Hacettepe University

Computer Science and Engineering Department

Name and Surname : Emin Orçun Sancar and Musa ALİŞAR

Identity Number : 21427306 and 21426567

Course : BBM 473 Data Management Systems Laboratory

Experiment : Project Phase-1

Subject : Online Food Ordering System ER Database Design

Data Due : 12.11.2018

Advisors : Research Teaching Assistant Aysun KOÇAK

Project Definition

We choose Online Food Ordering System project from samples in the website. This system is similar to Yemek Sepeti . We can easily choose which food to order from which restaurant in this system. It will allow us to find online restaurants from system.We can choose menu or only a food from system .Snacks are added to food in menu. Every restaurant has db admin,db admin can add delete update foods in the restaurant system.We can suggest some food or menu according to Customers orders.Every Customer has a membership type and according to membership type every Customer has his/her own discount. Customers are sorted by order number in their address.

Projects main functions

Order: Every Customer can order menu or food from online restaurants in the system.

Comment: Every Customer can comment on the restaurant or dinner which his/her want.

Discount: Every Customer have a discount according to his/her membership type.

Customer Payment: Every Customer can pay price of food/menu.

Sort by Address: Every Customer sort by his/her address according to total order number

Favorite food: System can suggest food to customer according to his/her order records.

Favorite menu: System can suggest menu to customer according to his/her order records.

Favorite restaurant: System can suggest restaurant to customer according to his/her order records.

Restaurant payment: Each restaurant pay some money to manager according to Restaurant Membership Type.

Restaurant Db Admin Manipulation: Each restaurant has one admin, and each db admin can add, delete, update foods, menu, snack in restaurant.

Db Admin Manipulation: Each db admin can add, delete, update informations about Manager and db operations.

The role of all entity sets and relations in the project

Tables and entities are defined as a diagram below

Food(fid:int not null pk, foodType:varchar(23) not null, foodName:varchar(23) not null, foodPrice:double not null, rid:int not null fk);

Restaurant(rid:int not null pk, restName:varchar(23) not null, restAccount:double not null, restaurantDBAdminID:int not null fk);

Customer(custID:int not null pk, custName:varchar(23) not null, custUsername:varchar(23) not null, custPassword:varchar(23) not null, custAccount:double not null, custPayID:int not null fk);

Order(orderID:int not null pk, paymentID:int not null fk, menuID:int not null fk, fid:int not null fk, rid:int not null fk, custID:int not null fk);

Menu(menuID:int not null pk, menuType:varchar(23) not null, menuName:varchar(23) not null, menuPrice:double not null, fid:int not null fk, snackID:int not null fk);

Snack(snackID:int not null pk, snackType:varchar(23) not null, snackName:varchar(23) not null, snackPrice:double not null);

Comments(commentID:int not null pk, custID:int not null fk, rid:int not null fk,);

CustomerMembership(custMemID:int not null pk, discount:double not null, custID:int not null fk);

CustomerGoldMember(goldMember:int not null pk, discountPer:double not null, custMemID:int not null fk);

CustomerSilverMember(silverMember:int not null pk, discountPer:double not null, custMemID:int not null fk);

CustomerBronzeMember(bronzeMember:int not null pk, discountPer:double not null, custMemID:int not null fk);

RestaurantDBAdmin(restDBAdminid:int not null pk, name:varchar(23) not null, username:varchar(23) not null, password:varchar(23) not null);

DBAdmin(dbAdminid:int not null pk, name:varchar(23) not null, username:varchar(23) not null, password:varchar(23) not null, managerID:int not null fk);

MyFavourites(favID:int not null pk, orderID:int not null fk);

RecentMovements(movID:int not null pk, orderID:int not null fk);

ScorBoard(scorboardID:int not null pk, orderID:int not null fk);

Manager(manID:int not null pk, manName:varchar(23) not null, manUsername:varchar(23) not null, manPassword:varchar(23) not null, manAccount:double not null, manPayID:int not null fk);

RestaurantMembership(restMemID:int not null pk, discount:double not null, restID:int not null fk);

Adress(adressID:int not null pk, telephone:varchar(23) not null, mail:varchar(23) not null, cityID:int not null fk);

City(cityID:int not null pk, cityName:varchar(23) not null, districtID:int not null fk);

District(districtID:int not null pk, districtName:varchar(23) not null);

RestaurantGoldMember(goldMember:int not null pk, discountPer:double not null, restMemID:int not null fk);

RestaurantSilverMember(silverMember:int not null pk, discountPer:double not null, restMemID:int not null fk);

RestaurantBronzeMember(bronzeMember:int not null pk, discountPer:double not null, restMemID:int not null fk);

CustomerPayment(customerPaymentID:int not null pk, custMemID:int not null pk);

RestaurantPayment(restaurantPaymentID:int not null pk, restMemID:int not null pk);

When creating a database, common sense dictates that we use separate tables for different types of entities. Some examples are: customers, orders, items, messages etc... But we also need to have relationships between these tables. For instance, customers make orders, and orders contain items. These relationships need to be represented in the database.

There are several types of database relationships. Today we are going to cover the following:

- One to One Relationships

- One to Many and Many to One Relationships

- Many to Many Relationships

One to One Relationships

Let's say you have a table for customers:

CUSTOMERS		
customer_id	**customer_name**	**customer_address**
101	John Doe	12 Main St., Houston TX 77001
102	Bruce Wayne	1007 Mountain Dr., Gotham NY 10286

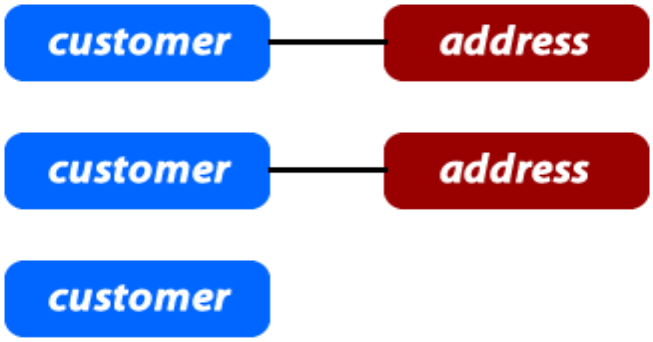
We can put the customer address information on a separate table:

CUSTOMERS		
customer_id	customer_name	address_id
101	John Doe	301
102	Bruce Wayne	302
ADDRESSES		
address_id	address	
301	12 Main St., Houston TX 77001	
302	1007 Mountain Dr., Gotham NY 10286	

Now we have a relationship between the Customers table and the Addresses table. If each address can belong to only one customer, this relationship is "One to One". Keep in mind that this kind of relationship is not very common. Our initial table that included the address along with the customer could have worked fine in most cases.

Notice that now there is a field named "address_id" in the Customers table, that refers to the matching record in the Address table. This is called a "Foreign Key" and it is used for all kinds of database relationships. We will cover this subject later in the article.

We can visualize the relationship between the customer and address records like this:



Note that the existence of a relationship can be optional, like having a customer record that has no related address record.

One to Many and Many to One Relationships

This is the most commonly used type of relationship. Consider an e-commerce website, with the following:

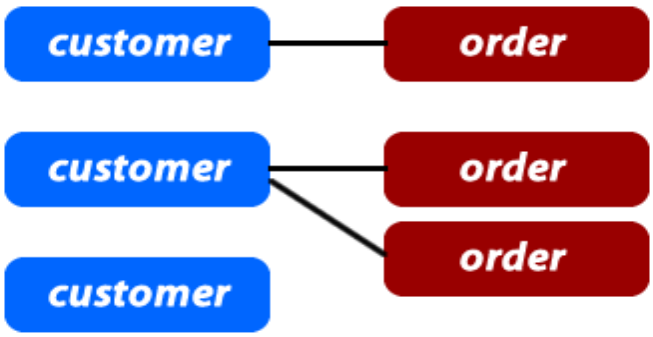
- Customers can make many orders.
- Orders can contain many items.
- Items can have descriptions in many languages.

In these cases we would need to create "One to Many" relationships. Here is an example:

CUSTOMERS	
customer_id	customer_name
101	John Doe
102	Bruce Wayne

ORDERS			
order_id	customer_id	order_date	amount
555	101	12/24/09	$156.78
556	102	12/25/09	$99.99
557	101	12/26/09	$75.00

Each customer may have zero, one or multiple orders. But an order can belong to only one customer.



Many to Many Relationships

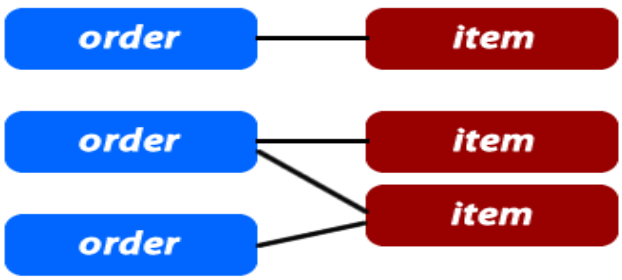
In some cases, you may need multiple instances on both sides of the relationship. For example, each order can contain multiple items. And each item can also be in multiple orders.

For these relationships, we need to create an extra table:

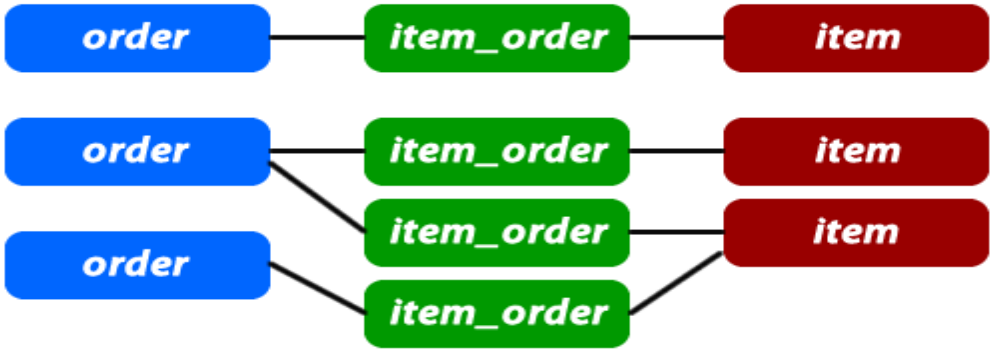
ORDERS			
order_id	customer_id	order_date	amount
555	101	12/24/09	$156.78
556	102	12/25/09	$99.99
ITEMS			
item_id	item_name	item_description	
201	Tickle Me Elmo	It wants to be tickled	
202	District 9 DVD	Awesome sci-fi movie	
203	Batarang	It is very sharp	
ITEMS_ORDERS			
order_id	item_id		
555	201		
555	202		
556	202		
556	203		

The Items_Orders table has only one purpose, and that is to create a "Many to Many" relationship between the items and the orders.

Here is a how we can visualize this kind of relationship:



If you want to include the items_orders records in the graph, it may look like this:



Foreign Keys

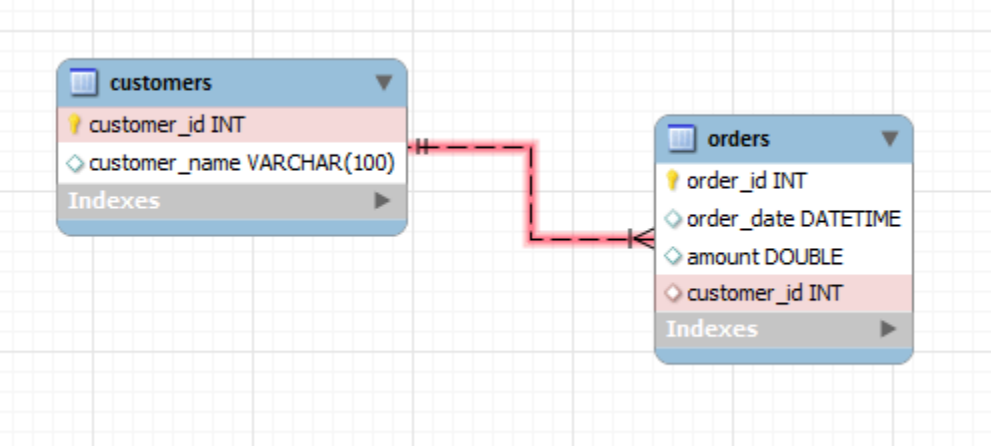
So far we have only learned about some of the concepts. Now it is time to bring them to life using SQL. For this part, we need to understand what Foreign Keys are.

In the relationship examples above, we always had these "****_id" fields that referenced a column in another table. In this example, the customer_id column in the Orders table is a Foreign Key column:

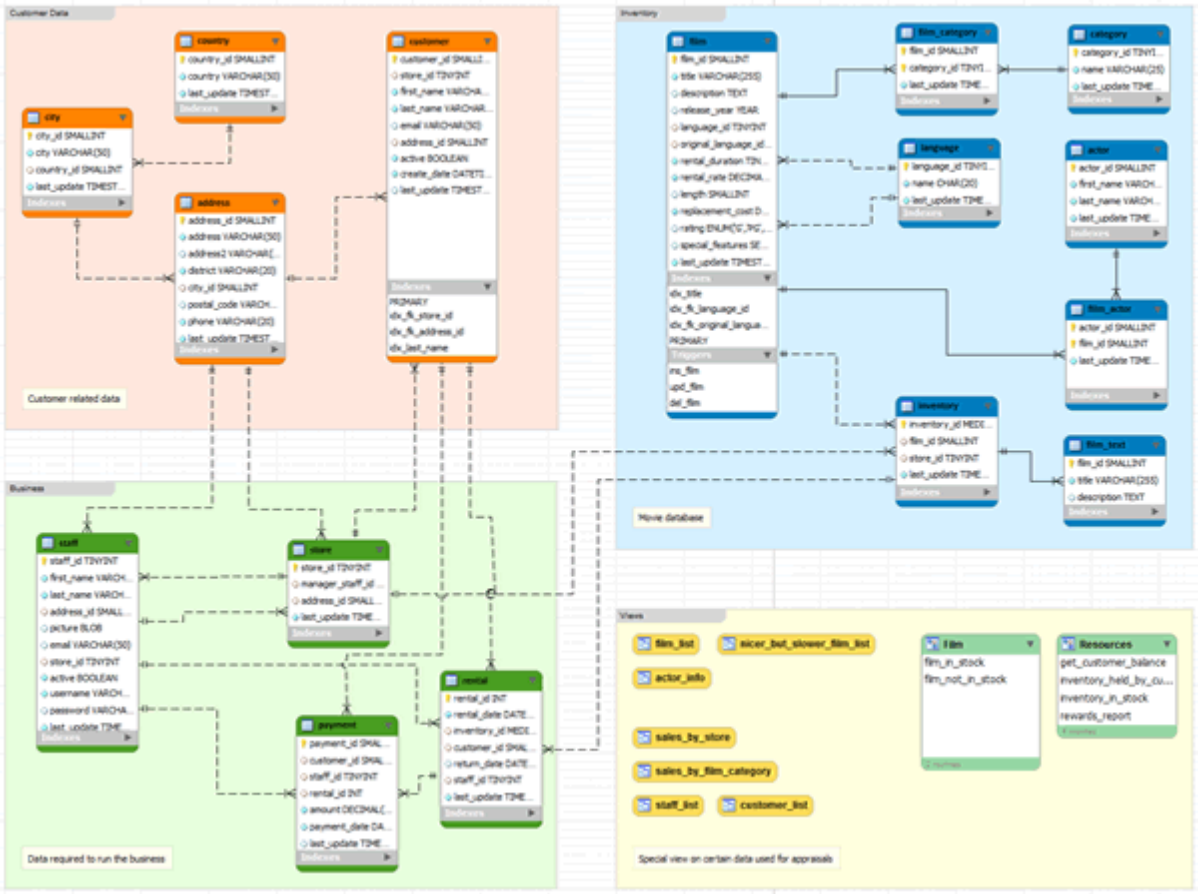
CUSTOMERS	
customer_id	customer_name
101	John Doe
102	Bruce Wayne

ORDERS			
order_id	customer_id	order_date	amount
555	101	12/24/09	$156.78
556	102	12/25/09	$99.99
557	101	12/26/09	$75.00

Visualizing the Relationships



Once you design your database, you can export the SQL and run it on your server. This comes in very handy for bigger and more complex database designs.



Relationships

-Each Restaurant cooks many food(one to many-by using food id)

-Each Restaurant has many Menu(one to many-by using menu id)

-Many Restaurant buy many Orders(many to many)

-Each Restaurant has one address(one to one)

-Many Restaurant gets many comments(many to many)

- Many Order has many menu(many to many)

- Many Order has many food(many to many)

-Each Customer order many Orders(many to one-customerID)

-Each Order has only one CustomerPayment(exactly one)

- Customer favourites finds according to Orders(one to many-customerID)

-RecentMoves lists according to Orders(one to many-orderID)

-Scorboard lists according to Orders(one to many-orderID)

-Each Order has one adress(exactly one)

-Many menu has many food(food weak entity)

-Each Customer comment many Comment(one to many-customerID)

-Each Customer has one CustomerMembership(one to one)

- Each Customer has one adress(one to one)

-Each Customer has one CustomerPayment(one to one)

-Adress has 2 subclasses such as City an District(2 Subclasses isA relation)

- Restaurant Db admin has one adress(one to one)

-Many Menu contains many snack(many to many)

-CustomerMembership has 3 subclasses such as Gold,Silver,Bronze(3 subclasses has one descriptive atrribute as discountPer)

-CustomerMembership has one CustomerPayment(one to one)

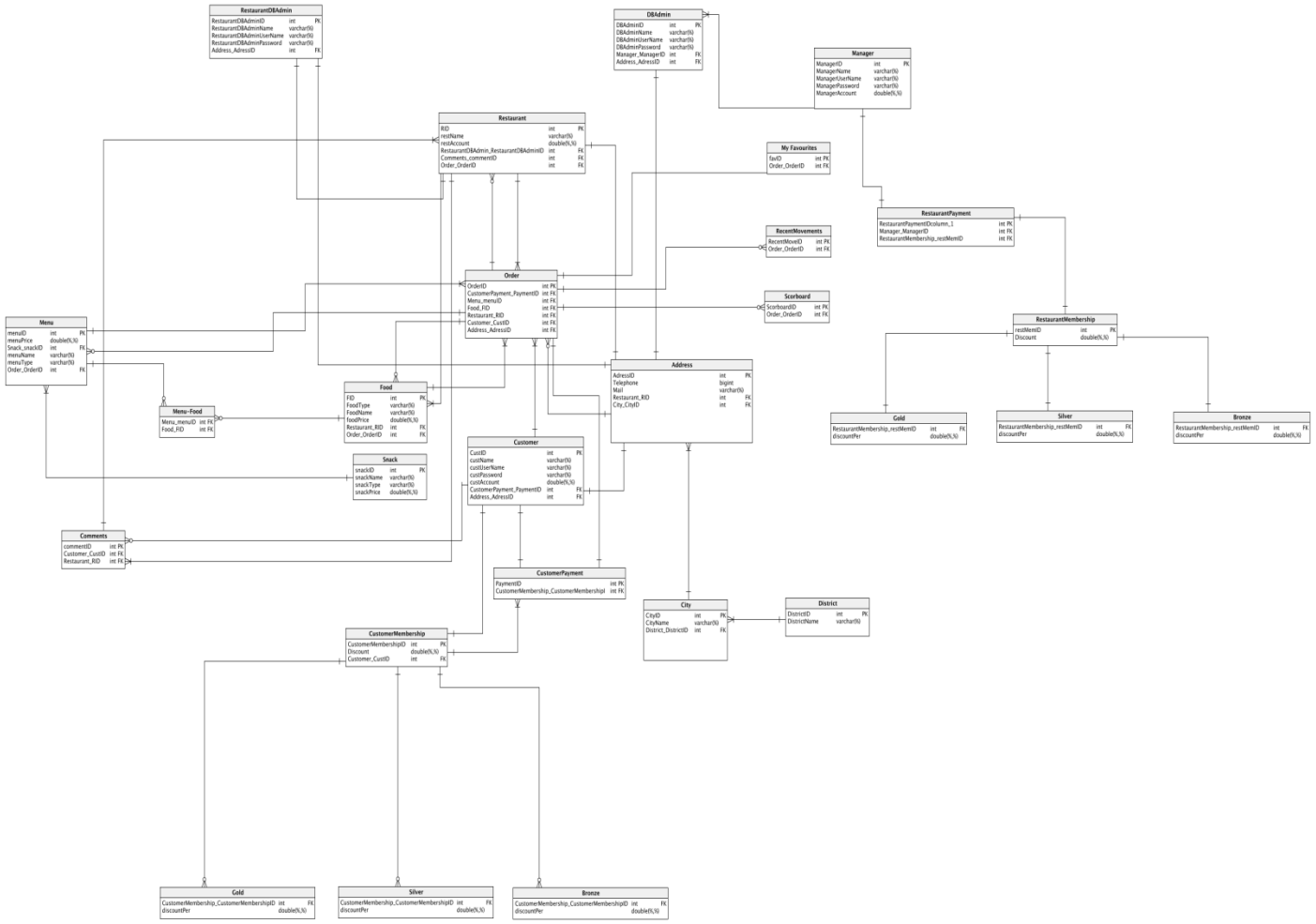
-RestaurantMembership has 3 subclasses such as Gold,Silver,Bronze(3 subclasses has one descriptive atrribute as discountPer)

-RestaurantMembership has one RestaurantPayment(one to one)

-One db admin create many manager.(one to many-managerID)

-Each Manager follows one restaurantPayment(one to one)

ER Diagram



Vertabelo

References

1)https://database.guide/the-3-types-of-relationships-in-database-design/

2) https://www.techrepublic.com/article/relational-databases-defining-relationships-between-database-tables/

3) https://www.quora.com/What-are-the-types-of-relationships-in-DBMS

4) https://www.careerride.com/DB-types-of-relationships.aspx

5)http://etutorials.org/SQL/Database+design+for+mere+mortals/Part+II+The+Design+Process/Chapter+10.+Table+Relationships/Types+of+Relationships/