NAME: NAZIA KARIM KHAN OISHEE

STUDENT ID: 200042137

COURSE CODE: CSE 4308

SUBJECT: DATABASE MANAGEMENT SYSTEMS LAB

Introduction

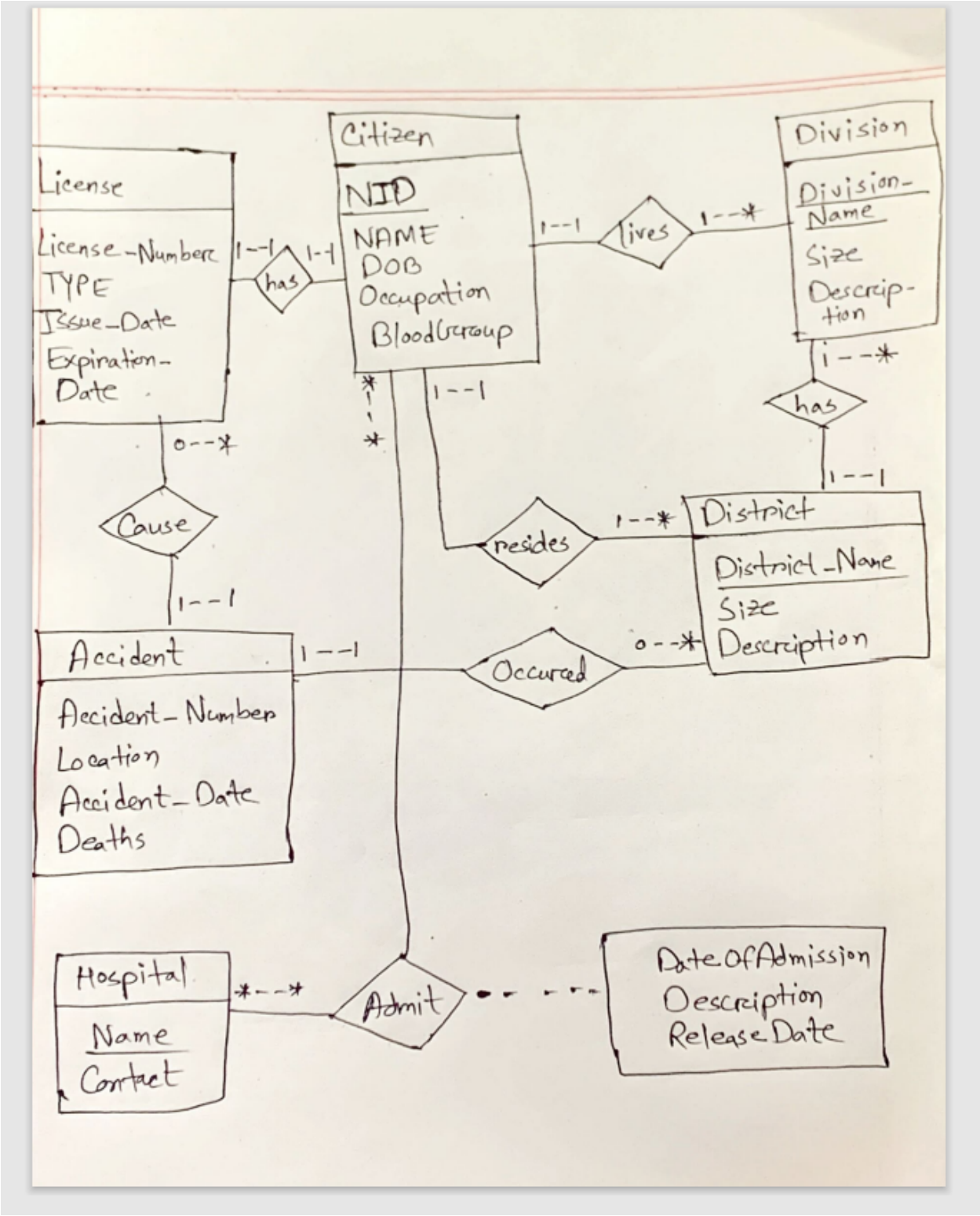
Our task in this lab was to draw an ER Diagram for the given scenario, convert the ER Diagram into DDL and write SQL statements to execute given queries.

ER Diagram stands for Entity Relationship Diagram. ERD displays the relationship of entity sets stored in a database. ER Diagram consists of three basic components- entities, attributes of the entities and relation among the entities. ER Diagram provides visual representation of the database and makes the process of designing schema of the database easier.

Task1

Our first task was to create the ERD. I drew the ERD with pen and paper.Here I am providing the image of the ERD.

Solution:


Citizen
NID
NAME
DOB
Occupation
BloodGroup
License
License_Number
TYPE
Issue_Date
Expiration_Date
has
1--1
1-1
lives
1--*
Division
Division_Name
Size
Description
1--*
has
1--1
0--*
Cause
1--1
resides
1--*
District
District_Name
Size
Description
Accident
Accident_Number
Location
Accident_Date
Deaths
1--1
Occured
0--*
Hospital
Name
Contact
--
Admit
Date_Of_Admission
Description
Release_Date

Link of the ERD; Lab7.pdf

Explanation:

In the diagram, I have created 7 entities along with their necessary attributes.

Citizen entity has 5 attributes- NID, Name,DOB,Occupation and BloodGroup. Here NID is the primary key.

Division entity has 3 attributes- Division_Name,Size and Description. Here Division_Name is the primary key.

District entity also has 3 attributes- District_Name,Size and Description. Here Dsitrict_Name is the primary key.

License entity has 5 attributes- License_Number, TYPE, Issue_Date, Expiration_Date. Here License_Number is the primary key.

Accident entity has 4 attributes- Accident_Number, Location,Accident_Date and Deaths. Here Accident_Number is the primary key.

Hospital entity has 2 attributes- Name and Contact. Here Name is the primary key.

Each citizen can live only in one division or district. That's why Citizen has one to many relation with Division and District.

Each district can belong to only one division. So district and and division also have one to many relation.

Each License can only belong to one citizen. Considering the given scenario I assumed each citizen can have only one driving license. So license and citizen have a one to one relation.

Each accident can be caused by only one driver. So only one license holder is to blame for a particular accident. But a license holder can cause zero or many accidents. So license and accident have one to many relation.

Each Accident occurs in a particular district. Many accidents can occur in a district including zero accidents. So accident and district have one to many relation.

Each citizen can get admitted to many hospitals in his/her lifespan. Each hospital can treat many citizens. So hospital and citizen have many to many relation. A junction table named Admit is created for this many to many relation. Admit has additional attributes - DateOfAdmission, Description and ReleaseDate.

All the relations in the ERD are binary relations.

Task2

Our second task was to convert the ER Diagram into DDL using SQL.

Solution:

```
CREATE TABLE Citizen(
    NID NUMBER,
    NAME VARCHAR(255),
    DOB DATE,
    Occupation TEXT,
    BloodGroup VARCHAR(255),
    Division_Name VARCHAR(255),
    District_Name VARCHAR(255),
    CONSTRAINT PK_Citizen PRIMARY KEY (NID),
    CONSTRAINT FK_Citizen1 FOREIGN KEY (Division_Name) REFERENCES
Division(Division_Name),
    CONSTRAINT FK_Citizen2 FOREIGN KEY (District_Name) REFERENCES
District(District_Name)

);
CREATE TABLE Division(
    Division_Name VARCHAR(255),
    Size NUMBER,
    Description TEXT,
    CONSTRAINT PK_Division PRIMARY KEY (NAME)
```

```
);
CREATE TABLE District(
    District_Name VARCHAR(255),
    Division_Name VARCHAR(255),
    Size NUMBER,
    Description TEXT,
    CONSTRAINT PK_District PRIMARY KEY (NAME),
    CONSTRAINT FK_District FOREIGN KEY (Division_Name) REFERENCES
Division(Division_Name)
);
CREATE TABLE License(
License_Number NUMBER,
TYPE VARCHAR2(20),
Issue_Date DATE,
Expiration_Date DATE,
NID NUMBER,
CONSTRAINT PK_LICENSE PRIMARY KEY (License_Number),
CONSTRAINT Fk_License FOREIGN KEY (NID) REFERENCES Citizen(NID)
);
CREATE TABLE Accident(
Accident_Number NUMBER,
Location VARCHAR2(20),
Accident_Date DATE,
Deaths NUMBER,
License_Number NUMBER,
District_Name VARCHAR(255),
CONSTRAINT PK_Accident PRIMARY KEY (Accident_Number),
CONSTRAINT Fk_Accident1 FOREIGN KEY (License_Number) REFERENCES
License(License_Number) ,
CONSTRAINT Fk_Accident2 FOREIGN KEY (District_Name) REFERENCES
District(District_Name)
);

CREATE TABLE Hospital(
Hospital_Name VARCHAR(255),
TYPE CONTACT_INFO IS VARRAY(50) OF VARCHAR2(11),
CONSTRAINT Hospital_PK PRIMARY KEY Hospital_Name
```

```
);

CREATE TABLE Admit(
    Hospital_Name VARCHAR(255),
    NID NUMBER,
    DateOfAdmission DATE,
    Description VARCHAR(255),
    ReleaseDate DATE,
    CONSTRAINT PK_Admit1 PRIMARY KEY (Hospital_Name),
    CONSTRAINT PK_Admit2 PRIMARY KEY (NID),
    CONSTRAINT Fk_Admit1 FOREIGN KEY (Hospital_Name) REFERENCES
Hospital(Hospital_Name),
    CONSTRAINT Fk_Admit2 FOREIGN KEY (NID) REFERENCES Citizen(NID)
    );
```

Explanation:

To convert the ERD into a table we need to use the 'CREATE TABLE' command.

To implement many to one relation a foriegn key is introduced at the many part referencing the other entity. I used constraints such as primary key and foreign key to impose the constraints.

Task3

(a) Find the list of divisions along with its total number of districts

.Query

```
SELECT Division_Name , COUNT(District_Name)
FROM DISTRICT
GROUP BY Division_Name;
```

Using GROUP BY I separated districts belonging to the same district and then counted the number of members in each group.

(b) Find the list of districts having at least 20,000 people living there.

Query

```
SELECT District_Name , COUNT(NID)
FROM Citizen
GROUP BY District_Name
HAVING COUNT(NID)>=20000;
```

Using GROUP BY I separated citizens belonging to the same district and then counted the number of citizens in each group.

(c) Find the number of accidents that involve a citizen whose NID is 210.

Query

```
SELECT COUNT(Accident_Number)
FROM Accident NATURAL JOIN License
WHERE NID = 210;
```

License has NID of the license holder. Joining License with Accident we can access both-NID and number of accidents.

(d) Find the list of top 5 hospitals based on the number of patients admitted so far.

Query

```
SELECT Hospital_Name
FROM(
SELECT Hospital_Name, ,COUNT(NID)
FROM Admit
GROUP BY Hospital_Name
ORDER BY DESC)
```

```
WHERE ROWNUM<=5;
```

After arranging the hospital in descending order based on the number of patients admitted so far , I chose top 5 hospitals using ROWNUM.

(e) Find the blood group of all the patients admitted to different hospitals.

Query

```
SELECT BloodGroup,NID
FROM Admit NATURAL JOIN Citizen;
```

Admit has the NID of the patients and Citizen has the citizens' bloodgroup. That's why I joined these two entities to access the information.

(f) Find the population density for each division.

Query

```
SELECT POPULATION/Division.Size
FROM( SELECT Division_Name , COUNT(NID) AS POPULATION
FROM Citizen
GROUP BY Division_Name) , DIVISION;
```

Counting the population of each Division ,I divided the population by the division's size to calculate the density.

(g) Find the top 3 densely populated districts.

Query

```
SELECT DISTRICT_NAME , DENSITY
FROM (
SELECT District_Name , POPULATION/District.Size AS DENSITY
FROM(
SELECT District_Name , COUNT(NID) AS POPULATION
```

```
FROM Citizen
GROUP BY District_Name;
), District)
ORDER BY DENSITY DESC
WHERE ROWNUM<=3;
```

Using the sub-query I first calculated the density and then using ROWNUM selected the top 3 records.

(h) Find the number of accidents that occurred in each district.

Query

```
SELECT District_Name, COUNT(*)
FROM Accident
GROUP_BY District_Name;
```

Using GROUP BY I separated accidents occured in the same district and then counted the number of accidents in each group.

(i) Find the division where the least amount of accidents occurred.

Query

```
SELECT District_Name,NumberOfAccident
FROM(
SELECT Division_Name, COUNT(*) AS NumberOfAccident
FROM (Accident NATURAL JOIN District) NATURAL JOIN Division
GROUP_BY Division_Name
ORDER BY NumberOfAccident ASC)
WHERE ROWNUM<2;
```

Using the sub-query I counted the number of accidents in each division and arranged them in ascending order . UsinG ROWNUM I selected the top record with the least number of accidents.

(j) Find the number of accidents caused by ‘non-professional’ and ‘professional’ license holders.

Query

```
SELECT TYPE , COUNT(*)
FROM License NATURAL JOIN Accident
GROUP BY TYPE;
```

Using GROUP BY I separated accidents occurred by different type of license holder and then counted the number of accidents in each group.

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

Query

```
SELECT NAME
FROM(
SELECT NAME
FROM (
SELECT NID, (ReleaseDate - DateOfAdmission) AS Duration
FROM Admit
ORDER BY Duration AS DESC
) NATURAL JOIN Citizen
)
WHERE ROWNUM<=1;
```

Using the sub-query I counted the duration of the patients and arranged them in descending order and joined the information with Citizen to get the patients’ name. UsinG ROWNUM I selected the top record with the longest time duration.

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

Query

```
SELECT *
FROM(
SELECT Division_Name,COUNT(NID) as People
FROM Citizen
WHERE TRUNC(MONTHS_BETWEEN(sysdate, DOB) / 12) >=15 AND
TRUNC(MONTHS_BETWEEN(sysdate,DOB) / 12) <=30
ORDER BY COUNT(NID) ASC
)
WHERE ROWNUM<=1;
```

Using MONTHS_BETWEEN I calculated the age and then arranging the records in ascending order chose the top record.

(m) Find the people whose licenses expired.

Query

```
SELECT *
FROM License
WHERE SYSDATE > Expiration_Date;
```

SYSDATE is used to compare the expiration date with the current date.

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

Query

```
SELECT COUNT(*)
FROM Accident NATURAL JOIN License
WHERE SYSDATE > Expiration_Date;
```

SYSDATE is used to compare the expiration date with the current date and Accident is joined to count the number of accidents .

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

Query

```
SELECT License_Number
FROM License
WHERE License_Number NOT IN (
SELECT  License_Number
FROM Accident);
```

NOT IN is used to separate the license holder from the ones who caused any accident.

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

Query

```
SELECT Division_Name , SUM(Deaths)
FROM Accident NATURAL JOIN District
GROUP_BY Division_Name;
```

District contains the division name. So I joined district and accident for this query.

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

Query

```
SELECT NAME
FROM License NATURAL JOIN Citizen
WHERE TRUNC(MONTHS_BETWEEN(sysdate,DOB)/12)<22 OR
TRUNC(MONTHS_BETWEEN(sysdate,DOB)/12) >40;
```

Citizen contains citizens' names. So I joined citizen with license for the query.

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

Query

```
SELECT NAME
FROM Accident ,(Admit NATURAL JOIN Citizen)
WHERE Accident_Date = DateOfAdmission;
```

I joined admit with citizen to access the name of the citizens and select those records which have the same date.

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

Query

```
SELECT Hospital_Name
(SELECT Hospital_Name , COUNT(*) AS Patients
FROM Admit NATURAL JOIN Citizen
WHERE Citizen.Division_Name ='DHAKA'
GROUP_BY Hospital_Name
ORDER BY Patients DESC)
WHERE ROWNUM<=1;
```

Using the sub-query I selected the hospitals which admitted patients from Dhaka, arranged the records in descending order and then using ROWNUM selected the top record.

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

Query

```
SELECT NAME
FROM Accident NATURAL JOIN License NATURAL JOIN Citizen
WHERE Accident.District_Name != Citizen.District_Name;
```

I joined accident ,license and citizen to identify the record where the location of the accident is not the same as the citizen's district.