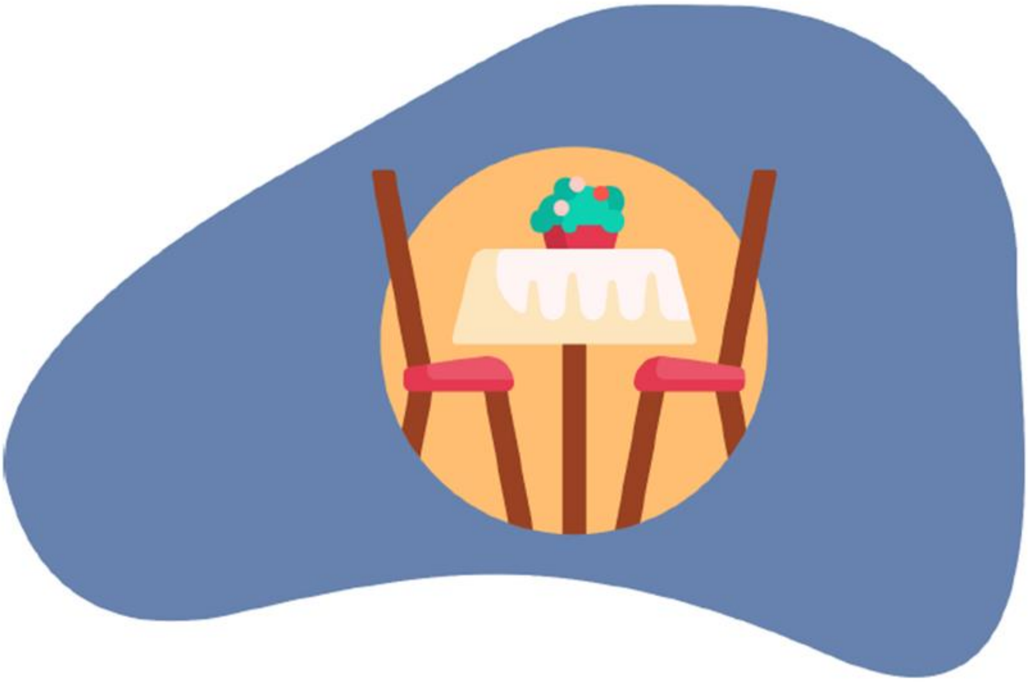
RESTAURANT DBMS PROJECT



Group 20

Gencay Turgut	280201056
Muhammed Efe İncir	270201029
Cem Kağan Kayılı	280201019
Matin Huseynzade	280201086
Yasir Duman	280201101

1- INTRODUCTION

A key component of implementing IT systems that power business applications and deliver analytical data to support operational decision-making and strategic planning by company leaders, business managers, and other end users is efficient data management. Database management systems (DBMS), offer benefits to important workplaces such as visibility, reliability, security and scalability. With the integration of DBMSs, restaurants now can manage orders to the tables, operate on orders digitally, and view their sales. Some types of DBMSs for restaurants include Point of Sale (POS), restaurant reservation software, table management, food costing software, inventory management systems etc.

In our project, we will develop a DBMS specially designed for the use of restaurants. This DBMS aims to create delivery monitoring for our restaurant. Our DBMS will manage the orders, waiters, bills, tables, reservations and products. A customer sits at a table then chooses his/her demands from the menu. An order is created from products using the Order-Product relationship. Products consist of ingredients. Required ingredients will be bought from their suppliers using the Supplier-Ingredient relationship. If an order has prepared status, a trigger will create the bill using the Bill-Order relationship. Reservations will grant the customers the flexibility of time and will be connected to a Table entity via Reservation-Table relationship. Also Waiters' will be assigned to tables via the Table-Waiter relationship. Instead of generating bills and orders per customer, bills, orders, and waiters will be assigned to tables. Thus providing the ease of management and flexibility. These are the optimal relationships and entities to manage a restaurant.

2- DESIGN ATTRIBUTES

Entity Sets and Attributes

- **RESERVATION(<u>RESERVATION_ID</u>, RESERVATION_CUSTOMER_NAME, RESERVATION_TIME,RESERVATION_PHONE_NUMBER):** Reservation is an arrangement that tells us the name and telephone number of our customers and when our customers will come to our restaurant. It is an entity that holds the RESERVATION_ID, RESERVATION_CUSTOMER_NAME , RESERVATION_TIME and RESERVATION_PHONE_NUMBER information of a reservation. RESERVATION_ID is a primary key. RESERVATION_CUSTOMER_NAME attribute is for the name of the customer to be served. RESERVATION_TIME attribute is the information about when the reservation was made. It is the DATE data type. All attributes cannot be null. We also take the phone number, as the customer name may not be unique.
- **TABLE(<u>TABLE_ID</u>, TABLE_SIZE, TABLE_LOCATION, TABLE_FLOOR):** Table is a piece of furniture that our customers use to receive service. It is an entity that holds the TABLE_ID, TABLE_CAPACITY, TABLE_LOCATION, TABLE_FLOOR information of a table. TABLE_ID is the primary key and it is used for the table number information. So, every table number is unique. TABLE_SIZE attribute is the information that shows the maximum number of people that can be served to the table. TABLE_LOCATION attribute is for the information about where customers will be served in the restaurant. In our restaurant, there are 3 locations which are 'garden', 'terrace' and 'inside'. The system will check if the entered location information is one of them. TABLE_FLOOR attribute is for the information that says on which floor the table is in the restaurant. If the TABLE_LOCATION is entered as a 'garden', the TABLE_FLOOR attribute is restricted to 0 and other TABLE_FLOOR numbers

cannot be written(**). TABLE_ID, TABLE_LOCATION, TABLE_FLOOR attributes cannot be null.

- **WAITER(<u>WAITER_ID</u>, WAITER_NAME, WAITER_PHONE_NUMBER, WAITER_DATE_OF_BIRTH)**

  Waiter is a man/woman who works in our restaurant and serves our customers. It is an entity that holds the WAITER_ID, WAITER_NAME, WAITER_PHONE_NUMBER, WAITER_DATE_OF_BIRTH information of a waiter. WAITER_ID is a primary key. The WAITER_NAME attribute is for the waiter's name. WAITER_PHONE_NUMBER is the phone number of the waiter. WAITER_DATE_OF_BIRTH attribute is the date of birth of the waiter. It is the DATE data type. The system checks if the waiter is older than 18 or not. All attributes cannot be null.

- **ORDER(<u>ORDER_ID</u> , ORDER_DATE)**

  Order is a request for serving in return for payment and it is created after our customers place their order. It is an entity that holds the ORDER_ID and ORDER_DATE information of an order. ORDER_ID is the primary key.ORDER_DATE attribute holds the information about when the order was made. ORDER_ID and ORDER_DATE can not be null.

- **BILL(<u>BILL_ID</u>, BILL_DATE, BILL_STATUS, BILL_PAYMENT_METHOD)**

  Bill is used to tells us how much our customers pay for a service. It allows us to find the amount to be paid together with the queries. It is an entity that holds the BILL_ID, BILL_DATE, BILL_STATUS, BILL_PAYMENT_METHOD information of a bill. BILL_ID attribute is the primary key. BILL_DATE attribute is for the information about when the bill was created. BILL_STATUS attribute is for the information that shows whether the bill has been paid or not. It can be 'paid' or 'not paid'. BILL_PAYMENT_METHOD attribute is for the information that shows how the payment is made. It can be 'cash' or 'credit card'. All attributes cannot be null. A bill will be created automatically after a product in an order has been prepared.

- **PRODUCT(<u>PRODUCT_ID</u>, PRODUCT_COST , PRODUCT_PRICE , PRODUCT_DESCRIPTION, PRODUCT_TYPE):**

  Product is what we serve to the customers. It is an entity that holds the PRODUCT_ID, PRODUCT_COST, PRODUCT_PRICE, PRODUCT_DESCRIPTION, PRODUCT_TYPE information of a product. PRODUCT_ID attribute is the primary key. PRODUCT_COST attribute is the information that shows how much the product cost to make it. It is a DECIMAL data type with 6 precision and 2 scales. PRODUCT_PRICE attribute is the sales price information of the product. It is a DECIMAL data type with 6 precision and 2 scales. PRODUCT_TYPE attribute is the information that shows what type the product is. It can be either 'food' or 'beverage'.

- **INGREDIENT (<u>INGREDIENT_ID</u>, INGREDIENT_NAME)**

Ingredient entity set holds information about all the ingredients that our products may possibly include, therefore, enabling us to check for any allergic response from our customers. This entity set has two attributes: INGREDIENT_ID, INGREDIENT_NAME. Each tuple in the relation represents a single ingredient and has a unique value for the INGREDIENT_ID attribute, a value for the INGREDIENT_NAME attribute, and a value for the SUPPLIER_ID attribute that corresponds to the ID of the supplier that supplies the ingredient.

- **SUPPLIER (<u>SUPPLIER_ID</u>, SUPPLIER_NAME, SUPPLIER_TELEPHONE_NUMBER)**

  This entity set has three attributes: SUPPLIER_ID, SUPPLIER_NAME, and SUPPLIER_TELEPHONE_NUMBER. Each tuple in the relation represents a single supplier and has a unique value for the SUPPLIER_ID attribute and a value for the SUPPLIER_NAME and SUPPLIER_TELEPHONE_NUMBER attributes

Relationship Sets and Attributes

- **RESERVATION-TABLE(<u>RESERVATION_ID, TABLE_ID</u>):** This relationship shows us which table the customers reserve. It includes the RESERVATION_ID and TABLE_ID as primary keys. Each reservation is associated with a table. A table is associated with at most 1(including 0) reservation.
- **TABLE-WAITER(<u>TABLE_ID, WAITER_ID</u>):** This relationship shows us which table the waiters serve. It includes the TABLE_ID and WAITER_ID as primary keys. A table can have 0 or 1 waiter to serve. A waiter can serve 0 or more tables.
- **TABLE_ORDER(<u>TABLE_ID, ORDER_ID</u>):** This relationship shows us which table the order belongs to. It includes the TABLE_ID and ORDER_ID as primary keys. A table is associated with several (including 0) orders. Each order is associated with a table.
- **BILL_ORDER(<u>BILL_ID, ORDER_ID</u>):** This relationship shows us which orders are in the bill. It includes the BILL_ID and ORDER_ID as primary keys. Each bill is associated with at least 1 order. Each order is associated with several bills.
- **ORDER-PRODUCT(<u>ORDER_PRODUCT_ID, ORDER_ID, PRODUCT_ID,</u> ORDER_PRODUCT_STATUS):** This relationship shows us which products are in the order. It includes the ORDER_PRODUCT as a primary key, ORDER_ID and PRODUCT_ID as foreign keys and ORDER_PRODUCT_STATUS as an attribute.ORDER_PRODUCT_STATUS tells us whether the products are made or not. Each order has a product. A product is associated with 0 or more orders.
- **FOOD-INGREDIENT(<u>FOOD-INGREDIENT_ID,FOOD_ID, INGREDIENT_ID</u>):** This relationship shows the ingredients in a food. INGREDIENT_ID and PRODUCT_ID as foreign keys as an attribute.Ther is many to many relation-ship between product(food) and ingredient. This relationship is not created for beverages.
- **SUPPLIER-INGREDIENT(<u>SUPPLIER_ID, INGREDIENT_ID</u>):** This relationship shows the source supplier of the ingredient. INGREDIENT_ID and SUPPLIER_ID as foreign keys as an attribute.There is one to many relationship. One ingredient can be obtained from one supplier, and one supplier can supply more than one ingredient.

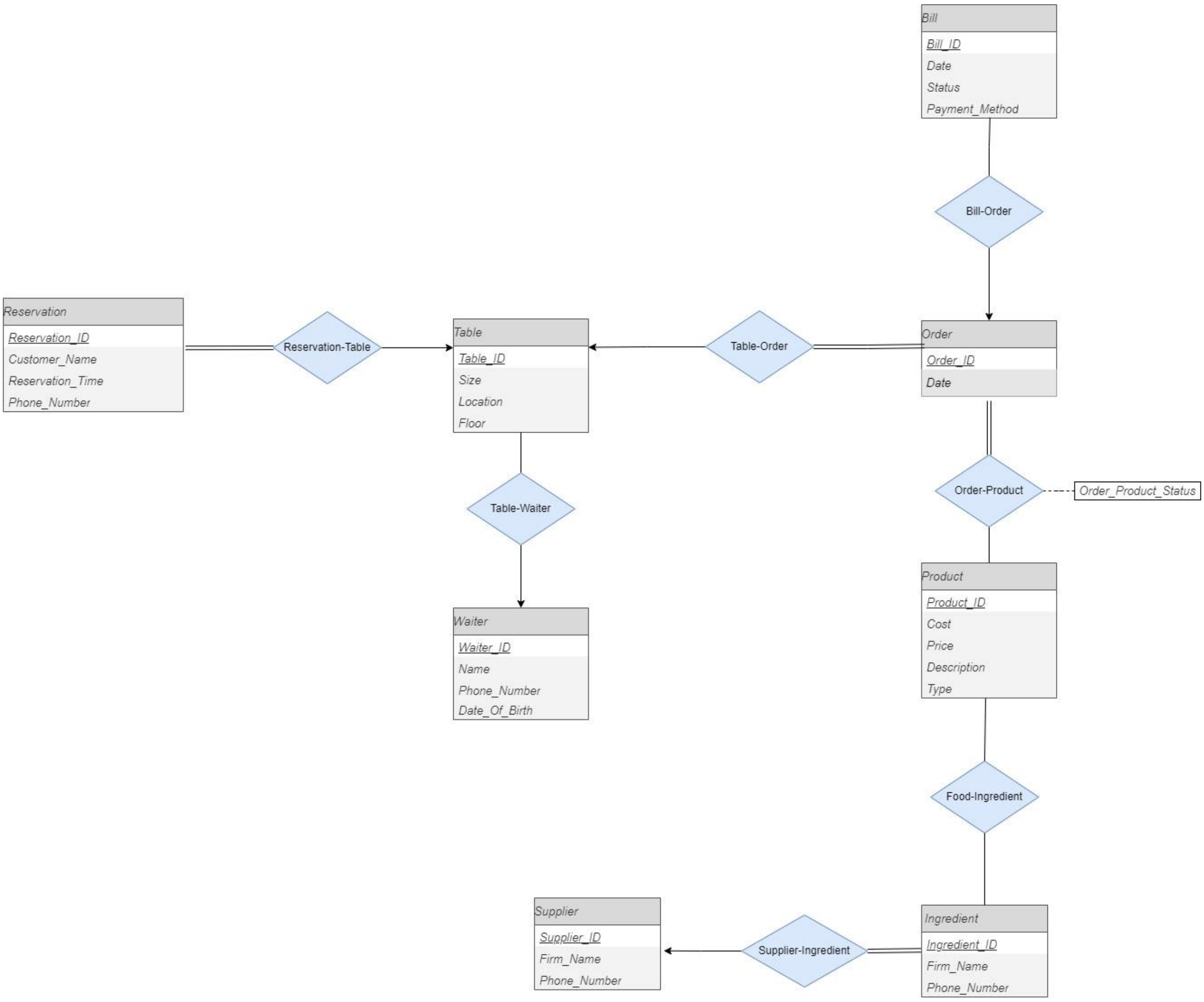
Users

- Waiter
- Restaurant Manager
- Cashier
- Cheff

Business Rules

- Waiters must be over the age of 18.
- Product prices cannot be less than zero.
- After the order is prepared, the order status must be updated as prepared.
- Existing bills and orders cannot be assigned to more than one table.
- Two reservations at the same time cannot be made for the same table.
- Table location should be one of the garden, inside and terrace.
- The order status must be prepared or unprepared.
- The bill status must be paid or unpaid.
- Product cost cannot be less than zero.
- The product type must be beverage or food.
- Payment type must be credit card or cash.
- Table size can not be less than one.
- Floor cannot be more than five.
- If a table is removed, reservations for it should be removed as well.
- If a product is a beverage it cannot include ingredients.

ER Diagram:



RELATIONAL SCHEMA

Entity Sets

- TABLE(TABLE_ID, WAITER_ID, TABLE_SIZE, TABLE_LOCATION, TABLE_FLOOR)
- RESERVATION(RESERVATION_ID, TABLE_ID, RESERVATION_CUSTOMER_NAME, RESERVATION_TIME, RESERVATION_PHONE_NUMBER)
- WAITER(WAITER_ID, WAITER_NAME, WAITER_PHONE_NUMBER, WAITER_DATE_OF_BIRTH)
- ORDER(ORDER_ID, TABLE_ID, ORDER_DATE)
- PRODUCT(PRODUCT_ID, PRODUCT_COST, PRODUCT_PRICE, PRODUCT_DESCRIPTION, PRODUCT_TYPE)
- BILL(BILL_ID, ORDER_PRODUCT_ID, BILL_DATE, BILL_STATUS, BILL_PAYMENT_METHOD)

- INGREDIENT(INGREDIENT_ID,SUPPLIER_ID, INGREDIENT_NAME)
- SUPPLIER(SUPPLIER_ID, SUPPLIER_NAME, SUPPLIER_TELEPHONE_NUMBER)

Relationship Sets

- ~~RESERVATION-TABLE(RESERVATION_ID, TABLE_ID~~**)**
- ~~TABLE-WAITER(TABLE_ID, WAITER_ID~~**)**
- ~~TABLE_ORDER(TABLE_ID, ORDER_ID)~~
- ~~BILL_ORDER(BILL_ID, ORDER_ID~~**)**
- ~~SUPPLIER-INGREDIENT(SUPPLIER_ID,INGREDIENT_ID)~~
- ORDER-PRODUCT(ORDER_PRODUCT_ID, ORDER_ID, PRODUCT_ID, ORDER_PRODUCT_STATUS)
- FOOD-INGREDIENT(FOOD_ID, INGREDIENT_ID , FOOD_INGREDIENT_ID)

REFERENCES:

https://www.techtarget.com/searchdatamanagement/definition/data-management

https://www.tableau.com/learn/articles/what-is-data-management#:~:text=Data%20management%20helps%20minimize%20potential,market%20changes%20and%20customer%20needs.

https://bloomfire.com/blog/data-vs-information/

https://www.posist.com/restaurant-times/features/data-management-crucial-restaurants-2.html