

第三次 “数据库原理及应用” 课程作业

在一个成绩管理系统数据库 GradeDB 中，定义如下关系表：

STUDENT(SID,SName,Age,Sex)

GRADE(SID,CID,Score, Note)

COURSE(CID,CName,Teacher)

- (1) 编写 SQL 程序创建数据库关系表，并插入 10 个学生的 2 门课程（“数据库原理及应用”、“数据结构与算法”）成绩数据；
- (2) 编写触发器程序实现 STUDENT 表、COURSE 表与 GRADE 表之间的级联更新、级联删除；
- (3) 编写存储过程程序实现统计各课程不及格学生人数，并在屏幕输出；
- (4) 用 Mybatis+Servlet+jsp 编程实现查询课程学生成绩列表页面输出功能。

作业要求：在 PostgreSQL 数据库中，使用 SQL 程序创建 GradeDB 数据库及其数据库表，并插入样本数据。分别使用 PL/pgSQL 编程方法和 Java Web 编程方法，实现数据库后端编程和应用前端编程程序。给出每个问题解决的步骤、SQL 程序、PL/pgSQL 程序和 Java Web 程序，及其运行结果界面，并对结果进行说明。

作业文件格式：作业 3_学号_姓名.doc

作业成绩评价标准：

正确完成情况	优	优	优	良	良	良	良	良	中	中
作业过程情况	优	优	良	优	良	良	良	中	中	中
文档规范性	优	良	良	优	优	良	中	中	中	差
作业评分	100-98	97-95	94-92	91-89	88-85	84-82	81-79	78-76	75-73	72-70

(1)

利用 PowerDesigner 建模工具对数据库进行简单的建模，考虑到本次实验的实际运用，主键的类型不使用代理键（serial），防止意外的情况产生。grade 表中的 sid 和 cid 既是主键又是外键，建模结果如图 1-1 所示。

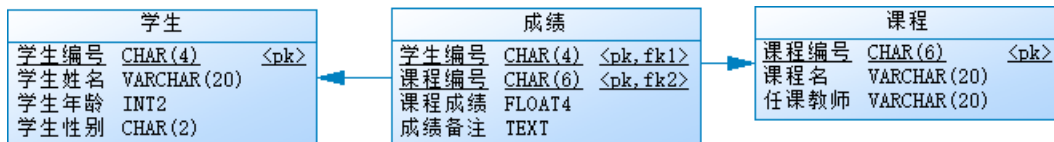


图 1-1 GradeDB 物理模型

生成 SQL 文件并导入到 PostgreSQL 数据库中，结果如图 1-2 所示。SQL 代码可见附录中的代码 1。考虑到本次实验的实际运用，在此处取消 grade 表中 sid 和 cid 的外键约束。

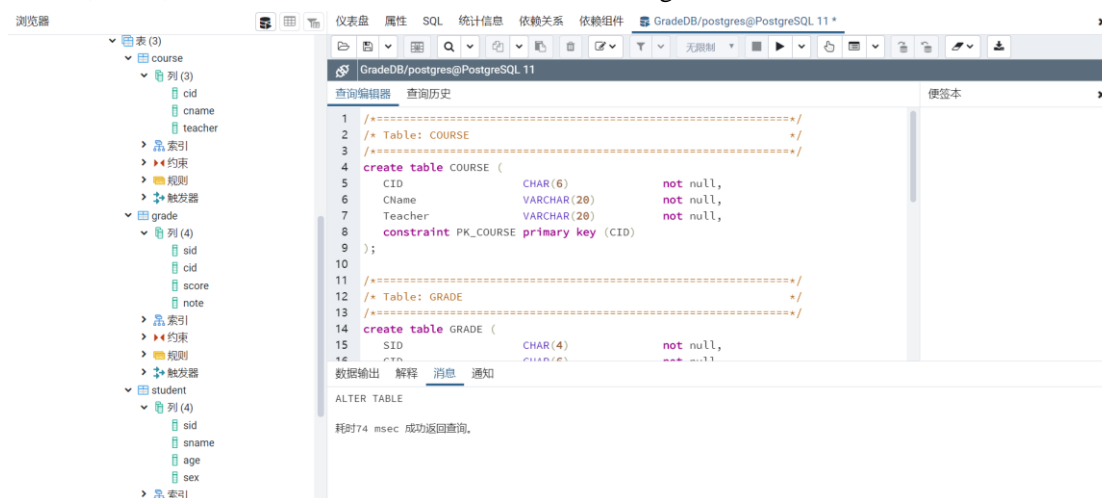


图 1-2 创建 GradeDB 数据库

在 GradeDB 数据库的 student 数据表中插入十个学生信息，此处参考 2018 年全国姓名报告发布结果，分别选择男生和女生五个登记使用最多的名字作为学生姓名，结果如图 1-3 所示。

	sid [PK] character (4)	sname character varying (20)	age smallint	sex character (2)
1	1001	浩宇	21	男
2	1002	浩然	21	男
3	1003	宇轩	20	男
4	1004	宇航	22	男
5	1005	宇泽	20	男
6	1006	梓涵	21	女
7	1007	一诺	22	女
8	1008	欣怡	19	女
9	1009	诗涵	20	女
10	1010	依诺	21	女

图 1-3 插入学生信息

在 course 数据表中插入“数据库原理及应用”和“数据结构与算法”两门课程信息，结果如图 1-4 所示。

	cid [PK] character (6)	cname character varying (20)	teacher character varying (20)
1	201801	数据库原理及应用	文军
2	201802	数据结构与算法	张栋梁

图 1-4 插入课程信息

在 grade 数据表中插入十个学生的两门课程的成绩信息，结果如图 1-5 所示。

	sid [PK] character (4)	cid [PK] character (6)	score real	note text
1	1001	201801	99	无
2	1002	201801	94	无
3	1003	201801	90	无
4	1004	201801	86	无
5	1005	201801	86	无
6	1006	201801	77	无
7	1007	201801	60	无
8	1008	201801	55	无
9	1009	201801	45	无
10	1010	201801	22	无
11	1010	201802	97	无
12	1009	201802	88	无
13	1008	201802	88	无
14	1007	201802	77	无
15	1006	201802	59	无
16	1005	201802	55	无
17	1004	201802	44	无
18	1003	201802	44	无
19	1002	201802	41	无
20	1001	201802	40	无

图 1-5 插入成绩信息

(2)

首先分别创建触发器函数 `grade_update_by_student()`和 `grade_update_by_course()`，分别实现 `student` 表和 `course` 表的修改的级联更新和级联删除，如代码 2-1 和代码 2-2 所示。

代码 2-1 `grade_update_by_student` 触发器函数

```
CREATE FUNCTION grade_update_by_student() RETURNS TRIGGER AS $grade_update_by_student$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        DELETE FROM public.grade
        WHERE sid=OLD.sid;
        RETURN OLD;
    ELSIF (TG_OP = 'UPDATE') THEN
        UPDATE public.grade
        SET sid=NEW.sid
        WHERE sid=OLD.sid;
        RETURN NEW;
    END IF;
    RETURN NULL;
END;
$grade_update_by_student$ LANGUAGE plpgsql;
```

代码 2-2 `grade_update_by_course` 触发器函数

```
CREATE FUNCTION grade_update_by_course() RETURNS TRIGGER AS $grade_update_by_course$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        DELETE FROM public.grade
        WHERE cid=OLD.cid;
        RETURN OLD;
    ELSIF (TG_OP = 'UPDATE') THEN
        UPDATE public.grade
        SET cid=NEW.cid
        WHERE cid=OLD.cid;
        RETURN NEW;
    END IF;
    RETURN NULL;
END;
$grade_update_by_course$ LANGUAGE plpgsql;
```

接下来使用 `CREATE TRIGGER` 创建触发器，绑定触发器函数，如代码 2-3 和代码 2-4 所示。

代码 2-3 `updateGradeInfo_byStudent` 触发器

```
CREATE TRIGGER updateGradeInfo_byStudent
AFTER DELETE OR UPDATE ON public.student
FOR EACH ROW EXECUTE PROCEDURE grade_update_by_student();
```

代码 2-4 updateGradeInfo_byCourse 触发器

```
CREATE TRIGGER updateGradeInfo_byCourse
AFTER DELETE OR UPDATE ON public.course
FOR EACH ROW EXECUTE PROCEDURE grade_update_by_course();
```

在测试触发器之前，创建 public 架构的备份。

更新或删除 student 表和 course 表中的数据行，测试触发器。删除 student 表中的某一个学生信息，查看 grade 表中记录的结果，如代码 2-5 和图 2-1 所示；修改 course 表中 cid 的值，查看 grade 表中记录的结果，如代码 2-6 和图 2-2 所示。

代码 2-5 删除学生信息

```
DELETE FROM public.student
WHERE sid='1001';
```

	sid [PK] character (4)	cid [PK] character (6)	score real	note text
1	1010	201802	97	无
2	1009	201802	88	无
3	1008	201802	88	无
4	1007	201802	77	无
5	1006	201802	59	无
6	1005	201802	55	无
7	1004	201802	44	无
8	1003	201802	44	无
9	1002	201802	41	无
10	1002	201801	94	无
11	1003	201801	90	无
12	1004	201801	86	无
13	1005	201801	86	无
14	1006	201801	77	无
15	1007	201801	60	无
16	1008	201801	55	无
17	1009	201801	45	无
18	1010	201801	22	无

图 2-1 删除学生信息结果

代码 2-6 修改课程信息

```
UPDATE public.course
SET cid='201803'
```

WHERE cid='201801';





	 sid [PK] character (4)	 cid [PK] character (6)	 score real	 note text
1	1010	201802	97	无
2	1009	201802	88	无
3	1008	201802	88	无
4	1007	201802	77	无
5	1006	201802	59	无
6	1005	201802	55	无
7	1004	201802	44	无
8	1003	201802	44	无
9	1002	201802	41	无
10	1001	201802	40	无
11	1001	201803	99	无
12	1002	201803	94	无
13	1003	201803	90	无
14	1004	201803	86	无
15	1005	201803	86	无
16	1006	201803	77	无
17	1007	201803	60	无
18	1008	201803	55	无
19	1009	201803	45	无
20	1010	201803	22	无

图 2-2 修改课程信息结果

(3)

创建存储过程 count_fail_records(character), 如代码 3-1 所示, 其中括号中的参数为课程的课程编号。该存储过程返回值为 grade 表中 score 小于 60 且 cid 等于传入参数的行数, 即该课程不及格的人数。

代码 3-1 创建存储过程 count_fail_records(character)

```
CREATE FUNCTION count_fail_records(char(6))
RETURNS integer AS $body$
DECLARE
    countNum integer;
BEGIN
    SELECT COUNT(*) INTO countNum FROM public.grade WHERE score < 60 AND cid = $1;
    RETURN countNum;
END;
$body$ LANGUAGE plpgsql;
```

创建视图 course_fail_records_view 实现对查询各科目不及格人数功能的封装, 如代码 3-2 所示。

代码 3-2 创建视图 course_fail_records_view

```
CREATE OR REPLACE VIEW course_fail_records_view AS
SELECT c.cname AS 课程名称, count_fail_records(c.cid) AS 不及格人数
FROM course AS c;
```

调用视图得到结果, 如代码 3-3 和图 3-1 所示。

代码 3-3 调用视图 course_fail_records_view

```
SELECT * FROM course_fail_records_view;
```

	课程名称 character varying (20)	不及格人数 integer
1	数据结构与算法	6
2	数据库原理及应用	3

图 3-1 查询不及格人数结果

(4)

在 Tomcat 的官方网站下载安装稳定的 [Tomcat](#) 9.0.35 版本，更新 Java 至 jre1.8.0_251，下载 PostgreSQL 驱动程序 [JDBC](#) 42.2.12，选择 IntelliJ IDEA 2020.1 作为 JavaWeb 开发 IDE 并配置开发环境。

创建视图 `course_student_score_view` 实现对查询不同课程学生的分数功能的封装，如代码 4-1 所示。

代码 4-1 创建视图 `course_student_score_view`

```
CREATE OR REPLACE VIEW course_student_score_view AS
SELECT c.cname AS 课程名称, s.sname AS 学生姓名, g.score AS 分数
FROM course AS c, student AS s, grade AS g
WHERE c.cid=g.cid AND s.sid=g.sid;
```

编写 Java 类 `PgsqlConnect`，实现访问本地的 `GradeDB` 数据库和获取需要的学生成绩信息，如代码 4-2 所示。其中 `getScoreInfo (String courseName)` 函数的参数 `courseName` 为查询的课程名称，该函数的返回值为存储 `HashMap` 的 `ArrayList`，`HashMap` 的值如 {分数=94.0, 学生姓名=浩然} 的形式存储。

代码 4-2 `pgsql.PgsqlConnect.java`

```
package pgsql;

import java.sql.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class PgsqlConnect {
    public static ArrayList<Map<String, String>> getScoreInfo (String courseName) {
        Connection conn = null;
        Statement stat = null;
        ArrayList<Map<String, String>> arrayList = new ArrayList<>();
        try {
            Class.forName("org.postgresql.Driver");
            String url = "jdbc:postgresql://127.0.0.1:5432/GradeDB";
            String username = "postgres";
            String password = "root";
            conn = DriverManager.getConnection(url, username, password);
            String sql = "SELECT * FROM course_student_score_view";
            stat = conn.createStatement();
            ResultSet result = stat.executeQuery(sql);
            while (result.next()) {
                String getCourseName = result.getString("课程名称");
                if (getCourseName.equals(courseName)) {
                    float scoreDouble = result.getFloat("分数");
                    String score = Float.toString(scoreDouble);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return arrayList;
    }
}
```



```

        Map<String, String> map = new HashMap<>();
        map.put("学生姓名", result.getString("学生姓名"));
        map.put("分数", score);

        arrayList.add(map);
    }
}
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (stat != null) {
            stat.close();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    try {
        if (conn != null) {
            conn.close();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
return arrayList;
}
}
}

```

编写 JSP 文件 index.jsp，在不考虑任何美观的情况下实现对学生成绩的查询，如代码 4-3 所示。设置一个 form 实现下拉选单查询，点击查询按钮后，通过 Servlet 获取欲查询的课程名称，调用 PgsqIConnect.getScoreInfo(courseName)获取数据库查询结果，最后将数据显示在页面上，如图 4-1 和 4-2 所示。

代码 4-3 index.jsp

```

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page import="pgsql.PgsqIConnect" %>
<%@ page import="java.util.ArrayList" %>
<%@ page import="java.util.Map" %>
<html>
    <head>
        <title>学生成绩查询系统</title>
    </head>
    <body>

```

```

<form>
  <select name="courseSelect">
    <option value="数据库原理及应用">数据库原理及应用</option>
    <option value="数据结构与算法">数据结构与算法</option>
  </select>
  <input type="submit" value="查询">
</form>
<pre>
<%
    request.setCharacterEncoding("UTF-8");
    String courseName = request.getParameter("courseSelect");
    out.println("\n 查询课程: " + courseName);
    ArrayList<Map<String, String>> arrayList = PgsqlConnect.getScoreInfo(courseName);
    out.println("学生姓名"+"\""+ "分数");
    for (Map<String, String> stringStringMap : arrayList) {
        out.println(stringStringMap.get("学生姓名") + "\"\t\" + stringStringMap.get("分数"));
    }
%>
</pre>
</body>
</html>

```

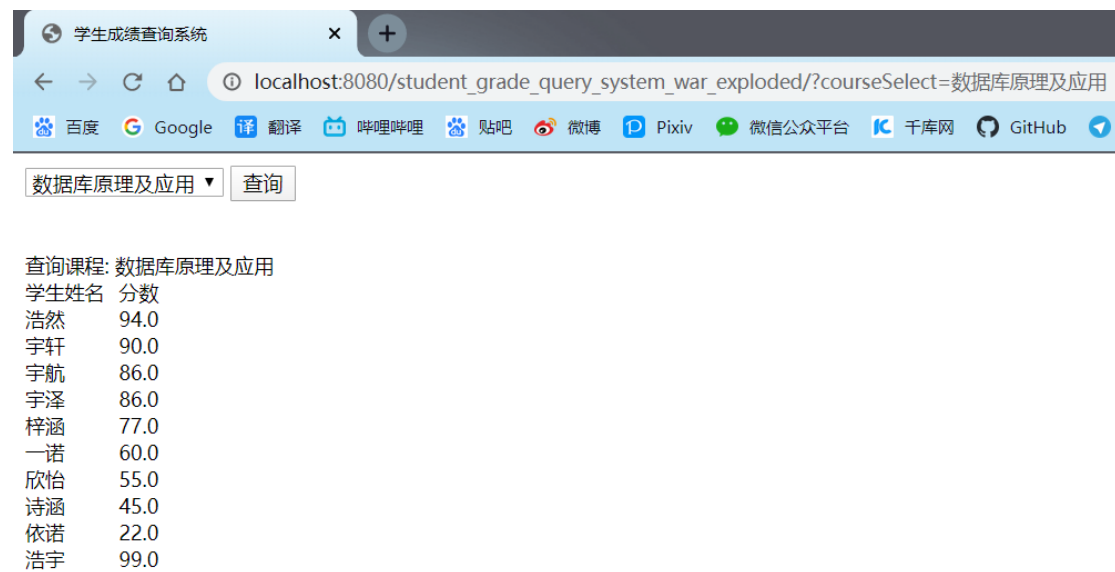


图 4-1 查询数据库原理及应用结果

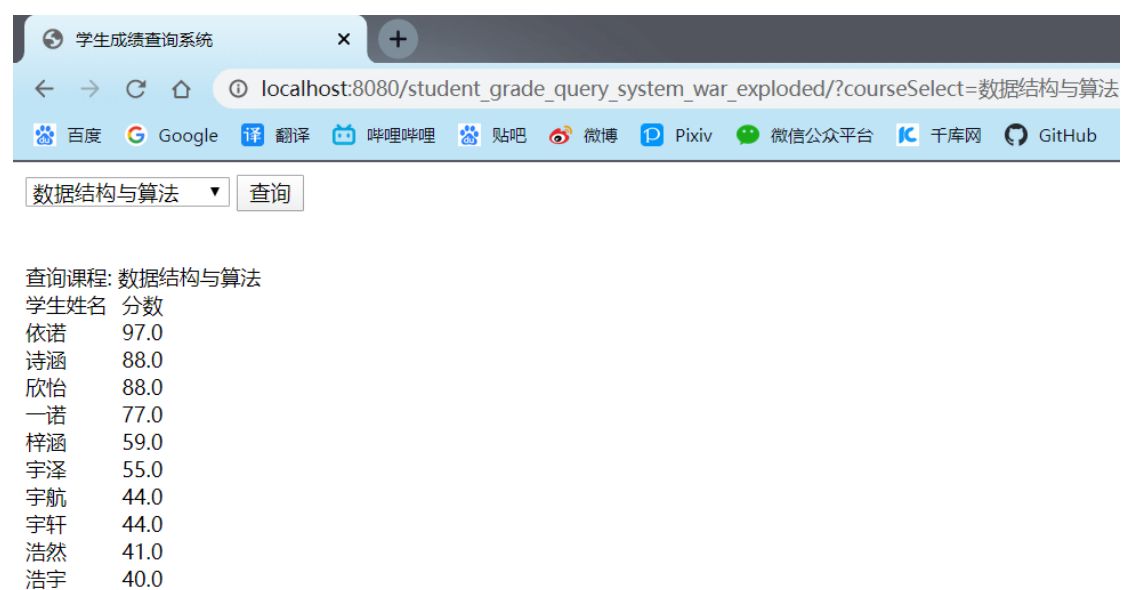


图 4-2 查询数据结构与算法结果

附录

代码 1 数据库建模代码

```
/*=====*/
/* DBMS name:      PostgreSQL 9.x                      */
/*=====*/

drop table COURSE;

drop table GRADE;

drop table STUDENT;

/*=====*/
/* Table: COURSE                                     */
/*=====*/
create table COURSE (
    CID          CHAR(6)          not null,
    CName        VARCHAR(20)      not null,
    Teacher      VARCHAR(20)      not null,
    constraint PK_COURSE primary key (CID)
);

/*=====*/
/* Table: GRADE                                       */
/*=====*/
create table GRADE (
    SID          CHAR(4)          not null,
    CID          CHAR(6)          not null,
    Score        FLOAT4          not null,
    Note         TEXT            null,
    constraint PK_GRADE primary key (SID, CID)
);

/*=====*/
/* Table: STUDENT                                    */
/*=====*/
create table STUDENT (
    SID          CHAR(4)          not null,
    SName        VARCHAR(20)      not null,
    Age          INT2            not null,
    Sex          CHAR(2)         not null,
    constraint PK_STUDENT primary key (SID)
);
```

```
alter table GRADE
```

```
  add constraint FK_GRADE_REFERENCE_STUDENT foreign key (SID)  
    references STUDENT (SID)  
    on delete restrict on update restrict;
```

```
alter table GRADE
```

```
  add constraint FK_GRADE_REFERENCE_COURSE foreign key (CID)  
    references COURSE (CID)  
    on delete restrict on update restrict;
```