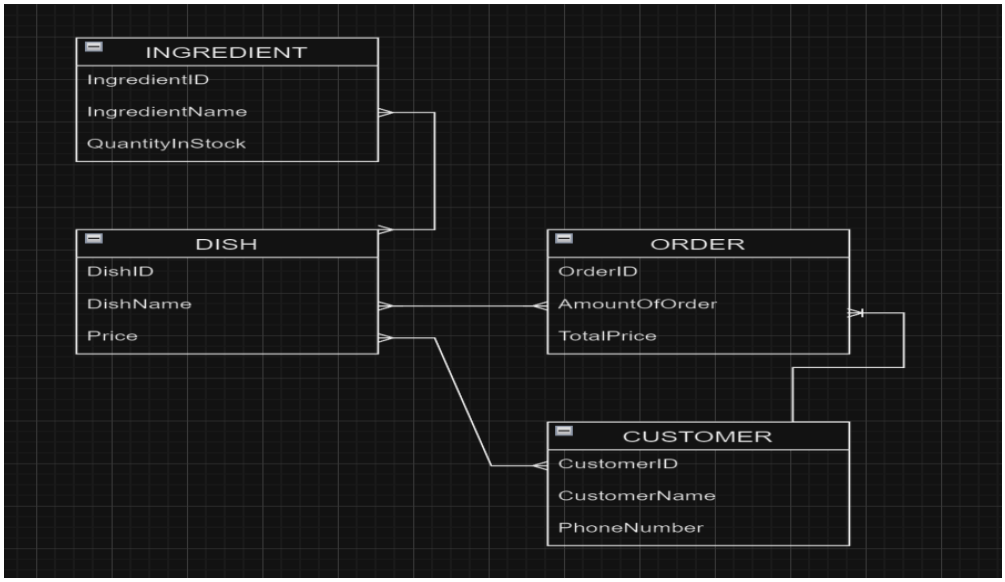
Sean Ashton V. Regalado BSSE-2 09/27/24

SE 2141 – Laboratory 2

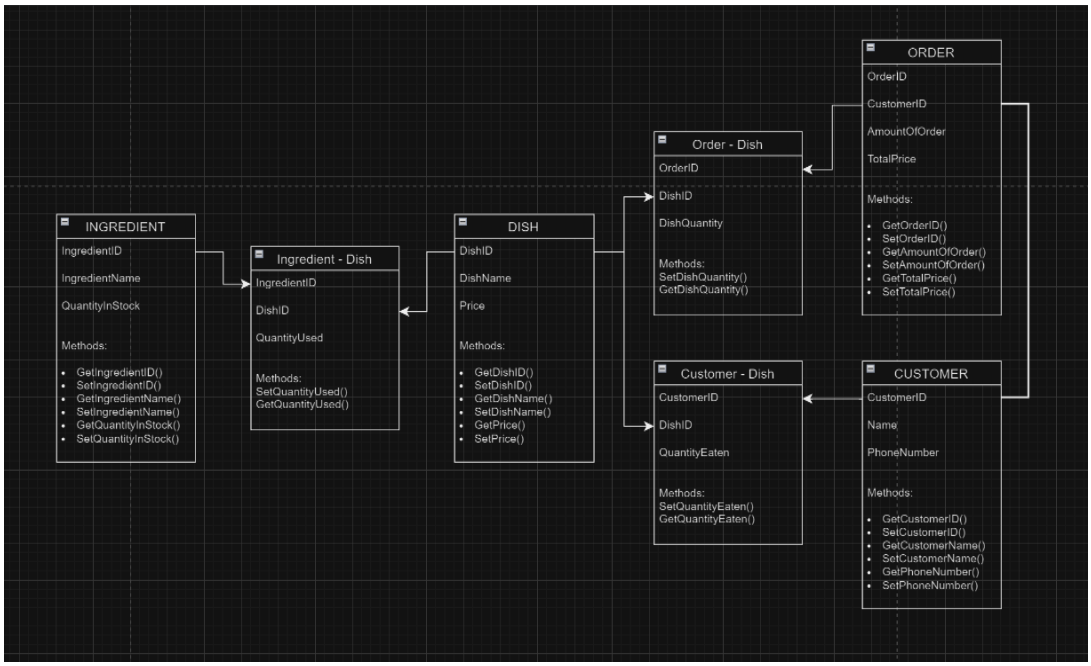
Part 1: Entity-Relationship Models



Relationships:

- Ingredient-Dish (M:M): Each dish can use many ingredients and each ingredient can be use in different dishes.
- Order-Dish (M:M): An order can consist of multiple dishes and a dish can appear in multiple orders.
- Customer-Dish (M:M): A customer can eat multiple dishes and a dish can be eaten by multiple customers.
- Customer-Order (1:M): A customer can have multiple orders but one order can only belong to a single customer.

Part 2: Object-Oriented Data Models



This is the OODM version of the ER diagram from part 1. In the OODM diagram, I added methods for each entity which are a bunch of setter and getter methods for each attribute. Additionally, a relationship table was also added to visualize the relationship between entities. In the relationship table, it has an attribute of the primary key of both the entity in relationship. It also has an extra attribute like QuantityUsed in the Ingredient – Dish relationship, to store the data on the quantity of ingredients that were used in that dish. It also has a method which is a getter and setter method to get and store data to the extra attribute.

Part 3: Relational Data Models

SQL Tables and Associative Tables with inserted Sample Data

Database: PostgreSQL v15 ▼ Run Update Fork

Schema SQL

```sql
-- INIT database
CREATE TABLE CUSTOMER (
  CustomerID INT PRIMARY KEY,
  CustomerName VARCHAR(100),
  PhoneNumber VARCHAR(15)
);

CREATE TABLE CUSTOMER_ORDER (
  OrderID INT PRIMARY KEY,
  AmountOfOrder INT,
  TotalPrice DECIMAL(7, 2),
  CustomerID INT,
  FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID)
);

CREATE TABLE DISH (
  DishID INT PRIMARY KEY,
  DishName VARCHAR(100),
  Price DECIMAL (5,2)
);

CREATE TABLE INGREDIENT (
  IngredientID INT PRIMARY KEY,
  IngredientName VARCHAR(100),
  QuantityInStock INT
);

CREATE TABLE DishIngredient (
  DishID INT,
  IngredientID INT,
  QuantityUsed INT,
  PRIMARY KEY (DishID, IngredientID),
  FOREIGN KEY (DishID) REFERENCES DISH(DishID),
  FOREIGN KEY (IngredientID) REFERENCES INGREDIENT(IngredientID)
);
```

Database: PostgreSQL v15 ▼ Run Update Fork Load Ex

Schema SQL

```sql
-- Many-to-many relationship between Customer and Dish
CREATE TABLE DishCustomer (
  DishID INT,
  CustomerID INT,
  QuantityEaten INT,  -- Quantity of a specific dish eaten by the customer
  PRIMARY KEY (DishID, CustomerID),
  FOREIGN KEY (DishID) REFERENCES DISH(DishID),
  FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID)
);

-- Many-to-many relationship between Dish and Order
CREATE TABLE DishOrder (
  DishID INT,
  OrderID INT,
  DishQuantity INT,  -- Quantity of a specific dish in an order
  PRIMARY KEY (DishID, OrderID),
  FOREIGN KEY (DishID) REFERENCES DISH(DishID),
  FOREIGN KEY (OrderID) REFERENCES CUSTOMER_ORDER(OrderID)
);



INSERT INTO CUSTOMER (CustomerID, CustomerName, PhoneNumber)
VALUES
(1, 'John Smith', '535-648-093'),
(2, 'Luminous Valentine', '221-664-932');

INSERT INTO CUSTOMER_ORDER (OrderID, AmountOfOrder, TotalPrice, CustomerID)
VALUES
(1, 3, 1000, 1),
(2, 2, 450, 2);
```

Database: PostgreSQL v15 ▼ Run Update Fork L

Schema SQL

```sql
INSERT INTO DISH (DishID, DishName, Price)
VALUES
(1, 'Spaghetti', 120),
(2, 'Tacos', 70),
(3, 'Pizza', 250);

INSERT INTO INGREDIENT (IngredientID, IngredientName, QuantityInStock)
VALUES
(1, 'Tomato', 50),
(2, 'Cheese', 100),
(3, 'Meat', 50),
(4, 'Lettuce', 30);

INSERT INTO DishIngredient (DishID, IngredientID, QuantityUsed)
VALUES
(1, 1, 5),   -- Spaghetti uses Tomato
(1, 2, 4),   -- Spaghetti uses Cheese
(2, 3, 6),   -- Tacos use Meat
(2, 4, 1),   -- Tacos use Lettuce
(3, 2, 5),   -- Pizza uses Cheese
(3, 1, 3);   -- Pizza uses Tomato

-- Insert data into DishCustomer (Dishes eaten by customers)
INSERT INTO DishCustomer (DishID, CustomerID, QuantityEaten)
VALUES
(1, 1, 2),   -- John Smith ate 2 Spaghetti
(2, 1, 1),   -- John Smith ate 1 Taco
(3, 2, 1);   -- Luminous Valentine ate 1 Pizza

-- Insert data into DishOrder (Dishes in each order)
INSERT INTO DishOrder (DishID, OrderID, DishQuantity)
VALUES
(1, 1, 2),   -- Order 1 includes 2 Spaghetti
(2, 1, 1),   -- Order 1 includes 1 Taco
(3, 2, 1);   -- Order 2 includes 1 Pizza
```

SQL Queries and its results

Database: PostgreSQL v15 ▼ Run Update

Query SQL

```sql
-- Get the dishes in Order 1
SELECT d.DishName, d_order.DishQuantity
FROM DishOrder d_order
JOIN DISH d ON d_order.DishID = d.DishID
WHERE d_order.OrderID = 1;

-- Get ingredients used in Spaghetti
SELECT i.IngredientName, di.QuantityUsed
FROM DishIngredient di
JOIN INGREDIENT i ON di.IngredientID = i.IngredientID
WHERE di.DishID = 1;

-- Get orders for Customer 1
SELECT co.OrderID, co.TotalPrice
FROM CUSTOMER_ORDER co
WHERE co.CustomerID = 1;
```

Database: PostgreSQL v15 ▼ Run Update Fork Load Example Star PRO Embed PRO Collaborate Sign in Have any feedback?

Results

Copy as Markdown

Query #1 Execution time: 1.27ms

dishname	dishquantity
Spaghetti	2
Tacos	1

Query #2 Execution time: 0.43ms

ingredientname	quantityused
Tomato	5
Cheese	4

Query #3 Execution time: 0.3ms

orderid	totalprice
1	1000.00

This is the relational model for the ER model in part 1. To create a table, you use CREATE TABLE <Entity Name> and supply each entity with their corresponding attributes with The EntityID being the PRIMARY KEY. VARCHAR(number) is used assigned a length on how long can the Attribute column can store characters. In the associative tables, they all have the primary key of the entity that are in a relationship with and each one of them have an extra attribute to associate the relationship of the two. To insert data to these entities, you use INSERT INTO <Entity Name> (attributes, …) VALUES. You then supply it with data that matches the attribute like for example your attribute is Name so you need to add “Name of person” to that attribute. Lastly to create SQL queries, you can use the SELECT, FROM, JOIN AND WHERE keywords. SELECT keyword is used to specify which columns (or data) you want to retrieve from the database. Example, d.Dishname is used to retrieve the name of the dish in the DISH table. Next, FROM is used to specify the table(s) from which you are retrieving data from. An example is DishOrder d_order. DishOrder is the table that I am retrieving data from and d_order is an alias or shortcut for the DishOrder table so that I can refer to the table more easily later in the query. Next is the JOIN keyword. It is used to combine rows from two or more tables based on a related column. An example is JOIN DISH d ON d_order.DishID = d.DishID. What this essentially do is that the DishID attribute from the DISH table and is being matched with the DishID attribute that can be found in the DishOrder associative table. Lastly, WHERE is used to filter the rows returned by the query, so that only the rows meeting certain conditions are selected. In the example WHERE d_order.OrderID = 1 , I am specifying that I only want to see the dishes in Order 1.

Part 4: Report

This laboratory was all about creating different kinds of data models. Entity-Relationship model in part 1, Object Oriented Data Model in part 2 and Relational Data Model in part 3. In the ER model in part 1, I identified the kinds of entities that can be found in a restaurant system and then determine their attributes. Those entities are customer, dish, ingredient, and order. They all have their specific attributes to them especially their primary key which is important for understanding relationship between two different entities.

In part 2, my task is to create a OODM based from the ER model that I created in part 1 In making it, I just kept the model from part 1 and I added each entity with their own personal methods. I also added a relationship table where primary keys of both entity in relationship is linked together. This acts like a bridge that connects both entities and show how they interact with each other.

In last part, part3, it was also tasked to create a model based from the ER model but this time it a relational data model. To do this we were instruct to use SQL to create tables and associative tables with inserted data samples and SQL queries to retrieve data from those tables based on certain conditions. On the pictures in part 3, I first created tables for the four entities and their associative relationships using CREATE TABLE <Entity Name> and associating them with their own attributes. Next, I inserted data to each one of them using INSERT INTO <Entity Name> (attributes…). This supplies data to each attribute of every entity that will be used in sending queries. Lastly, I created some SQL queries like getting the dishes on order 1, get ingredients used in spaghetti and getting orders from customer 1. The results of the queries were also successful as it shows the correct data that were expected to be receive.