v_day_log_path
    v_filename=$v_user"_"$v_procname"_"$v_acct_day".log"
else
    v_path=$v_mon_log_path
    v_filename=$v_user"_"$v_procname"_"$v_acct_day".log"
fi
v_logfile=$v_path$v_filename
echo "v_logfile:"$v_logfile



#创建日志文件
cd $v_shell_path
echo `./p_pub_logfile.sh $v_path $v_filename` >>/dev/null
#echo "******LOG FILE:"$v_logfile
```

```
#echo "********ACTUAL:"$v_expected ":" $v_actual



typeset -l v_user_procname_low

v_user_procname_low=$v_user_procname



if [ -f $v_shell_path/sql/${v_user_procname_low}.sh ]

then

        echo '文件存在 '

else

        echo "$v_shell_path/sql/${v_user_procname_low}.sh      文件不存在  "

        v_retcode="FAIL"

        v_retinfo='sql file not found!'

        v_rowline=0

        echo `./p_pub_update_log.sh $v_acct_day $v_pkg $v_procname """"$v_retinfo          $v_log""""
$v_retcode $v_rowline` >>/dev/null

        exit 1

fi



#前置判断

if [ $v_expected == $v_actual ]; then



   ################################### HIVE sql 脚本  ###################################

   #执行的  sh 脚本模式为       echo "       select * from tablename where month=$v_month          "



    v_sql=`sh ./sql/$v_user_procname_low'.sh'`
```

```
        echo "GOD start $(date     +%Y%m%d_%T)

        ********************************************************************************
****

        $v_sql

        "



echo "RUN....     $v_user_procname_low.sh"


hive -e "
    use $v_user;
    $v_sql
    " 2>&1 |tee $v_logfile >>/dev/null


        echo "GOD stop $(date     +%Y%m%d_%T)

        ********************************************************************************
****

        "



   #获取  sql 执行结果信息

   v_result=`cat $v_logfile | grep -s "FAILED" | awk -F ":" '{print $1}'` >>/dev/null


    echo "********RESULT:"$v_result


#如果  sql 执行成功，执行结果状态置为       SUCCESS 并从执行日志中获取执行记录数

if     [ "$v_result" != "FAILED" ]; then

    v_retcode="SUCCESS"
```

```
    v_retinfo="SUCCESS"


#如果  sql 执行失败，执行结果状态置为       FAIL，并获取执行日志中的失败信息
else
    v_retcode="FAIL"
    v_retinfo=`cat $v_logfile | grep -s "FAILED"` >>/dev/null
        #v_retinfo=${v_retinfo//\'/\"}
        #v_retinfo=${v_retinfo// /|}
    fi
else
    v_retcode="WAIT"
    v_retinfo="WAIT"
    echo 'WAIT'
fi


#更新日志
cd $v_shell_path
echo "./p_pub_update_log.sh $v_acct_day $v_pkg $v_procname $v_retinfo $v_retcode 0"
echo `./p_pub_update_log.sh $v_acct_day $v_pkg $v_procname """$v_retinfo          $v_log"""
$v_retcode 0` >>/dev/null
```

2） hive-sql　加工 shell　脚本

Hive-sql 加工脚本中只包含业务逻辑处理　　sql 语句，方便开发人员编写脚本。

此例文件名：　p_ui_l_user_info_m

```
#/bin/bash
echo " INSERT OVERWRITE TABLE UI.UI_L_USER_INFO_M partition
                        (month_part = '$v_month')
SELECT '$v_month',
                    T1.AREA_NO,
```

```
T1.USER_ID,
T1.DEVICE_NUMBER,
T1.CUST_ID,
CASE
   WHEN T1.AGE >=6 AND T1.AGE <=10 THEN'1' --儿童
   WHEN T1.AGE >=11 AND T1.AGE <= 18 THEN'2' --少年
   WHEN T1.AGE >=19 AND T1.AGE <= 25 THEN'3' --青年
   WHEN T1.AGE >=26 AND T1.AGE <= 40 THEN'4' --成年
   WHEN T1.AGE >=41 AND T1.AGE <= 59 THEN '5'中年
   WHEN T1.AGE >=60 THEN '6' -老年
      ELSE '0'
END AGE, -年龄分档
NVL(T1.SEX, 0) SEX,性别
CASE
   WHEN T1.INNET_LENGTH = 1 THEN
    '1'
WHEN T1.INNET_LENGTH = 2 THEN
    '2'
WHEN T1.INNET_LENGTH = 3 THEN
    '3'
WHEN T1.INNET_LENGTH >= 4 AND T1.INNET_LENGTH <= 6 THEN
    '4'
WHEN T1.INNET_LENGTH >= 7 AND T1.INNET_LENGTH <= 12 THEN
    '5'
   WHEN T1.INNET_LENGTH > 12 THEN
     '6'
   ELSE
     '0'
END INNET_TIME, -入网时长
```

```
CASE
   WHEN T1.INNET_METHOD = '1' THEN
    '1' --自备机入网
   WHEN T1.INNET_METHOD = '2' THEN
    '2' --租机入网
   WHEN T1.INNET_METHOD = '3' THEN
    '3' --赠送手机入网
   ELSE
    '0' --其他方式
END INNET_TYPE, 入网方式
NVL(T4.CHNL_KIND_ID, '0') INNET_CHANNEL,发展渠道
CASE
   WHEN T1.CUST_TYPE = '30' THEN
    '1' --大客户
   WHEN T1.CUST_TYPE = '20' THEN
    '2' --商务客户
   WHEN T1.CUST_TYPE = '50' THEN
    '3' --校园客户
   WHEN T1.CUST_TYPE = '40' THEN
    '4' --公众客户
   ELSE
    '0' --其他集团客户
END CUST_GRADE,客户级别
CASE
   WHEN T1.VIP_TYPE = '0' THEN
    '1' --钻石卡
   WHEN T1.VIP_TYPE = '1' THEN
    '2' --金卡
   WHEN T1.VIP_TYPE = '2' THEN
```

```
    '3' --银卡
  WHEN T1.VIP_TYPE = '4' THEN
    '4' --至尊卡
  WHEN T1.VIP_TYPE = '9' THEN
    '5' --未发卡
  ELSE
      '0' --非ＶＩＰ会员
  END MEMBER_GRADE, 会员等级
  NVL(C1.LAN_ID, '0') LAN_ID, --区域标识
  C1.AREA_ID_DESC_PROV AREA_ID_DESC,区域描述
  T2.CITY_DESC CITY_ID_DESC,区县描述
  NVL(T3.CHANNEL_TYPE_NAME,'其它渠道  ') CHANNEL_DESC, --入网渠道描述
  NVL(T4.CHANNEL_TYPE_NAME,'其它渠道  ') OPER_CHANNEL_DESC,发展渠道描述
  T1.OPER_NO, --发展人
  T1.GENT_TAG, --业务标识
  T5.BLC_FEE 帐户余额
FROM (SELECT
    *
      FROM SRC.DWA_V_M_USE_MB_USER T
    WHERE T.month_part = '$v_month'
      ) T1
  left join     DIM.DIM_CITY_NO t2 on T1.CITY_ID = T2.CITY_NO
  left join
  DIM.DIM_CHANNEL_TYPE T3 on  T1.CHANNEL_TYPE = T3.CHANNEL_TYPE --入网渠道
  left join
  (SELECT T.CHANNEL_NO, TT.CHANNEL_TYPE_NAME, TT.CHNL_KIND_ID
      FROM DIM.DIM_CHANNEL_NO T
```

```
                        join DIM.DIM_CHANNEL_TYPE TT on T.CHANNEL_TYPE =
TT.CHANNEL_TYPE) T4 on T1.OPER_CHANNEL_ID = T4.CHANNEL_NO(发展渠道
                        left join
                        (SELECT
                            USER_ID, SUM(BLC_FEE) BLC_FEE
                             FROM SRC.DWA_V_M_ACC_MB_ACCT T
                            WHERE T.month_part = '$v_month'
                            GROUP BY USER_ID) T5 on T1.USER_ID = T5.USER_ID
                       join (SELECT * FROM DIM.DIM_AREA_NO WHERE AREA_NO <> '018') c1 on
t1.AREA_NO=c1.AREA_NO;"
```

3） shell　执行参考语句：

```
# exec_task_main.sh +  过程文件名    +  账期
sh exec_task_main.sh       p_ui_l_user_info_m 201408
# exec_task_main.sh +  过程文件名    +  账期
sh exec_task_main.sh       p_ui_l_user_info_m 20140801
```

1.3.5 Hive UDF函数开发及规范

1.3.5.1 开发步骤

UDF简称自定义函数，它是　Hive 函数库的扩展，自定义函数　UDF在 MapReduce执行阶段发挥作用。开发步骤如下：

1） 给 hive.ql.exec.UDF　包开发一个自定义函数类，从　UDF继承。自定义函数类实现 evaluate　方法。

2） 在 FunctionRegistry　类中注册开发的自定义函数类。

3） 打包发布至　Hive 客户端。

1.3.5.2 开发工具

Eclipse 是一款开源的、基于 Java的可扩展开发平台。 Hadoop 开发人员可通过在 Eclipse 上面开发 UDF。

1.3.5.3 UDF 函数案例

1）开发 UDF函数类

文件名及路径：

/hive-0.12.0/src/ql/src/java/org/apache/hadoop/hive/ql/udf/UDFHelloWorld.java

```
package org.apache.hadoop.hive.ql.udf;

import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;

public class UDFHelloWorld extends UDF {
    public String evaluate(String str) {
        if (str == null ) {
            return null ;
        }
        return "HelloWorld " + str;
    }
    public static void main(String[] args) {

        helloUDF uf = new helloUDF();
            //Text t = new Text("gfsg");
            System. out .println(uf.evaluate( "nihao" ).toString());
        }

}
```

2） UDF类注册，注册方法

文件名及路径：

/hive-0.12.0/src/ql/src/java/org/apache/hadoop/hive/ql/exec/FunctionRegistry.ja

va

```
package org.apache.hadoop.hive.ql.exec;

import org.apache.hadoop.hive.ql.udf.UDFHelloWorld;
```

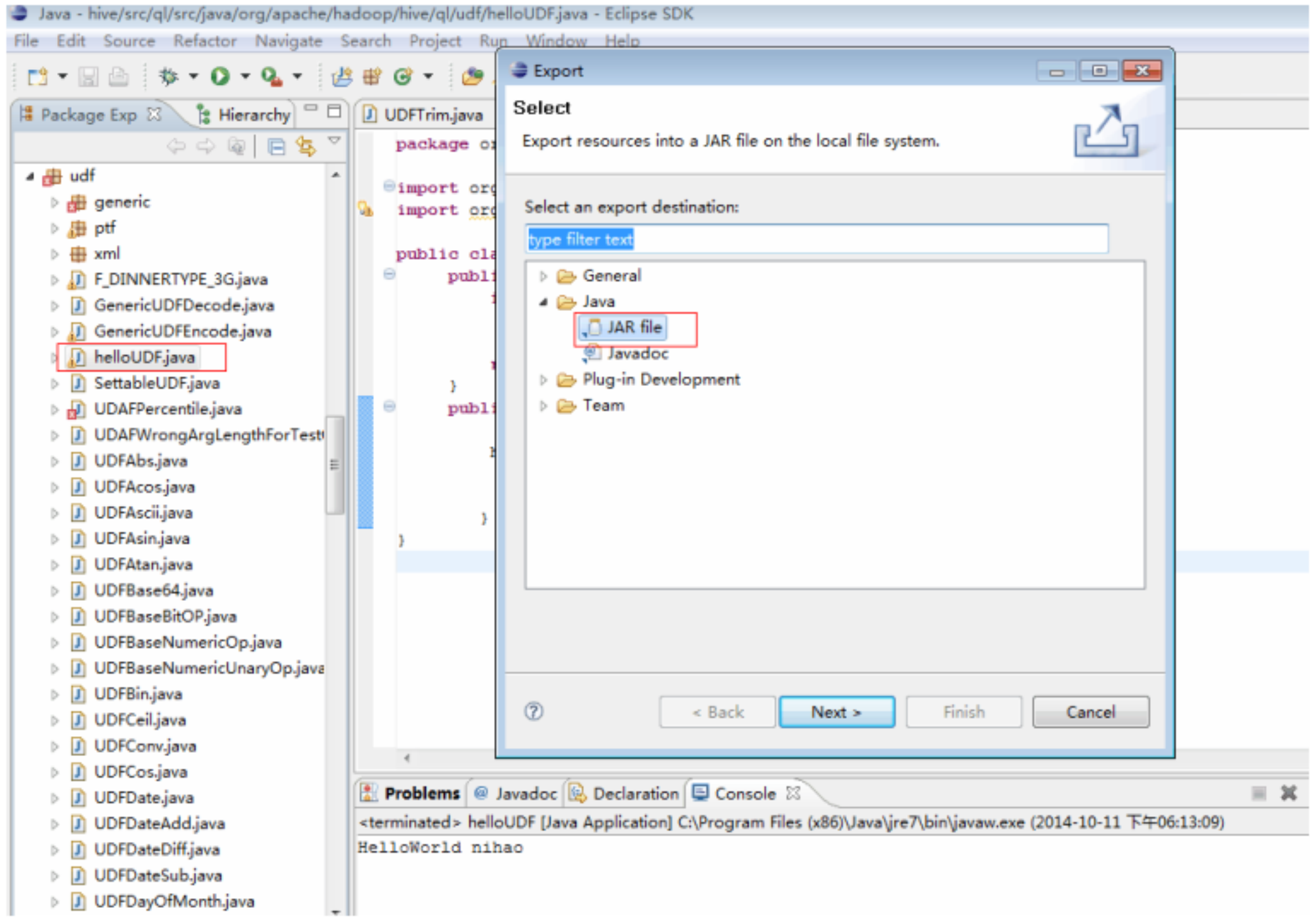
```
/**
 * FunctionRegistry.
 */
public final class FunctionRegistry {
    static {
        registerGenericUDF("concat", GenericUDFConcat.class);
registerUDF("substr", UDFSubstr.class, false);
registerUDF("substring", UDFSubstr.class, false);
registerUDF("space", UDFSpace.class, false);
registerUDF("repeat", UDFRepeat.class, false);
registerUDF("ascii", UDFAscii.class, false);
registerUDF("lpad", UDFLpad.class, false);
registerUDF("rpad", UDFRpad.class, false);
registerUDF("Hello", UDFHelloWorld.class, false);

        registerGenericUDF("size", GenericUDFSize.class);
    ……
    ……
```

3）Jar 包发布路径

发布路径： /opt/boh/hive/lib/hive-exec-0.12.0-cdh5.0.0.jar

上传至 hadoop 集群执行脚本的 hive 客户端。



1.3.5.4 Hive UDF 函数

1.3.5.4.1 UDF函数列表

函数清单及其功能

TO_DATE(string date,'format')	格式化所需要的日期
ADD_MONTHS(Timestamp date,int n)	增加月数
date_tostring(Timestamp date,'format')	转换 Date 类型为指定格式字符串
MONTHS_BETWEEN（Timestamp date1 ， Timestamp date2 ）	返回两个日期之间的月数

f_age(string identityId)	验证身份证合法性并返回性别年龄
f_checkidcard(string identityId)	验证身份证合法性

1.3.5.4.2 UDF函数说明

TO_DATE函数

Select to_date('20140909111111','YYYYMMDDHH24miss') from test;

返回结果： 2014-09-09 11:11:11

ADD_MONTHS函数

select add_months(to_date('20140909111111','YYYYMMDDHH24miss'),1) from test;

返回结果： 2014-10-09 11:11:11

date_tostring 函数

select

date_tostring(to_date('20140909111111','YYYYMMDDHH24miss'),'YYYY-MM-DD') from test;

返回结果： 2014-09-09

MONTHS_BETWEEN函数

select

MONTHS_BETWEEN(to_date('20140909111111','YYYYMMDDHH24miss'),to_date('20140706111111','YYYYMMDDHH24miss')) from test;

返回结果： 2.096774193548387

f_age 函数

select f_age('511024198710148199') from test;

返回结果： 127

f_checkidcard 函数

select f_checkidcard('511024198710148199') from test;

返回结果： 1