MySQL学习笔记（Day012：子查询/INSERT/UPDATE/DELETE/REPLACE）

MySQL学习

一. 子查询

> 子查询就是指在一个select语句中嵌套另一个select语句。同时，子查询必须包含括号。
> MySQL 5.6.x 版本之前，MySQL的子查询性能较差，但是从5.6开始，不存在性能差的问题。

```
select a from t1 where a > any(select a from t2);
```

1. select a from t1 是外部查询(outer query)
2. (select a from t2) 是子查询

一般说来，子查询嵌套于外部查询中，可以将两个或两个以上的子查询进行嵌套

1. 子查询的使用

1.1. ANY / SOME

如果外部查询的列的结果和子查询的列的结果比较得到为True的话，则返回比较值为True的（外查询）的记录

```
mysql> create table  t1 (a int);
Query OK, 0 rows affected (0.15 sec)

mysql> create table  t2 (a int);
Query OK, 0 rows affected (0.14 sec)

mysql> insert into t1 values(10),(4);
Query OK, 2 rows affected (0.02 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> insert into t2 values(12),(13),(5);
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select a from t1;
+------+
| a    |
+------+
|   10 |
|    4 |
+------+
2 rows in set (0.00 sec)

mysql> select * from t2;
+------+
| a    |
+------+
|   12 |   -- t1中10，4比12小
|   13 |   -- t1中10，4比13小
|    5 |   -- t1中10比5大，4比5小
+------+
3 rows in set (0.00 sec)

mysql> select a from t1
    -> where a > any
    -> (select a from t2); -- 返回(12，13，4)
                           -- t1中a列的值，只要大于(12,13,4)中任意一值
                           -- 即t1.a > t2.a为True，则返回对应的t1.a
+------+
| a    |
+------+
|   10 |  -- 10 比 5 大为True，则返回该值，4比t2中所有的a值小，为False
+------+
1 row in set (0.00 sec)

-- 这个查询可以解释为，t1表内a列的值 大于 t2表中a列的任意(any)一个值（t1.a > any(t2.a) == true）,则返回t1.a的记录
```

> ANY 关键词必须与一个 比较操作符 一起使用： =， >， <， >=， <=， <> *（这个是!=的意思）*
>
> 子查询中 **SOME** 和 **ANY** 是同一个意思

1.2. IN

in 是 ANY 的一种特殊情况： **"in"** equals **"= any"**

```
mysql> insert into t1 values(5);  -- 向t1中插入一个t2中存在的值 5
Query OK, 1 row affected (0.03 sec)

mysql> select a from t1 where a = any(select a from t2); -- t1.a==t2.a 的只有5
+------+
| a    |
+------+
|    5 |
+------+
1 row in set (0.00 sec)

mysql> select a from t1 where a in (select a from t2); -- in的结果等同于 =any 的结果
+------+
| a    |
+------+
|    5 |
+------+
1 row in set (0.00 sec)
```

> select a from s1 where a in (select a in t2); 是用的比较多的一种语法

1.3. ALL

如果外部查询的列的结果和子查询的列的 所有结果 比较得到为True的话，则返回比较值为True的（外查询）的记录

```
mysql> truncate t1;   -- 清空t1
Query OK, 0 rows affected (0.07 sec)

mysql> truncate t2;   -- 清空t2
Query OK, 0 rows affected (0.10 sec)

mysql> insert into t1 values(10),(4);
Query OK, 2 rows affected (0.02 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> insert into t2 values(5),(4),(3);
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select a from t1 where a > all(select a from t2);
+------+
| a    |
+------+
|   10 |  -- (10 > 5, 4, 3 为 True) 而 (4 >5, 4, 3 为 False)
+------+
1 row in set (0.00 sec)
```

> ALL 关键词必须与一个 比较操作符 一起使用
> NOT IN 是 <> ALL 的别名

2. 子查询的分类

- 独立子查询
  - 不依赖外部查询而运行的子查询

```
mysql> select a from t1 where a in (1,2,3,4,5);
+------+
| a    |
+------+
|    4 |
+------+
1 row in set (0.00 sec)
```

- 相关子查询
  - 引用了外部查询列的子查询

```
-- 在这个例子中，子查询中使用到了外部的列t2.a
mysql> select a from t1 where a in (select * from t2 where t1.a = t2.a);
+------+
| a    |
+------+
|    4 |
+------+
1 row in set (0.00 sec)
```

3. 子查询的优化

- MySQL5.6之前
  在 MySQL5.6 之前，优化器会把子查询重写成 exists 的形式

```
select a from t1 where a in (select a from t2); -- 这个是一条独立的子查询，时间复杂度 O(M*N)
--
-- 经过优化器重写后
--
select a from t1 where exists (select 1 from t2 where t1.a = t2.a); -- 这是相关子查询，复杂度O(M*N + M)
```

所以在 MySQL 5.6 之前，部分的子查询需要重写成join的形式 *（注意表的大小）*

```
mysql> select t1.a from t1 join t2 on t1.a = t2.a;
+------+
| a    |
+------+
|    4 |
+------+
1 row in set (0.00 sec)
```

- MySQL 5.6之后
  在 MySQL 5.6 之后，优化器 不会 将子查询 重写 成 exists 的形式，而是自动优化，性能有了大幅提升

> 可通过 explain extended 来查看子查询优化的结果。由于 explain 还未讲到，该部分暂时跳过

4. 包含NULL值的NOT IN

```
mysql> select null in ('a', 'b', null);
+---------------------------+
| null in ('a', 'b', null) |
+---------------------------+
|                      NULL |
+---------------------------+
1 row in set (0.00 sec)
```

> MySQL数据库的 比较操作 ，除了返回 1(True)， 0(False) 之外，还会返回 NULL
> NULL 和 NULL 的比较，返回的还是 NULL

```
mysql> select null not in ('a', 'b', null);
+-------------------------------+
| null not in ('a', 'b', null) |
+-------------------------------+
|                          NULL |  -- null不在('a', 'b', null)中，返回的还是null，因为有null和null的比较
+-------------------------------+
1 row in set (0.00 sec)

mysql> select 'a' not in ('a', 'b', null);
+-----------------------------+
| 'a' not in ('a', 'b', null) |
+-----------------------------+
|                           0 |  -- a 不在 ('a', 'b', null)中，返回0,即False
+-----------------------------+
1 row in set (0.00 sec)

mysql> select 'c' not in ('a', 'b');
+-----------------------+
| 'c' not in ('a', 'b') |
+-----------------------+
|                     1 |  -- 这个返回值可以理解 'c'不在('a', 'b')中，返回1，即为True
+-----------------------+
1 row in set (0.00 sec)

mysql> select 'c' not in ('a', 'b', null);
+-----------------------------+
| 'c' not in ('a', 'b', null) |
+-----------------------------+
|                        NULL | -- 理论上应该是返回1，即True的。但是包含了null值。则返回null
+-----------------------------+
1 row in set (0.00 sec)
```

> 对于包含了 NULL 值的 IN 操作，总是返回 True 或者 NULL
> NOT IN 返回 NOT True (False) 或者 NOT NULL (NULL)

```
--
-- SQL语句一 使用 EXISTS
--
select customerid, companyname
    from customers as A
    where country = 'Spain'
        and not exists
            ( select * from orders as B
              where A.customerid = B.customerid );

--
-- SQL语句二 使用 IN
--
select customerid, companyname
    from customers as A
    where country = 'Spain'
        and customerid not in (select customerid from orders);

-----
-- 当结果集合中没有NULL值时，上述两条SQL语句查询的结果是一致的
-----

--
-- 插入一个NULL值
--
insert into orders(orderid) values (null);

-----
-- SQL语句1 : 返回和之前一致
-- SQL语句2 : 返回为空表，因为子查询返回的结果集中存在NULL值。not in null 永远返回False或者NULL
--            此时 where (country = 'Spain' and (False or NULL)) 为 False OR NULL，条件永远不匹配
-----

--
-- SQL语句2 改写后
--
select customerid, companyname
    from customers as A
    where country = 'Spain'
        and customerid not in (select customerid from orders
                               where customerid is not null);  -- 增加这个过滤条件，使用is not，而不是<>


--
-- 和 null比较，使用is和is not， 而不是 = 和 <>
--
mysql> select null = null;
+-------------+
| null = null |
+-------------+
|        NULL |
+-------------+
1 row in set (0.00 sec)

mysql> select null <> null;
+--------------+
| null <> null |
+--------------+
|         NULL |
+--------------+
1 row in set (0.00 sec)

mysql> select null is null;
+--------------+
| null is null |
+--------------+
|            1 |  -- 返回 True
+--------------+
1 row in set (0.00 sec)

mysql> select null is not  null;
+--------------------+
| null is not  null |
+--------------------+
|                  0 |  -- 返回 False
+--------------------+
1 row in set (0.00 sec)
```

> EXISTS 不管返回值是什么，而是看是否有 行 返回，所以 EXISTS 中子查询都是 select * 、 select 1 等，因为只关心返回是否有行（结果集）

二. INSERT

> 官方文档

```
mysql> insert into t1 values(1);  -- 插入一个值
Query OK, 1 row affected (0.03 sec)

mysql> insert into t1 values(2),(3),(-1);  -- 插入多个值，MySQL独有
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> insert into t1 select 8;   -- insert XXX select XXX 语法，MySQ独有
Query OK, 1 row affected (0.02 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> create table t3 (a int, b int); -- 有多个列
Query OK, 0 rows affected (0.15 sec)

mysql> insert into t3 select 8;  -- 没有指定列，报错
ERROR 1136 (21S01): Column count doesn't match value count at row 1

mysql> insert into t3(a) select 8;  -- 指定列a
Query OK, 1 row affected (0.04 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> insert into t3 select 8, 9;  -- 不指定列，但是插入值匹配列的个数和类型
Query OK, 1 row affected (0.03 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> select * from t3;
+------+------+
| a    | b    |
+------+------+
|    8 | NULL |
|    8 |    9 |
+------+------+
2 rows in set (0.00 sec)

mysql> insert into t3(b) select a from t2;  -- 从t2表中查询数据并插入到t3(a)中，注意指定列
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from t3;
+------+------+
| a    | b    |
+------+------+
|    8 | NULL |
|    8 |    9 |
| NULL |    5 |
| NULL |    4 |
| NULL |    3 |
+------+------+
5 rows in set (0.00 sec)

--
-- 如果想快速增长表格中的数据，可以使用如下方法，使得数据成倍增长
--
mysql> insert into t3 select * from t3;
Query OK, 5 rows affected (0.03 sec)  -- 插入了5列
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from t3;
+------+------+
| a    | b    |
+------+------+
|    8 | NULL |
|    8 |    9 |
| NULL |    5 |
| NULL |    4 |
| NULL |    3 |
|    8 | NULL |
|    8 |    9 |
| NULL |    5 |
| NULL |    4 |
| NULL |    3 |
+------+------+
10 rows in set (0.00 sec)

mysql> insert into t3 select * from t3;
Query OK, 10 rows affected (0.03 sec)  -- 插入了10列，成倍增长
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from t3;
+------+------+
| a    | b    |
+------+------+
|    8 | NULL |
|    8 |    9 |
| NULL |    5 |
| NULL |    4 |
| NULL |    3 |
|    8 | NULL |
|    8 |    9 |
| NULL |    5 |
| NULL |    4 |
| NULL |    3 |
|    8 | NULL |
|    8 |    9 |
| NULL |    5 |
| NULL |    4 |
| NULL |    3 |
|    8 | NULL |
|    8 |    9 |
| NULL |    5 |
| NULL |    4 |
| NULL |    3 |
+------+------+
20 rows in set (0.00 sec)
```

三. DELETE

> 官方文档

```
mysql> delete from t3 where a is null;  -- 根据过滤条件删除
Query OK, 12 rows affected (0.03 sec)

mysql> select * from t3;
+------+------+
| a    | b    |
+------+------+
|    8 | NULL |
|    8 |    9 |
|    8 | NULL |
|    8 |    9 |
|    8 | NULL |
|    8 |    9 |
|    8 | NULL |
|    8 |    9 |
+------+------+
8 rows in set (0.00 sec)

mysql> delete from t3;   -- 删除整个表
Query OK, 8 rows affected (0.03 sec)

mysql> select * from t3;
Empty set (0.00 sec)
```

四. UPDATE

> 官方文档

```
mysql> insert into t3 select 1,2;
Query OK, 1 row affected (0.03 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> select * from t3;
+------+------+
| a    | b    |
+------+------+
|    1 |    2 |
+------+------+
1 row in set (0.00 sec)

mysql> update t3 set a=10 where a=1;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from t3;
+------+------+
| a    | b    |
+------+------+
|   10 |    2 |
+------+------+
1 row in set (0.00 sec)

--
-- 关联后更新
--
mysql> select * from t1;
+------+
| a    |
+------+
|   10 |
|    4 |  -- 和t2中的4相等
|    1 |
|    2 |
|    3 |  -- 和t2中的3相等
|   -1 |
|    8 |
+------+
7 rows in set (0.00 sec)

mysql> select * from t2;
+------+
| a    |
+------+
|    5 |
|    4 |  -- 和t1中的4相等
|    3 |  -- 和t1中的3相等
+------+
3 rows in set (0.00 sec)

mysql> update t1 join t2 on t1.a = t2.a set t1.a=100;  -- 先得到t1.a=t2.a的结果集
                                                       -- 然后将结果集中的t1.a设置为100
Query OK, 2 rows affected (0.03 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from t1;
+------+
| a    |
+------+
|   10 |
|  100 |  -- 该行被更新成100
|    1 |
|    2 |
|  100 |  -- 该行被更新成100
|   -1 |
|    8 |
+------+
7 rows in set (0.00 sec)
```

五. REPLACE

> 官方文档

```
mysql> create table t4(a int primary key auto_increment, b int);
Query OK, 0 rows affected (0.15 sec)

mysql> insert into t4 values(NULL, 10);
Query OK, 1 row affected (0.02 sec)

mysql> insert into t4 values(NULL, 11);
Query OK, 1 row affected (0.03 sec)

mysql> insert into t4 values(NULL, 12);
Query OK, 1 row affected (0.03 sec)

mysql> select * from t4;
+---+------+
| a | b    |
+---+------+
| 1 |   10 |
| 2 |   11 |
| 3 |   12 |
+---+------+
3 rows in set (0.00 sec)

mysql> insert into t4 values(1, 100);  -- 报错，说存在重复的主键记录 "1"
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'

mysql> replace into t4 values(1, 100); -- 替换该主键对应的值
Query OK, 2 rows affected (0.03 sec)   -- 两行记录受到影响

mysql> select * from t4;
+---+------+
| a | b    |
+---+------+
| 1 |  100 |  -- 已经被替换
| 2 |   11 |
| 3 |   12 |
+---+------+
3 rows in set (0.00 sec)
-----
-- replace的原理是：先delete，在insert
-----

mysql> replace into t4 values(5, 50);  -- 没有替换对象时，类似插入效果
Query OK, 1 row affected (0.03 sec)    -- 只影响1行

mysql> select * from t4;
+---+------+
| a | b    |
+---+------+
| 1 |  100 |
| 2 |   11 |
| 3 |   12 |
| 5 |   50 |  -- 插入了1行
+---+------+
4 rows in set (0.00 sec)

--
-- replace原理更明显的例子
--

mysql> create table t6
    -> (a int primary key,
    -> b int auto_increment,   -- b是auto_increment的int型数据
    -> c int, key(b));
Query OK, 0 rows affected (0.15 sec)

mysql> insert into t6 values(10, NULL, 100),(20,NULL,200);  -- b自增长
Query OK, 2 rows affected (0.02 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from t6;
+----+---+------+
| a  | b | c    |
+----+---+------+
| 10 | 1 |  100 |  -- b为1
| 20 | 2 |  200 |  -- b为2
+----+---+------+
2 rows in set (0.00 sec)

mysql> replace into t6 values(10,NULL,150);  -- 将a=10的替换掉
Query OK, 2 rows affected (0.03 sec)

mysql> select * from t6;
+----+---+------+
| a  | b | c    |
+----+---+------+
| 10 | 3 |  150 |  -- 替换后b从1变成了3，说明是先删除，再插入
| 20 | 2 |  200 |
+----+---+------+
2 rows in set (0.00 sec)

-----

--
-- insert on duplicate 效果和 replace类似
--
mysql> insert into t4 values(1,1);  -- 插入报错，存在key为1的记录
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'

mysql> insert into t4 values(1,1) on duplicate key update b=1;  -- 带上on duplicate参数
                                                                -- 非SQL标准，不推荐
Query OK, 2 rows affected (0.03 sec)

mysql> select * from t4;
+---+------+
| a | b    |
+---+------+
| 1 |    1 |  -- 该行的b列从100被替换成1
| 2 |   11 |
| 3 |   12 |
| 5 |   50 |
+---+------+

--
-- insert ignore
--
mysql> insert ignore into t4 values(1,1);   -- 忽略重复的错误
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> show warnings;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Warning | 1062 | Duplicate entry '1' for key 'PRIMARY' |
+---------+------+----------------------------------------+
1 row in set (0.00 sec)
```

六. 其他知识点

- **更新有关系的值**

```
mysql> create table t5 (a int, b int);
Query OK, 0 rows affected (0.14 sec)

mysql> insert into t5 values(1,1);
Query OK, 1 row affected (0.03 sec)

mysql> select * from t5;
+------+------+
| a    | b    |
+------+------+
|    1 |    1 |
+------+------+
1 row in set (0.00 sec)

mysql> update t5 set a=a+1, b=a where a=1;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from t5;
+------+------+
| a    | b    |
+------+------+
|    2 |    2 |  -- SQL Server和Oracle中得到的值是2, 1
+------+------+
1 row in set (0.00 se
```

- **显示行号(RowNumber)**

```
--
-- 方法一
--
mysql> use employees ;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> set @rn:=0;  -- 产生 SESSION(会话)级别的变量
Query OK, 0 rows affected (0.00 sec)

mysql> select @rn:=@rn+1 as rownumber, emp_no, gender from employees limit 10;  -- : = 是赋值的意思
+-----------+--------+--------+
| rownumber | emp_no | gender |
+-----------+--------+--------+
|        11 |  10001 | M      |
|        12 |  10002 | F      |
|        13 |  10003 | M      |
|        14 |  10004 | M      |
|        15 |  10005 | M      |
|        16 |  10006 | F      |
|        17 |  10007 | F      |
|        18 |  10008 | M      |
|        19 |  10009 | F      |
|        20 |  10010 | F      |
+-----------+--------+--------+
10 rows in set (0.00 sec)

--
-- 方法二 （推荐）
--
mysql> select @rn1:=@rn1+1 as rownumber, emp_no, gender from employees, (select @rn1:=0) as a limit 10;
+-----------+--------+--------+
| rownumber | emp_no | gender |
+-----------+--------+--------+
|         1 |  10001 | M      |
|         2 |  10002 | F      |
|         3 |  10003 | M      |
|         4 |  10004 | M      |
|         5 |  10005 | M      |
|         6 |  10006 | F      |
|         7 |  10007 | F      |
|         8 |  10008 | M      |
|         9 |  10009 | F      |
|        10 |  10010 | F      |
+-----------+--------+--------+
10 rows in set (0.00 sec)

-- MySQL 自定义变量，根据每一记录进行变化的

mysql> select @rn1:=0;
+----------+
| @rn1:=0 |
+----------+
|        0 |  -- 只有一行记录
+----------+
1 row in set (0.00 sec)

-- 相当于 把 employees 和 (select @rn1:=0)做了笛卡尔积，然后使用@rn1:=@rn + 1，根据每行进行累加

--
-- ":=" 和 "="
--
mysql> set @a:=1;  -- 赋值为1
Query OK, 0 rows affected (0.00 sec)

mysql> select @a;
+------+
| @a   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)

mysql> set @a:=10;  -- 赋值为10
Query OK, 0 rows affected (0.00 sec)

mysql> select @a;
+------+
| @a   |
+------+
|   10 |
+------+
1 row in set (0.00 sec)

mysql> select @a=9;  -- 进行比较
+------+
| @a=9 |
+------+
|    0 |  -- 返回为False
+------+
1 row in set (0.00 sec)

--
-- 作业：通过子查询或者其他方式，计算出employees的行号
--
```