漏斗模型（路径转化）

归总

业务分析

# 每一步相对于上一步的转化率

要得到一列是一步,另一列是上一步这样就好说了

2.每一步相对于第一步的转化率

要的到第一步,对应的是其他的步骤就好了

-- 求两个指标：

-- 第一个指标：每一步相对于第一步的转化率

-- 先把每一步的访问的人数求出

-- 自关联自己,求出数据

-- 第二个指标：每一步相对于上一步的转化率

-- 使用模型生成的数据，可以满足我们的转化率的求取

load data inpath '/weblog/clickstream/pageviews/click-part-r-00000' overwrite into table ods\_click\_pageviews partition(datestr='20130920');

----------------------------------------------------------

---1、查询每一个步骤的总访问人数

-- 1.1根据下面给出的具体的步骤,找

Step1、 /item 1000 相对上一步 相对第一步

Step2、 /category 800 0.8 0.8

Step3、 /index 500 0.625 0.5

Step4、 /order 100 0.2 0.1

create table dw\_oute\_numbs as

select 'step1' as step,count(distinct remote\_addr) as numbs from ods\_weblog\_origin

where datestr='20130920'

and request like '/item%'

union all

select 'step2' as step,count(distinct remote\_addr) as numbs from ods\_weblog\_origin

where datestr='20130920'

and request like '/category%'

union all

select 'step3' as step,count(distinct remote\_addr) as numbs from ods\_weblog\_origin where datestr='20130920' and request like '/order%'

union all

select 'step4' as step,count(distinct remote\_addr) as numbs from ods\_weblog\_origin where datestr='20130920' and request like '%index%';

+---------------------+----------------------+--+

| dw\_oute\_numbs.step | dw\_oute\_numbs.numbs |

+---------------------+----------------------+--+

| step1 | 1029 |

| step2 | 1029 |

| step3 | 1028 |

| step4 | 1018 |

+---------------------+----------------------+--+

----------------------------------------------------------------------------

--2、查询每一步骤相对于路径起点人数的比例

--级联查询，自己跟自己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;

-- 自我实现 : 每一步相对于第一步的转化率

-- 1. 自join后结果如下图所示：

SELECT a.step aa,a.numbs an, b.step bs, b.numbs bn

FROM dw\_oute\_numbs a

INNER JOIN

dw\_oute\_numbs b

+--------+-------+--------+-------+--+

| aa | an | bs | bn |

+--------+-------+--------+-------+--+

| step1 | 1029 | step1 | 1029 |

| step2 | 1029 | step1 | 1029 |

| step3 | 1028 | step1 | 1029 |

| step4 | 1018 | step1 | 1029 |

| step1 | 1029 | step2 | 1029 |

| step2 | 1029 | step2 | 1029 |

| step3 | 1028 | step2 | 1029 |

| step4 | 1018 | step2 | 1029 |

| step1 | 1029 | step3 | 1028 |

| step2 | 1029 | step3 | 1028 |

| step3 | 1028 | step3 | 1028 |

| step4 | 1018 | step3 | 1028 |

| step1 | 1029 | step4 | 1018 |

| step2 | 1029 | step4 | 1018 |

| step3 | 1028 | step4 | 1018 |

| step4 | 1018 | step4 | 1018 |

+--------+-------+--------+-------+--+

-- 自我实现

-- 2. 通过观察我们如果能在同一张表中取到上图中如下的数据就好办了

+--------+-------+--------+-------+--+

| aa | an | bs | bn |

+--------+-------+--------+-------+--+

| step1 | 1029 | step1 | 1029 |

| step2 | 1029 | step1 | 1029 |

| step3 | 1028 | step1 | 1029 |

| step4 | 1018 | step1 | 1029 |

select tempTab.rnnumbs/tempTab.rrnumbs

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 where rr.step = 'step1'

) tempTab;

+---------+----------+---------+----------+--+

| rnstep | rnnumbs | rrstep | rrnumbs |

+---------+----------+---------+----------+--+

| step1 | 1029 | step1 | 1029 |

| step2 | 1029 | step1 | 1029 |

| step3 | 1028 | step1 | 1029 |

| step4 | 1018 | step1 | 1029 |

+---------+----------+---------+----------+--+

-- 讲师实现

--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';

-- 自我实现

--

SELECT tmp.aa, tmp.an/tmp.bn

FROM(

SELECT a.step aa,a.numbs an,b.step bs,b.numbs bn

FROM dw\_oute\_numbs a

right JOIN

(SELECT \*

FROM dw\_oute\_numbs WHERE step='step1')b

) tmp

+---------+----------+---------+----------+--+

| rnstep | rnnumbs | rrstep | rrnumbs |

+---------+----------+---------+----------+--+

| step1 | 1029 | step1 | 1029 |

| step2 | 1029 | step1 | 1029 |

| step3 | 1028 | step1 | 1029 |

| step4 | 1018 | step1 | 1029 |

+---------+----------+---------+----------+--+

+---------+---------------------+--+

| tmp.aa | \_c1 |

+---------+---------------------+--+

| step1 | 1.0 |

| step2 | 1.0 |

| step3 | 0.9990281827016521 |

| step4 | 0.989310009718173 |

+---------+---------------------+--+

==============================================================

==============================================================

--------------------------------------------------------------------------------

# 查询每一步骤相对于上一步骤的漏出率

--首先通过自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;

-- 自我实现

-- 通过观察自关联的总表,如果出现如下的表的话就好办了

+--------+-------+--------+-------+--+

| step1 | 1029 | step2 | 1029 |

| step2 | 1029 | step3 | 1028 |

| step3 | 1028 | step4 | 1018 |

+--------+-------+--------+-------+--+

-- 自我代码实现

SELECT a.step aa,a.numbs an, b.step bs, b.numbs bn

FROM dw\_oute\_numbs a

INNER JOIN

dw\_oute\_numbs b

-- 经过观察我们需要的一条数据是: 有前一步和上一步,还有对应的访问人数

WHERE CAST(SUBSTR(b.step,5,1)AS INT)-1= CAST(SUBSTR(a.step,5,1)AS INT)

-- 这里cast()函数是类型转换函数,语法如上,这里是吧string-->int

-- 这里substr()函数是字符串截取的函数,5表示从1开始数取第五个位子后的一个

select newTable.rnnumbs/newTable.rrnumbs from (

select \* 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

) tmpTable

where cast(substr(tmpTable.rrStep,5,1) as int ) = cast(substr(tmpTable.rnstep,5,1) as int )-1

) newTable

-- 教案例子

where temTable.rrstep.截串 >= temTable.rnstep.截串

-- 注意：cast为hive的内置函数，主要用于类型的转换

-- 用例：

select cast(1 as float);

select cast('2018-06-22' as date);

+---------+----------+---------+----------+--+

| rnstep | rnnumbs | rrstep | rrnumbs |

+---------+----------+---------+----------+--+

| step1 | 1029 | step2 | 1029 |

| step2 | 1029 | step3 | 1028 |

| step3 | 1028 | step4 | 1018 |

+---------+----------+---------+----------+--+

--然后就可以非常简单的计算出每一步相对上一步的漏出率

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;

-- 自我实现

-- 通过上一步的实现这里就非常的简单了,就是两个字段做除法就可以了,得到的数据如下

+--------------+---------------------+--+

| \_c0 | \_c1 |

+--------------+---------------------+--+

| step2-step1 | 1.0 |

| step3-step2 | 0.9990281827016521 |

| step4-step3 | 0.9902723735408561 |

+--------------+---------------------+--+

-----------------------------------------------------------------------------------

--4、汇总以上两种指标

select abs.step,abs.numbs,abs.rate as abs\_ratio,rel.rate as leakage\_rate

from

(

-- 这个是得到每步相对于第一步的转化率 这个查出来作为一张临时表abs

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 -- 直接水平连接对应的两张表

(

-- 这个得到每步相对于上一步的转化率,这个查出来作为一张临时表rel

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;

-- 最终的结果如下

-- 第4步相对于第3步的转化率是0.989310009718173,相对于第1步的转化率是0.9902723735408561

+-----------+------------+---------------------+---------------------+--+

| abs.step | abs.numbs | abs\_ratio | leakage\_rate |

+-----------+------------+---------------------+---------------------+--+

| step1 | 1029 | 1.0 | NULL |

| step2 | 1029 | 1.0 | 1.0 |

| step3 | 1028 | 0.9990281827016521 | 0.9990281827016521 |

| step4 | 1018 | 0.989310009718173 | 0.9902723735408561 |

+-----------+------------+---------------------+---------------------+--+