show columns from name; //显示表列

desc name; //显示表列

describe name; //显示表列

第四章 检索数据

查询：

1.检索单个列

select part from name; //从name中检索一个名为part的列

2.检索多个列

select part1, part2 from name; //从name中检索名为part1和part2的列 //记得加逗号间隔

3.检索所有列

select \* from name; //检索所有列

4.检索不同的行 //去重

select distinct \* from name; //去重 //distinct应用于其后的所有列

//根据distinct后面的内容来进行去重判断，比如\*的话则必须所有列的内容相同才可去重，只是id的话，只要id重复就去重

5.选择行的输出

select part from name limit 5; //输出表中的前5行

select part from name limit 5,5; //向下偏移5行，输出表中的6-10行

select part from name limit 5,5; //指示MySQL返回从行5开始的5行 //标准解释

//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

MySQL 5支持的另一种替代语法

limit 4 offset 3 //从行3开始取4行，同 limit 3, 4

6.使用完全限定的表名

select name.part from name;

//表名也可以是完全限定的

select table.part from database.table;

//select exam\_result.id, exam\_result.name from self.exam\_result;

第五章 排序检索数据

//检索出的数据并不是以纯粹的随机顺序显示的，如果不排序，数据一般将以它在底层表中出现的顺序显示，这可以是数据最初添加到表中的顺序，但是，如果数据后来进行过更新或删除，则此顺序会受到MySQL重用回收存储空间的影响，因此，如果不明确规定排序顺序，则不应该假定检索出的数据的顺序有意义。

1.单列排序

//ASC为升序 (从小到大)

//DESC为降序 (从大到小)

//默认为ASC

select part from name order by part<可以和前面的part相同也可以不同> [ASC|DESC];

2.多列排序

select part from name order by part2, part3 [ASC|DESC]; //part3排序的条件part2在多个行具有相同的数据，如果part2列中的所有的值都是唯一的，则part3不起任何作用

//DESC关键字值应用到直接位于其前面的列名，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，ASC因为是默认的可以不写

第六章 过滤数据

where语句

1.检查单个值

select part1, part2 from name where condition;

2.不匹配检查

select part1, part2 from name where part3 <> ?? / part3 != ??; //<>是标准的 !=和<>没大区别

3.范围值检查

select part1, part2 from name where part3 between A and B;

select id,name from exam\_result where chinese between 60 and 80;

4.空值检查

select part1, part2 from name where part3 if [not] null;

第七章 数据过滤

1.组合where子句

1).AND操作符

select part1, part2 from name where condition AND condition;

2).OR操作符

select part1, part2 from name where condition OR condition;

3).计算次序

AND的优先级高于OR，如果不明确优先级，记得加上括号

2.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

select part1, part2 from name where part3 IN(??, ??); //IN和OR具有相同的功能

IN的优点：

1.在使用长的合法选项清单时，IN操作符的语法更清楚且更直观

2.在使用IN时，计算的次序更容易管理(因为使用的操作符更少)

3.IN操作符一般比OR操作符清单执行更快

4.最大的优点是可以包含其他SELECT语句，使得能够更动态地建立WHERE语句。

3.NOT操作符

select part1, part2 from name where NOT part3; //从前面选择不满足part3的

第八章 用通配符进行过滤

1.LIKE 操作符

1).百分号(%)通配符 //%表示任何字符出现任意次数 //不能匹配值为NULL

select part1, part2 from name where part3 LIKE '???%'; //显示part3中满足???开头的part1和part2

2).下划线(\_)通配符 //\_只能匹配单个字符

第九章 用正则表达式进行搜索 //只支持一部分正则表达式

1.基本字符匹配

select part1, part2 from name where part3 REGEXP '???'; //NULL查不到

select part1, part2 from name where part3 REGEXP '.???'; //.表示匹配任意字符 //和%相似

2.进行OR匹配

select part1, part2 from name where part3 REGEXP 'condition1 | condition2';

| 运算符与OR类似多个OR条件可并入单个正则表达式 'condition1 | condition2 | condition3'

3.匹配几个字符之一

select part1, part2 from name where part3 REGEXP '[condition1condition2condition3]???';

[]内的含义是匹配condition1或condition2或condition3，因此condition1???,condition2???,conditon3???都匹配且返回 <前提是存在>

[123] = [1|2|3]

[^123]匹配除这些字符外的任何东西 //^否定含义

4.匹配范围

[0123456789] = [0-9] //范围可以不连续

[a....z] = [a-z]

5.匹配特殊字符 比如正则表达式中的 |, [], .等必须以\\为前导

\\-表示查找-，\\.表示查找.

6.匹配字符类

截图

7.难以描述去看书吧

第十章 创建计算字段

1.拼接字段

Concat()函数拼接两个列

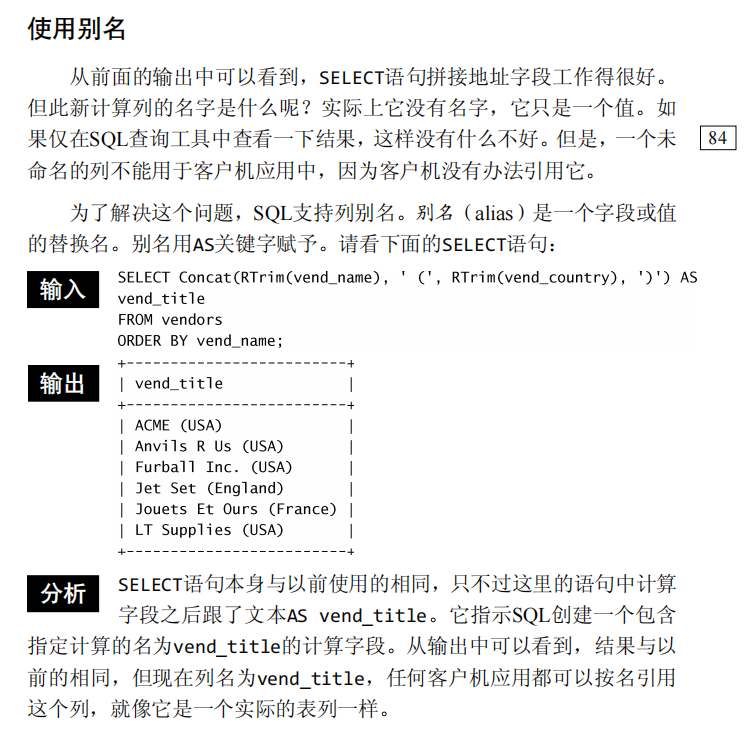


RTrim()函数:删除数据右侧多余的空格

LTrim()函数:删除数据左侧多余的空格

Trim()函数:删除数据两边的空格

2.别名

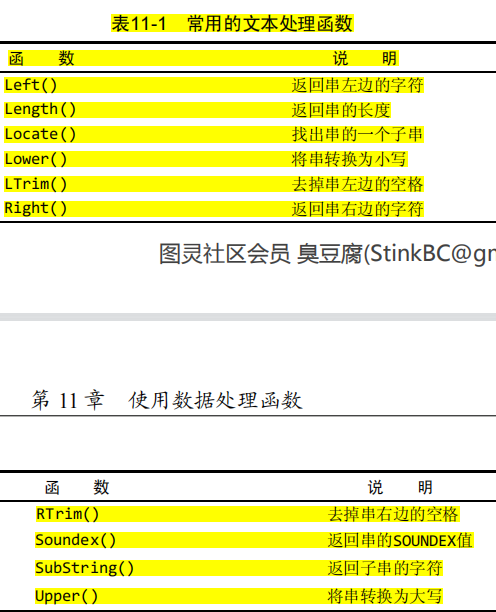


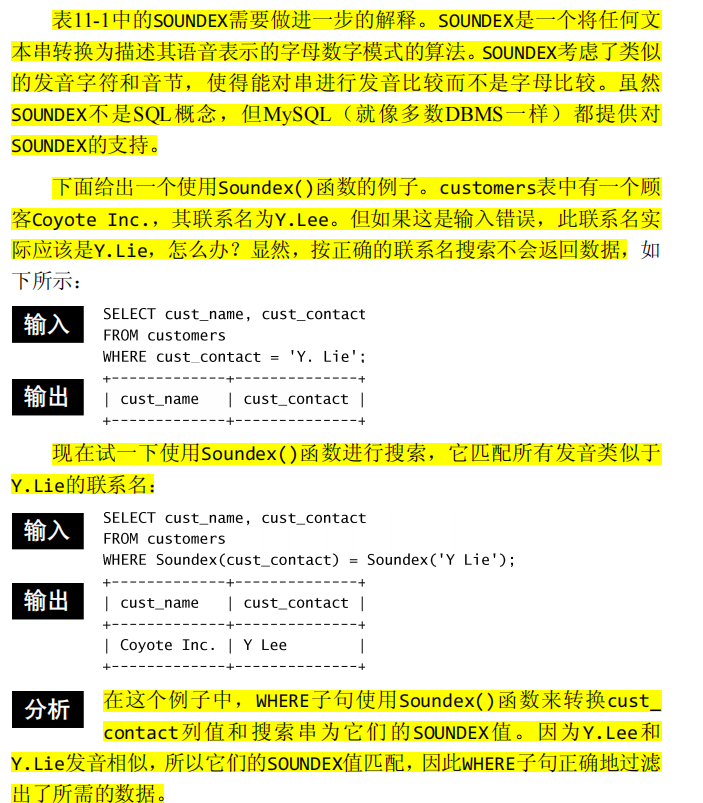
3.执行算术运算

没啥说的

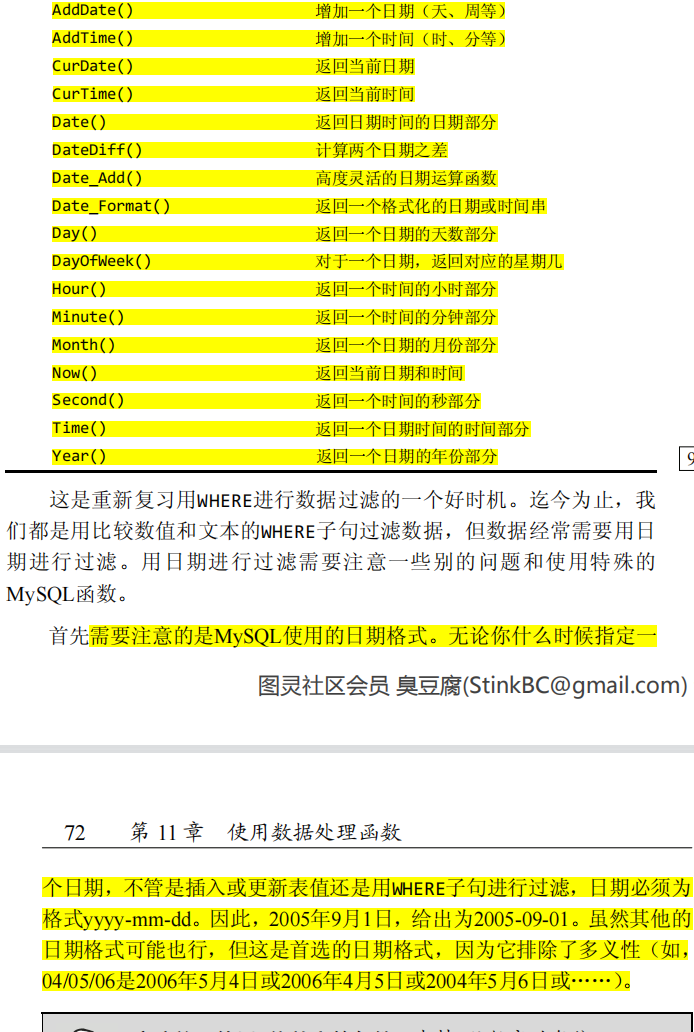
第十一章 使用数据处理函数 //函数的可移植性不强

1.文本处理函数

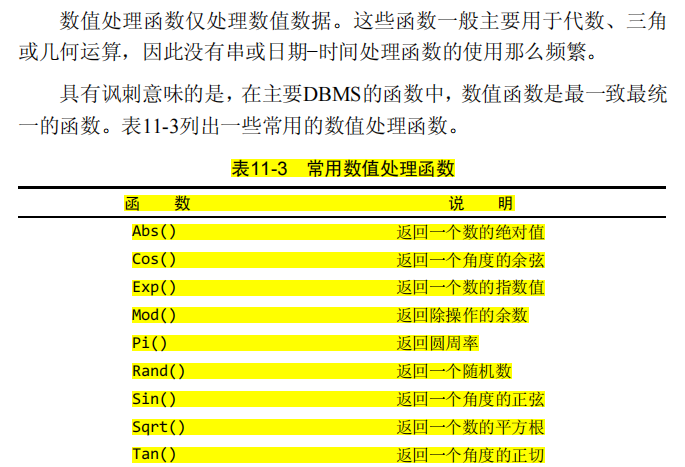




2.日期和时间处理函数 //没有验证



3.数值处理函数



第十二章 汇总数据

1.AVG() 返回某列的平均值

1).只能用于单个列，为了获得多个列的平均值，必须使用多个AVG()函数

2).NULL值 AVG()函数忽略值为NULL的行

2.COUNT() 返回某列的行数

1).COUNT(\*) 对表中行的数目进行计数，不管表中是否包含NULL

2).COUNT(column\_name) 对特定列中具有值的行进行计数，忽略NULL值

3.MAX() 返回某列的最大值

4.MIN() 返回某列的最小值

5.SUM() 返回某列值之和

注：

1.子句(clause) SQL语句由子句构成，有些子句是必需的，而有的是可选的，一个子句通常由一个关键字和所提供的数据组成。子句的例子有SELECT语句的FROM子句。

2.ORDER BY子句应该保证它位于FROM子句之后，如果使用LIMIT，它必须位于ORDER BY之后，WHERE在ORDER BY之前

3.MySQL支持使用NOT对IN、BETWEEN和EXISTS字句取反