目录

[1. 实验一：大数据平台业务处理系统应用 2](#_Toc11003)

[1.1. 实验目的 2](#_Toc876)

[1.2. 实验要求 2](#_Toc11824)

[1.3. 实验环境 2](#_Toc25040)

[1.4. 实验过程 2](#_Toc31385)

[1.4.1. 实验任务一：生产环境数据导入大数据平台 2](#_Toc26354)

[1.4.2. 实验任务二：业务处理层数据清洗分析 4](#_Toc18615)

[1.4.3. 实验任务三：运行业务处理相关清洗任务 13](#_Toc3630)

[1.4.4. 实验任务四：数据进入数据仓库 18](#_Toc8167)

[1.4.5. 实验任务五：业务应用层大数据分析 20](#_Toc7441)

[1.4.6. 实验任务六：数据仓库数据导出数据访问层 22](#_Toc17539)

[1.4.7. 实验任务七：数据访问层大数据可视化 24](#_Toc3976)

# 实验一：大数据平台业务处理系统应用

## 实验目的

完成本实验，您应该能够：

* 熟悉MapReduce数据清洗的一般流程
* 熟悉Hive的基本语法
* 熟悉Sqoop数据传输的基本语法
* 熟悉大数据业务处理的全流程

## 实验要求

* 熟悉常用Linux操作系统命令
* 熟悉Hadoop操作命令
* 了解Sqoop运行原理

## 实验环境

本实验所需之主要资源环境如表1-1所示。

表1-1 资源环境

|  |  |
| --- | --- |
| 服务器集群 | 3个以上节点，节点间网络互通，各节点最低配置：双核CPU、8GB内存、100G硬盘 |
| 运行环境 | CentOS 7.4 |
| 大数据平台 | H3C教学与实践管理系统 |
| 服务和组件 | 完成前面章节的实验，其他服务及组件根据实验需求安装 |

## 实验过程

### 实验任务一：生产环境数据导入大数据平台

本教材用到的数据源来自于实际的生产环境的MySQL数据

#### 步骤一：上传数据源到大数据平台

数据源是来自于招聘网站的初始数据，并且数据都存放在实际生产环境的MySQL数据库中。存储在MySQL的数据源查看可知字段介绍如表17-1所示。

表17-1 数据源

|  |  |
| --- | --- |
| 字段 | 中文解析 |
| com\_name | 公司名称 |
| city | 工作城市 |
| j\_require | 工作要求 |
| number\_recruit | 招聘人数 |
| salary\_up | 工资上限 |
| salary\_down | 工资下限 |
| skill | 工作技术 |
| release\_time | 发布时间 |
| sex | 性别 |
| company\_detail | 公司描述 |
| edu | 学历要求 |
| j\_name | 岗位名称 |

在实际的生产环境中，业务处理层的操作是结合大数据平台进行操作运行的，存放在MySQL的数据不利于后续的业务处理层（即数据清洗、分析、挖掘）的使用，所以需要使用Sqoop从MySQL导入数据到HDFS。如图17-2所示。

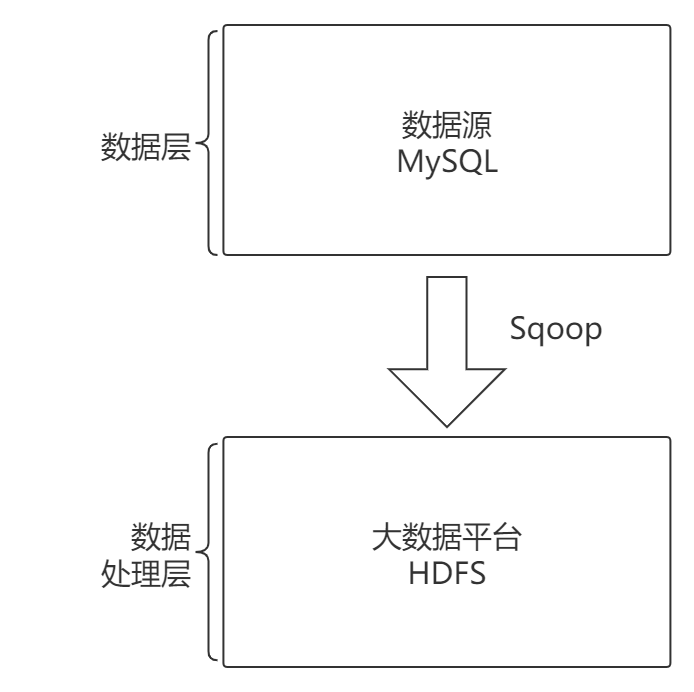


图17-2 数据源层与数据处理层进行数据传输

使用Sqoop命令，将数据库中表数据导出到HDFS，具体参数如表17-2所示。

表17-2 Sqoop导出数据的参数

|  |  |
| --- | --- |
| 数据库名 | recruit |
| 数据库表 | data |
| HDFS路径 | /recruitdata |

导入完成后，使用Hadoop命令查看数据源的行数，命令如下。

[hadoop@master ~]$ hdfs dfs –mkdir /recruitdata

[hadoop@master ~]$ hdfs dfs –put /opt/data/recruit.json /recruitdata

[hadoop@master ~]$ hdfs dfs -cat /recruitdata/\* | wc -l

**123711**

### 实验任务二：业务处理层数据清洗分析

本章节使用的招聘网站初始数据集来自于生产环境，因此数据集中不可避免地存在一些脏数据，即源数据不在给定的范围内或对于实际业务毫无意义，或是数据格式非法，以及在源系统中存在不规范的编码和含糊的业务逻辑。

通过使用MapReduce实现业务数据在Hadoop平台离线清洗处理，将结果保存到HDFS，本小节主要是介绍具体代码，如图17-3所示。

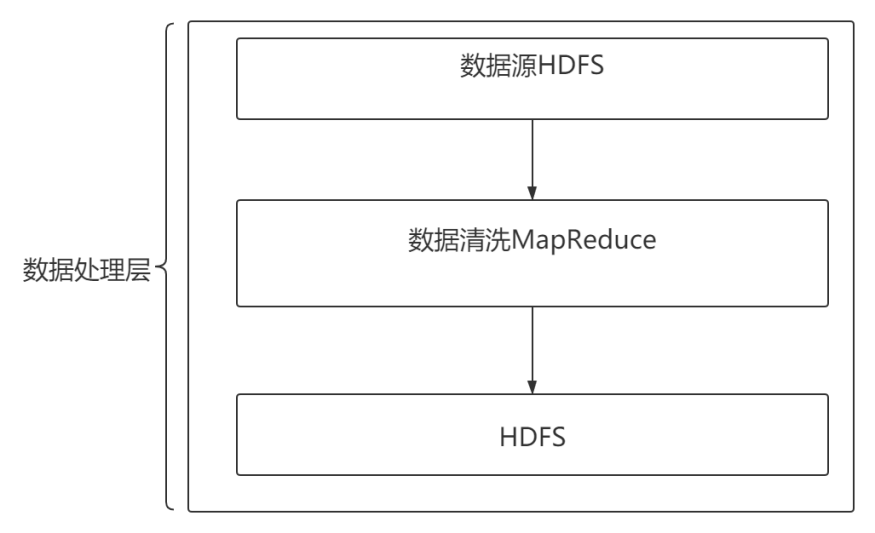


图17-3 业务处理层数据清洗分析

#### 步骤一：剔除不合规数据

对于原始数据集字段缺失，会采用填充默认值、均值、众数、KNN填充、以及把缺失值作为新的Label方式处理，当缺失信息较少时可采用删除的方式进行处理。同时，不当的填充可能会令后续的分析结果出现导向性偏差，所以需要对数据业务逻辑进行全面分析后，确定不合规数据处理方式。

已经上传到HDFS的/recruitdata/清洗前的数据源样式如下:

[{"name": "\*\*\*", "detail": {"公司名称": "\*\*\*", "工作城市": "\*\*\*", "工作要求": "\*\*\*", "招聘人数": \*\*, "薪资上限": "\*\*\*", "薪资下限": "\*\*\*", "工作技术": \*\*\*, "发布时间": "\*\*\*", "招聘性别": "\*", "公司描述": "\*\*\*", "学历要求": " \*\*\*"}}]

完成去除字段中特殊符号清洗任务执行后，数据样式如下所示：

岗位名称|公司名称|工作城市|工作要求|招聘人数|薪资上限|薪资下限|工作技术|发布时间|招聘性别|公司描述|学历要求

（1）编辑Recruit MR/src/com/org/task1/ReportDelete.java文件，在枚举类中添加MapReduce自定义计数器，用于计算被删除的数据条数，在ReportDelete枚举类中添加如下代码：

Delete\_number

（2）编辑Recruit MR/src/com/org/task1/CleanMapper.java文件，编写MapReduce数据预处理Map程将下面的每个代码片添加到map方法中。

解析该Json文件，字段间以“|”分隔，并存放在数组中，并定义存放字段数据的数组。

String rawJson = value.toString().replaceAll("\\\\r|\\\\n", "");

String[] datas = new String[12];

去除字段信息中特殊符号，保留有效信息。通过查看json可知，如果读取的行包含[]，或者读取的行为空，则不做任何处理；由于上一条语句可能会造成读取的行数据末尾缺少一个},所以需确认读取的行结尾是否}}结尾，即是否符合一条json数据格式，如不符合则添加多一个“}”。这里使用外部Jar包的JSONObject对象，解析json数据，提取相应的字段。

if(rawJson.contains("[")||rawJson.contains("]")||rawJson==null||rawJson.equals("")) {

return;

}

//删除读取每行最后的逗号

rawJson = rawJson.substring(0, rawJson.length()-1);

if(!rawJson.substring(rawJson.length()-2, rawJson.length()).equals("}}")) {

rawJson = rawJson+"}";

}

//json解析行数据

JSONObject jsonObject = JSONObject.parseObject(rawJson);

//提取name字段的数据

String name = jsonObject.getString("name");

//判空

if(name == null||name.equals("")) {

return;

}

//提取detail字段

String values = jsonObject.getString("detail");

//detail字段的值又是一条需要解析的json数据，再用json解析

JSONObject jsonObject1 = JSONObject.parseObject(values);

//提取相应的字段值，并赋值给datas数组

String companyname = jsonObject1.getString("公司名称");

String city = jsonObject1.getString("工作城市");

String require = jsonObject1.getString("工作要求");

String man = jsonObject1.getString("招聘人数");

String cost\_max = jsonObject1.getString("薪资上限");

String cost\_min = jsonObject1.getString("薪资下限");

String skill = jsonObject1.getString("工作技术");

String time = jsonObject1.getString("发布时间");

String sex = jsonObject1.getString("招聘性别");

String introduce = jsonObject1.getString("公司描述");

String need = jsonObject1.getString("学历要求");

// 将每字段的数据存放在对应的数据位置，例如：公司名称的字段数据存放在数组的第一个位置

datas[0] = name;

datas[1] = companyname;

datas[2] = city;

datas[3] = require;

datas[4] = man;

datas[5] = cost\_max;

datas[6] = cost\_min;

datas[7] = skill;

datas[8] = time;

datas[9] = sex;

datas[10] = introduce;

datas[11] = need;

//遍历判空，for循环遍历数据，通过return完成数据数据的剔除操作

for(String i : datas) {

if(i==null||i.equals("")) {

return;

}

}

剔除不合规“工资”数据，并在控制台输出删除记录的条数。薪资字段如果出现-，则表示薪资存在负值，需要剔除,并使计数器+1；将工作技术的“【】”剔除。

//以下为剔除不合规的工资数据

if(datas[5].contains("-")||datas[6].contains("-")) {

context.getCounter(ReportDelete.Delete\_number).increment(1);

return;

}

//将工作技术的【】剔除

datas[7] = datas[7].replaceAll("【|】", "");

//给数据添加|分隔符

String result = "";

for(String i : datas) {

if(i.equals(datas[datas.length-1])) {

result = result+i.trim();

}else {

result = result+i.trim()+"|";

}

}

//输出数据

context.write(key, new Text(result));

（3）编辑Recruit MR/src/com/org/task1/CleanReducer.java文件，根据MapReduce规则，返回的Key-values的value值为一个Iterator的对象，编写MapReduce的Reduce程序在reduce方法中添加如下代码，对reduce任务不做任何操作，直接输出结果。

for(Text t : values) {

context.write(NullWritable.get(), t);

}

（4）编辑Recruit MR/src/com/org/task1/CleanJob.java文件，编写MapReduce启动程序将下面的所有代码片添加到main方法中。初始化Hadoop集群的java配置对象，设置Job类，Job类是java调用MapReduce的java对象，其中setJarByClass方法的参数值需要给到main方法所在的类。

// TODO Auto-generated method stub

Configuration conf = new Configuration();

Job job = Job.getInstance(conf);

//设置job类

job.setJarByClass(CleanJob.class);

分别设置MapReduce的Map类和Reduce类，以及输入输出数据的数据类型，分别完成两个过程的业务逻辑。

//设置map和reduce的类

job.setMapperClass(CleanMapper.class);

job.setReducerClass(CleanReducer.class);

//设置map输出key，value类型

job.setMapOutputKeyClass(LongWritable.class);

job.setMapOutputValueClass(Text.class);

//设置reduce输出的key，value类型

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

设置数据源输入输出路径，这里设置数据源文件存放于HDFS的recruitdata目录，将结果输出至HDFS的recruitoutput1目录。

FileInputFormat.addInputPath(job, new Path("/recruitdata"));

//设置输出的hdfs路径，注意：hdfs不能存在输出的文件夹，mapreduce会自动创建

FileOutputFormat.setOutputPath(job, new Path("/recruitoutput1"));

System.exit(job.waitForCompletion(true)?0:1);

Counters counters = job.getCounters();

System.out.println("counters getGroupNames:"+counters.getGroupNames());

#### 步骤二：清洗字段数据

数据源中的“薪资”字段的数据，薪资的上限和下限是分开的两个字段，以及薪资存在不规范，例如负数或者极大极小的数据，因此需要处理工资字段不合规数据，合并上限下限薪资数据格式，使该字段数据格式统一，为了后续方便查看以及统计分析。将清洗后的数据存入指定数据表或数据文件中。

实则将薪资的下限与上限进行了字段的合并，清洗前其中一条数据：

Java工程师|阿里巴巴|北京|熟练使用RPC框架，具备相关的分布式开发经验|3|12000|7000|Spark, HBase, Linux, Python|2019-03-28|女|阿里巴巴网络技术有限公司（简称：阿里巴巴集团）是以曾担任英语教师的马云为首的18人于1999年在浙江杭州创立|专科及其以上

完成清洗任务,合并薪资上下限执行后，数据样式如下所示：

Java工程师|阿里巴巴|北京|熟练使用RPC框架，具备相关的分布式开发经验|3|7000-12000|Spark, HBase, Linux, Python|2019-03-28|女|阿里巴巴网络技术有限公司（简称：阿里巴巴集团）是以曾担任英语教师的马云为首的18人于1999年在浙江杭州创立|专科及其以上

（1）编辑Recruit MR/src/com/org/task2/ReportUpdata.java文件,在枚举类中添加MapReduce自定义计数器，Upadata\_number用于计算需要被修改的薪资数据的总数，error\_lines用户统计薪资字段数据不规范的数据条数，error\_line2用于统计数据中不包含“|”的总数。

Updata\_number,error\_lines,error\_lines2

（2）编辑Recruit MR/src/com/org/task2/CleanMapper.java文件，编写MapReduce数据预处理Map程序，将下面的所有代码片添加到main方法中。根据上题清洗结果，请编写MapReduce程序，分析“工资”数据格式，读取数据源并根据规则进行字段分割。

String raw = value.toString();

String[] datas = raw.split("\\|");

将工资格式统一为“下限-上限”的形式，并在控制台输出需要调整次序的记录条数。

String cost = "";

if(!raw.contains("|")) {

context.getCounter(ReportUpdata.error\_lines2).increment(1);

}

//判断薪资的大小，组成 下限-上限 的格式

if(datas.length!=12) {

context.getCounter(ReportUpdata.error\_lines).increment(1);

return;

}

if(Integer.parseInt(datas[5])>=Integer.parseInt(datas[6])) {

cost = datas[6]+"-"+datas[5];

}else {

//如果上限小于下限，则需要修正，并使计数器+1

context.getCounter(ReportUpdata.Updata\_number).increment(1);

cost = datas[5]+"-"+datas[6];

}

String[] result = new String[11];

int x = 0;

//遍历赋值

for(int i=0;i<datas.length;i++) {

if(i == 5) {

result[x] = cost;

x++;

}else if(i == 6) {

//不做任何操作

}else {

result[x] = datas[i];

x++;

}

}

给数据源重新设置分割符。

String str = "";

for(String i : result) {

if(i.equals(result[result.length-1])) {

str = str+i;

}else {

str = str+i+"|";

}

}

//输出数据

context.write(key, new Text(str));

（3）编辑Recruit MR/src/com/org/task2/CleanReducer.java文件，根据MapReduce规则，返回的Key-values的value值为一个Iterator的对象，编写MapReduce数据预处理Reduce程序，在reduce方法中添加如下代码，对reduce任务不做任何操作，直接输出结果。

for(Text t : values) {

context.write(NullWritable.get(), t);

}

（4）编辑Recruit MR/src/com/org/task2/CleanJob.java文件，编写MapReduce启动程序，将下面的所有代码片添加到main方法中。初始化Hadoop集群的java配置对象，设置Job类，Job类是java调用MapReduce的java对象，其中setJarByClass方法的参数值需要给到main方法所在的类。

// TODO Auto-generated method stub

Configuration conf = new Configuration();

Job job = Job.getInstance(conf);

//设置job类

job.setJarByClass(CleanJob.class);

分别设置MapReduce的Map类和Reduce类，以及输入输出数据的数据类型，分别完成两个过程的业务逻辑。

//设置map和reduce的类

job.setMapperClass(CleanMapper.class);

job.setReducerClass(CleanReducer.class);

//设置map输出key，value类型

job.setMapOutputKeyClass(LongWritable.class);

job.setMapOutputValueClass(Text.class);

//设置reduce输出的key，value类型

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

设置数据源输入输出路径，这里设置数据源文件存放于HDFS的recruitdata1目录，将结果输出至HDFS的recruitoutput2目录，设置输出的hdfs路径，注意：hdfs不能存在输出的文件夹，mapreduce会自动创建。

FileInputFormat.addInputPath(job, new Path("/recruitoutput1"));

FileOutputFormat.setOutputPath(job, new Path("/recruitoutput2"));

System.exit(job.waitForCompletion(true)?0:1);

Counters counters = job.getCounters();

System.out.println("counters getGroupNames:"+counters.getGroupNames());

#### 步骤三：统一日期格式

题目具体参数要求，将原始数据集中格式不一致的数据进行标准化处理，并存入指定数据表或数据文件中。清洗任务三数据清洗前，数据中日期存在几种情况的数据，如表17-3所示：

表17-3 发布时间数据样式

|  |
| --- |
| 发布时间数据存在的格式 |
| "发布时间": "12-29-2008" |
| "发布时间": "29 Dec 2008 16:25:46" |
| "发布时间": "18th,Apr,2019,10:52:53:000000" |
| "发布时间": "Apr,18th,2019,10:46 AM" |
| "发布时间": "18th,Apr,2019 10:52:53" |
| "发布时间": "Dec 29 2008 11:45 PM" |

完成清洗分析任务，统一日期格式执行后的数据样式后，统一日期为yyyy-mm-dd格式如下所示：

Java工程师|阿里巴巴|北京|熟练使用RPC框架，具备相关的分布式开发经验|3|7000-12000|Spark, HBase, Linux, Python|2019-03-28|女|阿里巴巴网络技术有限公司（简称：阿里巴巴集团）是以曾担任英语教师的马云为首的18人于1999年在浙江杭州创立|专科及其以上

（1）编辑Recruit MR/src/com/org/task3/CleanMapper.java文件,编写MapReduce数据预处理Map程序，下面的所有代码片添加到main方法中。上一节清洗结果保存在/recruitoutput2中，编写MapReduce程序，分析“发布日期”数据格式。

读取数据，根据数据源的分隔符进行分割。

String raw = value.toString();

String[] datas = raw.split("\\|");

建立时间的数据映射，通过正则表示是，将日期格式统一为yyyy-mm-dd的形式。

// 创建一个时间的map集合，存放月份的英文和数字的转换字符串。

HashMap<String,String> timemap = new HashMap<String, String>();

// 将月份英文缩写和数字以键值对的形式存放在map集合中。

timemap.put("Jan", "01");

timemap.put("Feb", "02");

timemap.put("Mar", "03");

timemap.put("Apr", "04");

timemap.put("May", "05");

timemap.put("Jun", "06");

timemap.put("Jul", "07");

timemap.put("Aug", "08");

timemap.put("Sept", "09");

timemap.put("Oct", "10");

timemap.put("Nov", "11");

timemap.put("Dec", "12");

//根据三种格式，创建正则表达式

//Dec 29 2008 11:45 PM

String partition1 = "([a-zA-Z]{3,4}) ([0-9]{2}) ([0-9]{4}) [0-9]{1,2}:[0-9]{1,2} [a-zA-Z]{2}";

//12-29-2008

String partition2 = "([0-9]{2})-([0-9]{2})-([0-9]{4})";

//29 Dec 2008 16:25:46

String partition3 = "([0-9]{2}) ([a-zA-Z]{3,4}) ([0-9]{4}) [0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2}";

//18th,Apr,2019,10:52:53:000000

String partition4 = "([0-9]{2})[a-z]{2},([a-zA-Z]{3,4}),([0-9]{4}),[0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2}:[0-9]{6}";

//Apr,18th,2019,10:46 AM

String partition5 = "([a-zA-Z]{3,4}),([0-9]{2})[a-z]{2},([0-9]{4}),[0-9]{1,2}:[0-9]{1,2} [a-zA-Z]{2}";

//18th,Apr,2019 10:52:53

String partition6 = "([0-9]{2})[a-z]{2},([a-zA-Z]{3,4}),([0-9]{4}) [0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2}";

//编译正则表达式

Pattern r1 = Pattern.compile(partition1);

Pattern r2 = Pattern.compile(partition2);

Pattern r3 = Pattern.compile(partition3);

Pattern r4 = Pattern.compile(partition4);

Pattern r5 = Pattern.compile(partition5);

Pattern r6 = Pattern.compile(partition6);

//对datas[7]时间字段进行匹配，每一层if判断语句都包含每个字段的分割匹配

Matcher m = r1.matcher(datas[7]);

if(m.find()) {

datas[7] = m.group(3)+"-"+timemap.get(m.group(1))+"-"+m.group(2);

}else {

m = r2.matcher(datas[7]);

if(m.find()) {

datas[7] = m.group(3)+"-"+m.group(1)+"-"+m.group(2);

}else {

m = r3.matcher(datas[7]);

if(m.find()) {

datas[7] = m.group(3)+"-"+timemap.get(m.group(2))+"-"+m.group(1);

}else {

m = r4.matcher(datas[7]);

if(m.find()) {

datas[7] = m.group(3)+"-"+timemap.get(m.group(2))+"-"+m.group(1);

}else {

m = r5.matcher(datas[7]);

if(m.find()) {

datas[7] = m.group(3)+"-"+timemap.get(m.group(1))+"-"+m.group(2);

}else {

m = r6.matcher(datas[7]);

if(m.find()) {

datas[7] = m.group(3)+"-"+timemap.get(m.group(2))+"-"+m.group(1);

}

}

}

}

}

}

给数据源重新设置分割符。

String str = "";

for(String i : datas) {

if(i.equals(datas[datas.length-1])) {

str = str+i;

}else {

str = str+i+"|";

}

}

//输出数据

context.write(key, new Text(str));

（2）编辑Recruit MR/src/com/org/task3/CleanReducer.java文件,编写MapReduce数据预处理Reduce程序，在reduce方法中添加如下代码

//reduce不做任何操作，直接输出。

for(Text t : values) {

context.write(NullWritable.get(), t);

}

（3）编辑Recruit MR/src/com/org/task3/CleanJob.java文件，编写MapReduce启动程序，将下面的所有代码片添加到main方法中。初始化Hadoop集群的java配置对象，设置Job类，Job类是java调用MapReduce的java对象，其中setJarByClass方法的参数值需要给到main方法所在的类。

// TODO Auto-generated method stub

Configuration conf = new Configuration();

Job job = Job.getInstance(conf);

//设置job类

job.setJarByClass(CleanJob.class);

分别设置MapReduce的Map类和Reduce类，以及输入输出数据的数据类型，分别完成两个过程的业务逻辑。

//设置map和reduce的类

job.setMapperClass(CleanMapper.class);

job.setReducerClass(CleanReducer.class);

//设置map输出key，value类型

job.setMapOutputKeyClass(LongWritable.class);

job.setMapOutputValueClass(Text.class);

//设置reduce输出的key，value类型

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

设置数据源输入输出路径，这里设置数据源文件存放于HDFS的recruitdata2目录，将结果输出至HDFS的recruitoutput3目录，设置输出的hdfs路径，注意：hdfs不能存在输出的文件夹，mapreduce会自动创建。

FileInputFormat.addInputPath(job, new Path("/recruitoutput2"));

FileOutputFormat.setOutputPath(job, new Path("/recruitoutput3"));

System.exit(job.waitForCompletion(true)?0:1);

Counters counters = job.getCounters();

System.out.println("counters getGroupNames:"+counters.getGroupNames());

### 实验任务三：运行业务处理相关清洗任务

#### 步骤一：打包程序

MapReduce基于Hadoop集群以及JVM运行，所以需要将MapReduce程序打包成一个jar包，才能在集群中进行运行相关清洗任务的程序。

本小节的MapReduce程序已经被打包并上传到大数据平台的/opt/data/目录中，jar包的名字为recruit.jar。

#### 步骤二：.运行MapReduce程序

[hadoop@master ~]$ hadoop jar /opt/data/recruit.jar com.org.task1.CleanJob

**20/05/11 11:22:55 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032**

**20/05/11 11:22:56 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.**

**20/05/11 11:22:57 INFO input.FileInputFormat: Total input paths to process : 1**

**20/05/11 11:22:58 INFO mapreduce.JobSubmitter: number of splits:2**

**20/05/11 11:22:58 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1589167350629\_0001**

**20/05/11 11:23:04 INFO impl.YarnClientImpl: Submitted application application\_1589167350629\_0001**

**20/05/11 11:23:04 INFO mapreduce.Job: The url to track the job: http://hadoop:8088/proxy/application\_1589167350629\_0001/**

**20/05/11 11:23:04 INFO mapreduce.Job: Running job: job\_1589167350629\_0001**

**20/05/11 11:23:18 INFO mapreduce.Job: Job job\_1589167350629\_0001 running in uber mode : false**

**20/05/11 11:23:18 INFO mapreduce.Job: map 0% reduce 0%**

**20/05/11 11:23:41 INFO mapreduce.Job: map 7% reduce 0%**

**20/05/11 11:23:44 INFO mapreduce.Job: map 19% reduce 0%**

**20/05/11 11:23:47 INFO mapreduce.Job: map 34% reduce 0%**

**20/05/11 11:23:50 INFO mapreduce.Job: map 52% reduce 0%**

**20/05/11 11:23:53 INFO mapreduce.Job: map 72% reduce 0%**

**20/05/11 11:23:57 INFO mapreduce.Job: map 79% reduce 0%**

**20/05/11 11:23:59 INFO mapreduce.Job: map 82% reduce 0%**

**20/05/11 11:24:03 INFO mapreduce.Job: map 94% reduce 0%**

**20/05/11 11:24:06 INFO mapreduce.Job: map 100% reduce 0%**

**20/05/11 11:24:24 INFO mapreduce.Job: map 100% reduce 67%**

**20/05/11 11:24:29 INFO mapreduce.Job: map 100% reduce 80%**

**20/05/11 11:24:31 INFO mapreduce.Job: map 100% reduce 97%**

**20/05/11 11:24:32 INFO mapreduce.Job: map 100% reduce 100%**

**20/05/11 11:24:33 INFO mapreduce.Job: Job job\_1589167350629\_0001 completed successfully**

**20/05/11 11:24:33 INFO mapreduce.Job: Counters: 50**

**File System Counters**

**FILE: Number of bytes read=258405630**

**FILE: Number of bytes written=411700949**

**FILE: Number of read operations=0**

**FILE: Number of large read operations=0**

**FILE: Number of write operations=0**

**HDFS: Number of bytes read=194107320**

**HDFS: Number of bytes written=151336320**

**HDFS: Number of read operations=9**

**HDFS: Number of large read operations=0**

**HDFS: Number of write operations=2**

**Job Counters**

**Launched map tasks=2**

**Launched reduce tasks=1**

**Data-local map tasks=2**

**Total time spent by all maps in occupied slots (ms)=81704**

**Total time spent by all reduces in occupied slots (ms)=31136**

**Total time spent by all map tasks (ms)=81704**

**Total time spent by all reduce tasks (ms)=31136**

**Total vcore-milliseconds taken by all map tasks=81704**

**Total vcore-milliseconds taken by all reduce tasks=31136**

**Total megabyte-milliseconds taken by all map tasks=83664896**

**Total megabyte-milliseconds taken by all reduce tasks=31883264**

**Map-Reduce Framework**

**Map input records=123712**

**Map output records=113577**

**Map output bytes=152472090**

**Map output materialized bytes=152926410**

**Input split bytes=232**

**Combine input records=0**

**Combine output records=0**

**Reduce input groups=113577**

**Reduce shuffle bytes=152926410**

**Reduce input records=113577**

**Reduce output records=113577**

**Spilled Records=306454**

**Shuffled Maps =2**

**Failed Shuffles=0**

**Merged Map outputs=2**

**GC time elapsed (ms)=2846**

**CPU time spent (ms)=24290**

**Physical memory (bytes) snapshot=539246592**

**Virtual memory (bytes) snapshot=6314057728**

**Total committed heap usage (bytes)=301146112**

**Shuffle Errors**

**BAD\_ID=0**

**CONNECTION=0**

**IO\_ERROR=0**

**WRONG\_LENGTH=0**

**WRONG\_MAP=0**

**WRONG\_REDUCE=0**

**com.org.task1.ReportDelete**

**Delete\_number=10**

**File Input Format Counters**

**Bytes Read=194107088**

**File Output Format Counters**

**Bytes Written=151336320**

执行Hadoop命令，查看HDFS文件系统上是否已生成输出文件。命令如下：

[hadoop@master ~]# hdfs dfs -ls /recruitoutput1

**Found 2 items**

**-rw-r--r-- 1 hadoop supergroup 0 2020-05-11 11:24 /recruitoutput1/\_SUCCESS #成功标记**

**-rw-r--r-- 1 hadoop supergroup 151336320 2020-05-11 11:24 /recruitoutput1/part-r-00000 #输出文件**

执行MapReduce任务，完成合并薪资上下限的数据清洗任务。命令如下：

[hadoop@master ~]# hadoop jar /opt/data/recruit.jar com.org.task2.CleanJob

**20/05/11 11:27:32 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032**

**20/05/11 11:27:33 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.**

**20/05/11 11:27:34 INFO input.FileInputFormat: Total input paths to process : 1**

**20/05/11 11:27:34 INFO mapreduce.JobSubmitter: number of splits:2**

**20/05/11 11:27:34 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1589167350629\_0002**

**20/05/11 11:27:40 INFO impl.YarnClientImpl: Submitted application application\_1589167350629\_0002**

**20/05/11 11:27:40 INFO mapreduce.Job: The url to track the job: http://hadoop:8088/proxy/application\_1589167350629\_0002/**

**20/05/11 11:27:40 INFO mapreduce.Job: Running job: job\_1589167350629\_0002**

**20/05/11 11:27:51 INFO mapreduce.Job: Job job\_1589167350629\_0002 running in uber mode : false**

**20/05/11 11:27:51 INFO mapreduce.Job: map 0% reduce 0%**

**20/05/11 11:28:07 INFO mapreduce.Job: map 50% reduce 0%**

**20/05/11 11:28:09 INFO mapreduce.Job: map 71% reduce 0%**

**20/05/11 11:28:11 INFO mapreduce.Job: map 78% reduce 0%**

**20/05/11 11:28:15 INFO mapreduce.Job: map 83% reduce 0%**

**20/05/11 11:28:19 INFO mapreduce.Job: map 88% reduce 0%**

**20/05/11 11:28:21 INFO mapreduce.Job: map 100% reduce 0%**

**20/05/11 11:28:34 INFO mapreduce.Job: map 100% reduce 70%**

**20/05/11 11:28:37 INFO mapreduce.Job: map 100% reduce 97%**

**20/05/11 11:28:39 INFO mapreduce.Job: map 100% reduce 100%**

**20/05/11 11:28:40 INFO mapreduce.Job: Job job\_1589167350629\_0002 completed successfully**

**20/05/11 11:28:40 INFO mapreduce.Job: Counters: 51**

**File System Counters**

**FILE: Number of bytes read=288578686**

**FILE: Number of bytes written=441874014**

**FILE: Number of read operations=0**

**FILE: Number of large read operations=0**

**FILE: Number of write operations=0**

**HDFS: Number of bytes read=151340654**

**HDFS: Number of bytes written=151336320**

**HDFS: Number of read operations=9**

**HDFS: Number of large read operations=0**

**HDFS: Number of write operations=2**

**Job Counters**

**Killed map tasks=1**

**Launched map tasks=3**

**Launched reduce tasks=1**

**Data-local map tasks=3**

**Total time spent by all maps in occupied slots (ms)=51368**

**Total time spent by all reduces in occupied slots (ms)=29385**

**Total time spent by all map tasks (ms)=51368**

**Total time spent by all reduce tasks (ms)=29385**

**Total vcore-milliseconds taken by all map tasks=51368**

**Total vcore-milliseconds taken by all reduce tasks=29385**

**Total megabyte-milliseconds taken by all map tasks=52600832**

**Total megabyte-milliseconds taken by all reduce tasks=30090240**

**Map-Reduce Framework**

**Map input records=113577**

**Map output records=113577**

**Map output bytes=152472090**

**Map output materialized bytes=152926410**

**Input split bytes=238**

**Combine input records=0**

**Combine output records=0**

**Reduce input groups=113577**

**Reduce shuffle bytes=152926410**

**Reduce input records=113577**

**Reduce output records=113577**

**Spilled Records=329553**

**Shuffled Maps =2**

**Failed Shuffles=0**

**Merged Map outputs=2**

**GC time elapsed (ms)=1087**

**CPU time spent (ms)=13170**

**Physical memory (bytes) snapshot=568848384**

**Virtual memory (bytes) snapshot=6302265344**

**Total committed heap usage (bytes)=326156288**

**Shuffle Errors**

**BAD\_ID=0**

**CONNECTION=0**

**IO\_ERROR=0**

**WRONG\_LENGTH=0**

**WRONG\_MAP=0**

**WRONG\_REDUCE=0**

**com.org.task2.ReportUpdata**

**Updata\_number=17**

**File Input Format Counters**

**Bytes Read=151340416**

**File Output Format Counters**

**Bytes Written=151336320**

执行Hadoop命令，查看HDFS文件系统上是否已生成输出文件。命令如下：

[hadoop@master ~]# hdfs dfs -ls /recruitoutput2

**Found 2 items**

**-rw-r--r-- 1 hadoop supergroup 0 2020-05-11 11:24 /recruitoutput2/\_SUCCESS #成功标记**

**-rw-r--r-- 1 hadoop supergroup 151336320 2020-05-11 11:24 /recruitoutput2/part-r-00000 #输出文件**

执行MapReduce任务，完成统一日期格式的数据清洗任务。命令如下：

[hadoop@master ~]# hadoop jar /opt/data/recruit.jar com.org.task3.CleanJob

**20/05/11 11:29:28 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032**

**20/05/11 11:29:29 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.**

**20/05/11 11:29:30 INFO input.FileInputFormat: Total input paths to process : 1**

**20/05/11 11:29:30 INFO mapreduce.JobSubmitter: number of splits:2**

**20/05/11 11:29:30 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1589167350629\_0003**

**20/05/11 11:29:31 INFO impl.YarnClientImpl: Submitted application application\_1589167350629\_0003**

**20/05/11 11:29:31 INFO mapreduce.Job: The url to track the job: http://hadoop:8088/proxy/application\_1589167350629\_0003/**

**20/05/11 11:29:31 INFO mapreduce.Job: Running job: job\_1589167350629\_0003**

**20/05/11 11:29:47 INFO mapreduce.Job: Job job\_1589167350629\_0003 running in uber mode : false**

**20/05/11 11:29:47 INFO mapreduce.Job: map 0% reduce 0%**

**20/05/11 11:30:04 INFO mapreduce.Job: map 50% reduce 0%**

**20/05/11 11:30:05 INFO mapreduce.Job: map 59% reduce 0%**

**20/05/11 11:30:08 INFO mapreduce.Job: map 71% reduce 0%**

**20/05/11 11:30:11 INFO mapreduce.Job: map 75% reduce 0%**

**20/05/11 11:30:14 INFO mapreduce.Job: map 80% reduce 0%**

**20/05/11 11:30:18 INFO mapreduce.Job: map 83% reduce 0%**

**20/05/11 11:30:20 INFO mapreduce.Job: map 100% reduce 0%**

**20/05/11 11:30:37 INFO mapreduce.Job: map 100% reduce 91%**

**20/05/11 11:30:38 INFO mapreduce.Job: map 100% reduce 100%**

**20/05/11 11:30:39 INFO mapreduce.Job: Job job\_1589167350629\_0003 completed successfully**

**20/05/11 11:30:40 INFO mapreduce.Job: Counters: 50**

**File System Counters**

**FILE: Number of bytes read=288567422**

**FILE: Number of bytes written=441857118**

**FILE: Number of read operations=0**

**FILE: Number of large read operations=0**

**FILE: Number of write operations=0**

**HDFS: Number of bytes read=151340654**

**HDFS: Number of bytes written=151330688**

**HDFS: Number of read operations=9**

**HDFS: Number of large read operations=0**

**HDFS: Number of write operations=2**

**Job Counters**

**Killed map tasks=1**

**Launched map tasks=3**

**Launched reduce tasks=1**

**Data-local map tasks=3**

**Total time spent by all maps in occupied slots (ms)=61871**

**Total time spent by all reduces in occupied slots (ms)=31048**

**Total time spent by all map tasks (ms)=61871**

**Total time spent by all reduce tasks (ms)=31048**

**Total vcore-milliseconds taken by all map tasks=61871**

**Total vcore-milliseconds taken by all reduce tasks=31048**

**Total megabyte-milliseconds taken by all map tasks=63355904**

**Total megabyte-milliseconds taken by all reduce tasks=31793152**

**Map-Reduce Framework**

**Map input records=113577**

**Map output records=113577**

**Map output bytes=152466458**

**Map output materialized bytes=152920778**

**Input split bytes=238**

**Combine input records=0**

**Combine output records=0**

**Reduce input groups=113577**

**Reduce shuffle bytes=152920778**

**Reduce input records=113577**

**Reduce output records=113577**

**Spilled Records=329553**

**Shuffled Maps =2**

**Failed Shuffles=0**

**Merged Map outputs=2**

**GC time elapsed (ms)=1586**

**CPU time spent (ms)=15730**

**Physical memory (bytes) snapshot=570970112**

**Virtual memory (bytes) snapshot=6299836416**

**Total committed heap usage (bytes)=328450048**

**Shuffle Errors**

**BAD\_ID=0**

**CONNECTION=0**

**IO\_ERROR=0**

**WRONG\_LENGTH=0**

**WRONG\_MAP=0**

**WRONG\_REDUCE=0**

**File Input Format Counters**

**Bytes Read=151340416**

**File Output Format Counters**

**Bytes Written=151330688**

执行Hadoop命令，查看HDFS文件系统上是否已生成输出文件，命令如下：

[hadoop@master ~]# hdfs dfs -ls /recruitoutput3

**Found 2 items**

**-rw-r--r-- 1 hadoop supergroup 0 2020-05-11 11:24 /recruitoutput3/\_SUCCESS #成功标记**

**-rw-r--r-- 1 hadoop supergroup 151336320 2020-05-11 11:24 /recruitoutput3/part-r-00000 #输出文件**

### 实验任务四：数据进入数据仓库

#### 步骤一：将数据存储到Hive数据仓库

若要将清洗后的数据存储到数据文件中，需要将数据的不同字段使用某种分隔符分隔开后，再写入数据文件中。后续将数据文件再导入数据库时，同样以该分隔符进行字段划分。请根据题目具体参数要求，将清洗后的数据以指定数据分隔符进行分隔，存入指定数据文件中，再使用数据转移工具将数据导入数据库中。（本节将以17.4.3的输出结果作为数据源。）

本小节数据将对清洗分析之后存放在HDFS的数据，通过Sqoop导入到Hive数据仓库，如图17-4所示。

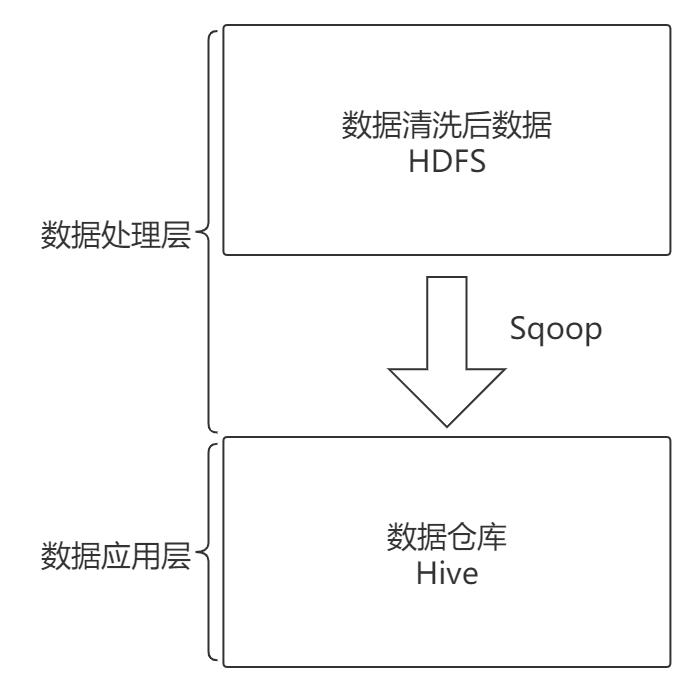


图17-4 数据进入数据仓库

（1）启动Hadoop集群之后，进入Hive命令终端,

（2）创建表rawdata，数据中包含字段，清洗后的职位数据以“|”分隔，如表17-4所示。

[hadoop@master ~]# hive

hive> create table rawdata

> (name STRING, com\_name STRING, city STRING, requirement STRING,

> number STRING, salary STRING, skill STRING, time STRING, gender STRING,

> intro STRING, edu STRING)

> row format delimited fields terminated by "|" stored as textfile

> location '/recruitoutput3'/;

表17-4 数据源

|  |  |
| --- | --- |
| 字段 | 中文解析 |
| com\_name | 公司名称 |
| city | 工作城市 |
| requirement | 工作要求 |
| number | 招聘人数 |
| salary | 工资 |
| skill | 技能要求 |
| time | 布时间 |
| gender | 性别 |
| intro | 公司描述 |
| edu | 学历要求 |
| name | 岗位名称 |

（3）导入前面17.4.3清洗之后输出到HDFS中/recruitoutput3的数据到新建的rawdata表中，在Hive中inpath参数后面添加的是数据源的数据路径，默认情况下读取的是HDFS文件系统上的数据，使用Hive命令，查看导入的数据源总行数如下：

hive> select count(\*) from rawdata;

**WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.**

**Query ID = hadoop\_20200518131919\_f0e8ba3c-9ef8-4204-9a35-49fd000d2344**

**Total jobs = 1**

**Launching Job 1 out of 1**

**Number of reduce tasks determined at compile time: 1**

**In order to change the average load for a reducer (in bytes):**

**set hive.exec.reducers.bytes.per.reducer=<number>**

**In order to limit the maximum number of reducers:**

**set hive.exec.reducers.max=<number>**

**In order to set a constant number of reducers:**

**set mapreduce.job.reduces=<number>**

**Starting Job = job\_1589776556088\_0002, Tracking URL = http://hadoop:8088/proxy/application\_1589776556088\_0002/**

**Kill Command = /usr/local/src/hadoop/bin/hadoop job -kill job\_1589776556088\_0002**

**Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1**

**2020-05-18 13:19:43,441 Stage-1 map = 0%, reduce = 0%**

**2020-05-18 13:19:59,446 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.15 sec**

**2020-05-18 13:20:17,732 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 8.04 sec**

**MapReduce Total cumulative CPU time: 8 seconds 40 msec**

**Ended Job = job\_1589776556088\_0002**

**MapReduce Jobs Launched:**

**Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.04 sec HDFS Read: 151343731 HDFS Write: 106 SUCCESS**

**Total MapReduce CPU Time Spent: 8 seconds 40 msec**

**OK**

**113577**

**Time taken: 60.788 seconds, Fetched: 1 row(s)**

### 实验任务五：业务应用层大数据分析

本章节着重在大数据业务处理系统的业务应用层的数据分析模块，17.4.4已经将处理好的数据存放到了Hive中，使用Hadoop生态组件Hive进行数据分析。

业务需求中有其中一项为：统计“大数据”相关岗位的招聘信息。统计相关岗位的招聘信息，可以得出该岗位的招聘热度，从而为求职者提供应聘的提示：什么岗位比较容易被录取，什么岗位比较容易多人同时竞争，同时也为招聘者提供招聘提示：什么岗位最为稀缺人才，什么岗位的应聘者较多充足，以下将进行对“大数据”相关岗位的招聘信息进行分析统计。

#### 步骤一：统计“大数据”相关职位招聘信息

（1）rawdata为数据源，统计“大数据”相关职位招聘信息。字段信息如表17-3所示。

表17-4 数据源

|  |  |
| --- | --- |
| 字段 | 中文解析 |
| com\_name | 公司名称 |
| city | 工作城市 |
| requirement | 工作要求 |
| number | 招聘人数 |
| salary | 工资 |
| skill | 技能要求 |
| time | 布时间 |
| gender | 性别 |
| intro | 公司描述 |
| edu | 学历要求 |
| name | 岗位名称 |

创建新表用来存放数据，hive命令如下：

hive> create table bigdata\_work

> (name STRING, com\_name STRING, city STRING, requirement STRING,

> number STRING, salary STRING, skill STRING, time STRING, gender STRING,

> intro STRING, edu STRING)

> row format delimited fields terminated by "|" stored as textfile;

**OK**

**Time taken: 0.386 seconds**

（2）hive查询将结果保存到新表中，hive命令如下：

hive> insert overwrite table bigdata\_work select \* from rawdata where name like '%大数据%';

**Query ID = hadoop\_20200425222626\_403811b5-cd59-42e3-a05b-9ddfe12e9af4**

**Total jobs = 3**

**Launching Job 1 out of 3**

**Number of reduce tasks is set to 0 since there's no reduce operator**

**Starting Job = job\_1585553665184\_0010, Tracking URL = http://master:8088/proxy/application\_1585553665184\_0010/**

**Kill Command = /usr/local/src/hadoop/bin/hadoop job -kill job\_1585553665184\_0010**

**Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 0**

**2020-04-25 22:26:11,674 Stage-1 map = 0%, reduce = 0%**

**2020-04-25 22:26:20,366 Stage-1 map = 33%, reduce = 0%, Cumulative CPU 1.68 sec**

**2020-04-25 22:26:21,402 Stage-1 map = 67%, reduce = 0%, Cumulative CPU 4.28 sec**

**2020-04-25 22:26:22,425 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 6.95 sec**

**MapReduce Total cumulative CPU time: 6 seconds 950 msec**

**Ended Job = job\_1585553665184\_0010**

**Stage-4 is filtered out by condition resolver.**

**Stage-3 is selected by condition resolver.**

**Stage-5 is filtered out by condition resolver.**

**Launching Job 3 out of 3**

**Number of reduce tasks is set to 0 since there's no reduce operator**

**Starting Job = job\_1585553665184\_0011, Tracking URL = http://master:8088/proxy/application\_1585553665184\_0011/**

**Kill Command = /usr/local/src/hadoop/bin/hadoop job -kill job\_1585553665184\_0011**

**Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0**

**2020-04-25 22:26:31,875 Stage-3 map = 0%, reduce = 0%**

**2020-04-25 22:26:36,013 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.23 sec**

**MapReduce Total cumulative CPU time: 1 seconds 230 msec**

**Ended Job = job\_1585553665184\_0011**

**Loading data to table hive\_data\_jobs.test**

**Table hive\_data\_jobs.test stats: [numFiles=1, numRows=197754, totalSize=2551545, rawDataSize=2353791]**

**MapReduce Jobs Launched:**

**Stage-Stage-1: Map: 3 Cumulative CPU: 6.95 sec HDFS Read: 622031566 HDFS Write: 2551792 SUCCESS**

**Stage-Stage-3: Map: 1 Cumulative CPU: 1.23 sec HDFS Read: 2553835 HDFS Write: 2551545 SUCCESS**

**Total MapReduce CPU Time Spent: 8 seconds 180 msec**

**OK**

**Time taken: 32.788 seconds**

（3）查询结果hive分析的结果总条数。

hive> select count(\*) from bigdata\_work;

**OK**

**9911**

**Time taken: 0.662 seconds, Fetched: 1 row(s)**

hive> exit;

### 实验任务六：数据仓库数据导出数据访问层

本教材搭建的大数据业务处理系统中各个业务层面的数据传输都使用Hadoop生态组件Sqoop完成数据传输，17.4.4已经将统计分析出的结果存放到了Hive新数据表里。

该流程是通过Sqoop组件，将Hive数据仓库中的数据导出到MySQL数据库中，用做数据可视化，如图17-5所示。

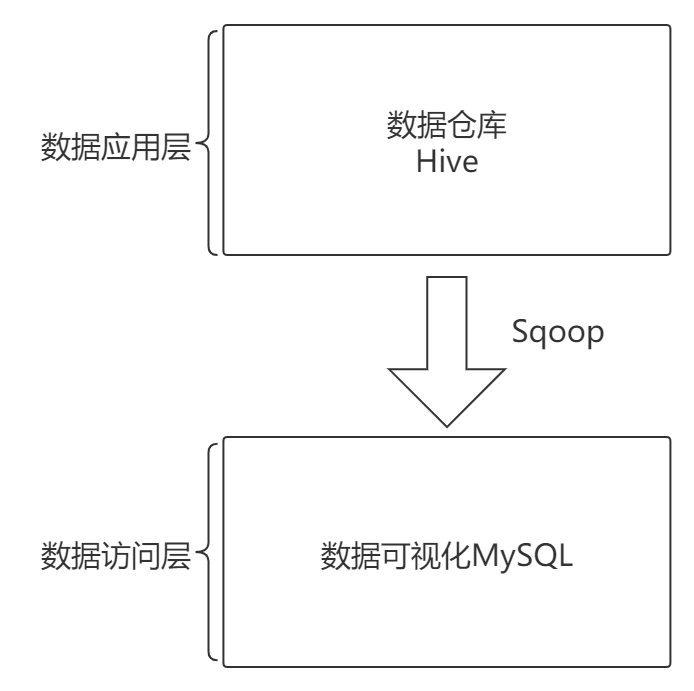


图17-5 数据仓库数据导出数据访问层

#### 步骤一：使用Sqoop组件完成数据从Hive导出到MySQL

（1）实验之前需要启动MySQL服务，启动Hadoop集群，输入Sqoop命令，Sqoop组件会调用MapReduce将Hive数据导入MySQL。

（注意）数据被导入到MySQL之后，Hive的数据会被清空，参数解析如表17-5。

表17-5 sqoop传输参数

|  |  |
| --- | --- |
| export | 表示sqoop操作用于数据导出 |
| connect | MySQL连接地址 |
| username | MySQL用户名 |
| password | MySQL连接密码 |
| table | MySQL表 |
| --input-fields-terminated-by | 设置分隔符 |
| --export-dir | hive表的数据位置 |

[hadoop@master ~]# mysql -uroot -pPassword123$

mysql> show databases;

**+--------------------+**

**| Database |**

**+--------------------+**

**| information\_schema |**

**| hive |**

**| mysql |**

**| performance\_schema |**

**| sample |**

**| sys |**

**+--------------------+**

**6 rows in set (0.01 sec)**

mysql> create database visiondata;

**Query OK, 1 row affected (0.00 sec)**

mysql> use visiondata;

**Database changed**

mysql> SET FOREIGN\_KEY\_CHECKS=0;

**Query OK, 0 rows affected (0.00 sec)**

mysql> DROP TABLE IF EXISTS `bigdata\_work`;

**Query OK, 0 rows affected, 1 warning (0.00 sec)**

mysql> CREATE TABLE `bigdata\_work` (

-> `job\_name` varchar(255) DEFAULT NULL,

-> `company\_name` varchar(255) DEFAULT NULL,

-> `city` varchar(255) DEFAULT NULL,

-> `job\_require` text,

-> `recruit\_number` varchar(255) DEFAULT NULL,

-> `money` varchar(255) DEFAULT NULL,

-> `skill\_require` varchar(255) DEFAULT NULL,

-> `release\_date` date DEFAULT NULL,

-> `sex` varchar(255) DEFAULT NULL,

-> `company\_detail` varchar(255) DEFAULT NULL,

-> `education` varchar(255) DEFAULT NULL

-> ) ;

**Query OK, 0 rows affected (0.02 sec)**

mysql> exit;

**Bye**

[hadoop@master ~]# sqoop export --connect "jdbc:mysql://master:3306/visiondata?useUnicode=true&characterEncoding=utf8" --username root --password Password123$ --table bigdata\_work -input-fields-terminated-by "|" --export-dir /user/hive/warehouse/bigdata\_work

**20/04/27 15:24:54 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5**

**20/04/27 15:24:56 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.**

**INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-hadoop/compile/cebe706d23ebb1fd99c1f063ad51ebd7/emp.jar**

**-----------------------------------------------------**

**O mapreduce.Job: map 0% reduce 0%**

**20/04/27 15:28:08 INFO mapreduce.Job: map 100% reduce 0%**

**20/04/27 15:28:16 INFO mapreduce.Job: Job job\_1419242001831\_0001 completed successfully**

**-----------------------------------------------------**

**-----------------------------------------------------**

**20/04/27 15:28:17 INFO mapreduce.ImportJobBase: Transferred 145 bytes in 177.5849 seconds (0.8165 bytes/sec)**

**20/04/27 15:28:17 INFO mapreduce.ImportJobBase: Retrieved 5 records.**

（2）查看数据是否导出MySQL，启动MySQL，命令如下：

[hadoop@master ~]$ mysql -u root -pPassword123$

**mysql: [Warning] Using a password on the command line interface can be insecure.**

**Welcome to the MySQL monitor. Commands end with ; or \g.**

**Your MySQL connection id is 10**

**Server version: 5.7.18 MySQL Community Server (GPL)**

**Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.**

**Oracle is a registered trademark of Oracle Corporation and/or its**

**affiliates. Other names may be trademarks of their respective**

**owners.**

**Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.**

mysql>

#使用数据库。

mysql> use visiondata;

**Reading table information for completion of table and column names**

**You can turn off this feature to get a quicker startup with -A**

**Database changed**

#添加自增id字段

mysql> alter table bigdata\_work add column id int auto\_increment primary key;

#设置自增起始值

mysql> alter table bigdata\_work auto\_increment=1;

#取消**ONLY\_FULL\_GROUP\_BY** 模式

mysql> select @@global.sql\_mode;

**+-------------------------------------------------------------------------------------------------------------------------------------------+**

**| @@global.sql\_mode |**

**+-------------------------------------------------------------------------------------------------------------------------------------------+**

**| ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION |**

**+-------------------------------------------------------------------------------------------------------------------------------------------+**

**1 row in set (0.00 sec)**

mysql> set @@global.sql\_mode='STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION';

mysql> select @@global.sql\_mode;

**+------------------------------------------------------------------------------------------------------------------------+**

**| @@global.sql\_mode |**

**+------------------------------------------------------------------------------------------------------------------------+**

**| STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION |**

**+------------------------------------------------------------------------------------------------------------------------+**

**1 row in set (0.00 sec)**

查询导入的数据库表，至此数据已经成功从业务应用层导出到了数据访问层，也为后续数据可视化或者其他业务需求提供了较为合理的参考数据，通过MySQL命令查询导入数据的总条数。

mysql> select count(\*) from bigdata\_work;

**OK**

**9911**

### 实验任务七：数据访问层大数据可视化

#### 步骤一：大数据可视化业务需求

分析业务应用层Hive统计分析之后的数据，统计了“大数据”相关职位招聘信息，由于业务部门希望了解2018年以及2019年上半年“大数据”相关职位每个月的招聘情况，从而总结2018年以及2019年上半年的大数据岗位行情，为企业2019年下半年招聘计划提供准备，为应聘者2019年下半年应聘计划提供参考，所以为了让企业和应聘这能直观便捷地看到实际情况，应用Flask程序完成相关数据的展示。

#### 步骤二：运行Flask程序

本教材提供一个已经编写好的Flask可视化项目，需要使用安装PyCharm工具运行该项目即可完成数据可视化。

（1）对编写好的Flask程序编译运行，打开PyCharm的Terminal命令栏，控制台返回的信息如下,说明Flask启动成功，结果如图17-6所示。

[root@master ~]# cd /opt/data/ViewData

[root@master ~]# python3 manage.py runserver

**\* Serving Flask app "app" (lazy loading)**

**\* Environment: production**

**WARNING: This is a development server. Do not use it in a production deployment.**

**Use a production WSGI server instead.**

**\* Debug mode: on**

**\* Restarting with stat**

**\* Debugger is active!**

**\* Debugger PIN: 283-891-031**

**\* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)**

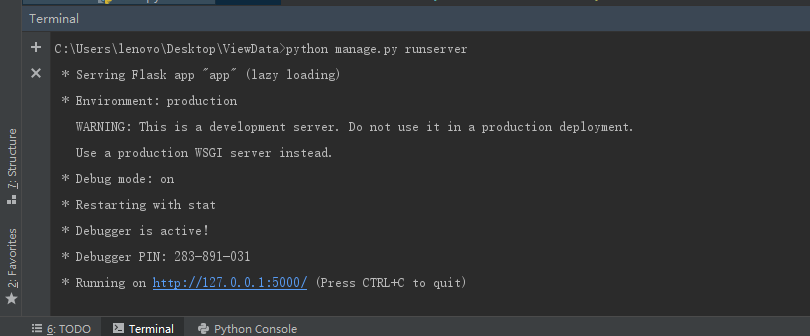


图17-6 Flask启动

（2）打开浏览器，输入WEB可视化项目的URL，输入网站地址为：

http://127.0.0.1:5000/index/

当网站地址被客户端浏览器访问的时候，触发Flask程序查询到的数据结合Echarts前端大数据可视化组件，将数据清楚地展示带浏览器中。如图17-7所示，图形标题为“大数据相关岗位招聘数据”，并且饼形图的左边为2018年度各个岗位的招聘情况，右边为2019年上半年各个岗位的招聘情况。

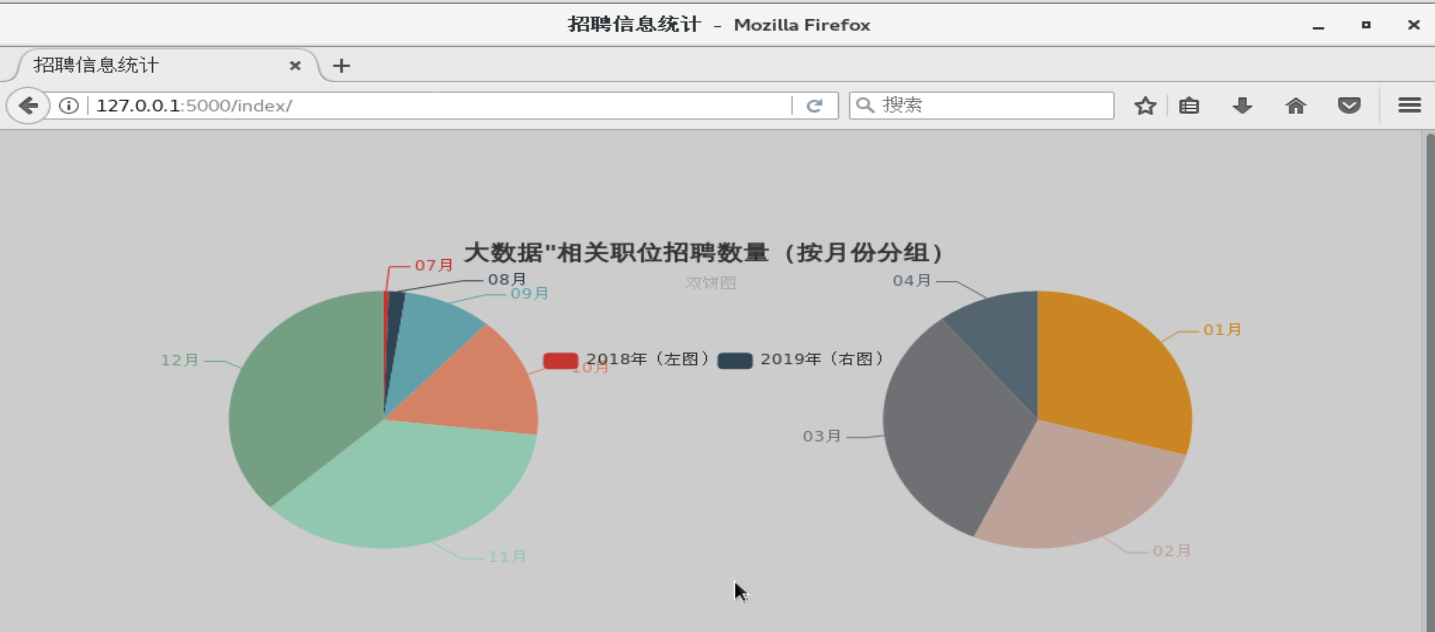


图17-7 可视化