



UNIVERSITATEA TEHNICĂ "GH ASACHI" IAȘI FACULTATEA AUTOMATICĂ ȘI
CALCULATOARE
SPECIALIZAREA CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI
DISCIPLINA BAZE DE DATE PROIECT

Gestiunea unui magazin online de dulciuri

**Coordonator,
Cătălin Mironeanu**

**Student,
Hriscu Cornelia-Ștefana
GRUPA 1310A**

Iași, 2022



TITLU PROIECT: Gestiunea unui magazin online de dulciuri

Proiectul se va concentra pe analizarea problemei administrării unui magazin online de dulciuri, proiectarea și implementarea unei baze de date care să permită modelarea fluxului de solicitări printr-un astfel de magazin.

DESCRIEREA CERINTELOR ȘI MODUL DE ORGANIZARE AL PROIECTULUI

Datorită volumului mare de solicitări pentru diversele articole vândute în mediul online, problema administrării unui magazin care funcționează în acest mediu este de mare importanță, întrucât toate solicitările trebuie concentrate, analizate și onorate în cel mai eficient mod pentru a minimiza timpul de așteptare al fiecărui client.

O aplicație care se ocupă de gestionarea magazinului online trebuie mai întâi să înregistreze toți clienții care doresc să comande produsele pe care le vinde, astfel încât comenzile și tranzacțiile să poată fi efectuate în siguranță.

Un alt aspect important îl reprezintă evidențele de inventar, care trebuie monitorizate în timp real pentru a preveni plasarea comenzilor pentru produse care nu sunt în stoc. În plus, aplicația trebuie să permită stocarea istoricului comenzilor precum și a istoricului aprovizionării pentru a putea calcula statistici privind investițiile realizate și profitul magazinului.

DATELE DE INTERES

Acestea sunt gestionate de baza de date și sunt legate de:

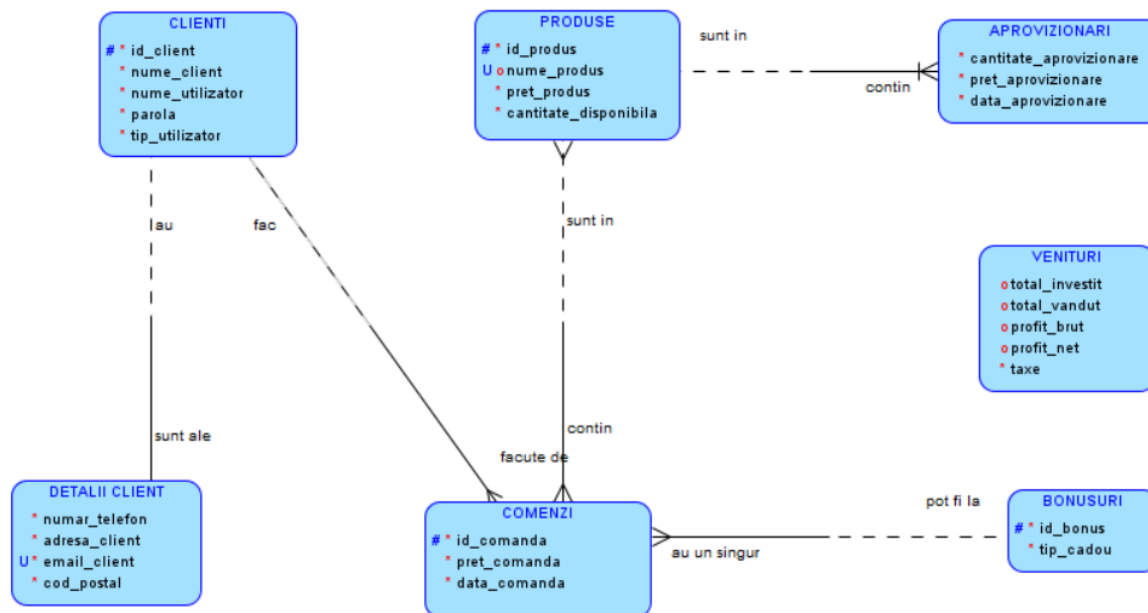
- **CLIEȚI:** Trebuie să păstrăm o listă a clienților înregistrați la magazinul online pe baza e-mailului, adresei (stocate în tabela [DETALII_CLIENT](#)), numelui de utilizator și parolei (stocate în tabela [CLIEȚI](#)).
- **COMENZI:** Pentru a putea monitoriza inventarul, precum și veniturile, trebuie să păstrăm un istoric al comenzilor din magazin. Monitorizarea acestora se va face în două tabele. Un tabel va reține clientul care a plasat comanda, id-ul unic al comenzii, prețul total și data la care a fost plasată comanda, iar celălalt tabel va reține lista de produse și cantitatea corespunzătoare asociată cu fiecare comandă.



- **APROVIZIONARE:** Monitorizarea informațiilor legate de aprovizionarea stocului de produse al magazinului se va face într-un tabel care va păstra fiecare comandă de aprovizionare sub forma unei intrări, câmpurile de interes sunt ID-ul produsului, cantitatea și prețul de achiziție.
- **PRODUSE:** Vom folosi un tabel în care vom păstra toate datele legate de fiecare produs cum ar fi numele, prețul, ID.
- **STOC:** Pentru a putea gestiona stocul rapid și eficient, se va folosi de coloana *cantitate_disponibila* din tabela **PRODUSE**, astfel se vor observa cantitățile disponibile din fiecare produs. Prin urmare, vom putea verifica înainte de fiecare comandă dacă stocul permite ca respectiva comandă să fie îndeplinită.
- **VÂNZĂRI:** Pentru ca aplicația să monitorizeze profitabilitatea magazinului, vom folosi un tabel pentru a reține investiția totală, vânzările totale și profitul magazinului calculat din momentul în care magazinul și-a început activitatea, până în prezent.

DESCRIEREA DETALIATĂ A MODELULUI LOGIC ȘI A ENTITĂȚILOR

DIAGRAMA MODELULUI LOGIC:





ENTITĂȚI ȘI CONSTRÂNGERI FOLOSITE:

- **CLIENTI** - entitate folosită în gestionarea clienților magazinului
 - Id_client - **NUMBER(2)** NOT NULL
 - clienti_pk (Primary Key)
 - Nume_client - **VARCHAR2(50)** NOT NULL
 - nume_client_ck : verificarea ca numele să aibă minim cinci caractere și să nu conțină cifre (cu regex)
 - Nume_utilizator - **VARCHAR2(20)** NOT NULL
 - nume_util_ck : impune o dimensiune minimă de 5 caractere și să poată conține litere și cifre, fără caractere speciale (cu regex)
 - Parola - **VARCHAR2(31)** NOT NULL
 - parola_ck : verificare ca parola să aibă mai mult de 8 caractere și să poată conține litere, cifre și caractere speciale, fără spații (cu regex)
 - Tip_utilizator - **VARCHAR2(20)** NOT NULL
 - tip_utilizator_ck : aparține următoarei liste {'Administrator', 'Utilizator'}
- **DETALII CLIENT** - entitate folosită în evidențierea informațiilor clienților magazinului
 - Număr_telefon - **VARCHAR2(10)** NOT NULL
 - nr_telefon_ck : verificarea ca numărul de telefon să conțină doar 10 cifre (<nume>@<domeniu>.<suffix>) (cu regex)
 - La acest atribut, inițial, m-am gândit să fie cheie unică, uitând de posibilitatea că un om își poate schimba numărul de telefon. Astfel, când un client ar fi dat o comandă cu un număr de telefon, pentru a efectua comenzi în viitor, acesta ar fi fost obligat să își păstreze numărul de telefon inițial => CAZ ERONAT
 - Adresa_client - **VARCHAR2(80)** NOT NULL
 - Adresa_client_ck : verificarea ca adresa clientului să conțină mai



mult de 5 caractere și să poată avea litere, cifre și spațiere, dar fără caractere speciale (cu regex)

- La acest atribut, inițial, m-am gândit să fie cheie unică, uitând de posibilitatea că un om își poate schimba adresa domiciliului. Astfel, când un client ar fi dat o comandă cu o anumită adresă, pentru a efectua comenzi în viitor, acesta ar fi fost obligat să își păstreze adresa inițială => CAZ ERONAT

- Email_client - **VARCHAR2(40)** NOT NULL
 - email_client_ck :verificarea ca email-ul să corespundă formatului corect (<nume>@<domeniu>.<sufix>)
- Cod_postal - **VARCHAR2(6)** NOT NULL
 - cod_postal_ck: încadrarea codului poștal în normele standardizate (cu regex)
- Id_client - **NUMBER(3)** NOT NULL
 - clienti_fk (Foreign Key) : asigură legătura cu tabela **CLIENTI**

➤ **COMENZI** - tabelă folosită pentru gestionarea comenzilor date de către clienți

- Id_comanda - **NUMBER(2)** NOT NULL
 - comenzi_pk (Primary Key)
- Id_client - **NUMBER(2)** NOT NULL
 - clienti_fk (Foreign Key) : asigură legătura cu tabela **CLIENTI**
- Id_bonus - **NUMBER(2)** NOT NULL
 - bonusuri_fk(Foreign Key):asigură legătura cu tabela **BONUSURI**
- Pret_comanda - **NUMBER(5)** NOT NULL
 - Default: 0
 - Comenzi_pret_comanda_ck: impune ca prețul comenzii să existe
- Data_comanda - **DATE** NOT NULL
 - Default : SYSDATE



- **APROVIZIONĂRI** - tabela folosită pentru gestionarea aprovizionarilor magazinului
 - Id_produș - **NUMBER(2)** NOT NULL
 - produse_fk (Foreign Key) : asigură legătura cu tabela PRODUSE
 - Cantitate_aprovizionare - **NUMBER(3)** NOT NULL
 - aprovizionari_cantitate_ck : impune o cantitate pozitivă
 - Pret_aprovizionare - **NUMBER(4)** NOT NULL
 - aprovizionari_pret_apro_ck : impune ca prețul să fie între 10 și 9999
 - Data_aprovizionare - **DATE** NOT NULL
 - Default : SYSDATE

- **PRODUSE** - tabela ce conține informații despre produsele din magazin
 - Id_produș - **NUMBER(2)** NOT NULL
 - produse_pk (Primary Key)
 - Nume_produș - **VARCHAR2(20)**
 - produse_nume_produș_uk : asigură unicitatea numelor produselor
 - produse_nume_produș_ck : asigură ca numele produsului să existe
 - Pret_produș - **NUMBER(4)** NOT NULL
 - produse_pret_produș_ck : impune ca prețul să fie între 10 și 9999
 - Cantitate_disponibilă - **NUMBER(3)** NOT NULL
 - produse_cantitate_disp_ck : impune ca cantitatea să fie mai mare sau egală cu 0
 - Pret_transport – am ales să renunț la acest atribut deoarece nu avea un rol anume. Inițial, dacă clientul meu depășea o anumită sumă voiam să îi ofer ca bonus transportul gratuit, atributul fiind pe 0. Însă, pe parcurs, m-am răzgândit și am ales să ofer ca bonus ceva dulce. Pentru a evita să mă complic, l-am eliminat.



- **VENITURI** - tabela folosită pentru monitorizarea cheltuielilor și profitului total magazinului
- Total_investit - **NUMBER(5)** DEFAULT 0
 - venituri_total_investit_ck : impune ca totalul investit să existe
 - Total_vandut - **NUMBER(5)** DEFAULT 0
 - venituri_total_vandut_ck : impune ca totalul vândut să existe
 - Profit_brut - **NUMBER(5)**
 - venituri_profit_brut_ck : impune ca profitul brut să existe
 - profit_net - **NUMBER(5)**
 - venituri_profit_net_ck : impune ca profitul net să existe
 - Taxe - **NUMBER(2)** NOT NULL
 - venituri_taxe_ck : impune ca taxa să existe

ÎN PROIECTAREA BAZEI DE DATE S-AU STABILIT URMĂTOARELE TIPURI DE RELAȚII:

→ 1:1

- ◆ Între **CLIENTI** și **DETALII_CLIENT**, deoarece fiecare intrare în tabela **CLIENTI** va avea un corespondent unic în tabela **DETALII_CLIENT**, dar și fiecare informație va avea un corespondent unic în tabela **CLIENTI**.

→ 1:n

- ◆ Între **CLIENTI** și **COMENZI**, deoarece tabela **COMENZI** conține istoricul tuturor comenzilor, existând posibilitatea ca un client să dea mai multe comenzi, dar o comandă nu poate fi dată de mai mulți clienți.
- ◆ Între **PRODUSE** și **APROVIZIONĂRI**, deoarece pot fi mai multe aprovizionări făcute pe același produs, dar o intrare în tabela **APROVIZIONĂRI** conține un singur produs.
- ◆ Între **COMENZI** și **BONUSURI**, deoarece pot fi mai multe bonusuri făcute pe aceeași comandă, dar o intrare în tabela **BONUSURI** conține o singură comandă.



→ N:m

- ◆ Între **COMENZI** și **PRODUSE**, deoarece o comanda poate conține mai multe produse, iar un produs poate apărea în mai multe intrări din tabela **COMENZI**.
- ◆ Această legătură generează o tabelă de intersecție, numită **DETALII_COMANDA**.

NORMALIZAREA BAZEI DE DATE

Normalizarea a fost folosită la nivelul entităților **DETALII_CLIENT**, **APROVIZIONĂRI** și **COMENZI** prin folosirea foreign key-urilor pentru a le descompune în entități mai mici care stochează aceleași date ca și entitatea inițială astfel încât să fie eliminate redundanța în date.

Normalizarea entității s-a realizat astfel:

➤ **DETALII_CLIENT**

- id_client - id-ul clientului ar fi fost specificat de fiecare dată când s-ar fi inserat date în tabelă, această repetiție fiind redundantă. S-a folosit tabela **CLIENTI** care va stoca fiecare client, pentru a se evita popularea excesivă a entității **DETALII_CLIENT**

➤ **APROVIZIONĂRI**

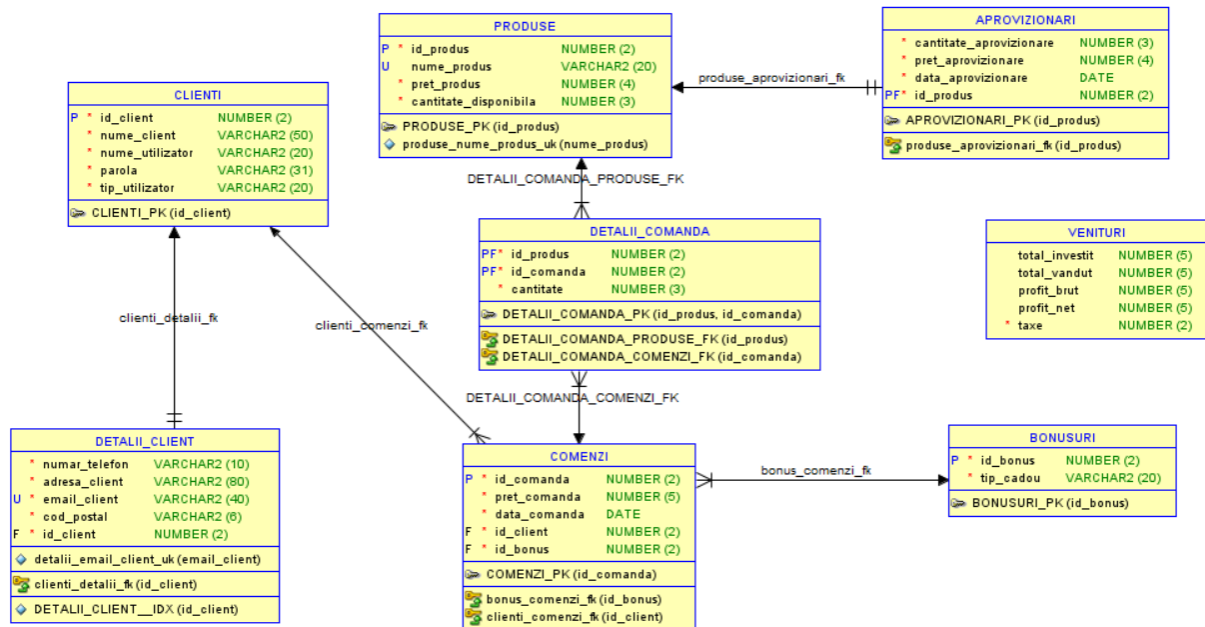
- id_produs - id-ul produsului ar fi fost specificat de fiecare dată când s-ar fi inserat date în tabelă, această repetiție fiind redundantă. S-a folosit tabela **PRODUSE** care va stoca fiecare produs, pentru a se evita popularea excesivă a entității **APROVIZIONĂRI**

➤ **COMENZI**

- id_produs, id_bonus – id-ul produsului și cel al bonusului ar fi fost specificate de fiecare dată când s-ar fi inserat date în tabelă, această repetiție fiind redundantă. Pentru a nu se popula în exces entitatea, se vor folosi tabelele **PRODUSE** și **BONUSURI** care vor stoca fiecare produs, respectiv bonus



DESCRIEREA DETALIATĂ A MODELULUI RELAȚIONAL



Pentru relația **n:m** dintre **COMENZI** și **PRODUSE** a fost generată o tabelă de intersecție pentru a avea posibilitatea să adăugăm mai multe produse în aceeași comandă (legate de același id_comanda).

- **DETALII_COMANDĂ**: tabela folosită pentru a monitoriza produsele dintr-o comandă
 - Detalii_comanda_pk (Primary Key) : cheie primară compusă (id_comanda, id_produș)
 - Id_comanda - **NUMBER(2)** NOT NULL
 - detalii_comanda_comenzi_fk (Foreign Key) : asigură legătura cu tabela **COMENZI**
 - Id_produș - **NUMBER(2)** NOT NULL
 - detalii_comanda_produș_fk (Foreign Key) : asigură legătura cu tabela **PRODUSE**
 - Cantitate - **NUMBER(3)** NOT NULL



Pentru cheile primare **CLIENTI.id_client**, **COMENZI.id_comanda**, **PRODUS.id_produs** s-a ales folosirea funcției de **autoincrement** pusă la dispoziție de data modeller, care are la bază folosirea unei **secvențe**, și a unui **before trigger**. La restul primary key-urilor nu s-a folosit autoincrementul deoarece este necesară introducerea la mână a acestora, de exemplu bonusul.

În implementarea bazei de date s-au adăugat obiecte de tip **trigger**, cu două scopuri:

- Pentru a detecta anumite scenarii, cu scopul de a împiedica unele acțiuni
- Pentru a declanșa o acțiune (INSERT, UPDATE)

Obiecte de tip **trigger** folosite:

- **cantitate_trg** : verifică înainte de fiecare insert în tabela **DETALII_COMANDA**, dacă stocul actual din tabela **PRODUSE** permite efectuarea comenzii. În caz contrar, va semnală o eroare.
- **pret_aprovizionare_trg**: verifică înainte de insert în tabela **APROVIZIONĂRI**, dacă prețul de cumpărare este mai mic decât cel de vânzare. În caz contrar, va semnală o eroare.
- **detalii_comanda_trg** : după fiecare insert în tabela **DETALII_COMANDA**, declanșează o acțiune în care se actualizează prețul total al comenzii (**COMENZI.pret**), precum și câmpul **VENITURI.total_vandut**. Se actualizează de asemenea și stocul magazinului (**PRODUSE.cantitate_disponibila**)
- **aprovizionari_trg** : după fiecare insert în tabela **APROVIZIONĂRI**, declanșează o acțiune în care se actualizează stocul magazinului (**PRODUSE.cantitate_disponibila**), precum și câmpul **VENITURI.total_vandut**.



- **arovizionari_data_trg** : verifică după înainte de insert în tabela **APROVIZIONĂRI**, dacă data de aprovizionare este data curentă. În caz contrar, va semnală o eroare.

FUNCȚII folosite:

- **găseste_id_by_username**: primește ca parametru **numele** unui client și returnează **id-ul** acestuia dacă este găsit. În caz contrar, va semnală o eroare.