**数据收集与数据分析工具指导说明书**

目录

[任务1. 数据收集阶段 1](#_Toc44270649)

[任务描述 1](#_Toc44270650)

[任务指导 1](#_Toc44270651)

[任务实现 2](#_Toc44270652)

[任务2. 数据分析工具编写 22](#_Toc44270653)

[任务描述 22](#_Toc44270654)

[任务指导 22](#_Toc44270655)

[任务实现 23](#_Toc44270656)

# 数据收集

## 任务描述

**知识点**：数据收集整理、导入数据到MySql、使用爬虫抓取网络数据

**重点**：数据收集整理、导入数据到MySql

**难点**：使用爬虫抓取网络数据

## 任务指导

1. 给出项目所需的数据种类和格式
2. 要求学生通过爬虫等手段自行获取数据，图书信息可以豆瓣、京东、china-pub等网站获取
3. 为防止学生无法自行获取数据，老师需提供已经准备好的静态数据（.csv格式的数据）

## 任务实现

1. **项目所需的数据种类和格式介绍**

* 业务数据：

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 所属表 | 所属页面 | 说明 |
| 登录 | user | 登录页面 |  |
| 图书列表展示 | Book\_info、book\_category | 主页面 | 分页 |
| 图书详情显示 | Book\_info、book\_desc | 图书详情页面 |  |
| 图书打分（0~10分） | Bx\_book\_ratings | 图书详情页面、购物车页面 |  |
| 生成订单和订单明细 | orders、order\_detail | 购物车页面 |  |
| 模拟支付 | orders、order\_shipping | 购物车页面、支付页面 |  |

* 日志数据：

|  |  |  |
| --- | --- | --- |
| 数据 | 说明 |  |
| 网站访问日志 | 用于用户点击流日志分析 | 1. coolshell\_20140212.log（演示用） 2. apache\_simple.log(样例日志数据) 3. 本项目Web端产生的Apache访问日志 |

1. **数据库设计**

* 数据库说明：

|  |  |  |
| --- | --- | --- |
| 数据库名称 | books | |
| 表名 | 功能 | 备注 |
| users | 用户表 |  |
| book\_info | 图书信息表 |  |
| bx\_book\_ratings | 用户图书评分表 |  |
| book\_category | 图书类别表 |  |
| book\_desc | 图书描述表 |  |
| orders | 订单表 |  |
| order\_detail | 订单详情表 |  |
| order\_shipping | 订单邮寄表 |  |
| privilege | 权限表 |  |
| reply | 回复表 |  |
| role | 角色表 |  |
| role\_privilege | 权限角色对应表 |  |
| store | 店铺表 |  |
| user\_role | 用户角色表 |  |
| comment | 评论表 |  |
| global\_parameter | 全局参数表 |  |

* 表说明：
* 表users：用户表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| userid | bigint |  | 主键、非空 | 用户id |  |
| username | varchar | 100 | 非空 | 用户名 |  |
| nickname | varchar | 32 |  | 昵称 |  |
| password | varchar | 100 | 非空 | 登录密码 |  |
| gender | char | 2 |  | 性别 |  |
| email | varchar | 20 |  | 邮件地址 |  |
| Phone | varchar | 20 |  | 手机号 |  |
| zip\_code | varchar | 20 |  | 邮编 |  |
| location | varchar | 20 |  | 地区 |  |
| detail\_address | varchar | 100 | 无 | 详细地址 |  |
| country | varchar | 100 | 无 | 国家 |  |
| age | int |  | 无 | 年龄 |  |
| sex | varchar | 10 | 无 | 性别 |  |
| identity | varchar | 20 |  | 身份 |  |
| Active | varchar | 1 |  | 是否有效 |  |
| Code | varchar | 100 |  |  |  |
| updated | datetime |  |  | 更新时间 |  |
| Created | datetime |  |  | 创建时间 |  |

* 表book\_info：图书信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| bookid | bigint |  | 主键、非空 | 图书id |  |
| book\_category\_id | Int |  |  | 图书目录ID | 关联到book\_category |
| store\_id | Int |  |  | 店铺id |  |
| Name | varchar | 100 |  | 图书名称 |  |
| outline | varchar | 300 | 非空 | 概述 |  |
| detail | varchar | 100 |  | 图书详情 |  |
| press | varchar | 100 |  | 出版社 |  |
| author | varchar | 300 | 无 | 作者 |  |
| Publish\_date | varchar | 300 | 无 | 出版日期 |  |
| size | varchar | 10 | 无 | 图书大小 |  |
| version | varchar | 20 | 无 | 版本号 |  |
| translator | varchar | 20 | 无 | 译者 |  |
| isbn | varchar | 100 | 无 | Isbn号 |  |
| price | bigdecimal |  | 非空 | 价格 |  |
| pages | Int |  | 无 | 页数 |  |
| catalog | varchar |  | 无 | 目录 |  |
| market\_price | bigdecimal |  | 无 | 市场价 |  |
| member\_price | bigdecimal |  | 无 | 会员价 |  |
| deal\_mount | Int |  | 无 | 交易数量 |  |
| look\_mount | Int |  |  | 查看数量 |  |
| discount | bigdecimal |  |  | 折扣 |  |
| image\_url | varchar | 200 |  | 图片链接 |  |
| store\_mount | Int |  |  | 库存数量 |  |
| store\_time | Datetime |  |  | 库存日期 |  |
| pack\_style | varchar |  |  | 包装类型 |  |
| is\_shelf | Char |  |  | 是否上架 |  |
| cname | varchar | 100 |  |  |  |
| description | varchar | 100 | 无 | 描述 |  |
| cata | varchar | 100 |  | 目录 |  |
| content | varchar | 200 |  | 内容 |  |

* 表book\_category：图书类别表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| cate\_id | int |  | 主键、非空 | 类别id |  |
| parent\_id | int |  |  | 父类别ID |  |
| name | varchar | 100 |  | 名称 |  |
| Status | Char |  |  | 状态 |  |
| sort\_order | Int |  |  | 排序号 |  |
| is\_parent | Tinyint |  |  | 是否父类 |  |
| created | Datetime |  |  | 创建日期 |  |
| Updated | Datetime |  |  | 更新日期 |  |

* 表book\_desc：图书描述表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| book\_id | bigint |  | 非空 | 图书id |  |
| book\_desc | bigint |  |  | 图书描述 |  |
| created | Datetime |  |  | 创建日期 |  |
| Updated | Datetime |  |  | 更新日期 |  |

* 表bx\_book\_ratings：用户图书评分表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| userid | bigint |  | 非空 | 用户id |  |
| bookid | bigint |  | 非空 | 图书id |  |
| score | int |  | 非空 | 图书评分 |  |

* 表orders：订单表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| orderid | bigint |  | 主键、非空 | 订单id |  |
| userid | bigint |  | 非空 | 用户id |  |
| payment | Varchar |  |  | 实付金额 |  |
| payment\_type | Int |  |  | 付款类型 |  |
| post\_fee | Varchar |  |  | 邮费 |  |
| create\_time | Datetime |  |  | 创建时间 |  |
| update\_time | Datetime |  |  | 更新时间 |  |
| order\_mount | Int |  |  | 订单数量 |  |
| payment\_time | Datetime |  |  | 付款时间 |  |
| end\_time | Datetime |  |  | 交易完成时间 |  |
| close\_time | Datetime |  |  | 交易关闭时间 |  |
| shipping\_name | Varchar |  |  | 物流名称 |  |
| shipping\_code | Varchar |  |  | 物流单号 |  |
| buyer\_message | Varchar |  |  | 买家信息 |  |
| buyer\_rate | Int |  |  | 买家评级 |  |
| status | int |  | 非空 | 订单状态 | 0：已下单；  1：未付款；  2：已结算 |

* 表order\_detail：订单明细表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| order\_detail\_id | bigint |  |  | 订单详情id |  |
| order\_number | Varchar |  |  | 订单号 |  |
| book\_id | bigint |  |  | 图书id |  |
| order\_id | bigint |  |  | 订单id |  |
| store\_id | int |  |  | 店铺id |  |
| mount | int |  |  | 数量 |  |
| unit\_price | bigDecimal |  |  | 单价 |  |
| total\_price | bigDecimal |  |  | 总价 |  |
| post\_status | Varchar |  |  | 支付，发货状态 |  |
| delivery\_time | Datetime |  |  | 修订时间 |  |
| receive\_status | Varchar |  |  | 收货状态 |  |
| image\_url | Varchar |  |  | 图片地址 |  |
| book\_name | Varchar |  |  | 书名 |  |
| score | int |  |  | 评分 |  |

* 表 Order\_shipping

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| order\_id | bigint |  | 非空 | 订单id |  |
| receiver\_name | Varchar |  | 非空 | 收件人姓名 |  |
| receiver\_phone | Varchar |  |  | 收件人座机 |  |
| receiver\_mobile | Varchar |  |  | 收件人电话 |  |
| receiver\_state | Varchar |  |  | 省份 |  |
| receiver\_city | Varchar |  |  | 城市 |  |
| receiver\_district | Varchar |  |  | 区/县 |  |
| receiver\_address | Varchar |  |  | 收货地址 |  |
| receiver\_zip | Varchar |  |  | 邮政编码 |  |
| Created | Datetime |  |  | 创建时间 |  |
| Update | Datetime |  |  | 更新时间 |  |

* 表privilege 权限表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| priv\_id | bigint |  | 非空 | 权限ID |  |
| name | Varchar |  |  | 权限名称 |  |
| Code | Varchar |  |  | 编码 |  |
| url | Varchar |  |  | 访问路径 |  |
| parent\_id | Int |  |  | 父id |  |
| is\_parent | Int |  |  | 是否是父节点 |  |
| Created | Datetime |  |  | 创建日期 |  |
| Update | Datetime |  |  | 更新日期 |  |

* 表role 角色表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| role\_id | bigint |  | 非空 | 角色ID |  |
| name | Varchar |  |  | 角色名称 |  |
| Code | Varchar |  |  | 编码 |  |
| Description | Varchar |  |  | 描述 |  |
| Created | Datetime |  |  | 创建时间 |  |
| Update | Datetime |  |  | 更新时间 |  |

* 表Role\_privilege 角色-权限表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| role\_id | bigint |  | 非空 | 角色ID |  |
| privilege\_id | Int |  |  | 权限ID |  |
| Created | Datetime |  |  | 创建时间 |  |
| Update | Datetime |  |  | 更新时间 |  |

* 表user\_role 用户-角色表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| user\_id | int |  |  | 用户ID |  |
| role\_id | Int |  |  | 角色ID |  |
| Created | Datetime |  |  | 创建时间 |  |
| Update | Datetime |  |  | 更新时间 |  |

* 表store ：店铺表 储存店铺的基本信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 大小 | 约束 | 说明 | 备注 |
| store\_id | 16 | 主键 | 店铺ID |  |
| store\_manager\_id | 16 |  | 管理ID |  |
| store\_phone\_num | 20 |  | 店铺电话 |  |
| store\_telephone | 20 |  | 店铺固话 |  |
| store\_name | 64 |  | 店铺名称 |  |
| created |  |  | 创建时间 |  |
| updated |  |  | 更新时间 |  |
| store\_position | 128 |  | 店铺地址 |  |

* 表reply 回复表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| reply\_id | int | 16 | 主键 | 回复ID |  |
| type | varchar | 20 |  | 回复类型 |  |
| title | varchar | 20 |  | 标题 |  |
| content | varchar | 100 |  | 内容 |  |
| user\_id | int | 16 |  | 用户id |  |
| username | varchar | 20 |  | 用户名 |  |
| date | datetime |  |  | 时间 |  |

* 表comment 评论表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 | 备注 |
| comment\_id | int | 16 | 主键 | 评论ID |  |
| book\_id | int | 16 |  | 书本id |  |
| user\_id | int | 16 |  | 用户id |  |
| username | varchar | 100 |  | 用户名 |  |
| date | datetime |  |  | 评论时间 |  |
| content | varchar | 1000 |  | 评论内容 |  |
| flag | varchar | 100 |  | 标记 |  |

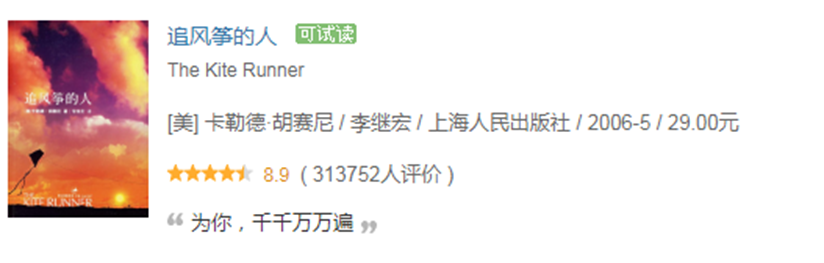
* 表bx\_clickstream\_log：点击流日志表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 大小 | 约束 | 说明 |
| ipaddress | varchar | 200 | 无 | IP地址 |
| uniqueid | varchar | 200 | 无 | 用户唯一编号 |
| url | varchar | 200 | 无 | 用户访问的链接 |
| sessionid | varchar | 200 | 无 | Session的唯一编号 |
| sessiontimes | varchar | 200 | 无 | Session的次数 |
| areaaddress | varchar | 200 | 无 | 发生点击行为的地区 |
| localaddress | varchar | 200 | 无 | 发生点击行为的详细地址 |
| browsertype | varchar | 200 | 无 | 用户浏览器信息 |
| operationsys | varchar | 200 | 无 | 用户操作系统信息 |
| referurl | varchar | 200 | 无 | 上一个浏览的网页 |
| receivetime | varchar | 200 | 无 | 日志接收服务端接收时间 |
| userid | varchar | 200 | 无 | 用户的账号 |
| csvp | varchar | 200 | 无 | 该点击行为在其Session中的顺序 |

1. **使用scrapy爬取豆瓣图书TOP250**

* **爬取的内容为：**图书id、书名、作者、出版日期、出版社、小图、中图、大图、类别id、类别名称、单价、短评。
* 爬取豆瓣网图书，仅作为学习使用，不得用于任何商业目的。（爬取过程可能会被封IP）
* 采集工作有两层连接，第一层从[https://book.douban.com/top250?start=0](https://link.jianshu.com?t=https%3A%2F%2Fbook.douban.com%2Ftop250%3Fstart%3D0" \t "_blank) 以25的位数递增，第二层为每本图书的详情页面。
* 采集方法可以使用：xpath方法或css方法。
  + - Xpath方法

使用浏览器开发者工具，定位元素后，查找xpath路径。



例如：

每本书的路径： "//[@id="content"]/div/div[1]/div/table"

令 book = response.xpath('//[@id="content"]/div/div[1]/div/table')则后面的可以表示为：

• 标题路径：book.xpath("./tr/td[2]/div[1]/a/@title")

• 评价人数路径：book.xpath("./tr/td[2]/p[2]/span/text()")

* + - css方法

使用浏览器开发者工具，定位元素后观察下方栏的css标签节点。



例如：每本书的路径：'tr.item'，令 book=response.css('tr.item')，则可以表示为：

• 标题路径：book.css('div.pl2 a::text')

• 图片路径：book.css('a.nbg img').xpath('@src')

• 其它同理

* 打开前面创建的爬虫项目，然后编写代码
  + - 编写items

|  |
| --- |
| class ScrapyBookItem(scrapy.Item):  # 图书id  isbn = scrapy.Field()  # 书名  title = scrapy.Field()  # 作者  author = scrapy.Field()  # 出版日期  pub\_date = scrapy.Field()  # 出版社  publisher = scrapy.Field()  # 小图  s\_img = scrapy.Field()  # 中图  m\_img = scrapy.Field()  # 大图  b\_img = scrapy.Field()  # 类别id  category\_id = scrapy.Field()  # 类别名称  category = scrapy.Field()  # 单价  price = scrapy.Field()  # 图书一句话描述  scrible = scrapy.Field()  # 图书详情页的url  sub\_url = scrapy.Field() |

* + - 编写基于xpath的spider

|  |
| --- |
| class SpiderForXPath(scrapy.Spider):  name = 'spider\_xpath\_douban'  def start\_requests(self):  for a in range(10):  url = 'https://book.douban.com/top250?start={}'.format(a \* 25)  yield scrapy.Request(url=url, callback=self.parse)  def parse(self, response):  items = []  for book in response.xpath('//\*[@id="content"]/div/div[1]/div/table'):  item = ScrapyBookItem()  title1 = book.xpath("./tr/td[2]/div[1]/a/@title").extract\_first().replace('\n', '').strip()  title2 = "无" if book.xpath("./tr/td[2]/div[1]/span/text()").extract\_first() == None else book.xpath(  "./tr/td[2]/div[1]/span/text()").extract\_first().replace('\n', '').strip()  item['title'] = title1 + "(" + title2 + ")"  item['s\_img'] = book.xpath("./tr/td[1]/a/img/@src").extract\_first().replace('\n', '').strip()  item['scrible'] = "无" if book.xpath("./tr/td[2]/p[2]/span/text()").extract\_first() == None else book.xpath(  "./tr/td[2]/p[2]/span/text()").extract\_first().replace('\n', '').strip()  sub\_url = book.xpath("./tr/td[2]/div/a/@href").extract\_first().replace('\n', '').strip()  items.append(item)  # meta={"item":item} 传递item引用SinaItem对象  yield scrapy.Request(url=sub\_url, callback=self.parse\_second, meta={"item": item})  def parse\_second(self, response):  item = response.meta["item"]  item["category\_id"] = ""  item["category"] = ""  # book = response.xpath('//div[@class="indent"]/div').extract\_first()  # item["author"] = book.xpath("./div[1]/a[1]/text()").extract\_first().replace('\n', '').strip()  # item["publisher"] = book.xpath("./div[1]/a/@href").extract\_first().replace('\n', '').strip()  # item["pub\_date"] = book.xpath("./div[1]/a/@href").extract\_first().replace('\n', '').strip()  # item["price"] = book.xpath("./div[1]/a/@href").extract\_first().replace('\n', '').strip()  # item["m\_img"] = book.xpath("./div[1]/a/@href").extract\_first().replace('\n', '').strip()  # item["b\_img"] = book.xpath("./div[1]/a/@href").extract\_first().replace('\n', '').strip()  # item["isbn"] = book.xpath("./div[2]/a[1]/text()").extract\_first().replace('\n', '').strip()  yield item |

* + - 学生需独立完成基于css的spider

|  |
| --- |
| class SpiderForCSS(scrapy.Spider):  name = 'spider\_css\_douban'  def start\_requests(self):  for a in range(10):  url = 'https://book.douban.com/top250?start={}'.format(a \* 25)  yield scrapy.Request(url=url, callback=self.parse)  def parse(self, response):  items = []  for book in response.css('tr.item'):  item = ScrapyBookItem()  item['title'] = book.css('div.pl2 a::text').extract\_first().replace('\n', '').strip()  item['score'] = book.css('div.star.clearfix span.rating\_nums::text').extract\_first().replace('\n',  '').strip()  item['scrible'] = book.css('p.quote span.inq::text').extract\_first().replace('\n', '').strip()  item['num'] = book.css('div.star.clearfix span.pl::text').extract\_first().strip("(").strip(")").replace(  '\n', '').strip()  item['img'] = book.css('a.nbg img').xpath('@src').extract\_first().replace('\n', '').strip()  items.append(item)  print(items)  return items |

* + - 编写pipelines

|  |
| --- |
| import codecs  import json  class ScrapyBookPipeline(object):  def \_\_init\_\_(self):  self.file = codecs.open('data.json', 'w', encoding='utf-8')  def process\_item(self, item, spider):  line = json.dumps(dict(item), ensure\_ascii=False) + "\n"  self.file.write(line)  return item  def spider\_closed(self, spider):  self.file.close() |

* + - 配置settings，在文件中修改user-agent，伪装成浏览器。修改内容如下：

|  |
| --- |
| USER\_AGENT = 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36'  ITEM\_PIPELINES = {  'douban.pipelines.DoubanPipeline': 300,  } |

* + - 编写start.py

|  |
| --- |
| # -\*- coding: utf-8 -\*-  from scrapy import cmdline  # 执行基于xpath的spider  cmdline.execute('scrapy crawl spider\_xpath\_douban'.split())  # 执行基于css的spider  # cmdline.execute('scrapy crawl spider\_xpath\_douban'.split()) |

1. **将业务数据导入到MySql数据库**

* 在MySql中创建数据库和表
* 使用Navicat将数据导入到MySql中

1. **对日志数据进行清理**

* **日志格式**

本项目所需的访问日志的格式为标准的Apache访问日志：

|  |  |  |
| --- | --- | --- |
| 1 | 10.178.123.55 | 远端主机 |
| 2 | - | 远端登录名 |
| 3 | - | 远程用户名 |
| 4 | [11/Dec/2018:10:00:32 +0800] | 服务器接收时间 |
| 5 | "GET /\_\_utm.gif HTTP/1.1" | 请求的第一行(请求的路径) |
| 6 | 200 | 最后请求的状态 |
| 7 | 35 | 以CLF格式显示的除HTTP头以外传送的字节数 |
| 8 | "http://easternmiles.ceair.com/flight/index.html" | 上一个访问页面 |
| 9 | "Mozilla/5.0 (Windows NT 6.1;WOW64) AppleWebkit/  537.36 (KHTML,like Gecko) Chrome/31.0.1650.63 Safari/537.36" | 用户浏览器信息 |
| 10 | "BIGipServermu\_122.119.122.14=192575345.20480.000;Webtrends=  120.196.145.58.1386724976245806; " | Cookie信息 |
| 11 | 1482 | 接收的字节数，包括请求头的数据，且不能为零 |
| 12 | 352 | 发送的字节数，包括请求头的数据，且不能为零 |
| 13 | - | %{X-Forwarded-For}i |
| 14 | easternmiles.ceair.com | 访问主机地址（域名） |
| 15 | 749 | 服务器处理本请求所用的时间，以微秒为单位 |

* **获取日志数据**
* 数据来源是由本项目的Web端程序运行所产生的访问日志。
* 修改Tomcat日志格式，使Tomcat服务器产生的日志格式符合上面表格中的格式要求：

如果使用的是纯Apache服务器，那么访问日志在Apache根目录的conf文件夹内的httpd.conf文件中配置。如果使用的是Apache tomcat服务器，那么访问日志在Apache根目录的conf文件夹内的server.xml文件中配置。

pattern:属性值由字符串常量和pattern标识符加上前缀"%"组合而成。pattern标识符加上前缀"%"，用来代替当前请求/响应中的对应的变量值。目前支持如下的pattern：

%a - 远端IP地址

%A - 本地IP地址

%b - 发送的字节数，不包括HTTP头，如果为零即没有发送字节，使用"－"

%B - 发送的字节数，不包括HTTP头

%h - 远端主机名(如果resolveHost=false，远端的IP地址）

%H - 请求协议

%l - 从identd返回的远端逻辑用户名（总是返回 '-'）

%m - 请求的方法（GET，POST，等）

%p - 收到请求的本地端口号

%q - 查询字符串(如果存在，以 '?'开始)

%r - 请求的第一行，包含了请求的方法和URI

%s - 响应的状态码

%S - 用户的session ID

%t - 日志和时间，使用通常的Log格式

%u - 认证以后的远端用户（如果存在的话，否则为'-'）

%U - 请求的URI路径

%v - 本地服务器的名称

%D - 处理请求的时间，以毫秒为单位

%T - 处理请求的时间，以秒为单位

另外，还可以将request请求的查询参数、session会话变量值、cookie值或HTTP请求/响应头内容的变量值等内容写入到日志文件。

它仿照了apache的语法：

％{XXX}i xxx代表传入的头(HTTP Request)

％{XXX}o xxx代表传出的响应头(Http Resonse)

％{XXX}c xxx代表特定的Cookie名

％{XXX}r xxx代表ServletRequest属性名

％{XXX}s xxx代表HttpSession中的属性名

注意：每次更改请重启tomcat服务器

例如当pattern设置为：

pattern="%h %l %u %t "%r" %s %b"

对应的实际的一条访问记录如下：

216.35.116.91 - - [19/Aug/2000:14:47:37 -0400] "GET / HTTP/1.0" 200 654

* 将Apache Tomcat访问日志的最终格式设置为：

pattern="%h %l %u %t &quot;%r&quot; %s %b &quot;%{Referer}i&quot; &quot;%{User-Agent}i&quot; &quot;%{uuid}c;%{userId}c;%{st}c;&quot; %I %O %{X-Forwarded-For}i %v %D"

%I 接收的字节数，包括请求头的数据，并且不能为零。%O 发送的字节数，包括请求头的数据，并且不能为零。要使用这两个指令你必须启用mod\_logio模块。

这样就可以收集项目所需格式的数据了。如果使用IDEA直接部署的项目，日志文件保存在：C:\Users\gaogu\.IntelliJIdea2019.1\system\tomcat\Tomcat\_8\_5\_32\_bx\_web\logs

* 在没有获取大量的访问日志前，在这里我们提供一些样例日志数据（**apache\_simple.log**），为后面的点击流日志分析模块的开发提供数据支撑。
* **演示一个日志处理的示例代码，供学生参考使用**

示例数据来源于酷壳（CoolShell.cn）网站上的HTTP访问日志数据。这份数据是2014/2/12一天的访问日志（附件coolshell\_20140212.log列分隔符为"`"）。

**数据格式：**

$remote\_addr $remote\_user $time\_local $request $status $body\_bytes\_sent $http\_referer $http\_user\_agent

|  |  |  |
| --- | --- | --- |
| **序号** | **字段名称** | **字段说明** |
| 1 | $remote\_addr | 发送请求的客户端IP地址。 |
| 2 | $remote\_user | 客户端登录名 |
| 3 | $time\_local | 服务器本地时间 |
| 4 | $request | 请求，包括HTTP请求类型+请求URL+HTTP协议版本号 |
| 5 | $status | 服务端返回状态码 |
| 6 | $body\_bytes\_sent | 返回给客户端的字节数（不含header） |
| 7 | $http\_referer | 该请求的来源URL |
| 8 | $http\_user\_agent | 发送请求的客户端信息，如使用的浏览器等 |

**数据示例：**

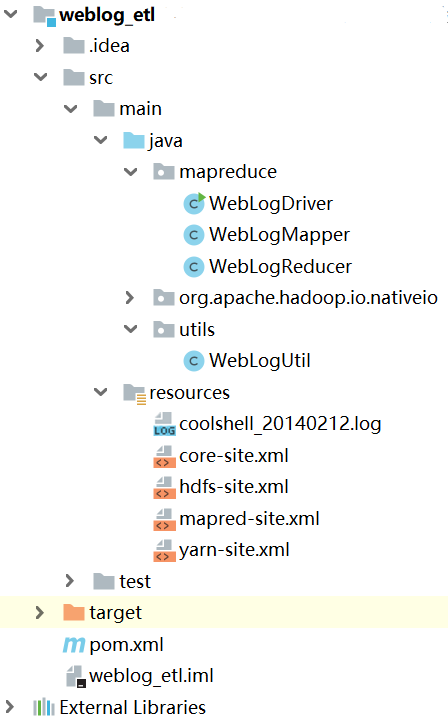
153.65.16.10``2014-02-12 05:45:09`GET //wp-content/plugins/google-syntax-highlighter/Scripts/shBrushDelphi.js HTTP/1.1`200`989`http://coolshell.cn/ articles/968.html`Mozilla/5.0 (X11; Linux x86\_64; rv:2.0b12) Gecko/20110222 Firefox/4.0b12

* **方案一：使用python脚本parse.py对源log文件清洗，将数据清洗成如下格式：**

192.190.75.20||2014-02-12 03:08:28|GET //wp-includes/js/jquery/jquery-migrate.min.js?ver=1.2.1 HTTP/1.1|304|0||Mozilla/4.0 (compatible;)

|  |
| --- |
| # parse.py  #!/usr/bin/python  import sys  import re  import time  COL\_DELIMITER = '|';  def convertTime(str):  # convert: 12/Feb/2014:03:17:50 +0800 to: 2014-02-12 03:17:50  if str:  return time.strftime('%Y-%m-%d %H:%M:%S',  time.strptime(str[:-6], '%d/%b/%Y:%H:%M:%S'))  def parseLog(inFile, outFile, dirtyFile):  file = open(inFile)  output = open(outFile, "w")  dirty = open(dirtyFile, "w")  items = [  r'(?P<ip>\S+)', # ip  r'\S+', # indent -, not used  r'(?P<user>\S+)', # user  r'\[(?P<time>.+)\]', # time  r'"(?P<request>.\*)"', # request  r'(?P<status>[0-9]+)', # status  r'(?P<size>[0-9-]+)', # size  r'"(?P<referer>.\*)"', # referer  r'"(?P<agent>.\*)"', # user agent  r'(.\*)', # unknown info  ]  pattern = re.compile(r'\s+'.join(items)+r'\s\*\Z')  for line in file:  m = pattern.match(line)  if not m:  dirty.write(line)  else:  dict = m.groupdict()  dict["time"] = convertTime(dict["time"])  if dict["size"] == "-":  dict["size"] = "0"  for key in dict:  if dict[key]=="-":  dict[key] = ""  #ip,user,time,request,status,size,referer  output.write("%s\n" % (COL\_DELIMITER.join(  (dict["ip"],dict["user"],dict["time"],  dict["request"],dict["status"],dict["size"],  dict["referer"],dict["agent"] ))))  output.close()  dirty.close()  file.close()  if \_\_name\_\_ == "\_\_main\_\_":  if len(sys.argv) != 4:  print "Usage: %s <input\_file> <output\_file> <dirty\_file>" % sys.argv[0]  sys.exit(1)  parseLog(sys.argv[1], sys.argv[2], sys.argv[3]); |

* **方案二：使用Hadoop MapReduce对日志进行过滤，并为日志添加SessionID**
* 使用IDEA创建一个Maven项目：



* 在pom.xml文件中导入开发Hadoop MapReduce所需的jar包：

|  |
| --- |
| <dependencies>  <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-lang3</artifactId>  <version>3.9</version>  </dependency>  <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-client</artifactId>  <version>2.7.5</version>  </dependency>  <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-common</artifactId>  <version>2.7.5</version>  </dependency>  <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-mapreduce-client-common</artifactId>  <version>2.7.5</version>  </dependency>  <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-mapreduce-client-core</artifactId>  <version>2.7.5</version>  </dependency>  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.17</version>  </dependency>  </dependencies> |

* 编写一个通用类WebLogUtil.java，主要包括两个方法，用于对日志进行过滤和为日志添加SessionID。

|  |
| --- |
| public class WebLogUtil {  public static SimpleDateFormat sdf\_standard = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  //为每一条访问记录添加一个SessionID  public static List<String> distinctLogInfoBySession(List<String> logInfoGroup)  throws ParseException {  List<String> logInfoBySession = new ArrayList<String>();  long lastRequestTime = 0;  String lastSessionID = "";  for (String logInfo : logInfoGroup) {  //某IP的用户第一次访问网站的记录做为该用户的第一个session日志  if (lastRequestTime == 0) {  lastSessionID = UUID.randomUUID().toString();  //将该次访问日志记录拼上sessionID并放进按session分类的日志信息数组中  logInfoBySession.add(lastSessionID + "|" + logInfo);  //记录该次访问日志的时间,并用户和下一条访问记录比较,看时间间隔是否超过30分钟,是的话就代表新Session开始  lastRequestTime = sdf\_standard.parse(logInfo.split("\\|")[2]).getTime();  } else {  //当前日志记录和上一次的访问时间相比超过30分钟,所以认为是一个新的Session,重新生成sessionID  if (sdf\_standard.parse(logInfo.split("\\|")[2]).getTime() - lastRequestTime >= 30 \* 60 \* 1000) {  //和上一条访问记录相比,时间间隔超过了30分钟,所以当做一次新的session,并重新生成sessionID  lastSessionID = UUID.randomUUID().toString();  logInfoBySession.add(lastSessionID + "|" + logInfo);  //记录该次访问日志的时间,做为一个新session开始的时间,并继续和下一条访问记录比较,看时间间隔是否又超过30分钟  lastRequestTime = sdf\_standard.parse(logInfo.split("\\|")[2]).getTime();  } else {  //当前日志记录和上一次的访问时间相比没有超过30分钟,所以认为是同一个Session,继续沿用之前的sessionID  logInfoBySession.add(lastSessionID + "|" + logInfo);  }  }  }  return logInfoBySession;  }  //过滤掉信息不全或者格式不正确的日志信息  public static String weblogParser(String logLine) throws ParseException {  boolean isStandardLogInfo = logLine.split("`").length >= 6;  if (isStandardLogInfo) {  //过滤掉多余的符号  String newLogLine = logLine.replace("- - ", "")  .replaceFirst("\\[", "").replace(" +0000]", "");  //如果访问时间不存在，也是一个不正确的日志信息  List<String> logInfoGroup = Arrays.asList(newLogLine.split("`"));  String oldDateFormat = logInfoGroup.get(2);  if (oldDateFormat == "-") return "";  //判断是否符合正则表达式要求  String[] items = {  "(?<ip>\\S+)", // ip  "(?<user>\\S\*)", // user  "(?<time>.+)", // time  "(?<request>.\*)", // request  "(?<status>[0-9]+)", // status  "(?<size>[0-9-]+)", // size  "(?<referer>.\*)", // referer  "(?<agent>.\*)" // user agent  };  Pattern re = Pattern.compile(StringUtils.join(items, "`") + "\\s\*\\Z");  Matcher m = re.matcher(newLogLine);  boolean res = m.matches();  if (res) {  Hashtable<Integer, String> dict = new Hashtable<>();  dict.put(8, m.group("ip"));  dict.put(7, m.group("user"));  dict.put(6, m.group("time"));  dict.put(5, m.group("request"));  dict.put(4, m.group("status"));  dict.put(3, m.group("size"));  dict.put(2, m.group("referer"));  dict.put(1, m.group("agent"));  if (dict.get(3).equals("-")) {  dict.put(3, "0");  }  for (int key : dict.keySet()) {  if (dict.get(key) == "-")  dict.replace(key, "");  }  return StringUtils.join(dict.values(), "|");  } else {  return "";  }  } else {  return "";  }  }  } |

* 编写MapReduce实现对日志过滤，并添加SessionID的工作

**Mapper类：WebLogMapper.java：**

|  |
| --- |
| public class WebLogMapper extends Mapper<LongWritable, Text, Text, Text> {  @Override  protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  String logLine = value.toString();  try {  String valueout = WebLogUtil.weblogParser(logLine);  String keyout = valueout.split("\\|")[0];  context.write(new Text(keyout), new Text(valueout));  } catch (Exception e) {  e.printStackTrace();  }  }  } |

**Reducer类：WebLogReducer.java：**

|  |
| --- |
| public class WebLogReducer extends Reducer<Text, Text, Text, Text> {  @Override  protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {  List<String> loginfoGroup = new ArrayList<>();  try {  values.forEach(value -> {  loginfoGroup.add(value.toString());  });  List<String> loginfoBySession=WebLogUtil.distinctLogInfoBySession(loginfoGroup);  for (String valueout : loginfoBySession) {  context.write(new Text(""), new Text(valueout));  }  } catch (Exception e) {  e.printStackTrace();  }  }  } |

**驱动类：WebLogDriver.java：**

|  |
| --- |
| public class WebLogDriver {  public static void main(String[] args) throws Exception {  Configuration conf = new Configuration();  conf.addResource("classpath:/resources/core-site.xml");  conf.addResource("classpath:/resources/hdfs-site.xml");  conf.addResource("classpath:/resources/mapred-site.xml");  conf.addResource("classpath:/resources/yarn-site.xml");  Job job = null;  job = Job.getInstance(conf);  job.setJarByClass(WebLogDriver.class);  job.setJobName("Min temperature");  job.setMapperClass(WebLogMapper.class);  job.setReducerClass(WebLogReducer.class);  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(FloatWritable.class);  FileInputFormat.addInputPath(job, new Path("hdfs://10.10.0.10:9000/weblog/in/coolshell\_20140212.log"));  FileOutputFormat.setOutputPath(job, new Path("hdfs://10.10.155.101:9000/weblog/out/"));  // System.exit(job.waitForCompletion(true) ? 0 : 1);  job.submit();  }  } |

# 数据分析工具编写

## 任务描述

1. 数据全量和增量的导入
2. 编写SqoopUtil工具类
3. 数据导出
4. 编写数据分析工具类HiveUtil
5. 数据分析和报表

## 任务指导

1. 数据分析工具的业务需求

主要包括**数据导入导出工具**和**数据分析工具**。

1. 数据全量和增量的导入

数据导出模块和数据导入模块都是基于Sqoop做二次开发，在系统上线时，需要一次性将前一天的数据全量导入，上线后，需要定期执行将前一天的数据增量导入HDFS中。

1. 编写SqoopUtil工具类

使用SqoopUtil工具，可以实现通过读取配置文件的方式在Python执行数据的导入和导出操作。

1. 数据导出

同数据导入类似。

1. 编写数据分析工具类HiveUtil

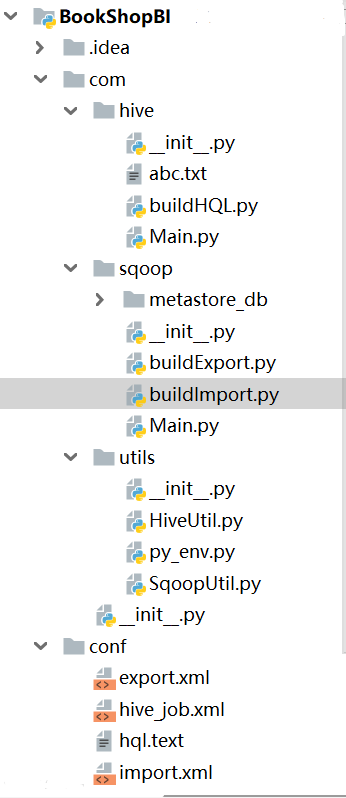
数据分析工具是一个可以执行HQL语句的工具，本系统中的大部分数据分析是使用HQL查询数据仓库中的表完成的，HiveUtil类，通过读取配置文件中的HQL，在Python中执行数据分析的操作。

1. 数据分析和报表

利用数据分析工具可以轻松地进行一些常规的数据分析，再将分析的结果数据导回至关系型数据库。需要了解OLAP和Hive的区别，以及OLAP和多维模型。

## 任务实现

1. 项目目录结构



1. 数据全量和增量的导入

* 编写配置文件：主要目的是告诉Sqoop，导入的表以及导入方式。在conf目录下创建一个名为import.xml的文件。

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <root>  <!-- import\_type="all|add"，all为全量导入； add为增量导入-->  <sqoop-shell type="import" table="book\_category" import\_type="all">  <param key="connect">jdbc:mysql://master:3306/books</param>  <param key="username">hive</param>  <param key="password">hive</param>  <param key="query">"select cid,cname from book\_category where date\_format(created,'%Y-%m-%d hh24:mi:ss') \$dt and \$CONDITIONS"</param>  <param key="target-dir">/data/warehouse/book\_category</param>  <param key="hive-import"></param>  <param key="hive-database">test</param>  <param key="hive-table">book\_category</param>  <param key="hive-partition-key">dt</param>  <param key="hive-partition-value">$dt</param>  <!-- create-sqoop-table 建表，如果表已经存在，该操作会报错！  <param key="create-hive-table"></param>  <param key="hive-overwrite"></param>-->  <param key="hive-delims-replacement">'\\t'</param>  <!--使用3个map任务并行导入数据-->  <param key="m">3</param>  <param key="null-string">'\\N'</param>  <param key="null-non-string">'\\N'</param>  <param key="split-by">cate\_id</param>  </sqoop-shell>  <sqoop-shell type="import" table="orders" import\_type="all">  <param key="connect">jdbc:mysql://master:3306/books</param>  <param key="username">hive</param>  <param key="password">hive</param>  <param key="query">"select \* from orders where date\_format(update\_time,'%Y-%m-%d hh24:mi:ss') \$dt and \$CONDITIONS"</param>  <param key="target-dir">/data/warehouse/orders</param>  <param key="hive-import"></param>  <param key="hive-database">test</param>  <param key="hive-table">orders</param>  <param key="hive-partition-key">dt</param>  <param key="hive-partition-value">$dt</param>  <!-- create-sqoop-table 建表，如果表已经存在，该操作会报错！  <param key="create-hive-table"></param>  <param key="hive-overwrite"></param>-->  <param key="hive-delims-replacement">'\\t'</param>  <param key="m">3</param>  <param key="null-string">'\\N'</param>  <param key="null-non-string">'\\N'</param>  <param key="split-by">orderid</param>  </sqoop-shell>  <sqoop-shell type="import" table="order\_detail" import\_type="all">  <param key="connect">jdbc:mysql://master:3306/books</param>  <param key="username">hive</param>  <param key="password">hive</param>  <param key="query">"select \* from order\_detail where date\_format(delivery\_time,'%Y-%m-%d hh24:mi:ss') \$dt and \$CONDITIONS"</param>  <param key="target-dir">/data/warehouse/order\_detail</param>  <param key="hive-import"></param>  <param key="hive-database">test</param>  <param key="hive-table">order\_detail</param>  <param key="hive-partition-key">dt</param>  <param key="hive-partition-value">$dt</param>  <!-- create-sqoop-table 建表，如果表已经存在，该操作会报错！  <param key="create-hive-table"></param>  <param key="hive-overwrite"></param>-->  <param key="hive-delims-replacement">'\\t'</param>  <param key="m">3</param>  <param key="null-string">'\\N'</param>  <param key="null-non-string">'\\N'</param>  <param key="split-by">order\_id</param>  </sqoop-shell>  <sqoop-shell type="import" table="bx\_book\_ratings" import\_type="all">  <param key="connect">jdbc:mysql://master:3306/books</param>  <param key="username">hive</param>  <param key="password">hive</param>  <param key="query">"select \* from bx\_book\_ratings where date\_format('1900-01-01','%Y-%m-%d hh24:mi:ss') \$dt and \$CONDITIONS"</param>  <param key="target-dir">/data/warehouse/bx\_book\_ratings</param>  <param key="hive-import"></param>  <param key="hive-database">test</param>  <param key="hive-table">bx\_book\_ratings</param>  <param key="hive-partition-key">dt</param>  <param key="hive-partition-value">$dt</param>  <!-- create-sqoop-table 建表，如果表已经存在，该操作会报错！  <param key="create-hive-table"></param>  <param key="hive-overwrite"></param>-->  <param key="hive-delims-replacement">'\\t'</param>  <param key="m">3</param>  <param key="null-string">'\\N'</param>  <param key="null-non-string">'\\N'</param>  <param key="split-by">userid</param>  </sqoop-shell>  </root> |

* 编写一个buildImport.py文件，用于解析配置文件，然后拼接成将数据从MySql导入到HDFS的Sqoop命令

|  |
| --- |
| # 其中dt为昨天的日期，将由调度模块传入  def resolve\_conf(dt):  # 获取配置文件路径  conf\_file = env.PROJECT\_CONF\_DIR + "import.xml"  # 解析配置文件  xml\_tree = ElementTree.parse(conf\_file)  # 获取所有的sqoop-shell元素  shells = xml\_tree.findall('./sqoop-shell')  # 用来保存待执行的Sqoop命令的集合  cmds = []  for shell in shells:  # 获取sqoop命令类型，如：import或export  sqoop\_cmd\_type = shell.attrib["type"]  # 获取导入类型，如：增量导入或全量导入  import\_type = shell.attrib["import\_type"]  # 获取  params = shell.findall("./param")  # 首先组装sqoop命令头  command = "sqoop " + sqoop\_cmd\_type  for param in params:  if (import\_type == "all"):  import\_condition = "< '" + dt + "'"  elif (import\_type == "add"):  import\_condition = "= '" + dt + "'"  else:  raise Exception  key = param.attrib["key"]  value = param.text  if (value == None or value == "" or value == " "):  value = ""  if (key == "query"):  value = value.replace("\$dt", import\_condition)  if (key == "hive-partition-value"):  value = value.replace("$dt", dt)  command += " --" + key + " " + value  cmds.append(command)  return cmds |

* 编写SqoopUtil工具类，用于实现在Python中调用Sqoop完成数据导入的操作。

|  |
| --- |
| class SqoopUtil(object):  '''  sqoop operation  '''  def \_\_init\_\_(self):  pass  @staticmethod  def execute\_shell(shell, sqoop\_path=SQOOP\_PATH):  # 执行传入的shell命令  status, output = subprocess.getstatusoutput(sqoop\_path + shell)  if status != 0:  print("failed:the number is " + str(status))  print(output)  return None  else:  print("success")  output = str(output).split("\n")  return output |

* 在sqoop目录下编写一个Main.py，用于整合上面的操作

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  today = datetime.date.today() + datetime.timedelta(-1)  dt = today.strftime('%Y-%m-%d')  cmds = imp.resolve\_conf(dt)  for cmd in cmds:  print(cmd)  # 执行导入过程  SqoopUtil.execute\_shell(cmd) |

1. 数据导出

* 编写配置文件：主要目的是告诉Sqoop，导出的方式。在conf目录下创建一个名为export.xml的文件。

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <root>  <!-- 为每一个“从Hive导出到数据库中表”的命令定义sqoop shell参数 -->  <!-- Hive to mysql 全部导出-->  <sqoop-shell type="export">  <param key="connect">jdbc:mysql://master:3306/books</param>  <param key="username">hive</param>  <param key="password">hive</param>  <!--mysql上的表-->  <param key="table">book\_category</param>  <!--hive上的schema-->  <param key="hcatalog-database">test</param>  <!--hive上的表-->  <param key="hcatalog-table">book\_category</param>  <param key="m">1</param>  </sqoop-shell>  <!-- Hive to mysql 部分导出-->  <sqoop-shell type="export">  <param key="connect">jdbc:mysql://master:3306/books</param>  <param key="username">hive</param>  <param key="password">hive</param>  <!--mysql上的表-->  <param key="table">temp</param>  <!--要导入的列-->  <param key="columns">"cate\_id,name"</param>  <!--hive上的schema-->  <param key="hcatalog-database">test</param>  <!--hive上的表-->  <param key="hcatalog-table">book\_category</param>  <param key="m">1</param>  </sqoop-shell>  <!-- HDFS to mysql  <sqoop-shell type="export">  <param key="connect">jdbc:mysql://master1:3306/BX</param>  <param key="username">hive</param>  <param key="password">hive</param>  <param key="table">category</param>  <param key="export-dir">/tmp/BX/category</param>  <param key="m">1</param>  </sqoop-shell>  -->  </root> |

* 编写一个buildExport.py文件，用于解析配置文件，然后拼接成将数据从HDFS导出到MySql的Sqoop命令

|  |
| --- |
| # 数据导出：生成“将数据从Hive导出到MySql”的命令。  #  # 1.全部导出  # Sqoop export --connect jdbc:mysql://127.0.0.1:3306/dbname  # --username mysql(mysql用户名) --password 123456(密码)  # --table student(mysql上的表) --hcatalog-database sopdm(hive上的schema)  # --hcatalog-table student(hive上的表)  #  # 2.部分导出  # Sqoop export --connect jdbc:mysql://127.0.0.1:3306/dbname  # --username mysql(mysql用户名) --password 123456(密码)  # --table student(mysql上的表) --columns "id,name,age" --hcatalog-database sopdm(hive上的schema)  # --hcatalog-table student(hive上的表)  def resolve\_conf(dt):  # 导出配置文件路径  conf\_file = env.PROJECT\_CONF\_DIR + "export.xml"  # 解析配置文件  xml\_tree = ElementTree.parse(conf\_file)  # 获取sqoop-shell元素  shells = xml\_tree.findall('./sqoop-shell')  # 用来保存待执行的Sqoop命令的集合  cmds = []  for shell in shells:  # sqoop命令类型  sqoop\_cmd\_type = shell.attrib["type"]  # 首先组装成sqoop命令头  command = "sqoop " + sqoop\_cmd\_type  # 获取  params = shell.findall('./param')  for param in params:  # 获取key属性值  key = param.attrib["key"]  # 获取param标签的中间值  value = param.text  # 如果不是键值对形式的命令选项  if (value == None or value == "" or value == " "):  value = ""  # 继续组装命令  command += " --" + key + " " + value  cmds.append(command)  return cmds |

* 在sqoop目录下编写一个Main.py，用于整合上面的操作

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  today = datetime.date.today() + datetime.timedelta(-1)  dt = today.strftime('%Y-%m-%d')  cmds = exp.resolve\_conf(dt)  for cmd in cmds:  print(cmd)  # 执行导出过程  SqoopUtil.execute\_shell(cmd) |

1. 编写数据分析工具类HiveUtil

* hive\_job.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <root>  <!-- 表示可执行的type -->  <job type="analysis">  <!--将查询导出一个文件-->  <hql>"select \* from test.book\_category" > abc.txt</hql>  <!--导出到HDFS-->  <hql>"insert overwrite directory '/data/warehouse/book\_category' select \* from test.book\_category"</hql>  <!--导出到另外的hive表-->  <hql>"insert into table test.temp select orderid,userid,orderdate from test.orders where status=0"</hql>  </job>  <job type="etl\_db">  <hql>""</hql>  </job>  <job type="etl\_data\_dedup">  <hql>  "insert overrite table orders PARTITION(dt)  select \* from (  select \* from (  select \* from orders where dt=$dt  union all  select \* from orders where dt<$dt) t1 DISTRIBUTE BY orderid SORT BY orderid,update\_time) t2  WHERE rownum(orderid)=1"  </hql>  <!--使用HQL进行数据去重，公式：结果=（全量数据减去（全量数据与增量数据的交集））再与增量数据取并集 -->  <hql>  "insert overwrite table test.dedup\_orders select \* from (select t1.\* from (select \* from test.orders  where dt<$dt) t1 left outer join (select \* from test.orders where dt=$dt) t2 on t1.orderid=t2.orderid  where t2.orderid is null UNION ALL select \* from test.orders where dt=$dt) t3"  </hql>  </job>  </root> |

* buildHQL.py用于解析hive\_job.xml文件

|  |
| --- |
| def resolve\_conf(type, dt):  # 获取配置文件路径  confFile = PROJECT\_CONF\_DIR + "hive\_job.xml"  # 解析配置文件  xmlTree = ET.parse(confFile)  # 获取job元素  jobs = xmlTree.findall('./job')  # 用来保存hql  hqls = []  # 遍历所有job的子元素，获得所需参数  for job in jobs:  # 如果job的type是需要执行的type  if job.attrib["type"] == type:  for hql in job.getchildren():  # 获取hql标签的text值，并去掉两边可能出现的空字符  hql = hql.text.strip()  # 检查hql有效性，无效则抛出异常  if len(hql) == 0 or hql == "" or hql == None:  raise Exception("hql语法不正确")  else:  # 将$t替换为时间  hql = hql.replace("\$dt", "< '" + dt + "'")  hql = hql.replace("$dt", "'" + dt + "'")  hqls.append(hql)  return hqls |

* HiveUtil.py用于执行Hive查询

|  |
| --- |
| class HiveUtil(object):  def \_\_init\_\_(self):  pass  @staticmethod  def execute\_shell(hql, hive\_path=HIVE\_PATH):  # 将hql语句进行字符串转义  # hql = hql.replace("\"", "'")  # 执行查询，并取得执行的状态和输出  status, output = subprocess.getstatusoutput(cmd=hive\_path + "hive -S -e " + hql)  if status != 0:  print("failed:the number is " + str(status))  print(output)  return None  else:  print("success")  output = str(output).split("\n")  return output |

* Main.py

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  today = datetime.date.today() + datetime.timedelta(-1)  dt = today.strftime('%Y-%m-%d')  hqls = bq.resolve\_conf("analysis", dt)  # print hqls[3]  # HiveUtil.execute\_shell(hqls[3])  for hql in hqls:  print(hql)  HiveUtil.execute\_shell(hql) |

1. 数据分析和报表

* Hive由于MapReduce计算框架的原因，执行响应时间是分钟级别的，本身并不适合做OLAP，所以一般情况下都是先使用Hive进行离线计算，将计算结果导出到关系型数据库中进行OLAP的操作。

**Hive**

**MySql**

**报表系统**

* 利用Hive离线计算能力强的优点对大数据进行数据分析，使用MySql之类的小型数据库轻松应对报表系统的OLAP操作和并发量。
* 需要学生自行扩展功能实现使用Hive分析数据，并将结果保存到MySql，用于生成报表。