9.MySQL SQL多表连接和子查询√

9.0 多表连接的介绍

> 在关系数据库中，一个查询往往会涉及多个表，因为很少有数据库只有一个表，而如果大多查询只涉及到一个表的，那么那个表也往往低于第三范式，存在大量冗余和异。所以我们需要将多张表连接在一起。

连接(Join)就是一种把多个表连接成一个表的重要手段.

9.1 多表连接环境准备

1.创建两张练习表 a 和 b

创建a表表结构

```
mysql> create table a(id int,name varchar(20));
```

创建b表表结构

```
mysql> create table b(id int,addr varchar(20),telnum char(11),aid int);
```

2.插入表种数据

插入a表数据

```
mysql> insert into a values(1,'zs'),(2,'ls'),(3,'ww');
mysql> insert into a values(11,'a'),(22,'b'),(33,'c');
```

插入b表数据

```
mysql> insert into b values(1001,'bj','110',1),(1002,'hn','111',2),(1003,'hn','112',3),(1004,'zz','113',4);
```

查看a和b表

Bash | Copy

```
mysql> select * from a;
+------+------+
| id   | name |
+------+------+
|    1 | zs   |
|    2 | ls   |
|    3 | ww   |
|   11 | a    |
|   22 | b    |
|   33 | c    |
+------+------+
mysql> select * from b;
+------+------+--------+------+
| id   | addr | telnum | aid  |
+------+------+--------+------+
| 1001 | bj   | 110    |    1 |
| 1002 | hn   | 111    |    2 |
| 1003 | hn   | 112    |    3 |
| 1004 | zz   | 113    |    4 |
+------+------+--------+------+
```

9.2 多表连接算法类型

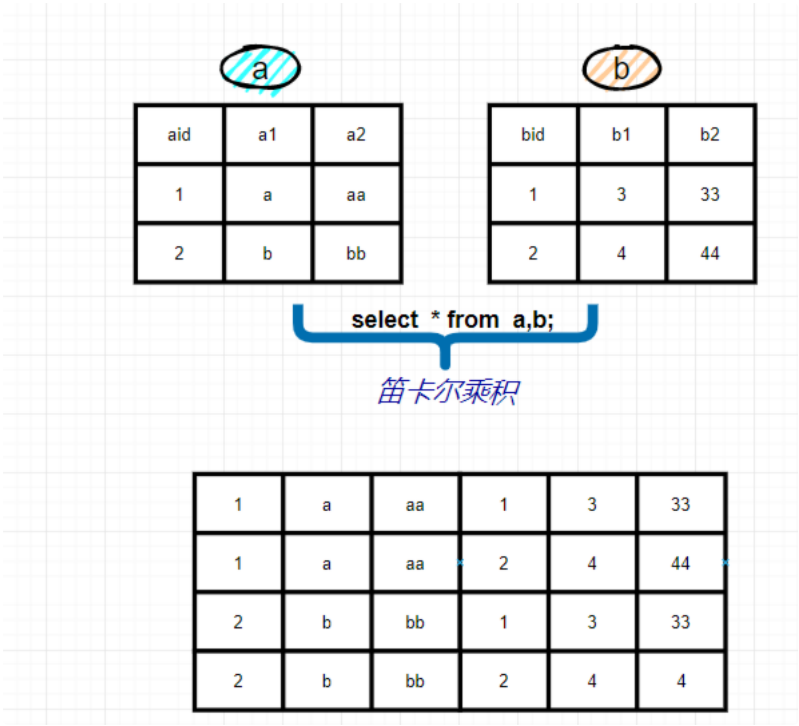
1.笛卡尔乘积

1.0 介绍

> 1.笛卡尔积在SQL中的实现方式既是交叉连接(Cross Join)。所有连接方式都会先生成临时笛卡尔积表，笛卡尔积是关系代数里的一个概念，表示两个表中的每一行数据任意组合,两个表连接即为笛卡尔积(交叉连接)
> 2.在实际应用中，笛卡尔积本身大多没有什么实际用处，只有在两个表连接时加上限制条件，才会有实际意义

`mysql> select * from a,b;`
`mysql > select * from a join b;`
就可以触发笛卡尔乘积

1.1原理图


a
aid
a1
a2
1
a
aa
2
b
bb
b
bid
b1
b2
1
3
33
2
4
44
select * from a,b;
笛卡尔乘积
1
a
aa
1
3
33
1
a
aa
2
4
44
2
b
bb
1
3
33
2
b
bb
2
4
4

1.2 语句执行

Bash | Copy

```
mysql> select * from a,b;
+------+------+------+------+---------+------+
| id   | name | id   | addr | telnum | aid  |
+------+------+------+------+---------+------+
|    1 | zs   | 1004 | zz   | 113    |    4 |
|    1 | zs   | 1003 | hn   | 112    |    3 |
|    1 | zs   | 1002 | hn   | 111    |    2 |
|    1 | zs   | 1001 | bj   | 110    |    1 |
|    2 | ls   | 1004 | zz   | 113    |    4 |
|    2 | ls   | 1003 | hn   | 112    |    3 |
|    2 | ls   | 1002 | hn   | 111    |    2 |
|    2 | ls   | 1001 | bj   | 110    |    1 |
|    3 | ww   | 1004 | zz   | 113    |    4 |
|    3 | ww   | 1003 | hn   | 112    |    3 |
|    3 | ww   | 1002 | hn   | 111    |    2 |
|    3 | ww   | 1001 | bj   | 110    |    1 |
|   11 | a    | 1004 | zz   | 113    |    4 |
|   11 | a    | 1003 | hn   | 112    |    3 |
|   11 | a    | 1002 | hn   | 111    |    2 |
|   11 | a    | 1001 | bj   | 110    |    1 |
|   22 | b    | 1004 | zz   | 113    |    4 |
|   22 | b    | 1003 | hn   | 112    |    3 |
|   22 | b    | 1002 | hn   | 111    |    2 |
|   22 | b    | 1001 | bj   | 110    |    1 |
|   33 | c    | 1004 | zz   | 113    |    4 |
|   33 | c    | 1003 | hn   | 112    |    3 |
|   33 | c    | 1002 | hn   | 111    |    2 |
|   33 | c    | 1001 | bj   | 110    |    1 |
+------+------+------+------+---------+------+
24 rows in set (0.00 sec)
```

2.内连接(取交集)

`mysql > select * from a join b on a.id=b.aid;`

2.0 介绍

> 如果分步骤理解的话，内连接可以看做先对两个表进行了交叉连接后，再通过加上限制条件(SQL中通过关键字on)剔除不符合条件的行的子集,得到的结果就是内连接

伪代码理解相当于两次for循环

```
for each_row in a        ---->驱动表
    for each_row in b    ---->非驱动表
    if a.id=b.aid
    marge
```

2.1 语句执行

```
mysql> select * from a,b where  a.id=b.aid;     或者 mysql> select * from a join b on a.id=b.aid;
+------+------+------+------+--------+------+
| id   | name | id   | addr | telnum | aid  |
+------+------+------+------+--------+------+
|    1 | zs   | 1001 | bj   | 110    |    1 |
|    2 | ls   | 1002 | hn   | 111    |    2 |
|    3 | ww   | 1003 | hn   | 112    |    3 |
+------+------+------+------+--------+------+
3 rows in set (0.00 sec)
```

3.外连接

3.0 介绍

外连接分为左外连接和右外连接，join语句前面的表称为左表，join语句后面的表称为右表。

3.1 左外连接

> 左表所有的数据行信息与右边进行条件匹配，条件不满足的数据行用NULL填充

3.2 左外连接语句执行

```
mysql> select * from a left join b on a.id=b.aid;
+------+------+------+------+--------+------+
| id   | name | id   | addr | telnum | aid  |
+------+------+------+------+--------+------+
|    1 | zs   | 1001 | bj   | 110    |    1 |
|    2 | ls   | 1002 | hn   | 111    |    2 |
|    3 | ww   | 1003 | hn   | 112    |    3 |
|   11 | a    | NULL | NULL | NULL   | NULL |
|   22 | b    | NULL | NULL | NULL   | NULL |
|   33 | c    | NULL | NULL | NULL   | NULL |
+------+------+------+------+--------+------+
```

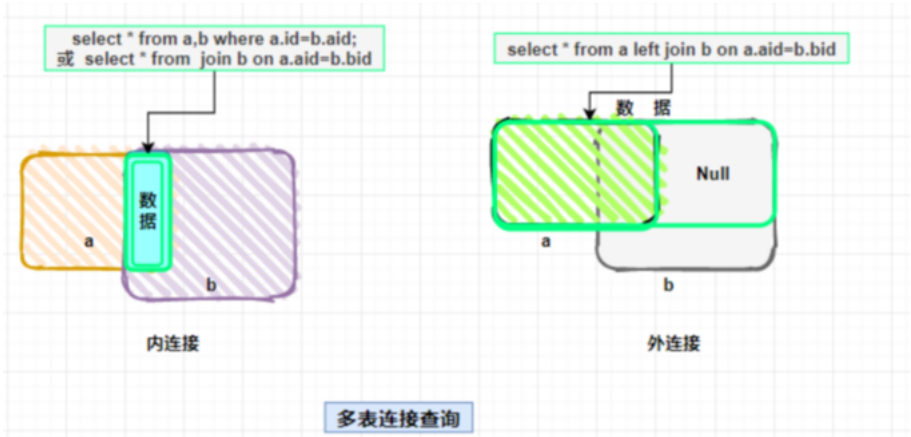
3.3 右外连接

> 右表所有的数据行信息与右边进行条件匹配，条件不满足的数据行用NULL填充

3.4 右外连接语句执行

```
mysql> select * from a right  join  b on a.id=b.aid;
+------+------+------+------+--------+------+
| id   | name | id   | addr | telnum | aid  |
+------+------+------+------+--------+------+
|    1 | zs   | 1001 | bj   | 110    |    1 |
|    2 | ls   | 1002 | hn   | 111    |    2 |
|    3 | ww   | 1003 | hn   | 112    |    3 |
| NULL | NULL | 1004 | zz   | 113    |    4 |
+------+------+------+------+--------+------+
```

4.内连接外连接原理图



9.3 多表连接书写套路

```
- 查询oldguo老师教的课程名
a. 通过需求找到所有需要的表
# teacher : 教师表
tno:      教师编号
tname:    教师名字

# course : 课程表
cno:   课程编号
cname: 课程名字
tno:   教师编号

b. 找直接和间接关联条件

teacher.tno = course.tno

c. 组合到一起()

from  teacher
join  course
on
teacher.tno = course.tno

d. 罗列其他查询条件
mysql> select teacher.tname ,course.cname
from teacher
join  course
on teacher.tno = course.tno
where teacher.tname='oldguo';
```

9.4 多表连接练习题

Bash | Copy

```
# student : 学生表
sno:    学号
sname: 学生姓名
sage:  学生年龄
ssex:  学生性别

# teacher : 教师表
tno:     教师编号
tname: 教师名字

# course : 课程表
cno:   课程编号
cname: 课程名字
tno:   教师编号

# sc  : 成绩表
sno:   学号
cno:   课程编号
score: 成绩

-- 1.  每位学生学习的课程门数
SELECT student.sname,COUNT(*)
FROM student
JOIN sc
ON student.sno=sc.sno
GROUP BY student.sno

-- 2.  每位老师所教的课程门数
SELECT  teacher.tname,COUNT(*)
FROM teacher
JOIN course
ON teacher.tno=course.tno
GROUP BY teacher.tno
-- 3.  每位老师所教的课程门数和名称
SELECT teacher.tname,COUNT(*),GROUP_CONCAT(course.cname)
FROM teacher
```

9.5 子查询

> 整个查询语句中，子查询作为内嵌语句，是外围查询语句的条件。

9.5.1 基于from语句后的子查询

子查询是查询其他存在表的结果集作为再次查询的条件

Bash | Copy

```
0.from+子查询例子
mysql>SELECT a,sum_po
FROM (SELECT district AS a ,SUM(population) AS sum_po FROM city WHERE countrycode='CHN'GROUP BY district) t1
WHERE t1.sum_po>5000000
ORDER BY sum_po;
1.我们拆分理解，先拿出子查询
mysql> SELECT district AS a ,SUM(population) AS sum_po
FROM city
WHERE countrycode='CHN'
GROUP BY district
子查询的意思是：先过滤出中国的省份，再对中国的省份进行分组，统计出中国每个省份的总人口数。
2.分析子查询后放入到整个查询语句中就是
mysql> SELECT a,sum_po from（中国每个省份的总人口数） t1
WHERE t1.sum_po>5000000
ORDER BY sum_po;
整个语句的意思：统计出中国每个省份的总人口数，再过滤中国每个身份总人口数大于5000000结果集，进行排序。

3.优化子查询（因为子查询不是真实的表，所以无法添加索引，不能走索引）
优化一：改写成多表连接 join on
改写成多表连接可以使用having语句，后过滤出中国每个身份总人口数大于5000000，再进行排序。
优化二：先执行子查询得到常量结果，再把常量结果带入到外围查询语句
```

9.5.2 基于where语句后面的子查询

Bash | Copy

```
0.where+子查询列子
mysql> SELECT t2.name, t2.`SurfaceArea`
FROM country AS t2 WHERE t2.code = (SELECT a.countrycode AS c3 FROM city AS a WHERE a.population<100) ;
1.我们拆分理解，先拿出子查询
mysql> SELECT a.countrycode AS c3 FROM city AS a WHERE a.population<100
子查询的意思是：查询人口数小于100的国家代码
2.分析子查询后放入到整个查询语句中就是
mysql> SELECT t2.name, t2.`SurfaceArea`
FROM country AS t2 WHERE t2.code = （查询人口数小于100的国家代码）;
整个语句的意思：查询人口数小于100的国家代码的国家名字和国土面积
3.优化子查询
优化一：改写成多表连接 join on
select t2.name, t2.`SurfaceArea`
from country as t2
right join city as t2
where t2.population<100;
优化二：先执行子查询得到常量结果，再把常量结果带入到外围查询语句
```

Bash | Copy

```
0.where+双层内嵌子查询
mysql> select name from hosts
where hostid = (select hostid from items as a,
(select itemid,min(value) from history_uint where itemid in (1,2,.5))as b where a.itemid = b.itemid);
1.我们拆分理解，先拿出最内嵌的子查询
mysql> select itemid,min(value)
from history_uint
where itemid in (1,2,.5);
子查询意思是：查询history_uint表中itemid是1或2或5的value列的最小值
2.再拿出第二层内嵌的子查询
mysql> select hostid from items as a, (查询history_uint表中itemid是1或2或5的value列的最小值)as b where a.itemid =
 b.itemid;
子查询意识是：将查询history_uint表中itemid是1或2或5的value列的最小值的结果集取别名为b,将items表取别名为a. 通过两个表相关条
件 a.itemid = b.itemid进行两个表的连接。查出hostid常量
3.最后将子查询的结果集带入到整个查询语句中。
mysql> select name from hosts where hostid =(子查询结果集查出的hostid常量);
整个查询语句的意识是：查询hosts表中hostid =(子查询结果集查出的hostid常量)的姓名列信息。
4.优化子查询
优化一：改写成多表连接 join on
我们假设把history_uint取别名为a表，把items表取别名为b表，把hosts表取别名为c表

mysql> select
c.name
from history_items as a
left join items as b
on  a.itemid = b.itemid
join  hosts as c
on b.hostid=c.hostid
where a.itemid in (1,2,5)
order by a.value limit 1;

为什么使用a left join a on b?
首先a相对于b表是小表，所以用a表做驱动表
其次我们需要先得到history_uint表中itemid是1或2或5的value列的最小值，我们使用左外连接就是a表中所有的信息列取匹配b表中所有的信
息。防止因为内连接a.itemid = b.itemid a表中itemid是1或2或5的value列的最小值丢失。

优化二：先执行子查询得到常量结果，再把常量结果带入到外围查询语句
```