Islamic University of Technology (IUT)

Report on Lab 07

Submitted By

Shanta Maria (ID:200042172)

CSE 4308 Database Management Systems Lab

Submitted To

Md. Bakhtiar Hasan

Lecturer, Department of CSE

Zannatun Naim Srsity

Lecturer, Department of CSE

November 16, 2022

Introduction

In the lab class, we were asked to draw an ER Diagram by analysing a given problem set.

1 Task

1 Lab Task

Nationa ID (NID) is an integrated collection of citizens' information such as Name, Date of Birth, Occupation, Blood Group. Each citizen has his/her own NID. In order to investigate the population density, the country has been divided into divisions. Each divison has its name, size (in square KM), and a brief description. Again, each division has a number of districts with similar attributes. Citizen information must be connected to its corresponding division and district.
Each citizen may have exactly one driving license where information such as type of license, issue date, expiration date are maintained. Whenever any accident occurs, it is logged in the central system. The system stores relevant information such as date and time of accident, location of accident, number of deaths (if any), etc.
There are a number of hospitals in the country having name and contact information. Each hospital may have more than one contact number. Citizens may avail treatment in any hospitals they prefer. Whenever any patient (i.e., citizen) is admitted, the system keeps the record of his/here date of admission, a brief description, and release date.
Now, your task is to:

1. Draw an ER Diagram, without any data redundancy, specifying the cardinality explicitly. You may add additional attributes only if it is needed.

2. Convert the ER Diagram into DDL using standard SQL denoting the appropriate constraints.

3. Write SQL statements for the following queries:

   (a) Find the list of divisions along with its total number of districts.
   (b) Find the list of districts having at least 20,000 people living there.
   (c) Find the number of accidents that involved a citizen whose NID is 210.
   (d) Find the list of top 5 hospitals based on the number of patients admitted so far.
   (e) Find the blood group of all the patients admitted to different hospitals.
   (f) Find the population density for each division.
   (g) Find the top 3 densely populated districts.
   (h) Find the number of accidents that occurred in each district.
   (i) Find the division where the least amount of accidents occurred.
   (j) Find the number of accidents caused by 'non-professional' and 'professional' license holders.

(k) Find the person who was admitted to the hospital for the longest period of time.

(l) Find the division where the number of young people ($15 \leq \text{age} \leq 30$) is the lowest.

(m) Find the people whose licenses expired.

(n) Find the number of accidents caused by people whose licenses expired.

(o) Find the license holders who were not involved in any accident so far.

(p) Find the number of deaths due to any accident for each division.

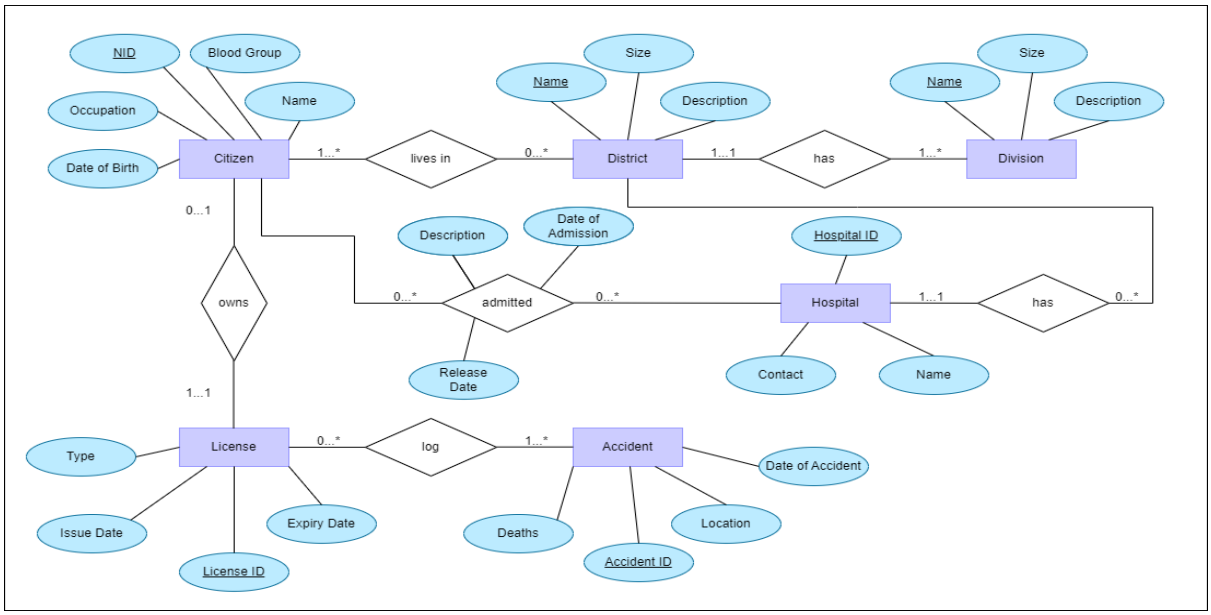
(q) Find the name of the people who got their license before the age of 22 or after the age of 40.

(r) Find the list of citizens who were admitted to the hospital on the same day they got into an accident.

(s) Find the hospital where people from Dhaka division were admitted the most.

(t) Find the list of people who caused an accident outside their own district.

2 Solution for Task 1



2.1 Analysis and Explanation

There are a total of 7 tables:

1. `central_system`

   (a) Primary Key: `nid` and `hospital_id`

(b) Foreign Keys: `nid` from `citizen`, `hospital_id` from `hospital`

(c) Cardinality :

i. One to One Relationships: `none`

ii. One to Many Relationships: `citizen, hospital`

iii. Many to Many Relationships: `none`

2. `division`

(a) Primary Key: `division_name`

(b) Foreign Keys: `district_name` from `district`

(c) Cardinality :

i. One to One Relationships: `none`

ii. One to Many Relationships: `district`

iii. Many to Many Relationships: `none`

3. `accident`

(a) Primary Key: `accident_id`

(b) Foreign Keys: `license_id` from `license`

(c) Cardinality :

i. One to One Relationships: `none`

ii. One to Many Relationships: `license`

iii. Many to Many Relationships: `none`

4. `license`

(a) Primary Key: `license_id`

(b) Foreign Keys: `nid` from `citizen`

(c) Cardinality :

i. One to One Relationships: `citizen`

ii. One to Many Relationships: `accident`

iii. Many to Many Relationships: `none`

5. `citizen`

   (a) Primary Key: `nid`

   (b) Foreign Keys: `district_name` from `district`

   (c) Cardinality :

      i. One to One Relationships: `none`

      ii. One to Many Relationships: `district, central_system, license`

      iii. Many to Many Relationships: `none`

6. `hospital`

   (a) Primary Key: `hospital_id`

   (b) Foreign Keys: `district_name` from `district`

   (c) Cardinality :

      i. One to One Relationships: `district`

      ii. One to Many Relationships: `hospital`

      iii. Many to Many Relationships: `none`

7. `district`

   (a) Primary Key: `district_name`

   (b) Foreign Keys: `none`

   (c) Cardinality :

      i. One to One Relationships: `division`

      ii. One to Many Relationships: `hospital, citizen`

      iii. Many to Many Relationships: `none`

2.2 Difficulties

I faced difficulties in figuring out how each entity is connected to one another and what the relationship is between them. I found it challenging to draw the ER diagram as well.

3 Solution for Task 2

```
drop table central_system;
drop table division;
drop table accident;
drop table license;
drop table citizen;
drop table hospital;
drop table district;

create table district
(
    district_name varchar2(25) not null,
    district_size number,
    district_description varchar2(50),
    constraint pk_district_name primary key(district_name)
);

create table citizen
(
    nid number not null,
    citizen_name varchar2(50),
    date_of_birth date,
    occupation varchar2(25),
```

```
    blood_group varchar2(5),
    district_name varchar2(25),
    constraint pk_nid primary key(nid),
    constraint fk_district_name foreign key(district_name)
    references district(district_name)
);

create table division
(
    division_name varchar2(25) not null,
    division_size number,
    division_description varchar2(50),
    district_name varchar2(25),
    constraint pk_division_name primary key(division_name),
    constraint fk_div_district_name foreign key(district_name)
    references district(district_name)
);

create table license
(
    license_id number not null,
    license_type varchar2(25),
    issue_date date,
    expired_date date,
    nid number,
    constraint pk_license_id primary key(license_id),
    constraint fk_nid foreign key(nid)
    references citizen(nid)
);
```

```
create table hospital
(
    hospital_id number not null,
    contact number,
    hospital_name varchar2(25),
    district_name varchar2(25),
    constraint pk_hospital_id primary key(hospital_id),
    constraint fk_hosp_district_name foreign key(district_name)
    references district(district_name)
);

create table accident
(
    accident_id number not null,
    no_deaths number,
    place_of_accident varchar2(25),
    date_of_accident date,
    license_id number,
    constraint pk_accident_id primary key(accident_id),
    constraint fk_license_id foreign key(license_id)
    references license(license_id)
);

create table central_system
(
    nid number not null,
    hospital_id number not null,
    accident_desc varchar2(50),
```

```
    date_of_admission date,
    release_date date,
    constraint pk_log primary key(hospital_id, nid),
    constraint fk_cs_nid foreign key(nid)
    references citizen(nid),
    constraint fk_cs_hospital_id foreign key(hospital_id)
    references hospital(hospital_id)
);
```

3.1 Analysis and Explanation

These are the DDL statements according to the cardinality, foreign key and primary key constraints mentioned in the explanation of Task 1.

3.2 Difficulties

This is easy to solve and no mentionable issues were encountered.

4 Solution for Task 3

```
---task(a)---
select max(division_name), max(division_size),
max(division_description), count(district_name)
from division
group by division_name;

---task(b)---
---view for number of citizens in a district---
```

```
create or replace view district_citizens as
select count(nid) as num
from citizen
group by district_name;

select *
from district
where district_name in (select district.district_name
                        from district, district_citizens
                        where district_citizens.num>=20000);

---task(c)---
select count(accident_id)
from accident natural join license natural join citizen
where nid=210;

---task(d)---
select hospital_id
from (select hospital_id, count(nid)
        from hospital natural join central_system
        group by hospital_id
        order by count(nid) desc)
where rownum<=5;

---task(e)---
select blood_group
from central_system natural join citizen;

---task(f)---
```

```
select division_name, count(nid) as density
from citizen natural join district natural join division
group by division_name;

---task(g)---
select district_name
from (select district_name, count(nid)
        from district natural join citizen
        group by district_name)
where rownum<=3;

---task(h)---
select count(accident_id)
from citizen natural join license natural join accident
group by district_name;

---task(i)---
select division_name
from (select division_name, count(accident_id)
        from citizen natural join license natural join
        accident natural join district natural join division
        group by division_name
        order by count(accident_id))
where rownum<=1;

---task(j)---
select count(accident_id)
from accident natural join license
where license_type='non-professional' or license_type='professional';
```

```
---task(k)---
select citizen_name
from citizen natural join central_system
where (date_of_admission-release_date) in
    (select min(date_of_admission-release_date)
    from central_system);

---task(l)---
create or replace view citizen_age as
select ((extract(year from sysdate))-
(extract(year from date_of_birth))) as age, nid
from citizen;

select division_name
from (select division_name, count(nid)
        from division natural join district natural join
        citizen natural join citizen_age
        where age<=30 and age>=15
        group by division_name
        order by count(nid))
where rownum<=1;

---task(m)---
select citizen_name
from citizen natural join license
where expired_date<sysdate;

---task(n)---
```

```
select count(accident_id)
from accident natural join license
where expired_date<sysdate;

---task(o)---
(select license_id
from license)
minus
(select license_id
from accident);

---task(p)---
select division_name, sum(no_deaths)
from accident natural join license natural join
citizen natural join district natural join division
group by division_name;

---task(q)---
create or replace view citizen_age_license as
select ((extract(year from issue_date))-
(extract(year from date_of_birth))) as age, nid
from citizen natural join license;

select citizen_name
from citizen natural join license natural join citizen_age_license
where age<22 or age>40;

---task(r)---
select nid, citizen_name
```

```
from citizen natural join accident natural join
central_system natural join license
where date_of_accident=date_of_admission;

---task(s)---
select max(hospital_name), max(patients)
from (select hospital_id, count(nid) as patients
        from hospital natural join central_system natural join
        citizen natural join district natural join division
        where division_name='Dhaka'
        group by hospital_id) natural join hospital
where rownum<=1;

---task(t)---
select nid
from citizen natural join license natural join accident
where district_name<>place_of_accident;
```

4.1 Analysis and Explanation

Task (a)

I used max() functions for the attributes in the select statement since group by function is used to group the results by each division since we need the total number of districts for each division.

Task (b)

Here I created a view to return the number of citizens in a district. Then I used a nested query for the query which returns all the district name with the citizen count greater than the specified amount. I returned the details for the districts which was in the nested query result.

Task (c)

Here I used natural join to join several tables to extract the accident id of the specified citizen and count the number of accidents they caused using count() function.

Task (d)

I used a nested query here to return the hospitals with the number of admitted patients in descending order and then used rownum to output the top 5 results from the nested query.

Task (e)

I used natural join to connect the tables and get the blood group from different hospitals.

Task (f)

I used group by function to group the results by division name since I used count() function to calculate the number of citizens. This will give the density for each division.

Task (g)

This task was very similar to the last task except here I grouped the results by district name and put the query in a nested query so that I can use rownum for the outer query to show the top 3 results only.

Task (h)

I used natural join to get information from several tables at once and counted the number of accident IDs grouped by district name.

Task (i)

For this task, I used the query from the last question in a nested query. I ordered the results in ascending order then printed the top result using rownum which would show the least count.

Task (j)

Here I counted the accident IDs for the license types specified in the question from the table created by the natural joining of accident and license table.

Task (k)

Here I used min() function to find the smallest date since the difference would be maximum then. The query used to find the minimum date is nested since aggregate function is used.

Task (l)

I used extract function here which returns the date part specified inside the function. I created a view at first which returned the citizen's age using the current date and their date of birth. Then I used that view to apply constraints on a nested query which returns the division name and number of citizens there. The citizens have constraints on their age as mentioned in the question. The results are grouped by division name so the count for each division is shown and the results are ordered so when only the first row is printed using rownum, the least count is shown.

Task (m)

Here I checked the license expiry date against the current date. If the date is lower then the license is expired.

Task (n)

I used count() to count the number of accident IDs and checked if the license was expired or not using the same method as in the last task.

Task (o)

I subtracted the license IDs from the accident table to get the license IDs which were not involved in any accident at all.

Task (p)

Here I used sum() to calculate the total number of deaths and grouped the results by division name so that the death count for each division is shown.

Task (q)

To calculate the age that the person got their license at, I used extract() function again to find the difference between the issue date of the license and their date of birth. I created a view to find this age at first. Then I used the view in the query and put the constraints on the age as specified in the question.

Task (r)

Here I used natural join to join several tables and checked if the date of admission and date of accident were the same. I printed the results for that condition.

Task (s)

At first I used a nested query to return the results for the hospital and their patient count by grouping the results by hospital ID for the hospitals in the division specified. Natural joining several tables made this possible. I also used max() function for the number of patients and printed the first result only to show the hospital with the highest patients.

Task (t)

Here I used not equal to operator to check if the district name of the citizen matched or not with the place of the accident.

4.2 Difficulties

It was difficult to understand if the queries were showing the correct results or not without any data in the tables. I also found it difficult to figure out how to calculate the citizen age.

Conclusion

As shown in the report, I have solved and tested the solutions for all four tasks given in the lab. All the commands used were written in VS Code which was then saved with .sql extension. The .sql file was then run through the `SQL` command line to execute all the commands.