

Project Member Management Database

Development

– CODE Details

Using SQL & Python

CONTENTS

- 상황 설명
- DB 개념&논리 설계
- python을 이용한 서비스 구현



상황 설명

팀프로젝트 본질적 가치 하락

- 팀 프로젝트의 본질적 목적은 타인과의 의사소통 능력 향상인데 현재 많은 프로젝트가 친한 사람들과 팀이 꾸려지는 경향 존재
 - 이에 새로운 방식의 팀원 구성의 필요성 증진

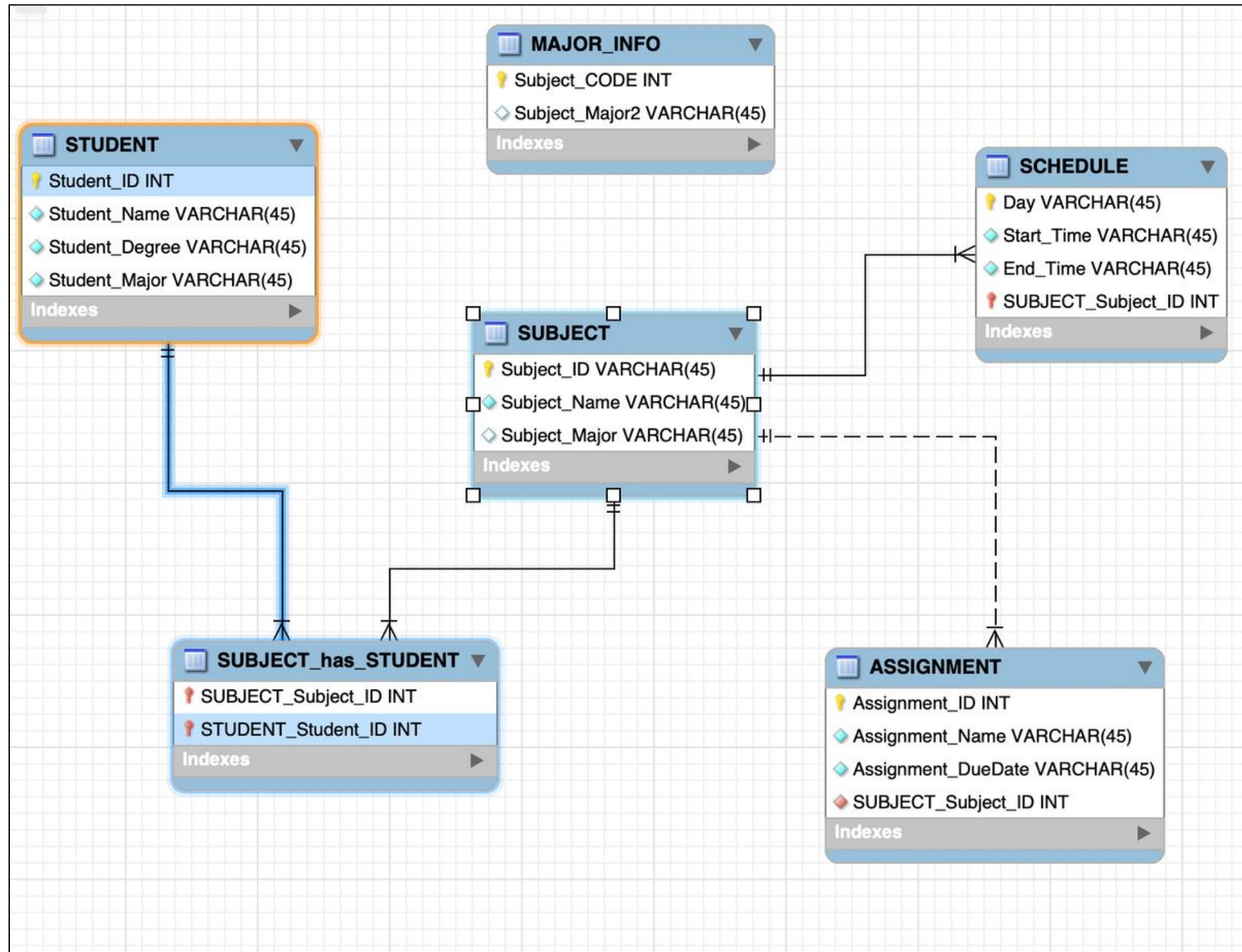
인원 랜덤 배정 시 시간 조정의 어려움

최대한 공강 시간이 겹치는 학우들끼리 팀 구성을 통해 회의 시간 조정의 효율성을 증대시키고자 함.

팀원 구성 시 불편함 및 비효율 문제

1차적으로 원하는 구성원끼리 메일 작성하여 구성 후 나머지는 랜덤 배정하는 현재의 방식은 비효율적이라 생각하여 본질적 의미에 더하여 랜덤 팀원 구성의 필요성을 느낌.

DB의 필요성

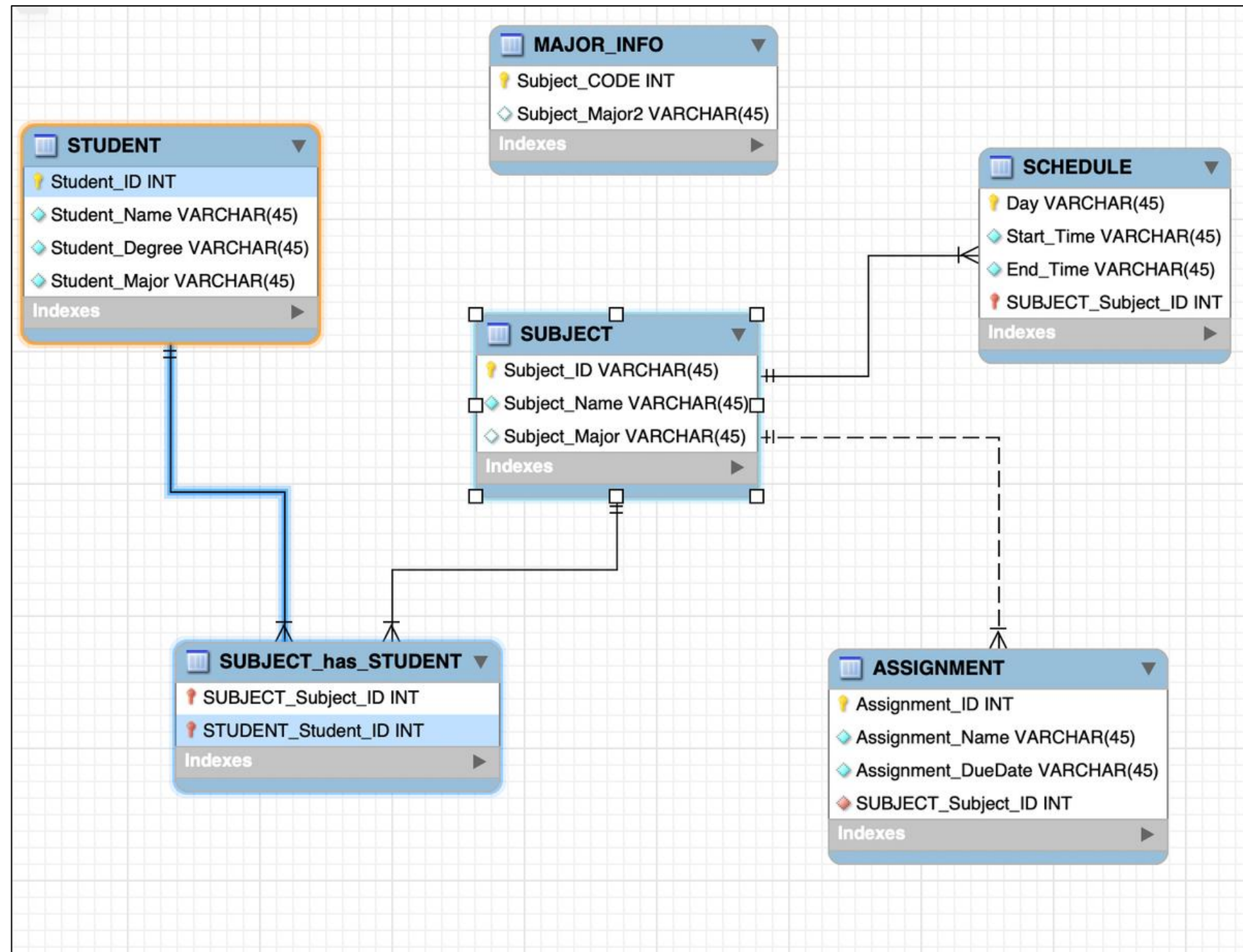


팀프로젝트 팀을 구성하는 입장에서 인원들끼리 자발적으로 팀을 구성하여
배정되지 않은 인원들만 랜덤하게 배정되는 시스템에 비효율성이 있다고
생각하였다. 따라서 다음과 같은 테이블을 설계하여 개인의 스케줄을 고려
하여 최적의 팀 구성을 실현하는 것을 목표로 삼았다.

또한 과제를 입력하는 관리자가 GUI에 해당 과목의 과제를 입력하면 해당
과제에 따른 우리가 설정한 기준에 부합하는 가장 최적의 팀 구성을 출력할
수 있다.

각 테이블에 대한 자세한 설명은 다음 페이지와 같다.

개념 설계



1. STUDENT & SUBJECT

STUDENT Table에는 다음과 같은 개인의 정보가 담겨있고 SUBJECT Table와 N:M 관계를 갖게 된다. 따라서 ERD 진행 시 두 Entity간에 Intersection Table이 필요하다.

2. SUBJECT & SCHEDULE

두 Table은 1:N 관계를 갖기에 SUBJECT Table의 Subject_ID(Pk)를 SCHEDULE Table이 Fk로 갖게 되고 이 테이블은 강의 시작 시간과 끝 시간을 포함한다. 주 5회 평일을 기준으로 오전9시 시작을 1교시로 가정하여 하루에 8교시까지 존재하고, 화요일 1교시부터는 9교시부터 1씩 증가하며 숫자를 할당한다.

3. SUBJECT & ASSIGNMENT

각 과목에 대해 여러 과제를 가질 수 있는 1:N의 관계를 갖는 두 테이블이다. ASSIGNMENT Table은 SUBJECT Table의 Pk를 Fk로 갖고 과제에 대한 정보를 포함하고 있다.

4. MAJOR_INFO

학과의 학정번호 앞 3자리와 학과명을 포함하고 있는 테이블이다.

CREATE TABLE

```
1 • CREATE TABLE STUDENT (  
2     Student_ID INT PRIMARY KEY,  
3     Student_Name VARCHAR(255) NOT NULL,  
4     Student_Degree VARCHAR(50) NOT NULL,  
5     Student_Major VARCHAR(100) NOT NULL  
6 );  
7  
8 CREATE TABLE SUBJECT (  
9     Subject_ID VARCHAR(30) PRIMARY KEY,  
10    Subject_Name VARCHAR(255) NOT NULL,  
11    Subject_Major VARCHAR(100) NOT NULL  
12 );  
13  
14 CREATE TABLE STUDENT_HAS_SUBJECT (  
15     SUBJECT_Subject_ID VARCHAR(30),  
16     STUDENT_Student_ID INT,  
17     PRIMARY KEY (SUBJECT_Subject_ID, STUDENT_Student_ID),  
18     FOREIGN KEY (SUBJECT_Subject_ID) REFERENCES SUBJECT(Subject_ID),  
19     FOREIGN KEY (STUDENT_Student_ID) REFERENCES STUDENT(Student_ID)  
20 );  
21
```

```
21  
22 • CREATE TABLE SCHEDULE (  
23     SUBJECT_Subject_ID VARCHAR(30),  
24     Day VARCHAR(10) NOT NULL,  
25     Start_Time INT NOT NULL,  
26     End_Time INT NOT NULL,  
27     FOREIGN KEY (SUBJECT_Subject_ID) REFERENCES SUBJECT(Subject_ID)  
28 );  
29  
30 • CREATE TABLE ASSIGNMENT (  
31     Assignment_ID VARCHAR(30) PRIMARY KEY,  
32     Assignment_Name VARCHAR(255) NOT NULL,  
33     Assignment_DueDate DATE NOT NULL,  
34     Assignment_DueTime INT NOT NULL,  
35     SUBJECT_Subject_ID VARCHAR(30),  
36     FOREIGN KEY (SUBJECT_Subject_ID) REFERENCES SUBJECT(Subject_ID)  
37 );
```

SQL의 CREATE 문을 사용하여 ERD에서 사용한 Data Model에 RDB 과정을 수행한다.

SQL INSERT 문을 이용해서 입력될 row 수가 적은 SUBJECT 테이블에만 적용시켰다.

row의 개수가 많이 필요한 STUDENT 테이블의 경우, CSV 파일을 미리 작성하여 파이썬에서 불러왔다.

```
2 • INSERT INTO SUBJECT (Subject_ID, Subject_Name, Subject_Major)  
3 VALUES  
4 ('IIE2102.01-00', '산업정보관리론', '산업공학'),  
5 ('IIE3102.01-00', '생산계획론', '산업공학'),  
6 ('EC03119.01-00', '금융공학이해', '경제학'),  
7 ('BIZ9014.03-00', '전략경영', '경영학');
```

TRIGGER & FUNCTION

```
1 DELIMITER //
2
3 • CREATE TRIGGER Insert_Assignment
4 BEFORE INSERT ON ASSIGNMENT
5 FOR EACH ROW
6
7 BEGIN
8
9     IF NOT EXISTS (SELECT * FROM SUBJECT WHERE SUBJECT.Subject_ID = NEW.SUBJECT.Subject_ID)
10 THEN
11     SIGNAL SQLSTATE '45000'
12     SET MESSAGE_TEXT = 'Insert not allowed: Only subjects with major "산업공학" are allowed.';
13 END IF;
14
15 END
16 //
17 DELIMITER ;
18
```

사용자로부터 입력받은 과목명과 일치하는 Subject_ID(학정번호)를 찾고 해당 Subject_ID의 .을 기준으로 앞 문자열만 출력하는 사용자 정의 함수를 구현하였다. 예를 들면, 산업정보관리론 과목에 대해서 "IE2102"이라는 리턴값을 출력한다.

```
#FUNCTION
#사용자로부터 과목명 입력받고 학정번호만 출력해주는 함수
DELIMITER //

CREATE FUNCTION GetSubjectCode(subjectName VARCHAR(255))
RETURNS VARCHAR(255)
DETERMINISTIC
BEGIN
    DECLARE subjectID VARCHAR(255);
    DECLARE subjectCode VARCHAR(255);

    #과목명과 일치하는 Subject_ID 찾기
    SELECT Subject_ID INTO subjectID
    FROM SUBJECT
    WHERE Subject_Name = subjectName;

    #'.' 이전의 문자열 (과목 코드) 추출
    SET subjectCode = SUBSTRING_INDEX(subjectID, '.', 1);

    RETURN subjectCode;
END //

DELIMITER ;
```

ASSIGNMENT에 DATA를 INSERT 하기 전에 발동하는 TRIGGER로, M-O 관계를 유지하기 위해 SUBJECT에 없는 값이 ASSIGNMENT에 삽입되려 하면 오류코드를 발생시키고 입력할 수 없게 한다.

PROCEDURE

```
DELIMITER //
```

- **CREATE FUNCTION** GetSubjectIDFirstThree(subject_ID **VARCHAR(30)**)
RETURNS VARCHAR(3)
DETERMINISTIC
BEGIN
 RETURN LEFT(subject_ID, 3);
END //

```
DELIMITER ;
```

```
DELIMITER //
```

사용자로부터 입력받은 varSubject_ID(학정번호)를 미리 정의한 User-defined function으로 학정번호 앞글자 3개만 출력해서 프로시저 작동을 진행한다. 다음 과정을 통해 VarSubject_Name(과목명)에서 출력한 3글자를 수강 정보가 담겨있는 MAJOR_INFO 테이블의 학정번호와 비교하여 동일한 학정번호에 대한 학과명을 도출한다.

```
DELIMITER //
```

- **CREATE PROCEDURE** InsertProcedure
 (
 IN varSubject_ID **VARCHAR(30)**, **IN** varSubject_Name **VARCHAR(30)**
)
BEGIN
 DECLARE firstThreeChars **VARCHAR(3)**;
 DECLARE MajorName **VARCHAR(10)** ;
 SET firstThreeChars = GetSubjectIDFirstThree(varSubject_ID);
 SELECT Subject_Major **INTO** MajorName
 FROM MAJOR_INFO
 WHERE Subject_CODE = firstThreeChars ;

 INSERT INTO SUBJECT (Subject_ID, Subject_Name, Subject_Major) **VALUES** (varSubject_ID, varSubject_Name, MajorName);
END
 //
DELIMITER ;

이를 통해서 최종적으로 INSERT INTO 문을 통해서 Subject_ID(학정번호)와 Subject_Name(과목명) 입력만을 통해 Subject_Major(학과명)이 자동적으로 입력되도록 PROCEDURE를 활용하였다.

SERVICE BY PYTHON

-CSV 파일 MYSQL에 IMPORT

```
import add_delete as ad

def delete_button_clicked():
    subject_id = subject_id_entry.get()
    assignment_name = assignment_name_entry.get()
    if ad.delete_assignment(subject_id, assignment_name):
        result_label.config(text="과제가 성공적으로 삭제되었습니다!")
    else:
        result_label.config(text="해당 과목에 해당 이름으로 등록된 과제가 없습니다.")

def submit_button_clicked():
    subject_id = subject_id_entry.get()
    assignment_name = assignment_name_entry.get()
    assignment_due_date = assignment_due_date_entry.get()
    assignment_due_time = assignment_due_time_entry.get()
    if ad.add_assignment(subject_id, assignment_name, assignment_due_date, assignment_due_time):
        result_label.config(text="과제가 성공적으로 등록되었습니다!")
    else:
        result_label.config(text="과제가 성공적으로 수정되었습니다!")

# UI 생성
root = tk.Tk()
root.title("과제 추가/수정/삭제하기")

# Subject ID 입력 필드
subject_id_label = ttk.Label(root, text="학정번호:")
subject_id_label.grid(row=0, column=0, padx=10, pady=10)
subject_id_entry = ttk.Entry(root)
subject_id_entry.grid(row=0, column=1, padx=10, pady=10)

# Assignment 이름 입력 필드
assignment_name_label = ttk.Label(root, text="과제명:")
assignment_name_label.grid(row=1, column=0, padx=10, pady=10)
```

```
import pandas as pd
import add_delete as ad

def import_csv(table_name):
    query = f'''
    DELETE FROM {table_name};
    '''

    ad.execute_query(query)
    df = pd.read_csv(f'{table_name}.csv')
    for index, row in df.iterrows():
        query = f'''
        INSERT INTO {table_name}
        VALUES(''
        i = 0
        for value in row.values:
            if i != 0:
                query += ', '
            if type(value) != int:
                query += f'''{str(value)}'''
            else:
                query += str(value)
            i+=1
        query += ');'
        ad.execute_query(query)

import_csv("STUDENT")
import_csv("SUBJECT")
import_csv("SCHEDULE")
```

첨부한 Import_CSV_to_mySQL.py 파일을 실행하여 CSV 파일을 MYSQL 테이블에 IMPORT 하였다.

SERVICE BY PYTHON

1.과제(ASSIGNMENT) 추가/삭제

```
import pymysql

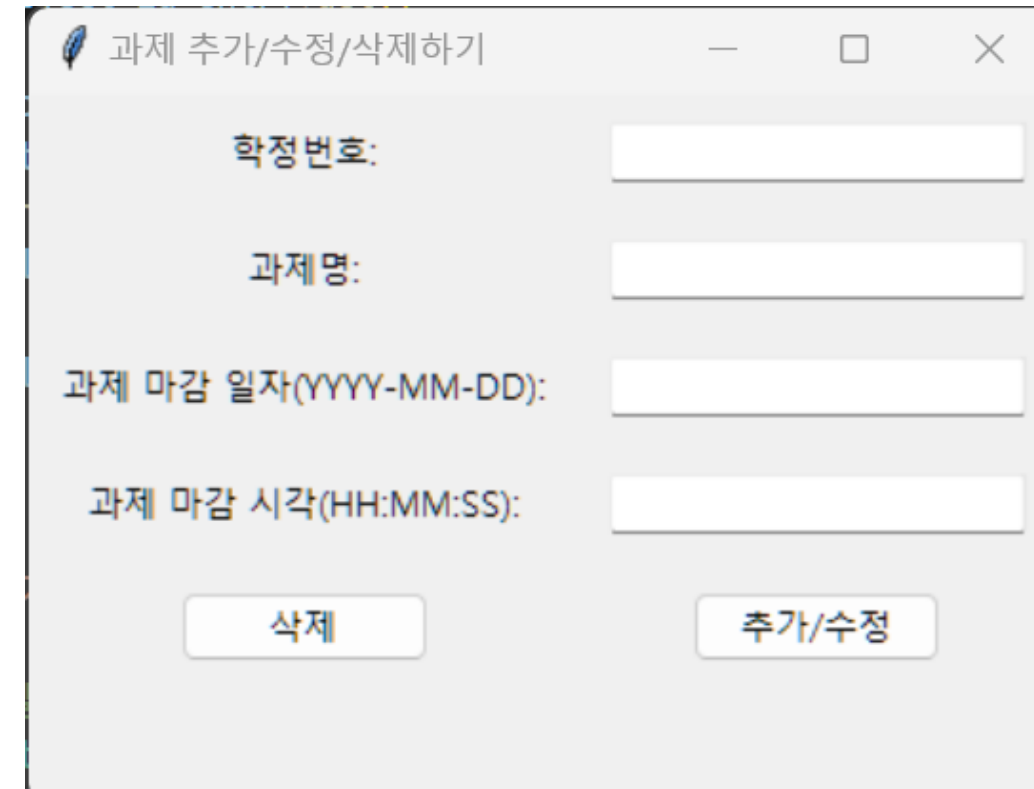
def execute_query(sql_query):
    try:
        # MySQL 연결
        connection = pymysql.connect(host='localhost',
                                     user='root',
                                     password='357456',
                                     database='term_project',
                                     charset='utf8mb4',
                                     cursorclass=pymysql.cursors.DictCursor)

        # SQL 쿼리 수행 예제
        with connection.cursor() as cursor:
            # 여기에 SQL 쿼리를 작성합니다.
            cursor.execute(sql_query)
            connection.commit()
            return cursor.fetchall()
        # 변경사항을 커밋

    finally:
        # 연결 종료
        connection.close()

def add_assignment(subjectID, assignment_name, assignment_duedate, assignment_duetime):
    query = f'''
        SELECT * FROM ASSIGNMENT
        WHERE Subject_ID = '{subjectID}' AND Assignment_Name = '{assignment_name}';
    '''

    num_of_rows = len(execute_query(query))
    if num_of_rows == 0:
        query = f'''
            INSERT INTO ASSIGNMENT(Subject_Id, Assignment_Name, Assignment_DueDate, Assignment_DueTime)
            VALUES('{subjectID}', '{assignment_name}', '{assignment_duedate}', '{assignment_duetime}');
        '''
        execute_query(query)
    return 1
```



과제 추가/수정/삭제하기

학정번호:

과제명:

과제 마감 일자(YYYY-MM-DD):

과제 마감 시각(HH:MM:SS):

첨부한 add_delete.py 파일을 실행하여 다음과 같은 인터페이스를 통해 관리자가 과제를 추가/삭제할 수 있다. 이때 Trigger를 사용해 M-O 관계가 강제 되도록 하였다.

SERVICE BY PYTHON

2.조별 과제 조 구성

```
def free_time(subject):
    subject_free = []
    Start = subject['Start_Time']
    End = subject['End_Time']
    if subject['Day'] == 'MON':
        free_time_list = list(range(Start, End))
    elif subject['Day'] == 'TUE':
        free_time_list = list(range(Start+8, End+8))
    elif subject['Day'] == 'WED':
        free_time_list = list(range(Start+16, End+16))
    elif subject['Day'] == 'THU':
        free_time_list = list(range(Start+24, End+24))
    elif subject['Day'] == 'FRI':
        free_time_list = list(range(Start+32, End+32))
    subject_free = subject_free + free_time_list
    return subject_free

def get_schedule_time_dict():
    result = ad.execute_query("SELECT * FROM SCHEDULE")

    free_time_dic = {}
    for i in result:
        free_time_dic[i['SUBJECT_Subject_ID']] = []
    for i in result:
        free_time_dic[i['SUBJECT_Subject_ID']] = free_time_dic[i['SUBJECT_Subject_ID']] + free_time(i)

    for key in free_time_dic.keys():
        free_time_dic[key] = set(free_time_dic[key])
    return free_time_dic
```

```
from Schedule_Time import get_schedule_time_dict
import add_delete as ad

def get_students_free_time(studentID):
    rows = ad.execute_query(f"SELECT SUBJECT_Subject_ID FROM STUDENT_HAS_SUBJECT WHERE STUDENT_Student_ID = {studentID}")
    student_free_time = set()
    for row in rows:
        student_free_time = set(student_free_time | get_schedule_time_dict()[row['SUBJECT_Subject_ID']])
    student_free_time = set(range(1, 41)) - student_free_time
    return student_free_time

def get_intersection(studentIDs):
    inter_ft = set(range(1, 41))
    for studentID in studentIDs:
        inter_ft = inter_ft & get_students_free_time(studentID)

    return len(inter_ft)
```

각 학생이 수강하고 있는 과목별로 시간표를 구성하여 공강 시간이 최대한 겹치는 조원들로 팀을 구성하였다.

SERVICE BY PYTHON

2.조별 과제 조 구성

```
import add_delete as ad
import random

student_rows = ad.execute_query("SELECT Student_ID FROM STUDENT")
subject_rows = ad.execute_query("SELECT Subject_ID FROM SUBJECT")
subject_list = []
for row in subject_rows: ...

student_subject = {}

for row in student_rows: ...
ad.execute_query("DELETE FROM STUDENT_HAS_SUBJECT")
for key in student_subject.keys():
    for subjectID in student_subject[key]:
        query = f'''
        INSERT INTO STUDENT_HAS_SUBJECT (STUDENT_Student_ID, SUBJECT_Subject_ID)
        VALUES({key}, '{subjectID}');
        '''
        #print(query)
        ad.execute_query(query)
```

```
team_list = []
assigned_student = set()

while len(intersection_list) != 0:
    for key in intersection_dict.keys():
        if len(set(key) & assigned_student) == 0 and intersection_dict[key] >= intersection_list[0]:
            team_list.append(key)
            for student in key:
                assigned_student.add(student)
            intersection_list.pop(0)

student_left = list(set(studentIDs) - assigned_student)

with open('Team_Created.csv', 'w', newline='') as csv_file:
    for team in team_list:
        csv_file.write(f'{team[0]},{team[1]}\n')
    for student in student_left:
        csv_file.write(f'남은 학생, {student}\n')
    messagebox.showinfo('팀 생성 완료', f"{len(student_left)}명의 학생이 남았습니다.\n남은 학생: {student}")

def on_submit():
    subject_ID = entry_subject_ID.get()
    query = f"SELECT STUDENT_Student_ID FROM STUDENT_HAS_SUBJECT WHERE SUBJECT_Subject_ID = '{subject_ID}'"
    rows = execute_query(query) # 가상의 함수로 대체

    studentIDs = [row['STUDENT_Student_ID'] for row in rows]

    create_team_csv(subject_ID, studentIDs)

# GUI 생성
root = tk.Tk()
```

개인의 공강시간을 구한 후 각 조합별로 공강 시간이 최대한 겹치는 경우를 구하여 CSV 파일로 정리해 조 구성 파일을 생성한다.