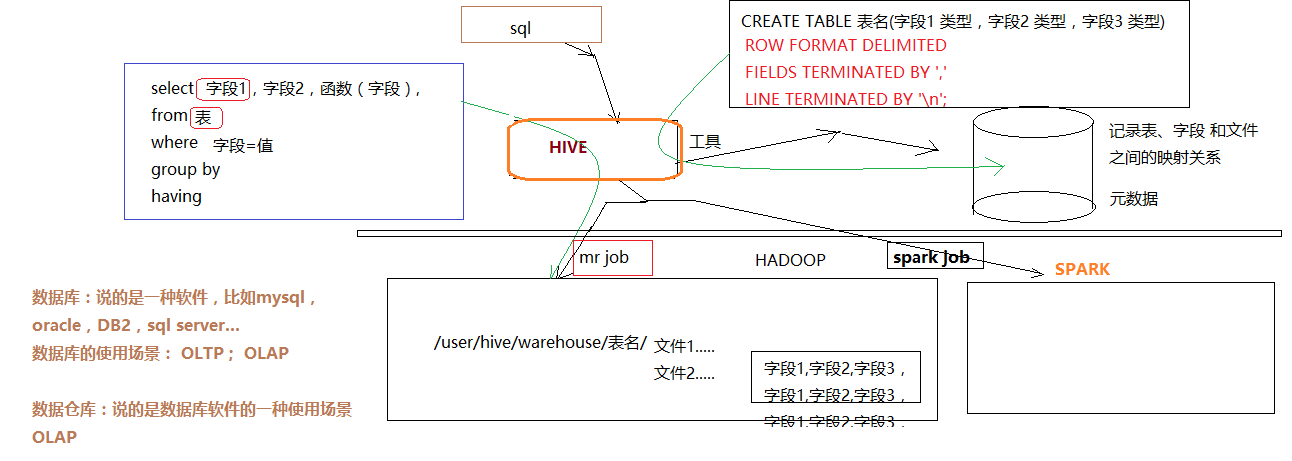
# 什么是hive

## hive基本思想

Hive是基于Hadoop的一个数据仓库工具(离线)，可以将结构化的数据文件映射为一张数据库表，并提供类SQL查询功能。



## 为什么使用Hive

* 直接使用hadoop所面临的问题

人员学习成本太高

项目周期要求太短

MapReduce实现复杂查询逻辑开发难度太大

* 为什么要使用Hive

操作接口采用类SQL语法，提供快速开发的能力。

避免了去写MapReduce，减少开发人员的学习成本。

功能扩展很方便。

## Hive的特点

* 可扩展

Hive可以自由的扩展集群的规模，一般情况下不需要重启服务。

* 延展性

Hive支持用户自定义函数，用户可以根据自己的需求来实现自己的函数。

* 容错

良好的容错性，节点出现问题SQL仍可完成执行。

# hive的基本架构



Jobtracker是hadoop1.x中的组件，它的功能相当于：

Resourcemanager+MRAppMaster

TaskTracker 相当于：

Nodemanager + yarnchild

# hive安装

|  |  |
| --- | --- |
| 知  识  补  充 | 补充：先将hadoop集群的机器配置时间同步  时间同步  yum install ntpdate -y ## 安装时间同步客户端  ntpdate 0.asia.pool.ntp.org 　　## 与互联网时间服务器同步  若上面的时间服务器不可用，也可以选择以下服务器同步时间  time.nist.gov  time.nuri.net  0.asia.pool.ntp.org  1.asia.pool.ntp.org  2.asia.pool.ntp.org  3.asia.pool.ntp.org |

## 最简安装：用内嵌derby作为元数据库

准备工作：安装hive的机器上应该有HADOOP环境（安装目录，HADOOP\_HOME环境变量）

安装：直接解压一个hive安装包即可

此时，安装的这个hive实例使用其内嵌的derby数据库作为记录元数据的数据库

此模式不便于让团队成员之间共享协作

## 标准安装：将mysql作为元数据库

### mysql安装

1. 上传mysql安装包
2. 解压：

[root@mylove ~]# tar -xvf MySQL-5.6.26-1.linux\_glibc2.5.x86\_64.rpm-bundle.tar

1. 安装mysql的server包

[root@mylove ~]# rpm -ivh MySQL-server-5.6.26-1.linux\_glibc2.5.x86\_64.rpm

依赖报错：

缺perl

yum install perl

安装完perl后 ，继续重新安装mysql-server

（可以配置一个本地yum源进行安装：

1、先在vmware中给这台虚拟机连接一个光盘镜像

2、挂在光驱到一个指定目录：mount -t iso9660 -o loop /dev/cdrom /mnt/cdrom

3、将yum的配置文件中baseURL指向/mnt/cdrom

）

[root@mylove ~]# rpm -ivh MySQL-server-5.6.26-1.linux\_glibc2.5.x86\_64.rpm

又出错：包冲突conflict with

移除老版本的冲突包：mysql-libs-5.1.73-3.el6\_5.x86\_64

[root@mylove ~]# rpm -e mysql-libs-5.1.73-3.el6\_5.x86\_64 --nodeps

继续重新安装mysql-server

[root@mylove ~]# rpm -ivh MySQL-server-5.6.26-1.linux\_glibc2.5.x86\_64.rpm

成功后，注意提示：里面有初始密码及如何改密码的信息

初始密码：/root/.mysql\_secret

改密码脚本：/usr/bin/mysql\_secure\_installation

1. 安装mysql的客户端包：

[root@mylove ~]# rpm -ivh MySQL-client-5.6.26-1.linux\_glibc2.5.x86\_64.rpm

1. 启动mysql的服务端：

[root@mylove ~]# service mysql start

Starting MySQL. SUCCESS!

1. 修改root的初始密码：

[root@mylove ~]# /usr/bin/mysql\_secure\_installation 按提示

1. 测试：

用mysql命令行客户端登陆mysql服务器看能否成功

[root@mylove ~]# mysql -uroot -proot

mysql> show databases;

1. 给root用户授予从任何机器上登陆mysql服务器的权限：

mysql> grant all privileges on \*.\* to 'root'@'%' identified by '你的密码' with grant option;

Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;

Query OK, 0 rows affected (0.00 sec)

注意点：要让mysql可以远程登录访问

最直接测试方法：从windows上用Navicat去连接，能连，则可以，不能连，则要去mysql的机器上用命令行客户端进行授权：

在mysql的机器上,启动命令行客户端：

mysql -uroot -proot

mysql>grant all privileges on \*.\* to 'root'@'%' identified by 'root的密码' with grant option;  
mysql>flush privileges;

### hive的元数据库配置

vi conf/hive-site.xml

|  |
| --- |
| <configuration>  <property>  <name>javax.jdo.option.ConnectionURL</name>  <value>jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true</value>  <description>JDBC connect string for a JDBC metastore</description>  </property>  <property>  <name>javax.jdo.option.ConnectionDriverName</name>  <value>com.mysql.jdbc.Driver</value>  <description>Driver class name for a JDBC metastore</description>  </property>  <property>  <name>javax.jdo.option.ConnectionUserName</name>  <value>root</value>  <description>username to use against metastore database</description>  </property>  <property>  <name>javax.jdo.option.ConnectionPassword</name>  <value>root</value>  <description>password to use against metastore database</description>  </property>  </configuration> |

1. 上传一个mysql的驱动jar包到hive的安装目录的lib中
2. 配置HADOOP\_HOME 和HIVE\_HOME到系统环境变量中：/etc/profile
3. source /etc/profile

5、hive启动测试

然后用命令启动hive交互界面：

[root@hdp20-04 ~]# hive

# hive使用方式

## 最基本使用方式

启动一个hive交互shell

bin/hive

hive>

设置一些基本参数，让hive使用起来更便捷，比如：

1、让提示符显示当前库：

hive>set hive.cli.print.current.db=true;

1. 显示查询结果时显示字段名称：

hive>set hive.cli.print.header=true;

但是这样设置只对当前会话有效，重启hive会话后就失效，解决办法：

在linux的当前用户目录中，编辑一个.hiverc文件，将参数写入其中：

vi .hiverc

|  |
| --- |
| set hive.cli.print.header=true;  set hive.cli.print.current.db=true; |

## 启动hive服务使用

启动hive的服务：

[root@hdp20-04 hive-1.2.1]# bin/hiveserver2 ~~-hiveconf hive.root.logger=DEBUG,console~~

上述启动，会将这个服务启动在前台，如果要启动在后台，则命令如下：

*nohup bin/hiveserver2 1>/dev/null 2>&1 &*

启动成功后，可以在别的节点上用beeline去连接

* 方式（1）

[root@hdp20-04 hive-1.2.1]# bin/beeline 回车，进入beeline的命令界面

输入命令连接hiveserver2

beeline> !connect jdbc:hive2//mini1:10000

（hadoop01是hiveserver2所启动的那台主机名，端口默认是10000）

* 方式（2）

启动时直接连接：

bin/beeline -u jdbc:hive2://mini1:10000 -n root

接下来就可以做正常sql查询了

## 脚本化运行

大量的hive查询任务，如果用交互式shell来进行输入的话，显然效率及其低下，因此，生产中更多的是使用脚本化运行机制：

该机制的核心点是：hive可以用一次性命令的方式来执行给定的hql语句

[root@hdp20-04 ~]# hive -e "insert into table t\_dest select \* from t\_src;"

然后，进一步，可以将上述命令写入shell脚本中，以便于脚本化运行hive任务，并控制、调度众多hive任务，示例如下：

vi t\_order\_etl.sh

|  |
| --- |
| #!/bin/bash  hive -e "select \* from db\_order.t\_order"  hive -e "select \* from default.t\_user"  hql="create table default.t\_bash as select \* from db\_order.t\_order"  hive -e "$hql" |

如果要执行的hql语句特别复杂，那么，可以把hql语句写入一个文件：

vi x.hql

|  |
| --- |
| select \* from db\_order.t\_order;  select count(1) from db\_order.t\_user; |

然后，用hive -f /root/x.hql 来执行

# hive建库建表与数据导入(DDL)

data define language

## 建库

hive中有一个默认的库：

库名： default

库目录：hdfs://hdp20-01:9000/user/hive/warehouse

新建库：

create database db\_order;

库建好后，在hdfs中会生成一个库目录：

hdfs://hdp20-01:9000/user/hive/warehouse/db\_order.db

## 建表

### 基本建表语句

use db\_order;

create table t\_order(id string,create\_time string,amount float,uid string);

表建好后，会在所属的库目录中生成一个表目录

/user/hive/warehouse/db\_order.db/t\_order

只是，这样建表的话，hive会认为表数据文件中的字段分隔符为 ^A（\001）

正确的建表语句为：

create table t\_order(id string,create\_time string,amount float,uid string)

row format delimited

fields terminated by ',';

这样就指定了，我们的表数据文件中的字段分隔符为 ","

*注意：hive是不会检查用户导入表中的数据的！如果数据的格式跟表定义的格式不一致，hive也不会做任何处理（能解析就解析，解析不了就是null）；*

### 删除表

drop table t\_order;

删除表的效果是：

hive会从元数据库中清除关于这个表的信息；

hive还会从hdfs中删除这个表的表目录；

### 内部表与外部表

内部表(MANAGED\_TABLE)：表目录按照hive的规范来部署，位于hive的仓库目录/user/hive/warehouse中

外部表(EXTERNAL\_TABLE)：表目录由建表用户自己指定

create external table t\_access(ip string,url string,access\_time string)

row format delimited

fields terminated by ','

location '/access/log';

外部表和内部表的特性差别：

1. 内部表的目录在hive的仓库目录中 VS 外部表的目录由用户指定
2. drop一个内部表时：hive会清除相关元数据，并删除表数据目录
3. drop一个外部表时：hive只会清除相关元数据；

一个hive的数据仓库，最底层的表，一定是来自于外部系统，为了不影响外部系统的工作逻辑，在hive中可建external表来映射这些外部系统产生的数据目录；

然后，后续的etl操作，产生的各种中间表建议用managed\_table（内部表）

### 分区表

**分区表的实质是**：在表目录中为数据文件创建分区子目录，以便于在查询时，MR程序可以针对指定的分区子目录中的数据进行处理，缩减读取数据的范围，提高效率！

比如，网站每天产生的浏览记录，浏览记录应该建一个表来存放，但是，有时候，我们可能只需要对某一天的浏览记录进行分析

这时，就可以将这个表建为分区表，每天的数据导入其中的一个分区；

当然，每日的分区目录，应该有一个目录名（分区字段）

一个分区字段的实例：

示例如下：

1. 创建带分区的表

|  |
| --- |
| create table t\_access(ip string,url string,access\_time string)  partitioned by(dt string)  row format delimited  fields terminated by ','; |

注意：分区字段不能是表定义中的已存在字段

1. 向分区中导入数据

load data local inpath '/root/access.log.2017-08-04.log' into table t\_access partition(dt='20170804');

load data local inpath '/root/access.log.2017-08-05.log' into table t\_access partition(dt='20170805');

1. 针对分区数据进行查询

*a、统计8月4号的总PV：*

*select count(\*) from t\_access where dt='20170804';*

*实质：就是将分区字段当成表字段来用，就可以使用where子句指定分区了*

*b、统计表中所有数据总的PV：*

*select count(\*) from t\_access;*

*实质：不指定分区条件即可*

多个分区字段示例

建表：

create table t\_partition(id int,name string,age int)

partitioned by(department string,sex string,howold int)

row format delimited fields terminated by ',';

导数据：

load data local inpath '/root/p1.dat' into table t\_partition partition(department='xiangsheng',sex='male',howold=20);

### CTAS建表语法

可以通过已存在表来建表：

1、create table t\_user\_2 like t\_user;

新建的t\_user\_2表结构定义与源表t\_user一致，但是没有数据

1. 在建表的同时插入数据

|  |
| --- |
| create table t\_access\_user  as  select ip,url from t\_access; |

*t\_access\_user会根据select查询的字段来建表，同时将查询的结果插入新表中！*

补充：将查询出来的数据保存到一张表中：

方式1：create t\_x as select .......

方式2：

如果事先存在一张表t\_x

可以将select查询出来的结果数据insert到这张已存在的表中;

insert into t\_x select .......

## 数据导入导出

### 将数据文件导入hive的表

方式1：导入数据的一种方式：

手动用hdfs命令，将文件放入表目录；

方式2：在hive的交互式shell中用hive命令来导入本地数据到表目录

hive>load data local inpath '/root/order.data.2' into table t\_order;

方式3：用hive命令导入hdfs中的数据文件到表目录

hive>load data inpath '/access.log.2017-08-06.log' into table t\_access;

*注意：导本地文件和导HDFS文件的区别：*

*本地文件导入表：复制*

*hdfs文件导入表：移动*

方式4：如果目标表是一个分区表

hive> load data [local] inpath ‘......’ into table t\_dest partition(p=’value’);

### 将hive表中的数据导出到指定路径的文件

1. 将hive表中的数据导入HDFS的文件

*insert overwrite directory '/root/access-data'*

*row format delimited fields terminated by ','*

*select \* from t\_access;*

1. 将hive表中的数据导入本地磁盘文件

*insert overwrite local directory '/root/access-data'*

*row format delimited fields terminated by ','*

*select \* from t\_access limit 100000;*

### hive文件格式

HIVE支持很多种文件格式： SEQUENCE FILE | TEXT FILE | PARQUET FILE | RC FILE

create table t\_text(movie string,rate int) stored as textfile;

create table t\_seq(movie string,rate int) stored as sequencefile;

create table t\_pq(movie string,rate int) stored as parquetfile;

演示：

1、先建一个存储文本文件的表  
create table t\_access\_text(ip string,url string,access\_time string)

row format delimited fields terminated by ','

stored as textfile;

导入文本数据到表中：

load data local inpath '/root/access-data/000000\_0' into table t\_access\_text;

1. 建一个存储sequence file文件的表：

create table t\_access\_seq(ip string,url string,access\_time string)

stored as sequencefile;

从文本表中查询数据插入sequencefile表中，生成数据文件就是sequencefile格式的了：

insert into t\_access\_seq

select \* from t\_access\_text;

1. 建一个存储parquet file文件的表：

create table t\_access\_parq(ip string,url string,access\_time string)

stored as parquetfile;

## 数据类型

### 数字类型

[TINYINT](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types" \l "LanguageManualTypes-tinyint) (1字节整数)

[SMALLINT](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types" \l "LanguageManualTypes-smallint) (2字节整数)

[INT](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types" \l "LanguageManualTypes-int)/INTEGER (4字节整数)

[BIGINT](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types" \l "LanguageManualTypes-bigint) (8字节整数)

FLOAT (4字节浮点数)

DOUBLE (8字节双精度浮点数)

*示例：*

*create table t\_test(a string ,b int,c bigint,d float,e double,f tinyint,g smallint)*

### 时间类型

[TIMESTAMP](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types" \l "LanguageManualTypes-timestamp) (时间戳) (包含年月日时分秒毫秒的一种封装)

[DATE](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types" \l "LanguageManualTypes-date) (日期)（只包含年月日）

示例，假如有以下数据文件：

|  |
| --- |
| 1,zhangsan,1985-06-31  2,lisi,1986-07-10  3,wangwu,1985-08-09 |

那么，就可以建一个表来对数据进行映射

create table t\_customer(id int,name string,birthday date)

row format delimited fields terminated by ',';

然后导入数据

load data local inpath '/root/customer.dat' into table t\_customer;

然后，就可以正确查询

### 字符串类型

[STRING](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types" \l "LanguageManualTypes-string)

[VARCHAR](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types" \l "LanguageManualTypes-varchar)(20) (字符串1-65535长度，超长截断)

[CHAR](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types" \l "LanguageManualTypes-char) (字符串，最大长度255)

### 其他类型

BOOLEAN（布尔类型）：true false

~~BINARY (二进制)：~~

举例：

1,zs,28,true

2,ls,30,false

3,ww,32,false

4,lulu,18,true

*create table t\_p(id int,name string,age int,is\_married boolean)*

*select*

*from t\_p where is\_married;*

### 复合（集合）类型

array数组类型

arrays: ARRAY<data\_type> )

示例：array类型的应用

假如有如下数据需要用hive的表去映射：

|  |
| --- |
| 战狼2,吴京:吴刚:余男,2017-08-16  三生三世十里桃花,刘亦菲:痒痒,2017-08-20  羞羞的铁拳,沈腾:玛丽:艾伦,2017-12-20 |

设想：如果主演信息用一个数组来映射比较方便

建表：

create table t\_movie(moive\_name string,actors array<string>,first\_show date)

row format delimited fields terminated by ','

collection items terminated by ':';

导入数据：

load data local inpath '/root/movie.dat' into table t\_movie;

查询：

select \* from t\_movie;

select moive\_name,actors[0] from t\_movie;

select moive\_name,actors from t\_movie where array\_contains(actors,'吴刚');

select moive\_name,size(actors) from t\_movie;

map类型

maps: MAP<primitive\_type, data\_type>

1. **假如有以下数据：**

|  |
| --- |
| 1,zhangsan,father:xiaoming#mother:xiaohuang#brother:xiaoxu,28  2,lisi,father:mayun#mother:huangyi#brother:guanyu,22  3,wangwu,father:wangjianlin#mother:ruhua#sister:jingtian,29  4,mayun,father:mayongzhen#mother:angelababy,26 |

可以用一个map类型来对上述数据中的家庭成员进行描述

1. **建表语句：**

create table t\_person(id int,name string,family\_members map<string,string>,age int)

row format delimited fields terminated by ','

collection items terminated by '#'

map keys terminated by ':';

1. **查询**

select \* from t\_person;

*## 取map字段的指定key的值*

select id,name,family\_members['father'] as father from t\_person;

## 取map字段的所有key

select id,name,map\_keys(family\_members) as relation from t\_person;

## 取map字段的所有value

select id,name,map\_values(family\_members) from t\_person;

select id,name,map\_values(family\_members)[0] from t\_person;

*## 综合：查询有brother的用户信息*

*方式1：*

|  |
| --- |
| select id,name,father  from  (select id,name,family\_members['brother'] as brother from t\_person) tmp  where brother is not null; |

*方式2：*

*select \* from t\_map where array\_contains(map\_keys(family),'sister');*

struct类型

struct: STRUCT<col\_name : data\_type, ...>

1. **假如有如下数据：**

|  |
| --- |
| 1,zhangsan,18:male:beijing  2,lisi,28:female:shanghai |

其中的用户信息包含：年龄：整数，性别：字符串，地址：字符串

设想用一个字段来描述整个用户信息，可以采用struct

1. **建表：**

*create table t\_person\_struct(id int,name string,info struct<age:int,sex:string,addr:string>)*

*row format delimited fields terminated by ','*

*collection items terminated by ':';*

1. **查询**

*select \* from t\_person\_struct;*

*select id,name,info.age from t\_person\_struct;*

## 修改表定义

仅修改Hive元数据，不会触动表中的数据，用户需要确定实际的数据布局符合元数据的定义。

修改表名：

ALTER TABLE table\_name RENAME TO new\_table\_name

示例：alter table t\_1 rename to t\_x;

修改分区名：

alter table t\_partition partition(department='xiangsheng',sex='male',howold=20) rename to partition(department='1',sex='1',howold=20);

添加分区：

alter table t\_partition add partition (department='2',sex='0',howold=40);

删除分区：

alter table t\_partition drop partition (department='2',sex='2',howold=24);

修改表的文件格式定义：

ALTER TABLE table\_name [PARTITION partitionSpec] SET FILEFORMAT file\_format

修改表的某个文件格式定义：

alter table t\_partition partition(department='2',sex='0',howold=40 ) set fileformat sequencefile;

修改列名定义：

ALTER TABLE table\_name CHANGE [COLUMN] col\_old\_name col\_new\_name column\_type [COMMENTcol\_comment] [FIRST|(AFTER column\_name)]

alter table t\_user change price jiage float first;

增加/替换列：

ALTER TABLE table\_name ADD|REPLACE COLUMNS (col\_name data\_type[COMMENT col\_comment], ...)

alter table t\_user add columns (sex string,addr string);

alter table t\_user replace columns (id string,age int,price float);

# hive查询语法

sql是一门面向集合的编程语言；

select 1;

提示：在做小数据量查询测试时，可以让hive将mrjob提交给本地运行器运行，可以在hive会话中设置如下参数：

hive> set hive.exec.mode.local.auto=true;

## 基本查询示例

select \* from t\_access;

select count(\*) from t\_access;

select max(ip) from t\_access;

## 条件查询

select \* from t\_access where access\_time<'2017-08-06 15:30:20'

select \* from t\_access where access\_time<'2017-08-06 16:30:20' and ip>'192.168.33.3';

## join关联查询示例

假如有a.txt文件

|  |
| --- |
| a,1  b,2  c,3  d,4 |

假如有b.txt文件

|  |
| --- |
| a,xx  b,yy  d,zz  e,pp |

进行各种join查询：

1. inner join（join）

|  |
| --- |
| select  a.name as aname,  a.numb as anumb,  b.name as bname,  b.nick as bnick  from t\_a a  join t\_b b  on a.name=b.name |

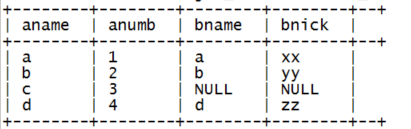
结果：

|  |
| --- |
| +--------+--------+--------+--------+--+  | aname | anumb | bname | bnick |  +--------+--------+--------+--------+--+  | a | 1 | a | xx |  | b | 2 | b | yy |  | d | 4 | d | zz |  +--------+--------+--------+--------+--+ |

1. left outer join（left join）

|  |
| --- |
| select  a.name as aname,  a.numb as anumb,  b.name as bname,  b.nick as bnick  from t\_a a  left outer join t\_b b  on a.name=b.name |

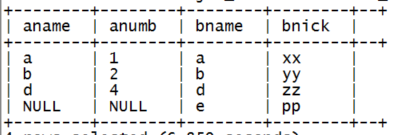
结果：



1. right outer join（right join）

|  |
| --- |
| select  a.name as aname,  a.numb as anumb,  b.name as bname,  b.nick as bnick  from t\_a a  right outer join t\_b b  on a.name=b.name |

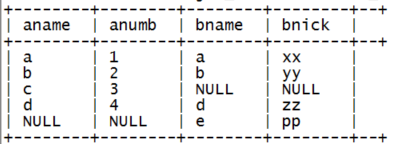
结果：



1. full outer join（full join）

|  |
| --- |
| select  a.name as aname,  a.numb as anumb,  b.name as bname,  b.nick as bnick  from t\_a a  full join t\_b b  on a.name=b.name; |

结果：



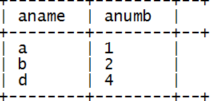
## left semi join

*Left semi join ：相当于join连接两个表后产生的数据中的左半部分*

hive中不支持exist/IN子查询，可以用left semi join来实现同样的效果：

|  |
| --- |
| select  a.name as aname,  a.numb as anumb  from t\_a a  left semi join t\_b b  on a.name=b.name; |

结果：



*注意： left semi join的 select子句中，不能有右表的字段*

## group by分组聚合

|  |
| --- |
| 20170804,192.168.33.66,http://www.edu360.cn/job  20180804,192.168.33.40,http://www.edu360.cn/study  20180805,192.168.20.18,http://www.edu36.cn/job  20180805,192.168.20.28,http://www.edu36.cn/login  20180806,192.168.20.38,http://www.edu36.cn/job  20180806,192.168.20.38,http://www.edu36.cn/study  20180807,192.168.33.40,http://www.edu36.cn/login  20180807,192.168.20.88,http://www.edu36.cn/job |

select dt,count(\*),max(ip) as cnt from t\_access group by dt;

select dt,count(\*),max(ip) as cnt from t\_access group by dt having dt>'20170804';

select

dt,count(\*),max(ip) as cnt

from t\_access

where url='http://www.edu360.cn/job'

group by dt having dt>'20170804';

注意： 一旦有group by子句，那么，在select子句中就不能有 （分组字段，聚合函数） 以外的字段

## 为什么where必须写在group by的前面，为什么group by后面的条件只能用having

因为，where是用于在真正执行查询逻辑之前过滤数据用的

having是对group by聚合之后的结果进行再过滤；

*上述语句的执行逻辑：*

1. *where过滤不满足条件的数据*
2. *用聚合函数和group by进行数据运算聚合，得到聚合结果*
3. *用having条件过滤掉聚合结果中不满足条件的数据*

## 子查询

|  |
| --- |
| 1,zhangsan,father:xiaoming#mother:xiaohuang#brother:xiaoxu,28  2,lisi,father:mayun#mother:huangyi#brother:guanyu,22  3,wangwu,father:wangjianlin#mother:ruhua#sister:jingtian,29  4,mayun,father:mayongzhen#mother:angelababy,26 |

-- 查询有兄弟的人

select id,name,brother

from

(select id,name,family\_members['brother'] as brother from t\_person) tmp

where brother is not null;

另一种写法：

select id,name,family\_members[‘brother’]

from t\_person where array\_contains(map\_keys(family\_members),”brother”);

# hive函数使用

测试函数小技巧：

***直接用常量来测试函数即可***

select substr("abcdefg",1,3);

而且，可以将hive的本地运行自动模式开启：

hive>set hive.exec.mode.local.auto=true;

HIVE 的所有函数手册：

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF#LanguageManualUDF-Built-inTable-GeneratingFunctions(UDTF)

## 常用内置函数

类型转select cast("5" as int) ~~;~~

select cast("2017-08-03" as date) ;

select cast(current\_timestamp as date);

*示例：*

### 换函数

|  |  |  |
| --- | --- | --- |
| *1* | *1995-05-05 13:30:59* | *1200.3* |
| *2* | *1994-04-05 13:30:59* | *2200* |
| *3* | *1996-06-01 12:20:30* | *80000.5* |

*create table t\_fun(id string,birthday string,salary string)*

*row format delimited fields terminated by ',';*

*select id,cast(birthday as date) as bir,cast(salary as float) from t\_fun;*

### 数学运算函数

select round(5.4); ## 5 四舍五入

select round(5.1345,3) ; ##5.135

select ceil(5.4) ; // select ceiling(5.4) ; ## 6 向上取整

select floor(5.4); ## 5 向下取整

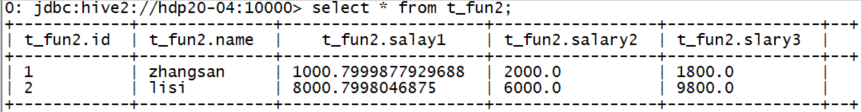
select abs(-5.4) ; ## 5.4 绝对值

select greatest(id1,id2,id3) ; ## 6 单行函数

select least(3,5,6) ; ##求多个输入参数中的最小值

示例：

有表如下：



select greatest(cast(s1 as double),cast(s2 as double),cast(s3 as double)) from t\_fun2;

结果：

+---------+--+

| \_c0 |

+---------+--+

| 2000.0 |

| 9800.0 |

+---------+--+

select max(age) from t\_person group by ..; 分组聚合函数

select min(age) from t\_person group by...; 分组聚合函数

### 字符串函数

substr(string str, int start) ## 截取子串

substring(string str, int start)

*示例：select substr("abcdefg",2) ;*

**substr(string, int start, int len)**

substring(string, int start, int len)

*示例：select substr("abcdefg",2,3) ; ## bcd*

**concat**(string A, string B...) ## 拼接字符串

**concat\_ws**(string SEP, string A, string B...)

*示例：select concat("ab","xy") ; ## abxy*

*select concat\_ws(".","192","168","33","44") ; ## 192.168.33.44*

**length**(string A)

*示例：select length("192.168.33.44"); ## 13*

**split(string str, string pat) ## 切分字符串，返回数组**

示例：~~select split("192.168.33.44",".") ;~~ 错误的，因为.号是正则语法中的特定字符

select split("192.168.33.44","\\.") ;

**upper**(string str) ##转大写

**lower**(string str) ##转小写

### 时间函数

select current\_timestamp; ## 返回值类型：timestamp，获取当前的时间戳(详细时间信息)

select current\_date; ## 返回值类型：date，获取当前的日期

## unix时间戳转字符串格式——**from\_unixtime**

**from\_unixtime**(bigint unixtime[, string format])

示例：select from\_unixtime(unix\_timestamp());

select from\_unixtime(unix\_timestamp(),"yyyy/MM/dd HH:mm:ss");

## 字符串格式转unix时间戳——**unix\_timestamp：返回值是一个长整数类型**

## 如果不带参数，取当前时间的秒数时间戳long--(距离格林威治时间1970-1-1 0:0:0秒的差距)

select **unix\_timestamp()**;

**unix\_timestamp**(string date, string pattern)

示例： select unix\_timestamp("2017-08-10 17:50:30");

select unix\_timestamp("2017-08-10 17:50:30","yyyy-MM-dd HH:mm:ss");

## 将字符串转成日期date

select **to\_date**("2017-09-17 16:58:32");

### 条件控制函数

IF

*select id,if(age>25,'working','worked') from t\_user;*

*select moive\_name,if(array\_contains(actors,'吴刚'),'好电影',’烂片儿’)*

*from t\_movie;*

case when

语法：

CASE [ expression ]

WHEN condition1 THEN result1

WHEN condition2 THEN result2

...

WHEN conditionn THEN resultn

ELSE result

END

示例：

|  |
| --- |
| 1,zhangsan,18,beijing,20000  2,lisi,28,shanghai,1500  3,wangwu,38,wuhan,4000  4,zhaoliu,35,changsha,10050  5,tianqi,23,shijiazhuang,30000  6,wangba,45,qinhuangdao,28000  7,wushuai,55,haerbin,80000 |

查询：

1,zhangsan,少年,beijing,中产

|  |
| --- |
| 建表：  create table t10(id int,name string,age int,addr string,income int)  row format delimited fields terminated by ',';  查询：  select id,name,  case  when age<=20 then '少年'  when age>20 and age<=40 then '青年'  when age>40 and age<=60 then '中年'  when age>60 then '老年'  end as status,  addr,  income,  case  when income<=5000 then '穷屌丝'  when income>5000 and income<=15000 then '富屌丝'  when income>15000 and income<=30000 then '还行'  when income>30000 and income<=60000 then '涛哥'  when income>60000 then '土豪'  end as level  from t10; |

结果：

|  |
| --- |
| +-----+-----------+---------+---------------+---------+--------+--+  | id | name | status | addr | income | level |  +-----+-----------+---------+---------------+---------+--------+--+  | 1 | zhangsan | 少年 | beijing | 20000 | 还行 |  | 2 | lisi | 青年 | shanghai | 1500 | 穷屌丝 |  | 3 | wangwu | 青年 | wuhan | 4000 | 穷屌丝 |  | 4 | zhaoliu | 青年 | changsha | 10050 | 富屌丝 |  | 5 | tianqi | 青年 | shijiazhuang | 30000 | 还行 |  | 6 | wangba | 中年 | qinhuangdao | 28000 | 还行 |  | 7 | wushuai | 中年 | haerbin | 80000 | 土豪 |  +-----+-----------+---------+---------------+---------+--------+--+ |

### 集合函数

**array(3,5,8,9) 构造一个整数数组**

**array(‘hello’,’moto’,’semense’,’chuizi’,’xiaolajiao’) 构造一个字符串数组**

**array\_contains**(Array<T>, value) 返回boolean值

示例：

*select moive\_name,array\_contains(actors,'吴刚') from t\_movie;*

*select array\_contains(array('a','b','c'),'c') ;*

**sort\_array**(Array<T>) 返回排序后的数组

*示例：*

*select sort\_array(array('c','b','a')) ;*

*select 'haha',sort\_array(array('c','b','a')) as xx from (select 0) tmp;*

**size**(Array<T>) 返回一个集合的长度，int值

*示例：*

*select moive\_name,size(actors) as actor\_number from t\_movie;*

**size**(Map<K.V>) 返回一个imap的元素个数，int值

size(array<T>) 返回一个数组的长度,int值

**map\_keys**(Map<K.V>) 返回一个map字段的所有key，结果类型为：数组

**map\_values**(Map<K.V>) 返回一个map字段的所有value，结果类型为：数组

### 常见分组聚合函数

sum(字段) : 求这个字段在一个组中的所有值的和

avg(字段) ： 求这个字段在一个组中的所有值的平均值

max(字段) ：求这个字段在一个组中的所有值的最大值

min(字段) ：求这个字段在一个组中的所有值的最小值

分组聚合函数练习：

|  |
| --- |
| 聚合函数练习：  原始数据：  1,zhangsan,18:male:北京  2,lisi,28:female:上海  3,wangwu,30:male:天津  4,zhaoliu,32:female:石家庄  建表语句：  create table t\_user\_info(id int,name string,info struct<age:int,gender:string,addr:string>)  row format delimited fields terminated by ','  collection items terminated by ':';  1.求男性及女性分别的年龄总和  select info.gender,sum(info.age)  from t\_user\_info  group by info.gender;  select gender,sum(age)  from  (select id,  name,  info,  info.age as age,  info.gender as gender  from t\_user\_info) tmp  group by tmp.gender  ;  2.求男性及女性分别的最大年龄  select info.gender,max(info.age)  from t\_user\_info  group by info.gender;  3.求男性及女性分别的最小年龄  select info.gender,min(info.age)  from t\_user\_info  group by info.gender;  4.求男性及女性分别的平均年龄  select info.gender,avg(info.age)  from t\_user\_info  group by info.gender;  5.求所有人的信息，及男、女性别各自的平均年龄  1,zhangsan,18:male:北京,  2,lisi,28:female:上海,  3,wangwu,30:male:天津  4,zhaoliu,32:female:石家庄  select a.\*,b.avg\_age  from  t\_user\_info a  join  (select info.gender as gender,avg(info.age) as avg\_age  from t\_user\_info  group by info.gender) b  on a.info.gender=b.gender  where a.info.age > b.avg\_age  ; |

**count()：**求一个组中的满足某条件的数据条数！

举例说明：

1,mary,female,jiuye

2,tom,male,meijiuye

3,kitty,female,meijiuye

4,white,male,jiuye

5,jack,male,jiuye

6,rose,female,meijiuye

建表语句：

create table t\_count(id int,name string,gender string,job string)

row format delimited fields terminated by ‘,’;

请求出，男生和女生中分别已就业的人数

方式1 ：在count计数时进行判断是否需要计入

|  |
| --- |
| select sex,count(if(job='jiuye',1,null))  from t11  group by sex; |

方式3：在count计数时判断是否需要计入：

|  |
| --- |
| select sex,count(case when job='jiuye' then 1 else null end)  from t11  group by sex; |

方式3：先过滤掉不需要计入的数据，再分组计数

|  |
| --- |
| select sex,count(1)  from  (  select \*  from t11  where job='jiuye') o1  group by o1.sex; |

**collect\_set() :将某个字段在一组中的所有值形成一个集合（数组）返回**

举例：

|  |
| --- |
| 1,zhangsan,数学  1,zhangsan,化学  1,zhangsan,语文  1,zhangsan,搭讪学  2,lisi,数学  2,lisi,化学  2,lisi,聊骚  2,lisi,成人搏斗学  3,wangwu,防狼术  3,wangwu,跆拳道 |

create table t13(id int,name string,subject string)

row format delimited fields terminated by ‘,’;

需要查询出如下结果：

|  |  |  |
| --- | --- | --- |
| 1 | zhangsan | 数学 化学 语文 搭讪学 |
| 2 | lisi | 数学 化学 聊骚 |
| ..... | .... | .... |

select id,name,collect\_set(subject)

from t13

group by id,name;

|  |
| --- |
| +-----+-----------+------------------------------------------------------------+--+  | id | name | \_c2 |  +-----+-----------+--------------------------------------------------------------+--+  | 1 | zhangsan | ["数学","化学","语文","街头搭讪学"] |  | 2 | lisi | ["数学","化学","聊骚","成人搏斗学"] |  | 3 | wangwu | ["防狼术","跆拳道"] |  +-----+-----------+----------------------------------------------------------------+--+ |

**disctinct**

数据：

|  |
| --- |
| 1,zhangsan,28,beijing  1,lisi,29,shanghai  1,wangwu,30,beijing  2,zhaoliu,18,tianjin  2,tianqi,19,shenzhen  3,wangba,21,shenzhen |

**建表：**

|  |
| --- |
| create table t\_distinct(id int,  name string,  age int,  addr string)  row format delimited fields terminated by ',';  load data local inpath '/root/hivedata/distinct.dat' into table t\_distinct; |

### 表生成函数

行转列函数：explode()

假如有以下数据：

|  |
| --- |
| 1,zhangsan,化学:物理:数学:语文  2,lisi,化学:数学:生物:生理:卫生  3,wangwu,化学:语文:英语:体育:生物 |

映射成一张表：

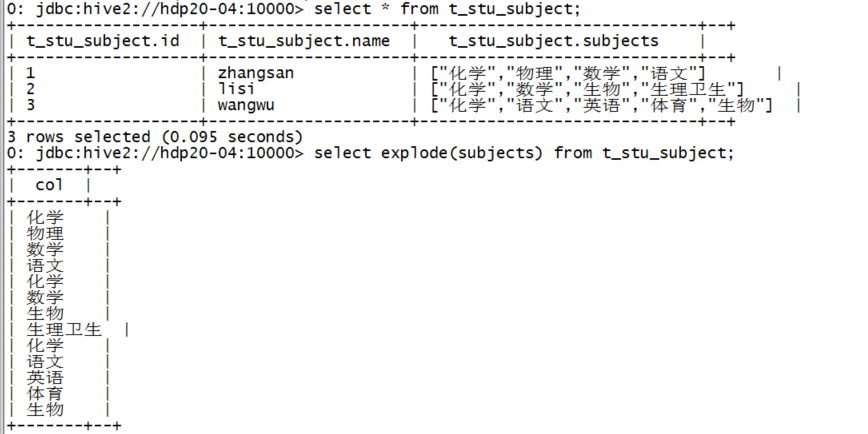
create table t\_stu\_subject(id int,name string,subjects array<string>)

row format delimited fields terminated by ','

collection items terminated by ':';

load data local inpath ‘/root/sb.dat’ into table t\_stu\_subject;

使用explode()对数组字段“炸裂”



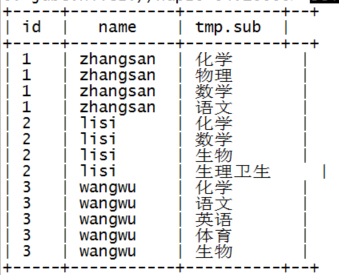
然后，我们利用这个explode的结果，来求去重的课程：

|  |
| --- |
| select distinct tmp.sub  from  (select explode(subjects) as sub from t\_stu\_subject) tmp; |

表生成函数lateral view

select id,name,tmp.sub

from t\_stu\_subject lateral view explode(subjects) tmp as sub;



*理解： lateral view 相当于两个表在join*

*左表：是原表*

*右表：是explode(某个集合字段)之后产生的表*

*而且：这个join只在同一行的数据间进行*

*那样，可以方便做更多的查询：*

*比如，查询选修了生物课的同学*

*select a.id,a.name,a.sub from*

(select id,name,tmp.sub as sub from t\_stu\_subject lateral view explode(subjects) tmp as sub) a

where sub='生物';

### json解析函数：表生成函数

需求：有如下json格式的电影评分数据：

|  |
| --- |
| {"movie":"1193","rate":"5","timeStamp":"978300760","uid":"1"}  {"movie":"661","rate":"3","timeStamp":"978302109","uid":"1"}  {"movie":"914","rate":"3","timeStamp":"978301968","uid":"1"}  {"movie":"3408","rate":"4","timeStamp":"978300275","uid":"1"}  {"movie":"2355","rate":"5","timeStamp":"978824291","uid":"1"}  {"movie":"1197","rate":"3","timeStamp":"978302268","uid":"1"} |

需要做各种统计分析。

发现，直接对json做sql查询不方便，需要将json数据解析成普通的结构化数据表。可以采用hive中内置的json\_tuple()函数

实现步骤：

1. 创建一个原始表用来对应原始的json数据

create table t\_json(json string);

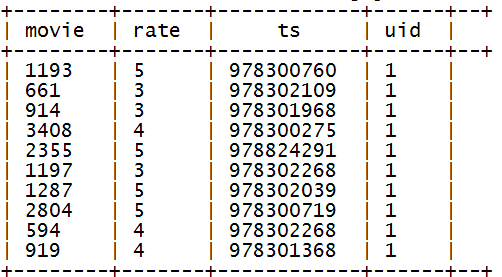
load data local inpath ‘/root/rating.json’ into table t\_json;

1. 利用**json\_tuple**进行json数据解析

测试，示例：

select json\_tuple(json,'movie','rate','timeStamp','uid') as(movie,rate,ts,uid) from t\_json limit 10;

产生结果：



真正解析整张json表，将解析结果数据插入一张新表

create table t\_movie\_rate

as

select json\_tuple(json,'movie','rate','timeStamp','uid') as(movie,rate,ts,uid) from t\_json;

-- TODO 练习--

1/ 统计每部电影的平均得分

create table t\_rate\_m\_avg as select movie,avg(rate) from t\_rate group by movie;

2/ 统计数据中一共有多少部电影

select count(distinct movie) from t\_rate;

3/ 统计每部电影的被评分次数（热门度）

create table t\_rate\_m\_cnt as select movie,count(1) as cnt from t\_rate group by movie order by cnt desc;

4/ 统计每个人的所有电影评分的平均值

create table t\_rate\_u\_avg as select uid,avg(rate) from t\_rate group by uid;

5/ 查询出每个人评分最高的20部电影信息

利用json\_tuple从原始json数据表中，etl出一个详细信息表：

|  |
| --- |
| create table t\_rate  as  select  uid,  movie,  rate,  year(from\_unixtime(cast(ts as bigint))) as year,  month(from\_unixtime(cast(ts as bigint))) as month,  day(from\_unixtime(cast(ts as bigint))) as day,  hour(from\_unixtime(cast(ts as bigint))) as hour,  minute(from\_unixtime(cast(ts as bigint))) as minute,  from\_unixtime(cast(ts as bigint)) as ts  from  (select  json\_tuple(rateinfo,'movie','rate','timeStamp','uid') as(movie,rate,ts,uid)  from t\_json) tmp  ; |

### 窗口分析函数：

#### row\_number() over()——分组TOPN

需求

有如下数据：

|  |
| --- |
| 1,18,a,male  2,19,b,male  3,22,c,female  4,16,d,female  5,30,e,renyao  6,26,f,female  7,16,d,renyao  8,30,e,renyao  9,26,f,female  10,14,b,male |

需要查询出每种性别中年龄最大的2条数据

create table t\_rn(id int,age int,name string,sex string)

row format delimited fields terminated by ‘,’;

实现：

使用row\_number函数，对表中的数据按照性别分组，按照年龄倒序排序并进行标记

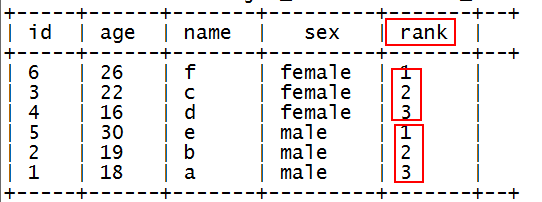
hql代码：

select id,age,name,sex,

row\_number() over(partition by sex order by age desc) as rank

from t\_rownumber

产生结果：



然后，利用上面的结果，查询出rank<=2的即为最终需求

select id,age,name,sex

from

(select id,age,name,sex,

row\_number() over(partition by sex order by age desc) as rank

from t\_rownumber) tmp

where rank<=2;

*练习：求出电影评分数据中，每个用户评分最高的topn条数据*

#### sum() over()——级联求和

需求：同8.2.1

(共享单车：本月新增用户数以及累积到本月的新增用户数...

本月充值额以及累积到本月的充值额....

本月用户骑行总时长以及累计到本月的骑行总时长......

........

)

实现：使用sum()over()窗口分析函数来实现

步骤1：先求出每家店铺每个月的总金额

create table t\_tmp as

select name,month,sum(sale) as amt from shop group by name,month;

+-------+--------+-------+--+

| name | month | amt |

+-------+--------+-------+--+

| a | 01 | 350 |

| a | 02 | 5000 |

| a | 03 | 600 |

| b | 01 | 7800 |

| b | 02 | 2500 |

| c | 01 | 470 |

| c | 02 | 630 |

+-------+--------+-------+--+

步骤2：实现级联求和

select name,month,amt,

sum(amt) over(partition by name order by month rows between unbounded preceding and current row) as accumulate

from t\_tmp;

drop table t\_tmp;

|  |  |
| --- | --- |
| 重要提示 | sum()over()的累加范围指定语法：  *sum() over(partition by x order by y rows between 8 preceding and current row)*  *sum() over(partition by x order by y rows between 8 preceding and 5 following)*  *sum() over(partition by x order by y rows between unbounded preceding and 5 following)*  *sum() over(partition by x order by y rows between unbounded preceding and unbounded following)* |

#### 窗口分析函数综合应用案例

有以下数据：

|  |
| --- |
| a,2017-02-05,200  a,2017-02-06,300  a,2017-02-07,200  a,2017-02-08,400  a,2017-02-10,600  b,2017-02-05,200  b,2017-02-06,300  b,2017-02-08,200  b,2017-02-09,400  b,2017-02-10,600  c,2017-01-31,200  c,2017-02-01,300  c,2017-02-02,200  c,2017-02-03,400  c,2017-02-10,600  a,2017-03-01,200  a,2017-03-02,300  a,2017-03-03,200  a,2017-03-04,400  a,2017-03-05,600 |

请找出那些有过连续4天都有销售记录的店铺；

（需求还可以演化成：有过连续10天登录app的用户；

有过连续10天扫码骑车的用户；...

有过连续观看公司推送的广告的用户；

有过连续n月充值的用户统计.....

有过连续N天登录本公司网站的用户统计.....

）

结果数据：

|  |
| --- |
| b  d |

答案sql：

步骤1：先按店铺分窗口打rownumber

select shop,s\_date,amt,

row\_number() over(partition by shop order by s\_date) as rn

from t18;

|  |
| --- |
| +-------+-------------+------+-----+--+  | shop | s\_date | amt | rn |  +-------+-------------+------+-----+--+  | a | 2017-02-05 | 200 | 1 |  | a | 2017-02-06 | 300 | 2 |  | a | 2017-02-07 | 200 | 3 |  | a | 2017-02-08 | 400 | 4 |  | a | 2017-02-10 | 600 | 5 |  | b | 2017-02-05 | 200 | 1 |  | b | 2017-02-06 | 300 | 2 |  | b | 2017-02-08 | 200 | 3 |  | b | 2017-02-09 | 400 | 4 |  | b | 2017-02-10 | 600 | 5 |  | c | 2017-01-31 | 200 | 1 |  | c | 2017-02-01 | 300 | 2 |  | c | 2017-02-02 | 200 | 3 |  | c | 2017-02-03 | 400 | 4 |  | c | 2017-02-10 | 600 | 5 |  +-------+-------------+------+-----+--+ |

然后，用 日期-rownumber得到一个差值

|  |
| --- |
| select shop,s\_date,amt,date\_sub(s\_date,rn) as diff  from  (  select shop,s\_date,amt,  row\_number() over(partition by shop order by s\_date) as rn  from t18) o1; |

|  |
| --- |
| +-------+-------------+------+-------------+--+  | shop | s\_date | amt | diff |  +-------+-------------+------+-------------+--+  | a | 2017-02-05 | 200 | 2017-02-04 |  | a | 2017-02-06 | 300 | 2017-02-04 |  | a | 2017-02-07 | 200 | 2017-02-04 |  | a | 2017-02-08 | 400 | 2017-02-04 |  | a | 2017-02-10 | 600 | 2017-02-05 |  | b | 2017-02-05 | 200 | 2017-02-04 |  | b | 2017-02-06 | 300 | 2017-02-04 |  | b | 2017-02-08 | 200 | 2017-02-05 |  | b | 2017-02-09 | 400 | 2017-02-05 |  | b | 2017-02-10 | 600 | 2017-02-05 |  | c | 2017-01-31 | 200 | 2017-01-30 |  | c | 2017-02-01 | 300 | 2017-01-30 |  | c | 2017-02-02 | 200 | 2017-01-30 |  | c | 2017-02-03 | 400 | 2017-01-30 |  | c | 2017-02-10 | 600 | 2017-02-05 |  +-------+-------------+------+-------------+--+ |

按店铺,差值 分组计数，并过滤出技术值>=4的数据，并对店铺去重，就是那些有过连续4天以上销售记录的店铺

|  |
| --- |
| select distinct shop  from  (  select shop,count(1) as cnt  from  (  select shop,s\_date,amt,date\_sub(s\_date,rn) as diff  from  (  select shop,s\_date,amt,  row\_number() over(partition by shop order by s\_date) as rn  from t18) o1) o2  group by shop,diff  having cnt>=4  ) o3  ; |

|  |
| --- |
| +-------+--+  | shop |  +-------+--+  | a |  | c |  +-------+--+ |

## 自定义函数

### 需求1：

有如下数据：

|  |
| --- |
| id,salay,bonus,subsidy  a,100,50,120  b,220,150,20  c,220,450,220 |

3个字段分别表示： 用户id，基本工资，业绩提成，股权收益

需要查询出每个人的三类收益中最高的是哪一种收益

|  |
| --- |
| a,100,50,120,subsidy  b,220,150,20,salary  c,220,450,220,bonus |

hive中有一个函数greatest(f1,f2,f3)可以求出n个字段中的最大值，满足不了本案例的需求。此时，我们可以考虑自己开发一个hive的函数（hive具备这个机制）

create table t19(id int, salary int,bonus int,subsidy int)

row format delimited fields terminated by ‘,’;

通过需求分析，可得，如果有如下函数

select id,highest\_income\_type(salary,bonus,subsidy) from t19

highest\_income\_type(salary,bonus,subsidy) :能够接收3个整数，返回哪一个是最大的

就很容易实现需求；

可惜这样的函数在hive中没有；

解决：可以自定义一个这样的函数；

### 实现思路：

hive的函数无非也就是一个逻辑的封装，可以接收参数，返回结果，跟java中的方法本质上没有区别

hive就允许用户开发一个java的方法，来实现你想要的函数的功能；

然后在hive中定义一个自己命名的函数，并将这个函数跟你的java方法所在的类关联起来即可。

### 实现步骤：

1/ 开发一个java类继承（HIVE的父类UDF），重载一个方法： evaluate()

方法的功能：

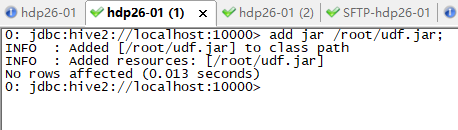
输入：3个整数值

返回：最大值所在的序号

|  |
| --- |
| **public** **class** HighestIncomType **extends** UDF{  // 重载evaluate方法，而且必须是public  **public** **int** evaluate(**int** a,**int** b,**int** c) {  // 求出最大值是几  **int** max = a;  **if**(a<b) {  max = b;  }  **if**(max<c) {  max = c;  }      //判断谁是最大值  **if**(a==max) {  **return** 0;  }  **if**(b==max) {  **return** 1;  }**else**{  **return** 2;  }  }  } |

2/ 将java工程打成jar包，上传到hive所在的机器上

3/ 在hive的提示符中，将jar包添加到hive的运行时classpath



4/ 在hive的提示中，用hive语法声明一个自定义函数，并与jar包中的java类关联

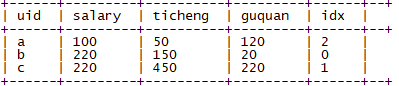
|  |
| --- |
| 0:jdbc:hive2://localhost:10000> create temporary function highest\_income\_type as 'top.ganhoo.hive.udf.HighestIncomType';  No rows affected (0.016 seconds) |

5/ 就可以在sql查询中使用这个函数了

|  |
| --- |
| a,100,50,120  b,220,150,20  c,220,450,220 |

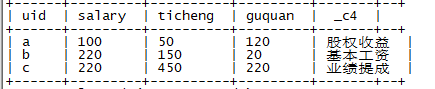
使用函数实现需求：

|  |
| --- |
| select id,  case  when type=0 then '基本工资'  when type=1 then '股权收益'  else '津贴'  end as type  from  (select id,highest\_income\_type(salary,bonus,subsidy) as type from t19) o1; |



用case-when改改显示值：

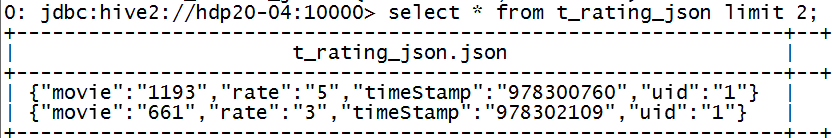
|  |
| --- |
| select uid,salary,ticheng,guquan,  case  when get\_max\_index(salary,ticheng,guquan) =0 then '基本工资'  when get\_max\_index(salary,ticheng,guquan) =1 then '业绩提成'  when get\_max\_index(salary,ticheng,guquan) =2 then '股权收益'  end  from t\_employee; |



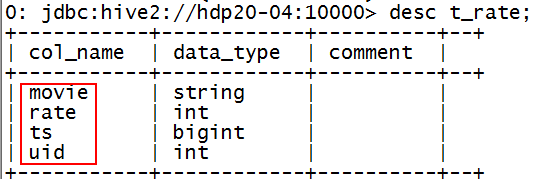
### 需求2：

需要对json数据表中的json数据写一个自定义函数，用于传入一个json，返回一个数据值的数组

json原始数据表：



需要做ETL操作，将json数据变成普通表数据，插入另一个表中：



### 实现步骤：

1、开发JAVA的UDF类

|  |
| --- |
| **public** **class** ParseJson **extends** UDF{    // 重载 ：返回值类型 和参数类型及个数，完全由用户自己决定  // 本处需求是：给一个字符串，返回一个数组  **public** String[] evaluate(String json) {    String[] split = json.split("\"");  String[] res = **new** String[]{split[3],split[7],split[11],split[15]};  **return** res;  }  } |

2、打jar包

在eclipse中使用export即可

1. 上传jar包到运行hive所在的linux机器
2. 在hive中创建临时函数：

在hive的提示符中：

*hive> add jar /root/jsonparse.jar;*

然后，在hive的提示符中，创建一个临时函数：

hive>CREATE TEMPORARY FUNCTION jsonp AS 'cn.edu360.hdp.hive.ParseJson';

1. 开发hql语句，利用自定义函数，从原始表中抽取数据插入新表

|  |
| --- |
| insert into table t\_rate  select  split(jsonp(json),',')[0],  cast(split(jsonp(json),',')[1] as int),  cast(split(jsonp(json),',')[2] as bigint),  cast(split(jsonp(json),',')[3] as int)  from  t\_rating\_json; |

*注：临时函数只在一次hive会话中有效，重启会话后就无效*

如果需要经常使用该自定义函数，可以考虑创建永久函数：

拷贝jar包到hive的类路径中：

cp wc.jar apps/hive-1.2.1/lib/

创建了：

create function pfuncx as 'com.doit.hive.udf.UserInfoParser';

删除函数：

DROP TEMPORARY FUNCTION [IF EXISTS] function\_name

DROP FUNCTION[IF EXISTS] function\_name

# 综合查询案例

## 用hql来做wordcount

有以下文本文件：

|  |
| --- |
| hello tom hello jim  hello rose hello tom  tom love rose rose love jim  jim love tom love is what  what is love |

需要用hive做wordcount

-- 建表映射

create table t\_wc(sentence string);

-- 导入数据

load data local inpath '/root/hivetest/xx.txt' into table t\_wc;

hql答案：

|  |
| --- |
| SELECT word  ,count(1) as cnts  FROM (  SELECT explode(split(sentence, ' ')) AS word  FROM t\_wc  ) tmp  GROUP BY word  order by cnts desc  ; |

## 略难报表查询

### 级联累计报表查询

有如下数据：

|  |
| --- |
| A,2015-01-08,5  A,2015-01-11,15  B,2015-01-12,5  A,2015-01-12,8  B,2015-01-13,25  A,2015-01-13,5  C,2015-01-09,10  C,2015-01-11,20  A,2015-02-10,4  A,2015-02-11,6  C,2015-01-12,30  C,2015-02-13,10  B,2015-02-10,10  B,2015-02-11,5  A,2015-03-20,14  A,2015-03-21,6  B,2015-03-11,20  B,2015-03-12,25  C,2015-03-10,10  C,2015-03-11,20 |

建表映射：

|  |
| --- |
| create table t\_access\_times(username string,month string,counts int)  row format delimited fields terminated by ','; |

需要要开发hql脚本，来统计出如下累计报表：

|  |  |  |  |
| --- | --- | --- | --- |
| 店铺 | 月份 | 月总额 | 累计到当月的总额 |
| A | 2015-01 | 33 | 33 |
| A | 2015-02 | 10 | 43 |
| A | 2015-03 | 30 | 73 |
| B | 2015-01 | 30 | 30 |
| B | 2015-02 | 15 | 45 |
|  |  |  |  |

#### 实现方式1：

——利用自jion

*答案： 参见《复杂sql案例--套路--累计报表.txt》*

#### 实现方式2

——利用sum()over()窗口分析函数：

|  |
| --- |
| select shopid,month,amount,  sum(amount) over(partition by shopid order by month rows between unbounded preceding and current row ) as accumulate\_amount  from (  select shopid,substr(dt,1,7) as month,sum(sale) as amount  from t\_jd group by shopid,substr(dt,1,7)  ) tmp; |

### 连续销售记录查询

请查出整个数据集中，那些有过连续3天销售记录的店；

|  |
| --- |
| A,2017-10-11,300  A,2017-10-12,200  B,2017-10-11,400  B,2017-10-12,200  A,2017-10-13,100  B,2017-10-15,600  A,2017-10-15,100  C,2017-10-11,350  A,2017-10-16,300  C,2017-10-13,250  A,2017-10-17,150  C,2017-10-14,300  C,2017-10-15,400  A,2017-10-18,340  A,2017-10-19,360  C,2017-10-16,200  D,2017-10-13,500  E,2017-10-14,600  E,2017-10-15,500  D,2017-10-14,600 |

create table t\_continue (id string,sale\_d date,sale int)

row format delimited fields terminated by ‘,’;

思路：让连续的日期行，都附带一个相同的差值

|  |
| --- |
| id , day ,sale,rn,diff  A,2017-10-11,300,1,2017-10-10  A,2017-10-12,200,2,2017-10-10  A,2017-10-13,100,3,2017-10-10  A,2017-10-15,100,4,2017-10-11  A,2017-10-16,300,5,2017-10-11  A,2017-10-17,150,6,2017-10-11  A,2017-10-18,340,7,2017-10-11  A,2017-10-19,360,8,2017-10-11 |

最终答案：

|  |
| --- |
| SELECT DISTINCT shopid  FROM (  SELECT shopid  ,count(1) AS lianxu  FROM (  SELECT shopid  ,day  ,sale  ,date\_sub(day, rn) AS diff  FROM (  SELECT shopid  ,day  ,sale  ,row\_number() OVER (  PARTITION BY shopid ORDER BY day  ) AS rn  FROM t\_jd\_2  ) tmp  ) tmp2  GROUP BY shopid  ,diff  HAVING lianxu >= 3  ) tmp3; |