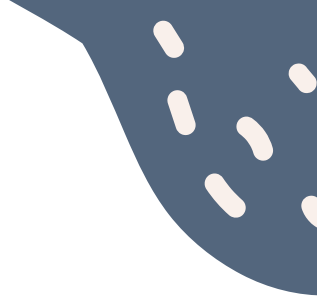


Department of Mathematics

M.Sc. Mathematics & Computing

D.B.M.S. Project

Course Name: Database Management System

Student Name: Nishant Kanwar

Roll No.: 3024030021

Instructor: Professor Anil Vashisht

Semester: 2nd

Project Overview

The Pizza Delivery System is a database management system (DBMS) project designed to streamline the operations of a pizza delivery service. The goal is to design and implement a relational database that efficiently stores, manages, and traces key information related to customers, pizzas, and orders. This system enables a pizza shop to manage its business operations with ease, maintain up to date records, and monitor the status of deliveries effectively.

By leveraging core DBMS concepts such as normalization, relational integrity, triggers, and sequences, this project demonstrates how a real world food delivery service can be structured into a robust and scalable database solution

System Requirements and Features

The database system must support the following core functionalities:

1. **Customer Management**

The system will maintain a detailed customer database, which includes:

* Name of the customer
* Phone number (used as a unique contact reference)
* Delivery address

This information is essential for placing and delivering orders accurately and is designed to avoid redundancy and ensure data consistency through normalization.

2. **Menu Management**

The pizza menu will be stored and managed in a dedicated table, including:

* Pizza Name (e.g., Margherita, Pepperoni, Veggie Delight)
* Size (Small, Medium, Large)
* Price (based on size and type)

This allows for dynamic updates to the menu and provides a reliable reference for order placements and billing.

3. **Order Processing**
The system will record and track customer orders. Each order includes:

* Order ID (automatically generated using a sequence)
* Customer ID (linked to the customer table)
* Pizza ID (linked to the menu table)
* Quantity
* Order Date
* Delivery Status

This relational setup ensures referential integrity and enables efficient tracking of order details.

Project Objectives:

This project serves multiple academic and practical goals:

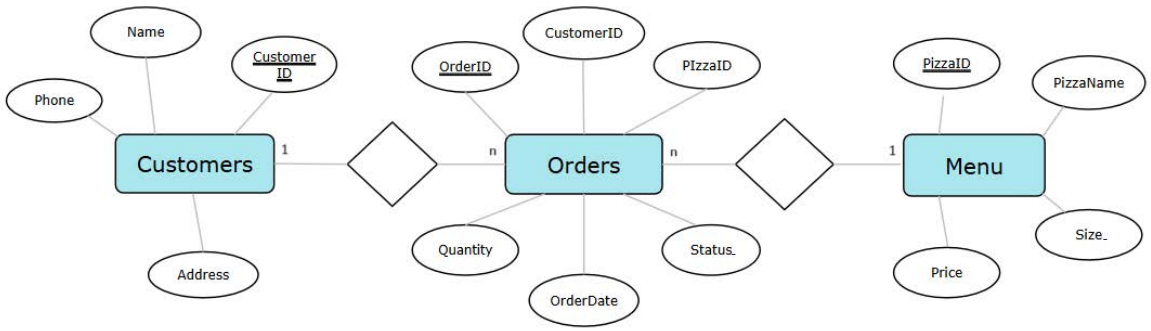
* Demonstrate understanding of relational database design and normalization.
* Apply DBMS concepts such as triggers, sequences, foreign keys, and stored procedures.
* Develop a system that reflects real-world requirements of a food delivery business.
* Emphasize data accuracy, efficiency, and automation.

Conclusion:

The Pizza Delivery System DBMS project provides a functional, organized, and efficient way to handle the core operations of a pizza delivery business.

----------x----------

ER diagram:



Normalization of the Pizza Delivery System Database:

First Normal Form (1NF)

* All tables contain only atomic (indivisible) values.

* There are no repeating groups or multi valued attributes.

Second Normal Form (2NF)

* The database is already in 1NF.

* All non key columns must depend on the entire primary key, not part of it.

* We use single_column primary keys (no composite keys), so partial dependencies cannot occur.

Third Normal Form (3NF)

* The database is in 2NF.

* No non key column is transitively dependent on the primary key (i.e., non key columns do not depend on other non key columns).

The Pizza Delivery System schema meets 3NF:

1NF - atomic values, unique rows.

2NF - no partial dependencies .

3NF - no transitive dependencies .

Each customer has only one name, one phone number, and one address stored separately. Similarly, each order stores one pizza ID and one quantity.

In the Orders table, attributes like CustomerID, PizzaID, Quantity, OrderDate, and Status all depend on the full OrderID and not on a part of it.

In the Orders table, the Status depends only on OrderID and not indirectly on CustomerID or PizzaID.

After normalization:

Data redundancy is minimized.

Data integrity is maintained.

The database becomes easier to update and manage.

An example of table Orders is given below,you can find the sample data below.

ORDERID	CUSTOMERID	PIZZAID	QUANTITY	ORDERDATE	STATUS_
1	1	2	1	02-MAY-25	Order Placed
2	2	1	2	02-MAY-25	Order Placed

----------x----------

Summary of Project Components

1. **Tables and Keys:**

a. Customers
- Primary Key: CustomerID
- Other Columns: Name, Phone, Address

b. Menu
- Primary Key: PizzaID
- Other Columns: PizzaName, Size, Price

c. Orders
- Primary Key: OrderID
- Foreign Keys:
  - CustomerID → Customers(CustomerID)
  - PizzaID → Menu(PizzaID)
- Other Columns: Quantity, OrderDate, Status

2. Procedure used:

UpdateStatus

Its takes input of two things: Orderid and it's status then changes the status of the order which was previously set by a using a trigger.

3.Triggers used:

OrderGiven

Whenever a new order details are entered in table orders,this trigger automatically sets the status of that new order to order placed.

Autoval

It uses a sequence to automatically set the order id,which would make it a more faster and convinient way to insert order details.

4.Sequence used:

SEQ

It is used in OrderID,it starts from 1,has an increment of 1 and has no cache or cycle.

Statements:

Tables

```
CREATE TABLE Customers (
   CustomerID NUMBER PRIMARY KEY,
   Name VARCHAR2(100),
   Phone VARCHAR2(15),
   Address VARCHAR2(200)
);
CREATE TABLE Menu (
     PizzaID NUMBER PRIMARY KEY,
     PizzaName VARCHAR2(100),
     Size_ VARCHAR2(20),
     Price NUMBER(6,2)
);
CREATE TABLE Orders (
   OrderID NUMBER PRIMARY KEY,
   CustomerID NUMBER,
   PizzaID NUMBER,
   Quantity NUMBER,
   OrderDate DATE,
   Status_ VARCHAR2(30),
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
   FOREIGN KEY (PizzaID) REFERENCES Menu(PizzaID)
);
```

Sample data

```
-- Insert Customers
INSERT INTO Customers VALUES (1, 'Ravi Kumar',
'9876543210', 'Delhi');

INSERT INTO Customers  VALUES (2, 'Anjali Mehra',
'9988776655', 'Mumbai');

-- Insert Pizzas into Menu
INSERT INTO Menu VALUES (1, 'Margherita', 'Medium', 250);

INSERT INTO Menu VALUES (2, 'Pepperoni', 'Large', 450);

INSERT INTO Menu VALUES (3, 'Farmhouse', 'Small', 200);

-- Insert Orders
INSERT INTO Orders (CustomerID, PizzaID, Quantity, OrderDate)
VALUES (1, 2, 1, SYSDATE);

INSERT INTO Orders (CustomerID, PizzaID, Quantity, OrderDate)
VALUES ( 2, 1, 2, SYSDATE);
```

Procedure,triggers and sequence:

1. Procedure:

```
CREATE OR REPLACE PROCEDURE UpdateStatus (
   p_OrderID IN NUMBER,
   p_Status IN VARCHAR2
) AS
BEGIN
   UPDATE Orders
   SET Status_ = p_Status
WHERE OrderID = p_OrderID;

   COMMIT;
END;
```

Execution:

```
BEGIN
   UpdateStatus(1, 'Delivered');
END;
```

Before Execution:

ORDERID	CUSTOMERID	PIZZAID	QUANTITY	ORDERDATE	STATUS_
1	1	2	1	02-MAY-25	Order Placed
2	2	1	2	02-MAY-25	Order Placed

After Execution:

ORDERID	CUSTOMERID	PIZZAID	QUANTITY	ORDERDATE	STATUS_
1	1	2	1	02-MAY-25	Delivered
2	2	1	2	02-MAY-25	Order Placed

2. Triggers:

```
CREATE OR REPLACE TRIGGER OrderGiven
BEFORE INSERT ON Orders
FOR EACH ROW
BEGIN
   :NEW.Status_ := 'Order Placed';
END;
```

```
CREATE OR REPLACE TRIGGER Autoval
   BEFORE INSERT ON Orders
   FOR EACH ROW
   BEGIN
   IF :NEW.OrderID is NULL THEN
   :NEW.OrderID :=SEQ.NEXTVAL;
END IF;
END;
```

3. Sequence:

```
CREATE SEQUENCE SEQ
   START WITH 1
   INCREMENT BY 1
   NOCACHE
   NOCYCLE;
```

----------x----------