Islamic University of Technology
Lab Task 07
CSE 4308 - DBMS Lab

Submitted To :

MD Bakhtiar Hasan
Ast. Professor, CSE Department
Islamic University of Technology

Zannatun Naim Sristy
Lecturer, CSE Department
Islamic University of Technology

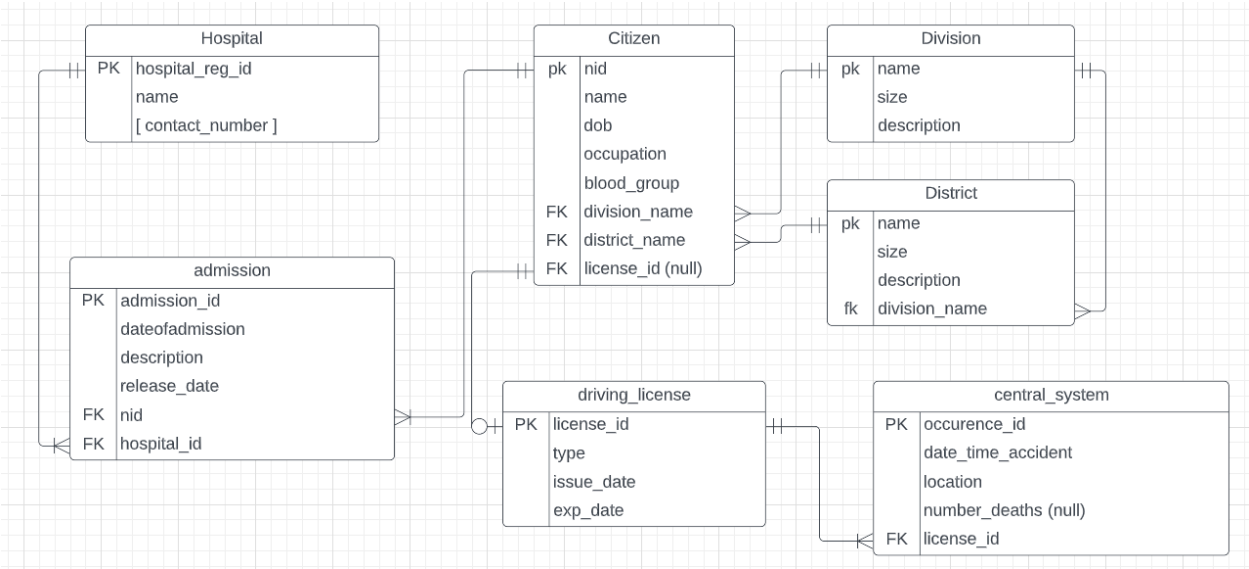
Submitted By :

Sian Ashsad
ID : 200042151
Prog. : SWE
Dept. : CSE

Report :

Analysis : In this task, we need to draw an ER diagram with appropriate cardinality and negating data redundancy. Afterwards, the ER diagram has to be converted into DDL using standard SQL denoting the appropriate constraints. Finally, given queries need to be performed on the tables.

ER Diagram :



DDL :

```
create table Citizen(
    nid INT,
    name VARCHAR2(100),
    dob DATE,
    occupation VARCHAR2(50),
    blood_group VARCHAR2(3),
    division_name VARCHAR2(100),
    district_name VARCHAR2(100),
    license_id INT NULL
    CONSTRAINT pk_nid primary key (nid)
```

```
    CONSTRAINT fk_division FOREIGN KEY(division_name) references
Division(name)
    CONSTRAINT fk_district FOREIGN KEY(district_name) references
District(name)
    CONSTRAINT fk_division FOREIGN KEY(license_id) references
License(license_id)
);


create table License(
    license_id INT,
    type VARCHAR2(20),
    issue_date DATE,
    exp_date DATE

    CONSTRAINT pk_license_id primary key (license_id)
);


create table Division(
    name VARCHAR2(100),
    size INT,
    description VARCHAR2(2000)

    constraint pk_div_name primary key (name)
);

create table District(
    name VARCHAR2(100),
    size INT,
    description VARCHAR2(2000),
    division_name VARCHAR2(100)

    CONSTRAINT pk_dis_name primary key(name)
    CONSTRAINT fk_div_name foreign key (division_name) references
Division(name)
);


create table central_system(
```

```
    occurence_id VARCHAR2(20),
    date_time_accident DATE,
    LOCATION VARCHAR2(100),
    number_deaths INT NULL,
    license_id INT
    CONSTRAINT pk_occ_id primary key (occurence_id)
    CONSTRAINT fk_license_id foreign key(license_id) references
License(license_id)

);

CREATE TABLE Admission(
    admission_id VARCHAR2(20),
    dateofadmission DATE,
    description VARCHAR2(2000),
    release_date DATE,
    nid INT,
    hospital_id INT

    CONSTRAINT pk_admission_id primary key(admission_id)
    CONSTRAINT fk_nid foreign key (nid) references Citizen(nid)
    CONSTRAINT fk_hospital_id FOREIGN KEY (hospital_id) references
Hospital(hospital_reg_id)
);

create or replace TYPE vcontact AS VARRAY(10) OF VARCHAR2(11);


create table Hospital(
    hospital_reg_id INT,
    name VARCHAR2(100),
    contact_number vcontact

    CONSTRAINT pk_hospital_id PRIMARY key (hospital_reg_id)

);
```

Explanation of solution : The ER diagram has 7 tables.

1. Citizen table :
    a. Primary key : nid
    b. Foreign key :division_name from Division table, district_name form District table and license_id from License table.
    c. Relationships :
        i. One to one : License table
        ii. One to many : Admission, Division, District table
        iii. Many to many : None
2. License table :
    a. Primary key : license_id
    b. Foreign key : None
    c. Relationships :
        i. One to one : Citizen table
        ii. One to many : central_system table
        iii. Many to many : None
3. Division table :
    a. Primary key : name
    b. Foreign key : None
    c. Relationships :
        i. One to one : None
        ii. One to many : District and Citizen table
        iii. Many to many : None
4. District table :
    a. Primary key : name
    b. Foreign key : division_name from Division table
    c. Relationships :
        i. One to one : None
        ii. One to many : Division and Citizen table
        iii. Many to many : None
5. Central_system table :
    a. Primary key : occurrence_id
    b. Foreign key : license_id form License table
    c. Relationships :
        i. One to one : None
        ii. One to many : License table
        iii. Many to many : None

6. Admission table :
    a. Primary key : admission_id
    b. Foreign key : nid from Citizen table and hospital_id from Hospital table
    c. Relationships :
        i. One to one : None
        ii. One to many : Hospital and Citizen table
        iii. Many to many : None
7. Hospital table :
    a. Primary key : hospital_reg_id
    b. Foreign key : None.
    c. Relationships :
        i. One to one : None
        ii. One to many : Admission table
        iii. Many to many : None

Queries :

```
--> a <--

SELECT D.NAME, COUNT(DS.NAME)
FROM DIVISION D NATURAL JOIN DISTRICT DS
GROUPBY D.NAME;

--> b <--
SELECT C.DISTRICT, COUNT(C.NID)
FROM CITIZEN C NATURAL JOIN DISTRICT D
HAVING COUNT(C.NID) >= 20000
GROUPBY C.DISTRICT;

--> c <--
SELECT COUNT(OCCURENCE_ID)
FROM CENTRAL_SYSTEM CS, CITIZEN C, LICENSE L
WHERE C.NID = "210" AND C.LICENSE_ID = L.LICENSE_ID AND D.LICENSE_ID =
CS.LICENSE_ID;

--> d <--
SELECT ROWNUM AS RANK, H.NAME
FROM(SELECT H.NAME, COUNT(A.ADMISSION_ID)
     FROM HOSPITAL H, ADMISSION A
```

```
     WHERE H.HOSPITAL_REG_ID = A.HOSPITAL_ID
     GROUPBY H.NAME
     ORDERBY COUNT(A.ADMISSION_ID) DESC)
WHERE ROWNUM <= 5;

--> e <--
SELECT C.BLOOD_GROUP
FROM CITIZEN C, HOSPITAL H, ADMISSION A
WHERE C.NID = A.NID AND A.NID = H.NID;

--> f <--
SELECT D.NAME, (TOTAL/D.SIZE)
FROM(SELECT D.NAME, COUNT(C.NID) AS TOTAL
     FROM DIVISION D, CITIZEN C
     WHERE C.DIVISION_NAME = D.NAME
     GROUPBY D.NAME);

--> g <--
SELECT ROWNUM AS RANK, D.NAME
FROM(SELECT D.NAME, (TOTAL/D.SIZE)
     FROM(SELECT D.NAME, COUNT(C.NID) AS TOTAL
     FROM DIVISION D, CITIZEN C
     WHERE C.DIVISION_NAME = D.NAME
     GROUPBY D.NAME))
WHERE ROWNUM <= 3;

--> h <--
SELECT DS.NAME, COUNT(CS.OCCURENCE_ID)
FROM DISTRICT DS, CITIZEN C, LICENSE L, CENTRAL_SYSTEM CS
WHERE DS.NAME = C.DISTRICT AND C.LICENSE_ID = D.LICENSE_ID AND
D.LICENSE_ID = CS.LICENSE_ID
GROUPBY DS.NAME;

--> i <--
SELECT D.NAME
FROM(SELECT D.NAME, COUNT(CS.OCCURENCE_ID) AS OCCURENCE
        FROM DIVISION D, CITIZEN C, LICENSE L, CENTRAL_SYSTEM CS
        WHERE D.NAME = C.DIVISION AND C.LICESNSE_ID = L.LICENSE_ID AND
L.LICENSE_ID=CS.LICENSE_ID
        GROUPBY D.NAME
```

```
        ORDERBY OCCURENCE ASC)
WHERE ROWNUM <= 1;

--> j <--
SELECT COUNT(CS.OCCURENCE_ID)
FROM LICENSE L, CENTRAL_SYSTEM CS
WHERE L.TYPE = "PROFESSIONAL" AND L.TYPE = "NON-PROFESSIONAL";

--> k <--
SELECT C.NAME
FROM(SELECT C.NAME, (A.RELEASE_DATE - A.DATEOFADMISSION) AS TIME
    FROM CITIZEN C, ADMISSION A
    WHERE C.NID = A.NID
    ORDERBY TIME DESC)
WHERE ROWNUM <= 1;

--> l <--
SELECT D.NAME
FROM(SELECT D.NAME, COUNT(C.NID) AS TOTAL
    FROM DIVISION D, CITIZEN C
    WHERE D.NAME=C.DIVISION AND GETDATE()-C.DOB>= 15 AND GETDATE()-C.DOB
<= 30
    ORDERBY TOTAL)
WHERE ROWNUM <= 1;

--> m <--
SELECT C.NAME
FROM CITIZEN C, LICENSE L
WHERE C.LICENSE_ID = D.LICENSE_ID AND GETDATE()>D.EXP_DATE;

--> n <--
SELECT C.NAME, COUNT(CS.OCCURENCE_ID)
FROM CITIZEN C, LICENSE L, CENTRAL_SYSTEM CS
WHERE C.LICENSE_ID = D.LICENSE_ID AND D.LICENSE_ID = CS.LICENSE_ID AND
GETDATE()>D.EXP_DATE;

--> o <--
SELECT C.NAME
FROM CITIZEN C, LICENSE L
WHERE C.LICENSE_ID = D.LICENSE_ID;
```

```
MINUS
SELECT C.NAME
FROM CITIZEN C, LICENSE L, CENTRAL_SYSTEM CS
WHERE C.LICENSE_ID = D.LICENSE_ID AND D.LICENSE_ID = CS.LICENSE_ID;

--> p <--
SELECT D.NAME, CS.NUMBER_DEATHS
FROM DIVISION D, CITIZEN C, LICENSE L, CENTRAL_SYSTEM CS
WHERE D.NAME = C.DIVISION AND C.LICENSE_ID = L.LICENSE_ID AND L.LICENSE_ID
= CS.LICENSE_ID;

--> q <--
SELECT C.NAME
FROM CITIZEN C, LICENSE L
WHERE (L.ISSUE_DATE - GETDATE())<=22 OR (L.ISSUE_DATE - GETDATE())>=40;

--> r <--
SELECT C.NAME
FROM CITIZEN C, LICENSE L, CENTRAL_SYSTEM CS, ADMISSION A
WHERE C.LICENSE_ID = L.LICENSE_ID AND L.LICENSE_ID = CS.LICENSE_ID AND
C.NID = A.NID AND CS.DATE_TIME_ACCIDENT = A.DATEOFADMISSION;

--> s <--
SELECT H.NAME
FROM(SELECT H.NAME, COUNT(C.NID) AS TOTAL
     FROM HOSPITAL H, ADMISSION A, CITIZEN C
     WHERE C.DIVISION_NAME = "DHAKA" AND C.NID = A.NID AND A.HOSPITAL_ID =
H.HOSPITAL_REG_ID
     GROUPBY H.NAME
     ORDERBY TOTAL DESC)
WHERE ROWNUM <= 1;

--> t <--
SELECT C.NAME
FROM CITIZEN C, LICENSE L, CENTRAL_SYSTEM CS
WHERE C.LICENSE_ID = L.LICENSE_ID AND L.LICENSE_ID = CS.LICENSE_ID
MINUS
SELECT C.NAME
FROM CITIZEN C, LICENSE L, CENTRAL_SYSTEM CS
```

```
WHERE C.LICENSE_ID = L.LICENSE_ID AND L.LICENSE_ID = CS.LICENSE_ID AND
C.DISTRICT_NAME = CS.LOCATION;
```

a) Select names of divisions and the count of the number of districts using the count aggregate function.
b) Select names of districts and the count of the number of citizens in each district using the count aggregate function. Using the having clause so that the query shows only the districts with at least 20000 living in it.
c) Shows the number of accidents that involved a citizen whose NID is 210 using the count aggregate function.
d) Uses the rownum function to show the top 5 hospitals with the most admitted patients from a nested query.
e) Shows the blood group of all the patients admitted to different hospitals.
f) Shows the population density of each division by finding out the total number of citizens in that division and dividing it by the size of the division through nested queries and count aggregate function.
g) Similar to (f) but uses rownum to find the top 3.
h) Uses the count aggregate function to find out the number of accidents that occurred in each district.
i) Same as (h) but user rownum and orderby ASC to find the required division.
j) Uses the count aggregate function to complete this query.
k) Finds out the person who was in a hospital for the longest time through a nested query. Here the attributes, release date and date of admission were subtracted to find the time period of stay. Then orderby DESC and rownum functions were used to complete the query.
l) Uses the getdate() function to find out the age of a citizen by subtracting it with DOB. Afterwards, necessary operations were performed to find out the required division. Count aggregate function and rownum were used.
m) Uses the getdate() function to find out required citizens by comparing it to the expiration date of license.
n) Same as (m) only an extra column is shown using the count aggregate function. This shows the accidents occurred by citizens with expired licenses.
o) Uses the minus operator to rule out the citizens who were not involved in any accidents so far.
p) Shows the number of deaths due to any accident for each division.
q) Similar to (l), uses the getdate() function for the age comparison.
r) The query checks if the date of accident is the same as the date of admission.

s) Nested query shows the names of hospitals with the most citizens from Dhaka district in a descending order, using the count aggregate function and orderby DESC function. The rownum function is used to find complete the query
t) Uses the Minus operator to complete the query by removing all the citizens who had an accident in their own district from citizens who were in an accident.

Findings : Visualizing different relationships between entities is the key to making a successful ER diagram. Fundamental knowledge of cardinality and data redundancy is a must for this purpose. In case of implementation, knowing the correct syntax for DDL and appropriate constraints is also imperative.
Solving given queries also required the need of certain functions which had to be learned in order to solve the problem. The getdate() function was one of those functions.

Problems : Figuring out the schema of the database proved to be a challenge. While drawing the ER diagram, cardinality of certain tables were difficult to apprehend. Checking data redundancy was also a stone in the way. Finding out the validation of the queries stood out to be a hindrance since inputting data in each table manually after performing sufficient DDL is very scrutinous. Therefore, some queries were left unvalidated relying purely on logical intuition.