# 目标

定义DataFrame接口，与Hive交互，读取和写入数据。

当然，DataFrame也不仅限于与Hive进行数据交互，还可以与其他类型数据库交互；还可以将DataFrame类型的结果序列化和反序列化，使得不同项目的程序之间直接以文件的形式高效交换数据。

# 实现方法

## 从Hive中读取数据

### 过程

* 建立HiveContext hiveCtx
* 使用hiveCtx执行sql语句，读取hive数据到DataFrame lines
* 将lines转为RDD<Row>格式，即可进行后续的数据处理。

### 代码示例

|  |
| --- |
| SparkConf sparkConf = new SparkConf().setAppName("item\_based\_CF");  JavaSparkContext ctx = new JavaSparkContext(sparkConf);  //从hive读入数据  HiveContext hiveCtx = new HiveContext(ctx.sc());  hiveCtx.sql("use tmalldb");//指定数据库  DataFrame lines=hiveCtx.sql("select \* from user\_info");//读取数据，存入dataframe  //一次用户行为：将lines转为RDD格式，并通过判断用户行为类别保留浏览行为。读入的Hive数据转为RDD后为JavaRDD<Row>，具体Row操作见下面示例  JavaPairRDD<Long, Long> user\_behavior=lines.toJavaRDD().filter(new Function<Row, Boolean>() {  @Override  public Boolean call(Row row) throws Exception {  return Long.parseLong(row.getString(2))==1L;  }  }).mapToPair(new PairFunction<Row, Long, Long>() {  @Override  public Tuple2<Long, Long> call(Row row) throws Exception {  return new Tuple2<Long, Long>(Long.parseLong(row.getString(0)),Long.parseLong(row.getString(1)));  }  }); |

## 将数据存入Hive

### 过程

* 新建一个类，比如public static class outfile\_result implements Serializable，里面包含所需要储存的数据结构（注意：由于该类要用于RDD操作，必须加上static，由于要转换为DataFrame结构，因此必须加上可序列化标识Serializable）；
* 定义一个RDD为JavaRDD<outfile\_result> outfile，并将结果数据存入；
* 利用outfile生成DataFrame结构，并用该DataFrame注册一个spark虚拟出的临时表table1
* 用sql语句将table1的数据插入目标表prm14\_item\_based\_CF\_result

### 代码示例

新建一个类

|  |
| --- |
| public static class outfile\_result implements Serializable {  public Long item\_id;  public String item\_list;  outfile\_result(Long i1,String list) {  item\_id=i1;  item\_list=list;  }  } |

生成临时表table1，并将table1的数据插入目标表的分区中（注意：一般的商业化运营中，都是采用分区表的形式储存全量或者增量数据，这样可以在程序产出出错时及时数据回流；在将数据插入目标表前，需要先确认是否存在该表，如果不存在，则建立该表）

|  |
| --- |
| JavaRDD<outfile\_result> outfile=……  hiveCtx.sql("create external table if not exists prm14\_item\_based\_CF\_result(item\_id bigint,item\_list string) partitioned by (ds string)");  hiveCtx.createDataFrame(outfile, outfile\_result.class).registerTempTable("table1");  hiveCtx.sql("insert into prm14\_item\_based\_CF\_result partition(ds=from\_unixtime(unix\_timestamp(),'yyyy-MM-dd')) select item\_id,item\_list from table1"); |