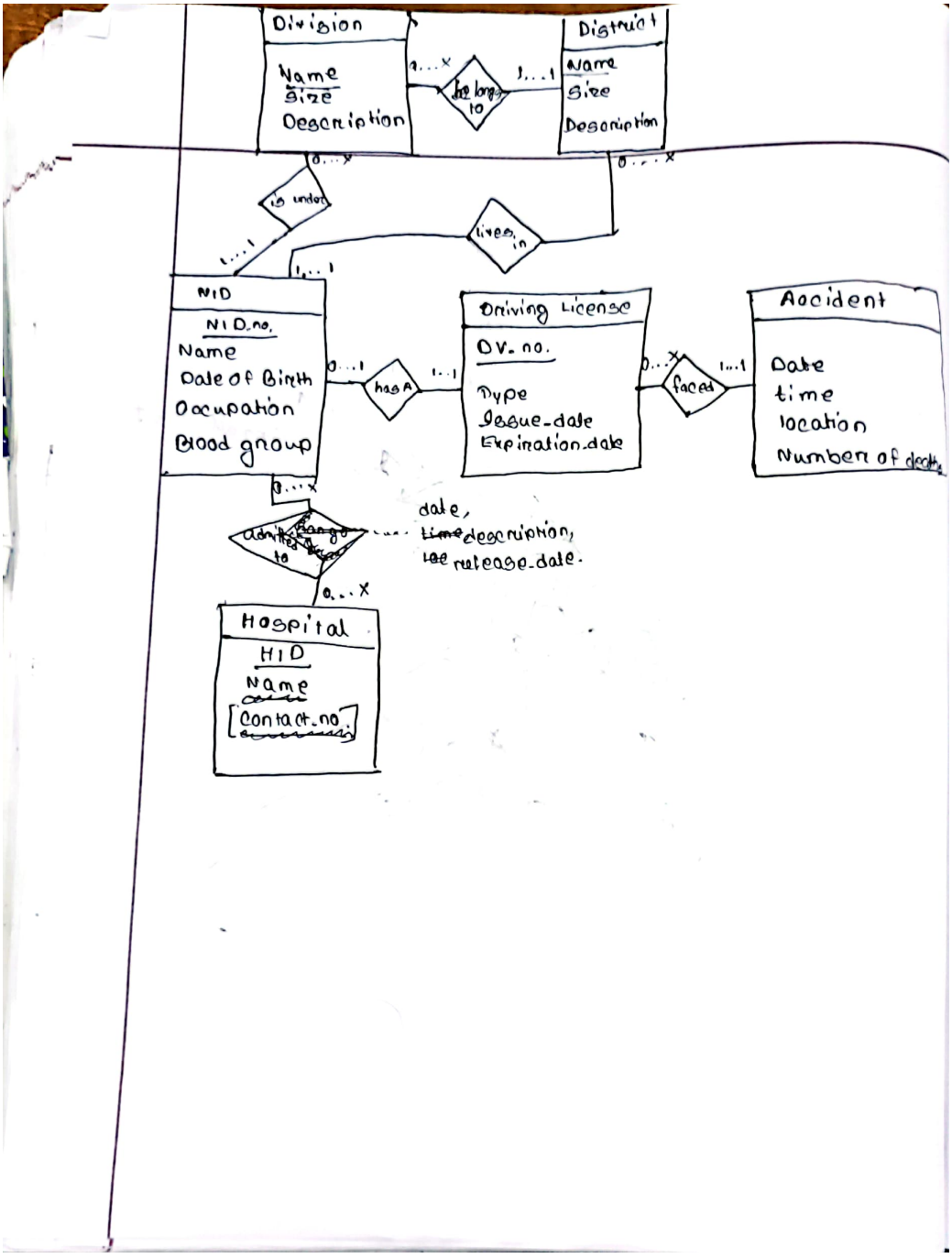
Database Management Systems Lab
Lab 7
CSE 4308

A Z Hasnain Kabir
200042102
Software Engineering

26 October 2022

Task1

We are to draw Entity Relationship diagram for the following problem statement.
National 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.


Division
Name
Size
Description
belongs to
District
Name
Size
Description
is under
lives in
NID
NID_no.
Name
Date of Birth
Occupation
Blood group
has A
Driving License
DV_no.
Type
Issue_date
Expiration_date
faces
Accident
Date
time
location
Number of death
admitted to
date,
description,
release_date.
Hospital
HID
Name
Contact_no

Analysis of the problem

We can clearly state from the problem statement that we need a minimum of 6 tables initially for division, district, NID, Driving_license, Accident and Hospital.

Explanation of the solution

We create six tables with the relationships mentioned on the diagram. Each table has its own attributes described in the table. There is one many to many relationship present inside the diagram. Contact_no is a multivalued attribute inside Hospital table.

Findings

We do not need to create a relationship from NID to district has each district contains a relationship with a division even through it is inferred in the problem statement that each citizen belongs inside a district. We also do not need to create a citizen table as the information is present inside NID table and citizen table will be redundant here.

Task 2

We have to convert the ER diagram to DDL statements using SQL denoting the appropriate constraints.

Analysis of the problem

We can write DDL SQL statements for the given tables. We need to create junction table for many to many relationships.

Code

```
CREATE OR REPLACE TYPE vmobiles as varray(10) of varchar2(20);

CREATE TABLE NID(
    NID_no INT primary key,
    Name varchar2(20),
    Date_of_birth DATE,
    Occupation varchar2(20),
    Blood_group varchar2(5)
```

```
);

CREATE TABLE driving_license(
    DV_no INT primary key,
    Type_of_license varchar2(20),
    Issue_date DATE,
    Expiration_date DATE,
    NID references NID NOT NULL
);

CREATE TABLE accident(
    date_of_accident Date primary key,
    time_of_accident timestamp,
    location varchar2(20),
    number_of_deaths number,
    license_no references driving_license NOT NULL
);

CREATE TABLE division(
    Name varchar2(20) primary key,
    size_of_division number,
    Description varchar2(250)
);


CREATE TABLE district(
    Name varchar2(20) primary key,
    size_of_district number,
    Description varchar2(250),
    division references division NOT NULL
);


ALTER TABLE NID ADD division references division;
ALTER TABLE NID ADD district references district;

CREATE TABLE hospital(
    HID number primary key,
    Name varchar2(20),
    contact_no vmobiles
);
```

```
CREATE TABLE admittedTo(
    date_of_admission date,
    Description varchar2(250),
    date_of_release date,
    NID references NID NOT NULL,
    HID references hospital NOT NULL
);
```

Explanation of the solution

The solution contains multivalued attribute vmobile as an array of varchar. This multivalued attribute is used to store multiple contact numbers of hospital table. The other SQL statements are pretty basic SQL statements to define a database along with its constraints.

Findings

Since the tables such as district and division are created after NID table, we have to add foreign key constraints later by altering the already created table.

Task 3

We are to write SQL statements for the given queries

- Find the list of divisions along with its total number of districts.
- Find the list of districts having at least 20,000 people living there.
- Find the number of accidents that involved a citizen whose NID is 210.
- Find the list of top 5 hospitals based on the number of patients admitted so far.
- Find the blood group of all the patients admitted to different hospitals.
- Find the population density for each division.
- Find the top 3 densely populated districts.
- Find the number of accidents that occurred in each district.

- Find the division where the least amount of accidents occurred.
- Find the number of accidents caused by 'non-professional' and 'professional' license holders.
- Find the person who was admitted to the hospital for the longest period of time.
- Find the division where the number of young people (15 ¡= age ¡= 30) is the lowest.
- Find the people whose licenses expired.
- Find the number of accidents caused by people whose licenses expired.
- Find the license holders who were not involved in any accident so far.
- Find the number of deaths due to any accident for each division.
- Find the name of the people who got their license before the age of 22 or after the age of 40.
- Find the list of citizens who were admitted to the hospital on the same day they got into an accident.
- Find the hospital where people from Dhaka division were admitted the most.
- Find the list of people who caused an accident outside their own district.

Analysis of the problem

We need to write SQL queries that we learned in the previous classes to solve the given queries.

Code

```
--A
SELECT count(Name), division FROM district GROUP BY division;

--B
SELECT district FROM NID GROUP BY district HAVING count(Name) >
↪  20000;
```

```sql
--C
SELECT count(date_of_accident) FROM accident GROUP BY
↪  license_no HAVING license_no =
(SELECT DV_no FROM driving_license WHERE NID = 210);

--D
SELECT name FROM hospital WHERE HID = (SELECT HID FROM
↪  admittedTo GROUP BY HID
ORDER BY  count(NID) DESC HAVING ROWNUM < 6);

--E
SELECT Blood_group FROM NID WHERE NID.NID_no in (SELECT NID
↪  FROM admittedTo);

--F
SELECT name, pop/size_of_division as density FROM
(SELECT division, count(NID_no) as pop FROM NID GROUP BY
↪  division), division
WHERE name = division ;

--G
SELECT name, pop/size_of_division as density FROM
(SELECT division, count(NID_no) as pop FROM NID GROUP BY
↪  division), division
WHERE name = division GROUP BY name ORDER BY density DESC
↪  HAVING ROWNUM <= 3;

--H
SELECT district, COUNT(NID_NO) FROM NID WHERE NID_NO IN
(SELECT NID FROM driving_license,accident WHERE
DV_no =license_no) GROUP BY district;

--I
SELECT division, COUNT(NID_NO) AS NUM FROM NID WHERE NID_NO IN
(SELECT NID FROM driving_license,accident WHERE
DV_no =license_no) GROUP BY division ORDER BY NUM WHERE ROWNUM
↪  <=1;

--J
```

```
SELECT Type_of_license, COUNT(DV_no) FROM driving_license WHERE
↪  DV_no
IN (SELECT license_no FROM accident) GROUP BY Type_of_license;

--K
SELECT NID.name
FROM NID,AdmittedTO
WHERE NID.NID_NO=AdmittedTO.NID AND longestDate IN (
    SELECT
↪  max(datediff(AdmittedTO.date_of_release,Admitted.date_of_admission))
↪  longestDate
        FROM AdmittedTO
);

--M
SELECT DV_no FROM driving_license WHERE Expiration_date <
↪  CURRENT_DATE;

--N
SELECT COUNT(DV_no) FROM driving_license WHERE Expiration_date
↪  < CURRENT_DATE
AND DV_no IN (SELECT license_no FROM accident);

--O
SELECT DV_no FROM driving_license
AND DV_no NOT IN (SELECT license_no FROM accident);


--P
SELECT MAX(number_of_deaths), DIV.name FROM Accident A,
driving_license D, NID N, Division DIV WHERE A.LICENSE_NO =
↪  D.DV_no
AND N.NID_NO = D.NID AND N.Division = DIV.Name GROUP BY
↪  DIV.NAME;

--Q
SELECT N.NAME FROM NID N, driving_license D WHERE N.NID_NO =
↪  D.NID AND
DATE_FORMAT(FROM_DAYS(DATEDIFF(D.Issue_date,N.Date_of_birth)),
↪  '%Y') + 0
```

```
in (SELECT
↪  DATE_FORMAT(FROM_DAYS(DATEDIFF(D.Issue_date,Date_of_birth)),
↪  '%Y') + 0 AS AGE
FROM NID,driving_license
where AGE<=22 and AGE <=30);

--R
SELECT N.NAME FROM NID N, admittedTo ADT WHERE N.NID_NO =
↪  ADT.NID AND
ADT.date_of_admission IN (SELECT date_of_accident FROM
↪  Accident);

--S
SELECT MAX(H.NAME), DIV.NAME FROM (
    SELECT H.NAME, DIV.NAME  FROM Hospital H, admittedTo ADT,
↪  NID N,
    division DIV  WHERE H.HID = ADT.HID AND
    N.NID_NO = ADT.NID AND  DIV.NAME = N.DIVISION
) GROUP BY DIV.NAME HAVING DIV.NAME = "DHAKA";
```

Explanation of the solution

Other than the normal SQL statements, oracle offers a wide range of keywords such as CURRENT_DATETIME, datediff, etc. to identify differences between dates. Also, we have used many subqueries to write SQL statements that clearly designs the logic of our tasks.

Findings

According to my opinion, using subqueries instead of join or cartesian operations describe the underlying logic of a query more accurately.