GNU/Linux-MariaDB

第二章 数据查询

select语句

从一个或多个表中检索信息，检索表中数据需指定你要检索什么信息以及从哪里检索

检索单独的列

```
>select prod_name
  from products;
```

检索多列 （列名与列名之间使用逗号，但最后一个列名后不需要逗号）

```
>select  prod_id,prod_name,prod_price
  from  products;
```

SQL语句是不区分大小写的，因此SELECT与select是一样的，但是标识符（如数据库，表和列名），另外SQL语句可以用很长一行表示，或者分成多行，许多开发者认为将语句分成多行书写便于阅读且易于调试。

GNU/Linux-MariaDB

第二章 数据查询

检索所有列

```
>select *
  from products;
```

DISTINCT关键字

从products表中检索所有供应商ID不需要重复值

```
>select  distinct vend_id
   from  prouducts
```

注：distinct关键字应用于所有的列，而不仅是其后的一列。

GNU/Linux-MariaDB

第二章 数据查询

LIMIT子句

select语句返回所有匹配的行，但只想返回第一行或者前几行使用limit子句

```
>select prod_name
   from products
   limit  5;
```

上述语句的意思是从0行（检索的第一行是行0）开始显示5行，要显示5-10行执行以下语句

```
>select prod_name
  from  products
  limit 5,5;
```

同理limit 3,4即从第三行开始返回四行，这样特别容易混淆，MariaDB还支持 limit 4 offset 3 意思是从第三行开始获取四行，功能同上

GNU/Linux-MariaDB

第二章 数据查询

完全限定表名，列名检索

```
>select products.prod_name
  from crashcourse.prouducts;
```

注释

随着SQL语句长度和复杂度的增长，你将会需要包含一些描述性的注释以便将来参考或者接下来接手这个项目的人理解SQL语句，显然它们不是让MariaDB来处理而是嵌入到SQL脚本中。

注释同样试用于顶层的SQL文件，或许包含程序员交流信息，描述和注意事项等

临时阻止SQL代码的执行，例如你正在使用一条很长的SQL语句，但是你只要测试其中一部分，注释掉其他的就可以了

GNU/Linux-MariaDB

第二章 数据查询

注释的三种形式

>select prod_name --this is a comment “--”后面的内容是注释文本

from products;

># this is a comment 以”#”开头让整行称为注释

select prod_name

from products;

>/* select prod_name,vend_id “/*”开始一个注释，“*/”

from products; */ 结束一个注 释，通常用来注释掉代码

select prod_name

from products;

GNU/Linux-MariaDB

第二章 数据查询

Order by 子句

对使用select语句检索出的数据进行排序，默认以正序（ASC）排序(a-z)

```
>select prod_name
  from  products
  order  by prod_name;
```

通常 order by 子句中使用的列就是需要显示的列，然而用非检索的列排序也是完全合法的

GNU/Linux-MariaDB

第二章 数据查询

对多列进行排序

```
>select prod_id,prod_price,prod_name
  from products
  order by  prod_price,prod_name;
```

上述语句先对价格按照从高到底的顺序排序，价格相同的再按照名字正序排序

另一种表示方法（功能同上）

```
>select prod_id,prod_price,prod_name
  from products
  order by  2,3;      ---指定在select语句中指定的列的编号
```

GNU/Linux-MariaDB

第二章 数据查询

降序排序

```
>select prod_id,prod_price,prod_name
  from products
  order  by prod_price  desc;
>select prod_id,prod_price,prod_name
  from products
  order  by prod_price  desc,prod_name;
```

注：DESC仅作用于位于其前面的列名，因此prod_price按降序排序，而prod_name列（对同一价格）以升序排序。

GNU/Linux-MariaDB

第二章 数据查询

order by 与limit结合使用可显示prod_price列的最大值或最小值

```
>select prod_price
  from products
  order by prod_price desc
  limit 1;
```

各子句位置顺序

指定order by子句时要确保跟在from子句后；若使用limit，limit必须跟在order by之后。

GNU/Linux-MariaDB

第二章 数据查询

Where 子句

在select语句中可通过where子句指定搜索条件来过滤数据，where子句在表名（from子句）之后指定。

```
>select prod_name,prod_price
  from products
  where prod_price = 2.50;
```

若同时使用order by和where子句，要确保order by子句跟在where子句之后，否则会产生错误

GNU/Linux-MariaDB

第二章 数据查询

where子句支持的操作符

操作符	说明
=	相等
<>	不等
！=	不等
<	小于
<=	小于等于
>	大于
>=	大于等于
between	在两个特定值之间

GNU/Linux-MariaDB

第二章 数据查询

where子句示例

1.匹配prod_name为fuses的行

```
>select prod_name,prod_price
  from products
  where prod_name = 'fuses';
```

2.列出所有价格低于10的产品

```
>select prod_name,prod_price
  from  products
  where prod_price < 10;
```

GNU/Linux-MariaDB

第二章 数据查询

3.列出所有非供应商1003制造的产品

```
>select vend_id,prod_name
  from products
  where vend_id <>1003;
```

4.查询价格在5~10之间的所有产品

```
>select prod_name,prod_price
  from products
  where prod_price between 5 and 10;
```

5.检查空值

```
>select cust_id
  from customers
  where cust_email is NULL;
```

GNU/Linux-MariaDB

第二章 数据查询

上述where子句都是使用单一标准过滤数据，为了更好的控制过滤，MariaDB允许指定多个where子句，这些子句可以通过AND子句或OR子句实现。

操作符　在where子句中用来连接或者改变子句的关键字，也称为逻辑操作符。

AND操作符　对where子句添加条件

```
>select prod_id,prod_price,prod_name
  from products
  where vend_id=1003 and  prod_price <= 10;
```

GNU/Linux-MariaDB

第二章 数据查询

OR操作符 指示MariaDB检索匹配其中某个条件的行。

```
>select prod_name,prod_price
  from products
  where vend_id=1002 or vend_id=1003;
```

OR与AND优先级顺序

and的优先级要高于or

```
>select prod_name,prod_price
  from products
  where vend_id = 1002  or vend_id=1003 and prod_price >=10;
```

本意想要检索供应商1002或1003生产的，且价格高于或等于10的产品，上述语句将先执行and子句再执行or子句，它将读取供应商1002生产的产品，而不管其价格是多少，以及任何由供应商1003生产的价格高于或等于10的产品，由于优先级问题并未达到预期效果，解决的方法如下

GNU/Linux-MariaDB

第二章 数据查询

>select prod_name,prod_price
from products
where (vend_id = 1002 or vend_id=1003) and prod_price >=10

IN操作符 用来指定一组条件，只要匹配其中任何一个条件即可。

一个用于where子句的关键字，用来指定一系列使用or进行匹配 的值。

>select prod_name,prod_price
from products
where vend_id in (1002,1003)
order by prod_name;

检索所有由供应商1002和供应商1003提供的商品， where vend_id=1002 or vend_id=1003 与关键字in效果相同

IN操作符的优点

当你使用很长的值列表选项时，IN操作符语法更清晰易读

更容易管理优先级顺序

比一系列or操作符执行的快

可以创建较为动态的where子句

GNU/Linux-MariaDB

第二章 数据查询

NOT操作符　用来否定条件的where子句中一个关键字

列出除了供应商1002和1003所生产的所有产品

```
>select prod_name,prod_price
  from  products
  where vend_id not in (1002,1003)
  order by prod_name;
```

GNU/Linux-MariaDB

第二章 数据查询

使用通配符过滤数据

术语

通配符　　用来匹配值的某个部分的特殊字符

搜索模式　由纯文本，通配符，或者两者结合构成的匹配条件

LIKE操作符　指示MariaDB接下来的搜索模式是使用通配符匹配，而不是纯粹的相等匹配。

GNU/Linux-MariaDB

第二章 数据查询

通配符

通配符	说明
%	任意数量的字符（包括0个字符）
_	仅匹配一个字符

查找以jet开头的产品

```
>select prod_id,prod_name
   from products
   where prod_name like 'jek%';
```

GNU/Linux-MariaDB

第二章 数据查询

在搜索模式中，通配符可以用在任何地方，也可以使用多个通配符，如‘%anvil%'的意思是匹配任何位置包含文本anvil的值；’s%e'代表以s开头以e结尾的产品，但注意%无法匹配NULL

```
>select prod_id,prod_name
  from products
  where prod_name  like '_ ton anvil';
```

与”%”不同，可以匹配0-多个字符；“_”总是匹配一个字符，上述语句将返回匹配的1 ton anvil,2 ton anvil两行数据，若使用%通配符将会返回1 ton anvil,2 ton anvil,.5 ton anvil三行数据

GNU/Linux-MariaDB

第二章 数据查询

正则表达式搜索

正则表达式所做的事情是匹配文本，即通过一个模式（正则表达式）和文本字符串之间的比较。

REGEXP 操作符　指示MariaDB接下来的内容被当做正则表达式处理。

GNU/Linux-MariaDB

第二章 数据查询

正则表达式表示方法:

\ \ 忽略正则表达式中特殊字符的原有含义

^ 匹配以特定字符或者字符串开头的记录

$ 匹配以特定字符或者字符串结尾的记录

| select语句中使用or功能相似，将多个or的条件综合到一个 正则表达式中

[n - m] 匹配字符集合中任意一个字符

[^ n - m] 匹配除了字符集合中的任意一个字符

. 匹配字符串中任意一个字符,包括回车或者换行等

GNU/Linux-MariaDB

第二章 数据查询

示例

1.检索prod_name中包含X000的记录

```
>select prod_name
 from products
 where prod_name regexp '.000'
 order by prod_name;
```

注 MariaDB中的正则表达式匹配时不区分大小写，为强制区分大小写，可以使用binary关键字，如“where prod_name regexp binary 'JetPack .000'”

2.执行or匹配

```
>select prod_name
 from products
 where prod_name regexp '1000|2000|3000'
 order by prod_name;
```

3.定义字符集，匹配特定字符

```
>select prod_name
 from products
 where prod_name regexp '[123] ton'
 order by prod_name;
```

GNU/Linux-MariaDB

第二章 数据查询

4.匹配特殊字符”. ”

```
>select vend_name
  from vendors
  where vend_name regexp '\\.'
  order by vend_name;
```

“\\.”匹配”.”,因此只会返回Furball Inc. 一行数据

GNU/Linux-MariaDB

第二章 数据查询

“\\”也用于元字符（具有特殊含义的字符）

元字符	描述
\\f	换页符
\\n	换行符
\\r	回车
\\t	水平制表符
\\v	垂直制表符

你会发现自己会频繁使用许多匹配内容（数字，所有字母字符，或者所有数字和字母组成的字符等。为了简化工作，你可以使用预定义的字符集，即字符类。

GNU/Linux-MariaDB

第二章 数据查询

类	描述
[:alnum:]	任何字母或数字（同[a-zA-Z0-9]
[:alpha:]	任何字母（同[a-zA-Z]
[:blank:]	空格或者制表符（同[\\t])
[:cntrl:]	ASCII控制字符（ASCII表中0!～31和127
[:digit:]	任何数字（如[0-9])
[:graph :]	与[:print:]一样，不过去除空格
[:lower:]	任何小写字母（如[a-z]）
[:print:]	任何可打印的字符
[:punct:]	任何不在[:alnum:]和[:cntrl:]中的字符
[:space:]	任何空白字符包括空格（如同[\\f\\n\\r\\t\\v])
[:upper:]	任何大写字母（如[A-Z]）
[:xdigit:]	任何十六进制数字（如同[a-fA-F0-9])

GNU/Linux-MariaDB

第二章 数据查询

有时候你需要对匹配的数量进行更强大的控制，例如，你可能想要匹配所有的号码，而无论该号码包含多少个数字，或者你想要定位一个单词，并且也可适用于尾部有s的情况等等，可以通过适用正则表达式的量词元字符来实现

元字符	描述
*	匹配0个或多个
+	匹配1个或多个（等价于{1,}
?	匹配0个或1个（等价于{0,1}
{n}	匹配指定次数
{n,}	匹配至少n次
{m,n}	匹配一个范围（m不超过255）

GNU/Linux-MariaDB

第二章 数据查询

示例

1.输入

```
>select prod_name
  from prouducts
  where prod_name regexp '\\([0-9] sticks?\\)'
  order by prod_name;
```

输出

```
+---------------+
| prod_name     |
+---------------+
| TNT (1 stick) |
| TNT (5 sticks) |
+----------------+
```

其中'\\([0-9] sticks?\\)'，“\\(”匹配“（”，“[0-9]”匹配任何数字，“sticks？”匹配stick或sticks（?让s变得可选，因为?匹配0次或1次它前面的内容），“ \\) ”匹配闭括号“）”。

2.输入

```
>select prod_name
  from products
 where prod_name regexp '[[:digit:]]{4}'
 order by prod_name;
```

输出

```
+--------------+
| prod_name    |
+--------------+
| JetPack 1000 |
| JetPack 2000 |
+--------------+
```

其中'[[:digit:]]{4}'，'[[:digit:]]'匹配任何数字，”“{4}”需要四次精确匹配它遵循的内容（即任何数字），因此整体就是匹配任何四个连续的数字。当然一个特定的正则表达式总是可以用多种方法表示，前面这个例子还可以这样表示：

'[0-9][0-9][0-9][0-9]'或'[0-9]{4}'

3.匹配prod_name以“.”或者任何数字开头的字符串

```
>select prod_name
  from products
  where prod_name regexp '^[0-9\\.]'
  order by prod_name;
```

^的两种用法。在集合内用来否定集合，否则就是用来代表字符串的开始

GNU/Linux-MariaDB

第二章 数据查询

子查询　　子查询就是查询中又嵌套的查询,嵌套的级数随各数据库厂商的设定而有所不同,一般最大嵌套数不超过15级,实际应用中,一般不要超过2级,否则代码难以理解.一般来说,所有嵌套子查询都可改写为非嵌套的查询,但是这样将导致代码量增大.子查询就如递归函数一样,有时侯使用起来能达到事半功倍之效,只是其执行效率同样较低,有时用自身连接可代替某些子查询,另外,某些相关子查询也可改写成非相关子查询.

子查询常用于复杂的SQL操作中,包括select,insert,delete,update等语句中都可嵌套子查询.子查询分为两种:相关子查询和不相关子查询,顾名思义,相关子查询不能独自运行,必须依赖于外部父查询提供某些值才能运行.

GNU/Linux-MariaDB

第二章 数据查询

在此课件中使用的数据库表都是关系表，例如订单数据存储在两个表中，orders表每行存储一份订单，包括订单号，客户ID，订单日期；每个订单购买的物品存储在orderitems表中；orders表不包含客户信息，它只包含客户ID，实际的客户信息存储在customers表中。

GNU/Linux-MariaDB

第二章 数据查询

例 检索所有订单包含物品TNT2的客户信息

```
>select cust_name,cust_contact
  from customers
  where cust_id in (select cust_id
                          from orders
                           where order_num in (select order_num
                                                  from orderitems
                                                  where prod_id
='TNT2'));
```

子查询是从最内层的select语句开始往外处理，上述select语句处理完成，MariaDB实际上执行了三个操作。当然利用三个单独的select语句也可完成上述功能。

GNU/Linux-MariaDB

第二章 数据查询

子查询的另外一个用处是创建计算字段

例 显示customers表中每个客户订单总量，但是customers表中只有相关客户信息，订单信息存储在orders表中，两表唯一关联的就是cust_id

```
>select cust_name,cust_state,
        (select count(*)
         from orders
         where orders.cust_id=customers.cust_id) as orders
from customers
order by cust_name;
```

注 count（*）为聚合函数，表示计算表中行的总数

as 给列赋予别名

上述这种类型的查询叫做相关子查询，既引用外部查询的子查询，这种查询的语法要求当列名存在歧义时，必须完全限定列名。

创建安全的带有子查询的查询的方式是增量式的构建，即首先创建和测试最内层的查询，然后使用硬编码数据创建外部查询。验证嵌入的子查询可以正常运行时，接着再进行测试

GNU/Linux-MariaDB

第二章 数据查询

联合查询　　同时执行多个查询（即select语句）并且将结果作为一个单一查询结果集返回，又叫复合查询。在大多数情况下，对同一个表进行的多次查询的联合与使用多个where子句条件的单一查询完成同样的功能。SQL联合查询是通过UNION操作符实现的，指定select语句并且在他们之间放置union即可。

GNU/Linux-MariaDB

第二章 数据查询

例 检索价格低于或者等于5的产品列表的同时检索所有供应商为1001、1002的产品，不管价格是多少

```
>select vend_id,prod_id,prod_price
  from products
  where prod_price <=5
  union
  select vend_id,prod_id,prod_price
  from products
 where vend_id in (1001,1002);
```

GNU/Linux-MariaDB

第二章 数据查询

作为参考，以下语句可达到同样功能

```
select vend_id,prod_id,prod_price
   from products
   where prod_price <=5
   or vend_id  in (1001,1002);
```

UNION规则

1.一个union必须包含两个或两个以上select语句

2.在union中的每个查询都必须包含相同的列、表达式或者聚合函数

3.列数据类型必须是兼容的，他们不必是完全相同的类型，但必须是MariaDB能够转换的

4.union从查询结果集中自动移除任何重复行，如果要返回所有匹配结果，用union all 代替 union

5.当使用union联合查询时只可以使用一个order by子句，并且必须出现在最后一个select语句后面

6.union也适用于对不同的表进行联合查询

GNU/Linux-MariaDB

第二章 数据查询

全文本搜索　　使用全文本搜索时，MariaDB不需要单独地看每一行，不需要单独分析和处理每个词，而是创建单词索引（在指定的列），可针对这些单词进行搜索。MariaDB能够快速、有效地确定哪些单词匹配（那些行包含它们），那些不匹配，以及匹配的频率等。

但需要注意并非所有引擎都支持全文本搜索，MariaDB支持多种底层的数据库引擎。MariaDB ARIA引擎支持全文本搜索，并且在create.sql的CREATE TABLE 语句中，所有的crashcourse表都是使用ARIA引擎（通过指定ENGINE=Aria)创建的。为了执行全文本搜索，要搜索的列必须创建索引并且数据改变时重新建立索引，当表的列被制定后，MariaDB会自动创建索引或重新创建索引。创建完索引后，select语句可以与Match()和Against()一起使用来执行搜索。

GNU/Linux-MariaDB

第二章 数据查询

一般来说当一个表创建时全文本搜索是开启的。CREATE TABLE语句中接受FULLTEXT子句，这是一个以逗号隔开的要索引的列的列表。列：

```
create table productnotes
 （
note_id  int            not NULL atto_increment,
prod_id char(10)     not NULL,
note_date datetime  not NULL,
note_text  text        NULL,
primary key(note_id),
fulltext(note_text)
)enging=Aria;
```

GNU/Linux-MariaDB

第二章 数据查询

上述语句新建表productnotes，其中有一列叫做note_text，通过fulltext(note_text)建立索引，用于全文本搜索，fulltext可定义多列（用逗号隔开），fulltext可在创建表时指定，或者之后指定。特别提示导入数据时不要使用instfulltext。

建立索引后，使用Match()指定需要搜索的列，Against()指定需要使用的搜索表达式。

```
>select note_text
 from productnotes
 where match(note_text) against('rabbit');
```

执行结果将返回包含rabbit 的两行

注意：传递给match的值必须与fulltext()定义的值一样，若指定了多列，需要将所有指定列出且以正确的顺序；搜索时不区分大小写，除非使用binary

GNU/Linux-MariaDB

第二章 数据查询

事实上，上面的搜索可以用like子句实现，但二者返回的数据的顺序不同，前者会按照数据文本匹配的程度排序。

扩展查询 不仅匹配搜索标准的行，还匹配含有搜索标准返回行中出现过的单词的行，例：

```
>select note_text
 from  productnotes
where match(note_text) against('anvils' with query expansion)
```

自行与where match(note_text) against('anvils') 输出结果作比较

GNU/Linux-MariaDB

第二章 数据查询

布尔文本搜索 MariaDB支持额外的全文本搜索格式，叫做布尔模式（boolean mode），在没有定义fulltext索引也能使用。在布尔模式下可提供如下特定信息

匹配的单词

排除的单词(如果一行中包含这个单词，就不返回改行，即使匹配上其他指定的单词）

排位提示（指定哪些单词比其他单词更重要可以使其排位更高

表达式分组

GNU/Linux-MariaDB

第二章 数据查询

示例 匹配行中包含heavy单词，但是不包含以repo开头的单词

```
>select note_text
  from productnotes
  where match(note_text) against('heavy -repo*' in boolean mode);
```

注 “- ” “*”为两个全文本搜索布尔操作符

GNU/Linux-MariaDB

第二章 数据查询

全文本布尔操作符

操作符	描述
+	包含，必须出现的单词
_	排除，不能出现的单词
>	包含，增加排位值
<	包含，降低排位值
()	将单词分组为子表达式（允许它们被包含、排位、排除，如此作为一个组
~	对一个单词的排位值取否
*	单词末尾的通配符
“ ”	定义短语（相对与个别单词的列表，整个短语要么全部包含，要么全部排除）

GNU/Linux-MariaDB

第二章 数据查询

操作示例

1.搜索匹配同时包含rabbit和bait的行

```
>select note_text
from productnotes
where match(note_text) against('+rabbit +bait' in boolean mode);
```

2.搜索至少包含rabbit或bait一次的行

```
>select note_text
from productnotes
where match(note_text) against('rabbit bait' in boolean mode);
```

GNU/Linux-MariaDB

第二章 数据查询

3.同时匹配rabbit和carrot，提升前者排位，降低后者排位

```
>select note_text
from productnotes
where match(note_text) against('>rabbit  <carrot' in boolean mode);
```

4.同时匹配单词safe和combination,降低后者排位

```
>select note_text
from productnotes
where match(note_text) against('+safe +(<combination)' in boolean mode);
```

GNU/Linux-MariaDB

第二章 数据查询

连接

关系表　　设计关系表时,将信息分割到多个表中,每个表包含一种数据类型,表与表之间通过相同的值联系起来。例如products表仅存储产品信息，并且除了供应商ID（vendors表的主键）之外，没有供应商的其他信息。供应商ID称为外键（即来自其他表的主键值），通过它关联vendors表和products表。

这样将数据分割到多个表中，可以使存储更高效，操作更简洁，并且有更强的可扩展性。数据存储在多个表中，如何通过一个单独的select语句检索数据？答案是使用连接（join）。

GNU/Linux-MariaDB

第二章 数据查询

连接　使用select语句关联表与表之间的一种机制，使用一个特殊的语法，多个表连接起来，因此得到一个单一的输出集合。

连接类型

1.内连接

又称为等值连接，一个基于测试两表相等的连接

2.自连接

自连接常用来代替从相同的表中返回数据作为外部语句的子查询，尽管结果相同，但执行速度比子查询更快

3.自然连接

一种特殊的等值连接，它要求两个关系中进行比较的分量必须是相同的属性组，并且在结果中把重复的属性列去掉。而等值连接并不去掉重复的属性列。

GNU/Linux-MariaDB

第二章 数据查询

在连接条件中使用等于(=)运算符比较被连接列的列值，但它使用选择列表指出查询结果集合中所包括的列，并删除连接表中的重复列。

所谓自然连接就是在等值连接的情况下，当连接属性X与Y具有相同属性组时，把在连接结果中重复的属性列去掉。是在广义笛卡尔积R×S中选出同名属性上符合相等条件元组，再进行投影，去掉重复的同名属性，组成新的关系。

1）等值连接中不要求相等属性值的属性名相同，而自然连接要求相等属性值的属性名必须相同，即两关系只有在同名属性才能进行自然连接。

2）等值连接不将重复属性去掉，而自然连接去掉重复属性，也可以说，自然连接是去掉重复列的等值连接。事实上，我们一般使用的都是自然连接。

4．外连接

outer join会返回每个满足第一个（顶端）输入与第二个（底端）输入的联接的行。它还返回任何在第二个输入中没有匹配行的第一个输入中的行。外连接分为三种：左外连接，右外连接。对应SQL：LEFT/RIGHT JOIN。

在左外连接和右外连接时都会以一张表为基表，该表的内容会全部显示，然后加上两张表匹配的内容。如果基表的数据在另一张表没有记录。那么在相关联的结果集行中列显示为空值（NULL）。

GNU/Linux-MariaDB

第二章 数据查询

内连接(等值连接）

```
>select vend_name,prod_name,prod_price
from vendors,products
where vendors.vend_id=products.vend_id
order by vend_name,prod_name;
```

或者

```
>select vend_name,prod_name,prod_price
  from vendors  inner  join products
   on   vendors.vend_id=products.vend_id
  order by vend_name,prod_name;
```

一般采用第二种语法形式

GNU/Linux-MariaDB

第二章 数据查询

第一种语法形式若不加where字句，如下

```
>select vend_name,prod_name,prod_price
from vendors,products
order by vend_name,prod_name;
```

返回的数据匹配了每个供应商的每一个产品，包括错误供应商的产品（甚至有些供应商根本没有产品），把这种没有使用连接条件的表关系的返回结果称为**笛卡儿乘积**，返回的行数是第一个表的行数乘以第二个表的行数。这种返回笛卡儿乘积的连接叫做交叉连接。

GNU/Linux-MariaDB

第二章 数据查询

可以连接多个表，如子查询中的例子检索订购了TNT2产品的客户列表

```
>select cust_name,cust_contact
 from customers
  where cust_id in (select cust_id
                    from orders
                     where order_num in (select order_num
                                         from orderitems
                                         where prod_id ='TNT2'));
```

使用连接可进行同样的查询

```
>select cust_name,cust_contact
 from customers,orders,orderitems
 where customers.cust_id=orders.cust_id
 and orderitems.order_num=orders.order_num
 and prod_id='TNT2';
```

GNU/Linux-MariaDB

第二章 数据查询

SQL中可以给列起别名，也允许对表使用别名

例:

```
>select cust_name,cust_contact
    from customers as c,orders as o ,orderitems as oi
    where c.cust_id=o.cust_id
    and oi.order_num=o.order_num
    and prod_id='TNT2';
```

但表别名只能应用于查询，不同于列的别名，表别名决不会返回客户端

GNU/Linux-MariaDB

第二章 数据查询

自连接

例 现有一产品被发现存在问题（物品ID为DTNTR),因此要检索该产品供应商生产的所有产品，以确定其他产品是否也存在这个问题，此例有两种解决方案

1.子查询

```
>select prod_id,prod_name
  from products
  where vend_id=(select vend_id
                   from  products
                  where prod_id='DTNTR');
```

GNU/Linux-MariaDB

第二章 数据查询

2.自连接

```
>select p1.prod_id,p1.prod_name
  from products as p1,products as p2
  where p1.vend_id=p2.vend_id
  and  p2.prod_id='DTNTR';
```

GNU/Linux-MariaDB

第二章 数据查询

3．内连接　　实现方式是对一个表使用通配符(select*)和其他表使用列的明确子集

```
>select c.*,o.order_num,o.order_date,oi.prod_id,
        oi.quantity,oi.item_price
  from customers as c,orders as o, orderitems as oi
  where c.cust_id=o.cust_id
  and oi.order_num=o.order_num
  and prod_id='FB';
```

GNU/Linux-MariaDB

第二章 数据查询

4.外连接

```
>select customers.cust_id,orders.order_num
  from customers left outer join orders
  on customers.cust_id=orders.cust_id;
```

输出

```
+---------+-----------+
| cust_id | order_num |
+---------+-----------+
|   10001 |     20005 |
|   10001 |     20009 |
|   10002 |      NULL |
|   10003 |     20006 |
|   10004 |     20007 |
|   10005 |     20008 |
+---------+-----------+
```

GNU/Linux-MariaDB

第二章 数据查询

```
>select customers.cust_id,orders.order_num
  from customers right outer join orders
  on customers.cust_id=orders.cust_id;
```

输出

```
+---------+-----------+
| cust_id | order_num |
+---------+-----------+
|   10001 |     20005 |
|   10003 |     20006 |
|   10004 |     20007 |
|   10005 |     20008 |
|   10001 |     20009 |
+---------+-----------+
```

GNU/Linux-MariaDB

第二章 数据查询

聚合函数可与各种连接一起使用　　示例

１．获取下过订单的客户信息及其订单量

```
>select customers.cust_name,
  customers.cust_id,
  count(orders.order_num) as num_ord
  from customers.cust_id=orders.cust_id
  group by customers.cust_id;
```

GNU/Linux-MariaDB

第二章 数据查询

2.显示所有的客户信息，包括那些没有下过订单的客户

```
>select customers.cust_name,
  customers.cust_id,
  count(orders.order_num) as num_ord
  from  customers left outer join orders
  on  customers.cust_id=orders.cust_id
  group by customers.cust_id;
```

GNU/Linux-MariaDB

第二章 数据查询

示例 1 输出

```
+---------------+---------+---------+
| cust_name     | cust_id | num_ord |
+---------------+---------+---------+
| Coyote Inc.   |   10001 |       2 |
| Wascals       |   10003 |       1 |
| Yosemite Place |   10004 |       1 |
| E Fudd        |   10005 |       1 |
+---------------+---------+---------+
```

GNU/Linux-MariaDB

第二章 数据查询

示例 2　输出

```
+---------------+---------+---------+
| cust_name     | cust_id | num_ord |
+---------------+---------+---------+
| Coyote Inc.   |   10001 |       2 |
| Mouse House   |   10002 |       0 |
| Wascals       |   10003 |       1 |
| Yosemite Place|   10004 |       1 |
| E Fudd        |   10005 |       1 |
+---------------+---------+---------+
```