Homework Assignment 1

Author: Marios Toparopoulos
AM: 7115112400018
University of Athens
M149 - Fall 2024

1. Page 32 - Problem 1.11:

The transaction manager. More specifically the concurrency-control system. The database ensures that only one student's transaction will successfully allocate the last available seat by implementing a locking mechanism or optimistic concurrency control.

2. Page 32 - Problem 1.15

Users, Posts, Likes

Users:

```
CREATE TABLE Users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    date_of_birth DATE,
    profile_picture_url VARCHAR(255)
);
```

Posts:

```
CREATE TABLE Posts (
    post_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    content TEXT NOT NULL,
    media_url VARCHAR(255)
);
```

Likes

```
CREATE TABLE Likes (
    like_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    post_id INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (post_id) REFERENCES Posts(post_id)
);
```

3. Page 62 - Problem 2.11

s_id alone can no longer be the primary key of the advisor relation. This is because a primary key must uniquely identify each row in the table, and with the possibility of a student having multiple advisors, **s_id** would no longer uniquely identify each row.

4. Page 62 - Problem 2.14

a: \Pi_{person_name}(employee \bowtie_{employee.person_name = works.person_name} \sigma_{company_name="BigBank"}(works))

b: \Pi_{person_name, city}(employee \bowtie_{employee.person_name = works.person_name} \sigma_{company_name="BigBank"}(works))

c: \Pi_{person_name, street, city}(employee \bowtie_{employee.person_name = works.person_name} \sigma_{company_name="BigBank" \wedge salary > 10000 }(works))

5. Page 116 - Problem 3.2

a:

```
select sum(credits * points)
from takes, course, grade_points
where takes.grade = grade_points.grade and
      takes.course_id = course.course_id and
      ID = '12345';
```

b:

```
select sum(credits * points)/sum(credits) as GPA
from takes, course, grade_points
where takes.grade = grade_points.grade and
      takes.course_id = course.course_id and
      ID = '12345';
```

c:

```
select ID, sum(credits * points)/sum(credits) as GPA
from takes, course, grade_points
where takes.grade = grade_points.grade and
      takes.course_id = course.course_id
group by ID;
```

d:
The queries above, include a test of equality on grade between grade_points and takes. Grades with null are going to be eliminated

6. Page 118 - Problem 3.9

a:

```
select e.ID, e.person name, city
from employee as e, works as w
```

```
where w.company name = 'First Bank Corporation' and w.ID = e.ID;
```

b:

```
select * from employee
where ID in (
    select ID from works
    where  company_name = 'First Bank Corporation' and salary > 10000
);
```

c:

```
select ID from works
where company_name <> 'First Bank Corporation';
```

d:

```
select ID from works
where salary > all (
    select salary from works
    where  company_name = 'Small Bank Corporation');
```

e:

```
select S.company_name
from company as S
where not exists ((
    select city
    from company
    where company_name = 'Small Bank Corporation')
  except (
    select city from  company as T
    where S.company_name = T.company_name));
```

f:

```
select company_name
from works
group by company_name
having  count (distinct ID) >= all
    (select count (distint ID)
    from works
    group by  company_name)
```

g:

```
select company_name
from works
group by company_name
having avg (salary) > (select avg (salary)
from works
where company_name = 'First Bank Corporation');
```