《数据库系统》课内实验报告

班级：计算机 2201 姓名：余凯越 学号：2223515348

目录

1 实验环境配置

这个实验基于 VirtualBox 虚拟化平台，在本地环境下配置并安装了 openEuler 20.03-LTS 操作系统，数据库采用 openGauss 1.1.0 版本。为了提升实验效率和开发体验，在标准实验环境配置指导书的基础上，额外配置了以下环境：

1.1 GCC 编译器升级

openEuler 20.03-LTS 默认搭载的 libstdc++ 版本为 3.4.24，该版本无法满足 Visual Studio Code SSH Remote 插件的远程连接需求（要求 libstdc++ 版本 ≥3.4.25）。由于 openEuler 20.03 默认软件仓库中的 GCC 版本较低，本实验采用手动编译的方式升级 GCC 编译器：

- 将 GCC 从默认版本升级至 10.3.0
- 同步更新 libstdc++ 版本至 3.4.28
- 实现 VSCode 远程连接，利用其丰富的插件生态系统

```
[root@db1 omm]# gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/local/gcc-10.3.0/libexec/gcc/x86_64-pc-linux-gnu/10.3.0/lto-wrapper
Target: x86_64-pc-linux-gnu
Configured with: ./configure --prefix=/usr/local/gcc-10.3.0 --enable-languages=c,c++ --disable-mult
ilib
Thread model: posix
Supported LTO compression algorithms: zlib
gcc version 10.3.0 (GCC)
[root@db1 omm]#
```

图 1: 更新 GCC 版本

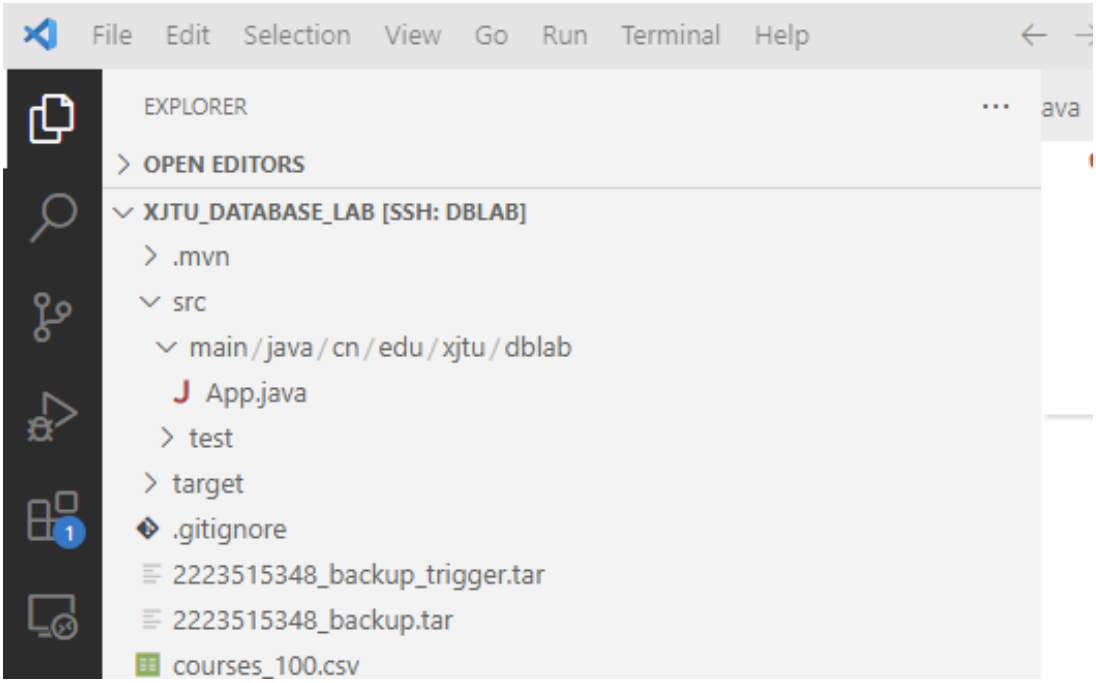

图 2: VSCODE 连接

1.2 Java 开发环境

为满足实验中使用 JDBC 导入数据的需求，配置了完整的 Java 开发环境：

- **Java 运行时环境**：安装 openEuler 20.03-LTS 官方仓库中的 OpenJDK（版本 1.8.0_242）
- **依赖管理工具**：手动安装 Apache Maven，实现高效的 Java 包管理
- **数据库连接组件**：导入 openGauss-JDBC 依赖，实现与数据库的连接
- **数据处理工具**：集成 OpenCSV 依赖包，简化 CSV 文件的数据导入流程

```
[root@db1 xjtu_database_lab]# mvn -v
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: /opt/maven
Java version: 1.8.0_242, vendor: Huawei Technologies Co., Ltd, runtime: /usr/lib/jvm/java-1.8.0-ope
njdk-1.8.0.242.b08-1.h5.oe1.x86_64/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.19.90-2003.4.0.0036.oe1.x86_64", arch: "amd64", family: "unix"
[root@db1 xjtu_database_lab]#
```

图 3: Maven 环境

```
    <dependency>
      <groupId>org.opengauss</groupId>
      <artifactId>opengauss-jdbc</artifactId>
      <version>5.0.0-og</version>
    </dependency>
    <dependency>
      <groupId>com.opencsv</groupId>
      <artifactId>opencsv</artifactId>
      <version>5.7.1</version>
    </dependency>
  </dependencies>
```

图 4: Maven 导入相关依赖

1.3 项目目录结构

```
XJTU_DATABASE_LAB/
├── .mvn/
├── src/
│   ├── main/java/cn/edu/xjtu/dblab/
│   │   └── App.java
│   └── test/java/cn/edu/xjtu/dblab/
│       └── AppTest.java
├── target/
└── .gitignore
```

```
├── pom.xml
├── README.md
├── 2024 国赛.pdf
├── 2223515348_backup.tar
├── 2223515348_backup_trigger.tar
├── MYDB_backup.tar
├── parsepdf.py
├── gencdata.py
├── genscdata.py
├── request.py
├── students_name.csv
├── students_1000.csv
├── students_5000.csv
├── students_data.csv
├── xjtu_courses.csv
├── courses_100.csv
├── courses_1000.csv
├── stu_course_20000.csv
├── stu_course_200000.csv
└── 实验报告.pdf
```

Java 项目文件 如下:

- `src/main/java/cn/edu/xjtu/dblab/App.java` — 主程序入口
- `src/test/java/cn/edu/xjtu/dblab/AppTest.java` — 单元测试
- `pom.xml` — Maven 项目配置文件
- `target/` — Maven 编译输出目录

数据库备份文件 如下:

- `2223515348_backup.tar` — 未完成触发器实验的数据库备份
- `2223515348_backup_trigger.tar` — 完成所有实验后的完整备份

数据处理脚本 如下:

- `parsepdf.py` — 从 PDF 文件中提取学生姓名
- `gencdata.py` — 处理原始数据生成符合格式的插入数据
- `genscdata.py` — 生成学生选课关系数据
- `request.py` — 爬取西安交通大学教务网站课程信息

数据文件 如下:

- `2024 国赛.pdf` — 学生姓名数据源
- `students_name.csv` — 从 PDF 提取的姓名表格
- `students_1000.csv` — 约 1000 条学生记录
- `students_5000.csv` — 约 5000 条学生记录
- `xjtu_courses.csv` — 爬取的原始课程信息
- `courses_100.csv` — 约 100 门课程数据
- `courses_1000.csv` — 约 1000 门课程数据
- `stu_course_20000.csv` — 约 20000 条选课记录
- `stu_course_200000.csv` — 约 200000 条选课记录

实验报告.pdf 即本篇实验报告。

2 创建数据库和数据表

在 openGauss 中创建 MYDB 数据库，并在 MYDB 中建立学生、课程、选课三个表。
根据命名规则，构建数据表如下:

表名	表结构
S348	S#，SNAME，SEX，BDATE，HEIGHT，DORM
C348	C#，CNAME，PERIOD，CREDIT，TEACHER
SC348	S#，C#，GRADE（其中 S#、C# 均为外键）

表 1: 数据表结构概览

2.1 学生表属性设计

主键设计

S#（学号）: 参照数据格式，应为 8 位定长的纯数字，因此类型设计为 `CHAR(8)`。

基本属性设计

SNAME（学生姓名） 设计为 `VARCHAR(60)`。考虑到姓名长度的可变性，同时为支持少数民族和外国人的姓名，按照常用汉字 3 字节编码计算，可支持最多 20 个汉字或 60 个 ASCII 字符。

SEX（学生性别） 该字段值限定为' 男'、' 女' 二选一，故设计为 `CHAR(3)` 定长字符串类型。

BDATE（出生日期） 采用标准的 `DATE` 类型存储。

HEIGHT（身高） 根据数据插入格式要求，需保留两位小数精度，故设计为 `DEC(3,2)` 类型。

DORM（宿舍） 宿舍编号格式为 {方向}{楼号} 舍 {层号}{门号}，根据我设计的具体约束，设计为 `VARCHAR(20)` 类型，具体如下：

- 方向：’ 东’ 或’ 西’
- 楼号：1-30
- 层号：2-7
- 门号：01-40

字段名	数据类型	约束	说明
S#	CHAR(8)	主键	8 位定长学号
SNAME	VARCHAR(60)	非空	支持中外文姓名
SEX	CHAR(3)	-	CHECK ’ 男’ 或’ 女’
BDATE	DATE	非空	出生日期
HEIGHT	DECIMAL(3,2)	-	身高，保留两位小数
DORM	VARCHAR(20)	-	宿舍编号

表 2: 学生表（S348）字段详细设计

```
mydb=> CREATE TABLE S348 (
S# CHAR(8) PRIMARY KEY,
SNAME VARCHAR(60) NOT NULL,
SEX CHAR(3) CHECK (SEX IN ('男', '女')),
BDATE DATE NOT NULL,
HEIGHT DEC(3,2),
DORM VARCHAR(20))
;
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "s348_pkey" for table "s348"
CREATE TABLE
```

图 5: 创建学生表

2.2 课程表属性设计

主键设计

C#（课程号）：参照数据格式，格式为专业简写-两位编号，考虑到专业简写可能并非定长，因此类型设计为 `VARCHAR(10)`。

基本属性设计

CNAME（课程名） 设计为 `VARCHAR(150)`。按照常用汉字 3 字节编码计算，可以支持最多 50 个汉字和 150 个 ASCII 字符。

PERIOD（课时） 一般来说，课时不会超过 SMALLINT 的范围（-32,768 到 32,767），故设计为 `SMALLINT` 类型，且检查课时一定为正数。

CREDIT（学分数） 考虑到存在 X.5 学分的课程，设计为 `DEC(3,1)` 类型，并检查学分一定为正。

TEACHER（授课教师） 与学生表的学生姓名相同，设计为 `VARCHAR(60)` 类型。

字段名	数据类型	约束	说明
C#	VARCHAR(10)	主键	课程编号
CNAME	VARCHAR(150)	非空	课程名
PERIOD	SMALLINT	-	CHECK PERIOD > 0
CREDIT	DEC(3,1)	-	CHECK CREDIT > 0
TEACHER	VARCHAR(60)	-	授课教师

表 3: 课程表（C348）字段详细设计

```
mydb=> CREATE TABLE C348 (
C# VARCHAR(10) PRIMARY KEY,
CNAME VARCHAR(150) NOT NULL,
PERIOD SMALLINT CHECK (PERIOD > 0),
CREDIT DEC(3, 1) CHECK (CREDIT > 0),
TEACHER VARCHAR(60));
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "c348_pkey" for table "c348"
CREATE TABLE
```

图 6: 创建课程表

2.3 选课表属性设计

主键设计

S#（学号）：关联到学生表的外键，类型设计为 `CHAR(8)`。由于学生可以被删除，被删除后需要级联删除对应的选课记录，所以删除规则设计为 CASCADE。

C#（课程号）：关联到课程表的外键，类型设计为 `VARCHAR(10)`。由于已开课的课程不应该被删除，所以删除规则设计为 RESTRICT。

以上两个属性共同作为选课表的主键。

基本属性设计

GRADE（成绩） 设计为 `DEC(4,1)`，表示学生的成绩，可为空且默认为空。

字段名	数据类型	约束	说明
S#	CHAR(8)	主键/外键	学生学号，外键引用学生表，级联删除（CASCADE）
C#	VARCHAR(10)	主键/外键	课程编号，外键引用课程表，限制删除（RESTRICT）
GRADE	DEC(4,1)	-	学生成绩，允许为空，默认值为 NULL

表 4: 选课表（SC348）字段详细设计

```
mydb=> CREATE TABLE SC348(
mydb(> S# CHAR(8),
mydb(> C# VARCHAR(10),
mydb(> GRADE DEC(4,1) DEFAULT NULL,
mydb(> PRIMARY KEY (S#, C#),
mydb(> FOREIGN KEY (S#) REFERENCES S348(S#) ON DELETE CASCADE,
mydb(> FOREIGN KEY (C#) REFERENCES C348(C#) ON DELETE RESTRICT);
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "sc348_pkey" for table "sc348"
CREATE TABLE
```

图 7: 创建选课表

3 将数据插入数据表

3.1 录入数据

```
mydb=> INSERT INTO S348 VALUES('01032010', '王涛', '男', '2003-4-5', 1.72, '东6舍221');
INSERT 0 1
mydb=> INSERT INTO S348 VALUES('01032023', '孙文', '男', '2004-6-10', 1.80, '东6舍221');
INSERT 0 1
mydb=> INSERT INTO S348 VALUES('01032001', '张晓梅', '女', '2003-11-17', 1.58, '东1舍312');
INSERT 0 1
mydb=> INSERT INTO S348 VALUES('01032005', '刘静', '女', '2003-1-10', 1.63, '东1舍312');
INSERT 0 1
mydb=> INSERT INTO S348 VALUES('01032112', '董喆', '男', '2003-2-20', 1.71, '东6舍221');
INSERT 0 1
mydb=> INSERT INTO S348 VALUES('03031011', '王倩', '女', '2004-12-20', 1.66, '东2舍104');
INSERT 0 1
mydb=> INSERT INTO S348 VALUES('03031014', '赵思扬', '男', '2002-6-6', 1.85, '东18舍421');
INSERT 0 1
mydb=> INSERT INTO S348 VALUES('03031051', '周剑', '男', '2002-5-8', 1.68, '东18舍422');
INSERT 0 1
mydb=> INSERT INTO S348 VALUES('03031009', '田菲', '女', '2003-8-11', 1.60, '东2舍104');
INSERT 0 1
mydb=> INSERT INTO S348 VALUES('03031033', '蔡明明', '男', '2003-3-12', 1.75, '东18舍423');
INSERT 0 1
mydb=> INSERT INTO S348 VALUES('03031056', '曹子衿', '女', '2005-12-15', 1.65, '东2舍305');
INSERT 0 1
mydb=>
```

图 8: 录入学生数据

```
mydb=> INSERT INTO C348 VALUES('CS-01', '数据结构', 60, 3, '张军');
INSERT 0 1
mydb=>  INSERT INTO C348 VALUES('CS-02', '计算机组成原理', 80, 4, '王亚伟');
INSERT 0 1
mydb=>  INSERT INTO C348 VALUES('CS-04', '人工智能', 40, 2, '李蕾');
INSERT 0 1
mydb=>  INSERT INTO C348 VALUES('CS-05', '深度学习', 40, 2, '崔昀');
INSERT 0 1
mydb=>  INSERT INTO C348 VALUES('EE-01', '信号与系统', 60, 3, '张明');
INSERT 0 1
mydb=>  INSERT INTO C348 VALUES('EE-02', '数字逻辑电路', 100, 5, '胡海东');
INSERT 0 1
mydb=>  INSERT INTO C348 VALUES('EE-03', '光电子学与光子学', 40, 2, '石韬');
INSERT 0 1
mydb=>
```

图 9: 录入课程数据

```
mydb=> INSERT INTO SC348 VALUES('01032010', 'CS-01', 82.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032010', 'CS-02', 91.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032010', 'CS-04', 83.5);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032001', 'CS-01', 77.5);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032001', 'CS-02', 85.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032001', 'CS-04', 83.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032005', 'CS-01', 62.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032005', 'CS-02', 77.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032005', 'CS-04', 82.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032023', 'CS-01', 55.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032023', 'CS-02', 81.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032023', 'CS-04', 76.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032112', 'CS-01', 88.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032112', 'CS-02', 91.5);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032112', 'CS-04', 86.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('01032112', 'CS-05');
INSERT 0 1
```

图 10: 录入选课数据（CS 课程）

```
mydb=> INSERT INTO SC348 VALUES('03031033', 'EE-01', 93.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('03031033', 'EE-02', 89.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('03031009', 'EE-01', 88.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('03031009', 'EE-02', 78.5);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('03031011', 'EE-01', 91.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('03031011', 'EE-02', 86.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('03031051', 'EE-01', 78.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('03031051', 'EE-02', 58.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('03031014', 'EE-01', 79.0);
INSERT 0 1
mydb=> INSERT INTO SC348 VALUES('03031014', 'EE-02', 71.0);
INSERT 0 1
mydb=>
```

图 11: 录入选课数据（EE 课程）

3.2 各表及属性信息

```
mydb=> SELECT table_catalog, table_schema, table_name, table_type
FROM information_schema.tables
WHERE table_name IN ('s348', 'c348', 'sc348')
ORDER BY table_name;
 table_catalog | table_schema | table_name | table_type
---------------+--------------+------------+------------
 mydb          | joe          | c348       | BASE TABLE
 mydb          | joe          | s348       | BASE TABLE
 mydb          | joe          | sc348      | BASE TABLE
(3 rows)
```

图 12: 各表信息

```
mydb=> SELECT table_name AS NAME,
       data_type AS TYPE,
       character_maximum_length AS MAXCHAR,
       numeric_scale AS SCALE,
       numeric_precision AS PRECISION,
       is_nullable AS NULLABLE,
       column_default AS DEFAULT
FROM information_schema.columns
WHERE table_name IN ('s348', 'c348', 'sc348')
ORDER BY table_name ASC;
 name  |            type             | maxchar | scale | precision | nullable |    default
-------+-----------------------------+---------+-------+-----------+----------+---------------
 c348  | character varying           |      10 |       |           | NO       |
 c348  | character varying           |     150 |       |           | NO       |
 c348  | smallint                    |         |     0 |        16 | YES      |
 c348  | numeric                     |         |     1 |         3 | YES      |
 c348  | character varying           |      60 |       |           | YES      |
 s348  | character varying           |      20 |       |           | YES      |
 s348  | character                   |       8 |       |           | NO       |
 s348  | character varying           |      60 |       |           | NO       |
 s348  | character                   |       3 |       |           | YES      |
 s348  | timestamp without time zone |         |       |           | NO       |
 s348  | numeric                     |         |     2 |         3 | YES      |
 sc348 | character                   |       8 |       |           | NO       |
 sc348 | character varying           |      10 |       |           | NO       |
 sc348 | numeric                     |         |     1 |         4 | YES      | NULL::numeric
(14 rows)
```

图 13: 各表属性信息

4 SQL 语句编写与执行

4.1 数据查询操作

题目 1 查询电子工程系（EE）所开课程的课程编号、课程名称及学分数。

```
mydb=> SELECT C#, CNAME, CREDIT FROM
mydb-> C348 WHERE C# LIKE 'EE-%';
  c#   |      cname       | credit
-------+------------------+--------
 EE-01 | 信号与系统       |    3.0
 EE-02 | 数字逻辑电路     |    5.0
 EE-03 | 光电子学与光子学 |    2.0
(3 rows)
```

图 14: 题目 1

题目 2 查询未选修课程”CS-02” 的女生学号及其已选各课程编号、成绩。

```
mydb=> SELECT S348.S#, COALESCE(SC348.C#, '未选修课程') AS C# , SC348.GRADE FROM S348
mydb-> LEFT OUTER JOIN SC348 ON S348.S# = SC348.S#
mydb-> WHERE S348.SEX = '女' AND S348.S# NOT IN
mydb->  (SELECT DISTINCT S# FROM SC348 WHERE C#='CS-02');
    s#    |     c#     | grade
----------+------------+-------
 03031011 | EE-01      |  91.0
 03031011 | EE-02      |  86.0
 03031009 | EE-01      |  88.0
 03031009 | EE-02      |  78.5
 03031056 | 未选修课程 |
(5 rows)
```

图 15: 题目 2

题目 3 查询 2004 年 ~ 2005 年出生学生的基本信息。

```
mydb=> SELECT S348.S#, S348.SNAME, COALESCE(SUM(C348.CREDIT),0.0) AS "Total Credits" FROM S348
LEFT OUTER JOIN SC348 ON S348.S# = SC348.S#
LEFT OUTER JOIN C348 ON C348.C# = SC348.C# GROUP BY S348.S#;
    s#    | sname  | Total Credits
----------+--------+---------------
 03031014 | 赵思扬 |           8.0
 03031033 | 蔡明明 |           8.0
 01032023 | 孙文   |           9.0
 03031009 | 田菲   |           8.0
 03031056 | 曹子衿 |           0.0
 03031011 | 王倩   |           8.0
 01032112 | 董喆   |          11.0
 03031051 | 周剑   |           8.0
 01032001 | 张晓梅 |           9.0
 01032010 | 王涛   |           9.0
 01032005 | 刘静   |           9.0
(11 rows)
```

图 16: 题目 3

题目 4 查询每位学生的学号、学生姓名及其已选修课程的学分总数。

```
mydb=>  SELECT * FROM S348 WHERE BDATE >= '2004-01-01' AND BDATE <= '2004-12-31';
    s#    | sname | sex |        bdate        | height |   dorm
----------+-------+-----+---------------------+--------+----------
 01032023 | 孙文  | 男  | 2004-06-10 00:00:00 |   1.80 | 东6舍221
 03031011 | 王倩  | 女  | 2004-12-20 00:00:00 |   1.66 | 东2舍104
(2 rows)
```

图 17: 题目 4

题目 5 查询选修课程”CS-01” 的学生中成绩第二高的学生学号。

```
mydb=> SELECT S# FROM SC348
mydb-> WHERE C#='CS-01' AND GRADE = (
mydb(> SELECT MAX(GRADE) FROM SC348
mydb(> WHERE C#='CS-01' AND GRADE < (
mydb(> SELECT MAX(GRADE) FROM SC348 WHERE C#='CS-01'
mydb(> ));
    s#
----------
 01032010
(1 row)
```

图 18: 题目 5

题目 6 查询平均成绩超过”王涛”同学的学生学号、姓名和平均成绩，并按学号进行降序排列。

```
mydb=> SELECT S348.S#,
mydb->        S348.SNAME,
mydb->        AVG(SC348.GRADE) AS AVERAGE
mydb-> FROM S348, SC348
mydb-> WHERE S348.S# = SC348.S#
mydb-> GROUP BY S348.S#, S348.SNAME
mydb-> HAVING AVG(SC348.GRADE) > (
mydb(>  SELECT AVG(SC348.GRADE)
mydb(>  FROM S348, SC348
mydb(>  WHERE S348.S# = SC348.S#
mydb(>   AND S348.SNAME = '王涛'
mydb(> )
mydb-> ORDER BY S348.S# DESC;
    s#    | sname  |        average
----------+--------+----------------------
 03031033 | 蔡明明 | 91.0000000000000000
 03031011 | 王倩   | 88.5000000000000000
 01032112 | 董喆   | 88.5000000000000000
(3 rows)
```

图 19: 题目 6

题目 7 查询选修了计算机专业全部课程（课程编号为”CS-××”）的学生姓名及已获得的学分总数。

```
mydb=> SELECT S348.SNAME,
 SUM(C1.CREDIT) AS TOTAL
FROM S348, SC348, C348 AS C1
WHERE S348.S# = SC348.S# AND
 C1.C# = SC348.C#
AND NOT EXISTS (
SELECT * FROM C348 AS C2
WHERE C2.C# LIKE 'CS-%'
AND NOT EXISTS (
 SELECT * FROM SC348
 WHERE SC348.S# = S348.S# AND
  SC348.C# = C2.C#
))
GROUP BY S348.S#, S348.SNAME;
 sname | total
-------+-------
 董喆  |  11.0
(1 row)
```

图 20: 题目 7

题目 8 查询选修了 3 门以上课程（包括 3 门）的学生中平均成绩最高的同学学号及姓名。

```
mydb=> SELECT S348.S#, S348.SNAME
FROM S348, SC348
WHERE S348.S# = SC348.S#
GROUP BY S348.S#, S348.SNAME
HAVING COUNT(SC348.C#) >= 3
AND AVG(SC348.GRADE) >= ALL(
 SELECT AVG(SC348.GRADE)
 FROM S348, SC348
 WHERE S348.S# = SC348.S#
 GROUP BY S348.S#, S348.SNAME
 HAVING COUNT(SC348.C#) >= 3);
    s#    | sname
----------+-------
 01032112 | 董喆
(1 row)
```

图 21: 题目 8

4.2 数据插入操作

题目 1 在学生表中插入记录：('01032005', ' 刘竞', ' 男', '2003-12-10', 1.75, ' 东 14 舍 312')。

```
mydb=> INSERT INTO S348 VALUES('01032005', '刘竞', '男', '2003-12-10', 1.75, '东14舍312');
ERROR:  duplicate key value violates unique constraint "s348_pkey"
DETAIL:  Key ("s#")=(01032005) already exists.
```

图 22: 插入操作 1：违反主键约束

可以看到，这个插入操作插入失败，理由是'01032005' 作为主键已经在 S348 表中存在。

题目 2 在课程表中插入记录：('CS-03', ' 离散数学', 64, 4, ' 陈建明')。

```
mydb=> INSERT INTO C348 VALUES('CS-03', '离散数学', 64, 4, '陈建明');
INSERT 0 1
```

图 23: 插入操作 2：插入成功

4.3 数据删除操作

将学生表中已修学分数大于 60 的学生记录删除。

```
mydb=> DELETE FROM S348
WHERE S# IN (
 SELECT S348.S#
 FROM S348, SC348, C348
 WHERE S348.S# = SC348.S# AND
  C348.C# = SC348.C#
 GROUP BY S348.S#, S348.SNAME
 HAVING SUM(C348.CREDIT) > 60);
DELETE 0
```

图 24: 删除操作

可以看到，没有学生记录被删除，因为没有学分已修学分数大于 60.

4.4 数据更新操作

将” 张明” 老师负责的” 数字电子技术” 课程的学时数调整为 36，同时增加一个学分。

```
DELETE 0
mydb=> UPDATE C348 SET PERIOD = 36, CREDIT = CREDIT + 1
mydb-> WHERE TEACHER = '张明' AND CNAME = '数字电子技术';
UPDATE 0
mydb=>
```

图 25: 更新操作

可以看到，没有课程记录被更新，这是因为张明老师并不负责“数字电子技术”课程。

4.5 创建视图操作

视图 1 居住在”东 18 舍”的男生视图，包括学号、姓名、出生日期、身高等属性。

```
mydb=> CREATE VIEW S_E18_MALE
mydb-> (S#, SNAME, BDATE, HEIGHT)
mydb-> AS SELECT S#, SNAME, BDATE, HEIGHT
mydb->  FROM S348
mydb->  WHERE SEX = '男' AND
mydb->   DORM LIKE '东18舍%';
CREATE VIEW
mydb=> SELECT * FROM S_E18_MALE;
    s#    | sname  |        bdate        | height
----------+--------+---------------------+--------
 03031014 | 赵思扬 | 2002-06-06 00:00:00 |   1.85
 03031051 | 周剑   | 2002-05-08 00:00:00 |   1.68
 03031033 | 蔡明明 | 2003-03-12 00:00:00 |   1.75
(3 rows)
```

图 26: 视图 1

视图 2 ”张明”老师所开设课程情况的视图，包括课程编号、课程名称、平均成绩等属性。

```
mydb=> CREATE VIEW C_ZHANGMING
mydb-> (C#, CNAME, AVG_GRADE) AS
mydb->  SELECT C348.C#, C348.CNAME,
mydb->   AVG(SC348.GRADE)
mydb->  FROM C348, SC348
mydb->  WHERE C348.C# = SC348.C# AND
mydb->   C348.TEACHER = '张明'
mydb->  GROUP BY C348.C#, C348.CNAME;
CREATE VIEW
mydb=> SELECT * FROM C_ZHANGMING;
  c#   |   cname    |      avg_grade
-------+------------+---------------------
 EE-01 | 信号与系统 | 85.8000000000000000
(1 row)
```

图 27: 视图 2

视图 3 所有选修了”人工智能”课程的学生视图，包括学号、姓名、成绩等属性。

```
mydb=> CREATE VIEW S_AI (S#, SNAME, GRADE)
mydb-> AS SELECT S348.S#, S348.SNAME, SC348.GRADE
mydb->  FROM S348, C348, SC348
mydb->  WHERE C348.C# = SC348.C# AND
mydb->   S348.S# = SC348.S# AND
mydb->   C348.CNAME = '人工智能';
CREATE VIEW
mydb=> SELECT * FROM C_AI;
ERROR:  relation "c_ai" does not exist on dn_6001
LINE 1: SELECT * FROM C_AI;
                      ^
mydb=> SELECT * FROM S_AI;
    s#     | sname  | grade
-----------+--------+-------
 01032010 | 王涛   |  83.5
 01032001 | 张晓梅 |  83.0
 01032005 | 刘静   |  82.0
 01032023 | 孙文   |  76.0
 01032112 | 董喆   |  86.0
(5 rows)
```

图 28: 视图 3

5 数据补充与查询优化

5.1 数据生成方法

基于前述学生表（表 2）、课程表（表 8）和选课表（表 4）的约束条件，本次实验基于以下策略进行数据生成：

5.1.1 学生表数据生成

姓名数据获取 通过爬取 2024 年全国大学生数学建模竞赛获奖名单（PDF 格式）获取真实姓名数据。采用 Python 的 `pdfplumber` 库进行 PDF 表格解析，提取所有获奖者姓名。由于数学建模竞赛每队包含 3 名成员，使用 `pandas` 库进行数据处理，将所有姓名汇总至统一表格。

相较于使用常见姓名库生成数据，这一方法的优势在于确保获取的姓名符合当代大学生的命名特征，避免使用过时的常用姓名。

其他字段生成规则 姓名确定后，其余字段按以下规则分配：

S# 学号 生成 8 位定长学号，采用集合数据结构确保唯一性。根据数据格式要求，第 2 位和第 4 位固定为 0，其余位数随机分配。

SEX 性别 遵循生物学规律，男女比例设定为 1:1。

BDATE 出生日期 参考预先提供的数据的特征，出生日期随机分布在 2001 年 1 月 1 日至 2005 年 12 月 31 日之间。

HEIGHT 身高 基于性别差异化生成。假设人类身高服从正态分布，其中男性身高服从 $N(1.75, 0.08^2)$，女性身高服从 $N(1.65, 0.08^2)$，通过正态分布随机采样生成。

DORM 宿舍 格式为 {方向}{楼号} 舍 {层号}{门号}，基于如下具体约束随机生成：

- 方向：东、西
- 楼号：1–30
- 层号：2–7
- 门号：01–40

参考实际宿舍管理规范，采用了优先填满宿舍，确保每间宿舍不超过 4 人，且男女分楼居住的分配策略进行分配，使得生成的数据更加符合实际情况。

5.1.2 课程表数据生成

采用 Python 的 `requests` 库，通过填入访问西安交通大学本科教务平台的 cookie 信息，爬取全部课程数据。数据处理包括：删除所有 0 学分课程；对于多名授课教师的课程，仅保留第一位教师信息。

针对课程编号格式转换：西安交通大学课程编号采用”四位字母（专业标识）+ 六位数字（序号）”格式。为满足数据库约束要求，提取四位字母中的第 1 位和第 3 位，以及六位数字中的中间两位，并进行筛选以确保课程编号的唯一性。

5.1.3 选课表数据生成

制定”单个学生选课学分数量适中”的生成规则，具体策略如下：

- 设定约 10% 的课程尚未出成绩；
- 对于已出成绩的课程，假设成绩服从正态分布 $N(75, 10^2)$，取值范围限定在 40–100 分；
- 设置学生平均获得学分为 250；
- 假设学生选课学分数服从以该期望值为均值的正态分布，据此进行数据分配。

数据生成后，存入 CSV 文件，上传至虚拟机，随后，按照要求使用 JDBC 连接数据库，多线程插入数据，详见实验代码。

5.2 小规模补充与随机删除

在学生表中补充数据至约 1000 行，在课程表中补充数据至约 100 行，在选课表中补充数据至约 20000 行。在向选课表中补充数据的过程中，随机选择成绩低于 60 分（含 NULL）的 200 行选课记录删除。

具体实现上，我们使用多线程进行分批插入，由于插入 20000 行选课数据时需要同时随机删除 200 条数据，所以这里选择的 batch size 为 100。对于每个选课数据，每个新线程负责插入 100 条数据，并删除 1 条成绩低于 60 分或 NULL 的数据。对于学生表、课程表的数据插入，除了不用删除数据之外，和选课表相同。

```
读取到 114 条记录
pool-2-thread-1 完成插入 100 条记录
pool-2-thread-2 完成插入 14 条记录
所有数据插入完成，耗时: 366ms
```

图 29: 插入课程数据

```
INFO: Connect complete. ID: 74e33f9a-0ec1
连接成功!
读取到 1024 条记录
pool-1-thread-1 完成插入 100 条记录
pool-1-thread-3 完成插入 100 条记录
pool-1-thread-5 完成插入 100 条记录
pool-1-thread-4 完成插入 100 条记录
pool-1-thread-2 完成插入 100 条记录
pool-1-thread-1 完成插入 100 条记录
pool-1-thread-3 完成插入 100 条记录
pool-1-thread-8 完成插入 100 条记录
pool-1-thread-7 完成插入 100 条记录
pool-1-thread-7 完成插入 24 条记录
pool-1-thread-6 完成插入 100 条记录
所有数据插入完成，耗时: 2389ms
```

图 30: 插入学生数据

```
读取到 20100 条记录
pool-3-thread-1 完成插入 100 条记录
pool-3-thread-2 完成插入 100 条记录
pool-3-thread-4 完成插入 100 条记录
pool-3-thread-8 完成插入 100 条记录
pool-3-thread-7 完成插入 100 条记录
pool-3-thread-6 完成插入 100 条记录
随机选择了1条记录进行删除
开始删除记录...
pool-3-thread-5 完成插入 100 条记录
pool-3-thread-3 完成插入 100 条记录
随机选择了1条记录进行删除
开始删除记录...
随机选择了1条记录进行删除
开始删除记录...
随机选择了1条记录进行删除
开始删除记录...
随机选择了1条记录进行删除
随机选择了1条记录进行删除
开始删除记录...
开始删除记录...
随机选择了1条记录进行删除
开始删除记录...
pool-4-thread-1完成删除记录
随机选择了1条记录进行删除
开始删除记录...
pool-6-thread-1完成删除记录
pool-5-thread-1完成删除记录
```

图 31: 插入选课数据（节选）

```
[omm@db1 xjtu_database_lab]$ gsql -d mydb -p 26000 -U joe -W OpenGauss@1234 -r
gsql ((openGauss 1.1.0 build 392c0438) compiled at 2020-12-31 20:08:21 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

mydb=> select COUNT(*) FROM S348;
 count
-------
  1035
(1 row)

mydb=> select COUNT(*) FROM SC348;
 count
-------
 19926
(1 row)

mydb=> select COUNT(*) FROM C348;
 count
-------
   122
(1 row)

mydb=>
```

图 32: 插入结果

结果数据成功补充到要求范围，其中注意到选课记录数量在 csv 源文件中是 20100 行，算上已有的 26 行，在删除 200 条记录后为 19926 行.

5.3 大规模补充与查询优化

在学生表中补充数据至约 5000 行，在课程表中补充数据至约 1000 行，在选课表中补充数据至约 200000 行。

在 5.2 代码的基础上，修改导入文件名，把插入选课表数据的方法的代码中的删除语句进行注释。同时为了提高执行效率，我们将 batch size 修改为 2000。

```
[root@db1 xjtu_database_lab]#  cd /root/xjtu_database_lab ; /usr/bi
ubvytuf758ec8qd3x.jar cn.edu.xjtu.dblab.App
Jun 13, 2025 6:58:30 PM org.opengauss.core.v3.ConnectionFactoryImpl
INFO: [f9cda8a8-5d6e-4661-82e8-0945d50f1bf0] Try to connect. IP: 10
Jun 13, 2025 6:58:31 PM org.opengauss.core.v3.ConnectionFactoryImpl
INFO: [10.0.2.15:45660/10.0.2.15:26000] Connection is established.
Jun 13, 2025 6:58:31 PM org.opengauss.core.v3.ConnectionFactoryImpl
INFO: Connect complete. ID: f9cda8a8-5d6e-4661-82e8-0945d50f1bf0
连接成功!
读取到 4163 条记录
pool-1-thread-1 完成插入 2000 条记录
pool-1-thread-3 完成插入 163 条记录
pool-1-thread-2 完成插入 2000 条记录
所有数据插入完成，耗时: 2094ms
读取到 1000 条记录
pool-2-thread-1 完成插入 1000 条记录
所有数据插入完成，耗时: 197ms
```

图 33: 插入学生和选课数据

```
pool-3-thread-7 完成插入 2000 条记录
pool-3-thread-1 完成插入 2000 条记录
pool-3-thread-8 完成插入 2000 条记录
pool-3-thread-3 完成插入 2000 条记录
pool-3-thread-6 完成插入 2000 条记录
pool-3-thread-4 完成插入 2000 条记录
pool-3-thread-5 完成插入 2000 条记录
pool-3-thread-2 完成插入 2000 条记录
pool-3-thread-5 完成插入 2000 条记录
pool-3-thread-5 完成插入 2000 条记录
pool-3-thread-4 完成插入 2000 条记录
pool-3-thread-6 完成插入 2000 条记录
pool-3-thread-3 完成插入 2000 条记录
pool-3-thread-8 完成插入 2000 条记录
pool-3-thread-1 完成插入 2000 条记录
pool-3-thread-7 完成插入 2000 条记录
pool-3-thread-7 完成插入 500 条记录
pool-3-thread-1 完成插入 2000 条记录
pool-3-thread-8 完成插入 2000 条记录
pool-3-thread-3 完成插入 2000 条记录
pool-3-thread-6 完成插入 2000 条记录
pool-3-thread-4 完成插入 2000 条记录
pool-3-thread-5 完成插入 2000 条记录
pool-3-thread-2 完成插入 2000 条记录
所有数据插入完成，耗时: 175337ms
```

图 34: 插入选课数据（节选）

```
mydb=> SELECT COUNT(*) FROM S348;
 count
-------
  5198
(1 row)

mydb=> SELECT COUNT(*) FROM C348;
 count
-------
  1122
(1 row)

mydb=> SELECT COUNT(*) FROM SC348;
 count
--------
 200425
(1 row)
```

图 35: 插入结果

接着，插入完成后，我按照要求为三个查询编写不同的 SQL 语句实现，并分析了其运行效率。

题目 2 查询未选修课程”CS-02” 的女生学号及其已选各课程编号、成绩。

优化前 查询语句

```sql
SELECT S348.S#,
 COALESCE(SC348.C#, '未选修课程') AS C#,
 SC348.GRADE
FROM S348
LEFT OUTER JOIN SC348 ON SC348.S# = S348.S#
WHERE S348.SEX = '女' AND
 S348.S# NOT IN (
  SELECT DISTINCT S# FROM SC348
  WHERE SC348.C# = 'CS-02'
 );
```

代码 1: 优化前

```
 Hash Right Join  (cost=3948.80..8948.25 rows=98334 width=21) (actual time=57.985..207.419
rows=97601 loops=1)
   Hash Cond: (joe.sc348."s#" = s348."s#")
   (Buffers: shared hit=2580)
   ->  Seq Scan on sc348  (cost=0.00..3265.17 rows=200217 width=21) (actual time=0.011..50.
083 rows=200425 loops=1)
         (Buffers: shared hit=1263)
   ->  Hash  (cost=3917.05..3917.05 rows=2540 width=9) (actual time=57.762..57.762 rows=254
4 loops=1)
          Buckets: 32768  Batches: 1  Memory Usage: 102kB
         (Buffers: shared hit=1317)
         ->  Hash Anti Join  (cost=3766.30..3917.05 rows=2540 width=9) (actual time=54.517.
.56.965 rows=2544 loops=1)
               Hash Cond: (s348."s#" = joe.sc348."s#")
               (Buffers: shared hit=1317)
               ->  Seq Scan on s348  (cost=0.00..118.98 rows=2546 width=9) (actual time=0.0
24..1.504 rows=2546 loops=1)
                     Filter: (sex = '女'::bpchar)
                     Rows Removed by Filter: 2652
                     (Buffers: shared hit=54)
               ->  Hash  (cost=3766.24..3766.24 rows=5 width=9) (actual time=54.292..54.292
 rows=5 loops=1)
                      Buckets: 32768  Batches: 1  Memory Usage: 1kB
                     (Buffers: shared hit=1263)
                     ->  HashAggregate  (cost=3766.14..3766.19 rows=5 width=9) (actual time
=54.273..54.275 rows=5 loops=1)
                           Group By Key: joe.sc348."s#"
                           (Buffers: shared hit=1263)
                           ->  Seq Scan on sc348  (cost=0.00..3765.71 rows=169 width=9) (ac
tual time=0.012..54.250 rows=5 loops=1)
                                 Filter: ("c#" = 'CS-02'::bpchar)
                                 Rows Removed by Filter: 200420
                                 (Buffers: shared hit=1263)
 Total runtime: 221.684 ms
(26 rows)
```

图 36: 优化前查询计划

通过查询计划看到，原查询一直使用顺序扫描来获取结果，查询效率较低。

优化后 查询语句

```sql
SELECT S348.S#,
 COALESCE(SC348.C#, '未选修课程') AS C#,
 SC348.GRADE
FROM S348
LEFT OUTER JOIN SC348 ON SC348.S# = S348.S#
WHERE S348.SEX = '女' AND
 NOT EXISTS (
  SELECT * FROM SC348
  WHERE SC348.S# = S348.S# AND
   SC348.C# = 'CS-02'
 );
```

代码 2: 优化后

```
------------------------------------------------------------------------------------------
-------------------------------------------------
 Hash Right Join  (cost=1278.44..6198.52 rows=90401 width=21) (actual time=15.965..141.347
rows=97601 loops=1)
   Hash Cond: (joe.sc348."s#" = s348."s#")
   (Buffers: shared hit=8971)
   ->  Seq Scan on sc348  (cost=0.00..3265.17 rows=200217 width=21) (actual time=0.006..40.
229 rows=200425 loops=1)
         (Buffers: shared hit=1263)
   ->  Hash  (cost=1249.26..1249.26 rows=2335 width=9) (actual time=15.807..15.807 rows=254
4 loops=1)
          Buckets: 32768  Batches: 1  Memory Usage: 102kB
         (Buffers: shared hit=7708)
         ->  Nested Loop Anti Join  (cost=0.00..1249.26 rows=2335 width=9) (actual time=0.0
57..15.048 rows=2544 loops=1)
               ->  Seq Scan on s348  (cost=0.00..118.98 rows=2546 width=9) (actual time=0.0
18..1.710 rows=2546 loops=1)
                     Filter: (sex = '女'::bpchar)
                     Rows Removed by Filter: 2652
                     (Buffers: shared hit=54)
               ->  Index Only Scan using sc348_pkey on sc348  (cost=0.00..3.83 rows=120 wid
th=9) (actual time=11.748..11.748 rows=2 loops=2546)
                     Index Cond: (("s#" = s348."s#") AND ("c#" = 'CS-02'::bpchar))
                     Heap Fetches: 2
                     (Buffers: shared hit=7654)
 Total runtime: 153.611 ms
(18 rows)
```

图 37: 优化后查询计划

通过执行时间可以发现，优化后查询相比优化前效率提升约 30%。查看查询计划，发现最后一个阶段并不采用顺序扫描，而直接采用了索引，显示了 NOT EXISTS 的高效。

题目 7 查询选修了计算机专业全部课程（课程编号为"CS-××"）的学生姓名及已获得的学分总数。

优化前 查询语句

```sql
SELECT S348.SNAME,
 SUM(C1.CREDIT) AS TOTAL_CREDIT
FROM S348, SC348, C348 C1
WHERE S348.S# = SC348.S# AND
```

```
 C1.C# = SC348.C# AND
 NOT EXISTS(
  SELECT * FROM C348 C2
  WHERE C2.C# LIKE 'CS-%' AND
   NOT EXISTS (
    SELECT * FROM SC348
    WHERE SC348.S# = S348.S# AND
     SC348.C# = C2.C#
   )
 )
GROUP BY S348.S#, S348.SNAME;
```

代码 3: 优化前

```
 HashAggregate  (cost=549875.56..549927.54 rows=5198 width=55) (actual time=196.514..196.514 rows=0 loops
=1)
   Group By Key: s348."s#", s348.sname
   (Buffers: shared hit=6385)
   ->  Hash Join  (cost=40.61..549124.75 rows=100108 width=23) (actual time=196.488..196.488 rows=0 loops
=1)
         Hash Cond: (joe.sc348."c#" = (c1."c#")::bpchar)
         (Buffers: shared hit=6385)
         ->  Nested Loop  (cost=4.37..547712.02 rows=100108 width=24) (actual time=196.486..196.486 rows=
0 loops=1)
               ->  Nested Loop Anti Join  (cost=4.37..539431.50 rows=2599 width=18) (actual time=196.483.
.196.483 rows=0 loops=1)
                     Join Filter: (NOT (alternatives: SubPlan 1 or hashed SubPlan 2))
                     Rows Removed by Join Filter: 13
                     ->  Index Scan using s348_pkey on s348  (cost=0.00..414.22 rows=5198 width=18) (actu
al time=0.020..5.226 rows=5198 loops=1)
                           (Buffers: shared hit=5119)
                     ->  Materialize  (cost=4.37..16.15 rows=23 width=6) (actual time=0.999..1.011 rows=5
211 loops=5198)
                           (Buffers: shared hit=3)
                           ->  Bitmap Heap Scan on c348 c2  (cost=4.37..16.04 rows=23 width=6) (actual ti
me=0.248..0.254 rows=5 loops=1)
                                 Filter: (("c#")::text ~~ 'CS-%'::text)
                                 (Buffers: shared hit=3)
                                 ->  Bitmap Index Scan on c348_pkey  (cost=0.00..4.36 rows=11 width=0) (a
ctual time=0.030..0.030 rows=27 loops=1)
                                       Index Cond: ((("c#")::text >= 'CS-'::text) AND (("c#")::text < 'CS
.'::text))
                                       (Buffers: shared hit=2)
                     SubPlan 1
                       ->  Index Only Scan using sc348_pkey on sc348  (cost=0.00..8.31 rows=1 width=0) (A
ctual time: never executed)
                             Index Cond: (("s#" = s348."s#") AND ("c#" = (c2."c#")::bpchar))
                             Heap Fetches: 0
                     SubPlan 2
                       ->  Seq Scan on sc348  (cost=0.00..3265.17 rows=200217 width=15) (actual time=0.01
4..55.947 rows=200425 loops=1)
               ->  Index Only Scan using sc348_pkey on sc348  (cost=0.00..2.80 rows=39 width=15) (Actual
time: never executed)
                     Index Cond: ("s#" = s348."s#")
                     Heap Fetches: 0
         ->  Hash  (cost=22.22..22.22 rows=1122 width=11) (Actual time: never executed)
               Buckets: 0  Batches: 0  Memory Usage: 0kB
               ->  Seq Scan on c348 c1  (cost=0.00..22.22 rows=1122 width=11) (Actual time: never execute
d)
 Total runtime: 198.113 ms
(33 rows)
```

图 38: 优化前查询计划

可以看到，这个查询使用了 NOT EXISTS 进行查询，将问题转化为了查找没有任何 CS 课程未选修的学生，因此对每个学生都进行了一次 CS 课程表的查询，即使有上一部分说的优势，仍然性能较低。

优化后 查询语句

```sql
SELECT S348.SNAME,
 SUM(C1.CREDIT) AS TOTAL_CREDIT
FROM S348, SC348, C348 C1
WHERE S348.S# = SC348.S# AND
 C1.C# = SC348.C# AND
 S348.S# IN (
  SELECT SC348.S#
  FROM SC348, C348 C2
  WHERE SC348.C# = C2.C# AND
   C2.C# LIKE 'CS-%'
  GROUP BY SC348.S#
  HAVING COUNT(SC348.C#) = (
   SELECT COUNT(*) FROM C348
   WHERE C# LIKE 'CS-%'
  )
 )
GROUP BY S348.S#, S348.SNAME;
```

代码 4: 优化后

```
                                                                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=11411.60..11463.58 rows=5198 width=55) (actual time=68.889..68.889 rows=0 loops=1)
   Group By Key: s348."s#", s348.sname
   (Buffers: shared hit=1296)
   ->  Hash Join  (cost=4262.01..10660.79 rows=100108 width=23) (actual time=68.845..68.845 rows=0 loops=
1)
         Hash Cond: (joe.sc348."c#" = (c1."c#")::bpchar)
         (Buffers: shared hit=1296)
         ->  Hash Join  (cost=4225.77..9248.06 rows=100108 width=24) (actual time=68.080..68.080 rows=0 l
oops=1)
               Hash Cond: (joe.sc348."s#" = s348."s#")
               (Buffers: shared hit=1285)
               ->  Seq Scan on sc348  (cost=0.00..3265.17 rows=200217 width=15) (actual time=0.007..0.007
 rows=1 loops=1)
                     (Buffers: shared hit=1)
               ->  Hash  (cost=4193.28..4193.28 rows=2599 width=27) (actual time=67.883..67.883 rows=0 lo
ops=1)
                      Buckets: 32768  Batches: 1  Memory Usage: 0kB
                     (Buffers: shared hit=1284)
                     ->  Hash Join  (cost=4033.07..4193.28 rows=2599 width=27) (actual time=67.881..67.88
1 rows=0 loops=1)
                           Hash Cond: (s348."s#" = joe.sc348."s#")
                           (Buffers: shared hit=1284)
                           ->  Seq Scan on s348  (cost=0.00..105.98 rows=5198 width=18) (actual time=0.00
5..0.005 rows=1 loops=1)
                                 (Buffers: shared hit=1)
                           ->  Hash  (cost=3981.53..3981.53 rows=4123 width=9) (actual time=67.767..67.76
7 rows=0 loops=1)
                                  Buckets: 32768  Batches: 1  Memory Usage: 0kB
                                 (Buffers: shared hit=1283)
                                 ->  HashAggregate  (cost=3888.77..3940.30 rows=4123 width=23) (actual ti
me=67.763..67.763 rows=0 loops=1)
                                       Group By Key: joe.sc348."s#"
                                       Filter: (count(joe.sc348."c#") = $0)
```

图 39: 优化后查询计划（前半）

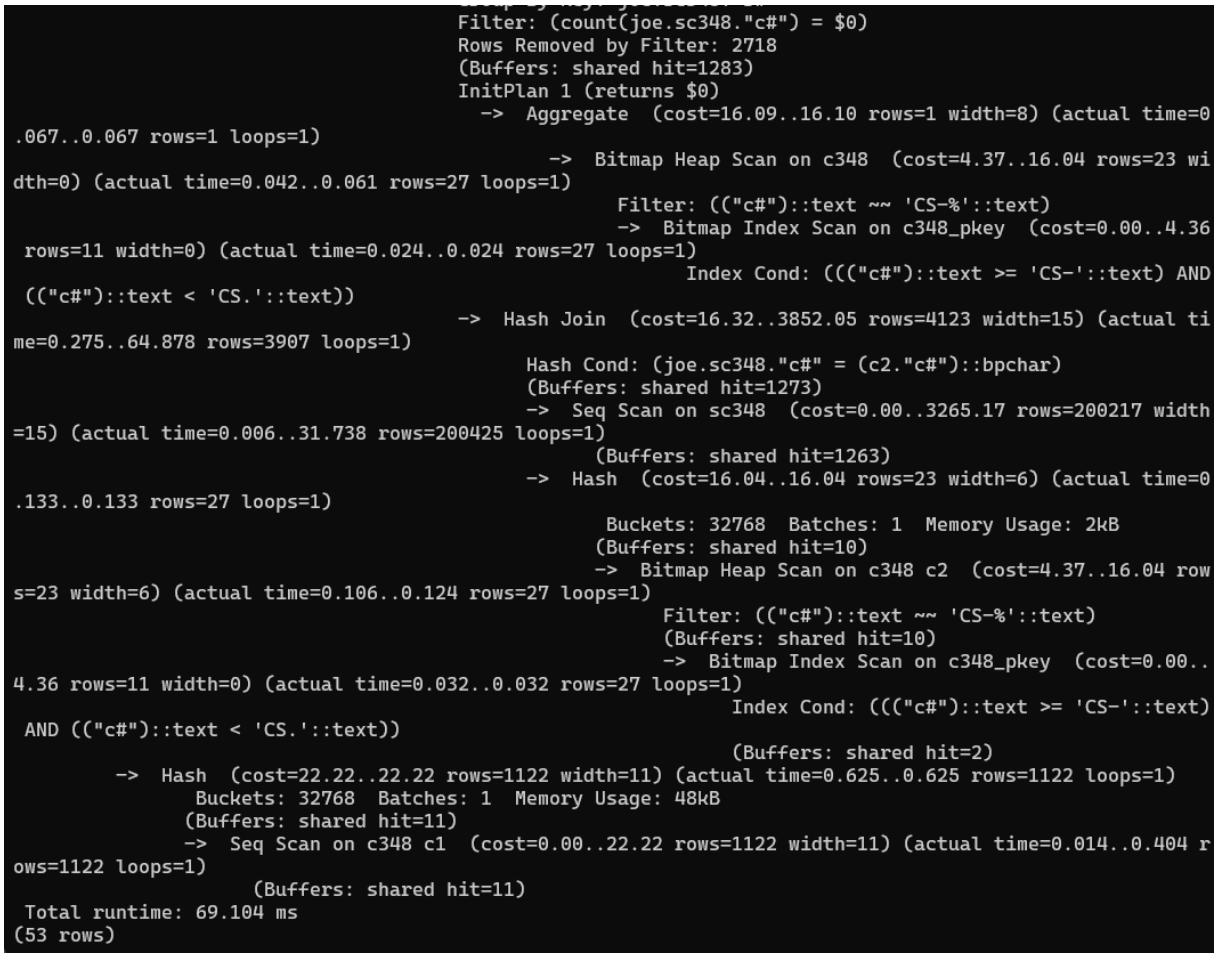

图 40: 优化后查询计划（后半）

可以看到，优化后的查询计划更为复杂。这个优化主要是通过分组聚合函数进行分组计数，从而获得更高的执行效率。

题目 8 查询选修了 3 门以上课程（包括 3 门）的学生中平均成绩最高的同学学号及姓名。

优化前 查询语句

```sql
SELECT S348.S#, S348.SNAME
FROM S348, SC348
WHERE S348.S# = SC348.S#
GROUP BY S348.S#, S348.SNAME
HAVING COUNT(SC348.C#) >= 3 AND
 AVG(SC348.GRADE) >= ALL(
  SELECT AVG(SC348.GRADE)
  FROM S348, SC348
  WHERE S348.S# = SC348.S#
  GROUP BY S348.S#, S348.SNAME
  HAVING COUNT(SC348.C#) >= 3
 );
```

代码 5: 优化前

```
 HashAggregate  (cost=16382.56..320439.57 rows=5198 width=70) (actual time=922.349..938.644 rows=1 loop
s=1)
   Group By Key: joe.s348."s#", joe.s348.sname
   Filter: ((count(joe.sc348."c#") >= 3) AND (SubPlan 1))
   Rows Removed by Filter: 5196
   (Buffers: shared hit=2634)
   ->  Hash Join  (cost=170.96..6189.11 rows=200217 width=30) (actual time=4.595..183.697 rows=200425 l
oops=1)
         Hash Cond: (joe.sc348."s#" = joe.s348."s#")
         (Buffers: shared hit=1317)
         ->  Seq Scan on sc348  (cost=0.00..3265.17 rows=200217 width=21) (actual time=0.076..50.988 ro
ws=200425 loops=1)
               (Buffers: shared hit=1263)
         ->  Hash  (cost=105.98..105.98 rows=5198 width=18) (actual time=4.164..4.164 rows=5198 loops=1
)
                Buckets: 32768  Batches: 1  Memory Usage: 257kB
               (Buffers: shared hit=54)
               ->  Seq Scan on s348  (cost=0.00..105.98 rows=5198 width=18) (actual time=0.044..1.466 r
ows=5198 loops=1)
                     (Buffers: shared hit=54)
   SubPlan 1
     ->  Materialize  (cost=8191.28..8295.24 rows=5198 width=70) (actual time=444.482..463.783 rows=548
62 loops=5191)
           (Buffers: shared hit=1317)
           ->  HashAggregate  (cost=8191.28..8269.25 rows=5198 width=70) (actual time=443.941..449.001
rows=5191 loops=1)
                 Group By Key: joe.s348."s#", joe.s348.sname
                 Filter: (count(joe.sc348."c#") >= 3)
                 Rows Removed by Filter: 6
                 (Buffers: shared hit=1317)
                 ->  Hash Join  (cost=170.96..6189.11 rows=200217 width=30) (actual time=3.393..175.925
 rows=200425 loops=1)
                       Hash Cond: (joe.sc348."s#" = joe.s348."s#")
                       (Buffers: shared hit=1317)
                       ->  Seq Scan on sc348  (cost=0.00..3265.17 rows=200217 width=21) (actual time=0.
011..46.321 rows=200425 loops=1)
                             (Buffers: shared hit=1263)
                       ->  Hash  (cost=105.98..105.98 rows=5198 width=18) (actual time=3.206..3.206 row
s=5198 loops=1)
                              Buckets: 32768  Batches: 1  Memory Usage: 257kB
                             (Buffers: shared hit=54)
                             ->  Seq Scan on s348  (cost=0.00..105.98 rows=5198 width=18) (actual time=
0.007..1.552 rows=5198 loops=1)
                                   (Buffers: shared hit=54)
 Total runtime: 939.951 ms
(34 rows)
```

图 41: 优化前查询计划

可以看到，这个查询进行了进行了嵌套查询，内层查询筛选了所有选修了 3 门以上课程的学生，然后在这里面筛选出均分最高的学生信息，复杂度较高，效率比较低。

优化后 查询语句

```sql
SELECT S#, SNAME
FROM (
 SELECT S348.S#,
  S348.SNAME,
  AVG(SC348.GRADE),
  RANK() OVER (ORDER BY AVG(SC348.GRADE) DESC) AS RK
 FROM S348
 JOIN SC348 ON S348.S# = SC348.S#
```

```
 GROUP BY S348.S#, S348.SNAME
 HAVING COUNT(SC348.C#) >= 3
)
WHERE RK = 1;
```

代码 6: 优化后

```
 Subquery Scan on __unnamed_subquery__  (cost=8590.06..8746.00 rows=26 width=18) (actual time=532.705..
542.312 rows=1 loops=1)
   Filter: (__unnamed_subquery__.rk = 1)
   Rows Removed by Filter: 5190
   ->  WindowAgg  (cost=8590.06..8681.03 rows=5198 width=70) (actual time=532.688..540.789 rows=5191 lo
ops=1)
         (Buffers: shared hit=1320)
         ->  Sort  (cost=8590.06..8603.06 rows=5198 width=70) (actual time=526.261..528.009 rows=5191 l
oops=1)
               Sort Key: (avg(sc348.grade)) DESC
               Sort Method: quicksort  Memory: 598kB
               (Buffers: shared hit=1320)
               ->  HashAggregate  (cost=8191.28..8269.25 rows=5198 width=70) (actual time=516.485..521.
733 rows=5191 loops=1)
                     Group By Key: s348."s#", s348.sname
                     Filter: (count(sc348."c#") >= 3)
                     Rows Removed by Filter: 6
                     (Buffers: shared hit=1317)
                     ->  Hash Join  (cost=170.96..6189.11 rows=200217 width=30) (actual time=3.133..200
.192 rows=200425 loops=1)
                           Hash Cond: (sc348."s#" = s348."s#")
                           (Buffers: shared hit=1317)
                           ->  Seq Scan on sc348  (cost=0.00..3265.17 rows=200217 width=21) (actual tim
e=0.011..52.428 rows=200425 loops=1)
                                 (Buffers: shared hit=1263)
                           ->  Hash  (cost=105.98..105.98 rows=5198 width=18) (actual time=2.980..2.980
 rows=5198 loops=1)
                                 Buckets: 32768  Batches: 1  Memory Usage: 257kB
                                 (Buffers: shared hit=54)
                                 ->  Seq Scan on s348  (cost=0.00..105.98 rows=5198 width=18) (actual t
ime=0.008..1.609 rows=5198 loops=1)
                                       (Buffers: shared hit=54)
 Total runtime: 543.098 ms
(25 rows)
```

图 42: 优化后查询计划

可以看到，优化后的查询使用了窗口函数避免了嵌套循环，故提升了效率。

6 备份恢复与数据评价

6.1 备份、恢复备份

在命令行通过 gs_dump 命令备份数据库：

```
[omm@db1 xjtu_database_lab]$ gs_dump -U joe -W OpenGauss@1234 -f ~/2223515348_backup.tar -p 26000 m
ydb -F t
gs_dump[port='26000'][mydb][2025-06-04 18:13:29]: The total objects number is 411.
gs_dump[port='26000'][mydb][2025-06-04 18:13:29]: [100.00%] 411 objects have been dumped.
gs_dump[port='26000'][mydb][2025-06-04 18:13:30]: dump database mydb successfully
gs_dump[port='26000'][mydb][2025-06-04 18:13:30]: total time: 3829  ms
[omm@db1 xjtu_database_lab]$ ls
ls: cannot open directory '.': Permission denied
[omm@db1 xjtu_database_lab]$ cd ~
[omm@db1 ~]$ ls
2223515348_backup.tar                              userdatabase_backup.tar
```

图 43: 备份数据库

创建一个新数据库 WANGDB，并在通过 gs_restore 命令在其中恢复王添一同学的数据；

```
[omm@db1 ~]$ gs_restore -p 26000 -U joe -W OpenGauss@1234 -d wangdb MYDB_backup.tar
start restore operation ...
table c544 complete data imported !
table s544 complete data imported !
table sc544 complete data imported !
Finish reading 19 SQL statements!
end restore operation ...
restore operation successful
total time: 12687  ms
```

图 44: 恢复数据库

6.2 评价数据表设计

```
wangdb=> SELECT table_name AS NAME,
       data_type AS TYPE,
       character_maximum_length AS MAXCHAR,
       numeric_scale AS SCALE,
       numeric_precision AS PRECISION,
       is_nullable AS NULLABLE,
       column_default AS DEFAULT
FROM information_schema.columns
WHERE table_name IN ('s544', 'c544', 'sc544')
ORDER BY table_name ASC;
 name  |            type             | maxchar | scale | precision | nullable | default
-------+-----------------------------+---------+-------+-----------+----------+---------
 c544  | character                   |      10 |       |           | NO       |
 c544  | character varying           |      90 |       |           | NO       |
 c544  | smallint                    |         |     0 |        16 | YES      |
 c544  | smallint                    |         |     0 |        16 | YES      |
 c544  | character varying           |      20 |       |           | YES      |
 s544  | character                   |       8 |       |           | NO       |
 s544  | character varying           |      20 |       |           | NO       |
 s544  | character varying           |       3 |       |           | YES      |
 s544  | timestamp without time zone |         |       |           | YES      |
 s544  | numeric                     |         |     2 |         3 | YES      |
 s544  | character varying           |      20 |       |           | YES      |
 sc544 | character varying           |       8 |       |           | NO       |
 sc544 | character                   |      10 |       |           | NO       |
 sc544 | numeric                     |         |     1 |         4 | YES      |
(14 rows)
```

图 45: 王添一同学的数据库设计

从截图中汇总表格如下：

字段名	数据类型	约束	说明
S#	CHAR(8)	主键	8 位定长学号
SNAME	VARCHAR(20)	非空	学生姓名
SEX	VARCHAR(3)	-	学生性别
BDATE	DATE	-	出生日期
HEIGHT	DECIMAL(3,2)	-	身高，保留两位小数
DORM	VARCHAR(20)	-	宿舍编号

表 5: 合作同学学生表字段详细设计

字段名	数据类型	约束	说明
C#	CHAR(10)	主键	课程编号
CNAME	VARCHAR(90)	非空	课程名
PERIOD	SMALLINT	-	学时
CREDIT	SMALLINT	-	学分
TEACHER	VARCHAR(20)	-	授课教师

表 6: 合作同学课程表字段详细设计

字段名	数据类型	约束	说明
S#	VARCHAR(8)	主键/外键	学生学号，外键引用学生表
C#	CHAR(10)	主键/外键	课程编号，外键引用课程表
GRADE	DEC(4,1)	-	学生成绩，允许为空

表 7: 合作同学选课表字段详细设计

整体上数据类型设计较为合理，学号使用定长 CHAR(8)，符合学号固定位数的特点，对于部分字段使用 DECIMAL，精确控制小数位数。

但是与此同时，也存在如下问题：

- 选课表中学号字段使用 VARCHAR(8) 而非 CHAR(8)，与学生表的类型不一致；
- SNAME 和 CNAME 字段长度设置较短，可能不够；
- 学分设置为 SMALLINT，无法照顾到小数学分的情况。

6.3 评价数据质量

```
wangdb=> SELECT * FROM S544;
    s#    |  sname   | sex |        bdate        | height |   dorm
----------+----------+-----+---------------------+--------+----------
 01032010 | 王涛     | 男  | 2003-04-05 00:00:00 |   1.72 | 东6舍221
 01032023 | 孙文     | 男  | 2004-06-10 00:00:00 |   1.80 | 东6舍221
 01032001 | 张晓梅   | 女  | 2003-11-17 00:00:00 |   1.58 | 东1舍312
 01032005 | 刘静     | 女  | 2003-01-10 00:00:00 |   1.63 | 东1舍312
 01032112 | 董喆     | 男  | 2003-02-20 00:00:00 |   1.71 | 东6舍221
 03031011 | 王倩     | 女  | 2004-12-20 00:00:00 |   1.66 | 东2舍104
 03031014 | 赵思扬   | 男  | 2002-06-06 00:00:00 |   1.85 | 东18舍421
 03031051 | 周剑     | 男  | 2002-05-08 00:00:00 |   1.68 | 东18舍422
 03031009 | 田菲     | 女  | 2003-08-11 00:00:00 |   1.60 | 东2舍104
 03031033 | 蔡明明   | 男  | 2003-03-12 00:00:00 |   1.75 | 东18舍423
 03031056 | 曹子衿   | 女  | 2005-12-15 00:00:00 |   1.65 | 东2舍305
 22256438 | 李凯     | 女  | 2005-04-04 00:00:00 |   1.61 | 西18舍518
 22166872 | 薛文     | 女  | 2008-02-03 00:00:00 |   1.57 | 东22舍622
 22420683 | 刘帆     | 女  | 2004-02-04 00:00:00 |   1.56 | 西16舍204
 22209140 | 马凤英   | 女  | 2002-04-26 00:00:00 |   1.70 | 东13舍906
 22453837 | 左芳     | 女  | 2006-05-29 00:00:00 |   1.62 | 西8舍604
 22239288 | 陈丽     | 男  | 2003-02-14 00:00:00 |   1.66 | 东15舍425
 22332241 | 王丹     | 女  | 2001-06-10 00:00:00 |   1.64 | 西20舍913
 22345904 | 杨建华   | 女  | 2009-03-11 00:00:00 |   1.60 | 西15舍616
 22306347 | 冯建华   | 男  | 2002-03-23 00:00:00 |   1.79 | 西6舍526
```

图 46: 王添一同学的学生表数据

上图截取了合作同学生成的部分学生数据。可以看到生成数据质量较高，姓名为正常汉字姓名，男女比大约为 1：1，不过学号数据与预先提供的学号数据格式不同。

```
wangdb=> SELECT * FROM C544;
    c#     |              cname                 | period | credit | teacher
-----------+------------------------------------+--------+--------+----------
 CS-01     | 数据结构                           |     60 |      3 | 张军
 CS-02     | 计算机组成原理                     |     80 |      4 | 王亚伟
 CS-04     | 人工智能                           |     40 |      2 | 李蕾
 CS-05     | 深度学习                           |     40 |      2 | 崔昀
 EE-02     | 数字逻辑电路                       |    100 |      5 | 胡海东
 EE-03     | 光电子学与光子学                   |     40 |      2 | 石韬
 CS-03     | 离散数学                           |     64 |      4 | 陈建明
 EE-01     | 信号与系统                         |     36 |      4 | 张明
 BI-32     | 遗传学                             |     48 |      3 | 杨铁林
 BI-33     | 生物化学-2                         |     32 |      2 | 龙建纲
 BI-34     | 化学生物学                         |     48 |      3 | 李菲
 BI-35     | 分离综合开放实验                   |     48 |      2 | 张雅利
 BI-36     | 生物化学-2                         |     48 |      3 | 李剑君
 BI-37     | 生物信息学                         |     64 |      3 | 杨铁林
 BI-38     | 发酵综合开放实验                   |     48 |      2 | 高美丽
 BI-39     | 微生物学                           |     56 |      3 | 张雅利
 BI-40     | 现代仪器分析                       |     56 |      3 | 武亚艳
 BI-41     | 生物创新综合实验-1                 |     64 |      2 | 孔宇
 BI-42     | 神经生物学与脑科学                 |     32 |      2 | 李延海
 BI-43     | 合成生物学                         |     32 |      2 | 谭丹
 BI-44     | 生物制药技术                       |     32 |      2 | 孔令洪
 BI-45     | 发酵技术与应用                     |     32 |      2 | 高美丽
 BI-46     | 基因组学                           |     32 |      2 | 郭燕
 BI-47     | 基因编辑与基因治疗                 |     32 |      2 | 丁健
 BI-48     | 生物组学与精准医学                 |     32 |      2 | 王昌河
 BI-49     | 衰老生物学                         |     32 |      2 | 刘健康
 BI-50     | 干细胞与再生医学                   |     32 |      2 | 耿晶
 CH-01     | 生化仪器分析                       |     32 |      2 | 武亚艳
 CO-36     | 生命起源与生物进化                 |     32 |      2 | 刘晓刚
 CO-37     | 华夏文明探源工程概述               |     32 |      2 | 张虎勤
 CO-38     | 纳米科技前沿                       |     32 |      2 | 姚翠萍
 CO-39     | 营养与健康                         |     32 |      2 | 丁岩
 GN-84     | 探索生命奥秘-生命科学史            |     32 |      2 | 高琨
```

图 47: 王添一同学的课程表数据

上图截取了合作同学生成的部分课程数据。可以看到课程号为递增序列，课程名和课程号的专业相匹配，整体数据非常合理。

```
wangdb=> SELECT * FROM SC544;
    s#     |     c#     | grade
-----------+------------+-------
 01032010 | CS-01      |  82.0
 01032010 | CS-02      |  91.0
 01032010 | CS-04      |  83.5
 01032001 | CS-01      |  77.5
 01032001 | CS-02      |  85.0
 01032001 | CS-04      |  83.0
 01032005 | CS-01      |  62.0
 01032005 | CS-02      |  77.0
 01032005 | CS-04      |  82.0
 01032023 | CS-02      |  81.0
 01032023 | CS-04      |  76.0
 01032112 | CS-01      |  88.0
 01032112 | CS-02      |  91.5
 01032112 | CS-04      |  86.0
 03031033 | EE-01      |  93.0
 03031033 | EE-02      |  89.0
 03031009 | EE-01      |  88.0
 03031009 | EE-02      |  78.5
 03031011 | EE-01      |  91.0
 03031011 | EE-02      |  86.0
 03031051 | EE-01      |  78.0
 03031014 | EE-01      |  79.0
 03031014 | EE-02      |  71.0
 22470245 | XJ-03      |  82.5
 22208196 | MA-62      |  72.9
 22261647 | IN-17      |  84.7
 22200746 | PH-08      |  97.8
 22177300 | MA-82      |  71.5
 22249181 | GN-36      |  71.5
 22170939 | PH-33      |  98.7
 22479987 | ME-23      |  86.5
 22421033 | LA-06      |  68.0
 22143062 | FR-04      |  83.1
 22342501 | AA-10      |  68.0
 22157694 | EC-17      |  68.0
 22436468 | LA-11      |  78.6
 22326536 | MA-36      |  46.3
 22463049 | LI-13      |  49.1
 22254292 | LA-08      |  66.6
```

图 48: 王添一同学的选课表数据

上图截取了合作同学生成的部分选课表数据。可以看到生成数据质量很高，不过成绩为空的选

课记录数量较少。

7 触发器与删除记录

7.1 设计触发器

设计触发器，实现删除数据的记录功能。新建表 BK348（S，C，GRADE，DDATE），其中 DDATE 表示“删除时间”信息，我们设计为 TIMESTAMP 类型，其余字段类型与 SC348 表一致。对于主键，由于考虑到选课记录可能会被重复添加、删除，我们选择主键为 (S, C, DDATE)。

具体设计如下：

字段名	数据类型	约束	说明
S#	CHAR(8)	主键	学生编号
C#	VARCHAR(10)	主键	课程编号
GRADE	DEC(4,1)	-	学生成绩，默认为空
DDATE	TIMESTAMP	主键	删除时间，默认为当前时间戳

表 8: 删除记录表（BK348）字段详细设计

```
mydb=> CREATE TABLE BK348 (
S# CHAR(8),
C# VARCHAR(10),
GRADE DECIMAL(4,1) DEFAULT NULL,
DDATE TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
PRIMARY KEY (S#, C#, DDATE));
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "bk348_pkey" for table "bk348"
CREATE TABLE
mydb=> CREATE FUNCTION log_delete()
mydb-> RETURNS TRIGGER AS $$
mydb$> BEGIN
mydb$> INSERT INTO BK348(S#, C#, GRADE, DDATE)
mydb$> VALUES(OLD.S#, OLD.C#, OLD.GRADE, CURRENT_TIMESTAMP);
mydb$> RETURN OLD;
mydb$> END;
mydb$> $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

图 49: 创建删除记录表与记录函数

```
mydb=> CREATE TRIGGER trigger_log_delete
BEFORE DELETE ON SC348
FOR EACH ROW
EXECUTE PROCEDURE log_delete();
CREATE TRIGGER
```

图 50: 创建触发器

```
mydb=> CREATE TRIGGER trigger_log_delete
BEFORE DELETE ON SC348
FOR EACH ROW
EXECUTE PROCEDURE log_delete();
CREATE TRIGGER
mydb=> SELECT
mydb->     trigger_name,
mydb->     event_manipulation,
mydb->     event_object_table,
mydb->     action_timing
mydb-> FROM information_schema.triggers
mydb-> WHERE trigger_name = 'trigger_log_delete';
    trigger_name    | event_manipulation | event_object_table | action_timing
--------------------+--------------------+--------------------+---------------
 trigger_log_delete | DELETE             | sc348              | BEFORE
(1 row)
```

图 51: 在系统表中查询触发器信息

```
mydb=> DELETE FROM SC348
mydb-> WHERE S# IN (
mydb(>     SELECT S# FROM SC348 WHERE GRADE IS NULL LIMIT 1
mydb(> )
mydb-> AND C# IN (
mydb(>     SELECT C# FROM SC348 WHERE GRADE IS NULL LIMIT 1
mydb(> )
mydb-> AND GRADE IS NULL;
DELETE 1
mydb=> SELECT COUNT(*) as backup_count FROM BK348;
 backup_count
--------------
            1
(1 row)
```

图 52: 删除一条数据进行测试

```
mydb=> SELECT * FROM BK348;
    s#     |  c#   | grade |           ddate
-----------+-------+-------+----------------------------
 01032112 | CS-05 |       | 2025-06-04 19:08:46.448771
(1 row)
```

图 53: 测试结果：成功删除并添加到 BK 数据表中

7.2 删除记录及结果

随后，请合作同学编写程序连接至自己的 MYDB 数据库，从 SC348 表成绩为 NULL 的选课记录中随机选择 100 条记录逐条删除，触发器将删除的记录备份至 BK348，同时记录该记录被删除的时间。这里我连接到了与我合作同学的数据库，调用方法删除了对方的 100 条记录。

接着，我也邀请对方删除了我的数据库的 100 条数据，删除后结果如下：

```
mydb=> SELECT * FROM BK348;
    s#     |  c#   | grade |           ddate
-----------+-------+-------+---------------------------
 01032112 | CS-05 |       | 2025-06-04 19:08:46.448771
 04016546 | RS-05 |       | 2025-06-04 19:46:44.278877
 05015052 | AT-57 |       | 2025-06-04 19:46:44.291772
 08027247 | JS-03 |       | 2025-06-04 19:46:44.311191
 04076063 | BE-10 |       | 2025-06-04 19:46:44.323589
 06024137 | MM-02 |       | 2025-06-04 19:46:44.324095
 05050296 | PY-16 |       | 2025-06-04 19:46:44.34044
 09065770 | EG-21 |       | 2025-06-04 19:46:44.387031
 02055336 | EL-16 |       | 2025-06-04 19:46:44.387541
 07026660 | EE-76 |       | 2025-06-04 19:46:44.387763
 02018681 | IF-38 |       | 2025-06-04 19:46:44.388095
 05013285 | IF-36 |       | 2025-06-04 19:46:44.388532
 05017559 | AA-14 |       | 2025-06-04 19:46:44.499876
 06081986 | PY-25 |       | 2025-06-04 19:46:44.510998
 06089932 | FA-02 |       | 2025-06-04 19:46:44.511529
 04096643 | CE-11 |       | 2025-06-04 19:46:44.511884
 04075704 | AC-17 |       | 2025-06-04 19:46:44.512074
 06070845 | LW-38 |       | 2025-06-04 19:46:44.512438
 02055093 | MG-22 |       | 2025-06-04 19:46:44.5128
 08076497 | 0Y-35 |       | 2025-06-04 19:46:44.52022
 04083268 | FR-06 |       | 2025-06-04 19:46:44.526261
 04083767 | PL-54 |       | 2025-06-04 19:46:44.527436
 04023282 | CV-03 |       | 2025-06-04 19:46:44.530213
 03054544 | PY-35 |       | 2025-06-04 19:46:44.531681
 06011853 | FA-57 |       | 2025-06-04 19:46:44.53232
 04097880 | EG-15 |       | 2025-06-04 19:46:44.53256
 08025464 | SO-26 |       | 2025-06-04 19:46:44.533826
 03083631 | BS-15 |       | 2025-06-04 19:46:44.535014
 03030349 | FE-05 |       | 2025-06-04 19:46:44.535627
 01078882 | EL-16 |       | 2025-06-04 19:46:44.537416
```

图 54: 删除后（前半）

```
 06078489 | MT-27 |       | 2025-06-04 19:46:44.660172
 09097921 | AS-28 |       | 2025-06-04 19:46:44.661012
 05024649 | EN-01 |       | 2025-06-04 19:46:44.661455
 03096052 | ST-01 |       | 2025-06-04 19:46:44.662369
 07055635 | NR-04 |       | 2025-06-04 19:46:44.662732
 03084589 | LW-12 |       | 2025-06-04 19:46:44.66491
 03054544 | EG-28 |       | 2025-06-04 19:46:44.665371
 01086810 | CV-58 |       | 2025-06-04 19:46:44.667034
 01091137 | NC-10 |       | 2025-06-04 19:46:44.670674
 01035245 | LW-34 |       | 2025-06-04 19:46:44.673142
 03095980 | AC-42 |       | 2025-06-04 19:46:44.675322
 05011830 | JP-04 |       | 2025-06-04 19:46:44.675835
 06050403 | MG-39 |       | 2025-06-04 19:46:44.676325
 02090660 | LW-13 |       | 2025-06-04 19:46:44.677033
 08053442 | EL-31 |       | 2025-06-04 19:46:44.677188
 09073372 | ID-44 |       | 2025-06-04 19:46:44.677481
 06087079 | CN-19 |       | 2025-06-04 19:46:44.679321
 07033047 | PL-53 |       | 2025-06-04 19:46:44.683442
 05065246 | MM-31 |       | 2025-06-04 19:46:44.683606
 06036693 | JP-27 |       | 2025-06-04 19:46:44.683709
 08019210 | JP-04 |       | 2025-06-04 19:46:44.683848
 03048868 | EG-52 |       | 2025-06-04 19:46:44.684256
 02024225 | FA-24 |       | 2025-06-04 19:46:44.6853
 07053767 | CS-15 |       | 2025-06-04 19:46:44.68589
 04057743 | LW-04 |       | 2025-06-04 19:46:44.68619
 05057135 | EL-02 |       | 2025-06-04 19:46:44.686371
 09051886 | AT-75 |       | 2025-06-04 19:46:44.688158
 07033989 | LT-43 |       | 2025-06-04 19:46:44.688311
(101 rows)
```

图 55: 删除后（后半）

可以看到，在测试删除的 1 行数据的基础上，删除后添加了 100 行新数据，且被删除的数据均为成绩为空的数据。