Spring 2021 CS307 Project Report

1. Title:

Educational Administration System Inplementing and Testing

2. Group info and Contribution

Student Name:	Student ID:	Specific contribution content	Percentage
Tan Sixu(谈思序)	11911627	Prerequisite design, Python server	33.3%
Zhao Yunlong(赵云龙)	11911309	Database import, web	33.3%
Jiang Runzhe(蒋润喆)	11912511	Database design, High currency,	33.3%
Andy Lau(于德华)	88015127	Spiritual leader(逃げる)	0.1%

3. Catalog

4. Brief Introduction

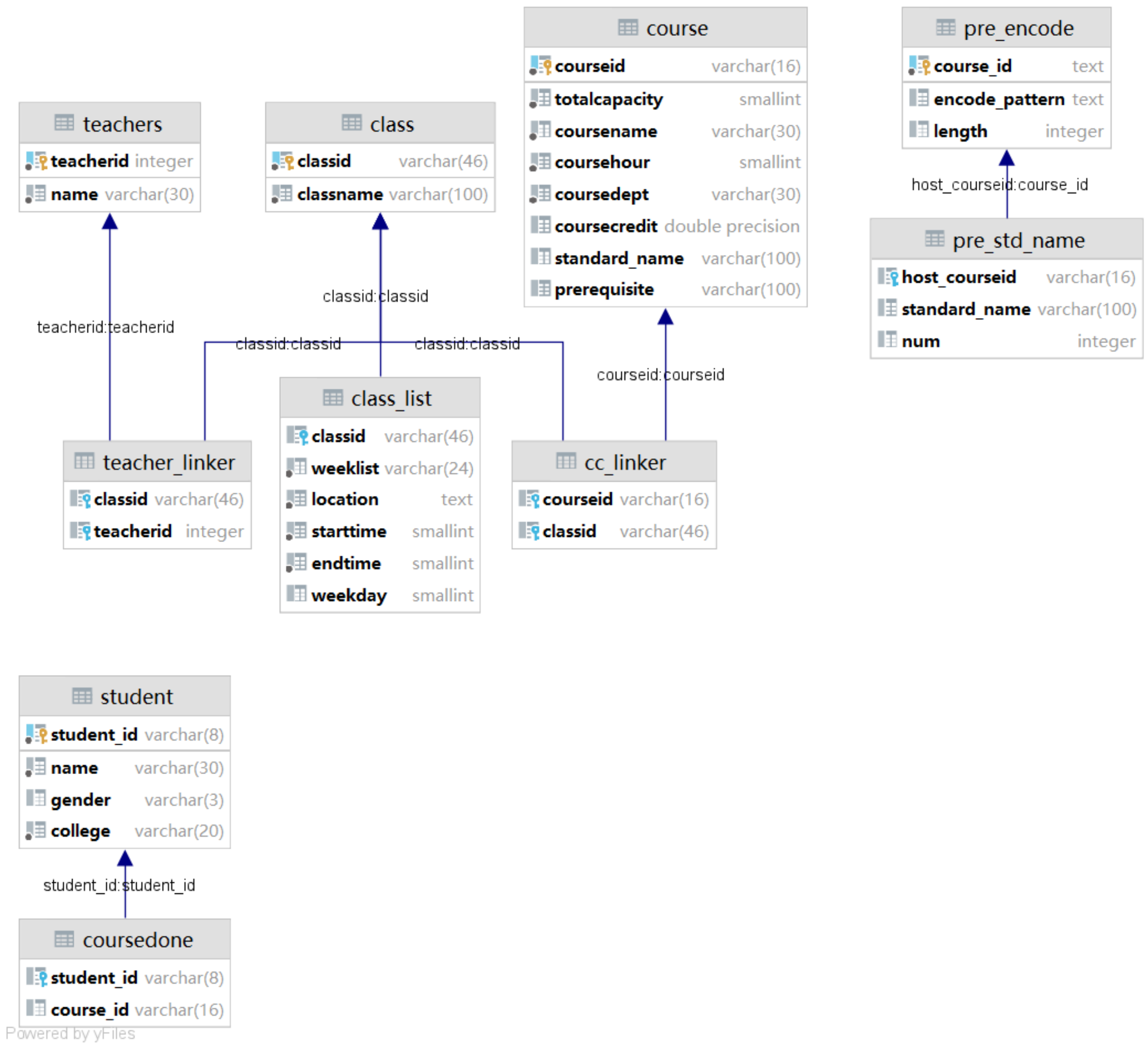
The 21th century is filled with machines and data, thus we call it an era of big data and AI. Therefore, how to manage these data may be a very challenging as well as troublesome task. With the development of science and technology, DBMS (database management system) springs up and shows how powerful it is, helping numerous computer scientists and engineers to study and research, in order to push up the efficiency and accuracy of dealing with data. Nowadays, these systems are totally commercial and somehow mature to certain extent.

In this project, we want to use DBMS to implement a multi-dimension educational information database which is easy to extend and can have further evolved into an educational administration system(this part is also roughly accomplished by the time we submit our report). We import all the given data and assured the proper functioning of our database. All kinds of instructions were given for testing the performance and efficiency of our database as well.

5. DB Design

Task 1 Database Design

<1.1> The structure of the whole table



<1.2> Data Table(except prerequisite)

Three given files are respectively provided , with many columns which have not been separated and are not total clear. In order to show the pristine organization of the data, here presents several tables with original data, data type and extra short explanation. (basic info can be easily recognized from columns' name, so only important parts are explained specifically)

Table: course

Columns' Name	Data Type	Short Explanation
courseid	varchar(16)	not null, primary key
totalcapacity	smallint	not null , will do the check to assure it's> 0
coursename	varchar(30)	not null, course name
coursehour	smallint	not null , will do the check to assure it's>= 0
coursedept	varchar(30)	not null, get the department
coursecredit	double precision	will do the check to assure it's>= 0 and<100
standard_name	varchar(100)	
prerequisite	varchar(100)	

Table: class_list

Columns' Name	Data Type	Short Explanation
classid	varchar(46)	foreign key
weeklist	weeklist(24)	not null
location	text	not null
starttime	smallint	not null , will do the check to assure it's between 1 to 11
endtime	smallint	not null , will do the check to assure it's between 2 to 12
weekday	smallint	will do the check to assure it's between 1 to 8

Table: class

Columns' Name	Data Type	Short Explanation
classId	varchar(46)	primary key
className	varchar(20)	

Table: teachers

Columns' Name	Data Type	Short Explanation
teacherId	serial	primary key
name	varchar(30)	not null

Table: student

Columns' Name	Data Type	Short Explanation
student_id	varchar(8)	primary key
name	varchar(30)	not null
gender	varchar(3)	m/f/null
college	varchar(20)	not null

Table: coursedone

Columns' Name	Data Type	Short Explanation
student_id	varchar(8)	foreign key
course_id	varchar(16)	

<1.3> Linker Table

Table: teacher_linker

connect table-class to table-teachers

Columns' Name	Data Type	Short Explanation
classid	varchar(46)	foreign key
teacherid	int	foreign key

Table: cc_linker

connect table-course to table-class

Columns' Name	Data Type	Short Explanation
courseId	varchar(16)	foreign key
classId	varchar(30)	foreign key

<1.4> Prerequisite Table

This part is significant for its unique implementation

Table: pre_std_name

Columns' Name	Data Type	Short Explanation
courseid	varchar(16)	
standard_name	varchar(30)	
num	integer	

Table: pre_encode

Columns' Name	Data Type	Short Explanation
course_id	text	
encode_pattern	text	
length	integer	

Explanation:

This design works with any pattern of prerequisite, no matter how complicated the brackets and logical connections are.
To design a pattern that fit all kinds of given string of prerequisite, we construct Two assistant tables, namely *pre_encode* and *pre_std_name*.

Our basic idea is to avoid interpreting the annoying string of prerequisites. In order to fulfill this, the *Regular expression* is employed to replace all names of prerequisite courses with placeholder "%d", forming the encode_pattern which will be stored in table *pre_encode* while leaving the original brackts and logical connections unchanged. Meanwhile, the names of prerequisites are stored in the table *pre_std_name*. When checking whether the prerequisites are satisfied, we can again replace the placeholders with 0 or 1 that returned from queries in database. The by dynamically excute the encode_pattern in Python, we can get the result.

> e.g. rough example may look like this:

> Given course "BIO304" with prerequisites"(普通生物学 或者 普通生物学) 并且 (概率论与数理统计 或者 (概率论 并且 (数理统计 或者 数理统计)))"

After transition, the tables will be like:

pre_encode:

host_course_id	encode_pattern
BIO304	(%d or %d) and (%d or (%d and (%d or %d)))

pre_std_name:

host_course_id	std_name
BIO304	'普通生物学'
BIO304	'普通生物学'
BIO304	'概率论与数理统计'
BIO304	'概率论'
BIO304	'数理统计'
BIO304	'数理统计'

Suppose there is a student, say A, who has taken the following courses:

Name	courses_done
A	'普通生物学'
A	'数理统计'

Then by the index of joining *course_done* and *pre_std_name*, we replace the placeholder with [1,0,0,0,1,0], the encode_pattern will then be like

$$(1 or 0) and (0 or (0 and (1 or 0))) \tag{1}$$

The we just evaluate the logic expression by

```
    expression=f"satisfied= {encoded}"%tuple(logic)     #replace
    exec(expression)
```

Hence in this way we can check the satisfaction of prerequisite.

<1.5> Code

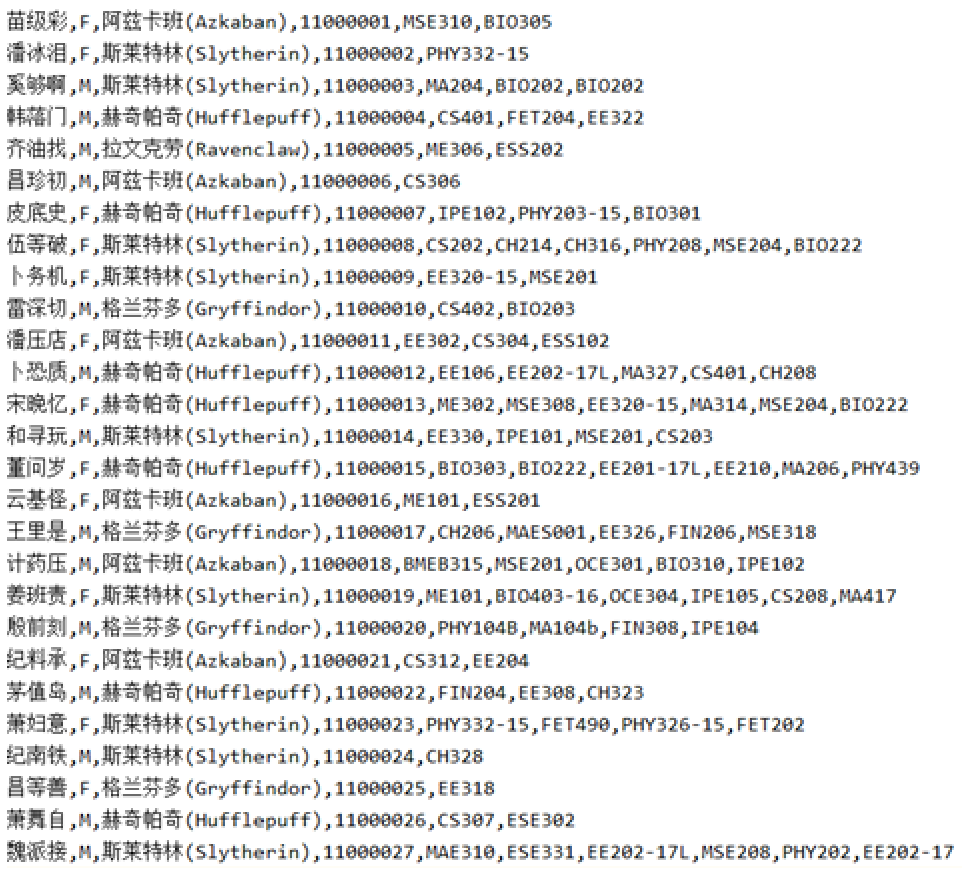
```sql
CREATE TABLE course(
    courseId varchar(16)  not null primary key ,
    totalCapacity smallint not null check ( totalCapacity > 0 ),
    courseName varchar(30)  not null,
    courseHour smallint not null check ( courseHour>=0 ),
    courseDept varchar(20) not null,
    courseCredit float check((courseCredit>=0) and (courseCredit<100)),
    standard_name varchar(20),
    prerequisite varchar(100)
);

CREATE TABLE class(
    classId varchar(46) primary key,
    className varchar(20) not null
);

CREATE TABLE class_list(
    classId varchar(46) references class(classId),
    --TODO: change this str to another table
    weekList varchar(24) not null,
    location varchar(20) not null,
    -- start time should between 1 and 10,between是左闭右开[,)[1,11)
    startTime smallint not null check ( startTime between 1 and 11),
    -- [2,12)
    endTime smallint not null check ( endTime>startTime and endTime between 2 and 12),
    weekday smallint check(weekday between 1 and 8)--[1,8)
);

CREATE TABLE cc_linker(
    courseId varchar(16) references course(courseId),
    classId varchar(46) references class(classId)
);


CREATE TABLE Teachers(
    teacherId serial primary key,
    --因为有可能会有一个很长的英文名:GARG NAVEEN KUMAR
    name varchar(30) not null
);

CREATE TABLE Teacher_linker(
    courseId varchar(46) references course(courseId),
    teacherId int references Teachers(teacherId)
);

CREATE TABLE Student(
      name varchar(30) not null,
      gender varchar(3),
      college varchar(20) not null,
      student_id varchar(8) primary key

);
```

```sql

CREATE TABLE CourseDone(
    student_id varchar(8) references Student(student_id),
    course_id varchar(16) references course(courseId)
);

create table pre_encode(
    courseid varchar(16),
    encode varchar(100),
    length integer
);

create table pre_std_name(
    courseid varchar(16),
    standard_name varchar(100),
    num integer
);
```

Task 2 Import Data

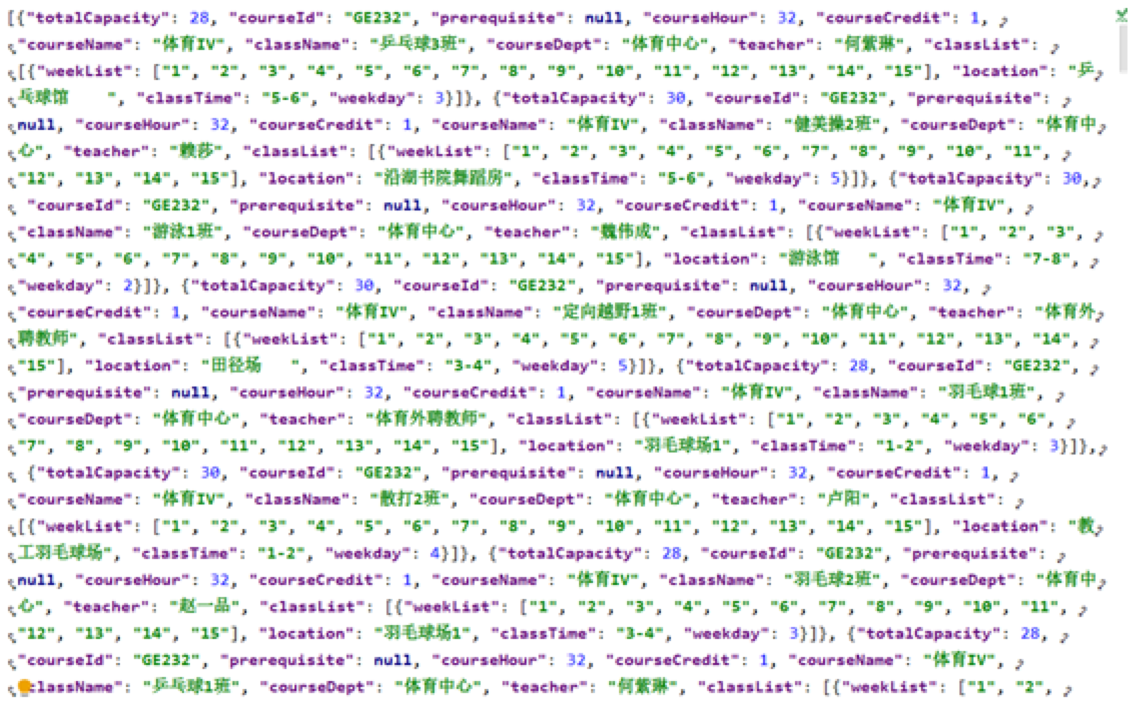
<2.1> Data preprocessing

if we directly open the select_course.csv, course_info.json , They will perform in form below:



```
苗级彩,F,阿兹卡班(Azkaban),11000001,MSE310,BIO305
潘冰泪,F,斯莱特林(Slytherin),11000002,PHY332-15
奚够啊,M,斯莱特林(Slytherin),11000003,MA204,BIO202,BIO202
韩落门,M,赫奇帕奇(Hufflepuff),11000004,CS401,FET204,EE322
齐油找,M,拉文克劳(Ravenclaw),11000005,ME306,ESS202
昌珍切,M,阿兹卡班(Azkaban),11000006,CS306
皮底史,F,赫奇帕奇(Hufflepuff),11000007,IPE102,PHY203-15,BIO301
伍等破,F,斯莱特林(Slytherin),11000008,CS202,CH214,CH316,PHY208,MSE204,BIO222
卜务机,F,斯莱特林(Slytherin),11000009,EE320-15,MSE201
雷深切,M,格兰芬多(Gryffindor),11000010,CS402,BIO203
潘压店,F,阿兹卡班(Azkaban),11000011,EE302,CS304,ESS102
卜恐质,M,赫奇帕奇(Hufflepuff),11000012,EE106,EE202-17L,MA327,CS401,CH208
宋晚忆,F,赫奇帕奇(Hufflepuff),11000013,ME302,MSE308,EE320-15,MA314,MSE204,BIO222
和寻玩,M,斯莱特林(Slytherin),11000014,EE330,IPE101,MSE201,CS203
董问岁,F,赫奇帕奇(Hufflepuff),11000015,BIO303,BIO222,EE201-17L,EE210,MA206,PHY439
云基怪,F,阿兹卡班(Azkaban),11000016,ME101,ESS201
王里是,M,格兰芬多(Gryffindor),11000017,CH206,MAES001,EE326,FIN206,MSE318
计药压,M,阿兹卡班(Azkaban),11000018,BMEB315,MSE201,OCE301,BIO310,IPE102
姜班责,F,斯莱特林(Slytherin),11000019,ME101,BIO403-16,OCE304,IPE105,CS208,MA417
殷前刻,M,格兰芬多(Gryffindor),11000020,PHY104B,MA104b,FIN308,IPE104
纪料承,F,阿兹卡班(Azkaban),11000021,CS312,EE204
茅值岛,M,赫奇帕奇(Hufflepuff),11000022,FIN204,EE308,CH323
萧妇意,F,斯莱特林(Slytherin),11000023,PHY332-15,FET490,PHY326-15,FET202
纪南铁,M,斯莱特林(Slytherin),11000024,CH328
昌等善,F,格兰芬多(Gryffindor),11000025,EE318
萧舞自,M,赫奇帕奇(Hufflepuff),11000026,CS307,ESE302
魏派接,M,斯莱特林(Slytherin),11000027,MAE310,ESE331,EE202-17L,MSE208,PHY202,EE202-17
```

select_course.csv



```
[{"totalCapacity": 28, "courseId": "GE232", "prerequisite": null, "courseHour": 32, "courseCredit": 1,
"courseName": "体育IV", "className": "乒乓球3班", "courseDept": "体育中心", "teacher": "何紫琳", "classList":
[{"weekList": ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15"], "location": "乒
乓球馆    ", "classTime": "5-6", "weekday": 3}]}, {"totalCapacity": 30, "courseId": "GE232", "prerequisite":
null, "courseHour": 32, "courseCredit": 1, "courseName": "体育IV", "className": "健美操2班", "courseDept": "体育中
心", "teacher": "赖莎", "classList": [{"weekList": ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",
"12", "13", "14", "15"], "location": "沿湖书院舞蹈房", "classTime": "5-6", "weekday": 5}]}, {"totalCapacity": 30,
 "courseId": "GE232", "prerequisite": null, "courseHour": 32, "courseCredit": 1, "courseName": "体育IV",
"className": "游泳1班", "courseDept": "体育中心", "teacher": "魏伟成", "classList": [{"weekList": ["1", "2", "3",
"4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15"], "location": "游泳馆   ", "classTime": "7-8",
"weekday": 2}]}, {"totalCapacity": 30, "courseId": "GE232", "prerequisite": null, "courseHour": 32,
"courseCredit": 1, "courseName": "体育IV", "className": "定向越野1班", "courseDept": "体育中心", "teacher": "体育外
聘教师", "classList": [{"weekList": ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14",
"15"], "location": "田径场   ", "classTime": "3-4", "weekday": 5}]}, {"totalCapacity": 28, "courseId": "GE232",
"prerequisite": null, "courseHour": 32, "courseCredit": 1, "courseName": "体育IV", "className": "羽毛球1班",
"courseDept": "体育中心", "teacher": "体育外聘教师", "classList": [{"weekList": ["1", "2", "3", "4", "5", "6",
"7", "8", "9", "10", "11", "12", "13", "14", "15"], "location": "羽毛球场1", "classTime": "1-2", "weekday": 3}]},
 {"totalCapacity": 30, "courseId": "GE232", "prerequisite": null, "courseHour": 32, "courseCredit": 1,
"courseName": "体育IV", "className": "散打2班", "courseDept": "体育中心", "teacher": "卢阳", "classList":
[{"weekList": ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15"], "location": "教
工羽毛球场", "classTime": "1-2", "weekday": 4}]}, {"totalCapacity": 28, "courseId": "GE232", "prerequisite":
null, "courseHour": 32, "courseCredit": 1, "courseName": "体育IV", "className": "羽毛球2班", "courseDept": "体育中
心", "teacher": "赵一品", "classList": [{"weekList": ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",
"12", "13", "14", "15"], "location": "羽毛球场1", "classTime": "3-4", "weekday": 3}]}, {"totalCapacity": 28,
"courseId": "GE232", "prerequisite": null, "courseHour": 32, "courseCredit": 1, "courseName": "体育IV",
"className": "乒乓球1班", "courseDept": "体育中心", "teacher": "何紫琳", "classList": [{"weekList": ["1", "2",
```

course_info.json

As we can see here, it is far away from our expectation. So we turn to the Java to do some pre-processing. By the way, now our data have some error format like the mix-use of "," and "，". We also corrected these errors in order to smoothly load the data.

select_course.csv-java-preprocessing:

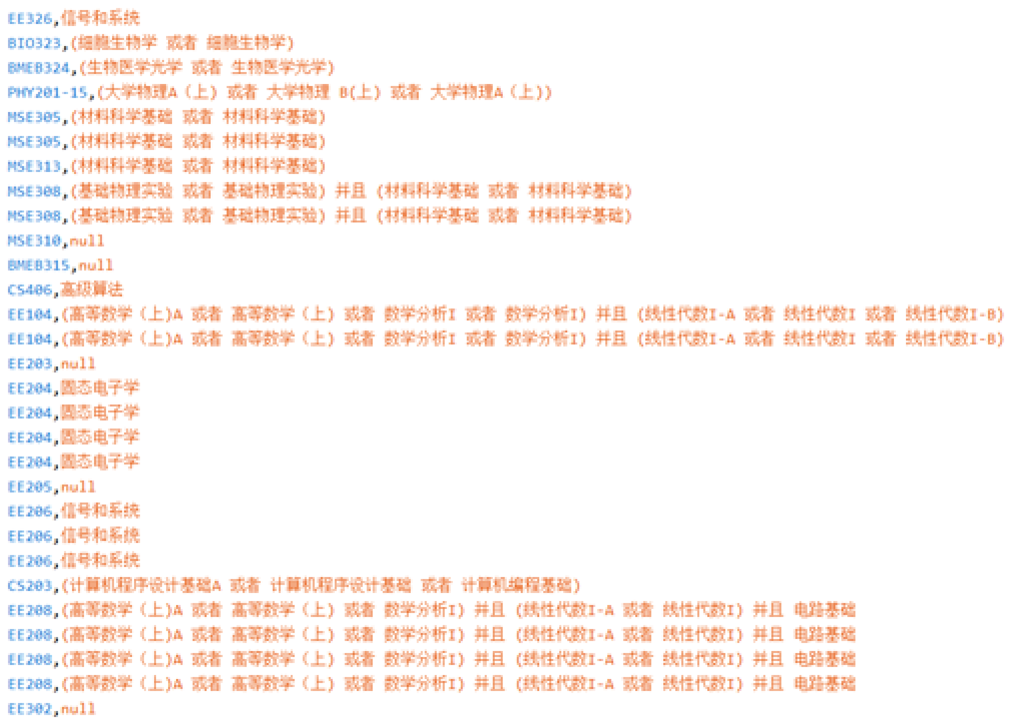
```java
public static void parseStudent() throws IOException {
        students = new ArrayList<>();
        File student_info = new File("src/main/java/data/select_course.csv");
        BufferedReader reader = null;
        try {
            String one_student = null;
            reader = new BufferedReader(new FileReader(student_info));
            while ((one_student = reader.readLine()) != null) {
                String[] s_info = one_student.split(",");
                Student student = new Student(s_info);
                students.add(student);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            reader.close();
        }
    }

static void putStudentIntoData() throws IOException {
        databaseConnnect = new DatabaseConnnect("jdbc:postgresql://localhost:5432/CS307_SustechStudentClass",
                "byll",
                "123456");

        //将学生信息导入数据库
        DatabaseConnnect.SendToDataBase(students);
        DatabaseConnnect.SendToDataBase(students,10);
        DatabaseConnnect.CloseConnection();
        //将学生信息写入sql文件
        DatabaseConnnect.writeToFileS(students);

    }
```

course_info.json-java-preprocessing:

```
//用作导入json文件
    public static void parseCourseJson() throws IOException {
        Path path = Path.of("src/main/java/data/course_info.json");
        String content = Files.readString(path);
        content = content.replaceAll("）", ")");
        content = content.replaceAll("（", "(");
        Gson gson = new Gson();
        courses = gson.fromJson(content, new TypeToken<List<CourseRAW>>() {
        }.getType());
    }
//用于导出先修课文件
    public static void exportPre() {
        HashSet<String> hasAdded = new HashSet<>();
        FileOutputStream out = null;
        OutputStreamWriter osw = null;
        BufferedWriter bw = null;
        try {
            out = new FileOutputStream("src/main/java/data/Pre.csv");
            osw = new OutputStreamWriter(out, StandardCharsets.UTF_8);
            bw = new BufferedWriter(osw);


            //加上UTF-8文件的标识字符
            out.write(new byte[]{(byte) 0xEF, (byte) 0xBB, (byte) 0xBF});

            for (CourseRAW c : courses) {
                if (hasAdded.contains(c.courseId)) continue;
                String insert = String.format("%s,%s\n", c.courseId, c.prerequisite);
                System.out.println(insert);
                bw.append(insert);
                hasAdded.add(c.courseId);
            }

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                assert bw != null;
                bw.close();
                osw.close();
                out.close();
            } catch (Exception e) {
                e.printStackTrace();
            }

        }
```

```java
    }
//用于解析从json导入的数据,放进自己想要的数据结构中
public static void parseCourseRAW() {
        courseHashMap = new HashMap<>();
        classes = new ArrayList<>();
        teachers = new HashSet<>();
        teacherHashMap=new HashMap<>();
        for (CourseRAW course_raw : courses) {
            //Course info_去重
            if (!courseHashMap.containsKey(course_raw.courseName.trim())) {
                Course co = new Course(course_raw.courseId.trim(),
                        course_raw.totalCapacity,
                        course_raw.courseName.trim(),
                        course_raw.courseHour,
                        course_raw.courseCredit,
                        course_raw.courseDept);
                co.prerequisite=course_raw.prerequisite;
                courseHashMap.put(co.course_name, co);
            }
            //Class info 不需要去重
            Class clAss = new Class(course_raw.className.trim(), courseHashMap.get(course_raw.courseName.trim()));
            for (ClassListRAW cl : course_raw.classList) {
                ClassList classList = new ClassList(cl);
                clAss.class_info_list.add(classList);
            }
            if(course_raw.teacher!=null){

                if (course_raw.teacher.contains(",")) {
                    String[] manyTeacher = course_raw.teacher.split(",");
                    for (String oneTeacher : manyTeacher) {

                        Teacher teacher = new Teacher(oneTeacher.trim());
                        teachers.add(teacher);
                        teacherHashMap.put(teacher.names, teacher);
                        clAss.teachers.add(teacher);
                    }
                } else {
                    Teacher teacher = new Teacher(course_raw.teacher.trim());
                    clAss.teachers.add(teacher);
                    teachers.add(teacher);
                    teacherHashMap.put(teacher.names, teacher);
                }
            }

            classes.add(clAss);

        }
    }
//将course信息导入数据库
    public static void putJWXTinData(){
        databaseConnnect = new DatabaseConnnect("jdbc:postgresql://localhost:5432/CS307_SustechStudentClass",
                "byll",
                "123456");

        for (Course c:courseHashMap.values()) {
            DatabaseConnnect.SendToDataBase(c);

        }

        for (Teacher teacher:teachers){
            DatabaseConnnect.SendToDataBase(teacher);
        }

        for (Class c:classes){
            DatabaseConnnect.SendToDataBase(c);
            DatabaseConnnect.SendToDataBase(c.course,c);
            for (ClassList classList:c.class_info_list){
                DatabaseConnnect.SendToDataBase(classList,c);
            }
            DatabaseConnnect.SendToDataBase(c,1);
        }


        DatabaseConnnect.CloseConnection();
    }
```

In order to solve prerequisite more conveniently, we load out the prerequisite and save them into file: Pre.csv. In order to make our database as easy to expand as possible. We had to split the long string of prerequisite and separately store them into the table. We used library pandas in python to do this part.



```
EE326,信号和系统
BIO323,(细胞生物学 或者 细胞生物学)
BMEB324,(生物医学光学 或者 生物医学光学)
PHY201-15,(大学物理A (上) 或者 大学物理 B(上) 或者 大学物理A (上))
MSE305,(材料科学基础 或者 材料科学基础)
MSE305,(材料科学基础 或者 材料科学基础)
MSE313,(材料科学基础 或者 材料科学基础)
MSE308,(基础物理实验 或者 基础物理实验) 并且 (材料科学基础 或者 材料科学基础)
MSE308,(基础物理实验 或者 基础物理实验) 并且 (材料科学基础 或者 材料科学基础)
MSE310,null
BMEB315,null
CS406,高级算法
EE104,(高等数学 (上)A 或者 高等数学 (上) 或者 数学分析I 或者 数学分析I) 并且 (线性代数I-A 或者 线性代数I 或者 线性代数I-B)
EE104,(高等数学 (上)A 或者 高等数学 (上) 或者 数学分析I 或者 数学分析I) 并且 (线性代数I-A 或者 线性代数I 或者 线性代数I-B)
EE203,null
EE204,固态电子学
EE204,固态电子学
EE204,固态电子学
EE204,固态电子学
EE205,null
EE206,信号和系统
EE206,信号和系统
EE206,信号和系统
CS203,(计算机程序设计基础A 或者 计算机程序设计基础 或者 计算机编程基础)
EE208,(高等数学 (上)A 或者 高等数学 (上) 或者 数学分析I) 并且 (线性代数I-A 或者 线性代数I) 并且 电路基础
EE208,(高等数学 (上)A 或者 高等数学 (上) 或者 数学分析I) 并且 (线性代数I-A 或者 线性代数I) 并且 电路基础
EE208,(高等数学 (上)A 或者 高等数学 (上) 或者 数学分析I) 并且 (线性代数I-A 或者 线性代数I) 并且 电路基础
EE208,(高等数学 (上)A 或者 高等数学 (上) 或者 数学分析I) 并且 (线性代数I-A 或者 线性代数I) 并且 电路基础
EE302,null
```

Pre.csv

(extra)Pre.csv-python-preprocessing:

```python
import re
import pandas as pd
import csv
import numpy as np
def encode(raw_pre):
    """
    Parameters
    ----------
    raw_pre : TYPE
        raw_pre is a string that contain the raw infomation of pre

    Returns
    -------
    final : TYPE
        return the encoded format of pre, with ease to do logic calculation to check pre
    """
    raw_pre=re.sub(r"（","(", raw_pre)     #change to standard ()
    raw_pre=re.sub(r"）",")", raw_pre)
    discard=re.sub(r"[(][^(\(|\))]{,9}[)]","-",raw_pre)   #remove all inner () and set flag
    discard=re.sub(r"\s","",discard)            #remove space, save relational ()
    remove_all_p=noP=re.sub(r"\(|\)", "",discard)   #remove all () to split couse name
    names=re.split("或者|并且", remove_all_p)     #get names
    final=discard
    for name in names:
        final=re.sub(name, "%d", final, count=1)          #change all names to %d
    final=re.sub("或者"," or ", final)                 #change to or and
    final=re.sub("并且", " and ", final)
    return final, len(names)
def get_course_name(raw_pre):
    """
    This function is used to get list of name of courses, in standard form(without any "()")
    Parameters
    ----------
    raw_pre : Str
        DESCRIPTION.
    Returns
    -------
    coursese : List
        list of course names
    """
    raw_pre=re.sub(r"（","(", raw_pre)     #change to standard ()
    raw_pre=re.sub(r"）",")", raw_pre)
    clean_pre=re.sub(r"\(|\)|\s", "", raw_pre)     #remove all () both inner and outer or space
    courses=re.split("或者|并且", clean_pre)
    return courses
def check_satisfy(encoded, pre_list, satified_list):
    """
    THIS function check whether pre are satisfied.
    Parameters
    ----------
    encoded : TYPE
        DESCRIPTION.
    pre_list : TYPE
        DESCRIPTION.
    satified_list : TYPE
        DESCRIPTION.
    Returns
    -------
    TYPE
        Note that to get local var, locals() needed.

    """
    logic=[0 for i in range(len(pre_list))]     #initialize logical list

    for index in satified_list:                # flag satisfied courses
index as 1
        logic[index]=1


    loc=locals()                                #this very tricky
    expression=f"satisfied= {encoded}"%tuple(logic)     #replace

    exec(expression)

    return loc['satisfied']
if __name__=='__main__':
    pre=open("Pre.csv", 'r',encoding='utf-8')

    with pre:
        tt=pd.read_csv(pre, names=['course','pre'])
    pattern=[]
    lenth=[]
    courses=[]
    nums=[]
    cid=[]
    raw_id=tt['course']
    raws=tt['pre']
    raws[raws.isnull()]=0
    for k in range(len(raws)):
        c=raw_id[k]
        r=raws[k]
        if r==0:
            pattern.append(nan)
            lenth.append(0)
            continue
        tmp_p, tmp_len=encode(r)
        pattern.append(tmp_p)
        lenth.append(tmp_len)
        tmp=get_course_name(r)
        for i in range(len(tmp)):
            cid.append(c)
            courses.append(tmp[i])
            nums.append(i)


    # pattern={'encode': pattern}
    # lenth={'len': lenth}

    # a=pd.DataFrame(pattern)
    # b=pd.DataFrame(lenth)

    # tt=tt.append(a, axis=1)
    # tt=tt.append(b,axis=1)
    code=np.array(tt['course']).tolist()

    df=pd.DataFrame({'course':code, 'encode':pattern,'len':lenth})
    df.to_csv('encode.csv',index=False, header=False)

    df2=pd.DataFrame({'id':cid, 'pre':courses, 'num':nums})
    df2.to_csv('pre_course.csv', index=False,
header=False,encoding='utf_8_sig')
```

<2.2> Insert data in DBMS

First of all , we created a database "CS307_SustechStudentClass" with user "byll" and set the password "123456". Then we wrote a java file "DatabaseConnect.java" to insert all the data from file into our database. In this file ,we firstly get connection with our database:

```
 DatabaseConnnect(String url, String user, String password) {
        try {
            connection = DriverManager.getConnection(url, user, password);
            System.out.println("Connection success" + connection);
            statement = connection.createStatement();
            statement.execute("set search_path = \"Public\"");
//
//              ResultSet resultSet = statement.executeQuery("select * from
course");
//
//              while (resultSet.next()) {
//                  System.out.println(resultSet.getString(1));
//              }

        } catch (Exception e) {
            System.out.println("Connection failed");
            e.printStackTrace();
        }

    }
```

Then we load the data from .csv file and store them into our DB step by step. We can see below part which insert the student information into two tables as an example:

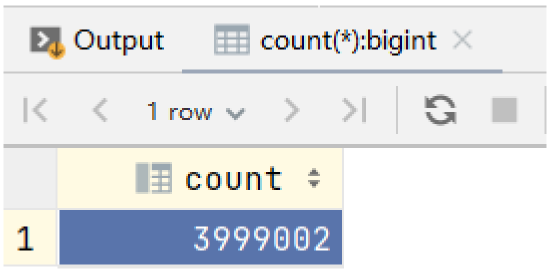
```
    static void SendToDataBase(Student student) throws IOException {
        if (connection != null) {
            String sql = "insert into Student
(name,gender,college,student_id) values (?,?,?,?)  ON CONFLICT DO NOTHING";
            String sql2 = "insert into CourseDone (student_id,course_id)
values (?,?)  ON CONFLICT DO NOTHING";
            try {
                PreparedStatement preparedStatement =
connection.prepareStatement(sql);
                preparedStatement.setString(1, student.name);
                preparedStatement.setString(2, student.gender);
                preparedStatement.setString(3, student.college);
                preparedStatement.setString(4, student.student_id);

                preparedStatement.execute();

                for (String course : student.courses_done) {
                    preparedStatement = connection.prepareStatement(sql2);
                    preparedStatement.setString(1, student.student_id);
                    preparedStatement.setString(2, course);
                    preparedStatement.execute();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            } finally {

            }
        }
    }
```

<2.3> Follow-up Check

At last, we use count() command as well as the error warning in datagrib itself to ensure all the data are successfully imported into our table.

```
select count(*)
from student
```

Result:



```
Output    count(*):bigint
1 row
  count
1 3999002
```

Which matches the number of lines of csv.

<2.4> Import perfomace

By using jdbc,

Import Student we use Run `1071337 ms` about 17.8min.

Import Student's learned lectures we use `1764935 ms` about 29.4min.

But when we use `copy` it just takes `6s` to import student from file csv.

Task 3 Compare Database And File

<3.1> Data and environment statement

All the data are remained. We never change any origin data for convenience. All the disposals are accomplished through the program.

All of our team-members' operating system is windows-10. We chose Java and Python to organize the data.

<3.2> Efficiency comparison about query between DBMS and JAVA

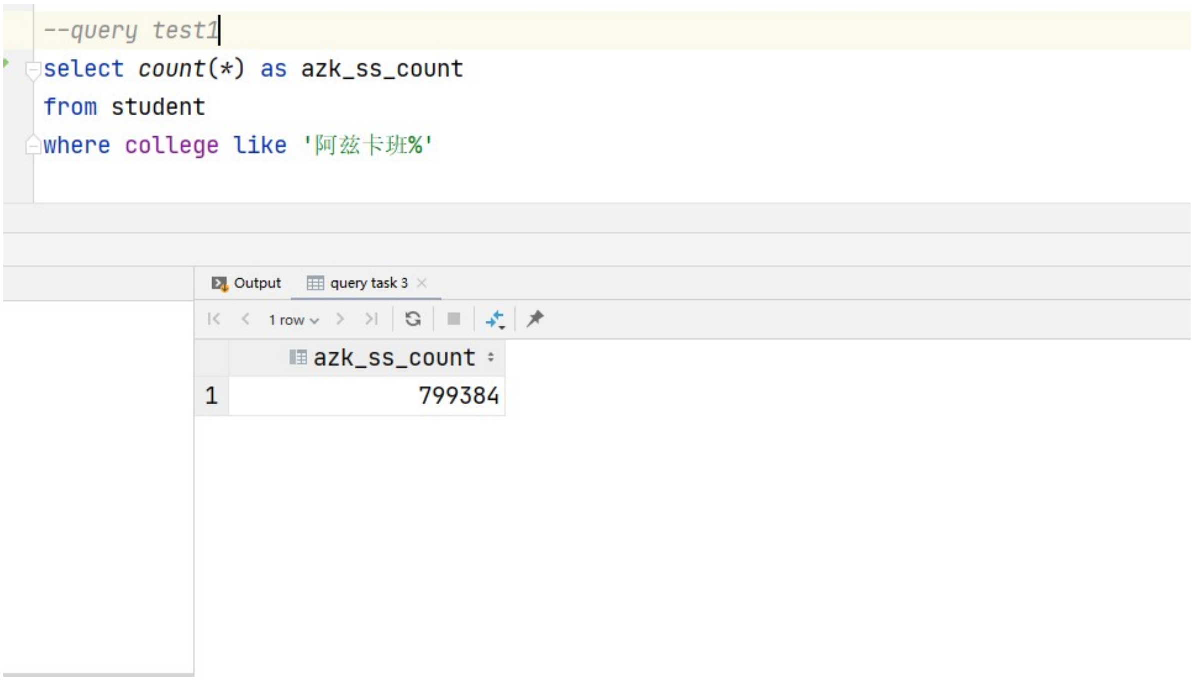
This is the comparison diagram we make between DBMS and JAVA for some query command.

According to the result of these four comparison, we found that It's much more efficient to use DBMS when we do command like count and fuzzy match for DBMS can do much more better than simply iteration and match in JAVA. And for other query command like select, DBMS also wins the JAVA , but the superiority is not obivious.

comparison-1:

DBMS:

```sql
--query test1
select count(*) as azk_ss_count
from student
where college like '阿兹卡班%';
```

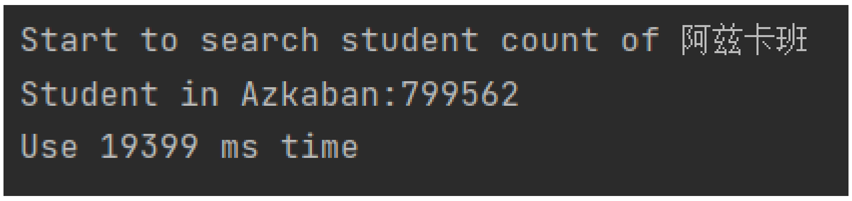


```sql
--query test1
select count(*) as azk_ss_count
from student
where college like '阿兹卡班%'
```

```
  azk_ss_count
1       799384
```

JAVA:

```java
static void q1() throws IOException {
        System.out.println("Start to search student count of 阿兹卡班");
        long startTime=System.currentTimeMillis();
        JwxtParser.parseStudent();
        long cnt=0;
        for (Student s:JwxtParser.students){
            if(s.college.equals("阿兹卡班(Azkaban)"))
                cnt++;
        }
        System.out.println("Student in Azkaban:"+cnt);
        System.out.println(String.format("Use %d ms 
time",System.currentTimeMillis()-startTime));
    }
```

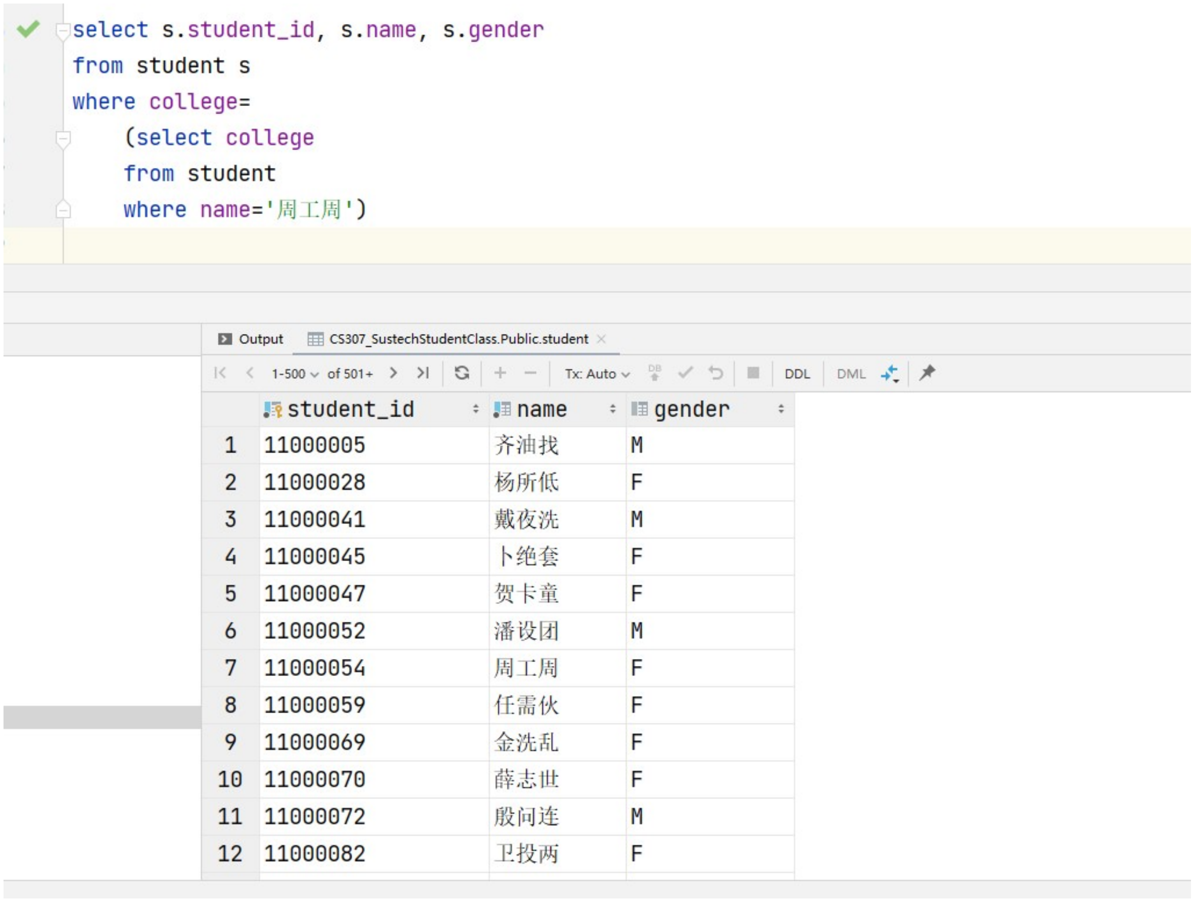


```
Start to search student count of 阿兹卡班
Student in Azkaban:799562
Use 19399 ms time
```

comparison-2:

DBMS:

```sql
--query test 2
select s.student_id, s.name, s.gender, s.college
from student s
where college =
      (select college
       from student
       where name = '周工周');
```



```sql
select s.student_id, s.name, s.gender
from student s
where college=
    (select college
    from student
    where name='周工周')
```

	student_id	name	gender
1	11000005	齐油找	M
2	11000028	杨所低	F
3	11000041	戴夜洗	M
4	11000045	卜绝套	F
5	11000047	贺卡童	F
6	11000052	潘设团	M
7	11000054	周工周	F
8	11000059	任需伙	F
9	11000069	金洗乱	F
10	11000070	薛志世	F
11	11000072	殷问连	M
12	11000082	卫投两	F

JAVA:

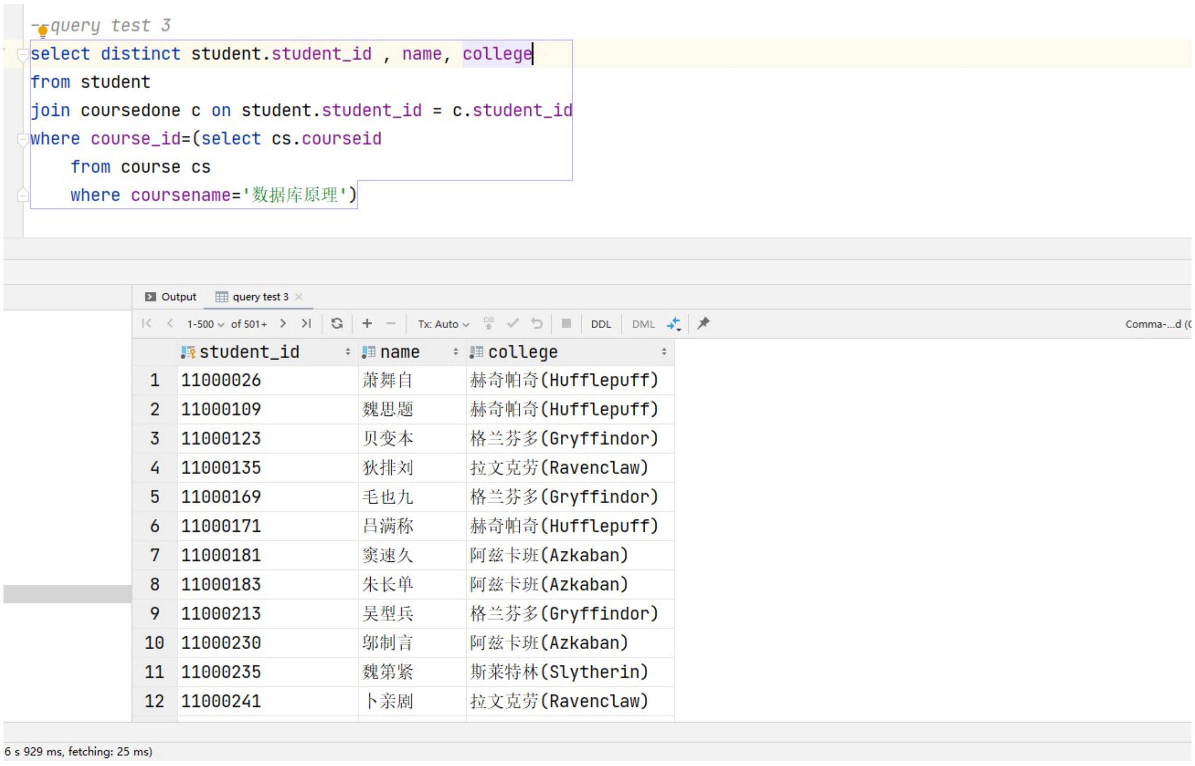
```java
 static void q2() throws IOException {
        System.out.println("Start to search the students with the same college of ZhouGongZhou");
        long startTime=System.currentTimeMillis();
        JwxtParser.parseStudent();
        String name="周工周";
        ArrayList<Student>[] students=new ArrayList[5];
        for (int i = 0; i < 5; i++) {
            students[i]=new ArrayList<>();
        }
        int index=-1;
        for (Student s:JwxtParser.students){
            if(s.name.equals(name)){
                switch (s.college){
                    case "阿兹卡班(Azkaban)":
                        index=0;
                        break;
                    case "斯莱特林(Slytherin)":
                        index=1;
                        break;
                    case"拉文克劳(Ravenclaw)":
                        index=2;
                        break;
                    case"格兰芬多(Gryffindor)":
                        index=3;
                        break;
                    case"赫奇帕奇(Hufflepuff)":
                        index=4;
                        break;
                }
            }
            switch (s.college){
                case "阿兹卡班(Azkaban)":
                    students[0].add(s);
                    break;
                case "斯莱特林(Slytherin)":
                    students[1].add(s);
                    break;
                case"拉文克劳(Ravenclaw)":
                    students[2].add(s);
                    break;
                case"格兰芬多(Gryffindor)":
                    students[3].add(s);
                    break;
                case"赫奇帕奇(Hufflepuff)":
                    students[4].add(s);
                    break;
            }
        }
        for (Student s:students[index]){
            System.out.println(s.toString());
        }
        System.out.println(String.format("Use %d ms time",System.currentTimeMillis()-startTime));
    }

```

```
Student{name='沈肯适', gender='F', college='拉文克劳(Ravenclaw)', student_id='14999888'}
Student{name='熊释酒', gender='F', college='拉文克劳(Ravenclaw)', student_id='14999889'}
Student{name='贺短头', gender='F', college='拉文克劳(Ravenclaw)', student_id='14999890'}
Student{name='彭交西', gender='F', college='拉文克劳(Ravenclaw)', student_id='14999893'}
Student{name='汪烟续', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999896'}
Student{name='章河罪', gender='F', college='拉文克劳(Ravenclaw)', student_id='14999897'}
Student{name='魏抱反', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999909'}
Student{name='汤型民', gender='F', college='拉文克劳(Ravenclaw)', student_id='14999913'}
Student{name='祁农身', gender='F', college='拉文克劳(Ravenclaw)', student_id='14999916'}
Student{name='俞治顿', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999918'}
Student{name='方父准', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999926'}
Student{name='余息纸', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999927'}
Student{name='殷得校', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999929'}
Student{name='罗自听', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999931'}
Student{name='邬活读', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999937'}
Student{name='明土曾', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999942'}
Student{name='黄题求', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999945'}
Student{name='孟白石', gender='F', college='拉文克劳(Ravenclaw)', student_id='14999947'}
Student{name='喻味班', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999950'}
Student{name='马禁良', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999956'}
Student{name='庞类变', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999967'}
Student{name='庞森候', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999975'}
Student{name='唐达屋', gender='F', college='拉文克劳(Ravenclaw)', student_id='14999981'}
Use 12013 ms time
```

comparison-3:

DBMS:

```sql
--query test 3
select distinct student.student_id, name, college
from student
        join coursedone c on student.student_id = c.student_id
where course_id = (select cs.courseid
                   from course cs
                   where coursename = '数据库原理');
```



```sql
--query test 3
select distinct student.student_id , name, college
from student
join coursedone c on student.student_id = c.student_id
where course_id=(select cs.courseid
    from course cs
    where coursename='数据库原理')
```

	student_id	name	college
1	11000026	萧舞自	赫奇帕奇(Hufflepuff)
2	11000109	魏思题	赫奇帕奇(Hufflepuff)
3	11000123	贝变本	格兰芬多(Gryffindor)
4	11000135	狄排刘	拉文克劳(Ravenclaw)
5	11000169	毛也九	格兰芬多(Gryffindor)
6	11000171	吕满称	赫奇帕奇(Hufflepuff)
7	11000181	窦速久	阿兹卡班(Azkaban)
8	11000183	朱长单	阿兹卡班(Azkaban)
9	11000213	吴型兵	格兰芬多(Gryffindor)
10	11000230	邬制言	阿兹卡班(Azkaban)
11	11000235	魏第紧	斯莱特林(Slytherin)
12	11000241	卜亲剧	拉文克劳(Ravenclaw)

6 s 929 ms, fetching: 25 ms)

JAVA:

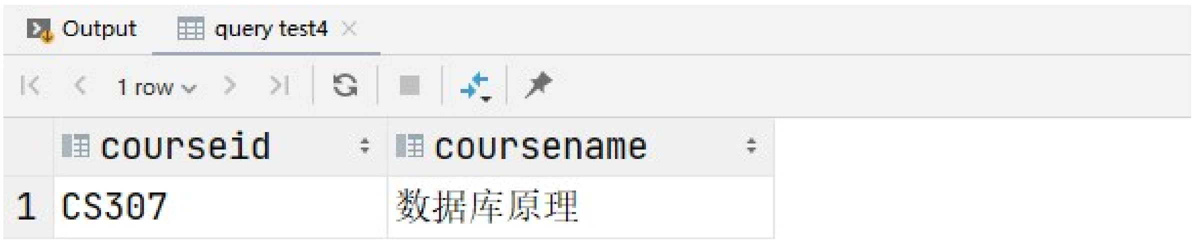
```java
static void q3() throws IOException {
        System.out.println("Start to search the students learned database");
        long startTime=System.currentTimeMillis();
        JwxtParser.parseCourseJson();
        JwxtParser.parseCourseRAW();
        JwxtParser.parseStudent();
        String name="数据库原理";
        String course_id="";
        for (Course c:JwxtParser.courseHashMap.values()) {
            if(c.course_name.equals(name)){
                course_id=c.course_id;
                break;
            }
        }

        for (Student s:JwxtParser.students){
            for (String cc:s.courses_done){
                if(cc.equals(course_id)){
                    System.out.println(s.toString());
                }
            }
        }

        System.out.println(String.format("Use %d ms time",System.currentTimeMillis()-startTime));
    }

```

```
Student{name='邵梦守', gender='F', college='格兰芬多(Gryffindor)', student_id='14999876'}
Student{name='毛没著', gender='M', college='斯莱特林(Slytherin)', student_id='14999882'}
Student{name='钱朝民', gender='F', college='赫奇帕奇(Hufflepuff)', student_id='14999883'}
Student{name='贺短头', gender='F', college='拉文克劳(Ravenclaw)', student_id='14999890'}
Student{name='潘们现', gender='M', college='格兰芬多(Gryffindor)', student_id='14999922'}
Student{name='余息纸', gender='M', college='拉文克劳(Ravenclaw)', student_id='14999927'}
Student{name='穆弄课', gender='F', college='格兰芬多(Gryffindor)', student_id='14999934'}
Student{name='鲍舞窦', gender='F', college='斯莱特林(Slytherin)', student_id='14999971'}
Use 8104 ms time
```

comparison-4:

DBMS:

```sql
--query test4
with csc as
         (select courseid, count(*) cnt
          from class c
                   join cc_linker cl on c.classid = cl.classid
          where courseid like 'CS%'
          group by courseid),

     greater as
         (select courseid
          from csc
          where cnt > 3),

     cntp as
         (select courseid, count(*) cnt2
          from greater
                   join coursedone cd on greater.courseid = cd.course_id
          group by courseid),

     mx as
         (select max(cnt2) m
          from cntp
         )

select cntp.courseid, coursename
from cntp
         join course on cntp.courseid = course.courseid
where cnt2 = (select m from mx);
```

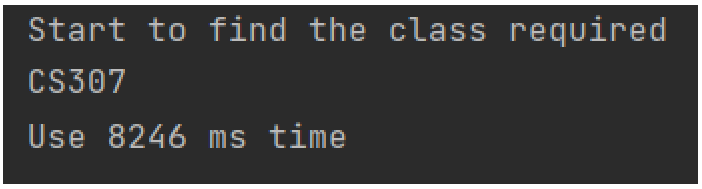

Output   query test4

1 row

	courseid	coursename
1	CS307	数据库原理

JAVA:

```java
static void q4() throws IOException{
        System.out.println("Start to find the class required");
        long startTime=System.currentTimeMillis();
        JwxtParser.parseCourseJson();
        JwxtParser.parseCourseRAW();
        JwxtParser.parseStudent();
        ArrayList<String> courses=new ArrayList<>();
        for (Course c:JwxtParser.courseHashMap.values()) {
            if(c.classes.size()>=3 && c.course_departure.equals("计算机科学与工程系"))
                courses.add(c.course_id);
        }
        HashMap<String,Long> course_count = new HashMap<>();
        Long max_cnt=0l;
        String max_str="";
        for (Student s:JwxtParser.students) {
            for (String c:courses) {

                Long tmp=course_count.get(c);
                if(tmp==null)tmp=0l;
                if(s.courses_done.contains(c))
                    course_count.put(c,tmp+1);
                if(tmp+1>max_cnt){
                    max_cnt=tmp+1;
                    max_str=c;
                }
            }
        }
        System.out.println(max_str);
        System.out.println(String.format("Use %d ms time",System.currentTimeMillis()-startTime));
    }

    static void inJson() throws IOException{
        System.out.println("Input json");
        long startTime=System.currentTimeMillis();
        JwxtParser.parseCourseJson();
        JwxtParser.parseCourseRAW();
        System.out.println(String.format("Use %d ms time",System.currentTimeMillis()-startTime));
        System.out.println("Input csv");
        startTime=System.currentTimeMillis();
        JwxtParser.parseStudent();
        System.out.println(String.format("Use %d ms time",System.currentTimeMillis()-startTime));

    }
```



```
Start to find the class required
CS307
Use 8246 ms time
```

<3.3> High concurrency and transaction management

A project starts with the goal of implementing basic functionality, and as versions and features iterate, big data and high concurrency become necessary to be considered!

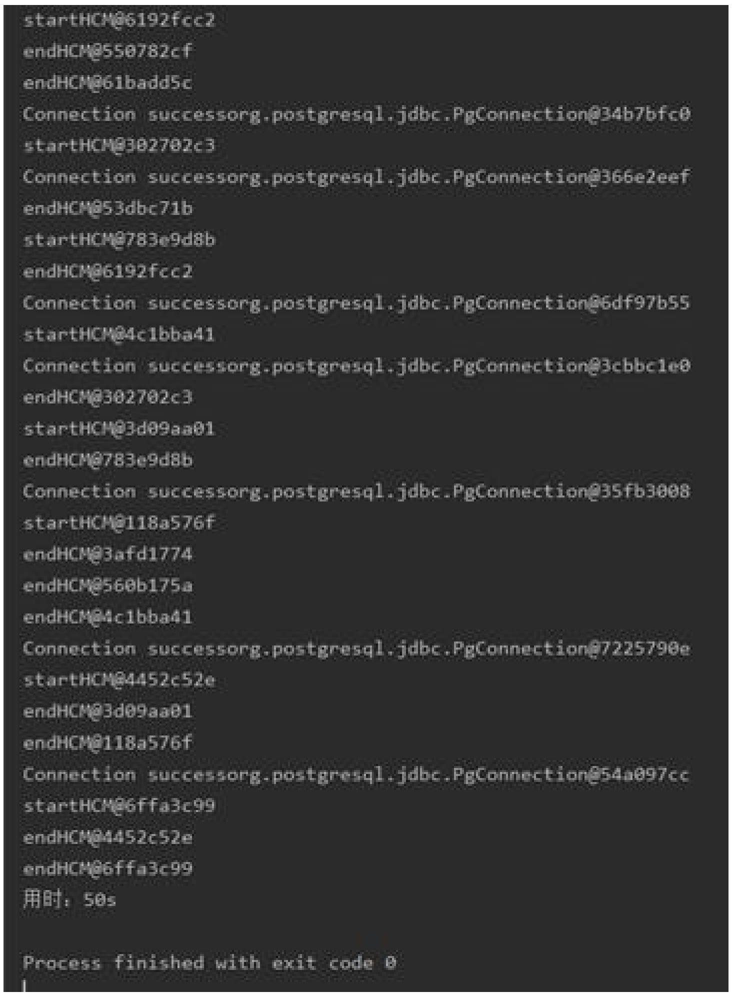
The essence is very simple, one is slow, one is to wait.

This two reasons are interrelated, slow causes waiting, waiting results in slow.

Our group wrote the following .java file to test the high concurrency problem:(only the core part is shown below)

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class HCM implements Runnable {
    public static void main(String[] args) throws Exception {
        start=0;
        for(long i=0;i<100;i++) {
            HCM hcm=new HCM("jdbc:postgresql://10.17.118.214:5432/CS307_SustechStudentClass","byll","123456");
            Thread thread=new Thread(hcm);
            thread.start();
            start++;
        }
    }
    static long end;
    static long start;
    static long sTime;
    static long eTime;
    HCM(String url, String user, String password) {
        try {
            connection = DriverManager.getConnection(url, user, password);
            System.out.println("Connection success" + connection);
            statement = connection.createStatement();
            statement.execute("set search_path = \"Public\"");
        } catch (Exception var) {
            System.out.println("Connection failed");
            var.printStackTrace();
        }
    }

    Connection connection;
    Statement statement;
    ResultSet rst = null;
    @Override
    public void run() {
        if(start==0){
            sTime=System.nanoTime();
        }
        System.out.println("start"+this);
        if (connection != null) {
            String sql = "select c.student_id from \"CS307_SustechStudentClass\".\"Public\".coursedone c\n" +
                    "where c.course_id='CH316';";
            try {
                statement = connection.createStatement();
                rst = statement.executeQuery(sql);
                rst.close();
                statement.close();
                connection.close();
            } catch (SQLException var) {
                var.printStackTrace();
            } finally {
                end++;
                System.out.println("end"+this);
                if(end>=100){
                    eTime = System.nanoTime();
                    System.out.println("用时：" + (eTime - sTime)/1000000000+"s");
                }
            }
        }
    }
}
```

In this file, we simulate 100 users using the select function at almost the same time and record the total time. This help test the efficiency of our database. The test time is:

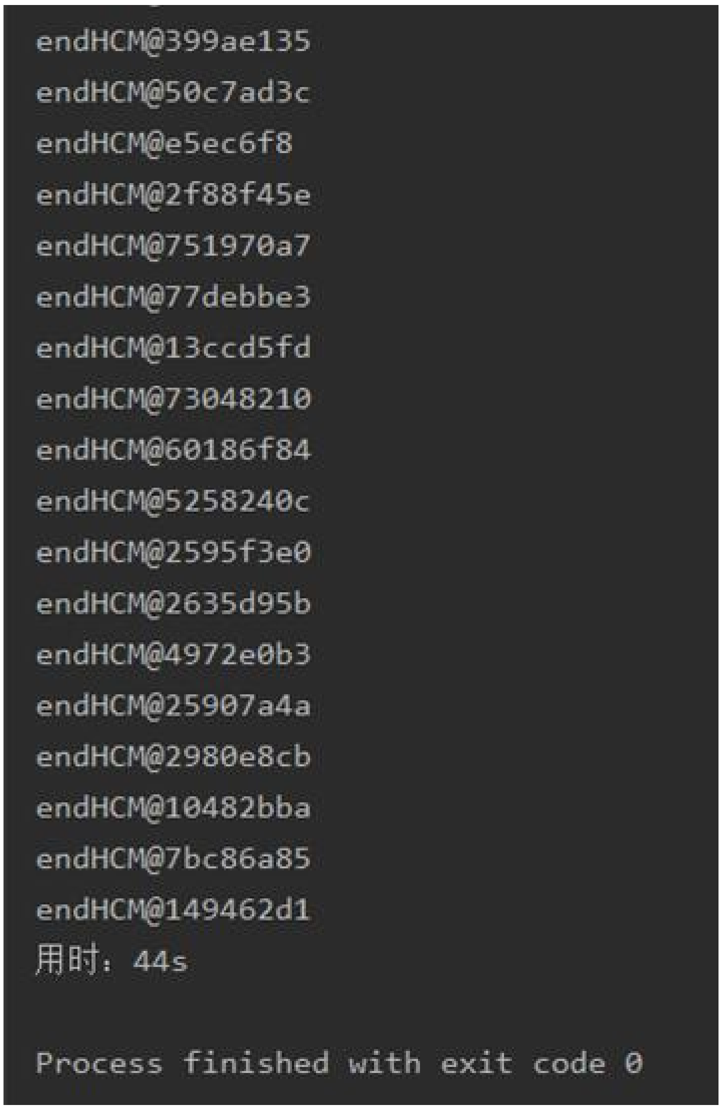


```
startHCM@6192fcc2
endHCM@550782cf
endHCM@61badd5c
Connection successorg.postgresql.jdbc.PgConnection@34b7bfc0
startHCM@302702c3
Connection successorg.postgresql.jdbc.PgConnection@366e2eef
endHCM@53dbc71b
startHCM@783e9d8b
endHCM@6192fcc2
Connection successorg.postgresql.jdbc.PgConnection@6df97b55
startHCM@4c1bba41
Connection successorg.postgresql.jdbc.PgConnection@3cbbc1e0
endHCM@302702c3
startHCM@3d09aa01
endHCM@783e9d8b
Connection successorg.postgresql.jdbc.PgConnection@35fb3008
startHCM@118a576f
endHCM@3afd1774
endHCM@560b175a
endHCM@4c1bba41
Connection successorg.postgresql.jdbc.PgConnection@7225790e
startHCM@4452c52e
endHCM@3d09aa01
endHCM@118a576f
Connection successorg.postgresql.jdbc.PgConnection@54a097cc
startHCM@6ffa3c99
endHCM@4452c52e
endHCM@6ffa3c99
用时：50s

Process finished with exit code 0
```

We can see from the result that there still remains a lot of room for improvement. Our group member still learned a lot through analyzing this problem and did a lot of research on it. We list some solution for further study and optimizing:

1. Create unique key.
2. The infrequently queried ones are put in a table, and the frequently queried ones are put in another table.
3. Do not go through the full table query, this will be slow.
4. use a UUID or a self-incrementing sequence by date.
5. ....

Since the ddl is close, we don't have much time to practice them one by one . So, we chose the first solution only: created the unique key to improve the performance, The second time result is shown below:



```
endHCM@399ae135
endHCM@50c7ad3c
endHCM@e5ec6f8
endHCM@2f88f45e
endHCM@751970a7
endHCM@77debbe3
endHCM@13ccd5fd
endHCM@73048210
endHCM@60186f84
endHCM@5258240c
endHCM@2595f3e0
endHCM@2635d95b
endHCM@4972e0b3
endHCM@25907a4a
endHCM@2980e8cb
endHCM@10482bba
endHCM@7bc86a85
endHCM@149462d1
用时：44s

Process finished with exit code 0
```

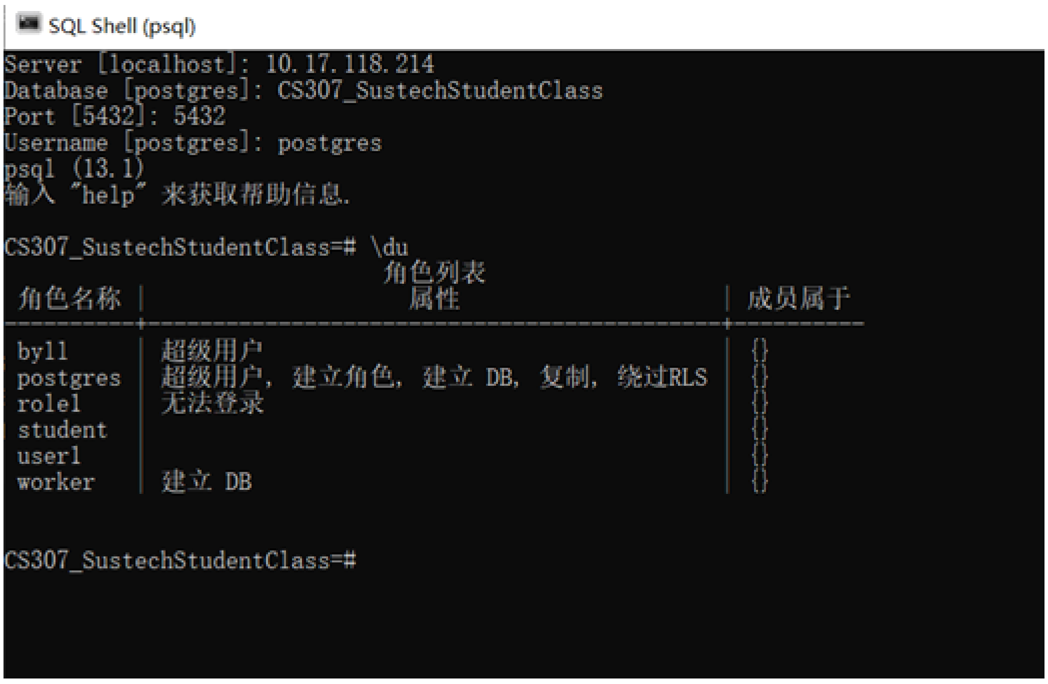
As we can see, the performance is improved. In the future, we will do more modification to manage high concurrency problems.

<3.4> User privileges management

User is a very significant key to the dataset with DBMS. For example, we want user worker just has the access of select privileges, that is he/she cannot change the dataset in any attempt (inserting, drop, alter attribute, etc.). Also, there should exist superusers who can do anything to the dataset without limitation. In the aspect, we call it user privileges.
In the DBMS, we can easily create ,give him some privilege and drop a user, the codes is displayed below:

```sql
CREATE USER worker PASSWORD '123456' ;  --创建用户时授权可创建数据库,并赋密码
ALTER USER worker CREATEDB; --赋权worker可创建数据库
GRANT CONNECT ON DATABASE "CS307_SustechStudentClass" TO worker; --将数据库的连接权限赋予给worker用户
```

And then we check out the result with SQL shell:

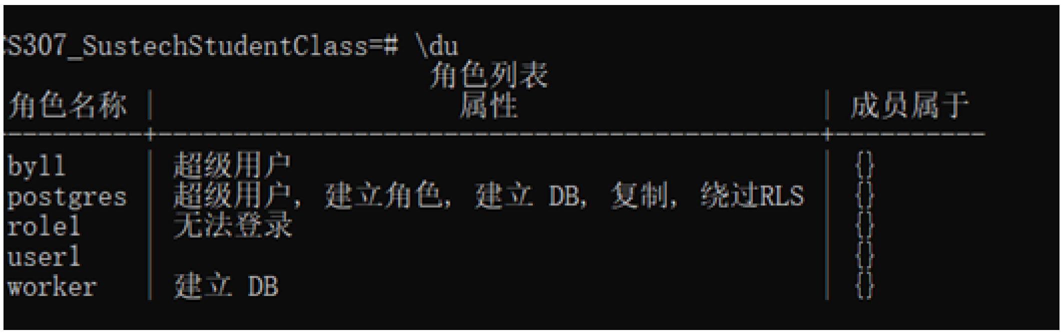


```
SQL Shell (psql)
Server [localhost]: 10.17.118.214
Database [postgres]: CS307_SustechStudentClass
Port [5432]: 5432
Username [postgres]: postgres
psql (13.1)
输入 "help" 来获取帮助信息.

CS307_SustechStudentClass=# \du
                        角色列表
 角色名称 |                      属性                      | 成员属于
----------+------------------------------------------------+----------
 byll     | 超级用户                                       | {}
 postgres | 超级用户, 建立角色, 建立 DB, 复制, 绕过RLS     | {}
 role1    | 无法登录                                       | {}
 student  |                                                | {}
 user1    |                                                | {}
 worker   | 建立 DB                                        | {}


CS307_SustechStudentClass=#
```

Then we drop user student(which was created before to test) and grant all privileges to worker:

```sql
drop user student;
GRANT ALL PRIVILEGES ON DATABASE "CS307_SustechStudentClass" TO worker;
```

Then we turn to SQL Shell to check the updated result:



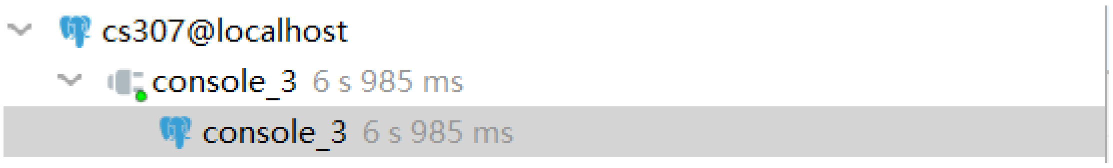
```
S307_SustechStudentClass=# \du
                        角色列表
角色名称 |                      属性                      | 成员属于
---------+------------------------------------------------+----------
byll     | 超级用户                                       | {}
postgres | 超级用户, 建立角色, 建立 DB, 复制, 绕过RLS     | {}
role1    | 无法登录                                       | {}
user1    |                                                | {}
worker   | 建立 DB                                        | {}
```

We can say that it's very simple and convenient for user privilege operations in DBMS.

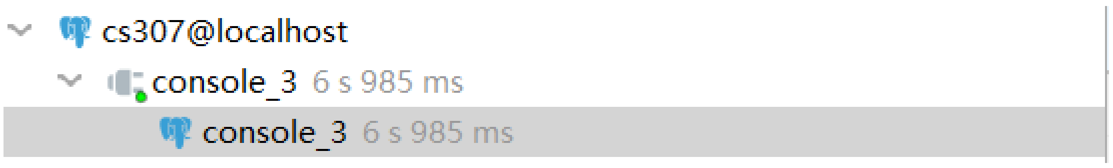
<3.5> Database index and file IO

Search without index

```sql
select *
from student
where name='喻古春'
```



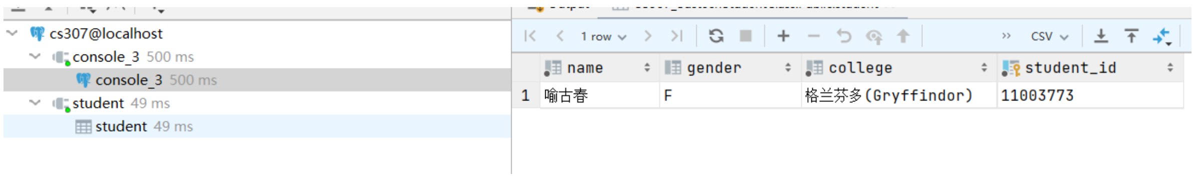
```
cs307@localhost
  console_3 6 s 985 ms
    console_3 6 s 985 ms
```



```
cs307@localhost
  console_3 6 s 985 ms
    console_3 6 s 985 ms
```

Then we add

```
create index student_name on student(student_id)
```



```
cs307@localhost
  console_3 500 ms
    console_3 500 ms
  student 49 ms
    student 49 ms
```

	name	gender	college	student_id
1	喻古春	F	格兰芬多(Gryffindor)	11003773

And the second search, the speed will be faster

<3.6> Performance comparison

query test

> 1. Count the number of all students that belong to Azkaban college.

```sql
select count(*) as azk_ss_count
from student
where college like '阿兹卡班%';
```

Using java

```java
static void q1() throws IOException {
        System.out.println("Start to search student count of 阿兹卡班");
        long startTime=System.currentTimeMillis();
        JwxtParser.parseStudent();
        long cnt=0;
        for (Student s:JwxtParser.students){
            if(s.college.equals("阿兹卡班(Azkaban)"))
                cnt++;
        }
        System.out.println("Student in Azkaban:"+cnt);
        System.out.println(String.format("Use %d ms time",System.currentTimeMillis()-startTime));
    }
```

> 2. Output the sid, name, and gender of students who are in the same college as "周工周"

```sql
select s.student_id, s.name, s.gender, s.college
from student s
where college =
      (select college
       from student
       where name = '周工周');
```

Using java

```java
static void q2() throws IOException {
        System.out.println("Start to search the students with the same college of ZhouGongZhou");
        long startTime=System.currentTimeMillis();
        JwxtParser.parseStudent();
        String name="周工周";
        ArrayList<Student>[] students=new ArrayList[5];
        for (int i = 0; i < 5; i++) {
            students[i]=new ArrayList<>();
        }
        int index=-1;
        for (Student s:JwxtParser.students){
            if(s.name.equals(name)){
                switch (s.college){
                    case "阿兹卡班(Azkaban)":
                        index=0;
                        break;
                    case "斯莱特林(Slytherin)":
                        index=1;
                        break;
                    case"拉文克劳(Ravenclaw)":
                        index=2;
                        break;
                    case"格兰芬多(Gryffindor)":
                        index=3;
                        break;
                    case"赫奇帕奇(Hufflepuff)":
                        index=4;
                        break;
                }
            }
            switch (s.college){
                case "阿兹卡班(Azkaban)":
                    students[0].add(s);
                    break;
                case "斯莱特林(Slytherin)":
                    students[1].add(s);
                    break;
                case"拉文克劳(Ravenclaw)":
                    students[2].add(s);
                    break;
                case"格兰芬多(Gryffindor)":
                    students[3].add(s);
                    break;
                case"赫奇帕奇(Hufflepuff)":
                    students[4].add(s);
                    break;
            }
        }
        for (Student s:students[index]){
            System.out.println(s.toString());
        }
        System.out.println(String.format("Use %d ms 
time",System.currentTimeMillis()-startTime));
    }
```

> 3. The sid, name and college of students who have taken the course named "数据库原理"

```sql
select distinct student.student_id, name, college
from student
         join coursedone c on student.student_id = c.student_id
where course_id = (select cs.courseid
                   from course cs
                   where coursename = '数据库原理');
```

Using java

```java
static void q3() throws IOException {
        System.out.println("Start to search the students learned database");
        long startTime=System.currentTimeMillis();
        JwxtParser.parseCourseJson();
        JwxtParser.parseCourseRAW();
        JwxtParser.parseStudent();
        String name="数据库原理";
        String course_id="";
        for (Course c:JwxtParser.courseHashMap.values()) {
            if(c.course_name.equals(name)){
                course_id=c.course_id;
                break;
            }
        }

        for (Student s:JwxtParser.students){
            for (String cc:s.courses_done){
                if(cc.equals(course_id)){
                    System.out.println(s.toString());
                }
            }
        }

        System.out.println(String.format("Use %d ms
time",System.currentTimeMillis()-startTime));
    }
```

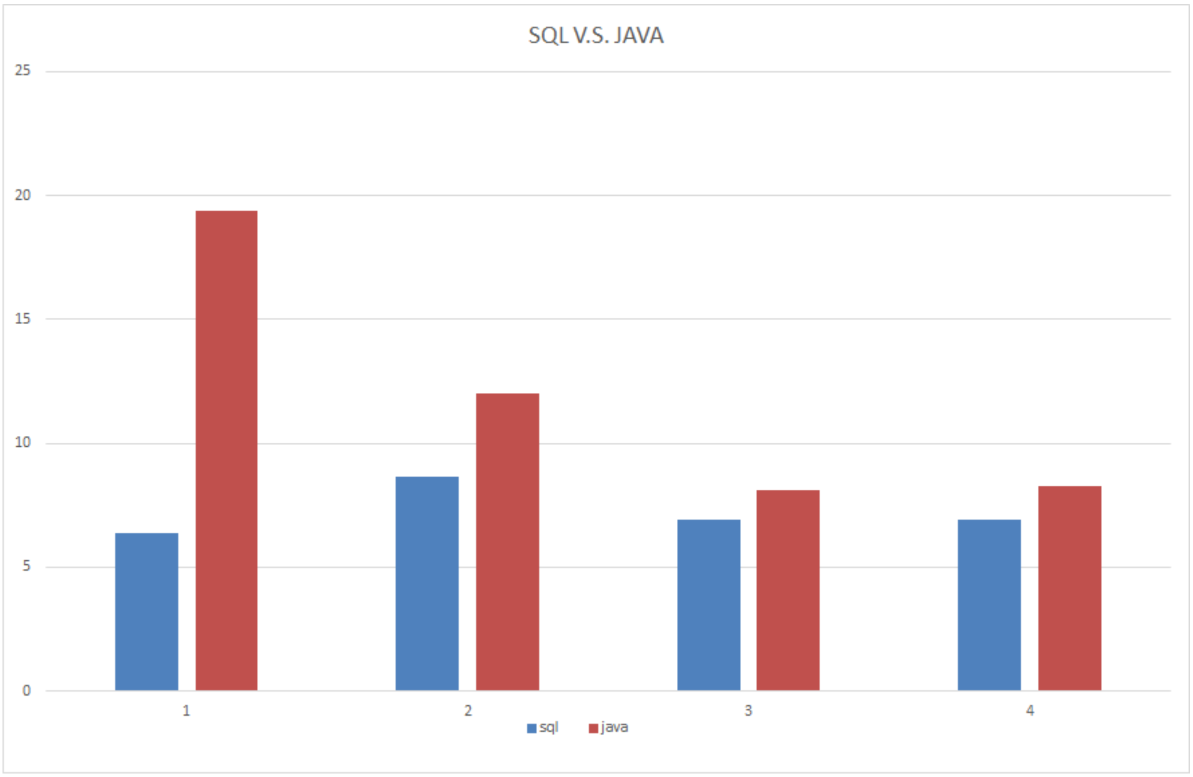
> 4. The course_id of such a course that has most number of students who have taken it among the courses conducted by CS department that have more than 3 different classes.

```sql
with csc as
         (select courseid, count(*) cnt
          from class c
                   join cc_linker cl on c.classid = cl.classid
          where courseid like 'CS%'
          group by courseid),

     greater as
         (select courseid
          from csc
          where cnt > 3),

     cntp as
         (select courseid, count(*) cnt2
          from greater
                   join coursedone cd on greater.courseid = cd.course_id
          group by courseid),

     mx as
         (select max(cnt2) m
          from cntp
         )
select cntp.courseid, coursename
```

```sql
from cntp
         join course on cntp.courseid = course.courseid
where cnt2 = (select m from mx);
```

Using java

```java
static void q4() throws IOException{
        System.out.println("Start to find the class required");
        long startTime=System.currentTimeMillis();
        JwxtParser.parseCourseJson();
        JwxtParser.parseCourseRAW();
        JwxtParser.parseStudent();
        ArrayList<String> courses=new ArrayList<>();
        for (Course c:JwxtParser.courseHashMap.values()) {
            if(c.classes.size()>=3 && c.course_departure.equals("计算机科学与工程系"))
                courses.add(c.course_id);
        }
        HashMap<String,Long> course_count = new HashMap<>();
        Long max_cnt=0l;
        String max_str="";
        for (Student s:JwxtParser.students) {
            for (String c:courses) {

                Long tmp=course_count.get(c);
                if(tmp==null)tmp=0l;
                if(s.courses_done.contains(c))
                    course_count.put(c,tmp+1);
                if(tmp+1>max_cnt){
                    max_cnt=tmp+1;
                    max_str=c;
                }
            }
        }
        System.out.println(max_str);
        System.out.println(String.format("Use %d ms time",System.currentTimeMillis()-startTime));
    }
```



<3.7> Accessing database by web

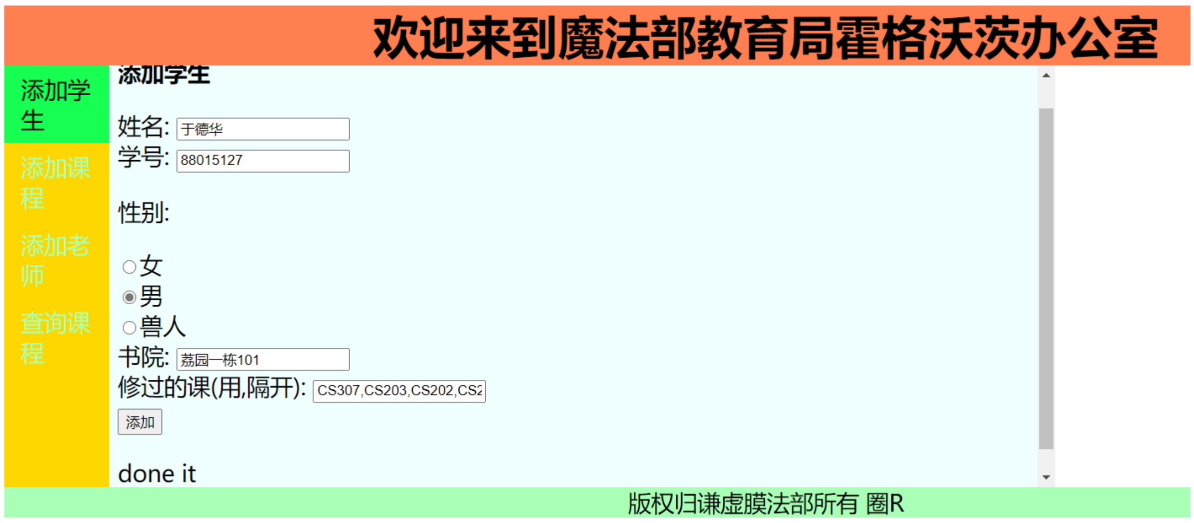
It's may be hard for us to manipulate database directly to access some data. So we create a web application for easier show our powerful database.

Environment

We use python as server and web as application for cross platform access. For python we use package **flask** but for web we make it for ourselves(so it may be ugly).

Accessing as a admin

When we login as admin, we can see a panel and we put data like this:



欢迎来到魔法部教育局霍格沃茨办公室

添加学生

添加课程

添加老师

查询课程

添加学生

姓名: 于德华

学号: 88015127

性别:

○女
●男
○兽人

书院: 荔园一栋101

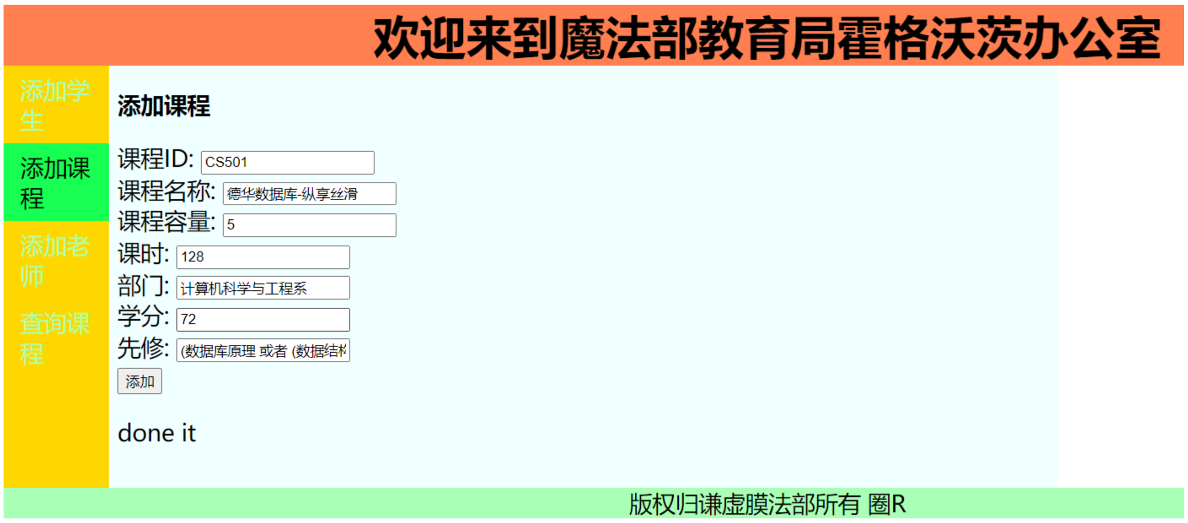
修过的课(用,隔开): CS307,CS203,CS202,CS2

添加

done it

版权归谦虚膜法部所有 圈R

We made the most powerful prerequisite design, so we must show it off.


欢迎来到魔法部教育局霍格沃茨办公室

添加学生
添加课程
添加老师
查询课程

添加课程

课程ID: CS501
课程名称: 德华数据库-纵享丝滑
课程容量: 5
课时: 128
部门: 计算机科学与工程系
学分: 72
先修: (数据库原理 或者 (数据结
添加

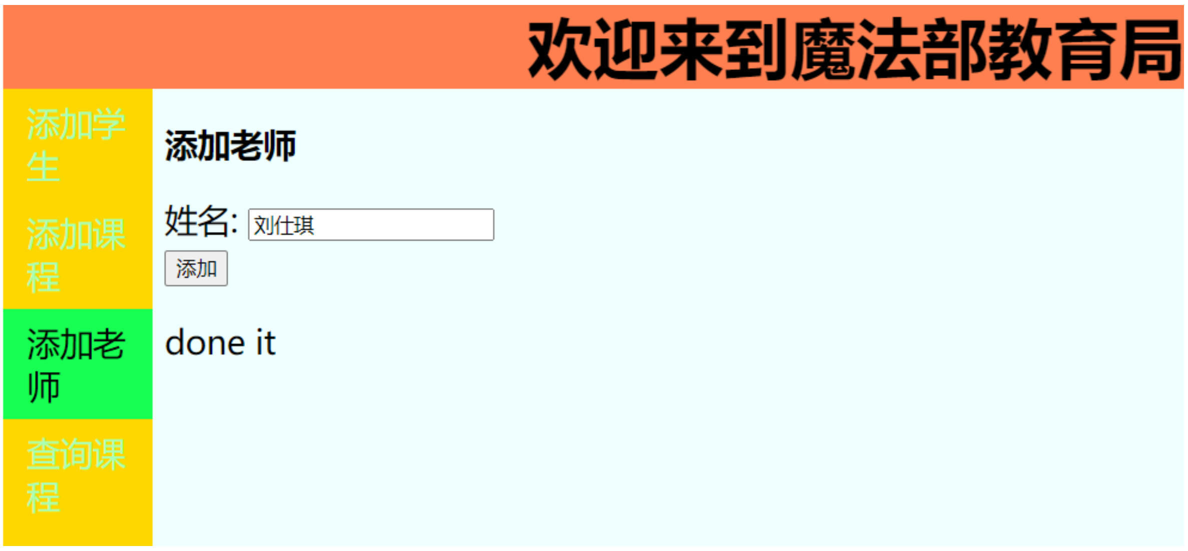
done it

版权归谦虚膜法部所有 圈R

The **prerequisite** like this:

```
(数据库原理 或者 (数据结构与算法分析 并且 (计算机组成原理 或者 数字逻辑) )或者 如何像于德华一样帅)
```

We can even add a teacher:


欢迎来到魔法部教育局

添加学生
添加课程
添加老师
查询课程

添加老师

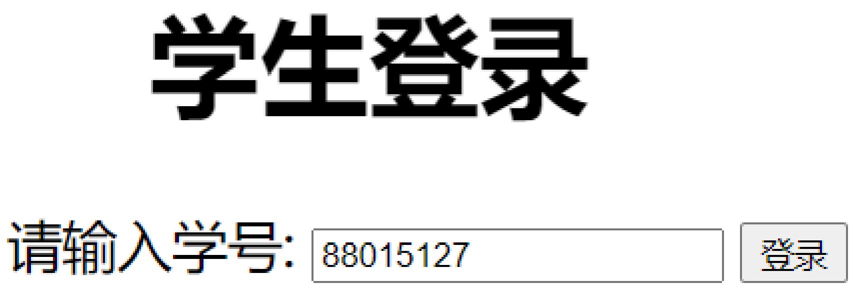
姓名: 刘仕琪
添加

done it

Accessing as a student

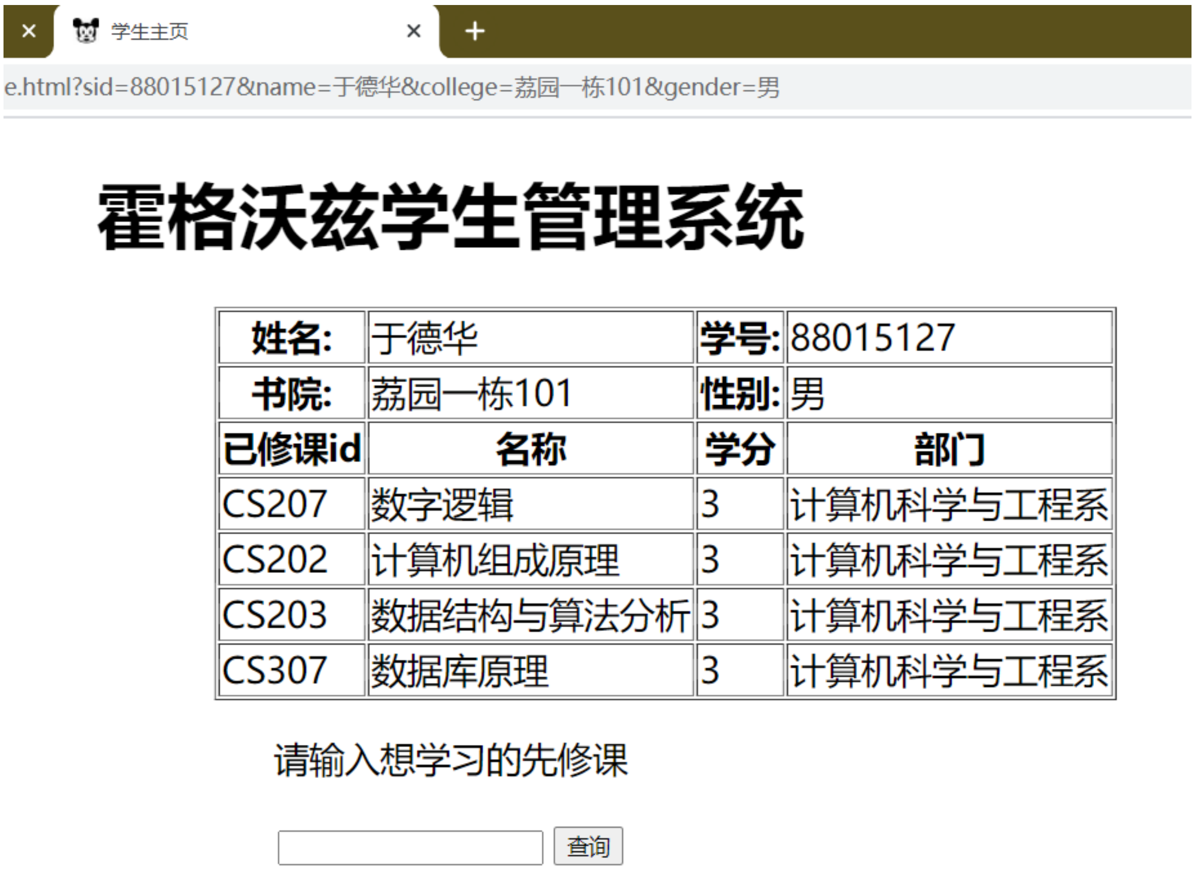
Now it's time for us to check our insert result.

Firstly we login.


学生登录

请输入学号: 88015127
登录

And we got chart like this:



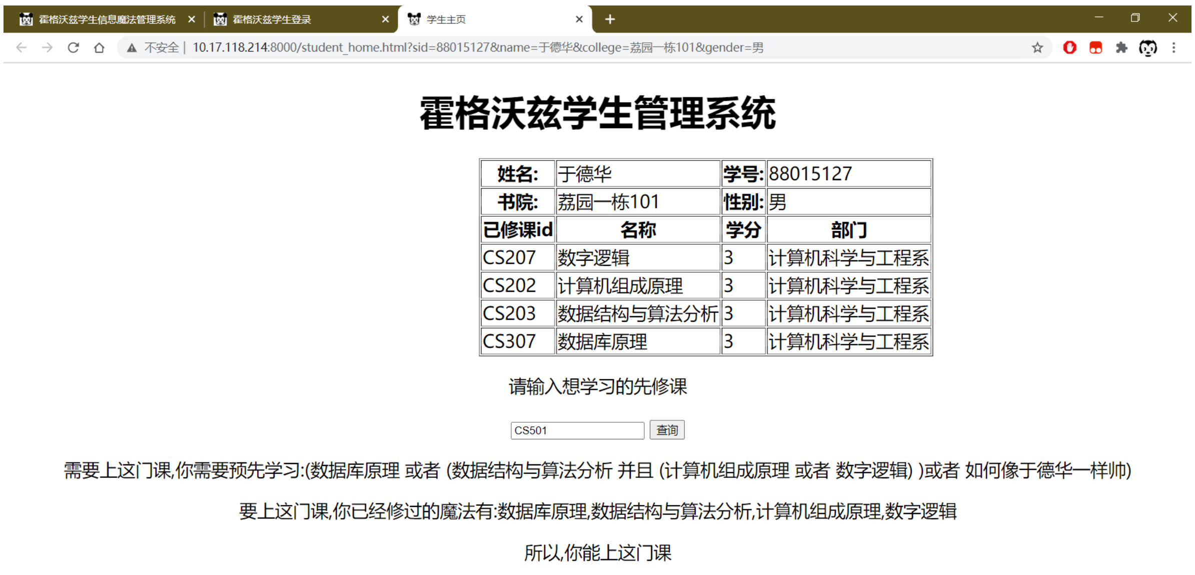
e.html?sid=88015127&name=于德华&college=荔园一栋101&gender=男

霍格沃兹学生管理系统

姓名:	于德华	学号:	88015127
书院:	荔园一栋101	性别:	男
已修课id	名称	学分	部门
CS207	数字逻辑	3	计算机科学与工程系
CS202	计算机组成原理	3	计算机科学与工程系
CS203	数据结构与算法分析	3	计算机科学与工程系
CS307	数据库原理	3	计算机科学与工程系

请输入想学习的先修课

We can check if this student qualified for some lecture:



不安全 | 10.17.118.214:8000/student_home.html?sid=88015127&name=于德华&college=荔园一栋101&gender=男

霍格沃兹学生管理系统

姓名:	于德华	学号:	88015127
书院:	荔园一栋101	性别:	男
已修课id	名称	学分	部门
CS207	数字逻辑	3	计算机科学与工程系
CS202	计算机组成原理	3	计算机科学与工程系
CS203	数据结构与算法分析	3	计算机科学与工程系
CS307	数据库原理	3	计算机科学与工程系

请输入想学习的先修课

CS501

需要上这门课,你需要预先学习:(数据库原理 或者 (数据结构与算法分析 并且 (计算机组成原理 或者 数字逻辑))或者 如何像于德华一样帅)

要上这门课,你已经修过的魔法有:数据库原理,数据结构与算法分析,计算机组成原理,数字逻辑

所以,你能上这门课

That's all for what we made to make the data more comfortable to get. We won't post the detail about how we realize our server and web(But you can check this source code `https://github.com/baiyanlali/sustech_student_class_web`). But we can share how we interact with database.

We fetch data rows and process them to make them like json, and put them back to web.

```python
def pre(cid, sid):
    db = psy.connect(database='CS307_SustechStudentClass', user='byll', password='123456', host='10.17.118.214',
                     port='5432')
    cur = db.cursor()
    cur.execute("set search_path = 'Public'")

    # get pre list and done
    cur.execute("""select p.standard_name, p.num
        from
        (select standard_name, num
        from pre_std_name
        where host_courseid='%s')p
        join (select c.standard_name
            from coursedone
            join course c
            on c.courseid=coursedone.course_id
            where coursedone.student_id='%s')q
        on p.standard_name=q.standard_name; """ % (cid, sid))
    rows = cur.fetchall()
    done = []
    pre_list = []
    for i, j in rows:
        done.append(i)
        pre_list.append(j)

    # get encode
    cur.execute("""select encode_pattern, length
            from pre_encode
            where course_id='%s'""" % (cid))
    rows2 = cur.fetchall()
    encode_r = rows2[0][0]
    length_r = rows2[0][1]


    #get raw expression of pre
    cur.execute("""select prerequisite
                from course
                where courseid='%s'""" % (cid))
    rows3 = cur.fetchall()
    raw_pre=rows3[0][0]

    check=check_satisfy(encode_r,length_r, pre_list)

    if check==1 or length_r==0:
        reply=True
    else:
        reply=False
    t={'list':done,'qualified':reply, 'pres':raw_pre}
    t=json.dumps(t)
    tt='%s(%s)'%('pre_course_query',t)
    return tt
```

6. Conclusion

Through this project, we have better understood the principles and paradigms of database design. Specifically, we learnt to utilize E-R diagram to assist designing and clarify our flow. Meanwhile, we also discovered that everything is more sophisticated than aforehand considering when you zoom in to a certain extant. By implementing comparison between files and DML, we eventually agree that database is very clever.

Now, it's 2:49 AM, It's time to sleep.

お疲れ様でした。