一、 课程计划

目录

二、 模块开发----数据仓库设计

1. 维度建模基本概念

维度建模(dimensional modeling)是专门用于分析型数据库、数据仓库、数据集市建模的方法。数据集市可以理解为是一种"小型数据仓库"。

维度表(dimension)

维度表示你要对数据进行分析时所用的一个量,比如你要分析产品销售情况,你可以选择按类别来进行分析,或按区域来分析。这样的按..分析就构成一个维度。再比如"昨天下午我在星巴克花费 200 元喝了一杯卡布奇诺"。那么以消费为主题进行分析，可从这段信息中提取三个维度：时间维度(昨天下午)，地点维度(星巴克)， 商品维度(卡布奇诺)。通常来说维度表信息比较固定，且数据量小。

事实表(fact table)

表示对分析主题的度量。事实表包含了与各维度表相关联的外键，并通过 JOIN 方式与维度表关联。事实表的度量通常是数值类型，且记录数会不断增加，表规模迅速增长。比如上面的消费例子，它的消费事实表结构示例如下：

消费事实表：Prod_id(引用商品维度表)， TimeKey(引用时间维度表)，Place_id(引用地点维度表)， Unit(销售量)。

总的说来，在数据仓库中不需要严格遵守规范化设计原则。因为数据仓库的主导功能就是面向分析，以查询为主，不涉及数据更新操作。事实表的设计是以能够正确记录历史信息为准则，维度表的设计是以能够以合适的角度来聚合主题内容为准则。

2. 维度建模三种模式

2.1. 星型模式

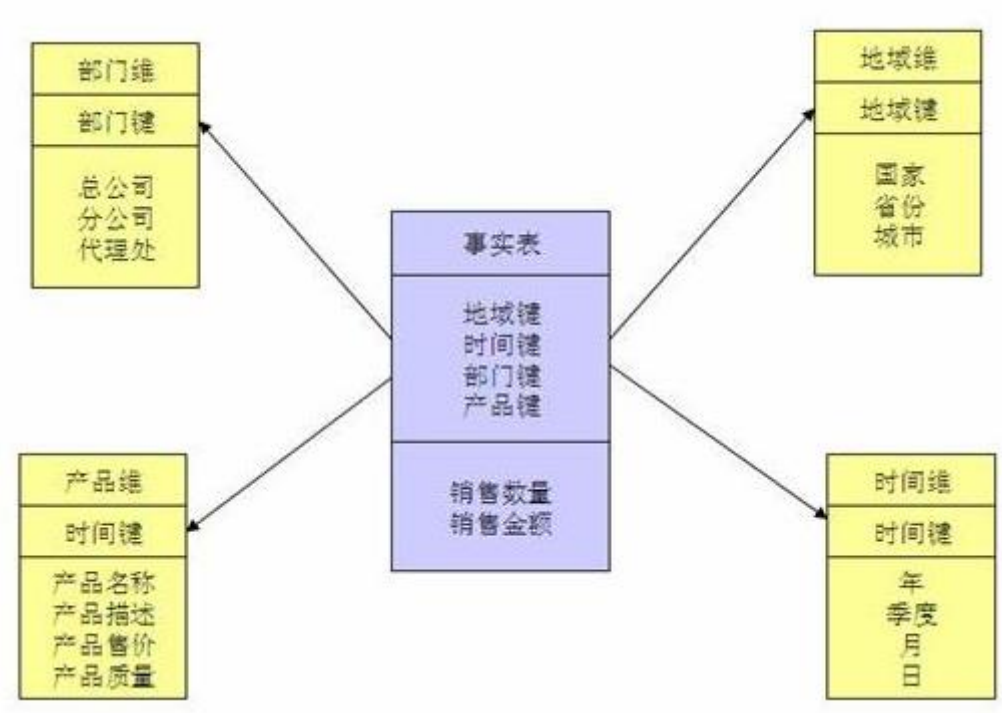
星形模式(Star Schema)是最常用的维度建模方式。星型模式是以事实表为中心，所有的维度表直接连接在事实表上，像星星一样。

星形模式的维度建模由一个事实表和一组维表成，且具有以下特点：

a. 维表只和事实表关联，维表之间没有关联；

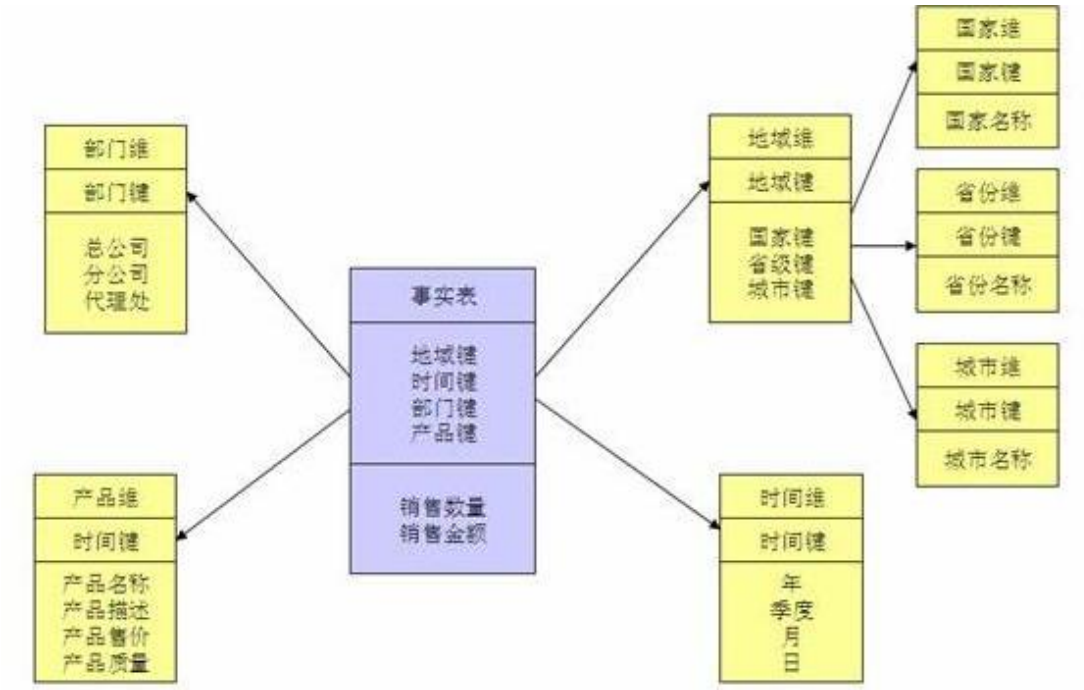
b. 每个维表主键为单列，且该主键放置在事实表中，作为两边连接的外键；

c. 以事实表为核心，维表围绕核心呈星形分布；



2.2. 雪花模式

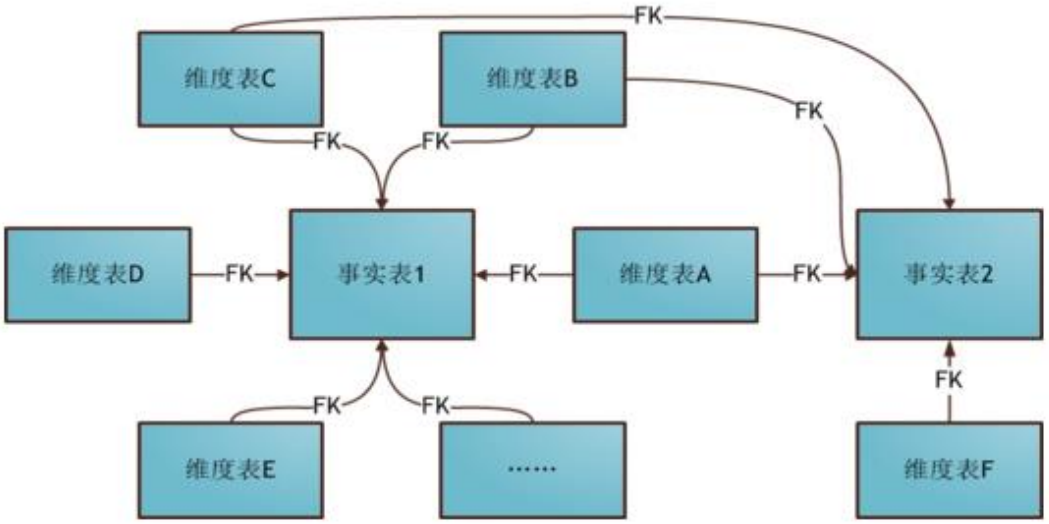
雪花模式(Snowflake Schema)是对星形模式的扩展。雪花模式的维度表可以拥有其他维度表的，虽然这种模型相比星型更规范一些，但是由于这种模型不太容易理解，维护成本比较高，而且性能方面需要关联多层维表，性能也比星型模型要低。所以一般不是很常用。



2.3. 星座模式

星座模式是星型模式延伸而来，星型模式是基于一张事实表的，而星座模式是基于多张事实表的，而且共享维度信息。

前面介绍的两种维度建模方法都是多维表对应单事实表，但在很多时候维度空间内的事实表不止一个，而一个维表也可能被多个事实表用到。在业务发展后期，绝大部分维度建模都采用的是星座模式。



3. 本项目中数据仓库的设计

注：采用星型模型

3.1. 事实表设计

原始数据表: ods_weblog_origin =>对应 mr 清洗完之后的数据		
valid	string	是否有效
remote_addr	string	访客 ip
remote_user	string	访客用户信息
time_local	string	请求时间
request	string	请求 url
status	string	响应码
body_bytes_sent	string	响应字节数
http_referer	string	来源 url
http_user_agent	string	访客终端信息

访问日志明细宽表：dw_weblog_detail		
valid	string	是否有效
remote_addr	string	访客 ip
remote_user	string	访客用户信息
time_local	string	请求完整时间
daystr	string	访问日期
timestr	string	访问时间
month	string	访问月
day	string	访问日
hour	string	访问时
request	string	请求 url 整串
status	string	响应码
body_bytes_sent	string	响应字节数
http_referer	string	来源 url
ref_host	string	来源的 host
ref_path	string	来源的路径
ref_query	string	来源参数 query
ref_query_id	string	来源参数 query 值
http_user_agent	string	客户终端标识

3.2. 维度表设计

时间维度 t_dim_time
date_Key
year
month
day
hour

访客地域维度 t_dim_area
area_ID
北京
上海
广州
深圳

终端类型维度 t_dim_termination
uc
firefox
chrome
safari
ios
android

网站栏目维度 t_dim_section
跳蚤市场
房租信息
休闲娱乐
建材装修
本地服务
人才市场

注意：

维度表的数据一般要结合业务情况自己写脚本按照规则生成，也可以使用工具生成，方便后续的关联分析。

比如一般会事前生成时间维度表中的数据，跨度从业务需要的日期到当前日期即可. 具体根据你的分析粒度, 可以生成年，季，月，周，天，时等相关信息，用于分析。

三、 模块开发----ETL

ETL 工作的实质就是从各个数据源提取数据，对数据进行转换，并最终加载填充数据到数据仓库维度建模后的表中。只有当这些维度/事实表被填充好，ETL 工作才算完成。

本项目的数据分析过程在 hadoop 集群上实现，主要应用 hive 数据仓库工具，因此，采集并经过预处理后的数据，需要加载到 hive 数据仓库中，以进行后续的分析过程。

1. 创建 ODS 层数据表

1.1. 原始日志数据表

```
drop table if exists ods_weblog_origin;
create table ods_weblog_origin(
valid string,
remote_addr string,
remote_user string,
time_local string,
request string,
status string,
body_bytes_sent string,
http_referer string,
http_user_agent string)
partitioned by (datestr string)
row format delimited
fields terminated by '\001';
```

1.2. 点击流模型 pageviews 表

```
drop table if exists ods_click_pageviews;
create table ods_click_pageviews(
session string,
remote_addr string,
remote_user string,
time_local string,
```

```
request string,
visit_step string,
page_staylong string,
http_referer string,
http_user_agent string,
body_bytes_sent string,
status string)
partitioned by (datestr string)
row format delimited
fields terminated by '\001';
```

1.3. 点击流 visit 模型表

```
drop table if exist ods_click_stream_visit;
create table ods_click_stream_visit(
session          string,
remote_addr string,
inTime           string,
outTime          string,
inPage           string,
outPage          string,
referal          string,
pageVisits    int)
partitioned by (datestr string)
row format delimited
fields terminated by '\001';
```

2. 导入 ODS 层数据

```
load data inpath '/weblog/preprocessed/' overwrite into table
ods_weblog_origin partition(datestr='20130918');--数据导入
show partitions ods_weblog_origin;---查看分区
select count(*) from ods_weblog_origin; --统计导入的数据总数
```

点击流模型的两张表数据导入操作同上。

注：生产环境中应该将数据 load 命令，写在脚本中，然后配置在 azkaban 中定时运行，注意运行的时间点，应该在预处理数据完成之后。

3. 生成 ODS 层明细宽表

3.1. 需求实现

整个数据分析的过程是按照数据仓库的层次分层进行的，总体来说，是从 ODS 原始数据中整理出一些中间表（比如，为后续分析方便，将原始数据中的时间、url 等非结构化数据作结构化抽取，将各种字段信息进行细化，形成明细表），然后再在中间表的基础之上统计出各种指标数据。

3.2. ETL 实现

- 建明细表 ods_weblog_detail:

```
drop table ods_weblog_detail;
create table ods_weblog_detail(
valid                 string, --有效标识
remote_addr           string, --来源 IP
remote_user           string, --用户标识
time_local            string, --访问完整时间
daystr                string, --访问日期
timestr               string, --访问时间
month                 string, --访问月
day                   string, --访问日
hour                  string, --访问时
request               string, --请求的 url
status                string, --响应码
body_bytes_sent string, --传输字节数
http_referer          string, --来源 url
ref_host              string, --来源的 host
ref_path              string, --来源的路径
ref_query             string, --来源参数 query
ref_query_id          string, --来源参数 query 的值
http_user_agent string --客户终端标识
)
partitioned by(datestr string);
```

- 通过查询插入数据到明细宽表 ods_weblog_detail 中

1、 抽取 refer_url 到中间表 t_ods_tmp_referurl

也就是将来访 url 分离出 host path query query id

```
drop table if exists t_ods_tmp_referurl;
create table t_ods_tmp_referurl as
SELECT a.*,b.*
FROM ods_weblog_origin a
LATERAL VIEW parse_url_tuple(regexp_replace(http_referer, "\"", ""),
'HOST', 'PATH','QUERY', 'QUERY:id') b as host, path, query, query_id;
```

注：lateral view 用于和 split, explode 等 UDTF 一起使用，它能够将一列数据拆成多行数据。

UDTF(User-Defined Table-Generating Functions) 用来解决输入一行输出多行(On-to-many maping) 的需求。Explode 也是拆列函数，比如 Explode (ARRAY) ，array 中的每个元素生成一行。

2、抽取转换 time_local 字段到中间表明细表 t_ods_tmp_detail

```
drop table if exists t_ods_tmp_detail;
create table t_ods_tmp_detail as
select b.*,substring(time_local,0,10) as daystr,
substring(time_local,12) as tmstr,
substring(time_local,6,2) as month,
substring(time_local,9,2) as day,
substring(time_local,11,3) as hour
from t_ods_tmp_referurl b;
```

3、以上语句可以合成一个总的语句

```
insert into table shizhan.ods_weblog_detail partition(datestr='2013-09-18')
select c.valid,c.remote_addr,c.remote_user,c.time_local,
substring(c.time_local,0,10) as daystr,
substring(c.time_local,12) as tmstr,
substring(c.time_local,6,2) as month,
```

```
    substring(c.time_local,9,2) as day,
    substring(c.time_local,11,3) as hour,
    c.request,c.status,c.body_bytes_sent,c.http_referer,c.ref_host,c.ref_
path,c.ref_query,c.ref_query_id,c.http_user_agent
    from
    (SELECT
    a.valid,a.remote_addr,a.remote_user,a.time_local,
    a.request,a.status,a.body_bytes_sent,a.http_referer,a.http_user_agent
,b.ref_host,b.ref_path,b.ref_query,b.ref_query_id
    FROM shizhan.ods_weblog_origin a LATERAL VIEW
    parse_url_tuple(regexp_replace(http_referer,  "\"",  ""),  'HOST',
'PATH','QUERY', 'QUERY:id') b as ref_host, ref_path, ref_query,
     ref_query_id) c;
```

四、 模块开发----统计分析

数据仓库建设好以后，用户就可以编写 Hive SQL 语句对其进行访问并对其中数据进行分析。

在实际生产中，究竟需要哪些统计指标通常由数据需求相关部门人员提出，而且会不断有新的统计需求产生，以下为网站流量分析中的一些典型指标示例。

注：每一种统计指标都可以跟各维度表进行钻取。

1. 流量分析

1.1. 多维度统计 PV 总量

按时间维度

```
--计算每小时 pvs，注意 gruop by 语法
select count(*) as pvs,month,day,hour from ods_weblog_detail group by month,day,hour;
```

方式一：直接在 ods_weblog_detail 单表上进行查询

```
--计算该处理批次（一天）中的各小时 pvs
drop table dw_pvs_everyhour_oneday;
create table dw_pvs_everyhour_oneday(month string,day string,hour string,pvs bigint) partitioned by(datestr string);

insert into table dw_pvs_everyhour_oneday partition(datestr='20130918')
select a.month as month,a.day as day,a.hour as hour,count(*) as pvs from ods_weblog_detail a
where a.datestr='20130918' group by a.month,a.day,a.hour;

--计算每天的 pvs
drop table dw_pvs_everyday;
create table dw_pvs_everyday(pvs bigint,month string,day string);

insert into table dw_pvs_everyday
select count(*) as pvs,a.month as month,a.day as day from ods_weblog_detail a
group by a.month,a.day;
```

方式二：与时间维表关联查询

```
--维度：日
drop table dw_pvs_everyday;
create table dw_pvs_everyday(pvs bigint,month string,day string);

insert into table dw_pvs_everyday
select count(*) as pvs,a.month as month,a.day as day from (select distinct month, day from t_dim_time) a
join ods_weblog_detail b
on a.month=b.month and a.day=b.day
group by a.month,a.day;

--维度：月
drop table dw_pvs_everymonth;
create table dw_pvs_everymonth (pvs bigint,month string);

insert into table dw_pvs_everymonth
select count(*) as pvs,a.month from (select distinct month from t_dim_time)  a
join ods_weblog_detail b on a.month=b.month group by a.month;

--另外，也可以直接利用之前的计算结果。比如从之前算好的小时结果中统计每一天的
Insert into table dw_pvs_everyday
Select sum(pvs) as pvs,month,day from dw_pvs_everyhour_oneday group by month,day having day='18';
```

按终端维度

数据中能够反映出用户终端信息的字段是 http_user_agent。

User Agent 也简称 UA。它是一个特殊字符串头，是一种向访问网站提供所使用的浏览器类型及版本、操作系统及版本、浏览器内核、等信息的标识。例如：

```
User-Agent,Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.276 Safari/537.36
```

上述 UA 信息就可以提取出以下的信息：

chrome 58.0、浏览器 chrome、浏览器版本 58.0、系统平台 windows

浏览器内核 webkit

这里不再拓展相关知识，感兴趣的可以查看参考资料如何解析 UA。

可以用下面的语句进行试探性统计，当然这样的准确度不是很高。

```
select distinct(http_user_agent) from ods_weblog_detail where http_user_agent like '%Chrome%' limit 200;
```

按栏目维度

网站栏目可以理解为网站中内容相关的主题集中。体现在域名上来看就是不同的栏目会有不同的二级目录。比如某网站网址为 www.xxxx.cn,旗下栏目可以通过如下方式访问：

栏目维度：../job

栏目维度：../news

栏目维度：../sports

栏目维度：../technology

那么根据用户请求 url 就可以解析出访问栏目，然后按照栏目进行统计分析。

按 referer 维度

```
--统计每小时各来访 url 产生的 pv 量
drop table dw_pvs_referer_everyhour;
create table dw_pvs_referer_everyhour(referer_url string,referer_host string,month string,day string,hour string,pv_referer_cnt bigint) partitioned by(datestr string);

insert into table dw_pvs_referer_everyhour partition(datestr='20130918')
select http_referer,ref_host,month,day,hour,count(1) as pv_referer_cnt
from ods_weblog_detail
group by http_referer,ref_host,month,day,hour
having ref_host is not null
order by hour asc,day asc,month asc,pv_referer_cnt desc;
```

```
--统计每小时各来访 host 的产生的 pv 数并排序
drop table dw_pvs_refererhost_everyhour;
create table dw_pvs_refererhost_everyhour(ref_host string,month string,day string,hour string,ref_host_cnts bigint) partitioned by(datestr string);

insert into table dw_pvs_refererhost_everyhour partition(datestr='20130918')
select ref_host,month,day,hour,count(1) as ref_host_cnts
from ods_weblog_detail
```

```
group by ref_host,month,day,hour
having ref_host is not null
order by hour asc,day asc,month asc,ref_host_cnts desc;
```

注：还可以按来源地域维度、访客终端维度等计算

```
+-------------------------+-------+------+-------+-------+--------------+--+
| blog.fens.me            | 09    | 19   | 00    | 111   | 2013-09-18   |
| www.fens.me             | 09    | 19   | 00    | 13    | 2013-09-18   |
| h2w.iask.cn             | 09    | 19   | 00    | 6     | 2013-09-18   |
| www.google.com.hk       | 09    | 19   | 00    | 3     | 2013-09-18   |
| angularjs.cn            | 09    | 19   | 00    | 3     | 2013-09-18   |
| cnodejs.org             | 09    | 19   | 00    | 1     | 2013-09-18   |
| www.leonarding.com      | 09    | 19   | 00    | 1     | 2013-09-18   |
| www.itpub.net           | 09    | 19   | 00    | 1     | 2013-09-18   |
| blog.fens.me            | 09    | 19   | 01    | 89    | 2013-09-18   |
| cos.name                | 09    | 19   | 01    | 3     | 2013-09-18   |
+-------------------------+-------+------+-------+-------+--------------+--+
```

1.2. 人均浏览量

需求描述：统计今日所有来访者平均请求的页面数。

人均浏览量也称作人均浏览页数，该指标可以说明网站对用户的粘性。

人均页面浏览量表示用户某一时段平均浏览页面的次数。

计算方式：总页面请求数/去重总人数

remote_addr 表示不同的用户。可以先统计出不同 remote_addr 的 pv 量，然后累加（sum）所有 pv 作为总的页面请求数，再 count 所有 remote_addr 作为总的去重总人数。

```
--总页面请求数/去重总人数
drop table dw_avgpv_user_everyday;
create table dw_avgpv_user_everyday(
day string,
avgpv string);

insert into table dw_avgpv_user_everyday
select '20130918',sum(b.pvs)/count(b.remote_addr) from
(select remote_addr,count(1) as pvs from ods_weblog_detail where datestr='20130918' group by remote_addr) b;
```

1.3. 统计 pv 总量最大的来源 TOPN (分组 TOP)

需求描述：统计每小时各来访 host 的产生的 pvs 数最多的前 N 个（topN）。

row_number()函数

- 语法：row_number() over (partition by xxx order by xxx) rank，rank 为分组的别名，相当于新增一个字段为 rank。
- partition by 用于分组，比方说依照 sex 字段分组
- order by 用于分组内排序，比方说依照 sex 分组，组内按照 age 排序
- 排好序之后，为每个分组内每一条分组记录从 1 开始返回一个数字
- 取组内某个数据，可以使用 where 表名.rank>x 之类的语法去取

以下语句对每个小时内的来访 host 次数倒序排序标号:

```
select ref_host,ref_host_cnts,concat(month,day,hour),
row_number() over (partition by concat(month,day,hour) order by ref_host_cnts desc) as od from dw_pvs_refererhost_everyhour;
```

效果如下：

ref_host	ref_host_cnts	_c2	od	
blog.fens.me	68	0918 06	1	
www.angularjs.cn	3	0918 06	2	
www.google.com	2	0918 06	3	
www.baidu.com	1	0918 06	4	
cos.name	1	0918 06	5	
blog.fens.me	711	0918 07	1	
www.google.com.hk	20	0918 07	2	
www.angularjs.cn	20	0918 07	3	
www.dataguru.cn	10	0918 07	4	
www.fens.me	6	0918 07	5	
r.dataguru.cn	6	0918 07	6	
cnodejs.org	5	0918 07	7	
cos.name	5	0918 07	8	
www.google.com	3	0918 07	9	
it.dataguru.cn	2	0918 07	10	
f.dataguru.cn	2	0918 07	11	
blog.chinaunix.net	2	0918 07	12	
www.baidu.com	1	0918 07	13	
www.weibo.com	1	0918 07	14	
www.google.fr	1	0918 07	15	
74.125.128.160	1	0918 07	16	
image.baidu.com	1	0918 07	17	
blog.fens.me	1556	0918 08	1	
www.fens.me	26	0918 08	2	

根据上述 row_number 的功能，可编写 hql 取各小时的 ref_host 访问次数 topn

```
drop table dw_pvs_refhost_topn_everyhour;
create table dw_pvs_refhost_topn_everyhour(
hour string,
toporder string,
ref_host string,
ref_host_cnts string
)partitioned by(datestr string);

insert into table dw_pvs_refhost_topn_everyhour partition(datestr='20130918')
select t.hour,t.od,t.ref_host,t.ref_host_cnts from
  (select ref_host,ref_host_cnts,concat(month,day,hour) as hour,
row_number() over (partition by concat(month,day,hour) order by ref_host_cnts desc) as od
from dw_pvs_refererhost_everyhour) t where od<=3;
```

结果如下：

```
0: jdbc:hive2://localhost:10000> select * from dw_ref_host_topn_h;
+---------------------------+-------------------------------+-------------------------------+-----------------------------------+
| dw_ref_host_topn_h.hour   | dw_ref_host_topn_h.toporder   | dw_ref_host_topn_h.ref_host   | dw_ref_host_topn_h.ref_host_cnts  |
+---------------------------+-------------------------------+-------------------------------+-----------------------------------+
| Sep1806                   | 1                             | blog.fens.me                  | 68                                |
| Sep1806                   | 2                             | www.angularjs.cn              | 3                                 |
| Sep1806                   | 3                             | www.google.com                | 2                                 |
| Sep1807                   | 1                             | blog.fens.me                  | 706                               |
| Sep1807                   | 2                             | www.angularjs.cn              | 20                                |
| Sep1807                   | 3                             | www.google.com.hk             | 20                                |
| Sep1808                   | 1                             | blog.fens.me                  | 1550                              |
| Sep1808                   | 2                             | www.fens.me                   | 26                                |
| Sep1808                   | 3                             | www.baidu.com                 | 15                                |
| Sep1809                   | 1                             | blog.fens.me                  | 1037                              |
| Sep1809                   | 2                             | www.baidu.com                 | 27                                |
| Sep1809                   | 3                             | www.google.com.hk             | 11                                |
| Sep1810                   | 1                             | blog.fens.me                  | 306                               |
| Sep1810                   | 2                             | www.baidu.com                 | 15                                |
| Sep1810                   | 3                             | www.angularjs.cn              | 7                                 |
| Sep1811                   | 1                             | blog.fens.me                  | 245                               |
| Sep1811                   | 2                             | fens.me                       | 12                                |
| Sep1811                   | 3                             | www.angularjs.cn              | 7                                 |
| Sep1812                   | 1                             | blog.fens.me                  | 394                               |
| Sep1812                   | 2                             | cnodejs.org                   | 3                                 |
| Sep1812                   | 3                             | f.dataguru.cn                 | 2                                 |
| Sep1813                   | 1                             | blog.fens.me                  | 243                               |
| Sep1813                   | 2                             | www.baidu.com                 | 27                                |
| Sep1813                   | 3                             | cos.name                      | 6                                 |
| Sep1814                   | 1                             | blog.fens.me                  | 259                               |
| Sep1814                   | 2                             | www.angularjs.cn              | 6                                 |
```

2． 受访分析（从页面的角度分析）

2.1． 各页面访问统计

主要是针对数据中的 request 进行统计分析，比如各页面 PV ，各页面 UV 等。

以上指标无非就是根据页面的字段 group by。例如：

```
--统计各页面 pv
select request as request,count(request) as request_counts from
ods_weblog_detail group by request having request is not null order by request_counts desc limit 20;
```

2.2． 热门页面统计

```
--统计每日最热门的页面 top10
drop table dw_hotpages_everyday;
create table dw_hotpages_everyday(day string,url string,pvs string);

insert into table dw_hotpages_everyday
select '20130918',a.request,a.request_counts from
(select request as request,count(request) as request_counts from ods_weblog_detail where datestr='20130918'
group by request having request is not null) a
order by a.request_counts desc limit 10;
```

3． 访客分析

3.1． 独立访客

需求描述：按照时间维度比如小时来统计独立访客及其产生的 pv。

对于独立访客的识别，如果在原始日志中有用户标识，则根据用户标识即很好实现;此处，由于原始日志中并没有用户标识，以访客 IP 来模拟，技术上是一样的，只是精确度相对较低。

```
--时间维度：时
drop table dw_user_dstc_ip_h;
create table dw_user_dstc_ip_h(
remote_addr string,
pvs         bigint,
hour        string);

insert into table dw_user_dstc_ip_h
select remote_addr,count(1) as pvs,concat(month,day,hour) as hour
from ods_weblog_detail
Where datestr='20130918'
group by concat(month,day,hour),remote_addr;

在此结果表之上，可以进一步统计，如每小时独立访客总数：
select count(1) as dstc_ip_cnts,hour from dw_user_dstc_ip_h group by hour;
```

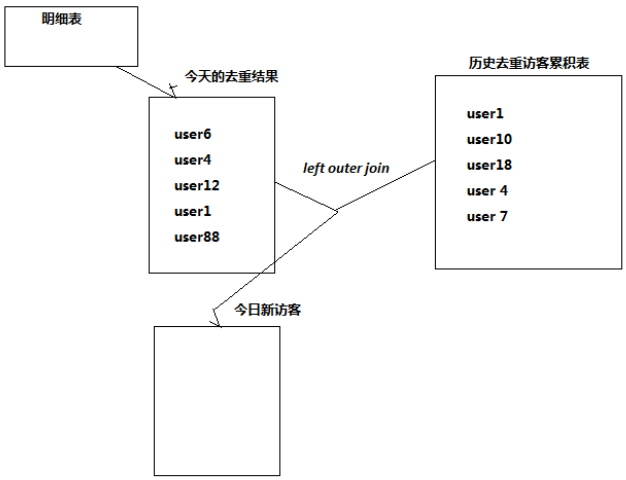
```
--时间维度：日
select remote_addr,count(1) as counts,concat(month,day) as day
from ods_weblog_detail
Where datestr='20130918'
group by concat(month,day),remote_addr;
```

```
--时间维度：月
select remote_addr,count(1) as counts,month
from ods_weblog_detail
group by month,remote_addr;
```

3.2. 每日新访客

需求：将每天的新访客统计出来。

实现思路：创建一个去重访客累积表，然后将每日访客对比累积表。




```
--历日去重访客累积表
drop table dw_user_dsct_history;
create table dw_user_dsct_history(
day string,
ip string
)
partitioned by(datestr string);

--每日新访客表
drop table dw_user_new_d;
create table dw_user_new_d (
day string,
ip string
)
partitioned by(datestr string);

--每日新用户插入新访客表
insert into table dw_user_new_d partition(datestr='20130918')
select tmp.day as day,tmp.today_addr as new_ip from
(
select today.day as day,today.remote_addr as today_addr,old.ip as old_addr
from
```

```
(select distinct remote_addr as remote_addr,"20130918" as day from ods_weblog_detail where datestr="20130918") today
left outer join
dw_user_dsct_history old
on today.remote_addr=old.ip
) tmp
where tmp.old_addr is null;

--每日新用户追加到累计表
insert into table dw_user_dsct_history partition(datestr='20130918')
select day,ip from dw_user_new_d where datestr='20130918';
```

验证查看：

```
select count(distinct remote_addr) from ods_weblog_detail;

select count(1) from dw_user_dsct_history where datestr='20130918';

select count(1) from dw_user_new_d where datestr='20130918';
```

注：还可以按来源地域维度、访客终端维度等计算

4. 访客 Visit 分析（点击流模型）

4.1. 回头/单次访客统计

需求：查询今日所有回头访客及其访问次数。

visit表

sessionid	userid	inTime	outTime	inPage	outPage	pageCnts	.....
s001	usera	13:01:50	13:08:20	/a	/e	5	
s002	userb	13:01:50	13:08:20	/a	/e	5	
s003	userc	13:01:50	13:08:20	/a	/e	5	
s004	usera	13:31:50	13:38:20	/a	/f	6	

实现思路：上表中出现次数>1 的访客，即回头访客；反之，则为单次访客。

```
drop table dw_user_returning;
create table dw_user_returning(
day string,
remote_addr string,
acc_cnt string)
partitioned by (datestr string);

insert overwrite table dw_user_returning partition(datestr='20130918')
select tmp.day,tmp.remote_addr,tmp.acc_cnt
from
(select '20130918' as day,remote_addr,count(session) as acc_cnt from ods_click_stream_visit group by remote_addr) tmp
where tmp.acc_cnt>1;
```

4.2. 人均访问频次

需求：统计出每天所有用户访问网站的平均次数（visit）

总 visit 数/去重总用户数

```
select sum(pagevisits)/count(distinct remote_addr) from ods_click_stream_visit where datestr='20130918';
```

5. 关键路径转化率分析（漏斗模型）

5.1. 需求分析

转化：在一条指定的业务流程中，各个步骤的完成人数及相对上一个步骤的百分比。

step	numbs	rate	
1	10000	100%	
2	4000	40%	40%
3	2000	20%	50%
4	1000	10%	50%
5	500	5%	50%
6	200	2%	40%

5.2. 模型设计

定义好业务流程中的页面标识，下例中的步骤为：

Step1、 /item

Step2、 /category

Step3、 /index

Step4、 /order

5.3. 开发实现

- 查询每一个步骤的总访问人数

```
--查询每一步人数存入 dw_oute_numbs
create table dw_oute_numbs as
select 'step1' as step,count(distinct remote_addr)  as numbs from ods_click_pageviews where datestr='20130920'
and request like '/item%'
union
select 'step2' as step,count(distinct remote_addr)  as numbs from ods_click_pageviews where datestr='20130920'
and request like '/category%'
union
select 'step3' as step,count(distinct remote_addr)  as numbs from ods_click_pageviews where datestr='20130920'
and request like '/order%'
union
select 'step4' as step,count(distinct remote_addr)  as numbs from ods_click_pageviews where datestr='20130920'
and request like '/index%';
```

注：UNION 将多个 SELECT 语句的结果集合并为一个独立的结果集。

- 查询每一步骤相对于路径起点人数的比例

思路：级联查询，利用自 join

```
--dw_oute_numbs 跟自己 join
select rn.step as rnstep,rn.numbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs  from dw_oute_numbs rn
inner join
dw_oute_numbs rr;
```

```
--每一步的人数/第一步的人数==每一步相对起点人数比例
select tmp.rnstep,tmp.rnnumbs/tmp.rrnumbs as ratio
from
(
select rn.step as rnstep,rn.numbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs  from dw_oute_numbs rn
inner join
dw_oute_numbs rr) tmp
where tmp.rrstep='step1';
```

- 查询每一步骤相对于上一步骤的漏出率

```
--自 join 表过滤出每一步跟上一步的记录
select rn.step as rnstep,rn.numbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs  from dw_oute_numbs rn
inner join
dw_oute_numbs rr
where cast(substr(rn.step,5,1) as int)=cast(substr(rr.step,5,1) as int)-1;
```

```
select tmp.rrstep as step,tmp.rrnumbs/tmp.rnnumbs as leakage_rate
from
(
select rn.step as rnstep,rn.numbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs  from dw_oute_numbs rn
inner join
dw_oute_numbs rr) tmp
where cast(substr(tmp.rnstep,5,1) as int)=cast(substr(tmp.rrstep,5,1) as int)-1;
```

- 汇总以上两种指标

```
select abs.step,abs.numbs,abs.rate as abs_ratio,rel.rate as leakage_rate
from
(
select tmp.rnstep as step,tmp.rnnumbs as numbs,tmp.rnnumbs/tmp.rrnumbs as rate
```

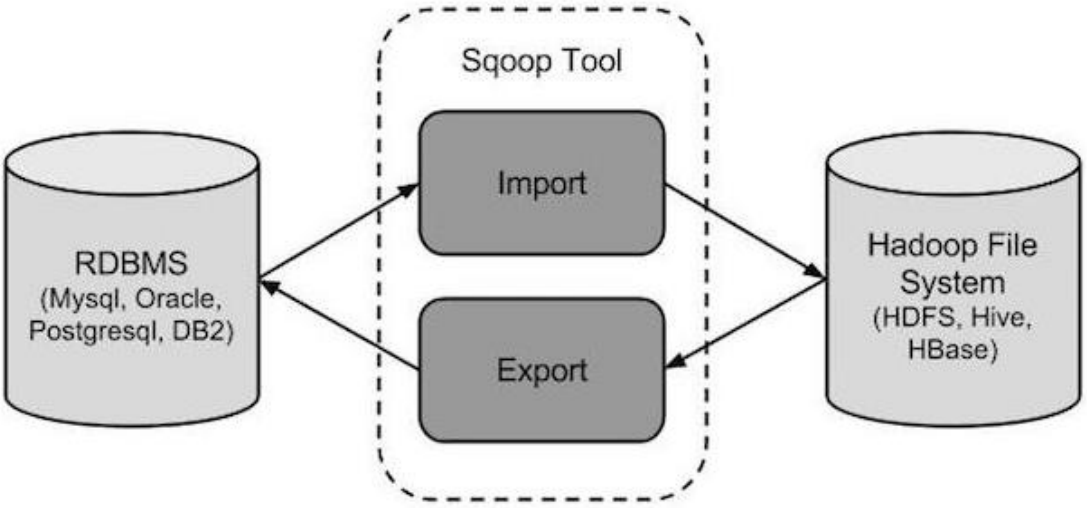
```
from
(
select rn.step as rnstep,rn.numbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs    from dw_oute_numbs rn
inner join
dw_oute_numbs rr) tmp
where tmp.rrstep='step1'
) abs
left outer join
(
select tmp.rrstep as step,tmp.rrnumbs/tmp.rnnumbs as rate
from
(
select rn.step as rnstep,rn.numbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs    from dw_oute_numbs rn
inner join
dw_oute_numbs rr) tmp
where cast(substr(tmp.rnstep,5,1) as int)=cast(substr(tmp.rrstep,5,1) as int)-1
) rel
on abs.step=rel.step;
```

五、 模块开发----结果导出

1. Apache Sqoop

Sqoop 是 Hadoop 和关系数据库服务器之间传送数据的一种工具。它是用来从关系数据库如：MySQL，Oracle 到 Hadoop 的 HDFS，并从 Hadoop 的文件系统导出数据到关系数据库。由 Apache 软件基金会提供。

Sqoop：“SQL 到 Hadoop 和 Hadoop 到 SQL”。



Sqoop 工作机制是将导入或导出命令翻译成 mapreduce 程序来实现。

在翻译出的 mapreduce 中主要是对 inputformat 和 outputformat 进行定制。

sqoop 安装

安装 sqoop 的前提是已经具备 java 和 hadoop 的环境。

最新稳定版： 1.4.6

配置文件修改：

```
cd $SQOOP_HOME/conf
mv sqoop-env-template.sh sqoop-env.sh
vi sqoop-env.sh
export HADOOP_COMMON_HOME=/root/apps/hadoop/
export HADOOP_MAPRED_HOME=/root/apps/hadoop/
```

export HIVE_HOME=/root/apps/hive

加入 mysql 的 jdbc 驱动包

cp /hive/lib/mysql-connector-java-5.1.28.jar $SQOOP_HOME/lib/

验证启动

```
bin/sqoop list-databases --connect jdbc:mysql://localhost:3306/ --username root --password hadoop
```

本命令会列出所有 mysql 的数据库。

到这里，整个 Sqoop 安装工作完成。

2. Sqoop 导入

“导入工具”导入单个表从 RDBMS 到 HDFS。表中的每一行被视为 HDFS 的记录。所有记录都存储为文本文件的文本数据（或者 Avro、sequence 文件等二进制数据）。

下面的语法用于将数据导入 HDFS。

```
$ sqoop import (generic-args) (import-args)
```

Sqoop 测试表数据

在 mysql 中创建数据库 userdb，然后执行参考资料中的 sql 脚本：

创建三张表: emp emp_add emp_conn。

2.1. 导入 mysql 表数据到 HDFS

下面的命令用于从 MySQL 数据库服务器中的 emp 表导入 HDFS。

```
bin/sqoop import \
--connect jdbc:mysql://node-21:3306/sqoopdb \
--username root \
--password hadoop \
--target-dir /sqoopresult \
--table emp --m 1
```

其中--target-dir 可以用来指定导出数据存放至 HDFS 的目录；

mysql jdbc url 请使用 ip 地址。

为了验证在 HDFS 导入的数据，请使用以下命令查看导入的数据：

```
hdfs dfs -cat /sqoopresult/part-m-00000
```

可以看出它会用逗号,分隔 emp 表的数据和字段。

```
1201,gopal,manager,50000,TP
1202,manisha,Proof reader,50000,TP
1203,khalil,php dev,30000,AC
1204,prasanth,php dev,30000,AC
1205,kranthi,admin,20000,TP
```

2.2. 导入 mysql 表数据到 HIVE

将关系型数据的表结构复制到 hive 中

```
bin/sqoop create-hive-table \
--connect jdbc:mysql://node-21:3306/sqoopdb \
--table emp_add \
--username root \
--password hadoop \
--hive-table test.emp_add_sp
```

其中：

--table emp_add 为 mysql 中的数据库 sqoopdb 中的表。

--hive-table emp_add_sp 为 hive 中新建的表名称。

从关系数据库导入文件到 hive 中

```
bin/sqoop import \
--connect jdbc:mysql://node-21:3306/sqoopdb \
--username root \
--password hadoop \
--table emp_add \
--hive-table test.emp_add_sp \
--hive-import \
--m 1
```

2.3. 导入表数据子集

--where 可以指定从关系数据库导入数据时的查询条件。它执行在各自的数据库服务器相应的 SQL 查询，并将结果存储在 HDFS 的目标目录。

```
bin/sqoop import \
--connect jdbc:mysql://node-21:3306/sqoopdb \
--username root \
--password hadoop \
--where "city ='sec-bad'" \
```

```
--target-dir /wherequery \
--table emp_add --m 1
```

复杂查询条件：

```
bin/sqoop import \
--connect jdbc:mysql://node-21:3306/sqoopdb \
--username root \
--password hadoop \
--target-dir /wherequery12 \
--query 'select id,name,deg from emp WHERE  id>1203 and $CONDITIONS' \
--split-by id \
--fields-terminated-by '\t' \
--m 1
```

2.4. 增量导入

增量导入是仅导入新添加的表中的行的技术。

--check-column (col)	用来作为判断的列名，如 id
--incremental (mode)	append：追加，比如对大于 last-value 指定的值之后的记录进行追加导入。lastmodified：最后的修改时间，追加 last-value 指定的日期之后的记录
--last-value (value)	指定自从上次导入后列的最大值（大于该指定的值），也可以自己设定某一值

假设新添加的数据转换成 emp 表如下：

```
1206, satish p, grp des, 20000, GR
```

下面的命令用于在 EMP 表执行增量导入：

```
bin/sqoop import \
--connect jdbc:mysql://node-21:3306/sqoopdb \
--username root \
--password hadoop \
--table emp --m 1 \
--incremental append \
--check-column id \
--last-value 1205
```

3. Sqoop 导出

将数据从 HDFS 导出到 RDBMS 数据库导出前，目标表必须存在于目标数据库中。

默认操作是从将文件中的数据使用 INSERT 语句插入到表中，更新模式下，是生成 UPDATE 语句更新表数据。

以下是 export 命令语法：

$ sqoop export (generic-args) (export-args)

3.1. 导出 HDFS 数据到 mysql

数据是在 HDFS 中“emp/”目录的 emp_data 文件中：

```
1201,gopal,manager,50000,TP
1202,manisha,preader,50000,TP
1203,kalil,php dev,30000,AC
1204,prasanth,php dev,30000,AC
1205,kranthi,admin,20000,TP
1206,satishp,grpdes,20000,GR
```

首先需要手动创建 mysql 中的目标表：

```
mysql> USE sqoopdb;
mysql> CREATE TABLE employee (
    id INT NOT NULL PRIMARY KEY,
    name VARCHAR(20),
    deg VARCHAR(20),
    salary INT,
    dept VARCHAR(10));
```

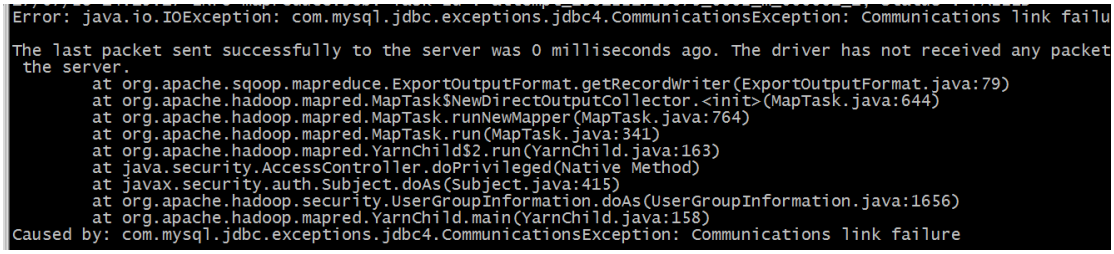
然后执行导出命令：

```
bin/sqoop export \
--connect jdbc:mysql://node-21:3306/sqoopdb \
--username root \
--password hadoop \
--table employee \
--export-dir /emp/emp_data
```

还可以用下面命令指定输入文件的分隔符

```
--input-fields-terminated-by '\t'
```

如果运行报错如下:



则需要把 localhost 更改为 ip 或者域名。

示例如下，将点击流模型表导出到 mysql

```
sqoop export \
--connect jdbc:mysql://hdp-node-01:3306/webdb --username root --password root    \
--table click_stream_visit    \
--export-dir /user/hive/warehouse/dw_click.db/click_stream_visit/datestr=2013-09-18 \
--input-fields-terminated-by '\001'
```

<更多表的导出，可参照修改>

六、 模块开发----工作流调度

整个项目的数据按照处理过程，从数据采集到数据分析，再到结果数据的导出，一系列的任务可以分割成若干个 azkaban 的 job 单元，然后由工作流调度器调度执行。

调度脚本的编写难点在于 shell 脚本。但是一般都是有固定编写模式。大家可以参考资料中的脚本进行编写。大体框架如下：

```
#!/bin/bash

#set java env

#set hadoop env

#设置一些主类、目录等常量

#获取时间信息

#shell 主程序、结合流程控制（if....else）去分别执行 shell 命令。
```

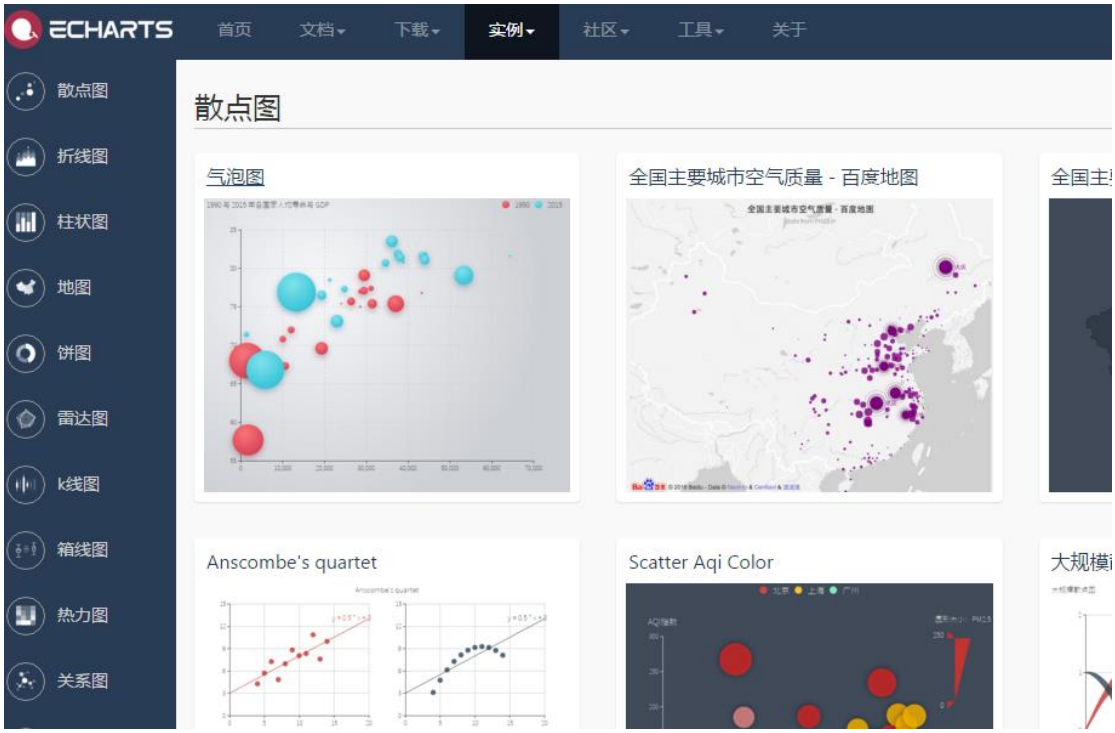
更多工作流及 hql 脚本定义见参考资料。

七、 模块开发----数据可视化

1. Echarts 介绍

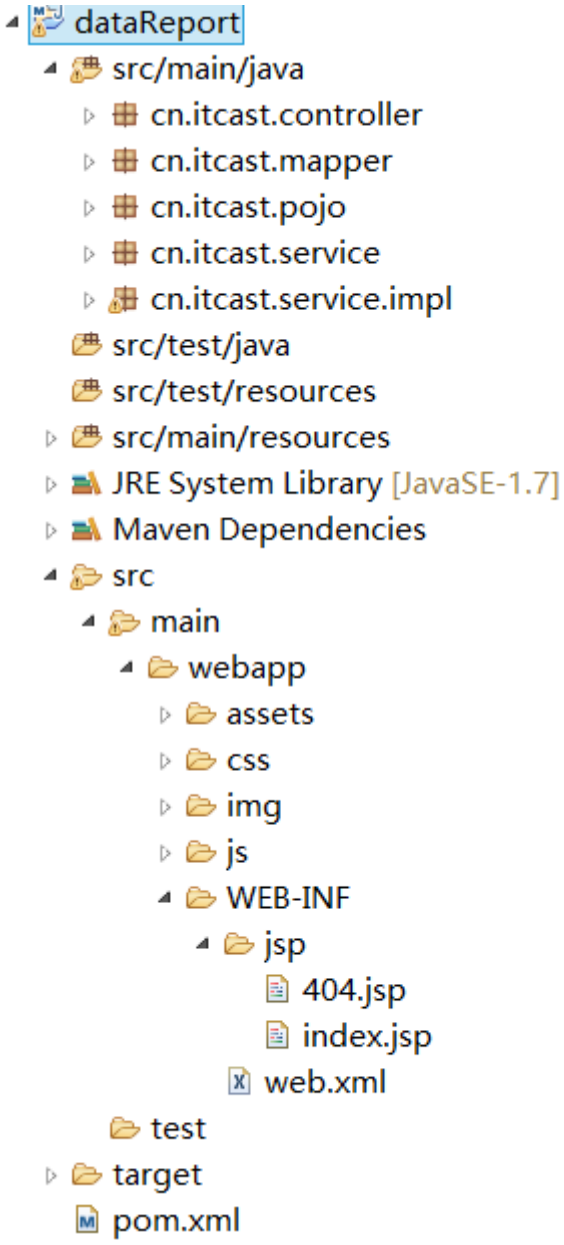
ECharts 是一款由百度前端技术部开发的，基于 Javascript 的数据可视化图表库，提供直观，生动，可交互，可个性化定制的数据可视化图表。

提供大量常用的**数据可视化图表**，底层基于 ZRender（一个全新的轻量级 canvas 类库），创建了坐标系，图例，提示，工具箱等基础组件，并在此上构建出折线图（区域图）、柱状图（条状图）、散点图（气泡图）、饼图（环形图）、K 线图、地图、力导向布局图以及和弦图，同时支持任意维度的堆积和多图表混合展现。



2. Web 程序工程结构

本项目是个纯粹的 JavaEE 项目，基于 ssm 的框架整合构建。使用 maven 的 tomcat 插件启动项目。



3. 感受 Echarts—简单入门

3.1. 下载 Echarts

从官网下载界面选择你需要的版本下载，根据开发者功能和体积上的需求，提供了不同打包的下载，如果在体积上没有要求，可以直接下载完整版本。**开发环境建议下载源代码版本，包含了常见的错误提示和警告。**

3.2. 页面引入 Echarts

ECharts 3 开始只需要像普通的 JavaScript 库一样用 script 标签引入即可。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <!-- 引入 ECharts 文件 -->
    <script src="echarts.min.js"></script>
</head>
</html>
```

3.3. 绘制一个简单的图表

在绘图前我们需要为 ECharts 准备一个具备高宽的 DOM 容器：

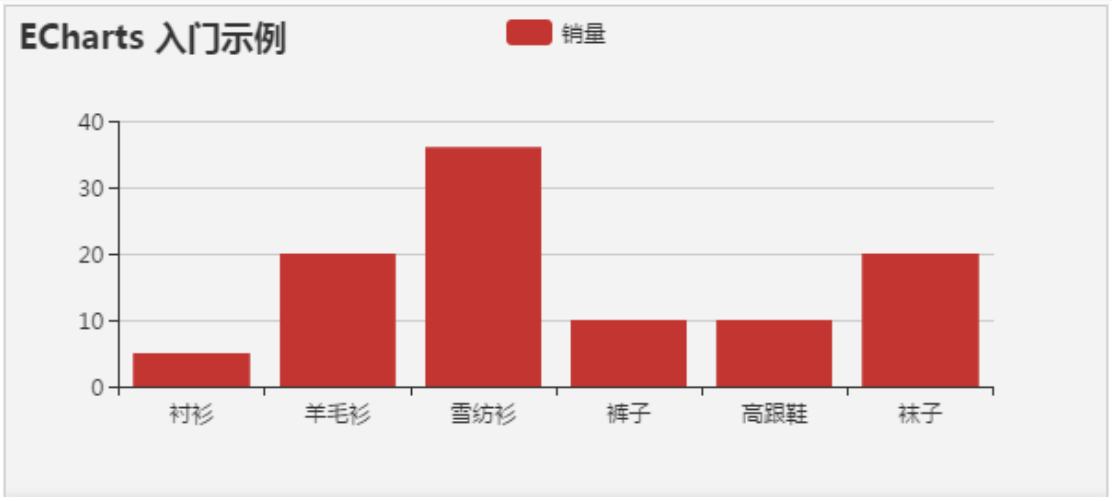
```
<body>
    <!-- 为 ECharts 准备一个具备大小（宽高）的 DOM -->
    <div id="main" style="width: 600px;height:400px;"></div>
</body>
```

然后就可以通过 echarts.init 方法初始化一个 echarts 实例并通过 setOption 方法生成一个简单的柱状图，下面是完整代码。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>ECharts</title>
    <!-- 引入 echarts.js -->
    <script src="echarts.min.js"></script>
</head>
```

```
<body>
    <!-- 为 ECharts 准备一个具备大小（宽高）的 Dom -->
    <div id="main" style="width: 600px;height:400px;"></div>
    <script type="text/javascript">
        // 基于准备好的 dom，初始化 echarts 实例
        var myChart = echarts.init(document.getElementById('main'));
        // 指定图表的配置项和数据
        var option = {
            title: {
                text: 'ECharts 入门示例'
            },
            tooltip: {},
            legend: {
                data:['销量']
            },
            xAxis: {
                data: ["衬衫","羊毛衫","雪纺衫","裤子","高跟鞋","袜子"]
            },
            yAxis: {},
            series: [{
                name: '销量',
                type: 'bar',
                data: [5, 20, 36, 10, 10, 20]
            }]
        };
        // 使用刚指定的配置项和数据显示图表。
        myChart.setOption(option);
    </script>
</body>
</html>
```

不出意外的话你就可以看见如下的图表：



4. 数据可视化的展现

课程中带领大家去实现一个从后端到前端展示数据的过程。其他图表类型都是大同小异，大家重点掌握异步数据加载的方法。

几个开发细节注意点：

4.1. Mybatis example 排序问题

example.setOrderByClause("`dateStr` ASC");

查询结果便可以根据 dataStr 字段正序排列（从小到大）

4.2. Echarts 前端数据格式问题

注意，当异步加载数据的时候，前端一般需要的是数据格式是数组。一定要对应上。在这里我们可以使用 Java Bean 封装数据，然后转换成 json 扔到前端，对应上相应的字段即可。

```
ObjectMapper om = new ObjectMapper();
beanJson = om.writeValueAsString(bean);
```

4.3. Controller 返回的 json

```
@RequestMapping(value="/xxxx",produces="application/json;charset=UTF-8")
@ResponseBody
```