本笔记为阿里云天池龙珠计划SQL训练营的学习内容，链接为：https://tianchi.aliyun.com/specials/promotion/aicampsql

Task02：SQL 基础查询与排序

一、SELECT 语句基础

1.1 从表中选取数据

SELECT 语句

从表中选取数据时需要使用 SELECT 语句，也就是只从表中选出（SELECT）必要数据的意思。通过 SELECT 语句查询并选取出必要数据的过程称为匹配查询或查询（query）。

基本 `SELECT` 语句包含了 `SELECT` 和 `FROM` 两个子句（clause）。示例如下：

```
SELECT < 列名 >,< 列名 >
FROM < 表名 >;
```

其中，SELECT 子句中列举了希望从表中查询出的列的名称，而 FROM 子句则指定了选取出数据的表的名称。

1.2 从表中选取符合条件的数据

WHERE 语句

当不需要取出全部数据，而是选取出满足“商品种类为衣服”“销售单价在 1000 日元以上”等某些条件的数据时，使用 WHERE 语句。

SELECT 语句通过 WHERE 子句来指定查询数据的条件。在 WHERE 子句中可以指定“某一列的值和这个字符串相等”或者“某一列的值大于这个数字”等条件。执行含有这些条件的 SELECT 语句，就可以查询出只符合该条件的记录了。

```
SELECT < 列名 >, ……
FROM < 表名 >
WHERE < 条件表达式 >;
```

比较下面两者输出结果的不同：

```
-- 用来选取 product_type 列为衣服的记录的 SELECT 语句
SELECT product_name, product_type
FROM product
```

```
WHERE product_type = '衣服';
-- 也可以选取出不是查询条件的列（条件列与输出列不同）
SELECT product_name
FROM product
WHERE product_type = '衣服';
```

1.3 相关法则

- 星号（*）代表全部列的意思。
- SQL 中可以随意使用换行符，不影响语句执行（但不可插入空行）。
- 设定汉语别名时需要使用双引号（"）括起来。
- 在 SELECT 语句中使用 **DISTINCT** 可以删除重复行。
- 注释是 SQL 语句中用来标识说明或者注意事项的部分。分为单行注释 "--" 和多行注释两种 "/* */"。

```
-- 想要查询出全部列时，可以使用代表所有列的星号（*）。
SELECT *
FROM < 表名 >;
-- SQL 语句可以使用 AS 关键字为列设定别名（用中文时需要双引号（""））。
SELECT product_id     AS id,
       product_name   AS name,
       purchase_price AS "进货单价"
FROM product;
-- 使用 DISTINCT 删除 product_type 列中重复的数据
SELECT DISTINCT product_type
FROM product;
```

二、算术运算符和比较运算符

2.1 算术运算符

SQL 语句中可以使用的四则运算的主要运算符如下：

含义	运算符
加法	+
减法	-
乘法	*
除法	/

2.2 比较运算符

```sql
-- 选取出 sale_price 列为 500 的记录
SELECT product_name, product_type
FROM product
WHERE sale_price = 500;
```

SQL 常见比较运算符如下：

运算符	含义
=	和 *某* 相等
<>	和 *某* 不相等
>=	大于等于 *某*
>	大于 *某*

<=	小于等于 *某*
<	小于 *某*

2.3 常用法则

- SELECT 子句中可以使用常数或者表达式。
- 使用比较运算符时一定要注意**不等号**和**等号**的位置。
- 字符串类型的数据原则上按照字典顺序进行排序，不能与数字的大小顺序混淆。
- 希望选取 `NULL` 记录时，需要在条件表达式中使用 `IS NULL` 运算符。希望选取不是 NULL 的记录时，需要在条件表达式中使用 IS NOT NULL 运算符。

相关代码如下：

```
-- SQL 语句中也可以使用运算表达式
SELECT
    product_name,
    sale_price,
    sale_price * 2 AS "sale_price x2"
FROM product;
-- WHERE 子句的条件表达式中也可以使用计算表达式
SELECT product_name, sale_price, purchase_price
FROM product
WHERE sale_price-purchase_price >= 500;
```

```
/* 对字符串使用不等号
首先创建 chars 并插入数据
选取出大于'2'的 SELECT 语句 */

-- DDL：创建表
```

```sql
CREATE TABLE chars(
    chr CHAR (3) NOT NULL,
    PRIMARY KEY(chr)
);
-- 选取出大于 '2' 的数据的 SELECT 语句 ('2' 为字符串)
SELECT chr
FROM chars
WHERE chr > '2';
```

```sql
-- 选取 NULL 的记录
SELECT product_name, purchase_price
FROM product
WHERE purchase_price IS NULL;
-- 选取不为 NULL 的记录
SELECT product_name, purchase_price
FROM product
WHERE purchase_price IS NOT NULL;
```

三、逻辑运算符

3.1 NOT 运算符

想要表示“不是……”时，除了前文的 <> 运算符外，还存在另外一个表示否定、使用范围更广的运算符：`NOT`。

NOT 不能单独使用，如下例：

```sql
-- 选取出销售单价大于等于 1000 日元的记录
SELECT product_name, product_type, sale_price
FROM product
WHERE sale_price >= 1000;
```

```sql
-- 向上面代码清单的查询条件中添加 NOT 运算符
SELECT product_name, product_type, sale_price
FROM product
WHERE NOT sale_price >= 1000;
```

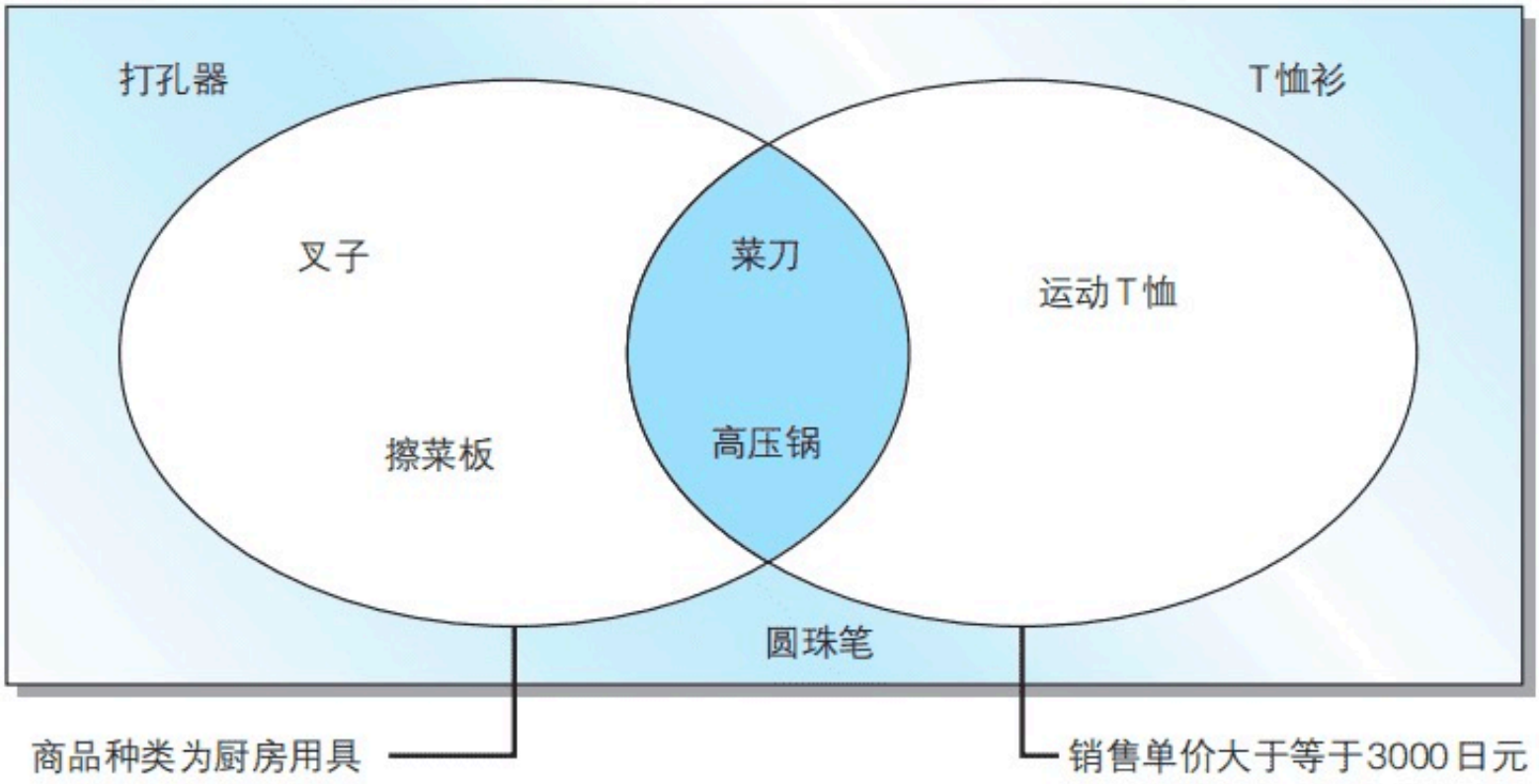
3.2 AND 运算符和 OR 运算符

当希望同时使用多个查询条件时，可以使用 `AND` 或者 `OR` 运算符。

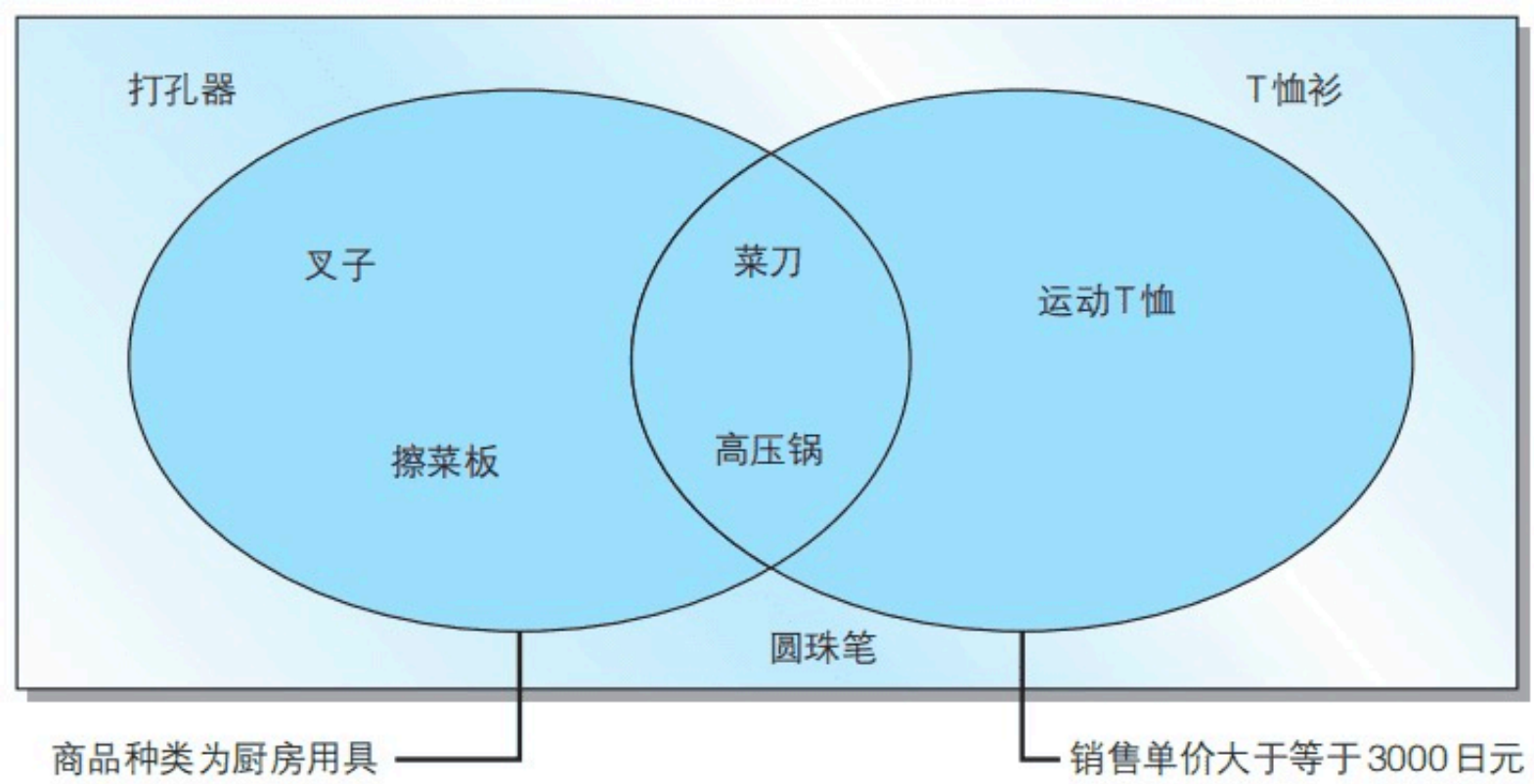
- AND 相当于“并且”，类似数学中的取交集；
- OR 相当于“或者”，类似数学中的取并集。

如下图所示：

- AND 运算符工作效果图



- OR 运算符工作效果图



3.3 通过括号优先处理

如果要查找这样一个商品，该怎么处理?

> “商品种类为办公用品”并且“登记日期是 2009 年 9 月 11 日或者 2009 年 9 月 20 日”
> 理想结果为“打孔器”，但当你输入以下信息时，会得到错误结果

```
-- 将查询条件原封不动地写入条件表达式，会得到错误结果
SELECT product_name, product_type, regist_date
FROM product
WHERE product_type = '办公用品'
AND regist_date = '2009-09-11'
```

```
OR regist_date = '2009-09-20';
```

错误的原因是 **是 AND 运算符优先于 OR 运算符**，想要优先执行 OR 运算，可以使用 **括号**：

```
-- 通过使用括号让 OR 运算符先于 AND 运算符执行
SELECT product_name, product_type, regist_date
FROM product
WHERE product_type = '办公用品'
AND ( regist_date = '2009-09-11'
OR regist_date = '2009-09-20');
```

3.4 真值表

复杂运算时该怎样理解?

当碰到条件较复杂的语句时，理解语句含义并不容易，这时可以采用 **真值表** 来梳理逻辑关系。

什么是真值?

本节介绍的三个运算符 NOT、AND 和 OR 称为逻辑运算符。

这里所说的逻辑就是对真值进行操作的意思。**真值** 就是值为真（TRUE）或假 （FALSE）其中之一的值。

例如，对于 `sale_price >= 3000` 这个查询条件来说，由于 product_name 列为 '运动 T 恤' 的记录的 sale_price 列的值是 2800，因此会返回假（FALSE），而 product_name 列为 '高压锅' 的记录的 sale_price 列的值是 5000，所以返回真（TRUE）。

AND 运算符：两侧的真值都为真时返回真，除此之外都返回假。

OR 运算符：两侧的真值只要有一个不为假就返回真，只有当其两侧的真值都为假时才返回假。

NOT 运算符：只是单纯的将真转换为假，将假转换为真。

真值表

AND

P	Q	P AND Q
真	真	真
真	假	假
假	真	假
假	假	假

OR

P	Q	P OR Q
真	真	真
真	假	真
假	真	真
假	假	假

NOT

P	NOT P
真	假
假	真

查询条件为 `P AND (Q OR R)` 的真值表

P **AND** (Q **OR** R)

P	Q	R	Q OR R	P AND (Q OR R)
真	真	真	真	真
真	真	假	真	真
真	假	真	真	真
真	假	假	假	假
假	真	真	真	假
假	真	假	真	假
假	假	真	真	假
假	假	假	假	假

P：商品种类为办公用品
Q：登记日期是2009年9月11日
R：登记日期是2009年9月20日
Q OR R：登记日期是2009年9月11日或者2009年9月20日
P AND（Q OR R）：商品种类为办公用品，并且，登记日期是2009年9月11日或者2009年9月20日

3.5 含有 NULL 时的真值

NULL 的真值结果既不为真，也不为假，因为并不知道这样一个值。

那该如何表示呢?

这时真值是除真假之外的第三种值——**不确定**（UNKNOWN）。一般的逻辑运算并不存在这第三种值。SQL 之外的语言也基本上只使用真和假这两种真值。与通常的逻辑运算被称为二值逻辑相对，只有 SQL 中的逻辑运算被称为三值逻辑。

三值逻辑下的 AND 和 OR 真值表为：

AND

P	Q	P AND Q
真	真	真
真	假	假
真	不确定	不确定
假	真	假
假	假	假
假	不确定	假
不确定	真	不确定
不确定	假	假
不确定	不确定	不确定

OR

P	Q	P OR Q
真	真	真
真	假	真
真	不确定	真
假	真	真
假	假	假
假	不确定	不确定
不确定	真	真
不确定	假	不确定
不确定	不确定	不确定

练习题 - 第一部分

练习题 1

编写一条 SQL 语句，从 product（商品）表中选取出“登记日期（regist 在 2009 年 4 月 28 日之后”的商品，查询结果要包含 product_name 和 regist_date 两列。

```
SELECT product_name, regist_date
FROM product
WHERE regist_date > '2009-04-28';
```

练习题 2

请说出对 product 表执行如下 3 条 SELECT 语句时的返回结果。

```
SELECT *
FROM product
WHERE purchase_price = NULL;
```

```
SELECT *
FROM product
WHERE purchase_price <> NULL;
```

```
SELECT *
FROM product
WHERE product_name > NULL;
```

练习题 3

代码清单 2-22（2-2 节）中的 SELECT 语句能够从 product 表中取出“销售单价（saleprice）比进货单价（purchase price）高出 500 日元以上”的商品。请写出两条可以得到相同结果的 SELECT 语句。执行结果如下所示。

```
product_name | sale_price | purchase_price
------------ + ---------- + ------------
T 恤衫         |    1000    | 500
运动 T 恤      |    4000    | 2800
高压锅         |    6800    | 5000
```

```
-- SELECT 语句 1
SELECT product_name, sale_price, purchase_price
```

```
FROM product
WHERE sale_price >= purchase_price + 500;

-- SELECT 语句 2
SELECT product_name, sale_price, purchase_price
FROM product
WHERE sale_price - 500 >= purchase_price;
```

练习题 4

请写出一条 SELECT 语句，从 product 表中选取出满足“销售单价打九折之后利润高于 100 日元的办公用品和厨房用具”条件的记录。查询结果要包括 product_name 列、product_type 列以及销售单价打九折之后的利润（别名设定为 profit）。

提示：销售单价打九折，可以通过 saleprice 列的值乘以 0.9 获得，利润可以通过该值减去 purchase_price 列的值获得。

```
SELECT product_name, product_type,
       sale_price * 0.9 - purchase_price AS profit
FROM product
WHERE sale_price * 0.9 - purchase_price > 100
   AND (   product_type = '办公用品'
        OR product_type = '厨房用具');
```

```
SELECT product_type, SUM(sale_price), SUM(purchase_price)
FROM product
GROUP BY product_type
HAVING SUM(sale_price) > SUM(purchase_price) * 1.5;
```

上面这段代码是什么意思呢？如何翻译成中文！

四、对表进行聚合查询

4.1 聚合函数

SQL 中用于汇总的函数叫做聚合函数。以下五个是最常用的聚合函数：

- COUNT：计算表中的记录数（行数）
- SUM：计算表中数值列中数据的合计值
- AVG：计算表中数值列中数据的平均值
- MAX：求出表中任意列中数据的最大值
- MIN：求出表中任意列中数据的最小值

在使用 COUNT 的时候要特别注意的一点是,COUNT(*) 和 COUNT(字段) 的区别.

请沿用第一章的数据，使用以下操作熟练函数：

```
-- 计算全部数据的行数（包含 NULL）
SELECT COUNT(*)
FROM product;

-- 计算 NULL 以外数据的行数
SELECT COUNT(purchase_price)
FROM product;

-- 计算销售单价和进货单价的合计值
SELECT SUM(sale_price), SUM(purchase_price)
FROM product;

-- 计算销售单价和进货单价的平均值
SELECT AVG(sale_price), AVG(purchase_price)
FROM product;

-- MAX 和 MIN 也可用于非数值型数据
SELECT MAX(regist_date), MIN(regist_date)
```

```
FROM product;
```

MySQL 聚合函数主要用于数据统计，比如计数、求和、求平均数等。聚合函数结合分组操作，您可以在更多的维度统计数据。

- AVG
  MySQL AVG() 函数计算并返回表达式的平均值。
- BIT_AND
  MySQL BIT_AND() 函数是一个聚合函数，它对所有的非 null 输入值执行"按位与"运算。
- BIT_OR
  MySQL BIT_OR() 函数是一个聚合函数，它对所有的非 null 输入值执行"按位或"运算。
- BIT_XOR
  MySQL BIT_XOR() 函数是一个聚合函数，它对所有的非 null 输入值执行"按位异或"运算。
- COUNT
  MySQL COUNT() 函数用于统计表达式的数量。
- GROUP_CONCAT
  MySQL GROUP_CONCAT() 函数将一个分组中指定的列或表达式的值连接成一个字符串并返回。
- JSON_ARRAYAGG
  MySQL JSON_ARRAYAGG() 函数将指定的列或者表达式的值聚合为一个 JSON 数组。
- JSON_OBJECTAGG
  MySQL JSON_OBJECTAGG() 函数将由第一个参数作为键和第二个参数作为值的键值对聚合为一个 JSON 对象。
- MAX
  MySQL MAX() 函数返回表达式的最大值。
- MIN
  MySQL MIN() 函数返回表达式的最小值。
- STD
  MySQL STD() 计算所有非 null 输入值的总体标准差并返回结果。
- STDDEV

MySQL STDDEV() 函数计算所有非 null 输入值的总体标准差并返回结果。

- STDDEV_POP
  MySQL STDDEV_POP() 函数计算所有非 null 输入值的总体标准差并返回结果。
- STDDEV_SAMP
  MySQL STDDEV_SAMP() 函数计算所有非 null 输入值的样本标准差并返回结果。
- SUM
  MySQL SUM() 函数计算所有指定的非 NULL 值的总和并返回。
- VAR_POP
  MySQL VAR_POP() 函数计算所有非 null 输入值的总体方差（总体标准差的平方）并返回结果。
- VAR_SAMP
  MySQL VAR_SAMP() 函数计算所有非 null 输入值的样本方差（样本标准差的平方）并返回结果。
- VARIANCE
  MySQL VARIANCE() 函数计算所有非 null 输入值的总体方差（总体标准差的平方）并返回结果。

4.2 使用聚合函数删除重复值

```sql
-- 计算去除重复数据后的数据行数
SELECT COUNT(DISTINCT product_type)
FROM product;

-- 是否使用 DISTINCT 时的动作差异（SUM 函数）
SELECT SUM(sale_price), SUM(DISTINCT sale_price)
FROM product;
```

4.3 常用法则

- COUNT 函数的结果根据参数的不同而不同。COUNT(*) 会得到包含 NULL 的数据行数，而 COUNT(< 列名 >) 会得到 NULL 之外的数据行数。

- 聚合函数会将 NULL 排除在外。但 COUNT(*) 例外，并不会排除 NULL。
- MAX/MIN 函数几乎适用于所有数据类型的列。SUM/AVG 函数只适用于数值类型的列。
- 想要计算值的种类时，可以在 COUNT 函数的参数中使用 DISTINCT。
- 在聚合函数的参数中使用 DISTINCT，可以删除重复数据。

五、对表进行分组

5.1 GROUP BY 语句

之前使用聚合函数都是会整个表的数据进行处理，当你想将进行分组汇总时（即：将现有的数据按照某列来汇总统计），GROUP BY 可以帮助你：

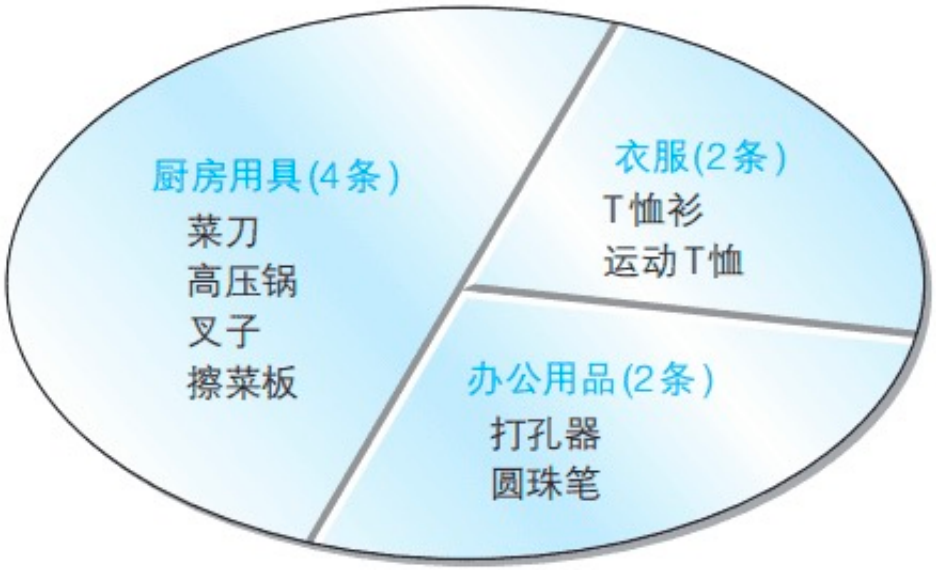
```
SELECT < 列名 1>,< 列名 2>, < 列名 3>, ……
FROM < 表名 >
GROUP BY < 列名 1>, < 列名 2>, < 列名 3>, ……;
```

看一看是否使用 GROUP BY 语句的差异：

```
-- 按照商品种类统计数据行数
SELECT product_type, COUNT(*)
FROM product
GROUP BY product_type;

-- 不含 GROUP BY
SELECT product_type, COUNT(*)
FROM product
```

按照商品种类对表进行切分



这样，GROUP BY 子句就像切蛋糕那样将表进行了分组。在 GROUP BY 子句中指定的列称为 **聚合键** 或者 **分组列**。

5.2 聚合键中包含 NULL 时

将进货单价（purchase_price）作为聚合键举例：

```
SELECT purchase_price, COUNT(*)
FROM product
GROUP BY purchase_price;
```

此时会将 NULL 作为一组特殊数据进行处理

5.3 GROUP BY 书写位置

GROUP BY 的子句书写顺序有严格要求，不按要求会导致 SQL 无法正常执行，目前出现过的子句 **书写顺序** 为：

1.SELECT → 2. FROM → 3. WHERE → 4. GROUP BY

其中前三项用于筛选数据，GROUP BY 对筛选出的数据进行处理

5.4 在 WHERE 子句中使用 GROUP BY

```
SELECT purchase_price, COUNT(*)
FROM product
WHERE product_type = '衣服'
GROUP BY purchase_price;
```

5.5 常见错误

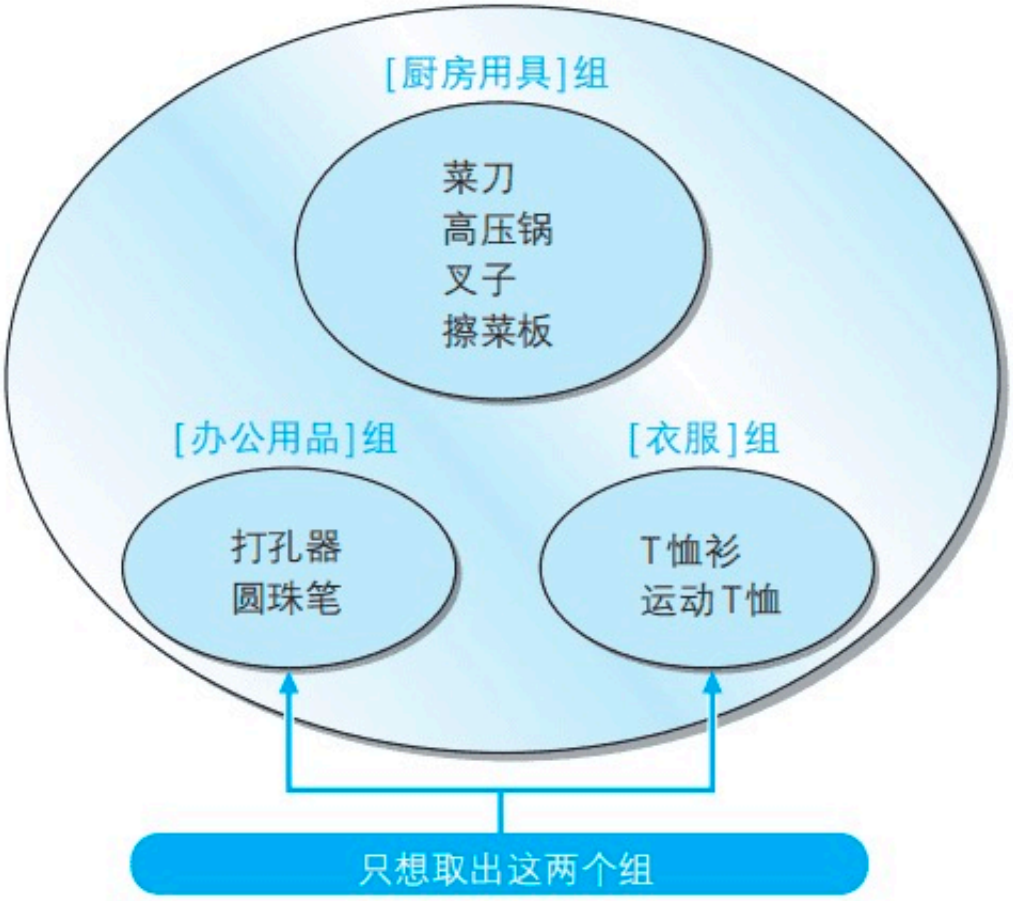
在使用聚合函数及 GROUP BY 子句时，经常出现的错误有：

1. 在聚合函数的 SELECT 子句中写了聚合键以外的列使用 COUNT 等聚合函数时，SELECT 子句中如果出现列名，**只能**是 GROUP BY 子句中指定的列名（也就是聚合键）。
2. 在 GROUP BY 子句中使用列的别名 SELECT 子句中可以通过 AS 来指定别名，但在 GROUP BY 中**不能**使用别名。因为在 DBMS 中 ,**SELECT 子句在 GROUP BY 子句后执行**。
3. 在 WHERE 中使用聚合函数，原因是聚合函数的使用前提是结果集已经确定，而 WHERE 还处于确定结果集的过程中，所以相互矛盾会引发错误。 如果想指定条件，可以在 SELECT，HAVING 以及 ORDER BY 子句中使用聚合函数。

六、为聚合结果指定条件

6.1 用 HAVING 得到特定分组

将表使用 GROUP BY 分组后，怎样才能只取出其中两组?



这里 WHERE 不可行，因为，WHERE 子句只能指定记录（行）的条件，而不能用来指定组的条件（例如，“数据行数为 2 行”或者“平均值为 500”等）。

可以在 GROUP BY 后使用 HAVING 子句。

HAVING 的用法类似 WHERE

6.2 HAVING 特点

HAVING 子句用于对分组进行过滤，可以使用数字、聚合函数和 GROUP BY 中指定的列名（聚合键）。

```sql
-- 数字
SELECT product_type, COUNT(*)
FROM product
GROUP BY product_type
HAVING COUNT(*) = 2;

-- 错误形式（因为 product_name 不包含在 GROUP BY 聚合键中）
SELECT product_type, COUNT(*)
FROM product
GROUP BY product_type
HAVING product_name = '圆珠笔';
```

七、对查询结果进行排序

7.1 ORDER BY

SQL 中的执行结果是随机排列的，当需要按照特定顺序排序时，可已使用 **ORDER BY** 子句。

```sql
SELECT < 列名 1>, < 列名 2>, < 列名 3>, ……
FROM < 表名 >
ORDER BY < 排序基准列 1>, < 排序基准列 2>, ……
```

默认为升序排列，降序排列为 DESC

```sql
-- 降序排列
SELECT product_id, product_name, sale_price, purchase_price
FROM product
ORDER BY sale_price DESC;
```

```sql
-- 多个排序键
SELECT product_id, product_name, sale_price, purchase_price
FROM product
ORDER BY sale_price, product_id;

-- 当用于排序的列名中含有 NULL 时，NULL 会在开头或末尾进行汇总。
SELECT product_id, product_name, sale_price, purchase_price
FROM product
ORDER BY purchase_price;
```

7.2 ORDER BY 中列名可使用别名

前文讲 GROUP BY 中提到，GROUP BY 子句中不能使用 SELECT 子句中定义的别名，但是在 ORDER BY 子句中却可以使用别名。为什么在 GROUP BY 中不可以而在 ORDER BY 中可以呢?

这是因为 SQL 在使用 HAVING 子句时 SELECT 语句的 **执行顺序** 为：

FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY

其中 SELECT 的执行顺序在 GROUP BY 子句之后，ORDER BY 子句之前。也就是说，当在 ORDER BY 中使用别名时，已经知道了 SELECT 设置的别名存在，但是在 GROUP BY 中使用别名时还不知道别名的存在，所以 **在 ORDER BY 中可以使用别名，但是在 GROUP BY 中不能使用别名。**

练习题 - 第二部分

练习题 5

请指出下述 SELECT 语句中所有的语法错误。

```sql
-- 本 SELECT 语句中存在错误。
```

```sql
SELECT product_id, SUM (product_name)
FROM product
GROUP BY product_type
WHERE regist_date > '2009-09-01';
```

- 错误 1
  字符型字段 product_name 不可以进行 SUM 聚合
- 错误 2
  WHERE 语句应该书写在 GROUP BY 语句之前（ FROM 语句之后）
- 错误 3
  GROUP BY 字段（product_type ）与 SELECT 字段不同（product_id ）

练习题 6

请编写一条 SELECT 语句，求出销售单价（sale_price 列）合计值大于进货单价（purchase_price 列）合计值 1.5 倍的商品种类。执行结果如下所示。


```sql
SELECT product_type, SUM(sale_price),SUM(purchase_price)
FROM product
GROUP BY product_type
HAVING SUM(sale_price) > SUM(purchase_price) * 1.5;
```

练习题 7

此前我们曾经使用 SELECT 语句选取出了 product（商品）表中的全部记录。当时我们使用了 ORDERBY 子句来指定排列顺序，但现在已经无法记起当时如何指定的了。请根据下列执行结果，思考 ORDERBY 子句的内容。

product_id	product_name	product_type	sale_price	purchase_price	regist_date
0003	运动T恤	衣服	4000	2800	
0008	圆珠笔	办公用品	100		2009-11-11
0006	叉子	厨房用具	500		2009-09-20
0001	T恤衫	衣服	1000	500	2009-09-20
0004	菜刀	厨房用具	3000	2800	2009-09-20
0002	打孔器	办公用品	500	320	2009-09-11
0005	高压锅	厨房用具	6800	5000	2009-01-15
0007	擦菜板	厨房用具	880	790	2008-04-28

```sql
SELECT *
FROM product
ORDER BY regist_date DESC, sale_price;
```