

```
第二节——HiveSQL基础技能2
  课堂目标
  表连接
      [inner join] (25min)
        举例说明:
        练习:
          1、在2019年购买后又退款的用户:
          2、在2017年和2018年都购买的用户:
          3、在2017年、2018年、2019都有交易的用户:
      [left join] (25min)
        举例说明:
        练习:
          1、在2019年购买,但是没有退款的用户:
          2、在2019年购买用户的学历分布:
          3、在2017和2018年都购买,但是没有在2019年购买的用户:
      [full join] (10min)
        举例说明:
        练习:
          1、user_list_1和user_list_2的所有用户:
      [union all] (20min)
        举例说明:
        练习:
          1、2017-2019年有交易的所有用户数:
          2、2019年每个用户的支付和退款金额汇总:
  重点练习(20min)
  总结
  作业
```

第二节——HiveSQL基础技能2

课堂目标

- 1、掌握inner join
- 2、掌握left join
- 3、掌握full join和union all
- 4、掌握各种连接的组合

表连接

[inner join] (25min)

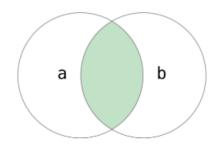






表1: user_list_1

user_id	user_name
10001	Abby
10002	Ailsa
10003	Alice
10004	Alina
10005	Allison
10006	Angelia

表2: user_list_2

user_id	user_name
10001	Abby
10003	Alice
10004	Alina
10007	Amanda
10008	Anne
10009	Ann

找出既在user_list_1也在user_list_2的用户:



```
1 --既在user_list_1也在user_list_2的用户--
2 SELECT *
3 FROM user_list_1 a JOIN user_list_2 b ON a.user_id=b.user_id;
```

```
10001 Abby 10001 Abby
10003 Alice 10003 Alice
10004 Alina 10004 Alina
Time taken: 26.062 seconds, Fetched: 3 row(s)
```

注意:

- 表连接时,必须进行重命名
- on后面使用的连接条件必须起到唯一键值的作用
- inner可省略不写,效果一样

练习:

user_trade列名	举例
user_name	Amy, Dennis
piece	购买数量
price	价格
pay_amount	支付金额
goods_category	food, clothes, book, computer, electronics, shoes
pay_time	1323308943,时间戳
dt	partition,'yyyy-mm-dd'

user_refund列名	举例
user_name	Amy, Dennis
refund_piece	退款件数
refund_amount	退款金额
refund_time	1323308943,时间戳
dt	partition,'yyyy-mm-dd'

1、在2019年购买后又退款的用户:

```
--在2019年购买后又退款的用户--
 2
    SELECT a.user_name
3
    FROM
 4
          (SELECT distinct user name
5
          FROM user_trade
          WHERE year(dt)=2019)a
 6
        JOIN
          (SELECT distinct user_name
8
9
          FROM user_refund
          WHERE year(dt)=2019)b on a.user name=b.user name;
10
```

常见错误: 没有写别名

```
SELECT user_name
1
2
   FROM
3
          (SELECT distinct user name
4
         FROM user trade
5
         WHERE year(dt)=2019)a
       JOIN
6
7
          (SELECT distinct user_name
         FROM user_refund
8
9
          WHERE year(dt)=2019)b on a.user name=b.user name;
```

如果不进行去重的话,会怎么样?

```
SELECT count(a.user name),
           count(distinct a.user_name)
2
3
    FROM
4
          (SELECT user name
          FROM user_trade
5
          WHERE year(dt)=2019)a
 6
 7
        JOIN
8
           (SELECT user name
9
          FROM user refund
          WHERE year(dt)=2019)b on a.user_name=b.user_name;
10
```

```
61 31
Time taken: 38.773 seconds, Fetched: 1 row(s)
```

!! 注意: 一定要先去重, 再做表连接, 养成良好习惯

PS: 虽然可以先连接后再去重,但是这么做的话,执行效率会低。



2、在2017年和2018年都购买的用户:

```
--在2017年和2018年都购买的用户--
2
    SELECT a.user name
3
    FROM
          (SELECT distinct user name
4
          FROM user trade
         WHERE year(dt)=2017)a
6
7
          (SELECT distinct user_name
8
9
          FROM user trade
10
          WHERE year(dt)=2018)b on a.user_name=b.user_name;
```

trade_2017&2018&2019列名	举例
user_name	Amy, Dennis
amount	金额
trade_time	交易时间,2017-02-05 06:31:50

3、在2017年、2018年、2019都有交易的用户:

```
--在2017年、2018年、2019都有交易的用户--
 2
    ##第一种写法
 3
    SELECT distinct a.user name
   FROM trade 2017 a
 5
    JOIN trade_2018 b on a.user_name=b.user_name
    JOIN trade 2019 c on b.user name=c.user name;
 6
 8
    ##第二种写法
9
    SELECT a.user_name
10
    FROM
11
          (SELECT distinct user name
12
          FROM trade_2017)a
13
        JOIN
          (SELECT distinct user name
          FROM trade_2018)b on a.user_name=b.user_name
15
16
          (SELECT distinct user_name
17
          FROM trade 2019)c on b.user name=c.user name;
18
```

- 在表的数据量级很大时,推荐第二种写法
- 为什么写a.user_name呢? b.user_name或者c.user_name可以吗? ——当然可以。

因为JOIN是取表的交集,所以不管取哪个表的这个字段都是一样的结果。

初学者常犯错误写法:

```
1 ----错误写法----
2 SELECT distinct a.user_name
3 FROM trade_2017 a
4 JOIN trade_2018 b
5 JOIN trade_2019 c on a.user_name=b.user_name=c.user_name;
```

[left join] (25min)

首先,我们先来看一下,对表1和表2进行左连接后,发生了什么。

```
1 | SELECT *
2 | FROM user_list_1 a LEFT JOIN user_list_2 b ON a.user_id=b.user_id;
```

```
Abby
                         Abby
10001
                10001
10002
        Ailsa
                NULL
                         NULL
10003
        Alice
                10003
                         Alice
10004
                10004
        Alina
                         Alina
10005
        Allison NULL
                         NULL
10006
        Angelia NULL
                         NULL
Time taken: 25.652 seconds, Fetched: 6 row(s)
```

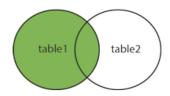
进行左连接后,以左边的表1为全集,返回能够匹配上的右边表2的匹配结果,没有匹配上的则显示NULL。

拓展:

right join:以右边的表为全集,返回能够匹配上的左表的匹配结果,没有匹配上的则显示NULL但其完全可以由left join改写出同样的结果,所以较少使用

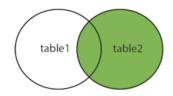
LEFT JOIN 关键字从左表(table1)返回所有的行,即使右表(table2)中没有匹配。如果右表中没有匹配,则结果为 NULL。

LEFT JOIN



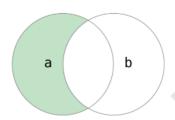
RIGHT JOIN 关键字从右表(table2)返回所有的行,即使左表(table1)中没有匹配。如果左表中没有匹配,则结果为 NULL。

RIGHT JOIN



举例说明:

如何取出,在user_list_1表中但是不在user_list_2的用户?



```
--在user_list_1表中但是不在user_list_2的用户--

SELECT a.user_id,

a.user_name

FROM user_list_1 a LEFT JOIN user_list_2 b ON a.user_id=b.user_id

WHERE b.user_id is null;
```



练习:

1、在2019年购买,但是没有退款的用户:

```
--在2019年购买,但是没有退款的用户--
2
    SELECT a.user_name
3
    FROM
4
          (SELECT distinct user_name
5
          FROM user trade
6
          WHERE year(dt)=2019)a
7
       LEFT JOIN
          (SELECT distinct user_name
8
9
          FROM user refund
          WHERE year(dt)=2019)b on a.user_name=b.user_name
10
   WHERE b.user_name is null;
11
```

2、在2019年购买用户的学历分布:

user_info 列名	举例	
user_id	10001,10002	
user_name	Amy, Dennis	
sex	[male, female]	
age	[13,70]	
city	beijing, shanghai	
firstactivetime	2019-04-19 15:40:00	
level	[1,10]	
extra1	string类型:{"systemtype":"ios","education":"master","marriage_status":"1","phonebrand":"iphone X"}	
extra2	map <string,string>类型: {"systemtype":"ios","education":"master","marriage_status":"1","phonebrand":"iphone X"}</string,string>	

```
--在2019年购买用户的学历分布--
 1
 2
    SELECT b.education,
 3
           count(a.user_name)
 4
    FROM
          (SELECT distinct user_name
 5
 6
          FROM user trade
 7
          WHERE year(dt)=2019)a
        LEFT JOIN
 8
9
          (SELECT user name,
                  get_json_object(extra1, '$.education') as education
10
11
          FROM user_info)b on a.user_name=b.user_name
12
    GROUP BY b.education;
```

```
bachelor 19
doctor 19
master 25
Time taken: 56.78 seconds, Fetched: 3 row(s)
```

注意: get_json_object(extra1, '\$.education')可以换成extra2['education']

3、在2017和2018年都购买,但是没有在2019年购买的用户:

trade_2017&2018&2019列名	举例
user_name	Amy, Dennis
amount	金额
trade_time	交易时间,2017-02-05 06:31:50

```
--在2017和2018年都购买,但是没有在2019年购买的用户--
2
    SELECT a.user_name
 3
    FROM
          (SELECT distinct user_name
4
          FROM trade_2017)a
5
        JOIN
 6
          (SELECT distinct user_name
7
          FROM trade 2018)b on a.user name=b.user name
8
9
       LEFT JOIN
10
          (SELECT distinct user_name
          FROM trade_2019)c on b.user_name=c.user_name
11
    WHERE c.user_name is null;
12
```

● a.user_name换成b.user_name也可以

这种写法行不行? ——可以的!

```
SELECT c.user_name

FROM

(SELECT distinct a.user_name

FROM trade_2017 a JOIN trade_2018 b on a.user_name=b.user_name)c

LEFT JOIN

(SELECT distinct user_name

FROM trade_2019)d on c.user_name=d.user_name

WHERE d.user_name is null;
```

```
SELECT distinct a.user_name

FROM trade_2017 a

JOIN trade_2018 b on a.user_name=b.user_name

LEFT JOIN trade_2019 c on b.user_name=c.user_name

WHERE c.user_name is null;
```

注意: 如果表比较小的时候,这样写影响不大。但是有分区的大表,这样写执行速度很慢。

[full join] (10min)

举例说明:

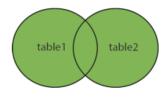
首先,我们先来看一下,对表1和表2进行全连接后,发生了什么。

```
1   SELECT *
2   FROM user_list_1 a FULL JOIN user_list_2 b ON a.user_id=b.user_id;
```

```
10001
        Abby
                 10001
                         Abby
10002
        Ailsa
                NULL
                         NULL
10003
        Alice
                 10003
                         Alice
10004
        Alina
                 10004
                         Alina
10005
        Allison NULL
                         NULL
10006
        Angelia NULL
                         NULL
NULL
        NULL
                         Amanda
                 10007
NULL
        NULL
                         Anne
                 10008
NULL
        NULL
                 10009
                         Ann
Time taken: 26.858 seconds, Fetched: 9 row(s)
```

FULL OUTER JOIN 关键字只要左表(table1)和右表(table2)其中一个表中存在匹配,则返回行。FULL OUTER JOIN 关键字结合了 LEFT JOIN 和 RIGHT JOIN 的结果。





练习:

1、user list 1和user list 2的所有用户:

```
1 --user_list_1和user_list_2的所有用户--
2 SELECT coalesce(a.user_name,b.user_name)
3 FROM user_list_1 a FULL JOIN user_list_2 b on a.user_id=b.user_id;
```

```
Abby
Ailsa
Alice
Alina
Allison
Angelia
Amanda
Anne
Anne
Time taken: 26.578 seconds, Fetched: 9 row(s)
```

拓展: coalesce是一个函数, (expression_1, expression_2, ..., expression_n)依次参考各参数表达式,遇到非null值即停止并返回该值。如果所有的表达式都是空值,最终将返回一个空值。

[union all] (20min)

表1: user_list_1

user_id	user_name
10001	Abby
10002	Ailsa
10003	Alice
10004	Alina
10005	Allison
10006	Angelia

表3: user_list_3

user_id	user_name
10290	Michael Kojkeloo C
10291	Avery
10292	Reilly
10293	Dillon
10294	Walton

union all: 联合所有

举例说明:

将user_list_1和user_list_3合并在一起:

```
1  SELECT user_id,
2          user_name
3  FROM user_list_1
4  UNION ALL
5  SELECT user_id,
6          user_name
7  FROM user_list_3;
```

```
10001
        Abby
10002
        Ailsa
10003
        Alice
10004
        Alina
10005
        Allison
10006
        Angelia
10290
        Michael
10291
        Avery
10292
        Reilly
10293
        Dillon
        Walton
10294
Time taken: 22.26 seconds, Fetched: 11 row(s)
```

注意:

- 字段名称必须一致
- 字段顺序必须一致
- 没有连接条件

```
----错误写法----
1
2
   SELECT user_name,
3
          user id
   FROM user_list_1
4
5
   UNION ALL
6
   SELECT user id,
7
          user name
8
   FROM user_list_3;
```

```
Abby
        10001
Ailsa
        10002
Alice
        10003
Alina
        10004
Allison 10005
Angelia 10006
10290
        Michael
10291
        Avery
        Reilly
10292
10293
        Dillon
10294
        Walton
Time taken: 17.473 seconds, Fetched: 11 row(s)
```

练习:

1、2017-2019年有交易的所有用户数:

trade_2017&2018&2019列名	举例
user_name	Amy, Dennis
amount	金额
trade_time	交易时间,2017-02-05 06:31:50

```
--2017-2019年有交易的所有用户数--
 2
    ##写法一
 3
    SELECT count(distinct a.user_name),
 4
           count(a.user_name)
 5
    FROM
 6
          SELECT user_name
 8
          FROM trade_2017
        UNION ALL
9
10
          SELECT user name
          FROM trade 2018
11
12
        UNION ALL
          SELECT user_name
13
          FROM trade_2019)a;
15
16
17
    ##写法二
    SELECT count(distinct a.user_name),
18
19
           count(a.user_name)
20
    FROM
21
          SELECT user_name
22
          FROM trade_2017
23
24
        UNION
25
          SELECT user_name
26
          FROM trade_2018
27
        UNION
28
          SELECT user name
          FROM trade_2019)a;
29
```

● UNION ALL和UNION的区别:

对比	UNION ALL	UNION
对重复结果的处 理	不会去除重复记录	在进行表连接后会筛选掉重复的记 录
对排序的处理	只是简单的将两个结果合并后就返 回	将会按照字段的顺序进行排序
效率	更快	更慢
总述	不去重不排序	去重且排序

PS: 如果表很大时推荐先去重,再进行union all。

● 常见错误——没有对union all后的表进行重命名:

```
----错误写法----
    SELECT count(distinct user name)
 3
    FROM
 4
5
          SELECT user_name
          FROM trade 2017
 6
        UNION ALL
 7
8
          SELECT user_name
          FROM trade 2018
9
10
        UNION ALL
          SELECT user_name
11
          FROM trade_2019);
12
```

● 常见错误——直接对表进行UNION ALL:

```
---错误写法---

SELECT count(distinct user_name)

FROM trade_2017

UNION ALL trade_2018

UNION ALL trade_2019;
```

2、2019年每个用户的支付和退款金额汇总:

user_trade列名	举例
user_name	Amy, Dennis
piece	购买数量
price	价格
pay_amount	支付金额
goods_category	food, clothes, book, computer, electronics, shoes
pay_time	1323308943,时间戳
dt	partition,'yyyy-mm-dd'

user_refund列名	举例		
user_name	Amy, Dennis koj keloo c		
refund_piece	退款件数		
refund_amount	退款金额		
refund_time	1323308943,时间戳		
dt	partition,'yyyy-mm-dd'		

```
--2019年每个用户的支付和退款金额汇总--
 2
    SELECT a.user_name,
 3
           sum(a.pay_amount),
           sum(a.refund_amount)
 4
 5
    FROM
 6
 7
          SELECT user_name,
 8
                  sum(pay_amount) as pay_amount,
 9
                  0 as refund_amount
10
          FROM user trade
          WHERE year(dt)=2019
11
12
          GROUP BY user_name
        UNION ALL
13
          SELECT user_name,
14
15
                  0 as pay_amount,
16
                  sum(refund_amount) as refund_amount
17
          FROM user_refund
18
          WHERE year(dt)=2019
19
          GROUP BY user_name
20
        ) a
    GROUP BY a.user_name;
21
```

● 如何用full join来实现该题目?

```
SELECT coalesce(a.user_name,b.user_name),
 2
           a.pay_amount,
 3
           b.refund amount
 4
    FROM
 5
          (SELECT user_name,
 6
                  sum(pay amount) as pay amount
          FROM user trade
          WHERE year(dt)=2019
 8
9
          GROUP BY user_name)a
        FULL JOIN
1.0
           (SELECT user name,
11
                  sum(refund_amount) as refund_amount
12
          FROM user refund
13
          WHERE year(dt)=2019
14
15
          GROUP BY user_name)b on a.user_name=b.user_name;
```

与第一种写法有什么结果差异?——没有退款的人,退款金额显示NULL

如果用第二种写法,如何把NULL都变成0呢?

```
SELECT coalesce(a.user name, b.user name),
 2
           if(a.pay_amount is null,0,a.pay_amount),
           if(b.refund_amount is null,0,b.refund_amount)
 3
 4
    FROM
 5
         (SELECT user name,
 6
                  sum(pay_amount) as pay_amount
 7
          FROM user_trade
 8
          WHERE year(dt)=2019
          GROUP BY user_name)a
 9
10
        FULL JOIN
11
           (SELECT user_name,
                 sum(refund_amount) as refund_amount
12
          FROM user_refund
13
          WHERE year(dt)=2019
14
          GROUP BY user_name)b on a.user_name=b.user_name;
```

问题变形: 2019年每个支付用户的支付金额和退款金额

```
SELECT a.user_name,
 2
           a.pay_amount,
 3
           b.refund_amount
 4
    FROM
 5
          (SELECT user_name,
                  sum(pay_amount) as pay_amount
 6
 7
          FROM user_trade
 8
          WHERE year(dt)=2019
          GROUP BY user_name)a
9
10
        LEFT JOIN
11
          (SELECT user name,
                 sum(refund_amount) as refund_amount
12
          FROM user_refund
13
          WHERE year(dt)=2019
14
          GROUP BY user_name)b on a.user_name=b.user_name;
15
```

重点练习(20min)

1、首次激活时间在2017年,但是一直没有支付的用户年龄段分布(使用user_trade和user_info两个表)

user_info 列名	举例
user_id	10001,10002
user_name	Amy, Dennis
sex	[male, female]
age	[13,70]
city	beijing, shanghai
firstactivetime	2019-04-19 15:40:00
level	[1,10]
extra1	string类型:{"systemtype":"ios","education":"master","marriage_status":"1","phonebrand":"iphone X"}
extra2	map <string,string>类型: {"systemtype":"ios","education":"master","marriage_status":"1","phonebrand":"iphone X"}</string,string>

user_trade列名	举例		
user_name	Amy, Dennis		kaikeba.co
piece	购买数量		
price	价格		
pay_amount	支付金额		
goods_category	food, clothes, book, computer, electronics, shoes		
pay_time	1323308943,时间戳		
dt	partition,'yyyy-mm-dd'		

```
--首次激活时间在2017年,但是一直没有支付的用户年龄段分布--
 2
    SELECT a.age_level,
           count(a.user_name)
 3
    FROM
 4
 5
            (SELECT user_name,
                    case when age<20 then '20岁以下'
 6
                        when age>=20 and age<30 then '20-30岁'
 7
                        when age>=30 and age<40 then '30-40岁'
 8
9
                        else '40岁以上' end as age_level
10
            FROM user_info
            WHERE year(firstactivetime)=2019)a
11
          LEFT JOIN
12
13
            (SELECT distinct user name
14
            FROM user_trade
            WHERE dt>0)b on a.user_name=b.user_name
15
16
    WHERE b.user name is null
    GROUP BY a.age_level;
17
```

```
20-30岁 6
20岁以下 8
30-40岁 6
40岁以上 22
Time taken: 86.026 seconds, Fetched: 4 row(s)
```

常见错误——没有对子查询中的字段进行重命名:

```
----错误写法----
 2
    SELECT a.age,
 3
           count(a.user_name)
 4
    FROM
 5
            (SELECT user_name,
                    case when age<20 then '20岁以下'
 6
 7
                        when age>=20 and age<30 then '20-30岁'
 8
                        when age>=30 and age<40 then '30-40岁'
                        else '40岁以上' end
9
10
            FROM user info
            WHERE year(firstactivetime)=2019)a
11
          LEFT JOIN
12
            (SELECT distinct user_name
13
            FROM user_trade
14
15
            WHERE dt>0)b on a.user_name=b.user_name
    WHERE b.user_name is null
16
    GROUP BY a.age;
17
```

2、2018、2019年交易的用户,其激活时间段分布(使用trade_2018、trade_2019和user_info三个表)

trade_2017&2018&2019列名	举例
user_name	Amy, Dennis
amount	金额
trade_time	交易时间,2017-02-05 06:31:50

user_info 列名	举例
user_id	10001,10002
user_name	Amy, Dennis
sex	[male, female]
age	[13,70]
city	beijing, shanghai
firstactivetime	2019-04-19 15:40:00
level	[1,10]
extra1	string类型:{"systemtype":"ios","education":"master","marriage_status":"1","phonebrand":"iphone X"}
extra2	map <string,string>类型: {"systemtype":"ios","education":"master","marriage_status":"1","phonebrand":"iphone X"}</string,string>

```
--2018、2019年交易的用户,其激活时间段分布--
 2
    SELECT hour(firstactivetime),
 3
           count(a.user_name)
    FROM
 4
 5
          (
 6
            SELECT user name
 7
            FROM trade_2018
 8
          UNION
9
            SELECT user_name
10
            FROM trade 2019)a
          LEFT JOIN user_info b on a.user_name=b.user_name
11
12
    GROUP BY hour(firstactivetime);
```

```
10
         4
         5
5
6
         5
         6
         7
         15
10
         3
11
         5
12
         10
13
         6
14
         6
15
         4
16
         10
17
         10
18
         6
19
         13
20
         14
21
         12
22
         9
23
Time taken: 53.367 seconds, Fetched: 24 row(s)
```

总结

- 1. 在实际业务场景中,熟练选择JOIN、LEFT JOIN来解决具体问题
- 2. 区分好FULL JOIN和UNION ALL的使用场景
- 3. 在多表连接时,注意各种细节和业务逻辑
- 4. 避免常见错误

作业

作业1: 在2019年购买后又退款的用户性别分布(使用user_trade、user_refund)

作业2: 在2018年购买,但是没在2019年购买的用户城市城市分布(使用user_trade、user_refund)

作业3: 2017-2019年,有交易但是没退款的用户的手机品牌分布(使用trade_2017、trade_2018、

trade_2019 \ user_refund \ user_info)

