Hadoop案例之单表关联输出祖孙关系

# 1.案例描述

实例中给出**child-parent**（孩子——父母）表，要求输出**grandchild-grandparent**（孙子——爷奶）表。

    样例**输入**如下所示。

**file：**

child        parent

Tom        Lucy

Tom        Jack

Jone        Lucy

Jone        Jack

Lucy        Mary

Lucy        Ben

Jack        Alice

Jack        Jesse

Terry        Alice

Terry        Jesse

Philip        Terry

Philip        Alma

Mark        Terry

Mark        Alma

    家族**树状**关系谱：

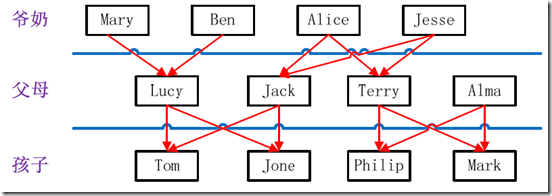
[](http://images.cnblogs.com/cnblogs_com/xia520pi/201206/201206041339478410.png)

图1 家族谱

    样例输出如下所示。

**file：**

grandchild        grandparent

Tom            　　Alice

Tom            　　Jesse

Jone            　　Alice

Jone           　　 Jesse

Tom            　　Mary

Tom            　　Ben

Jone           　　 Mary

Jone           　　 Ben

Philip          　　  Alice

Philip            　　Jesse

Mark           　　 Alice

Mark           　　 Jesse

# 2. 案例分析

分析这个实例，显然需要进行单表连接，连接的是**左表**的**parent**列和**右表**的**child**列，且**左表**和**右表**是**同一个表**。

**连接结果**中**除去**连接的两列就是所需要的结果——"grandchild--grandparent"表。要用MapReduce解决这个实例，**首先**应该考虑如何实现**表**的**自连接**；**其次**就是**连接列**的**设置**；**最后**是**结果**的**整理**。

      考虑到MapReduce的shuffle过程会将相同的key会连接在一起，所以可以将map结果的**key**设置成**待连接**的**列**，然后列中相同的值就自然会连接在一起了。再与最开始的分析联系起来：

　　要连接的是左表的parent列和右表的child列，且左表和右表是同一个表，所以在**map阶段**将**读入数据分割**成**child**和**parent**之后，会将**parent**设置成**key**，**child**设置成**value**进行输出，并作为**左表**；再将**同一对child**和**parent**中的**child**设置成**key**，**parent**设置成**value**进行输出，作为**右表**。为了**区分**输出中的**左右表**，需要在输出的**value**中**再**加上**左右表**的**信息**，比如在value的String最开始处加上**字符1**表示**左表**，加上**字符2**表示**右表**。这样在map的结果中就形成了左表和右表，然后在shuffle过程中完成连接。reduce接收到连接的结果，其中每个key的value-list就包含了"grandchild--grandparent"关系。取出每个key的value-list进行解析，将**左表**中的**child**放入一个**数组**，**右表**中的**parent**放入一个**数组**，然后对**两个数组求笛卡尔积**就是最后的结果了。

# 3. 程序代码

package hadoop\_STjoin;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.GenericOptionsParser;

public class SingleTableJoin {

public static int times = 0;

/\*

\* map将输出分割child和parent，然后正序输出一次作为右表，

\* 反序输出一次作为左表，需要注意的是在输出的value中必须

\* 加上左右表的区别标识。

\*/

public static class Map extends Mapper<Object, Text, Text, Text>

{

protected void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

String childName = new String(); //孩子名字

String parentName = new String(); //父母名字

String relationType = new String(); //左右表标志

//解析每一行

StringTokenizer iter = new StringTokenizer(value.toString());

String[] values = new String[2];

int i=0;

while(iter.hasMoreTokens())

{

values[i] = iter.nextToken();

i++;

}

if (i == 2 && values[0].compareTo("child") != 0) {

childName = values[0];

parentName = values[1];

//输出左表

relationType = "1";

context.write(new Text(parentName),new Text(relationType+"+"+childName+"+"+parentName));

//输出右表

relationType = "2";

context.write(new Text(childName), new Text(relationType+"+"+childName+"+"+parentName));

}

}

}

//reduce

public static class Reduce extends Reducer<Text, Text, Text, Text>

{

protected void reduce(Text key, Iterable<Text> values, Context cont)

throws IOException, InterruptedException {

if (times == 0) {

cont.write(new Text("grandChild"), new Text("grandParen"));

times++;

}

int grandChildNum = 0;

String[] grandChild = new String[10];

int grandParentNum = 0;

String[] grandParent = new String[10];

System.out.println("key="+key.toString());

for(Text temp : values)

{

String record = temp.toString();

System.out.println("record="+record);

if (record.length() == 0) {

System.out.println("continue;");

continue;

}

String childName = new String();

String parenName = new String();

int i = 2;

while (record.charAt(i) != '+')

{

childName += record.charAt(i);

i++;

}

i++;

while(i < record.length())

{

parenName += record.charAt(i);

i++;

}

char relationType = record.charAt(0);

//左表

if ('1' == relationType) {

grandChild[grandChildNum] = childName;

grandChildNum++;

}

if ('2' == relationType) {

grandParent[grandParentNum] = parenName;

grandParentNum++;

}

}

System.out.println("grandChildNum="+grandChildNum+"grandParentNum="+grandParentNum);

// grandchild和grandparent数组求笛卡尔儿积

if (grandChildNum != 0 && grandParentNum != 0) {

for(int m = 0; m < grandChildNum; m++)

{

for(int n = 0; n < grandParentNum; n++)

{

cont.write(new Text(grandChild[m]), new Text(grandParent[n]));

}

}

}

}

}

//main

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

Configuration conf = new Configuration();

String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();

if (otherArgs.length != 2) {

System.err.println("Usage: Single table join <in> <out>");

System.exit(2);

}

Job job = new Job(conf, "single table join");

job.setJarByClass(SingleTableJoin.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(otherArgs[0]));

FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));

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

}

}

# 4.程序启动

root@node1:/usr/local/hadoop/hadoop-2.5.2/myJar# hadoop jar SingleTableJoin.jar hadoop\_STjoin.SingleTableJoin /usr/local/hadooptempdata/input/stjoin /usr/local/hadooptempdata/output/stjoin

16/12/28 22:38:22 INFO client.RMProxy: Connecting to ResourceManager at node1/192.168.233.129:8032

16/12/28 22:38:25 INFO input.FileInputFormat: Total input paths to process : 1

16/12/28 22:38:25 INFO mapreduce.JobSubmitter: number of splits:1

16/12/28 22:38:25 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1482934139129\_0003

16/12/28 22:38:26 INFO impl.YarnClientImpl: Submitted application application\_1482934139129\_0003

16/12/28 22:38:26 INFO mapreduce.Job: The url to track the job: http://node1:8088/proxy/application\_1482934139129\_0003/

16/12/28 22:38:26 INFO mapreduce.Job: Running job: job\_1482934139129\_0003

16/12/28 22:39:16 INFO mapreduce.Job: Job job\_1482934139129\_0003 running in uber mode : false

16/12/28 22:39:16 INFO mapreduce.Job: map 0% reduce 0%

16/12/28 22:39:53 INFO mapreduce.Job: map 100% reduce 0%

16/12/28 22:40:34 INFO mapreduce.Job: map 100% reduce 100%

16/12/28 22:40:36 INFO mapreduce.Job: Job job\_1482934139129\_0003 completed successfully

16/12/28 22:40:36 INFO mapreduce.Job: Counters: 49

File System Counters

FILE: Number of bytes read=565

FILE: Number of bytes written=198627

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=406

HDFS: Number of bytes written=148

HDFS: Number of read operations=6

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Launched map tasks=1

Launched reduce tasks=1

Data-local map tasks=1

Total time spent by all maps in occupied slots (ms)=34937

Total time spent by all reduces in occupied slots (ms)=37274

Total time spent by all map tasks (ms)=34937

Total time spent by all reduce tasks (ms)=37274

Total vcore-seconds taken by all map tasks=34937

Total vcore-seconds taken by all reduce tasks=37274

Total megabyte-seconds taken by all map tasks=35775488

Total megabyte-seconds taken by all reduce tasks=38168576

Map-Reduce Framework

Map input records=15

Map output records=28

Map output bytes=503

Map output materialized bytes=565

Input split bytes=124

Combine input records=0

Combine output records=0

Reduce input groups=12

Reduce shuffle bytes=565

Reduce input records=28

Reduce output records=13

Spilled Records=56

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=254

CPU time spent (ms)=2270

Physical memory (bytes) snapshot=298164224

Virtual memory (bytes) snapshot=3775070208

Total committed heap usage (bytes)=139894784

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=282

File Output Format Counters

Bytes Written=148

# 5. 输出结果

root@node1:/usr/local/hadoop/hadoop-2.5.2/myJar# hdfs dfs -cat /usr/local/hadooptempdata/output/stjoin/\*

grandChild grandParen

Tom Alice

Tom Jesse

Jone Alice

Jone Jesse

Tom Ben

Tom Mary

Jone Ben

Jone Mary

Philip Alice

Philip Jesse

Mark Alice

Mark Jesse