INHA UNIVERSITY

ICE4016 데이터베이스 설계

<제목>

SQL, Express

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.
2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.
3. 나는 보고서의 내용을 조작하지 않겠습니다.
4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.
5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2022년 10월 1일

학부 정보통신공학
학년 3
성명 김민겸
학번 12201863

1. 개요

- 'Newbenefits' 프로젝트에 참여하는 모든 사원의 급여를 10% 올린 경우의 급여를 제시하라. (Fname, Lname, Increased_sal)
- 급여가 30,000달러에서 40,000달러 사이에 있는 5번 부서의 모든 사원을 검색하라.
- 모든 사원을 1. 급여(높은 순서) 2. 생년월일(나이가 많은 순서)을 제시하라.
- 상사가 없는 모든 사원의 이름(Fname, Lname)을 검색하라.
- 부양가족의 성별(Sex)과 사원의 성별이 같은 사원의 이름(Fname, Lname)을 검색하라.
- 부양가족이 없는 종업원들의 이름(Fname, Lname)을 검색하라.
- 프로젝트 번호 1,2,3에서 일하는 사원의 주민등록번호(Essn)를 검색하라.
- 사원의 급여의 합, 최고 급여, 최저 급여, 평균 급여를 구하라.

- 회사의 총 사원수를 제시하라.
- 각 부서에서 근무하는 사원의 수를 검색하라. (부서이름과 소속 사원수를 제시)
- 각 부서에 대해서 부서이름, 부서에 소속된 사원의 수와 최고급여와 평균 급여를 구하라.
- 프로젝트에 대해서 프로젝트 번호, 프로젝트 이름, 그 프로젝트에서 근무하는 사원들의 수를 검색하라.
- 세 명 이상의 사원이 근무하는 프로젝트에 대해서 프로젝트 번호, 프로젝트 이름, 그 프로젝트에서 근무하는 사원들의 수를 검색하라.
- 프로젝트에 대해서 프로젝트 번호, 프로젝트 이름, 5번 부서에 속하면서 프로젝트에서 근무하는 사원의 수를 검색하라.
- 3명 이상의 사원이 근무하는 각 부서에 대해서 부서 번호와 40,000달러가 넘는 급여를 받는 사원의 ssn 및 salary를 검색하라.

- STEP 1 : 4주차 과제에서 만든 Inha 데이터베이스 확장
  - Inha 데이터베이스에 Club, 테이블 추가
  - 추가된 테이블을 참고하여 기존 DB의 관계(Relation)를 재구성 수행

Table Name	Attribute	Attribute	Attribute	Attribute	Attribute	Attribute	Attribute
Club	Id	Name					
Employee	Id	Name	Position				

- STEP 2 : 확장한 Inha Db를 Express와 연동
  - Page 13을 참고하여 sql.js 파일에서 insert문 실행
  - Insert한 결과를 sql.js 파일에서 select문으로 조회한 결과를 캡처

2. 상세 설계 내용

1. SQL문 작성하기

(1) 'Newbenefits' 프로젝트에 참여하는 모든 사원의 급여를 10% 올린 경우의 급여를 제시하라. (Fname, Lname, Increased_sal).

```
select fname, lname, salary * 1.1 as Increased_sal
        from employee, project, works_on
        where project.Pname="Newbenefits" and works_on.Pno=project.Pnumber and
       works_on.Essn=employee.Ssn;
```

```
mysql> select fname, lname, salary * 1.1 as Increased_sal from employee, project, works_on where project.Pname="Newbe
nefits" and works_on.Pno=project.Pnumber and works_on.Essn=employee.Ssn;
+----------+---------+---------------+
| fname    | lname   | Increased_sal |
+----------+---------+---------------+
| Jennifer | Wallace |       47300.0 |
| Ahmad    | Jabbar  |       27500.0 |
| Alicia   | Zelaya  |       27500.0 |
+----------+---------+---------------+
3 rows in set (0.01 sec)
```

(2) 급여가 30,000달러에서 40,000달러 사이에 있는 5번 부서의 모든 사원을 검색하라.

```
select * from employee where salary >= 30000 and salary <= 40000;
```

```
mysql> select * from employee where salary >= 30000 and salary <= 40000;
+----------+-------+---------+-----------+------------+------------------------+------+--------+-----------+-----+
| Fname    | Minit | Lname   | Ssn       | Bdate      | Address                | Sex  | Salary | Super_ssn | Dno |
+----------+-------+---------+-----------+------------+------------------------+------+--------+-----------+-----+
| John     | B     | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston TX | M    |  30000 | 333445555 |   5 |
| Franklin | T     | Wong    | 333445555 | 1965-12-08 | 638 Voss, Houston TX    | M    |  40000 | 888665555 |   5 |
| Ramesh   | K     | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX | M    |  38000 | 333445555 |   5 |
+----------+-------+---------+-----------+------------+------------------------+------+--------+-----------+-----+
3 rows in set (0.01 sec)
```

(3) 모든 사원을 1. 급여(높은 순서) 2. 생년월일(나이가 많은 순서)을 제시하라.

```
select * from employee order by salary desc;
select * from employee order by bdate asc;
```

```
mysql> select * from employee order by salary desc;
+----------+-------+----------+-----------+------------+--------------------------+------+--------+-----------+-----+
| Fname    | Minit | Lname    | Ssn       | Bdate      | Address                  | Sex  | Salary | Super_ssn | Dno |
+----------+-------+----------+-----------+------------+--------------------------+------+--------+-----------+-----+
| James    | E     | Borg     | 888665555 | 1937-11-10 | 450 Stone, Houston TX    | M    |  55000 | NULL      |   1 |
| Jennifer | S     | Wallace  | 987654321 | 1941-06-20 | 291 Berry, Bellaire TX   | F    |  43000 | 888665555 |   4 |
| Franklin | T     | Wong     | 333445555 | 1965-12-08 | 638 Voss, Houston TX     | M    |  40000 | 888665555 |   5 |
| Ramesh   | K     | Narayan  | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX  | M    |  38000 | 333445555 |   5 |
| John     | B     | Smith    | 123456789 | 1965-01-09 | 731 Fondren, Houston TX  | M    |  30000 | 333445555 |   5 |
| Joyce    | A     | English  | 453453453 | 1972-07-31 | 5631 Rice, Houston TX    | F    |  25000 | 333445555 |   5 |
| Ahmad    | V     | Jabbar   | 987987987 | 1969-03-29 | 980 Dallas, Houston TX   | M    |  25000 | 987654321 |   4 |
| Alicia   | J     | Zelaya   | 999887777 | 1968-01-19 | 3321 Castle, Spring TX   | F    |  25000 | 987654321 |   4 |
+----------+-------+----------+-----------+------------+--------------------------+------+--------+-----------+-----+
8 rows in set (0.00 sec)

mysql> select * from employee order by bdate asc;
+----------+-------+----------+-----------+------------+--------------------------+------+--------+-----------+-----+
| Fname    | Minit | Lname    | Ssn       | Bdate      | Address                  | Sex  | Salary | Super_ssn | Dno |
+----------+-------+----------+-----------+------------+--------------------------+------+--------+-----------+-----+
| James    | E     | Borg     | 888665555 | 1937-11-10 | 450 Stone, Houston TX    | M    |  55000 | NULL      |   1 |
| Jennifer | S     | Wallace  | 987654321 | 1941-06-20 | 291 Berry, Bellaire TX   | F    |  43000 | 888665555 |   4 |
| Ramesh   | K     | Narayan  | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX  | M    |  38000 | 333445555 |   5 |
| John     | B     | Smith    | 123456789 | 1965-01-09 | 731 Fondren, Houston TX  | M    |  30000 | 333445555 |   5 |
| Franklin | T     | Wong     | 333445555 | 1965-12-08 | 638 Voss, Houston TX     | M    |  40000 | 888665555 |   5 |
| Alicia   | J     | Zelaya   | 999887777 | 1968-01-19 | 3321 Castle, Spring TX   | F    |  25000 | 987654321 |   4 |
| Ahmad    | V     | Jabbar   | 987987987 | 1969-03-29 | 980 Dallas, Houston TX   | M    |  25000 | 987654321 |   4 |
| Joyce    | A     | English  | 453453453 | 1972-07-31 | 5631 Rice, Houston TX    | F    |  25000 | 333445555 |   5 |
+----------+-------+----------+-----------+------------+--------------------------+------+--------+-----------+-----+
8 rows in set (0.00 sec)
```

(4) 상사가 없는 모든 사원의 이름(Fname, Lname)을 검색하라.

```
select fname, lname from employee where super_ssn is null;
```

```
mysql> select fname, lname from employee where super_ssn is null;
+-------+-------+
| fname | lname |
+-------+-------+
| James | Borg  |
+-------+-------+
1 row in set (0.00 sec)
```

(5) 부양가족의 성별(Sex)과 사원의 성별이 같은 사원의 이름(Fname, Lname) 을 검색하라.

```
select fname, lname
        from employee, dependent
        where dependent.sex=employee.sex and dependent.Essn=employee.ssn;
```

```
mysql> select fname, lname from employee, dependent where depende
nt.sex=employee.sex and dependent.Essn=employee.ssn;
+----------+-------+
| fname    | lname |
+----------+-------+
| John     | Smith |
| Franklin | Wong  |
+----------+-------+
2 rows in set (0.00 sec)
```

(6) 부양가족이 없는 종업원들의 이름(Fname, Lname)을 검색하라

```
select distinct fname, lname
        from employee, dependent
        where ssn NOT IN (select ssn from employee, dependent where
       employee.ssn=dependent.Essn);
```

```
mysql> select distinct fname, lname from employee, dependent where ssn NOT IN (sel
ect ssn from employee, dependent where employee.ssn=dependent.Essn);
+--------+---------+
| fname  | lname   |
+--------+---------+
| Joyce  | English |
| Ramesh | Narayan |
| James  | Borg    |
| Ahmad  | Jabbar  |
| Alicia | Zelaya  |
+--------+---------+
5 rows in set (0.00 sec)
```

(7) 프로젝트 번호 1,2,3에서 일하는 사원의 주민등록번호(Essn)를 검색하라.

```
select distinct essn from works_on where pno=1 or pno=2 or pno=3;
```

```
mysql> select distinct essn from works_on where pno=1 or pno=2 or pno=3;
+-----------+
| essn      |
+-----------+
| 123456789 |
| 453453453 |
| 333445555 |
| 666884444 |
+-----------+
4 rows in set (0.01 sec)
```

(8) 사원의 급여의 합, 최고 급여, 최저 급여, 평균 급여를 구하라

```
select sum(salary), max(salary), min(salary), avg(salary) from employee;
```

```
mysql> select sum(salary), max(salary), min(salary), avg(salary) from employee;
+-------------+-------------+-------------+-------------+
| sum(salary) | max(salary) | min(salary) | avg(salary) |
+-------------+-------------+-------------+-------------+
|      281000 |       55000 |       25000 |  35125.0000 |
+-------------+-------------+-------------+-------------+
1 row in set (0.01 sec)
```

(9) 회사의 총 사원수를 제시하라.

```
select count(*) from employee;
```

```
mysql> select count(*) from employee;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.01 sec)
```

(10) 각 부서에서 근무하는 사원의 수를 검색하라. (부서이름과 소속 사원수를 제시)

```
select department.Dname, count(employee.dno) as count
        from employee left join department
       on(department.dnumber=employee.dno) group by department.dnumber;
```

```
mysql> select department.Dname, count(employee.dno) as count from employee l
eft join department on (department.dnumber=employee.dno) group by department
.dnumber;
+----------------+-------+
| Dname          | count |
+----------------+-------+
| Headquarters   |     1 |
| Administration |     3 |
| Research       |     4 |
+----------------+-------+
3 rows in set (0.01 sec)
```

(11) 각 부서에 대해서 부서이름, 부서에 소속된 사원의 수와 최고급여와 평균 급여를 구하라.

```
select Dname, count(employee.dno) as count,
        max(employee.salary) as max, avg(employee.salary) as avg
        from employee left join department on (department.dnumber=employee.dno)
```

```
mysql> select Dname, count(employee.dno) as count, max(employee.salary) as max, avg(emplo
yee.salary) as avg from employee left join department on (department.dnumber=employee.dno
) group by department.dnumber;
+----------------+-------+-------+------------+
| Dname          | count | max   | avg        |
+----------------+-------+-------+------------+
| Research       |     4 | 40000 | 33250.0000 |
| Headquarters   |     1 | 55000 | 55000.0000 |
| Administration |     3 | 43000 | 31000.0000 |
+----------------+-------+-------+------------+
3 rows in set (0.00 sec)
```

(12) 프로젝트에 대해서 프로젝트 번호, 프로젝트 이름, 그 프로젝트에서 근무하는 사원들의 수를 검색하라.

```
select Pnumber, Pname, count(*) as count from project
        left join works_on on (works_on.Pno=project.Pnumber)
        group by works_on.Pno;
```

```
mysql> select Pnumber, Pname, count(*) as count from project left join works
_on on (works_on.Pno=project.Pnumber) group by works_on.Pno;
+---------+-----------------+-------+
| Pnumber | Pname           | count |
+---------+-----------------+-------+
|      10 | Computerization |     3 |
|      30 | Newbenefits     |     3 |
|       1 | ProductX        |     2 |
|       2 | ProductY        |     3 |
|       3 | ProductZ        |     2 |
|      20 | Reorganization  |     3 |
+---------+-----------------+-------+
```

```
mysql> select Pnumber ,Pname, count(*) as count from works_on, proj
ect where works_on.pno=project.pnumber group by works_on.Pno;
+---------+-----------------+-------+
| Pnumber | Pname           | count |
+---------+-----------------+-------+
|      10 | Computerization |     3 |
|      30 | Newbenefits     |     3 |
|       1 | ProductX        |     2 |
|       2 | ProductY        |     3 |
|       3 | ProductZ        |     2 |
|      20 | Reorganization  |     3 |
+---------+-----------------+-------+
6 rows in set (0.00 sec)
```

두 가지 방법으로 해결하였다.

(13) 세 명 이상의 사원이 근무하는 프로젝트에 대해서 프로젝트 번호, 프로젝트 이름, 그 프로젝트에서 근무하는 사원들의 수를 검색하라.

- 각 프로젝트에서 근무하는 사원의 수(COUNT)
- 사원이 세 명 이상인 것만 출력 (Where COUNT>=3)

```
select pnumber, pname, count(*) as count from works_on, project
        where works_on.pno=project.pnumber
        group by works_on.pno having count(*) >= 3;
```

```
mysql> select pnumber, pname, count(*) as count from works_on, project where works_on.pno=p
roject.pnumber group by works_on.pno having count(*) >= 3;
+---------+-----------------+-------+
| pnumber | pname           | count |
+---------+-----------------+-------+
|      10 | Computerization |     3 |
|      30 | Newbenefits     |     3 |
|       2 | ProductY        |     3 |
|      20 | Reorganization  |     3 |
+---------+-----------------+-------+
4 rows in set (0.00 sec)
```

이 부분에서 많이 헷갈렸는데, having이라는 키워드를 사용하여 해결할 수 있었다. 위에서 select문을 사용할 때 group by를 사용해서 결과를 출력하였는데, 여기서 조건을 정하기 위해 having을 추가하였다.

(14) 프로젝트에 대해서 프로젝트 번호, 프로젝트 이름, 5번 부서에 속하면서 프로젝트에서 근무하는 사원의 수를 검색하라.

- 5번 부서 사원은 총 4명

- 5번 부서에게 할당된 프로젝트는 1, 2, 3 총 3개

- 프로젝트 1, 2, 3에 참여하는 직원은 총 7명

즉 모든 프로젝트의 번호와 이름을 출력하되 각 프로젝트에서 근무하는 사원의 수는 5번 부서에 속하는 만큼만 출력하도록 한다.

```
select pnumber, pname, count(employee.dno) from project, works_on, employee
       where employee.dno=5 and project.pnumber=works_on.pno and
       works_on.essn=employee.ssn group by works_on.pno;
```

```
mysql> select pnumber, pname, count(employee.dno) from project, works_on, employee where employee.dno=5
 and project.pnumber=works_on.pno and works_on.essn=employee.ssn group by works_on.pno;
+---------+-----------------+---------------------+
| pnumber | pname           | count(employee.dno) |
+---------+-----------------+---------------------+
|       1 | ProductX        |                   2 |
|       2 | ProductY        |                   3 |
|       3 | ProductZ        |                   2 |
|      10 | Computerization |                   1 |
|      20 | Reorganization  |                   1 |
+---------+-----------------+---------------------+
5 rows in set (0.00 sec)
```

(15) 3명 이상의 사원이 근무하는 각 부서에 대해서 부서 번호와 40,000달러가 넘는 급여를 받는 사원의 ssn 및 salary를 검색하라

- 3명 이상의 사원이 있는 부서 총 2개의 부서

- 40,000 달러가 넘는 급여를 받는 사원 총 3명

- 4번 부서에 대해서 40,000달러가 넘는 급여를 받는 사원인 Jennifer의 ssn과 salary, 그리고 5번 부서에 대해서 40,000달러가 넘는 급여를 받는 사원인 Franklin의 ssn과 salary를 출력하면 된다.

먼저 3명 이상의 사원이 있는 부서는 4, 5번 부서이다. 이 부서에 속하는 사람을 거르기 위해 count와 inner join을 이용해서 각 부서의 총 사원 수를 구하는 실습을 진행하였다.

```
mysql> select dname, count(*) from department as d inner join empl
oyee as e on d.dnumber=e.dno group by d.dname;
+----------------+----------+
| dname          | count(*) |
+----------------+----------+
| Administration |        3 |
| Headquarters   |        1 |
| Research       |        4 |
+----------------+----------+
3 rows in set (0.00 sec)

mysql> select dname, count(*) from department as d inner join empl
oyee as e on d.dnumber=e.dno group by d.dname having count(*)>=3 ;

+----------------+----------+
| dname          | count(*) |
+----------------+----------+
| Administration |        3 |
| Research       |        4 |
+----------------+----------+
2 rows in set (0.00 sec)
```

이전과 동일하게 group by와 having을 함께 사용하여 원하는 결과를 출력하였다.

```
mysql>  select department.dnumber, employee.ssn, employee.salary from employee, department where depart
ment.dnumber=employee.dno and employee.salary>=40000;
+---------+-----------+--------+
| dnumber | ssn       | salary |
+---------+-----------+--------+
|       5 | 333445555 |  40000 |
|       1 | 888665555 |  55000 |
|       4 | 987654321 |  43000 |
+---------+-----------+--------+
3 rows in set (0.00 sec)
```

위와 같이 부서별로 40000달러가 넘는 Salary를 가지고 있는 사원을 출력하였고, 이제 각 부서마다 3명 이상의 부원이 있는지 확인하는 쿼리를 합하여 출력해보자.

```
mysql> select dnumber, ssn, salary, count(dno) from department as d inner join employee as e on d.dnumb
er=e.dno where e.salary>= 40000 group by dno;
+---------+-----------+--------+------------+
| dnumber | ssn       | salary | count(dno) |
+---------+-----------+--------+------------+
|       5 | 333445555 |  40000 |          1 |
|       1 | 888665555 |  55000 |          1 |
|       4 | 987654321 |  43000 |          1 |
+---------+-----------+--------+------------+
3 rows in set (0.00 sec)
```

위와 같은 방법으로 1차로 40000 이상의 salary를 받는 사람을 거르고, 2차로 count를 수행해 부서별 사원의 수를 계산하고자 하였더니 count값이 1차에서 걸러진 만큼 집계되는 것을 알 수 있다. Group by 와 inner join의 순서를 바꾸어 출력해보았다.

Group by와 inner join의 순서를 바꾸기 위해 서브 쿼리를 사용하고자 하였고, count(*)가 총 3 이상인 부서에 속하는 사람들의 리스트를 뽑아서 그 중에서 salary가 40000이상인 사람들을 걸러내었다.

```
mysql> select dno from employee group by dno having count(*) >= 3;
+-----+
| dno |
+-----+
|   4 |
|   5 |
+-----+
2 rows in set (0.00 sec)
```

3명 이상의 부원이 속하는 부서를 뽑고, 이 테이블을 조회하는 쿼리를 서브로 하여 아래 쿼리를 만들었다.

```
mysql> select fname, minit, lname, ssn, bdate, address, sex, salary, super_ssn, employee.dno from employee, (select dno
from employee group by dno having count(*)>=3) as dno_table where dno_table.dno=employee.dno;
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
| fname    | minit | lname   | ssn       | bdate      | address                 | sex  | salary | super_ssn | dno |
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
| John     | B     | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston TX | M    |  30000 | 333445555 |   5 |
| Franklin | T     | Wong    | 333445555 | 1965-12-08 | 638 Voss, Houston TX    | M    |  40000 | 888665555 |   5 |
| Joyce    | A     | English | 453453453 | 1972-07-31 | 5631 Rice, Houston TX   | F    |  25000 | 333445555 |   5 |
| Ramesh   | K     | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX | M    |  38000 | 333445555 |   5 |
| Jennifer | S     | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire TX  | F    |  43000 | 888665555 |   4 |
| Ahmad    | V     | Jabbar  | 987987987 | 1969-03-29 | 980 Dallas, Houston TX  | M    |  25000 | 987654321 |   4 |
| Alicia   | J     | Zelaya  | 999887777 | 1968-01-19 | 3321 Castle, Spring TX  | F    |  25000 | 987654321 |   4 |
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
7 rows in set (0.00 sec)
```

이를 바탕으로 3명 이상의 부원이 속한 부서에 속하는 부원들(4, 5 부서의 부원)의 리스트를 뽑았다. 여기서 dno가 중복되어 두 개 나타나는 문제가 있어 직접 필드를 하나하나 쳐서 출력하였다.

이제 이렇게 만들어진 리스트를 쿼리로 서브쿼리로 또 넣어서 최종적으로 아래 쿼리를 완성하였다.

```
select dnumber, ssn, salary
        from department, (select fname, minit, lname, ssn, bdate, address, sex,
       salary, super_ssn, employee.dno
                from employee, (select dno from employee group by dno having
       count(*)>= 3) as dno_table where dno_table.dno=employee.dno) as employee
       where employee.salary >= 40000;
```

```
mysql> select dnumber, ssn, salary from department, (select fname, minit, lname, ssn, bdate, address, sex, salary, super
_ssn, employee.dno from employee, (select dno from employee group by dno having count(*)>= 3) as dno_table where dno_tab
le.dno=employee.dno) as employee where employee.salary >= 40000;
+---------+-----------+--------+
| dnumber | ssn       | salary |
+---------+-----------+--------+
|       5 | 987654321 |  43000 |
|       5 | 333445555 |  40000 |
|       1 | 987654321 |  43000 |
|       1 | 333445555 |  40000 |
|       4 | 987654321 |  43000 |
|       4 | 333445555 |  40000 |
+---------+-----------+--------+
6 rows in set (0.00 sec)
```

결국에 서브 쿼리를 중복하여 사용해서 구현하였고, 더 좋은 방법이 있는지 고민해볼 필요가 있다고 느꼈다.

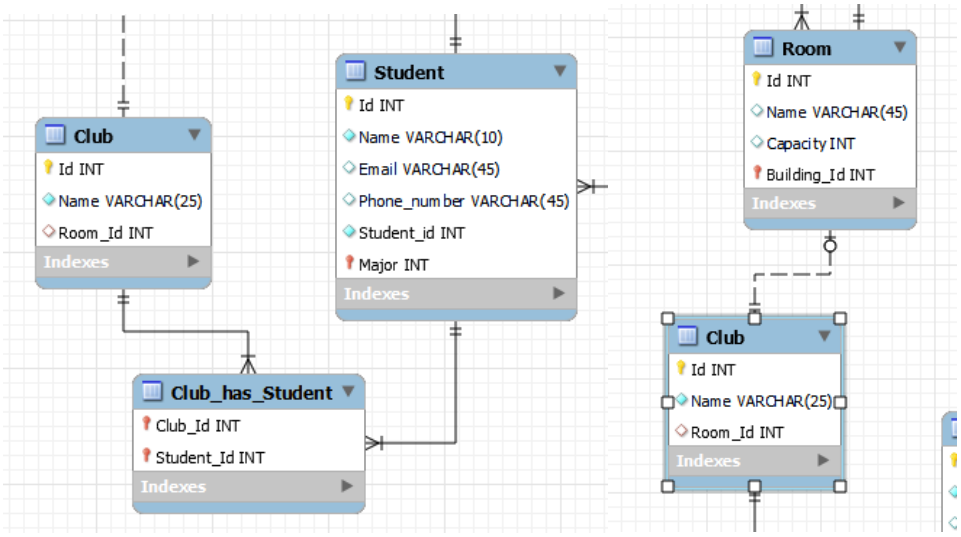
Step2) 데이터베이스 확장하기

Table Name: Club

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Id	INT	☑	☑	☐	☐	☐	☐	☑	☐	
Name	VARCHAR(25)	☐	☑	☑	☐	☐	☐	☐	☐	
Room_Id	INT	☐	☐	☐	☐	☐	☐	☐	☐	

그림 1 Club 테이블 description

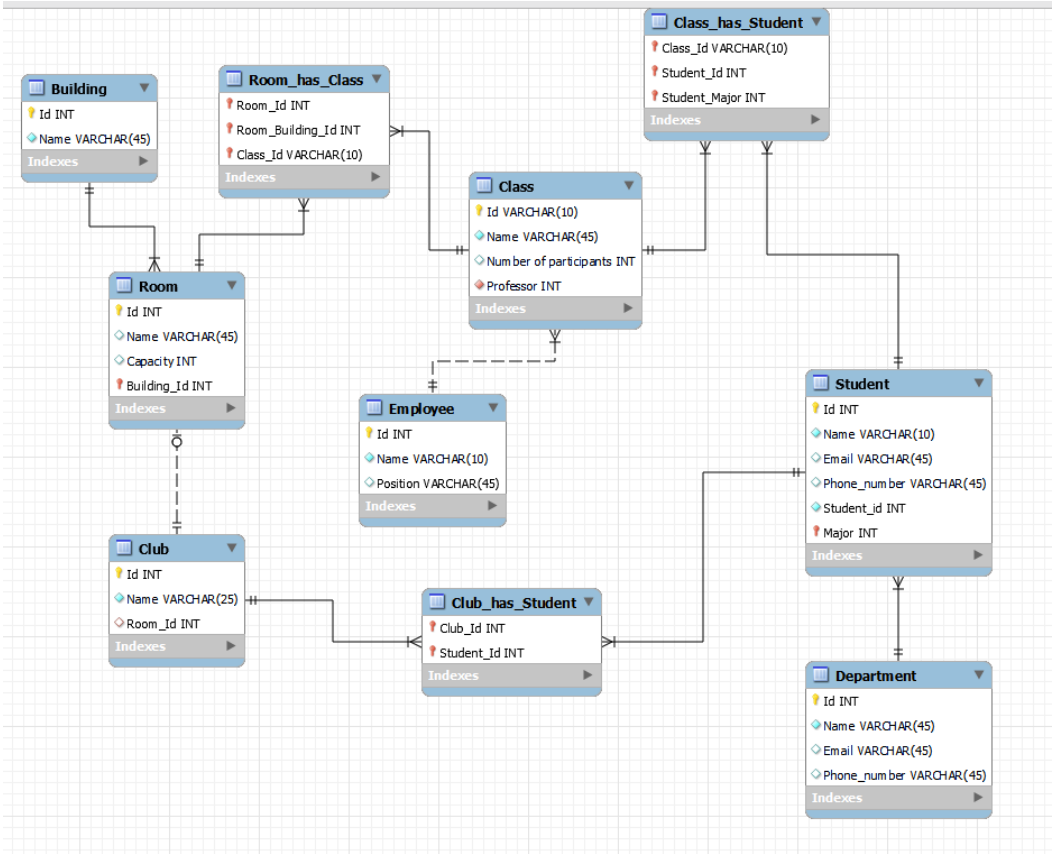
Club 테이블의 경우 위와 같이 Id, Name을 정의해주었다. 또한 기존 테이블 중에서 Room, 그리고 Student와 연관을 지어 주었는데, 각각 동아리방과 동아리원의 의미를 가진다. 따라서 테이블 관계도도 아래와 같이 설정해주었다.



Club은 Student와 다대다 관계를 가진다. 학생은 여러 동아리에 가입할 수 있고, 하나의 동아리는 여러 동아리원을 보유하고 있기 때문이다. 이에 따라 Club_has_Student라는 매핑 테이블이 생기게 되고, 여기에는 Club Id와 Student Id가 포함된다.

그리고 동아리는 보통 동아리방을 소유하고 있기 때문에, Room 테이블과 연관을 지어줬고 이때 동아리방이 없는 동아리도 있을 수 있으므로 Club 테이블의 Room_id는 NULL이 가능한 상태로 설정하였다. 동아리는 하나의 동아리방을, 하나의 동아리방은 하나의 동아리로 지정되어 있기 때문에 1:1 매핑을 수행하였다.

아래는 확장한 데이터베이스의 전체 도식도이다.



이제 VSCode에서 insert문을 작성해 테이블에 데이터를 입력하고자 하였다.

```
const sql = {
    getEmployee: async () => {
        const results = await promisePool.query(`
        insert into club(Name, Room_id) values ("이미지", 1);
        insert into club(Name, Room_id) values ("멍냥멍냥", 2);
        insert into club(Name, Room_id) values ("인하장학회", 3);
        insert into club(Name, Room_id) values ("트리키", 4);
        insert into club(Name, Room_id) values ("페다고지", 5);

        insert into employee(Name,  Position) values ("홍길동", "정보통신공학과 사무실 조교");
        insert into employee(Name,  Position) values ("김철수", "전기공학과 교수");
        insert into employee(Name,  Position) values ("김영희", "행정실 팀장");
        insert into employee(Name,  Position) values ("박개똥", "시설팀 부팀장");
        insert into employee(Name,  Position) values ("이미영", "기계실 팀장");
        `);
        return results;
    },
```

위와 같이 insert query문을 작성하였다. Connection과 다르게 Pool은 쿼리문 여러 개를 병렬적으로 처리할 수 있다. 그러나 아쉽게도 실행을 시켜보니 Node.js에서는 이런 방식으로의 Multiple query를 실행해주지 않고 두 번째 쿼리문부터 Syntax Error를 발생시키는 것을 알 수 있었다.

```
const sql = {
    getEmployee: async () => {

        const stmt_multiple_result_club = 'insert into club(Name, Room_id) values ?';
        var values_club = [
            ['이미지', 1],
            ['멍냥멍냥', 2],
            ['인하장학회', 3],
            ['트리키', 4],
            ['페다고지', 5]
        ];

        const stmt_mutiple_result_employee = 'insert into employee(Name, Position) values ?';
        var values_employee = [
            ['홍길동', '정보통신공학과 사무실 조교'],
            ['김철수', '전기공학과 교수'],
            ['김영희', '행정실 팀장'],
            ['박개똥', '시설팀 부팀장'],
            ['이미영', '기계실 팀장']
        ]

        let results = await promisePool.query(stmt_multiple_result_club, [values_club]);
        results = await promisePool.query(stmt_mutiple_result_employee, [values_employee]);

        return results;
    },
};
```

https://cheese10yun.github.io/mysql-multiple-insert/ 결국 이 블로그를 참고해서 위와 같이 코드를 수정하였고, 정상적으로 테이블에 값이 Insert 되는 것을 확인할 수 있었다.

```
mysql> select * from club;
+----+------------+---------+
| Id | Name       | Room_Id |
+----+------------+---------+
| 18 | 이미지     |       1 |
| 19 | 멍냥멍냥   |       2 |
| 20 | 인하장학회 |       3 |
| 21 | 트리키     |       4 |
| 22 | 페다고지   |       5 |
+----+------------+---------+
5 rows in set (0.00 sec)
```

```
mysql> select * from employee;
+----+--------+----------------------------+
| Id | Name   | Position                   |
+----+--------+----------------------------+
| 21 | 홍길동 | 정보통신공학과 사무실 조교 |
| 22 | 김철수 | 전기공학과 교수            |
| 23 | 김영희 | 행정실 팀장                |
| 24 | 박개똥 | 시설팀 부팀장              |
| 25 | 이미영 | 기계실 팀장                |
+----+--------+----------------------------+
5 rows in set (0.00 sec)
```

이제 select 문을 이용해서 테이블의 값을 조회해보자.

```
const sql = {
    getEmployee: async () => {
        const results = await promisePool.query(`select * from employee`)
        return results;
    },
    getClub: async () => {
        const results = await promisePool.query(`select * from club`)
        return results;
    }
};
```

그림 2 sql.js의 일부

```
app.get("/employee", async (req, res) =>
{   const employee = await sql.getEmployee();
    res.json({"Employee": employee})
});

app.get('/club', async (req, res) => {
    const club = await sql.getClub();
    res.json({"Club": club})
});
```

그림 3 index.js의 일부

기존의 index.js와 sql.js를 위와 같이 변경해주었고, npm run start로 실행 후 localhost:3000에 접속하니 아래와 같이 정상적으로 Employee와 Club 데이터가 뜨는 것을 확인할 수 있었다.

App.get에 employee와 club을 호출하는 select문을 다르게 둬서 localhost:3000/employee와 localhost:3000/club에 각각 접근했을 때 지정한 table의 data가 호출되도록 지정하여 위와 같은 결과가 나올 수 있었다.

3. 실행 화면

실행 화면은 상세 설계에 첨부한 사진으로 대체하도록 하겠습니다.

4. 결론

sql문을 설계할 때 count(*)에 의한 group by와 having문, 그리고 inner join과 where절을 혼합해서 사용하는 부분에서 순서를 지정하는 것이 헷갈렸습니다. 이를 서브 쿼리를 사용하는 것으로 해결하였으며 group by와 inner join을 동시에 사용하는 경우 inner

join이 우선시 되어 필터링 된 후 group by가 수행된다는 점을 깨닫게 되었습니다.