第3章

DML 和 DQL

本章重点：

使用 SQL 语句实现数据添加、修改、查询
查询指定学生考试成绩
查询某学期开设的课程
查询某课程最近一次考试缺考的学生名单

本章目标：

掌握 MySQL 常用的存储引擎
掌握增删查改数据库表的 SQL 语句
掌握简单子查询的用法
掌握 IN 子查询的用法

本章任务

学习本章，需要完成以下 3 个工作任务。请记录下来学习过程中所遇到的问题，可以通过自己的努力或访问 kgc.cn 解决。

任务 1：使用 SQL 语句实现数据添加、修改、查询

任务 2：查询指定学生考试成绩

任务 3：查询某学期开设的课程

任务 1　使用 SQL 语句实现数据添加、修改、查询

在前面的章节中学习了数据库的相关概念，那么针对于 MySQL 数据库，数据最终以什么样的形式保存，以及数据保存在硬盘的什么位置？基于以上两个问题，本章将详细介绍 MySQL 数据库的存储引擎。

在前面的章节中学习了 MySQL 数据库和 SQL 语句的基础知识，以及如何使用 SQL 语句中的 DDL 语句操作数据库、数据库表，本章将继续讲解 SQL 语句中的 DML 和 DQL，并使用 SQL 语句完成对数据库表数据的增加、删除、查询和修改。

在前面的章节中学习了 MySQL 客户端工具 SQLyog 的基本使用，本章中将继续使用 SQLyog 客户端对 MySQL 进行管理。

3.1.1　MySQL 的存储引擎

通过前面章节的学习我们知道 MySQL 属于数据库管理系统，其中包括数据库以及用于数据库访问管理的接口系统。数据库负责存储数据，接口系统负责管理数据库。由于不同用户对数据的容量、访问速度、数据安全性有不同的要求。为了满足不同用户的业务需求，MySQL 的数据库采用多种存储引擎进行数据存储，接下来介绍 MySQL 数据库常用的存储引擎。

存储引擎指定了表的存储类型，即如何存储和索引数据、是否支持事务等，同时存储引擎也决定了表在计算机中的存储方式。MySQL 5.5 支持的存储引擎有 InnoDB、MyISAM、Memory、MRG_MyISAM、Archive、Federated、CSV、BLACKHOLE 等九种，可以使用 SHOW ENGINES 语句查看系统所支持的引擎类型，执行结果如图 3.1 所示。

1. 常用的存储引擎

不同的存储引擎都有各自的特点，以适应不同的需求，本课程重点介绍两种常用

的存储引擎 InnoDB 和 MyISAM，为了做出选择，首先需要对比它们所提供的功能。表 3-1 中列出了在 MySQL 5.5 版本中二者的部分差异点。

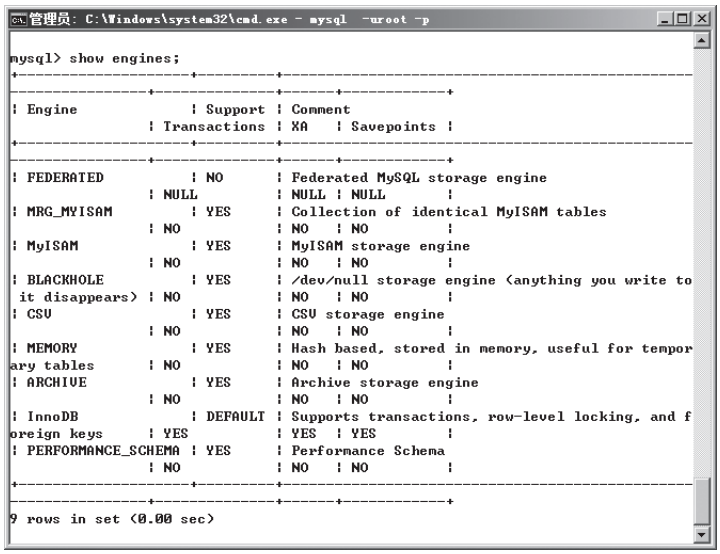

图 3.1　显示所有存储引擎信息

表 3-1　InnoDB 和 MyISAM 存储引擎比较

功能	InnoDB	MyISAM
支持事务	支持	不支持
支持全文索引	不支持	支持
外键约束	支持	不支持
表空间大小	较大	较小
数据行锁定	支持	不支持

InnoDB 和 MyISAM 各自的使用场合如下：

- MyISAM 存储引擎：该存储引擎不支持事务，也不支持外键，访问速度比较快。因此对不需要事务处理，以访问为主的应用适合使用该引擎。
- InnoDB 存储引擎：该存储引擎在事务处理上有优势（事务相关内容将在第 5 章讲解），即支持具有提交、回滚和崩溃恢复能力的事务控制，所以比 MyISAM 引擎占用更多的磁盘空间，因此需要进行频繁的更新、删除操作，同时还对事务的完整性要求比较高，需要实现并发控制，适合使用该存储引擎。

2. 操作默认存储引擎

安装 MySQL 5.5 以上版本默认存储引擎是 InnoDB，可以通过以下语句来查看当前的默认存储引擎：

SHOW VARIABLES LIKE 'storage_engine%';

其中，LIKE 后要查询的关键字为“storage_engine%”，表示查询默认存储引擎。执行结果如图 3.2 所示。

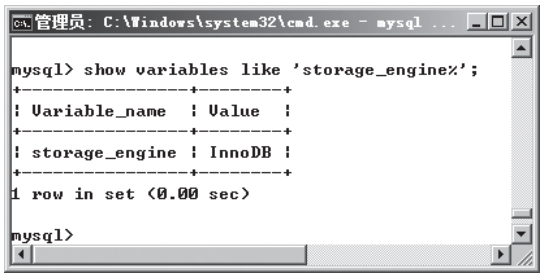

图 3.2　查询默认存储引擎

如需修改默认存储引擎，可以通过配置向导，也可以通过修改配置文件 my.ini 来实现。修改配置文件 my.ini 时，只修改如下内容：

defaulstorage-engine= InnoDB

例如，需将默认存储引擎改为 MyISAM，只需修改 defaulstorage-engine 参数值为 MyISAM 即可。

注意：

如果让修改后的参数生效，必须要重新启动 MySQL 服务。

重新启动服务后，再次查看默认存储引擎，执行结果如图 3.3 所示。

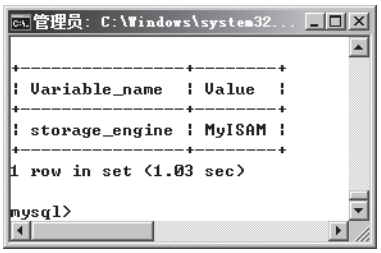

图 3.3　修改默认存储引擎结果

3. 指定表的存储引擎

数据表默认使用当前 MySQL 默认的存储引擎，有时为了达到数据表的特殊功能要求，也可重新设置表的存储类型。语法如下：

```
CREATE TABLE 表名 (
  # 省略代码
)ENGINE= 存储引擎 ;
```

例如：创建 myisam 表并设置为 MyISAM 类型，SQL 语句如下：

```
CREATE TABLE 'myisam' (
```

```
  id  INT(4)
)ENGINE=MyISAM;
```

注意：

在有些参考资料中，表的存储引擎也称为表类型。

4. MySQL 的数据文件

通过上述的课程，就会知道 MySQL 的存储引擎如何设置。那么 MySQL 的数据文件又是怎样的？它们是如何存放的？在 MySQL 中，不同的存储引擎下，数据文件有所不同，下面来创建两个不同类型的数据表。

示例 1

```
CREATE DATABASE enginedb;
USE enginedb;
/* 创建表类型为 MyISAM 的表 */
DROP TABLE IF EXISTS 'myisam' ;
CREATE TABLE 'myisam'(
  sid INT(4)
)ENGINE=MyISAM;

/* 创建表类型为 InnoDB 的表 */
DROP TABLE IF EXISTS 'innodb' ;
CREATE TABLE 'innodb'(
  sid INT(4)
)ENGINE=InnoDB;
```

以上代码，在 enginedb 数据库中创建了两个表，其中 myisam 表为 MyISAM 类型，innodb 表为 InnoDB 类型。

（1）数据文件的存储位置。

不同操作系统数据文件的默认存储位置不同，在 Windows 7 操作系统中，MySQL 数据文件的默认存储为 C:\ProgramData\MySQL\MySQL Server 5.5\data，可配置文件 my.ini 中参数 datadir 获取或修改该路径，例如：

```
datadir="C:/ProgramData/MySQL/MySQL Server 5.5/data/"
```

该目录下，每个数据库相关文件存放在以数据库命名的文件夹中，找到 enginedb 数据库文件夹，如图 3.4 所示。

（2）MyISAM 类型的表文件。

进入 enginedb 文件夹，找到类型为 MyISAM 的表 myisam 的数据文件有三个，扩展名分别为 .frm、.MYI、.MYD，如图 3.5 所示。

- .frm 文件：表结构定义文件。其主要存放表的元数据，包括表结构定义信息等。该文件与存储引擎无关，即任何存储类型的表都会有这个文件。
- MYI 文件：索引文件。其主要存放 MyISAM 类型表的索引信息，每个

MyISAM 类型的表会有一个 .MYI 文件，存放的位置与 .frm 文件相同。

- .MYD 文件：数据文件。其存放表中数据的文件。

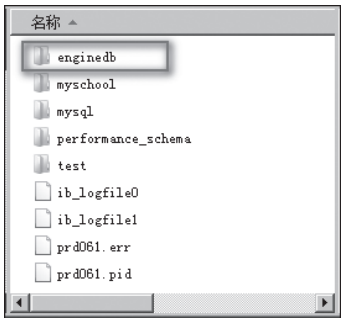

图 3.4　MySQL 数据文件目录

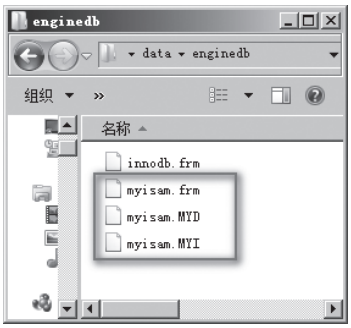

图 3.5　数据表文件

（3）InnoDB 类型的表文件。

enginedb 数据库文件目录下，InnoDB 类型的表 innodb.frm 相关文件如下：

- .frm 文件：表结构定义文件。其作用与 MyISAM 类型的 .frm 文件相同，如图 3.5 所示。
- ibdata1 文件：数据文件。其保存所有 InnoDB 类型表的数据。这个文件保存位置与 .frm 文件不同，可以通过 my.ini 文件中的参数 innodb_data_home_dir 查询或修改，例如：

  innodb_data_home_dir="D:/MySQL Datafiles/"

  打开目录“D:/MySQL Datafiles/”，结果如图 3.6 所示。

图 3.6　ibdata1 文件

3.1.2　使用 DML 插入数据

我们已经了解了如何创建表、修改表的结构和添加约束，现在需要学习如何向表中添加数据，这里介绍两种方式。

（1）在 SQLyog 中插入数据比较简单，只要右击表，在弹出的快捷菜单中打开表，就可以向表中直接输入数据行。

（2）使用 SQL 可以向表中添加新数据，也可以将现有表中的数据添加到新创建的表中。

1. 使用 INSERT 插入数据

使用数据库的之前，先要为数据表添加数据。

（1）插入单行数据。

语法如下：

```
INSERT INTO 表名 [( 字段名列表 )] VALUES ( 值列表 );
```

其中：

- 表的字段是可选的，如果省略，则依次插入所有字段。
- 多个列表和多个值之间使用逗号分隔。
- 值列表必须和字段名列表数量相同，且数据类型相符。
- 如果插入的是表中部分列的数据，字段名列表必须填写。

例如，向 student 表中插入一条记录：

```
INSERT INTO 'student'('loginPwd','studentName','gradeId','phone','birthday')
VALUES('123',' 黄小平 ',1,'13956799999','1996-5-8');
```

（2）插入多行数据。

在 MySQL 中 INSERT 语句支持一次插入多条记录，插入时可指定多个值列表，每个值列表之间用逗号分隔。

语法如下：

```
INSERT INTO 新表（字段名列表）VALUES( 值列表 1),( 值列表 2),……,( 值列表 n);
```

例如，一次向 subject 表中插入三条数据，SQL 语句如下：

```
INSERT INTO 'subject'('subjectName','classHour','gradeID')
VALUES('Logic Java',220,1),('HTML',160,1),('Java OOP',230,2);
```

注意：

在使用 INSERT 语句插入记录时，如果不包含字段名称，VALUES 后值列表中各字段的顺序必须和表定义中各字段的顺序相同，如果表结构变了（如执行了添加数据操作），则值列表也要变化，否则会出现错误。如果指定了插入的字段名，就会避免这个问题，因此，建议在插入数据时指定具体字段名。

（3）将查询结果插入到新表。

语法如下：

```
CREATE TABLE 新表 (SELECT 字段 1, 字段 2,……FROM 原表 );
```

例如，将 student 表中的 studentName、phone 字段数据保存到新表 phoneList 中，SQL 语句如下：

```
CREATE TABLE 'phoneList'(SELECT 'studentName','phone' FROM 'student');
```

以上 SQL 语句在执行该语句的同时创建新表 phoneList，无须提前创建，SQLyog 工具中执行结果如图 3.7 所示。如已存在表 phoneList，则执行该语句会报错。

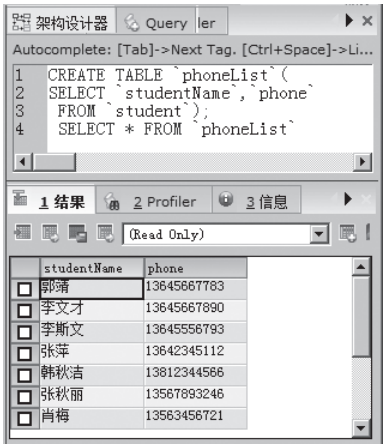

图 3.7　将查询结果插入到新表

2. 技能训练

上机练习 1：为科目表、学生表、成绩表添加数据

➢　需求说明

（1）为科目表添加数据，如表 3-2 所示。要求使用一条 INSERT 语句。

表 3-2　科目表结构

科目编号	科目名	课时数	学期编号
1	LogicJava	220	1
2	HTML	160	1
3	JavaOOP	230	2

（2）为学生表添加数据，如表 3-3 所示。

表 3-3　学生表结构

学号	密码	学生姓名	性别	年级编号	手机号码	地址	出生日期
10000	123	郭靖	男	1	13645667783	天津市河西区	1990-09-08
10001	123	李文才	男	1	13645667890	地址不详	1994-04-12
10002	123	李斯文	男	1	13645556793	河南洛阳	1993-07-23
10003	123	张萍	女	1	13642345112	地址不详	1995-06-10
10004	123	韩秋洁	女	1	13812344566	北京市海淀区	1995-07-15
10005	123	张秋丽	女	1	13567893246	北京市东城区	1994-01-17
10006	123	肖梅	女	1	13563456721	河北省石家庄市	1991-02-17
10007	123	秦洋	男	1	13056434411	上海市卢湾区	1992-04-18
10008	123	何晴晴	女	1	13053445221	广州市天河区	1997-07-23

续表

学号	密码	学生姓名	性别	年级编号	手机号码	地址	出生日期
20000	123	王宝宝	男	2	15076552323	地址不详	1996-06-05
20010	123	何小华	女	2	13318877954	地址不详	1995-09-10
30011	123	陈志强	女	3	13689965430	地址不详	1994-09-27
30012	123	李露露	女	3	13685678854	地址不详	1992-09-27

（3）使用提供的 SQL 脚本为成绩表添加数据。

3.1.3　使用 DML 更新数据

数据更新是经常发生的事情，使用 SQL 可以进行数据更新。使用 SQL 更新表中某行的语法格式如下：

```
UPDATE 表名 SET 列名 = 更新值 [WHERE 更新条件 ]
```

其中：

（1）SET 后面可以紧随多个“列名 = 更新值”，修改多个数据列的值，不限一个，使用逗号分隔。

（2）WHERE 子句是可选的，用来限制更新数据的条件。若不限制，则整个表的所有数据行将被更新。

需要注意的是，使用 UPDATE 语句，可能更新一行数据，也可能更新多行数据，还可能不会更新任何数据。

例如，在学生信息表中，要把所有学生的性别都改为 0（女生）。

```
UPDATE student SET sex = ' 女 '
```

对于地址为“北京女子职业技术学校刺绣班”的学生，若这个班级改为家政班了，则需要按照条件进行更新。

```
UPDATE student
SET address =' 北京女子职业技术学校家政班 '
WHERE address = ' 北京女子职业技术学校刺绣班 '
```

前面已经提到，在 SQL 表达式中，可以使用列名和数值。如果学生在考试时有一道题目的标准答案错了，导致评分失误，事后需要在成绩表中更新成绩，所有低于或等于 95 分的成绩都在原来的基础上加 5 分，更新的 SQL 语句如下：

```
UPDATE result
SET studentResult = studentResult + 5
WHERE studentResult <= 95
```

注意：

在更新数据的时候，一般都有条件限制，别忘了书写 WHERE 条件语句，否则将更新表中所有行的数据，这就可能导致有效数据的丢失。

3.1.4 使用 DML 删除数据

删除数据行也是经常会用到的操作，使用 SQL 语句来操作相对比较简单。

1. 使用 DELETE 删除数据

使用 SQL 删除表中的数据，语法格式如下：

```
DELETE [FROM] 表名 [WHERE < 删除条件 >]
```

在学生信息表中删除姓名为“王宝宝”的数据的 SQL 语句如下：

```
DELETE FROM student WHERE studentName =' 王宝宝 '
```

还有一种情况，如果要删除的行的主键值被其他表引用，例如，分数表中的 studentID 引用了学生信息表中的 studentNo 列，那么删除被引用的行时：

```
DELETE FROM student WHERE studentNo = 22
```

MySQL 将报告与约束冲突的错误信息。

注意：

DELETE 语句删除的是整条记录，不会只删除单个列，所以在 DELETE 后不能出现列名，例如，以下语句：

```
DELETE address FROM student;
```

MySQL 将报告错误信息。

2. 使用 TRUNCATE TABLE 删除数据

TRUNCATE TABLE 用来删除表中的所有行，功能上类似于没有 WHERE 子句的 DELETE 语句。

例如，要删除学生信息表中的所有记录行，可以使用以下语句：

```
TRUNCATE TABLE student
```

但 TRUNCATE TABLE 比 DELETE 执行速度快，使用的系统资源和事务日志资源更少，并且删除数据后表的标识列会重新开始编号。

注意：

TRUNCATE TABLE 删除表中的所有行，但是表的结构、列、约束、索引等不会被改动。TRUNCATE TABLE 不能用于有外键约束引用的表，这种情况下，需要使用 DELETE 语句。

实际工作中，不建议使用 TRUNCATE TABLE 语句，因为使用它删除的数据不能恢复还原。

3. 技能训练

上机练习 2：修改学生表、科目表数据

- 需求说明

（1）将学生表中学号为 20000 的学生的地址修改为“西直门东大街 215 号”，密码改为 000。

（2）将科目表中课时数大于 200 且学期编号为 1 的科目的课时减少 10 课时。

（3）将所有年级编号为 1 的学生姓名、性别、出生日期、手机号码信息保存到新表 student_grade1 中。

3.1.5　DQL 语句

1. 使用 SELECT 语句进行查询

查询使用 SELECT 语句，最简单的查询语句的格式如下：

```
SELECT < 列名 | 表达式 | 函数 | 常量 >
FROM < 表名 >
[WHERE < 查询条件表达式 >]
[ORDER BY < 排序的列名 >[ASC 或 DESC]]
```

其中，WHERE 条件是可选的，若不限制，则查询返回所有行的数据项。ORDER BY 是用来排序的，后续内容将会详细介绍。

注意：

在查询中还可以使用很多其他的关键字，或者实现其他特殊的要求。有关 SELECT 语句的详细语法请参考 MySQL 文档和教程。

注意：

查询语句可以分为多个子句部分，例如，上面的查询语法可以划分为 SELECT…FROM…WHERE…ORDER BY 四个子句，对于复杂的 SQL 语句，可以将每个子句单独写成一行，以方便调试和查找错误。

（1）查询所有的数据行和列。

把表中的所有行和列都列举出来比较简单，这时候可以使用“*”表示所有的列：

```
SELECT * FROM student
```

（2）查询部分行和列。

查询部分列需要列举不同的列名，而查询部分行需要使用 WHERE 子句进行条件限制，例如：

```
SELECT studentNo,studentName,address FROM student WHERE address = ' 河南新乡 '
```

以上的查询语句，将只查询地址为“河南新乡”的学生，并且只显示编号、姓名和地址列。同理，以下语句用来查询地址不是“河南新乡”的学生信息。

```
SELECT studentNo,studentName,address FROM student WHERE address <> ' 河南新乡 '
```

（3）在查询中使用列的别名。

AS 子句可以用来改变结果集中列的名称，也可以为组合或者计算出的列指定名称，

还有一种情况是让标题列的信息更易懂，例如，把 studentNo 列名查询后显示为“学生编号”。

在 SQL 中重新命名列名可以使用 AS 子句，例如：

```
SELECT studentNo AS 学生编号 ,studentName AS 学生姓名 ,address AS 学生地址
FROM student
WHERE address <> ' 河南新乡 '
```

还有一种情况是使用计算、合并得到新列的命名。例如，假设在某数据库的雇员表 employee 中存在 firstName 列和 lastName 列，现在需要将这两列合并成一个叫作“姓名”的列，可以使用以下查询语句：

```
SELECT firstName+'.'+lastName AS 姓名 FROM employee
```

（4）查询空值。

在 SQL 语句中采用“IS NULL”或者“IS NOT NULL”来判断是否为空，因此，如果要查询学生信息表中没有填写 Email 信息的学生，可以使用以下查询语句：

```
SELECT studentName FROM student WHERE email IS NULL
```

（5）在查询中使用常量列。

有时候，需要将一些常量的默认信息添加到查询输出中，以方便统计或计算。例如，查询学生信息的时候，学校名称统一都是“北京新兴桥”，查询语句如下：

```
SELECT studentName AS 姓名 ,address AS 地址 ,' 北京新兴桥 ' AS 学校名称 FROM student
```

查询输出多了一列“学校名称”，该列的所有数据都是“北京新兴桥”。

2. 常用函数

SQL 语言中的函数将一些常用的处理数据的操作封装起来，这样就大大简化了程序员的工作，提高了开发效率。因此，除了会使用 SQL 语句之外，还需要掌握一些常用函数。以下分类列出了 MySQL 中的常用函数。

（1）聚合函数。

MySQL 中的聚合函数用来对已有数据进行汇总，如求和、平均值、最大值、最小值等。MySQL 中常用聚合函数如表 3-4 所示。

表 3-4　常用聚合函数

函数名	作用
AVG()	返回某字段的平均值
COUNT()	返回某字段的行数
MAX()	返回某字段的最大值
MIN()	返回某字段的最小值
SUM()	返回某字段的和

很多需求中我们需要计算学员的总成绩，在这里就可以使用 SUM() 函数。SQL 语句如下：

```
SELECT SUM (studentResult) FROM result ;
```

同上，如果我们要计算学员的平均成绩 , 在这里可以使用 AVG() 函数。

```
SELECT AVG (studentResult) FROM result ;
```

（2）字符串函数。

字符串函数中常用函数之一，用来对字符串进行各类处理，MySQL 中使用频率较高的字符串函数如表 3-5 所示。

表 3-5　常用字符串函数

函数名	作用	举例
CONCAT(str1,str1,…, strn)	连接字符串 str1、str2、……、strn 为一个完整字符串	SELECT CONCAT('My','S','QL'); 返回：MySQL
INSERT(str,pos,len, newstr)	将字符串 str 从 pos 位置开始，len 个字符长的子串替换为字符串 newstr	SELECT INSERT(' 这是 MySQL 数据库 ',3,10,'MySQL'); 返回：这是 MySQL 数据库
LOWER(str)	将字符串 str 中所有字符变为小写	SELECT LOWER('MySQL'); 返回：mysql
UPPER(str)	将字符串 str 中所有字符变为大写	SELECT UPPER('MySQL'); 返回：MYSQL
SUBSTRING (str,num,len)	返回字符串 str 的第 num 个位置开始长度为 len 的子字符串	SELECT SUBSTRING('JavaMySQLOracle',5,5); 返回：MySQL

如果需求中规定要对学员姓名的首字母进行大写，采用 UPPER(str) 函数的 SQL 语句如下：

```
SELECT UPPER (studentName) FROM student ;
```

（3）时间日期函数。

除了聚合函数和日期函数之外，日期函数也是一类常用函数，表 3-6 列出了常用的日期函数。

表 3-6　常用日期函数

函数名	作用	举例（部分结果与当前日期有关）
CURDATE()	获取当前日期	SELECT CURDATE(); 返回：2016-08-08
CURTIME()	获取当前时间	SELECT CURTIME(); 返回：19:19:26
NOW()	获取当前日期和时间	SELECT NOW(); 返回：2016-08-08 19:19:26
WEEK(date)	返回日期 date 为一年中的第几周	SELECT WEEK(NOW()); 返回：26
续 YEAR(date)	返回日期 date 的年份	SELECT YEAR(NOW()); 返回：2016

续表

函数名	作用	举例（部分结果与当前日期有关）
HOUR(time)	返回时间 time 的小时值	SELECT HOUR(NOW()); 返回：9
MINUTE(time)	返回时间 time 的分钟值	SELECT MINUTE(NOW()); 返回：43
DATEDIFF(date1,date2)	返回日期参数 date1 和 date2 之间相隔的天数	SELECT DATEDIFF(NOW(),'2008-8-8'); 返回：2881
ADDDATE(date,n)	计算日期参数 date 加上 n 天后的日期	SELECT ADDDATE(NOW(),5); 返回：2016-09-02 09:37:07

如果需求中规定我们要对学员的出生年份进行统计，采用 YEAR (date) 函数的 SQL 语句如下：

```
SELECT YEAR(birthday) FROM student ;
```

（4）数学函数。

在使用 SQL 语句进行数据操作时，有时也会需要进行数值运算，MySQL 中支持的常用数学函数如表 3-7 所示。

表 3-7　常用数学函数

函数名	作用	举例
CEIL(x)	返回大于或等于数值 x 的最小整数	SELECT CEIL(2.3) 返回：3
FLOOR(x)	返回小于或等于数值 x 的最大整数	SELECT FLOOR(2.3) 返回：2
RAND()	返回 0~1 间的随机数	SELECT RAND() 返回：0.5525468583708134

如果需求中规定我们要对学员的分数继续取整，采用 CEIL (x) 函数的 SQL 语句如下：

```
SELECT CEIL (studentResult) FROM result ;
```

3. 技能训练

上机练习 3：查询学生相关基本信息

➢ 需求说明

（1）查询所在学期 ID 为 1 的全部学生信息。

（2）查询所在学期 ID 为 2 的全部学生的姓名和电话。

（3）查询所在学期 ID 为 1 的全部女同学的信息。

（4）查询课时超过 60 的科目信息。

（5）保存 SQL 为“查询学生相关基本信息 .sql”文件。

上机练习 4：查询学生相关复杂信息

➢　训练要点

使用 SELECT 语句查询数据。

➢　需求说明

（1）查询所在学期 ID 为 1 的科目名称。

（2）查询所在学期 ID 为 2 的所有男同学的姓名和住址。

（3）查询无电子邮件的学生姓名和年级信息。

（4）查询所在学期 ID 为 2 的学生中所有在 1990 年后出生的学生姓名。

（5）查询参加了日期为 2013 年 2 月 15 日的“HTML 和 CSS 网页技术”科目考试的学生成绩信息。

（6）查询参加 HTML 科目考试的学员的总成绩。

（7）查询参加 CSS 网页技术科目考试的学员的平均成绩。

➢　难点分析

注意各个表之间的关系。例如，通过查看所在学期 ID 为 2 的年级编号在科目表中查询对应科目。

➢　实现思路及关键代码

（1）在 SQLyog 中，新建一个查询窗口，选择数据库 MySchool 或在查询窗口中输入语句“USE MySchool”。

（2）查询学期 ID 为 1 的学期所开设的科目名称，参考如下的 SQL 语句：

```
SELECT sbjectName FROM sbject WHERE gadeId=1
```

注意这里的 GradeId 对应数据表 Grade 中 U2 的 GradeId。

（3）相同思路完成需求 2、需求 4、需求 5。

（4）需求 3 参考如下的 SQL 语句条件。

```
SELECT studentName,gadeId FROM student WHERE email IS NULL
```

（5）保存 SQL 为“查询学生相关复杂信息 .sql”文件。

4．ORDER BY 子句

如果需要按照一定顺序排列查询语句选中的行，则需要使用 ORDER BY 子句，并且排序可以是升序（ASC）或者降序（DESC）。如果不指定 ASC 或者 DESC，结果集按默认 ASC 升序排序。

上面讲述过的 SQL 语句，都可以在其后面再加上 ORDER BY 来进行排序。

例如，查询学生成绩的时候，如果把所有成绩都降低 10% 后加 5 分，再查询及格成绩并按照成绩高低来进行排列，SQL 语句如下：

```
SELECT studentID AS 学生编号 ,(studentResult*0.9+5) AS 综合成绩
FROM result
WHERE (studentResult*0.9+5)>60
ORDER BY studentResult
```

还可以按照多个列进行排序。例如，要在学生成绩排序的基础上，再按照课程 ID

进行排序的语句如下：

```
SELECT studentID AS 学生编号 , courseID AS 课程 ID, studentResult AS 成绩
FROM result
WHERE studentResult>60
ORDER BY studentResult,courseID
```

注意：

如果成绩按升序，课程编号按降序，该如何编写？

5. 技能训练

上机练习 5：使用排序查询学生相关信息

➢ 训练要点

（1）使用 SELECT 语句查询数据。

（2）使用 ORDER BY 将结果排序。

➢ 需求说明

（1）按照出生日期查询学期 ID 为 1 的学生信息。

（2）按日期先后、成绩由高到低的次序查询编号为 1 的科目考试信息。

（3）查询 2013 年 3 月 22 日参加“面向对象程序设计”考试的前五名学生的成绩信息。

（4）查询 Y2 的课时最多的科目名称。

（5）查询年龄最小的学生的姓名及所在的年级。

（6）查询 2013 年 3 月 22 日参加考试的最低分出现在哪个科目。

（7）查询学号为“10000”的学生参加过的所有考试信息，并按照时间先后次序显示。

（8）查询学号为“10000”的学生参加过的所有考试中的最高分及时间、科目。

（9）保存 SQL 为“使用排序查询学生相关信息 .sql”文件。

注意：

按照多个列进行排序时，在每列之间使用逗号分隔，并且在每列后面考虑排序的升降序，例如，…ORDER BY 列 1 ASC, 列 2 DESC。

6. LIMIT 子句

以上操作中实现了基本的对数据库表的查询操作，但是展示的是一个数据库表中的全部数据。但实际开发中，可能只要求显示指定位置指定行数的记录。下面将介绍如何通过 DQL 语句限制查询出的数据的位置和数目。

语法如下：

```
SELECT < 字段名列表 >
```

```
FROM < 表名或视图 >
[WHERE < 查询条件 >]
[GROUP BY< 分组的字段名 >]
[ORDER BY < 排序的列名 >[ASC 或 DESC]]
[LIMIT [ 位置偏移量 ], 行数 ]
```

在上述语法中的 LIMIT 部分介绍如下：

- 位置偏移量指从结果集中第几条数据开始显示（第 1 条记录的位置偏移量是 0，第 2 条记录的位置偏移量是 1……），此参数可选，当省略时默认从第一条记录开始显示。
- 行数指显示记录的条数。

LIMIT 子句可以实现数据的分页查询，即从一批结果数据中，规定每页显示多少条数据，可以查询中间某页记录。LIMIT 子句经常与 ORDER BY 子句一起使用，即先对查询结果进行排序，然后根据 LIMIT 的参数显示其中部分数据。例如，查询所有年级编号为 1 的学员信息，按学号升序显示前 4 条记录。SQL 语句如下：

```
SELECT 'studentNo','studentName','phone','address','bornDate'
FROM 'student'
WHERE 'gradeId' = 1
ORDER BY studentNo
LIMIT 4;
```

执行结果如图 3.8 所示。

studentNo	studentName	phone	address	bornDate
10000	郭靖	13645667783	天津市河西区	1990-09-08 00:00:00
10001	李文才	13645667890	地址不详	1994-04-12 00:00:00
10002	李斯文	13645556793	河南洛阳	1993-07-23 00:00:00
10003	张萍	13642345112	地址不详	1995-06-10 00:00:00

图 3.8　使用 LIMIT 子句指定行数记录

以上示例省略位置偏移量，从第 1 条记录开始显示，如果每页显示 4 条数据，要求显示第二页全部数据，经过计算，应从第 5 条记录开始显示 4 条数据，则 SQL 语句如下：

```
SELECT 'studentNo','studentName','phone','address','bornDate'
FROM 'student'
WHERE 'gradeId' = 1
ORDER BY studentNo
LIMIT 4,4;
```

执行结果如图 3.9 所示。

studentNo	studentName	phone	address	bornDate
10004	韩秋洁	13812344566	北京市海淀区	1995-07-15 00:00:00
10005	张秋丽	13567893246	北京市东城区	1994-01-17 00:00:00
10006	肖梅	13563456721	河北省石家庄市	1991-02-17 00:00:00
10007	秦洋	13056434411	上海市卢湾区	1992-04-18 00:00:00

图 3.9　使用 LIMIT 子句实现分页查询

从图 3.9 看出，通过使用“LIMIT 4,4”实际是从第 5 条记录开始显示，注意第 1 条记录的位置为 0。

7. 技能训练

上机练习 6：查询学生信息

➢ 需求说明

（1）查询 2016 年 2 月 17 日考试前 5 名的学生的学号和分数。

（2）将所有女学生按年龄从大到小排序，从第 2 条记录开始显示 6 名女学生的姓名、年龄、出生日期、手机号码信息。

（3）按出生年份分组统计学生人数，将各组中人数达到两人的年份和人数显示出来。

（4）查询参加 2016 年 2 月 17 日考试的所有学员的最高分、最低分、平均分。

任务 2　查询指定学生考试成绩

子查询

1. 简单子查询

前面学习了 MySQL 中如何使用 SELECT、INSERT、UPDATE 和 DELETE 语句对数据进行简单访问和更新操作。在此基础上，我们开始学习子查询。那么，究竟什么是子查询？子查询有什么作用？带着这些疑问，我们不妨先来解决本节第一个问题。

student 表的数据如图 3.10 所示。

studentNo	login	studentName	sex	grade	phone	address	bornDate
10000	123	郭靖	男	1	13645667783	天津市河西区	1990-09-08 00:00:00
10001	123	李文才	男	1	13645667890	地址不详	1994-04-12 00:00:00
10002	123	李斯文	男	1	13645556793	河南洛阳	1993-07-23 00:00:00
10003	123	张萍	女	1	13642345112	地址不详	1995-06-10 00:00:00
10004	123	韩秋洁	女	1	13812344566	北京市海淀区	1995-07-15 00:00:00
10005	123	张秋丽	女	1	13567893246	北京市东城区	1994-01-17 00:00:00
10006	123	肖梅	女	1	13563456721	河北省石家庄市	1991-02-17 00:00:00
10007	123	秦洋	男	1	13056434411	上海市卢湾区	1992-04-18 00:00:00
10008	123	何晴晴	女	1	13053445221	广州市天河区	1997-07-23 00:00:00
20000	000	王宝宝	男	2	15076552323	地址不详	1996-06-05 00:00:00
20010	123	何小华	女	2	13318877954	地址不详	1995-09-10 00:00:00
30011	123	陈志强	男	3	13689965430	地址不详	1994-09-27 00:00:00
30012	123	李露露	女	3	13685678854	地址不详	1992-09-27 00:00:00

图 3.10　学生表数据

问题：

根据图3.10所示的数据，查看年龄比“李斯文”小的学生，要求显示这些学生的信息。

分析：

学生的信息可以从 student 表中查询，但条件是年龄比“李斯文”小。如何实现呢？

（1）查找出“李斯文”的出生日期。

（2）利用 WHERE 语句筛选出生日期比“李斯文”大的学生。

根据以上分析思路，以上问题可以分两步实现，如示例 2 所示。

实现方法 1：分两步实现。

示例 2

```
# 查找出 " 李斯文 " 的出生日期
SELECT 'birthday' FROM 'student' WHERE 'studentname' = ' 李斯文 ';
# 利用 WHERE 语句筛选出生日期比 " 李斯文 " 大的学生
SELECT studentNo, studentName,sex,birthday,address FROM 'student' WHERE birthday > '1993-07-23';
```

分别执行上述两条语句的运行结果如图 3.11 所示。

在示例 2 中，共使用了两个查询语句：首先，通过第一条 SELECT 语句将从 student 表中查出“李斯文”的出生日期是“1993-07-23”；然后，利用第二条 SELECT 语句查询出生日期比“1993-07-23”大的学生记录，即可得到年龄比“李斯文”小的学生信息。有没有更简洁的语句呢？答案是肯定的。

	studentNo	studentN	sex	bornDate	address
□	10001	李文才	男	1994-04-12 00:00:00	地址不详
□	10003	张萍	女	1995-06-10 00:00:00	地址不详
□	10004	韩秋洁	女	1995-07-15 00:00:00	北京市海淀区
□	10005	张秋丽	女	1994-01-17 00:00:00	北京市东城区
□	10008	何睛睛	女	1997-07-23 00:00:00	广州市天河区
□	20000	王宝宝	男	1996-06-05 00:00:00	地址不详
□	20010	何小华	女	1995-09-10 00:00:00	地址不详
□	30011	陈志强	男	1994-09-27 00:00:00	地址不详

图 3.11　查询比“李斯文”年龄小的学生

实现方法 2：采用子查询实现，SQL 语句如示例 3 所示。

示例 3

```
SELECT 'studentNo','studentName','sex','birthday','address' FROM 'student'
 WHERE 'birthday' >
(SELECT 'birthday' FROM 'student' WHERE 'studentName'=' 李斯文 ');
```

从示例 3 代码中，可以发现示例 2 的两条查询语句已合并成为一条 SQL 语句。其中，示例 2 中的第一步查询语句“（SELECT 'birthday' FROM 'student' WHERE 'studentName' =' 李斯文 '）”就是子查询，因为它嵌入到外层查询语句“SELECT 'studentNo','studentName', 'sex','birthday','address' FROM 'student'”中作为 WHERE 条件的一部分。

子查询在 WHERE 语句中的一般用法如下：

```
SELECT …… FROM 表 1 WHERE 字段 1 比较运算符 ( 子查询 );
```

其中，子查询语句必须放置在一对圆括号内；比较运算符包括 >、=、<、>=、<=。

习惯上，外面的查询称为父查询，圆括号中嵌入的查询称为子查询。执行时，先

执行子查询部分，求出子查询部分的值，再执行整个父查询，返回最后的结果。

因为子查询作为 WHERE 条件的一部分，所以还可以和 UPDATE、INSERT、DELETE 一起使用，语法类似于 SELECT 语句。

注意：

将子查询和比较运算符联合使用，必须保证子查询返回的值不能多于一个。

注意：

SELECT 语句中使用 SELECT * FROM 'student' 语句的执行效率会低于 SELECT 'studentNo', 'studentName', 'sex',' birthday', 'address' FROM 'student'，因为前者获得表中所有字段值所占用的资源将大于后者获得的指定字段值所占资源。另外，后者的可维护性高于前者。因此，在编写查询语句时，建议大家采用以下格式：

SELECT 字段列表 FROM 表名 WHERE 条件表达式

示例 3 的子查询将两个查询的结果集合并在一起，除此之外，子查询还可以在多表间查询符合条件的数据，请参照以下问题。result 表的数据如图 3.12 所示。

studentNo	subjectNo	examDate	studentResult
10000	1	2016-02-15 00:00:00	71
10000	1	2016-02-17 00:00:00	60
10001	1	2016-02-17 00:00:00	46
10002	1	2016-02-17 00:00:00	83
10003	1	2016-02-17 00:00:00	65
10004	1	2016-02-17 00:00:00	70
10005	1	2016-02-17 00:00:00	95
10006	1	2016-02-17 00:00:00	93
10007	1	2016-02-17 00:00:00	23
20000	3	2016-07-09 00:00:00	68
20010	3	2016-07-09 00:00:00	90

图 3.12　result 表的数据

问题：

查询 Logic Java 课程至少一次考试刚好等于 60 分的学生名单。

分析：

（1）查询 subject 表，获得 Logic Java 课程的课程编号。

（2）根据课程编号，查询 result 表中成绩是 60 分的学生的学号。

（3）根据学号，查询 student 表得到学生姓名。

实现方法：采用子查询，如示例 4 所示。

示例 4

```
SELECT 'studentName' FROM 'student' WHERE 'studentNo' = (
   SELECT 'studentNo' FROM 'result'
      INNER JOIN 'Subject' ON result.subjectNo= subject.subjectNo
```

```
    WHERE 'studentResult'=60 AND 'subjectName'=’ Logic Java’
)
```

其中，括号中的子查询查询出刚好 60 分的学生的学号。

上述语句的运行结果如图 3.13 所示。

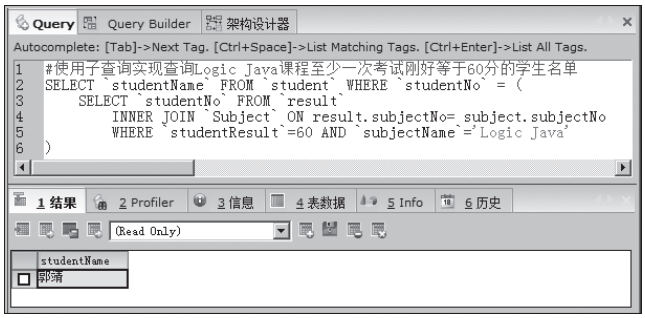

图 3.13　查询刚好及格的学生

2. 技能训练

上机练习 7：查询指定学生的考试成绩

➢　训练要点

使用子查询返回单条记录。

➢　需求说明

查询参加最近一次 Logic Java 考试成绩的学生的最高分和最低分。

➢　实现思路

（1）查询获得 Logic Java 课程的课程编号。

（2）查询获得 Logic Java 课程最近一次的考试日期。

（3）根据课程编号查询考试成绩的最高分和最低分。

前面学习了如何使用两层嵌套子查询语句完成简单的数据查询功能。但在实际的软件开发中，程序员可能需要通过使用多层嵌套的子查询来实现更复杂的查询功能。

任务 3　查询某学期开设的课程

IN 和 NOT IN 子查询

使用 IN 关键字可以使父查询匹配子查询返回的多个单字段值。

1. IN 子查询

使用 =、> 等比较运算符时，要求子查询只能返回一条或空的记录。在 MySQL 中，

当子查询跟随在 =、!=、<、<=、> 和 >= 之后时，不允许子查询返回多条记录。例如，示例 4 中查询 Logic Java 课程至少一次考试刚好等于 60 分的学生名单。在 result 表中刚好只有 1 条记录满足条件：郭靖（学号 10000）的 Logic Java 课程的考试成绩刚好是 60 分。如果有多条记录满足条件，即有多个学生的 Logic Java 课程考试成绩为 60 分，则采用上述子查询将出现编译错误。

在使用示例 4 的查询代码之前，使用 INSERT 语句向成绩表中添加 1 条记录学号为 10008，她的 Logic Java 课程考试成绩刚好也是 60 分。此时，执行示例 4 的代码，出现的语法错误如图 3.14 所示。

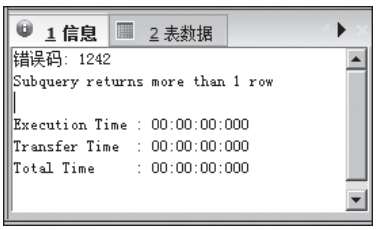

图 3.14　比较运算符后的子查询不允许返回多条记录

出错信息“Subquery returns more than 1 row”的意思是“子查询返回值不唯一”。如何解决呢？答案很简单，只需将“=”改为“IN”即可，如示例 5 所示。

示例 5

```
SELECT 'studentName' FROM 'student'
WHERE 'studentNo' IN(
   SELECT 'studentNo' FROM 'result'
   WHERE 'subjectNo' = (
      SELECT 'subjectNo' FROM 'subject'
      WHERE 'subjectName'=’ Logic Java’
   )AND 'studentResult' = 60
);
```

上述语句的运行结果如图 3.15 所示。

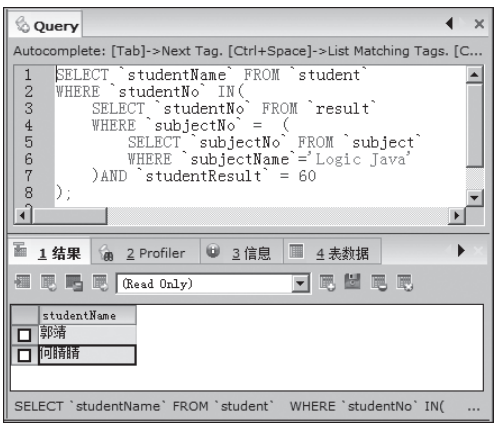

图 3.15　IN 子查询

从示例 5 可以看出，IN 后面的子查询可以返回多条记录，用于限制学号的筛选范围。尽管示例 4 与示例 5 子查询语句所完成的功能是相同的，都是查询 Logic Java 课程考试成绩至少一次是 60 分的学生记录，而示例 5 则在子查询语句中又嵌套了一个子查询，用于查询获得 Logic Java 课程的课程编号。因此，示例 5 是一个三层嵌套的子查询语句。

注意：

能找出示例 5 中具有嵌套关系的三个查询语句吗？能否描述一下示例 5 的执行步骤？

下面再看一个新问题。

问题：

查询参加 Logic Java 课程最近一次考试的在读学生名单。

分析：

（1）获得 Logic Java 课程的课程编号。

```
SELECT 'subjectNo' FROM 'subject' WHERE 'subjectName'='Logic Java';
```

（2）根据课程编号查询得到 Logic Java 课程最近一次的考试日期。

```
SELECT MAX('examDate') FROM 'result' WHERE 'subjectNo'= (
SELECT 'subjectNo' FROM 'subject' WHERE 'subjectName'='Logic Java' )
```

（3）根据课程编号和最近一次的考试日期查询学生信息。

实现这个需求需要查询三个表：课程表 subject、成绩表 result 和学生表 student，具体的 SQL 语句如示例 6 所示。

示例 6

```
/* 采用 IN 子查询获得参加考试的在读学生名单 */
SELECT 'studentNo', 'studentName' FROM 'student' WHERE 'studentNo'
IN(
  SELECT 'studentNo' FROM 'result'
  WHERE 'subjectNo' = (
    # 获得参加 Logic Java 课程最近一次考试的学生学号
    SELECT 'subjectNo' FROM 'subject'
    WHERE 'subjectName'='Logic Java'
  ) AND 'examDate' = (
     # 获得 Logic Java 课程最近一次的考试日期
     SELECT MAX('examDate') FROM 'result'
     WHERE 'subjectNo' = (
        # 获得 Logic Java 课程的课程编号
        SELECT 'subjectNo' FROM 'subject'
        WHERE 'subjectName'='Logic Java'
     )
  )
);
```

上述语句的运行结果如图 3.16 所示。

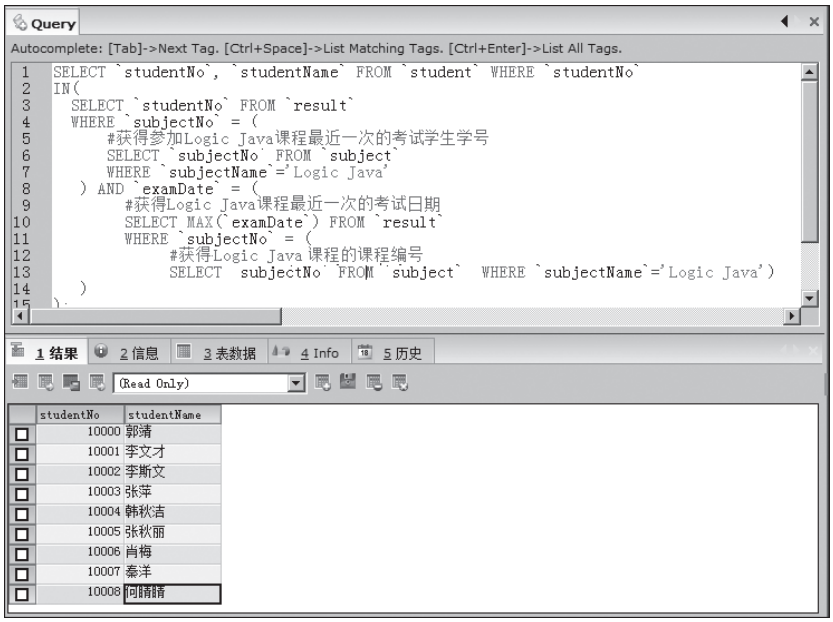

图 3.16　查询参加“Logic Java”课程最近一次考试的在读学生名单

仔细阅读示例 6 的代码，会发现这是一个包含了四层嵌套子查询的查询语句。第四层（即最内层）子查询用于获得 Logic Java 课程的课程编号，对应的代码如下：

```
SELECT 'subjectNo' FROM 'subject' WHERE 'subjectName' = 'Logic Java';
```

第三层子查询是在第四层子查询获得 Logic Java 课程编号的基础上，获得此课程最近一次的考试日期，代码如下：

```
SELECT MAX('examDate') FROM 'result'
WHERE 'subjectNo' = (
    SELECT 'subjectNo' FROM 'subject'
    WHERE 'subjectName' = 'Logic Java'
);
```

有了 Logic Java 课程最近一次的考试日期后，执行第二层子查询，即可获得参加 Logic Java 课程最近一次考试的所有学生的学号，代码如下：

```
SELECT 'studentNo' FROM 'result'
WHERE 'subjectNo' = (
   # 获得参加 Logic Java 课程最近一次考试的在读学生学号
    SELECT 'subjectNo' FROM 'subject'
    WHERE 'subjectName'='Logic Java'
  ) AND 'examDate' = (
     # 获得 Logic Java 课程最近一次的考试日期
     SELECT MAX('examDate') FROM 'result'
     WHERE 'subjectNo' = (
        # 获得 Logic Java 课程的课程编号
```

```
            SELECT 'subjectNo' FROM 'subject'
            WHERE 'subjectName'='Logic Java'
        )
);
```

最外层查询（第一层查询）是依据第二层子查询的查询结果——所有参加 Logic Java 课程最近一次考试的在读学生的学号从 student 表中查找出对应的学生姓名。至此，通过嵌套的四层子查询得到了参加 Logic Java 课程最近一次考试的在读学生名单。

2. NOT IN 子查询

如何查询得到没有参加 Logic Java 课程最近一次考试的在读学生名单？你一定想到了，在示例 6 代码的 IN 关键字之前加上否定的 NOT 即可获得未参加考试的学生名单。查询语句如示例 7 所示。

示例 7

```
/* 采用 NOT IN 子查询 , 查看未参加考试的在读学生名单 */
SELECT 'studentNo', 'studentName' FROM 'student' WHERE 'studentNo'
NOT IN(
 SELECT 'studentNo' FROM 'result'
 WHERE 'subjectNo' = (
   # 获得参加 Logic Java 课程最近一次考试的学生学号
   SELECT 'subjectNo' FROM 'subject'
   WHERE 'subjectName'='Logic Java'
 ) AND 'examDate' = (
    # 获得 Logic Java 课程最近一次的考试日期
    SELECT MAX('examDate') FROM 'result'
    WHERE 'subjectNo' = (
       # 获得 Logic Java 课程的课程编号
       SELECT 'subjectNo' FROM 'subject'
       WHERE 'subjectName'='Logic Java'
    )
 )
);
```

上述语句的运行结果如图 3.17 所示。

在图 3.17 中看到，示例 7 中代码的运行结果集数据不仅包含了 Logic Java 课程所在学期 ID 为 1 的学生记录，还包括了学期 ID 为 2 和第二学年等高年级学生的名单，这与我们的初衷有所差异。我们希望获得 Logic Java 课程所在学期（即学期 ID 为 1）在读学生没有参加这门课程最近一次考试的学生名单，那么，如何在示例 7 中代码的基础上进行完善呢？其实不难，只需要增加一个查询条件以限制 Logic Java 课程所在学期的学生即可。

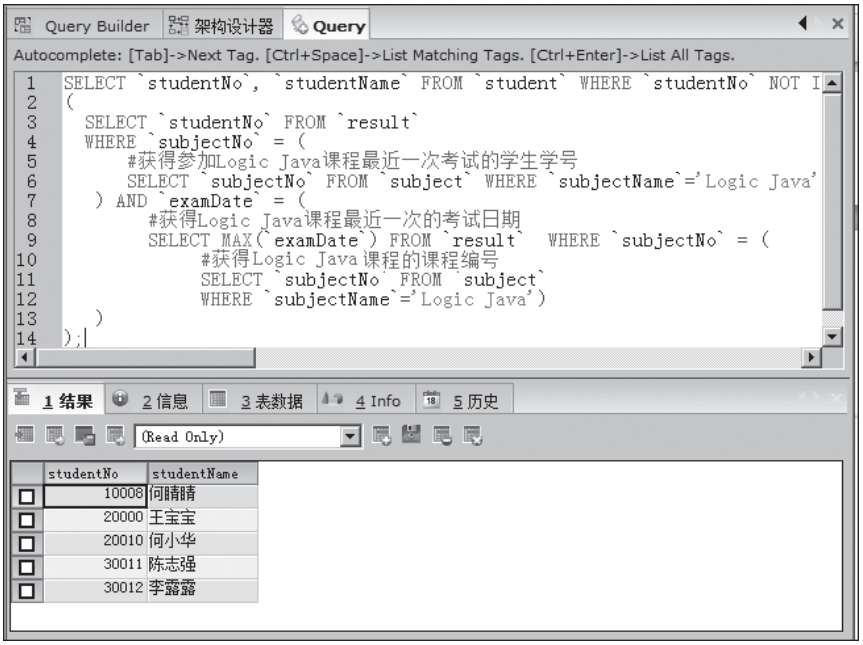

图 3.17　查看未参加 Logic Java 课程最近一次考试的在读学生名单

修改示例 7 中的代码以实现查询得到没有参加 Logic Java 课程最近一次考试在读学生姓名的功能，代码如下：

```
SELECT 'studentNo', 'studentName' FROM 'student' WHERE 'studentNo'
NOT IN(
  SELECT 'studentNo' FROM 'result'
  WHERE 'subjectNo' = (
    # 获得参加 Logic Java 课程最近一次考试的学生学号
    SELECT 'subjectNo' FROM 'subject' WHERE 'subjectName'='Logic Java'
  ) AND 'examDate' = (
     # 获得 Logic Java 课程最近一次的考试日期
     SELECT MAX('examDate') FROM 'result'  WHERE 'subjectNo' = (
        # 获得 Logic Java 课程的课程编号
        SELECT 'subjectNo' FROM 'subject'
        WHERE 'subjectName'='Logic Java')
  )
)
AND 'gradeID' = (
  SELECT 'gradeID' FROM 'subject'
  WHERE 'subjectName' = 'Logic Java'
);
```

3．技能训练

上机练习 8：查询某学期开设的课程

➢　训练要点

使用子查询返回多条记录。

➢　需求说明

使用 IN 关键字的子查询来查询学期 ID 为 1 开设的课程。

上机练习 9：查询某课程最近一次考试缺考的学生名单

➢　需求说明

使用 NOT IN 关键字的子查询来查询未参加 HTML 课程最近一次考试的在读学生名单。

提示：

（1）查询没有参加 HTML 课程最近一次考试的学生名单。

（2）限定 HTML 课程所在学期。

本章总结

本章学习了以下知识点：

- MySQL 的存储引擎。
  - 常用存储引擎：InnoDB、MyISAM。
  - InnoDB：支持事务处理、外键、占用空间比 MyISAM 大，适合于需要事务处理、更新、删除频繁的场景。
  - MyIASM：不支持事务和外键，占用空间较小，访问速度快，适合于不需事务处理，频繁查询的应用场景。
- MySQL 中的 DML 语句。
  - 插入数据记录（INSERT）：MySQL 中 INSERT INTO……VALUES……语句可同时插入多条记录。
  - 更新数据记录（UPDATE）。
  - 删除数据记录（DELETE/TRUNCATE）。
- 学习 MySQL 中的 SELECT 语句。
  - SELECT 语法：查询所有、查询部分，查询中使用别名、查询空值、查询使用常量。
  - WHERE 子句：对查询结果进行限定。
  - LIMIT 子句：对查询结果进行限定。
  - 常用函数分类：聚合函数、字符串函数、时间日期函数、数学函数。
- 可以通过子查询——将一个查询嵌套在另一个查询中。
- 比较运算符后面的子查询只能返回单个数值。
- IN 子查询后面可跟随返回多条记录的子查询，用于检测某字段的值是否存在于某个范围中。

本章练习

一、选择题

1. 已知MySQL数据库中有一个储户表和一个储蓄卡表，储蓄卡表中的“储户编号”字段引用了储户表的“编号”字段，则关于下面的查询语句的说法正确的是（　　）。

SELECT * FROM 储户表 WHERE 编号 NOT IN(SELECT 储户编号 FROM 储蓄卡表）

A．查询没有储户的储蓄卡

B．查询没有储蓄卡的储户

C．该 SQL 语句有语法错误，不能执行

D．以上都不正确

2．若实现查询储蓄卡中账户余额最多的 5 个储蓄卡卡号，以下 SQL 语句正确的是（　　）。

A．SELECT TOP 5 卡号 FROM 储蓄卡 ORDER BY 账户余额 ;

B．SELECT 卡号 FROM 储蓄卡 ORDER BY 账户余额 DESC LIMIT 1,5;

C．SELECT 卡号 FROM 储蓄卡 LIMIT 5 ORDER BY 账户余额 DESC ;

D．SELECT 卡号 FROM 储蓄卡 ORDER BY 账户余额 DESC LIMIT 5 ;

3．以下（　　）方式可以实现多表关联查询。

A．设定权限　　B．子查询

C．表连接　　D．角色

4．下列（　　）函数可以获取当前的日期和时间。

A．CURDATE()　　B．CURTIME()

C．NOW()　　D．GETDATE()

5．下列（　　）可用于将已存在表的数据填充到新表。

A．INSERT INTO……SELECT　　B．UNION

C．子查询　　D．表连接

6．下列有关子查询和表连接的说法，错误的是（　　）。

A．子查询一般可以代替表连接

B．表连接能代替所有的子查询，所以一般优先采用子查询

C．如果需要显示多表数据，则优先考虑表连接

D．如果只是作为查询的条件部分，则一般考虑子查询

二、简答题

1．请说明在 MySQL 中如何限定查询结果显示的条数。

2．请说明表连接和子查询是否可以相互转换，及各自的应用场景。

3．使用子查询获得当前没有被读者借阅的图书信息。要求：输出图书名称、图书编号、作者姓名、出版社和单价。

4．使用子查询获得今年的所有缴纳罚款的读者清单，要求输出的数据包括读者姓名、图书名称、罚款日期和缴纳罚金等。

5．使用子查询获得地址为空的所有读者尚未归还的图书清单。要求：按读者编号从高到低、借书日期由近至远的顺序输出读者编号、读者姓名、图书名称、借书日期和应归还日期。

注意：第 2 题 ~ 第 5 题使用第 2 章作业中创建的图书馆管理系统数据库。