Czech University of Life Sciences

Faculty of Economics and Management

Database Systems project:

Database for Flowers E-Shop

Autor: Adina Sarsenbayeva

Introduction

This project designs a database for Flowers E-Shop. It captures basic information which can be used in e-shop management system. This project provides a clear and comprehensive overview of the data and processes involved in the business, and it serves as a useful reference for designing and maintaining the database for the e-shop. It's based on following assumptions:

- Each customer can place many orders, each order can have multiple order items.
- Each flower can be part of multiple order items, each order item can be flowers.
- Each order can be a subject of payment (bought/sold).
- Each order has one delivery, each delivery belongs to only one order.
- Each supplier can supply many flowers.

This project is a simple example of an e-shop management system. It does not solve the problem of flower storage. It may also have additional entities and important attributes.

Outline

1. Possible use cases for the model

- Get the total number of flowers of each type that have been ordered
- Find the names of customers who have ordered flowers from a specific supplier
- Find the names of customers who have placed at least one order
- Find the names of all customers and delivery addresses who have received flowers within a specific date
- Find the names and descriptions of flowers that have been ordered by a specific customer
- Find all customers name who spend more than 500 CZK

2. Entity relationship diagrams

Following section captures the proposed structure of database using entity relationship diagram. The diagrams were created in Lucidchart.

2.1. Conceptual ERD

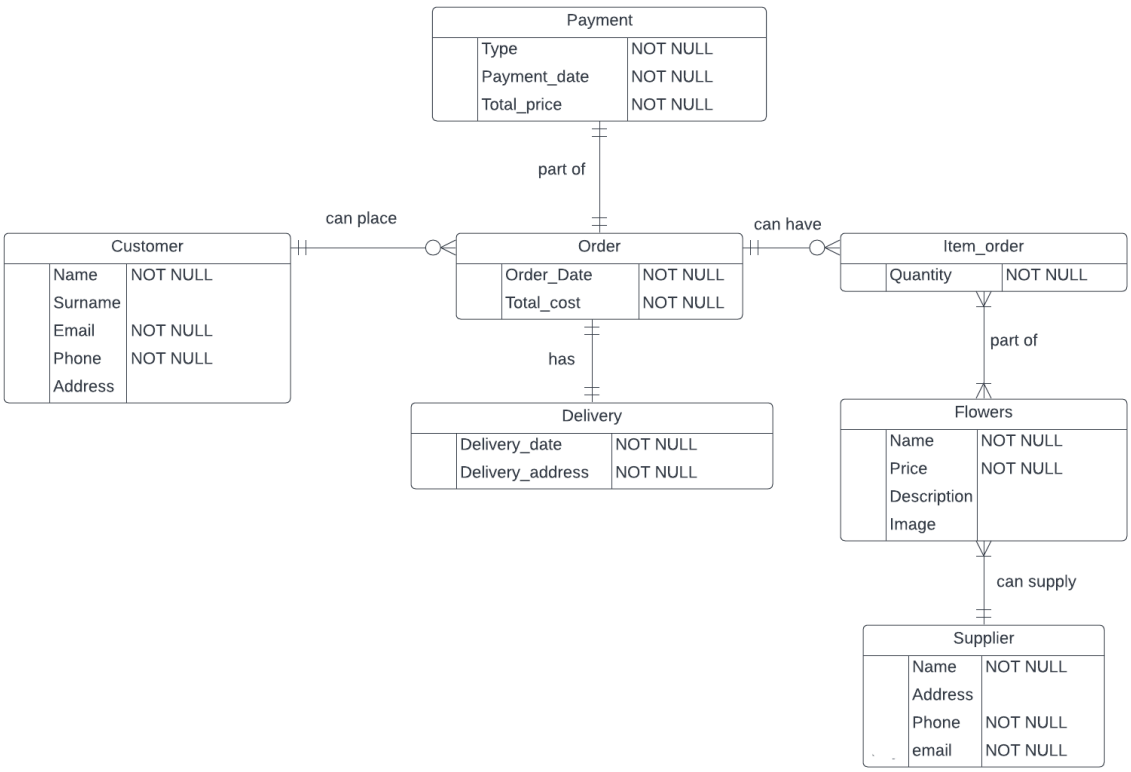

Figure 1: Conceptual ERD

2.2. Logical ERD

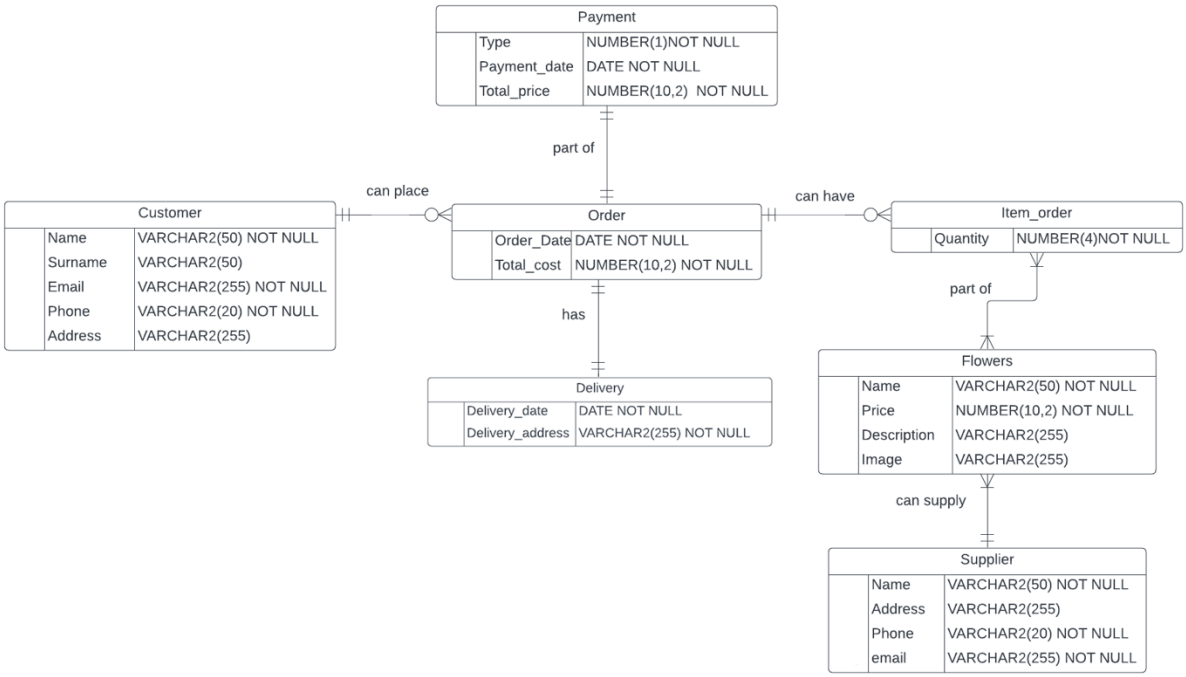

Figure 2: Logical ERD

2.3. Physical ERD

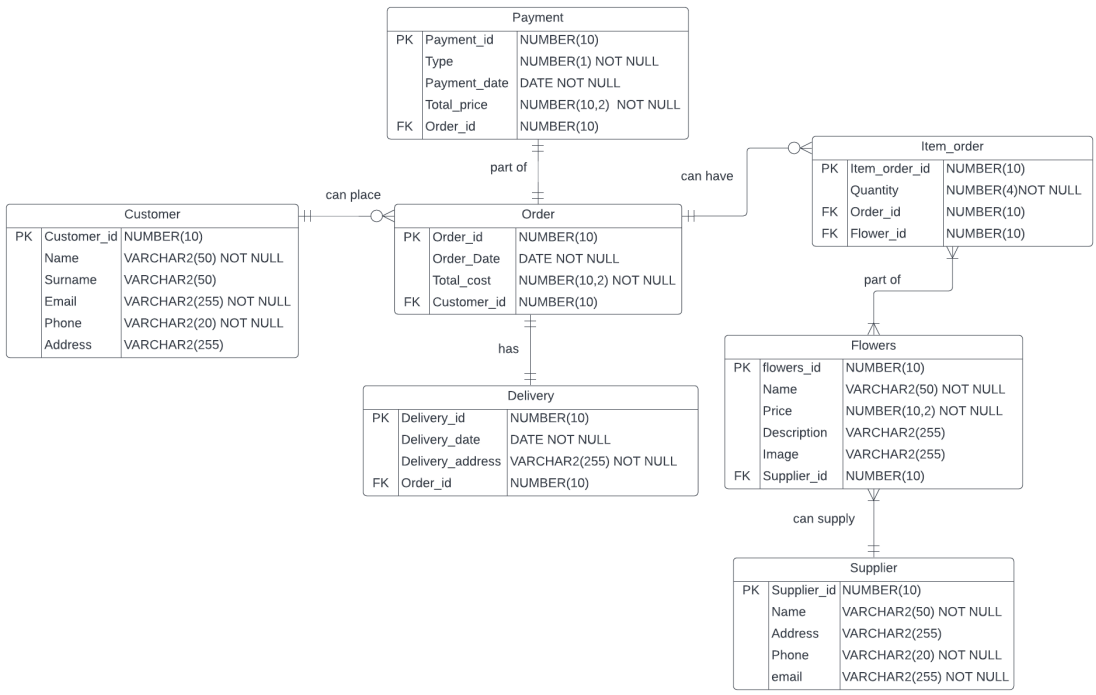

Figure 3: Physical ERD

3. SQL Implementation

The database was implemented in Oracle Application Express, which uses Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production as a DBMS.

3.1. DDL : Defining the database objects

CREATE TABLE Customer (Customer_id NUMBER(10), Name VARCHAR2(50) NOT NULL, Surname VARCHAR2(50), Email VARCHAR2(255) NOT NULL, Phone VARCHAR2(20) NOT NULL, Address VARCHAR2(255), PRIMARY KEY (Customer_id));

CREATE TABLE Orders (Order_id NUMBER(10), Order_Date DATE NOT NULL, Total_cost NUMBER(10,2) NOT NULL, Customer_id NUMBER(10), PRIMARY KEY (Order_id));

CREATE TABLE Flowers (flowers_id NUMBER(10), Name VARCHAR2(50) NOT NULL, Price NUMBER(10,2) NOT NULL, Description VARCHAR2(255), Image VARCHAR2(255), Supplier_id NUMBER(10), PRIMARY KEY (flowers_id));

CREATE TABLE Item_order (Item_order_id NUMBER(10), Quantity NUMBER(4)NOT NULL, Order_id NUMBER(10), Flower_id NUMBER(10), PRIMARY KEY (Item_order_id));

CREATE TABLE Delivery (Delivery_id NUMBER(10), Delivery_date DATE NOT NULL, Delivery_address VARCHAR2(255) NOT NULL, Order_id NUMBER(10), PRIMARY KEY (Delivery_id));

CREATE TABLE Supplier (Supplier_id NUMBER(10), Name VARCHAR2(50) NOT NULL, Address VARCHAR2(255), Phone VARCHAR2(20) NOT NULL, email VARCHAR2(255) NOT NULL, PRIMARY KEY (Supplier_id));

CREATE TABLE Payment (Payment_id NUMBER(10), Type NUMBER(1) NOT NULL, Payment_date DATE NOT NULL, Total_price NUMBER(10,2) NOT NULL, Order_id NUMBER(10), PRIMARY KEY (Payment_id));

Constraints:

ALTER TABLE Orders ADD CONSTRAINT "can place" FOREIGN KEY (Customer_id) REFERENCES Customer (Customer_id);

ALTER TABLE Delivery ADD CONSTRAINT "has" FOREIGN KEY (Order_id) REFERENCES Orders (Order_id);

ALTER TABLE Payment ADD CONSTRAINT "part of" FOREIGN KEY (Order_id) REFERENCES Orders (Order_id);

ALTER TABLE Flowers ADD CONSTRAINT "can supply" FOREIGN KEY (Supplier_id) REFERENCES Supplier (Supplier_id);

ALTER TABLE Item_order ADD CONSTRAINT "part_of" FOREIGN KEY (Order_id) REFERENCES Orders (Order_id);

ALTER TABLE Item_order ADD CONSTRAINT "can_have" FOREIGN KEY (flower_id) REFERENCES Flowers (flowers_id);

3.2. DML: Inserting the data (examples)

INSERT INTO Customer(Customer_id, Name, Surname, Email, Phone, Address) VALUES(1, 'Anna', 'Jarkovska', 'annaJ@gmail.com', '+420456678234', 'Praha, Nove Mesto, Stepanska 1, 11000');

INSERT INTO Customer(Customer_id, Name, Surname, Email, Phone, Address) VALUES(2, 'Martin', 'Pavlicek', 'MartinP@gmail.com', '+420345234123', 'Praha, Dejvice, Buzulucka 8, 16000');

INSERT INTO Customer(Customer_id, Name, Surname, Email, Phone, Address) VALUES(3, 'Pavel', 'Umarov', 'PavelU@gmail.com', '+4203453289', 'Praha, Nove Mesto, Italska 6, 11000');

INSERT INTO Orders(Order_id, Order_date, Total_cost, Customer_id) VALUES(1, '12/13/2022', 500, 3);

INSERT INTO Orders(Order_id, Order_date, Total_cost, Customer_id) VALUES(2, '12/20/2022', 400, 1);

INSERT INTO Orders(Order_id, Order_date, Total_cost, Customer_id) VALUES(3, '12/23/2022', 1000, 2);

INSERT INTO Delivery(Delivery_id, Delivery_date, Delivery_address, Order_id) VALUES(1, '12/14/2022', 'Praha, Nove Mesto, Italska 6, 11000', 1);

INSERT INTO Delivery(Delivery_id, Delivery_date, Delivery_address, Order_id) VALUES(2, '12/20/2022', 'Praha, Nove Mesto, Stepanska 1, 11000',2);

INSERT INTO Delivery(Delivery_id, Delivery_date, Delivery_address, Order_id) VALUES(3, '12/24/2022', 'Praha, Nove Mesto, Pricna 8, 11000', 3);

INSERT INTO Payment(Payment_id, Type, Payment_date, Total_price, Order_id) VALUES(1, 1, '12/14/2022', 550, 1);

INSERT INTO Payment(Payment_id, Type, Payment_date, Total_price, Order_id) VALUES(2, 1, '12/20/2022', 450, 2);

INSERT INTO Payment(Payment_id, Type, Payment_date, Total_price, Order_id) VALUES(3, 1, '12/23/2022', 1050, 3);

INSERT INTO Supplier(Supplier_id, Name, Address, Phone, Email) VALUES(1, 'Kvetiny po Praze', 'Praha, Nove Mesto, Italska 8, 11000', '+420456768456', 'KevityPraha@gmail.com');

INSERT INTO Supplier(Supplier_id, Name, Address, Phone, Email) VALUES(2, 'Best Flowers', 'Praha, Dejvice, Buzulucka 10, 11000', '+420455345666', 'BestF@gmail.com');

INSERT INTO Flowers(flowers_id, Name, Price, Description, Image, Supplier_id) VALUES(1, 'Red Rose', 100, 'Flowers for your love!', 'www.flowersdelivery.cz/rose.jpeg', 1);

INSERT INTO Flowers(flowers_id, Name, Price, Description, Image, Supplier_id) VALUES(2, 'Peony', 50, 'Symbolic of love, honor.', 'www.flowersdelivery.cz/peony.jpeg',1);

INSERT INTO Flowers(flowers_id, Name, Price, Description, Image, Supplier_id) VALUES(3, 'Chamomile', 20, 'May all your dreams and wishes be fulfilled', 'www.flowersdelivery.cz/ch.jpeg',2);

INSERT INTO Item_order(item_order_id, Quantity, Order_id, Flower_id) VALUES(1, 25, 1, 3);

INSERT INTO Item_order(item_order_id, Quantity, Order_id, Flower_id) VALUES(2, 8, 2, 2);

INSERT INTO Item_order(item_order_id, Quantity, Order_id, Flower_id) VALUES(3, 10, 3, 1);

3.3. SQL Queries

- Get the total number of flowers of each type that have been ordered

SELECT f.name, SUM(io.quantity) as total_quantity FROM Item_order io INNER JOIN Flowers f ON io.flower_id = f.flowers_id GROUP BY f.name

- Find the names of customers who have ordered flowers from a specific supplier

SELECT c.name FROM Customer c INNER JOIN Orders o ON c.customer_id = o.customer_id INNER JOIN Item_order io ON o.order_id = io.order_id INNER JOIN Flowers

f ON io.flower_id = f.flowers_id INNER JOIN Supplier s ON f.supplier_id = s.supplier_id WHERE s.name = 'Kvetiny po Praze' GROUP BY c.name

- Find the names of customers who have placed at least one order

SELECT c.name FROM Customer c INNER JOIN Orders o ON o.customer_id = c.customer_id

- Find the names of all customers and delivery addresses who have received flowers within a specific date

SELECT c.name FROM Customer c INNER JOIN Orders o ON o.customer_id = c.customer_id INNER JOIN Delivery d On d.order_id = o.order_id WHERE d.Delivery_date = '12/14/2022'

- Find the names and descriptions of flowers that have been ordered by a specific customer

SELECT f.name, f.description FROM Flowers f INNER JOIN item_order io ON io.flower_id = f.flowers_id INNER JOIN Orders o ON o.order_id = io.order_id INNER JOIN Customer c ON c.customer_id = o.customer_id WHERE c.name = 'Anna'

- Find all customers name who spend more than 500 CZK

SELECT c.name FROM Customer c INNER JOIN Orders o ON o.customer_id = c.customer_id INNER JOIN Payment p ON p.order_id = o.order_id WHERE p.total_price > 500

4. Conclusion

In conclusion, this work designs a database that contains essential information about e-shop management system. The database is organized into three main categories: people, process, and product and it includes several entities that represent the various components of the business, such as customers, orders, flowers, and deliveries. This project contains definitions of important database objects, and example of possible use cases realised in the form of SQL queries. Based on this, we can assume that this project has basic entities, attributes and use cases, so it can be improved and made into a full-scale system.