RESTAURANT MANAGEMENT SYSTEM

RIYA MATE

INTRODUCTION

A Restaurant Management System (RMS) is a software solution designed to streamline and enhance various aspects of restaurant operations. It integrates multiple functions to help restaurant owners, managers, and staff manage their business more efficiently. Here are key features and components typically found in a restaurant management system like

1) Order management for online delivery
2) Inventory management

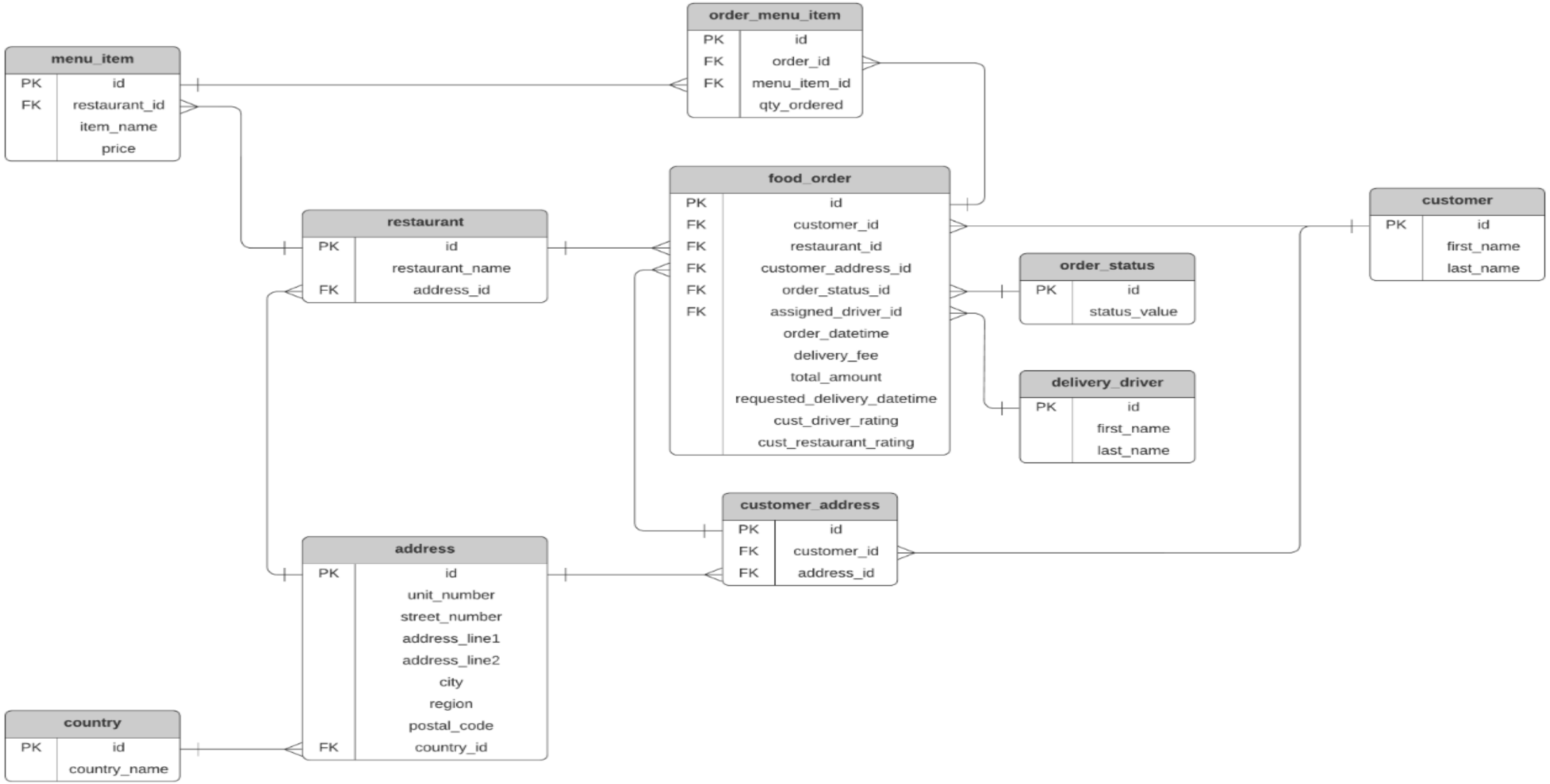
DESCRIPTION

1. **Place order**: Customers can choose a restaurant and place an order for food from the restaurant
2. **Menu items**: Each restaurant has different menu items and prices. Out of scope - customising the order by adding and removing ingredients
3. **Delivery time**: Customers can specify a delivery time, either ASAP (when the order is placed) or at a future time
4. **Delivery address**: Customers can store multiple delivery addresses on their profile
5. **Order status**: Restaurants can accept the order and update the status of the order as it is made
6. **Assign order**: Delivery drivers can pick an order to deliver to assign it to themselves
7. **Pick up order**: Delivery drivers can pick up an order that is ready from the restaurant and deliver it to the customer
8. **Ratings**: Customers can rate delivery drivers and restaurants

Subtitle

ER DIAGRAM

ER


order_menu_item
PK id
FK order_id
FK menu_item_id
qty_ordered
menu_item
PK id
FK restaurant_id
item_name
price
food_order
PK id
FK customer_id
FK restaurant_id
FK customer_address_id
FK order_status_id
FK assigned_driver_id
order_datetime
delivery_fee
total_amount
requested_delivery_datetime
cust_driver_rating
cust_restaurant_rating
customer
PK id
first_name
last_name
restaurant
PK id
restaurant_name
FK address_id
order_status
PK id
status_value
delivery_driver
PK id
first_name
last_name
customer_address
PK id
FK customer_id
FK address_id
address
PK id
unit_number
street_number
address_line1
address_line2
city
region
postal_code
FK country_id
country
PK id
country_name

NORMALIZATION PROCESS

1) First Normal Form (1NF):Ensure that each table has a primary key (id in your case).Make sure that each column contains atomic (indivisible) values.The address table seems to violate 1NF due to multiple address components. Consider breaking it down into separate columns.
2) Second Normal Form (2NF):Meet 1NF requirements. Remove partial dependencies**. The food_order table might have partial dependencies on customer_id and restaurant_id. Separate these into a new table, say *customer_order.***
3) Third Normal Form (3NF):Meet 2NF requirements. Remove transitive dependencies. **The food_order table might have a transitive dependency on address through customer_address_id. Consider creating a new table for *customer_address* and linking it to the customer and address tables.**

customer_address

RELATION SHIP

1) customer:
2) One-to-Many relationship with customer_address: One customer can have multiple addresses.
3) One-to-Many relationship with customer_address: One address can be associated with multiple customers.
4) One-to-Many relationship with restaurant: One address can be associated with multiple restaurants
5) .One-to-Many relationship with food order: One address can be associated with multiple food orders. Country:
6) One-to-Many relationship with address: One country can have multiple addresses. Restaurant:
7) Many-to-One relationship with address: Many restaurants can share the same address.
8) One-to-Many relationship with menu_item: One restaurant can have multiple menu items.
9) One-to-Many relationship with food_order: One restaurant can receive multiple food orders.menu_item:
10) Many-to-One relationship with restaurant: Many menu items can belong to the same restaurant.
11) One-to-Many relationship with order_menu_item: One menu item can be part of multiple food orders.food_order:
12) Many-to-One relationship with customer: Many food orders can be placed by the same customer.




PROCEDURES

- -- Calculate Average Customer Rating for a Restaurant
- DELIMITER //
- CREATE PROCEDURE calculate_avg_rating_for_restaurant(IN p_restaurant_id INT, OUT p_avg_rating DECIMAL(5, 2))BEGIN DECLARE total_ratings DECIMAL(10, 2); DECLARE total_customers INT; -- Calculate the sum of ratings and the count of customers who rated SELECT SUM(cust_restaurant_rating) INTO total_ratings FROM food_order WHERE restaurant_id = p_restaurant_id AND cust_restaurant_rating IS NOT NULL; SELECT COUNT(*) INTO total_customers FROM food_order WHERE restaurant_id = p_restaurant_id AND cust_restaurant_rating IS NOT NULL; -- Calculate the average rating IF total_customers > 0 THEN SET p_avg_rating = total_ratings / total_customers; ELSE SET p_avg_rating = NULL; END IF;END //DELIMITER ;
- -- Call the stored procedure for Restaurant with ID 1
- CALL calculate_avg_rating_for_restaurant(1, @avg_rating);-- Retrieve the resultSELECT @avg_rating AS average_rating;DELIMITER //

TRIGGERS

- -- Created a BEFORE INSERT trigger on the restaurant table to check if the order is placed correctly

- DELIMITER //CREATE TRIGGER before_insert_restaurantBEFORE INSERT ON restaurant FOR EACH ROWBEGIN -- Check if the address_id exists in the address table IF NOT EXISTS (SELECT 1 FROM address WHERE id = NEW.address_id) THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid address_id. Cannot insert into restaurant.'; END IF;END;//DELIMITER ;

VIEWS

- Create the view for customer information
- CREATE VIEW customer_info ASSELECT id AS customer_id, first_name, last_nameFROM customer;
- -- Create the view for restaurant information-- This view, named restaurant_info, combines information from the restaurant, address, and country tables to provide details about each restaurant along with its full address.
- CREATE VIEW restaurant_info ASSELECT r.id AS restaurant_id, r.restaurant_name, a.unit_number, a.street_number, a.address_line1, a.address_line2, a.city, a.region, a.postal_code, c.country_nameFROM restaurant rJOIN address a ON r.address_id = a.idJOIN country c ON a.country_id = c.id;-- Select all rows from the customer view
- SELECT * FROM customer_info;select * FROM restaurant_info;
- -- Select specific columns from the restaurant view
- SELECT restaurant_name, city, country_nameFROM restaurant_info;

ANALYTICS

1) Find top customers-based orders
2) Find busiest hours of food delivery
3) Find popular trend by monthly orders
4) Improvise solutions for efficient Delivery management.

SOME QUERIES!

- SQL **ANALYTIC** QUERIES –
- *Top Customers by Total Order Amount*:

SELECT c.first_name, c.last_name, SUM(fo.total_amount) AS total_order_amountFROM customer c JOIN food_order fo ON c.id = fo.customer_idGROUP BY c.first_name, c.last_nameORDER BY total_order_amount DESCLIMIT 5;

- *-- Monthly Order Trend by Restaurant*:

SELECT r.restaurant_name, EXTRACT(MONTH FROM fo.order_datetime) AS order_month, COUNT(fo.id) AS monthly_ordersFROM restaurant r LEFT JOIN food_order fo ON r.id = fo.restaurant_idGROUP BY r.restaurant_name, order_monthORDER BY r.restaurant_name, order_month;

- *-- Busiest Hours for Food Orders:*

SELECT EXTRACT(HOUR FROM fo.order_datetime) AS order_hour, COUNT(fo.id) AS orders_countFROM food_order foGROUP BY order_hourORDER BY orders_count DESCLIMIT 5;

THANK YOU

Riya Mate

mate.r@northeastern.edu