# 开发规范

## 数据开发规范

### hive数据目录规范

#### 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。

#### 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

#### 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。

#### 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' ;

### hbase数据目录规范

#### hdfs存放位置

表数据文件在hdfs存放位置：/hbase /表名

示例：

/hbase/test

#### 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

## 脚本开发规范

### 脚本目录规范

#### 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。

1. 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
4. shell脚本目录说明
5. 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,sg,…]/sh

1. 加工脚本目录:存放shell脚本，例如Hive Shell加工脚本

/home/[hadev]/app/[gn,bi,sg,…]/sh/sql

1. 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,sg,…]/sh/log

1. 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,sg,…]/sh/param

1. 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,sg,…]/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。

#### 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

### 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 |

### 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。

### 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：xxxxxxxxxxxxxxxxxxxxxxxx

#日志文件地址：xxxxxxxxxxxxxxxxxxxxxxxx

#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

###################################################################

### 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

### 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

1. 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

1. 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。

### 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：



* 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 $v\_sqlset"SELECT REPLACE(t.Condition, '$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`;

###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

### 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=2048;

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

## Hive开发规范

### 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。

### 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;

1. 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;

### 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

1. 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="

EXECUTE DECLARE BEGIN 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;

### 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

### Hive UDF函数开发及规范

#### 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。

#### 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。

#### 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.java

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);

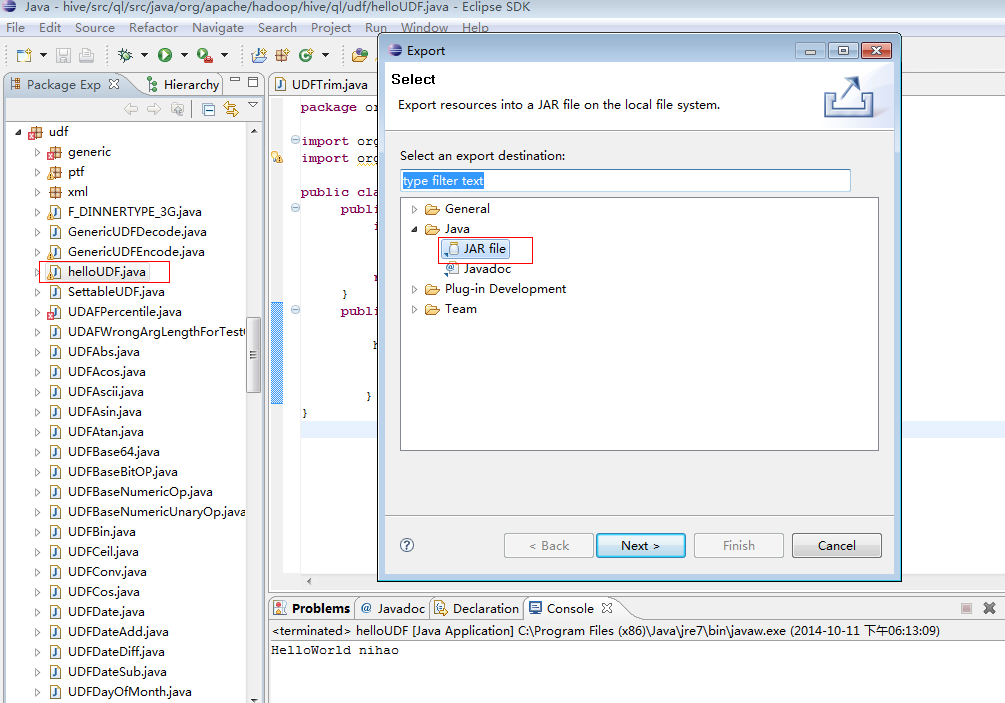
……………………………………………………

……………………………………………………

**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。



#### Hive UDF函数

##### 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 |

##### 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