山西工程技术学院

**实验报告**

（ 2024 -2025 学年第2学期）

课程名称： 大型数据库系统开发

专业班级： 22计算机科学与技术一班

学 号： 2210708130

学生姓名： 郝泓毅

任课教师： 王晓霞

2025年 6 月

实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 数据库、表的创建与管理 | | | 指导教师 | 王晓霞 |
| 实验类型 | 验证型 | 实验学时 | **4** | 实验时间 | 2.19 |
| 1. **实验目的与要求**   1.掌握创建、修改数据库的方法及管理数据库的方法  2.掌握创建、修改表结构的方法及插入、更新和删除表数据的方法。 | | | | | |
| **二、实验环境**  **MySQL 8.0** | | | | | |
| **三、实验内容和步骤**  (1)利用MySQL8.0创建一个名为teaching数据库,初始大小为10MB,增长速度为10%,其他均采用默认设置。  (2)在查询编辑器中输入创建表的代码，分别创建student、course、score、teacher、 class、teach\_class这6张表  (3)分别对这6张表输入记录（每张表不少于5条记录）  (4)向student表插入、删除、修改一条记录  1. CREATE DATABASE teaching  ON PRIMARY  (      NAME = 'teaching\_data',      FILENAME = 'C:\teaching\_data.mdf',      SIZE = 10MB,      FILEGROWTH = 10%  )  LOG ON  (      NAME = 'teaching\_log',      FILENAME = 'C:\teaching\_log.ldf',      SIZE = 5MB,      FILEGROWTH = 5MB  );    2.  CREATE TABLE student (      sno CHAR(8) PRIMARY KEY,      sname VARCHAR(20) NOT NULL,      sex CHAR(2) CHECK (sex IN ('男', '女')),      birthday DATE,      classno CHAR(6),      point DECIMAL(5,2)  );  CREATE TABLE course (      courseno CHAR(6) PRIMARY KEY,      cname VARCHAR(30) NOT NULL,      credit SMALLINT,      period SMALLINT,      priorcourse CHAR(6)  );  CREATE TABLE score (      sno CHAR(8),      courseno CHAR(6),      daily DECIMAL(5,2),      final DECIMAL(5,2),      PRIMARY KEY (sno, courseno)  );  CREATE TABLE teacher (      tno CHAR(6) PRIMARY KEY,      tname VARCHAR(20) NOT NULL,      sex CHAR(2) CHECK (sex IN ('男', '女')),      prof CHAR(10),      depart VARCHAR(30)  )  CREATE TABLE class (      classno CHAR(6) PRIMARY KEY,      classname VARCHAR(30) NOT NULL,      monitor CHAR(8),      tno CHAR(6)  );  CREATE TABLE teach\_class (      tno CHAR(6),      classno CHAR(6),      courseno CHAR(6),      PRIMARY KEY (tno, classno, courseno)  );    3.  INSERT INTO student VALUES  ('180101', '张三', '男', '2000-05-15', '180501', 85.5),  ('180102', '李四', '女', '2000-08-20', '180501', 78.0),  ('180103', '王五', '男', '1999-11-10', '180502', 92.5),  ('180104', '赵六', '女', '2000-03-25', '180502', 88.0),  ('180105', '钱七', '男', '2000-07-12', '180503', 76.5);    INSERT INTO course VALUES  ('c05103', '数据库原理', 4, 64, NULL),  ('c05108', '数据结构', 4, 64, NULL),  ('c05109', '离散数学', 3, 48, NULL),  ('c05127', '操作系统', 4, 64, 'c05108'),  ('c05138', '计算机网络', 3, 48, 'c05108'),  ('c05222', '软件工程', 3, 48, 'c05103');    INSERT INTO score VALUES  ('180101', 'c05103', 85.0, 90.0),  ('180101', 'c05108', 78.0, 82.0),  ('180102', 'c05103', 90.0, 88.0),  ('180102', 'c05109', 82.0, 85.0),  ('180103', 'c05127', 76.0, 80.0);    INSERT INTO teacher VALUES  ('t05001', '刘老师', '男', '教授', '计算机学院'),  ('t05002', '张老师', '女', '副教授', '计算机学院'),  ('t05003', '李老师', '男', '讲师', '计算机学院'),  ('t05004', '王老师', '女', '助教', '数学学院'),  ('t05017', '陈老师', '男', '讲师', '计算机学院');    INSERT INTO class VALUES  ('180501', '计算机1801班', '180101', 't05001'),  ('180502', '计算机1802班', '180103', 't05002'),  ('180503', '计算机1803班', '180105', 't05003'),  ('180504', '数学1801班', NULL, 't05004'),  ('180505', '数学1802班', NULL, NULL);    INSERT INTO teach\_class VALUES  ('t05001', '180501', 'c05103'),  ('t05001', '180501', 'c05108'),  ('t05002', '180502', 'c05109'),  ('t05003', '180503', 'c05127'),  ('t05004', '180504', 'c05138');    4.  INSERT INTO student VALUES ('180106', '孙八', '男', '2000-09-18', '180503', 81.0);    DELETE FROM student WHERE sno = '180106';    UPDATE student SET point = 90.0 WHERE sno = '180101'; | | | | | |
| 1. **实验小结和思考**   **本次实验让我掌握了数据库和数据表的基本操作，学习了创建、修改和删除数据库及数据表的 SQL 语句。通过实际操作，我加深了对数据类型、主键、默认值等字段属性的理解，为后续数据库设计打下了基础。同时也意识到在设计数据表结构时应充分考虑数据的完整性与规范性。** | | | | | |
| 实验成绩 |  | 批阅日期 |  | 批阅人 |  |

实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 数据完整性的设置 | | | 指导教师 | 王晓霞 |
| 实验类型 | 验证型 | 实验学时 | **4** | 实验时间 | 3.5 |
| 1. **实验目的与要求**   创建外键、UNIKUE约束  创建check约束、规则  掌握数据完整性的类型和实现机制  (2)掌握约束、规则对象的创建和修改 | | | | | |
| **二、实验环境**  **MySQL 8.0** | | | | | |
| **三、实验内容和步骤**  (1)利用MySQL将teaching数据库中score表的courseno列设置为引用表course的外键；  (2)在teaching数据库中class表的classname创建UNIQUE约束；  (3)为teaching数据库中student表的birthday列创建check约束,规定学生的年龄在17~25之间,为course表的credit列创建check约束,规定学分的取值范围为1~6,删除check约束；  (4)为teaching数据库创建规则prof\_rule,规定教师职称取值只能为”助教”、“讲师”、“副教授”、“教授”，并将其绑定到teacher表的Prof列，删除创建的规则。  1.  ALTER TABLE score  ADD CONSTRAINT fk\_score\_courseno  FOREIGN KEY (courseno) REFERENCES course(courseno);    2.  ALTER TABLE class  ADD CONSTRAINT unique\_classname  UNIQUE (classname);    3.  CREATE TRIGGER trg\_check\_birthday  BEFORE INSERT ON student  FOR EACH ROW  BEGIN  IF TIMESTAMPDIFF(YEAR, NEW.birthday, CURDATE()) < 17 OR  TIMESTAMPDIFF(YEAR, NEW.birthday, CURDATE()) > 25 THEN  SIGNAL SQLSTATE '45000'  SET MESSAGE\_TEXT = '年龄必须在17到25岁之间';  END IF;  END; (MySQL 的限制：在 CHECK 约束中 不允许使用非确定性函数,故改用触发器实验)    插入合法数据，成功插入    非法数据，报错  4.  ALTER TABLE teacher  MODIFY COLUMN prof ENUM('助教', '讲师', '副教授', '教授');    插入合法数据    插入非法数据报则报错 | | | | | |
| 1. **实验小结和思考**   **通过本次实验，我学会了如何设置实体完整性、参照完整性以及用户自定义完整性等约束条件，有效提高数据的准确性和一致性。同时，通过对表中数据的增删改查操作，加深了对 SQL 基本语法和数据管理操作的理解，提升了数据库的实用能力。** | | | | | |
| 实验成绩 |  | 批阅日期 |  | 批阅人 |  |

实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 单表数据检索 | | | 指导教师 | 王晓霞 |
| 实验类型 | 验证型 | 实验学时 | **4** | 实验时间 | 3.19 |
| 1. **实验目的与要求**   (1)会使用SELECT语句对单表进行查询,会使用LIKE进行模糊查询,会用WHERE加学号、班级进行查询  (2)会用聚合函数COUNT进行统计、MAX求最大值、MIN求最小值、TOP5求成绩前5名学生信息，会用ORDER进行排序，GROUP BY按学号分组。  (2)掌握约束、规则对象的创建和修改  (1)掌握SELECT各个字句的功能和检索数据的方法  (2)掌握WHERE子句中LIKE、IN、BETWEEN、IS等逻辑运算符的使用  (3)掌握聚合函数的使用 | | | | | |
| **二、实验环境**  **MySQL 8.0** | | | | | |
| **三、实验内容和步骤**  (1)查询所有课程的课程编号、课程名和学分，查询160501班所有学生的基本信息  (2)查询student表中所有年龄大于20岁的男生的姓名和年龄  (3)查询计算机学院教师的专业名称  (4)查询每名学生的学号、选修课程数目、总成绩，并将查询结果存放到生成的“学生选课统计表”  (5)查询student表中所有学生的基本信息,查询结果按班级号classno升序排序,同一班级中的学生按入学成绩point降序排列  (6)查询各班学生的人数(按班级分组),查询各班期末成绩的最高分和最低分  (7)查询教授两门及以上课程的教师编号、课程编号和任课班级  (8)查询课程编号以c05开头,被三名及以上学生选修,且期末成绩的平均分高于75分的课程号、选修人数和期末成绩平均分，并按平均分降序排序。  1.  SELECT courseno, cname, credit FROM course;    SELECT \* FROM student WHERE classno = '160501';    插入数据后重新查询：    2.  SELECT sname, TIMESTAMPDIFF(YEAR, birthday, CURDATE()) AS age  FROM student WHERE sex = '男' AND TIMESTAMPDIFF(YEAR, birthday, CURDATE()) > 20;    3.  SELECT DISTINCT depart FROM teacher WHERE depart = '计算机学院';    4.  SELECT s.sno, COUNT(\*) AS course\_count, SUM(final) AS total\_score  FROM score s  GROUP BY s.sno;    5.  SELECT \* FROM student ORDER BY classno ASC, point DESC;    6.  SELECT classno, COUNT(\*) AS num\_students FROM student GROUP BY classno;    SELECT classno,  MAX(sc.final) AS max\_score,  MIN(sc.final) AS min\_score  FROM student s  JOIN score sc ON s.sno = sc.sno  GROUP BY classno;    7.  SELECT t.tno, t.courseno, t.classno  FROM teach\_class t  WHERE t.tno IN (  SELECT tno  FROM teach\_class  GROUP BY tno  HAVING COUNT(DISTINCT courseno) >= 2  )  ORDER BY t.tno, t.courseno, t.classno;    8.  SELECT  s.courseno AS 课程号,  COUNT(\*) AS 选修人数,  AVG(s.final) AS 期末成绩平均分  FROM  score s  WHERE  s.courseno LIKE 'c05%'  GROUP BY  s.courseno  HAVING  COUNT(\*) >= 3  AND AVG(s.final) > 75  ORDER BY  AVG(s.final) DESC; | | | | | |
| 1. **实验小结和思考**   **本次实验使我掌握了对单个数据表进行各种查询的能力，包括条件查询、模糊查询、聚合函数、分组排序等操作。通过练习不同的查询方式，我认识到 SQL 查询的灵活性与强大功能，也增强了我分析数据和提取有效信息的能力。** | | | | | |
| 实验成绩 |  | 批阅日期 |  | 批阅人 |  |

实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 多表查询与子查询 | | | 指导教师 | 王晓霞 |
| 实验类型 | 验证型 | 实验学时 | **4** | 实验时间 | 4.2 |
| 1. **实验目的与要求**   (1)会使用JOIN ON连接两个及以上表  (2)会在SELECT查询语句中再嵌套一个SELECT语句  (3)会使用游标对查询结果集进行处理(查询结果集必须是大于等于两条记录)  (4)掌握多表连接查询及多表连接的各种方法,包括内连接、外连接  (5)掌握子查询的方法，包括相关子查询和不相关子查询  (6)掌握游标处理结果集的基本过程 | | | | | |
| **二、实验环境**  **MySQL 8.0** | | | | | |
| 三、实验内容和步骤  (1)查询所有班级的期末成绩平均分,并按照平均分降序排序  (2)查询教师基本信息和教授课程信息,其中包括未分配课程的教师信息  (3)查询两门及以上课程的期末成绩超过80分的学生姓名及其平均成绩  (4)查询没有被任何学生选修的课程编号、课程名称和学分（子查询）  (5)查询入学成绩最高的学生学号、姓名和入学成绩（子查询）  (6)查询同时教授c05127号和c05109号课程的教师信息(子查询)  (7)查询每门课程的课程号、课程名和选修该课程的学生人数，并按所选人数升序排序  (8)使用游标输出学生姓名、选修课程名称和期末考试成绩  1.  SELECT class.classno, AVG(score.final) AS avg\_score  FROM class           JOIN teach\_class ON class.classno = teach\_class.classno           JOIN score ON teach\_class.courseno = score.courseno  GROUP BY class.classno  ORDER BY avg\_score DESC;    2.  SELECT teacher.\*, course.courseno, course.cname  FROM teacher           LEFT JOIN teach\_class ON teacher.tno = teach\_class.tno           LEFT JOIN course ON teach\_class.courseno = course.courseno;    3.、  SELECT student.sname, AVG(score.final) AS avg\_score  FROM student           JOIN score ON student.sno = score.sno  WHERE score.final > 80  GROUP BY student.sname  HAVING COUNT(score.courseno) >= 2;    4.  SELECT courseno, cname, credit  FROM course  WHERE courseno NOT IN (SELECT DISTINCT courseno FROM score);    5.  SELECT courseno, cname, credit  FROM course  WHERE courseno NOT IN (SELECT DISTINCT courseno FROM score);    6.  SELECT \*  FROM teacher  WHERE tno IN (      SELECT tno      FROM teach\_class      WHERE courseno IN ('c05103', 'c05108')      GROUP BY tno      HAVING COUNT(DISTINCT courseno) = 2  );    7.  SELECT course.courseno, course.cname, COUNT(score.sno) AS student\_count  FROM course           LEFT JOIN score ON course.courseno = score.courseno  GROUP BY course.courseno, course.cname  ORDER BY student\_count ASC;    8.  CREATE PROCEDURE ShowStudentCourseScores()  BEGIN      -- 声明变量      DECLARE sname VARCHAR(50);      DECLARE cname VARCHAR(50);      DECLARE final\_score DECIMAL(5, 2);      -- 声明游标      DECLARE student\_cursor CURSOR FOR      SELECT student.sname, course.cname, score.final      FROM student      JOIN score ON student.sno = score.sno      JOIN course ON score.courseno = course.courseno;      -- 声明处理游标结束的条件      DECLARE done INT DEFAULT FALSE;      DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;      -- 打开游标      OPEN student\_cursor;      -- 循环遍历游标      read\_loop: LOOP          -- 从游标中获取数据          FETCH student\_cursor INTO sname, cname, final\_score;          -- 检查是否到达游标末尾          IF done THEN              LEAVE read\_loop;          END IF;          -- 输出信息          SELECT CONCAT('学生姓名: ', sname, ', 选修课程名称: ', cname, ', 期末考试成绩: ', final\_score) AS info;      END LOOP;      -- 关闭游标      CLOSE student\_cursor;  END //  DELIMITER ; | | | | | |
| 1. **实验小结和思考**   **本实验让我系统学习了多表连接查询、嵌套查询等内容，初步掌握了内连接、外连接的使用方法。通过练习多个数据表之间的逻辑关系和查询语句的编写，理解了数据间的关联性，为复杂业务逻辑的数据提取奠定了技术基础。** | | | | | |
| 实验成绩 |  | 批阅日期 |  | 批阅人 |  |

实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 索引和视图 | | | 指导教师 | 王晓霞 |
| 实验类型 | 验证型 | 实验学时 | **4** | 实验时间 | 4.16 |
| 1. **实验目的与要求**   (1)掌握创建、维护索引的方法  (2)掌握创建、修改视图的方法，并通过视图插入、修改、删除基本表中数据的方法 | | | | | |
| **二、实验环境**  **MySQL 8.0** | | | | | |
| **三、实验内容和步骤**  (1)在teaching数据库的student表的classno字段创建非聚集非唯一索引UC\_classno  (2)在teaching数据库中的teacher表的tname列上创建非聚集唯一索引UQ\_name ,若该索引已存在,则删除后重建  (3)在course表上创建视图v\_course\_avg,查询每门课程的课程号、课程名及选修该课程的学生的期末成绩平均分，并且按平均分降序排序  (4)修改v\_course\_avg视图的定义,添加WITH CHECK OPTION选项  (5)在teaching 数据库中创建视图v\_teacher\_course，包含教师编号、教师姓名、职称、课程号、课程名和任课班级，通过视图v\_teacher\_course将教师编号为t05017的教师职称更改为“副教授”  (6)用SQL语句删除创建的索引和视图  1.  CREATE INDEX UC\_classno ON student (classno);  GO    2.  SHOW INDEX FROM teacher WHERE Key\_name = 'UQ\_name';  DROP INDEX UQ\_name ON teacher;    3.  CREATE VIEW v\_course\_avg  AS  SELECT course.courseno, course.cname, AVG(score.final) AS avg\_score  FROM course           JOIN score ON course.courseno = score.courseno  GROUP BY course.courseno, course.cname  ORDER BY avg\_score DESC;  GO      4.  ALTER VIEW v\_course\_avg  AS  SELECT course.courseno, course.cname, AVG(score.final) AS avg\_score  FROM course           JOIN score ON course.courseno = score.courseno  GROUP BY course.courseno, course.cname  ORDER BY avg\_score DESC  WITH CHECK OPTION;  GO    5.  CREATE OR REPLACE VIEW v\_course\_avg AS  SELECT course.courseno, course.cname, AVG(score.final) AS avg\_score  FROM course  JOIN score ON course.courseno = score.courseno  GROUP BY course.courseno, course.cname  ORDER BY avg\_score DESC;    UPDATE v\_teacher\_course  SET prof = '副教授'  WHERE tno = 't05017';  GO    6.  DROP INDEX UC\_classno ON student;  DROP INDEX UQ\_name ON teacher;  DROP VIEW v\_course\_avg;  DROP VIEW v\_teacher\_course;  GO | | | | | |
| 1. **实验小结和思考**   **在本次实验中，我学习了如何创建和管理视图与索引。视图简化了复杂查询，提升了数据的安全性；索引则提高了查询效率。实验中我体会到合理使用视图和索引对于大型数据库性能优化的重要作用，并加深了对数据库抽象和效率管理的理解。** | | | | | |
| 实验成绩 |  | 批阅日期 |  | 批阅人 |  |

实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 存储过程和触发器 | | | 指导教师 | 王晓霞 |
| 实验类型 | 验证型 | 实验学时 | **4** | 实验时间 | 4.30 |
| 1. **实验目的与要求**   (1)分别会创建没有参数的存储过程、带有输入参数和输出参数的存储过程  (2)会创建AFTER触发器,会回滚事务,会删除触发器 | | | | | |
| **二、实验环境**  **MySQL 8.0** | | | | | |
| **三、实验内容和步骤**  (1)创建一个存储过程ProcNum，查询每个班级中学生的人数，按班级号升序排序  (2)利用SQL语句创建一个带有参数的存储过程ProcInsert，向score表插入一条选课记录，并查询该学生的姓名、选修的所有课程名称、平时成绩和期末成绩  (3)利用SQL语句创建一个存储过程ProcAvg，查询指定课程的平均分。班级号和课程名称由输入参数确定，计算出的平均分通过输出参数返回，若该存储过程已存在，则删除后重建  (4)创建一个AFTER触发器trigsex，当插入或修改student表中性别字段sex时，检查数据是否只为“男”或“女”  (5)利用SQL语句创建一个AFTER触发器trigforeign，当向score表中插入或修改记录时，如果插入或修改的数据与student表中数据不匹配，即没有对应的学号存在，则将此记录删除  (6)利用SQL语句创建一个AFTER触发器trigclassname，当向class表中插入或修改数据时，如果出现班级名称重复则回滚事务，若该触发器已存在，则删除后重建  1.  DELIMITER $$  DROP PROCEDURE IF EXISTS ProcNum $$  CREATE PROCEDURE ProcNum()  BEGIN      SELECT classno, COUNT(\*) AS student\_count      FROM student      GROUP BY classno      ORDER BY classno ASC;  END $$  DELIMITER ;    2.  DELIMITER $$  DROP PROCEDURE IF EXISTS ProcInsert $$  CREATE PROCEDURE ProcInsert(      IN p\_sno CHAR(8),      IN p\_courseno CHAR(6),      IN p\_daily DECIMAL(5,2),      IN p\_final DECIMAL(5,2)  )  BEGIN      INSERT INTO score (sno, courseno, daily, final)      VALUES (p\_sno, p\_courseno, p\_daily, p\_final);      SELECT s.sname, c.cname, sc.daily, sc.final      FROM student s      JOIN score sc ON s.sno = sc.sno      JOIN course c ON sc.courseno = c.courseno      WHERE s.sno = p\_sno;  END $$  DELIMITER ;    3.  DELIMITER $$  DROP PROCEDURE IF EXISTS ProcAvg $$  CREATE PROCEDURE ProcAvg(      IN p\_classno CHAR(6),      IN p\_cname VARCHAR(40),      OUT p\_avg DECIMAL(5,2)  )  BEGIN      SELECT AVG(s.final)      INTO p\_avg      FROM score s      JOIN student st ON s.sno = st.sno      JOIN course c ON s.courseno = c.courseno      WHERE st.classno = p\_classno AND c.cname = p\_cname;  END $$  DELIMITER ;    4.  DELIMITER $$  DROP TRIGGER IF EXISTS trigsex\_insert $$  CREATE TRIGGER trigsex\_insert  BEFORE INSERT ON student  FOR EACH ROW  BEGIN      IF NEW.sex NOT IN ('男', '女') THEN          SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = '性别只能是 男 或 女';      END IF;  END $$  DROP TRIGGER IF EXISTS trigsex\_update $$  CREATE TRIGGER trigsex\_update  BEFORE UPDATE ON student  FOR EACH ROW  BEGIN      IF NEW.sex NOT IN ('男', '女') THEN          SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = '性别只能是 男 或 女';      END IF;  END $$  DELIMITER ;    5.  DELIMITER $$  DROP TRIGGER IF EXISTS trigforeign\_insert $$  CREATE TRIGGER trigforeign\_insert  AFTER INSERT ON score  FOR EACH ROW  BEGIN      IF NOT EXISTS (SELECT 1 FROM student WHERE sno = NEW.sno) THEN          DELETE FROM score WHERE sno = NEW.sno AND courseno = NEW.courseno;      END IF;  END $$  DROP TRIGGER IF EXISTS trigforeign\_update $$  CREATE TRIGGER trigforeign\_update  AFTER UPDATE ON score  FOR EACH ROW  BEGIN      IF NOT EXISTS (SELECT 1 FROM student WHERE sno = NEW.sno) THEN          DELETE FROM score WHERE sno = NEW.sno AND courseno = NEW.courseno;      END IF;  END $$  DELIMITER ;  6.  CREATE TRIGGER trigclassname  AFTER INSERT OR UPDATE ON class  FOR EACH ROW  BEGIN      DECLARE classNameCount INT;        SELECT COUNT(\*)      INTO classNameCount      FROM class      WHERE classname = NEW.classname      AND classid != NEW.classid;        IF classNameCount > 0 THEN          SIGNAL SQLSTATE '45000'          SET MESSAGE\_TEXT = '班级名称不能重复';      END IF;  END$$ | | | | | |
| 1. **实验小结和思考**   **通过本次实验，我系统学习了存储过程和触发器的基本用法。掌握了如何创建不带参数、带输入参数以及带输出参数的存储过程，并能利用 SQL 编写逻辑实现数据插入、查询与计算等操作。尤其是在使用输出参数获取计算结果（如平均分）时，加深了对过程式 SQL 编程的理解。** | | | | | |
| 实验成绩 |  | 批阅日期 |  | 批阅人 |  |