

# 目录

MySQL学习笔记	1.1
-----------	-----

## Part I 查询

简单查询 (SELECT 语句)	2.1
分组查询 (Group By语句)	2.2
筛选数据 (WHERE子句)	2.3
MySQL库函数	2.4

## Part II 增删改

We love feedback	3.1
Better tools for authors	3.2

## Part III 高级特性

We love feedback	4.1
用户管理	4.2

- [Introduction](#)

## Introduction

本文是《MySQL必知必会》，人民邮电出版社，的学习笔记，请购买正版图书。

Copyright © dujiaju.net 2020 all right reserved, powered by Gitbook该文件修订时间： 2020-01-10 16:24:00

- 简单查询语句(SELECT 查询语句)
  - 检索单个列
  - 检索多个列
  - 检索所有列
  - 检索不同的行（去重复值）
  - 限制结果(返回行数)
  - 使用完全限定的表名

## 简单查询语句(SELECT 查询语句)

### 检索单个列

```
SELECT 列名 FROM 表名
```

### 检索多个列

```
SELECT 列名1,列名2 FROM 表名
```

### 检索所有列

```
SELECT * FROM 表名
```

一般，除非你确实需要表中的每个列，否则最好别使用 \* 通配符。虽然使用通配符可能会使你省事，不用明确列出所需列，但检索不需要的列通常会降低检索和应用程序的性能。使用通配符有一个大优点。由于不明确指定列名(因为星号检索每个列)，所以能检索出名字未知的列。

### 检索不同的行（去重复值）

```
SELECT DISTINCT 列名 FROM 表名`
```

不能部分使用**DISTINCT** DISTINCT关键字应用于所有列而不仅是前置它的列。如果给出SELECT DISTINCT 列1, 列2 FROM 表名，除非指定的两个列都不同，否则所有行都将被检索出来。

### 限制结果(返回行数)

```
SELECT * FROM 表名 LIMIT 开始行数, 返回行数
SELECT * FROM 表名 LIMIT 返回行数 OFFSET 开始行数  仅MySQL 5以上支持
```

例：返回表中前五条数据

```
SELECT * FROM 表名 LIMIT 0,5
或MySQL5+ `SELECT * FROM 表名 LIMIT 5 OFFSET 0
```

返回表中第6~10条数据

```
SELECT * FROM 表名 LIMIT 5,5  
或MySQL5+ SELECT * FROM 表名 LIMIT 5 OFFSET 5
```

**行0** 检索出来的第一行为行0而不是行1。因此，LIMIT 1,1 将检索出第二行而不是第一行。

**在行数不够时** LIMIT中指定要检索的行数为检索的最大行数。如果没有足够的行(例如，给出LIMIT 10, 5，但只有13 行)，MySQL将只返回它能返回的那么多行。

**MySQL 5的LIMIT语法** LIMIT 3, 4的含义是从行4开始的3 行还是从行3开始的4行？如前所述，它的意思是从行3开始的4 行，这容易把人搞糊涂。由于这个原因，MySQL 5支持LIMIT的另一种替代语法。LIMIT 4 OFFSET 3意为从行3开始取4行，就像LIMIT 3, 4一样。

## 使用完全限定的表名

```
SELECT 表名.列名 FROM 表名
```

完全限定名字可以在联合查询的时候避免多表的列重名的情况。

Copyright © dujiaju.net 2020 all right reserved, powered by Gitbook该文件修订时间：2020-01-10 13:54:27

- ORDER BY语句
  - 排序数据
  - 按多个列排序
  - 指定排序方向

## ORDER BY语句

### 排序数据

```
SELECT * FROM 表名 ORDER BY 列名 (要排序的列)
```

**通过非选择列进行排序** 通常，ORDERBY子句中使用的列将是为显示所选择的列。但是，实际上并不一定要这样，用非检索的列排序数据是完全合法的。

### 按多个列排序

```
SELECT * FROM 表名 ORDER BY 列名1 (要排序的列), 列名2
```

按照列1先排序，当列1相同时，按照列2排序

### 指定排序方向

默认是升序（A~Z字母顺序）排列

```
SELECT * FROM 表名 ORDER BY 列名  
SELECT * FROM 表名 ORDER BY 列名 ASC
```

降序排列(从Z到A)

```
SELECT * FROM 表名 ORDER BY 列名 DESC
```

**在多个列上降序排序** 如果想在多个列上进行降序排序，必须对每个列指定DESC关键字

**区分大小写和排序顺序** 在对文本性的数据进行排序时，A与a相同吗？a位于B之前还是位于Z之后？这些问题不是理论问题，其答案取决于数据库如何设置。

在字典(dictionary)排序顺序中，A被视为与a相同，这是MySQL (和大多数数据库管理系统)的默认行为。但是，许多数据库 管理员能够在需要时改变这种行为(如果你的数据库包含大量外语字符，可能必须这样做)。

这里，关键的问题是，如果确实需要改变这种排序顺序，用简单的ORDER BY子句做不到。你必须请求数据库管理员的帮助

Copyright © dujiaju.net 2020 all right reserved, powered by Gitbook该文件修订时间： 2020-01-10 16:01:23

- 过滤数据（WHERE）子句
  - 过滤数据
  - WHERE子句操作符
  - 单值查询
  - 不匹配查询
  - 范围查询
  - 空值查询
  - 并列查询（AND操作符）
  - 任一查询（OR操作符）
  - AND OR 优先级
  - 范围查询（IN操作符）
  - 不匹配查询（NOT IN操作符）
  - 用通配符进行过滤（LIKE操作符）
  - 正则表达式

## 过滤数据（WHERE）子句

### 过滤数据

```
SELECT * FROM 表名 WHERE 条件
```

例：

```
SELECT * FROM products WHERE prod_price='14.99'
```

**WHERE子句的位置** 在同时使用ORDERBY和WHERE子句时，应该让ORDER BY位于WHERE之后，否则将会产生错误！

例：

```
SELECT * FROM products WHERE prod_price='14.99' ORDER BY prod_price
```

### WHERE子句操作符

操作符	说明
=	等于
<>	不等于
!=	不等于
<	小于
<=	小于等于
>	大于
>=	大于等于
BETWEEN	在指定的两个值之间

## 单值查询

注意！**MySQL在执行匹配时默认不区分大小写**，如fuses与Fuses匹配。

```
SELECT * FROM 表名 WHERE prod_name='fuses';
```

## 不匹配查询

```
SELECT * FROM 表名 WHERE 列名 <> '值';
或者 SELECT * FROM 表名 WHERE 列名 != '值';
```

如：输出不是fuses的结果

```
SELECT * FROM products WHERE prod_name <> 'fuses'
或者 SELECT * FROM products WHERE prod_name != 'fuses';
```

**何时使用引号** 如果仔细观察上述WHERE子句中使用的条件， 会看到有的值括在单引号内(如前面使用的'fuses')而有的值未括起来。单引号用来限定字符串。如果将值与串类型的列进行比较，则需要限定引号。用来与数值列进行比较的值不用引号。

## 范围查询

```
SELECT * FROM 表名 WHERE 列名 BETWEEN 条件A AND 条件B ;
```

例：查询价格在1~11之间的记录

```
SELECT * FROM products WHERE prod_price BETWEEN 1 and 11
```

## 空值查询

注意：**NULL 无值(no value)**，它与字段包含0、空字符串或仅仅包含空格不同。

```
SELECT * FROM 表名 WHERE 列名 IS NULL ;
```

## 并列查询（AND操作符）

```
SELECT * FROM 表名 WHERE 条件1 AND 条件2;
```

例：供应商1003，价格<30的产品信息

```
SELECT * FROM products WHERE prod_price <30 AND vend_id =1003;
```

## 任一查询（OR操作符）

```
SELECT * FROM 表名 WHERE 条件1 OR 条件2;
```

例：供应商是1003或者价格<30的产品信息

```
SELECT * FROM products WHERE prod_price <30 OR vend_id =1003;
```

## AND OR 优先级

SQL(像多数语言一样)在处理OR操作符前，优先处理AND操作符。

在WHERE子句中使用圆括号，消除奇异。

## 范围查询（IN操作符）

IN操作符用来指定条件范围，范围中的每个条件都可以进行匹配。IN取合法值的由逗号分隔的清单，全都括在圆括号中。

下列两行代码等价

```
SELECT * FROM 表名 WHERE 列名 IN (条件1,条件2);  
SELECT * FROM 表名 WHERE 列名 = 条件1 OR 列名 = 条件2;
```

IN操作符一般比OR操作符清单执行更快。

## 不匹配查询（NOT IN操作符）

NOT IN操作符在WHERE子句中用来否定后跟条件的关键字。

下列两行代码等价

```
SELECT * FROM 表名 WHERE 列名 NOT IN (条件1,条件2);  
SELECT * FROM 表名 WHERE 列名 != 条件1 AND 列名 != 条件2;
```

例：查询所有供应商不是1002，1003 的产品。



```
SELECT * FROM products WHERE vend_id NOT IN (1002,1003);
SELECT * FROM products WHERE vend_id !=1002 AND vend_id!=1003;
```

**MySQL中的NOT** MySQL支持使用NOT对IN、BETWEEN和EXISTS子句取反。

## 用通配符进行过滤（LIKE操作符）

**通配符(wildcard)** 用来匹配值的一部分的特殊字符。

**搜索模式(search pattern)** 由字面值、通配符或两者组合构成的搜索条件。

MySQL的通配符很有用，但这种功能是有代价的，通配符搜索的处理一般要比前面讨论的其他搜索所花时间更长。不要过度使用通配符，如果其他操作符能达到相同的目的，应该使用其他操作符。确实需要使用通配符时，除非绝对有必要，否则不要把它们用在搜索模式的开始处。把通配符置于搜索模式的开始处，搜索起来是最慢的。

## 百分号(%)通配符

最常使用的通配符是百分号(%)。在搜索串中，%表示任何字符出现任意次数。

```
SELECT * FROM 表名 WHERE 列名 LIKE '条件%'
```

例如，为了找出所有以词jet起头的顾客姓名，可使用以下SELECT 语句：

```
SELECT * FROM customers WHERE cust_name LIKE 'jet%'
```

jet结尾

```
SELECT * FROM customers WHERE cust_name LIKE '%jet'
```

包含jet

```
SELECT * FROM customers WHERE cust_name LIKE '%jet%'
```

注意：虽然似乎%通配符可以匹配任何东西，但有一个例外，即NULL。即使是WHERE prod\_name LIKE '%'也不能匹配用值NULL作为产品名的行。只能用下面语句匹配NULL

```
SELECT * FROM 表名 WHERE 列名 IS NULL;
```

## 下划线(\_)通配符

下划线的用途与%一样，但下划线只匹配单个字符而不是多个字符，\_总是匹配一个字符，不能多也不能少。

```
SELECT * FROM 表名 WHERE 列名 LIKE '条件%'
```

例：查询供应商ID（4位数）是100开头的，100\_

```
SELECT * FROM products WHERE vend_id LIKE '100_';
```

## 正则表达式

```
SELECT * FROM 表名 WHERE 列名 REGEXP '正则表达式'
```

Copyright © dujiaju.net 2020 all right reserved, powered by Gitbook该文件修订时间: 2020-01-11 11:46:55

- MySQL库函数
  - 文本处理函数
  - Soundex 函数（按音节查询）
  - Concat函数（拼接字符串）
  - Trim 函数（去除空格）
  - AS操作符（别名）
  - 算术操作符
  - 日期和时间处理函数
  - 数值处理函数
  - 聚集函数
  - 平均数函数（AVG）
  - 计数函数（COUNT）
  - 最大值函数（MAX）
  - 最小值函数（MIN）
  - 求和函数（SUM）
  - 排除重复值

## MySQL库函数

注意：函数没有SQL的可移植性强，大多数函数可能是MySQL独有的，或者名称不一样。

函数大多数按照如下分类。

- 用于处理文本串(如删除或填充值，转换值为大写或小写)的文本函数。
- 用于在数值数据上进行算术操作(如返回绝对值，进行代数运算)的数值函数。
- 用于处理日期和时间值并从这些值中提取特定成分(例如，返回两个日期之差，检查日期有效性等)的日期和时间函数。
- 返回DBMS正使用的特殊信息(如返回用户登录信息，检查版本细节)的系统函数。

### 文本处理函数

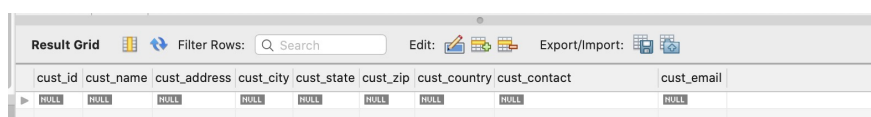
函数	说明
Left()	返回串左边的字符
Length()	返回串的长度
Locate()	找出串的一个子串
Lower()	将串转换为小写
LTrim()	去掉串左边的空格
Right()	返回串右边的字符
RTrim()	去掉串右边的空格
Soundex()	返回串的SOUNDEX值
SubString()	返回子串的字符
Upper()	将串转换为大写

## Soundex 函数（按音节查询）

SOUNDEX是一个将任何文本串转换为描述其语音表示的字母数字模式的算法。SOUNDEX考虑了类似的发音字符和音节，使得能对串进行发音比较而不是字母比较。虽然 SOUNDEX不是SQL概念，但MySQL(就像多数DBMS一样)都提供对SOUNDEX的支持。

下面给出一个使用Soundex()函数的例子。customers表中有一个顾客杰威尔公司，其联系名为周杰伦。但如果这是输入错误，成了周杰轮，怎么办？显然，按正确的联系名搜索不会返回数据，如下所示：

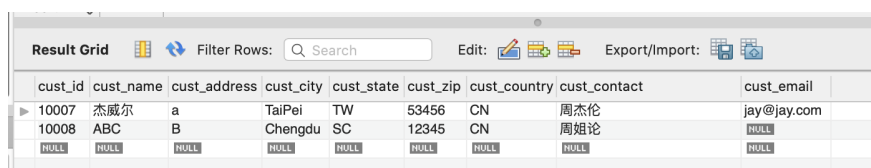
```
SELECT * FROM cc.customers where cust_contact='周杰轮';
```



cust_id	cust_name	cust_address	cust_city	cust_state	cust_zip	cust_country	cust_contact	cust_email
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

现在试一下使用Soundex()函数进行搜索，它匹配所有发音类似于 周杰伦的联系名：

```
SELECT * FROM cc.customers where Soundex(cust_contact)=Soundex('周杰伦');
```



cust_id	cust_name	cust_address	cust_city	cust_state	cust_zip	cust_country	cust_contact	cust_email
10007	杰威尔	a	Taipei	TW	53456	CN	周杰伦	jay@jay.com
10008	ABC	B	Chengdu	SC	12345	CN	周姐伦	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

## Concat函数（拼接字符串）

拼接(concatenate) 将值联结到一起构成单个值。

```
SELECT CONCAT(字符串1或者列名, '字符串2') FROM 表名;
```

例：把供应商名字和供应商国家连起来

```
SELECT CONCAT(vend_name, '(', vend_country, ')') FROM cc.vendors;
```

## Trim 函数（去除空格）

```
SELECT TRIM(列名) FROM 表名;
---- 去除左边空格
SELECT LTRIM(列名) FROM 表名;
---- 去除右边空格
SELECT RTRIM(列名) FROM 表名;
```

## AS操作符（别名）

```
SELECT 列名 AS 别名 FROM 表名;
```

## 算术操作符

操作符	说明
+	加
-	减
*	乘
/	除

```
SELECT 列名1 + 列名2或常量 FROM 表名;
```

## 日期和时间处理函数

日期和时间采用相应的数据类型和特殊的格式存储，以便能快速和有效地排序或过滤，并且节省物理存储空间。

函数	说明
AddDate()	增加一个日期(天、周等)
AddTime()	增加一个时间(时、分等)
CurDate()	返回当前日期
CurTime()	返回当前时间
Date()	返回日期时间的日期部分
DateDiff()	计算两个日期之差
Date_Add()	高度灵活的日期运算函数
Date_Format()	返回一个格式化的日期或时间串
Day()	返回一个日期的天数部分
DayOfWeek()	对于一个日期，返回对应的星期几
Hour()	返回一个时间的小时部分
Minute()	返回一个时间的分钟部分
Month()	返回一个日期的月份部分
Now()	返回当前日期和时间
Second()	返回一个时间的秒部分
Time()	返回一个日期时间的时间部分
Year()	返回一个日期的年份部分

## 数值处理函数

函数	说明
Abs()	返回一个数的绝对值
Cos()	返回一个角度的余弦
Exp()	返回一个数的指数值
Mod()	返回除操作的余数
Pi()	返回圆周率
Rand()	返回一个随机数
Sin()	返回一个角度的正弦
Sqrt()	返回一个数的平方根
Tan()	返回一个角度的正切

## 聚集函数

我们经常需要汇总数据而不用把它们实际检索出来，为此MySQL提供了专门的函数。使用这些函数，MySQL查询可用于检索数据，以便分析和报表生成。这种类型的检索例子有以下几种。

- 确定表中行数(或者满足某个条件或包含某个特定值的行数)。
- 获得表中行组的和。
- 找出表列(或所有行或某些特定的行)的最大值、最小值和平均值。

**聚集函数(aggregate function)** 运行在行组上，计算和返回单个值的函数。

函数	说明
AVG()	返回某列的平均值
COUNT()	返回某列的行数
MAX()	返回某列的最大值
MIN()	返回某列的最小值
SUM()	返回某列值之和

## 平均数函数 (AVG)

AVG函数，通过对表中行数计数并计算特定列值之和，求得该列的平均值。**NULL**值 AVG()函数忽略列值为NULL的行。

```
SELECT AVG(列名) FROM 表名;
```

## 计数函数 (COUNT)

COUNT()函数进行计数。可利用COUNT()确定表中行的数目或符合特定条件的行的数目。

COUNT()函数有两种使用方式。

- 使用COUNT(\*)对表中行的数目进行计数，不管表列中包含的是空值(NULL)还是非空值。

返回所有行

```
SELECT COUNT(*) FROM 表名;
```

- 使用COUNT(column)对特定列中具有值的行进行计数，忽略 NULL值。

返回所有非空行

```
SELECT COUNT(列名) FROM 表名;
```

## 最大值函数 (MAX)

返回列的最大值

```
SELECT MAX(列名) FROM 表名;
```

## 最小值函数 (MIN)

返回列的最小值

```
SELECT MIN(列名) FROM 表名;
```

## 求和函数 (SUM)

返回列的总和

```
SELECT SUM(列名) FROM 表名;
```

## 排除重复值

以上5个聚集函数都可以如下使用:

- 对所有的行执行计算, 指定ALL参数或不给参数(因为ALL是默认行为);
- 只包含不同的值, 指定DISTINCT参数。

**ALL**为默认 ALL参数不需要指定, 因为它是默认行为。如果 不指定DISTINCT, 则假定为ALL。

```
SELECT AVG(DISTINCT 列名) FROM 表名;
```

**注意:** 如果指定列名, 则DISTINCT只能用于COUNT()。DISTINCT 不能用于COUNT(\*), 因此不允许使用COUNT(DISTINCT), 否则会产生错误。类似地, DISTINCT必须使用列名, 不能用于计算或表达式。

将**DISTINCT**用于**MIN()**和**MAX()** 虽然DISTINCT从技术上可用于MIN()和MAX(), 但这样做实际上没有价值。一个列中的最小值和最大值不管是否包含不同值都是相同的。

Copyright © dujiaju.net 2020 all right reserved, powered by Gitbook该文件修订时间: 2020-01-12 10:15:03