2019 가을학기 데이터베이스 HW4 보고서

7조(조원: 김보창\*, 강태호, 박은천, 송영우)

개요

총 6개의 모듈로 구성되었다. Main, login, userAcc, DBconnection, studentMenu, instructorMenu의 6개 모듈로 구성되어 있으며 각각의 역할은 다음과 같다.

각 모듈에 대한 설명

Main.py

프로그램의 메인 모듈. Login 모듈의 login 함수를 호출하는 것이 하는 일의 전부다.

UserAcc.py

UserAcc 클래스를 정의한 모듈이다. UserAcc는 유저의 계정을 나타내며, 유저의 ID, name, 역할(student인지 instructor인지), 그리고 connection object를 인자로 가진다.

이 class가 connection object를 인자로 가지는 이유는, 한 명의 유저는 DB와 하나의 connection을 이용하기 때문이다.

또한, UserAcc 클래스의 인스턴스인 user\_acc 전역변수를 선언한다.

DBConnection.py

MySQLdb 모듈을 이용해서, database와의 connection을 담당하는 모듈.

Database의 정보를 이용해서 database와의 connection을 만들어서 전역변수인 connect\_pool에 넣어주고 관리해주는 함수들로 구성되어 있다.

connectDB() : db\_host(데이터 베이스 주소), db\_user, db\_pw, db\_name 변수를 이용하여 데이터베이스와의 connection을 만든다.

get\_connect() : connect\_pool에서 connection을 하나 빼온다. 만약 connect\_pool에 connection이 없다면, 새로운 conneciton을 만든다.

return\_connect() : connect\_pool로 conection을 다시 추가한다.

login.py

처음 사용자가 프로그램을 실행했을 때, 로그인하는 부분을 담당하는 부분이다. 다음 두 함수로 구성된다.

login() : 처음 사용자가 로그인을 시도할 때, id와 이름을 받아서 해당하는 메뉴를 불러주는 함수.

auth() : login()함수 내부에서 불리며, 해당하는 ID, name이 유효한지, 유효하다면 instructor인지 student인지 판단해준다.

이를 구현하기 위해서, 유저로부터 입력 받은 ID와 name이 먼저 student나 instructor DB에 존재하는지를 다음 코드(쿼리)를 이용해서 확인한다.

# 입력 받은 ID, name이 student DB에 존재하는지 확인  
c.execute("SELECT \* FROM student \  
 WHERE ID = \"%s\" and name = \"%s\"" % (ID, name))  
data = c.fetchone()  
  
if data is not None:  
 user\_acc.set\_attrs(ID, name, 0, user\_connect)  
 return

해당 쿼리로 리턴 받은 결과가 없으면 data는 None이므로,

user는 student는 아니라는 이야기가 되고,

student table에 해당하는 ID와 이름을 가진 데이터가 있으면, 이 사람은 student라는 뜻이므로 user\_acc의 attribute를 그에 맞게 고쳐준다.

student가 아닌 경우, instructor table에 해당하는 ID와 이름을 가진 데이터가 있는지 마찬가지로 확인하는 과정을 거치고, instructor에 해당 데이터가 존재하면 user\_acc의 attribute를 그에 맞게 고쳐주고, 그렇지 않은 경우 이 사람은 student도, instructor도 아니므로 잘못된 접근임을 알 수 있다. 따라서 이 경우에는 에러메시지를 띄우고 다시 입력을 받는다.

# 입력 받은 ID, name이 instructor DB에 존재하는지 확인  
c.execute("SELECT \* FROM instructor \  
 WHERE ID = \"%s\" and name = \"%s\"" % (ID, name))  
data = c.fetchone()  
  
if data is not None:  
 user\_acc.set\_attrs(ID, name, 1, user\_connect)  
 return  
  
# print that the id and the name are wrong.  
print("Wrong authentication.")  
return\_connect(user\_connect)

studentMenu.py

유저가 학생일 경우 연관된 작업들을 진행하는 모듈이다.

크게 student\_menu(), print\_stud\_report(), print\_time\_table() 함수로 구성된다.

student\_menu() : 메뉴를 띄우고, 입력을 받아서 해당 입력에 맞는 작업을 수행한다.

print\_stud\_report() : student report를 출력하는 부분을 담당하는 함수다.

먼저 user의 정보를 출력해야하므로, User\_acc에 user\_id와 user\_name정보가 있으므로 이를 이용해서 다음과 같이 DB에 쿼리를 날려 학생의 소속 부서와 총 학점을 가져올 수 있다.

먼저 connection object의 cursor()함수를 이용해 쿼리를 실행하기 위한 cursor object를 가져오고,

다음 코드를 이용하여 유저의 정보를 가져왔다.

c = user\_acc.conn.cursor()  
c.execute("SELECT \* FROM student \  
 WHERE ID = \"%s\" and name = \"%s\"" % (user\_acc.ID, user\_acc.name) )  
data = c.fetchone()

student table에 부서명과 총 credit이 존재하므로, 해당 쿼리를 사용하였다.

data는 쿼리 결과로 리턴된 정보들을 튜플로 가지고 있으므로, data[index]를 이용해서 필요한 정보를 가져왔다.

그 뒤에는 학생이 들은 과목들에 대한 레포트를 학기별로 가까운 시일부터 작성해야 하므로,

먼저 다음과 같은 코드를 통해 학생이 어떤 학기에 수업을 들었는지를 파악하였다.

c.execute("SELECT DISTINCT year, semester FROM takes \  
 WHERE ID = \"%s\" \  
 ORDER BY year DESC, \  
 case when semester like 'Winter' then 1 \  
 when semester like 'Fall' then 2 \  
 when semester like 'Summer' then 3 \  
 when semester like 'Spring' then 4 \  
 else 5 end" % (user\_acc.ID)) #

takes table에서 학생의 ID를 이용하면, 각 과목들의 year, semester 정보를 얻을 수 있고, 중복 없는 year, semester 정보를 얻기 위해 distinct를 이용하였다. 또한, 이러한 year, semester정보를 시간순으로 뒤에 있는 것부터 출력해야 하므로, order by문을 이용하여 연도, 학기의 계절에 따라 정렬되게 하였다. 여기서 case~ when then else문을 사용하였는데, 겨울일 경우 가장 작은 값, 봄일 경우 가장 큰 값을 가지므로 같은 year일때 계절이 겨울에 가까울수록 먼저 출력되게 하였고, 혹시나 학기에 대한 정보가 4계절이 아닐 경우는 같은 year에서 가장 마지막에 출력되게 하였다.

각 학기에 대한 정보가 있으므로, for문을 이용해 각 학기마다의 report를 출력한다.

각 학기마다 그 학생이 들은 과목들의 GPA를 계산해야 하므로,

c.execute("SELECT grade, credits FROM course natural join takes \  
 WHERE ID = \"%s\" and semester = \"%s\" and year = %s" % (user\_acc.ID, semester, year))

를 통해 그 학기에 학생이 들은 수업마다 grade들과 credit들을 얻어오고, 다음과 같이 for문을 이용해 총 gpa를 구했다.

grade\_credits = c.fetchall()  
  
credit\_sum = 0  
grade\_mul\_credit\_sum = 0  
  
for grade\_credit in grade\_credits:  
 grade = grade\_credit[0]  
 if(grade is None) : continue # null값을 가지고 있으면 cursor에서 리턴된것은 None값을 가짐. 이럴땐 무시.  
 if(grade\_credit[1] != None): credit = float(grade\_credit[1])  
 else: credit = 0  
  
 credit\_sum += credit  
 grade\_mul\_credit\_sum += gp\_to\_float(grade) \* credit

이때, grade의 경우 DB에서 null값을 가질수 있고, 이때는 해당 결과가 None값을 가지므로

grade가 None일경우 총 gpa를 계산하는 과정에서 이를 무시하도록 하였다.

## gpa가 NULL인경우 처리.  
if(credit\_sum == 0) :  
 print("\n%s\t%s\tGPA : " % (year, semester))  
else:  
 gpa = grade\_mul\_credit\_sum / credit\_sum  
 print("\n%s\t%s\tGPA : %f" % (year, semester, gpa))

그 뒤, 모든 과목의 grade 가 NULL인 경우는 credit\_sum 이 0이 될 것이므로, 이 경우는 따로 빼서 GPA를 출력하지 않도록 하였고, 그렇지 않은 경우는 GPA를 계산하여 출력하였다.

print("%10s\t%40s\t%15s\t%8s\t%8s" % ("course\_id", "title", "dept\_name", "credit", "grade"))  
c.execute("SELECT course\_id, title, dept\_name, credits, grade FROM course natural join takes \  
 WHERE ID = \"%s\" and semester = \"%s\" and year = %s" % (user\_acc.ID, semester, year))  
courses = c.fetchall()  
  
for course in courses:  
 course\_id = course[0]  
 title = course[1]  
 dept\_name = course[2]  
 credits = course[3]  
 grade = course[4]  
  
 if(grade is None):  
 credits = ""  
 grade = ""  
  
 print("%10s\t%40s\t%15s\t%8s\t%8s" % (course\_id, title, dept\_name, credits, grade))

그 후로는 위와 같은 코드를 이용해 데이터를 받아오고 출력한다.

학생의 ID와 언제 수업이 열렸는지를 이용하면 해당 학기에 그 학생이 들은 과목들에 대한 course\_id, title, dept\_name, credits, grade는 course와 takes 테이블을 natural join하면 얻어낼 수 있으므로, 이를 얻어낸 다음, 이 정보들을 형식에 맞게 출력해주었다.

이때, grade가 NULL인경우는 credits와 grade 부분을 비워서 출력하게 하였다.

print\_time\_table() : 학생이 수업을 들은 학기 중, 한 학기를 골라 그 학기의 time table을 출력해주는 함수다.

c = user\_acc.conn.cursor()  
  
# 수강한 학기, 연도정보 모두 가져오기 (distinct로 중복 계산 방지)  
  
c.execute("SELECT DISTINCT year, semester FROM takes \  
 WHERE ID = \"%s\" \  
 ORDER BY year DESC, \  
 case when semester like 'Winter' then 1 \  
 when semester like 'Fall' then 2 \  
 when semester like 'Summer' then 3 \  
 when semester like 'Spring' then 4 \  
 else 5 end" % (user\_acc.ID)) #  
  
year\_and\_semester = c.fetchall()

먼저 위와 같이 print\_stud\_report() 함수와 같이 중복되지 않게 그 학생이 들었던 year와 semester data를 가져온 다음, 만약 학생이 수업을 들은 학기가 없다면 year\_and\_semester의 값이 None이 될 것이고, 수업을 들은 학기가 있으면 해당 결과가 리스트의 형태로 리턴될 것이므로, if문을 이용해 두 경우를 처리하였다.

if year\_and\_semester:  
 print("Please select semester to view")  
 for (i,year\_semester) in enumerate(year\_and\_semester):  
 year = year\_semester[0]  
 semester = year\_semester[1]  
 print("%d) %s %s" % (i+1, year, semester))  
 input\_num = input()  
  
 # handle if input is uncorrect value  
 try:  
 num = int(input\_num)  
 if(num > len(year\_and\_semester) or num <= 0):  
 print("Please insert correct value")  
 c.close()  
 return  
  
 except Exception as e:  
 print("Please insert correct value")  
 c.close()  
 return

year\_and\_semester가 None이 아닌 경우, 각 semester들을 메뉴로 출력해주고, 유저가 어떤 semester에 대한 정보를 원하는지 받는다. 이때, 입력한 값이 정수가 아니거나, 정수지만 메뉴에 나온 값이 아닐 경우는 입력을 제대로 해달라는 메시지를 띄우고 프로그램의 첫부분으로 돌아간다. (student menu 부분)

# 타임테이블 출력  
print("\nTime Table\n")  
print("%10s\t%40s\t%15s\t%10s\t%10s" % ("course\_id", "title", "day", "start\_time", "end\_time"))  
year = year\_and\_semester[num-1][0]  
semester = year\_and\_semester[num-1][1]  
  
# time\_table 데이터 받아오기  
c.execute("SELECT course\_id, title, day, start\_hr, start\_min, end\_hr, end\_min FROM takes natural join course natural join section natural join time\_slot \  
 WHERE ID = \"%s\" and semester = \"%s\" and year = %s ORDER BY start\_hr desc, start\_min desc" % (user\_acc.ID, semester, year))  
  
time\_table\_rows = c.fetchall()

타임테이블을 나타내는 첫 문장을 출력한다음, 위 쿼리를 이용해 timetable을 출력하는데 필요한 정보를 받아온다. 이때, 학생의 id와 학기의 년과 계절을 이용해 필요한 정보를 가져와야 하는데, 각 course의 course\_id, title은 course table에, 학생의 id를 통해 어떤 course를 수강했는지는 takes table을 통해 알 수 있으므로 먼저 course와 takes 테이블을 natural join을 통해 join하고, 각 course가 열린 시간을 알기 위해서는 section테이블의 각 course\_id에 연관된 time\_slot\_id와, 각 time\_slot\_id와 연관된 시간 정보를 알기 위해서는 time\_slot table의 정보가 필요하므로,

결국 takes, course, section, time\_slot 테이블을 차례대로 natural join 한 다음, 주어진 id와 학기를 통해 필요한 정보를 얻어오도록 하였다.

또한, 각 정보를 시간의 역순으로 출력해야 하므로, start\_time (start\_hr와 start\_min)을 기준으로 하여 order by 문을 통해 이 두 변수의 내림차순으로 쿼리의 결과를 내놓게 하였다.

if time\_table\_rows:  
 # course에 해당하는 section table의 time\_slot\_id의 값이 timeslot table에 없는경우, 제외하고 출력함.  
 for time\_table\_row in time\_table\_rows:  
 course\_id = time\_table\_row[0]  
 title = time\_table\_row[1]  
 day = time\_table\_row[2]  
 start\_time = str(time\_table\_row[3]) + " : " + str(time\_table\_row[4])  
 end\_time = str(time\_table\_row[5]) + " : " + str(time\_table\_row[6])  
  
 print("%10s\t%40s\t%15s\t%10s\t%10s" % (course\_id, title, day, start\_time, end\_time))  
 else:  
 # 모든 course들의 time\_slot\_id의 값이 timeslot table에 없는 경우  
 print("there is no course that has a valid time\_slot id")  
else:  
 print("No option to Select")

그 후, 결과로 나온 데이터들을 포맷에 맞게 묶어서 출력하도록 하였다.’

이때, section table의 time\_slot\_id값이 timeslot table의 값으로 들어있지 않은 경우는 제외하고 출력하게 된다.

이때, 해당 학기의 모든 course들의 time\_slot\_id값이 timeslot table에 없어서 유효한 시간표를 가져오지 못할 경우, 관련 경고 메시지를 출력해준다.

아래의 else문의 경우는 학생이 수업을 들은 학기가 없을 때 출력하는 문장이다.

instructorMenu.py

유저가 강사일 경우 연관된 작업들을 진행하는 모듈이다.

크게 instructor\_menu(), print\_course\_report(), print\_advisee\_table() 함수로 구성된다.

instructor\_menu() : 메뉴를 띄우고, 입력을 받아서 해당 입력에 맞는 작업을 수행한다.

print\_course\_report() : course report를 출력하는 함수.

먼저 교수가 강의한 학기와 연도정보를 모두 가져와야 하므로 다음 코드로 가져왔다.

c = user\_acc.conn.cursor()  
  
# 강의한 학기, 연도정보 모두 가져오기 (distinct로 중복 계산 방지)  
  
c.execute("SELECT DISTINCT year, semester FROM teaches \  
 WHERE ID = \"%s\" \  
 ORDER BY year DESC, \  
 case when semester like 'Winter' then 1 \  
 when semester like 'Fall' then 2 \  
 when semester like 'Summer' then 3 \  
 when semester like 'Spring' then 4 \  
 else 5 end" % (user\_acc.ID))  
  
year\_and\_semester = c.fetchall()

table이 teaches로 바뀌었다는 것만 빼면 앞과 똑같은 코드이므로 설명은 생략한다.

이 쿼리의 결과로 나온 첫번째 행이 가장 최근에 교수가 강의했던 년도와 계절을 포함하고 있으므로, 그 정보를 활용하여 아래와 같이 코드를 작성하였다.

if year\_and\_semester:  
 # get most recent semester.  
 year = year\_and\_semester[0][0]  
 semester = year\_and\_semester[0][1]  
  
 # couser report - year semester 출력.  
 print("\nCourse report - %d %6s\n" % (year,semester))  
  
 # find courses at that semester  
 c.execute("SELECT course\_id from teaches \  
 where ID = \"%s\" and semester = \"%s\" and year = %s" % (user\_acc.ID, semester, year))  
  
 most\_recent\_courses = c.fetchall()

먼저 교수가 학교에서 강의한적이 있을 때, 가장 최근에 강의한 그 학기에 있는 course들의 id를 아래 쿼리로 찾아주었다.

for most\_recent\_course in most\_recent\_courses:  
 recent\_course\_id = most\_recent\_course[0]  
  
 # 각 course에 대해 course가 열렸던 위치, course의 정보 출력.  
 c.execute("SELECT title, building, room\_number, day, start\_hr, start\_min, end\_hr, end\_min FROM course natural join section natural join time\_slot \  
 WHERE course\_id = \"%s\" and semester = \"%s\" and year = %s" % (recent\_course\_id, semester, year))  
  
 course\_datas = c.fetchall()

그리고나서, 위와 같이 찾은 course\_id와 semester, year를 이용하여 각 course에 대해 course의 이름, building, 방 번호, 요일, 시작시간과 끝시간을 얻어왔다.

이때 course 테이블에는 course id와 이름 정보가, section에는 course id와 빌딩, 방 번호, 강의가 열렸던 학기 (년도와 계절정보), time\_slot id정보가 있고, time\_slot에는 각 강의의 시작시간, 끝 시간, 열리는 요일등의 데이터가 있으므로, 이 세 table을 natural join한다음 우리가 찾은 course\_id와 semester, year정보를 이용하면 우리가 원하는 정보를 얻어낼 수 있을 것이라 생각하여 위와 같이 관련 데이터를 구해주었다.

#course 정보 출력  
if course\_datas:  
 days = []  
 for course\_data in course\_datas: # course가 열린 day들 추출  
 days.append(course\_data[3])  
  
 title = course\_datas[0][0]  
 building = course\_datas[0][1]  
 room\_number = course\_datas[0][2]  
 day = ", ".join(days)  
 start\_time = str(course\_datas[0][4]) + " : " + str(course\_datas[0][5])  
 end\_time = str(course\_datas[0][6]) + " : " + str(course\_datas[0][7])  
  
 print("%10s\t%40s\t[%s %s]\t(%s %7s - %7s)" % (recent\_course\_id, title, building, room\_number, day, start\_time, end\_time))  
else:  
 # 만약 section table의 time\_slot\_id의 데이터가 timeslot table의 없는값인 경우, 부분 정보만 받아옴.  
 c.execute("SELECT title, building, room\_number FROM course natural join section \  
 WHERE course\_id = \"%s\" and semester = \"%s\" and year = %s" % (recent\_course\_id, semester, year))  
 course\_datas = c.fetchall()  
 if course\_datas:  
 title = course\_datas[0][0]  
 building = course\_datas[0][1]  
 room\_number = course\_datas[0][2]  
 print("%10s\t%40s\t[%s %s]\t" % (recent\_course\_id, title, building, room\_number))  
 else:  
 # 아예 해당하는 데이터가 없는경우, 오류메시지 출력.  
 print("course\_id %s have missing data. we can't get course data." % recent\_course\_id)

그 후에는, 위와 같이 course에 대한 정보를 출력해주었다. 이때, 만약 course에 대한 데이터 중 section table에서의 time\_slot\_id의 값이 timeslot table에 없는 값인 경우, natural join을 했을 때 결과가 없을 것이므로, 이 경우는 부분적인 정보만 출력하도록 하였다.

또한, join한 테이블의 결과의 누락 등으로 course의 정보 자체를 못 얻어온 경우에는, 그에 관련된 경고메시지를 else부분에 출력해주었다.

#course들을 수강했던 학생들 데이터 출력  
  
 c.execute("SELECT ID, name, dept\_name, grade FROM student natural join takes \  
 WHERE course\_id = \"%s\" and semester = \"%s\" and year = %s" % (recent\_course\_id, semester, year))  
  
 students = c.fetchall()  
  
 for student in students:  
 student\_id = student[0]  
 student\_name = student[1]  
 dept\_name = student[2]  
 grade = student[3]  
  
 if(grade == None) : grade = "" # handle grade is null  
  
 print("%10s\t%20s\t%15s\t%8s" % (student\_id, student\_name, dept\_name, grade))  
  
 print("")  
else:  
 print("there is no course instructor %s teaches" % user\_acc.name)

그 후, course를 수강했던 학생들의 데이터를 출력해주었다.

이미 course\_id와 semester, year정보가 있으므로, student와 takes 테이블을 natural join한다음 이 정보를 가지고 관련 데이터를 찾을 수 있었다.

마지막의 else 부분은 instuctor가 이 학교에서 강의한 전적이 없을 경우에 출력되는 문장이다.

print\_advisee\_report() : advisee report를 출력하는 함수이다.

c = user\_acc.conn.cursor()  
  
c.execute("SELECT student.ID, student.name, student.dept\_name, student.tot\_cred FROM student, advisor\  
 WHERE student.ID = advisor.s\_id and advisor.i\_id = \"%s\"" % user\_acc.ID)  
  
students = c.fetchall()  
  
print("%10s\t%20s\t%15s\t%8s" % ("ID", "name", "dept\_name", "tot\_cred"))  
  
for student in students:  
 student\_id = student[0]  
 student\_name = student[1]  
 dept\_name = student[2]  
 tot\_cred = student[3]  
 print("%10s\t%20s\t%15s\t%8s" % (student\_id, student\_name, dept\_name, tot\_cred))  
  
c.close()  
  
return

advisor의 id를 user\_acc.ID로 가지고 있으므로, 이 정보를 이용하면 간단하게 원하는 정보를 얻어낼 수 있다.

student table에는 학생의 ID, 이름, 부서이름과 총 학점이 있고, advisor 테이블에는 어떤 교수가 어떤 학생을 지도하는지가 나와있으므로, 이 두 테이블을 join하여 정보를 구하였다.

이때, 앞과는 다르게 natural join을 사용하지 않았는데, 그 이유는 두 테이블에서 join에 사용되어야할 column의 이름이 다르므로, (student table의 ID와 advisor table의 s\_id를 이용해 join해야함) 이를 명시적으로 where절에서 지정하여 join을 수행하였다.

그 후 for문을 이용해 데이터를 출력하고, cursor를 닫고 리턴한다.

역할분담

김보창 : 코드 작성 및 쿼리 작성

강태호 : 코드 작성 및 테스트

송영우 : 쿼리 작성 및 보고서 작성

박은천 : 쿼리 작성 및 테스트