第11章_数据处理之增删改

1. 插入数据

1.1 实际问题

解决方式：使用 INSERT 语句向表中插入数据

1.2 方式1：VALUES的方式添加

使用这种语法一次只能向表中插入**一条**数据

情况1：为表的所有字段按默认顺序插入数据

```
INSERT INTO 表名
VALUES (value1,value2,....);
```

值列表中需要为表的每一个字段指定值，并且值的顺序必须和数据表中字段定义时的顺序相同。

```
-- 切换数据库
USE atguigudb;
```

```
Database changed



-- 创建表
CREATE TABLE IF NOT EXISTS emp1(
                id INT,
                `name` VARCHAR(12),
                hire_date DATE,
                salary DOUBLE(10,2)  -- 共十位数字，其中小数占两位
);

Query OK, 0 rows affected, 1 warning (0.01 sec)



-- 查看表结构
DESC emp1;

+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| id        | int          | YES  |     | NULL    |       |
| name      | varchar(12)  | YES  |     | NULL    |       |
| hire_date | date         | YES  |     | NULL    |       |
| salary    | double(10,2) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

方式一：一条一条地添加数据

```
INSERT INTO emp1 VALUES(1,'Tom','2022-06-25',3400); -- 注意顺序（因为没有指明添加字段），按照生命的先后顺序添加

Query OK, 1 row affected (0.00 sec)



-- 验证插入
SELECT * FROM emp1;

+------+------+------------+---------+
| id   | name | hire_date  | salary  |
+------+------+------------+---------+
|    1 | Tom  | 2022-06-25 | 3400.00 |
+------+------+------------+---------+
1 row in set (0.00 sec)
```

情况2：为表的指定字段插入数据

```
INSERT INTO 表名(column1 [, column2, …, columnn])
VALUES (value1 [,value2, …, valuen]);
```

为表的指定字段插入数据，就是在INSERT语句中只向部分字段中插入值，而其他字段的值为表定义时的默认值。

在 INSERT 子句中随意列出列名，但是一旦列出，VALUES中要插入的value1,....valuen需要与column1,...columnn列一一对应

如果类型不同，将无法插入，并且MySQL会产生错误。

推荐的添加方法

```
INSERT INTO emp1 (id,name,salary)
VALUES(2,'Jetty',2300);
Query OK, 1 row affected (0.00 sec)



-- 验证插入
SELECT * FROM emp1;
+------+-------+------------+---------+
| id   | name  | hire_date  | salary  |
+------+-------+------------+---------+
|    1 | Tom   | 2022-06-25 | 3400.00 |
|    2 | Jetty | NULL       | 2300.00 |
+------+-------+------------+---------+
2 rows in set (0.00 sec)
```

情况3：同时插入多条记录

INSERT语句可以同时向数据表中插入多条记录，插入时指定多个值列表，每个值列表之间用逗号分隔开，基本语法格式如下：

```
INSERT INTO table_name
VALUES
(value1 [,value2, …, valuen]),
(value1 [,value2, …, valuen]),
……
(value1 [,value2, …, valuen]);
```

或者

```
INSERT INTO table_name(column1 [, column2, …, columnn])
VALUES
(value1 [,value2, …, valuen]),
(value1 [,value2, …, valuen]),
……
(value1 [,value2, …, valuen]);
```

推荐

```
INSERT INTO emp1(id,name,salary)
VALUES
      (4,'Ben',1000),
      (5,'JJ',10009);

Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0


-- 验证插入
SELECT * FROM emp1;

+------+-------+------------+----------+
| id   | name  | hire_date  | salary   |
+------+-------+------------+----------+
|    1 | Tom   | 2022-06-25 |  3400.00 |
|    2 | Jetty | NULL       |  2300.00 |
|    4 | Ben   | NULL       |  1000.00 |
|    5 | JJ    | NULL       | 10009.00 |
+------+-------+------------+----------+
4 rows in set (0.00 sec)
```

1.3 方式2：将查询结果插入到表中

INSERT还可以将SELECT语句查询的结果插入到表中，此时不需要把每一条记录的值一个一个输入，只需要使用一条INSERT语句和一条SELECT语句组成的组合语句即可快速地从一个或多个表中向一个表中插入多行。

基本语法格式如下：

```
INSERT INTO 目标表名
(tar_column1 [, tar_column2, …, tar_columnn])
SELECT
(src_column1 [, src_column2, …, src_columnn])
FROM 源表名
[WHERE condition]
```

- 在 INSERT 语句中加入子查询，即INSERT 语句后不接分号
- **不必书写 VALUES 子句**
- 子查询中的值列表应与 INSERT 子句中的列名**对应**

举例：

```
INSERT INTO emp1(id,name,salary,hire_date)

SELECT employee_id,last_name,salary,hire_date
FROM employees
WHERE department_id IN (70,60);

Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0



-- 验证
SELECT * FROM emp1;

+------+-----------+------------+----------+
| id   | name      | hire_date  | salary   |
+------+-----------+------------+----------+
|    1 | Tom       | 2022-06-25 |  3400.00 |
|    2 | Jetty     | NULL       |  2300.00 |
|    4 | Ben       | NULL       |  1000.00 |
|    5 | JJ        | NULL       | 10009.00 |
|  103 | Hunold    | 1990-01-03 |  9000.00 |
|  104 | Ernst     | 1991-05-21 |  6000.00 |
|  105 | Austin    | 1997-06-25 |  4800.00 |
|  106 | Pataballa | 1998-02-05 |  4800.00 |
|  107 | Lorentz   | 1999-02-07 |  4200.00 |
|  204 | Baer      | 1994-06-07 | 10000.00 |
+------+-----------+------------+----------+
10 rows in set (0.00 sec)
```

注意：

两种表中，同一个字段的数据类型；最好保持一致；

使用INSERT同时插入多条记录时，MySQL会返回一些在执行单行插入时没有的额外信息，这些信息的含义如下：

- Records：表明插入的记录条数。
- Duplicates：表明插入时被忽略的记录，原因可能是这些记录**包含了重复的主键值**
- Warnings：表明有问题的数据值，例如发生数据类型转换。

小结：

- VALUES也可以写成VALUE，但是VALUES是标准写法。

- 字符和日期型数据应包含在单引号中
- 一个同时插入多行记录的INSERT语句等同于多个单行插入的INSERT语句，但是多行的INSERT语句在处理过程中效率更高。
- 因为MySQL执行单条INSERT语句插入多行数据比使用多条INSERT语句快，所以在插入多条记录时最好选择使用单条INSERT语句的方式插入

2. 更新数据

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_F
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

更新 **EMPLOYEES** 表

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSIO
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	30	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	30	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	30	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

- 使用 UPDATE 语句更新数据。语法如下：

```
UPDATE TABLE

SET field1=xxx,

field2=xxx;

WHERE ……
```

- 可以一次更新**多条**数据
- 如果需要回滚数据，需要保证在DML前，进行设置：**SET AUTOCOMMIT = FALSE;**

- 使用 **WHERE** 子句指定需要更新的数据。
- 如果省略 WHERE 子句，则表中的所有数据都将被更新。

修改一个字段

```
UPDATE emp1
SET hire_date=CURDATE()
WHERE id=5;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0



SELECT * FROM emp1
WHERE id=5;

+------+------+------------+----------+
| id   | name | hire_date  | salary   |
+------+------+------------+----------+
|    5 | JJ   | 2022-06-25 | 10009.00 |
+------+------+------------+----------+
1 row in set (0.00 sec)
```

修改多个字段

```
UPDATE emp1
SET hire_date=CURDATE(),salary=1230
WHERE id=4;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0



SELECT * FROM emp1
WHERE id=4;

+------+------+------------+---------+
| id   | name | hire_date  | salary  |
+------+------+------------+---------+
|    4 | Ben  | 2022-06-25 | 1230.00 |
+------+------+------------+---------+
1 row in set (0.00 sec)
```

由于约束的问题导致更新失败

- **更新中的数据完整性错误**

```
UPDATE employees
SET    department_id = 55
WHERE  department_id = 110;
```

```
错误代码： 1452
Cannot add or update a child row: a foreign key
constraint fails (`myemployees`.`employees`,
CONSTRAINT `dept_id_fk` FOREIGN KEY (`department_id`)
REFERENCES `departments` (`department_id`))
```

> 说明：不存在 55 号部门

3. 删除数据

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
100	Finance		
50	Shipping	124	1500
60	IT	103	1400

从表DEPARTMENTS 中删除一条记录。

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
50	Shipping	124	1500
60	IT	103	1400

- 使用 DELETE 语句从表中删除数据

> DELETE FROM table_name [WHERE <condition>];

table_name指定要执行删除操作的表；

[WHERE <condition>] 为可选参数，指定删除条件，如果没有WHERE子句，DELETE语句将删除表中的所有记录

注意：

DML操作默认情况下，执行完以后都会自动提交数据；

如果希望执行完成以后不自动提交数据，则需要使用命令 SET autocommit =FALSE;

使用 WHERE 子句删除指定的记录。

```
DELETE FROM departments
WHERE  department_name = 'Finance';
```

如果省略 WHERE 子句，则表中的全部数据将被删除

```
DELETE FROM  copy_emp;
```

删除中的数据完整性错误

```
DELETE FROM departments
WHERE       department_id = 60;
```

```
错误代码： 1451
Cannot delete or update a parent row: a foreign key
constraint fails (`myemployees`.`employees`,
CONSTRAINT `dept_id_fk` FOREIGN KEY (`department_id`)
REFERENCES `departments` (`department_id`))
```

> 说明：You cannot delete a row that contains a primary key that is used as a foreign key in another table

4. MySQL8新特性：计算列

什么叫计算列呢?

简单来说就是某一列的值是通过别的列计算得来的

例如，a列值为1、b列值为2，c列不需要手动插入，定义a+b的结果为c的值，那么c就是计算列，是通过别的列计算得来的。

在MySQL 8.0中，**CREATE TABLE 和 ALTER TABLE 中都支持增加计算列。**下面以CREATE TABLE为例进行讲解。

举例：定义数据表tb1，然后定义字段id、字段a、字段b和字段c，其中字段c为计算列，用于计算a+b的值。

首先创建测试表tb1，语句如下：

```
USE atguigudb;

CREATE TABLE tb1(
id INT,
a INT,
b INT,
c INT GENERATED ALWAYS AS (a + b) VIRTUAL -- 声明计算列
);

Query OK, 0 rows affected (0.02 sec)
```

插入演示数据，语句如下：

```
INSERT INTO tb1(id,a,b) VALUES (1,100,200);
Query OK, 1 row affected (0.00 sec)
```

查询数据表tb1中的数据，结果如下：

```
SELECT * FROM tb1;

+------+------+------+------+
| id   | a    | b    | c    |
+------+------+------+------+
|    1 |  100 |  200 |  300 |
+------+------+------+------+
1 row in set (0.00 sec)
```

更新数据中的数据，语句如下：

```
UPDATE tb1 SET a = 500;

Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0



-- 验证数据
SELECT * FROM tb1;

+------+------+------+------+
| id   | a    | b    | c    |
```

```
+------+------+------+------+
|    1 |  500 |  200 |  700 |
+------+------+------+------+
1 row in set (0.00 sec)
```

5. 综合案例

```
# 1、创建数据库test01_library
CREATE DATABASE IF NOT EXISTS test01_library CHARACTER SET ='utf8';

Query OK, 1 row affected, 1 warning (0.00 sec)



# 2、创建表 books，表结构如下：
USE test01_library;

CREATE TABLE books(
        id INT,
        name VARCHAR(50),
        authors VARCHAR(100),
        price FLOAT,
        pubdate YEAR,
        note VARCHAR(100),
        num INT
);

Query OK, 0 rows affected (0.02 sec)



-- 查看表结构
DESC books;

+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| id      | int          | YES  |     | NULL    |       |
| name    | varchar(50)  | YES  |     | NULL    |       |
| authors | varchar(100) | YES  |     | NULL    |       |
| price   | float        | YES  |     | NULL    |       |
| pubdate | year         | YES  |     | NULL    |       |
| note    | varchar(100) | YES  |     | NULL    |       |
| num     | int          | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

字段名	字段说明	数据类型
id	书编号	INT
name	书名	VARCHAR(50)
authors	作者	VARCHAR(100)
price	价格	FLOAT
pubdate	出版日期	YEAR
note	说明	VARCHAR(100)
num	库存	INT

```
# 3、向books表中插入记录


# 1）不指定字段名称，插入第一条记录
INSERT INTO books
VALUES(1,'Tal of AAA','Dickes',23,1995,'novel',11);

Query OK, 1 row affected (0.00 sec)



# 2）指定所有字段名称，插入第二记录
INSERT INTO books (id,name,authors,price,pubdate,note,num)
VALUES(2,'EmmaT','Jane lura',35,1993,'joke',22);

Query OK, 1 row affected (0.00 sec)
# 3）同时插入多条记录（剩下的所有记录）

INSERT INTO books
VALUES
(3,'Story of Jane','Jane Tim',40,2001,'novel',0),
(4,'Lovey Day','George Byron',20,2005,'novel',30),
(5,'Old land','Honore Blade',30,2010,'law',0),
(6,'The Battle','Upton Sara',30,1999,'medicine',40);

Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0



-- 数据验证

SELECT * FROM books;

+------+---------------+--------------+-------+---------+----------+------+
| id   | name          | authors      | price | pubdate | note     | num  |
+------+---------------+--------------+-------+---------+----------+------+
|    1 | Tal of AAA    | Dickes       |    23 |    1995 | novel    |   11 |
|    2 | EmmaT         | Jane lura    |    35 |    1993 | joke     |   22 |
```

```
|    3 | Story of Jane | Jane Tim     |    40 |    2001 | novel    |    0 |
|    4 | Lovey Day     | George Byron |    20 |    2005 | novel    |   30 |
|    5 | Old land      | Honore Blade |    30 |    2010 | law      |    0 |
|    6 | The Battle    | Upton Sara   |    30 |    1999 | medicine |   40 |
+------+---------------+--------------+-------+---------+----------+------+
6 rows in set (0.00 sec)
```

id	name	authors	price	pubdate	note	num
1	Tal of AAA	Dickes	23	1995	novel	11
2	EmmaT	Jane lura	35	1993	joke	22
3	Story of Jane	Jane Tim	40	2001	novel	0
4	Lovey Day	George Byron	20	2005	novel	30
5	Old land	Honore Blade	30	2010	law	0
6	The Battle	Upton Sara	30	1999	medicine	40
7	Rose Hood	Richard haggard	28	2008	cartoon	28

```
# 4、将小说类型(novel)的书的价格都增加5
UPDATE books SET price=price+5 WHERE note = 'novel';

Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0


# 5、将名称为EmmaT的书的价格改为40，并将说明改为drama。
UPDATE books SET price=40,note='drama' WHERE name='EmmaT';

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

# 6、删除库存为0的记录。
DELETE FROM books WHERE num=0;

Query OK, 2 rows affected (0.00 sec)
```

```
# 7、统计书名中包含a字母的书
SELECT * FROM books WHERE name LIKE '%a%';

+------+------------+--------------+-------+---------+----------+------+
| id   | name       | authors      | price | pubdate | note     | num  |
+------+------------+--------------+-------+---------+----------+------+
|    1 | Tal of AAA | Dickes       |    28 |    1995 | novel    |   11 |
|    2 | EmmaT      | Jane lura    |    40 |    1993 | drama    |   22 |
|    4 | Lovey Day  | George Byron |    25 |    2005 | novel    |   30 |
|    6 | The Battle | Upton Sara   |    30 |    1999 | medicine |   40 |
+------+------------+--------------+-------+---------+----------+------+
4 rows in set (0.00 sec)
```

```
# 8、统计书名中包含a字母的书的数量和库存总量
SELECT COUNT(*),SUM(num) FROM books WHERE name LIKE '%a%';

+----------+----------+
| COUNT(*) | SUM(num) |
+----------+----------+
|        4 |      103 |
+----------+----------+
1 row in set (0.00 sec)


# 9、找出“novel”类型的书，按照价格降序排列
SELECT * FROM books
WHERE note = 'novel'
ORDER BY price DESC;

+------+------------+--------------+-------+---------+-------+------+
| id   | name       | authors      | price | pubdate | note  | num  |
+------+------------+--------------+-------+---------+-------+------+
|    1 | Tal of AAA | Dickes       |    28 |    1995 | novel |   11 |
|    4 | Lovey Day  | George Byron |    25 |    2005 | novel |   30 |
+------+------------+--------------+-------+---------+-------+------+
2 rows in set (0.00 sec)



# 10、查询图书信息，按照库存量降序排列，如果库存量相同的按照note升序排列
SELECT *
FROM books
ORDER BY num DESC,note ASC;


+------+------------+--------------+-------+---------+----------+------+
| id   | name       | authors      | price | pubdate | note     | num  |
+------+------------+--------------+-------+---------+----------+------+
|    6 | The Battle | Upton Sara   |    30 |    1999 | medicine |   40 |
|    4 | Lovey Day  | George Byron |    25 |    2005 | novel    |   30 |
|    2 | EmmaT      | Jane lura    |    40 |    1993 | drama    |   22 |
|    1 | Tal of AAA | Dickes       |    28 |    1995 | novel    |   11 |
+------+------------+--------------+-------+---------+----------+------+
4 rows in set (0.00 sec)


# 11、按照note分类统计书的数量
SELECT note,COUNT(*)
FROM books
GROUP BY note;

+----------+----------+
| note     | COUNT(*) |
+----------+----------+
| novel    |        2 |
| drama    |        1 |
```

```
| medicine |        1 |
+----------+----------+
3 rows in set (0.00 sec)

# 12、按照note分类统计书的库存量，显示库存量超过30本的
SELECT note,SUM(num)
FROM books
GROUP BY note
HAVING SUM(num)>30;

+----------+----------+
| note     | SUM(num) |
+----------+----------+
| novel    |       41 |
| medicine |       40 |
+----------+----------+
2 rows in set (0.00 sec)


# 13、查询所有图书，每页显示5本，显示第二页
SELECT *
FROM books
LIMIT 5,5;

Empty set (0.00 sec)


# 14、按照note分类统计书的库存量，显示库存量最多的
SELECT note,SUM(num) sum_num
FROM books
GROUP BY note
ORDER BY sum_num DESC
LIMIT 0,1;

+-------+---------+
| note  | sum_num |
+-------+---------+
| novel |      41 |
+-------+---------+
1 row in set (0.00 sec)


# 15、查询书名达到10个字符的书，不包括里面的空格
SELECT *
FROM books
WHERE CHAR_LENGTH(REPLACE(name,' ',''))>=10;

Empty set (0.00 sec)


# 16、查询书名和类型，其中note值为novel显示小说，law显示法律，medicine显示医药，cartoon
显示卡通，joke显示笑话
SELECT name AS "书名" ,note, CASE note
 WHEN 'novel' THEN '小说'
```

```
 WHEN 'law' THEN '法律'
 WHEN 'medicine' THEN '医药'
 WHEN 'cartoon' THEN '卡通'
 WHEN 'joke' THEN '笑话'
 END  "类型"
FROM books;

+------------+----------+------+
| 书名       | note     | 类型 |
+------------+----------+------+
| Tal of AAA | novel    | 小说 |
| EmmaT      | drama    | NULL |
| Lovey Day  | novel    | 小说 |
| The Battle | medicine | 医药 |
+------------+----------+------+
4 rows in set (0.00 sec)


# 17、查询书名、库存，其中num值超过30本的，显示滞销，大于0并低于10的，显示畅销，为0的显示需要无货
SELECT name,num,CASE
  WHEN num>30 THEN '滞销'
  WHEN num>0 AND num<10 THEN '畅销'
  WHEN num=0 THEN '无货'
  ELSE '正常'
  END  "库存状态"
FROM books;

+------------+------+----------+
| name       | num  | 库存状态 |
+------------+------+----------+
| Tal of AAA |   11 | 正常     |
| EmmaT      |   22 | 正常     |
| Lovey Day  |   30 | 正常     |
| The Battle |   40 | 滞销     |
+------------+------+----------+
4 rows in set (0.00 sec)



# 18、统计每一种note的库存量，并合计总量
SELECT IFNULL(note,'合计总库存量')  note,SUM(num) FROM books
GROUP BY note
WITH ROLLUP;


+--------------+----------+
| note         | SUM(num) |
+--------------+----------+
| drama        |       22 |
| medicine     |       40 |
| novel        |       41 |
| 合计总库存量 |      103 |
+--------------+----------+
```

```
4 rows in set, 1 warning (0.00 sec)
```

```
# 19、统计每一种note的数量，并合计总量
SELECT IFNULL(note,'合计总数')  note,COUNT(*)
FROM books
GROUP BY note
WITH ROLLUP;
```

```
+----------+----------+
| note     | COUNT(*) |
+----------+----------+
| drama    |        1 |
| medicine |        1 |
| novel    |        2 |
| 合计总数 |          4 |
+----------+----------+
4 rows in set, 1 warning (0.00 sec)
```

```
# 20、统计库存量前三名的图书
SELECT *
FROM books
ORDER BY num DESC
LIMIT 0,3;
```

```
+------+------------+--------------+-------+---------+----------+------+
| id   | name       | authors      | price | pubdate | note     | num  |
+------+------------+--------------+-------+---------+----------+------+
|    6 | The Battle | Upton Sara   |    30 |    1999 | medicine |   40 |
|    4 | Lovey Day  | George Byron |    25 |    2005 | novel    |   30 |
|    2 | EmmaT      | Jane lura    |    40 |    1993 | drama    |   22 |
+------+------------+--------------+-------+---------+----------+------+
3 rows in set (0.00 sec)
```

```
# 21、找出最早出版的一本书
SELECT *
FROM books
ORDER BY pubdate ASC
LIMIT 0,1;
```

```
+------+-------+-----------+-------+---------+-------+------+
| id   | name  | authors   | price | pubdate | note  | num  |
+------+-------+-----------+-------+---------+-------+------+
|    2 | EmmaT | Jane lura |    40 |    1993 | drama |   22 |
+------+-------+-----------+-------+---------+-------+------+
1 row in set (0.00 sec)
```

```
# 22、找出novel中价格最高的一本书
SELECT *
FROM books
WHERE note = 'novel'
ORDER BY price DESC
LIMIT 0,1;

+------+------------+---------+-------+---------+-------+------+
| id   | name       | authors | price | pubdate | note  | num  |
+------+------------+---------+-------+---------+-------+------+
|    1 | Tal of AAA | Dickes  |    28 |    1995 | novel |   11 |
+------+------------+---------+-------+---------+-------+------+
1 row in set (0.00 sec)



# 23、找出书名中字数最多的一本书，不含空格
SELECT *
FROM books
ORDER BY CHAR_LENGTH(REPLACE(name,' ','')) DESC -- 字符长度
LIMIT 0,1;

+------+------------+------------+-------+---------+----------+------+
| id   | name       | authors    | price | pubdate | note     | num  |
+------+------------+------------+-------+---------+----------+------+
|    6 | The Battle | Upton Sara |    30 |    1999 | medicine |   40 |
+------+------------+------------+-------+---------+----------+------+
1 row in set (0.00 sec)
```