Islamic University of Technology

Lab-1 Report: Relational Model

Submitted by:

Name : Mashrur Ahsan

ID 200042115

Program : SWE

Department : CSE

Course Code : CSE 4410

The Scenario:

Suppose you are given the task of automating the operations of an international food chain via a single platform. There are multiple franchises (KFC, Chester's, Pizza hut, Domino's Pizza) of the food chain spread across over 20 countries. Each of the franchises gets at least 10,0000 customers per year.
Customers can register themselves under different franchises to order food from different branches of that specific franchise. Each franchise has multiple branches spread around the country. Each branch has its own team of chefs. Each of them is an expert in a particular cuisine. And any customer can see the basic information of the chefs such as special menus developed by them (up to 5 menus). Each franchise has multiple menus (some are unique and some are common) that they offer to customers. The menus are identified by their own name, cuisine, main ingredients(optional), price, calorie_count etc.
Further to build a proper food recommendation system for the food chain, information about customer details, their personal preferred cuisines (a person can have multiple preferred ones) and customers' ratings of any food need to be stored.

Problem Statements

1. Convert the scenario into DDL using standard SQL denoting the appropriate constraints.

Here's the "franchise", "customer" and the "franchise_customer_list" table. The last has two attributes which are acting as a single primary key.

```
create table franchise
(
    f_name varchar2(50),
    CONSTRAINT PK_FRANCHISE PRIMARY KEY (f_name)
);


create table customer
(
    c_id varchar2(50),
    CONSTRAINT PK_CUSTOMER PRIMARY KEY (c_id)
);


create table franchise_customer_list
(
    f_name varchar2(50),
    c_id varchar2(50),
    CONSTRAINT PK_FRANCHISE_CUSTOMER_LIST PRIMARY KEY (f_name, c_id)
);
```

These are the “menu” and the “rating” table.

```
create table menu
(
    f_name varchar2(50),
    menu_name varchar2(50),
    CONSTRAINT FK_MENU FOREIGN KEY (f_name) REFERENCES franchise(f_name),
    CONSTRAINT PK_MENU PRIMARY KEY (f_name, menu_name)
);


create table Rating
(
    f_name varchar2(50),
    menu_name varchar2(50),
    menu_rating number,
    c_id varchar2(50),
    CONSTRAINT FK_RATING FOREIGN KEY (f_name, menu_name) REFERENCES menu(f_name, menu_name),
    CONSTRAINT PK_RATING PRIMARY KEY (c_id, f_name, menu_name)
);
```

And these are the “orders” and the “preference” table.

```
create table orders
(
    order_id varchar2(50),
    c_id varchar2(50),
    f_name varchar2(50),
    menu_name varchar2(50),
    CONSTRAINT FK_ORDER_CUSTOMER FOREIGN KEY (c_id) REFERENCES customer(c_id),
    CONSTRAINT FK_ORDER_MENU FOREIGN KEY (f_name, menu_name) REFERENCES menu(f_name, menu_name),
    CONSTRAINT PK_ORDER PRIMARY KEY (order_id)
);


create table preference
(
    c_id varchar2(50),
    prefered_food varchar2(50),
    CONSTRAINT PK_PREFERENCE FOREIGN KEY (c_id) REFERENCES customer(c_id)
);
```

Explanation and Analysis:

The “customer” entity will contain the customer details, customer_id being the primary key. The “franchise” table will contain information about the different franchises where the name of the franchise is the primary key.

There’s another entity that stores the data regarding the customers going to different franchises. One customer can order food from multiple franchises. Whenever someone orders a food from a food store, they will be enlisted in the table. So, the table will store the customer ids and the store’s name from where they are purchasing the food from.

The “menu” table stores the data about a cuisine or a food item from different stores. Multiple stores can offer the same food item or cuisine. To avoid inconsistent data the franchise name and the menu name together will act as the primary for this entity.

The “rating” table stores data regarding a food item’s rating and which customer gave it. It also stores the data regarding which food store offered that food to that customer. Therefore, the food store name and the menu name will be referencing the same attributes from the “menu” table. And the customer ids, franchise name and the menu name altogether will act as the primary key.

Now we come the “orders” table, which stores the orders from customers. A customer can order any food item however they like. They can order it from multiple franchises or order numerous different food items from different stores as many times as they want. To keep track of the orders there’s an order id. Which will act as the primary key. The other attributes will reference “menu” and the “customer” tables’ attributes.

Lastly, we have the “preference” table. This table will store data regarding a customer’s preference. Which item a customer prefers to eat. This table will contain a foreign key that references the customer ids. Then it will store that customer’s preferred food.

Problems Faced:

Properly understanding the whole scenario and deducing the entities from the scenario was a lengthy process. It was also crucial to find the relative relations between the entities, which was a bit time consuming. Other than that it was pretty straightforward.

Now we come to the queries.

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

The Solution:

```
-- a --
select count(*), f_name from franchise_customer_list
Group by f_name;
```

Explanation & Analysis:
We are using the “count” aggregative function to get the total number of customers for every franchise. It’s getting done with the help of “group by” keyword. Where we are grouping the number of customers according to the franchise name.

Problems Faced:
No problems faced.

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

The Solution:

```
-- b --
select avg(menu_rating),  menu_name
from rating
group by menu_name;
```

Explanation & Analysis:
Again, another aggregate function is being used here. The “agv” function is calculating the average rating for a particular menu item across all franchises. And, it’s been grouped by “menu_name”.

Problems Faced:
No problems faced.

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

The Solution:

```
-- c --
select ROWNUM as times_ordered, menu_name
from(select count(order_id), menu_name
from orders
group by menu_name
order by count(order_id) DESC)
where ROWNUM<=5;
```

Explanation & Analysis:
First we are sorting the menu names according to the number of orders placed. It will be sorted in a descending order. Then we are using the “ROWNUM” keyword to get the top 5 most ordered items (with the help of a condition at the end).

Problems Faced:
No problems faced.

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

The Solution:

```
-- d --
SELECT c_id from
(select menu_name from
 (select count(menu_name) as times, menu_name from menu
   group by menu_name)
where times>=2), preference
where menu_name=prefered_food;
```

Explanation & Analysis:

Here the concept of sub query was used since we know all that the result of a query is also a relation. If the number of menu name is more than 2 in the "menu" table, it indicates that the item is being offered in more than two franchises.

Therefore, at first we are selecting only those menu names and matching them with the customer's preferred foods. If there's a match then we are showing the customer ids. Although the question asked us to give customer names but we can find those in a similar way. I only provided the ids because it is now quite difficult for me to change the code for insertions. (The validation of the queries were checked with the help of data insertions into the database)

Problems Faced:

No problems faced.

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

The Solution:

```
-- e --
select Distinct customer.c_id
from Orders, customer
where customer.c_id
    not in (select c_id from orders);
```

Explanation & Analysis:

Here we are going through the "customer" table and the "orders" table and selecting only those customers who doesn't have their ids in the "orders" table. That would mean that, those customers didn't place any orders.

Again, although the question asked us to give customer names but we can find those in a similar way. I only provided the ids because it is now quite difficult for me to change the code for insertions. (The validation of the queries were checked with the help of data insertions into the database)

Problems Faced:

No problems faced.

Insertion Codes:

```
INSERT INTO franchise values ('Dominoes');
INSERT INTO franchise values ('Pizza Hut');
INSERT INTO franchise values ('Pizza Inn');


INSERT INTO customer values ('1');
INSERT INTO customer values ('2');
INSERT INTO customer values ('3');
INSERT INTO customer values ('4');
INSERT INTO customer values ('5');


INSERT INTO franchise_customer_list values ('Dominoes', '1');
INSERT INTO franchise_customer_list values ('Dominoes', '2');
INSERT INTO franchise_customer_list values ('Dominoes', '3');
INSERT INTO franchise_customer_list values ('Dominoes', '4');
INSERT INTO franchise_customer_list values ('Pizza Hut', '1');
INSERT INTO franchise_customer_list values ('Pizza Hut', '4');
INSERT INTO franchise_customer_list values ('Pizza Inn', '1');
INSERT INTO franchise_customer_list values ('Pizza Inn', '5');



INSERT INTO menu values ('Dominoes', 'dessert');
INSERT INTO menu values ('Pizza Hut', 'fries');
INSERT INTO menu values ('Dominoes', 'fries');
INSERT INTO menu values ('Pizza Inn', 'hot dog');
```

```
INSERT INTO rating values ('Dominoes', 'dessert', '8', '1');
INSERT INTO rating values ('Dominoes', 'fries', '10', '2');
INSERT INTO rating values ('Dominoes', 'fries', '7', '3');
INSERT INTO rating values ('Dominoes', 'fries', '9', '4');
INSERT INTO rating values ('Pizza Hut', 'fries', '3', '1');
INSERT INTO rating values ('Pizza Hut', 'fries', '5', '4');
INSERT INTO rating values ('Pizza Inn', 'hot dog', '10', '1');
INSERT INTO rating values ('Pizza Inn', 'hot dog', '9', '5');




INSERT INTO Orders values ('1', '1', 'Dominoes', 'dessert');
INSERT INTO Orders values ('2', '4', 'Pizza Hut', 'fries');
INSERT INTO Orders values ('3', '3', 'Dominoes', 'fries');
INSERT INTO Orders values ('4', '1', 'Pizza Inn', 'hot dog');


INSERT INTO preference values ('1', 'fries');
INSERT INTO preference values ('2','fries');
INSERT INTO preference values ('3', 'hot dog');
```