Restaurant Management System (Deliverable 1)

Group 62
Weiming Guo, Hengxian Jiang, Yingjie Xu, Helen Ren

Introduction:

purpose:

This application will store the information and constraints required to efficiently manage a restaurant system. This system could store two kinds of orders from customers and distribute staff with different responsibilities. The management system also demonstrates how staff interact with customers.

Database description:

Entities and their attributes

The following entities will be stored in the tables of a relational database.

Order: An order is a request for food or drink in a restaurant. An order is associated with two attributes, including order number and tips. The order number is the key to uniquely identify the order. The order has two subclasses, which are Dine-in Order and Delivery Order. There is a covering constraint on this entity, which means an order should be either dine-in order or delivery order.

Dine-in order: Dine in order is a subcategory of Order entity. It inherits all the attributes from the Order and interacts with customers in a different way than that of the other subclass.

Delivery order: Delivery order is a subcategory of the Order entity. It inherits all the attributes from Order. It also has its own attribute named “delivery fee”. The delivery order could be placed either on a third-party platform or by directly contacting the restaurant.

Platform: Platform is an entity with the attributes pname and the URL of their website, it can be uniquely identified by pname. A platform is an online ordering service provider (e.g.UberEats), which can be used to support our delivery order placed by customers. Customers can choose to place a delivery order through the platform.

Customer: Customer is an entity with attributes cname, address and phone number. A phone number is a key that can be used to uniquely identify the customer. A customer can place two different kinds of orders, which are Dine-in order and Delivery order. Customers can also make reservations.

Reservation: Reservation is made by the customer and it is a weak entity associated with the customer. It has attributes time slot and date. It can only be uniquely identified by the combination of the date and customer's phone number.

Dish: A dish is a food or a drink that exists in the restaurant menu. A dish is associated with two attributes, price and dish name. Dish name is the key to uniquely identify the dish entity.

Staff: A staff member is a person who is responsible for specific services in the restaurant. A Staff is associated with four attributes, including sname, working schedule, salary and id. Among these attributes, id is the key to uniquely identify a staff entity. The Staff entity has three subcategories, which are Chef, Waiter and Delivery guy respectively. There is a covering constraint on this entity, which means staff should be a chef, a waiter or a delivery guy.

Chef: A chef is a subcategory of the Staff entity. He/She has his/her own attributes, which are proficiency level (indicated by an integer from 0 to 5) and the cooking style. The chef is responsible for preparing dishes for customers.

Waiter: A waiter is a subcategory of the Staff entity, whose role is to serve food for the dine-in order placed by the customer. It inherits all attributes from the Staff entity.

Delivery guy: The delivery guy is a subcategory of the Staff entity, who is responsible for delivering orders placed by the customer. The delivery guy has attributes including delivery method and phone number.

Relationships

Place: A customer place a Dine-in order. This is a one-to-many relationship, a customer can place many dine-in orders. And each dine-in order must be placed by exactly one customer. So the relationship from dine-in order to a customer has both participation constraint and key constraint.

Contains: An order could contain several dishes. This is a many-to-many relationship with a participation constraint from Order to Dish because order must contain at least one dish but a dish does not necessarily have to be contained in a specific order. This relationship also has an attribute named quantity attached to it, indicating the number of dishes associated with that specific order.

Liaison: Liaison is a ternary relationship between customer, delivery order and platform. The relationship is many-to-one-to-one. A delivery order can only be placed by exactly one customer. However, each customer can place multiple delivery orders. Also, the customer can choose whether or not to place an order through a third-party platform. Each order can only have at most one platform associated with it.

Served by: Dine-in orders are served by waiters. This is a many-to-many relationship with a participation constraint between them. A waiter can serve more than one Dine-in orders at the same time, while one Dine-in order can only be served by one waiter.

Delivered by: Delivery orders are delivered by delivery guy. This is a many-to-one relationship with key and participation constraints from delivery orders to delivery guy. One delivery guy can deliver multiple orders, while one delivery order can only be delivered by one delivery guy.

Cooked by: Dish is cooked by a chef. This is a many-to-many relationship with a participation constraint from Dish to Chef because a dish must be cooked by at least one chef but a chef does not have to cook a dish.

Make: This is a weak relationship. Customers can make reservations. This is a one-to-many relationship because a customer can make several reservations on different days. However, a reservation must be made by exactly one customer. So, there are both participation and key constraints from reservation to a customer.

Application Description:

Overview

Our system provides impressive features. For example, we can manage multiple order types including both dine-in order and delivery order by using this single system. The restaurant could be advertised on the third-party platform to increase the customer base. This system provides a convenient way to facilitate employee management and work distribution. It provides seat reservation services so that customers can arrange their time in advance. Therefore, the system could keep everything in order and provide a good dining experience to customers.

Creativity and complexity

1. Our application domain stands out for user-friendly features as described above.
2. We have two instances of inheritances: chef, waiter and delivery guy are inherited from the staff. Dine-in order and delivery order are both subclasses of order.
3. We have one weak entity: the reservation is a weak entity, which can be

determined by the customer’s phone number and the reservation date together.

4. We have a ternary relationship: liaison is a ternary relationship between customer, delivery order and platform.
5. We have five key constraints: each dine-in order and each delivery order must be placed by one customer. Each reservation is made by only one customer. Each delivery order is delivered by only one delivery guy. Each order can be placed through at most platform

Relations:

Entities:

Order(order number, tips)

Dine-in order(order number, phone number), ***(order number ref Order) (phone number NOT NULL)***

Delivery order(order number, delivery fee, id, phone number, pname), ***(order number ref Order, pname ref Platform) (id NOT NULL, phone number NOT NULL)***

Platform(pname, URL)

Customer(phone number, cname, address)

Dish(dish name, price)

Staff(id, sname, working schedule, salary)

Chef(id, proficiency, cooking style), ***(id ref Staff)***

Waiter(id) ***(id ref Staff)***

Delivery guy(id, delivery method, phone number), ***(id ref Staff)***

Weak Entities:

Reservation(date, phone number, time slot), ***(phone number ref customer)***

Relationship:

Contains(order number, dish name, quantity) ***(order number ref Order, dish name ref Dish) (the participation constraint from order to dish could not be represented by the relational model)***

Cooked by(<u>dish name, id</u>) ***(id ref Chef, dish name ref Dish) (the participation constraint from dish to chef could not be represented by the relational model)***

Served by(<u>order number, id</u>), ***(id ref Staff, order number ref order) (the participation constraint from Dine-in Order to waiter could not be represented by the relational model)***

Delivered by ***This relationship doesn't need an extra table here because of the key and participation constraints***

Place ***This relationship doesn't need an extra table here because of the key and participation constraints***

Make ***This relationship doesn't need an extra table here because of the weak entity***

Ternary Relationship:

Liaison ***This relationship doesn't need an extra table here because of the key and participation constraints***