第 11 章 Hive 实战

11.1 需求描述

统计 Youtube 视频网站的常规指标，各种 TopN 指标：
--统计视频观看数 Top10
--统计视频类别热度 Top10
--统计视频观看数 Top20 所属类别
--统计视频观看数 Top50 所关联视频的所属类别 Rank
--统计每个类别中的视频热度 Top10
--统计每个类别中视频流量 Top10
--统计上传视频最多的用户 Top10 以及他们上传的视频
--统计每个类别视频观看数 Top10

11.2 项目

11.2.1 数据结构

1）视频表

字段	备注	详细描述
video id	视频唯一 id	11 位字符串
uploader	视频上传者	上传视频的用户名 String
age	视频年龄	视频上传日期和 2007 年 2 月 15 日之间的整数天（Youtube 的独特设定）
category	视频类别	上传视频指定的视频分类
length	视频长度	整形数字标识的视频长度
views	观看次数	视频被浏览的次数
rate	视频评分	满分 5 分
ratings	流量	视频的流量，整型数字
conments	评论数	一个视频的整数评论数
related ids	相关视频 id	相关视频的 id，最多 20 个

2）用户表

字段	备注	字段类型
uploader	上传者用户名	string
videos	上传视频数	int
friends	朋友数量	int

11.2.2 ETL 原始数据

通过观察原始数据形式，可以发现，视频可以有多个所属分类，每个所属分类用&符号分割，且分割的两边有空格字符，同时相关视频也是可以有多个元素，多个相关视频又用“\t”进行分割。为了分析数据时方便对存在多个子元素的数据进行操作，我们首先进行数据重组

清洗操作。即：将所有的类别用“&”分割，同时去掉两边空格，多个相关视频 id 也使用“&”进行分割。

1）ETL 之 ETLUtil

```
public class ETLUtil {
	public static String oriString2ETLString(String ori){
		StringBuilder etlString = new StringBuilder();
		String[] splits = ori.split("\t");
		if(splits.length < 9) return null;
		splits[3] = splits[3].replace(" ", "");
		for(int i = 0; i < splits.length; i++){
			if(i < 9){
				if(i == splits.length - 1){
					etlString.append(splits[i]);
				}else{
					etlString.append(splits[i] + "\t");
				}
			}else{
				if(i == splits.length - 1){
					etlString.append(splits[i]);
				}else{
					etlString.append(splits[i] + "&");
				}
			}
		}

		return etlString.toString();
	}
}
```

2）ETL 之 Mapper

```
import java.io.IOException;

import org.apache.commons.lang.StringUtils;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import com.z.youtube.util.ETLUtil;

public class VideoETLMapper extends Mapper<Object, Text, NullWritable, Text>{
	Text text = new Text();

	@Override
```

```
    protected void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
        String etlString = ETLUtil.oriString2ETLString(value.toString());

        if(StringUtils.isBlank(etlString)) return;

        text.set(etlString);
        context.write(NullWritable.get(), text);
    }
}
```

3）ETL 之 Runner

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class VideoETLRunner implements Tool {
    private Configuration conf = null;

    @Override
    public void setConf(Configuration conf) {
        this.conf = conf;
    }

    @Override
    public Configuration getConf() {

        return this.conf;
    }

    @Override
    public int run(String[] args) throws Exception {
        conf = this.getConf();
        conf.set("inpath", args[0]);
        conf.set("outpath", args[1]);
```

```
        Job job = Job.getInstance(conf, "youtube-video-etl");

        job.setJarByClass(VideoETLRunner.class);

        job.setMapperClass(VideoETLMapper.class);
        job.setMapOutputKeyClass(NullWritable.class);
        job.setMapOutputValueClass(Text.class);
        job.setNumReduceTasks(0);

        this.initJobInputPath(job);
        this.initJobOutputPath(job);

        return job.waitForCompletion(true) ? 0 : 1;
    }

    private void initJobOutputPath(Job job) throws IOException {
        Configuration conf = job.getConfiguration();
        String outPathString = conf.get("outpath");

        FileSystem fs = FileSystem.get(conf);

        Path outPath = new Path(outPathString);
        if(fs.exists(outPath)){
            fs.delete(outPath, true);
        }

        FileOutputFormat.setOutputPath(job, outPath);

    }

    private void initJobInputPath(Job job) throws IOException {
        Configuration conf = job.getConfiguration();
        String inPathString = conf.get("inpath");

        FileSystem fs = FileSystem.get(conf);

        Path inPath = new Path(inPathString);
        if(fs.exists(inPath)){
            FileInputFormat.addInputPath(job, inPath);
        }else{
            throw new RuntimeException("HDFS 中该文件目录不存在：" + inPathString);
        }
    }
```

```
    public static void main(String[] args) {
        try {
            int resultCode = ToolRunner.run(new VideoETLRunner(), args);
            if(resultCode == 0){
                System.out.println("Success!");
            }else{
                System.out.println("Fail!");
            }
            System.exit(resultCode);
        } catch (Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

4）执行 ETL

```
$ bin/yarn jar ~/softwares/jars/youtube-0.0.1-SNAPSHOT.jar \
com.z.youtube.etl.ETLYoutubeVideosRunner \
/youtube/video/2008/0222 \
/youtube/output/video/2008/0222
```

11.3 准备工作

11.3.1 创建表

创建表：youtube_ori，youtube_user_ori，
创建表：youtube_orc，youtube_user_orc
youtube_ori：

```
create table youtube_ori(
    videoId string,
    uploader string,
    age int,
    category array<string>,
    length int,
    views int,
    rate float,
    ratings int,
    comments int,
    relatedId array<string>)
row format delimited
fields terminated by "\t"
collection items terminated by "&"
```

```
stored as textfile;
```

youtube_user_ori：

```
create table youtube_user_ori(
    uploader string,
    videos int,
    friends int)
clustered by (uploader) into 24 buckets
row format delimited
fields terminated by "\t"
stored as textfile;
```

然后把原始数据插入到 orc 表中

youtube_orc：

```
create table youtube_orc(
    videoId string,
    uploader string,
    age int,
    category array<string>,
    length int,
    views int,
    rate float,
    ratings int,
    comments int,
    relatedId array<string>)
clustered by (uploader) into 8 buckets
row format delimited fields terminated by "\t"
collection items terminated by "&"
stored as orc;
```

youtube_user_orc：

```
create table youtube_user_orc(
    uploader string,
    videos int,
    friends int)
clustered by (uploader) into 24 buckets
row format delimited
fields terminated by "\t"
stored as orc;
```

11.3.2 导入 ETL 后的数据

youtube_ori：

```
load data inpath "/youtube/output/video/2008/0222" into table youtube_ori;
```

youtube_user_ori：

```
load data inpath "/youtube/user/2008/0903" into table youtube_user_ori;
```

11.3.3 向 ORC 表插入数据

youtube_orc：

```
insert into table youtube_orc select * from youtube_ori;
```

youtube_user_orc：

```
insert into table youtube_user_orc select * from youtube_user_ori;
```

11.4 业务分析

11.4.1 统计视频观看数 Top10

思路：使用 order by 按照 views 字段做一个全局排序即可，同时我们设置只显示前 10 条。
最终代码：

```
select
    videoId,
    uploader,
    age,
    category,
    length,
    views,
    rate,
    ratings,
    comments
from
    youtube_orc
order by
    views
desc limit
    10;
```

11.4.2 统计视频类别热度 Top10

思路：
1) 即统计每个类别有多少个视频，显示出包含视频最多的前 10 个类别。
2) 我们需要按照类别 group by 聚合，然后 count 组内的 videoId 个数即可。
3) 因为当前表结构为：一个视频对应一个或多个类别。所以如果要 group by 类别，需要先将类别进行列转行(展开)，然后再进行 count 即可。
4) 最后按照热度排序，显示前 10 条。
最终代码：

```
select
    category_name as category,
    count(t1.videoId) as hot
from (
    select
        videoId,
```

```
        category_name
    from
        youtube_orc lateral view explode(category) t_catetory as category_name) t1
group by
    t1.category_name
order by
    hot
desc limit
    10;
```

11.4.3 统计出视频观看数最高的 20 个视频的所属类别以及类别包含 Top20 视频的个数

思路：

1) 先找到观看数最高的 20 个视频所属条目的所有信息，降序排列
2) 把这 20 条信息中的 category 分裂出来(列转行)
3) 最后查询视频分类名称和该分类下有多少个 Top20 的视频

最终代码：

```
select
    category_name as category,
    count(t2.videoId) as hot_with_views
from (
    select
        videoId,
        category_name
    from (
        select
            *
        from
            youtube_orc
        order by
            views
        desc limit
            20) t1 lateral view explode(category) t_catetory as category_name) t2
group by
    category_name
order by
    hot_with_views
desc;
```

11.4.4 统计视频观看数 Top50 所关联视频的所属类别 Rank

思路：

1) 查询出观看数最多的前 50 个视频的所有信息(当然包含了每个视频对应的关联视频)，记为临时表 t1

```
t1:观看数前 50 的视频
select
	*
from
	youtube_orc
order by
	views
desc limit
	50;
```

2) 将找到的 50 条视频信息的相关视频 relatedId 列转行，记为临时表 t2

```
t2:将相关视频的 id 进行列转行操作
select
	explode(relatedId) as videoId
from
	t1;
```

3) 将相关视频的 id 和 youtube_orc 表进行 inner join 操作

```
t5:得到两列数据，一列是 category，一列是之前查询出来的相关视频 id
(select
	distinct(t2.videoId),
	t3.category
from
	t2
inner join
	youtube_orc t3 on t2.videoId = t3.videoId) t4 lateral view explode(category) t_catetory as
category_name;
```

4) 按照视频类别进行分组，统计每组视频个数，然后排行

最终代码：

```
select
	category_name as category,
	count(t5.videoId) as hot
from (
	select
		videoId,
		category_name
	from (
		select
			distinct(t2.videoId),
			t3.category
		from (
			select
				explode(relatedId) as videoId
			from (
				select
					*
```

```
                    from
                        youtube_orc
                    order by
                        views
                    desc limit
                        50) t1) t2
            inner join
                youtube_orc t3 on t2.videoId = t3.videoId) t4 lateral view explode(category)
t_catetory as category_name) t5
group by
    category_name
order by
    hot
desc;
```

11.4.5 统计每个类别中的视频热度 Top10，以 Music 为例

思路：

1) 要想统计 Music 类别中的视频热度 Top10，需要先找到 Music 类别，那么就需要将 category 展开，所以可以创建一张表用于存放 categoryId 展开的数据。

2) 向 category 展开的表中插入数据。

3) 统计对应类别（Music）中的视频热度。

最终代码：

创建表类别表：

```
create table youtube_category(
    videoId string,
    uploader string,
    age int,
    categoryId string,
    length int,
    views int,
    rate float,
    ratings int,
    comments int,
    relatedId array<string>)
row format delimited
fields terminated by "\t"
collection items terminated by "&"
stored as orc;
```

向类别表中插入数据：

```
insert into table youtube_category
    select
        videoId,
        uploader,
```

```
        age,
        categoryId,
        length,
        views,
        rate,
        ratings,
        comments,
        relatedId
    from
        youtube_orc lateral view explode(category) catetory as categoryId;
```

统计 Music 类别的 Top10（也可以统计其他）

```
select
    videoId,
    views
from
    youtube_category
where
    categoryId = "Music"
order by
    views
desc limit
    10;
```

11.4.6 统计每个类别中视频流量 Top10，以 Music 为例

思路：

1) 创建视频类别展开表（categoryId 列转行后的表）
2) 按照 ratings 排序即可

最终代码：

```
select
    videoId,
    views,
    ratings
from
    youtube_category
where
    categoryId = "Music"
order by
    ratings
desc limit
    10;
```

11.4.7 统计上传视频最多的用户 Top10 以及他们上传的观看次数在前 20 的视频

思路：

1) 先找到上传视频最多的 10 个用户的用户信息

```
select
	*
from
	youtube_user_orc
order by
	videos
desc limit
	10;
```

2) 通过 uploader 字段与 youtube_orc 表进行 join，得到的信息按照 views 观看次数进行排序即可。

最终代码：

```
select
	t2.videoId,
	t2.views,
	t2.ratings,
	t1.videos,
	t1.friends
from (
	select
		*
	from
		youtube_user_orc
	order by
		videos desc
	limit
		10) t1
join
	youtube_orc t2
on
	t1.uploader = t2.uploader
order by
	views desc
limit
	20;
```

11.4.8 统计每个类别视频观看数 Top10

思路：

1) 先得到 categoryId 展开的表数据

2) 子查询按照 categoryId 进行分区，然后分区内排序，并生成递增数字，该递增数字这一列起名为 rank 列

3) 通过子查询产生的临时表，查询 rank 值小于等于 10 的数据行即可。

最终代码：

```
select
	t1.*
from (
	select
		videoId,
		categoryId,
		views,
		row_number() over(partition by categoryId order by views desc) rank from
youtube_category) t1
where
	rank <= 10;
```