数据查询：基本子句

一、 `SELECT` 子句

1. 指定字段查询

示例1: 通过指定字段查询 `product` 表的所有数据

示例2：查询 `product` 表中 `product_id, product_name, price` 字段对应的数据

示例3：通过 `*` 查询 `product` 表的所有数据

示例4: 查询前10个 `product` 表中的商品记录

二、 `WHERE` 子句

1. 带关系运算符的查询

示例5：查询 `product` 表中 `product_place` 为 `'天津'` 的产品信息

示例6：使用 `SELECT` 语句查询 `product_name` 为 `“京瓷KM-4030复印机”` 的商品价格

示例7：查询 `product` 表中 `price` 大于等于1000的商品代码和名称

2. 带 `IN` 关键字的查询: 限定值的可选集

示例8：查找 `product` 表中产地为 `'天津', '北京', '日本'` 的商品的全部信息

3. 带 `BETWEEN ... AND...` 关键字的查询：限定值的范围

示例9：查询 `prodct` 表中 `Price` 值在200和500之间的商品信息

示例10：查询 `product` 表中 `Price` 值不在200和500之间的商品信息

4. `NULL` 判断：查询某些列值是否为空

示例11：查询 `product` 表中 `product_Place` 为空值的商品名称和价格

示例12：查询 `product` 表中 `product_place` 不为空值的记录

5. 字段前带 `DISTINCT` 关键字的查询：剔除查询结果中的重复行

示例13：查询 `product` 表中 `product_place` 字段的值，查询记录不能重复

示例14：找出 `product` 表中不同的根类别和子类别的组合

三、匹配字符串

1. 带 `LIKE` 关键字的字符串简单匹配

示例15：查找 `product` 表中商品名称含有 `复印机` 的商品的 `product_name, price, product_place`

示例16: 找出 `product` 表商品名称含有 `_` 或 `%` 的记录

示例17: 查找 `product` 表中 `product_name` 末尾字符串为 `'复印机'` 的 `product_name, price, product_place`

示例18: 查询 `product` 表中 `product_place` 为 `'江苏'` 的 `product_name, price` 和 `product_place`

四、复合查询

1. 带 `AND` 关键字的多条件查询: 连接两个或者多个查询条件

示例19：找出 `product` 表中 `product_name` 含 `复印机` 且 `product_place` 在 `天津` 的记录

2. 带 `OR` 关键字的多条件查询: 匹配其中的一个条件

示例20：找出 `product` 中 `product_name` 含 `'复印机'` 或 `'过胶机'` 的商品记录

示例21: 找出 `product_name` 含 `'复印机'` 和 `'过胶机'` ，且 `product_plance` 为 `天津` 的记录

练习

附: `MySQL` 数据的写入写出

1. 将查询结论写入外部文件

将 `product` 表的查询结果导出到文件

2.将外部文件数据导入到 `MySQL` 中

写入数据文件至 `MySQL`

数据查询：基本子句

> 准备工作: 创建 `purchase` 数据库，然后利用 `purchase.sql` 还原数据。

```
use purchase;
```

> `SQL` 中的子句： `select, from, where, group by, having, join, limit`

一、 `SELECT` 子句

1. 指定字段查询

> 相当于关系代数中的投影运算

- 列出特定字段名对应数据（即进行字段的筛选）的语法格式如下:

```
SELECT 字段名1, 字段名2, ..., 字段名n
FROM 表名;
```

示例1: 通过指定字段查询 `product` 表的所有数据

```
SELECT product_id, product_name, product_code, product_place, product_date, price,
unit, detail, subsort_id, sort_id
FROM product;
```

示例2：查询 `product` 表中 `product_id, product_name, price` 字段对应的数据

```
SELECT product_id, product_name, price
FROM product;
```

- 如果需要查询表中的所有字段，则可采用 `*` 匹配符替代。

```
SELECT *
FROM 表名;
```

示例3：通过 `*` 查询 `product` 表的所有数据

```
SELECT *
FROM product;
```

> 注意: 查询结果字段排序为 `DESC 表名` 中的字段排序。

- `LIMIT` 子句可限定查询返回的行数

当不需要显示所有查询结果时，可以通过 `LIMIT` 限定查询的行数, 语法格式为：

```
SELECT *|{字段名1,字段名2,……}
FROM 表名
LIMIT m [,n];
```

> 参数m为偏移量（即第一个返回的记录对应的序号），n为返回的个数

示例4: 查询前10个 `product` 表中的商品记录

```
SELECT *
FROM product
LIMIT 10;
```

二、 `WHERE` 子句

> 相当于关系代数中的选择运算

1. 带关系运算符的查询

语法格式:

```
SELECT 字段名1, 字段名2, ...
FROM 表名
WHERE 条件表达式
LIMIT [m,] n;
```

- 常见的关系运算符号： `=、<>、!=、<、<=、>、>=`

示例5：查询 `product` 表中 `product_place` 为 `'天津'` 的产品信息

```
SELECT *
FROM product
WHERE product_place = '天津';
```

示例6：使用 `SELECT` 语句查询 `product_name` 为 `“京瓷KM-4030复印机”` 的商品价格

```sql
SELECT price
FROM product
WHERE product_name = '京瓷KM-4030复印机';
```

示例7：查询 `product` 表中 `price` 大于等于1000的商品代码和名称

```sql
SELECT product_id, product_name
FROM product
WHERE price >= 1000;
```

- 需要注意，在 `SQL` 中， `a <= x <= b` 并不等价于 `x >= a and x <= b` ，请对比以下两个查询结果:

```sql
SELECT COUNT(product_id)
FROM product
WHERE 100 <= price <= 500;

SELECT COUNT(product_id)
FROM product
WHERE price >= 100 AND price <= 500;
```

> - `count(*)` 可用于统计返回满足条件的结果行数；
> - `100 <= price <= 500` 返回表中的记录数, 先运算 `100 <= price` ，再将运算结果(`true` 或 `false`)与 `500` 进行比较，显然对于每一行有 `price` 值且大于等于0的行都成立.

- 字符串与数值的大小比较

> 试分别运行以下两个查询：
>
> ```sql
> SELECT * FROM Product WHERE Product_ID >= 1 AND Product_ID <= 200;
> SELECT * FROM Product WHERE Product_ID >= '1' AND Product_ID <= '200';
> ```
>
> 理论上而言，字符串类型值小于所有的数值。
>
> ```sql
> SELECT 1 > 'afe12';
> ```
>
> 然而，需要注意SQL中的比较运算会自动转换类型，如果比较运算符左右为数值和能转换为数值的字符串，则现先将字符串转为数值后比较大小；如果其中的字符串不能转换为数值，则数值一方大于字符串。如果两方都为字符串数值，也按字符串比较规则确定大小。
>
> ```sql
> SELECT 1 = '1', 2 > 'a', 2 > '12', 2 > '12ABC', '2' > '12';
> ```

2. 带 `IN` 关键字的查询: 限定值的可选集

语法格式

```
SELECT *|字段名1,字段名2,……
FROM 表名
WHERE 字段名 [NOT] IN (元素1,元素2,……)
```

示例8：查找 `product` 表中产地为 `'天津', '北京', '日本'` 的商品的全部信息

```
SELECT *
FROM product
WHERE product_place IN ('天津', '北京', '日本');
```

- 查找 `product` 表中产地不为 `'天津', '北京', '日本'` 的商品的全部记录

```
SELECT *
FROM product
WHERE product_place NOT IN ('天津', '北京', '日本');
```

- `IN` 后的可选择集合可为子查询 `SELECT` 的结果集
  例如：

```
-- 查找product表sort_id属于sort表中sort_id的行
SELECT *
FROM product
WHERE sort_id IN (SELECT sort_id FROM sort);
```

3. 带 `BETWEEN ... AND...` 关键字的查询：限定值的范围

语法格式

```
SELECT *|{字段名1,字段名2,……}
FROM 表名
WHERE 字段名 [NOT] BETWEEN 值1 AND 值2;
```

示例9：查询 `prodct` 表中 `Price` 值在200和500之间的商品信息

```
SELECT *
FROM product
WHERE price BETWEEN 200 AND 500;

-- 或者
SELECT *
FROM product
WHERE price >= 200 AND price <= 500;
```

示例10：查询 `product` 表中 `Price` 值不在200和500之间的商品信息

```sql
SELECT *
FROM product
WHERE price NOT BETWEEN 200 AND 500;

-- 或者
SELECT *
FROM product
WHERE price < 200 OR price > 500;
```

4. `NULL` 判断：查询某些列值是否为空

语法结构：

```sql
SELECT *|字段名1,字段名2,……
FROM 表名
WHERE 字段名 IS [NOT] NULL;
```

示例11：查询 `product` 表中 `product_Place` 为空值的商品名称和价格

```sql
SELECT product_name, price
FROM product
WHERE product_place IS NULL;
```

示例12：查询 `product` 表中 `product_place` 不为空值的记录

```sql
SELECT product_name, price
FROM product
WHERE product_place IS NOT NULL;
```

5. 字段前带 `DISTINCT` 关键字的查询：剔除查询结果中的重复行

语法格式

```sql
SELECT DISTINCT 字段名
FROM 表名
[WHERE 条件表达式];
```

示例13：查询 `product` 表中 `product_place` 字段的值，查询记录不能重复

```sql
SELECT DISTINCT product_place
FROM product;
```

- `DISTINCT` 关键字作用于多个字段：向量比较，只有 `DISTINCT` 关键字后指定的所有字段值都相同，才会被认作是重复记录。

示例14：找出 `product` 表中不同的根类别和子类别的组合

```sql
SELECT DISTINCT sort_id, subsort_id
FROM product;
```

三、匹配字符串

1. 带 `LIKE` 关键字的字符串简单匹配

- 语法格式

```
SELECT *|{字段名1,字段名2,……}
FROM 表名
WHERE 字段名 [NOT] LIKE '匹配字符串';
```

> LIKE语法格式中的“匹配字符串”指定用来匹配的字符串，其值可以是一个普通字符串，也可以是包含百分号(%)和下划线(_)等通配字符串:
> (1) `%` : 可以匹配任意长度的任意字符串，包括空字符;
> (2) `_` : 下划线通配符只匹配1个任意字符，如果要匹配多个任意字符，需要使用多个下划线通配符

示例15：查找 `product` 表中商品名称含有 `复印机` 的商品的 `product_name, price, product_place`

```
SELECT product_name, price, product_place
FROM product
WHERE product_name LIKE '%复印机%';

SELECT product_id, product_name, price, product_place
FROM product
WHERE product_name LIKE '______复印机';
```

> 注意：百分号和下划线是通配符，它们在通配字符串中有特殊含义，因此，如果要匹配字符串中的百分号和下划线，就需要在通配字符串中使用右斜线(\ )对百分号和下划线进行转义。例如， `\%` 匹配百分号字面值， `\_` 匹配下划线字面值。

示例16: 找出 `product` 表商品名称含有 `_` 或 `%` 的记录

```
-- 插入示例行数据
INSERT INTO product(product_id, product_name)
VALUES ('33', '理光_复印机'),
  ('44', '理光%复印机');

SELECT *
FROM product
WHERE product_name LIKE '%\%%' OR product_name LIKE '%\_%';
```

*2. 利用正则表达式匹配字符串: `regexp` 或 `rlike`

模式	描述
`^`	匹配输入字符串的开始位置。
`$`	匹配输入字符串的结束位置。
`.`	匹配除 `"\n"` 之外的任何单个字符。要匹配包括 `'\n'` 在内的任何字符，请使用象 `'[.\n]'` 的模式。
`[...]`	字符集合。匹配所包含的任意一个字符。例如， `'[abc]'` 可以匹配 `"plain"` 中的 `'a'` 。
`[^...]`	负值字符集合。匹配未包含的任意字符。例如， `'[^abc]'` 可以匹配 `"plain"` 中的 `'p'` 。
`p1|p2|p3`	匹配 `p1` 或 `p2` 或 `p3` 。例如， `'z|food'` 能匹配 `"z"` 或 `"food"` 。 `'(z|f)ood'` 则匹配 `"zood"` 或 `"food"` 。
`*`	匹配前面的子表达式零次或多次。例如， `zo*` 能匹配 `"z"` 以及 `"zoo"` 。 `*` 等价于 `{0,}` 。
`+`	匹配前面的子表达式一次或多次。例如， `'zo+'` 能匹配 `"zo"` 以及 `"zoo"` ，但不能匹配 `"z"` 。 `+` 等价于 `{1,}` 。
`{n}`	`n` 是一个非负整数。匹配确定的 `n` 次。例如， `'o{2}'` 不能匹配 `"Bob"` 中的 `'o'` ，但是能匹配 `"food"` 中的两个 `o` 。
`{n,m}`	`m` 和 `n` 均为非负整数，其中 `n <= m` 。最少匹配 `n` 次且最多匹配 `m` 次。

示例17: 查找 `product` 表中 `product_name` 末尾字符串为 `'复印机'` 的 `product_name, price, product_place`

```
1   SELECT product_name, price, product_place
2   FROM product
3   WHERE product_name REGEXP '复印机$';
```

示例18: 查询 `product` 表中 `product_place` 为 `'江苏'` 的 `product_name, price` 和 `product_place`

```
1   SELECT *
2   FROM product
3   WHERE product_place REGEXP '^江苏';
```

四、复合查询

1. 带 `AND` 关键字的多条件查询: 连接两个或者多个查询条件

`AND` 或 `&&`

- 语法格式

```
SELECT *|{字段名1,字段名2,……}
FROM 表名
WHERE 条件表达式1 […… AND 条件表达式n];
```

示例19：找出 `product` 表中 `product_name` 含 `复印机` 且 `product_place` 在 `天津` 的记录

```
SELECT *
FROM product
WHERE product_place = '天津' AND product_name LIKE '%复印机%';

-- 或则
SELECT *
FROM product
WHERE product_place = '天津' AND product_name REGEXP '复印机';
```

2. 带 `OR` 关键字的多条件查询: 匹配其中的一个条件

`OR` 或 `||`

- 语法格式

```
SELECT *|{字段名1,字段名2,……}
FROM 表名
WHERE 条件表达式1 […… OR 条件表达式n];
```

示例20：找出 `product` 中 `product_name` 含 `'复印机'` 或 `'过胶机'` 的商品记录

- 错误写法

```
SELECT *
FROM product
WHERE product_name LIKE '%复印机%';

SELECT *
FROM product
WHERE '%过胶机%';

SELECT count(*)
FROM product
WHERE product_name LIKE '%复印机%' OR '%过胶机%';
```

- 正确写法

```sql
SELECT *
FROM product
WHERE product_name LIKE '%复印机%' OR product_name LIKE '%过胶机%';

-- 或者
SELECT *
FROM product
WHERE (product_name LIKE '%复印机%') OR (product_name LIKE '%过胶机%');

-- 或者
SELECT *
FROM product
WHERE product_name REGEXP '复印机|过胶机';
```

```sql
-- 找出product中商品类别为复印机的商品记录
SELECT a.product_id, a.product_name, a.price, a.subsort_id, a.sort_id
FROM product a, subsort b
WHERE a.subsort_id = b.subsort_id AND b.subsort_name = '复印机';
```

- `OR` 和 `AND` 关键字一起使用的情况: `AND` 的优先级高于 `OR` ，因此当两者在一起使用时，应该先运算 `AND` 两边的条件表达式，再运算 `OR` 两边的条件表达式。

示例21: 找出 `product_name` 含 `'复印机'` 和 `'过胶机'` ，且 `product_plance` 为 `天津` 的记录

```sql
SELECT *
FROM product
WHERE product_name LIKE '%复印机%' AND product_place = '天津'
  OR product_name LIKE '%过胶机%' AND product_place = '天津';
-- 等价于
SELECT *
FROM product
WHERE (product_name LIKE '%复印机%' OR product_name LIKE '%过胶机%') AND
product_place = '天津';

-- 等级于
SELECT *
FROM product
WHERE product_name REGEXP '复印机|过胶机' AND product_place = '天津';

-- 不等价于(找出产地为天津且商品名称含过胶机，以及所有商品名称含复印机的商品记录)
SELECT *
FROM product
WHERE product_name LIKE '%复印机%' OR product_name LIKE '%过胶机%' AND
product_place = '天津';

```

练习

在 `product` 表中完成以下查询：

- 找出零售价在 `500` 元到 `1000` 元的商品记录，显示 `Product_ID, Product_Name, Price, Product_Place, SubSort_ID` 和 `Sort_ID` 。
- 找出 `product_name` 中含有 `'理光'` 和 `'墨粉'` 的商品记录。
- 找出 `product_place` 不为空且 `product_name` 包含 `'计算机'` 的商品记录，显示前 `10` 条记录。
- 找出 `Product_Place` 的不同值。
- 找出价格在 `1000` 元以下的书柜和价格在 `1000` 元到 `2000` 元之间的保险柜商品记录。

附: `MySQL` 数据的写入写出

1. 将查询结论写入外部文件

> 准备工作：找到 `ProgramData` 中的 `my.ini` 文件（例如： `'C:\ProgramData\MySQL\MySQL Server 5.7\my.ini'` ），将其中的变量 `secure-file-priv` 的值设置为空(`secure-file-priv=""`)，设置好之后重启 `mysql` 服务，即可将数据存储到任意文件夹下或从任意文件夹中导入数据。

```
SELECT 字段1, 字段2, ... | *
FROM 表名
WHERE 条件表达式
INTO OUTFILE "文件路径+文件名+扩展名" CHARACTER SET 字符集
FILEDS TERMINATED BY 字段值分割符
OPTIONALLY ENCLOSED BY 字符串标识符
LINES TERMINATED BY 换行符;
```

将 `product` 表的查询结果导出到文件

```
SELECT product_id, product_name, product_place, price, sort_id
FROM product
WHERE price > 100
INTO OUTFILE 'e:/1.xls' CHARACTER SET gbk
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

2.将外部文件数据导入到 `MySQL` 中

> 准备工作：csv文本文件，例如 `e:/1.csv`

- 语法格式

```
LOAD DATA INFILE "文件路径+文件名+扩展名"
INTO TABLE 表名 CHARACTER SET 字符集
FILEDS TERMINATED BY 字段值分割符
OPTIONALLY ENCLOSED BY 字符串标识符
LINES TERMINATED BY 换行符;
```

写入数据文件至 MySQL

```
CREATE TABLE product_bak AS SELECT * FROM product WHERE 1=2;
SHOW CREATE TABLE product_bak;

LOAD DATA INFILE "e:/1.CSV"
INTO TABLE product_bak CHARACTER SET gbk
FILEDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```