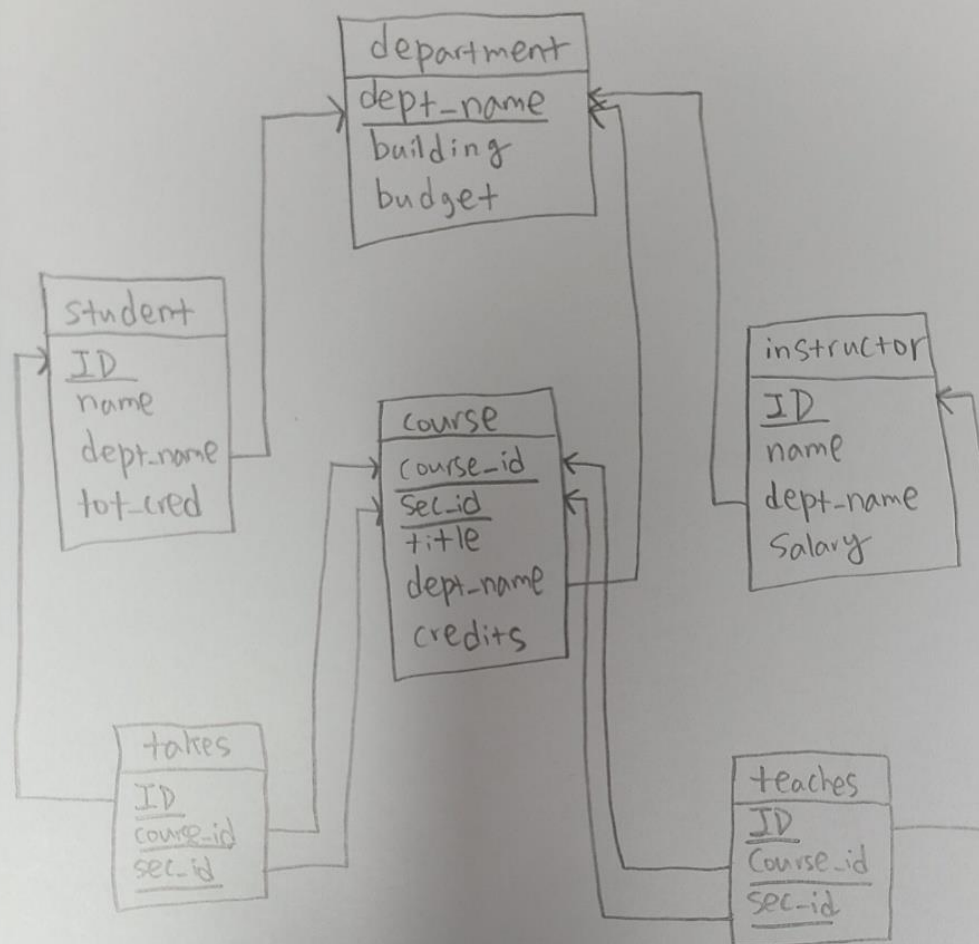


DB Application Homework

2018320147 문성준



Schema Diagram for
University-d

Schema Diagram

1. DB Application 소개

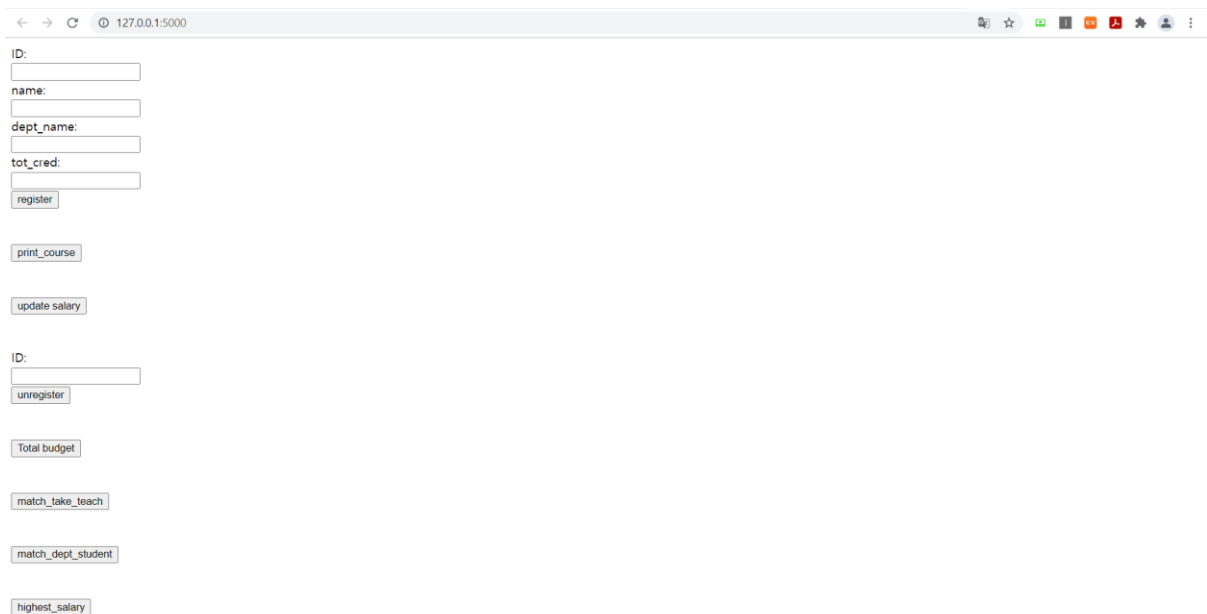
제가 만든 sql스크립트는 University Database를 응용하여 대학의 student(학생), department(학과), instructor(교수), course(과목), takes(수강에 관한 정보), teaches(가르치는 것에 관한 정보)를 담고 있습니다. 특정한 연도의 특정한 학기라고 가정하여 semester, year등의 내용은 담지 않았습니다.

또한 DB Application은 총 9개의 함수로 이루어져있고 이것은 html에 사용자가 입력한 정보를 읽어서 데이터베이스에 반영하는 것, html에서 특정한 버튼을 누르면 데이터베이스의 정보가 변경되는 것, 데이터베이스에서 정보를 추출하여 html로 출력하는 것 등으로 구성되어 있습니다. Application의 함수들이 일관된 목적으로 만들어진 것은 아니지만 데이터베이스에 저장된 대학의 정보를 바탕으로 여러 상황을 만들어 보고 출력해보는 것을 목표로 하였습니다.

2. 구현한 함수 설명

1) main함수

app.py를 실행시키면 메인함수가 먼저 실행됩니다.



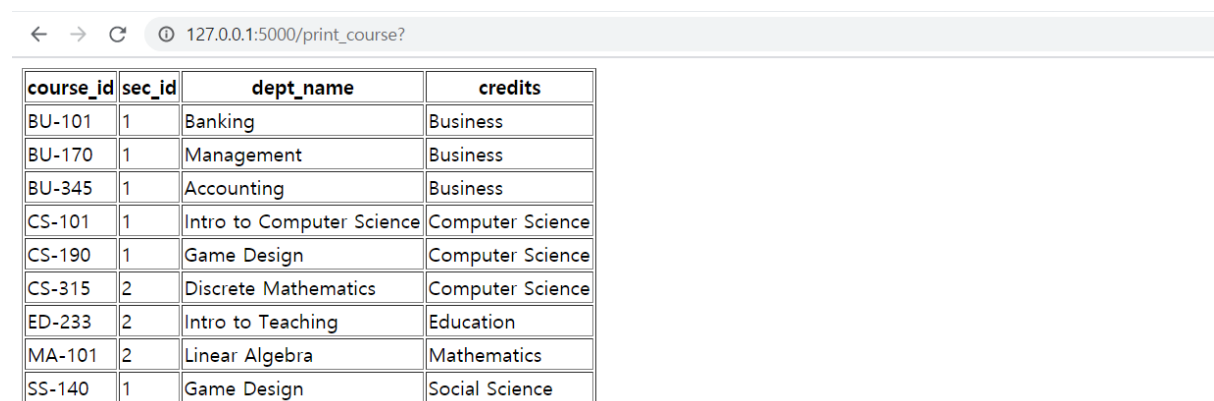
The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000". The page contains several input fields and buttons. The first section has input fields for "ID:", "name:", "dept_name:", and "tot_cred:", followed by a "register" button. Below this are buttons for "print_course" and "update salary". The second section has an "ID:" input field followed by an "unregister" button. The third section has a "Total budget" button. The fourth section has a "match_take_teach" button. The fifth section has a "match_dept_student" button. The sixth section has a "highest_salary" button.

app.py 실행 시 나오는 화면

문자를 입력할 수 있는 칸이 5개가 나오고 누를 수 있는 버튼이 8개 있습니다. 자세한 것은 다음 함수부터 설명하도록 하겠습니다.

2) print_course 함수

처음 화면에서 print_course를 누르면 현재 데이터베이스에 저장된 course 테이블에 대한 모든 정보가 html 파일에 테이블 형식으로 나옵니다.



course_id	sec_id	dept_name	credits
BU-101	1	Banking	Business
BU-170	1	Management	Business
BU-345	1	Accounting	Business
CS-101	1	Intro to Computer Science	Computer Science
CS-190	1	Game Design	Computer Science
CS-315	2	Discrete Mathematics	Computer Science
ED-233	2	Intro to Teaching	Education
MA-101	2	Linear Algebra	Mathematics
SS-140	1	Game Design	Social Science

위의 결과를 내기 위해 print_course.html을 작성하였고 코드는

```
@app.route('/print_course', methods=['GET'])
def print_course():
    cur.execute("SELECT * FROM course order by
course_id;")
    result = cur.fetchall()
    return render_template("print_course.html",
cous=result)
```

이 사용되었습니다.

3) create_student 함수

처음 화면에서 맨 위에 입력할 수 있는 칸이 4개가 있습니다. 이곳에 올바른 ID, name, dept_name, tot_cred을 입력하고 register를 누르면 새로운 student가 등록됩니다.

```
flask=# select*
flask=# from student;
```

id	name	dept_name	tot_cred
23121	Chavez	Computer Science	110
44553	Peltier	Mathematics	56
45678	Levy	Business	46
54321	Williams	Social Science	54
55739	Sanchez	Education	38

(5개 행)

```
flask=# select*
flask=# from student;
```

id	name	dept_name	tot_cred
23121	Chavez	Computer Science	110
44553	Peltier	Mathematics	56
45678	Levy	Business	46
54321	Williams	Social Science	54
55739	Sanchez	Education	38
46326	Seongjoon	Computer Science	341

(6개 행)

html파일 각각의 칸에 46326, Seongjoon, Computer Science,341를 입력하고 register를 눌렀더니 데이터베이스에 새로운 student가 추가되

었습니다.

코드는

```
@app.route('/create_student', methods=['POST'])
def create_student():
    try:
        id = request.form["id"]
        name = request.form["name"]
        dept_name = request.form["dept_name"]
        tot_cred = request.form["tot_cred"]

        cur.execute("INSERT INTO student
VALUES('{}','{}','{}','{}');".format(id, name,
dept_name, tot_cred))
        connect.commit()
        return render_template("main.html")
    except:
        return "create failed"
```

이 사용되었고 sql의 **insert** 를 사용했습니다.

4) delete_student 함수

처음의 html 화면에서 ID를 입력하는 곳이 있고 unregister 버튼이 있습니다. 이곳에 올바른 student ID를 입력하고 unregister를 누르면 student 테이블에서 해당 학생의 정보가 다 삭제됩니다.

```
flask=# select*
flask=# from student;
```

id	name	dept_name	tot_cred
23121	Chavez	Computer Science	110
44553	Peltier	Mathematics	56
45678	Levy	Business	46
54321	Williams	Social Science	54
55739	Sanchez	Education	38
46326	Seongjoon	Computer Science	341

(6개 행)


```
flask=# select*
flask=# from student;
```

id	name	dept_name	tot_cred
23121	Chavez	Computer Science	110
44553	Peltier	Mathematics	56
45678	Levy	Business	46
54321	Williams	Social Science	54
55739	Sanchez	Education	38

(5개 행)

ID에 46326을 입력하고 unregister를 눌렀더니 Seongjoon의 정보가 지워졌습니다.

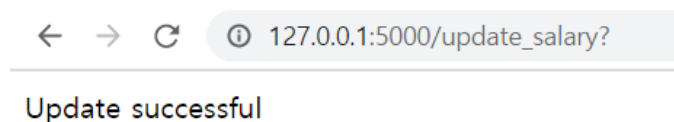
코드는

```
@app.route('/delete_student', methods=['POST'])
def delete_student():
    try:
        id = request.form["id"]
        cur.execute("DELETE FROM student where
ID=('{})'.format(id))
        connect.commit()
        return render_template("main.html")
    except:
        return "delete failed"
```

이 사용되었고 sql의 **delete**기능을 사용하였습니다.

5) update_salary 함수

처음의 화면에서 update salary 버튼이 있습니다. 이 버튼을 누르면 "update successful"이라는 문구가 나오고 모든 instructor의 salary가 1.03배 인상됩니다.



```
flask=# select salary
flask=# from instructor;
 salary
-----
 75000.00
 62000.00
 80000.00
 72000.00
 92000.00
(5개 행)
```

```
flask=# select salary
flask=# from instructor;
 salary
-----
 77250.00
 63860.00
 82400.00
 74160.00
 94760.00
(5개 행)
```

Update salary 버튼을 누르기 전보다 1.03배 salary가 많아진 것을 확인할 수 있습니다.

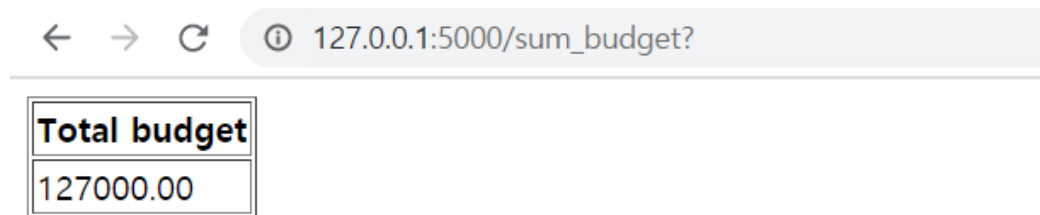
코드는

```
@app.route('/update_salary', methods=['GET'])
def update_salary():
    cur.execute("update instructor set
salary=salary*1.03;")
    connect.commit()
    return "Update successful"
```

이 사용되었고 sql의 **update**기능을 이용하였습니다.

6) sum_budget 함수

처음의 화면에서 Total budget을 누르면 모든 department의 budget의 합이 나옵니다.



코드는

```
@app.route('/sum_budget', methods=['GET'])
def sum_budget():
    cur.execute("SELECT SUM(budget) FROM department;")
    result = cur.fetchall()
    return render_template("budget.html", tot=result)
```

을 사용하였고 sql의 **aggregate function**을 이용하였습니다.

또한 이 함수를 구현하기 위해 budget.html을 만들었습니다.

7) match_take_teach 함수

처음의 화면에서 match_take_teach 버튼을 누르면 특정 과목의 특정 분반을 가르치는 교수님과 그 수업을 듣는 학생이 나옵니다. 어느 학생이 어느 교수님의 수업을 듣는지, 그리고 그 수업에 대한 정보를 출력하기 위해 만든 함수입니다.

← → ↻ 127.0.0.1:5000/match_take_teach?

takes_id	teaches_id	course_id	sec_id
45678	76543	BU-101	1
45678	76543	BU-170	1
23121	45565	CS-101	1
23121	45565	CS-190	1
55739	83821	ED-233	2
44553	58583	MA-101	2
54321	76766	SS-140	1

실행결과입니다.

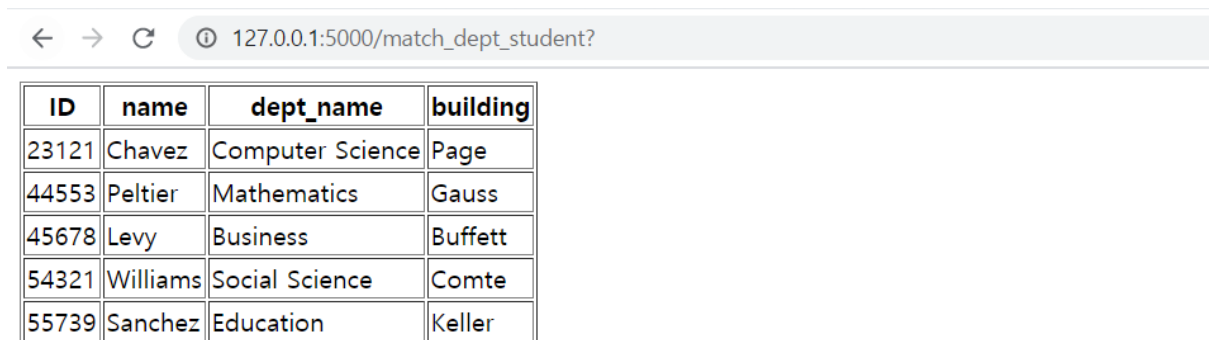
코드는

```
@app.route('/match_take_teach', methods=['GET'])
def match_take_teach():
    cur.execute("select
takes.ID,teaches.ID,takes.course_id,takes.sec_id "
                "from takes, teaches "
                "where takes.course_id=teaches.course_id
and takes.sec_id=teaches.sec_id;")
    result = cur.fetchall()
    return render_template("match_take_teach.html",
qu=result)
```

을 이용하였고 sql의 where절을 사용한 **Cartesian product**를 사용하였습니다. 또한 함수구현을 위해 match_take_teach.html을 만들었습니다.

8) match_dept_student 함수

처음 화면에서 match_dept_student 버튼을 누르면 student의 ID, 이름, dept_name, building이 나옵니다. 어떤 학생의 소속 단과대는 어디이고 그 단과대의 건물은 무엇인지 출력하기 위해 만든 함수입니다.



ID	name	dept_name	building
23121	Chavez	Computer Science	Page
44553	Peltier	Mathematics	Gauss
45678	Levy	Business	Buffett
54321	Williams	Social Science	Comte
55739	Sanchez	Education	Keller

실행결과입니다.

코드는

```
@app.route('/match_dept_student', methods=['GET'])
def match_dept_student():
    cur.execute("select ID,name,dept_name,building "
                "from student natural join department;")
    result = cur.fetchall()
    return render_template("match_dept_student.html",
                           pa=result)
```

을 사용하였고 sql의 **natural join** 기능을 사용하였습니다. 또한 함수 구현을 위해 match_dept_student.html을 만들었습니다.

9) highest_salary 함수

처음의 화면에서 highest_salary 버튼을 누르면 가장 salary가 많은 instructor의 정보가 나옵니다. 누가 가장 많은 돈을 받는지 보기 위해 만든 함수입니다.

← → ↻ ⓘ 127.0.0.1:5000/highest_salary?		
ID	Name	Salary
83821	Brandt	94760.00

실행결과입니다.

코드는

```
@app.route('/highest_salary', methods=['GET'])
def highest_salary():
    cur.execute("select ID,name,salary from instructor
where salary "
               ">=all(select salary from instructor);")
    result = cur.fetchall()
    return render_template("highest_salary.html",
yu=result)
```

이것을 사용하였고 sql의 **nested subquery**를 사용하였습니다. 또한 함수구현을 위해 highest_salary.html을 만들었습니다.

이상으로 Application 함수 설명을 마치겠습니다.