Pet Shop Management

Apostol Alin-Constantin

Pet Shop Management

This project aims to develop an information system for managing the activity of a pet shop. The main goal is to streamline the sales process and maintain a clear record of stock, customers, and employee performance.

Main functionalities covered:

- **Product Management:** Organizing products by categories (Food, Accessories, Pharmacy) and monitoring stock levels.
- **Order Processing:** Recording sales, automatically calculating totals, and linking each transaction to the employee who processed it.
- **Customer Loyalty:** A loyalty card system with tiers (Bronze, Silver, Gold) to encourage returning customers.
- **Employee Monitoring:**Keeping records of staff members and their roles (Manager, Cashier, Pharmacist).

The critical information managed includes customers' personal data (with strict validations), transaction details (what was sold and in what quantity), and price history.

Structure and relationships between entities

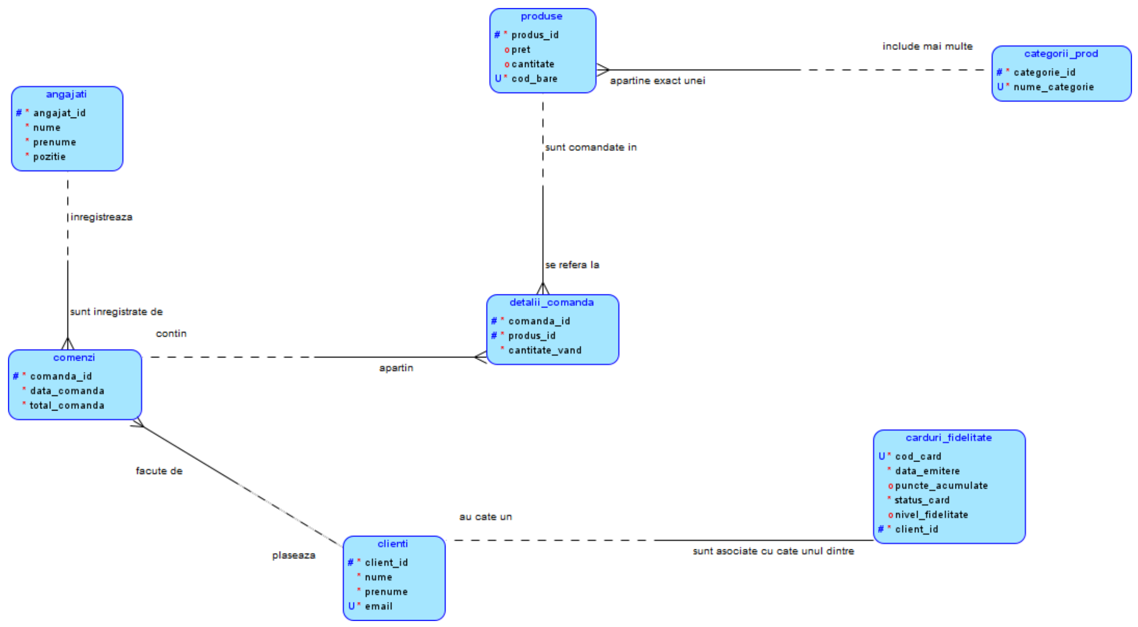
The database model was designed according to normalization principles. The following main entities were identified:

1. **CLIENTI (CUSTOMERS):** Stores contact information.
2. **CARDURI_FIDELITATE (LOYALTY_CARDS):** Extension of the Customers entity (1:1 relationship).
3. **ANGAJATI (EMPLOYEES):** Store staff.
4. **COMENZI (ORDERS):** Transactions performed (receipt header).
5. **DETALII_COMANDA (ORDER_DETAILS):** Transaction contents (receipt lines).
6. **PRODUSE (PRODUCTS):** Store inventory.
7. **CATEGORII_PROD (PRODUCT_CATEGORIES):** Product classification.

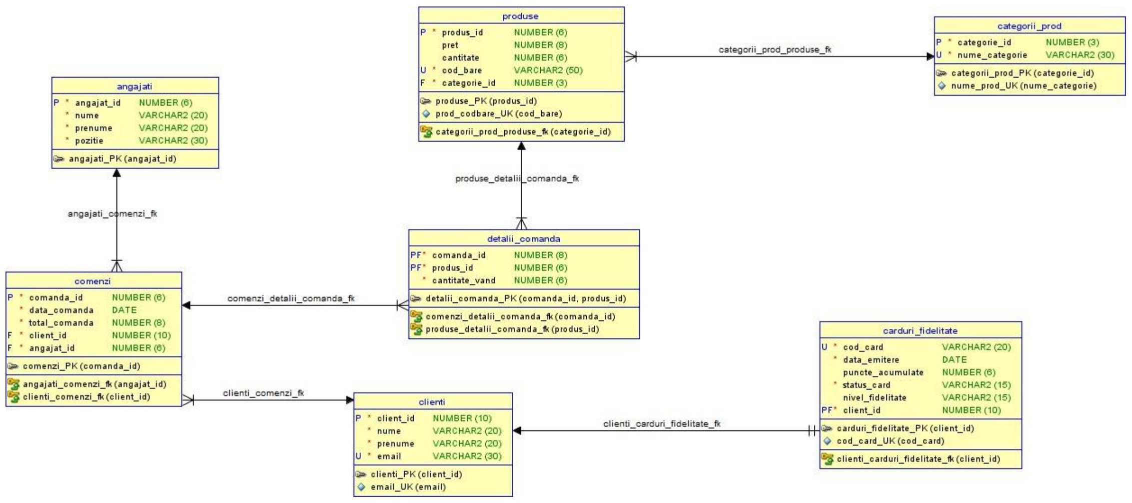
Types of relationships implemented:

- **1:1 (One-to-One):** Between **CUSTOMERS** and **LOYALTY_CARDS**
    - A customer can have at most one loyalty card, and a card belongs to exactly one customer.
    - Implementation: The primary key in **LOYALTY_CARDS** (client_id) is also a foreign key referencing **CUSTOMERS**.
- **1:N (One-to-Many):**
    - **Categories → Products:** One category contains multiple products.
    - **Employees → Orders:** One employee processes multiple orders.
    - **Customers → Orders:** One customer can place multiple orders.
- **M:N (Many-to-Many):** Between **ORDERS** and **PRODUCTS**
    - An order contains multiple products, and a product can appear in multiple orders.
    - **Solution:** This relationship was decomposed using the associative entity **ORDER_DETAILS**, which contains foreign keys to both parent tables and the relationship-specific attribute (cantitate_vand – sold quantity).

Logical Model



Relational Model


produse
P * produs_id NUMBER (6)
pret NUMBER (8)
cantitate NUMBER (6)
U * cod_bare VARCHAR2 (50)
F * categorie_id NUMBER (3)
produse_PK (produs_id)
prod_codbare_UK (cod_bare)
categorii_prod_produse_fk (categorie_id)
categorii_prod_produse_fk
categorii_prod
P * categorie_id NUMBER (3)
U * nume_categorie VARCHAR2 (30)
categorii_prod_PK (categorie_id)
nume_prod_UK (nume_categorie)
angajati
P * angajat_id NUMBER (6)
* nume VARCHAR2 (20)
* prenume VARCHAR2 (20)
* pozitie VARCHAR2 (30)
angajati_PK (angajat_id)
angajati_comenzi_fk
produse_detalii_comanda_fk
detalii_comanda
PF* comanda_id NUMBER (8)
PF* produs_id NUMBER (6)
* cantitate_vand NUMBER (6)
detalii_comanda_PK (comanda_id, produs_id)
comenzi_detalii_comanda_fk (comanda_id)
produse_detalii_comanda_fk (produs_id)
comenzi_detalii_comanda_fk
comenzi
P * comanda_id NUMBER (6)
* data_comanda DATE
* total_comanda NUMBER (8)
F * client_id NUMBER (10)
F * angajat_id NUMBER (6)
comenzi_PK (comanda_id)
angajati_comenzi_fk (angajat_id)
clienti_comenzi_fk (client_id)
clienti_comenzi_fk
clienti
P * client_id NUMBER (10)
* nume VARCHAR2 (20)
* prenume VARCHAR2 (20)
U * email VARCHAR2 (30)
clienti_PK (client_id)
email_UK (email)
clienti_carduri_fidelitate_fk
carduri_fidelitate
U * cod_card VARCHAR2 (20)
* data_emitere DATE
puncte_acumulate NUMBER (6)
* status_card VARCHAR2 (15)
nivel_fidelitate VARCHAR2 (15)
PF* client_id NUMBER (10)
carduri_fidelitate_PK (client_id)
cod_card_UK (cod_card)
clienti_carduri_fidelitate_fk (client_id)

Normalization aspects

The database follows normal forms to eliminate redundancy and anomalies:

- **First Normal Form (1NF):** All attributes are atomic (e.g., first name and last name are stored separately). There are no repeating groups (products in an order are not stored as a list in a single field, but in the separate **ORDER_DETAILS** table).
- **Second Normal Form (2NF):** All non-key attributes depend on the entire primary key. In **ORDER_DETAILS** (which uses a composite key), the sold quantity depends on the pair (Order + Product).
- **Third Normal Form (3NF):** There are no transitive dependencies. For example, the category name is not stored in the **PRODUCTS** table (where it would be unnecessarily repeated), but in **PRODUCT_CATEGORIES**, referenced only by an ID.

Column description (selection)

Data types:

- **NUMBER(6):** Used for IDs (allows up to 999,999 records) and stock values.
- **VARCHAR2(n):** Used for text (Name, Email). The size was chosen based on estimated length (e.g., 30 characters for Email).
- **DATE:** For order date and card issue date.
- **NUMBER(8):** For prices and totals, allowing large monetary values.

Constraints description (data validation)

To ensure data integrity, the following mechanisms were implemented:

A. Primary Keys
Automatically generated through Sequences and Triggers (BEFORE INSERT). This auto-increment mechanism ensures uniqueness for each record (e.g., angajat_id, comanda_id).
B. Foreign Keys
Ensure referential integrity:

- A product cannot be added to a non-existent category.
- An order cannot be created for a non-existent customer.

C. CHECK constraints (logical validation)

- **Email validation:** REGEX (regexp_like) was used to enforce the format text@domain.extension.
- **Name validation:** First name and last name must have at least 2 letters (length > 1).
- **Positive values:** Price, stock values, and order totals must be > 0.
- **Allowed values (value lists):**
    - status_card accepts only: 'Activ', 'Expirat', 'Suspendat'.
    - employee position (pozitie angajat) accepts only: 'Casier', 'Manager', 'Farmacist veterinar'.
- **Quantity limit:** A customer cannot buy more than 10 units of the same product on one receipt (BETWEEN 1 AND 10).

D. UNIQUE constraints

- **Email:** Two customers cannot have the same email address.
- **Card code / Barcode:** Physical identifiers must be unique in the system.

E. Complex triggers

- **Date validation:** Triggers (trg_comenzi_data, trg_card_data_emitere) were created to prevent inserting a date in the past (earlier than the current date). They raise a custom error (RAISE_APPLICATION_ERROR).