第1部分

实验指导

PART 1



实验指导部分根据课程的基本要求设计了12个实验，每个实验都和课程学习的知识点相配合，以帮助读者通过上机实践加深对课程内容的理解，更好地掌握数据库的基本操作。实验指导部分使用的数据库，如果没有特别指明，则为student数据库，该数据库在实验2创建。

为了达到理想的实验效果，希望读者能做到以下3点。

(1) 在实验前要认真准备，根据实验目的和实验内容，复习好实验中要用到的概念与操作步骤，做到胸有成竹，提高上机效率。

(2) 在实验过程中要积极思考，注意归纳各种操作的共同规律，分析操作结果以及各种屏幕信息的含义。

(3) 在实验后要认真总结，总结本次实验有哪些收获，还存在哪些问题，并写出实验报告。

实验 1　MySQL 8.0 的安装及管理工具的使用

一、实验目的

1. 了解安装 MySQL 8.0 对软、硬件的要求，掌握安装方法。
2. 熟悉 Workbench 管理工具的基本使用方法。
3. 了解数据库的结构及其包含的对象。
4. 了解在 Workbench 中执行 SQL 语句的方法。

二、实验内容

1. 确保当前软、硬件环境满足要求，安装 MySQL 8.0。

2. 在 Windows 桌面上单击“开始”按钮，选择 MySQL→MySQL Workbench 8.0 CE 命令，启动 Workbench 管理工具。

3. 在 Workbench 中查看已建立的数据库连接。重新创建一个连接，名称为“my”，连接方法为 Standard(TCP/IP)，主机名为“127.0.0.1”，端口为“3306”，用户名为“root”，密码为空。

4. 双击连接“my”，连接到服务器。

5. 在 Workbench 的 Navigator 窗口的 Administration(管理)选项卡中查看以下信息。

(1) Server Status：服务器状态。

(2) Client Connections：客户连接。

(3) Users and Privileges：用户和权限。

(4) Dashboard：仪表盘。

6. 切换到 Schemas(模式)选项卡，查看服务器上的数据库。

7. 在主窗口中选择 File→New Query Tab 命令，新建一个查询窗口。

8. 在查询窗口中输入 SQL 语句“USE student”，然后选择 Query→Execute Current Statement 命令执行该语句，将当前操作的数据库设置为 student。

9. 停止 MySQL 数据库服务，再启动 MySQL 数据库服务。

三、实验思考

1. Workbench 的作用是什么？如何启动 Workbench？
2. 如何创建数据库连接？创建连接的参数有哪些？
3. 在仪表盘窗口中可以查看哪些服务器的信息？
4. 查询窗口的作用是什么？如何打开查询窗口？

5. 使用什么 SQL 语句来选择当前操作的数据库？

6. 使用 Workbench 工具观察数据库中包含哪些对象。

实验 2　数据库和表

一、实验目的

1. 了解 MySQL 数据库的逻辑结构和物理结构的特点。

2. 学会使用 Workbench 创建数据库和表。

3. 学会使用 SQL 语句创建数据库和表。

4. 了解在 Workbench 中对表数据进行插入、修改和删除操作的方法。

二、实验内容

1. 在 Workbench 中创建数据库。

(1) 启动 Workbench，在 Navigator 窗口中切换到 Schemas 选项卡。

(2) 在窗口中右击鼠标，选择 Create Schema 命令，在显示的界面中输入要创建的数据库名称“student”，单击 Apply 按钮，然后再单击一次 Apply 按钮，创建数据库。

(3) 右击鼠标，选择 Refresh All 命令，显示刚才创建的数据库 student。

2. 右击 student 数据库，在弹出的快捷菜单中选择 Schema Inspector(数据库检查器)命令，然后分别查看 Info(信息)、Tables(表)、Columns(列)、Indexes(索引)、Triggers(触发器)、Stored Procedures(存储过程)、Functions(函数)、Grants(授权)和 Events(事件)等选项卡的内容。

3. 右击 student 数据库，选择 Alter Schema 命令，将字符集修改为 utf8，将排序修改为 Default Collation。

4. 右击 student 数据库，选择 Drop Schema 命令，删除数据库。

5. 在查询编辑器窗口中使用 SQL 语句 CREATE DATABASE 创建 student 数据库，然后通过 SHOW CREATE DATABASE student 语句查看数据库的信息。

6. 使用 Workbench 创建 student、course 表，使用 SQL 语句创建 score 表，表的数据结构如图 1-1～图 1-3 所示。

(1) student 表中各字段的解释如下。

① id：学号。

② name：姓名。

③ gender：性别。

④ address：家庭住址。

⑤ birthday：生日。

⑥ memo：备注。

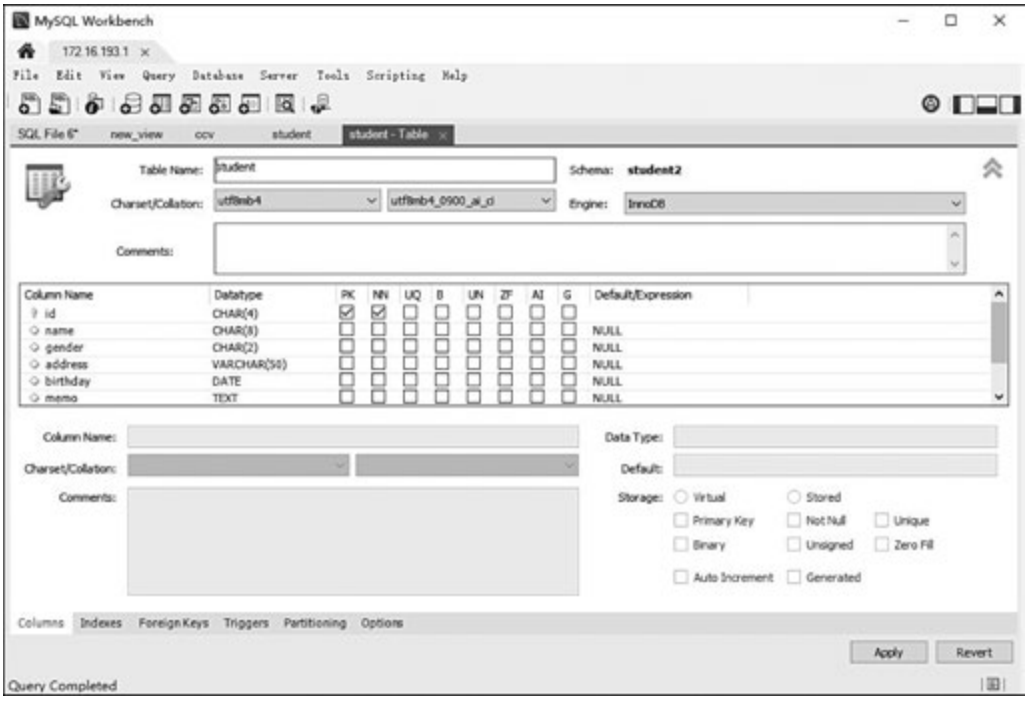


图 1-1　学生基本情况表 student 的结构

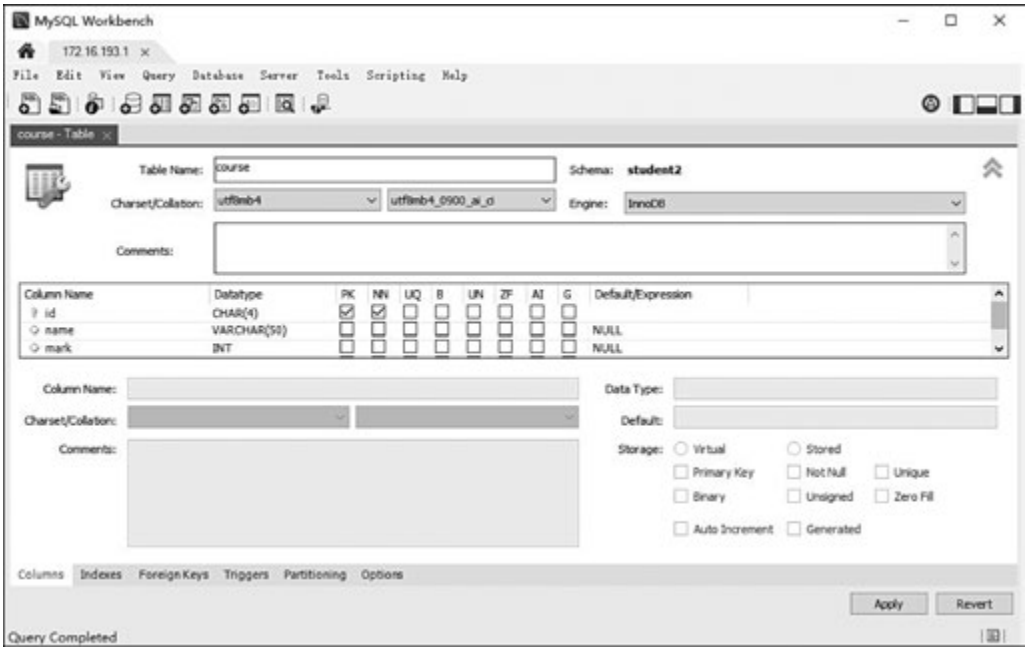


图 1-2　课程信息表 course 的结构

(2) course 表中各字段的解释如下。

① id：课程编号。

② name：课程名称。

③ mark：学分。

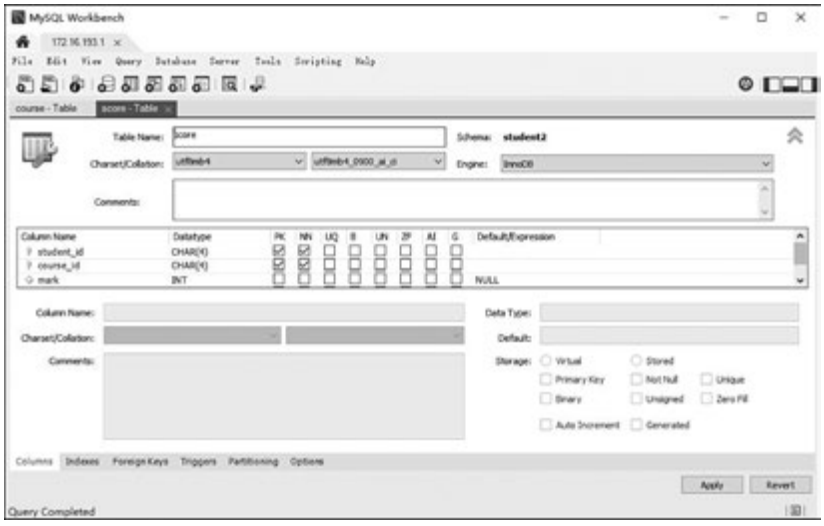


图 1-3　学生成绩表 score 的结构

(3) score 表中各字段的解释如下。

① student_id：学号。

② course_id：课程编号。

③ mark：分数。

7. 在 Workbench 中，将 student 表的学号列设置为主键，非空。

8. 在 Workbench 中，将 course 表的课程编号列设置为主键，非空。

9. 在 Workbench 中，将 score 表的学号列和课程编号列的组合设置为主键，非空。

10. student、course、score 表中的数据如图 1-4～图 1-6 所示。在 Workbench 中为 3 张表添加数据。

11. 使用 SQL 语句 ALTER...ADD 为 score 表添加一个名为“memo”的数据列，其数据类型为 VARCHAR(50)，允许为空。

```
ALTER TABLE score ADD memo VARCHAR(50) NULL
```

12. 使用 SQL 语句 ALTER...ADD 在 student 表的 name 列前添加一列 tall，类型为整型，NOT NULL。

```
ALTER TABLE student ADD tall int NOT NULL BEFORE name;
```

13. 删除 student 表的 tall 列。

```
ALTER TABLE student DROP tall;
```

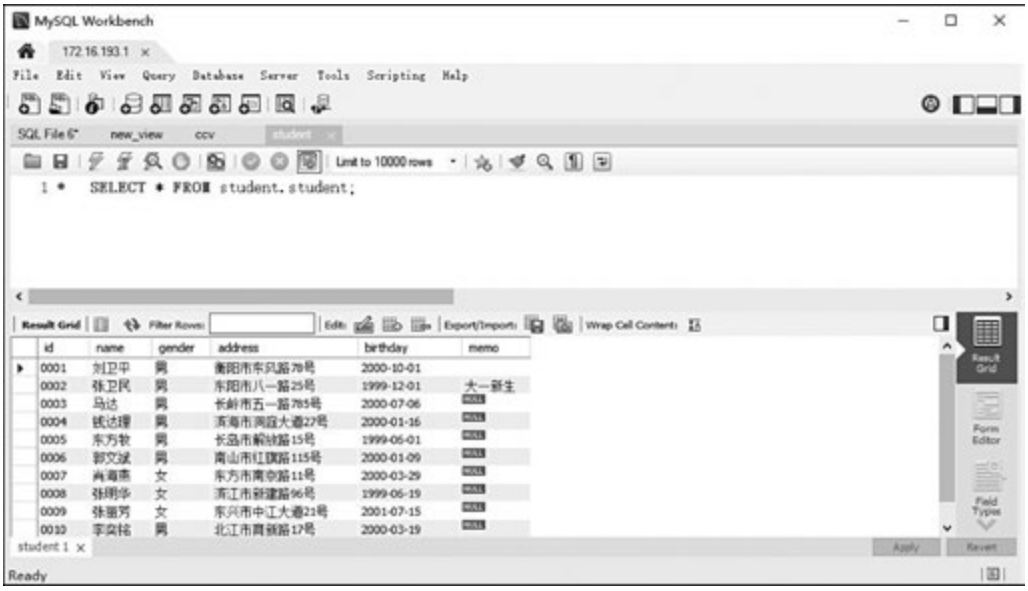


图 1-4　student 表中的数据

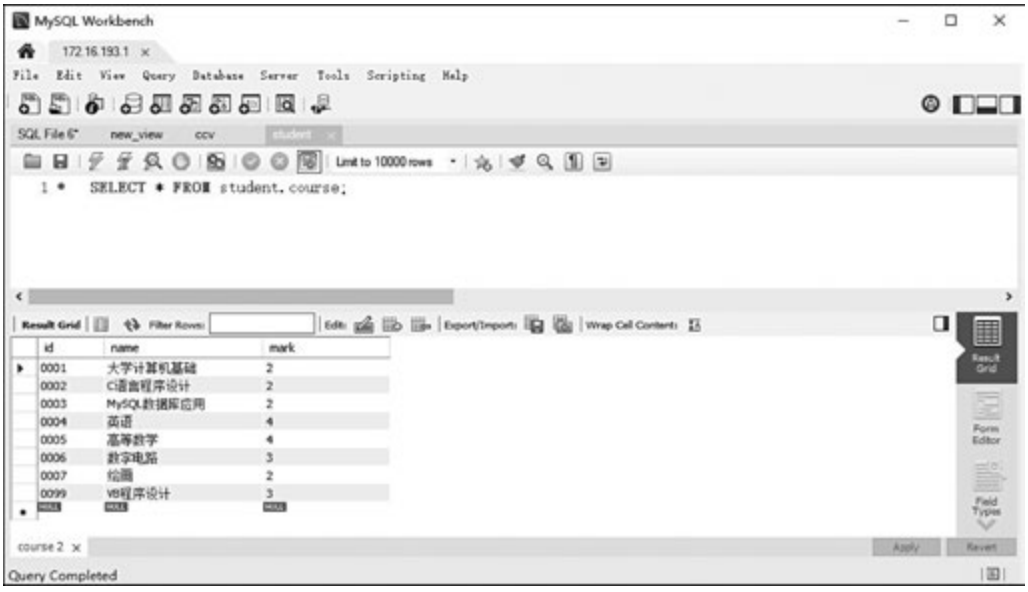


图 1-5　course 表中的数据

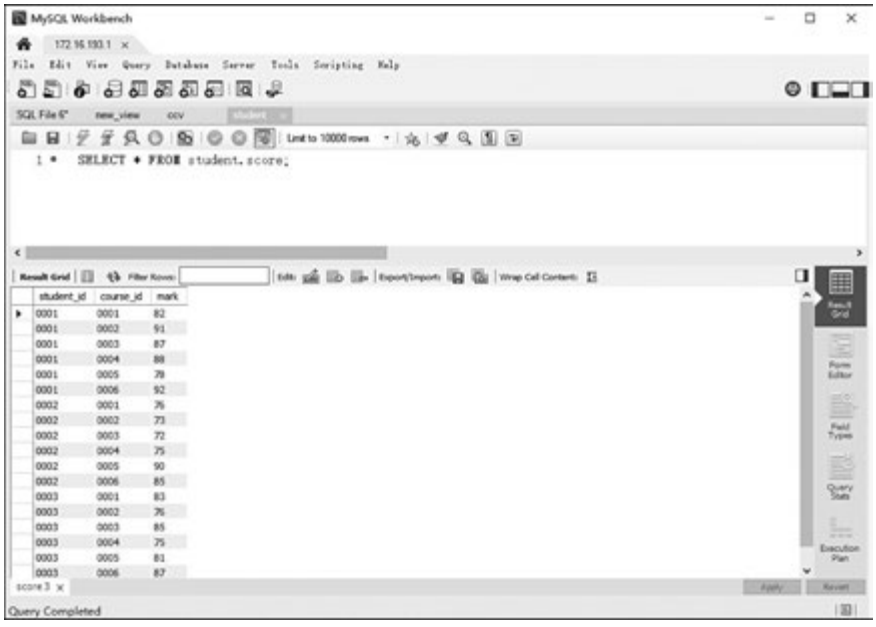


图 1-6　score 表中的数据

14. 使用 Workbench 删除 score 表。

15. 使用 SQL 语句 DROP TABLE 删除 course 表。

16. 使用 SQL 语句 DROP DATABASE 删除 student 数据库。

三、实验思考

1. 在创建数据库时产生了哪些文件？

2. 删除了数据库，其数据文件是否已经被删除？是否任何人都可以删除数据库？删除的数据库还有可能恢复吗？

3. 能够删除系统数据库 mysql 吗？

4. 使用 SQL 语句删除在 student 数据库的 score 表中添加的“memo”数据列。

5. 在 Workbench 中，向 student 数据库的 student 表中输入数据时，如果输入相同学号的记录将会出现什么现象？为什么？

6. 在 MySQL 8.0 中能够将数据表中的字段名和其数据类型同时改变吗？

实验 3　使用 SQL 操作数据库

一、实验目的

1. 掌握使用 SQL 语句进行数据插入、更新和删除的方法。

2. 掌握使用 SELECT 语句进行基本查询的方法。

3. 掌握使用 SELECT 语句进行条件查询的方法。

4. 理解 GROUP BY、ORDER BY 以及 UNION 子句在 SELECT 语句中的作用。

5. 学习嵌套查询的方法。

6. 了解连接查询的方法。

二、实验内容

1. 启动 Workbench，打开查询窗口，使用 SQL 语句 USE 设置 student 为当前操作的数据库。

2. 使用 SQL 语句 INSERT INTO... VALUES 向 student 数据库的 score 表中插入数据：学号为 0004，课程编号为 0001，分数为 80。

3. 使用 SQL 语句 UPDATE 修改 student 数据库的 score 表中学号为 0003 的学生信息，使其课程编号为 0005 的课程的分数为 90。

```
UPDATE score SET mark = 90 WHERE student_id = '0003' AND course_id = '0005'
```

4. 使用 SQL 语句 DELETE 删除 student 数据库的 score 表中学号为 0004 的学生的成绩记录。

```
DELETE FROM score WHERE student_id = '0004'
```

5. 在 student 数据库中使用 SELECT 语句进行基本查询。

(1) 在 student 表中查询每个学生的学号、姓名、出生日期等信息。

(2) 查询学号为 0002 的学生的姓名和家庭住址。

(3) 找出所有男同学的学号和姓名。

6. 使用 SELECT 语句进行条件查询。

(1) 在 score 表中查找分数为 80～90 的学生的学号和分数。

(2) 在 score 表中查询课程编号为 0003 的学生的平均分。

(3) 在 score 表中查询学习各门课程的人数。

(4) 将学生按出生日期升序排序。

(5) 查询所有姓“张”的学生的学号和姓名。

```
SELECT id,name FROM student WHERE name LIKE '张%'
```

7. 对 student 表按性别排序，列出学生的学号、姓名、性别、出生日期及家庭住址，性别相同的再按学号由小到大排序。

8. 使用 GROUP BY 查询子句列出各个学生的平均成绩。

9. 使用 UNION 运算符将 student 表中姓“刘”的学生的学号和姓名与姓“张”的学生的学号和姓名返回在一个表中，如图 1-7 所示。

```
SELECT * FROM student WHERE name LIKE '刘%'
UNION
```

```
SELECT * FROM student WHERE name LIKE '张%'
```

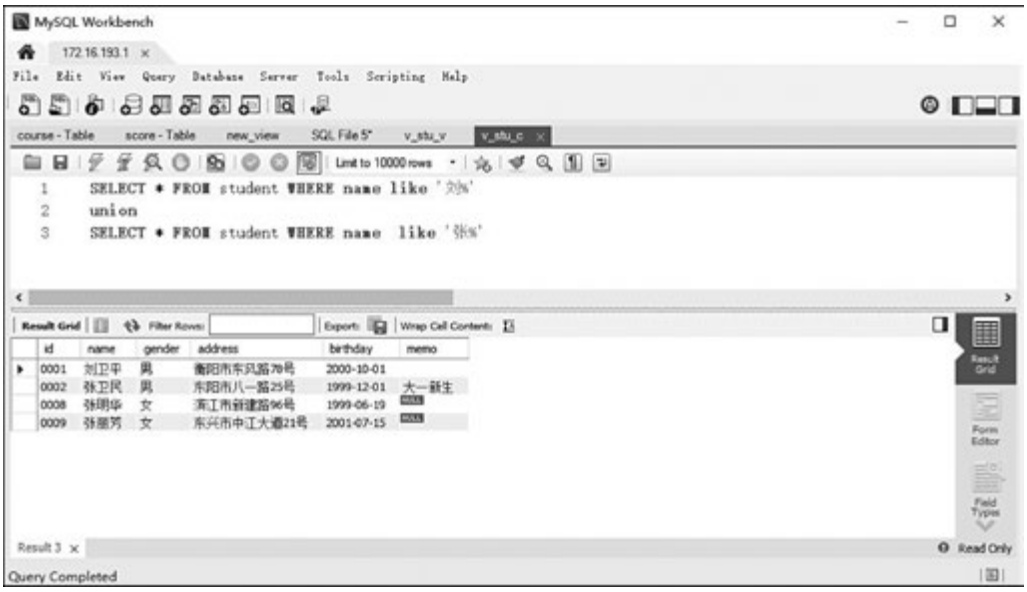


图 1-7　联合查询结果集

10. 嵌套查询。

(1) 在 student 表中查找与“刘卫平”性别相同的所有学生的姓名、出生日期。

```
SELECT name,birthday
FROM student
WHERE gender =
    (SELECT gender
     FROM student
     WHERE name = '刘卫平')
```

(2) 使用 IN 子查询查找所修课程编号为 0002、0005 的学生的学号、姓名、性别。

```
SELECT id,name,gender
FROM student
WHERE student.id IN
    (SELECT student_id
    FROM score
    WHERE course_id IN ('0002','0005'))
```

(3) 列出学号为 0001 的学生的分数比学号为 0002 的学生的最低分数高的课程编号和分数。

```
SELECT course_id,mark
FROM score
WHERE student_id = '0001' AND mark > ANY
    (SELECT mark FROM score
```

```
WHERE student_id = '0002')
```

(4) 列出学号为 0001 的学生的分数比学号为 0002 的学生的最高分数还要高的课程编号和分数。

11. 连接查询。

(1) 查询分数为 80～90 的学生的学号、姓名、分数。

```
SELECT student.id,student.name,score.mark
FROM student,score
WHERE student.id = score.student_id AND score.mark BETWEEN 80 AND 90
```

(2) 查询学习“C 语言程序设计”课程的学生的学号、姓名、分数。

```
SELECT student.id,student.name,student.mark
FROM student
INNER JOIN score ON student.id = score.student_id
INNER JOIN course ON score.course_id = course.id WHERE course.name = 'C 语言程序设计'
```

(3) 查询所有男同学的选课情况,要求列出学号、姓名、课程名称、分数。

(4) 查询每个学生的所选课程的最高成绩,要求列出学号、姓名、课程编号、分数。

(5) 查询所有学生的总成绩,要求列出学号、姓名、总成绩,没有选修课程的学生的总成绩为空。

提示:使用左外连接。

(6) 为 score 表添加数据行:学号为 0004、课程编号为 0008(course 表中没有的课程编号)、分数为 93。查询所有课程的选修情况,要求列出课程编号、课程名称、选修人数,course 表中没有的课程列值为空。

提示:使用右外连接。

(7) 查询所有平均分大于或等于 60 的学生的学号、姓名、平均分,并按平均分从高到低排序。

```
SELECT id,name,b.avg_mark FROM student a INNER JOIN (SELECT student_id,avg(mark) avg_mark
FROM score GROUP BY student_id HAVING avg(mark)>=60) b ON a.id = b.student_id ORDER BY b.avg_
mark DESC
```

三、实验思考

1. 查询所有没有选修课程的学生的信息,返回结果包括学号、姓名、性别。

2. 在 student 表和 score 表之间实现交叉连接。

3. 查询每个学生所选修课程的平均成绩。

4. 在查询语句中 SELECT、FROM 和 WHERE 关键字的作用是什么?

5. 在查询的 FROM 子句中实现表与表之间的连接有哪几种方式?对应的关键字分别是什么?

实验 4　索引与数据完整性

一、实验目的

1. 学会使用 Workbench 和 SQL 语句 CREATE INDEX 创建索引。
2. 学会使用 Workbench 查看索引。
3. 学会使用 Workbench 和 SQL 语句 DROP INDEX 删除索引。
4. 掌握使用 SQL 语句创建外键约束的方法。
5. 掌握使用 SQL 语句创建非空约束的方法。
6. 掌握使用 SQL 语句创建唯一性约束的方法。
7. 掌握使用 SQL 语句创建检查约束的方法。
8. 掌握使用 SQL 语句创建默认值约束的方法。

二、实验内容

1. 分别使用 Workbench 和 SQL 语句为 student 数据库的 student 表中的姓名列创建索引。

2. 分别使用 Workbench 和 SQL 语句为 student 数据库的 course 表中的课程名称列创建唯一索引。

3. 分别使用 Workbench 和 SQL 语句为 student 数据库的 score 表中的分数列创建索引，命名为 score_index。

```
CREATE INDEX score_index ON score(mark)
```

4. 为 student 数据库的 score 表中的“student_id”和“course_id”字段创建一个复合唯一索引，命名为 score_id_c_ind。

```
CREATE UNIQUE INDEX score_id_c_ind ON score(student_id,course_id)
```

5. 分别使用 Workbench 和 SQL 语句查看 score 表和 student 表上的索引信息。

```
SHOW KEYS FROM score;
SHOW KEYS FROM student;
```

6. 使用 Workbench 为 student 表的出生日期列创建一个索引。

7. 使用 SQL 语句将 score_index 索引更名为 score_ind。

```
ALTER TABLE student RENAME INDEX score_index TO score_ind;
```

8. 分别使用 Workbench 和 SQL 语句 DROP INDEX 删除 score_ind 索引。再次使用 SQL 语句 SHOW KEYS FROM score 查看 score 表上的索引信息。

```
DROP INDEX score_ind;
```

9. 为 score 表增加两个外键约束，student_id 字段参照引用 student 表中的 id 字段，course_id 字段参照引用 course 表中的 id 字段。

```
ALTER TABLE score ADD
CONSTRAINT fk_st_id FOREIGN KEY (student_id) REFERENCES student(id);
ALTER TABLE course ADD CONSTRAINT
fk_course_id FOREIGN KEY (course_id) REFERENCES course(id);
```

在 score 表中插入一条记录，学号为 0100，课程编号为 0099，分数为 78。观察 MySQL 会做何处理，为什么？如何解决所产生的问题？

使用 SQL 语句删除 score 表的外键 fk_course_id。

10. 为 course 表的学分列添加默认值约束，当插入记录时，如果学分为空，将其设置为 3。

```
ALTER TABLE course CHANGE COLUMN mark int NULL DEFAULT 3;
```

在 course 表中插入记录(id：0022，name：毕业设计)，观察该条记录插入后学分列的值。

11. 在 student 数据库中用 CREATE TABLE 语句创建 student_con 表，并同时创建约束。

(1) 在创建表的同时创建约束，表的结构如图 1-8 所示。

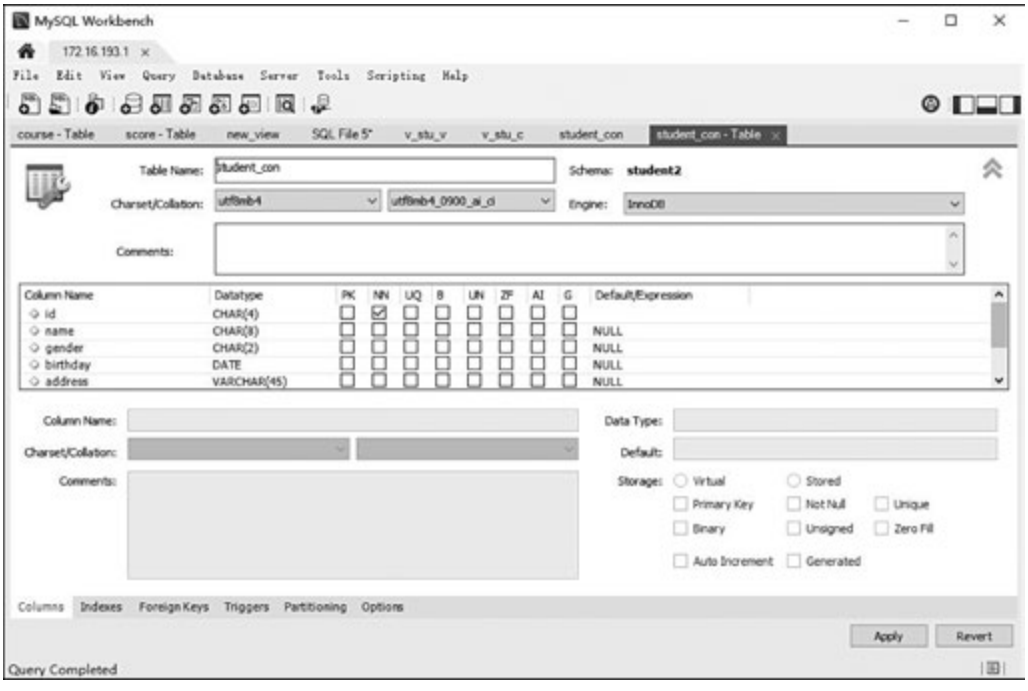


图 1-8　student_con 表的结构

约束要求如下。

① 将学号设置为主键(PRIMARY KEY),主键名为 pk_sid。

② 为姓名添加唯一约束(UNIQUE),约束名为 uk_name。

③ 为性别添加默认约束(DEFAULT),默认名称为 df_sex,其值为“男”。

(2) 为出生日期添加属性值约束(CHECK),约束名为 birthday_check,其检查条件为 birthday>'1995-07-01'。

```
ALTER TABLE student_con
ADD CONSTRAINT birthday_check
CHECK(birthday>'1995-07-01');
```

(3) 在 student_con 表中插入如表 1-1 所示的数据。

表 1-1 在 student_con 表中插入的数据

学号	姓名	性别	出生日期	家庭住址
0009	张小东		2000-04-06	
0010	李梅	女	1994-08-05	
0011	王强		2000-09-02	
0012	王强		2001-06-03	

分析各约束在插入记录时所起的作用,查看插入记录后表中的数据与所插入的数据是否一致。

三、实验思考

1. 什么是索引?在 MySQL 8.0 中有哪些索引类型?简单叙述它们的区别。
2. 在 MySQL 8.0 中有哪些约束?简单叙述每种约束的作用。
3. 创建索引的作用是什么?索引是否创建得越多越好?

实验 5 视图

一、实验目的

1. 掌握使用 Workbench 和 SQL 语句 CREATE VIEW 创建视图的方法。
2. 掌握使用 SQL 语句 ALTER VIEW 修改视图的方法。
3. 了解删除视图的 SQL 语句 DROP VIEW 的用法。

二、实验内容

1. 在 student 数据库中以 student 表为基础,使用 Workbench 建立名为 v_stu 的视

图，使视图显示学生的姓名、性别、家庭住址。

2. 在 student 数据库中，使用 SQL 语句 CREATE VIEW 建立一个名为 v_stu_c 的视图，显示学生的学号、姓名、所学课程的课程编号，并使用视图查询学号为 0003 的学生的情况。

3. 基于 student 表、course 表和 score 表，建立一个名为 v_stu_s 的视图，视图中具有所有学生的学号、姓名、课程名称、分数。使用视图 v_stu_s 查询学号为 0001 的学生的所有课程与成绩，如图 1-9 所示。

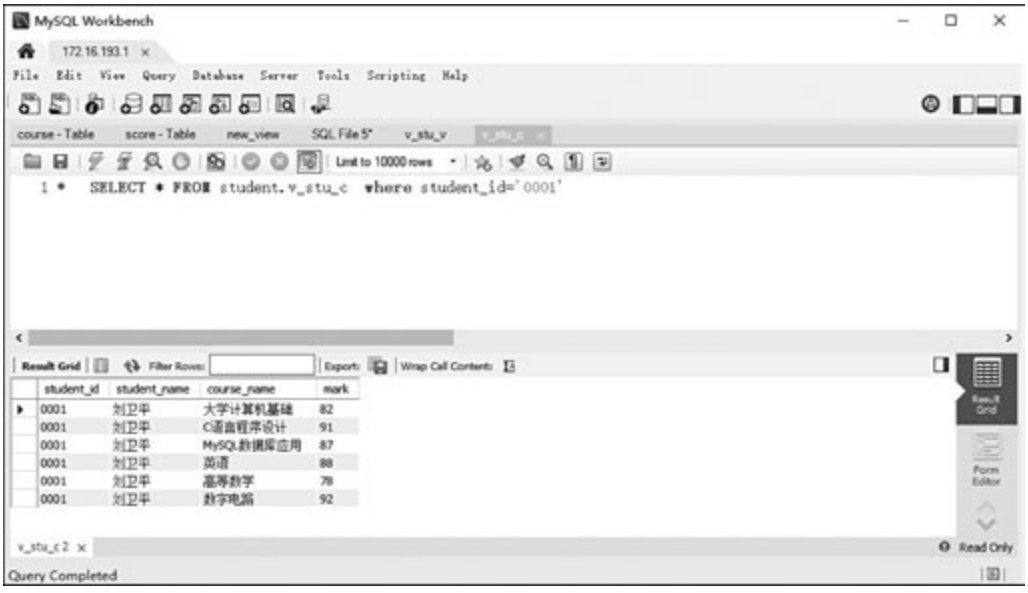


图 1-9　学号为 0001 的学生的视图信息

4. 使用 SQL 语句 ALTER VIEW 修改视图 v_stu_c，使之显示学生的学号、姓名以及每个学生所学课程的门数。

5. 使用 SQL 语句 ALTER VIEW 修改视图 v_stu，使其具有列名学号、姓名、性别。

```
ALTER VIEW v_stu AS SELECT id, name, gender FROM student
```

6. 使用 SQL 语句创建视图 v_score_avg，使其具有列名课程编号、课程名称、最高分、平均分，并按平均分从高到低排序。

```
CREATE VIEW v_score_avg AS SELECT course.id,course.name, max(score.mark), avg(score.mark)
avg_mark FROM course INNER JOIN score ON course.id = score.course_id GROUP BY course.id ORDER
BY avg_mark DESC;
```

7. 使用 SQL 语句查看视图 v_stu 的创建语句。

```
SHOW CREATE VIEW v_stu
```

8. 使用视图 v_stu 为 student 表添加一行数据：学号为 0015、姓名为陈婷、性别为女。

```
INSERT INTO v_stu VALUES('0015','陈婷','女')
```

9. 使用视图 v_stu 删除学号为 0015 的学生的记录。

10. 使用视图 v_stu_s 修改姓名为刘卫平的学生的高等数学的分数为 93。

11. 使用 SQL 语句 DROP VIEW 删除视图 v_stu_c 和 v_stu_s。

三、实验思考

1. 比较通过视图和基表操作表中数据的异同。

2. 可更新视图必须满足哪些条件?

3. 能否在视图上创建索引?

4. 在将视图所引用的基表删除后,视图还存在吗? 视图还能被查询吗?

实验 6　MySQL 程序设计

一、实验目的

1. 掌握 MySQL 的数据类型、变量、表达式等的概念和使用方法。

2. 掌握程序中的流程控制语句。

3. 掌握 MySQL 8.0 中内置函数的用法。

4. 掌握游标的概念和声明方法,以及使用游标进行数据的查询、遍历操作。

二、实验内容

1. 选择 student 数据库,新建查询编辑器,输入以下代码。

```
DELIMITER //
DROP PROCEDURE IF EXISTS example_01;
CREATE PROCEDURE example_01()
BEGIN
    DECLARE stu_name varchar(8);
    SELECT name INTO stu_name FROM student WHERE id = '0001';
    SELECT stu_name;
END
```

通过 CALL example_01()执行以上存储过程,观察显示的结果,与 student 表中的数据进行比较,stu_name 赋值的是 SELECT 结果集中的哪个数据?

2. 定义 int 型局部变量 score_max、score_min、score_sum,在 score 表中查找最高分、最低分和总分,分别赋给 score_max、score_min 和 score_sum 并显示。

```
DELIMITER //
CREATE PROCEDURE example_02()
BEGIN
    DECLARE score_max,score_min,score_sum int;
    SELECT max(mark),min(mark),sum(mark) INTO score_max,score_min,score_sum FROM score;
    SELECT score_max,score_min,score_sum;
END
```

3. 使用 SET 命令将查询结果集的记录数目赋值给 int 型局部变量 row。给下面代码中的画线处填上适当的内容,以完成上述操作。

```
DELIMITER //
CREATE PROCEDURE example_03()
BEGIN
    DECLARE _____;
    SET _____ = (SELECT count( * ) FROM score) ;
    _____ c      //显示 c 的值
END
```

4. 以下代码在 course 表中插入新记录:

```
DELIMITER //
CREATE PROCEDURE example_04()
BEGIN
    DECLARE cid,error_code int;
    INSERT INTO course(id,name,mark) VALUES('0099','VB 程序设计',3);
    SET cid = last_insert_id(),error_code = row_count();
    SELECT cid,error_code;
END
```

将该代码段连续执行两次,观察两次显示的信息及 course 表中数据的变化,为什么前后两次执行时显示的信息会不同?

5. 在 student 数据库的 student 表中以“gender”为分组条件,分别统计男生和女生的人数。

6. 在 score 表中,使用适当函数找出“高等数学”课程的最高分、最低分和平均分。

7. 定义一个 date 型局部变量 stu_date,以存储当前日期。计算 student 表中学生的年龄,并显示学生的姓名、年龄。在以下代码的画线部分填入适当内容,以实现上述功能。

```
DELIMITER //
CREATE PROCEDURE example_05()
BEGIN
    DECLARE _____ date;
    SET stu_date = _____;       //给 stu_date 赋值当前日期
    SELECT name,year(stu_date) - year(birthday) AS age FROM student;
END
```

8. 运行以下代码,写出运行结果。

```
DELIMITER //
CREATE PROCEDURE example_06()
BEGIN
    DECLARE a ,b int;
    SET a = 168;
    SET b = 73;
    SELECT a & b,a|b,a^b;
END
```

9. 在局部变量 stu_id 中存储了学号值。编写代码查询学号为 0001 的学生的各科平均成绩，如果平均成绩＞＝60，显示“你的平均成绩及格了，祝贺你!”，否则显示“你的平均成绩不及格”。

```
DELIMITER //
CREATE PROCEDURE example_07()
BEGIN
    DECLARE stu_id char(4);
    SET stu_id = '0001';
    IF (SELECT AVG(mark) FROM score WHERE student_id = stu_id)< 60 THEN
        SELECT  '你的平均成绩不及格';
    ELSE
        SELECT  '你的平均成绩及格了,祝贺你!';
    END IF;
END
```

10. 运行以下代码段，写出运行的结果。

```
DELIMITER //
CREATE PROCEDURE example_08()
BEGIN
    DECLARE counter int;
    SET counter = 1;
    WHILE counter < 10 DO
        SELECT 'counter 的值现在为: ' + convert(counter, CHAR);
        SET counter = counter + 1;
    END WHILE;
END
```

11. 查询 score 表。如果分数大于或等于 90，显示 A；如果分数大于或等于 80 小于 90，显示 B；如果分数大于或等于 70 小于 80，显示 C；如果分数大于或等于 60 小于 70，显示 D；其他显示 E。在以下代码的画线部分填入适当内容，完成上述功能。

```
SELECT student_id,mark,
CASE
 _________ mark > = 90 THEN 'A'
WHEN mark > = 80 AND mark < 90 _________'B'
WHEN mark > = 70 AND mark < 80 THEN 'C'
```

```
WHEN mark >= 60 AND mark < 70 THEN _________
ELSE 'E'
END AS level FROM score
```

12. 计算 score 表的分数列的平均值。如果分数小于 80,则分数增加其值的 5%；如果分数的最高值超过 95,则终止该操作。在以下代码的画线处填入适当的内容完成上述功能。

```
label1:LOOP
IF (SELECT _________(mark) FROM score)< 80 THEN
    UPDATE score SET mark = mark * 1.05;
END IF;
IF (SELECT MAX(mark) FROM score)>_________ THEN
    LEAVE label1;
END IF;
END LOOP label1;
```

13. 编写代码计算并显示 $n=1+2+3+\ldots+20$。

14. 编写代码计算并显示 1～100 的所有完全平方数。例如,81＝9^2,则称 81 为完全平方数。

15. 计算 1～100 的所有素数。

16. 在 student 数据库中使用游标查询数据。

(1) 声明一个游标 stu_cursor,要求返回 student 表中性别为“男”的学生记录。

(2) 打开 stu_cursor 游标。

(3) 获取并显示所有数据。

(4) 关闭该游标。

三、实验思考

1. MySQL 语言的运算符主要有哪些?

2. MySQL 中有哪些常用的流程控制语句?

3. 区分局部变量与会话变量的不同,思考会话变量的用处。

4. 什么函数能将字符串前面和尾部的空格去掉?

5. 使用什么语句可以打开游标? 在打开成功后,游标指针指向结果集的什么位置?

实验 7　函数与触发器

一、实验目的

1. 掌握通过 Workbench 和 SQL 语句 CREATE FUNCTION 创建函数的方法和

步骤。

2. 掌握使用 SQL 语句调用函数的方法。

3. 掌握通过 Workbench 和 SQL 语句 ALTER FUNCTION 修改函数的方法。

4. 掌握通过 Workbench 和 SQL 语句 DROP FUNCTION 删除函数的方法。

5. 掌握通过 Workbench 和 SQL 语句 CREATE TRIGGER 创建触发器的方法和步骤。

6. 了解触发器的执行机制。

7. 掌握使用 Workbench 或 SQL 语句修改和删除触发器的方法。

二、实验内容

1. 在查询编辑器中输入以下代码，创建一个函数 capitalize，将输入的小写字母转换为大写字母并返回。

```
SET GLOBAL log_bin_trust_function_creators = TRUE;
DELIMITER //
CREATE FUNCTION capitalize(x char(1))
RETURNS char(1)
BEGIN
    RETURN char(ASCII(x) - 32);
END
```

单击查询编辑器中的“执行查询”按钮，查看 student 数据库中是否存在 capitalize 函数。

执行 SQL 语句 SELECT capitalize('a')，观察函数的执行结果。

2. 输入以下代码，创建函数 avg_query，在执行时通过输入姓名可以查询该姓名对应的学生的各科平均成绩。

```
SET GLOBAL log_bin_trust_function_creators = TRUE;
DELIMITER //
CREATE FUNCTION avg_query(name char(8))
RETURNS int
BEGIN
    DECLARE s int;
    SELECT avg(mark) INTO s FROM score LEFT JOIN student ON score.student_id = student.id
WHERE student.name = name;
    return s;
END
```

执行 SQL 语句 SELECT avg_query('刘卫平')，观察函数的执行结果。

3. 使用 student 数据库中的 student 表、course 表、score 表。

(1) 创建一个函数 stu_score，根据输入的学号显示对应的姓名、课程名称、分数。

(2) 执行 stu_score 函数，查询学号为 0001 的学生的姓名、课程名称、分数。

(3) 使用 SQL 语句 SHOW CREATE FUNCTION stu_score 将显示 stu_score 函数的创建语句。

(4) 使用 SQL 语句 DROP FUNCITON stu_score 删除该函数。

4. 输入以下代码,复制 student 表并命名为 stu2,为 stu2 表创建一个触发器 stu_tr,当 stu2 表中插入一条记录时为该记录生成一个学号,该学号为学号列数据的最大值加 1。

(1) 复制 student 表并命名为 stu2。

```
CREATE TABLE stu2 (SELECT * FROM student);
```

(2) 为 stu2 表创建一个 INSERT 型触发器 stu_tr。

```
DELIMITER \\
CREATE TRIGGER stu_tr BEFORE INSERT
ON stu2 FOR EACH ROW
BEGIN
    DECLARE max char(4);
    SET max = (SELECT MAX(id) FROM stu2);
    SET max = max + 1;
    SET new.id = lpad(max,4,'0');
END
```

执行以上代码,查看 student 数据库中是否有 stu2 表,展开 stu2,查看其触发器项中是否有 stu_tr 触发器。

在查询编辑器的编辑窗口中输入以下代码:

```
INSERT INTO stu2(name,gender) VALUES('张婷','女');
```

运行以上代码,查看 stu2 表的变化情况,插入记录的学号是什么?

5. 为 score 表建立一个名为 insert_s_tr 的 INSERT 触发器,当用户向 score 表中插入记录时,如果插入的记录的课程编号在 course 表中不存在,提示用户不能插入记录,否则提示记录插入成功。在进行插入测试时,分别输入以下数据:

(1) 学号:0004,课程编号:0003,分数:76。

(2) 学号:0005,课程编号:0007,分数:69。

观察插入数据时的运行情况,说明为什么?

提示:当检测到插入的记录的课程编号在 course 表中不存在时,使用以下语句抛出异常,设置提示信息,停止插入。

```
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = '课程编号不存在,不能插入!';
```

6. 为 course 表创建一个名为 del_c_tr 的 DELETE 触发器,该触发器的作用是禁止删除 course 表中的记录。

7. 为 student 表创建一个名为 update_s_tr 的 UPDATE 触发器,该触发器的作用是禁止更新 student 表中“name”字段的内容。

提示:当检测到更新前后“name”字段发生变化时抛出异常,停止执行。

8. 使用 SQL 语句 DROP TRIGGER 删除 update_s_tr 触发器。