Day_13_ hive function

hadoop hive

java代码操作hive

创建maven工程，导入依赖文件

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>com.mycom.hive</groupId>
 <artifactId>hivetest</artifactId>
 <version>0.0.1-SNAPSHOT</version>

 <dependencies>
  <dependency>
   <groupId>org.apache.hive</groupId>
   <artifactId>hive-jdbc</artifactId>
   <version>2.3.0</version>
  </dependency>
  <dependency>
   <groupId>jdk.tools</groupId>
   <artifactId>jdk.tools</artifactId>
   <version>1.8</version>
   <scope>system</scope>
   <systemPath>${JAVA_HOME}/lib/tools.jar</systemPath>
  </dependency>

 </dependencies>

</project>
```

编写连接小工具

```
package top.xiesen.utils;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class HiveJdbcUtils {

 public static final String DRIVER_CLASS = "org.apache.hive.jdbc.HiveDriver";
 public static final String URL = "jdbc:hive2://master:10000/db14";
 public static final String USERNAME = "root";
 public static final String PASSWORD = "";
 private static Connection connection;
 /**
 * getConnection 获取连接
 * @param @return 参数
 * @return Connection 返回类型
 * @Exception 异常对象
 */
```

```
 public static Connection getConnection(){
  try {
   Class.forName(DRIVER_CLASS);
   connection = DriverManager.getConnection(URL,USERNAME,PASSWORD);
  } catch (Exception e) {
   e.printStackTrace();
  }
  return connection;
 }

 /**
 * close 关闭连接
 * @param @param connection 参数
 * @return void 返回类型
 * @Exception 异常对象
 */
 public static void close(Connection connection) {
  try {
   if (connection != null) {
    connection.close();
   }
  } catch (SQLException e) {
   e.printStackTrace();
  }
 }
}
```

hive的基本操作

创建表

```
public static void createTable() throws SQLException {
  Connection connection = HiveJdbcUtils.getConnection();
  Statement statement = connection.createStatement();
  StringBuilder stringBuilder = new StringBuilder();
  stringBuilder.append("create table table_form_java(");
  stringBuilder.append("test1 string");
  stringBuilder.append(",test2 int");
  stringBuilder.append(",test3 string");
  stringBuilder.append(")stored as textfile");
  statement.execute(stringBuilder.toString());
  ResultSet result = statement.executeQuery("show tables");
  while(result.next()){
   System.out.println(result.getString(1));
  }
 }
```

Hive 函数

hive 中内置了很多函数，具体使用用法详见官方文档https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF

- 查看所有函数 `show functions`
- 查看函数的具体用法 `describe function last_day` 或者 `describe function extended last_day`
- 返回当前月的最后一天

```
-- Returns the last day of the month
```

```
select last_day(to_date('2017-10-01'))
select last_day('2017-10-01')
```

- 字符串拼接

```
show tables;
select '姓名:' || emp_name || '\t薪水:'|| salary
from dw_employee

select concat(emp_name,':',salary)
from dw_employee

describe function extended concat
```

- hive支持正则表达式

```
describe function extended regexp
select * from dw_employee
where status regexp '(.{4})'
```

- 复杂类型构造方法

```
-- 复杂类型构造方法
select map(emp_name,status)
from dw_employee
```

数学函数

> 小数的数据类型 double float decimal numberic，一般我们使用decimal

- 四舍五入，参数1：小数；参数2：小数点后取的位数 `select round(1.222223,2);`
- 向下取整 `select floor(2.344)`
- 向上取整 `select ceil(2.344)`
- 获取随机值 `select rand();`
- 开平方 `select sqrt(4);`

日期类型函数

- 获取当前时间戳

```
select unix_timestamp();
select unix_timestamp('2017-01-02 00:00:00');
```

- 把字符串时间转换成时间戳,指定格式 `select unix_timestamp('2017-01-02 00:00:00','yyyy-MM-dd HH:mm:ss')`
- 时间戳转换成时间 `select from_unixtime(unix_timestamp(),'yyyyMMdd');`
- 把字符串转换成日期格式 yyyy-MM-dd HH:mm:ss `select to_date('2017-03-20 11:30:01');`
- 抽取日期的天数 `select extract(day from '2017-8-10')`
- 计算两个日期相隔天数 `select abs(datediff('2017-08-03','2017-09-02'));`
- 在一个日期上加天数 求新日期 `select date_add('2017-10-1',10);`

- 在一个日期上减天数 求新日期 `select date_add('2017-10-11',-10);select data_sub('2017-10-11',10);`
- 获取当前时间 `select current_date(); select current_timestamp();`
- 时间格式化

```
select date_format(current_date(),'yyyy--MM--dd');
select date_format(current_timestamp(),'yyyy--MM--dd');
select date-format('2017-10-11','yyyy--MM--dd');
```

条件函数

- if--else

```
select if(salary > 5000,'中等收入','低收入')
from dw_employee
```

- isnull

```
select emp_id
 ,emp_name
 ,isnull(status_salary)
from dw_employee
```

- isnotnull

```
select emp_id
 ,emp_name
 ,isnotnull(status_salary)
from dw_employee
```

- COALESCE 求非空

```
-- 取出每个人的上级id，如果没有上级部门id返回-1，如果有返回部门id
select emp_id
 ,emp_name
 ,COALESCE(leader_id,dep_id,-1)
from dw_employee
```

string函数

- 获取字符串长度`select character_length('2017')` 或者 `select length('2222');`
- 字符串转换成二进制 `select binary('你好');`
- 二进制转换成string,默认编码格式utf-8 `select decode(binary('你好'),'UTF-8'); select decode(binary('你好'),'GBK');`
- format_number对数字进行格式化，参数2是小数点后保留几位，返回值是字符串 `select format_number(1212.123,2);`
- lcase等同于lower `select lcase('asRfd');`
- trim ltrim rtrim 去除首尾空白字符

```
select trim(' aaa ');
```

```
select ltrim(' aaa ');
select rtrim(' aaa ');
```

- 正则表达式抽取 `select regexp_extract('张三:年龄19岁,email:xxx@163.com','.*(\\d+).*(\\w+@163.com)',1)`
- 正则表达式 `select regexp_replace('13522220064','^\\d{7}',"*******");`
- 替换电话号码 `select concat(substr('13134720265',1,3),'xxxx',substr('13134720265',8,4));`
- 字符串反转 `select reverse('abc');`

日志

日志信息结构分析

```
79.133.215.123 - - [14/Jun/2014:10:30:13 -0400] "GET /home HTTP/1.1" 200 1671 "-" "Moz
illa/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.19
16.153 Safari/537.36"
```

```
1. 用户访问的ip地址
2. 用户标识
3. 用户名
4. 访问时间
5. 请求信息(请求方法，请求url，协议类型)
6. 请求相应状态
7. 请求相应流量大小(byte)
8. 关联页面
9. 客户端的浏览器类型
```

> 对数据进行分析之后，发现使用基本数据类型很难将数据给分隔开来，因此，我们使用特殊的数据类型进行数据存储

创建日志对应的表

```
CREATE TABLE apachelog (
  host STRING,
  identity STRING,
  username STRING,
  time STRING,
  request STRING,
  status STRING,
  size STRING,
  referer STRING,
  agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  "input.regex" = "([^ ]*) ([^ ]*) ([^ ]*) (-|\\[[^\\]]*\\]) ([^ \"]*|\"[^\"]*\") (-|[
0-9]*) (-|[0-9]*)(?: ([^ \"]*|\"[^\"]*\") ([^ \"]*|\"[^\"]*\"))?"
  ,"output.format.string"="%1$s %2$s %3$s %4$s %5$s %6$s %7$s %8$s %9$s"
)
STORED AS TEXTFILE;
```

加载数据

```
load data inpath '/log' overwrite into table apachelog
select * from apachelog
describe formatted apachelog
```

计算当日网站的pv uv

```
-- 计算当日网站的pv uv
-- pv 用户的每个请求就是一个pv; uv(一个ip就是uv的数量)
select count(*) pv
 ,count(distinct host) uv
from apachelog
```

统计出网站用户访问使用windows系统和使用mac系统的占比和数量

> 这个题目相对比较复杂，我们先将windows，mac，orther用户打上标签，然后，将标记过的数据放到一张临时表中，我们队这个临时表进行统计计算

- 匹配不同的用户

```
select * from apachelog where agent rlike 'Windows NT'
select * from apachelog where agent rlike 'Mac OS'
```

- 创建临时表

```
create table tmp_user_sys
stored as orc
as
select host
 ,sys_type
from (
select host
 ,case
  when agent rlike 'Windows NT' then 'windows'
  when agent rlike 'Mac OS' then 'mac'
  else 'other'
 end sys_type
from apachelog
) a
group by host
 ,sys_type
```

- 通过创建的临时表，我们看到同一用户有两台电脑，这里我们把既有mac又有windows的用户看做是两个不同的用户

```
select count(1) p_num
 ,sum(case when sys_type='mac' then 1 else 0 end) mac_num
 ,sum(case when sys_type='windows' then 1 else 0 end) windows_num
 ,sum(case when sys_type='other' then 1 else 0 end) other_num
 ,sum(case when sys_type='mac' then 1 else 0 end)/count(1) mac_rate
 ,sum(case when sys_type='windows' then 1 else 0 end)/count(1) windows_rate
```

```
 ,sum(case when sys_type='other' then 1 else 0 end)/count(1) other_rate
from tmp_user_sys
```

自定义hive function

> 自定义函数，我们以时间格式转换为例

1. 编写时间转换格式代码

```
package top.xiesen.udf;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

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

/**
* 项目名称：udftest
* 类名称：LogDateConvert [14/Jun/2014:10:30:13 -0400] --》 2014-06-14 22:30:13
* 类描述：自定义udf需要继承UDF类，实现evaluate方法
* 返回值类型上，可以是java基础类型、writeable子类、string
* 创建人：Allen
* @version
*/
public class LogDateConvert extends UDF{
 public static final SimpleDateFormat FORMAT = new SimpleDateFormat("yyyy-MM-dd HH:mm:
ss");
 // 输出数据的日期格式是带有local和时区的，local是英文，可以使用英文格式进行匹配转换
 public static final SimpleDateFormat SRCFORMAT = new SimpleDateFormat("[dd/MMM/yyyy:H
H:mm:ss Z]",Locale.ENGLISH);


 /**
 * evaluate 将[14/Jun/2014:10:30:13 -0400] 格式转换成  2014-06-14 22:30:13
 * @param @param datestr [14/Jun/2014:10:30:13 -0400]格式的字符串
 * @return String 返回类型 ，可以是java基础类型、writeable子类、string
 * @Exception 异常对象
 */
 public String evaluate(String datestr){
  try {
   Date olDate = SRCFORMAT.parse(datestr);
   return FORMAT.format(olDate);
  } catch (ParseException e) {
   e.printStackTrace();
   return null;
  }
 }

 public static void main(String[] args) {
  LogDateConvert logDateConvert = new LogDateConvert();
  System.out.println(logDateConvert.evaluate("[14/Jun/2014:10:30:13 -0400]"));
 }
}
```

2. 将编写的代码，打成jar包，上传到linux环境下

```
add jar /root/udf.jar;
list jars;
```

3. 将jar包加载到hive上，创建function `create function logdate_convert as 'top.xiesen.udf.LogDateConvert';`
4. 使用自定义函数

```
-- 使用函数
create table hour_pvuv
stored as orc
as
select b.hour
 ,count(1) pv
 ,count(distinct b.host) uv
from (select hour(logdate_convert(time)) hour
 ,a.*
from apachelog a) b
group by b.hour
```

异常处理

1. 在创建hive function时出现 ClassNotFoundException 异常信息
   解决方案：将jar包上传到hdfs上，再执行添加操作

2. 在查询结果信息时，出现查询结果不一致的现象
   解决方案：出现这种现象的原因是虚拟机上linux的时间是英国的时区，老师的系统将时区修改为东八区了，我们可以将linux系统上的时区修改一下，也可以将我们的代码修改了。出现这个问题，给我们的程序没有关系，也可以选择沉默。