MySQL学习笔记

一、基础语言

1、select

1.1 基础查询

示例表：

```
//查询表中所有数据
SELECT * FROM T_Product;
```

id	product_name	stock	price	version	note
1	java书	8	55.00	2018	java基础学习
2	javascript书	5	50.00	2019	javascript基础学习
3	java书1	8	60.00	2017	java进阶
4	javascript书1	20	61.00	2017	javascript进阶
5	java书	16	88.00	2018	java高级
6	javascript书	30	99.00	2019	javascript高级学习

```
//查询表中某些字段
SELECT product_name,stock FROM T_Product;
```

product_name	stock
java书	8
javascript书	5
java书1	8
javascript书1	20
java书	16
javascript书	30

```
//去除查询字段中的重复值
SELECT DISTINCT product_name FROM T_Product;
```

product_name
java书
javascript书
java书1
javascript书1

```
//根据条件查询
SELECT * FROM T_Product where product_name="java书";
```

id	product_name	stock	price	version	note
1	java书	8	55.00	2018	java基础学习
5	java书	16	88.00	2018	java高级

1.2 根据查询条件

查询条件	谓词
比较	=, >, <, >=, <=, !=, <>(不等于), !>(不大于), !<(不小于);
确定范围	BETWEEN AND, NOT BETWEEN AND
确定集合	IN , NOT IN
字符匹配	LIKE, NOT LIKE
空值	IS NULL, IS NOT NULL
多重条件	AND , OR , NOT

1.2.1 根据范围(BETWEEN AND)

查找属性值在指定范围内的元组

```
//查询库存在8-30之间的产品名称、价格、库存
SELECT product_name,price,stock FROM T_Product WHERE stock BETWEEN 8 AND 30;
```

product_name	price	stock
java书	55.00	8
javascript书1	61.00	20
java书	88.00	16
javascript书	99.00	30

```
////查询库存不在8-30之间的产品名称、价格、库存
SELECT product_name,price,stock FROM T_Product WHERE stock NOT BETWEEN 8 AND 30;
```

product_name	price	stock
javascript书	50.00	5

1.2.2 确定集合(IN)

查找属性值属于指定集合的元组

```
SELECT product_name,price FROM T_Product WHERE VERSION IN (2019,2018);
```

product_name	price
java书	55.00
javascript书	50.00
java书	88.00
javascript书	99.00

```
SELECT product_name,price FROM T_Product WHERE VERSION NOT IN (2019,2018);
```

product_name	price
javascript书1	61.00

1.2.3 字符匹配(LIKE)

进行字符串的匹配：

语法：[NOT] LIKE '<匹配串>' [ESCAPE '<换码字符>']

<匹配串>可以是完整的字符串，也可以含有通配符%和_。

%代表任意长度的字符串。

_代表任意单个字符。

```
SELECT * FROM T_Product WHERE price LIKE 50.00;
```

id	product_name	stock	price	version	note
2	javascript书	5	50.00	2019	javascript基础学习

```
SELECT product_name,price FROM T_Product WHERE note  LIKE 'java%';
```

product_name	price
java书	55.00
javascript书	50.00
javascript书1	61.00
java书	88.00
javascript书	99.00

注意：若查询条件中的字符串本身就含有通配符 % 或 _，此时需要使用ESCAPE '<换码字符>'短语对通配符进行转义。

```
//此时%不再具有通配符的含义及作用
SELECT product_name,price FROM T_Product WHERE note  LIKE 'java\%高级' ESCAPE
'\';
```

1.2.4 空值查询(NULL)

```
SELECT product_name FROM T_Product WHERE price IS NULL;
```

注意： IS不能用（=）代替

1.2.5 多重条件查询

```
//and & or的应用
SELECT * FROM T_Product WHERE (product_name="java书" or stock=99) and price=55;
```

id	product_name	stock	price	version	note
1	java书	8	55.00	2018	java基础学习

1.2.6 ORDER BY子句

对查询结果按照一个或多个属性列的升序或降序（DESC）排列，默认为升序

```
//产品名称按照字母顺序排序，价格反序排列
SELECT product_name,stock FROM T_Product order by product_name,stock DESC;
```

product_name	stock
javascript书	30
javascript书	5
javascript书1	20
java书	16
java书	8

1.2.7GROUP BY

GROUP BY 语句用于结合合计函数，根据一个或多个列对结果集进行分组

```
//以产品名称分组，查询总价格
SELECT product_name,SUM(price) FROM T_Product GROUP BY product_name;
```

product_name	sum(price)
javascript书	149.00
javascript书1	61.00
java书	143.00

1.2.8 HAVING

```
//以产品名称分组，查询库存大于20的产品
SELECT product_name,SUM(stock) FROM T_Product GROUP BY product_name HAVING
SUM(stock)>20;
```

product_name	sum(stock)
javascript书	35
java书	24

2、insert

```
//增加一行数据
INSERT INTO T_Product(product_name,stock,price,VERSION,note) VALUES ('javascript
书',30,99,2019.3,'javascript高级学习');
```

3、update

```
//更改某个字段值
UPDATE T_Product SET product_name="java进阶教程" WHERE note="java进阶";
```

3	java进阶教程	8	60.00	2017	java进阶

4、delete

```
//删除某行
delete from T_Product where price=60;
```

二、SQL函数

1、avg()

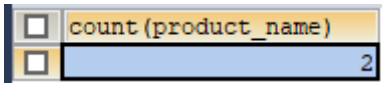
```
//查询价格平均值
SELECT AVG(price) FROM T_Product;
//查询价格大于平均值的产品名
SELECT product_name FROM T_Product WHERE price>(SELECT AVG(price) FROM
T_Product);
```

avg(price)
70.600000

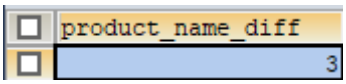
product_name
java书
javascript书

2、count()

```
//查询java书的总数
SELECT COUNT(product_name) FROM T_Product WHERE product_name="java书";
//查询不同产品名的数目
SELECT COUNT(DISTINCT product_name) AS product_name_diff FROM T_Product;
```



count(product_name)
2



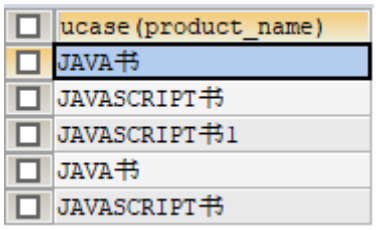
product_name_diff
3

3、ucase()/lcase()

UCASE 函数把字段的值转换为大写

LCASE函数把字段的值转换为小写

```
//将product_name字段值转换成大写
SELECT UCASE(product_name) FROM T_Product;
//将product_name字段值转换成小写
SELECT LCASE(product_name) FROM T_Product;
```

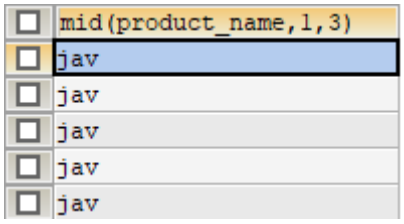


ucase(product_name)
JAVA书
JAVASCRIPT书
JAVASCRIPT书1
JAVA书
JAVASCRIPT书

4、mid()

MID 函数用于从文本字段中提取字符

```
//提取product_name字段中前三个字符
SELECT MID(product_name,1,3) FROM T_Product;
```



mid(product_name,1,3)
jav
jav
jav
jav
jav

5、len()

LEN 函数返回文本字段中值的长度

```
//查询字段中的字符长度
SELECT Length(product_name) as product_name_length FROM T_Product;
```

product_name_length
7
13
14
7
13

6、round()

ROUND 函数用于把数值字段舍入为指定的小数位数

```
//将库存字段的值的小数位设为1
SELECT product_name,ROUND(stock,1) AS price FROM T_Product;
```

product_name	price
java书	8.0
javascript书	5.0
javascript书1	20.0
java书	16.0
javascript书	30.0

7、now()

NOW 函数返回当前的日期和时间

```
//显示当前时间，各个产品的价格
select product_name,price, now() as PerDate from T_Product;
```

product_name	price	PerDate
java书	55.00	2019-10-30 09:33:25
javascript书	50.00	2019-10-30 09:33:25
javascript书1	61.00	2019-10-30 09:33:25
java书	88.00	2019-10-30 09:33:25
javascript书	99.00	2019-10-30 09:33:25

8、format()

FORMAT 函数用于对字段的显示进行格式化

```
//将日期显示为指定的格式
SELECT product_name,price, format(now(),'YYYY-MM-DD') as PreDate FROM T_Product;
```

三、连接查询

1、inner join

用于根据两个或多个表中的列之间的关系，从这些表中查询数据

```
SELECT T_Product.`product_name`,T_Product.`stock`,T_user.`name` FROM
T_Product,T_user WHERE T_Product.`id` = T_user.`pro_id`;
```

使用 inner join

```
SELECT T_Product.`product_name`,T_Product.`stock`,T_user.`name`
       FROM T_Product inner join T_user
       on T_Product.`id` = T_user.`pro_id`
       order by T_Product.`stock`;
```

product_name	stock	name
javascript书	5	小黑
java书	8	小绿
java书	8	小红
java书	16	小蓝
javascript书1	20	小紫

除了我们在上面的例子中使用的 INNER JOIN（内连接），还有其他几种连接：

JOIN: 如果表中有至少一个匹配，则返回行

LEFT JOIN: 即使右表中没有匹配，也从左表返回所有的行

RIGHT JOIN: 即使左表中没有匹配，也从右表返回所有的行

FULL JOIN: 只要其中一个表中存在匹配，就返回行

2、left join

LEFT JOIN 关键字会从左表 (table_name1) 返回所有的行，即使在右表 (table_name2) 中没有匹配的行。

语法：

```
SELECT column_name(s)  FROM table_name1 LEFT JOIN table_name2
       ON table_name1.column_name=table_name2.column_name
```

example:

```
SELECT T_Product.`product_name`,T_Product.`stock`,T_user.`name`
       FROM T_Product LEFT JOIN T_user
       ON T_Product.`id` = T_user.`pro_id`
       ORDER BY T_Product.`stock`;
```

product_name	stock	name
javascript书	5	小黑
java书	8	小绿
java书	8	小红
java书	16	小蓝
javascript书1	20	小紫
javascript书	30	*(NULL)*

3、right join

RIGHT JOIN 关键字会右表 (table_name2) 那里返回所有的行，即使在左表 (table_name1) 中没有匹配的行

语法：

```
SELECT column_name(s)
FROM table_name1
RIGHT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

example:

```
SELECT T_Product.`product_name`,T_Product.`stock`,T_user.`name`
       FROM T_Product RIGHT JOIN T_user
       ON T_Product.`id` = T_user.`pro_id`
       ORDER BY T_Product.`stock` DESC;
```

product_name	stock	name
javascript书	30	小彩
javascript书1	20	小紫
java书	16	小蓝
java书	8	小绿
java书	8	小红
javascript书	5	小黑
(NULL)	*(NULL)*	小青

4、full join

只要其中某个表存在匹配，FULL JOIN 关键字就会返回行

```
SELECT T_Product.`product_name`,T_Product.`stock`,T_user.`name`
       FROM T_Product
       FULL JOIN T_user
       ON T_Product.`id` = T_user.`pro_id`
       ORDER BY T_Product.`stock` DESC;
```

5、创建表

数据类型	描述
integer(size) int(size) smallint(size) tinyint(size)	仅容纳整数。在括号内规定数字的最大位数。
decimal(size,d) numeric(size,d)	容纳带有小数的数字。 "size" 规定数字的最大位数。"d" 规定小数点右侧的最大位数。
char(size)	容纳固定长度的字符串（可容纳字母、数字以及特殊字符）。 在括号中规定字符串的长度。
varchar(size)	容纳可变长度的字符串（可容纳字母、数字以及特殊的字符）。 在括号中规定字符串的最大长度。
date(yyyymmdd)	容纳日期。

5.1 UNIQ约束

约束唯一标识数据库表中的每条记录

UNIQUE 和 PRIMARY KEY 约束均为列或列集合提供了唯一性的保证。

PRIMARY KEY 拥有自动定义的 UNIQUE 约束。

请注意，每个表可以有多个 UNIQUE 约束，但是每个表只能有一个 PRIMARY KEY 约束

5.2 CHECK

CHECK 约束用于限制列中的值的范围

例：CHECK (Id_P>0)

5.3 DEFAULT

用于向列中插入默认值

例：City varchar(255) DEFAULT '西安'　　//设置城市的默认值为西安

5.4 Date

MySQL 使用的数据类型：

- DATE - 格式 YYYY-MM-DD
- DATETIME - 格式: YYYY-MM-DD HH:MM:SS
- TIMESTAMP - 格式: YYYY-MM-DD HH:MM:SS
- YEAR - 格式 YYYY 或 YY

四、动态SQL语句

1、if

根据条件查询，与 where 搭配使用

例：

```
<trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="name != null">
                name,
            </if>
            <if test="age != null">
                age,
            </if>
            <if test="hobby != null" >
                hobby,
            </if>
        </trim>
        <trim prefix="values(" suffix=")" suffixOverrides=",">
            <if test="name != null">
                #{name},
            </if>
            <if test="age != null">
                #{age},
            </if>
            <if test="hobby != null">
                #{hobby},
            </if>
        </trim>
```

2、choose、when、otherwise

当不想应用到所有的条件语句，只想从中择其一项。MyBatis 提供的 choose 元素，类似于 Java 中的 switch 语句。

```
<select id="findActiveBlogLike"
     resultType="Blog">
  SELECT * FROM BLOG WHERE state = ‘ACTIVE’
  <choose>
    <when test="title != null">
```

```
      AND title like #{title}
    </when>
    <when test="author != null and author.name != null">
      AND author_name like #{author.name}
    </when>
    <otherwise>
      AND featured = 1
    </otherwise>
  </choose>
</select>
```

3、trim、set、where

where:

```
<select id="findActiveBlogLike"
     resultType="Blog">
  SELECT * FROM BLOG
  <where>
    <if test="state != null">
         state = #{state}
    </if>
    <if test="title != null">
        AND title like #{title}
    </if>
    <if test="author != null and author.name != null">
        AND author_name like #{author.name}
    </if>
  </where>
</select>
```

trim:

```
<trim prefix="(" suffix=")" suffixOverrides=",">
```

此语句的作用是：为语句添加前缀和后缀值，并且移除语句中最后一个“，”

set:

```
<update id="updateAuthorIfNecessary">
  update Author
    <set>
      <if test="username != null">username=#{username},</if>
      <if test="password != null">password=#{password},</if>
      <if test="email != null">email=#{email},</if>
      <if test="bio != null">bio=#{bio}</if>
    </set>
  where id=#{id}
</update>
```

4、foreach

对一个集合进行遍历，通常是在构建 IN 条件语句的时候

```
<select id="selectPostIn" resultType="domain.blog.Post">
  SELECT *
  FROM POST P
  WHERE ID in
  <foreach item="item" index="index" collection="list"
      open="(" separator="," close=")">
        #{item}
  </foreach>
</select>
```