Databases Programming Project: **Final Report**

이름 (Name): 이강우

학과 (Department): 정보컴퓨터공학과

학번 (Student ID): 201824636

---

1. 프로젝트 개요 (Project topic)

사용한 언어 및 라이브러리: Python, psycopg2, random

프로젝트 개요:

CS 전공자의 취업 과정을 DB로 구현하였다. 기존에 부트캠프가 있었는데, 구현에 있어 기업과 유사하다고 생각하여 과감히 삭제하고, 피드백 받은 대로 강사의 역할을 확대하였다. 응시자, 기업, 강사의 3개의 ROLE이 있어 각자 접근할 수 있는 TABLE이 제한되고 강의의 Student Management System과 같이 서로 얽히고 설키는 상호작용을 구현한 프로젝트이고 다양한 sql문을 사용하였다.

프로젝트에서, 필요한 TABLE에 필요한 TUPLE이 없을 경우는 예외처리 되어 있어, 프로그램이 터지는 일 없이 매끄럽게 진행된다. 응시자는 12개의 옵션, 기업은 7개의 옵션, 강사는 8개의 옵션을 가지며 각각의 옵션들은 여러 테이블의 상호작용으로 구현되었다.

2. 사용자 (역할) (Users / Roles)

Solver: 응시자로서 응시자 등록, 문제를 풀거나, 응시자 간 질의응답, 모집공고 지원, 합격자 조회, 대회 지원, 대회 결과 조회, 강의 수강, 학원 등록 수행한다.

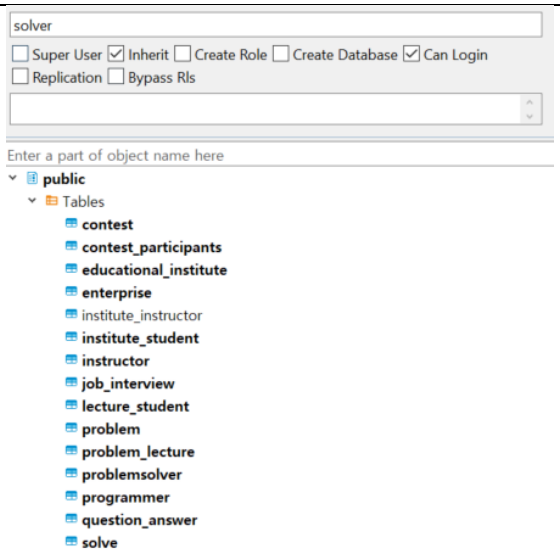

Figure 1 GRANT ON으로 Solver의 접근이 허용된 Table

Enterprise: 기업으로서 기업 등록, 모집공고를 등록하거나, 삭제, 합격여부 발표, 대회 개최, 대회 결과 발표를 수행한다.

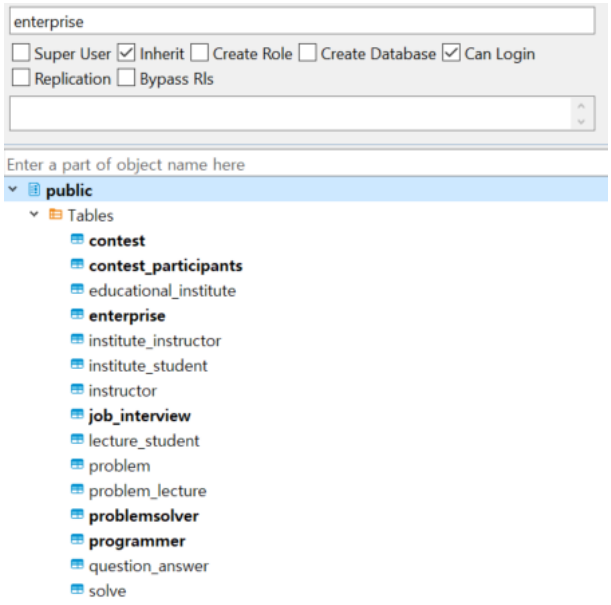

Figure 2 GRANT ON으로 Enterprise의 접근이 허용된 Table

Instructor: 강사로서 강사 등록, 강의 등록, 수강 학생 조회, 학원 창립, 학원 취원, 학원 조회, 문제 만들기를 수행한다.

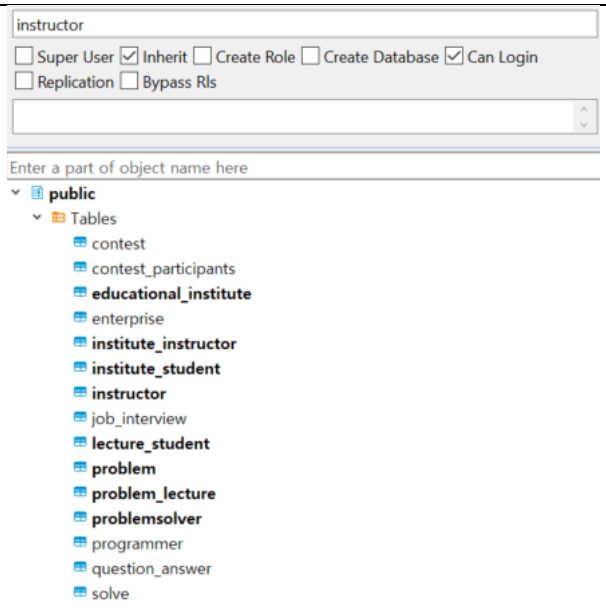

Figure 3 GRANT ON으로 Instructor의 접근이 허용된 Table

3. 기능 (Functions)

[공통 쿼리]

```
print("======================")
print("ROLE 을 선택하세요")
print("[1: solver  2: enterprise  3: instructor]")
print("======================")
n= int(input())
if n == 1:
    cursor.execute("set role solver")
    print("Solver 로 로그인 하셨습니다.")
    solver()
elif n == 2:
    cursor.execute("set role enterprise")
    print("Enterprise 로 로그인 하셨습니다.")
    recruitment(cursor)
elif n == 3:
    cursor.execute("set role instructor")
    print("Instructor 로 로그인 하셨습니다.")
```

```
    instructor()
cursor.execute("reset role")    # role 초기화
```

[3개의 ROLE을 나누기 위한 쿼리]

```
def get_id(cursor,table,column):
    query = f"SELECT MAX({column}) FROM {table}"
    cursor.execute(query)
    max_id = cursor.fetchone()[0]
    if max_id is None:
        max_id = 0
    return max_id
```

[테이블에 새로운 데이터를 추가할 때 필요한 최대 ID를 가져올 때 사용 쿼리]

[응시자 쿼리]

```
cursor.execute("insert into problemsolver
(sid,name,point,gpa,school) values(%s, %s,
0, %s, %s)",(id,name,gpa,school))
```

[새로운 응시자를 추가하는 쿼리]

```
 cursor.execute("DELETE FROM problemsolver where sid = %s",(id,))
```

[응시자를 삭제하는 쿼리]

```
ssid = get_id(cursor,"solve","ssid")+1;
success = random.choice([0, 1])
cursor.execute("insert into solve (ssid,sid,pid,success)
values(%s, %s, %s, %s)",(ssid, sid, pid, success))
```

[응시자가 문제를 푸는 쿼리]

```
cursor.execute("select sid, point, name, pid,tier,title,success
from solve natural join problemsolver natural join problem where
ssid = %s",(ssid,))
```

[응시자가 푼 문제의 정보를 가져오는 쿼리]

```
score ={'BRONZE':10,
'SILVER':20,'GOLD':30,'PLATINUM':40,'DIAMOND':50}
```

[딕셔너리로 저장되어 있는 문제의 점수]

```
cursor.execute("update problemsolver set point = %s where sid
= %s",(point+score[tier],sid))
```

[문제를 풀어냈다면 응시자의 코딩 점수 업데이트하는 쿼리]

```
cursor.execute("select sid,pid,name from solve natural join
problemsolver except select sid,pid,name from solve natural join
problemsolver where success = 1")
# success 하지 못한 sid,pid
```

[응시자가 여러 번의 시도에도 풀지 못한 문제를 except를 이용해 가져오는 쿼리]

```
post_id = get_id(cursor,"question_answer","post_id")+1
cursor.execute("insert into question_answer (post_id,qid,pid)
values(%s, %s, %s)", (post_id, sid, pid))
```

[응시자가 질문을 질의 응답 게시판에 등록 하는 쿼리]

```
cursor.execute("select post_id,pid,name from question_answer q join
problemsolver p on q.qid = p.sid  where resolve is NULL order by
post_id")
#해결하지 못한 문제의 post_id, pid, name 을 JOIN 을 이용해 가져온다.
```

[해결하지 못한 문제를 가져오는 쿼리]

```
cursor.execute("select distinct(sid),name from solve natural join
problemsolver where pid = (select pid from question_answer where
post_id = %s) and success = 1 order by sid",(post_id,))
```

[서브 쿼리를 이용해 해당 문제를 풀어낸 응시자가 있다면 가져오는 쿼리]

```
resolve = random.choice([0, 1])
cursor.execute("update question_answer set rid = %s,resolve = %s
where post_id = %s",(rid,resolve,post_id))
```

[랜덤으로 질의자가 도움을 받고 풀어냈다면 업데이트하는 쿼리]

```
cursor.execute(“select post_id,p.name as qname,p2.name as
rname ,pid,title,resolve from question_answer q left join
problemsolver p on q.qid = p.sid left join problemsolver p2 on
q.rid = p2.sid natural join problem order by post_id”)
#post_id, qname, rname, pid, title, resolve
```

[질의 응답 게시판을 구현하기 위해 겹치는 속성으로 join하는 쿼리]

```
cursor.execute("select eid,e_name from enterprise e where cut is
not null order by eid")
```

[모집공고가 등록(커트가 null이 아닌)된 공고를 가져오는 쿼리]

```
cursor.execute("select iid from instructor i order by iid")
results = cursor.fetchall()
if results:
    iids = [int(result[0]) for result in results]   #iid 리스트로
받아온다.
    iid = random.choice(iids)
    # 면접관중 한명 랜덤으로 뽑음
    cursor.execute("insert into job_interview
(sid,eid,iid,position)
values(%s, %s, %s, %s)",(sid,eid,iid,position))
```

[면접을 시행하는 테이블에 일단 넣어놓고 나중에 기업에서 발표할 쿼리]

```
cursor.execute("select name, e_name, position ,salary  from
programmer natural join problemsolver p natural join enterprise e")
```

[최종 면접에 붙은 응시자의 정보를 JOIN을 이용해 조회하는 쿼리]

```
cursor.execute("select contest_id , contest_name , prize, e_name
from contest c natural join enterprise e order by contest_id")
```

[현재 콘테스트의 정보를 JOIN으로 모두 가져오는 쿼리]

```
cursor.execute(“insert into contest_participants
(contest_id ,sid,rank) values(%s, %s, null)”,(contest_id,sid))
```

[참가자들을 rank를 NULL로 일단 받는다. 추후에 기업에서 rank를 UPDATE하는 쿼리]

```
cursor.execute("select distinct(contest_id)as id,
contest_name,e_name from contest_participants natural join contest
natural join enterprise where rank is not null")
#결과가 발표된 대회 가져온다
```

[rank를 보고 발표된 대회를 가져오는 쿼리]

```
cursor.execute("select rank,name from contest_participants natural
join problemsolver where contest_id = %s order by
rank",(contest_id,))
```

[rank로 정렬된 대회의 등수를 조회하는 쿼리]

```
cursor.execute("select iid, i_name, pid, title from problem_lecture
natural join instructor natural join problem")
```

[강사가 찍어놓은 강의 정보를 조회하는 쿼리]

```
cursor.execute("select distinct(sid), name from solve natural join
problemsolver p  where pid = %s and success = 0",(pid,))
```

[문제를 틀린 학생이 있다면 가져오는 쿼리]

```
cursor.execute("select eiid, name, i_name from
educational_institute ei join instructor i on i.iid = ei.ceo_id")
# 학원 정보 출력
```

[학원의 정보를 출력하는 쿼리]

```
cursor.execute("select sid,name from problemsolver s where sid not
in (select sid from institute_student where eiid = %s) order by
sid",(eiid,))
# 현재 학원에 다니지 않는 학생 선택
```

[학원에 다니지 않는 학생을 출력하는 쿼리]

```
cursor.execute("insert into institute_student(eiid,sid)
values(%s,%s)", (eiid, sid))
```

[학원에 다니는 학생 테이블에 삽입하는 쿼리]

[기업 쿼리]

```
cursor.execute("insert into enterprise (eid,e_name,scale,place)
values(%s, %s, %s, %s)", (eid, name, scale,place))
```

[새로운 기업을 추가하는 쿼리]

```
cursor.execute("select eid,e_name,scale,place from enterprise e
where cut is null order by eid")
```

[모집공고를 올리지 않은 기업 조회하는 쿼리]

```
cursor.execute("update enterprise set cut = %s where eid
= %s",(cut,eid))
```

[코딩 테스트 커트를 UPDATE해 모집공고를 올리는 쿼리]

```
cursor.execute("update enterprise set cut = NULL where eid = %s",
(eid,))
```

[커트를 없애 모집공고를 삭제하는 쿼리]

```
cursor.execute("select sid,eid,position,point from job_interview ji
natural join problemsolver p natural join enterprise e where ji.eid
=%s and cut <= point or sid in (select sid from contest natural
join contest_participants cp where rank <=3 and eid =
ji.eid)",(eid,))
#기업에서 주최한 대회에서 3등안에 들거나 OR 기업의 컷을 넘긴
지원자를 뽑는다.
cursor.execute("insert into programmer (sid, eid, position, salary)
values(%s, %s, %s, %s)",(sid,eid,position,point*10000))
#코테 점수 * 10000원
```

[서브쿼리와 조인을 이용해 대회에서 메달권 안에 들거나, 코테 점수를 넘기면 취업 테이블에 삽입하는 쿼리]

```
contest_id = get_id(cursor, "contest", "contest_id") + 1
cursor.execute("insert into contest (contest_id, eid,
contest_name,prize) values(%s, %s, %s, %s)",(contest_id, eid,
name,prize))
```

[대회를 생성하는 쿼리]

```
cursor.execute("select distinct(contest_id),contest_name from
contest_participants cp natural join contest where rank is null
order by contest_id")
# 발표가 나지 않은 콘테스트 출력 rank 가 null 인 distinct 한
contest_id
```

[발표가 나지 않는 대회를 가져오는 쿼리]

```
cursor.execute("select sid from contest_participants natural join
problemsolver p where contest_id = %s order by point desc,gpa
desc",(contest_id,))
# 코테 점수/ 그후 학점순으로 desc (내림차순) 으로 정렬하고 그
순서대로 sid 를 가져온다.
rank = 1
for row in rows:
    sid = row
    cursor.execute("update contest_participants set rank = %s where
sid = %s",(rank,sid))
    rank += 1
```

[발표가 나지 않은 대회에 참가한 참가자를 코테, 학점순으로 내림차순으로 정렬하여 rank UPDATE 하는 쿼리]

[강사 쿼리]

```
cursor.execute("insert into instructor (iid,i_name,point,salary)
values(%s, %s, %s, %s)",(iid, name, point, salary))
```

[새로운 강사를 추가하는 쿼리]

```
cursor.execute("select pid, title from problem order by pid ")
cursor.execute("select iid, i_name from instructor order by iid")
cursor.execute("insert into problem_lecture (iid,pid)
values(%s, %s)",(iid,pid))
```

[강사가 문제를 찍은 강의를 등록하는 쿼리].

```
cursor.execute("select i_name,name,pid,title from lecture_student
ls natural join problemsolver natural join problem join instructor
i on ls.iid = i.iid")
#강사의 수업을 수강하는 학생과 연결
```

[강사와 수강학생을 JOIN으로 연결하여 출력하는 쿼리]

```
cursor.execute("select iid,i_name from instructor i where iid not
in (select distinct(ceo_id) from educational_institute) order by
iid")
#ceo 가 아닌 강사 조회
cursor.execute("insert into educational_institute(eiid,name,ceo_id)
values(%s,%s, %s)", (eiid,name,iid))
```

[학원을 창립할 때, 대표강사를 지정하여 창립하는 쿼리]

```
cursor.execute("select eiid, name, i_name from
educational_institute ei join instructor i on i.iid = ei.ceo_id")
#학원 정보 출력
cursor.execute("select iid, i_name from instructor i where iid not
in (select iid from institute_instructor where eiid = %s)",(eiid,))
#현재 학원에 근무하지 않는 강사 선택
cursor.execute("insert into institute_instructor(eiid,iid)
values(%s,%s)", (eiid, iid))
```

[학원에 근무할 강사를 등록하는 쿼리]

```
cursor.execute("select ei.eiid, name,count(distinct ii.iid) as
instructorcnt, count(distinct id.sid) as solvercnt from
educational_institute ei left join institute_instructor ii on
ei.eiid = ii.eiid left join institute_student id on ei.eiid =
```

```
id.eiid group by ei.eiid order by eiid ")
#학원의 이름,강사 수, 응시자 수 반환
```

[JOIN을 이용해 학원의 이름, 강사 수, 응시자 수를 GROUP BY로 학원에 이름으로 그룹지어 출력하는 쿼리]

```
cursor.execute("insert into
problem(pid,title,algorithm,tier,tier_num) values(%s,%s,%s,%s,%s)",
(pid, title,algorithm,tier,level))
```

[새로운 문제를 등록하는 쿼리]

4. 데이터베이스 스키마 및 다이어그램 (Database schema / Schema diagram)

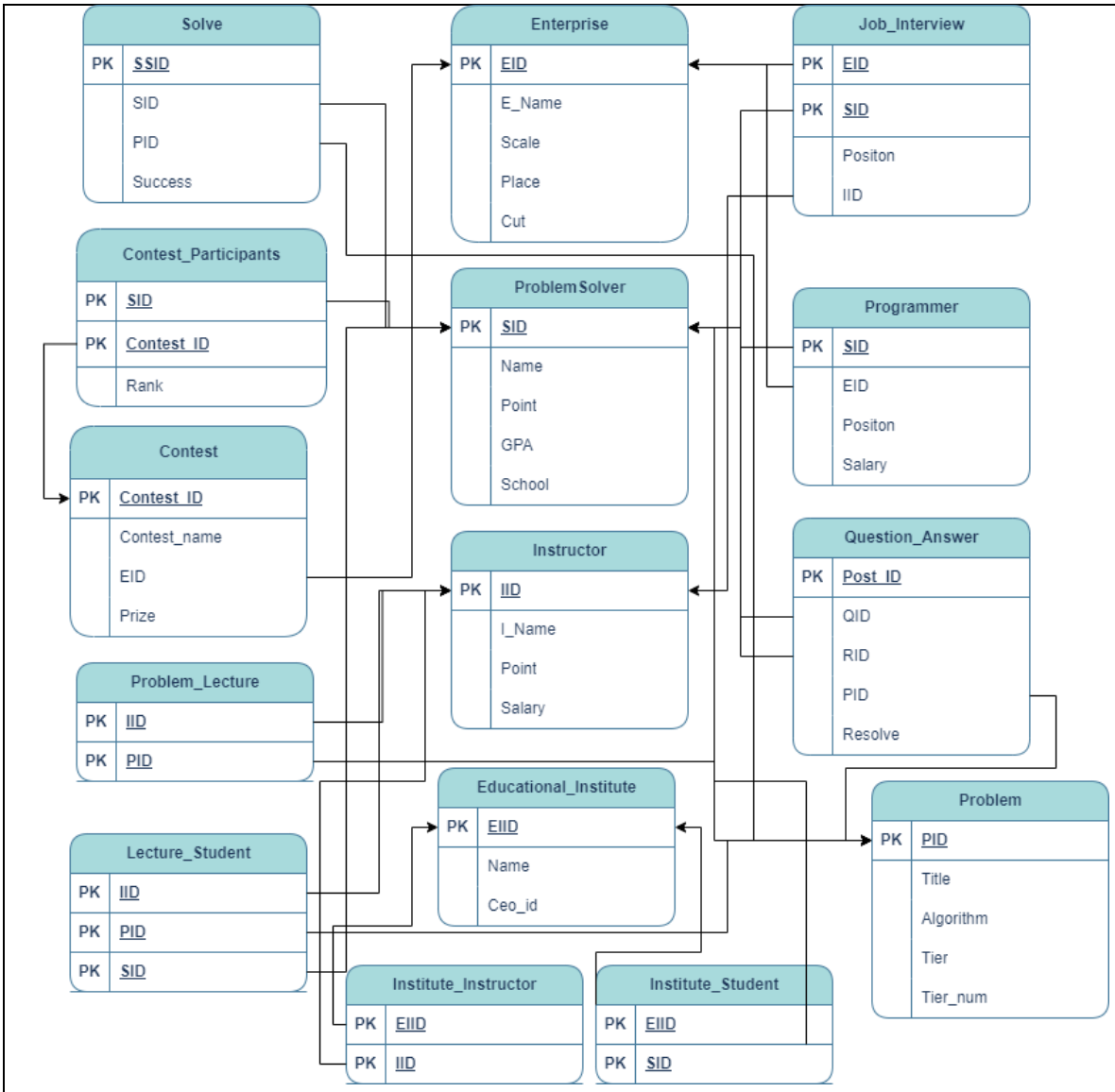

Figure 4 데이터베이스 다이어그램 (최종)

[테이블 쿼리문]

```
CREATE TABLE ProblemSolver (
        sid INT PRIMARY KEY,
        name VARCHAR(30) NOT NULL,
        point INT DEFAULT 0 NOT NULL,
        gpa NUMERIC(2,1) CHECK(gpa >= 0.0 AND gpa <=4.5) NOT NULL,
        school VARCHAR(30) NOT NULL
);

CREATE TABLE Enterprise (
        eid int primary key,
        e_name varchar(30) NOT NULL,
        scale varchar(30) NOT NULL,
        place varchar(30) NOT NULL,
        cut INT DEFAULT NULL
```

```
);
CREATE TABLE Instructor (
        iid int primary key,
        i_name varchar(30) NOT NULL,
        point INT DEFAULT 0,
        salary INT DEFAULT 2000000
);
CREATE TABLE Problem (
        pid int primary key,
        title varchar(30) NOT NULL,
        algorithm varchar(30) NOT NULL,
        tier varchar(30) NOT NULL,
        tier_num int CHECK (tier_num >= 1 AND tier_num <= 5) NOT NULL
);
CREATE TABLE Problem_Lecture (
        iid int,
        pid int,
        PRIMARY KEY (iid, pid),
        FOREIGN KEY (pid) REFERENCES Problem(pid) ON DELETE CASCADE,
        FOREIGN KEY (iid) REFERENCES Instructor(iid) ON DELETE CASCADE
);
CREATE TABLE Lecture_Student(
        iid int,
        pid int,
        sid int,
        PRIMARY KEY (iid, pid,sid),
        FOREIGN KEY (sid) REFERENCES ProblemSolver(sid) ON DELETE CASCADE,
        FOREIGN KEY (pid) REFERENCES Problem(pid) ON DELETE CASCADE,
        FOREIGN KEY (iid) REFERENCES Instructor(iid) ON DELETE CASCADE
);
CREATE TABLE Solve (
        ssid int primary key,
        sid int,
        pid int,
        success int CHECK(success >= 0 AND success <= 1),
        FOREIGN KEY (sid) REFERENCES ProblemSolver(sid) ON DELETE CASCADE,
        FOREIGN KEY (pid) REFERENCES Problem(pid) ON DELETE CASCADE
);
CREATE TABLE Contest (
        contest_id int primary key,
        eid int,
        contest_name varchar(30) NOT NULL,
        prize int NOT NULL,
        FOREIGN KEY (eid) REFERENCES ProblemSolver(sid) ON DELETE CASCADE
);
CREATE TABLE Contest_Participants (
        contest_id int,
        sid int,
        rank int,
```

```
        PRIMARY KEY (contest_id, sid),
        FOREIGN KEY (sid) REFERENCES ProblemSolver(sid) ON DELETE CASCADE,
        FOREIGN KEY (contest_id) REFERENCES Contest(contest_id) ON DELETE CASCADE
);
CREATE TABLE Question_Answer(
        post_id int primary key,
        qid int,
        rid int,
        pid int,
        resolve int CHECK(resolve >= 0 AND resolve <= 1),
        FOREIGN KEY (qid) REFERENCES ProblemSolver(sid) ON DELETE CASCADE,
        FOREIGN KEY (rid) REFERENCES ProblemSolver(sid) ON DELETE CASCADE,
        FOREIGN KEY (pid) REFERENCES Problem(pid) ON DELETE CASCADE
);

CREATE TABLE Job_Interview(
        sid int,
        eid int,
        iid int NOT NULL,
        position varchar(30) NOT NULL,
        PRIMARY KEY (sid, eid),
        FOREIGN KEY (sid) REFERENCES ProblemSolver(sid) ON DELETE CASCADE,
        FOREIGN KEY (eid) REFERENCES Enterprise(eid) ON DELETE CASCADE,
        FOREIGN KEY (iid) REFERENCES Instructor(iid) ON DELETE CASCADE
);
CREATE TABLE Programmer (
        sid int primary key,
        eid int,
        position varchar(30) NOT NULL,
        salary int DEFAULT 2000000,
        FOREIGN KEY (sid) REFERENCES ProblemSolver(sid) ON DELETE CASCADE,
        FOREIGN KEY (eid) REFERENCES Enterprise(eid) ON DELETE SET NULL
);
CREATE TABLE Educational_Institute (
        eiid int primary key,
        name varchar(30) NOT NULL,
        ceo_id int DEFAULT NULL,
        FOREIGN KEY (ceo_id) REFERENCES Instructor(iid) ON DELETE CASCADE
);
CREATE TABLE Institute_Instructor (
        eiid int,
        iid int,
        PRIMARY KEY (eiid, iid),
        FOREIGN KEY (iid) REFERENCES Instructor(iid) ON DELETE CASCADE,
        FOREIGN KEY (eiid) REFERENCES Educational_Institute(eiid) ON DELETE CASCADE
);
CREATE TABLE Institute_Student (
        eiid int,
        sid int,
```

```
        PRIMARY KEY (eiid, sid),
        FOREIGN KEY (sid) REFERENCES ProblemSolver(sid) ON DELETE CASCADE,
        FOREIGN KEY (eiid) REFERENCES Educational_Institute(eiid) ON DELETE CASCADE
);
```

5. (팀 프로젝트인 경우만 해당) 팀원의 역할 배분

※ 각 팀원이 수행한 업무를 기재해주세요.