Databasmodellering
Eshop

2024-01-30

av
Senai Amanuel, Elias Bakhshi och Daniel Mohagheghifard

TABLE OF CONTENT

Introduktion

En databasmodell av egen shop enligt kokboken [1].

Konceptuell modellering (kmom03)

Beskriv databasen i ett textstycke

Databasen är utformad för att stödja en e-handelsplattform och omfattar flera entiteter som är avgörande för hantering av **kunder**, **produkter**, **order**, **produktkategorier**, **lager**, **leveranser** och **fakturor**. Huvudentiteter inkluderar **kundregister**, **produktregister**, **produktkategorier**, **lagerregister**, **orderhantering**, **produkter i en order**, **leveranshantering**, **lagerhållning slog** och **fakturering**. Varje kund kan registreras med detaljer såsom namn, kontaktinformation och adress i **kundregistret**. Produkter kan registreras med information såsom produktbeskrivning, pris och lagerstatus i **produktregistret**. **Produktkategorierna** möjliggör organisering av produkter i olika kategorier för enklare navigering och sökning. **Lagerregistret** spårar lagerstatus för varje produkt och dess plats i lagret. **Orderhanteringen** tillåter skapande och uppdatering av order, medan **produkter i en order**-tabellen sparar detaljer om varje produkt som ingår i en order. **Leveranshanteringen** möjliggör skapande och spårning av leveranser baserat på befintliga ordrar. **Lagerhållningsloggen** registrerar händelser relaterade till lagerförvaltning. Slutligen genererar **faktureringsfunktionen** fakturor baserat på avslutade order.

Skriv ned alla entiteter

Här är en lista över entiteterna baserat på eshop:

Här är alla entiteter listade:

1. **Kundregister (Customer):**
   - Innehåller information om kunder, inklusive deras unika kund-ID, förnamn, efternamn, e-post, lösenord, adress och telefonnummer.

2. **Produktregister (Product):**
   - Lagrar detaljer om produkter som finns tillgängliga på plattformen, inklusive produkt-ID, beskrivning, namn, pris och lagerantal.

3. **Produktkategorier (Category):**
   - Representerar olika kategorier som produkter kan klassificeras i. Den inkluderar en unik kategori-ID och kategorinamn.

4. **Kopplingstabell för Produkt och Kategori (Product_Category):**

- Detta är en kopplingstabell som används för att etablera en många-till-många-relation mellan produkter och kategorier. Den inkluderar ID:n för både produkten och kategorin och bildar en sammansatt primärnyckel.

5. **Lagerregister (Warehouse):**
   - Hanterar lagerinventeringen genom att lagra information om lagret, som dess unika ID, ID för den lagrade produkten, hyllplacering och lagerkvantitet.

6. **Orderhantering (Order):**
   - Spårar information om ordrar som gjorts av kunder. Den inkluderar en order-ID, orderdatum, totalpris, kund-ID och status som indikerar orderns framsteg (t.ex. avvaktande, behandlas, avslutad, avbruten).

7. **Produkter i en order (Order_item):**
   - Representerar enskilda produkter inom en order. Den inkluderar en unik ID för orderposten, order-ID, produkt-ID, kvantitet och pris.

8. **Leveranshantering och plocklista (Delivery):**
   - Hanterar leveransprocesserna genom att spåra leveransinformation såsom leverans-ID, order-ID, leveransdatum och status (t.ex. packad, skickad, levererad).

9. **Lagerhållningslog (Inventory_Log):**
   - Underhåller en logg över lagerrelaterade händelser, inklusive händelse-ID, händelsebeskrivning och händelsedatum.

10. **Fakturering (Invoice):**
    - Hanterar faktureringsprocesserna genom att lagra faktureringsinformation såsom faktura-ID, order-ID, faktureringsdatum och totalpris.

Dessa entiteter tillsammans bildar databasens schema för en e-handelsplattform och möjliggör effektiv hantering av kunder, produkter, ordrar, lager, leveranser och fakturering.

:

Skriv ned alla relationer och visa i matris

```

**Beskrivning**:
```

Relationer:
Här är en mer detaljerad beskrivning av relationerna mellan entiteterna både framåt och bakåt:

1. **Customer - Order:**
   - Framåt: En kund kan placera flera ordrar, vilket resulterar i en 1:N-relation från kund till order.
   - Bakåt: Varje order är kopplad till en specifik kund, vilket skapar en N:1-relation från order till kund.

2. **Product - Category:**
   - Framåt: En produkt kan tillhöra flera kategorier, vilket ger upphov till en N:M-relation från produkt till kategori.
   - Bakåt: Varje kategori kan ha flera produkter, vilket resulterar i en N:M-relation från kategori till produkt.

3. **Order - Order_item:**
   - Framåt: En order kan innehålla flera orderposter (order_items), vilket skapar en 1:N-relation från order till order_item.
   - Bakåt: Varje orderpost är kopplad till en specifik order, vilket resulterar i en N:1-relation från order_item till order.

4. **Product - Warehouse:**
   - Framåt: Varje produkt kan vara lagrad på flera platser i lagret, vilket ger en 1:N-relation från produkt till lager.
   - Bakåt: En lagerplats är kopplad till en specifik produkt, vilket resulterar i en N:1-relation från lager till produkt.

5. **Order - Delivery:**
   - Framåt: En order kan leda till en leverans, vilket skapar en 1:1-relation från order till leverans.
   - Bakåt: Varje leverans är kopplad till en specifik order, vilket resulterar i en 1:1-relation från leverans till order.

6. **Order - Invoice:**
   - Framåt: Varje order kan generera en faktura, vilket ger upphov till en 1:1-relation från order till faktura.
   - Bakåt: En faktura är kopplad till en specifik order, vilket resulterar i en 1:1-relation från faktura till order.

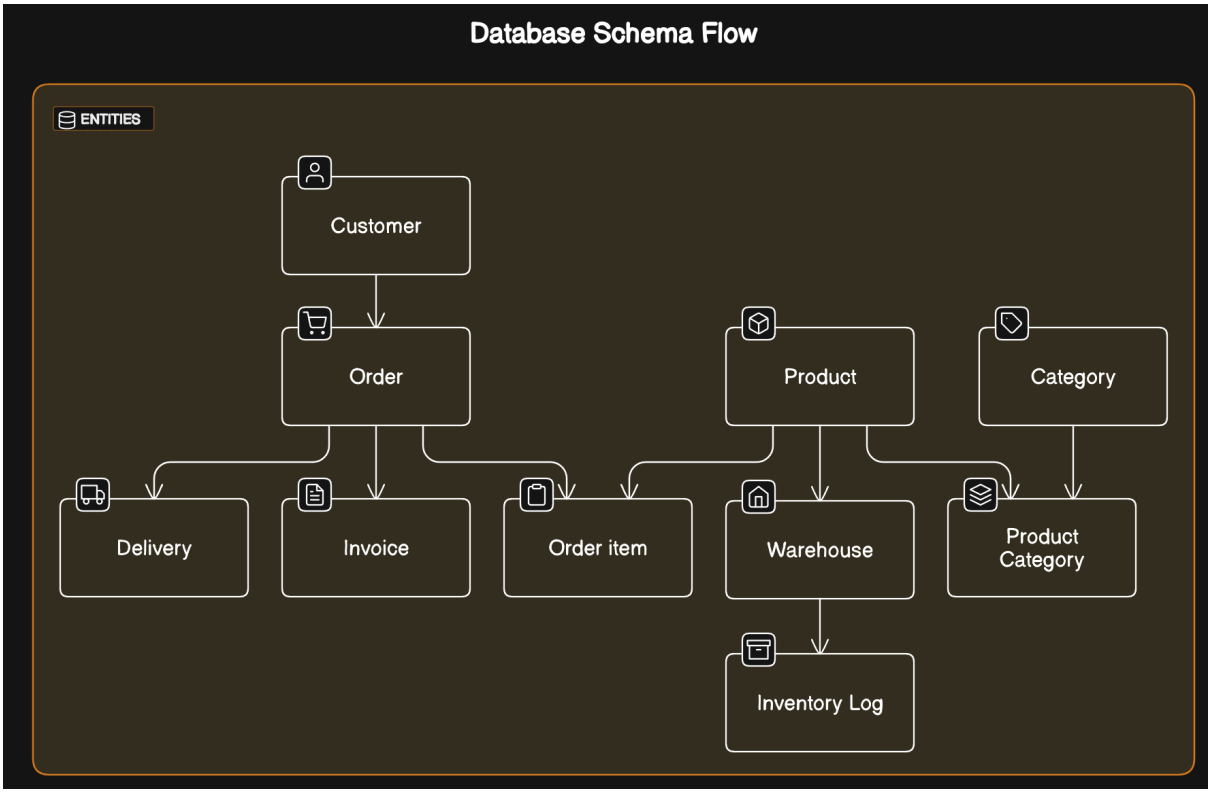
7. **Inventory_Log - Warehouse:**
   - Framåt: Lagerhållningsloggen spårar händelser relaterade till lagret, vilket skapar en 1:N-relation från lagerhållningsloggen till lager.
   - Bakåt: Varje händelse i lagerhållningsloggen är knuten till en specifik lagerplats, vilket resulterar i en N:1-relation från lager till lagerhållningsloggen.

Denna beskrivning tydliggör både fram- och bakåtrelationer mellan varje par av entiteter i databasen.

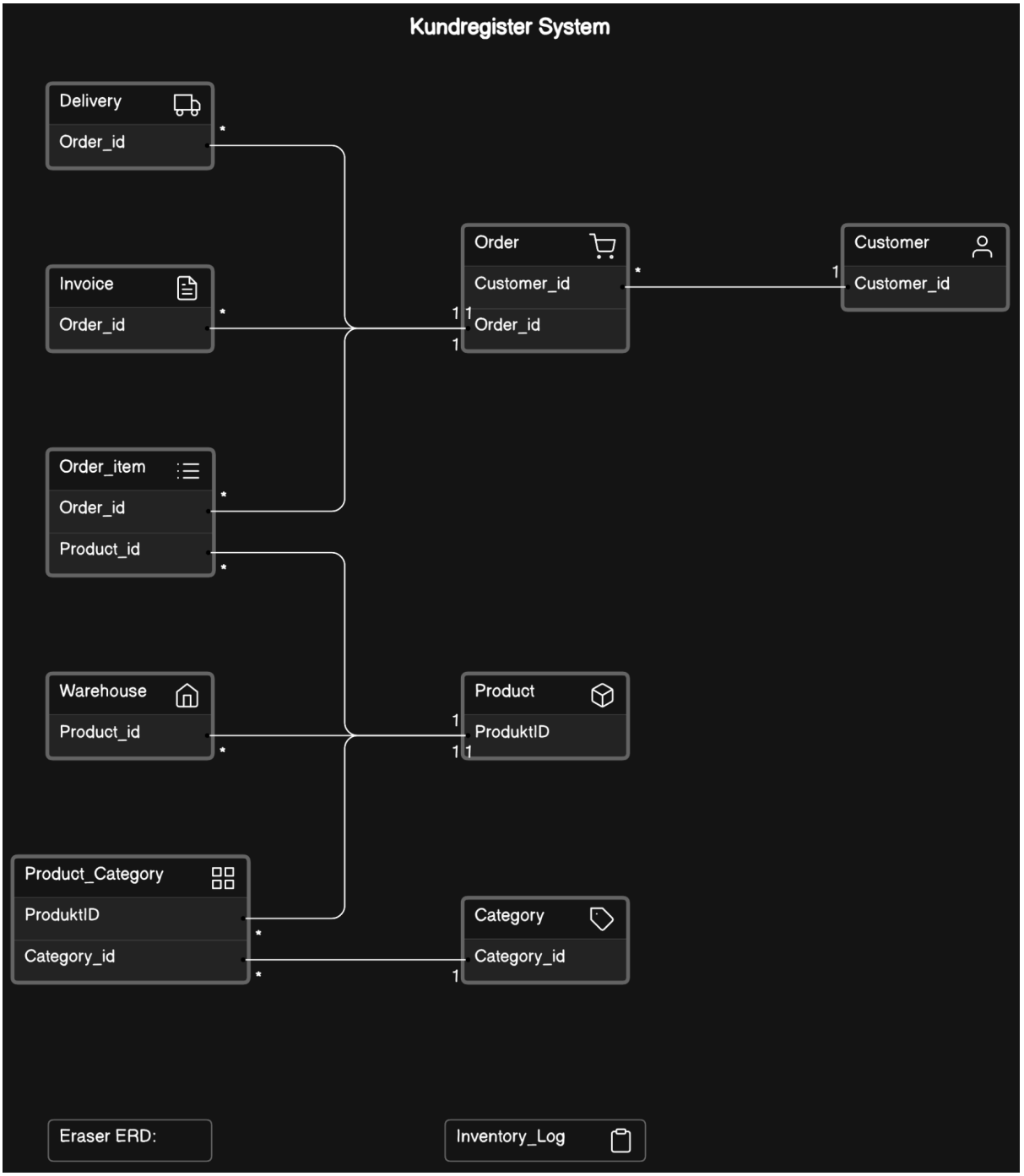
Matris

	Customer	Product	Category	Product_Category	**Warehouse**	Order	order_item	Delivery	Inventory_Log	Invoice
Customer										
Product	**X**									
Category		**X**								
Product_Category		**z**								
Warehouse										
Order										
order_item										
Delivery										
Inventory_Log										
Invoice										

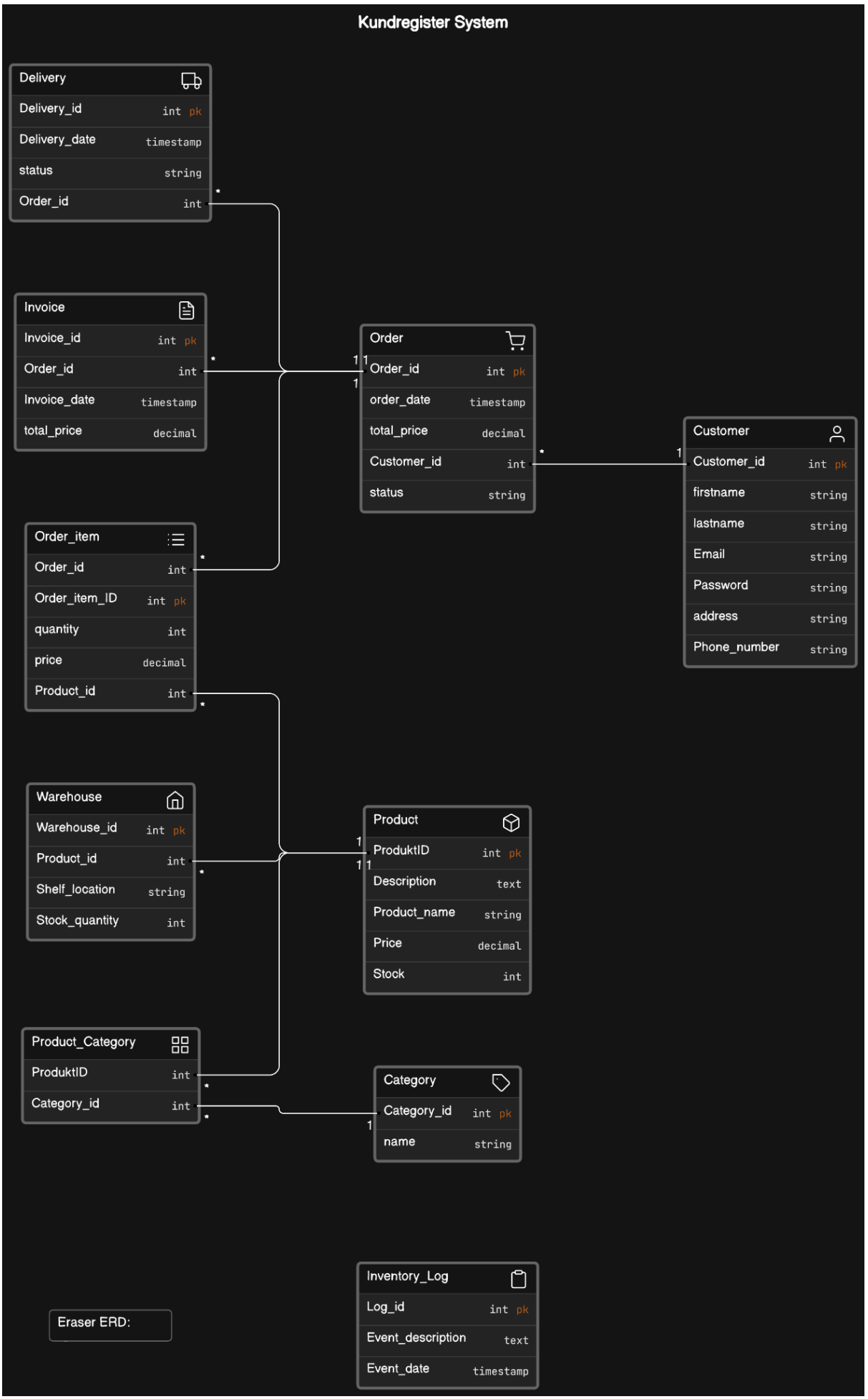
Rita enkelt ER-diagram med entiteter och relationer



Komplettera ER-diagram med kardinalitet

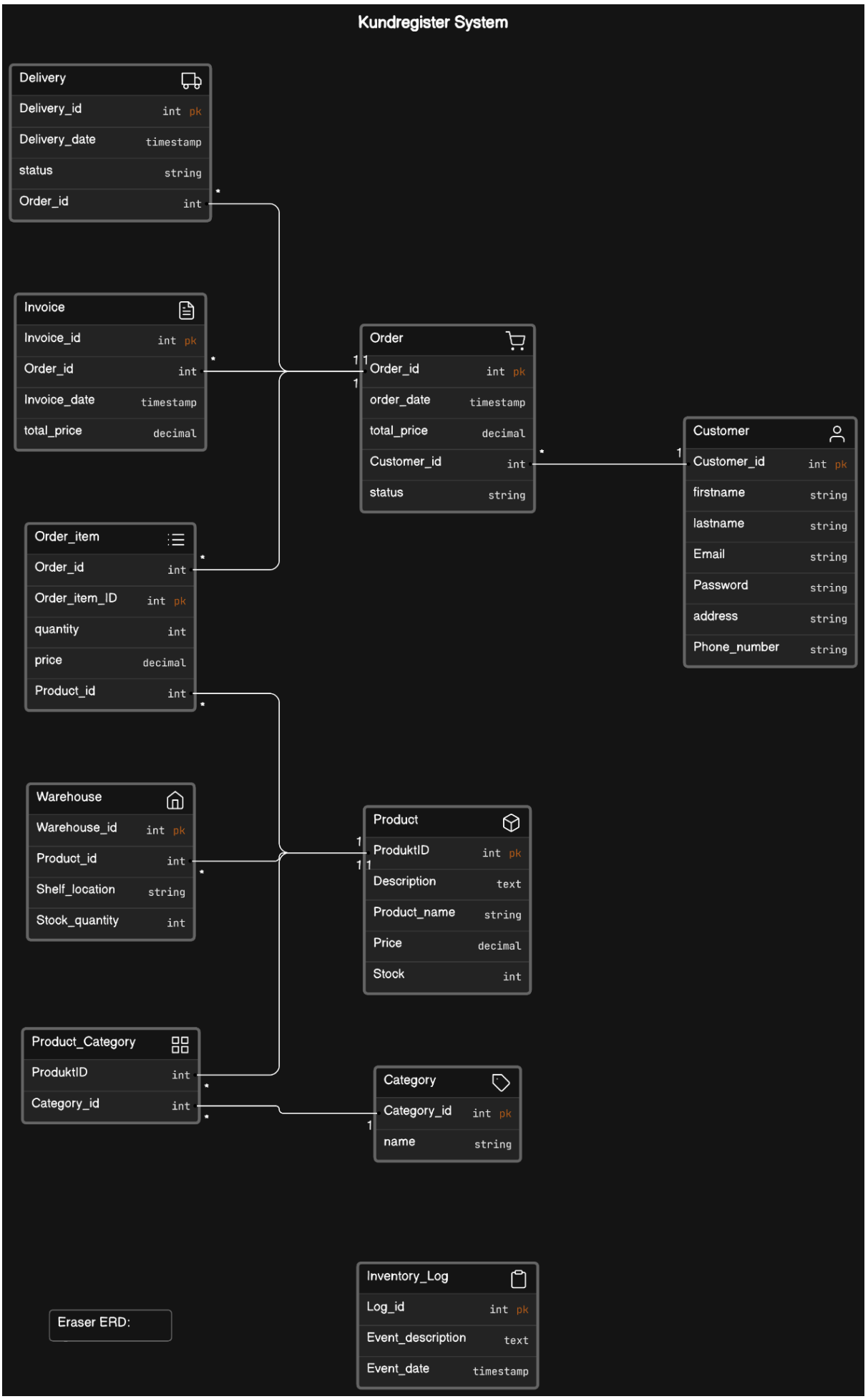


Komplettera ER-diagram med alla attribut samt kandidatnycklar

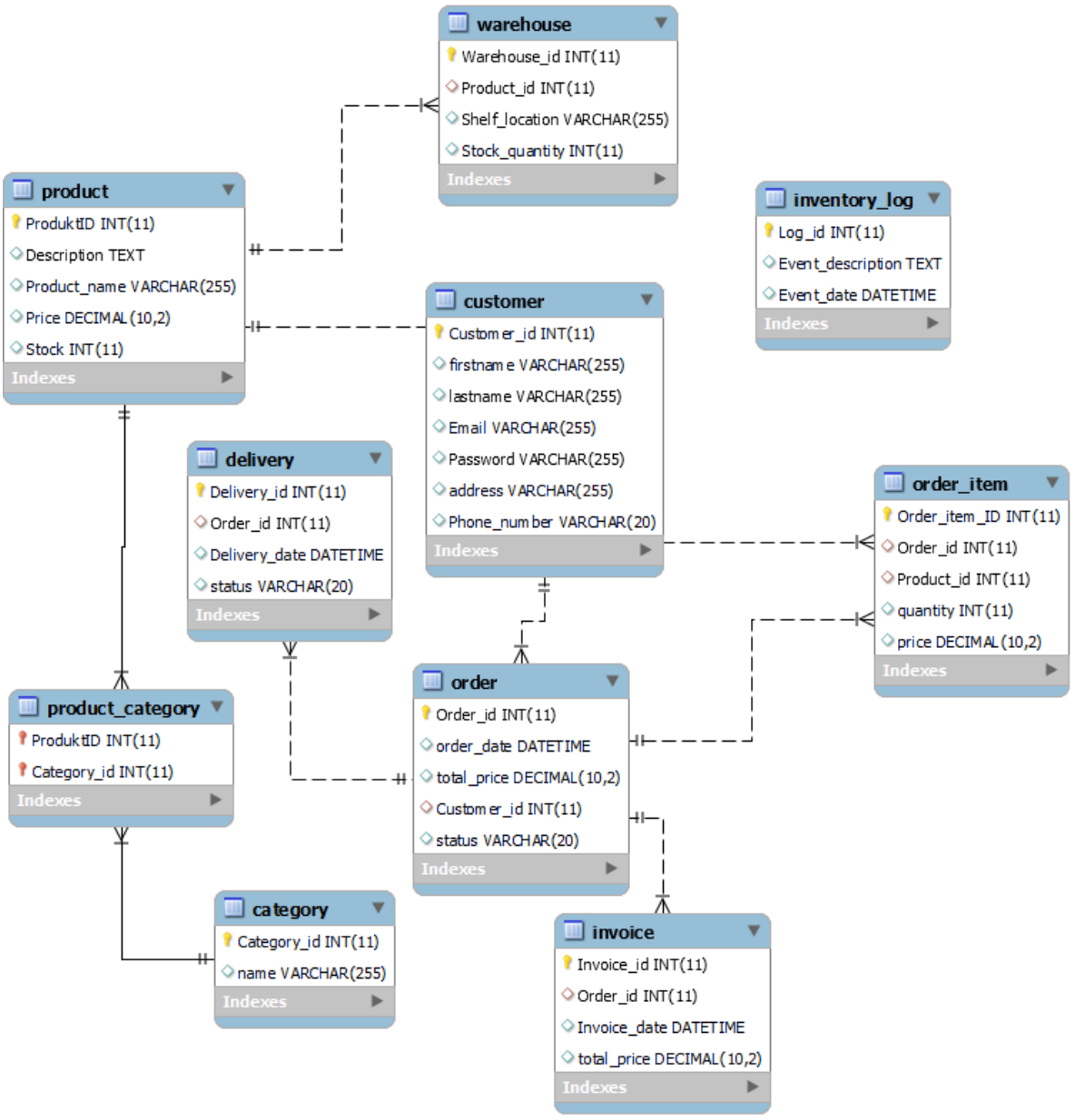

Kundregister System
Delivery
Delivery_id int pk
Delivery_date timestamp
status string
Order_id int
Invoice
Invoice_id int pk
Order_id int
Invoice_date timestamp
total_price decimal
Order
Order_id int pk
order_date timestamp
total_price decimal
Customer_id int
status string
Customer
Customer_id int pk
firstname string
lastname string
Email string
Password string
address string
Phone_number string
Order_item
Order_id int
Order_item_ID int pk
quantity int
price decimal
Product_id int
Warehouse
Warehouse_id int pk
Product_id int
Shelf_location string
Stock_quantity int
Product
ProduktID int pk
Description text
Product_name string
Price decimal
Stock int
Product_Category
ProduktID int
Category_id int
Category
Category_id int pk
name string
Inventory_Log
Log_id int pk
Event_description text
Event_date timestamp
Eraser ERD:

Logisk modellering (kmom04)

Modifiera ER-diagram enligt relationsmodellen


Kundregister System
Delivery
Delivery_id int pk
Delivery_date timestamp
status string
Order_id int
Invoice
Invoice_id int pk
Order_id int
Invoice_date timestamp
total_price decimal
Order
Order_id int pk
order_date timestamp
total_price decimal
Customer_id int
status string
Customer
Customer_id int pk
firstname string
lastname string
Email string
Password string
address string
Phone_number string
Order_item
Order_id int
Order_item_ID int pk
quantity int
price decimal
Product_id int
Warehouse
Warehouse_id int pk
Product_id int
Shelf_location string
Stock_quantity int
Product
ProduktID int pk
Description text
Product_name string
Price decimal
Stock int
Product_Category
ProduktID int
Category_id int
Category
Category_id int pk
name string
Inventory_Log
Log_id int pk
Event_description text
Event_date timestamp
Eraser ERD:

Utöka ER-diagram med primära/främmande nycklar samt kompletterande attribut



Fysisk modellering (kmom04)

Skapa SQL DDL för tabellerna

Absolut, här är ett bra svar:

Vi har genomfört en omfattande process för att skapa och strukturera våra databastabeller. Alla 10 tabellerna är nu skapade med nödvändiga primärnycklar och främmande nycklar för att säkerställa en korrekt koppling mellan dem. All vår SQL-kod har organiserats och lagrats i ddl.sql-filen för enkel åtkomst och referens. Dessutom har vi genomfört omvänd reverse engineering för att visualisera och förstå strukturen av våra tabeller och hur de är sammanlänkade.På så sätt har vi kunnat förstå allting bättre och kunnat förklara det bättre för andra.

Lista funktioner som databasen skall stödja (API)

Här är en lista över funktioner som databasen bör stödja genom dess API:

1. **Registrering och hantering av kunder:**
   - Skapa en ny kundprofil.
   - Uppdatera befintliga kunduppgifter (t.ex. kontaktinformation, lösenord).
   - Hämta kundinformation baserat på kund-ID eller e-postadress.

2. **Hantering av produkter:**
   - Lägga till nya produkter i produktregistret.
   - Uppdatera produktinformation (t.ex. namn, beskrivning, pris, lagerstatus).
   - Hämta produktinformation baserat på produkt-ID eller produktkategori.

3. **Hantering av kategorier:**
   - Skapa nya produktkategorier.
   - Uppdatera kategorinamn eller attribut.
   - Hämta kategorilista och dess tillhörande produkter.

4. **Orderhantering:**
   - Skapa en ny order för en kund.
   - Lägga till produkter i en befintlig order.
   - Uppdatera orderstatus (t.ex. behandling, leverans, slutförd, avbruten).
   - Hämta orderhistorik för en specifik kund eller order baserat på order-ID.

5. **Hantering av lager:**
   - Registrera lagra händelser (t.ex. mottagande av varor, flyttning av varor, justeringar av lagerstatus).
   - Uppdatera lagerstatus för enskilda produkter.
   - Hämta lagerstatus för produkter baserat på produkt-ID eller lagerplats.

6. **Hantering av leveranser och fakturor:**
   - Skapa en ny leverans från en befintlig order.
   - Uppdatera leveransstatus (t.ex. packad, skickad, levererad).
   - Skapa en faktura baserat på en befintlig order.
   - Hämta leverans- och faktureringsinformation baserat på order-ID.

7. **Loggning av händelser:**
   - Registrera händelser relaterade till kundinteraktioner (t.ex. inloggning, registrering, ändringar av uppgifter).

- Registrera händelser relaterade till orderhantering (t.ex. skapande, uppdatering, avbokning av order).
- Hämta loggposter baserade på händelsetyp, användar-ID eller datumintervall.

Genom att implementera dessa funktioner kan API:et möjliggöra en komplett hantering av kunder, produkter, order, lager, leveranser och fakturor inom e-handelsplattformen.

For the database to support functionality, we can define several APIs:

1. **Customer Management API:**
   - Create a new customer.
   - Retrieve customer information.
   - Update customer information.
   - Delete a customer.

2. **Order Management API:**
   - Place a new order.
   - Retrieve order details.
   - Update order details.
   - Cancel an order.

3. **Cart Management API:**
   - Add a product to the cart.
   - Remove a product from the cart.
   - Update quantity of products in the cart.
   - View cart contents.

4. **Payment Management API:**

   - Process payment for an order.

   - Retrieve payment details.

   - Refund payment.

5. **Product Management API:**

   - Add a new product.

   - Retrieve product information.

   - Update product information.

   - Delete a product.

6. **Order Item Management API:**

   - Add an item to an order.

   - Remove an item from an order.

   - Update quantity or price of an item in an order.

   - View items in an order.

7. **Category Management API:**

- Add a new category.

- Retrieve category information.

- Update category information.

- Delete a category.

APPENDIX DDL

```
drop database if exists eshop;

create database eshop;

use eshop;

-- Kundregister

CREATE TABLE Customer (

    Customer_id INT PRIMARY KEY,

    firstname VARCHAR(255),

    lastname VARCHAR(255),

    Email VARCHAR(255),

    Password VARCHAR(255),
```

```
    address VARCHAR(255),
    Phone_number VARCHAR(20)
);

-- Produktregister
CREATE TABLE Product (
    ProduktID INT PRIMARY KEY,
    Description TEXT,
    Product_name VARCHAR(255),
    Price DECIMAL(10, 2),
    Stock INT
);

-- Produktkategorier
CREATE TABLE Category (
    Category_id INT PRIMARY KEY,
    name VARCHAR(255)
);
```

```sql
--  detta är en koppling tabell för, Kopplingstabell för Produkt och Kategori (Många-till-många relation)

CREATE TABLE Product_Category (

    ProduktID INT,

    Category_id INT,

    PRIMARY KEY (ProduktID, Category_id),

    FOREIGN KEY (ProduktID) REFERENCES Product(ProduktID),

    FOREIGN KEY (Category_id) REFERENCES Category(Category_id)

);



-- Lagerregister

CREATE TABLE Warehouse (

    Warehouse_id INT PRIMARY KEY,

    Product_id INT,

    Shelf_location VARCHAR(255),

    Stock_quantity INT,

    FOREIGN KEY (Product_id) REFERENCES Product(ProduktID)

);



-- Orderhantering
```

```sql
CREATE TABLE `Order` (

    Order_id INT PRIMARY KEY,

    order_date DATETIME,

    total_price DECIMAL(10, 2),

    Customer_id INT,

    status VARCHAR(20), -- Pending, Processing, Completed, Cancelled

    FOREIGN KEY (Customer_id) REFERENCES Customer(Customer_id)

);



-- Produkter i en order

CREATE TABLE Order_item (

    Order_item_ID INT PRIMARY KEY,

    Order_id INT,

    Product_id INT,

    quantity INT,

    price DECIMAL(10, 2),

    FOREIGN KEY (Order_id) REFERENCES `Order`(Order_id),

    FOREIGN KEY (Product_id) REFERENCES Product(ProduktID)

);
```

```sql
-- Leveranshantering och plocklista
CREATE TABLE Delivery (
    Delivery_id INT PRIMARY KEY,
    Order_id INT,
    Delivery_date DATETIME,
    status VARCHAR(20), -- den berättar Packed, Shipped, Delivered
    FOREIGN KEY (Order_id) REFERENCES `Order`(Order_id)
);

-- Lagerhållningslog
CREATE TABLE Inventory_Log (
    Log_id INT PRIMARY KEY,
    Event_description TEXT,
    Event_date DATETIME
);

-- Fakturering
CREATE TABLE Invoice (
```

```
    Invoice_id INT PRIMARY KEY,

    Order_id INT,

    Invoice_date DATETIME,

    total_price DECIMAL(10, 2),

    FOREIGN KEY (Order_id) REFERENCES `Order`(Order_id)

);
```

REFERENSER

[1] Kokbok för databasmodellering, https://dbwebb.se/kunskap/kokbok-for-databasmodellering, visited 2024-01-31.