单表查询语句

使用 AppServ 的“MySQL Command Line Client”命令启动命令行窗口，输入 root 的登录密码，单元一进行系统安装的时候，设定 root 用户的密码是“12345678”。

启动“MySQL Command Line Client”的具体步骤如下：用 Windows 的“开始”菜单进入。安装好 AppServ 后，在 Windows 的“开始”菜单里，第一步点击“MySQL Command Line Client”选项后，系统弹出对话框，第二步输入“root”用户的登录密码，回车，就进入 MySQL 命令行模式。

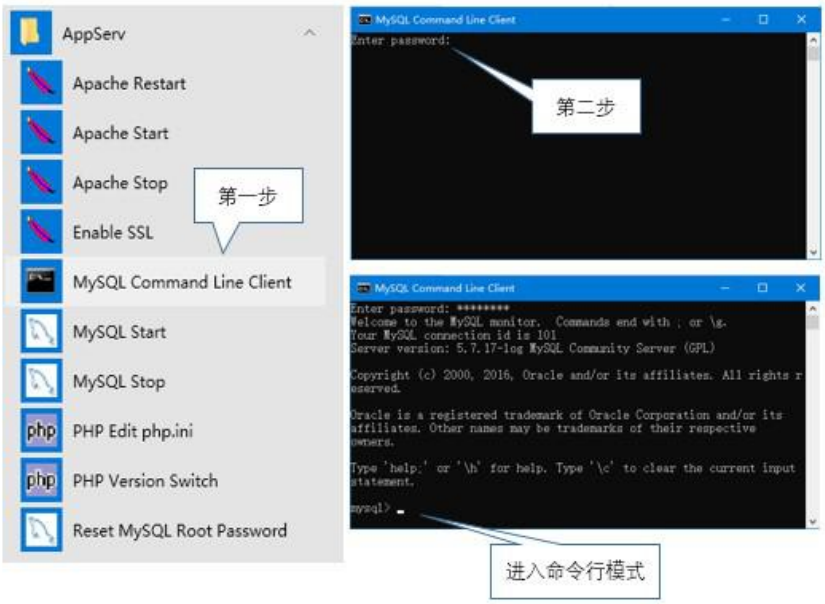

图 2-2 从 Windows 开始菜单进入 MySQL 命令行方式

进入之后，需要用“USE”语句将要使用的数据库切换到当前要操作的数据库“Lib”。

```
USE Lib;
```

系统会给出“database changed”的提示语句，这样表示已经切换到示例数据库“Lib”。

SELECT 语句可以按需提取表中的数据，功能强大，下面将通过若干实例、层层递进，学习有关 SELECT 语句的使用。

2.3.1 列出所有列

Lib 数据库中有五个表，其中 book 表中保存了当前图书馆的所有的书。下面的例子可以实现查看 book 表中所有的数据。

【例 2-1】显示 book 表中所有的数据。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT *
FROM book;
```

上述命令中，“SELECT…FROM…;”是 SELECT 语句的基本格式，可以理解为“从（FROM）……表中挑选（SELECT）……出来”。这里用“*”表示所有的列，FROM 后面引导是从哪个表提取数据，这里是 book 表；语句结束用半角英文的分号“;”表示，否则数据库系统不会执行该语句。请注意，所有 SQL 命令中的标点符号都是半角英文符号，在编辑 SQL 命令的时候，请注意输入法的切换。

当然，如果想显示其它几个表，就用表名放在 FROM 后面。例如：

```
SELECT *
FROM bookclass;
SELECT *
FROM borrow;
SELECT *
FROM reader;
SELECT *
FROM school;
```

可以看出，上述的语句具有一定的格式。SELECT 语句的格式可以简单的总结为下面的形式：

```
SELECT [ALL | DISTINCT]   输出列表达式, ...
[FROM  <表名 1> [ , <表名 2>] …]                    /*FROM 子句*/
[WHERE 条件]                                       /*WHERE 子句*/
[GROUP BY {列名 | 表达式 | 列编号}[ASC | DESC]      /*GROUP BY 子句*/
[HAVING 条件]                                      /*HAVING 子句*/
[ORDER BY {列名 | 表达式 | 列编号} DESC] , ...]    /*ORDER BY 子句*/
[LIMIT {[偏移量,] 行数|行数 OFFSET 偏移量}]        /*LIMIT 子句*/
```

SELECT…是语句必须的，方括号里的子句是根据应用需要。

2.3.2 列出指定列

在实际应用当中，不一定需要把表中所有列的数据都提取出来，往往都是按照需要，把特定的列所对应的数据提取出来。例如，仅提取 book 表中的书名。

【例 2-2】显示 book 表中的所有书籍的书名。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT bookName
FROM book;
```

此时，SELECT 后面跟着的是 book 表中表示书名的列名 bookName。

同样的，当需要显示读者的姓名的时候，可以用下面的语句：

```
SELECT readerName
FROM reader;
```

此时，SELECT 后面跟着的是 reader 表中表示读者姓名的列名 readerName。

2.3.3 WHERE 查询条件

上述的【例 2-1】【例 2-2】显示的都是指定列的所有的数据。而实际情况往往是需要聚焦到某个或某些数据，这样就需要一些条件进行限制。在 SELECT 语句中，条件可以用 WHERE 子句引导。

【例 2-3】从 book 表中，选出价格为 38 元的图书信息。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT *
FROM book
WHERE price=38;
```

上述语句将从 book 表中选出符合条件的数据显示出来。

WHERE 子句必须紧跟 FROM 子句之后，在 WHERE 子句中，使用一个条件从 FROM 子句的中间结果中选取行。其基本格式为：

```
WHERE 列名 运算符 值
```

WHERE 子句会根据条件对 FROM 子句的中间结果中的行一行一行地进行判断，当条件为 TRUE 的时候，一行就被包含到 WHERE 子句的中间结果中。

在 SQL 中，返回逻辑值（TRUE 或 FALSE）的运算符或关键字都可称为谓词。

判定运算包括比较运算、模式匹配、范围比较、空值比较和子查询。

1、比较运算

比较运算符用于比较（除 TEXT 和 BLOB 类型外）两个表达式值，MySQL 支持的比较运算符有：=（等于）、<（小于）、<=（小于等于）、>（大于）、>=（大于等于）、<=>（相等或都等于空）、<>（不等于）、!=（不等于）。

MySQL 有一个特殊的等于运算符“<=>”，当两个表达式彼此相等或都等于空值时，它的值为 TRUE，其中有一个空值或都是非空值但不相等，这个条件就是 FALSE。

当两个表达式值均不为空值（NULL）时，除了“<=>”运算符，其他比较运算返回逻辑值 TRUE(真)或 FALSE(假);而当两个表达式值中有一个为空值或都为空值时,将返回 UNKNOWN。

通过逻辑运算符（AND、OR、XOR 和 NOT）组成更为复杂的查询条件。逻辑运算操作的结果是“1”或“0”，分别表示“TRUE”或“FALSE”。

2、模式匹配（LIKE 运算符）

LIKE 运算符用于指出一个字符串是否与指定的字符串相匹配,其运算对象可以是 char、varchar、text、datetime 等类型的数据，返回逻辑值 TRUE 或 FALSE。

使用 LIKE 进行模式匹配时，常使用特殊符号_和%，可进行模糊查询。“%”代表 0 个或多个字符，“_”代表单个字符。由于 MySQL 默认不区分大小写，要区分大小写时需要更换字符集的校对规则。

3、范围比较

用于范围比较的关键字有两个：BETWEEN 和 IN。

当要查询的条件是某个值的范围时，可以使用 BETWEEN 关键字。BETWEEN 关键字指出查询范围，格式为：

```
表达式 [ NOT ] BETWEEN 表达式 1 AND 表达式 2
```

当不使用 NOT 时，若表达式 expression 的值在表达式 expression1 与 expression2 之间（包括这两个值），则返回 TRUE，否则返回 FALSE；使用 NOT 时，返回值刚好相反。

注意：表达式 1 的值不能大于表达式 2 的值。

使用 IN 关键字可以指定一个值表，值表中列出所有可能的值，当与值表中的任一个匹配时，即返回 TRUE，否则返回 FALSE。

使用 IN 关键字指定值表的格式为：

```
表达式 IN (表达式 1 [,…n])
```

4、空值比较

当需要判定一个表达式的值是否为空值时，使用 IS NULL 关键字，格式为：

```
表达式 IS [ NOT ] NULL
```

当不使用 NOT 时，若表达式的值为空值，返回 TRUE，否则返回 FALSE；当使用 NOT 时，结果刚好相反。

2.3.4 复合查询条件

从【例 2-3】可知，可以用 WHERE 子句引导筛选条件，条件越多，数据就越聚焦。

【例 2-4】从 book 表中，选出价格为 38 元且出版社为“长江出版社”的书籍信息。在

MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT *
FROM book
WHERE price=38 AND publishName='长江出版社';
```

这样，书名为《最漫画》的信息显示出来了。

请注意，上述命令中，WHERE 后面列出了两个条件，是“且”的关系，所以用关键字 AND。另外，由于“长江出版社”是一个字符串，所有在 WHERE 条件里，要用单引号。

2.3.5 多种查询条件

【例 2-4】中，查询条件是“且”的关系，使用了 AND 连接词。当查询条件有多个，将视情况使用“OR”、“AND”、“IN”或者“NOT IN”。

尝试将【例 2-4】的 WHERE 子句后的条件，用 OR 连接，观察运行的结果。

【例 2-5】查询书籍编号为“b004”、“b007”和“b013”的信息。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT *
FROM book
WHERE bookNo='b004' OR bookNo='b007' OR bookNo='b013';
```

当前的条件是“或”。该语句还可以写成如下形式：

```
SELECT *
FROM book
WHERE bookNo IN ('b004', 'b007', 'b013');
```

WHERE 后面用 IN 把若干或条件合成一个集合的形式。

2.3.6 NULL 查询条件

NULL 在数据库表中表示某一个值是没有的。例如，book 表中，由于某书籍信息不全，没有作者名字，工作人员没办法填那本书的作者名字，等找到该作者名字之后再录入。此时就不能空着那个“单元格”，作者列可以为空，今后可以用 NULL 作为条件进行查询。

【例 2-6】查询 book 表中，没有具体作者的书籍信息，显示书籍编号和书籍名称。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT bookNo, bookName
FROM book
WHERE author IS NULL;
```

请注意，当前的条件是“author IS NULL”，而不是“author=NULL”。

2.3.7 条件中不等号的使用

查询中，如果查找某一范围的信息，可以在条件中使用不等号。不等号有“>”、“<”、“>=”、“<=”或“<>”。

【例 2-7】在书籍表 book 中，查找单价小于 30 元的书籍信息，要求显示书籍名称和作者。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT bookName, author
FROM book
WHERE price < 30;
```

2.3.8 条件中 BETWEEN 的使用

当查询条件介于某个范围之内或者之外，除了可以用不等号构成查询条件外，还可以用“BETWEEN…AND…”的形式进行。

【例 2-8】查询书籍表 book 中，单价介于 20~30 元的书籍信息，要求显示书籍名称和作者。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT bookName, author
FROM book
WHERE price BETWEEN 20 AND 30;
```

上述语句还可以写成如下形式：

```
SELECT bookName, author
FROM book
WHERE price >= 20 AND price <= 30;
```

请注意，BETWEEN AND 语句是包括边界值的。

2.3.9 消除重复行

当查询书籍表 book 的出版社信息时，如果简单的使用

```
SELECT publishName FROM book;
```

结果将出现很多重复的出版社名称。如何消除重复的信息，可以在相关的列名前使用 DISTINCT。

【例 2-9】显示书籍表 book 的出版社名称，要求消除重复的名称。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT DISTINCT publishName
FROM book;
```

2.3.10　通配符的使用

通配符是数据库技术里常用的一种技术，例如“%”表示一个或多个字符，“_”表示一个字符。

【例 2-10】显示书籍表 book 的作者姓曹的信息，要求显示书名和作者名。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT bookName, author
FROM book
WHERE author LIKE '曹%';
```

如果作者姓名第二个字是“正”的，则语句改为：

```
SELECT bookName, author
FROM book
WHERE author LIKE '_正%';
```

如果查询书籍名称里包含“_”的书名和作者名，则语句可以为：

```
SELECT bookName, author
FROM book
WHERE bookName LIKE '%#_%' ESCAPE '#';
```

这里的“#”用 ESCAPE 说明，则其后面的下划线就失去通配符的意义。

2.3.11　改变结果列名

当执行查询命令时，显示结果的表头是真是数据表的列名。如果改变它们，则需要使用“AS”或者空格的方式。

【例 2-11】更改【例 2-10】的显示结果，要求 bookName 列显示为书名，author 列显示为作者。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT bookName AS '书名', author AS '作者'
FROM book
WHERE author LIKE '曹%';
或者
SELECT bookName '书名', author '作者'
FROM book
WHERE author LIKE '曹%';
```

2.3.12　显示计算结果

数据库表中的列值和列值之间可以进行计算。例如，在书籍表 book 中，有单价和存量数，可以求得该书籍的总价值。

【例 2-12】求每一图书的总价值，显示书名、单价、存量和总价值。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT bookName AS '书名' , price AS '单价',
        number AS '存量', price*number AS '总价值'
FROM book;
```

2.3.13　利用聚合函数计算

聚合函数对一组值执行计算，并返回单个值，也被称为组函数。例如库存图书很多，想知道最贵的书、最便宜的书和平均价格是多少，可以通过聚合函数 MAX()、MIN()、AVG()和 SUM()进行查询。

【例 2-13】在书籍表 book 中，查询最贵的书是平均价格的多少倍、最便宜的书与平均价格差多少。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT MAX(price)/AVG(price), MIN(price)-AVG(price)
FROM book;
```

2.3.14　使用“GROUP BY”进行分组和计算

GROUP BY 子句后通常包含列名或表达式。MySQL 对 GROUP BY 子句进行了扩展，可以在列的后面指定 ASC（升序）或 DESC（降序）。GROUP BY 可以根据一个或多个列进行分组，也可以根据表达式进行分组，经常和聚合函数一起使用。

分类汇总是数据库技术中经常要用到的操作，这种操作往往借助与聚合函数，需要经常与 SELECT 语句的 GROUP BY 子句的 HAVING 一同使用。

GROUP BY 是让数据分组，例如：

```
SELECT publishName
FROM book
GROUP BY publishName;
```

等价于

```
SELECT DISTINCT publishName
FROM book;
```

【例 2-14】统计一下书籍表 book 中，各个出版社的图书总数。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT publishName,COUNT(publishName)
FROM book
GROUP BY (publishName);
```

COUNT()也是聚合函数，表示计数。

2.3.15 使用“HAVING”子句作为查询条件

使用 HAVING 子句的目的与 WHERE 子句类似，不同的是 WHERE 子句是用来在 FROM 子句之后选择行，而 HAVING 子句用来在 GROUP BY 子句后选择行。其中，HAVING 条件的定义和 WHERE 子句中的条件类似，不过 HAVING 子句中的条件可以包含聚合函数，而 WHERE 子句中则不可以。

SQL 标准要求 HAVING 必须引用 GROUP BY 子句中的列或用于聚合函数中的列。不过，MySQL 支持对此工作性质的扩展，并允许 HAVING 引用 SELECT 清单中的列和外部子查询中的列。

【例 2-15】查询出版社所有书籍平均单价大于 25 的情况，列出出版社和平均单价。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT publishName,AVG(price)
FROM book
GROUP BY publishName
HAVING AVG(price)>25;
```

2.3.16 结果按要求排序

在一条 SELECT 语句中，如果不使用 ORDER BY 子句，结果中行的顺序是不可预料的。使用 ORDER BY 子句后可以保证结果中的行按一定顺序排列。将查询结果按一定次序排序后再显示，是一种常见工作场景。

ORDER BY 子句后可以是一个列、一个表达式或一个正整数。正整数表示按结果表中该位置上的列排序。例如，使用 ORDER BY 3 表示对 SELECT 的列清单上的第 3 列进行排序。

关键字 ASC 表示升序排列，DESC 表示降序排列，系统默认为 ASC。

【例 2-16】将书籍表 book 中的书名和对应的单价显示，按照单价排列显示。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT bookName, price
FROM book
ORDER BY price;
```

如果是降序显示，则需要关键字 DESC.

上述语句就为：

```
SELECT bookName, price
FROM book
ORDER BY price DESC;
```

2.3.17　显示指定行

如果要求显示结果是前几行；或者是从第几行开始，连续显示几行。这样的场景也常见于新闻列表或者多页显示结果。

【例 2-17】显示书籍表 book 单价最贵的前 5 行数据。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT *
FROM book
ORDER BY price DESC
LIMIT 0,5;
```

如果从第 6 行开始，显示 6~10 行，则语句为：

```
SELECT *
FROM book
ORDER BY price DESC
LIMIT 5,5;
```

请注意，表中的第一行在语句中是第 0 行。

2.3.18　子查询的使用。

在实际应用时，查询的条件往往不可知，需要先用一个查询得到结果后，然后再用这个结果作为条件进行新的查询。这种嵌套逻辑称作子查询。子查询除了可以用在 SELECT 语句中，还可以用在 INSERT、UPDATE 及 DELETE 语句中。子查询通常与 IN、EXIST 谓词及比较运算符结合使用。

【例 2-18】显示书籍表 book 中，单价比所有平均值高的信息，要求显示书名和单价。在 MySQL 命令行窗口中执行如下 SQL 语句：

```
SELECT bookName, price
FROM book
WHERE price > (SELECT AVG(price) FROM book);
```

先用一个查询语句查出所有书籍的平均价格，在用这个作为条件，让主查询使用。

比较子查询也是一种常见的技术。例如，查找书籍表 book 中所有比“清华大学出版社”图书价格都高的图书基本信息。

```
SELECT bookName,price
FROM  book
WHERE price>ALL (SELECT price FROM book WHERE publishName='清华大学出版社');
```

上述的子查询用 ALL 来表示所有。

如果上述问题改成，查找书籍表 book 中不低于“清华大学出版社”图书价格的图书基本信息，则把关键字 ALL 改成 SOME。

```
SELECT bookName,price
FROM  book
WHERE price>SOME
(SELECT price FROM book WHERE publishName='清华大学出版社');
```