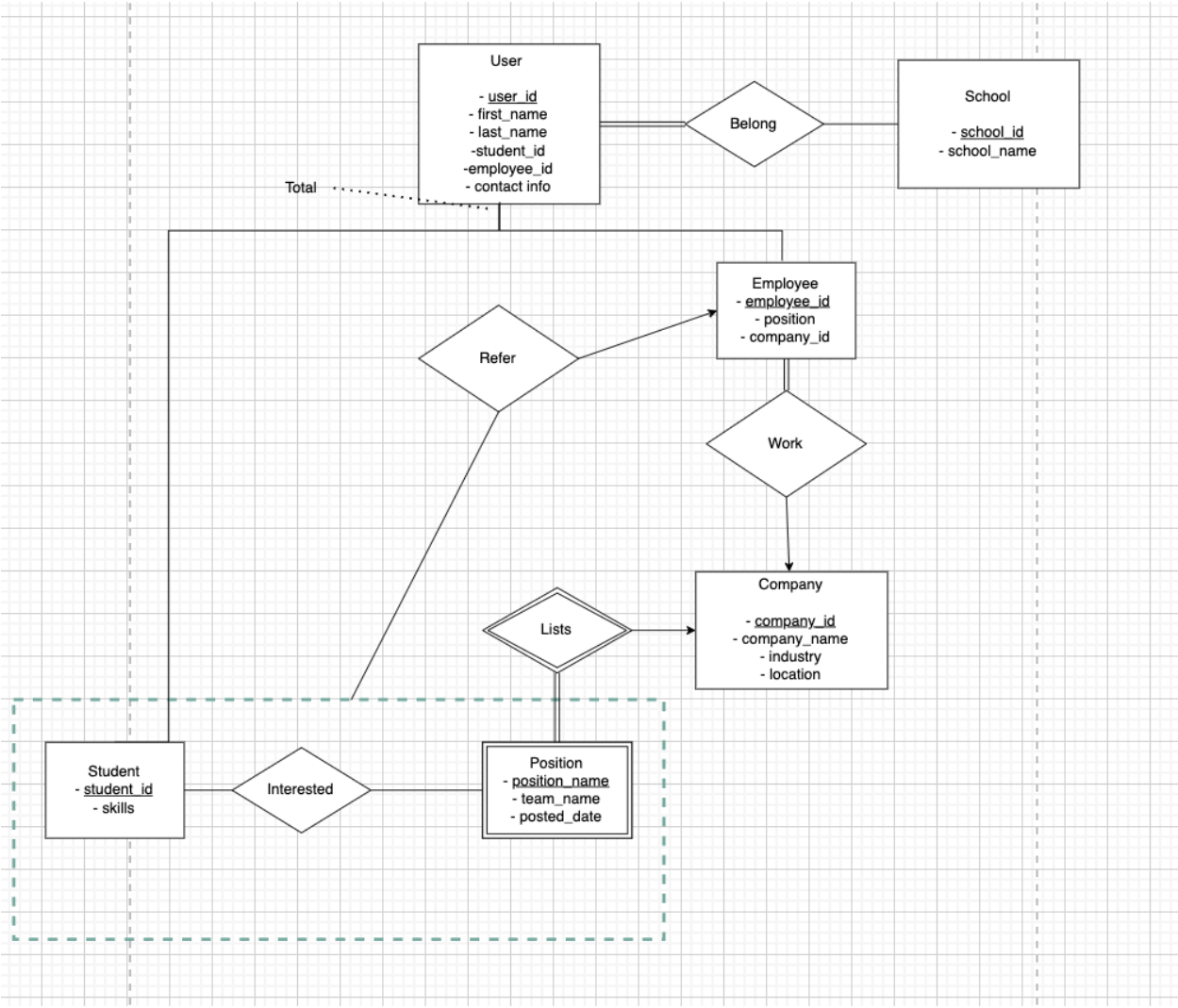
Team: Gyung Hyun Je (gj2353), Shreyans Kothari (sk4819)
Topic: Job referral web application
Date: October 18th
UNI for database: sk4819

Updated ER Diagram



CREATE statements

```
CREATE TABLE users (
user_id varchar PRIMARY KEY,
first_name varchar,
last_name varchar,
school_id int NOT NULL REFERENCES school,
student_id int REFERENCES student,
employee_id int REFERENCES employee,
contact_info text,
CHECK((NOT(student_id IS NULL AND employee_id IS NULL)) AND (NOT(student_id IS NOT
NULL AND employee_id IS NOT NULL )))
);

CREATE TABLE student (
student_id int PRIMARY KEY,
skills text,
user_id varchar UNIQUE NOT NULL REFERENCES user
);

CREATE TABLE employee (
employee_id int PRIMARY KEY,
company_id int REFERENCES company,
position varchar,
user_id varchar UNIQUE NOT NULL REFERENCES user
);

CREATE TABLE school(
school_id int PRIMARY KEY,
school_name text
);

CREATE TABLE Company (
    company_id serial PRIMARY KEY,
    company_name varchar UNIQUE NOT NULL,
    industry varchar(600),
    location varchar(600),
    description varchar(1000)
);
```

```
CREATE TABLE Position (
   PRIMARY KEY (position_title, company_id),
   position_title varchar(300),
   company_id int references Company ON DELETE CASCADE,
   team_name varchar(300),
   posted_date date
);

CREATE TABLE Student_interest(
   PRIMARY KEY (student_id, position_title, company_id),
   student_id int NOT NULL REFERENCES Student,
   position_title varchar,
   company_id int,
   FOREIGN KEY (position_title, company_id) REFERENCES Position(position_title,
company_id)
);

CREATE TABLE Refer(
PRIMARY KEY (student_id, position_title, company_id),
employee_id int NOT NULL REFERENCES Employee,
student_id int,
position_title varchar,
company_id int,
FOREIGN KEY (student_id, position_title, company_id) REFERENCES
Student_interest(student_id, position_title, company_id)
);
```

Three interesting queries

1. *Interested student:potential-referrer ratio for a given company*

```
SELECT a.company_id,
      c.company_name,
      count(distinct student_id)::float/count(distinct employee_id) as
interest_referrer_ratio
FROM Student_interest a
INNER JOIN Employee b on a.company_id = b.company_id
INNER JOIN Company c on b.company_id = c.company_id
group by 1,2;
```

2. *Count of all students who got referrals for every position they indicated as interested*

```
SELECT COUNT(DISTINCT a.student_id)
FROM Student_interest a
WHERE NOT EXISTS  (
   (SELECT a.student_id, a.position_title, a.company_id
   FROM Refer d, Student_interest
   )
   EXCEPT
   (select s.student_id, s.position_title, s.company_id
    FROM Student_interest s, Refer r
    WHERE s.student_id = r.student_id and s.position_title = r.position_title and
s.company_id = r.company_id
   )
);
```

3. *Returns school name, employee_id, and student_id if the referring employee is an alumni of student referee's school (i.e., if the referring employee graduated from the school the student is currently studying at):*

```
SELECT tmp.student_id, tmp.employee_id, school.school_name AS school
FROM (SELECT * FROM
                (SELECT s.student_id, u.school_id student_school
                        FROM student s, users u WHERE s.student_id = u.student_id) AS
                        stdnt,
                (SELECT e1.employee_id, u1.school_id emp_school
                        FROM employee e1, users u1 WHERE e1.employee_id =
                        u1.employee_id) AS emp
                WHERE student_school = emp_school) AS tmp
JOIN (SELECT employee_id, student_id FROM refer) AS ref
        ON tmp.student_id = ref.student_id AND tmp.employee_id = ref.employee_id
LEFT JOIN school
        ON tmp.student_school = school.school_id;
```

Description of changes made

ER diagram

- Underlined primary keys, instead of explicitly writing “primary key”
- Removed aggregation: Before, there was an aggregation among Student, Interested, Position, List, Companies. Now the aggregation is among Student, Interested, Position to express “student is interested in a specific position (listed by a company)”.
- Revised the “Refer” relation: With revision above in the aggregation, the employee now could refer many different students interested in a position. And each student interested in a position could get referred by at most one employee.

Create statement

- Revised Work_at primary key: since it should express “employee work at exactly one company”, changed the PRIMARY KEY (employee, company_id) to only PRIMARY KEY (employee).
- Removed Work_at (combined with employee table) and User_belongs_to_school (combined with users table) tables because redundant and can simply use User_id and Employee tables to enforce participation constraint

Implementations that were kept

Comment for revision: “Each school/company should have at least one user/employee”. But each school/company does not have to have at least one user/employee, since those would be predefined values. The scope of our application is limited, and thus it is only offered to employees at companies listed in the Company table (a list of top NYC companies), and users who went to one of the schools listed in the School table (a list of Columbia University schools)