**17. Hive统计**

**一、基础概述**

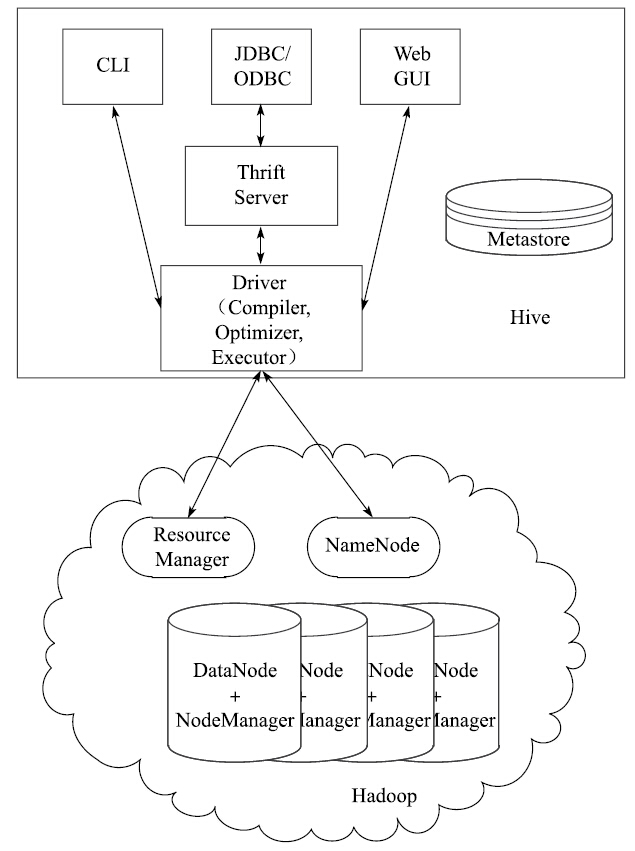
Hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供简单的SQL查询功能，可以将SQL语句转换为MapReduce任务进行运行。

其优点是学习成本低，可以通过类SQL语句快速实现简单的MapReduce统计，不必开发专门的MapReduce应用，十分适合数据仓库的统计分析。

Hive构建在基于静态批处理的Hadoop之上，由于Hadoop通常都有较高的延迟并且在作业提交和调度的时候需要大量的开销。因此，Hive并不适合那些需要低延迟的应用，它最适合应用在基于大量不可变数据的批处理作业，例如，网络日志分析。

Hive的特点是：可伸缩（在Hadoop集群上动态的添加设备）、可扩展、容错、输出格式的松散耦合。

Hive将元数据存储在关系型数据库（RDBMS）中，比如MySQL、Derby中。

[](http://192.168.1.241/doc/exper/127/img/01.png)

**二、需求概述**

在本节中，我们将使用Hive对以下指标进行统计并演示：

1. 移动端和PC端，用户比例

2. 用户评论周期（收到货后，一般多久进行评论）

3. 会员级别统计（判断购买此商品的用户级别）

4. 每天评论量（大体能反映出下单时间）

5. 自定义UDF，功能为：去掉评论时间的时分秒，只保留年月日

**三、实验步骤**

1. 获取本次实验使用的数据：

[hfut@master ~]$ cp ~/resources/hive-data/pinglun ~/hive-data/

2. 初始化操作

启动Hadoop；启动MySQL服务；启动Hive。

3. 在Hive创建一张表，用于存放清洗后的数据，表名为pinglun，字段名、字符类型、字段解释如下：

1. productid string 产品ID

2. commentcount int 评论数

3. goodcount int 好评数

4. generalcount int 中评数

5. poorcount int 差评数

6. goodrateshow float 好评率

7. generalrateshow float 中评率

8. poorrateshow float 差评率

9. guid string 随机生成ID

10. content string 评论内容

11. creationtime string 写评论的时间

12. score int 打分

13. nickname string 昵称

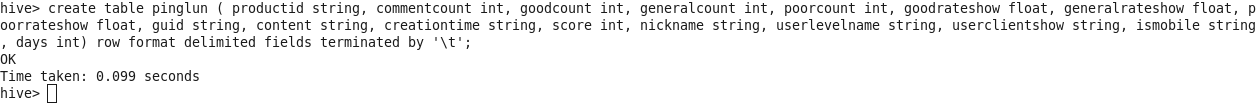
14. userlevelname string 会员级别

15. userclientshow string 评论设备

16. ismobile string 是否移动端

17. days int 评论时间距【收货/下单】时间多长时间

hive> create table pinglun ( productid string, commentcount int, goodcount int, generalcount int, poorcount int, goodrateshow float, generalrateshow float, poorrateshow float, guid string, content string, creationtime string, score int, nickname string, userlevelname string, userclientshow string, ismobile string, days int) row format delimited fields terminated by '\t';



创建成功后，查看pinglun表的表结构：



当然，也可以创建外部表：

hive> create external table if not exists pinglun ( productid string, commentcount int, goodcount int, generalcount int, poorcount int, goodrateshow float, generalrateshow float, poorrateshow float, guid string, content string, creationtime string, score int, nickname string, userlevelname string, userclientshow string, ismobile string, days int) row format delimited fields terminated by '\t';

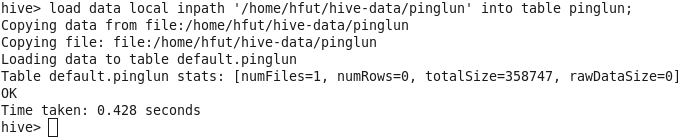
外部表的创建方法比内部表多了一个external，同时还加上了if判断，判断创建表之前，是否存在同样名称表。

Hive创建内部表时，会将数据移动到数据仓库指向的路径；创建外部表时，仅记录数据所在的路径， 不对数据的位置做任何改变。

在删除表的时候，内部表的元数据和数据会被一起删除， 而外部表只删除元数据，不删除数据。这样外部表相对来说更加安全些，数据组织也更加灵活，方便共享源数据，生产中常使用外部表。

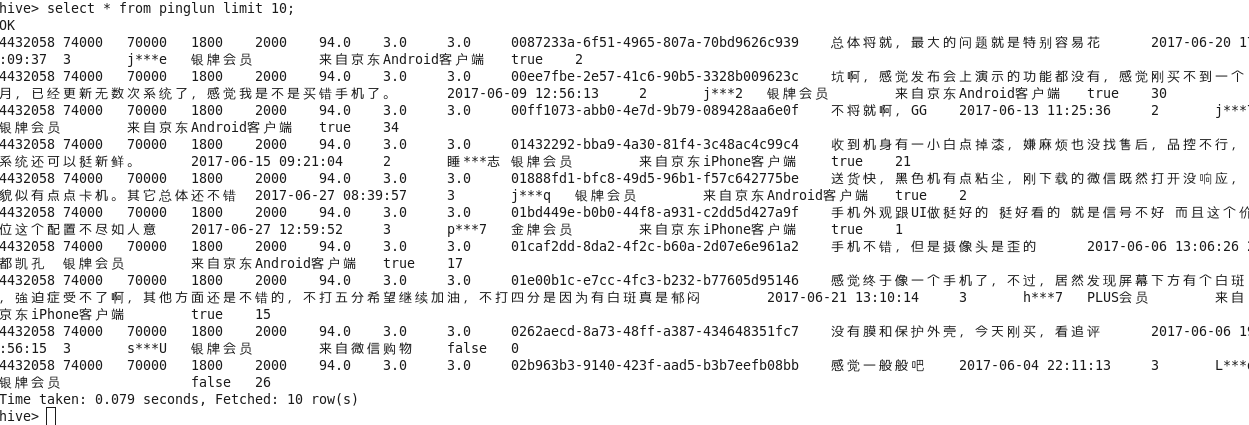
4. 表设计好以后，在Hive端使用load命令，将/home/hfut/hive-data下的pinglun导入Hive表中。

hive> load data local inpath '/home/hfut/hive-data/pinglun' into table pinglun;



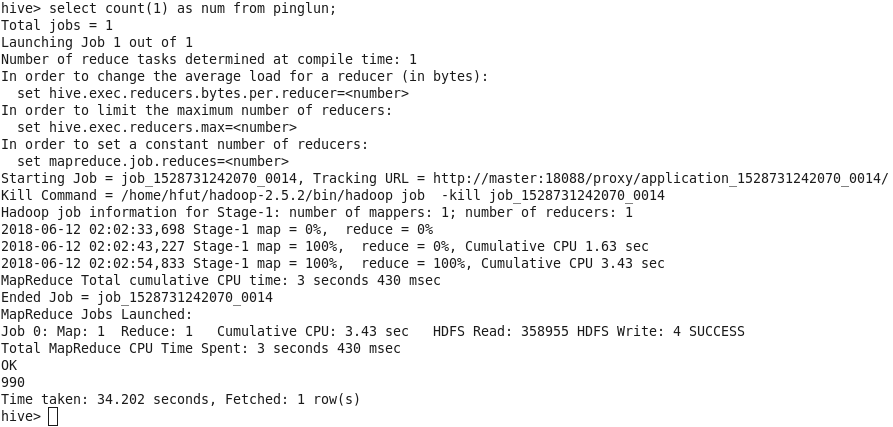
在hive中，执行查询操作，验证数据是否导入成功。

hive> select \* from pinglun limit 10;



查看数据条数

hive> select count(1) as num from pinglun;



**四、需求1：分析用户使用移动端购买还是PC端购买，及移动端和PC端的用户比例**

HQL语句：

select

case

when ismobile='true' then 1

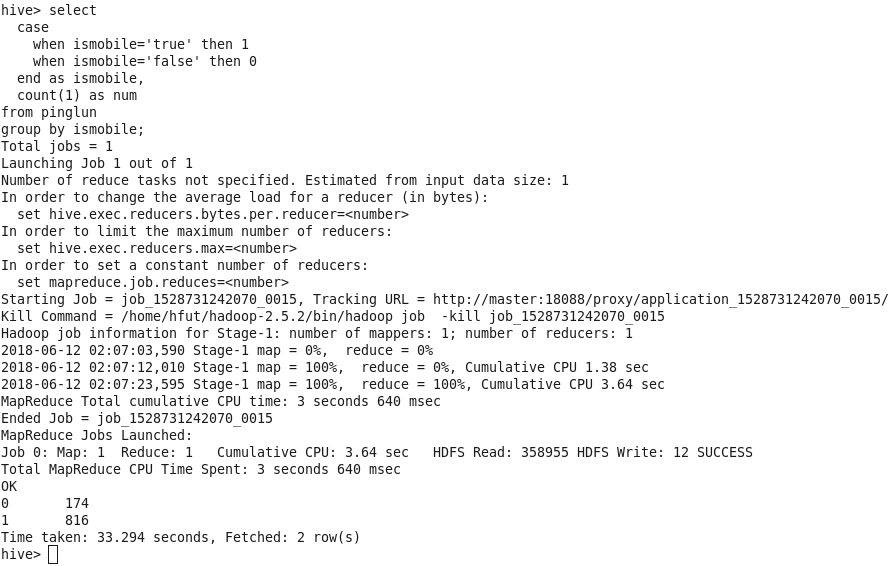
when ismobile='false' then 0

end as ismobile,

count(1) as num

from pinglun

group by ismobile;



其中1代表移动端购买，共816人，0代表PC端购买，共174人。

**五、需求2：分析用户评论周期（收到货后，一般多久进行评论）**

HQL语句：

select

days,

count(1) as num

from pinglun

group by days

order by num desc;



通过分析结果，我们可以清楚地了解到收到货后第一天，评论119人，第二天2天评论116人，3天评论的有90人，等等...

**六、需求3：分析会员级别（判断购买此商品的用户级别）**

HQL语句：

select

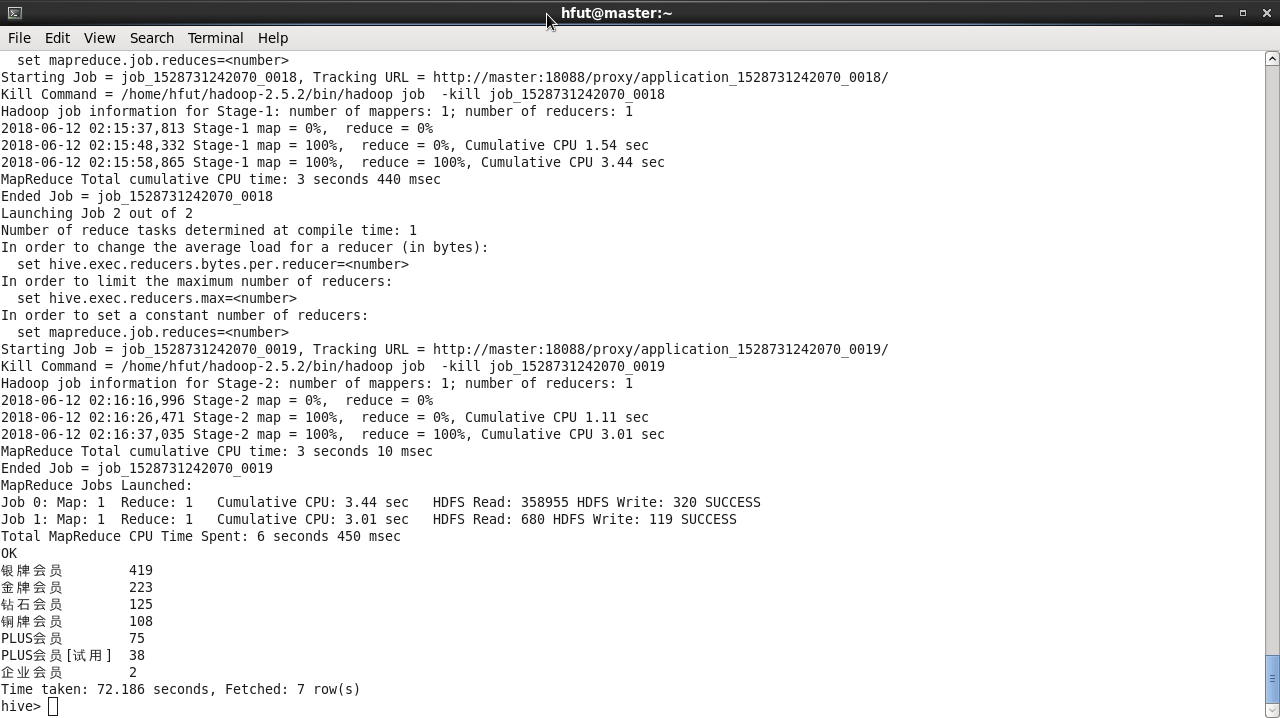
userlevelname,

count(1) as num

from pinglun

group by userlevelname

order by num desc;



通过分析结果，我们可以看到购买用户会员级别为银牌的最多，有419人，金牌会员有223人，钻石会员125人等。

具体数值，会根据采集来的数据的变化而变化。时间和商品不同，会导致数值有波动。

**七、需求4：分析每天评论量**

HQL语句：

select

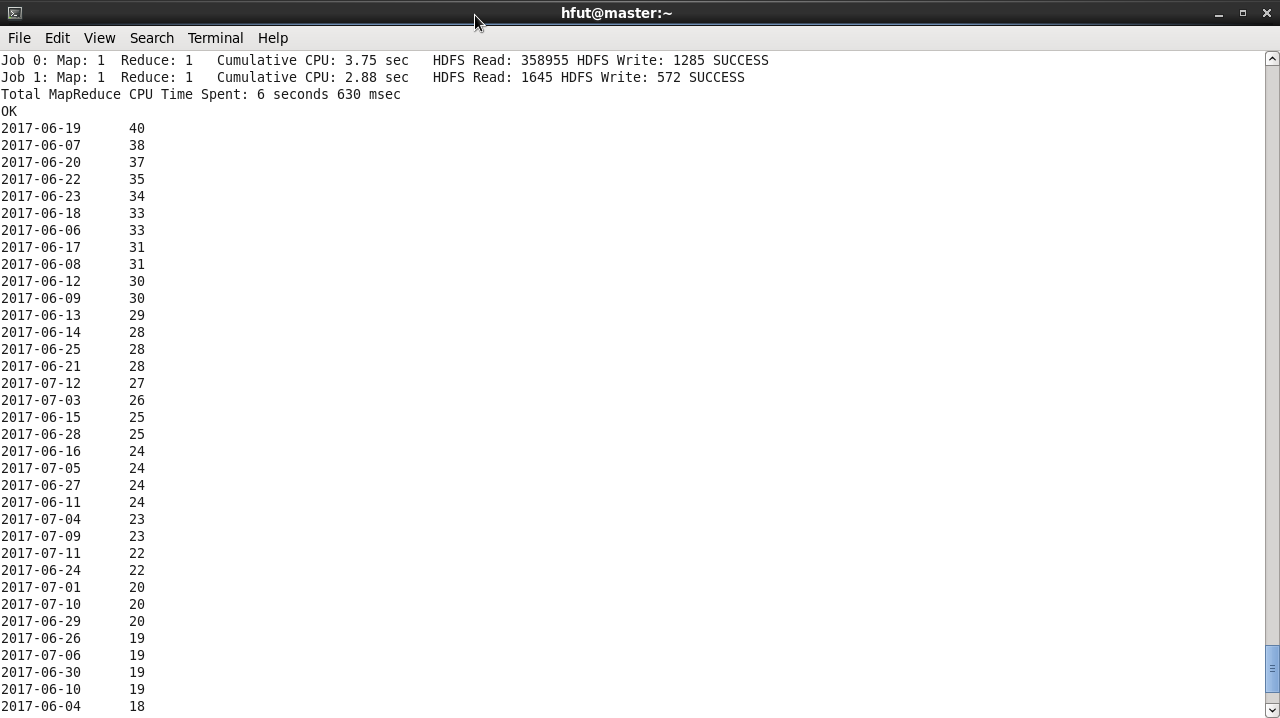
substr(creationtime, 0, 10) as dt,

count(1) as num

from pinglun

group by substr(creationtime, 0, 10)

order by num desc;

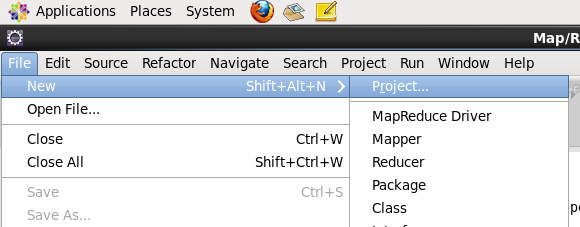


**八、使用UDF**

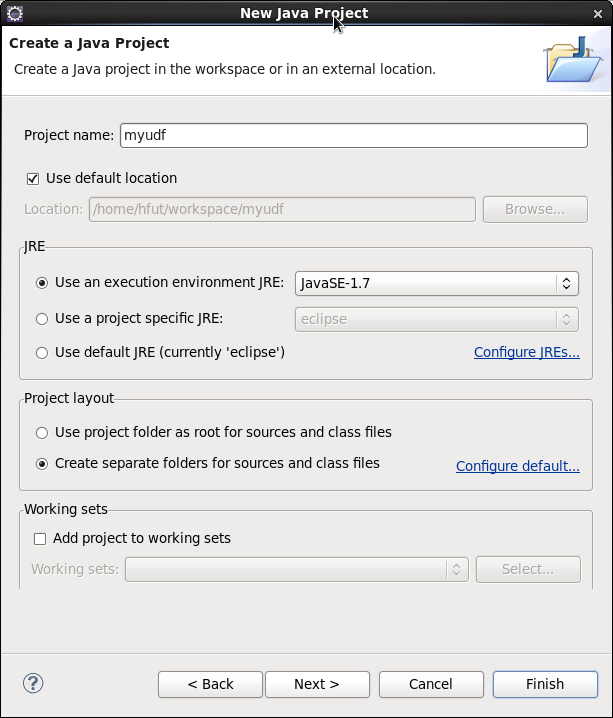
在需求4中，对日期的处理，我们使用了Hive中自带的，截取字符串的函数substr。有时这些函数功能较弱，需要增强。所以我们可以进行自定义。下面编写自定义函数，执行数据处理。这种函数叫UDF（User Defined Function）

下面，使用另一种方式，来处理需求4中的日期。

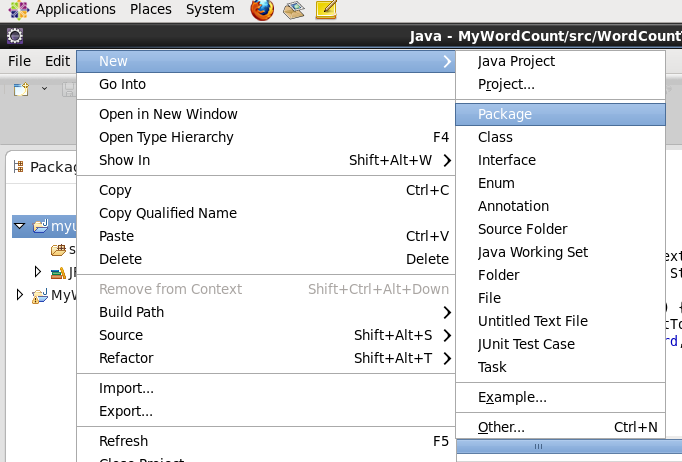
打开eclipse，创建Java项目



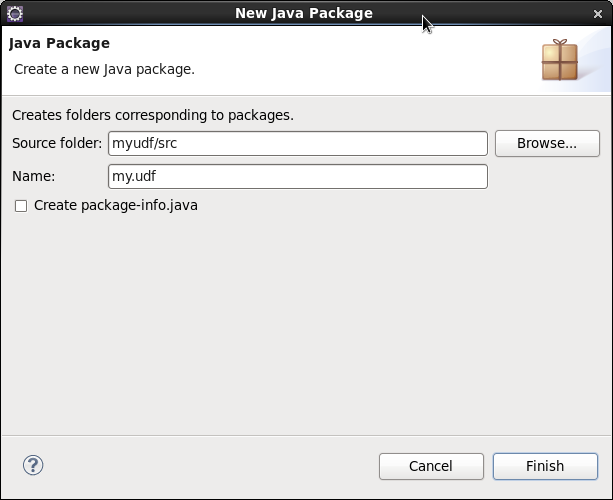
将项目命名为myudf



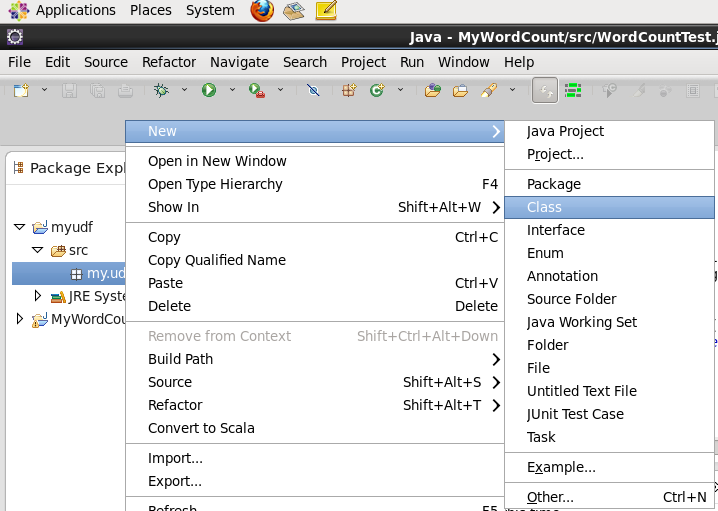
选中项目名myudf，右键，依次点击New->Package，创建包



将包命名为my.udf



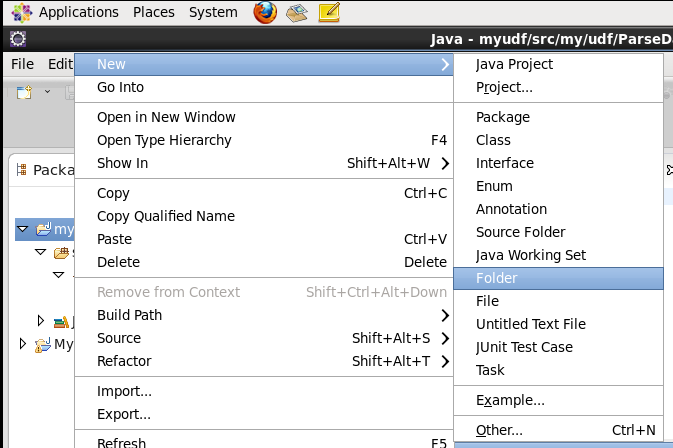
选中包my.udf，右键依次点击New->Class创建类



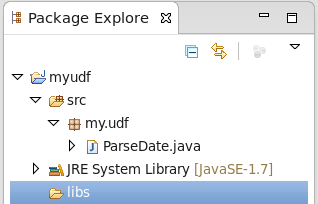
将类命名为ParseDate



选中项目名myudf，右键依次点击New->Folder，创建目录，并将目录命名为libs。用于存放项目所依赖的jar文件



最终项目框架，如下：



打开桌面终端模拟器，进入命令行，复制~/resources/software/hadoop/apache/hive-udf-libs.tar.gz到~/hive-data，hive-udf-libs.tar.gz为本次实验所属的包。

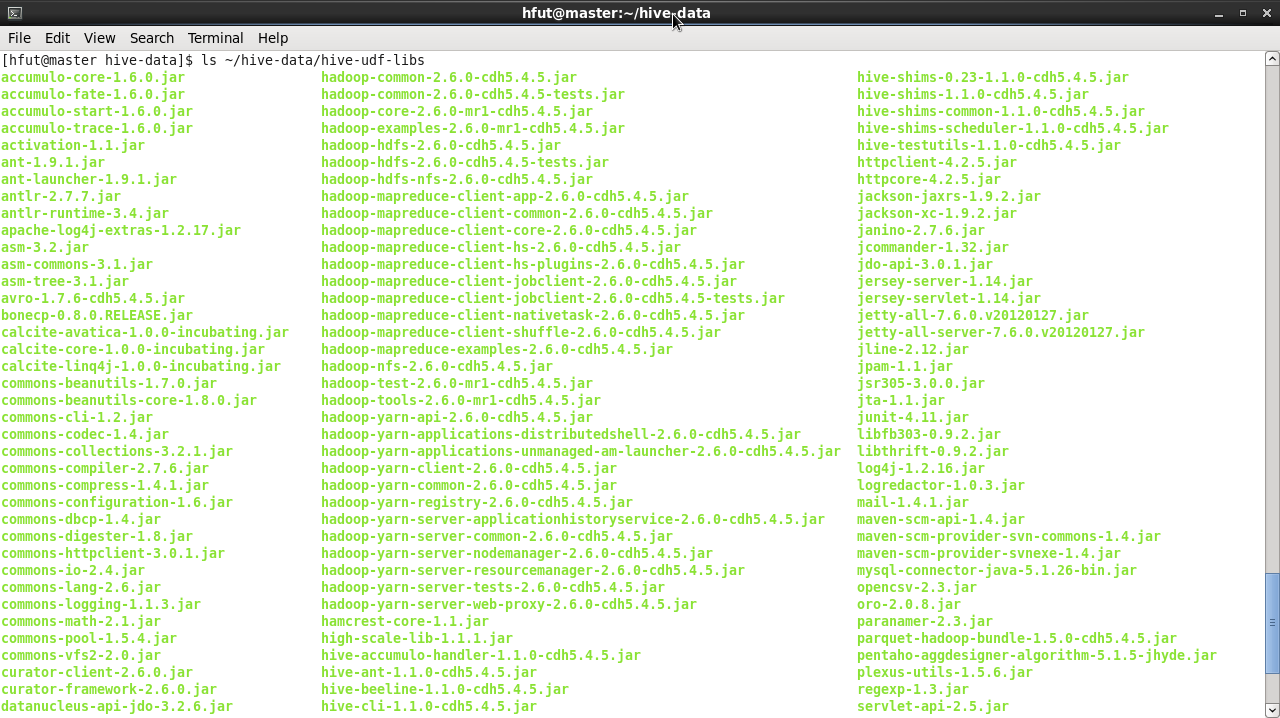
[hfut@master ~]$ cp ~/resources/software/hadoop/apache/hive-udf-libs.tar.gz ~/hive-data

将hive-udf-libs.tar.gz进行解压，并查看解压后的目录：

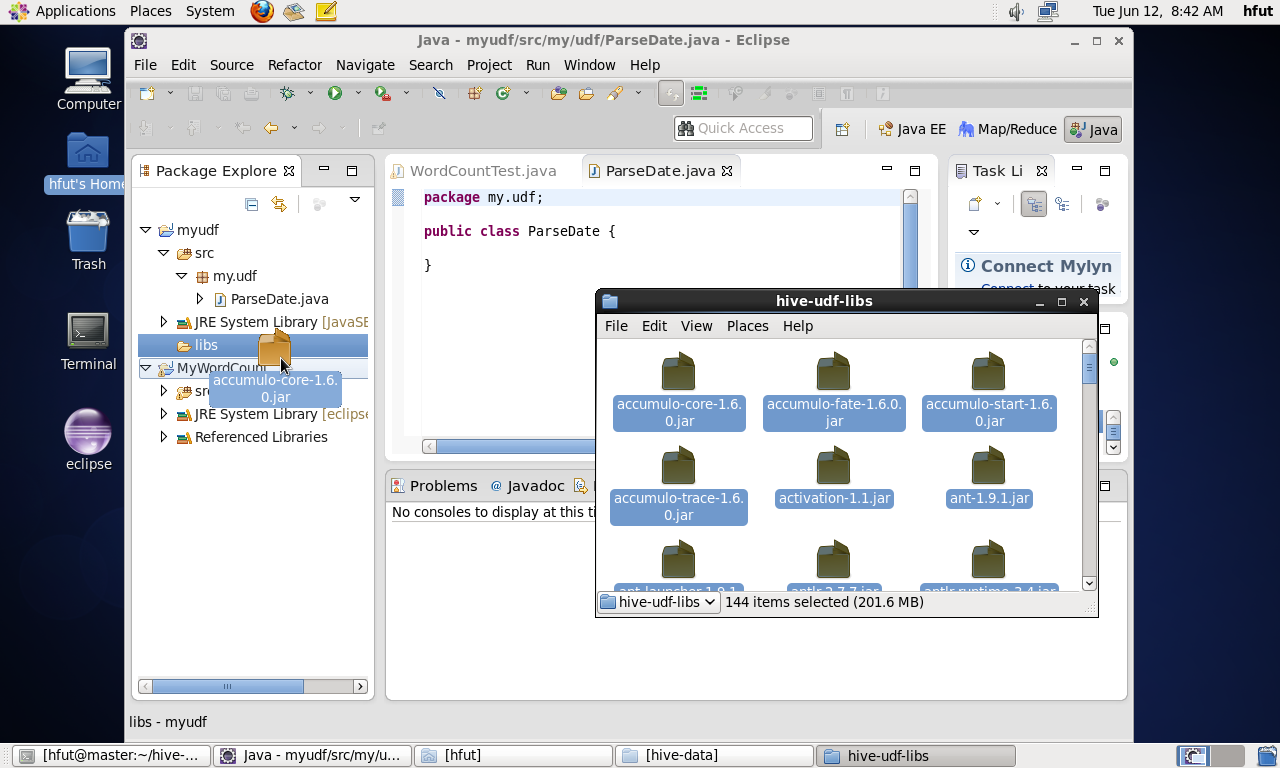
[hfut@master ~]$ cd hive-data/

[hfut@master hive-data]$ tar -zxvf hive-udf-libs.tar.gz

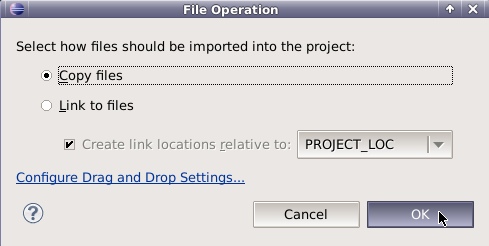
[hfut@master hive-data]$ ls ~/hive-data/hive-udf-libs



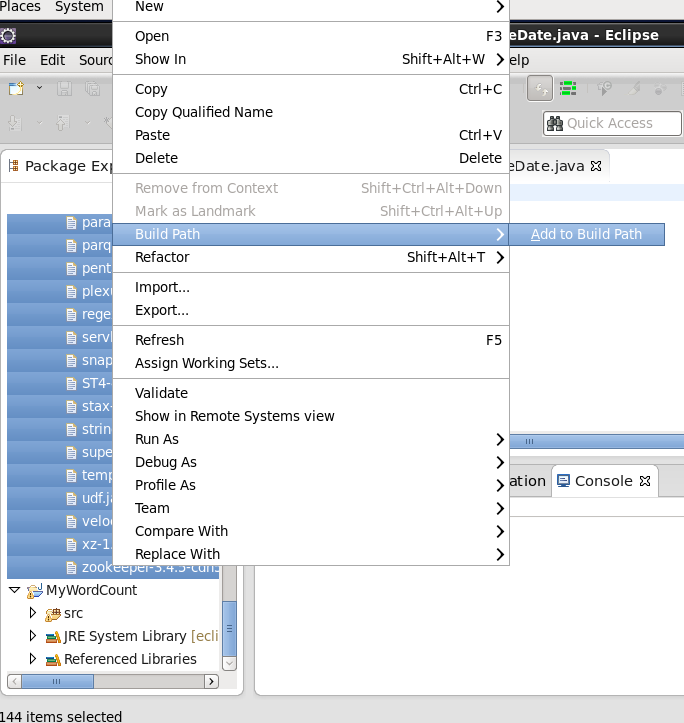
将~/hive-data/hive-udf-libs目录下所有jar包，导入拷贝到myud3项目的libs目录下



点击OK

[](http://192.168.1.241/doc/exper/127/img/24.png)

导入后，选中lib中的所有文件，单击右键并依次选择Build Path->Add to Build Path。



编写ParseData类中编写代码，实现UDF。要想自定义函数，需要使ParseData类继承UDF类，并重构evaluate函数即可。

package my.udf;

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

public class ParseDate extends UDF {

public String evaluate(String createiontime){

return null;

}

}

在这里evaluate函数，要实现的功能，就是对“2017-06-20 17:09:37”格式的日期数据，进行处理最终只保留日期部分“2017-06-20”。

public String evaluate(String createiontime) throws ParseException{

DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

Date dt = dateFormat.parse( createiontime );

return dateFormat.format(dt);

}

上面这段日期转换的代码很简单，我们也可以放到main函数中，进行测试

public static void main(String[] args) throws ParseException {

String dtString = "2017-1-1 12:10:10";

DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

Date dt = dateFormat.parse(dtString);

String result = dateFormat.format(dt);

System.out.println(result);

}

执行结果，最终会得到2017-01-01这样的结果

UDF的完整代码如下：

package my.udf;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

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

public class ParseDate extends UDF {

public String evaluate(String createiontime) throws ParseException{

DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

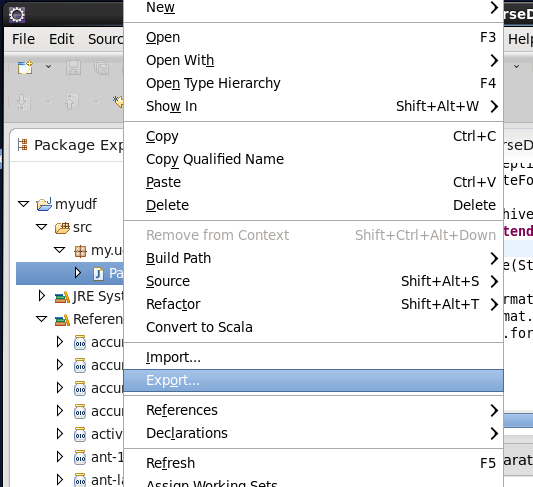
Date dt = dateFormat.parse( createiontime );

return dateFormat.format(dt);

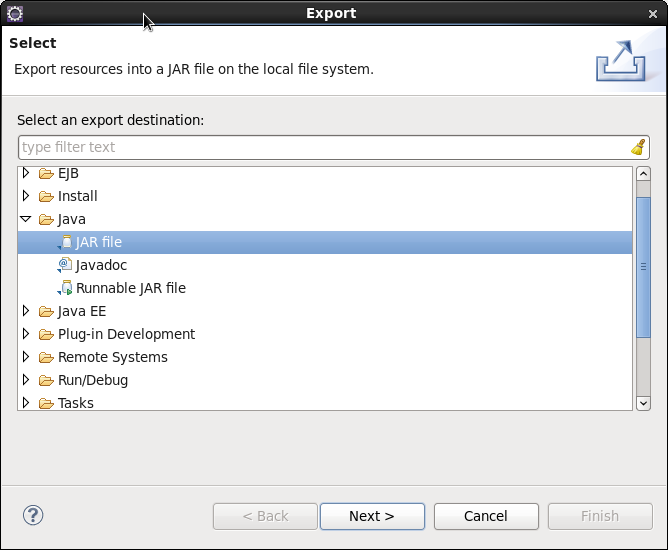
}

}

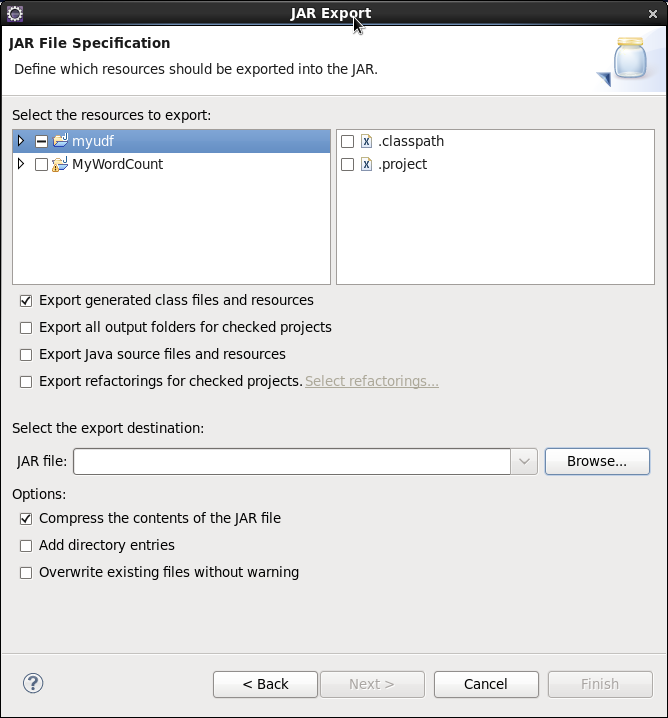
将我们自定义的UDF打包成Jar文件。右键类文件，点击Export



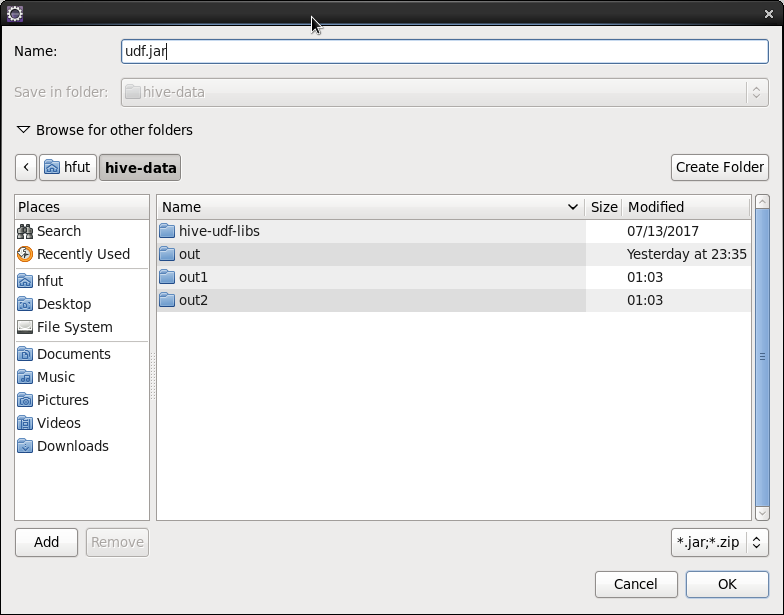
在弹出框中输入jar，在列出的可选项中，选择JAR file



在弹出的窗口中，点击Browser，设置要导出jar文件的位置



将Jar文件导出到/hive-data



执行测试。

重启hive命令行终端，在命令行下，直接输入命令

hive> add jar /home/hfut/hive-data/udf.jar;

这句话的意思是，将编写的自定义函数分发到集群中去。相比于第一种，不用改变集群环境。

剩余步骤与第一种方式相同。

直接输入命令，创建临时方法

hive> create temporary function udf as 'my.udf.ParseDate';

再执行需求4中查询命令

select

udf(creationtime) as dt,

count(1) as num

from pinglun

group by udf(creationtime)

order by num desc;

执行测试后，结果为：

