

Spark 实战应用

目 录

1 运行环境说明.....	3
1.1 硬软件环境.....	3
1.2 机器网络环境.....	3
2 SPARK基础应用.....	3
2.1 启动SPARK SHELL.....	4
2.1.1 环境设置.....	4
2.1.2 启动Hadoop.....	5
2.1.3 启动Spark.....	5
2.1.4 启动Spark-Shell.....	5
2.2 SQLCONTEXT演示.....	6
2.2.1 使用Case Class定义RDD演示.....	6
2.2.2 使用applySchema定义RDD演示.....	8
2.2.3 parquet演示.....	10
2.2.4 json演示.....	11
2.2.5 sqlContext中混合使用演示.....	13
2.3 HIVECONTEXT演示.....	13
2.3.1 启动hive.....	14
2.3.2 在SPARK_HOME/conf目录下创建hive-site.xml.....	14
2.3.3 查看数据库表.....	14
2.3.4 计算所有订单中每年的销售单数、销售总额.....	15
2.3.5 计算所有订单每年最大金额订单的销售额.....	17
2.3.6 计算所有订单中每年最畅销货品.....	18
2.3.7 hiveContext中混合使用演示.....	20
2.4 CACHE使用.....	21
2.5 DSL演示.....	24
3 SPARK综合应用.....	24
3.1 SQL ON SPARK.....	25
3.1.1 代码.....	25
3.1.2 操作.....	错误! 未定义书签。
3.2 HIVE ON SPARK.....	29
3.2.1 代码.....	29
3.2.2 操作.....	错误! 未定义书签。
3.3 店铺分类.....	31
3.3.1 代码.....	32
3.3.2 操作.....	错误! 未定义书签。
3.4 PAGERANK.....	36
3.4.1 创建表.....	36
3.4.2 代码.....	38
3.4.3 操作.....	错误! 未定义书签。
3.5 小结.....	40

Spark 实战应用

1 运行环境说明

1.1 硬软件环境

- 主机操作系统：Windows 64 位，双核 4 线程，主频 2.2G，10G 内存
- 虚拟软件：VMware® Workstation 9.0.0 build-812388
- 虚拟机操作系统：CentOS 64 位，单核
- 虚拟机运行环境：
 - JDK : 1.7.0_55 64 位
 - Hadoop : 2.2.0 (需要编译为 64 位)
 - Scala : 2.10.4
 - Spark : 1.1.0 (需要编译)
 - Hive : 0.13.1

1.2 机器网络环境

集群包含三个节点，节点之间可以免密码 SSH 访问，节点 IP 地址和主机名分布如下：

序号	IP 地址	机器名	类型	核数/内存	用户名	目录
1	192.168.0.61	hadoop1	NN/DN/RM Master/Worker	1 核/3G	hadoop	/app 程序所在路径 /app/scala-... /app/hadoop /app/complied
2	192.168.0.62	hadoop2	DN/NM/Worker	1 核/2G	hadoop	
3	192.168.0.63	hadoop3	DN/NM/Worker	1 核/2G	hadoop	

2 Spark 基础应用

SparkSQL 引入了一种新的 RDD——SchemaRDD，SchemaRDD 由行对象（Row）以及描述行对象中每列数据类型的 Schema 组成；SchemaRDD 很象传统数据库中的表。SchemaRDD 可以通过 RDD、Parquet 文件、JSON 文件、或者通过使用 hiveql 查询 hive 数

据来建立。SchemaRDD 除了可以和 RDD 一样操作外，还可以通过 registerTempTable 注册成临时表，然后通过 SQL 语句进行操作。

值得注意的是：

- Spark1.1 使用 registerTempTable 代替 1.0 版本的 registerAsTable
- Spark1.1 在 hiveContext 中，hql() 将被弃用，sql() 将代替 hql() 来提交查询语句，统一了接口。
- 使用 registerTempTable 注册表是一个临时表，生命周期只在所定义的 sqlContext 或 hiveContext 实例之中。换言之，在一个 sqlContext (或 hiveContext) 中 registerTempTable 的表不能在另一个 sqlContext (或 hiveContext) 中使用。

另外，Spark1.1 提供了语法解析器选项 spark.sql.dialect，就目前而言，Spark1.1 提供了两种语法解析器：sql 语法解析器和 hiveql 语法解析器。

- sqlContext 现在只支持 sql 语法解析器 (SQL-92 语法)
- hiveContext 现在支持 sql 语法解析器和 hivesql 语法解析器，默认为 hivesql 语法解析器，用户可以通过配置切换成 sql 语法解析器，来运行 hiveql 不支持的语法，如 select 1。
- 切换可以通过下列方式完成：
- 在 sqlContext 中使用 setconf 配置 spark.sql.dialect
- 在 hiveContext 中使用 setconf 配置 spark.sql.dialect
- 在 sql 命令中使用 set spark.sql.dialect=value

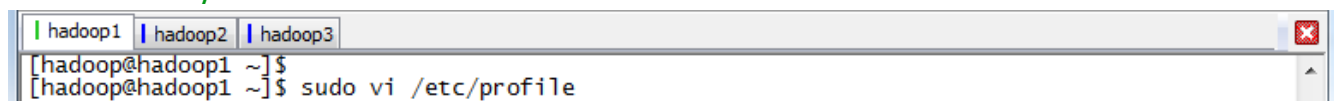
SparkSQL1.1 对数据的查询分成了 2 个分支：sqlContext 和 hiveContext。至于两者之间的关系，hiveSQL 继承了 sqlContext，所以拥有 sqlContext 的特性之外，还拥有自身的特性 (最大的特性就是支持 hive)。

2.1 启动 Spark shell

2.1.1 环境设置

使用如下命令打开/etc/profile 文件：

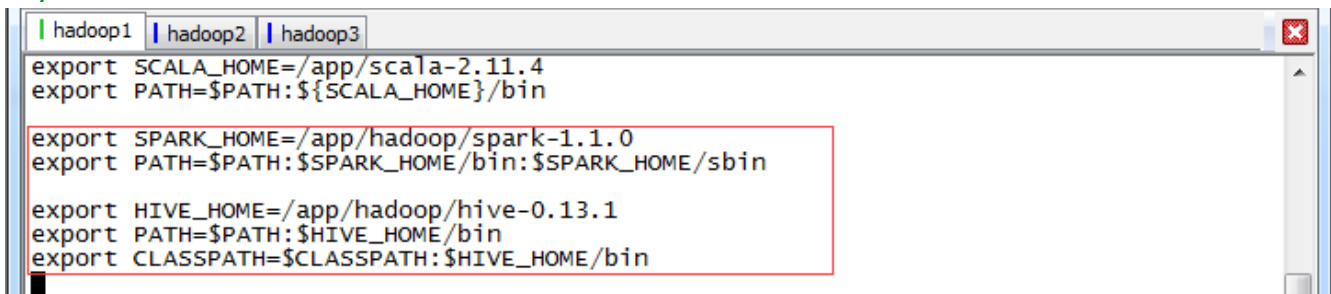
```
sudo vi /etc/profile
```



设置如下参数：

```
export SPARK_HOME=/app/hadoop/spark-1.1.0
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

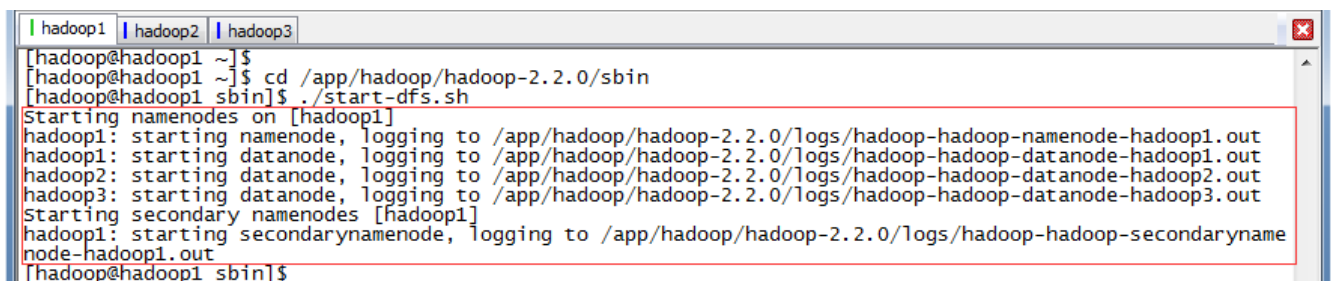
```
export HIVE_HOME=/app/hadoop/hive-0.13.1
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:$HIVE_HOME/bin
```



```
hadoop1 | hadoop2 | hadoop3
export SCALA_HOME=/app/scala-2.11.4
export PATH=$PATH:${SCALA_HOME}/bin
export SPARK_HOME=/app/hadoop/spark-1.1.0
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
export HIVE_HOME=/app/hadoop/hive-0.13.1
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:$HIVE_HOME/bin
```

2.1.2 启动 HDFS

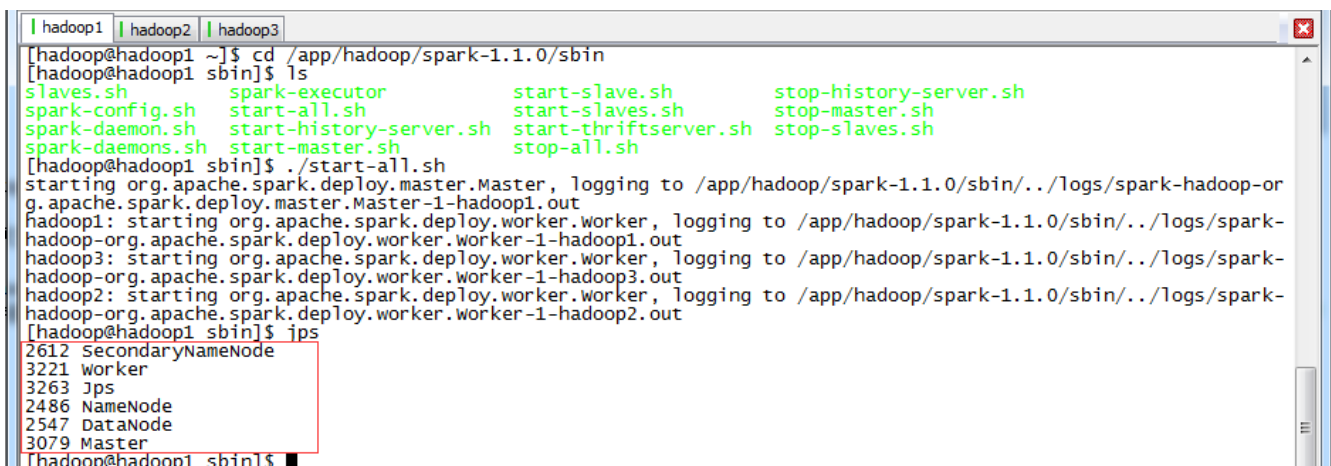
```
$cd /app/hadoop/hadoop-2.2.0/sbin
$./start-dfs.sh
```



```
hadoop1 | hadoop2 | hadoop3
[hadoop@hadoop1 ~]$
[hadoop@hadoop1 ~]$ cd /app/hadoop/hadoop-2.2.0/sbin
[hadoop@hadoop1 sbin]$ ./start-dfs.sh
Starting namenodes on [hadoop1]
hadoop1: starting namenode, logging to /app/hadoop/hadoop-2.2.0/logs/hadoop-hadoop-namenode-hadoop1.out
hadoop1: starting datanode, logging to /app/hadoop/hadoop-2.2.0/logs/hadoop-hadoop-datanode-hadoop1.out
hadoop2: starting datanode, logging to /app/hadoop/hadoop-2.2.0/logs/hadoop-hadoop-datanode-hadoop2.out
hadoop3: starting datanode, logging to /app/hadoop/hadoop-2.2.0/logs/hadoop-hadoop-datanode-hadoop3.out
Starting secondary namenodes [hadoop1]
hadoop1: starting secondarynamenode, logging to /app/hadoop/hadoop-2.2.0/logs/hadoop-hadoop-secondaryname
node-hadoop1.out
[hadoop@hadoop1 sbin]$
```

2.1.3 启动 Spark 集群

```
$cd /app/hadoop/spark-1.1.0/sbin
$./start-all.sh
```



```
hadoop1 | hadoop2 | hadoop3
[hadoop@hadoop1 ~]$ cd /app/hadoop/spark-1.1.0/sbin
[hadoop@hadoop1 sbin]$ ls
slaves.sh      spark-executor      start-slave.sh      stop-history-server.sh
spark-config.sh start-all.sh        start-slaves.sh     stop-master.sh
spark-daemon.sh start-history-server.sh start-thriftserver.sh stop-slaves.sh
spark-daemons.sh start-master.sh      stop-all.sh
[hadoop@hadoop1 sbin]$ ./start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /app/hadoop/spark-1.1.0/sbin/../logs/spark-hadoop-or
g.apache.spark.deploy.master.Master-1-hadoop1.out
hadoop1: starting org.apache.spark.deploy.worker.worker, logging to /app/hadoop/spark-1.1.0/sbin/../logs/spark-
hadoop-org.apache.spark.deploy.worker.worker-1-hadoop1.out
hadoop3: starting org.apache.spark.deploy.worker.worker, logging to /app/hadoop/spark-1.1.0/sbin/../logs/spark-
hadoop-org.apache.spark.deploy.worker.worker-1-hadoop3.out
hadoop2: starting org.apache.spark.deploy.worker.worker, logging to /app/hadoop/spark-1.1.0/sbin/../logs/spark-
hadoop-org.apache.spark.deploy.worker.worker-1-hadoop2.out
[hadoop@hadoop1 sbin]$ jps
2612 SecondaryNameNode
3221 worker
3263 Jps
2486 NameNode
2547 DataNode
3079 Master
[hadoop@hadoop1 sbin]$
```

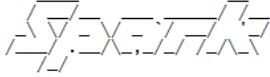
2.1.4 启动 Spark-Shell

在 spark 客户端 (在 hadoop1 节点) , 使用 spark-shell 连接集群

```
$cd /app/hadoop/spark-1.1.0/bin
```

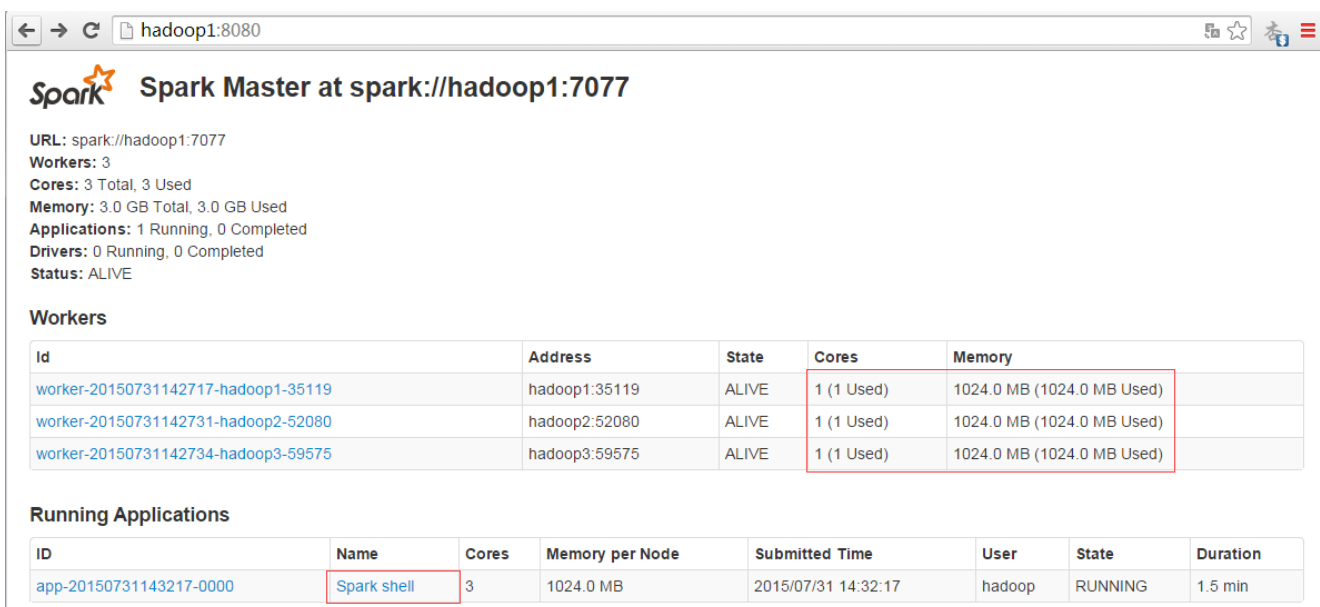
`./spark-shell --master spark://hadoop1:7077 --executor-memory 1g`

```
hadoop1 | hadoop2 | hadoop3
[hadoop@hadoop1 ~]$ cd /app/hadoop/spark-1.1.0/bin
[hadoop@hadoop1 bin]$ ./spark-shell --master spark://hadoop1:7077 --executor-memory 1g
Spark assembly has been built with Hive, including Datanucleus jars on classpath
15/07/31 14:31:52 INFO spark.SecurityManager: Changing view acls to: hadoop,
15/07/31 14:31:52 INFO spark.SecurityManager: Changing modify acls to: hadoop,
15/07/31 14:31:52 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view
Set(hadoop, ); users with modify permissions: Set(hadoop, )
15/07/31 14:31:52 INFO spark.HttpServer: Starting HTTP Server
15/07/31 14:31:52 INFO server.Server: jetty-8.y.z-SNAPSHOT
15/07/31 14:31:52 INFO util.AbstractConnector: Started SocketConnector@0.0.0.0:45666
15/07/31 14:31:52 INFO util.Utils: Successfully started service 'HTTP class server' on port 45666.
Welcome to

 version 1.1.0

using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_55)
Type in expressions to have them evaluated.
Type :help for more information.
15/07/31 14:32:03 INFO spark.SecurityManager: Changing view acls to: hadoop,
15/07/31 14:32:03 INFO spark.SecurityManager: Changing modify acls to: hadoop,
```

启动后查看启动情况，如下图所示：



Spark Master at spark://hadoop1:7077

URL: spark://hadoop1:7077
Workers: 3
Cores: 3 Total, 3 Used
Memory: 3.0 GB Total, 3.0 GB Used
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Id	Address	State	Cores	Memory
worker-20150731142717-hadoop1-35119	hadoop1:35119	ALIVE	1 (1 Used)	1024.0 MB (1024.0 MB Used)
worker-20150731142731-hadoop2-52080	hadoop2:52080	ALIVE	1 (1 Used)	1024.0 MB (1024.0 MB Used)
worker-20150731142734-hadoop3-59575	hadoop3:59575	ALIVE	1 (1 Used)	1024.0 MB (1024.0 MB Used)

Running Applications

ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20150731143217-0000	Spark shell	3	1024.0 MB	2015/07/31 14:32:17	hadoop	RUNNING	1.5 min

2.2 sqlContext 演示

Spark1.1.0 开始提供了两种方式将 RDD 转换成 SchemaRDD：

- 通过定义 Case Class，使用反射推断 Schema（case class 方式）
- 通过可编程接口，定义 Schema，并应用到 RDD 上（applySchema 方式）

前者使用简单、代码简洁，适用于已知 Schema 的源数据上；后者使用较为复杂，但可以在程序运行过程中实行，适用于未知 Schema 的 RDD 上。

2.2.1 使用 Case Class 定义 RDD 演示

对于 Case Class 方式，首先要定义 Case Class，在 RDD 的 Transform 过程中使用 Case Class 可以隐式转化成 SchemaRDD，然后再使用 registerTempTable 注册成表。注册成表后就可以在 sqlContext 对表进行操作，如 select、insert、join 等。注意，case class 可以是嵌套的，也可以使用类似 Sequences 或 Arrays 之类复杂的数据类型。

下面的例子是定义一个符合数据文件/sparksql/people.txt 类型的 case class (Person), 然后将数据文件读入后隐式转换成 SchemaRDD : people , 并将 people 在 sqlContext 中注册成表 rddTable , 最后对表进行查询 , 找出年纪在 13-19 岁之间的人名。

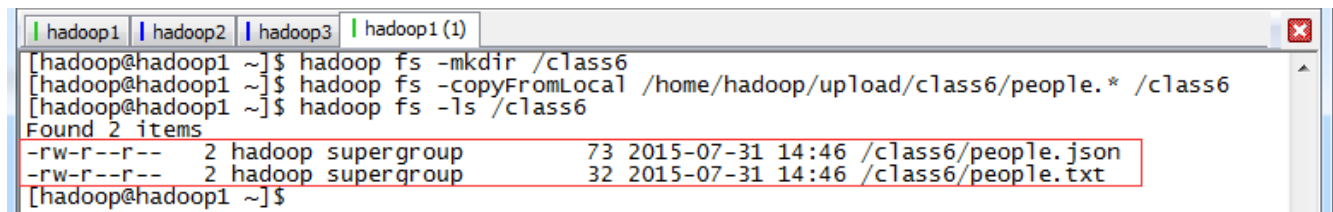
第一步 上传测试数据

在 HDFS 中创建/class6 目录 , 把配套资源/data/class5/ people.txt 上传到该目录上

```
$hadoop fs -mkdir /class6
```

```
$hadoop fs -copyFromLocal /home/hadoop/upload/class6/people.* /class6
```

```
$hadoop fs -ls /
```



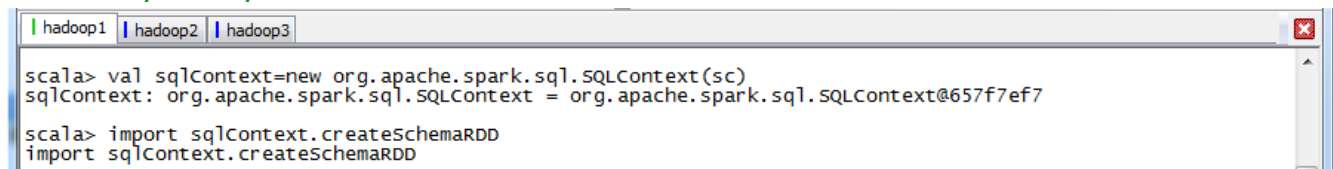
```
hadoop1 | hadoop2 | hadoop3 | hadoop1 (1)
[hadoop@hadoop1 ~]$ hadoop fs -mkdir /class6
[hadoop@hadoop1 ~]$ hadoop fs -copyFromLocal /home/hadoop/upload/class6/people.* /class6
[hadoop@hadoop1 ~]$ hadoop fs -ls /class6
Found 2 items
-rw-r--r--  2 hadoop supergroup          73 2015-07-31 14:46 /class6/people.json
-rw-r--r--  2 hadoop supergroup          32 2015-07-31 14:46 /class6/people.txt
[hadoop@hadoop1 ~]$
```

第二步 定义 sqlContext 并引入包

```
//sqlContext 演示
```

```
scala> val sqlContext=new org.apache.spark.sql.SQLContext(sc)
```

```
scala> import sqlContext.createSchemaRDD
```



```
hadoop1 | hadoop2 | hadoop3
scala> val sqlContext=new org.apache.spark.sql.SQLContext(sc)
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@657f7ef7

scala> import sqlContext.createSchemaRDD
import sqlContext.createSchemaRDD
```

第三步 定义 Person 类 , 读入数据并注册为临时表

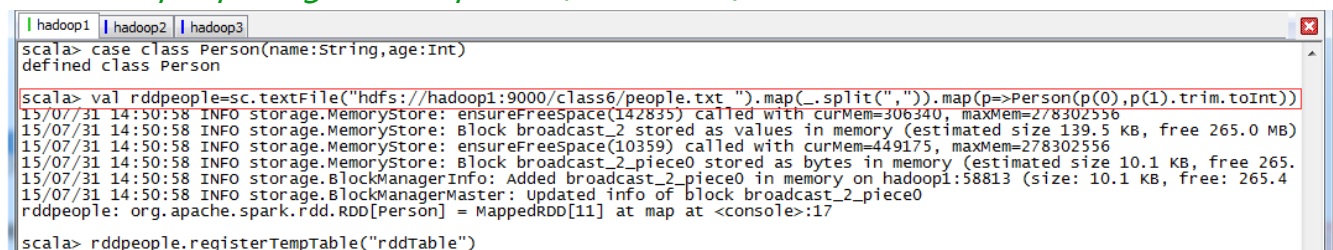
```
//RDD1 演示
```

```
scala> case class Person(name:String,age:Int)
```

```
scala> val
```

```
rddpeople=sc.textFile("hdfs://hadoop1:9000/class6/people.txt").map(_.split(",")).map(p
=>Person(p(0),p(1).trim.toInt))
```

```
scala> rddpeople.registerTempTable("rddTable")
```



```
hadoop1 | hadoop2 | hadoop3
scala> case class Person(name:String,age:Int)
defined class Person

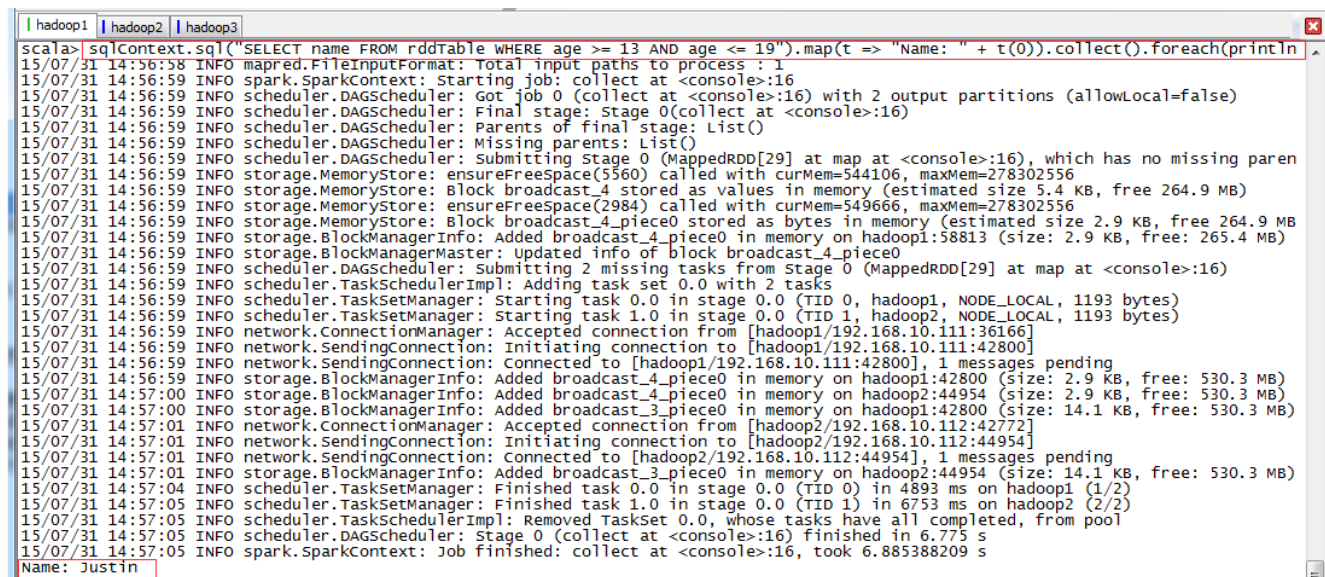
scala> val rddpeople=sc.textFile("hdfs://hadoop1:9000/class6/people.txt").map(_.split(",")).map(p=>Person(p(0),p(1).trim.toInt))
15/07/31 14:50:58 INFO storage.MemoryStore: ensureFreeSpace(142835) called with curMem=306340, maxMem=278302556
15/07/31 14:50:58 INFO storage.MemoryStore: Block broadcast_2 stored as values in memory (estimated size 139.5 KB, free 265.0 MB)
15/07/31 14:50:58 INFO storage.MemoryStore: ensureFreeSpace(10359) called with curMem=449175, maxMem=278302556
15/07/31 14:50:58 INFO storage.MemoryStore: Block broadcast_2_piece0 stored as bytes in memory (estimated size 10.1 KB, free 265.4 MB)
15/07/31 14:50:58 INFO storage.BlockManagerInfo: Added broadcast_2_piece0 in memory on hadoop1:58813 (size: 10.1 KB, free: 265.4 MB)
15/07/31 14:50:58 INFO storage.BlockManagerMaster: Updated info of block broadcast_2_piece0
rddpeople: org.apache.spark.rdd.RDD[Person] = MappedRDD[11] at map at <console>:17

scala> rddpeople.registerTempTable("rddTable")
```

第四步 在查询年纪在 13-19 岁之间的人员


```
scala>sqlContext.sql("SELECT name FROM rddTable WHERE age >= 13 AND age <= 19").map(t => "Name: " + t(0)).collect().foreach(println)
```

上面步骤均为 transform 未触发 action 动作，在该步骤中查询数据并打印触发了 action 动作，如下图所示：



```
scala>sqlContext.sql("SELECT name FROM rddTable WHERE age >= 13 AND age <= 19").map(t => "Name: " + t(0)).collect().foreach(println)
15/07/31 14:56:58 INFO mapred.FileInputFormat: Total input paths to process : 1
15/07/31 14:56:59 INFO spark.SparkContext: Starting job: collect at <console>:16
15/07/31 14:56:59 INFO scheduler.DAGScheduler: Got job 0 (collect at <console>:16) with 2 output partitions (allowLocal=false)
15/07/31 14:56:59 INFO scheduler.DAGScheduler: Final stage: Stage 0(collect at <console>:16)
15/07/31 14:56:59 INFO scheduler.DAGScheduler: Parents of final stage: List()
15/07/31 14:56:59 INFO scheduler.DAGScheduler: Missing parents: List()
15/07/31 14:56:59 INFO scheduler.DAGScheduler: Submitting Stage 0 (MappedRDD[29] at map at <console>:16), which has no missing paren
15/07/31 14:56:59 INFO storage.MemoryStore: ensureFreeSpace(3560) called with curMem=544106, maxMem=278302556
15/07/31 14:56:59 INFO storage.MemoryStore: Block broadcast_4 stored as values in memory (estimated size 3.4 KB, free 264.9 MB)
15/07/31 14:56:59 INFO storage.MemoryStore: ensureFreeSpace(2984) called with curMem=349666, maxMem=278302556
15/07/31 14:56:59 INFO storage.MemoryStore: Block broadcast_4_piece0 stored as bytes in memory (estimated size 2.9 KB, free 264.9 MB)
15/07/31 14:56:59 INFO storage.BlockManagerInfo: Added broadcast_4_piece0 in memory on hadoop1:58813 (size: 2.9 KB, free: 265.4 MB)
15/07/31 14:56:59 INFO storage.BlockManagerMaster: Updated info of block broadcast_4_piece0
15/07/31 14:56:59 INFO scheduler.DAGScheduler: Submitting 2 missing tasks from Stage 0 (MappedRDD[29] at map at <console>:16)
15/07/31 14:56:59 INFO scheduler.TaskSchedulerImpl: Adding task set 0.0 with 2 tasks
15/07/31 14:56:59 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, hadoop1, NODE_LOCAL, 1193 bytes)
15/07/31 14:56:59 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, hadoop2, NODE_LOCAL, 1193 bytes)
15/07/31 14:56:59 INFO network.ConnectionManager: Accepted connection from [hadoop1/192.168.10.111:36166]
15/07/31 14:56:59 INFO network.SendingConnection: Initiating connection to [hadoop1/192.168.10.111:42800]
15/07/31 14:56:59 INFO network.SendingConnection: Connected to [hadoop1/192.168.10.111:42800], 1 messages pending
15/07/31 14:56:59 INFO storage.BlockManagerInfo: Added broadcast_4_piece0 in memory on hadoop1:42800 (size: 2.9 KB, free: 530.3 MB)
15/07/31 14:57:00 INFO storage.BlockManagerInfo: Added broadcast_4_piece0 in memory on hadoop2:44954 (size: 2.9 KB, free: 530.3 MB)
15/07/31 14:57:00 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on hadoop1:42800 (size: 14.1 KB, free: 530.3 MB)
15/07/31 14:57:01 INFO network.ConnectionManager: Accepted connection from [hadoop2/192.168.10.112:42772]
15/07/31 14:57:01 INFO network.SendingConnection: Initiating connection to [hadoop2/192.168.10.112:44954]
15/07/31 14:57:01 INFO network.SendingConnection: Connected to [hadoop2/192.168.10.112:44954], 1 messages pending
15/07/31 14:57:01 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on hadoop2:44954 (size: 14.1 KB, free: 530.3 MB)
15/07/31 14:57:04 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 4893 ms on hadoop1 (1/2)
15/07/31 14:57:05 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
15/07/31 14:57:05 INFO scheduler.DAGScheduler: Stage 0 (collect at <console>:16) finished in 6.775 s
15/07/31 14:57:05 INFO spark.SparkContext: Job finished: collect at <console>:16, took 6.885388209 s
Name: Justin
```

通过监控页面，查看任务运行情况：

Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
0	collect at <console>:16	+details 2015/07/31 14:56:59	7 s	2/2	32.0 B		

Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input	Shuffle Read	Shuffle Write	Shuffle Spill (Memory)	Shuffle Spill (Disk)
1	hadoop1:42800	5 s	1	0	1	16.0 B	0.0 B	0.0 B	0.0 B	0.0 B
2	hadoop2:44954	7 s	1	0	1	16.0 B	0.0 B	0.0 B	0.0 B	0.0 B

Tasks

Index	ID	Attempt	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Accumulators	Input	Errors
0	0	0	SUCCESS	NODE_LOCAL	hadoop1	2015/07/31 14:56:59	4 s			16.0 B (hadoop)	
1	1	0	SUCCESS	NODE_LOCAL	hadoop2	2015/07/31 14:56:59	5 s			16.0 B (hadoop)	

2.2.2 使用 applySchema 定义 RDD 演示

applySchema 方式比较复杂，通常有 3 步过程：

- 从源 RDD 创建 rowRDD
- 创建与 rowRDD 匹配的 Schema
- 将 Schema 通过 applySchema 应用到 rowRDD

第一步 导入包创建 Schema

```
//导入 SparkSQL 的数据类型和 Row
```

```
scala>import org.apache.spark.sql._
```

```
//创建于数据结构匹配的 schema
```

```
scala>val schemaString = "name age"
```

```
scala>val schema =
```



```
StructType(
  schemaString.split(" ").map(fieldName => StructField(fieldName, StringType,
true)))
```

```
scala> import org.apache.spark.sql._
import org.apache.spark.sql._

scala> val schemaString = "name age"
schemaString: String = name age

scala> val schema = StructType(  schemaString.split(" ").map(fieldName => StructField(fieldName, s
tringType, true)))
schema: org.apache.spark.sql.catalyst.types.StructType = StructType(ArraySeq(StructField(name,StringT
ype,true), StructField(age,StringType,true)))
```

第二步 创建 rowRDD 并读入数据

//创建 rowRDD

```
scala> val rowRDD =
sc.textFile("hdfs://hadoop1:9000/class6/people.txt").map(_.split(",")).map(p =>
Row(p(0), p(1).trim))
```

//用 applySchema 将 schema 应用到 rowRDD

```
scala> val rddpeople2 = sqlContext.applySchema(rowRDD, schema)
```

```
scala> rddpeople2.registerTempTable("rddTable2")
```

```
scala> val rowRDD = sc.textFile("hdfs://hadoop1:9000/class6/people.txt").map(_.split(",")).map(p => Row(p(0), p(1).trim))
15/07/31 15:00:34 INFO storage.MemoryStore: ensureFreeSpace(182086) called with curMem=544106, maxMem=278302556
15/07/31 15:00:34 INFO storage.MemoryStore: ensureFreeSpace(14475) called with curMem=726192, maxMem=278302556
15/07/31 15:00:34 INFO storage.MemoryStore: Block broadcast_5 stored as values in memory (estimated size 177.8 KB, free 26
15/07/31 15:00:34 INFO storage.MemoryStore: Block broadcast_5_piece0 stored as bytes in memory (estimated size 14.1 KB, fr
15/07/31 15:00:34 INFO storage.BlockManagerInfo: Added broadcast_5_piece0 in memory on hadoop1:58813 (size: 14.1 KB, free:
15/07/31 15:00:34 INFO storage.BlockManagerMaster: updated info of block broadcast_5_piece0
rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.catalyst.expressions.Row] = MappedRDD[35] at map at <console>:18

scala> val rddpeople2 = sqlContext.applySchema(rowRDD, schema)
rddpeople2: org.apache.spark.sql.SchemaRDD =
SchemaRDD[36] at RDD at SchemaRDD.scala:103
== Query Plan ==
== Physical Plan ==
ExistingRdd [name#4,age#5], MappedRDD[35] at map at <console>:18

scala> rddpeople2.registerTempTable("rddTable2")
```

第三步 查询获取数据

```
scala> sqlContext.sql("SELECT name FROM rddTable2 WHERE age >= 13 AND age <=
19").map(t => "Name: " + t(0)).collect().foreach(println)
```

```
scala> sqlContext.sql("SELECT name FROM rddTable2 WHERE age >= 13 AND age <= 19").map(t => "Name: " + t(0)).collect().foreach(println)
15/07/31 15:03:24 INFO mapped.FileInputFormat: Total input paths to process : 1
15/07/31 15:03:24 INFO spark.SPARKContext: Starting job: collect at <console>:19
15/07/31 15:03:24 INFO scheduler.DAGScheduler: Got job 1 (collect at <console>:19) with 2 output partitions (allowLocal=false)
15/07/31 15:03:24 INFO scheduler.DAGScheduler: Final stage: stage 1(collect at <console>:19)
15/07/31 15:03:24 INFO scheduler.DAGScheduler: Parents of final stage: List()
15/07/31 15:03:24 INFO scheduler.DAGScheduler: Submitting Stage 1 (MappedRDD[38] at map at <console>:19), which has no missing parent
15/07/31 15:03:24 INFO storage.MemoryStore: ensureFreeSpace(5616) called with curMem=740667, maxMem=278302556
15/07/31 15:03:24 INFO storage.MemoryStore: Block broadcast_6 stored as values in memory (estimated size 5.5 KB, free 264.7 MB)
15/07/31 15:03:24 INFO storage.MemoryStore: ensureFreeSpace(3016) called with curMem=746283, maxMem=278302556
15/07/31 15:03:24 INFO storage.MemoryStore: Block broadcast_6_piece0 stored as bytes in memory (estimated size 2.9 KB, free 264.7 MB)
15/07/31 15:03:24 INFO storage.BlockManagerInfo: Added broadcast_6_piece0 in memory on hadoop1:58813 (size: 2.9 KB, free: 265.3 MB)
15/07/31 15:03:24 INFO storage.BlockManagerMaster: Updated info of block broadcast_6_piece0
15/07/31 15:03:24 INFO scheduler.DAGScheduler: Submitting 2 missing tasks from Stage 1 (MappedRDD[38] at map at <console>:19)
15/07/31 15:03:24 INFO scheduler.TaskSchedulerImpl: Adding task set 1.0 with 2 tasks
15/07/31 15:03:24 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 1.0 (TID 2, hadoop1, NODE_LOCAL, 1193 bytes)
15/07/31 15:03:24 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 1.0 (TID 3, hadoop2, NODE_LOCAL, 1193 bytes)
15/07/31 15:03:24 INFO storage.BlockManagerInfo: Added broadcast_6_piece0 in memory on hadoop2:44954 (size: 2.9 KB, free: 530.3 MB)
15/07/31 15:03:24 INFO storage.BlockManagerInfo: Added broadcast_6_piece0 in memory on hadoop1:42800 (size: 2.9 KB, free: 530.3 MB)
15/07/31 15:03:24 INFO storage.BlockManagerInfo: Added broadcast_5_piece0 in memory on hadoop1:42800 (size: 14.1 KB, free: 530.2 MB)
15/07/31 15:03:24 INFO storage.BlockManagerInfo: Added broadcast_5_piece0 in memory on hadoop2:44954 (size: 14.1 KB, free: 530.2 MB)
15/07/31 15:03:25 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 1.0 (TID 2) in 544 ms on hadoop1 (1/2)
15/07/31 15:03:25 INFO scheduler.DAGScheduler: stage 1 (collect at <console>:19) finished in 0.566 s
15/07/31 15:03:25 INFO scheduler.TaskSetManager: finished task 1.0 in stage 1.0 (TID 3) in 563 ms on hadoop2 (2/2)
15/07/31 15:03:25 INFO spark.SPARKContext: Job finished: collect at <console>:19, took 0.585591433 s
15/07/31 15:03:25 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
Name: Justin
```

通过监控页面，查看任务运行情况：

Completed Stages (2)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
1	collect at <console>:19	+details 2015/07/31 15:03:24	0.6 s	2/2	32.0 B		
0	collect at <console>:16	+details 2015/07/31 14:56:59	7 s	2/2	32.0 B		

Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input	Shuffle Read	Shuffle Write	Shuffle Spill (Memory)	Shuffle Spill (Disk)
1	hadoop1:42800	0.5 s	1	0	1	16.0 B	0.0 B	0.0 B	0.0 B	0.0 B
2	hadoop2:44954	0.6 s	1	0	1	16.0 B	0.0 B	0.0 B	0.0 B	0.0 B

Tasks

Index	ID	Attempt	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Accumulators	Input	Errors
0	2	0	SUCCESS	NODE_LOCAL	hadoop1	2015/07/31 15:03:24	0.5 s			16.0 B (hadoop)	
1	3	0	SUCCESS	NODE_LOCAL	hadoop2	2015/07/31 15:03:24	0.5 s			16.0 B (hadoop)	

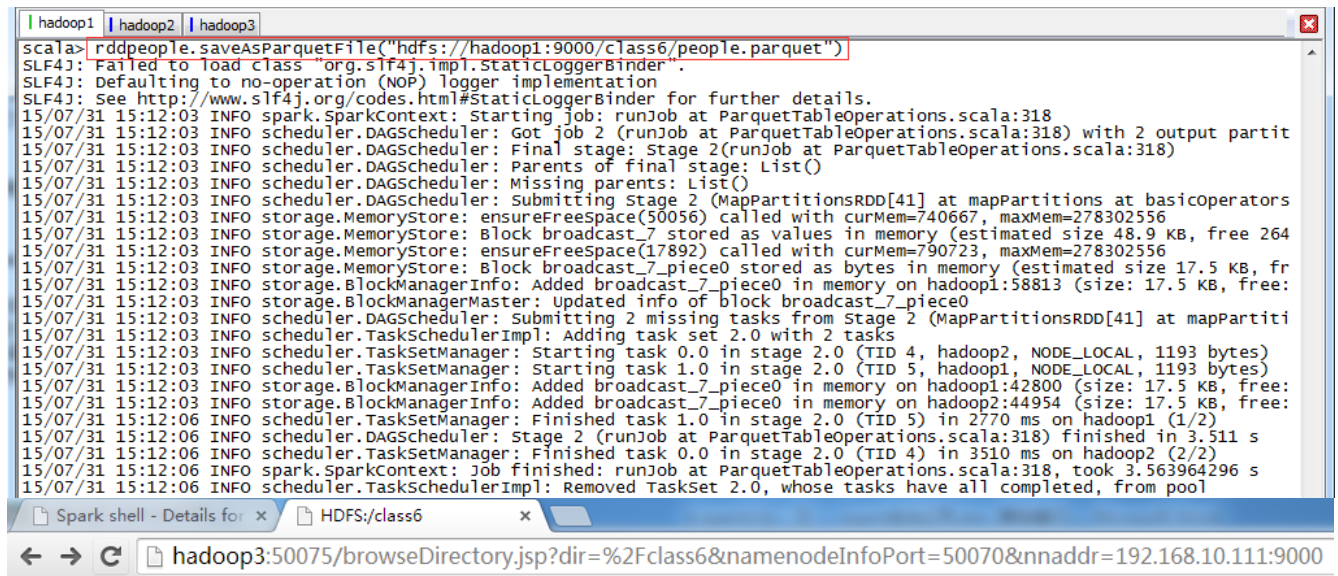
2.2.3 parquet 演示

同样得，sqlContext 可以读取 parquet 文件，由于 parquet 文件中保留了 schema 的信息，所以不需要使用 case class 来隐式转换。sqlContext 读入 parquet 文件后直接转换成 SchemaRDD，也可以将 SchemaRDD 保存成 parquet 文件格式。

第一步 保存成 parquest 格式文件

// 把上面步骤中的 rddpeople 保存为 parquet 格式文件到 hdfs 中

scala>rddpeople.saveAsParquetFile("hdfs://hadoop1:9000/class6/people.parquet")



Contents of directory /class6

Goto : go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
people.json	file	73 B	2	128 MB	2015-07-31 14:46	rw-r--r--	hadoop	supergroup
people.parquet	dir				2015-07-31 15:12	rw-r--r--	hadoop	supergroup
people.txt	file	32 B	2	128 MB	2015-07-31 14:46	rw-r--r--	hadoop	supergroup

第二步 读入 parquest 格式文件，注册表 parquetTable

//parquet 演示

```
scala> val parquetpeople =
```

```
sqlContext.parquetFile("hdfs://hadoop1:9000/class6/people.parquet")
```

```
scala> parquetpeople.registerTempTable("parquetTable")
```

```
scala> val parquetpeople = sqlContext.parquetFile("hdfs://hadoop1:9000/class6/people.parquet")
parquetpeople: org.apache.spark.sql.SchemaRDD =
SchemaRDD[44] at RDD at SchemaRDD.scala:103
== Query Plan ==
== Physical Plan ==
ParquetTableScan [name#10,age#11], (ParquetRelation hdfs://hadoop1:9000/class6/people.parquet, Some(Configuration: core-de
mapred-site.xml, yarn-default.xml, yarn-site.xml, hdfs-default.xml, hdfs-site.xml), org.apache.spark.sql.SQLContext@657f7
scala> parquetpeople.registerTempTable("parquetTable")
```

第三步 查询年龄大于等于 25 岁的人名

```
scala> sqlContext.sql("SELECT name FROM parquetTable WHERE age >= 25").map(t =>
"Name: " + t(0)).collect().foreach(println)
```

```
scala> sqlContext.sql("SELECT name FROM parquetTable WHERE age >= 25").map(t => "Name: " + t(0)).collect().foreach(println)
15/07/31 15:13:41 INFO storage.BlockManager: Removing broadcast_7
15/07/31 15:13:41 INFO storage.BlockManager: Removing block broadcast_7
15/07/31 15:13:41 INFO storage.MemoryStore: block broadcast_7 of size 50056 dropped from memory (free 277543997)
15/07/31 15:13:41 INFO storage.BlockManagerInfo: Removed broadcast_7_piece0 on hadoop2:44954 in memory (size: 17.5 KB, free: 530.3 MB)
15/07/31 15:13:41 INFO storage.BlockManager: Removing block broadcast_7_piece0
15/07/31 15:13:41 INFO storage.MemoryStore: block broadcast_7_piece0 of size 17892 dropped from memory (free 277561889)
15/07/31 15:13:41 INFO storage.BlockManagerInfo: Removed broadcast_7_piece0 on hadoop1:42800 in memory (size: 17.5 KB, free: 530.3 MB)
15/07/31 15:13:41 INFO storage.BlockManagerInfo: Removed broadcast_7_piece0 on hadoop1:58813 in memory (size: 17.5 KB, free: 265.4 MB)
15/07/31 15:13:41 INFO storage.BlockManagerMaster: Updated info of block broadcast_7_piece0
15/07/31 15:13:41 INFO spark.ContextCleaner: Cleaned broadcast_7
15/07/31 15:13:41 INFO storage.MemoryStore: ensureFreeSpace(192366) called with curMem=740667, maxMem=278302556
15/07/31 15:13:41 INFO storage.MemoryStore: block broadcast_8 stored as values in memory (estimated size 187.9 KB, free 264.5 MB)
15/07/31 15:13:42 INFO storage.MemoryStore: ensureFreeSpace(16356) called with curMem=933033, maxMem=278302556
15/07/31 15:13:42 INFO storage.MemoryStore: block broadcast_8_piece0 stored as bytes in memory (estimated size 16.0 KB, free 264.5 MB)
15/07/31 15:13:42 INFO storage.BlockManagerInfo: Added broadcast_8_piece0 in memory on hadoop1:58813 (size: 16.0 KB, free: 265.3 MB)
15/07/31 15:13:42 INFO storage.BlockManagerMaster: Updated info of block broadcast_8_piece0
15/07/31 15:13:42 INFO input.FileInputFormat: Total input paths to process : 2
15/07/31 15:13:42 INFO hadoop.ParquetInputFormat: Total input paths to process : 2
15/07/31 15:13:42 INFO hadoop.ParquetFileReader: reading summary file: hdfs://hadoop1:9000/class6/people.parquet/_metadata
15/07/31 15:13:42 INFO Configuration.deprecation: mapred.max.split.size is deprecated. Instead, use mapreduce.input.fileinputformat.split.maxsize
15/07/31 15:13:42 INFO Configuration.deprecation: mapred.min.split.size is deprecated. Instead, use mapreduce.input.fileinputformat.split.minsize
15/07/31 15:13:42 INFO spark.SparkContext: Starting job: collect at <console>:19
15/07/31 15:13:42 INFO scheduler.DAGScheduler: Got job 3 (collect at <console>:19) with 2 output partitions (allowLocal=false)
15/07/31 15:13:42 INFO scheduler.DAGScheduler: Final stage: Stage 3 (collect at <console>:19)
15/07/31 15:13:42 INFO scheduler.DAGScheduler: Parents of final stage: List()
15/07/31 15:13:42 INFO scheduler.DAGScheduler: Missing parents: List()
15/07/31 15:13:42 INFO scheduler.DAGScheduler: Submitting Stage 3 (MappedRDD[46] at map at <console>:19), which has no missing parents
15/07/31 15:13:42 INFO storage.MemoryStore: ensureFreeSpace(5416) called with curMem=949389, maxMem=278302556
15/07/31 15:13:42 INFO storage.MemoryStore: block broadcast_9 stored as values in memory (estimated size 5.3 KB, free 264.5 MB)
15/07/31 15:13:42 INFO storage.MemoryStore: ensureFreeSpace(2965) called with curMem=954805, maxMem=278302556
15/07/31 15:13:42 INFO storage.MemoryStore: block broadcast_9_piece0 stored as bytes in memory (estimated size 2.9 KB, free 264.5 MB)
15/07/31 15:13:42 INFO storage.BlockManagerInfo: Added broadcast_9_piece0 in memory on hadoop1:58813 (size: 2.9 KB, free: 265.3 MB)
15/07/31 15:13:42 INFO storage.BlockManagerMaster: Updated info of block broadcast_9_piece0
15/07/31 15:13:42 INFO scheduler.DAGScheduler: Submitting 2 missing tasks from stage 3 (MappedRDD[46] at map at <console>:19)
15/07/31 15:13:42 INFO scheduler.TaskSchedulerImpl: adding task set 3.0 with 2 tasks
15/07/31 15:13:42 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 3.0 (TID 6, hadoop2, NODE_LOCAL, 2062 bytes)
15/07/31 15:13:42 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 3.0 (TID 7, hadoop1, NODE_LOCAL, 2062 bytes)
15/07/31 15:13:42 INFO storage.BlockManagerInfo: Added broadcast_9_piece0 in memory on hadoop1:42800 (size: 2.9 KB, free: 530.2 MB)
15/07/31 15:13:42 INFO storage.BlockManagerInfo: Added broadcast_9_piece0 in memory on hadoop2:44954 (size: 2.9 KB, free: 530.2 MB)
15/07/31 15:13:42 INFO storage.BlockManagerInfo: Added broadcast_8_piece0 in memory on hadoop1:42800 (size: 16.0 KB, free: 530.2 MB)
15/07/31 15:13:42 INFO storage.BlockManagerInfo: Added broadcast_8_piece0 in memory on hadoop2:44954 (size: 16.0 KB, free: 530.2 MB)
15/07/31 15:13:42 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 3.0 (TID 6) in 685 ms on hadoop2 (1/2)
15/07/31 15:13:43 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 3.0 (TID 7) in 846 ms on hadoop1 (2/2)
15/07/31 15:13:43 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
15/07/31 15:13:43 INFO scheduler.DAGScheduler: Stage 3 (collect at <console>:19) finished in 0.849 s
15/07/31 15:13:43 INFO spark.SparkContext: Job finished: collect at <console>:19, took 0.879519739 s
Name: Michael
Name: Andy
```

2.2.4 json 演示

sparkSQL1.1.0 开始提供对 json 文件格式的支持，这意味着开发者可以使用更多的数据源，如鼎鼎大名的 NOSQL 数据库 MongoDB 等。sqlContext 可以从 jsonFile 或 jsonRDD 获取 schema 信息，来构建 SchemaRDD，注册成表后就可以使用。

- jsonFile - 加载 JSON 文件目录中的数据，文件的每一行是一个 JSON 对象
- jsonRdd - 从现有的 RDD 加载数据，其中 RDD 的每个元素包含一个 JSON 对象的字符串

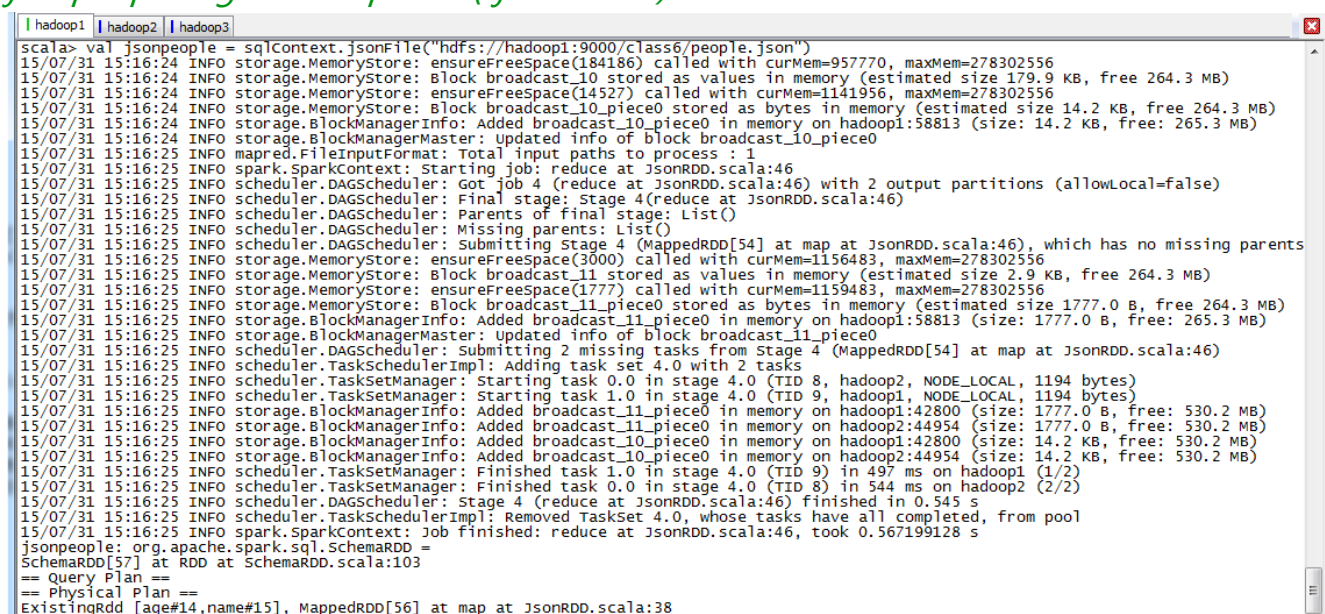
第一步 上传测试数据



第二步 读取数据并注册 jsonTable 表

//json 演示

```
scala> val jsonpeople = sqlContext.jsonFile("hdfs://hadoop1:9000/class6/people.json")
jsonpeople.registerTempTable("jsonTable")
```



第三步 查询年龄大于等于 25 的人名

```
scala> sqlContext.sql("SELECT name FROM jsonTable WHERE age >= 25").map(t =>
  "Name: " + t(0)).collect().foreach(println)
```

```
scala> sqlContext.sql("SELECT name FROM jsonTable WHERE age >= 25").map(t => "Name: " + t(0)).collect().foreach(println)
15/07/31 15:16:34 INFO storage.BlockManager: Removing broadcast 11
15/07/31 15:16:34 INFO storage.BlockManager: Removing block broadcast_11
15/07/31 15:16:34 INFO storage.MemoryStore: Block broadcast_11 of size 3000 dropped from memory (free 277144296)
15/07/31 15:16:34 INFO storage.BlockManager: Removing block broadcast_11_piece0
15/07/31 15:16:34 INFO storage.MemoryStore: Block broadcast_11_piece0 of size 1777 dropped from memory (free 277146073)
15/07/31 15:16:34 INFO storage.BlockManagerInfo: Removed broadcast_11_piece0 on hadoop1:58813 in memory (size: 1777.0 B, free: 265.3 MB)
15/07/31 15:16:34 INFO storage.BlockManagerMaster: updated info of block broadcast_11_piece0
15/07/31 15:16:34 INFO storage.BlockManagerInfo: Removed broadcast_11_piece0 on hadoop1:42800 in memory (size: 1777.0 B, free: 530.2 MB)
15/07/31 15:16:34 INFO storage.BlockManagerInfo: Removed broadcast_11_piece0 on hadoop2:44954 in memory (size: 1777.0 B, free: 530.2 MB)
15/07/31 15:16:34 INFO spark.ContextCleaner: Cleaned broadcast 11
15/07/31 15:16:34 INFO storage.BlockManager: Removing broadcast 9
15/07/31 15:16:34 INFO storage.MemoryStore: Block broadcast_9 of size 5416 dropped from memory (free 277151489)
15/07/31 15:16:34 INFO storage.BlockManager: Removing block broadcast_9_piece0
15/07/31 15:16:34 INFO storage.MemoryStore: Block broadcast_9_piece0 of size 2965 dropped from memory (free 277154454)
15/07/31 15:16:34 INFO storage.BlockManagerInfo: Removed broadcast_9_piece0 on hadoop1:58813 in memory (size: 2.9 KB, free: 265.3 MB)
15/07/31 15:16:34 INFO storage.BlockManagerMaster: updated info of block broadcast_9_piece0
15/07/31 15:16:34 INFO storage.BlockManagerInfo: Removed broadcast_9_piece0 on hadoop1:42800 in memory (size: 2.9 KB, free: 530.2 MB)
15/07/31 15:16:34 INFO storage.BlockManagerInfo: Removed broadcast_9_piece0 on hadoop2:44954 in memory (size: 2.9 KB, free: 530.2 MB)
15/07/31 15:16:34 INFO spark.ContextCleaner: Cleaned broadcast 9
15/07/31 15:16:34 INFO spark.SparkContext: Starting job: collect at <console>:19
15/07/31 15:16:34 INFO scheduler.DAGScheduler: Got job 5 (collect at <console>:19) with 2 output partitions (allowLocal=false)
15/07/31 15:16:34 INFO scheduler.DAGScheduler: Final stage: Stage 5 (collect at <console>:19)
15/07/31 15:16:34 INFO scheduler.DAGScheduler: Parents of final stage: List()
15/07/31 15:16:34 INFO scheduler.DAGScheduler: Missing parents: List()
15/07/31 15:16:34 INFO scheduler.DAGScheduler: Submitting Stage 5 (MappedRDD[59] at map at <console>:19), which has no missing parents
15/07/31 15:16:34 INFO storage.MemoryStore: ensureFreeSpace(5688) called with curMem=1148102, maxMem=278302556
15/07/31 15:16:34 INFO storage.MemoryStore: Block broadcast_12 stored as values in memory (estimated size 5.6 KB, free 264.3 MB)
15/07/31 15:16:34 INFO storage.MemoryStore: ensureFreeSpace(3154) called with curMem=1153790, maxMem=278302556
15/07/31 15:16:34 INFO storage.MemoryStore: Block broadcast_12_piece0 stored as bytes in memory (estimated size 3.1 KB, free 264.3 MB)
15/07/31 15:16:34 INFO storage.BlockManagerInfo: Added broadcast_12_piece0 in memory on hadoop1:58813 (size: 3.1 KB, free: 265.3 MB)
15/07/31 15:16:34 INFO storage.BlockManagerMaster: updated info of block broadcast_12_piece0
15/07/31 15:16:34 INFO scheduler.DAGScheduler: Submitting 2 missing tasks from Stage 5 (MappedRDD[59] at map at <console>:19)
15/07/31 15:16:34 INFO scheduler.TaskSchedulerImpl: Adding task set 5.0 with 2 tasks
15/07/31 15:16:34 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 5.0 (TID 10, hadoop1, NODE_LOCAL, 1194 bytes)
15/07/31 15:16:34 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 5.0 (TID 11, hadoop2, NODE_LOCAL, 1194 bytes)
15/07/31 15:16:34 INFO storage.BlockManagerInfo: Added broadcast_12_piece0 in memory on hadoop1:42800 (size: 3.1 KB, free: 530.2 MB)
15/07/31 15:16:34 INFO storage.BlockManagerInfo: Added broadcast_12_piece0 in memory on hadoop2:44954 (size: 3.1 KB, free: 530.2 MB)
15/07/31 15:16:34 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 5.0 (TID 11) in 129 ms on hadoop2 (1/2)
15/07/31 15:16:34 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 5.0 (TID 10) in 143 ms on hadoop1 (2/2)
15/07/31 15:16:34 INFO scheduler.DAGScheduler: Stage 5 (collect at <console>:19) finished in 0.140 s
15/07/31 15:16:34 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 5.0, whose tasks have all completed, from pool
15/07/31 15:16:34 INFO spark.SparkContext: Job finished: collect at <console>:19, took 0.167409676 s
Name: Andy
```

2.2.5 sqlContext 中混合使用演示

在 sqlContext 或 hiveContext 中来源于不同数据源的表在各自生命周期中可以混用，即 sqlContext 与 hiveContext 之间表不能混合使用

//sqlContext 中来自 rdd 的表 rddTable 和来自 parquet 文件的表 parquetTable 混合使用

scala> sqlContext.sql("select a.name,a.age,b.age from rddTable a join parquetTable b on a.name=b.name").collect().foreach(println)

```
scala> sqlContext.sql("select a.name,a.age,b.age from rddTable a join parquetTable b on a.name=b.name").collect().foreach(println)
15/07/31 15:19:33 INFO storage.BlockManager: Removing broadcast 19
15/07/31 15:19:33 INFO storage.BlockManager: Removing block broadcast_19
15/07/31 15:19:33 INFO storage.MemoryStore: Block broadcast_19 of size 6264 dropped from memory (free 276707880)
15/07/31 15:19:33 INFO storage.BlockManager: Removing block broadcast_19_piece0
15/07/31 15:19:33 INFO storage.MemoryStore: Block broadcast_19_piece0 of size 3500 dropped from memory (free 276711380)
15/07/31 15:19:33 INFO storage.BlockManagerInfo: Removed broadcast_19_piece0 on hadoop1:58813 in memory (size: 3.4 KB, free: 265.3 MB)
15/07/31 15:19:33 INFO storage.BlockManagerMaster: updated info of block broadcast_19_piece0
15/07/31 15:19:33 INFO storage.BlockManagerInfo: Removed broadcast_19_piece0 on hadoop2:44954 in memory (size: 3.4 KB, free: 530.2 MB)
15/07/31 15:19:33 INFO storage.BlockManagerInfo: Removed broadcast_19_piece0 on hadoop3:38322 in memory (size: 3.4 KB, free: 530.2 MB)
15/07/31 15:19:33 INFO spark.ContextCleaner: Cleaned broadcast 19
15/07/31 15:19:33 INFO storage.BlockManager: Removing broadcast 18
15/07/31 15:19:33 INFO storage.BlockManager: Removing block broadcast_18_piece0
15/07/31 15:18:30 INFO scheduler.TaskSetManager: Finished task 174.0 in stage 6.0 (TID 210) in 32 ms on hadoop2 (199/200)
15/07/31 15:18:30 INFO scheduler.TaskSetManager: Starting task 198.0 in stage 6.0 (TID 214, hadoop3, PROCESS_LOCAL, 1365 bytes)
15/07/31 15:18:30 INFO scheduler.TaskSetManager: Finished task 196.0 in stage 6.0 (TID 212) in 30 ms on hadoop3 (196/200)
15/07/31 15:18:30 INFO scheduler.TaskSetManager: Starting task 199.0 in stage 6.0 (TID 215, hadoop2, PROCESS_LOCAL, 1365 bytes)
15/07/31 15:18:30 INFO scheduler.TaskSetManager: Finished task 197.0 in stage 6.0 (TID 213) in 42 ms on hadoop2 (197/200)
15/07/31 15:18:30 INFO scheduler.TaskSetManager: Finished task 198.0 in stage 6.0 (TID 214) in 30 ms on hadoop3 (198/200)
15/07/31 15:18:30 INFO scheduler.TaskSetManager: Finished task 195.0 in stage 6.0 (TID 211) in 100 ms on hadoop1 (199/200)
15/07/31 15:18:30 INFO scheduler.TaskSetManager: Finished task 199.0 in stage 6.0 (TID 215) in 26 ms on hadoop2 (200/200)
15/07/31 15:18:30 INFO scheduler.DAGScheduler: Stage 6 (collect at SparkPlan.scala:85) finished in 5.151 s
15/07/31 15:18:30 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 6.0, whose tasks have all completed, from pool
15/07/31 15:18:30 INFO spark.SparkContext: Job finished: collect at SparkPlan.scala:85, took 13.126199935 s
[Justin,19,19]
[Andy,30,30]
[Michael,29,29]
```

2.3 hiveContext 演示

使用 hiveContext 之前首先要确认以下两点：

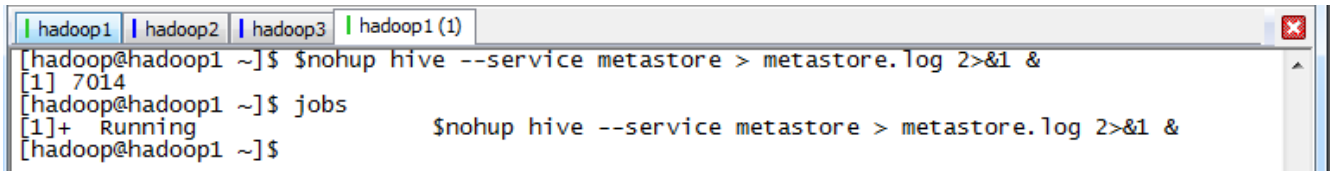
- 使用的 Spark 是支持 hive
- Hive 的配置文件 hive-site.xml 已经存在 conf 目录中

前者可以查看 lib 目录下是否存在以 datanucleus 开头的 3 个 JAR 来确定，后者注意是否在 hive-site.xml 里配置了 uris 来访问 Hive Metastore。

2.3.1 启动 hive

在 hadoop1 节点中使用如下命令启动 Hive

```
$nohup hive --service metastore > metastore.log 2>&1 &
```



```
[hadoop@hadoop1 ~]$ $nohup hive --service metastore > metastore.log 2>&1 &
[1] 7014
[hadoop@hadoop1 ~]$ jobs
[1]+  Running                  $nohup hive --service metastore > metastore.log 2>&1 &
[hadoop@hadoop1 ~]$
```

2.3.2 在 SPARK_HOME/conf 目录下创建 hive-site.xml

在 SPARK_HOME/conf 目录下创建 hive-site.xml 文件，修改配置后需要重新启动 Spark-Shell

【注】如果在第 6 课《SparkSQL (二) --SparkSQL 简介》配置，

```
<configuration>
```

```
<property>
```

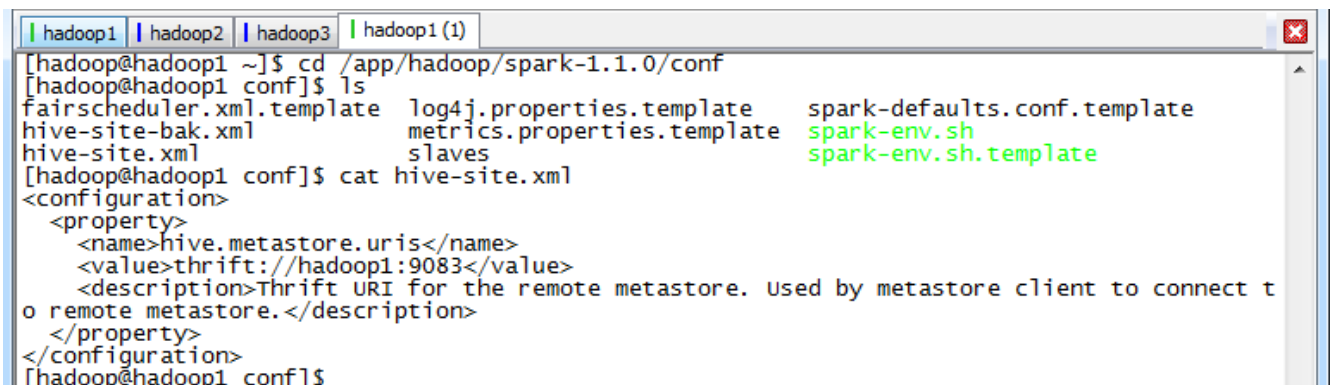
```
<name>hive.metastore.uris</name>
```

```
<value>thrift://hadoop1:9083</value>
```

```
<description>Thrift URI for the remote metastore. Used by metastore client to  
connect to remote metastore.</description>
```

```
</property>
```

```
</configuration>
```



```
[hadoop@hadoop1 ~]$ cd /app/hadoop/spark-1.1.0/conf
[hadoop@hadoop1 conf]$ ls
fairscheduler.xml.template  log4j.properties.template  spark-defaults.conf.template
hive-site-bak.xml           metrics.properties.template  spark-env.sh
hive-site.xml               slaves                       spark-env.sh.template
[hadoop@hadoop1 conf]$ cat hive-site.xml
<configuration>
  <property>
    <name>hive.metastore.uris</name>
    <value>thrift://hadoop1:9083</value>
    <description>Thrift URI for the remote metastore. Used by metastore client to connect t
o remote metastore.</description>
  </property>
</configuration>
[hadoop@hadoop1 conf]$
```

2.3.3 查看数据库表

要使用 hiveContext，需要先构建 hiveContext：

```
scala>val hiveContext = new org.apache.spark.sql.hive.HiveContext(sc)
```



```
scala> val hiveContext = new org.apache.spark.sql.hive.HiveContext(sc)
15/07/31 15:30:33 INFO Configuration.deprecation: mapred.input.dir.recursive is deprecated. Instead, use mapreduce.input.fileinputformat
15/07/31 15:30:33 INFO Configuration.deprecation: mapred.max.split.size is deprecated. Instead, use mapreduce.input.fileinputformat.spli
15/07/31 15:30:33 INFO Configuration.deprecation: mapred.min.split.size is deprecated. Instead, use mapreduce.input.fileinputformat.spli
15/07/31 15:30:33 INFO Configuration.deprecation: mapred.min.split.size.per.rack is deprecated. Instead, use mapreduce.input.fileinputputfo
15/07/31 15:30:33 INFO Configuration.deprecation: mapred.min.split.size.per.node is deprecated. Instead, use mapreduce.input.fileinputputfo
15/07/31 15:30:33 INFO Configuration.deprecation: mapred.reduce.tasks is deprecated. Instead, use mapreduce.job.reduces
15/07/31 15:30:33 INFO Configuration.deprecation: mapred.reduce.tasks.speculative.execution is deprecated. Instead, use mapreduce.reduce
hiveContext: org.apache.spark.sql.hive.HiveContext = org.apache.spark.sql.hive.HiveContext@16c6db21
```

然后就可以对 Hive 数据进行操作了，下面我们将使用 Hive 中的销售数据，首先切换数据库到 hive 并查看有几个表：

//销售数据演示

scala> hiveContext.sql("use hive")

```
scala> hiveContext.sql("use hive")
15/07/31 15:31:01 INFO parse.ParseDriver: Parsing command: use hive
15/07/31 15:31:01 INFO parse.ParseDriver: Parse Completed
15/07/31 15:31:02 INFO Configuration.deprecation: mapred.input.dir.recursive is deprecated. Instead, use mapreduce.input.fileinputformat
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=Driver.run>
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=TimeToSubmit>
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=compile>
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=parse>
15/07/31 15:31:02 INFO parse.ParseDriver: Parsing command: use hive
15/07/31 15:31:02 INFO parse.ParseDriver: Parse Completed
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=parse start=1438327862571 end=1438327862571 duration=0>
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=semanticAnalyze>
15/07/31 15:31:02 INFO ql.Driver: Semantic Analysis Completed
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=semanticAnalyze start=1438327862572 end=1438327862677 duration=105>
15/07/31 15:31:02 INFO ql.Driver: Returning Hive schema: Schema(fieldsSchemas:null, properties:null)
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=compile start=1438327862476 end=1438327862747 duration=271>
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=Driver.execute>
15/07/31 15:31:02 INFO Configuration.deprecation: mapred.job.name is deprecated. Instead, use mapreduce.job.name
15/07/31 15:31:02 INFO ql.Driver: Starting command: use hive
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=TimeToSubmit start=1438327862476 end=1438327862796 duration=320>
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=runTasks>
15/07/31 15:31:02 INFO ql.Driver: <PERFLOG method=task.DDL.Stage-0>
15/07/31 15:31:02 INFO hive.metastore: Trying to connect to metastore with URI thrift://hadoop1:9083
15/07/31 15:31:02 INFO hive.metastore: waiting 1 seconds before next connection attempt.
15/07/31 15:31:03 INFO hive.metastore: Connected to metastore.
15/07/31 15:31:04 INFO ql.Driver: <PERFLOG method=task.DDL.Stage-0 start=1438327862796 end=1438327864556 duration=1760>
15/07/31 15:31:04 INFO ql.Driver: <PERFLOG method=runTasks start=1438327862796 end=1438327864556 duration=1760>
15/07/31 15:31:04 INFO ql.Driver: <PERFLOG method=Driver.execute start=1438327862747 end=1438327864556 duration=1809>
15/07/31 15:31:04 INFO ql.Driver: OK
15/07/31 15:31:04 INFO ql.Driver: <PERFLOG method=releaseLocks>
15/07/31 15:31:04 INFO ql.Driver: <PERFLOG method=releaseLocks start=1438327864564 end=1438327864574 duration=10>
15/07/31 15:31:04 INFO ql.Driver: <PERFLOG method=Driver.run start=1438327862476 end=1438327864574 duration=2098>
15/07/31 15:31:04 INFO ql.Driver: <PERFLOG method=releaseLocks>
15/07/31 15:31:04 INFO ql.Driver: <PERFLOG method=releaseLocks start=1438327864576 end=1438327864576 duration=0>
res0: org.apache.spark.sql.SchemaRDD =
SchemaRDD[0] at RDD at SchemaRDD.scala:103
== query Plan ==
<Native command: executed by Hive>
```

scala> hiveContext.sql("show tables").collect().foreach(println)

```
scala> hiveContext.sql("show tables").collect().foreach(println)
15/07/31 15:31:14 INFO parse.ParseDriver: Parsing command: show tables
15/07/31 15:31:14 INFO parse.ParseDriver: Parse Completed
15/07/31 15:31:14 INFO Configuration.deprecation: mapred.input.dir.recursive is deprecated. Instead, use mapreduce.input.fileinputformat
15/07/31 15:31:14 INFO ql.Driver: <PERFLOG method=Driver.run>
15/07/31 15:31:14 INFO ql.Driver: <PERFLOG method=TimeToSubmit>
15/07/31 15:31:14 INFO ql.Driver: <PERFLOG method=compile>
15/07/31 15:31:14 INFO ql.Driver: <PERFLOG method=parse>
15/07/31 15:31:14 INFO parse.ParseDriver: Parsing command: show tables
15/07/31 15:31:14 INFO parse.ParseDriver: Parse Completed
15/07/31 15:31:14 INFO ql.Driver: <PERFLOG method=parse start=1438327874957 end=1438327874958 duration=1>
15/07/31 15:31:14 INFO ql.Driver: <PERFLOG method=semanticAnalyze>

15/07/31 15:31:16 INFO network.ConnectionManager: Accepted connection from [hadoop1/192.168.10.111:34679]
15/07/31 15:31:16 INFO network.SendingConnection: Initiating connection to [hadoop1/192.168.10.111:44561]
15/07/31 15:31:16 INFO network.SendingConnection: Connected to [hadoop1/192.168.10.111:44561], 1 messages pending
15/07/31 15:31:16 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on hadoop1:44561 (Size: 1105.0 B, free: 530.3 MB)
15/07/31 15:31:17 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 1064 ms on hadoop1 (1/1)
15/07/31 15:31:17 INFO scheduler.DAGScheduler: Stage 0 (collect at sparkPlan.scala:85) finished in 1.065 s
15/07/31 15:31:17 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
15/07/31 15:31:17 INFO spark.sparkContext: Job finished: collect at sparkPlan.scala:85, took 1.74213206 s
[sogouq1]
[sogouq2]
[tbdate]
[tbstock]
[tbstockdetail]
[testthrift]
```

2.3.4 计算所有订单中每年的销售单数、销售总额

//所有订单中每年的销售单数、销售总额

//三个表连接后以 count(distinct a.ordernumber)计销售单数，sum(b.amount)计销售总额

```
scala>hiveContext.sql("select c.theyear,count(distinct a.ordernumber),sum(b.amount)
from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c
on a.dateid=c.dateid group by c.theyear order by c.theyear").collect().foreach(println)
```

```
hadoop1 | hadoop2 | hadoop3
scala>hiveContext.sql("select c.theyear,count(distinct a.ordernumber),sum(b.amount) from tbstock a join tbstockdetail b on a.ordernumber=b.ordernumber join tbdate c
on a.dateid=c.dateid group by c.theyear order by c.theyear").collect().foreach(println)
15/07/31 15:43:59 INFO parse.ParseDriver: Parsing command: select c.theyear,count(distinct a.ordernumber),sum(b.amount) from tbstock a join tbstockdetail b on a.ordernumber=b.ordernumber join tbdate c on a.dateid=c.dateid group by c.theyear order by c.theyear
15/07/31 15:43:59 INFO parse.ParseDriver: Parse completed
15/07/31 15:43:59 INFO storage.MemoryStore: ensureFreeSpace(402134) called with curMem=2757284, maxMem=278302556
15/07/31 15:43:59 INFO storage.MemoryStore: block broadcast_23 stored as values in memory (estimated size 392.7 KB, free 262.4 MB)
15/07/31 15:43:59 INFO storage.MemoryStore: ensureFreeSpace(27437) called with curMem=3159418, maxMem=278302556
15/07/31 15:43:59 INFO storage.MemoryStore: block broadcast_23_piece0 stored as bytes in memory (estimated size 26.8 KB, free 262.4 MB)
15/07/31 15:43:59 INFO storage.BlockManagerInfo: Added broadcast_23_piece0 in memory on hadoop1:59535 (size: 26.8 KB, free: 265.2 MB)
15/07/31 15:43:59 INFO storage.BlockManagerMaster: Updated info of block broadcast_23_piece0
15/07/31 15:43:59 INFO storage.MemoryStore: ensureFreeSpace(401950) called with curMem=3186855, maxMem=278302556
15/07/31 15:43:59 INFO storage.MemoryStore: block broadcast_24 stored as values in memory (estimated size 392.5 KB, free 262.0 MB)
15/07/31 15:43:59 INFO storage.MemoryStore: ensureFreeSpace(27417) called with curMem=3588805, maxMem=278302556
15/07/31 15:43:59 INFO storage.MemoryStore: block broadcast_24_piece0 stored as bytes in memory (estimated size 26.8 KB, free 262.0 MB)
15/07/31 15:43:59 INFO storage.BlockManagerInfo: Added broadcast_24_piece0 in memory on hadoop1:59535 (size: 26.8 KB, free: 265.1 MB)
15/07/31 15:43:59 INFO storage.BlockManagerMaster: Updated info of block broadcast_24_piece0
15/07/31 15:43:59 INFO storage.MemoryStore: ensureFreeSpace(402046) called with curMem=3616222, maxMem=278302556
15/07/31 15:43:59 INFO storage.MemoryStore: block broadcast_25 stored as values in memory (estimated size 392.6 KB, free 261.6 MB)
15/07/31 15:43:59 INFO storage.MemoryStore: ensureFreeSpace(27479) called with curMem=4018268, maxMem=278302556

15/07/31 15:44:31 INFO scheduler.TaskSetManager: Starting task 3.0 in stage 33.0 (TID 2438, hadoop3, PROCESS_LOCAL, 948 bytes)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Starting task 4.0 in stage 33.0 (TID 2439, hadoop1, PROCESS_LOCAL, 948 bytes)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 33.0 (TID 2435) in 136 ms on hadoop1 (1/8)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 33.0 (TID 2437) in 131 ms on hadoop3 (2/8)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Starting task 5.0 in stage 33.0 (TID 2440, hadoop2, PROCESS_LOCAL, 948 bytes)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 33.0 (TID 2436) in 138 ms on hadoop2 (3/8)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Starting task 6.0 in stage 33.0 (TID 2441, hadoop1, PROCESS_LOCAL, 948 bytes)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Finished task 4.0 in stage 33.0 (TID 2439) in 30 ms on hadoop1 (4/8)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Starting task 7.0 in stage 33.0 (TID 2442, hadoop3, PROCESS_LOCAL, 948 bytes)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Finished task 3.0 in stage 33.0 (TID 2438) in 34 ms on hadoop3 (5/8)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Finished task 5.0 in stage 33.0 (TID 2440) in 32 ms on hadoop2 (6/8)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Finished task 7.0 in stage 33.0 (TID 2442) in 19 ms on hadoop3 (7/8)
15/07/31 15:44:31 INFO scheduler.TaskSetManager: Finished task 6.0 in stage 33.0 (TID 2441) in 24 ms on hadoop1 (8/8)
15/07/31 15:44:31 INFO scheduler.DAGScheduler: Stage 33 (collect at sparkPlan.scala:85) finished in 0.189 s
15/07/31 15:44:31 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 33.0, whose tasks have all completed, from pool
15/07/31 15:44:31 INFO spark.sparkContext: Job finished: collect at sparkPlan.scala:85, took 2.7635168 s
[2004,1094,3265696]
[2005,3828,13247234]
[2006,3772,13670416]
[2007,4885,16711974]
[2008,4861,14670698]
[2009,2619,6322137]
[2010,94,210924]
```

结果如下：

```
[2004,1094,3265696]
[2005,3828,13247234]
[2006,3772,13670416]
[2007,4885,16711974]
[2008,4861,14670698]
[2009,2619,6322137]
[2010,94,210924]
```

通过监控页面，查看任务运行情况：

Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
5	mapPartitions at Exchange.scala:48	+details (kill) 2015/07/31 15:40:29	16 s	<div><div>106/200</div></div>		755.3 KB	1424.7 KB

Completed Stages (4)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
4	mapPartitions at Exchange.scala:48	+details 2015/07/31 15:40:07	21 s	<div><div>2/2</div></div>	11.4 MB		1713.1 KB
3	mapPartitions at Exchange.scala:48	+details 2015/07/31 15:40:07	19 s	<div><div>2/2</div></div>	588.0 KB		413.0 KB
2	mapPartitions at Exchange.scala:48	+details 2015/07/31 15:40:07	14 s	<div><div>2/2</div></div>	167.5 KB		86.6 KB

Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input	Shuffle Read	Shuffle Write	Shuffle Spill (Memory)	Shuffle Spill (Disk)
0	hadoop3:45404	25 s	72	0	72	0.0 B	688.9 KB	970.2 KB	0.0 B	0.0 B
1	hadoop1:44561	25 s	56	0	56	0.0 B	352.5 KB	753.0 KB	0.0 B	0.0 B
2	hadoop2:35280	26 s	72	0	72	0.0 B	384.1 KB	975.9 KB	0.0 B	0.0 B

Tasks

Index	ID	Attempt	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Accumulators	Shuffle Read	Write Time	Shuffle Write	Errors
0	7	0	SUCCESS	PROCESS_LOCAL	hadoop1	2015/07/31 15:40:29	2 s			6.5 KB	4 ms	13.4 KB	
1	8	0	SUCCESS	PROCESS_LOCAL	hadoop2	2015/07/31 15:40:29	0.8 s			5.6 KB	2 ms	14.2 KB	
2	9	0	SUCCESS	PROCESS_LOCAL	hadoop3	2015/07/31 15:40:29	0.9 s			9.0 KB	3 ms	13.1 KB	
3	10	0	SUCCESS	PROCESS_LOCAL	hadoop2	2015/07/31 15:40:30	0.6 s			4.6 KB	2 ms	11.9 KB	
4	11	0	SUCCESS	PROCESS_LOCAL	hadoop3	2015/07/31 15:40:30	0.7 s			8.5 KB	2 ms	12.3 KB	

2.3.5 计算所有订单每年最大金额订单的销售额

第一步 实现分析

所有订单每年最大金额订单的销售额:

1、先求出每份订单的销售额以其发生时间

select a.dateid,a.ordernumber,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber group by a.dateid,a.ordernumber

2、以第一步的查询作为子表，和表 tbDate 连接，求出每年最大金额订单的销售额

select c.theyear,max(d.sumofamount) from tbDate c join (select a.dateid,a.ordernumber,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber group by a.dateid,a.ordernumber) d on c.dateid=d.dateid group by c.theyear sort by c.theyear

第二步 实现 SQL 语句

scala>hiveContext.sql("select c.theyear,max(d.sumofamount) from tbDate c join (select a.dateid,a.ordernumber,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber group by a.dateid,a.ordernumber) d on c.dateid=d.dateid group by c.theyear sort by c.theyear").collect().foreach(println)

```

hadoop1 | hadoop2 | hadoop3
scala>hiveContext.sql("select c.theyear,max(d.sumofamount) from tbDate c join (select a.dateid,a.ordernumber,sum(b.amount) as sumofamount from tbStock a join
tbStockDetail b on a.ordernumber=b.ordernumber group by a.dateid,a.ordernumber) d on c.dateid=d.dateid group by c.theyear sort by c.theyear").collect().foreach(println)
15/07/31 15:48:27 INFO parse.ParseDriver: Parsing command: select c.theyear,max(d.sumofamount) from tbDate c join (select a.dateid,a.ordernumber,sum(b.amount) as sumofamount from tbStock a join
tbStockDetail b on a.ordernumber=b.ordernumber group by a.dateid,a.ordernumber) d on c.dateid=d.dateid group by c.theyear sort by c.theyear
15/07/31 15:48:27 INFO parse.ParseDriver: Parse Completed
15/07/31 15:48:27 INFO storage.MemoryStore: ensureFreeSpace(402214) called with curMem=5518313, maxMem=278302556
15/07/31 15:48:27 INFO storage.MemoryStore: Block broadcast_43 stored as values in memory (estimated size 392.8 KB, free 259.8 MB)
15/07/31 15:48:27 INFO storage.MemoryStore: ensureFreeSpace(27498) called with curMem=5920527, maxMem=278302556
15/07/31 15:48:27 INFO storage.MemoryStore: Block broadcast_43_piece0 stored as bytes in memory (estimated size 26.9 KB, free 259.7 MB)
15/07/31 15:48:27 INFO storage.BlockManagerInfo: Added broadcast_43_piece0 in memory on hadoop1:59535 (size: 26.9 KB, free: 265.0 MB)
15/07/31 15:48:27 INFO storage.BlockManagerMaster: Updated info of block broadcast_43_piece0
15/07/31 15:48:27 INFO storage.MemoryStore: ensureFreeSpace(402430) called with curMem=5948025, maxMem=278302556
15/07/31 15:48:27 INFO storage.MemoryStore: Block broadcast_44 stored as values in memory (estimated size 393.0 KB, free 259.4 MB)
15/07/31 15:48:27 INFO storage.MemoryStore: ensureFreeSpace(27532) called with curMem=6350455, maxMem=278302556
15/07/31 15:48:27 INFO storage.MemoryStore: Block broadcast_44_piece0 stored as bytes in memory (estimated size 26.9 KB, free 259.3 MB)
15/07/31 15:48:27 INFO storage.BlockManagerInfo: Added broadcast_44_piece0 in memory on hadoop1:59535 (size: 26.9 KB, free: 264.9 MB)
15/07/31 15:48:27 INFO storage.BlockManagerMaster: Updated info of block broadcast_44_piece0
15/07/31 15:48:27 INFO storage.MemoryStore: ensureFreeSpace(402254) called with curMem=6377987, maxMem=278302556
15/07/31 15:48:27 INFO storage.MemoryStore: Block broadcast_45 stored as values in memory (estimated size 392.8 KB, free 258.9 MB)
15/07/31 15:48:27 INFO storage.MemoryStore: ensureFreeSpace(27500) called with curMem=6780241, maxMem=278302556

15/07/31 15:47:27 INFO scheduler.TaskSetManager: Starting task 197.0 in stage 40.0 (TID 3046, hadoop3, PROCESS_LOCAL, 948 bytes)
15/07/31 15:47:27 INFO scheduler.TaskSetManager: Finished task 191.0 in stage 40.0 (TID 3040) in 206 ms on hadoop3 (195/200)
15/07/31 15:47:27 INFO scheduler.TaskSetManager: Starting task 198.0 in stage 40.0 (TID 3047, hadoop2, PROCESS_LOCAL, 948 bytes)
15/07/31 15:47:27 INFO scheduler.TaskSetManager: Finished task 196.0 in stage 40.0 (TID 3045) in 21 ms on hadoop2 (196/200)
15/07/31 15:47:27 INFO scheduler.TaskSetManager: Starting task 199.0 in stage 40.0 (TID 3048, hadoop3, PROCESS_LOCAL, 948 bytes)
15/07/31 15:47:27 INFO scheduler.TaskSetManager: Finished task 197.0 in stage 40.0 (TID 3046) in 20 ms on hadoop3 (197/200)
15/07/31 15:47:27 INFO scheduler.TaskSetManager: Finished task 198.0 in stage 40.0 (TID 3047) in 17 ms on hadoop2 (198/200)
15/07/31 15:47:27 INFO scheduler.TaskSetManager: Finished task 199.0 in stage 40.0 (TID 3048) in 16 ms on hadoop3 (199/200)
15/07/31 15:47:27 INFO scheduler.TaskSetManager: Finished task 192.0 in stage 40.0 (TID 3041) in 204 ms on hadoop1 (200/200)
15/07/31 15:47:27 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 40.0, whose tasks have all completed, from pool
15/07/31 15:47:27 INFO scheduler.DAGScheduler: Stage 40 (collect at sparkPlan.scala:85) finished in 2.074 s
15/07/31 15:47:27 INFO spark.SparkContext: Job finished: collect at sparkPlan.scala:85, took 30.269685192 s
[2010,13063]
[2004,23612]
[2005,38180]
[2006,36124]
[2007,159126]
[2008,53828]
[2009,25810]

```

结果如下：

[2010,13063]

[2004,23612]

[2005,38180]

[2006,36124]

[2007,159126]

[2008,55828]

[2009,25810]

第三步 监控任务运行情况

Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
44	mapPartitions at Exchange.scala:48	+details (kill) 2015/07/31 15:46:59	13 s	192/200		1374.6 KB	1180.2 KB

Completed Stages (28)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
43	mapPartitions at Exchange.scala:48	+details 2015/07/31 15:46:57	2 s	2/2	11.4 MB		1713.1 KB
42	mapPartitions at Exchange.scala:48	+details 2015/07/31 15:46:57	1.0 s	2/2	588.0 KB		410.6 KB
41	mapPartitions at Exchange.scala:48	+details 2015/07/31 15:46:57	0.8 s	2/2	167.5 KB		87.3 KB

Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input	Shuffle Read	Shuffle Write	Shuffle Spill (Memory)	Shuffle Spill (Disk)
0	hadoop3:45404	2 s	73	0	73	0.0 B	12.3 KB	0.0 B	0.0 B	0.0 B
1	hadoop1:44561	2 s	55	0	55	0.0 B	15.6 KB	0.0 B	0.0 B	0.0 B
2	hadoop2:35280	2 s	72	0	72	0.0 B	10.8 KB	0.0 B	0.0 B	0.0 B

Tasks

Index	ID	Attempt	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Accumulators	Shuffle Read	Errors
0	2849	0	SUCCESS	PROCESS_LOCAL	hadoop3	2015/07/31 15:47:25	23 ms			0.0 B	
2	2851	0	SUCCESS	PROCESS_LOCAL	hadoop1	2015/07/31 15:47:25	30 ms			0.0 B	
1	2850	0	SUCCESS	PROCESS_LOCAL	hadoop2	2015/07/31 15:47:25	31 ms			0.0 B	
3	2852	0	SUCCESS	PROCESS_LOCAL	hadoop2	2015/07/31 15:47:25	9 ms			0.0 B	
4	2853	0	SUCCESS	PROCESS_LOCAL	hadoop1	2015/07/31 15:47:25	22 ms			0.0 B	

2.3.6 计算所有订单中每年最畅销货品

第一步 实现分析

所有订单中每年最畅销货品：

1、求出每年每个货品的销售金额

```
scala>select c.theyear,b.itemid,sum(b.amount) as sumofamount from tbStock a join  
tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid  
group by c.theyear,b.itemid
```

2、求出每年单品销售的最大金额

```
scala>select d.theyear,max(d.sumofamount) as maxofamount from (select  
c.theyear,b.itemid,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b
```

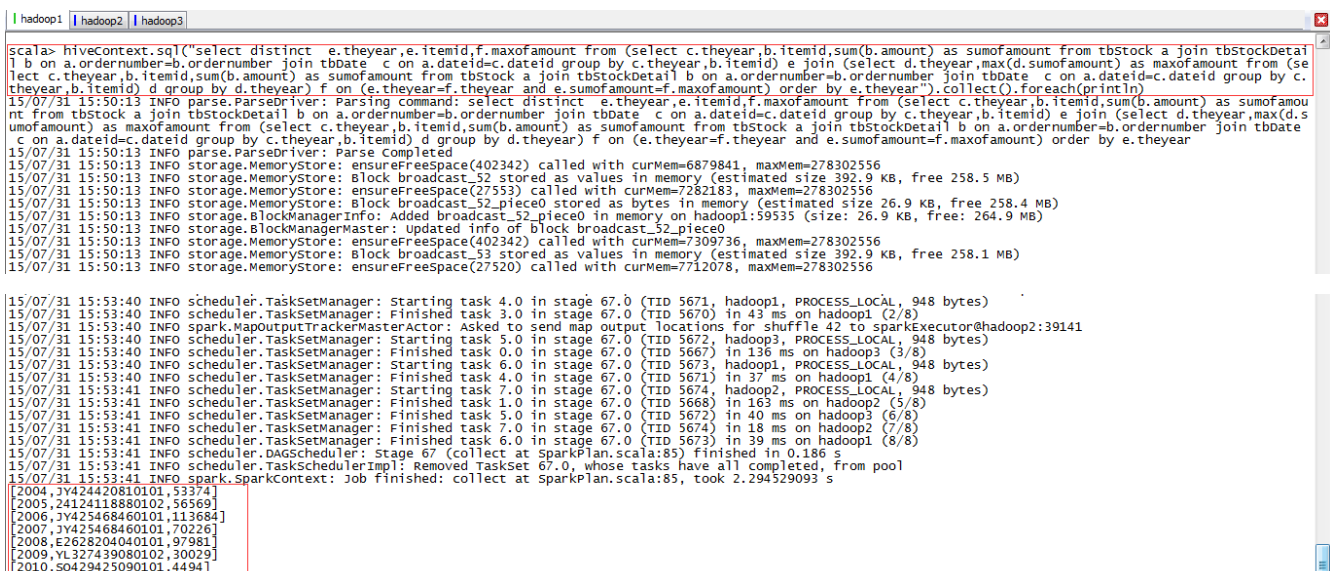

on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,b.itemid) d group by d.theyear

3、求出每年与销售额最大相符的货品就是最畅销货品

```
scala>select distinct e.theyear,e.itemid,f.maxofamount from (select c.theyear,b.itemid,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,b.itemid) e join (select d.theyear,max(d.sumofamount) as maxofamount from (select c.theyear,b.itemid,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,b.itemid) d group by d.theyear) f on (e.theyear=f.theyear and e.sumofamount=f.maxofamount) order by e.theyear
```

第二步 实现 SQL 语句

```
scala>hiveContext.sql("select distinct e.theyear,e.itemid,f.maxofamount from (select c.theyear,b.itemid,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,b.itemid) e join (select d.theyear,max(d.sumofamount) as maxofamount from (select c.theyear,b.itemid,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,b.itemid) d group by d.theyear) f on (e.theyear=f.theyear and e.sumofamount=f.maxofamount) order by e.theyear").collect().foreach(println)
```



```
scala>hiveContext.sql("select distinct e.theyear,e.itemid,f.maxofamount from (select c.theyear,b.itemid,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,b.itemid) e join (select d.theyear,max(d.sumofamount) as maxofamount from (select c.theyear,b.itemid,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,b.itemid) d group by d.theyear) f on (e.theyear=f.theyear and e.sumofamount=f.maxofamount) order by e.theyear").collect().foreach(println)
15/07/31 15:50:13 INFO parse.ParseDriver: Parsing command: select distinct e.theyear,e.itemid,f.maxofamount from (select c.theyear,b.itemid,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,b.itemid) e join (select d.theyear,max(d.sumofamount) as maxofamount from (select c.theyear,b.itemid,sum(b.amount) as sumofamount from tbStock a join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on a.dateid=c.dateid group by c.theyear,b.itemid) d group by d.theyear) f on (e.theyear=f.theyear and e.sumofamount=f.maxofamount) order by e.theyear
15/07/31 15:50:13 INFO parse.ParseDriver: Parse completed
15/07/31 15:50:13 INFO storage.MemoryStore: ensureFreeSpace(402342) called with curMem=6879841, maxMem=278302556
15/07/31 15:50:13 INFO storage.MemoryStore: block broadcast_52 stored as values in memory (estimated size 392.9 KB, free 258.5 MB)
15/07/31 15:50:13 INFO storage.MemoryStore: ensureFreeSpace(27553) called with curMem=7282183, maxMem=278302556
15/07/31 15:50:13 INFO storage.MemoryStore: block broadcast_52_piece0 stored as bytes in memory (estimated size 26.9 KB, free 258.4 MB)
15/07/31 15:50:13 INFO storage.BlockManagerInfo: Added broadcast_52_piece0 in memory on hadoop1:59535 (size: 26.9 KB, free: 264.9 MB)
15/07/31 15:50:13 INFO storage.BlockManagerMaster: updated info of block broadcast_52_piece0
15/07/31 15:50:13 INFO storage.MemoryStore: ensureFreeSpace(402342) called with curMem=7309736, maxMem=278302556
15/07/31 15:50:13 INFO storage.MemoryStore: block broadcast_53 stored as values in memory (estimated size 392.9 KB, free 258.1 MB)
15/07/31 15:50:13 INFO storage.MemoryStore: ensureFreeSpace(27520) called with curMem=7712078, maxMem=278302556
15/07/31 15:53:40 INFO scheduler.TaskSetManager: Starting task 4.0 in stage 67.0 (TID 5671, hadoop1, PROCESS_LOCAL, 948 bytes)
15/07/31 15:53:40 INFO scheduler.TaskSetManager: Finished task 3.0 in stage 67.0 (TID 5670) in 43 ms on hadoop1 (2/8)
15/07/31 15:53:40 INFO spark.MapOutputTrackerMasterActor: Asked to send map output locations for shuffle 42 to sparkExecutor@hadoop2:39141
15/07/31 15:53:40 INFO scheduler.TaskSetManager: Starting task 5.0 in stage 67.0 (TID 5672, hadoop3, PROCESS_LOCAL, 948 bytes)
15/07/31 15:53:40 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 67.0 (TID 5667) in 136 ms on hadoop3 (3/8)
15/07/31 15:53:40 INFO scheduler.TaskSetManager: Starting task 6.0 in stage 67.0 (TID 5673, hadoop1, PROCESS_LOCAL, 948 bytes)
15/07/31 15:53:40 INFO scheduler.TaskSetManager: Finished task 4.0 in stage 67.0 (TID 5671) in 37 ms on hadoop1 (4/8)
15/07/31 15:53:41 INFO scheduler.TaskSetManager: Starting task 7.0 in stage 67.0 (TID 5674, hadoop2, PROCESS_LOCAL, 948 bytes)
15/07/31 15:53:41 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 67.0 (TID 5668) in 163 ms on hadoop2 (5/8)
15/07/31 15:53:41 INFO scheduler.TaskSetManager: Finished task 5.0 in stage 67.0 (TID 5672) in 40 ms on hadoop3 (6/8)
15/07/31 15:53:41 INFO scheduler.TaskSetManager: Finished task 7.0 in stage 67.0 (TID 5674) in 18 ms on hadoop2 (7/8)
15/07/31 15:53:41 INFO scheduler.TaskSetManager: Finished task 6.0 in stage 67.0 (TID 5673) in 39 ms on hadoop1 (8/8)
15/07/31 15:53:41 INFO scheduler.DAGScheduler: stage 67 (collect at SparkPlan.scala:85) finished in 0.186 s
15/07/31 15:53:41 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 67.0, whose tasks have all completed, from pool
15/07/31 15:53:41 INFO spark.SparkContext: Job finished: collect at SparkPlan.scala:85, took 2.294529093 s
[2004,JY424420810101,53374]
[2005,24124118880102,56569]
[2006,JY425468460101,113684]
[2007,JY425468460101,70226]
[2008,E2628204040101,97981]
[2009,YL327439080102,30029]
[2010,SQ4229425090101,4494]
```

结果如下：

[2004,JY424420810101,53374]

[2005,24124118880102,56569]

[2006,JY425468460101,113684]
[2007,JY425468460101,70226]
[2008,E2628204040101,97981]
[2009,YL327439080102,30029]
[2010,SQ429425090101,4494]

第三步 监控任务运行情况

Active Stages (2)

Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
57	mapPartitions at Exchange.scala:48	+details (kill)	2015/07/31 15:50:30	22 s	157/200		3.0 MB	5.7 MB
63	mapPartitions at Exchange.scala:48	+details (kill)	2015/07/31 15:50:19	33 s	5/200		79.0 KB	127.1 KB

Completed Stages (44)

Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
56	mapPartitions at Exchange.scala:48	+details	2015/07/31 15:50:17	13 s	200/200		3.6 MB	5.7 MB
62	mapPartitions at Exchange.scala:48	+details	2015/07/31 15:50:14	5 s	2/2	11.4 MB		4.2 MB
61	mapPartitions at Exchange.scala:48	+details	2015/07/31 15:50:14	3 s	2/2	588.0 KB		410.6 KB
60	mapPartitions at Exchange.scala:48	+details	2015/07/31 15:50:14	1 s	2/2	167.5 KB		86.6 KB

Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input	Shuffle Read	Shuffle Write	Shuffle Spill (Memory)	Shuffle Spill (Disk)
0	hadoop3:45404	20 s	57	0	57	0.0 B	1367.1 KB	24.2 KB	0.0 B	0.0 B
1	hadoop1:44561	19 s	49	0	49	0.0 B	1192.9 KB	21.0 KB	0.0 B	0.0 B
2	hadoop2:35280	19 s	52	0	52	0.0 B	1350.6 KB	22.4 KB	0.0 B	0.0 B

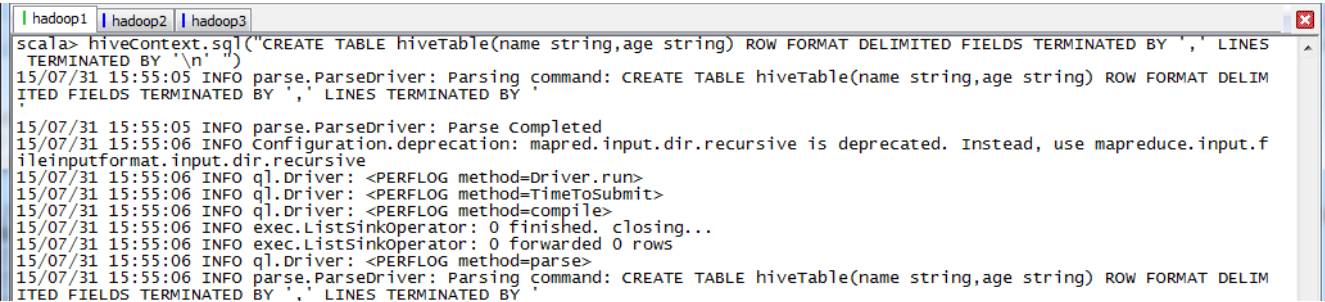
Tasks

Index	ID	Attempt	Status	Locality Level	Executor	Launch Time	Duration	GC Time	Accumulators	Shuffle Read	Write Time	Shuffle Write	Errors
0	4078	0	SUCCESS	PROCESS_LOCAL	hadoop3	2015/07/31 15:50:59	0.3 s			24.5 KB		445.0 B	
1	4079	0	SUCCESS	PROCESS_LOCAL	hadoop2	2015/07/31 15:50:59	0.3 s			21.3 KB		444.0 B	
2	4080	0	SUCCESS	PROCESS_LOCAL	hadoop1	2015/07/31 15:50:59	0.3 s			20.8 KB	1 ms	445.0 B	
3	4081	0	SUCCESS	PROCESS_LOCAL	hadoop3	2015/07/31 15:50:59	0.2 s			20.4 KB		445.0 B	

2.3.7 hiveContext 中混合使用演示

第一步 创建 hiveTable 从本地文件系统加载数据

//创建一个 hiveTable 并将数据加载 , 注意 people.txt 第二列有空格 , 所以 age 取 string 类型
scala>hiveContext.sql("CREATE TABLE hiveTable(name string,age string) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' ")



scala>hiveContext.sql("LOAD DATA LOCAL INPATH
'/home/hadoop/upload/class6/people.txt' INTO TABLE hiveTable")


```
hadoop1 | hadoop2 | hadoop3
scala> hiveContext.sql("LOAD DATA LOCAL INPATH '/home/hadoop/upload/class6/people.txt' INTO TABLE hiveTable")
15/07/31 15:56:25 INFO parse.ParseDriver: Parsing command: LOAD DATA LOCAL INPATH '/home/hadoop/upload/class6/people.txt' IN
TO TABLE hiveTable
15/07/31 15:56:25 INFO parse.ParseDriver: Parse Completed
15/07/31 15:56:25 INFO Configuration.deprecation: mapred.input.dir.recursive is deprecated. Instead, use mapreduce.input.fil
einputformat.input.dir.recursive
15/07/31 15:56:25 INFO ql.Driver: <PERFLOG method=Driver.run>
15/07/31 15:56:25 INFO ql.Driver: <PERFLOG method=TimeToSubmit>
15/07/31 15:56:25 INFO ql.Driver: <PERFLOG method=compile>
15/07/31 15:56:25 INFO ql.Driver: <PERFLOG method=parse>
15/07/31 15:56:25 INFO parse.ParseDriver: Parsing command: LOAD DATA LOCAL INPATH '/home/hadoop/upload/class6/people.txt' IN
TO TABLE hiveTable
15/07/31 15:56:25 INFO parse.ParseDriver: Parse Completed
15/07/31 15:56:25 INFO ql.Driver: <PERFLOG method=parse start=1438329385626 end=1438329385626 duration=0>
15/07/31 15:56:25 INFO ql.Driver: <PERFLOG method=semanticAnalyze>
15/07/31 15:56:26 INFO ql.Driver: Semantic Analysis completed
```

第二步 创建 parquet 表，从 HDFS 加载数据

//创建一个源自 parquet 文件的表 parquetTable2，然后和 hiveTable 混合使用

```
scala> hiveContext.parquetFile("hdfs://hadoop1:9000/class6/people.parquet").register
TempTable("parquetTable2")
```

```
hadoop1 | hadoop2 | hadoop3
scala> hiveContext.parquetFile("hdfs://hadoop1:9000/class6/people.parquet").registerTempTable("parquetTable2")
SLF4J: Failed to load class 'org.slf4j.impl.StaticLoggerBinder'.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
```

第三步 两个表混合使用

```
scala> hiveContext.sql("select a.name,a.age,b.age from hiveTable a join parquetTable2
b on a.name=b.name").collect().foreach(println)
```

```
hadoop1 | hadoop2 | hadoop3
scala> hiveContext.sql("select a.name,a.age,b.age from hiveTable a join parquetTable2 b on a.name=b.name").collect().foreach(println)
15/07/31 15:58:26 INFO parse.ParseDriver: Parsing command: select a.name,a.age,b.age from hiveTable a join parquetTable2 b on a.name=b.name
15/07/31 15:58:26 INFO parse.ParseDriver: Parse Completed
15/07/31 15:58:26 INFO storage.MemoryStore: ensureFreeSpace(405193) called with curMem=9832008, maxMem=278302556
15/07/31 15:58:26 INFO storage.MemoryStore: Block broadcast_75 stored as values in memory (estimated size 395.7 KB, free 255.6 MB)
15/07/31 15:58:26 INFO storage.MemoryStore: ensureFreeSpace(27655) called with curMem=10237201, maxMem=278302556
15/07/31 15:58:26 INFO storage.MemoryStore: Block broadcast_75_piece0 stored as bytes in memory (estimated size 27.0 KB, free 255.6 MB)
15/07/31 15:58:26 INFO storage.BlockManagerInfo: Added broadcast_75_piece0 in memory on hadoop1:59535 (size: 27.0 KB, free: 264.6 MB)
15/07/31 15:58:26 INFO storage.BlockManagerMaster: Updated info of block broadcast_75_piece0
15/07/31 15:58:26 INFO mapred.FileInputFormat: Total input paths to process : 1
15/07/31 15:58:26 INFO spark.SparkContext: Starting job: collect at SparkPlan.scala:85
15/07/31 15:58:26 INFO scheduler.DAGScheduler: Got job 11 (collect at SparkPlan.scala:85) with 2 output partitions (allowLocal=false)
15/07/31 15:58:26 INFO scheduler.DAGScheduler: Final stage: Stage 83(collect at SparkPlan.scala:85)
15/07/31 15:58:26 INFO scheduler.DAGScheduler: Parents of final stage: List()
15/07/31 15:58:26 INFO scheduler.DAGScheduler: Missing parents: List()
15/07/31 15:58:26 INFO scheduler.DAGScheduler: Submitting Stage 83 (MappedRDD[291] at map at SparkPlan.scala:85), which has no missing parents
15/07/31 15:58:26 INFO storage.MemoryStore: ensureFreeSpace(5448) called with curMem=10264856, maxMem=278302556
15/07/31 15:58:26 INFO storage.MemoryStore: Block broadcast_76 stored as values in memory (estimated size 5.3 KB, free 255.6 MB)
15/07/31 15:58:26 INFO storage.MemoryStore: ensureFreeSpace(3181) called with curMem=10270304, maxMem=278302556
15/07/31 15:58:26 INFO storage.MemoryStore: Block broadcast_76_piece0 stored as bytes in memory (estimated size 3.1 KB, free 255.6 MB)
15/07/31 15:58:26 INFO storage.BlockManagerInfo: Added broadcast_76_piece0 in memory on hadoop1:59535 (size: 3.1 KB, free: 264.6 MB)
15/07/31 15:58:26 INFO storage.BlockManagerMaster: Updated info of block broadcast_76_piece0
15/07/31 15:58:26 INFO scheduler.DAGScheduler: Submitting 2 missing tasks from Stage 83 (MappedRDD[291] at map at SparkPlan.scala:85)
15/07/31 15:58:26 INFO scheduler.TaskSchedulerImpl: Adding task set 83.0 with 2 tasks
15/07/31 15:58:28 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 84.0 (TID 5678) in 534 ms on hadoop2 (1/2)
15/07/31 15:58:28 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 84.0 (TID 5677) in 1285 ms on hadoop1 (2/2)
15/07/31 15:58:29 INFO scheduler.DAGScheduler: Stage 84 (collect at SparkPlan.scala:85) finished in 1.283 s
15/07/31 15:58:29 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 84.0, whose tasks have all completed, from pool
15/07/31 15:58:29 INFO spark.SparkContext: Job finished: collect at SparkPlan.scala:85, took 1.336995424 s
[Michael, 29,29]
[Andy, 30,30]
[Justin, 19,19]
[Justin, 19,19]
```

2.4 Cache 使用

sparkSQL 的 cache 可以使用两种方法来实现：

- CacheTable()方法
- CACHE TABLE 命令

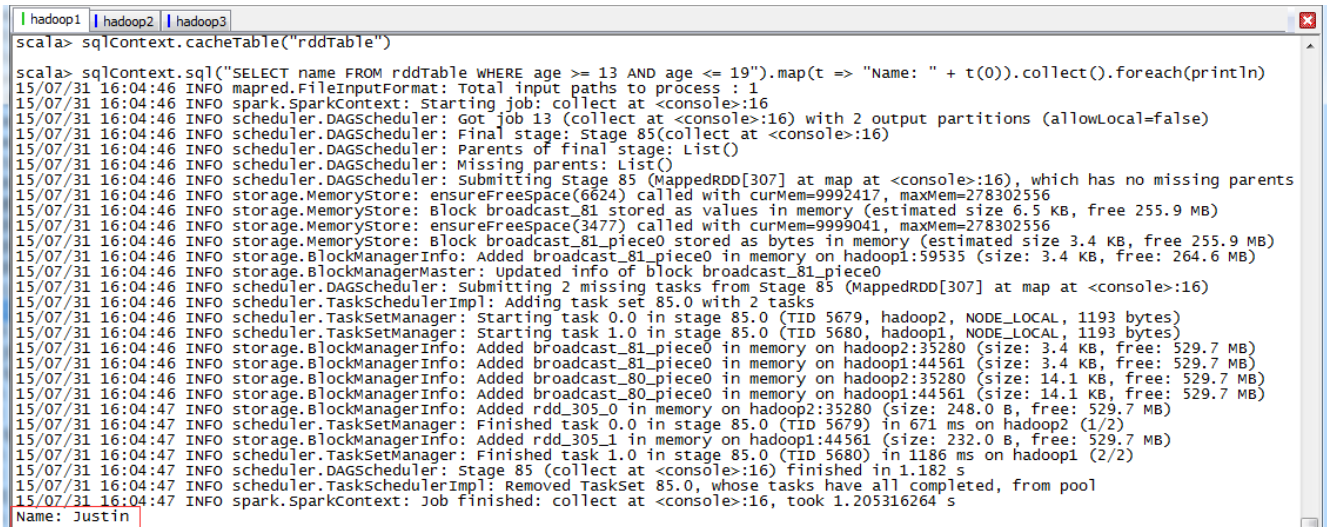
千万不要先使用 cache SchemaRDD，然后 registerAsTable；使用 RDD 的 cache()将使用原生态的 cache，而不是针对 SQL 优化后的内存列存储。

第一步 对 rddTable 表进行缓存

//cache 使用

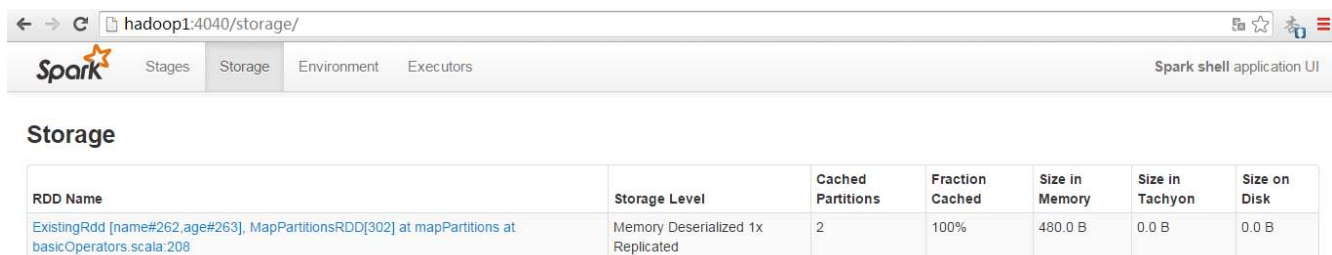
```
scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc)
scala> import sqlContext.createSchemaRDD
scala> case class Person(name:String,age:Int)
scala> val
rddpeople = sc.textFile("hdfs://hadoop1:9000/class6/people.txt").map(_._split(",")).map(p
=> Person(p(0),p(1).trim.toInt))
scala> rddpeople.registerTempTable("rddTable")

scala> sqlContext.cacheTable("rddTable")
scala> sqlContext.sql("SELECT name FROM rddTable WHERE age >= 13 AND age <=
19").map(t => "Name: " + t(0)).collect().foreach(println)
```



```
scala> sqlContext.cacheTable("rddTable")
scala> sqlContext.sql("SELECT name FROM rddTable WHERE age >= 13 AND age <= 19").map(t => "Name: " + t(0)).collect().foreach(println)
15/07/31 16:04:46 INFO mapred.FileInputFormat: Total input paths to process : 1
15/07/31 16:04:46 INFO spark.SparkContext: Starting job: collect at <console>:16
15/07/31 16:04:46 INFO scheduler.DAGScheduler: Got job 13 (collect at <console>:16) with 2 output partitions (allowLocal=false)
15/07/31 16:04:46 INFO scheduler.DAGScheduler: Final stage: Stage 85(collect at <console>:16)
15/07/31 16:04:46 INFO scheduler.DAGScheduler: Parents of final stage: List()
15/07/31 16:04:46 INFO scheduler.DAGScheduler: Missing parents: List()
15/07/31 16:04:46 INFO scheduler.DAGScheduler: Submitting Stage 85 (MappedRDD[307] at map at <console>:16), which has no missing parents
15/07/31 16:04:46 INFO storage.MemoryStore: ensureFreeSpace(6624) called with curMem=9992417, maxMem=278302556
15/07/31 16:04:46 INFO storage.MemoryStore: Block broadcast_81 stored as values in memory (estimated size 6.5 KB, free 255.9 MB)
15/07/31 16:04:46 INFO storage.MemoryStore: ensureFreeSpace(3477) called with curMem=9999041, maxMem=278302556
15/07/31 16:04:46 INFO storage.MemoryStore: Block broadcast_81_piece0 stored as bytes in memory (estimated size 3.4 KB, free 255.9 MB)
15/07/31 16:04:46 INFO storage.BlockManagerInfo: Added broadcast_81_piece0 in memory on hadoop1:59535 (size: 3.4 KB, free: 264.6 MB)
15/07/31 16:04:46 INFO storage.BlockManagerMaster: Updated info of block broadcast_81_piece0
15/07/31 16:04:46 INFO scheduler.DAGScheduler: Submitting 2 missing tasks from stage 85 (MappedRDD[307] at map at <console>:16)
15/07/31 16:04:46 INFO scheduler.TaskSchedulerImpl: Adding task set 85.0 with 2 tasks
15/07/31 16:04:46 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 85.0 (TID 5679, hadoop2, NODE_LOCAL, 1193 bytes)
15/07/31 16:04:46 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 85.0 (TID 5680, hadoop1, NODE_LOCAL, 1193 bytes)
15/07/31 16:04:46 INFO storage.BlockManagerInfo: Added broadcast_81_piece0 in memory on hadoop2:35280 (size: 3.4 KB, free: 529.7 MB)
15/07/31 16:04:46 INFO storage.BlockManagerInfo: Added broadcast_81_piece0 in memory on hadoop1:44561 (size: 3.4 KB, free: 529.7 MB)
15/07/31 16:04:46 INFO storage.BlockManagerInfo: Added broadcast_80_piece0 in memory on hadoop2:35280 (size: 14.1 KB, free: 529.7 MB)
15/07/31 16:04:46 INFO storage.BlockManagerInfo: Added broadcast_80_piece0 in memory on hadoop1:44561 (size: 14.1 KB, free: 529.7 MB)
15/07/31 16:04:47 INFO storage.BlockManagerInfo: Added rdd_305_0 in memory on hadoop2:35280 (size: 248.0 B, free: 529.7 MB)
15/07/31 16:04:47 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 85.0 (TID 5679) in 671 ms on hadoop2 (1/2)
15/07/31 16:04:47 INFO storage.BlockManagerInfo: Added rdd_305_1 in memory on hadoop1:44561 (size: 232.0 B, free: 529.7 MB)
15/07/31 16:04:47 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 85.0 (TID 5680) in 1186 ms on hadoop1 (2/2)
15/07/31 16:04:47 INFO scheduler.DAGScheduler: Stage 85 (collect at <console>:16) finished in 1.182 s
15/07/31 16:04:47 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 85.0, whose tasks have all completed, from pool
15/07/31 16:04:47 INFO spark.SparkContext: Job finished: collect at <console>:16, took 1.205316264 s
Name: Justin
```

在监控界面上看到该表数据已经缓存



RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size in Tachyon	Size on Disk
ExistingRdd [name#262,age#263], MapPartitionsRDD[302] at mapPartitions at basicOperators.scala:208	Memory Deserialized 1x Replicated	2	100%	480.0 B	0.0 B	0.0 B

第二步 对 parquetTable 表进行缓存

```
scala> val parquetpeople =
sqlContext.parquetFile("hdfs://hadoop1:9000/class6/people.parquet")
scala> parquetpeople.registerTempTable("parquetTable")
```

```
scala> sqlContext.sql("CACHE TABLE parquetTable")
```

```
scala> sqlContext.sql("SELECT name FROM parquetTable WHERE age >= 13 AND age <= 19").map(t => "Name: " + t(0)).collect().foreach(println)
```

```
hadoop1 | hadoop2 | hadoop3
scala> sqlContext.sql("CACHE TABLE parquetTable")
15/07/31 16:07:02 INFO storage.MemoryStore: ensureFreeSpace(186770) called with curMem=10002518, maxMem=278302556
15/07/31 16:07:02 INFO storage.MemoryStore: Block broadcast_82 stored as values in memory (estimated size 182.4 KB, free 255.7 MB)
15/07/31 16:07:02 INFO storage.MemoryStore: ensureFreeSpace(15013) called with curMem=10189288, maxMem=278302556
15/07/31 16:07:02 INFO storage.MemoryStore: Block broadcast_82_piece0 stored as bytes in memory (estimated size 14.7 KB, free 255.7 MB)
15/07/31 16:07:02 INFO storage.BlockManagerInfo: Added broadcast_82_piece0 in memory on hadoop1:59535 (size: 14.7 KB, free: 264.6 MB)
res21: org.apache.spark.sql.SchemaRDD =
SchemaRDD[313] at RDD at SchemaRDD.scala:103
== Query Plan ==
== Physical Plan ==
CacheCommand parquetTable, true

hadoop1 | hadoop2 | hadoop3
scala> sqlContext.sql("SELECT name FROM parquetTable WHERE age >= 13 AND age <= 19").map(t => "Name: " + t(0)).collect().foreach(println)
15/07/31 16:07:13 INFO storage.BlockManager: Removing broadcast_81
15/07/31 16:07:13 INFO storage.BlockManager: Removing block broadcast_81
15/07/31 16:07:13 INFO storage.MemoryStore: Block broadcast_81 of size 6624 dropped from memory (free 268104879)
15/07/31 16:07:13 INFO storage.BlockManager: Removing block broadcast_81_piece0
15/07/31 16:07:13 INFO storage.MemoryStore: Block broadcast_81_piece0 of size 3477 dropped from memory (free 268108356)
15/07/31 16:07:13 INFO storage.BlockManagerInfo: Removed broadcast_81_piece0 on hadoop1:59535 in memory (size: 3.4 KB, free: 264.6 MB)
15/07/31 16:07:13 INFO storage.BlockManagerInfo: Removed broadcast_81_piece0 on hadoop2:35280 in memory (size: 3.4 KB, free: 529.7 MB)
15/07/31 16:07:13 INFO storage.BlockManagerMaster: Updated info of block broadcast_81_piece0
15/07/31 16:07:13 INFO storage.BlockManagerInfo: Removed broadcast_81_piece0 on hadoop1:44561 in memory (size: 3.4 KB, free: 529.7 MB)
15/07/31 16:07:13 INFO spark.ContextCleaner: Cleaned broadcast 81
15/07/31 16:07:13 INFO input.FileInputFormat: Total input paths to process : 2
15/07/31 16:07:13 INFO hadoop.ParquetInputFormat: Total input paths to process : 2

15/07/31 16:07:13 INFO storage.BlockManagerInfo: Added rdd_318_1 in memory on hadoop1:44561 (size: 232.0 B, free: 529.6 MB)
15/07/31 16:07:13 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 86.0 (TID 5682) in 193 ms on hadoop1 (1/2)
15/07/31 16:07:14 INFO storage.BlockManagerInfo: Added rdd_318_0 in memory on hadoop2:35280 (size: 248.0 B, free: 529.7 MB)
15/07/31 16:07:14 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 86.0 (TID 5681) in 236 ms on hadoop2 (2/2)
15/07/31 16:07:14 INFO scheduler.DAGScheduler: Stage 86 (collect at <console>:16) finished in 0.237 s
15/07/31 16:07:14 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 86.0, whose tasks have all completed, from pool
15/07/31 16:07:14 INFO spark.SparkContext: Job finished: collect at <console>:16, took 0.265441671 s
Name: Justin
```

在监控界面上看到该表数据已经缓存

hadoop1:4040/storage/						
Spark Stages Storage Environment Executors Spark shell application UI						
Storage						
RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size in Tachyon	Size on Disk
ParquetTableScan [name#264,age#265], (ParquetRelation hdfs://hadoop1:9000/class6/people.parquet, Some(Configuration: core-default.xml, core-site.xml, mapred-default.xml, mapred-site.xml, yarn-default.xml, yarn-site.xml, hdfs-default.xml, hdfs-site.xml), org.apache.spark.sql.SQLContext@23f8cc50, [], [])	Memory Deserialized 1x Replicated	2	100%	480.0 B	0.0 B	0.0 B
ExistingRDD [name#262,age#263], MapPartitionsRDD[302] at mapPartitions at basicOperators.scala:208	Memory Deserialized 1x Replicated	2	100%	480.0 B	0.0 B	0.0 B

第三步 解除缓存

//uncache 使用

```
scala> sqlContext.uncacheTable("rddTable")
```

```
scala> sqlContext.sql("UNCACHE TABLE parquetTable")
```

```
hadoop1 | hadoop2 | hadoop3
scala> sqlContext.uncacheTable("rddTable")
15/07/31 16:09:11 INFO rdd.MapPartitionsRDD: Removing RDD 305 from persistence list
15/07/31 16:09:11 INFO storage.BlockManager: Removing RDD 305

scala> sqlContext.sql("UNCACHE TABLE parquetTable")
15/07/31 16:09:19 INFO rdd.MapPartitionsRDD: Removing RDD 318 from persistence list
15/07/31 16:09:19 INFO storage.BlockManager: Removing RDD 318
res24: org.apache.spark.sql.SchemaRDD =
SchemaRDD[326] at RDD at SchemaRDD.scala:103
== Query Plan ==
== Physical Plan ==
CacheCommand parquetTable, false
```


hadoop1.4040/storage/						
<div> <div>Spark</div> <div>StagesStorageEnvironmentExecutors</div> <div>Spark shell application UI</div> </div>						
Storage						
RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size in Tachyon	Size on Disk

2.5 DSL 演示

SparkSQL 除了支持 HiveQL 和 SQL-92 语法外，还支持 DSL (Domain Specific Language)。在 DSL 中，使用 Scala 符号 '+' 标示符表示基础表中的列，Spark 的 execution engine 会将这些标示符隐式转换成表达式。另外可以在 API 中找到很多 DSL 相关的方法，如 where()、select()、limit() 等等，详细资料可以查看 Catalyst 模块中的 DSL 子模块，下面为其中定义几种常用方法：

//DSL 演示

```
scala> import sqlContext._
```

```
scala> val teenagers_dsl = rddpeople.where('age >= 10).where('age <= 19).select('name)
```

```
scala> teenagers_dsl.map(t => "Name: " + t(0)).collect().foreach(println)
```

```

hadoop1 | hadoop2 | hadoop3
scala> import sqlContext._
import sqlContext._

scala> val teenagers_dsl = rddpeople.where('age >= 10).where('age <= 19).select('name)
teenagers_dsl: org.apache.spark.sql.SchemaRDD =
SchemaRDD[332] at RDD at SchemaRDD.scala:103
== Query Plan ==
== Physical Plan ==
Project [name#268]
  Filter ((age#269 >= 10) && (age#269 <= 19))
    ExistingRDD [name#268,age#269], MapPartitionsRDD[328] at mapPartitions at basicOperators.scala:208

hadoop1 | hadoop2 | hadoop3
scala> teenagers_dsl.map(t => "Name: " + t(0)).collect().foreach(println)
15/07/31 16:13:50 INFO spark.SparkContext: Starting job: collect at <console>:25
15/07/31 16:13:50 INFO scheduler.DAGScheduler: Got job 15 (collect at <console>:25) with 2 output partitions (allowLocal=false)
15/07/31 16:13:50 INFO scheduler.DAGScheduler: Final stage: Stage 87(collect at <console>:25)
15/07/31 16:13:50 INFO scheduler.DAGScheduler: Parents of final stage: List()
15/07/31 16:13:50 INFO scheduler.DAGScheduler: Missing parents: List()
15/07/31 16:13:50 INFO scheduler.DAGScheduler: Submitting Stage 87 (MappedRDD[333] at map at <console>:25), which has no missing parent
15/07/31 16:13:50 INFO storage.MemoryStore: ensureFreeSpace(5608) called with curMem=10194200, maxMem=278302556
15/07/31 16:13:50 INFO storage.MemoryStore: Block broadcast_84 stored as values in memory (estimated size 3.5 KB, free 255.7 MB)
15/07/31 16:13:50 INFO storage.MemoryStore: ensureFreeSpace(3034) called with curMem=10199808, maxMem=278302556
15/07/31 16:13:50 INFO storage.MemoryStore: Block broadcast_84_piece0 stored as bytes in memory (estimated size 3.0 KB, free 255.7 MB)
15/07/31 16:13:50 INFO storage.BlockManagerInfo: Added broadcast_84_piece0 in memory on hadoop1:59535 (size: 3.0 KB, free: 264.6 MB)
15/07/31 16:13:50 INFO storage.BlockManagerMaster: updated info of block broadcast_84_piece0
15/07/31 16:13:50 INFO scheduler.DAGScheduler: Submitting 2 missing tasks from Stage 87 (MappedRDD[333] at map at <console>:25)
15/07/31 16:13:50 INFO scheduler.TaskSchedulerImpl: Adding task set 87.0 with 2 tasks
15/07/31 16:13:50 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 87.0 (TID 5683, hadoop1, NODE_LOCAL, 1193 bytes)
15/07/31 16:13:50 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 87.0 (TID 5684, hadoop2, NODE_LOCAL, 1193 bytes)
15/07/31 16:13:50 INFO storage.BlockManagerInfo: Added broadcast_84_piece0 in memory on hadoop2:35280 (size: 3.0 KB, free: 529.7 MB)
15/07/31 16:13:50 INFO storage.BlockManagerInfo: Added broadcast_84_piece0 in memory on hadoop1:44561 (size: 3.0 KB, free: 529.6 MB)
15/07/31 16:13:50 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 87.0 (TID 5684) in 112 ms on hadoop2 (1/2)
15/07/31 16:13:50 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 87.0 (TID 5683) in 191 ms on hadoop1 (2/2)
15/07/31 16:13:50 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 87.0, whose tasks have all completed, from pool
15/07/31 16:13:50 INFO scheduler.DAGScheduler: Stage 87 (collect at <console>:25) finished in 0.191 s
15/07/31 16:13:50 INFO spark.SparkContext: Job finished: collect at <console>:25, took 0.228110263 s
Name: Justin

```

3 Spark 综合应用

Spark 之所以万人瞩目，除了内存计算还有其 ALL-IN-ONE 的特性，实现了 One stack rule them all。下面简单模拟了几个综合应用场景，不仅使用了 sparkSQL，还使用了其他 Spark 组件：

- SQL On Spark：使用 sqlContext 查询年纪大于等于 10 岁的人名

- Hive On Spark : 使用了 hiveContext 计算每年销售额
- 店铺分类, 根据销售额对店铺分类, 使用 sparkSQL 和 MLLib 聚类算法
- PageRank, 计算最有价值的网页, 使用 sparkSQL 和 GraphX 的 PageRank 算法

以下实验采用 IntelliJ IDEA 调试代码, 最后生成 LearnSpark.jar, 然后使用 spark-submit 提交给集群运行。

3.1 SQL On Spark

3.1.1 实现代码

在 src->main->scala 下创建 class6 包, 在该包中添加 SQLOnSpark 对象文件, 具体代码如下:

```
import org.apache.spark.rdd.RDD
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql.SQLContext

case class Person(name: String, age: Int)

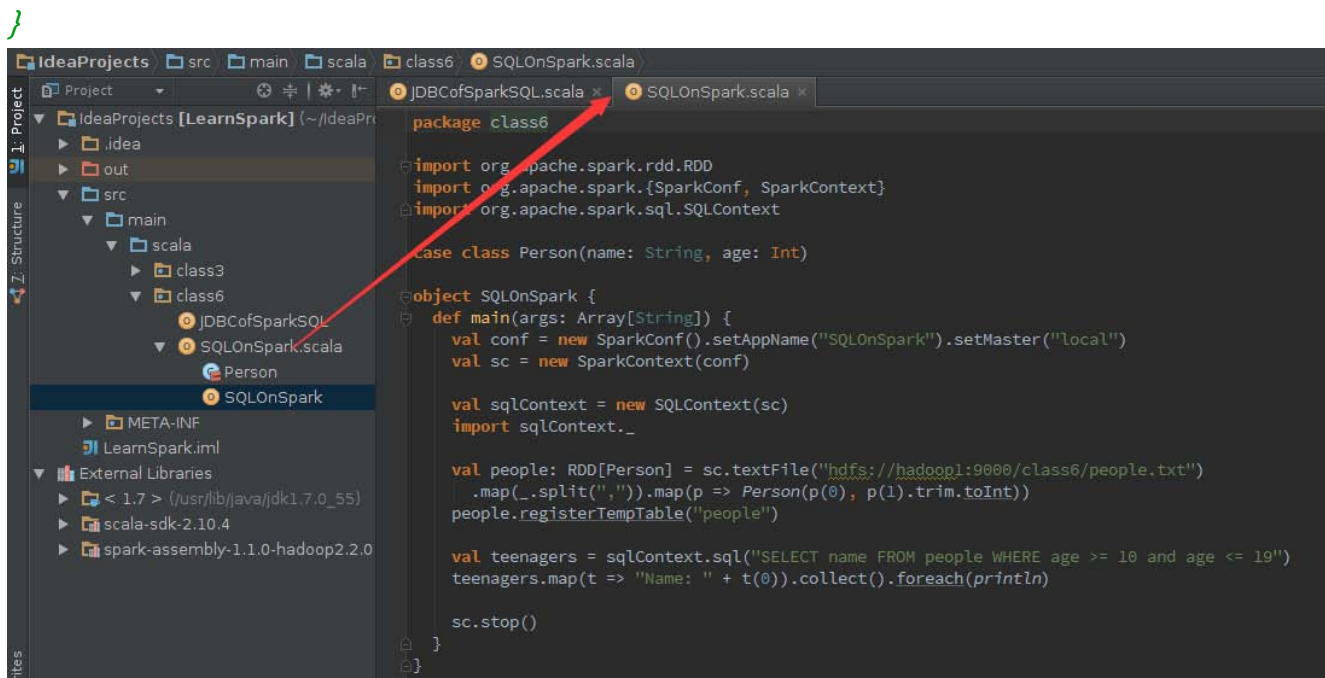
object SQLOnSpark {
  def main(args: Array[String]) {
    val conf = new SparkConf().setAppName("SQLOnSpark")
    val sc = new SparkContext(conf)

    val sqlContext = new SQLContext(sc)
    import sqlContext._

    val people: RDD[Person] = sc.textFile("hdfs://hadoop1:9000/class6/people.txt")
      .map(_.split(",")).map(p => Person(p(0), p(1).trim.toInt))
    people.registerTempTable("people")

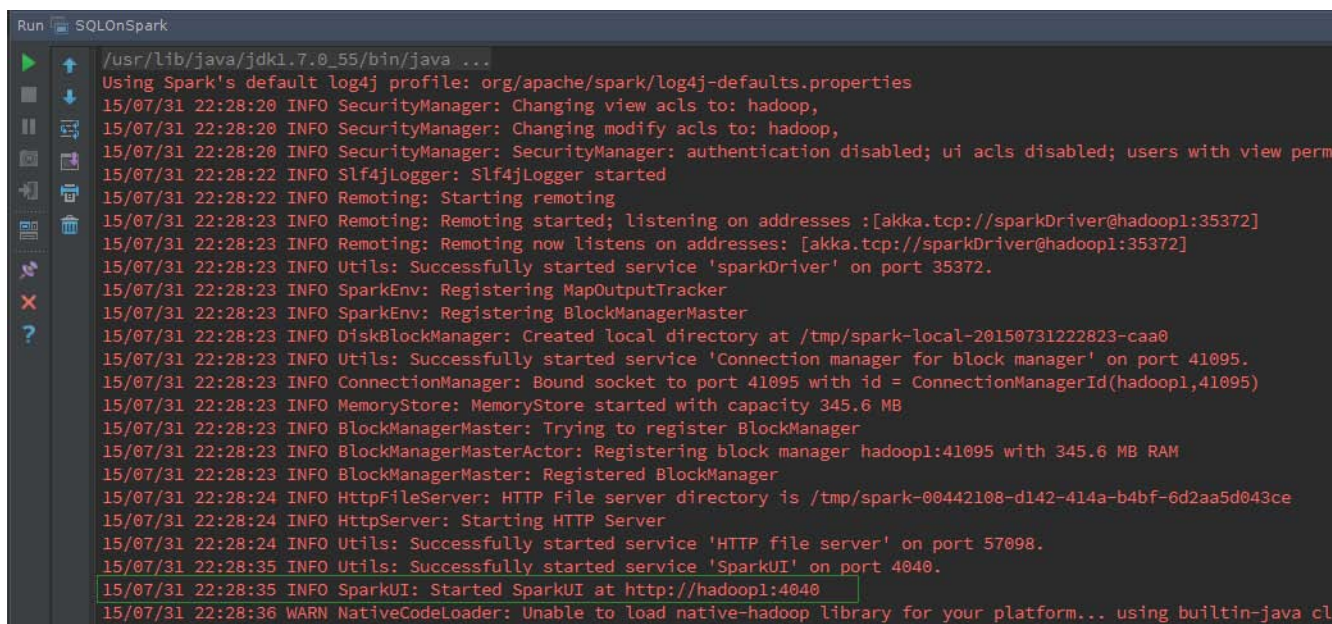
    val teenagers = sqlContext.sql("SELECT name FROM people WHERE age >= 10 and age <= 19")
    teenagers.map(t => "Name: " + t(0)).collect().foreach(println)

    sc.stop()
  }
}
```



3.1.2 IDEA 本地运行

先对该代码进行编译，然后运行该程序，需要注意的是在 IDEA 中需要在 SparkConf 添加 setMaster("local")设置为本地运行。运行时可以通过运行窗口进行观察：



打印运行结果

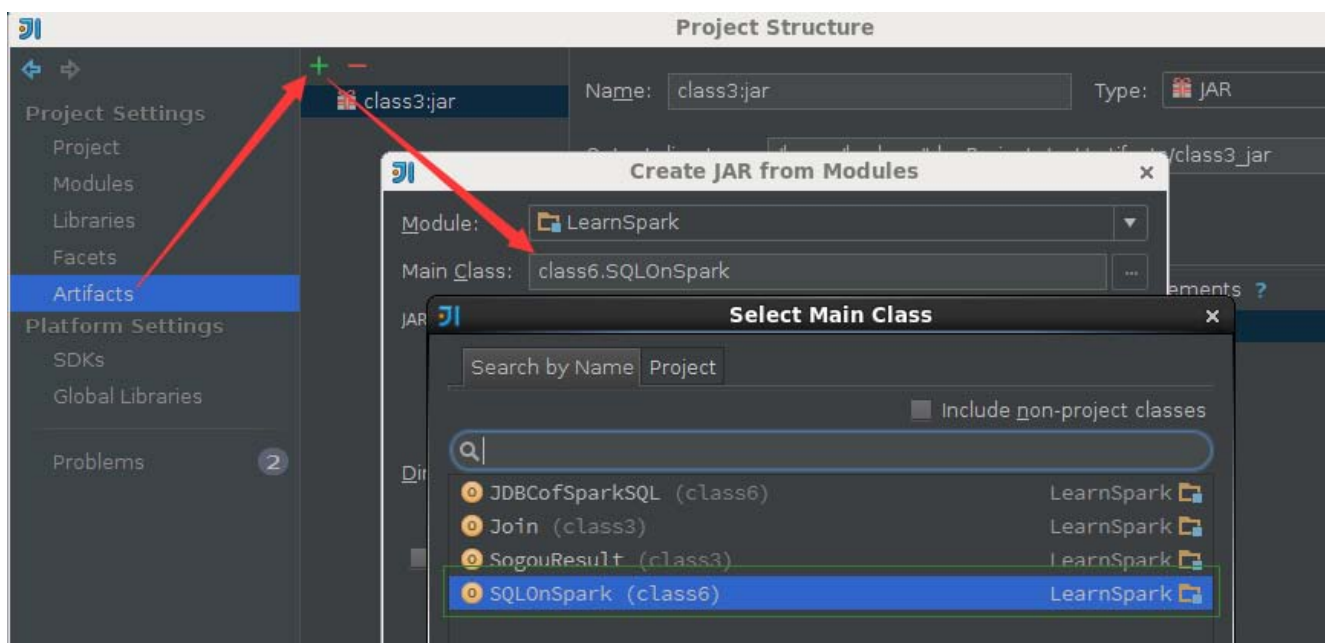

```
15/07/31 22:28:49 INFO deprecation: mapred.tip.id is deprecated. Instead, use mapreduce.task.id
15/07/31 22:28:49 INFO deprecation: mapred.task.id is deprecated. Instead, use mapreduce.task.attempt.id
15/07/31 22:28:49 INFO deprecation: mapred.task.is.map is deprecated. Instead, use mapreduce.task.ismap
15/07/31 22:28:49 INFO deprecation: mapred.task.partition is deprecated. Instead, use mapreduce.task.partition
15/07/31 22:28:49 INFO deprecation: mapred.job.id is deprecated. Instead, use mapreduce.job.id
15/07/31 22:28:50 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0). 1708 bytes result sent to driver
15/07/31 22:28:50 INFO DAGScheduler: Stage 0 (collect at SQLOnSpark.scala:22) finished in 1.569 s
15/07/31 22:28:50 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 1469 ms on localhost (1/1)
15/07/31 22:28:50 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
15/07/31 22:28:50 INFO SparkContext: Job finished: collect at SQLOnSpark.scala:22, took 2.342903206 s
Name: Justin
15/07/31 22:28:50 INFO SparkUI: Stopped Spark web UI at http://hadoop1:4040
15/07/31 22:28:50 INFO DAGScheduler: Stopping DAGScheduler
```

3.1.3 生成打包文件

【注】可以参见第3课《Spark 编程模型（下）--IDEA 搭建及实战》进行打包

第一步 配置打包信息

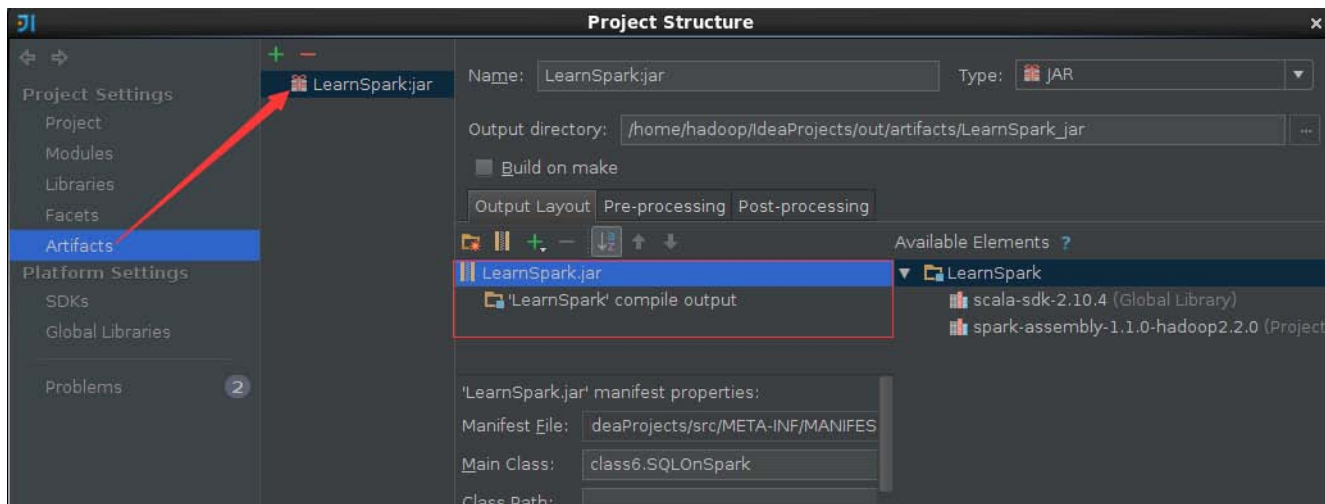
在项目结构界面中选择"Artifacts",在右边操作界面选择绿色"+"号,选择添加JAR包的"From modules with dependencies"方式,出现如下界面,在该界面中选择主函数入口为SQLOnSpark:



第二步 填写该JAR包名称和调整输出内容

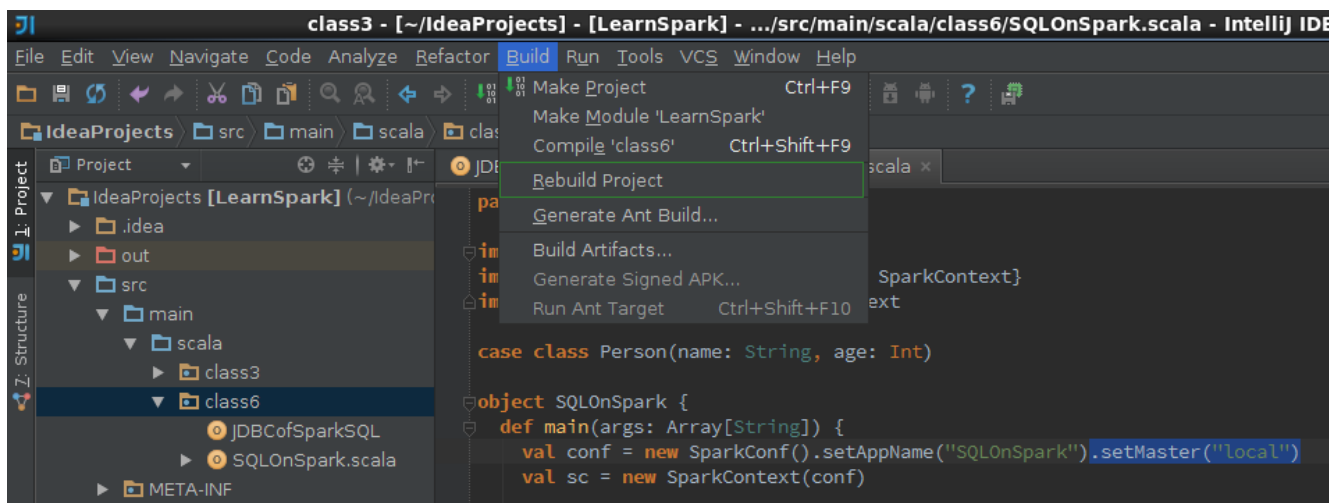
打包路径为/home/hadoop/IdeaProjects/out/artifacts/LearnSpark.jar

【注意】的是默认情况下"Output Layout"会附带Scala相关的类包,由于运行环境已经有Scala相关类包,所以在这里去除这些包只保留项目的输出内容



第三步 输出打包文件

点击菜单 Build->Build Artifacts, 弹出选择动作, 选择 Build 或者 Rebuild 动作

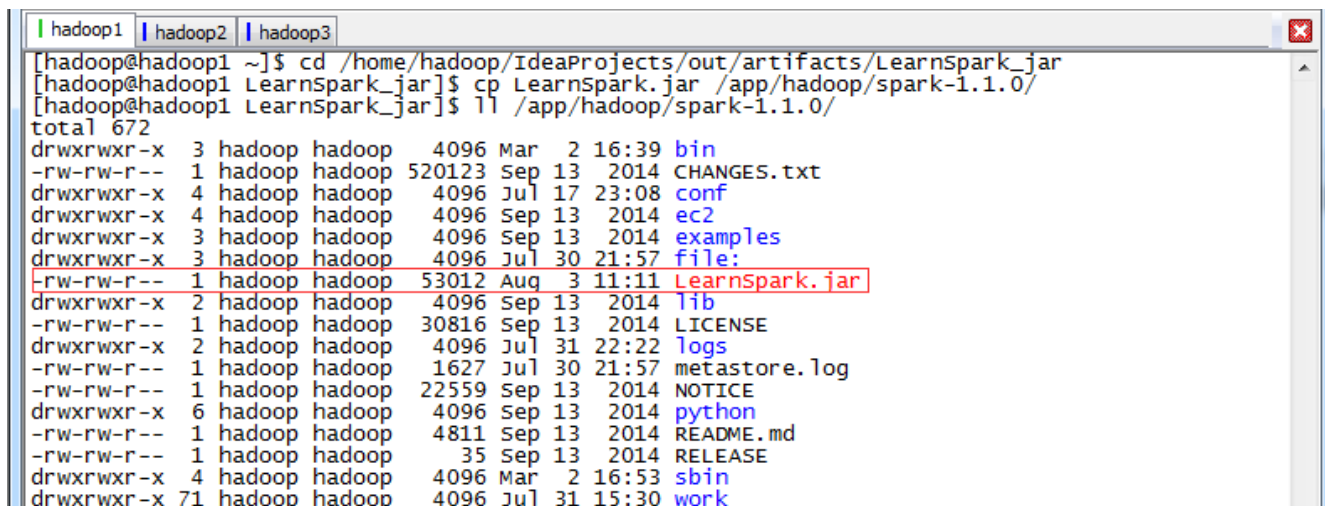


第四步 复制打包文件到 Spark 根目录下

cd /home/hadoop/IdeaProjects/out/artifacts/LearnSpark.jar

cp LearnSpark.jar /app/hadoop/spark-1.1.0/

ll /app/hadoop/spark-1.1.0/

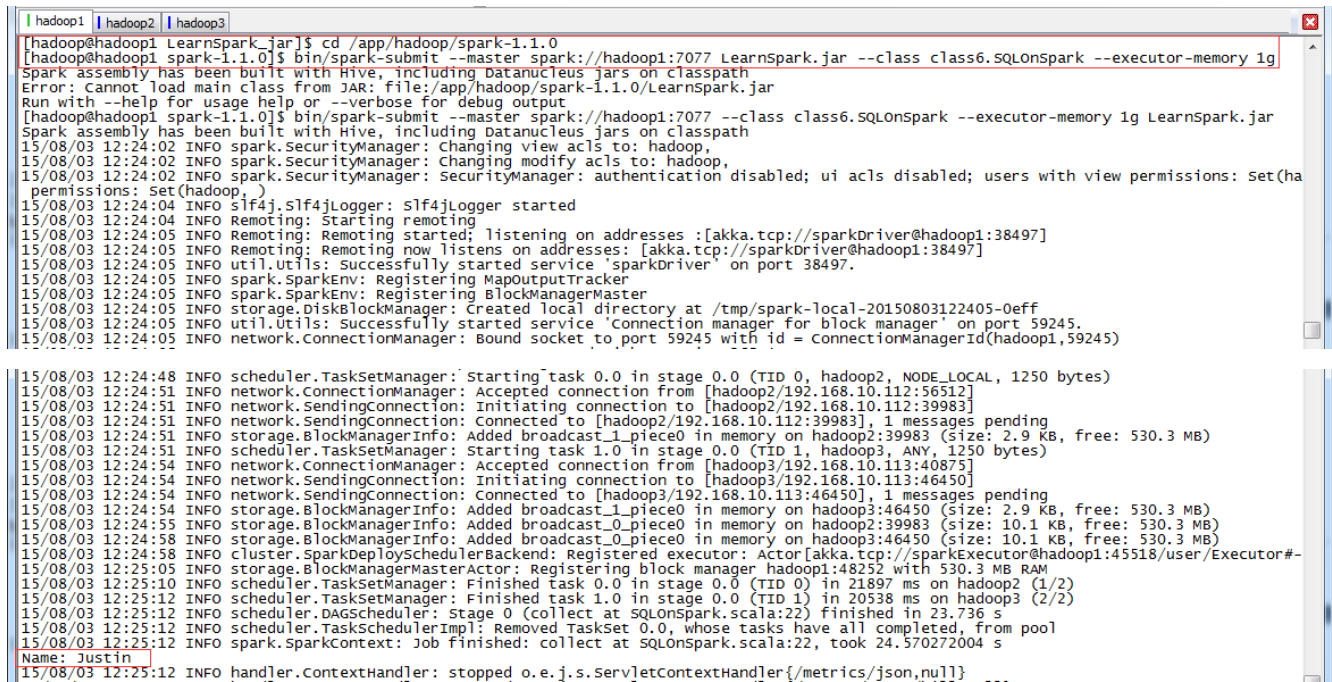


3.1.4 运行查看结果

通过如下命令调用打包中的 SQLOnSpark 方法，运行结果如下：

```
cd /app/hadoop/spark-1.1.0
```

```
bin/spark-submit --master spark://hadoop1:7077 --class class6.SQLOnSpark  
--executor-memory 1g LearnSpark.jar
```



```
[hadoop1@hadoop1 LearnSpark.jar]$ cd /app/hadoop/spark-1.1.0  
[hadoop1@hadoop1 spark-1.1.0]$ bin/spark-submit --master spark://hadoop1:7077 LearnSpark.jar --class class6.SQLOnSpark --executor-memory 1g  
Spark assembly has been built with Hive, including Datanucleus jars on classpath  
Error: cannot load main class from JAR: file:/app/hadoop/spark-1.1.0/LearnSpark.jar  
Run with --help for usage help or --verbose for debug output  
[hadoop1@hadoop1 spark-1.1.0]$ bin/spark-submit --master spark://hadoop1:7077 --class class6.SQLOnSpark --executor-memory 1g LearnSpark.jar  
Spark assembly has been built with Hive, including Datanucleus jars on classpath  
15/08/03 12:24:02 INFO spark.SecurityManager: Changing view acls to: hadoop,  
15/08/03 12:24:02 INFO spark.SecurityManager: Changing modify acls to: hadoop,  
15/08/03 12:24:02 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(ha  
permissions: Set(hadoop,  
15/08/03 12:24:04 INFO slf4j.Slf4jLogger: Slf4jLogger started  
15/08/03 12:24:04 INFO Remoting: Starting remoting  
15/08/03 12:24:05 INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sparkDriver@hadoop1:38497]  
15/08/03 12:24:05 INFO Remoting: Remoting now listens on addresses: [akka.tcp://sparkDriver@hadoop1:38497]  
15/08/03 12:24:05 INFO util.Utils: Successfully started service 'sparkDriver' on port 38497.  
15/08/03 12:24:05 INFO spark.SparkEnv: Registering MapOutputTracker  
15/08/03 12:24:05 INFO spark.SparkEnv: Registering BlockManagerMaster  
15/08/03 12:24:05 INFO storage.DiskBlockManager: Created local directory at /tmp/spark-local-20150803122405-0eff  
15/08/03 12:24:05 INFO util.Utils: Successfully started service 'Connection manager for block manager' on port 59245.  
15/08/03 12:24:05 INFO network.ConnectionManager: Bound socket to port 59245 with id = ConnectionManagerId(hadoop1,59245)  
  
15/08/03 12:24:48 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, hadoop2, NODE_LOCAL, 1250 bytes)  
15/08/03 12:24:51 INFO network.ConnectionManager: Accepted connection from [hadoop2/192.168.10.112:56512]  
15/08/03 12:24:51 INFO network.SendingConnection: Initiating connection to [hadoop2/192.168.10.112:39983]  
15/08/03 12:24:51 INFO network.SendingConnection: Connected to [hadoop2/192.168.10.112:39983], 1 messages pending  
15/08/03 12:24:51 INFO storage.BlockManagerInfo: Added broadcast_1_piece0 in memory on hadoop2:39983 (size: 2.9 KB, free: 530.3 MB)  
15/08/03 12:24:51 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, hadoop3, ANY, 1250 bytes)  
15/08/03 12:24:51 INFO network.ConnectionManager: Accepted connection from [hadoop3/192.168.10.113:40875]  
15/08/03 12:24:54 INFO network.SendingConnection: Initiating connection to [hadoop3/192.168.10.113:46450]  
15/08/03 12:24:54 INFO network.SendingConnection: Connected to [hadoop3/192.168.10.113:46450], 1 messages pending  
15/08/03 12:24:54 INFO storage.BlockManagerInfo: Added broadcast_1_piece0 in memory on hadoop3:46450 (size: 2.9 KB, free: 530.3 MB)  
15/08/03 12:24:55 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on hadoop2:39983 (size: 10.1 KB, free: 530.3 MB)  
15/08/03 12:24:55 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on hadoop3:46450 (size: 10.1 KB, free: 530.3 MB)  
15/08/03 12:24:58 INFO cluster.sparkDeploySchedulerBackend: Registered executor: Actor[akka.tcp://sparkExecutor@hadoop1:45518/user/Executor#-  
15/08/03 12:25:05 INFO storage.BlockManagerMasterActor: Registering block manager hadoop1:48252 with 530.3 MB RAM  
15/08/03 12:25:10 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 21897 ms on hadoop2 (1/2)  
15/08/03 12:25:12 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 20538 ms on hadoop3 (2/2)  
15/08/03 12:25:12 INFO scheduler.DAGScheduler: Stage 0 (collect at SQLOnSpark.scala:22) finished in 23.736 s  
15/08/03 12:25:12 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool  
15/08/03 12:25:12 INFO spark.SparkContext: Job finished: collect at SQLOnSpark.scala:22, took 24.570272004 s  
Name: Justin  
15/08/03 12:25:12 INFO handler.ContextHandler: stopped o.e.j.s.ServletContextHandler{/metrics/json,null} ...
```

3.2 Hive On Spark

3.2.1 实现代码

在 class6 包中添加 HiveOnSpark 对象文件，具体代码如下：

```
import org.apache.spark.{SparkConf, SparkContext}
```

```
import org.apache.spark.sql.hive.HiveContext
```

```
object HiveOnSpark {
```

```
  case class Record(key: Int, value: String)
```

```
  def main(args: Array[String]) {
```

```
    val sparkConf = new SparkConf().setAppName("HiveOnSpark")
```

```
    val sc = new SparkContext(sparkConf)
```

```
    val hiveContext = new HiveContext(sc)
```

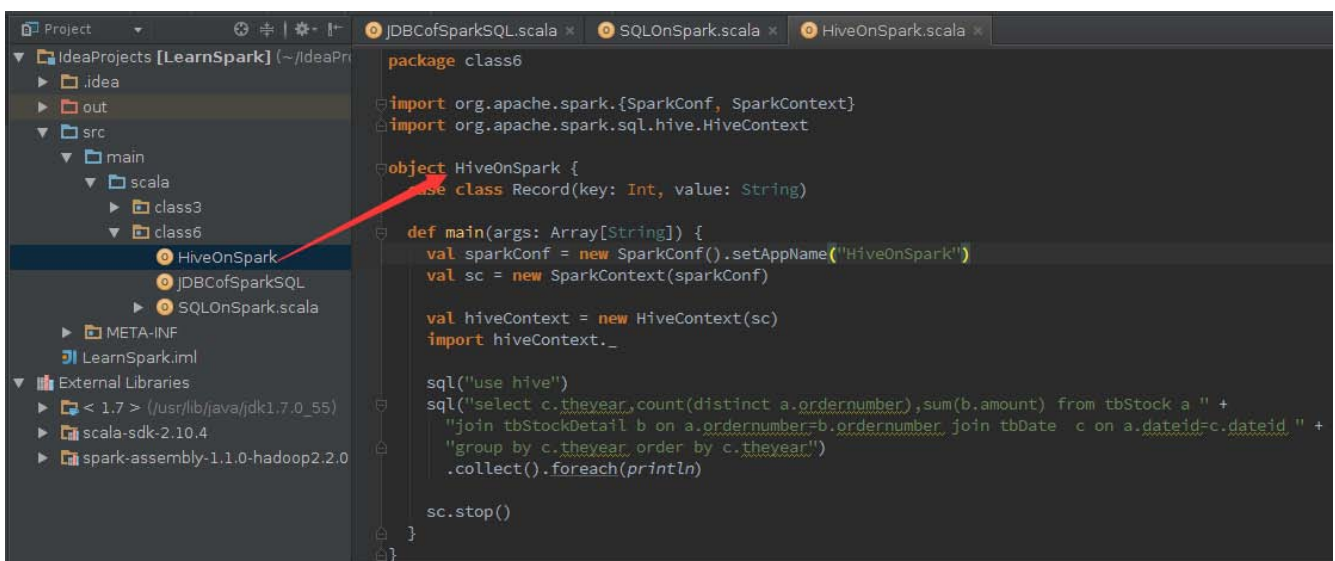
```
import hiveContext._
```

```
sql("use hive")
```

```
sql("select c.theyear,count(distinct a.ordernumber),sum(b.amount) from tbStock a  
join tbStockDetail b on a.ordernumber=b.ordernumber join tbDate c on  
a.dateid=c.dateid group by c.theyear order by c.theyear")  
.collect().foreach(println)
```

```
sc.stop()
```

```
}  
}
```



3.2.2 生成打包文件

按照 3.1.3 SQL On Spark 方法进行打包

3.2.3 运行查看结果

【注】 需要启动 Hive 服务，参见 2.3.1

通过如下命令调用打包中的 SQLOnSpark 方法，运行结果如下：

```
cd /app/hadoop/spark-1.1.0
```

```
bin/spark-submit --master spark://hadoop1:7077 --class class6.HiveOnSpark  
--executor-memory 1g LearnSpark.jar
```



```
hadoop@hadoop1 spark-1.1.0$ bin/spark-submit --master spark://hadoop1:7077 --class class6.HiveOnSpark --executor-memory 1g LearnSpark.jar
Spark assembly has been built with Hive, including datanucleus jars on classpath
15/08/03 14:02:28 INFO spark.SecurityManager: changing view acls to: hadoop,
15/08/03 14:02:28 INFO spark.SecurityManager: changing modify acls to: hadoop,
15/08/03 14:02:28 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(hadoop,); users with modify permissions: Set(hadoop,)
15/08/03 14:02:28 INFO slf4j.Slf4jLogger: Slf4jLogger started
15/08/03 14:02:28 INFO Remoting: Starting remoting
15/08/03 14:02:29 INFO Remoting: Remoting started; listening on addresses : [akka.tcp://sparkDriver@hadoop1:33917]
15/08/03 14:02:29 INFO Remoting: Remoting now listens on addresses: [akka.tcp://sparkDriver@hadoop1:33917]
15/08/03 14:02:29 INFO util.Utils: Successfully started service 'sparkDriver' on port 33917.
15/08/03 14:02:29 INFO spark.SparkEnv: Registering MapOutputTracker
15/08/03 14:02:29 INFO spark.SparkEnv: Registering BlockManagerMaster
15/08/03 14:02:29 INFO storage.DiskBlockManager: Created local directory at /tmp/spark-local-20150803140229-498a
15/08/03 14:02:29 INFO util.Utils: Successfully started service 'connection manager' on port 57359.
15/08/03 14:02:29 INFO network.ConnectionManager: Bound socket to port 57359 with id = connectionManagerId(hadoop1,57359)
15/08/03 14:02:29 INFO storage.MemoryStore: MemoryStore started with capacity 265.4 MB
15/08/03 14:02:29 INFO storage.BlockManagerMaster: Trying to register BlockManager
15/08/03 14:02:29 INFO storage.BlockManagerMasterActor: Registering block manager hadoop1:57359 with 265.4 MB RAM
15/08/03 14:02:29 INFO storage.BlockManagerMaster: Registered BlockManager
15/08/03 14:02:29 INFO spark.HttpFileServer: HTTP File server directory is /tmp/spark-02c8628c-388c-400e-9d8b-ba5ac061b051

15/08/03 13:59:08 INFO scheduler.TaskSetManager: Finished task 3.0 in stage 6.0 (TID 809) in 56 ms on hadoop1 (2/8)
15/08/03 13:59:08 INFO scheduler.TaskSetManager: Starting task 5.0 in stage 6.0 (TID 811, hadoop2, PROCESS_LOCAL, 1005 bytes)
15/08/03 13:59:08 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 6.0 (TID 806) in 225 ms on hadoop2 (3/8)
15/08/03 13:59:08 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 6.0 (TID 807) in 236 ms on hadoop3 (4/8)
15/08/03 13:59:08 INFO scheduler.TaskSetManager: Starting task 6.0 in stage 6.0 (TID 812, hadoop3, PROCESS_LOCAL, 1005 bytes)
15/08/03 13:59:08 INFO scheduler.TaskSetManager: Starting task 7.0 in stage 6.0 (TID 813, hadoop1, PROCESS_LOCAL, 1005 bytes)
15/08/03 13:59:08 INFO scheduler.TaskSetManager: Finished task 4.0 in stage 6.0 (TID 810) in 53 ms on hadoop1 (5/8)
15/08/03 13:59:08 INFO scheduler.TaskSetManager: Finished task 7.0 in stage 6.0 (TID 813) in 20 ms on hadoop1 (6/8)
15/08/03 13:59:08 INFO scheduler.TaskSetManager: Finished task 6.0 in stage 6.0 (TID 812) in 70 ms on hadoop3 (7/8)
15/08/03 13:59:08 INFO scheduler.TaskSetManager: Finished task 5.0 in stage 6.0 (TID 811) in 89 ms on hadoop2 (8/8)
15/08/03 13:59:08 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 6.0, whose tasks have all completed, from pool
15/08/03 13:59:08 INFO scheduler.DAGScheduler: Stage 6 (collect at SparkPlan.scala:85) finished in 0.304 s
15/08/03 13:59:08 INFO spark.SparkContext: Job finished: collect at SparkPlan.scala:85, took 5.044423314 s
[2004,1094,3265696]
[2005,3828,13247234]
[2006,3772,13670416]
[2007,4885,16711974]
[2008,4861,14670698]
[2009,2619,6322137]
[2010,94,2109241]
15/08/03 13:59:08 INFO handler.ContextHandler: stopped a e i s ServletContextHandler[/metrics/json null]
```

通过监控页面看到名为 HiveOnSpark 的作业运行情况：

Running Applications

ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20150803135655-0000	HiveOnSpark	3	1024.0 MB	2015/08/03 13:56:55	hadoop	RUNNING	60 s

Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
12	mapPartitions at Exchange.scala:71	+details (kill) 2015/08/03 13:59:03	4 s	187/200		1229.0 B	59.0 B

Completed Stages (6)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
0	RangePartitioner at Exchange.scala:79	+details 2015/08/03 13:58:57	6 s	200/200		143.7 KB	
5	mapPartitions at Exchange.scala:48	+details 2015/08/03 13:58:27	30 s	200/200		1909.0 KB	215.0 KB
4	mapPartitions at Exchange.scala:48	+details 2015/08/03 13:57:54	33 s	200/200		1402.7 KB	2.7 MB
3	mapPartitions at Exchange.scala:48	+details 2015/08/03 13:57:23	24 s	2/2	11.4 MB		1713.1 KB
2	mapPartitions at Exchange.scala:48	+details 2015/08/03 13:57:23	31 s	2/2	588.0 KB		410.6 KB
1	mapPartitions at Exchange.scala:48	+details 2015/08/03 13:57:23	31 s	2/2	167.5 KB		86.6 KB

3.3 店铺分类

分类在实际应用中非常普遍，比如对客户进行分类、对店铺进行分类等等，对不同类别采取不同的策略，可以有效的降低企业的营运成本、增加收入。机器学习中的聚类就是一种根据不同的特征数据，结合用户指定的类别数量，将数据分成几个类的方法。下面举个简单的例子，按照销售数量和销售金额这两个特征数据，进行聚类，分出 3 个等级的店铺。

3.3.1 实现代码

```
import org.apache.log4j.{Level, Logger}
import org.apache.spark.sql.catalyst.expressions.Row
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.sql.hive.HiveContext
import org.apache.spark.mllib.clustering.KMeans
import org.apache.spark.mllib.linalg.Vectors

object SQLMLlib {
  def main(args: Array[String]) {
    //屏蔽不必要的日志显示在终端上
    Logger.getLogger("org.apache.spark").setLevel(Level.WARN)
    Logger.getLogger("org.eclipse.jetty.server").setLevel(Level.OFF)

    //设置运行环境
    val sparkConf = new SparkConf().setAppName("SQLMLlib")
    val sc = new SparkContext(sparkConf)
    val hiveContext = new HiveContext(sc)

    //使用 sparksql 查出每个店的销售数量和金额
    hiveContext.sql("use hive")
    hiveContext.sql("SET spark.sql.shuffle.partitions=20")
    val sqldata = hiveContext.sql("select a.locationid, sum(b.qty)
totalqty,sum(b.amount) totalamount from tbStock a join tbStockDetail b on
a.ordernumber=b.ordernumber group by a.locationid")

    //将查询数据转换成向量
    val parsedData = sqldata.map {
      case Row(_, totalqty, totalamount) =>
        val features = Array[Double](totalqty.toString.toDouble,
totalamount.toString.toDouble)
        Vectors.dense(features)
    }

    //对数据集聚类，3 个类，20 次迭代，形成数据模型
```


//注意这里会使用设置的 partition 数 20

```
val numClusters = 3
```

```
val numIterations = 20
```

```
val model = KMeans.train(parsedData, numClusters, numIterations)
```

//用模型对读入的数据进行分类，并输出

//由于 partition 没设置，输出为 200 个小文件，可以使用 bin/hdfs dfs -getmerge 合并下载到本地

```
val result2 = sqldata.map {
```

```
  case Row(locationid, totalqty, totalamount) =>
```

```
    val features = Array[Double](totalqty.toString.toDouble,  
    totalamount.toString.toDouble)
```

```
    val linevectore = Vectors.dense(features)
```

```
    val prediction = model.predict(linevectore)
```

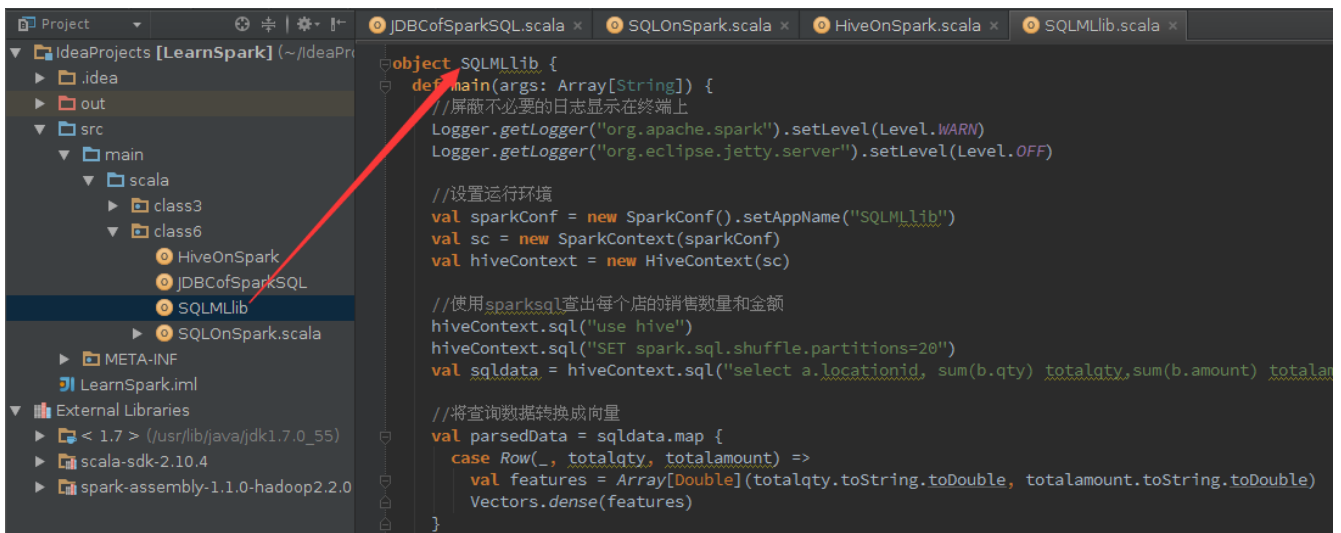
```
    locationid + " " + totalqty + " " + totalamount + " " + prediction
```

```
  }.saveAsTextFile(args(0))
```

```
sc.stop()
```

```
}
```

```
}
```



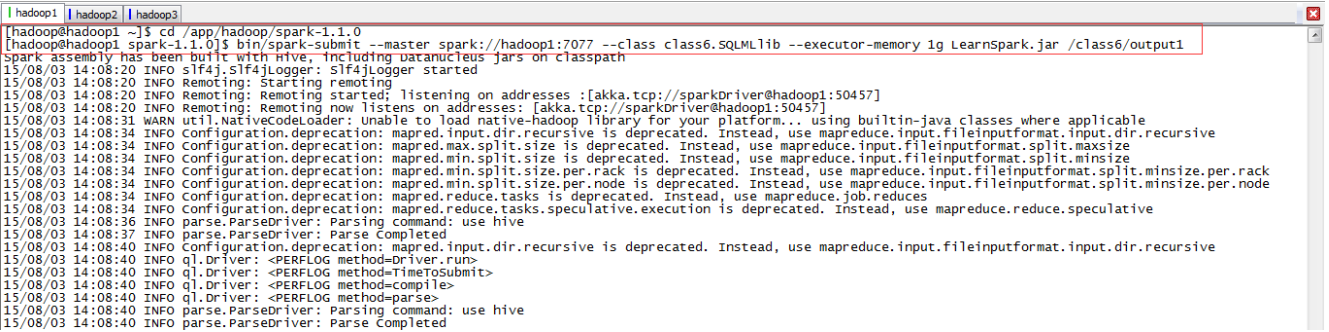
3.3.2 生成打包文件

按照 3.1.3 SQL On Spark 方法进行打包

3.3.3 运行查看结果

通过如下命令调用打包中的 SQLOnSpark 方法：

cd /app/hadoop/spark-1.1.0
bin/spark-submit --master spark://hadoop1:7077 --class class6.SQLMLlib
--executor-memory 1g LearnSpark.jar /class6/output1



运行过程，可以发现聚类过程都是使用 20 个 partition：

Active Stages (1)								
Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
62	mapPartitions at KMeans.scala:171	+details (kill)	2015/08/03 14:09:44	97 ms	0/20			

Completed Stages (22)								
Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Shuffle Read	Shuffle Write
53	collectAsMap at KMeans.scala:298	+details	2015/08/03 14:09:43	0.3 s	20/20		2.2 KB	
57	flatMap at KMeans.scala:294	+details	2015/08/03 14:09:42	0.8 s	20/20	824.0 B	11.4 KB	3.5 KB
49	collect at KMeans.scala:283	+details	2015/08/03 14:09:41	1 s	20/20	824.0 B	11.4 KB	
44	collectAsMap at KMeans.scala:275	+details	2015/08/03 14:09:41	0.2 s	20/20		2000.0 B	
48	flatMap at KMeans.scala:271	+details	2015/08/03 14:09:40	0.8 s	20/20	824.0 B	11.4 KB	2.3 KB
40	collect at KMeans.scala:283	+details	2015/08/03 14:09:38	1 s	20/20	824.0 B	11.4 KB	
35	collectAsMap at KMeans.scala:275	+details	2015/08/03 14:09:38	0.3 s	20/20		800.0 B	
39	flatMap at KMeans.scala:271	+details	2015/08/03 14:09:37	1 s	20/20	824.0 B	11.4 KB	2.3 KB
31	collect at KMeans.scala:283	+details	2015/08/03 14:09:36	0.9 s	20/20	824.0 B	11.4 KB	

查看运行结果，分为 20 个文件存放在 HDFS 中

Contents of directory [/class6/output1](#)

Goto : go

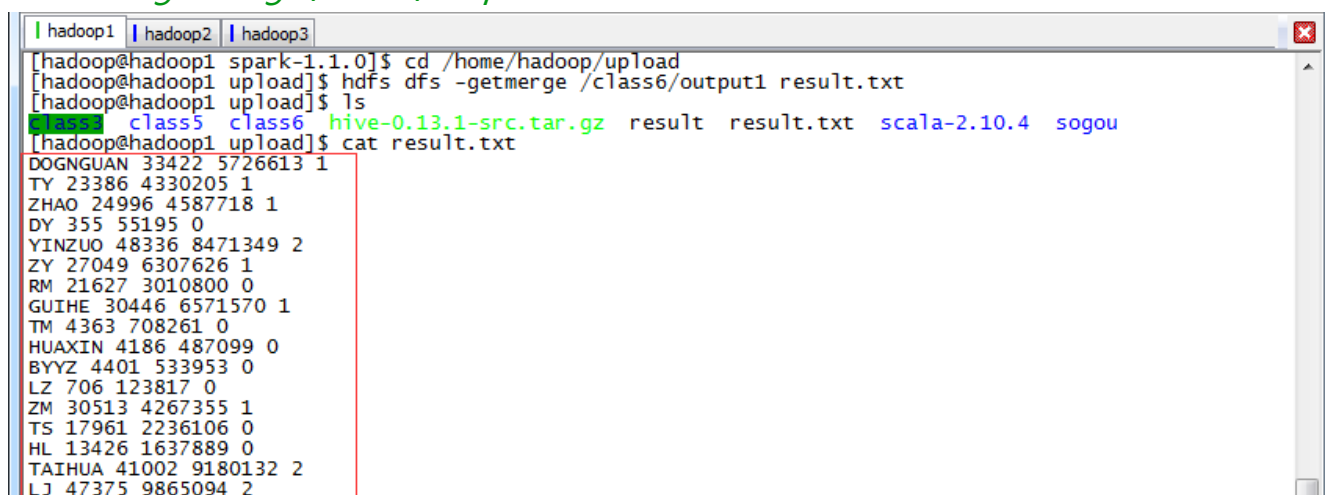
[Go to parent directory](#)

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 B	2	128 MB	2015-08-03 14:09	rw-r--r--	hadoop	supergroup
part-00000	file	0 B	3	128 MB	2015-08-03 14:09	rw-r--r--	hadoop	supergroup
part-00001	file	0 B	3	128 MB	2015-08-03 14:09	rw-r--r--	hadoop	supergroup
part-00002	file	44 B	3	128 MB	2015-08-03 14:09	rw-r--r--	hadoop	supergroup
part-00003	file	0 B	3	128 MB	2015-08-03 14:09	rw-r--r--	hadoop	supergroup
part-00004	file	0 B	3	128 MB	2015-08-03 14:09	rw-r--r--	hadoop	supergroup
part-00005	file	21 B	3	128 MB	2015-08-03 14:09	rw-r--r--	hadoop	supergroup
part-00006	file	15 B	3	128 MB	2015-08-03 14:09	rw-r--r--	hadoop	supergroup
part-00007	file	23 B	3	128 MB	2015-08-03 14:09	rw-r--r--	hadoop	supergroup
part-00008	file	38 B	3	128 MB	2015-08-03 14:09	rw-r--r--	hadoop	supergroup

使用 getmerge 将结果转到本地文件，并查看结果：

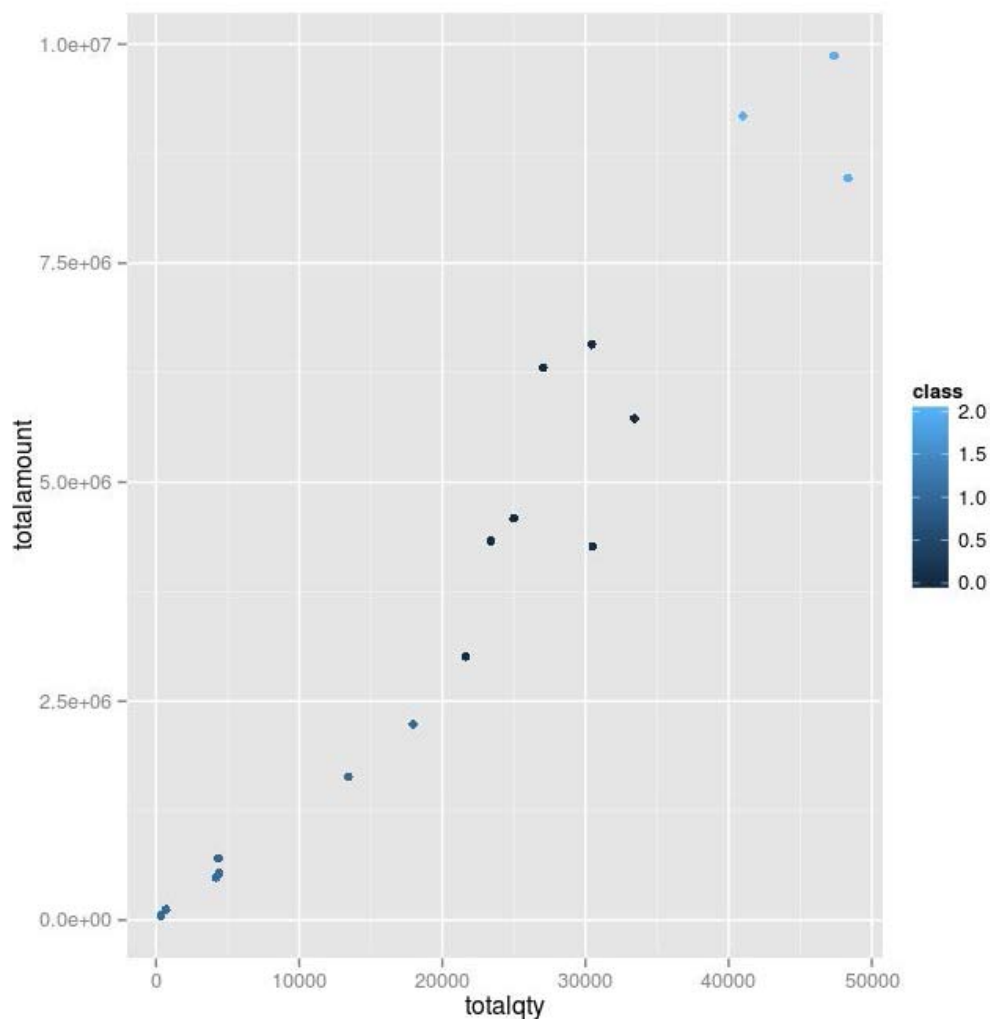
```
cd /home/hadoop/upload
```

```
hdfs dfs -getmerge /class6/output1 result.txt
```



```
hadoop1 | hadoop2 | hadoop3
[hadoop@hadoop1 spark-1.1.0]$ cd /home/hadoop/upload
[hadoop@hadoop1 upload]$ hdfs dfs -getmerge /class6/output1 result.txt
[hadoop@hadoop1 upload]$ ls
class5  class6  hive-0.13.1-src.tar.gz  result  result.txt  scala-2.10.4  sogou
[hadoop@hadoop1 upload]$ cat result.txt
DOGNUAN 33422 5726613 1
TY 23386 4330205 1
ZHAO 24996 4587718 1
DY 355 55195 0
YINZUO 48336 8471349 2
ZY 27049 6307626 1
RM 21627 3010800 0
GUIHE 30446 6571570 1
TM 4363 708261 0
HUAXIN 4186 487099 0
BYYZ 4401 533953 0
LZ 706 123817 0
ZM 30513 4267355 1
TS 17961 2236106 0
HL 13426 1637889 0
TAIHUA 41002 9180132 2
LJ 47375 9865094 2
```

最后使用 R 做示意图，用 3 种不同的颜色表示不同的类别。



3.4 PageRank

PageRank,即网页排名，又称网页级别、Google 左侧排名或佩奇排名，是 Google 创始人拉里·佩奇和谢尔盖·布林于 1997 年构建早期的搜索系统原型时提出的链接分析算法。目前很多重要的链接分析算法都是在 PageRank 算法基础上衍生出来的。PageRank 是 Google 用于用来标识网页的等级/重要性的一种方法，是 Google 用来衡量一个网站的好坏的唯一标准。在揉合了诸如 Title 标识和 Keywords 标识等所有其它因素之后，Google 通过 PageRank 来调整结果，使那些更具“等级/重要性”的网页在搜索结果中令网站排名获得提升，从而提高搜索结果的相关性和质量。

Spark GraphX 引入了 google 公司的图处理引擎 pregel，可以方便的实现 PageRank 的计算。

3.4.1 创建表

下面实例采用的数据是 wiki 数据中含有 Berkeley 标题的网页之间连接关系，数据为两个文件：graphx-wiki-vertices.txt 和 graphx-wiki-edges.txt，可以分别用于图计算的顶点和边。把这两个文件上传到本地文件系统/home/hadoop/upload/class6 目录中（注：这两个文件可

以从该系列附属资源/data/class6 中获取)

第一步 上传数据

```
hadoop1 | hadoop2 | hadoop3
[hadoop@hadoop1 ~]$ cd /home/hadoop/upload/class6
[hadoop@hadoop1 class6]$ ll
total 2164
-rwxrw-rw- 1 hadoop hadoop 1247306 Jun 19 2014 graphx-wiki-edges.txt
-rwxrw-rw- 1 hadoop hadoop 946608 Jun 19 2014 graphx-wiki-vertices.txt
-rw-rw-r-- 1 hadoop hadoop 838 Feb 28 17:14 nestjson.json
-rwxrw-rw- 1 hadoop hadoop 73 Aug 31 2014 people.json
-rwxrw-rw- 1 hadoop hadoop 32 Jun 2 2014 people.txt
drwxr-xr-x 2 hadoop hadoop 4096 Feb 4 14:07 wiki_parquet
[hadoop@hadoop1 class6]$
```

第二步 启动 SparkSQL

参见第 6 课《SparkSQL (一) --SparkSQL 简介》3.2.3 启动 SparkSQL

```
$cd /app/hadoop/spark-1.1.0
```

```
$bin/spark-sql --master spark://hadoop1:7077 --executor-memory 1g
```

Running Applications

ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20150803143919-0003	SparkSQL::hadoop1	3	1024.0 MB	2015/08/03 14:39:19	hadoop	RUNNING	3 s

第三步 定义表并加载数据

创建 vertices 和 edges 两个表并加载数据：

```
spark-sql>show databases;
```

```
spark-sql>use hive;
```

```
spark-sql>CREATE TABLE vertices(ID BigInt,Title String) ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'; LOAD DATA LOCAL INPATH
'/home/hadoop/upload/class6/graphx-wiki-vertices.txt' INTO TABLE vertices;
```

```
hadoop1 | hadoop2 | hadoop3
spark-sql> CREATE TABLE vertices(ID BigInt,Title String) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'; LOAD DATA LOCAL INPATH '/home/hadoop/upload/class6/graphx-wiki-vertices.txt' INTO TABLE vertices;
15/08/03 14:46:07 INFO parse.ParseDriver: Parsing command: CREATE TABLE vertices(ID BigInt,Title String) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
15/08/03 14:46:07 INFO parse.ParseDriver: Parse Completed
15/08/03 14:46:07 INFO Configuration.deprecation: mapreduce.input.dir.recursive is deprecated. Instead, use mapreduce.input.fileinputformat.input.dir.recursive
15/08/03 14:46:07 INFO ql.Driver: <PERFLOG method=Driver.run>
15/08/03 14:46:07 INFO ql.Driver: <PERFLOG method=TimeToSubmit>
15/08/03 14:46:07 INFO ql.Driver: <PERFLOG method=compile>
15/08/03 14:46:07 INFO ql.Driver: <PERFLOG method=parse>
15/08/03 14:46:07 INFO parse.ParseDriver: Parsing command: CREATE TABLE vertices(ID BigInt,Title String) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
15/08/03 14:46:07 INFO parse.ParseDriver: Parse Completed
15/08/03 14:46:07 INFO ql.Driver: <PERFLOG method=semanticAnalyze>
15/08/03 14:46:07 INFO ql.Driver: <PERFLOG method=semanticAnalyze>
15/08/03 14:46:07 INFO parse.SemanticAnalyzer: Starting Semantic Analysis
15/08/03 14:46:07 INFO parse.SemanticAnalyzer: Creating table vertices position=13
15/08/03 14:46:07 INFO ql.Driver: Semantic Analysis Completed
```

```
spark-sql>CREATE TABLE edges(SRCID BigInt,DISTID BigInt) ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'; LOAD DATA LOCAL INPATH
'/home/hadoop/upload/class6/graphx-wiki-edges.txt' INTO TABLE edges;
```

```
hadoop1 | hadoop2 | hadoop3
spark-sql> CREATE TABLE edges(SRCID BigInt,DISTID BigInt) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'; LOAD DATA LOCAL INPATH '/home/hadoop/upload/class6/graphx-wiki-edges.txt' INTO TABLE edges;
15/08/03 14:47:00 INFO parse.ParseDriver: Parsing command: CREATE TABLE edges(SRCID BigInt,DISTID BigInt) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
15/08/03 14:47:00 INFO parse.ParseDriver: Parse Completed
15/08/03 14:47:00 INFO Configuration.deprecation: mapred.input.dir.recursive is deprecated. Instead, use mapreduce.input.fileinputformat.input.dir.recursive
15/08/03 14:47:00 INFO ql.Driver: <PERFLOG method=Driver.run>
15/08/03 14:47:00 INFO ql.Driver: <PERFLOG method=TimeToSubmit>
15/08/03 14:47:00 INFO ql.Driver: <PERFLOG method=compile>
15/08/03 14:47:00 INFO ql.Driver: <PERFLOG method=parse>
15/08/03 14:47:00 INFO parse.ParseDriver: Parsing command: CREATE TABLE edges(SRCID BigInt,DISTID BigInt) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
15/08/03 14:47:00 INFO parse.ParseDriver: Parse Completed
15/08/03 14:47:00 INFO ql.Driver: </PERFLOG method=parse_start=1438584420271 end=1438584420272 duration=1>
15/08/03 14:47:00 INFO ql.Driver: <PERFLOG method=SemanticAnalyze>
15/08/03 14:47:00 INFO parse.SemanticAnalyzer: Starting Semantic Analysis
15/08/03 14:47:00 INFO parse.SemanticAnalyzer: Creating table edges position=13
15/08/03 14:47:00 INFO ql.Driver: Semantic Analysis completed
```

查看创建结果

spark-sql>show tables;

```
15/08/03 14:48:11 INFO ql.Driver: <PERFLOG method=releaseLocks>
15/08/03 14:48:11 INFO ql.Driver: </PERFLOG method=releaseLocks start=1438584491220 end=1438584491220 duration=0>
edges
hivetable
sogouq1
sogouq2
tbdate
tbstock
tbstockdetail
testthrift
vertices
Time taken: 0.118 seconds
15/08/03 14:48:11 INFO CliDriver: Time taken: 0.118 seconds
```

3.4.2 实现代码

```
import org.apache.log4j.{Level, Logger}
import org.apache.spark.sql.hive.HiveContext
import org.apache.spark.{SparkContext, SparkConf}
import org.apache.spark.graphx._
import org.apache.spark.sql.catalyst.expressions.Row
```

```
object SQLGraphX {
  def main(args: Array[String]) {
    //屏蔽日志
    Logger.getLogger("org.apache.spark").setLevel(Level.WARN)
    Logger.getLogger("org.eclipse.jetty.server").setLevel(Level.OFF)

    //设置运行环境
    val sparkConf = new SparkConf().setAppName("PageRank")
    val sc = new SparkContext(sparkConf)
    val hiveContext = new HiveContext(sc)

    //使用 sparksql 查出每个店的销售数量和金额
    hiveContext.sql("use hive")
    val verticesdata = hiveContext.sql("select id, title from vertices")
```

```

val edgesdata = hiveContext.sql("select srcid,distid from edges")

//装载顶点和边
val vertices = verticesdata.map { case Row(id, title) => (id.toString.toLong,
title.toString)}
val edges = edgesdata.map { case Row(srcid, distid) => Edge(srcid.toString.toLong,
distid.toString.toLong, 0)}

//构建图
val graph = Graph(vertices, edges, "").persist()

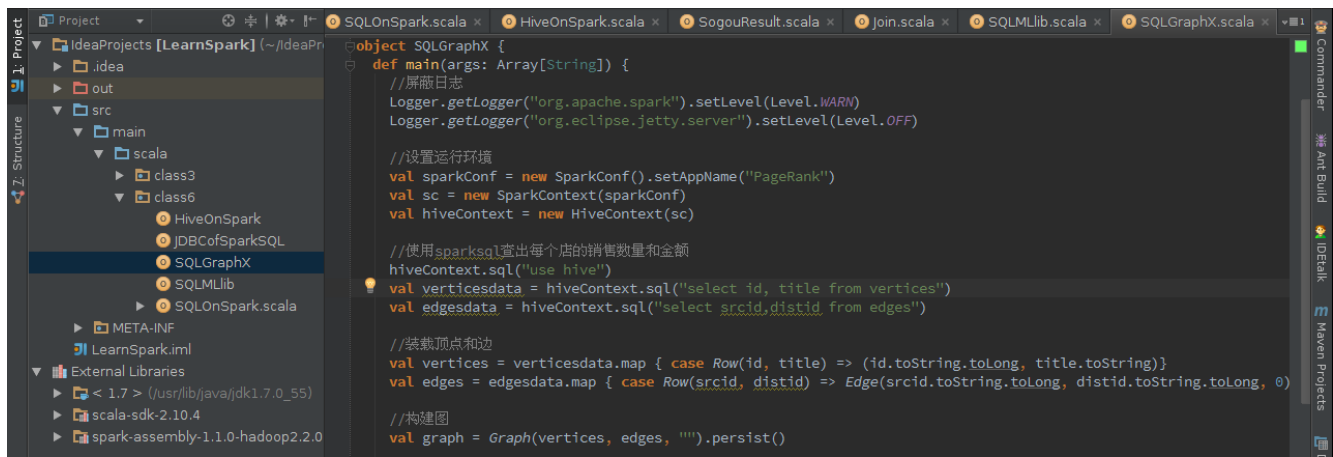
//pageRank 算法里面的时候使用了 cache() , 故前面 persist 的时候只能使用
MEMORY_ONLY
println("*****")
println("PageRank 计算 , 获取最有价值的数据")
println("*****")
val prGraph = graph.pageRank(0.001).cache()

val titleAndPrGraph = graph.outerJoinVertices(prGraph.vertices) {
  (v, title, rank) => (rank.getOrElse(0.0), title)
}

titleAndPrGraph.vertices.top(10) {
  Ordering.by((entry: (VertexId, (Double, String))) => entry._2._1)
}.foreach(t => println(t._2._2 + ": " + t._2._1))

sc.stop()
}
}

```



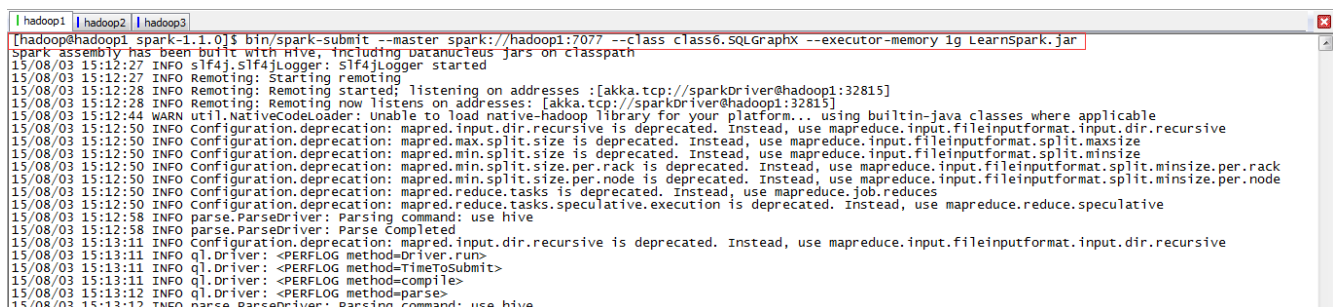
3.4.3 生成打包文件

按照 3.1.3SQL On Spark 方法进行打包

3.4.4 运行查看结果

通过如下命令调用打包中的 SQLOnSpark 方法：

```
cd /app/hadoop/spark-1.1.0
bin/spark-submit --master spark://hadoop1:7077 --class class6.SQLGraphX
--executor-memory 1g LearnSpark.jar
```



运行结果：

```
15/08/03 15:30:11 INFO parse.ParseDriver: Parsing command: select id, title from vertices
15/08/03 15:30:11 INFO parse.ParseDriver: Parse Completed
15/08/03 15:30:11 INFO parse.ParseDriver: Parsing command: select srcid,distid from edges
15/08/03 15:30:11 INFO parse.ParseDriver: Parse Completed
15/08/03 15:30:12 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
15/08/03 15:30:15 INFO mapred.FileInputFormat: Total input paths to process : 1
15/08/03 15:30:16 INFO mapred.FileInputFormat: Total input paths to process : 1
*****
PageRank??,?????????
*****
University of California, Berkeley: 1321.1117543121227
Berkeley, California: 664.8841977233989
Uc Berkeley: 162.5013274339786
Berkeley Software Distribution: 90.47860388486127
Lawrence Berkeley National Laboratory: 81.90404939642022
George Berkeley: 81.85226118458043
Busby Berkeley: 47.87199821801991
Berkeley Hills: 44.76406979519929
Xander Berkeley: 30.32407534728813
Berkeley County, South Carolina: 28.908336483710315
```

3.5 小结

在现实数据处理过程中，这种涉及多个系统处理的场景很多。通常各个系统之间的数据通过

磁盘落地再交给下一个处理系统进行处理。对于 Spark 来说，通过多个组件的配合，可以以流水线的方式来处理数据。从上面的代码可以看出，程序除了最后有磁盘落地外，都是在内存中计算的。避免了多个系统中交互数据的落地过程，提高了效率。这才是 spark 生态系统真正强大之处：One stack rule them all。另外 sparkSQL+sparkStreaming 可以架构当前非常热门的 Lambda 架构体系，为 CEP 提供解决方案。也正是如此强大，才吸引了广大开源爱好者的目光，促进了 Spark 生态的高速发展。