Data Management and Database Design- CRN 18472

Restaurant Database Management System

Contents

Team Members

S. No	Name	Email Id
1	Minghui Qiu	qiu.min@northeastern.edu
2	Xiaoyang Cai	cai.xiao@northeastern.edu
3	Manju Sharma	Sharma.manj@northeastern.edu
4	Kusha Choudhary	Choudhary.Kus@northeastern.edu

Project Topic

Restaurant Database Management System

Problem Statement

The existing system of the restaurant is maintained manually making it time consuming and tedious.

- All data is maintained manually, increasing the chances of error and complexity of maintenance.
- The same data is entered every day, which is a redundant process.
- No database available for the employees, customers, and the prime members.
- There is no system of reserving the table online.
- The food cannot be ordered online for delivery.
- The option of going contactless in the restaurant during covid was not available.
- Audit of inventory and transactions is difficult.

Objectives

The objective is to digitize the restaurant system with the below mentioned features:

- Online reservation of the table.
- Basis the past choices a custom menu will be available to each customer along with the traditional menu.
- Free delivery of birthday cake for the prime members.
- Contactless delivery of food.
- Assignment of delivery boys for specific areas.
- Online checkout at the table, ensuring no contact.

Proposed Solution for the Identified problems

1. **All data is maintained manually, increasing the chances of error and complexity of maintenance.**

   To overcome the said problem, a database management system has been created for the restaurant, where instead of maintaining the data manually, the data will be logged in its respective entity (such as customer, employees, orders, tables, menu …etc.) that will provide a structured data which can further be easily referred to, managed, and maintained. Moreover, different database DML functionalities, objects such as triggers, indexes, partitions, views, stored procedures, functions can be used which would not only fine tune the system but also play a vital role in maintaining the ETL process of database, hence making automating and processing of the data easier.

2. **The same data is entered every day, which is a redundant process.**

   Redundancy is being removed by the following design:

   i. Customer data need not be entered manually every time the customer is visiting the restaurant. All the relevant details will be present in the table ***customer***, such as **customer_id, customer_memberid, customer_lname, customer_fname** etc.
   ii. All the employee data will be maintained in the tables ***employee*** and attendant and it would be easy to keep track of all the employees.
   iii. The reservations will be taken care of by the database ensuring no two reservations for the same table and at the same time overlap.
   iv. All the tables will be normalized ensuring the data is unique and consistent.

3. **No database available for the employees, customers, and the prime members.**

   Three different tables for employees, customers and prime members have been created. Each table, ***employee***, ***customer*** and ***member*** have their unique ids, such as **employee_id**, **customer_id** and **member_id** respectively. These ids will help to determine each of the employee, customer, and member.

4. **There is no system of reserving the table online.**

   Reservation Entity is created in the database to reserve the table online, data in this table has **reservation_id** attribute as primary key (pk) to maintain uniqueness**.** The reservation table has the attribute **customer_id** and **reservation_tableId** (foreign key) which further helps to identify the table which has been reserved.
   There is a **reservation_timeIn** and **reservation_timeOut** attribute associated with the reservation table and the attribute **table_availability** helps in identifying if a table has been booked for a specific time thus avoiding duplicate and multiple bookings. Every time a customer will attempt to book an already booked table, a **trigger** will be generated.

5. The food cannot be ordered online for delivery.

The table ***deliverOrder*** has been created for storing orders for delivery. The key **order_customerId** identifies the customer which has placed the order, attribute **order_zipcode** is the primary key to the table ***area***, which stores the data of the assigned delivery agents for respective zip codes, using the key, **delivery_id.**

The **delivery _id** is the primary key for the table ***deliver*** which has a relationship with the ***employee*** table as the delivery agent will also be an employee with the restaurant.

6. Assignment of delivery agents for specific areas.

The table ***area*** is created which will contain all the zip codes within a certain range. The key **deliver_id** in the ***area*** table is the primary key to the entity ***deliver***. The ***deliver*** table has a relationship with the ***employee*** table, where the data of the delivery agent will be stored as the delivery agent is also the employee in the restaurant.

The key **deliver_availability**, will store the information of the availability of the delivery agent. If the assigned delivery agent is not available, then the assignment will be allocated to the next delivery agent.

7. The option of going contactless in the restaurant during covid was not available.

The issue of going contactless is being resolved by the below design

i. The option of food delivery will ensure contactless delivery of the food. If the customer checks the contactless option, then the same will be notified to the delivery agent.

ii. When a customer is seated in the restaurant there is an option of paying the bill without needing an attendant. The bill generated of the table is stored in the entity ***table***, in the attribute **table_bill**. The customer can make the payment against the bill generated and the data **payment_id**, **table_customerId**, **payment_time** will be stored in the entity payment.

8. Audit of inventory and transactions is difficult

With the creation of database auditing of the inventory and the transactions made is possible and easy to track

i. The table ***payment*** keeps the records of all the payments made. The attribute **payment_id** identifies the payment, **table_customerId** stores the ID of the customer who made the payment and attributes **payment_time** and **payment_mainValue** stores the information about the date and time of the payment made and the total value of the payment.

ii. The table ***inventory*** will keep the track of all the inventory present using the attributes **inventory_name** and **inventory_quant.** Whenever a dish is ordered, the database will calculate which and how many ingredients will be used and subtract the quantity from the inventory. This function can be realized by using the entity ***dishFormula***.

iii. Weekly, monthly, and annual reports can be easily generated to identify different aspects related to sales and inventory.

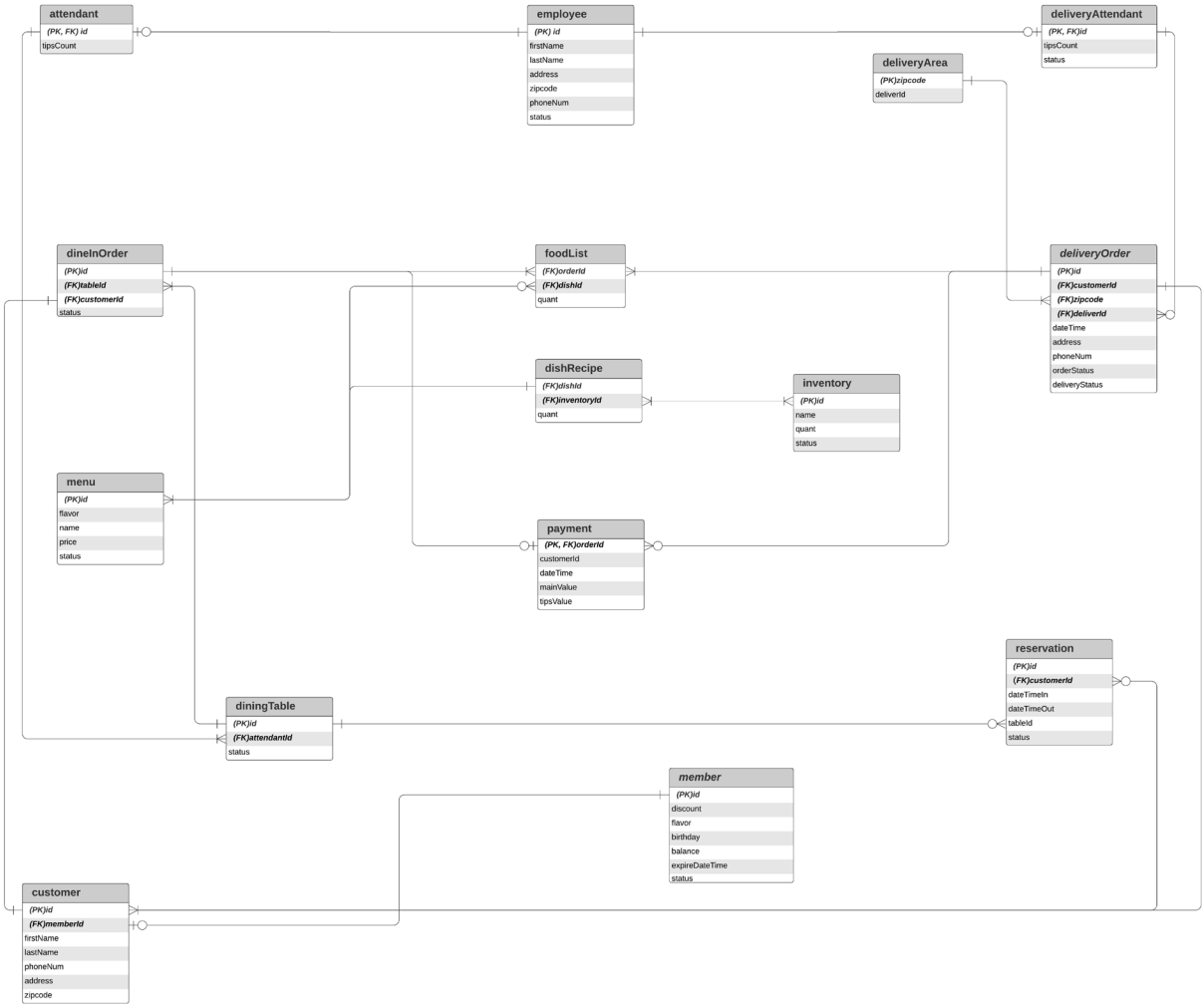
9. **Basis the past choices a custom menu will be available to each customer along with the traditional menu**

   Certain customers will be members which can be identified using the attribute **member_id** stored in the table ***member***. Basis the past choices of the members a custom menu will be developed, using an algorithm, for the customers. The choices of the customer will be saved in the attribute **member_flavour**.

10. **Free delivery of birthday cake for the prime members.**

    The birthdays of each of the members will be stored in the attribute **member_birth**. A trigger will be generated on their birthday , ensuring free delivery of birthday cake to each of the members

Entity – Relationship Diagram



Database Dictionaries

Database details for Restaurant Management System.

Table 1 - attendant

Field Name	Description	Type	Length
id	PK, FK	int	8
tipsCount	Tip for the attendant	float	4,2

Table2 - employee

Field Name	Description	Type	Length
id	Primary Key	int	8
firstName	First name of employee	char	255
lastName	Last name of employee	char	255
address	Address of employee	char	255
zipcode	Zip code of employee	int	5
phoneNum	Phone number of employees	char	255
status	Status of employees	ENUM("present", "absent")	

Table3 - deliveryAttendent

Field Name	Description	Type	Length
id	Id of delivery man	int, PK, FK	8
tipsCount	Tip for delivery	float	4,2
status	Status of delivery man	ENUM("delivered", "on the way", "preparing")	

Table4 - dineInOrder

Field Name	Description	Type	Length
id	Primary Key	int	10
tableid	Table number for dine	int, FK	4
customerId	Id of customer	Int, FK	8
status	Status of dine In order	ENUM("accepted", "rejected")	

Table5 - foodList

Field Name	Description	Type	Length
orderId	Foreign Key	int, FK	10
dishId	Foreign Key	int, FK	4
quant	Quantity of dish	int	4

Table6 - deliveryOrder

Field Name	Description	Type	Length
id	Primary Key	int, PK	10
customerid	Foreign Key	int, FK	8
zipcode	Foreign Key	int, FK	5
deliverid	Foreign Key	int, FK	8
paymentid	Foreign Key	int, FK	4
dateTime	Date of Order	datetime	-
address	Address for delivering order	char	255
phoneNum	Phone number	char	255
orderStatus	Status of order	ENUM("accepted", "rejected")	
deliveryStatus	Status of delivery	ENUM("delivered", "on the way", "preparing")	

Table7 - menu

Field Name	Description	Type	Length
id	Primary Key	int, PK	4
price	Price of dish	float	4,2
name	Dish Name	char	255
flavor	Dish Flavor	Char	255
status	Status of menu	ENUM("available", "unavailable")	

Table8 – dishRecipe

Field Name	Description	Type	Length
dishId	Foreign Key	int	4
inventoryId	Foreign Key	int	4
quant	Quantity of ingredient	int	4

Table9 - inventory

Field Name	Description	Type	Length
id	Primary Key	int	4
name	Name of inventory	char	255
quant	Quantity of inventory	int	4
status	Status of inventory	ENUM("in stock", "out of stock")	

Table10 – diningTable

Field Name	Description	Type	Length
id	Primary Key	int, PK	4
orderId	Foreign Key	int, FK	10
attendantId	Foreign Key	int, FK	8
status	Status of table	ENUM("available", "occupied")	

Table11 - reservation

Field Name	Description	Type	Length
id	Primary Key	int, PK	4
customerId	Foreign Key	int, FK	8
dateTimeIn	In time for reservation	datetime	-
dateTimeOut	Out time for reservation	datetime	-
tableId	Reserve table number	int	4
status	Status of reservation	ENUM("accepted", "rejected")	

Table12 - member

Field Name	Description	Type	Length
id	Primary Key	int, PK	8
discount	Foreign Key	int, FK	1
flavor	Favorite flavor of member	char	255
birthday	Member Birth date	date	-
balance	Member Balance Due	float	4,2
expireDateTime	Date of expiration	date	-
status	Status of member	ENUM("yes", "no")	

Table13 - customer

Field Name	Description	Type	Length
id	Primary Key	int, PK	8
memberId	Foreign Key	int, FK	8
firstName	First Name	char	255
lastName	Last Name	char	255
phoneNum	Phone Number	char	255
address	Address	char	255
zipcode	Zip Code of customer	int	5

Table14 – deliveryArea

Field Name	Description	Type	Length
zipcode	Primary Key	int, PK	5
deliverId	Id of deliver	int	4

Table15 – payment

Field Name	Description	Type	Length
id	Primary Key	int, PK	4
orderId	FK	int	10
customerId	Id of customer	int	8
dateTime	Time of Payment	datetime	-
mainValue	main value of payment	float	4,2
tipsValue	tips of payment	float	4,2

Business Rules

Following mentioned are the business rules

1. One ***attendant*** can serve one or many ***diningTable*** .
2. An **employee** may or may not be an ***attendant*****,** but an ***attendant*** must be an ***employee***.
3. An ***employee*** may or may not be a ***deliveryAttendant*** but a ***deliveryAttendant*** must be an employee.
4. One ***deliveryAttendant*** is assigned to a specific ***deliveryArea*** depending on the **zip** of ***deliveryArea*****.**
5. One ***deliveryAttendant*** can deliver none or many ***deliveryOrders***.
6. One **deliveryArea** can be associated with one or many **deliveryOrders.**
7. One ***deliveryOrder*** can have one or many orders, each having an **orderId** and **dishId** stored in the ***foodList*****.**
8. One ***deliveryOrder*** can be associated with one ***customer*****.**
9. ***member*** **status** can be updated only by the admin, **Ex: active, inactive**.
10. ***inventory*** **status** can be updated only by the admin, **Ex: in stock or out of stock**
11. ***reservation*** status is updated automatically to **available or booked** (using a view).
12. Every ***customer*** may or may not be a special **member** of the restaurant.
13. Every ***customer*** may or may not have a **reservation.**
14. One ***diningTable*** can have no or multiple ***customers*****.**
15. One ***dineinOrder*** can have multiple orders having **orderIds** present in the ***foodList*** and one ***foodList*** tuple can be associated with one order.
16. One ***diningTable*** can be associated with one or multiple orders having different **orderIds** in the ***orderList***
17. One ***diningTable*** can have no or only one ***reservation*** at any time.
18. One **diningTable** can be associated with one **payment.**
19. A ***diningTable*** can have multiple ***dineinOrder*** and one ***dineinOrder*** is associated with a ***diningTable*****.**

Security Rules: (User level access / Permissions)

1. **Admin**
   i. All level of access for all the entities.

2. **Manager**
   i. Read only access to ***payment***, ***dishRecipe*** , ***orderList*** , ***customer.***
   ii. READ/WRITE/UPDATE access to ***employee*** , ***inventory*** , ***menu*** , ***dinningTable*** , ***reservation***.

3. **Attendant**
   i. No access to ***payment***, ***inventory*** ,***customer*** , ***dishRecipe.***
   ii. Read Only access to ***reservation***, ***attendant*** , ***OrderList*** ,***dineInOrder***, ***menu.***
   iii. READ/WRITE/UPDATE access to ***employee.***

4. **Delivery Agent**
   i. No access to ***payment***, ***inventory*** ,***customer*** , ***dishRecipe.***
   ii. Read Only access to ***deliveryOrder***, ***deliveryArea.***
   iii. READ/WRITE/UPDATE access to ***employee.***

5. **Customer**
   i. No access to ***inventory*** ,***customer*** , ***dishRecipe ,***
   ii. Read Only access to the ***menu , payment*** only.
   iii. READ/WRITE/UPDATE access to ***customer.***