1.数据库结构

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_study |
+-----------------+
| customers       |
| orderitems      |
| orders          |
| productnotes    |
| products        |
| vendors         |
+-----------------+
6 rows in set (0.00 sec)
```

```
mysql> select * from customers;
+---------+---------------+--------------------+-----------+------------+----------+--------------+--------------+--------------------+
| cust_id | cust_name     | cust_address       | cust_city | cust_state | cust_zip | cust_country | cust_contact | cust_email         |
+---------+---------------+--------------------+-----------+------------+----------+--------------+--------------+--------------------+
|   10001 | Coyote Inc.   | 200 Maple Lane     | Detroit   | MI         | 44444    | USA          | Y Lee        | ylee@coyote.com    |
|   10002 | Mouse House   | 333 Fromage Lane   | Columbus  | OH         | 43333    | USA          | Jerry Mouse  | NULL               |
|   10003 | Wascals       | 1 Sunny Place      | Muncie    | IN         | 42222    | USA          | Jim Jones    | rabbit@wascally.com |
|   10004 | Yosemite Place | 829 Riverside Drive | Phoenix | AZ         | 88888    | USA          | Y Sam        | sam@yosemite.com   |
|   10005 | E Fudd        | 4545 53rd Street   | Chicago   | IL         | 54545    | USA          | E Fudd       | NULL               |
+---------+---------------+--------------------+-----------+------------+----------+--------------+--------------+--------------------+
5 rows in set (0.00 sec)
```

```
mysql> select * from orderitems;
+-----------+------------+---------+----------+------------+
| order_num | order_item | prod_id | quantity | item_price |
+-----------+------------+---------+----------+------------+
|     20005 |          1 | ANV01   |       10 |       5.99 |
|     20005 |          2 | ANV02   |        3 |       9.99 |
|     20005 |          3 | TNT2    |        5 |      10.00 |
|     20005 |          4 | FB      |        1 |      10.00 |
|     20006 |          1 | JP2000  |        1 |      55.00 |
|     20007 |          1 | TNT2    |      100 |      10.00 |
|     20008 |          1 | FC      |       50 |       2.50 |
|     20009 |          1 | FB      |        1 |      10.00 |
|     20009 |          2 | OL1     |        1 |       8.99 |
|     20009 |          3 | SLING   |        1 |       4.49 |
|     20009 |          4 | ANV03   |        1 |      14.99 |
+-----------+------------+---------+----------+------------+
11 rows in set (0.00 sec)
```

```
mysql> select * from orders;
+-----------+---------------------+---------+
| order_num | order_date          | cust_id |
+-----------+---------------------+---------+
|     20005 | 2005-09-01 00:00:00 |   10001 |
|     20006 | 2005-09-12 00:00:00 |   10003 |
|     20007 | 2005-09-30 00:00:00 |   10004 |
|     20008 | 2005-10-03 00:00:00 |   10005 |
|     20009 | 2005-10-08 00:00:00 |   10001 |
+-----------+---------------------+---------+
5 rows in set (0.00 sec)
```

```
mysql> select * from productnotes;
+---------+---------+---------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------+
| note_id | prod_id | note_date           | note_text
                          |
+---------+---------+---------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------+
|     101 | TNT2    | 2005-08-17 00:00:00 | Customer complaint:
Sticks not individually wrapped, too easy to mistakenly detonate all at once.
Recommend individual wrapping.                                                                                                                                          |
|     102 | OL1     | 2005-08-18 00:00:00 | Can shipped full, refills not available.
Need to order new can if refill needed.                                                                                                                                                                 |
|     103 | SAFE    | 2005-08-18 00:00:00 | Safe is combination locked, combination not provided with safe.
This is rarely a problem as safes are typically blown up or dropped by customers.                                                                                       |
|     104 | FC      | 2005-08-19 00:00:00 | Quantity varies, sold by the sack load.
All guaranteed to be bright and orange, and suitable for use as rabbit bait.                                                                                                                            |
|     105 | TNT2    | 2005-08-20 00:00:00 | Included fuses are short and have been known to detonate too quickly for some customers.
Longer fuses are available (item FU1) and should be recommended. |
|     106 | TNT2    | 2005-08-22 00:00:00 | Matches not included, recommend purchase of matches or detonator (item DTNTR).
                          |
|     107 | SAFE    | 2005-08-23 00:00:00 | Please note that no returns will be accepted if safe opened using explosives.
                          |
|     108 | ANV01   | 2005-08-25 00:00:00 | Multiple customer returns, anvils failing to drop fast enough or falling backwards on purchaser. Recommend that customer conside
rs using heavier anvils.  |
|     109 | ANV03   | 2005-09-01 00:00:00 | Item is extremely heavy. Designed for dropping, not recommended for use with slings, ropes, pulleys, or tightropes.
                          |
|     110 | FC      | 2005-09-01 00:00:00 | Customer complaint: rabbit has been able to detect trap, food apparently less effective now.
                          |
|     111 | SLING   | 2005-09-02 00:00:00 | Shipped unassembled, requires common tools (including oversized hammer).
                          |
|     112 | SAFE    | 2005-09-02 00:00:00 | Customer complaint:
Circular hole in safe floor can apparently be easily cut with handsaw.                                                                                                                                  |
|     113 | ANV01   | 2005-09-05 00:00:00 | Customer complaint:
Not heavy enough to generate flying stars around head of victim. If being purchased for dropping, recommend ANV02 or ANV03 instead.   |
|     114 | SAFE    | 2005-09-07 00:00:00 | Call from individual trapped in safe plummeting to the ground, suggests an escape hatch be added.
Comment forwarded to vendor.                        |
+---------+---------+---------------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------+
14 rows in set (0.00 sec)
```

```
mysql> select * from products;
+---------+---------+---------------+------------+-----------------------------------------------------------+
| prod_id | vend_id | prod_name     | prod_price | prod_desc                                                 |
+---------+---------+---------------+------------+-----------------------------------------------------------+
| ANV01   |    1001 | .5 ton anvil  |       5.99 | .5 ton anvil, black, complete with handy hook             |
| ANV02   |    1001 | 1 ton anvil   |       9.99 | 1 ton anvil, black, complete with handy hook and carrying case |
| ANV03   |    1001 | 2 ton anvil   |      14.99 | 2 ton anvil, black, complete with handy hook and carrying case |
| DTNTR   |    1003 | Detonator     |      13.00 | Detonator (plunger powered), fuses not included           |
| FB      |    1003 | Bird seed     |      10.00 | Large bag (suitable for road runners)                     |
| FC      |    1003 | Carrots       |       2.50 | Carrots (rabbit hunting season only)                      |
| FU1     |    1002 | Fuses         |       3.42 | 1 dozen, extra long                                       |
| JP1000  |    1005 | JetPack 1000  |      35.00 | JetPack 1000, intended for single use                     |
| JP2000  |    1005 | JetPack 2000  |      55.00 | JetPack 2000, multi-use                                   |
| OL1     |    1002 | Oil can       |       8.99 | Oil can, red                                              |
| SAFE    |    1003 | Safe          |      50.00 | Safe with combination lock                                |
| SLING   |    1003 | Sling         |       4.49 | Sling, one size fits all                                  |
| TNT1    |    1003 | TNT (1 stick) |       2.50 | TNT, red, single stick                                    |
| TNT2    |    1003 | TNT (5 sticks) |      10.00 | TNT, red, pack of 10 sticks                               |
+---------+---------+---------------+------------+-----------------------------------------------------------+
14 rows in set (0.00 sec)
```

```
mysql> select * from vendors;
+---------+----------------+-----------------+-------------+------------+----------+--------------+
| vend_id | vend_name      | vend_address    | vend_city   | vend_state | vend_zip | vend_country |
+---------+----------------+-----------------+-------------+------------+----------+--------------+
|    1001 | Anvils R Us    | 123 Main Street | Southfield  | MI         | 48075    | USA          |
|    1002 | LT Supplies    | 500 Park Street | Anytown     | OH         | 44333    | USA          |
|    1003 | ACME           | 555 High Street | Los Angeles | CA         | 90046    | USA          |
|    1004 | Furball Inc.   | 1000 5th Avenue | New York    | NY         | 11111    | USA          |
|    1005 | Jet Set        | 42 Galaxy Road  | London      | NULL       | N16 6PS  | England      |
|    1006 | Jouets Et Ours | 1 Rue Amusement | Paris       | NULL       | 45678    | France       |
+---------+----------------+-----------------+-------------+------------+----------+--------------+
6 rows in set (0.00 sec)
```

3.3了解数据库和表

```
SHOW DATABASES;
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| study              |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

```
USE study;
SHOW TABLES;
```

```
mysql> USE study;
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> SHOW TABLES;
+-----------------+
| Tables_in_study |
+-----------------+
| customers       |
| orderitems      |
| orders          |
| productnotes    |
| products        |
| vendors         |
+-----------------+
6 rows in set (0.00 sec)
```

```
SHOW COLUMNS FROM customers;
# 快捷实现为
DESC customers;
```

```
mysql> SHOW COLUMNS FROM customers;
+--------------+-----------+------+-----+---------+----------------+
| Field        | Type      | Null | Key | Default | Extra          |
+--------------+-----------+------+-----+---------+----------------+
| cust_id      | int(11)   | NO   | PRI | NULL    | auto_increment |
| cust_name    | char(50)  | NO   |     | NULL    |                |
| cust_address | char(50)  | YES  |     | NULL    |                |
| cust_city    | char(50)  | YES  |     | NULL    |                |
| cust_state   | char(5)   | YES  |     | NULL    |                |
| cust_zip     | char(10)  | YES  |     | NULL    |                |
| cust_country | char(50)  | YES  |     | NULL    |                |
| cust_contact | char(50)  | YES  |     | NULL    |                |
| cust_email   | char(255) | YES  |     | NULL    |                |
+--------------+-----------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

```
mysql> DESC customers;
+--------------+-----------+------+-----+---------+----------------+
| Field        | Type      | Null | Key | Default | Extra          |
+--------------+-----------+------+-----+---------+----------------+
| cust_id      | int(11)   | NO   | PRI | NULL    | auto_increment |
| cust_name    | char(50)  | NO   |     | NULL    |                |
| cust_address | char(50)  | YES  |     | NULL    |                |
| cust_city    | char(50)  | YES  |     | NULL    |                |
| cust_state   | char(5)   | YES  |     | NULL    |                |
| cust_zip     | char(10)  | YES  |     | NULL    |                |
| cust_country | char(50)  | YES  |     | NULL    |                |
| cust_contact | char(50)  | YES  |     | NULL    |                |
| cust_email   | char(255) | YES  |     | NULL    |                |
+--------------+-----------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

```
SHOW STATUS; # 用于显示广泛的服务器状态信息
```

```
mysql> SHOW STATUS;
+-----------------------------------------------+-----------------------------------------------
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
| Variable_name                                 | Value


+-----------------------------------------------+-----------------------------------------------
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
| Aborted_clients                               | 0

| Aborted_connects                              | 1

| Binlog_cache_disk_use                         | 0
```

```
SHOW CREATE DATABASE study;
```

```
mysql> SHOW CREATE DATABASE study;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| study    | CREATE DATABASE `study` /*!40100 DEFAULT CHARACTER SET latin1 */ |
+----------+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

```
SHOW CREATE TABLE customers;
```

```
mysql> SHOW CREATE TABLE customers;
+-----------+--------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------+
| Table     | Create Table
                                                                                                                        |
+-----------+--------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------+
| customers | CREATE TABLE `customers` (
  `cust_id` int(11) NOT NULL AUTO_INCREMENT,
  `cust_name` char(50) NOT NULL,
  `cust_address` char(50) DEFAULT NULL,
  `cust_city` char(50) DEFAULT NULL,
  `cust_state` char(5) DEFAULT NULL,
  `cust_zip` char(10) DEFAULT NULL,
  `cust_country` char(50) DEFAULT NULL,
  `cust_contact` char(50) DEFAULT NULL,
  `cust_email` char(255) DEFAULT NULL,
  PRIMARY KEY (`cust_id`)
) ENGINE=InnoDB AUTO_INCREMENT=10006 DEFAULT CHARSET=latin1 |
+-----------+--------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

```
# 显示服务器错误或警告信息
SHOW ERRORS;
SHOW WARNINGS;
HELP SHOW; # 显示帮助信息
```

```
mysql> HELP SHOW;
Name: 'SHOW'
Description:
SHOW has many forms that provide information about databases, tables,
columns, or status information about the server. This section describes
those following:

SHOW {BINARY | MASTER} LOGS
SHOW BINLOG EVENTS [IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count]
SHOW CHARACTER SET [like_or_where]
SHOW COLLATION [like_or_where]
SHOW [FULL] COLUMNS FROM tbl_name [FROM db_name] [like_or_where]
SHOW CREATE DATABASE db_name
SHOW CREATE EVENT event_name
SHOW CREATE FUNCTION func_name
SHOW CREATE PROCEDURE proc_name
SHOW CREATE TABLE tbl_name
```

2. 检索数据

SELECT

检索单个列

```
SELECT prod_name FROM products;
```

```
mysql> SELECT prod_name FROM products;
+----------------+
| prod_name      |
+----------------+
| .5 ton anvil   |
| 1 ton anvil    |
| 2 ton anvil    |
| Detonator      |
| Bird seed      |
| Carrots        |
| Fuses          |
| JetPack 1000   |
| JetPack 2000   |
| Oil can        |
| Safe           |
| Sling          |
| TNT (1 stick)  |
| TNT (5 sticks) |
+----------------+
14 rows in set (0.00 sec)
```

检索多个列

```
SELECT prod_id, prod_price, prod_name FROM products;
```

```
mysql> SELECT prod_id, prod_price, prod_name FROM products;
+---------+------------+----------------+
| prod_id | prod_price | prod_name      |
+---------+------------+----------------+
| ANV01   |       5.99 | .5 ton anvil   |
| ANV02   |       9.99 | 1 ton anvil    |
| ANV03   |      14.99 | 2 ton anvil    |
| DTNTR   |      13.00 | Detonator      |
| FB      |      10.00 | Bird seed      |
| FC      |       2.50 | Carrots        |
| FU1     |       3.42 | Fuses          |
| JP1000  |      35.00 | JetPack 1000   |
| JP2000  |      55.00 | JetPack 2000   |
| OL1     |       8.99 | Oil can        |
| SAFE    |      50.00 | Safe           |
| SLING   |       4.49 | Sling          |
| TNT1    |       2.50 | TNT (1 stick)  |
| TNT2    |      10.00 | TNT (5 sticks) |
+---------+------------+----------------+
14 rows in set (0.00 sec)
```

检索所有列

```
SELECT * FROM products;
```

```
mysql> SELECT * FROM products;
+---------+---------+---------------+------------+-----------------------------------------------------------------+
| prod_id | vend_id | prod_name     | prod_price | prod_desc                                                       |
+---------+---------+---------------+------------+-----------------------------------------------------------------+
| ANV01   |    1001 | .5 ton anvil  |       5.99 | .5 ton anvil, black, complete with handy hook                   |
| ANV02   |    1001 | 1 ton anvil   |       9.99 | 1 ton anvil, black, complete with handy hook and carrying case  |
| ANV03   |    1001 | 2 ton anvil   |      14.99 | 2 ton anvil, black, complete with handy hook and carrying case  |
| DTNTR   |    1003 | Detonator     |      13.00 | Detonator (plunger powered), fuses not included                 |
| FB      |    1003 | Bird seed     |      10.00 | Large bag (suitable for road runners)                           |
| FC      |    1003 | Carrots       |       2.50 | Carrots (rabbit hunting season only)                            |
| FU1     |    1002 | Fuses         |       3.42 | 1 dozen, extra long                                             |
| JP1000  |    1005 | JetPack 1000  |      35.00 | JetPack 1000, intended for single use                           |
| JP2000  |    1005 | JetPack 2000  |      55.00 | JetPack 2000, multi-use                                         |
| OL1     |    1002 | Oil can       |       8.99 | Oil can, red                                                    |
| SAFE    |    1003 | Safe          |      50.00 | Safe with combination lock                                      |
| SLING   |    1003 | Sling         |       4.49 | Sling, one size fits all                                        |
| TNT1    |    1003 | TNT (1 stick) |       2.50 | TNT, red, single stick                                          |
| TNT2    |    1003 | TNT (5 sticks) |      10.00 | TNT, red, pack of 10 sticks                                     |
+---------+---------+---------------+------------+-----------------------------------------------------------------+
14 rows in set (0.00 sec)
```

检索不同行

```
mysql> SELECT DISTINCT vend_id
    -> FROM products;
```

```
mysql> SELECT DISTINCT vend_id
    -> FROM products;
+---------+
| vend_id |
+---------+
|    1001 |
|    1002 |
|    1003 |
|    1005 |
+---------+
4 rows in set (0.00 sec)
```

DISTINCT应用于后边的所有字段

```
SELECT DISTINCT vend_id,prod_price  FROM products;
```

```
mysql> SELECT DISTINCT vend_id,prod_price  FROM products;
+---------+------------+
| vend_id | prod_price |
+---------+------------+
|    1001 |       5.99 |
|    1001 |       9.99 |
|    1001 |      14.99 |
|    1003 |      13.00 |
|    1003 |      10.00 |
|    1003 |       2.50 |
|    1002 |       3.42 |
|    1005 |      35.00 |
|    1005 |      55.00 |
|    1002 |       8.99 |
|    1003 |      50.00 |
|    1003 |       4.49 |
+---------+------------+
12 rows in set (0.00 sec)
```

限定结果

```
# 取前5个
mysql> SELECT prod_name
    -> FROM products
    -> LIMIT 5;
```

```
mysql> SELECT prod_name
    -> FROM products
    -> LIMIT 5;
+--------------+
| prod_name    |
+--------------+
| .5 ton anvil |
| 1 ton anvil  |
| 2 ton anvil  |
| Detonator    |
| Bird seed    |
+--------------+
5 rows in set (0.00 sec)
```

```
# 从第n行开始，取m个行
mysql> SELECT prod_name FROM products LIMIT 0,1;
```

```
mysql> SELECT prod_name FROM products LIMIT 0,1;
+--------------+
| prod_name    |
+--------------+
| .5 ton anvil |
+--------------+
1 row in set (0.00 sec)
```

使用完全限定的表名

```
mysql> SELECT products.prod_name
    -> FROM study.products;
```

```
mysql> SELECT products.prod_name
    -> FROM study.products;
+----------------+
| prod_name      |
+----------------+
| .5 ton anvil   |
| 1 ton anvil    |
| 2 ton anvil    |
| Detonator      |
| Bird seed      |
| Carrots        |
| Fuses          |
| JetPack 1000   |
| JetPack 2000   |
| Oil can        |
| Safe           |
| Sling          |
| TNT (1 stick)  |
| TNT (5 sticks) |
+----------------+
14 rows in set (0.00 sec)
```

3.排序检索数据

排序数据

```
# 不指定排序方向时，默认升序，A--Z，1--100
mysql> SELECT prod_name
    -> FROM study.products
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name
    -> FROM study.products
    -> ORDER BY prod_name;
+----------------+
| prod_name      |
+----------------+
| .5 ton anvil   |
| 1 ton anvil    |
| 2 ton anvil    |
| Bird seed      |
| Carrots        |
| Detonator      |
| Fuses          |
| JetPack 1000   |
| JetPack 2000   |
| Oil can        |
| Safe           |
| Sling          |
| TNT (1 stick)  |
| TNT (5 sticks) |
+----------------+
14 rows in set (0.00 sec)
```

按多个列排序

```
# 同一个prod_price的商品名字按升序排列
mysql> SELECT prod_id,prod_price,prod_name FROM products ORDER BY prod_price,prod_name;
```

```
mysql> SELECT prod_id,prod_price,prod_name FROM products ORDER BY prod_price,prod_name;
+---------+------------+----------------+
| prod_id | prod_price | prod_name      |
+---------+------------+----------------+
| FC      |       2.50 | Carrots        |
| TNT1    |       2.50 | TNT (1 stick)  |
| FU1     |       3.42 | Fuses          |
| SLING   |       4.49 | Sling          |
| ANV01   |       5.99 | .5 ton anvil   |
| OL1     |       8.99 | Oil can        |
| ANV02   |       9.99 | 1 ton anvil    |
| FB      |      10.00 | Bird seed      |
| TNT2    |      10.00 | TNT (5 sticks) |
| DTNTR   |      13.00 | Detonator      |
| ANV03   |      14.99 | 2 ton anvil    |
| JP1000  |      35.00 | JetPack 1000   |
| SAFE    |      50.00 | Safe           |
| JP2000  |      55.00 | JetPack 2000   |
+---------+------------+----------------+
14 rows in set (0.00 sec)
```

指定排序方向

DESC只对位于其前面的一个字段起作用，如果多个字段都要求降序，那么每个后面都要加DESC

```
# 按prod_price降序
mysql> SELECT prod_id,prod_price,prod_name
    -> FROM products
    -> ORDER BY prod_price DESC;
```

```
mysql> SELECT prod_id,prod_price,prod_name
    -> FROM products
    -> ORDER BY prod_price DESC;
+---------+------------+----------------+
| prod_id | prod_price | prod_name      |
+---------+------------+----------------+
| JP2000  |      55.00 | JetPack 2000   |
| SAFE    |      50.00 | Safe           |
| JP1000  |      35.00 | JetPack 1000   |
| ANV03   |      14.99 | 2 ton anvil    |
| DTNTR   |      13.00 | Detonator      |
| FB      |      10.00 | Bird seed      |
| TNT2    |      10.00 | TNT (5 sticks) |
| ANV02   |       9.99 | 1 ton anvil    |
| OL1     |       8.99 | Oil can        |
| ANV01   |       5.99 | .5 ton anvil   |
| SLING   |       4.49 | Sling          |
| FU1     |       3.42 | Fuses          |
| FC      |       2.50 | Carrots        |
| TNT1    |       2.50 | TNT (1 stick)  |
+---------+------------+----------------+
14 rows in set (0.00 sec)
```

```
mysql> SELECT prod_id,prod_price,prod_name FROM products ORDER BY prod_price DESC,prod_name;
```

```
mysql> SELECT prod_id,prod_price,prod_name FROM products ORDER BY prod_price DESC,prod_name;
+---------+------------+----------------+
| prod_id | prod_price | prod_name      |
+---------+------------+----------------+
| JP2000  |      55.00 | JetPack 2000   |
| SAFE    |      50.00 | Safe           |
| JP1000  |      35.00 | JetPack 1000   |
| ANV03   |      14.99 | 2 ton anvil    |
| DTNTR   |      13.00 | Detonator      |
| FB      |      10.00 | Bird seed      |
| TNT2    |      10.00 | TNT (5 sticks) |
| ANV02   |       9.99 | 1 ton anvil    |
| OL1     |       8.99 | Oil can        |
| ANV01   |       5.99 | .5 ton anvil   |
| SLING   |       4.49 | Sling          |
| FU1     |       3.42 | Fuses          |
| FC      |       2.50 | Carrots        |
| TNT1    |       2.50 | TNT (1 stick)  |
+---------+------------+----------------+
14 rows in set (0.00 sec)
```

ORDER BY 位于FROM 之后

```
mysql> SELECT prod_price
    -> FROM products
    -> ORDER BY prod_price DESC
    -> LIMIT 1;
+------------+
| prod_price |
+------------+
|      55.00 |
+------------+
1 row in set (0.00 sec)
```

4.过滤数据

使用WHERE子句

```
mysql> SELECT prod_name,prod_price
    -> FROM products
    -> WHERE prod_price = 2.5;
# 返回这两列的满足prod_price=2.5的行
```

```
mysql> SELECT prod_name,prod_price
    -> FROM products
    -> WHERE prod_price = 2.5;
+---------------+------------+
| prod_name     | prod_price |
+---------------+------------+
| Carrots       |       2.50 |
| TNT (1 stick) |       2.50 |
+---------------+------------+
2 rows in set (0.00 sec)
```

WHERE子句操作符使用

不区分大小写

```
mysql> SELECT prod_name,prod_price
    -> FROM products
    -> WHERE prod_name='fuses';
```

```
mysql> SELECT prod_name,prod_price
    -> FROM products
    -> WHERE prod_name='fuses';
+-----------+------------+
| prod_name | prod_price |
+-----------+------------+
| Fuses     |       3.42 |
+-----------+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT prod_name,prod_price FROM products WHERE prod_price<10;
```

```
mysql> SELECT prod_name,prod_price FROM products WHERE prod_price<10;
+---------------+------------+
| prod_name     | prod_price |
+---------------+------------+
| .5 ton anvil  |       5.99 |
| 1 ton anvil   |       9.99 |
| Carrots       |       2.50 |
| Fuses         |       3.42 |
| Oil can       |       8.99 |
| Sling         |       4.49 |
| TNT (1 stick) |       2.50 |
+---------------+------------+
7 rows in set (0.00 sec)
```

```
mysql> SELECT prod_name,prod_price FROM products WHERE prod_price≤10;
```

```
mysql> SELECT prod_name,prod_price FROM products WHERE prod_price<=10;
+----------------+------------+
| prod_name      | prod_price |
+----------------+------------+
| .5 ton anvil   |       5.99 |
| 1 ton anvil    |       9.99 |
| Bird seed      |      10.00 |
| Carrots        |       2.50 |
| Fuses          |       3.42 |
| Oil can        |       8.99 |
| Sling          |       4.49 |
| TNT (1 stick)  |       2.50 |
| TNT (5 sticks) |      10.00 |
+----------------+------------+
9 rows in set (0.00 sec)
```

```
mysql> SELECT vend_id,prod_name,prod_price FROM products WHERE vend_id<>1003;
# 不等于,和≠相同
```

```
mysql> SELECT vend_id,prod_name,prod_price FROM products WHERE vend_id<>1003;
+---------+--------------+------------+
| vend_id | prod_name    | prod_price |
+---------+--------------+------------+
|    1001 | .5 ton anvil |       5.99 |
|    1001 | 1 ton anvil  |       9.99 |
|    1001 | 2 ton anvil  |      14.99 |
|    1002 | Fuses        |       3.42 |
|    1005 | JetPack 1000 |      35.00 |
|    1005 | JetPack 2000 |      55.00 |
|    1002 | Oil can      |       8.99 |
+---------+--------------+------------+
7 rows in set (0.00 sec)
```

```
mysql> SELECT prod_name,prod_price FROM products WHERE prod_price BETWEEN 5 AND 10;
# 左右都是闭区间
```

```
mysql> SELECT prod_name,prod_price FROM products WHERE prod_price BETWEEN 5 AND 10;
+----------------+------------+
| prod_name      | prod_price |
+----------------+------------+
| .5 ton anvil   |       5.99 |
| 1 ton anvil    |       9.99 |
| Bird seed      |      10.00 |
| Oil can        |       8.99 |
| TNT (5 sticks) |      10.00 |
+----------------+------------+
5 rows in set (0.00 sec)
```

```
mysql> SELECT prod_name FROM products WHERE prod_price IS NULL;
```

```
mysql> SELECT prod_name FROM products WHERE prod_price IS NULL;
Empty set (0.00 sec)
```

```
mysql> SELECT cust_id FROM customers WHERE cust_email IS NULL;
```

```
mysql> SELECT cust_id FROM customers WHERE cust_email IS NULL;
+---------+
| cust_id |
+---------+
|   10002 |
|   10005 |
+---------+
2 rows in set (0.00 sec)
```

5.数据过滤

组合WHERE子句

AND

```
mysql> SELECT prod_id,prod_name,prod_price
    -> FROM products
    -> WHERE vend_id = 1003 AND prod_price ≤10;
```

```
mysql> SELECT prod_id,prod_name,prod_price
    -> FROM products
    -> WHERE vend_id = 1003 AND prod_price <=10;
+---------+----------------+------------+
| prod_id | prod_name      | prod_price |
+---------+----------------+------------+
| FB      | Bird seed      |      10.00 |
| FC      | Carrots        |       2.50 |
| SLING   | Sling          |       4.49 |
| TNT1    | TNT (1 stick)  |       2.50 |
| TNT2    | TNT (5 sticks) |      10.00 |
+---------+----------------+------------+
5 rows in set (0.00 sec)
```

OR

```
mysql> SELECT vend_id,prod_id,prod_name,prod_price
    -> FROM products
    -> WHERE vend_id = 1002 OR vend_id = 1003;
```

```
mysql> SELECT vend_id,prod_id,prod_name,prod_price FROM products WHERE vend_id = 1002 OR vend_id = 1003;
+---------+---------+----------------+------------+
| vend_id | prod_id | prod_name      | prod_price |
+---------+---------+----------------+------------+
|    1003 | DTNTR   | Detonator      |      13.00 |
|    1003 | FB      | Bird seed      |      10.00 |
|    1003 | FC      | Carrots        |       2.50 |
|    1002 | FU1     | Fuses          |       3.42 |
|    1002 | OL1     | Oil can        |       8.99 |
|    1003 | SAFE    | Safe           |      50.00 |
|    1003 | SLING   | Sling          |       4.49 |
|    1003 | TNT1    | TNT (1 stick)  |       2.50 |
|    1003 | TNT2    | TNT (5 sticks) |      10.00 |
+---------+---------+----------------+------------+
9 rows in set (0.00 sec)
```

计算次序

AND 优先于 OR,所以下面的例子一定要加括号

```
mysql> SELECT vend_id,prod_id,prod_name,prod_price
    -> FROM products
    -> WHERE (vend_id = 1002 OR vend_id = 1003) AND prod_price≤5;
```

```
mysql> SELECT vend_id,prod_id,prod_name,prod_price FROM products WHERE (vend_id = 1002 OR vend_id = 1003) AND prod_price<=5;
+---------+---------+--------------+------------+
| vend_id | prod_id | prod_name    | prod_price |
+---------+---------+--------------+------------+
|    1003 | FC      | Carrots      |       2.50 |
|    1002 | FU1     | Fuses        |       3.42 |
|    1003 | SLING   | Sling        |       4.49 |
|    1003 | TNT1    | TNT (1 stick) |       2.50 |
+---------+---------+--------------+------------+
4 rows in set (0.00 sec)
```

IN

```
mysql> SELECT prod_name,prod_price
    -> FROM products
    -> WHERE vend_id IN (1002,1003)
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name,prod_price
    -> FROM products
    -> WHERE vend_id IN (1002,1003)
    -> ORDER BY prod_name;
+----------------+------------+
| prod_name      | prod_price |
+----------------+------------+
| Bird seed      |      10.00 |
| Carrots        |       2.50 |
| Detonator      |      13.00 |
| Fuses          |       3.42 |
| Oil can        |       8.99 |
| Safe           |      50.00 |
| Sling          |       4.49 |
| TNT (1 stick)  |       2.50 |
| TNT (5 sticks) |      10.00 |
+----------------+------------+
9 rows in set (0.00 sec)
```

NOT

```
mysql> SELECT prod_name,prod_price
    -> FROM products
    -> WHERE vend_id NOT IN (1002,1003)
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name,prod_price FROM products WHERE vend_id NOT IN (1002,1003) ORDER BY prod_name;
+--------------+------------+
| prod_name    | prod_price |
+--------------+------------+
| .5 ton anvil |       5.99 |
| 1 ton anvil  |       9.99 |
| 2 ton anvil  |      14.99 |
| JetPack 1000 |      35.00 |
| JetPack 2000 |      55.00 |
+--------------+------------+
5 rows in set (0.00 sec)
```

6.使用通配符进行过滤

LIKE操作符

% ，表示任何字符出现任意次(包括0，但不包括NULL)

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE prod_name LIKE 'jet%';
```

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE prod_name LIKE 'jet%';
+---------+--------------+
| prod_id | prod_name    |
+---------+--------------+
| JP1000  | JetPack 1000 |
| JP2000  | JetPack 2000 |
+---------+--------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE prod_name LIKE '%anvil%';
```

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE prod_name LIKE '%anvil%';
+---------+--------------+
| prod_id | prod_name    |
+---------+--------------+
| ANV01   | .5 ton anvil |
| ANV02   | 1 ton anvil  |
| ANV03   | 2 ton anvil  |
+---------+--------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE prod_name LIKE 's%e';
```

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE prod_name LIKE 's%e';
+---------+-----------+
| prod_id | prod_name |
+---------+-----------+
| SAFE    | Safe      |
+---------+-----------+
1 row in set (0.00 sec)
```

下滑线（_）:只能匹配单个字符

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE prod_name LIKE '_ ton anvil';
```

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE prod_name LIKE '_ ton anvil';
+---------+-------------+
| prod_id | prod_name   |
+---------+-------------+
| ANV02   | 1 ton anvil |
| ANV03   | 2 ton anvil |
+---------+-------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE prod_name LIKE '% ton anvil';
```

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE prod_name LIKE '% ton anvil';
+---------+--------------+
| prod_id | prod_name    |
+---------+--------------+
| ANV01   | .5 ton anvil |
| ANV02   | 1 ton anvil  |
| ANV03   | 2 ton anvil  |
+---------+--------------+
3 rows in set (0.00 sec)
```

使用通配符的技巧

- 通配符搜索处理时间长，所以不要过度使用。
- 除非绝对必要，否则不要把通配符用在搜索模式开始处，因为这样是最慢的。
- 注意使用通配符的位置。

7.正则表达式

MYSQL仅支持正则表达式很小的一个子集

基本字符匹配

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '1000'
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '1000'
    -> ORDER BY prod_name;
+--------------+
| prod_name    |
+--------------+
| JetPack 1000 |
+--------------+
1 row in set (0.00 sec)
```

```
# .表示匹配任意一个字符
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '.000'
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name FROM products WHERE prod_name REGEXP '.000' ORDER BY prod_name;
+--------------+
| prod_name    |
+--------------+
| JetPack 1000 |
| JetPack 2000 |
+--------------+
2 rows in set (0.00 sec)
```

LIKE于REGEXP的差别

```
mysql> SELECT prod_name FROM products WHERE prod_name LIKE '1000' ORDER BY prod_name;
Empty set (0.00 sec)
```

这是将上面的REGEXP换为LIKE后的结果，没有返回值。因为LIKE是默认的匹配整个列值，如prod_name这一列有一个值为“JetPack 1000”的元素，那么使用LIKE就会从‘J’的位置开始匹配，直到‘0’的位置结束。而REGEXP的匹配是在列值内进行的，如果正则表达式中指定“^和$”定位符，就可以匹配整个列值了。

匹配不区分大小写，为区分大小写，可使用BINARY关键字，如WHERE prod_name REGEXP BINARY 'JetPack .000'

OR 匹配 ,匹配之一

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '1000|2000'
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '1000|2000'
    -> ORDER BY prod_name;
+--------------+
| prod_name    |
+--------------+
| JetPack 1000 |
| JetPack 2000 |
+--------------+
2 rows in set (0.00 sec)
```

匹配单一一个字符集合符号[]，集合的否定[^]

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '[123] Ton'
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '[123] Ton'
    -> ORDER BY prod_name;
+-------------+
| prod_name   |
+-------------+
| 1 ton anvil |
| 2 ton anvil |
+-------------+
2 rows in set (0.00 sec)
```

```
# 这里又涉及到了优先级，下面的正则意味包含'1'或者'2'或者'3 Ton'
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name
    -> REGEXP '1|2|3 Ton'
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name FROM products WHERE prod_name REGEXP '1|2|3 Ton' ORDER BY prod_name;
+---------------+
| prod_name     |
+---------------+
| 1 ton anvil   |
| 2 ton anvil   |
| JetPack 1000  |
| JetPack 2000  |
| TNT (1 stick) |
+---------------+
5 rows in set (0.00 sec)
```

```
# 修改
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name
    -> REGEXP '[1|2|3] Ton'
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name FROM products WHERE prod_name REGEXP '[1|2|3] Ton' ORDER BY prod_name;
+-------------+
| prod_name   |
+-------------+
| 1 ton anvil |
| 2 ton anvil |
+-------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name
    -> REGEXP '[^123] Ton'
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name FROM products WHERE prod_name REGEXP '[^123] Ton' ORDER BY prod_name;
+--------------+
| prod_name    |
+--------------+
| .5 ton anvil |
+--------------+
1 row in set (0.00 sec)
```

匹配范围

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name
    -> REGEXP '[1-5] Ton'
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name FROM products WHERE prod_name REGEXP '[1-5] Ton' ORDER BY prod_name;
+--------------+
| prod_name    |
+--------------+
| .5 ton anvil |
| 1 ton anvil  |
| 2 ton anvil  |
+--------------+
3 rows in set (0.00 sec)
```

匹配特殊字符，匹配【. - [ ] |】需要在前面加双斜杠

```
# 匹配包含.的
mysql> SELECT vend_name
    -> FROM vendors
    -> WHERE vend_name REGEXP '\\.'
    -> ORDER BY vend_name;
```

```
mysql> SELECT vend_name
    -> FROM vendors
    -> WHERE vend_name REGEXP '\\.'
    -> ORDER BY vend_name;
+--------------+
| vend_name    |
+--------------+
| Furball Inc. |
+--------------+
1 row in set (0.00 sec)
```

双斜杠也用来引用元字符

元字符	说明
\\f	换页
\\n	换行
\\r	回车
\\t	制表
\\v	纵向制表

\\\表示匹配反斜杠本身

匹配字符类

类	说明
[:alnum:]	任意字母和数字，同[a-zA-Z0-9]
[:alpha:]	任意字符,同[a-zA-Z]
[:blank:]	空格和制表,同[\\t]

匹配多个实例

元字符	说明
*	0个或多个匹配
+	1个或多个匹配等于{1,}
?	0个或1个匹配等于{0,1}
{n}	指定数目的匹配
{n,}	不少于置顶数目的匹配
{n,m}	匹配数目的范围，m<=255

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '\\([0-9] sticks?\\)'
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '\\([0-9] sticks?\\)'
    -> ORDER BY prod_name;
+----------------+
| prod_name      |
+----------------+
| TNT (1 stick)  |
| TNT (5 sticks) |
+----------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '[[:digit:]]{4}'
    -> ORDER BY prod_name;
```

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '[[:digit:]]{4}'
    -> ORDER BY prod_name;
+--------------+
| prod_name    |
+--------------+
| JetPack 1000 |
| JetPack 2000 |
+--------------+
2 rows in set (0.00 sec)
```

元字符	说明
^	文本开始
$	文本结束
[[:<:]]	词的开始
[[:>:]]	词的结束

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '^[0-9\\.]'
    -> ;
```

```
mysql> SELECT prod_name
    -> FROM products
    -> WHERE prod_name REGEXP '^[0-9\\.]'
    -> ;
+--------------+
| prod_name    |
+--------------+
| .5 ton anvil |
| 1 ton anvil  |
| 2 ton anvil  |
+--------------+
3 rows in set (0.00 sec)
```

8.创建计算字段

拼接字段

Mysql的不同之处：多数DBMS使用+或者||来实现拼接，MYSQL则使用Concat()函数来实现

```
mysql> SELECT CONCAT(vend_name, ' (', vend_country,')') FROM vendors ORDER BY vend_name;
```

```
mysql> SELECT CONCAT(vend_name, ' (', vend_country,')') FROM vendors ORDER BY vend_name;
+-----------------------------------------+
| CONCAT(vend_name, ' (', vend_country,')') |
+-----------------------------------------+
| ACME (USA)                              |
| Anvils R Us (USA)                       |
| Furball Inc. (USA)                      |
| Jet Set (England)                       |
| Jouets Et Ours (France)                 |
| LT Supplies (USA)                       |
+-----------------------------------------+
6 rows in set (0.00 sec)
```

原表

```
mysql> SELECT * FROM vendors;
+---------+----------------+----------------+-------------+------------+----------+--------------+
| vend_id | vend_name      | vend_address   | vend_city   | vend_state | vend_zip | vend_country |
+---------+----------------+----------------+-------------+------------+----------+--------------+
|    1001 | Anvils R Us    | 123 Main Street | Southfield  | MI         | 48075    | USA          |
|    1002 | LT Supplies    | 500 Park Street | Anytown     | OH         | 44333    | USA          |
|    1003 | ACME           | 555 High Street | Los Angeles | CA         | 90046    | USA          |
|    1004 | Furball Inc.   | 1000 5th Avenue | New York    | NY         | 11111    | USA          |
|    1005 | Jet Set        | 42 Galaxy Road  | London      | NULL       | N16 6PS  | England      |
|    1006 | Jouets Et Ours | 1 Rue Amusement | Paris       | NULL       | 45678    | France       |
+---------+----------------+----------------+-------------+------------+----------+--------------+
6 rows in set (0.00 sec)
```

```
mysql> SELECT CONCAT(vend_name, '----------is from (', vend_country,')') FROM vendors ORDER BY vend_name;
```

```
mysql> SELECT CONCAT(vend_name, '----------is from (', vend_country,')') FROM vendors ORDER BY vend_name;
+-----------------------------------------------------------+
| CONCAT(vend_name, '----------is from (', vend_country,')') |
+-----------------------------------------------------------+
| ACME----------is from (USA)                               |
| Anvils R Us----------is from (USA)                        |
| Furball Inc.----------is from (USA)                       |
| Jet Set----------is from (England)                        |
| Jouets Et Ours----------is from (France)                  |
| LT Supplies----------is from (USA)                        |
+-----------------------------------------------------------+
6 rows in set (0.00 sec)
```

去除右侧多余的空格，RTrim，左侧LTrim。两侧Trim

```
mysql> SELECT CONCAT(RTrim(vend_name), ' (', RTrim(vend_country),')')
    -> FROM vendors
    -> ORDER BY vend_name;
```

```
mysql> SELECT CONCAT(RTrim(vend_name), ' (', RTrim(vend_country),')') FROM vendors ORDER BY vend_name;
+-----------------------------------------------------+
| CONCAT(RTrim(vend_name), ' (', RTrim(vend_country),')') |
+-----------------------------------------------------+
| ACME (USA)                                          |
| Anvils R Us (USA)                                   |
| Furball Inc. (USA)                                  |
| Jet Set (England)                                   |
| Jouets Et Ours (France)                             |
| LT Supplies (USA)                                   |
+-----------------------------------------------------+
6 rows in set (0.00 sec)
```

使用别名(导出列)

```
mysql> SELECT CONCAT(RTrim(vend_name), ' (', RTrim(vend_country),')')
    -> AS vend_title
    -> FROM vendors
    -> ORDER BY vend_name;
```

```
mysql> SELECT CONCAT(RTrim(vend_name), ' (', RTrim(vend_country),')')  AS vend_title FROM vendors ORDER BY vend_name;
+------------------------+
| vend_title             |
+------------------------+
| ACME (USA)             |
| Anvils R Us (USA)      |
| Furball Inc. (USA)     |
| Jet Set (England)      |
| Jouets Et Ours (France) |
| LT Supplies (USA)      |
+------------------------+
6 rows in set (0.00 sec)
```

执行算数计算

```
mysql> SELECT prod_id,
    -> quantity,
    -> item_price,
    -> quantity*item_price AS expanded_price
    -> FROM orderitems
    -> WHERE order_num = 20005;
```

```
mysql> SELECT prod_id,
    -> quantity,
    -> item_price,
    -> quantity*item_price AS expanded_price
    -> FROM orderitems
    -> WHERE order_num = 20005;
+---------+----------+------------+----------------+
| prod_id | quantity | item_price | expanded_price |
+---------+----------+------------+----------------+
| ANV01   |       10 |       5.99 |          59.90 |
| ANV02   |        3 |       9.99 |          29.97 |
| TNT2    |        5 |      10.00 |          50.00 |
| FB      |        1 |      10.00 |          10.00 |
+---------+----------+------------+----------------+
4 rows in set (0.00 sec)
```

数量*单价=总价

mysql算数操作符

加 +　；减 -　；乘 *　；除 /　；

测验计算

```
mysql> SELECT NOW();
```

```
mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2020-11-24 09:58:16 |
+---------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT 2*3;
+-----+
| 2*3 |
+-----+
|   6 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT Trim(' ab  ');
+---------------+
| Trim(' ab  ') |
+---------------+
| ab            |
+---------------+
1 row in set (0.00 sec)
```

9.使用数据处理函数

文本处理函数

```
mysql> SELECT vend_name,Upper(vend_name) AS vend_name_upcase
    -> FROM vendors
    -> ORDER BY vend_name;
```

```
mysql> SELECT vend_name,Upper(vend_name) AS vend_name_upcase
    -> FROM vendors
    -> ORDER BY vend_name;
+----------------+------------------+
| vend_name      | vend_name_upcase |
+----------------+------------------+
| ACME           | ACME             |
| Anvils R Us    | ANVILS R US      |
| Furball Inc.   | FURBALL INC.     |
| Jet Set        | JET SET          |
| Jouets Et Ours | JOUETS ET OURS   |
| LT Supplies    | LT SUPPLIES      |
+----------------+------------------+
6 rows in set (0.00 sec)
```

常用的文本处理函数

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返回发音类似的串

```
mysql> SELECT cust_name,cust_contact
    -> FROM customers
    -> WHERE Soundex(cust_contact) = Soundex('Y Lie');
```

```
mysql> SELECT cust_name,cust_contact
    -> FROM customers
    -> WHERE Soundex(cust_contact) = Soundex('Y Lie');
+------------+--------------+
| cust_name  | cust_contact |
+------------+--------------+
| Coyote Inc. | Y Lee        |
+------------+--------------+
1 row in set (0.00 sec)
```

日期和时间处理函数

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

```
mysql> SELECT cust_id,order_num
    -> FROM orders
    -> WHERE order_date = '2005-09-01';
```

```
mysql> SELECT cust_id,order_num
    -> FROM orders
    -> WHERE order_date =
    -> '2005-09-01';
+---------+-----------+
| cust_id | order_num |
+---------+-----------+
|   10001 |     20005 |
+---------+-----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM orders;
+-----------+---------------------+---------+
| order_num | order_date          | cust_id |
+-----------+---------------------+---------+
|     20005 | 2005-09-01 00:00:00 |   10001 |
|     20006 | 2005-09-12 00:00:00 |   10003 |
|     20007 | 2005-09-30 00:00:00 |   10004 |
|     20008 | 2005-10-03 00:00:00 |   10005 |
|     20009 | 2005-10-08 00:00:00 |   10001 |
+-----------+---------------------+---------+
5 rows in set (0.00 sec)
```

```
mysql> desc orders;
+------------+----------+------+-----+---------+----------------+
| Field      | Type     | Null | Key | Default | Extra          |
+------------+----------+------+-----+---------+----------------+
| order_num  | int(11)  | NO   | PRI | NULL    | auto_increment |
| order_date | datetime | NO   |     | NULL    |                |
| cust_id    | int(11)  | NO   | MUL | NULL    |                |
+------------+----------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT cust_id,order_num
    -> FROM orders
    -> WHERE Date(order_date) = '2005-09-01';
```

```
mysql> SELECT cust_id,order_num
    -> FROM orders
    -> WHERE Date(order_date) = '2005-09-01';
+---------+-----------+
| cust_id | order_num |
+---------+-----------+
|   10001 |     20005 |
+---------+-----------+
1 row in set (0.00 sec)
```

查询出9月的所有订单

```
mysql> SELECT cust_id,order_num \
FROM orders \
WHERE Date(order_date) BETWEEN '2005-09-01' AND '2005-09-30';
```

```
mysql> SELECT cust_id,order_num FROM orders WHERE Date(order_date) BETWEEN '2005-09-01' AND '2005-09-30';
+---------+-----------+
| cust_id | order_num |
+---------+-----------+
|   10001 |     20005 |
|   10003 |     20006 |
|   10004 |     20007 |
+---------+-----------+
3 rows in set (0.01 sec)
```

但是如果2月有的年份是润年该怎么办，这么做

```
mysql> SELECT cust_id,order_num
    -> FROM orders
    -> WHERE Year(order_date)=2005 AND Month(order_date)=9;
```

```
mysql> SELECT cust_id,order_num
    -> FROM orders
    -> WHERE Year(order_date)=2005 AND Month(order_date)=9;
+---------+-----------+
| cust_id | order_num |
+---------+-----------+
|   10001 |     20005 |
|   10003 |     20006 |
|   10004 |     20007 |
+---------+-----------+
3 rows in set (0.00 sec)
```

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

10.汇总数据

聚集函数

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()	返回某列的值的和

AVG()函数默认忽略列值为NULL的行

```
mysql> SELECT AVG(prod_price) AS avg_price
    -> FROM products;
```

```
mysql> SELECT AVG(prod_price) AS avg_price
    -> FROM products;
+-----------+
| avg_price |
+-----------+
| 16.133571 |
+-----------+
1 row in set (0.00 sec)
```

返回vend_id为1003的价格的平均值

```
mysql> SELECT AVG(prod_price) AS avg_price
    -> FROM products
    -> WHERE vend_id=1003;
```

```
mysql> SELECT AVG(prod_price) AS avg_price
    -> FROM products
    -> WHERE vend_id=1003;
+-----------+
| avg_price |
+-----------+
| 13.212857 |
+-----------+
1 row in set (0.00 sec)
```

COUNT()

- COUNT(*)对表中行的数目进行计数，不管是否包含NULL

- COUNT(column)对特定列中具有值的行进行计数，忽略NULL值，也就是为NULL的不计数

```
mysql> SELECT COUNT(*) AS num_cust
    -> FROM customers;
```

```
mysql> SELECT COUNT(*) AS num_cust
    -> FROM customers;
+----------+
| num_cust |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(cust_email) AS num_cust
    -> FROM customers;
```

```
mysql> SELECT COUNT(cust_email) AS num_cust
    -> FROM customers;
+----------+
| num_cust |
+----------+
|        3 |
+----------+
1 row in set (0.00 sec)
```

MAX()

```
mysql> SELECT MAX(prod_price) AS max_price
    -> FROM products;
```

```
mysql> SELECT MAX(prod_price) AS max_price
    -> FROM products;
+-----------+
| max_price |
+-----------+
|     55.00 |
+-----------+
1 row in set (0.00 sec)
```

MAX()函数忽略列值为NULL的行

MIN()

```
mysql> SELECT MIN(prod_price) AS max_price FROM products;
```

```
mysql> SELECT MIN(prod_price) AS max_price FROM products;
+-----------+
| max_price |
+-----------+
|      2.50 |
+-----------+
1 row in set (0.00 sec)
```

SUM()函数,忽略NULL值

```
mysql> SELECT SUM(quantity) AS items_ordered
    -> FROM orderitems
    -> WHERE order_num = 20005;
```

```
mysql> SELECT SUM(quantity) AS items_ordered FROM orderitems WHERE order_num = 20005;
+--------------+
| items_ordered |
+--------------+
|           19 |
+--------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT quantity AS items_ordered FROM orderitems WHERE order_num = 20005;
+---------------+
| items_ordered |
+---------------+
|            10 |
|             3 |
|             5 |
|             1 |
+---------------+
4 rows in set (0.00 sec)
```

```
mysql> SELECT SUM(item_price*quantity) AS items_ordered FROM orderitems WHERE order_num = 20005;
```

```
mysql> SELECT SUM(item_price*quantity) AS items_ordered FROM orderitems WHERE order_num = 20005;
+--------------+
| items_ordered |
+--------------+
|       149.87 |
+--------------+
1 row in set (0.00 sec)
```

聚集不同的值

DISTINCT只考虑各个不同的价格

```
mysql> SELECT AVG(DISTINCT prod_price) AS avg_price
    -> FROM products
    -> WHERE vend_id = 1003;
+-----------+
| avg_price |
+-----------+
| 15.998000 |
+-----------+
1 row in set (0.00 sec)
```

组合聚集函数

```
mysql> SELECT COUNT(*) AS num_items,
    -> MIN(prod_price) AS price_min,
    -> MAX(prod_price) AS price_max,
    -> AVG(prod_price) AS price_avg
    -> FROM products;
```

```
mysql> SELECT COUNT(*) AS num_items,
    -> MIN(prod_price) AS price_min,
    -> MAX(prod_price) AS price_max,
    -> AVG(prod_price) AS price_avg
    -> FROM products;
+-----------+-----------+-----------+-----------+
| num_items | price_min | price_max | price_avg |
+-----------+-----------+-----------+-----------+
|        14 |      2.50 |     55.00 | 16.133571 |
+-----------+-----------+-----------+-----------+
1 row in set (0.00 sec)
```

11.分组数据

数据分组

返回供应商1003提供的产品数目

```
mysql> SELECT COUNT(*) AS num_prods
    -> FROM products
    -> WHERE vend_id = 1003;
```

```
mysql> SELECT COUNT(*) AS num_prods
    -> FROM products
    -> WHERE vend_id = 1003;
+-----------+
| num_prods |
+-----------+
|         7 |
+-----------+
1 row in set (0.00 sec)
```

创建分组

```
mysql> SELECT vend_id,COUNT(*) AS num_prods
    -> FROM products
    -> GROUP BY vend_id;
```

```
mysql> SELECT vend_id,COUNT(*) AS num_prods FROM products GROUP BY vend_id;
+---------+-----------+
| vend_id | num_prods |
+---------+-----------+
|    1001 |         3 |
|    1002 |         2 |
|    1003 |         7 |
|    1005 |         2 |
+---------+-----------+
4 rows in set (0.00 sec)
```

GROUP BY 指定按照vend_id排序并分组数据，这导致对每个vend_id而不是对整个表计算num_prods一次。

- GROUP BY子句可以包含任意数目的列。这使得能够对分组进行嵌套，为数据分组提供更细致的控制
- 如果在GROUP BY 子句中嵌套了分组，数据将在最后规定的分组上进行汇总。换句话说，在建立分组时，指定的所有列都一起计算（所以不能从个别的列取回数据）
- GROUP BY 子句中列出的每个列都必须是检索列或有效的表达式（但不能是聚集 函数）。如果在SELECT 中使用表达式，则必须在GROUP BY 子句中指定相同的表达式，不可以使用别名。
- 除了聚集函数以外，SELECT语句中的每个列都必须在GROUP BY 子句中给出。
- 如果分组列中具有NULL值，则NULL值将作为一个分组返回。如果列中有多行NULL值，他们将分为一组。
- GROUP BY 子句必须出现在WHERE子句之后，ORDER BY 子句之前。

过滤分组

WHERE 是用来过滤行的，在分组之前，HAVING 是用来过滤分组的，在分组之后，句法一样，只是关键字不同。

```
mysql> SELECT cust_id,COUNT(*) AS orders
    -> FROM orders
    -> GROUP BY cust_id
    -> HAVING COUNT(*)≥2;
```

```
mysql> SELECT cust_id,COUNT(*) AS orders
    -> FROM orders
    -> GROUP BY cust_id
    -> HAVING COUNT(*)>=2;
+---------+--------+
| cust_id | orders |
+---------+--------+
|   10001 |      2 |
+---------+--------+
1 row in set (0.00 sec)
```

```
mysql> SELECT vend_id,COUNT(*) AS num_prods
    -> FROM products
    -> WHERE prod_price≥10
    -> GROUP BY vend_id
    -> HAVING COUNT(*) ≥ 2;
```

```
mysql> SELECT vend_id,COUNT(*) AS num_prods FROM products WHERE prod_price>=10 GROUP BY ven
d_id HAVING COUNT(*) >= 2;
+---------+-----------+
| vend_id | num_prods |
+---------+-----------+
|    1003 |         4 |
|    1005 |         2 |
+---------+-----------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT vend_id,COUNT(*) AS num_prods
    -> FROM products
    -> GROUP BY vend_id
    -> HAVING COUNT(*) ≥ 2;
```

```
mysql> SELECT vend_id,COUNT(*) AS num_prods FROM products GROUP BY vend_id HAVING COUNT(*)
>= 2;
+---------+-----------+
| vend_id | num_prods |
+---------+-----------+
|    1001 |         3 |
|    1002 |         2 |
|    1003 |         7 |
|    1005 |         2 |
+---------+-----------+
4 rows in set (0.00 sec)
```

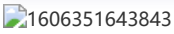
分组和排序

```
mysql> SELECT order_num,SUM(quantity*item_price) AS ordertotal
    -> FROM orderitems
    -> GROUP BY order_num
    -> HAVING SUM(quantity*item_price)≥50;
```

```
mysql> SELECT order_num,SUM(quantity*item_price) AS ordertotal
    -> FROM orderitems
    -> GROUP BY order_num
    -> HAVING SUM(quantity*item_price)>=50;
+-----------+------------+
| order_num | ordertotal |
+-----------+------------+
|     20005 |     149.87 |
|     20006 |      55.00 |
|     20007 |    1000.00 |
|     20008 |     125.00 |
+-----------+------------+
4 rows in set (0.00 sec)
```

使按总计订单价排序

```
mysql> SELECT order_num,SUM(quantity*item_price) AS ordertotal
    -> FROM orderitems
    -> GROUP BY order_num
    -> HAVING SUM(quantity*item_price)≥50
    -> ORDER BY ordertotal;
```



SELECT子句顺序

子句	说明	是否必须使用
SELECT	要返回的列或表达式	是
FROM	从中检索数据的表	仅在从表选择数据时使用
WHERE	行级过滤	否
GROUP BY	分组说明	仅在按组计算聚集时使用
HAVING	组级过滤	否
ORDER BY	输出排序顺序	否
LIMIT	要检索的行数	否

12.使用子查询

利用子查询进行过滤

1.检索包含物品TNT2的所有订单号

```
mysql> SELECT order_num
    -> FROM orderitems
    -> WHERE prod_id = 'TNT2';
```

```
mysql> SELECT order_num  FROM orderitems WHERE prod_id = 'TNT2';
+-----------+
| order_num |
+-----------+
|     20005 |
|     20007 |
+-----------+
2 rows in set (0.00 sec)
```

2.检索上述订单号所属的用户ID

```
mysql> SELECT cust_id
    -> FROM orders
    -> WHERE order_num IN (20005,20007);
```

```
mysql> SELECT cust_id
    -> FROM orders
    -> WHERE order_num IN (20005,20007);
+---------+
| cust_id |
+---------+
|   10001 |
|   10004 |
+---------+
2 rows in set (0.00 sec)
```

3.检索上述用户的信息

```
mysql> SELECT cust_name,cust_contact
    -> FROM customers
    -> WHERE cust_id IN (10001,10004);
```

```
mysql> SELECT cust_name,cust_contact FROM customers WHERE cust_id IN (10001,10004);
+----------------+--------------+
| cust_name      | cust_contact |
+----------------+--------------+
| Coyote Inc.    | Y Lee        |
| Yosemite Place | Y Sam        |
+----------------+--------------+
2 rows in set (0.00 sec)
```

将以上检索用一条语句

```
mysql> 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'));
```

```
mysql> 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'));
+----------------+--------------+
| cust_name      | cust_contact |
+----------------+--------------+
| Coyote Inc.    | Y Lee        |
| Yosemite Place | Y Sam        |
+----------------+--------------+
2 rows in set (0.00 sec)
```

作为计算字段使用子查询

相关子查询

```
mysql> SELECT cust_name,cust_state,(SELECT COUNT(*) FROM orders
    -> WHERE orders.cust_id = customers.cust_id) AS orders
    -> FROM customers
    -> ORDER BY cust_name;
```

```
mysql> SELECT cust_name,cust_state,(SELECT COUNT(*) FROM orders
  WHERE orders.cust_id = customers.cust_id) AS orders FROM customers ORDER BY cust_name;
+----------------+------------+--------+
| cust_name      | cust_state | orders |
+----------------+------------+--------+
| Coyote Inc.    | MI         |      2 |
| E Fudd         | IL         |      1 |
| Mouse House    | OH         |      0 |
| Wascals        | IN         |      1 |
| Yosemite Place | AZ         |      1 |
+----------------+------------+--------+
5 rows in set (0.00 sec)
```

13.联结表

如果数据存储在多个表中，怎样用单条SELECT语句检索出数据?

使用联结

创建联结

```
mysql> SELECT vend_name,prod_name,prod_price
    -> FROM vendors,products
    -> WHERE vendors.vend_id = products.vend_id
    -> ORDER BY vend_name,prod_name;
```

```
mysql> SELECT vend_name,prod_name,prod_price
    -> FROM vendors,products
    -> WHERE vendors.vend_id = products.vend_id
    -> ORDER BY vend_name,prod_name;
+-------------+----------------+------------+
| vend_name   | prod_name      | prod_price |
+-------------+----------------+------------+
| ACME        | Bird seed      |      10.00 |
| ACME        | Carrots        |       2.50 |
| ACME        | Detonator      |      13.00 |
| ACME        | Safe           |      50.00 |
| ACME        | Sling          |       4.49 |
| ACME        | TNT (1 stick)  |       2.50 |
| ACME        | TNT (5 sticks) |      10.00 |
| Anvils R Us | .5 ton anvil   |       5.99 |
| Anvils R Us | 1 ton anvil    |       9.99 |
| Anvils R Us | 2 ton anvil    |      14.99 |
| Jet Set     | JetPack 1000   |      35.00 |
| Jet Set     | JetPack 2000   |      55.00 |
| LT Supplies | Fuses          |       3.42 |
| LT Supplies | Oil can        |       8.99 |
+-------------+----------------+------------+
14 rows in set (0.00 sec)
```

实际运行时是利用了WHERE过滤后的数据进行了笛卡尔积。如果不加WHERE条件过滤，就会有很多的数据。

内部联结

table1 **INNER JOIN** table2 **ON** condition

```
mysql> SELECT vend_name,prod_name,prod_price
    -> FROM vendors INNER JOIN products
    -> ON vendors.vend_id = products.vend_id;
```

```
mysql> SELECT vend_name,prod_name,prod_price
    -> FROM vendors INNER JOIN products
    -> ON vendors.vend_id = products.vend_id;
+-------------+----------------+------------+
| vend_name   | prod_name      | prod_price |
+-------------+----------------+------------+
| Anvils R Us | .5 ton anvil   |       5.99 |
| Anvils R Us | 1 ton anvil    |       9.99 |
| Anvils R Us | 2 ton anvil    |      14.99 |
| LT Supplies | Fuses          |       3.42 |
| LT Supplies | Oil can        |       8.99 |
| ACME        | Detonator      |      13.00 |
| ACME        | Bird seed      |      10.00 |
| ACME        | Carrots        |       2.50 |
| ACME        | Safe           |      50.00 |
| ACME        | Sling          |       4.49 |
| ACME        | TNT (1 stick)  |       2.50 |
| ACME        | TNT (5 sticks) |      10.00 |
| Jet Set     | JetPack 1000   |      35.00 |
| Jet Set     | JetPack 2000   |      55.00 |
+-------------+----------------+------------+
14 rows in set (0.00 sec)
```

联结多个表

```
mysql> SELECT prod_name,vend_name,prod_price,quantity
    -> FROM orderitems,products,vendors
    -> WHERE products.vend_id=vendors.vend_id
    -> AND orderitems.prod_id = products.prod_id
    -> AND order_item = 20005;
```

```
mysql> SELECT prod_name,vend_name,prod_price,quantity FROM orderitems,products,vendors WHER
E products.vend_id=vendors.vend_id AND orderitems.prod_id = products.prod_id AND order_num
= 20005;
+----------------+-------------+------------+----------+
| prod_name      | vend_name   | prod_price | quantity |
+----------------+-------------+------------+----------+
| .5 ton anvil   | Anvils R Us |       5.99 |       10 |
| 1 ton anvil    | Anvils R Us |       9.99 |        3 |
| TNT (5 sticks) | ACME        |      10.00 |        5 |
| Bird seed      | ACME        |      10.00 |        1 |
+----------------+-------------+------------+----------+
4 rows in set (0.00 sec)
```

第14章的子查询可以使用联结查询替代

```
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';
```

```
mysql> 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';
+----------------+--------------+
| cust_name      | cust_contact |
+----------------+--------------+
| Coyote Inc.    | Y Lee        |
| Yosemite Place | Y Sam        |
+----------------+--------------+
2 rows in set (0.00 sec)
```

14.高级联结

表别名

```
mysql> SELECT Concat(RTrim(vend_name),' (',RTrim(vend_country),')') AS vend_title
    -> FROM vendors
    -> ORDER BY vend_name;
```

```
mysql> SELECT Concat(RTrim(vend_name),' (',RTrim(vend_country),')') AS vend_title FROM vend
ors ORDER BY vend_name;
+-------------------------+
| vend_title              |
+-------------------------+
| ACME (USA)              |
| Anvils R Us (USA)       |
| Furball Inc. (USA)      |
| Jet Set (England)       |
| Jouets Et Ours (France) |
| LT Supplies (USA)       |
+-------------------------+
6 rows in set (0.00 sec)
```

列别名

```
mysql> 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';
```

```
mysql> 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';
+----------------+--------------+
| cust_name      | cust_contact |
+----------------+--------------+
| Coyote Inc.    | Y Lee        |
| Yosemite Place | Y Sam        |
+----------------+--------------+
2 rows in set (0.00 sec)
```

表别名只在查询执行中使用，于列别名不一样，表别名不反回客户机。

自联结

找到物品ID为DTNTR的供应商，以及该供应商其他的产品。

以前使用这样的语句

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE vend_id = (SELECT vend_id
    -> FROM products
    -> WHERE prod_id = 'DTNTR');
```

```
mysql> SELECT prod_id,prod_name
    -> FROM products
    -> WHERE vend_id = (SELECT vend_id
    -> FROM products
    -> WHERE prod_id = 'DTNTR');
+---------+----------------+
| prod_id | prod_name      |
+---------+----------------+
| DTNTR   | Detonator      |
| FB      | Bird seed      |
| FC      | Carrots        |
| SAFE    | Safe           |
| SLING   | Sling          |
| TNT1    | TNT (1 stick)  |
| TNT2    | TNT (5 sticks) |
+---------+----------------+
7 rows in set (0.00 sec)
```

转换为自联结的方法

```
mysql> 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';
```

```
mysql> 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';
+---------+----------------+
| prod_id | prod_name      |
+---------+----------------+
| DTNTR   | Detonator      |
| FB      | Bird seed      |
| FC      | Carrots        |
| SAFE    | Safe           |
| SLING   | Sling          |
| TNT1    | TNT (1 stick)  |
| TNT2    | TNT (5 sticks) |
+---------+----------------+
7 rows in set (0.00 sec)
```

建议使用自联结而不是子查询

自然联结

```
mysql> 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';
```

```
mysql> mysql> 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';
+---------+------------+--------------+-----------+------------+----------+--------------+-------------+-----------------+-----------+---------------------+---------+----------+------------+
| cust_id | cust_name  | cust_address | cust_city | cust_state | cust_zip | cust_country | cust_contact | cust_email      | order_num | order_date          | prod_id | quantity | item_price |
+---------+------------+--------------+-----------+------------+----------+--------------+--------------+-----------------+-----------+---------------------+---------+----------+------------+
|   10001 | Coyote Inc. | 200 Maple Lane | Detroit   | MI         | 44444    | USA          | Y Lee        | ylee@coyote.com |     20005 | 2005-09-01 00:00:00 | FB      |        1 |      10.00 |
|   10001 | Coyote Inc. | 200 Maple Lane | Detroit   | MI         | 44444    | USA          | Y Lee        | ylee@coyote.com |     20009 | 2005-10-08 00:00:00 | FB      |        1 |      10.00 |
+---------+------------+--------------+-----------+------------+----------+--------------+--------------+-----------------+-----------+---------------------+---------+----------+------------+
2 rows in set (0.00 sec)
```

外部联结

LEFT OUTER JOIN

```
mysql> SELECT customers.cust_id,orders.order_num
    -> FROM customers LEFT OUTER JOIN orders
    -> ON customers.cust_id = orders.cust_id;
```

```
mysql> SELECT customers.cust_id,orders.order_num FROM customers LEFT OUTER JOIN orders ON customers.cust_id = o
rders.cust_id;
+---------+-----------+
| cust_id | order_num |
+---------+-----------+
|   10001 |     20005 |
|   10001 |     20009 |
|   10002 |      NULL |
|   10003 |     20006 |
|   10004 |     20007 |
|   10005 |     20008 |
+---------+-----------+
6 rows in set (0.00 sec)
```

使用带聚集函数的联结

```
mysql> SELECT customers.cust_name,
    -> customers.cust_id,
    -> COUNT(orders.order_num) AS num_prod
    -> FROM customers INNER JOIN orders
    -> ON customers.cust_id = orders.cust_id
    -> GROUP BY customers.cust_id;
```

```
mysql> SELECT customers.cust_name,
    -> customers.cust_id,
    -> COUNT(orders.order_num) AS num_prod
    -> FROM customers INNER JOIN orders
    -> ON customers.cust_id = orders.cust_id
    -> GROUP BY customers.cust_id;
+----------------+---------+----------+
| cust_name      | cust_id | num_prod |
+----------------+---------+----------+
| Coyote Inc.    |   10001 |        2 |
| Wascals        |   10003 |        1 |
| Yosemite Place |   10004 |        1 |
| E Fudd         |   10005 |        1 |
+----------------+---------+----------+
4 rows in set (0.00 sec)
```

外联结使用聚集函数

```
mysql> SELECT customers.cust_name,
    -> customers.cust_id,
    -> COUNT(orders.order_num) AS num_prod
    -> FROM customers LEFT OUTER JOIN orders
    -> ON customers.cust_id = orders.cust_id
    -> GROUP BY customers.cust_id;
```

```
mysql> SELECT customers.cust_name, customers.cust_id, COUNT(orders.order_num) AS num_prod FROM customers LEFT O
UTER JOIN orders ON customers.cust_id = orders.cust_id GROUP BY customers.cust_id;
+----------------+---------+----------+
| cust_name      | cust_id | num_prod |
+----------------+---------+----------+
| Coyote Inc.    |   10001 |        2 |
| Mouse House    |   10002 |        0 |
| Wascals        |   10003 |        1 |
| Yosemite Place |   10004 |        1 |
| E Fudd         |   10005 |        1 |
+----------------+---------+----------+
5 rows in set (0.00 sec)
```

15.组合查询

UNION

单条

```
SELECT vend_id,prod_id,prod_price
FROM products
WHERE prod_price ≤ 5;
```

```
mysql> SELECT vend_id,prod_id,prod_price
    -> FROM products
    -> WHERE prod_price <= 5;
+---------+---------+------------+
| vend_id | prod_id | prod_price |
+---------+---------+------------+
|    1003 | FC      |       2.50 |
|    1002 | FU1     |       3.42 |
|    1003 | SLING   |       4.49 |
|    1003 | TNT1    |       2.50 |
+---------+---------+------------+
4 rows in set (0.00 sec)
```

```
SELECT vend_id,prod_id,prod_price
FROM products
WHERE vend_id IN (1001,1002);
```

```
mysql> SELECT vend_id,prod_id,prod_price FROM products WHERE vend_id IN (1001,1002);
+---------+---------+------------+
| vend_id | prod_id | prod_price |
+---------+---------+------------+
|    1001 | ANV01   |       5.99 |
|    1001 | ANV02   |       9.99 |
|    1001 | ANV03   |      14.99 |
|    1002 | FU1     |       3.42 |
|    1002 | OL1     |       8.99 |
+---------+---------+------------+
5 rows in set (0.00 sec)
```

联合

```
mysql> SELECT vend_id,prod_id,prod_price
    -> FROM products
    -> WHERE prod_price ≤ 5
    -> UNION ALL
    -> SELECT vend_id,prod_id,prod_price
    -> FROM products
    -> WHERE vend_id IN (1001,1002);
```

```
mysql> 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);
+---------+---------+------------+
| vend_id | prod_id | prod_price |
+---------+---------+------------+
|    1003 | FC      |       2.50 |
|    1002 | FU1     |       3.42 |
|    1003 | SLING   |       4.49 |
|    1003 | TNT1    |       2.50 |
|    1001 | ANV01   |       5.99 |
|    1001 | ANV02   |       9.99 |
|    1001 | ANV03   |      14.99 |
|    1002 | OL1     |       8.99 |
+---------+---------+------------+
8 rows in set (0.00 sec)
```

还可使用多条WHERE子句

```
mysql> SELECT vend_id,prod_id,prod_price
    -> FROM products
    -> WHERE prod_price ≤ 5
    -> OR vend_id IN (1001,1002);
```

```
mysql> SELECT vend_id,prod_id,prod_price
    -> FROM products
    -> WHERE prod_price <= 5
    -> OR vend_id IN (1001,1002);
+---------+---------+------------+
| vend_id | prod_id | prod_price |
+---------+---------+------------+
|    1001 | ANV01   |       5.99 |
|    1001 | ANV02   |       9.99 |
|    1001 | ANV03   |      14.99 |
|    1003 | FC      |       2.50 |
|    1002 | FU1     |       3.42 |
|    1002 | OL1     |       8.99 |
|    1003 | SLING   |       4.49 |
|    1003 | TNT1    |       2.50 |
+---------+---------+------------+
8 rows in set (0.00 sec)
```

UNION规则

UNION中的每个查询必须包含相同的列、表达式或聚集函数

包含或取消重复的行

UNION ALL

```
SELECT vend_id,prod_id,prod_price
FROM products
WHERE prod_price ≤ 5
UNION ALL
SELECT vend_id,prod_id,prod_price
FROM products
WHERE vend_id IN (1001,1002);
```

```
mysql> SELECT vend_id,prod_id,prod_price
    -> FROM products
    -> WHERE prod_price <= 5
    -> UNION ALL
    -> SELECT vend_id,prod_id,prod_price
    -> FROM products
    -> WHERE vend_id IN (1001,1002);
+---------+---------+------------+
| vend_id | prod_id | prod_price |
+---------+---------+------------+
|    1003 | FC      |       2.50 |
|    1002 | FU1     |       3.42 |
|    1003 | SLING   |       4.49 |
|    1003 | TNT1    |       2.50 |
|    1001 | ANV01   |       5.99 |
|    1001 | ANV02   |       9.99 |
|    1001 | ANV03   |      14.99 |
|    1002 | FU1     |       3.42 |
|    1002 | OL1     |       8.99 |
+---------+---------+------------+
9 rows in set (0.00 sec)
```

返回值有两行是重复的，这是WHERE做不到的。

对组合结果排序

UNION不允许多条GROUP BY

```
mysql> SELECT vend_id,prod_id,prod_price
    -> FROM products
    -> WHERE prod_price ≤ 5
    -> UNION ALL
    -> SELECT vend_id,prod_id,prod_price
    -> FROM products
    -> WHERE vend_id IN (1001,1002)
    -> ORDER BY vend_id,prod_price;
```

```
mysql> SELECT vend_id,prod_id,prod_price
    -> FROM products
    -> WHERE prod_price <= 5
    -> UNION ALL
    -> SELECT vend_id,prod_id,prod_price
    -> FROM products
    -> WHERE vend_id IN (1001,1002)
    -> ORDER BY vend_id,prod_price;
+---------+---------+------------+
| vend_id | prod_id | prod_price |
+---------+---------+------------+
|    1001 | ANV01   |       5.99 |
|    1001 | ANV02   |       9.99 |
|    1001 | ANV03   |      14.99 |
|    1002 | FU1     |       3.42 |
|    1002 | FU1     |       3.42 |
|    1002 | OL1     |       8.99 |
|    1003 | FC      |       2.50 |
|    1003 | TNT1    |       2.50 |
|    1003 | SLING   |       4.49 |
+---------+---------+------------+
9 rows in set (0.00 sec)
```

16.全文本查询

启用全文本搜索支持

```
mysql> CREATE TABLE productexample(
    -> note_id              int         NOT NULL AUTO_INCREMENT,
    -> prod_id              char(10)    NOT NULL,
    -> note_date            datetime    NOT NULL,
    -> note_text            text        NULL,
    -> PRIMARY KEY(note_id),
    -> FULLTEXT(note_text)
    -> ) ENGINE=MyISAM;
```

进行全文本搜索，搜索不区分大小写

```
mysql> mysql> SELECT * FROM productnotes;
+---------+---------+---------------------+----------------------------------------------------------------------------------+
| note_id | prod_id | note_date           | note_text                                                                        |
+---------+---------+---------------------+----------------------------------------------------------------------------------+
|     101 | TNT2    | 2005-08-17 00:00:00 | Customer complaint:
Sticks not individually wrapped, too easy to mistakenly detonate all at once.
Recommend individual wrapping.                   |
|     102 | OL1     | 2005-08-18 00:00:00 | Can shipped full, refills not available.
Need to order new can if refill needed.                                         |
|     103 | SAFE    | 2005-08-18 00:00:00 | Safe is combination locked, combination not provided with safe.
This is rarely a problem as safes are typically blown up or dropped by customers.         |
|     104 | FC      | 2005-08-19 00:00:00 | Quantity varies, sold by the sack load.
All guaranteed to be bright and orange, and suitable for use as rabbit bait.            |
|     105 | TNT2    | 2005-08-20 00:00:00 | Included fuses are short and have been known to detonate too quickly for some customers.
Longer fuses are available (item FU1) and should be recommended. |
|     106 | TNT2    | 2005-08-22 00:00:00 | Matches not included, recommend purchase of matches or detonator (item DTNTR).  |
|     107 | SAFE    | 2005-08-23 00:00:00 | Please note that no returns will be accepted if safe opened using explosives.   |
|     108 | ANV01   | 2005-08-25 00:00:00 | Multiple customer returns, anvils failing to drop fast enough or falling backwards on purchaser. Recommend that customer considers using heavier anvils. |
|     109 | ANV03   | 2005-09-01 00:00:00 | Item is extremely heavy. Designed for dropping, not recommended for use with slings, ropes, pulleys, or tightropes. |
|     110 | FC      | 2005-09-01 00:00:00 | Customer complaint: rabbit has been able to detect trap, food apparently less effective now. |
|     111 | SLING   | 2005-09-02 00:00:00 | Shipped unassembled, requires common tools (including oversized hammer).        |
|     112 | SAFE    | 2005-09-02 00:00:00 | Customer complaint:
Circular hole in safe floor can apparently be easily cut with handsaw.                    |
|     113 | ANV01   | 2005-09-05 00:00:00 | Customer complaint:
Not heavy enough to generate flying stars around head of victim. If being purchased for dropping, recommend ANV02 or ANV03 instead.   |
|     114 | SAFE    | 2005-09-07 00:00:00 | Call from individual trapped in safe plummeting to the ground, suggests an escape hatch be added.
Comment forwarded to vendor.                   |
+---------+---------+---------------------+----------------------------------------------------------------------------------+
14 rows in set (0.00 sec)
```

对note_text这一列进行搜索,against()指定搜索内容

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE Match(note_text) Against('rabbit');
```

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE Match(note_text) Against('rabbit');
+----------------------------------------------------------------------------------------------+
| note_text                                                                                    |
+----------------------------------------------------------------------------------------------+
| Customer complaint: rabbit has been able to detect trap, food apparently less effective now. |
| Quantity varies, sold by the sack load.
All guaranteed to be bright and orange, and suitable for use as rabbit bait. |
+----------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

传递给Match的值必须与FULLTEXT()的相同。如果多个列，次序也要相同。

使用LIKE的相同查询,返回顺序不同，没有全文本搜索的优先级

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE note_text LIKE '%rabbit%';
```

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE note_text LIKE '%rabbit%';
+----------------------------------------------------------------------------------------------+
| note_text                                                                                    |
+----------------------------------------------------------------------------------------------+
| Quantity varies, sold by the sack load.
All guaranteed to be bright and orange, and suitable for use as rabbit bait. |
| Customer complaint: rabbit has been able to detect trap, food apparently less effective now. |
+----------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

机理：根据等级排序

```
mysql> SELECT note_text,Match(note_text) Against('rabbit') AS rank
    -> FROM productnotes;
```

```
mysql> mysql> SELECT note_text,Match(note_text) Against('rabbit') AS rank FROM productnotes;
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+--------------------+
| note_text                                                                                                                                                       | rank               |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+--------------------+
| Customer complaint:
Sticks not individually wrapped, too easy to mistakenly detonate all at once.
Recommend individual wrapping.                        |                0 |
| Can shipped full, refills not available.
Need to order new can if refill needed.                                                                  |                0 |
| Safe is combination locked, combination not provided with safe.
This is rarely a problem as safes are typically blown up or dropped by customers.         |                0 |
| Quantity varies, sold by the sack load.
All guaranteed to be bright and orange, and suitable for use as rabbit bait.                                    | 1.5905543565750122 |
| Included fuses are short and have been known to detonate too quickly for some customers.
Longer fuses are available (item FU1) and should be recommended. |                0 |
| Matches not included, recommend purchase of matches or detonator (item DTNTR).                                                                                 |                  0 |
| Please note that no returns will be accepted if safe opened using explosives.                                                                                   |                  0 |
| Multiple customer returns, anvils failing to drop fast enough or falling backwards on purchaser. Recommend that customer considers using heavier anvils.       |                  0 |
| Item is extremely heavy. Designed for dropping, not recommended for use with slings, ropes, pulleys, or tightropes.                                             |                  0 |
| Customer complaint: rabbit has been able to detect trap, food apparently less effective now.                                                                    | 1.6408053636550903 |
| Shipped unassembled, requires common tools (including oversized hammer).                                                                                        |                  0 |
| Customer complaint:
Circular hole in safe floor can apparently be easily cut with handsaw.                                                 |                0 |
| Customer complaint:
Not heavy enough to generate flying stars around head of victim. If being purchased for dropping, recommend ANV02 or ANV03 instead.   |                0 |
| Call from individual trapped in safe plummeting to the ground, suggests an escape hatch be added.
Comment forwarded to vendor.                          |                0 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+--------------------+
14 rows in set (0.00 sec)
```

可以看到不包含'rabbit'的行的rank为0，包含的行还根据‘rabbit'的位置计算rank，靠前的rank高。

使用查询扩展

一个简单的文本查询

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE Match(note_text) Against('anvils');
```

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE Match(note_text) Against('anvils');
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
| note_text                                                                                                                                                       |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Multiple customer returns, anvils failing to drop fast enough or falling backwards on purchaser. Recommend that customer considers using heavier anvils.       |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

这次使用查询扩展

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE Match(note_text) Against('anvils' WITH QUERY EXPANSION);
```

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE Match(note_text) Against('anvils');
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
| note_text                                                                                                                                                       |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Multiple customer returns, anvils failing to drop fast enough or falling backwards on purchaser. Recommend that customer considers using heavier anvils.       |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT note_text FROM productnotes WHERE Match(note_text) Against('anvils' WITH QUERY EXPANSION);
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
| note_text                                                                                                                                                       |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Multiple customer returns, anvils failing to drop fast enough or falling backwards on purchaser. Recommend that customer considers using heavier anvils.       |
| Customer complaint:
Sticks not individually wrapped, too easy to mistakenly detonate all at once.
Recommend individual wrapping.                        |
| Customer complaint:
Not heavy enough to generate flying stars around head of victim. If being purchased for dropping, recommend ANV02 or ANV03 instead.   |
| Please note that no returns will be accepted if safe opened using explosives.                                                                                   |
| Customer complaint: rabbit has been able to detect trap, food apparently less effective now.                                                                    |
| Customer complaint:
Circular hole in safe floor can apparently be easily cut with handsaw.                                                 |
| Matches not included, recommend purchase of matches or detonator (item DTNTR).                                                                                  |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
7 rows in set (0.00 sec)
```

这次返回了7行，第一行包含‘ anvils ‘所以等级最高，第二行不包含该词，但是包含了两个第一行有的词customer和recommend，第三行也有着两个词，但位置相对靠后，所以等级低。

布尔文本搜索 IN BOOLEAN MODE

即使没有定义FULLTEXT索引，也可以使用它。

为了匹配包含heavy但不包含任意以rope开始的词的行，可使用以下查询

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE Match(note_text) Against('heavy -rope*' IN BOOLEAN MODE);
```

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE Match(note_text) Against('heavy -rope*' IN BOOLEAN MODE);
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
| note_text                                                                                                                                                 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
| Customer complaint:
Not heavy enough to generate flying stars around head of victim. If being purchased for dropping, recommend ANV02 or ANV03 instead. |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

相较于下面，少了包含’ rope ‘的一行

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE Match(note_text) Against('heavy' IN BOOLEAN MODE);
```

```
mysql> SELECT note_text FROM productnotes WHERE Match(note_text) Against('heavy' IN BOOLEAN MODE);
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
| note_text                                                                                                                                                 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
| Item is extremely heavy. Designed for dropping, not recommended for use with slings, ropes, pulleys, or tightropes.                                      |
| Customer complaint:
Not heavy enough to generate flying stars around head of victim. If being purchased for dropping, recommend ANV02 or ANV03 instead. |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

全文本搜索布尔操作符-和*；-排除一个词，后者则是截断操作符。

布尔操作符	说明
+	包含，词必须存在
-	排除，词必须不出现
>	包含，而且增加等级值
<	包含，而且减少等级值
()	把词组成子表达式（允许这些子表达式被包含、排除，排列）
~	取消一个词的排序值
*	词尾的通配符
""	定义一个短语（与单个词的列表不一样，他匹配整个短语以包含或排除这个短语）

例子

1.包含词rabbit和bait的行

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE Match(note_text) Against('+rabbit +bait' IN BOOLEAN MODE);
```

```
mysql> SELECT note_text
    -> FROM productnotes
    -> WHERE Match(note_text) Against('+rabbit +bait' IN BOOLEAN MODE);
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
| note_text                                                                                                                                                 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
| Quantity varies, sold by the sack load.
All guaranteed to be bright and orange, and suitable for use as rabbit bait. |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

2.包含rabbit和bait中的至少一个词的行

```
mysql> SELECT note_text FROM productnotes WHERE Match(note_text) Against('rabbit bait' IN BOOLEAN MODE);
```

```
mysql> SELECT note_text FROM productnotes WHERE Match(note_text) Against('rabbit bait' IN BOOLEAN MODE);
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
| note_text                                                                                                                                                 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
| Quantity varies, sold by the sack load.
All guaranteed to be bright and orange, and suitable for use as rabbit bait. |
| Customer complaint: rabbit has been able to detect trap, food apparently less effective now.                                                              |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

3.匹配短语'rabbit bait'而不是两个分开的词

```
mysql> SELECT note_text FROM productnotes WHERE Match(note_text) Against('"rabbit bait"' IN BOOLEAN MODE);
```

```
mysql> SELECT note_text FROM productnotes WHERE Match(note_text) Against('"rabbit bait"' IN BOOLEAN MODE);
+-----------------------------------------------------------------------------------------------------------------------+
| note_text                                                                                                             |
+-----------------------------------------------------------------------------------------------------------------------+
| Quantity varies, sold by the sack load.
All guaranteed to be bright and orange, and suitable for use as rabbit bait. |
+-----------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

4.匹配rabbit和carrot，增加前者的等级，降低后者的等级

```
mysql> SELECT note_text FROM productnotes WHERE Match(note_text) Against('>rabbit <carrot' IN BOOLEAN MODE);
```

```
mysql> SELECT note_text FROM productnotes WHERE Match(note_text) Against('>rabbit <carrot' IN BOOLEAN MODE);
+-----------------------------------------------------------------------------------------------------------------------+
| note_text                                                                                                             |
+-----------------------------------------------------------------------------------------------------------------------+
| Quantity varies, sold by the sack load.
All guaranteed to be bright and orange, and suitable for use as rabbit bait. |
| Customer complaint: rabbit has been able to detect trap, food apparently less effective now.                          |
+-----------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

5.匹配safe 和combination，降低后者的等级

```
mysql> SELECT note_text FROM productnotes WHERE Match(note_text) Against('+safe +(<combination)' IN BOOLEAN MODE);
```

```
mysql> SELECT note_text FROM productnotes WHERE Match(note_text) Against('+safe +(<combination)' IN BOOLEAN MODE);
+-----------------------------------------------------------------------------------------------------------------------+
| note_text                                                                                                             |
+-----------------------------------------------------------------------------------------------------------------------+
| Safe is combination locked, combination not provided with safe.
This is rarely a problem as safes are typically blown up or dropped by customers. |
+-----------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

17.插入数据

数据插入

- 插入完整的行

```
INSERT INTO customers VALUES(NULL, 'D', 'A', 'S', 'SD', '9008', 'ASD', NULL, NULL);
```

```
mysql> INSERT INTO customers VALUES(NULL, 'D', 'A', 'S', 'SD', '9008', 'ASD', NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM customers;
+---------+---------------+---------------------+-----------+------------+----------+--------------+--------------+---------------------+
| cust_id | cust_name     | cust_address        | cust_city | cust_state | cust_zip | cust_country | cust_contact | cust_email          |
+---------+---------------+---------------------+-----------+------------+----------+--------------+--------------+---------------------+
|   10001 | Coyote Inc.   | 200 Maple Lane      | Detroit   | MI         | 44444    | USA          | Y Lee        | ylee@coyote.com     |
|   10002 | Mouse House   | 333 Fromage Lane    | Columbus  | OH         | 43333    | USA          | Jerry Mouse  | NULL                |
|   10003 | Wascals       | 1 Sunny Place       | Muncie    | IN         | 42222    | USA          | Jim Jones    | rabbit@wascally.com |
|   10004 | Yosemite Place | 829 Riverside Drive | Phoenix   | AZ         | 88888    | USA          | Y Sam        | sam@yosemite.com    |
|   10005 | E Fudd        | 4545 53rd Street    | Chicago   | IL         | 54545    | USA          | E Fudd       | NULL                |
|   10006 | D             | A                   | S         | SD         | 9008     | ASD          | NULL         | NULL                |
+---------+---------------+---------------------+-----------+------------+----------+--------------+--------------+---------------------+
6 rows in set (0.00 sec)
```

cust_id设置的是自增类型，插入时指定NULL就可以了。虽然这种语法简单，但是并不安全，应避免使用。编写依赖于特定次序的SQL语句是很不安全的。安全的做法如下

- 插入行的一部分

指定想要添加的列，这里没有cust_id，因为他是自增的

```
INSERT INTO customers(cust_name,
cust_address,
cust_state,
cust_email,
cust_contact,
cust_country,
cust_city,
cust_zip)
VALUES('as','ss','cs','vs','ccs','wws',NULL,NULL);
```

```
mysql> SELECT * FROM customers;
+---------+---------------+--------------------+-----------+------------+----------+--------------+--------------+--------------------+
| cust_id | cust_name     | cust_address       | cust_city | cust_state | cust_zip | cust_country | cust_contact | cust_email         |
+---------+---------------+--------------------+-----------+------------+----------+--------------+--------------+--------------------+
|   10001 | Coyote Inc.   | 200 Maple Lane     | Detroit   | MI         | 44444    | USA          | Y Lee        | ylee@coyote.com    |
|   10002 | Mouse House   | 333 Fromage Lane   | Columbus  | OH         | 43333    | USA          | Jerry Mouse  | NULL               |
|   10003 | Wascals       | 1 Sunny Place      | Muncie    | IN         | 42222    | USA          | Jim Jones    | rabbit@wascally.com |
|   10004 | Yosemite Place | 829 Riverside Drive | Phoenix   | AZ         | 88888    | USA          | Y Sam        | sam@yosemite.com   |
|   10005 | E Fudd        | 4545 53rd Street   | Chicago   | IL         | 54545    | USA          | E Fudd       | NULL               |
|   10006 | D             | A                  | S         | SD         | 9008     | ASD          | NULL         | NULL               |
|   10007 | a             | s                  | NULL      | c          | NULL     | ww           | cc           | v                  |
+---------+---------------+--------------------+-----------+------------+----------+--------------+--------------+--------------------+
7 rows in set (0.00 sec)
```

如果数据检索是最重要的，则可以在INSERT和INTO之间添加关键字LOW_PRIORITY,指示MYSQL降低INSERT语句的优先级。也适用于UPDATE和DELETE操作。

- 插入多行

使用逗号分隔多个values

```
INSERT INTO customers(cust_name,
cust_address,
cust_state,
cust_email,
cust_contact,
cust_country,
cust_city,
cust_zip)
VALUES('a','s','c','v','cc','ww',NULL,NULL),
      ('as','ss','cs','vs','ccs','wws',NULL,NULL);
```

- 插入某些查询的结果

创建一个新表custnew和customers有相同 的列。

```
CREATE TABLE `custnew` (
  `cust_id` int(11) NOT NULL AUTO_INCREMENT,
  `cust_name` char(50) NOT NULL,
  `cust_address` char(50) DEFAULT NULL,
  `cust_city` char(50) DEFAULT NULL,
  `cust_state` char(5) DEFAULT NULL,
  `cust_zip` char(10) DEFAULT NULL,
  `cust_country` char(50) DEFAULT NULL,
  `cust_contact` char(50) DEFAULT NULL,
  `cust_email` char(255) DEFAULT NULL,
  PRIMARY KEY (`cust_id`)
) ENGINE=InnoDB AUTO_INCREMENT=10008 DEFAULT CHARSET=latin1;
```

插入一些数据

```
INSERT INTO custnew(cust_name,
cust_address,
cust_state,
cust_email,
cust_contact,
cust_country,
cust_city,
cust_zip)
VALUES('sssa','dfds','fdc','fdv','gcc','wrew',NULL,NULL),
      ('afss','ssfs','cfss','vfss','cfscs','wwfds',NULL,NULL);
```

```
mysql> INSERT INTO custnew(cust_name,
    -> cust_address,
    -> cust_state,
    -> cust_email,
    -> cust_contact,
    -> cust_country,
    -> cust_city,
    -> cust_zip)
    -> VALUES('sssa','dfds','fdc','fdv','gcc','wrew',NULL,NULL),
    ->   ('afss','ssfs','cfss','vfss','cfscs','wwfds',NULL,NULL);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from custnew;
+---------+-----------+--------------+-----------+------------+----------+--------------+--------------+------------+
| cust_id | cust_name | cust_address | cust_city | cust_state | cust_zip | cust_country | cust_contact | cust_email |
+---------+-----------+--------------+-----------+------------+----------+--------------+--------------+------------+
|   10008 | sssa      | dfds         | NULL      | fdc        | NULL     | wrew         | gcc          | fdv        |
|   10009 | afss      | ssfs         | NULL      | cfss       | NULL     | wwfds        | cfscs        | vfss       |
+---------+-----------+--------------+-----------+------------+----------+--------------+--------------+------------+
2 rows in set (0.00 sec)
```

好了，现在把custnew用select检索出的数据插入都customers中。注意的是，虽然不必都写出所有的列，但前后的列的顺序要对应。SELECT子句可以使用WHERE。

```
INSERT INTO customers(cust_name,
cust_address,
cust_state,
cust_email,
cust_contact,
cust_country,
cust_city,
cust_zip)
SELECT cust_name,
cust_address,
cust_state,
cust_email,
cust_contact,
cust_country,
cust_city,
cust_zip
FROM custnew;
```

```
mysql> INSERT INTO customers(cust_name,
    -> cust_address,
    -> cust_state,
    -> cust_email,
    -> cust_contact,
    -> cust_country,
    -> cust_city,
    -> cust_zip)
    -> SELECT cust_name,
    -> cust_address,
    -> cust_state,
    -> cust_email,
    -> cust_contact,
    -> cust_country,
    -> cust_city,
    -> cust_zip
    -> FROM custnew;
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from customers;
+---------+----------------+---------------------+-----------+------------+----------+--------------+--------------+---------------------+
| cust_id | cust_name      | cust_address        | cust_city | cust_state | cust_zip | cust_country | cust_contact | cust_email          |
+---------+----------------+---------------------+-----------+------------+----------+--------------+--------------+---------------------+
|   10001 | Coyote Inc.    | 200 Maple Lane      | Detroit   | MI         | 44444    | USA          | Y Lee        | ylee@coyote.com     |
|   10002 | Mouse House    | 333 Fromage Lane    | Columbus  | OH         | 43333    | USA          | Jerry Mouse  | NULL                |
|   10003 | Wascals        | 1 Sunny Place       | Muncie    | IN         | 42222    | USA          | Jim Jones    | rabbit@wascally.com |
|   10004 | Yosemite Place | 829 Riverside Drive | Phoenix   | AZ         | 88888    | USA          | Y Sam        | sam@yosemite.com    |
|   10005 | E Fudd         | 4545 53rd Street    | Chicago   | IL         | 54545    | USA          | E Fudd       | NULL                |
|   10006 | D              | A                   | S         | SD         | 9008     | ASD          | NULL         | NULL                |
|   10007 | a              | s                   | NULL      | c          | NULL     | ww           | cc           | v                   |
|   10008 | sssa           | dfds                | NULL      | fdc        | NULL     | wrew         | gcc          | fdv                 |
|   10009 | afss           | ssfs                | NULL      | cfss       | NULL     | wwfds        | cfscs        | vfss                |
+---------+----------------+---------------------+-----------+------------+----------+--------------+--------------+---------------------+
9 rows in set (0.00 sec)
```

18.更新和删除数据

UPDATE

UPDATE,可以使用SELECT子查询。UPDATE在未执行完出错时的处理办法时恢复原来的数据，但可以指定不恢复的关键字IGNORE，在UPDATE和表明之间。

- 跟新特定的行

不要省略WHERE。

```
UPDATE customers
SET cust_email='elmer@fudd.com'
WHERE cust_id=10005;
```

置空某一行的某一列

```
UPDATE custnew
SET cust_email=NULL
WHERE cust_id=10008;
```

```
mysql> UPDATE custnew
    -> SET cust_email=NULL
    -> WHERE cust_id=10008;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from custnew;
+---------+-----------+-------------+-----------+------------+----------+-------------+-------------+------------+
| cust_id | cust_name | cust_address | cust_city | cust_state | cust_zip | cust_country | cust_contact | cust_email |
+---------+-----------+-------------+-----------+------------+----------+-------------+-------------+------------+
|   10008 | sssa      | dfds         | NULL      | fdc        | NULL     | wrew         | gcc          | NULL       |
|   10009 | afss      | ssfs         | NULL      | cfss       | NULL     | wwfds        | cfscs        | vfss       |
+---------+-----------+-------------+-----------+------------+----------+-------------+-------------+------------+
2 rows in set (0.00 sec)
```

- 跟新多个列,只需在SET中写多个值，用逗号隔开

```
UPDATE customers
SET cust_name='lijie',
    cust_email='elmer@fudd.com'
WHERE cust_id=10005;
```

- 跟新所有的行

不写WHERE.

```
mysql> UPDATE custnew SET cust_email=NULL;
Query OK, 1 row affected (0.00 sec)
Rows matched: 2  Changed: 1  Warnings: 0

mysql> select * from custnew;
+---------+-----------+-------------+-----------+------------+----------+-------------+-------------+------------+
| cust_id | cust_name | cust_address | cust_city | cust_state | cust_zip | cust_country | cust_contact | cust_email |
+---------+-----------+-------------+-----------+------------+----------+-------------+-------------+------------+
|   10008 | sssa      | dfds         | NULL      | fdc        | NULL     | wrew         | gcc          | NULL       |
|   10009 | afss      | ssfs         | NULL      | cfss       | NULL     | wwfds        | cfscs        | NULL       |
+---------+-----------+-------------+-----------+------------+----------+-------------+-------------+------------+
2 rows in set (0.00 sec)
```

DELETE

- 删除特定行

写WHERE.

```
DELETE FROM custnew
WHERE cust_id = 10009;
```

为了删除指定的列，使用UPDATE

```
?
```

- 删除所有行

不要使用DELETE.使用TRUNCATE，删除表，并重新创建了一个。快的。

```
TRUNCATE TABLE custnew;
```

19.创建和操纵表

创建表

DEFAULT指定默认值，不允许使用函数作为默认值，只能是常数。

```
CREATE TABLE `customers` IF NOT EXISTS (
  `cust_id` int(11) NOT NULL AUTO_INCREMENT,
  `cust_name` char(50) NOT NULL,
  `cust_address` char(50) DEFAULT NULL,
  `cust_city` char(50) DEFAULT NULL,
  `cust_state` char(5) DEFAULT NULL,
  `cust_zip` char(10) DEFAULT NULL,
  `cust_country` char(50) DEFAULT NULL,
  `cust_contact` char(50) DEFAULT NULL,
  `cust_email` char(255) DEFAULT NULL,
  PRIMARY KEY (`cust_id`)
) ENGINE=InnoDB AUTO_INCREMENT=10011 DEFAULT CHARSET=latin1;
```

获得AUTO_INCREAMENT的最后一个值

```
SELECT last_insert_id(cust_id) FROM customers;
```

```
mysql> SELECT last_insert_id(cust_id) FROM customers;
+------------------------+
| last_insert_id(cust_id) |
+------------------------+
|                  10001 |
|                  10002 |
|                  10003 |
|                  10004 |
|                  10005 |
|                  10007 |
|                  10008 |
|                  10009 |
+------------------------+
8 rows in set (0.00 sec)

mysql> select * from customers;
+---------+---------------+--------------------+-----------+------------+----------+--------------+--------------+--------------------+
| cust_id | cust_name     | cust_address       | cust_city | cust_state | cust_zip | cust_country | cust_contact | cust_email         |
+---------+---------------+--------------------+-----------+------------+----------+--------------+--------------+--------------------+
|   10001 | Coyote Inc.   | 200 Maple Lane     | Detroit   | MI         | 44444    | USA          | Y Lee        | ylee@coyote.com    |
|   10002 | Mouse House   | 333 Fromage Lane   | Columbus  | OH         | 43333    | USA          | Jerry Mouse  | NULL               |
|   10003 | Wascals       | 1 Sunny Place      | Muncie    | IN         | 42222    | USA          | Jim Jones    | rabbit@wascally.com |
|   10004 | Yosemite Place | 829 Riverside Drive | Phoenix   | AZ         | 88888    | USA          | Y Sam        | sam@yosemite.com   |
|   10005 | E Fudd        | 4545 53rd Street   | Chicago   | IL         | 54545    | USA          | E Fudd       | elmer@fudd.com     |
|   10007 | a             | s                  | NULL      | c          | NULL     | ww           | cc           | v                  |
|   10008 | sssa          | dfds               | NULL      | fdc        | NULL     | wrew         | gcc          | fdv                |
|   10009 | afss          | ssfs               | NULL      | cfss       | NULL     | wwfds        | cfscs        | vfss               |
+---------+---------------+--------------------+-----------+------------+----------+--------------+--------------+--------------------+
8 rows in set (0.00 sec)
```

引擎类型：

- InnoDB是一个可靠的事务处理引擎，它不支持全文本搜索
- MEMORY在功能上等同于MyISAM,但由于数据存储在内存中，速度很快。
- MyISAM是一个性能很高的引擎，它支持全文本搜索，但不支持事务处理。

外键不能跨引擎

更新表

给Vendors表增加一个新的列vend_phone。

```
ALTER TABLE vendors
ADD vend_phone CHAR(20);
```

删除该列

```
ALTER TABLE vendors
DROP COLUMN vend_phone;
```

常见的用途是定义外键

```
ALTER TABLE orderitems
ADD CONSTRAINT fk_orderitems_orders
FOREIGN KEY (order_num) REFERENCES orders (order_num);
```

```
ALTER TABLE orderitems
ADD CONSTRAINT fk_orderitems_products
FOREIGN KEY (prod_id) REFERENCES products (prod_id);
```

```
ALTER TABLE orders
ADD CONSTRAINT fk_orders_customers
FOREIGN KEY (cust_id) REFERENCES customers (cust_id);
```

```
ALTER TABLE products
ADD CONSTRAINT fk_products_vendors
FOREIGN KEY (vend_id) REFERENCES vendors (vend_id);
```

删除表

```
DROP TABLE customers2;
```

重命名表

重命名多个表，使用逗号隔开

```
RENAME TABLE customers2 TO customers,
             custnew    TO cust;
```

20.使用视图

视图是虚拟的表。他们包含的不是数据而是根据需要检索数据的查询。视图提供了一种MYsql的SELECT语句层次的封装，可用来简化数据处理以及重新格式化基础数据或保护基础数据。

可以把较为复杂的查询剃除出来组成视图，再次查询时，只需写少量的语句就可以了。

复杂1

```
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';
```

使用视图1

别忘了加prod_id

```
CREATE VIEW productcustomers AS
SELECT cust_name,cust_contact,prod_id
FROM customers,orders,orderitems
WHERE customers.cust_id = orders.cust_id
AND orderitems.order_num = orders.order_num
```

```
SELECT cust_name,cust_contact
FROM productcustomers
WHERE prod_id = 'TNT2';
```

```
mysql> CREATE VIEW productcustomers AS
    -> SELECT cust_name,cust_contact,prod_id
    -> FROM customers,orders,orderitems
    -> WHERE customers.cust_id = orders.cust_id
    -> AND orderitems.order_num = orders.order_num
    -> ;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT cust_name,cust_contact
    -> FROM productcustomers
    -> WHERE prod_id = 'TNT2';
+---------------+--------------+
| cust_name     | cust_contact |
+---------------+--------------+
| Coyote Inc.   | Y Lee        |
| Yosemite Place | Y Sam       |
+---------------+--------------+
2 rows in set (0.01 sec)
```

复杂2

```
SELECT CONCAT(RTrim(vend_name), ' (', RTrim(vend_country),')')
AS vend_title
FROM vendors
ORDER BY vend_name;
```

使用视图2

```
CREATE VIEW vendorlocations AS
SELECT CONCAT(RTrim(vend_name), ' (', RTrim(vend_country),')')
AS vend_title
FROM vendors
ORDER BY vend_name;
```

```
SELECT * FROM vendorlocations;
```

```
mysql> CREATE VIEW vendorlocations AS
    -> SELECT CONCAT(RTrim(vend_name), ' (', RTrim(vend_country),')')
    -> AS vend_title
    -> FROM vendors
    -> ORDER BY vend_name;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM vendorlocations;
+-------------------------+
| vend_title              |
+-------------------------+
| ACME (USA)              |
| Anvils R Us (USA)       |
| Furball Inc. (USA)      |
| Jet Set (England)       |
| Jouets Et Ours (France) |
| LT Supplies (USA)       |
+-------------------------+
6 rows in set (0.00 sec)
```

过滤掉不想要的数据

过滤掉没有邮箱的记录

```
CREATE VIEW customeremaillist AS
SELECT cust_id,cust_name,cust_email
FROM customers
WHERE cust_email IS NOT NULL;
```

```
SELECT * FROM customeremaillist;
```

```
mysql> CREATE VIEW customeremaillist AS
    -> SELECT cust_id,cust_name,cust_email
    -> FROM customers
    -> WHERE cust_email IS NOT NULL;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM customeremaillist;
+---------+----------------+--------------------+
| cust_id | cust_name      | cust_email         |
+---------+----------------+--------------------+
|   10001 | Coyote Inc.    | ylee@coyote.com    |
|   10003 | Wascals        | rabbit@wascally.com |
|   10004 | Yosemite Place | sam@yosemite.com   |
|   10005 | E Fudd         | elmer@fudd.com     |
|   10007 | a              | v                  |
|   10008 | sssa           | fdv                |
|   10009 | afss           | vfss               |
+---------+----------------+--------------------+
7 rows in set (0.00 sec)
```

使用视图与计算字段

```
SELECT prod_id,quantity,item_price,quantity*item_price AS expanded_price
FROM orderitems
WHERE order_num = 20005;
```

```
mysql> SELECT prod_id,quantity,item_price,quantity*item_price AS expanded_price
    -> FROM orderitems
    -> WHERE order_num = 20005;
+---------+----------+------------+----------------+
| prod_id | quantity | item_price | expanded_price |
+---------+----------+------------+----------------+
| ANV01   |       10 |       5.99 |          59.90 |
| ANV02   |        3 |       9.99 |          29.97 |
| TNT2    |        5 |      10.00 |          50.00 |
| FB      |        1 |      10.00 |          10.00 |
+---------+----------+------------+----------------+
4 rows in set (0.00 sec)
```

使用视图

别忘了加order_num

```
CREATE VIEW orderitemsexpanded AS
SELECT order_num,prod_id,quantity,item_price,quantity*item_price AS expanded_price
FROM orderitems;
```

```
SELECT * FROM orderitemsexpanded
WHERE order_num = 20005;
```

```
mysql> CREATE VIEW orderitemsexpanded AS
    -> SELECT order_num,prod_id,quantity,item_price,quantity*item_price AS expanded_price
    -> FROM orderitems;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM orderitemsexpanded
    -> WHERE order_num = 20005;
+-----------+---------+----------+------------+----------------+
| order_num | prod_id | quantity | item_price | expanded_price |
+-----------+---------+----------+------------+----------------+
|     20005 | ANV01   |       10 |       5.99 |          59.90 |
|     20005 | ANV02   |        3 |       9.99 |          29.97 |
|     20005 | TNT2    |        5 |      10.00 |          50.00 |
|     20005 | FB      |        1 |      10.00 |          10.00 |
+-----------+---------+----------+------------+----------------+
4 rows in set (0.00 sec)
```

删除视图

```
DROP VIEW orderitemsexpanded;
```