Islamic University of Technology

Department of Computer Science

Database Management Systems II Lab

Course Code- CSE 4410

Prepared By:
Nazia Krim Khan Oishee
ID: 200042137

Date: January 6, 2023

Contents

1 Introduction

In our first lab of DBMS II we were to convert a given scenario into DDL and to write SQL DML statements for given queries. The purpose behind these tasks were to revisit the concepts of entity relation and cardinality from DBMS I.

2 Task1

Explanation

We were given a scenario regarding food chain. Franchises such as KFC, Chester's, Pizza hut, Domino's Pizza have multiple branches and customers all around the world.

At first,I created an entity 'Franchise' containing one attribute 'Franchise_Name'.A franchise can have many branches and a branch organization can belong to only one franchise. So Franchise and Branch have One to Many relation.

A franchise have many customers and each customer can order food from multiple branches of multiple franchises.So customers and branches have many to many relation. I have denoted this relation as 'Preferences'.

Each branch of a particular franchise has their own team of chef. A chef is associated to only one branch of a particular franchise. So 'Chef' has one to many relation with both-'Branches' and 'Franchise'.

A chef can prepare multiple menus of various cuisines. Similarly a menu can be prepared by multiple chefs. So chefs and'.

menu have many to many relation.I have denoted this relation as 'Menu_PreparedBy'.

A menu can be offered by multiple franchises and a franchise has multiple menus. So they also have many to many relation which I denoted as 'Menu_OfferedBy'.

Now, a customer can order multiple menus from multiple franchises. An order can be distinguished by its order_id. An order can be placed by only one customer for only one menu from only one franchise. I have denoted this relation by 'ordertable'.

Lastly, a customer can rate a menu from a franchise. A particular menu from a franchise can be reviewed by many customers. I have denoted this relation as as 'Rating'.

I have inserted necessary attributes in all entities. I have introduced foreign key in many part of one to many relation.

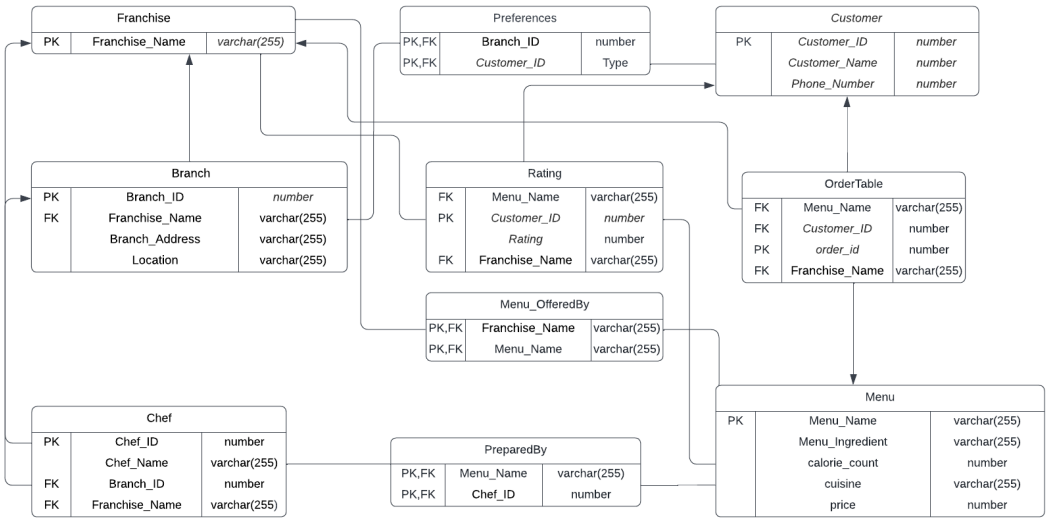

Figure 2.1: ER Diagram

Code Snippet

```
create table Franchise(
Franchise_Name varchar(255),
CONSTRAINT PK_Franchise PRIMARY KEY(Franchise_Name)
);
create table Customer(
    Customer_ID number,
    Phone_Number number,
    Customer_Name varchar(255),
    CONSTRAINT PK_Customer PRIMARY KEY(Customer_ID)
);
create table Branch (
    Branch_ID number ,
    Franchise_Name varchar(255),
    Branch_Address varchar(255),
    Location varchar(255),
    CONSTRAINT PK_Branch PRIMARY KEY(Branch_ID),
    CONSTRAINT FK_Branch Foreign KEY(Franchise_Name) references Franchise(
   Franchise_Name)
);
create table Preferences (
    Branch_ID number ,
    Customer_ID number,
CONSTRAINT PK_Preferences1 PRIMARY KEY(Branch_ID,Customer_ID),
    CONSTRAINT FK_Preferences1 Foreign KEY(Branch_ID) references Branch(
   Branch_ID),
    CONSTRAINT FK_Preferences2 Foreign KEY(Customer_ID) references Customer(
   Customer_ID)
);
create table Chef(
Chef_ID integer,
Chef_Name varchar(255),
Branch_ID integer,
Franchise_Name varchar(255),
```

```
Cuisine varchar(255),
CONSTRAINT PK_Chef PRIMARY KEY(Chef_ID),
CONSTRAINT FK_Chef1 Foreign KEY(Franchise_Name) references Franchise(Franchise_Name),
CONSTRAINT FK_Chef2 Foreign KEY(Branch_ID) references Branch(Branch_ID)
);
create table Menu(
    Menu_Name VARCHAR(255),
    Main_Ingredient VARCHAR(255),
    Cuisine varchar(255),
    price varchar(255),
    calorie_count integer,
    CONSTRAINT PK_Menu PRIMARY KEY (Menu_Name)
);
create table ordertable(
    order_id number,
        Customer_ID number,
     Menu_Name VARCHAR(255),
     Franchise_Name varchar(200),
     CONSTRAINT PK_order1 PRIMARY KEY (order_id),
        CONSTRAINT FK_order1 Foreign KEY(Menu_Name) references Menu(Menu_Name),
      CONSTRAINT FK_order2 Foreign KEY(Franchise_Name) references Franchise(Franchise_Name),
        CONSTRAINT FK_order3 Foreign KEY(Customer_ID) references Customer(Customer_ID)
);
create table Menu_PreparedBy(
        Menu_Name VARCHAR(255),
        Chef_ID integer,
        Franchise_Name varchar(255),
        CONSTRAINT PK_Menu_PreparedBy1 PRIMARY KEY(Chef_ID,Menu_Name),
        CONSTRAINT FK_Menu_PreparedBy1 Foreign KEY(Chef_ID) references Chef(Chef_ID),
        CONSTRAINT FK_Menu_PreparedBy2 Foreign KEY(Menu_Name) references Menu(Menu_Name),
        CONSTRAINT FK_Menu_PreparedBy3 Foreign KEY(Franchise_Name) references Franchise(Franchise_Name)
);
create table Rating(
        Menu_Name VARCHAR(255),
        Franchise_Name varchar(255),
       Customer_ID number,
       Rating number,
     CONSTRAINT PK_Rating PRIMARY KEY (Customer_ID),
    CONSTRAINT FK_Rating1 Foreign KEY(Menu_Name) references Menu(Menu_Name),
    CONSTRAINT FK_Rating2 Foreign KEY(Franchise_Name) references Francise(Franchise_Name)
);
create table Menu_OfferedBy(
        Menu_Name VARCHAR(255),
        Franchise_Name varchar(255),
     CONSTRAINT PK_MenuOfferedBy PRIMARY KEY (Menu_Name,Franchise_Name),
CONSTRAINT FK_Menu_OfferedBy1 Foreign KEY(Menu_Name) references Menu(Menu_Name),
    CONSTRAINT FK_Menu_OfferedBy2 Foreign KEY(Franchise_Name) references Franchise_Name(Franchise_Name)
);
```

3 Task2

Our 2nd task was to write SQL statements for the given queries.

a. Find the total number of customers for each franchise.

Solution

To count the total number of customers for each franchise, I joined the Preferences and Branch tables.Preferences has Branch_ID,Customer_ID and Branch has Franchise_Name. Firstly I divided the customers under Franchise_Name and then count the number of customers for each Franchise.

Code Snippet

```
Select count(*) ,Franchise_Name
from Preferences natural join Branch
group by Franchise_Name;
```

b. Find the avg rating for each menu item among all franchises.

Solution

For this query, I simply calculated the average rating for each Menu using the avg aggregate function from Rating table.

Code Snippet

```
 Select avg(Rating),Menu_Name
from Rating
group by Menu_Name;
```

c. Find the 5 top most popular items. It should be based on the number of times they were ordered.

Solution

At first I organized the Menus offered by franchises based on the number of orders in a descending order. Then I selected top 5 menus from the list using ROWNUM.

Code Snippet

```
Select * from
Menu
where Menu_Name IN (
Select Menu_Name from(
Select count(*) as order_count,Menu_Name,Franchise_Name
from ordertable
group by (Menu_Name,Franchise_Name)
order by order_count desc)
where ROWNUM <=5);
```

d. Find the names of all customers who have preferred food that is offered from at least 2 different franchises.

Solution

I selected those Menu which are offered by at least two franchises from Menu_OfferedBy and then from ordertable entities selected those customers who ordered them and finally selected their name from Customer entity.

Code Snippet

```
Select customer_name
from customer
where customer_id IN(
select customer_id
from ordertable
where Menu_Name IN(
Select count(*) as count_menu,Menu_Name
```

```
from Menu_OfferedBy
groupby (Franchise_Name)
having count(*)>=2));
```

e. Find the names of all customers who have not placed any orders.

Solution

I selected only those customer_name who were not present in ordertable which implies they did not place any orders.

Code Snippet

```
Select customer\_name
from customer
where customer_id NOT IN(
select customer_id
from ordertable
);
```

4 Problems I faced

The major problem I faced was figuring out the ERD. I made some mistakes at first such as I did not include the Menu_OfferedBy table. It took me quite some time to understand the whole scenario. But due to the fruitful discussion in the class I could understand the scenario and realized my mistakes. Later I improved my ERD. Due to not designing the ERD correctly, I at first made some mistakes in writing SQL statements correctly too. After mending the ERD,I could write the SQL statements again.