# 常用SQL语句

《SQL基础教程第2版》[日] MICK /著　孙淼、罗勇 /译

提取码：n9dg

温吉祥 2019-06-18

## 目录：



`product表`

| product_id | product_name | product_type | sale_price | purchase_price | regist_date |
|---|---|---|---|---|---|
| 0001 | T恤衫 | 衣服 | 2000 | 500 | 2009-09-20 |
| 0002 | 打孔器 | 办公用品 | 500 | 320 | 2009-09-11 |
| 0003 | 运动T恤 | 衣服 | 3000 | 2800 | NULL |
| 0004 | 菜刀 | 厨房用具 | 3000 | 2800 | 2009-09-20 |
| 0005 | 高压锅 | 厨房用具 | 6800 | 5000 | 2009-01-15 |
| 0006 | 叉子 | 厨房用具 | 500 | NULL | 2009-09-20 |
| 0007 | 擦菜板 | 厨房用具 | 880 | 790 | 2008-04-28 |
| 0008 | 圆珠笔 | 办公用品 | 100 | NULL | 2009-11-11 |

## 创建数据库和表(CREATE)

```
-- 创建数据库:
CREATE DATABASE <数据库名称>

-- 创建数据表:
CREATE TABLE <表名>
(<列名1> <数据类型> <该列所需约束>,
 <列名2> <数据类型> <该列所需约束>,
                  .
                  .
 <该表的约束1>,<该表的约束2>...);

CREATE TABLE Product
(product_id  CHAR(4) NOT NULL,
 product_name VARCHAR(100) NOT NULL,
 product_type VARCHAR(32) NOT NULL,
 sale_price INTEGER ,
 purchase_price INTEGER ,
 regist_date DATE ,
 PRIMARY KEY (product_id));
```

## 表的删除(DROP,DELETE,TRUNCATE)

```
-- DROP TABEL:将表完全删除
-- DELETE TABLE:保留表(容器),删除表中的数据,可以用where指定删除条件
-- TRUNCATE:与DELETE相比,只用删除表中的数据,但不用通过where指定删除条件,速度较快
DROP FROM <表名>;
DELETE FROM <表名>;
```

## 表定义的更新(ALTER)

### 1.添加列

```
-- Oracle 和SQL Server中不用写COLUMN
ALTER TABLE <表名> ADD COLUMN <列的定义>;

-- 如在Porduct表中添加商品名称列(product_name):
ALTER TABLE Product ADD COLUMN product_name VARCHAR(100);
```

### 2.删除列

```
ALTER TABLE <表名> DROP COLUMN <列名>;
-- 删除Product表中的商品名称列(product_name):
ALTER TABLE Product DROP COLUMN product_name;
```

### 3.更改表名

```sql
-- 把Product表改成Products

-- ORacle和PostgreSQL
ALTER TABLE Product RENAME TO Products;
-- SQL Server
sp_rename 'Product','Products';
-- MySQL
RENAME TEBLE Product to Products;
```

#### 4.向表中插入数据(INSERT)

```sql
INSERT INTO <表名> VALUES (列1,列2,...) VALUES (值1,值2,...);

INSERT INTO product (product,product_name,product_type,sale_price)
 VALUES ('0001','T恤','衣服','1000');
```

### 表的更新(UPDATE)

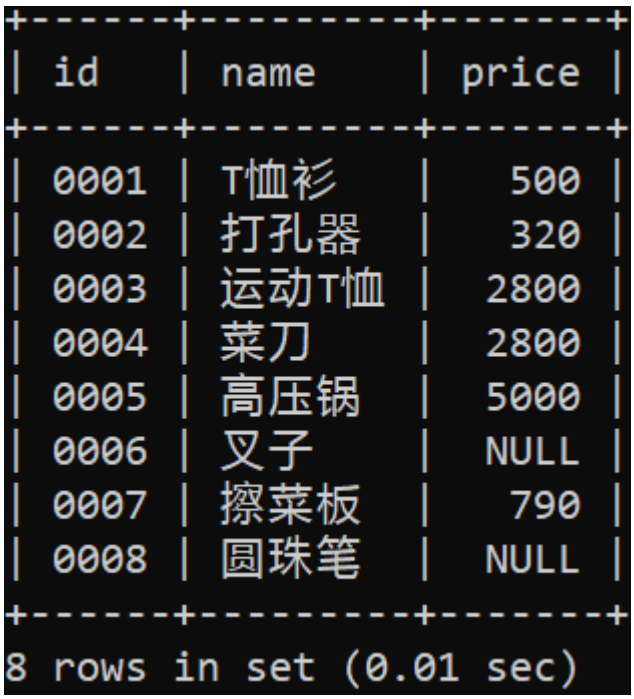
```sql
UPDATE <表名>
   SET <列名> = <表达式>;
-- 将登记日期更新为'2019-06-18'
UPDATE Product
   SET regist_date = '2019-06-18';
```

### 查询(SELECT)

```sql
-- 最基本的格式
SELECT <列名>, ……
   FROM <表名>;
```

#### 1.AS(设置别名)

```sql
SELECT
    product_id AS id,
    product_name AS name,
    purchase_price AS price
FROM
    Product;
```



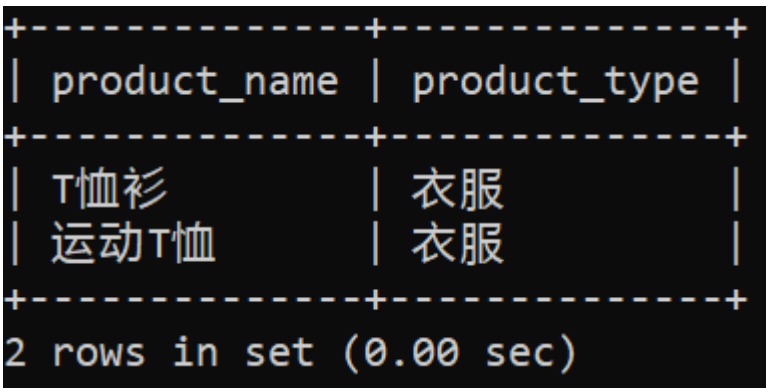
| id | name | price |
|---|---|---|
| 0001 | T恤衫 | 500 |
| 0002 | 打孔器 | 320 |
| 0003 | 运动T恤 | 2800 |
| 0004 | 菜刀 | 2800 |
| 0005 | 高压锅 | 5000 |
| 0006 | 叉子 | NULL |
| 0007 | 擦菜板 | 790 |
| 0008 | 圆珠笔 | NULL |

8 rows in set (0.01 sec)

**2.** `DISTINCT` **(删除重复行)**

**3.** `WHERE` **(选择记录)**

```
SELECT
    product_name, product_type
FROM
    Product
WHERE
    product_type = '衣服';
```

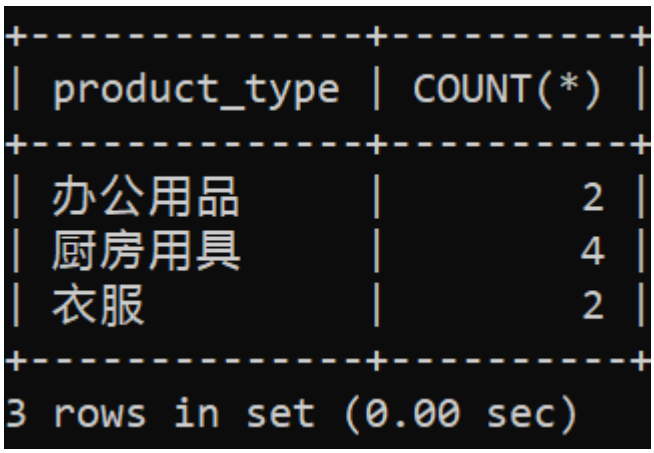


| product_name | product_type |
|---|---|
| T恤衫 | 衣服 |
| 运动T恤 | 衣服 |

2 rows in set (0.00 sec)

**4.** `GROUP BY` **(分组)**

```
SELECT
    product_type, COUNT(*)
FROM
    Product
GROUP BY product_type;
```

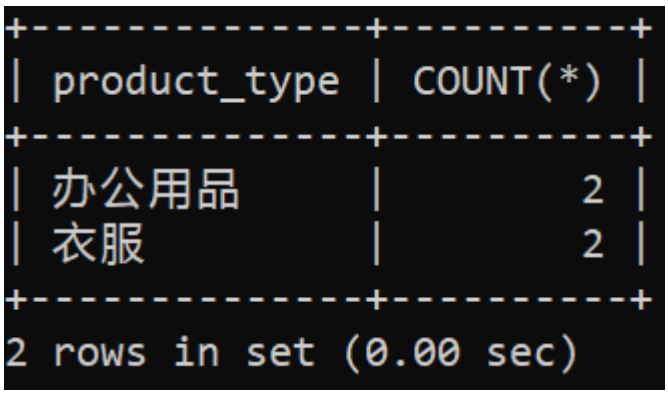


| product_type | COUNT(*) |
| --- | --- |
| 办公用品 | 2 |
| 厨房用具 | 4 |
| 衣服 | 2 |

3 rows in set (0.00 sec)

**5. `HAVING` (分组条件)**

```
SELECT
    product_type, COUNT(*)
FROM
    Product
GROUP BY product_type
HAVING COUNT(*) = 2; -- HAVING中可以放聚合函数,而WHERE不能
```



| product_type | COUNT(*) |
| --- | --- |
| 办公用品 | 2 |
| 衣服 | 2 |

2 rows in set (0.00 sec)

**6. `ORDER BY` (排序)**

```
-- 默认升序ASC，如果降序则使用DESC
SELECT
    product_id, product_name, sale_price, purchase_price
FROM
    Product
ORDER BY sale_price;
```

## 视图(`VIEW`)

> 视图与表的区别：表中保存的是实际的数据，二视图保存的是从表中去除数据所使用的SELECT语句。

```
-- 创建视图
CREATE VIEW 视图名称(<视图列名1>,<视图列名2>,...)
 AS <SELECT 语句>;

CREATE VIEW ProductSum (product_type,product_count)
AS SELECT product_type ,count(*)
    FROM Product GROUP BY product_type;
```

## 子查询

```
-- 在FROM子句中书写子查询
SELECT
    product_type, cnt_product
FROM
    (SELECT
        product_type, COUNT(*) AS cnt_product
    FROM
        Product
    GROUP BY product_type) AS ProductSum;
```

### 1.标量子查询

> 标量子查询就是返回单一值的子查询。无论是 SELECT 子句、GROUP BY 子句、HAVING 子句，还是 ORDER BY 子句，几乎所有的地方都可以使用标量子查询。

```
--   选取出销售单价（sale_price）高于全部商品的平均单价的商品
SELECT
    product_id, product_name, sale_price
FROM
    Product
WHERE
    sale_price > (SELECT
            AVG(sale_price)
        FROM
            Product);
```

| product_id | product_name | sale_price |
|---|---|---|
| 0003 | 运动T恤 | 3000 |
| 0004 | 菜刀 | 3000 |
| 0005 | 高压锅 | 6800 |

### 2.关联子查询

> 在细分的组内进行比较时，需要使用关联子查询。

```sql
-- 选取出各商品种类中高于该商品种类的平均销售单价的商品
SELECT
    product_type, product_name, sale_price
FROM
    Product AS P1
WHERE
    sale_price > (SELECT
            AVG(sale_price)
        FROM
            Product AS P2
        WHERE
            P1.product_type = P2.product_type
        GROUP BY product_type);
```

## 谓词

### 1. LIKE (一致查询)

表6-1 SampleLike表

| strcol（字符串） |
| --- |
| abcddd |
| dddabc |
| abdddc |
| abcdd |
| ddabc |
| abddc |

> " % ": 匹配任意数量字符串, " _ "(下划线): 匹配任意1个字符

- 前方一致：选取出"dd"开头的字符串

```sql
SELECT * FROM SamPleLike WHERE strcol LIKE "dd%";
```

- 中间一致：选取出包含"ddd"的字符串

```sql
SELECT * FROM SampleLike WHERE strcol LIKE "%dd%";
```

- 后方一致：选取出"abc+任意两个字符"的记录

```sql
SELECT * FROM SampleLikee WHERE strcol LIKE "abc__";
```

### 2. BETWEEN (范围查询)

```sql
-- 选取销售单价位于100~1000的商品信息
SELECT
    *
FROM
    product
WHERE
    sale_price BETWEEN 100 AND 1000;
```

| product_id | product_name | product_type | sale_price | purchase_price | regist_date |
|---|---|---|---|---|---|
| 0002 | 打孔器 | 办公用品 | 500 | 320 | 2009-09-11 |
| 0006 | 叉子 | 厨房用具 | 500 | NULL | 2009-09-20 |
| 0007 | 擦菜板 | 厨房用具 | 880 | 790 | 2008-04-28 |
| 0008 | 圆珠笔 | 办公用品 | 100 | NULL | 2009-11-11 |

**3. `IS NULL`(判断)**

```sql
-- 选取出进货价格(purchase_price)为NULL的商品信息
 SELECT
    *
FROM
    product
WHERE
    purchase_price IS NULL;

-- 选取出进货价格(purchase_price)不为NULL的商品信息
 SELECT
    *
FROM
    product
WHERE
    purchase_price IS NOT NULL;
```

**4. `IN`**

```sql
-- 通过IN来指定多个进货单价进行查询
SELECT
    product_name, purchase_price
FROM
    product
WHERE
    purchase_price IN (320 , 500, 5000);
```

| product_name | purchase_price |
| --- | --- |
| T恤衫 | 500 |
| 打孔器 | 320 |
| 高压锅 | 5000 |

```sql
-- 通过NOT IN来指定多个进货单价进行查询
SELECT
    product_name, purchase_price
FROM
    product
WHERE
    purchase_price NOT IN (320 , 500, 5000);
```

| product_name | purchase_price |
| --- | --- |
| 运动T恤 | 2800 |
| 菜刀 | 2800 |
| 擦菜板 | 790 |

> 使用子查询作为IN的参数

表6-2　ShopProduct（商店商品）表

| shop_id（商店） | shop_name（商店名称） | product_id（商品编号） | quantity（数量） |
|---|---|---|---|
| 000A | 东京 | 0001 | 30 |
| 000A | 东京 | 0002 | 50 |
| 000A | 东京 | 0003 | 15 |
| 000B | 名古屋 | 0002 | 30 |
| 000B | 名古屋 | 0003 | 120 |
| 000B | 名古屋 | 0004 | 20 |
| 000B | 名古屋 | 0006 | 10 |
| 000B | 名古屋 | 0007 | 40 |
| 000C | 大阪 | 0003 | 20 |
| 000C | 大阪 | 0004 | 50 |
| 000C | 大阪 | 0006 | 90 |
| 000C | 大阪 | 0007 | 70 |
| 000D | 福冈 | 0001 | 100 |

```
-- 选取出"大阪(000C)"在售商品(product_id)的销售单价
SELECT
    product_name, sale_price
FROM
    product
WHERE
    product_id IN (SELECT
            product_id
        FROM
            ShopProduct
        WHERE
            shop_id = '000C');
```

```
+--------------+------------+
| product_name | sale_price |
+--------------+------------+
| 运动T恤      |       3000 |
| 菜刀         |       3000 |
| 叉子         |        500 |
| 擦菜板       |        880 |
+--------------+------------+
```

### 5. EXISTS

> EXISTS判断是否存在满足某种条件的记录，如果存在则返回TRUE，否则返回FALSE.

```
-- 选取出"大阪(000C)"在售商品(product_id)的销售单价
SELECT
    product_name, sale_price
FROM
    product AS p
WHERE
    EXISTS( SELECT
            *
        FROM
            ShopProduct AS sp
        WHERE
            sp.shop_id = '000C'
                AND p.product_id = sp.product_id);
```

## CASE表达式

```
-- 简单CASE表达式
CASE <表达式>
     WHEN <表达式> THEN <表达式>
     WHEN <表达式> THEN <表达式>
                .
                .
     END <表达式>
END;
-- 根据product_type返回不同结果(简单CASE)
 SELECT
    product_name,
    CASE product_type
        WHEN '衣服' THEN CONCAT('A:', product_type)
        WHEN '办公用品' THEN CONCAT('B:', product_type)
        WHEN '厨房用具' THEN CONCAT('C:', product_type)
        ELSE NULL
    END AS abc_product_type
FROM
    product;

-- 搜索CASE表达式
CASE WHEN <求值表达式> THEN <表达式>
     WHEN <求值表达式> THEN <表达式>
                .
                .
     ELSE <表达式>
END;

-- 根据product_type返回不同结果(搜索CASE)
```

```sql
 SELECT
    product_name,
    CASE
        WHEN product_type = '衣服' THEN CONCAT('A:', product_type)
        WHEN product_type = '办公用品' THEN CONCAT('B:', product_type)
        WHEN product_type = '厨房用具' THEN CONCAT('C:', product_type)
    END AS abc_product_type
FROM
    product;
```

| product_name | abc_product_type |
| --- | --- |
| T恤衫 | A:衣服 |
| 打孔器 | B:办公用品 |
| 运动T恤 | A:衣服 |
| 菜刀 | C:厨房用具 |
| 高压锅 | C:厨房用具 |
| 叉子 | C:厨房用具 |
| 擦菜板 | C:厨房用具 |
| 圆珠笔 | B:办公用品 |

8 rows in set (0.00 sec)

```sql
-- 行列变换
 SELECT
    SUM(CASE
        WHEN product_type = '衣服' THEN sale_price
        ELSE 0
    END) AS sum_price_clothes,
    SUM(CASE
        WHEN product_type = '办公用品' THEN sale_price
        ELSE 0
    END) AS sum_price_office,
    SUM(CASE
        WHEN product_type = '厨房用具' THEN sale_price
        ELSE 0
    END) AS sum_price_kitchen
FROM
    product;
```

```
+--------------------+--------------------+--------------------+
| sum_price_clothes | sum_price_office | sum_price_kitchen |
+--------------------+--------------------+--------------------+
|               5000 |                600 |              11180 |
+--------------------+--------------------+--------------------+
```

## UNION(并集)

product表和product2表

```
mysql> select * from product;
+------------+--------------+--------------+------------+----------------+-------------+
| product_id | product_name | product_type | sale_price | purchase_price | regist_date |
+------------+--------------+--------------+------------+----------------+-------------+
| 0001       | T恤衫        | 衣服         |       2000 |            500 | 2009-09-20  |
| 0002       | 打孔器       | 办公用品     |        500 |            320 | 2009-09-11  |
| 0003       | 运动T恤      | 衣服         |       3000 |           2800 | NULL        |
| 0004       | 菜刀         | 厨房用具     |       3000 |           2800 | 2009-09-20  |
| 0005       | 高压锅       | 厨房用具     |       6800 |           5000 | 2009-01-15  |
| 0006       | 叉子         | 厨房用具     |        500 |           NULL | 2009-09-20  |
| 0007       | 擦菜板       | 厨房用具     |        880 |            790 | 2008-04-28  |
| 0008       | 圆珠笔       | 办公用品     |        100 |           NULL | 2009-11-11  |
+------------+--------------+--------------+------------+----------------+-------------+
8 rows in set (0.00 sec)

mysql> select * from product2;
+------------+--------------+--------------+------------+----------------+-------------+
| product_id | product_name | product_type | sale_price | purchase_price | regist_date |
+------------+--------------+--------------+------------+----------------+-------------+
| 0001       | T恤衫        | 衣服         |       1000 |            500 | 2008-09-20  |
| 0002       | 打孔器       | 办公用品     |        500 |            320 | 2009-09-11  |
| 0003       | 运动T恤      | 衣服         |       4000 |           2800 | NULL        |
| 0009       | 手套         | 衣服         |        800 |            500 | NULL        |
| 0010       | 水壶         | 厨房用具     |       2000 |           1700 | 2009-09-20  |
+------------+--------------+--------------+------------+----------------+-------------+
5 rows in set (0.03 sec)
```

```sql
-- 对表做加法
SELECT  * FROM product
UNION
SELECT  * FROM product
ORDER BY product_id;
-- 1.集合运算会去除重复的记录(如果保留重复行则使用UNION ALL)
-- 2.作为运算对象的记录的列数必须相同
-- 3.作为运算对象的记录的类型必须一致
-- 而已使用任意SELECT语句，但ORDER BY子句只能在最后使用
```

```
+------------+--------------+--------------+------------+----------------+-------------+
| product_id | product_name | product_type | sale_price | purchase_price | regist_date |
+------------+--------------+--------------+------------+----------------+-------------+
| 0001       | T恤衫        | 衣服         |       2000 |            500 | 2009-09-20  |
| 0002       | 打孔器       | 办公用品     |        500 |            320 | 2009-09-11  |
| 0003       | 运动T恤      | 衣服         |       3000 |           2800 | NULL        |
| 0004       | 菜刀         | 厨房用具     |       3000 |           2800 | 2009-09-20  |
| 0005       | 高压锅       | 厨房用具     |       6800 |           5000 | 2009-01-15  |
| 0006       | 叉子         | 厨房用具     |        500 |           NULL | 2009-09-20  |
| 0007       | 擦菜板       | 厨房用具     |        880 |            790 | 2008-04-28  |
| 0008       | 圆珠笔       | 办公用品     |        100 |           NULL | 2009-11-11  |
+------------+--------------+--------------+------------+----------------+-------------+
8 rows in set (0.00 sec)
```

## INTERSECT(交集)

```
-- MySQL不支持
SELECT product_id, product_name
    FROM Product
INTERSECT
SELECT product_id, product_name
    FROM Product2
ORDER BY product_id;
```

```
product_id| product_name
----------+--------------
0001      | T恤衫
0002      | 打孔器
0003      | 运动T恤
```

## EXCEPT(差集)

```
-- MySQL不支持
SELECT product_id, product_name
    FROM product
EXCEPT
SELECT product_id, product_name
    FROM product2
ORDER BY product_id;
```

```
product_id| product_name
----------+--------------
0004      | 菜刀
0005      | 高压锅
0006      | 叉子
0007      | 擦菜板
0008      | 圆珠笔
```

## JOIN(联结)

> 表product和表ShopProduct

```
mysql> select * from product;
+------------+--------------+--------------+------------+----------------+-------------+
| product_id | product_name | product_type | sale_price | purchase_price | regist_date |
+------------+--------------+--------------+------------+----------------+-------------+
| 0001       | T恤衫        | 衣服         |       2000 |            500 | 2009-09-20  |
| 0002       | 打孔器       | 办公用品     |        500 |            320 | 2009-09-11  |
| 0003       | 运动T恤      | 衣服         |       3000 |           2800 | NULL        |
| 0004       | 菜刀         | 厨房用具     |       3000 |           2800 | 2009-09-20  |
| 0005       | 高压锅       | 厨房用具     |       6800 |           5000 | 2009-01-15  |
| 0006       | 叉子         | 厨房用具     |        500 |           NULL | 2009-09-20  |
| 0007       | 擦菜板       | 厨房用具     |        880 |            790 | 2008-04-28  |
| 0008       | 圆珠笔       | 办公用品     |        100 |           NULL | 2009-11-11  |
+------------+--------------+--------------+------------+----------------+-------------+
8 rows in set (0.00 sec)

mysql> select * from ShopProduct;
+---------+-----------+------------+----------+
| shop_id | shop_name | product_id | quantity |
+---------+-----------+------------+----------+
| 000A    | 东京      | 0001       |       30 |
| 000A    | 东京      | 0002       |       50 |
| 000A    | 东京      | 0003       |       15 |
| 000B    | 名古屋    | 0002       |       30 |
| 000B    | 名古屋    | 0003       |      120 |
| 000B    | 名古屋    | 0004       |       20 |
| 000B    | 名古屋    | 0006       |       10 |
| 000B    | 名古屋    | 0007       |       40 |
| 000C    | 大阪      | 0003       |       20 |
| 000C    | 大阪      | 0004       |       50 |
| 000C    | 大阪      | 0006       |       90 |
| 000C    | 大阪      | 0007       |       70 |
| 000D    | 福冈      | 0001       |      100 |
+---------+-----------+------------+----------+
13 rows in set (0.00 sec)
```

**1.`INNER JOIN`(内联结)**

```sql
-- 连接表porduct和表ShopProduct，连接键：product_id
SELECT
    sp.shop_id,
    sp.shop_name,
    sp.product_id,
    p.product_id,
    p.product_name
FROM
    ShopProduct AS sp
        INNER JOIN
    product AS p ON sp.product_id = p.product_id
ORDER BY sp.product_id;
```

| shop_id | shop_name | product_id | product_id | product_name |
|---|---|---|---|---|
| 000A | 东京 | 0001 | 0001 | T恤衫 |
| 000D | 福冈 | 0001 | 0001 | T恤衫 |
| 000B | 名古屋 | 0002 | 0002 | 打孔器 |
| 000A | 东京 | 0002 | 0002 | 打孔器 |
| 000C | 大阪 | 0003 | 0003 | 运动T恤 |
| 000A | 东京 | 0003 | 0003 | 运动T恤 |
| 000B | 名古屋 | 0003 | 0003 | 运动T恤 |
| 000B | 名古屋 | 0004 | 0004 | 菜刀 |
| 000C | 大阪 | 0004 | 0004 | 菜刀 |
| 000C | 大阪 | 0006 | 0006 | 叉子 |
| 000B | 名古屋 | 0006 | 0006 | 叉子 |
| 000B | 名古屋 | 0007 | 0007 | 擦菜板 |
| 000C | 大阪 | 0007 | 0007 | 擦菜板 |

13 rows in set (0.00 sec)

```sql
-- 只查看东京店(000A)的信息,WHERE筛选
SELECT
    sp.shop_id,
    sp.shop_name,
    sp.product_id,
    p.product_id,
    p.product_name
FROM
    ShopProduct AS sp
        INNER JOIN
    product AS p ON sp.product_id = p.product_id
WHERE sp.shop_id="000A";
```

| shop_id | shop_name | product_id | product_id | product_name |
|---|---|---|---|---|
| 000A | 东京 | 0001 | 0001 | T恤衫 |
| 000A | 东京 | 0002 | 0002 | 打孔器 |
| 000A | 东京 | 0003 | 0003 | 运动T恤 |

3 rows in set (0.00 sec)

**2.`OUTER JOIN`(外联结)**

1.`LEFT JOIN`(左外联结)

```sql
-- 联结主键为左表主键,
SELECT
    sp.shop_id,
    sp.shop_name,
    sp.product_id,
    p.product_id,
    p.product_name
FROM
    ShopProduct AS sp
        LEFT JOIN
    product AS p ON sp.product_id = p.product_id
ORDER BY sp.product_id;
```

```
+---------+-----------+------------+------------+--------------+
| shop_id | shop_name | product_id | product_id | product_name |
+---------+-----------+------------+------------+--------------+
| 000A    | 东京      | 0001       | 0001       | T恤衫        |
| 000D    | 福冈      | 0001       | 0001       | T恤衫        |
| 000B    | 名古屋    | 0002       | 0002       | 打孔器       |
| 000A    | 东京      | 0002       | 0002       | 打孔器       |
| 000B    | 名古屋    | 0003       | 0003       | 运动T恤      |
| 000A    | 东京      | 0003       | 0003       | 运动T恤      |
| 000C    | 大阪      | 0003       | 0003       | 运动T恤      |
| 000C    | 大阪      | 0004       | 0004       | 菜刀         |
| 000B    | 名古屋    | 0004       | 0004       | 菜刀         |
| 000C    | 大阪      | 0006       | 0006       | 叉子         |
| 000B    | 名古屋    | 0006       | 0006       | 叉子         |
| 000C    | 大阪      | 0007       | 0007       | 擦菜板       |
| 000B    | 名古屋    | 0007       | 0007       | 擦菜板       |
+---------+-----------+------------+------------+--------------+
13 rows in set (0.00 sec)
```

2.`RIGHT JOIN`(右外联结)

```sql
-- 联结主键为右表主键,
SELECT
    sp.shop_id,
    sp.shop_name,
    sp.product_id,
    p.product_id,
    p.product_name
FROM
    ShopProduct AS sp
        LEFT JOIN
    product AS p ON sp.product_id = p.product_id
ORDER BY sp.product_id;
```

| shop_id | shop_name | product_id | product_id | product_name |
|---|---|---|---|---|
| NULL | NULL | NULL | 0008 | 圆珠笔 |
| NULL | NULL | NULL | 0005 | 高压锅 |
| 000A | 东京 | 0001 | 0001 | T恤衫 |
| 000D | 福冈 | 0001 | 0001 | T恤衫 |
| 000B | 名古屋 | 0002 | 0002 | 打孔器 |
| 000A | 东京 | 0002 | 0002 | 打孔器 |
| 000C | 大阪 | 0003 | 0003 | 运动T恤 |
| 000A | 东京 | 0003 | 0003 | 运动T恤 |
| 000B | 名古屋 | 0003 | 0003 | 运动T恤 |
| 000B | 名古屋 | 0004 | 0004 | 菜刀 |
| 000C | 大阪 | 0004 | 0004 | 菜刀 |
| 000C | 大阪 | 0006 | 0006 | 叉子 |
| 000B | 名古屋 | 0006 | 0006 | 叉子 |
| 000B | 名古屋 | 0007 | 0007 | 擦菜板 |
| 000C | 大阪 | 0007 | 0007 | 擦菜板 |

15 rows in set (0.00 sec)

**3.`CROSS JOIN`(交叉联结)**

```
-- 交叉联结做笛卡尔积运算，结果的记录数为联结表的乘积
-- 比如A表右13条记录，B表有8条记录，则A与B交叉联结的结果有13*8条记录
SELECT
    sp.shop_id,
    sp.shop_name,
    sp.product_id,
    p.product_id,
    p.product_name
FROM
    ShopProduct AS sp
        CROSS JOIN
    product AS p
ORDER BY sp.product_id;
```