Universitatea Tehnică „Gheorghe Asachi” Iași
Facultatea de Automatică și Calculatoare

Baze de Date

Proiect

Student: Vasile LEȘAN, grupa 1307B
Coordonator: Ș.l.dr.ing. Cătălin MIRONEANU

An Universitar 2024 - 2025

Tema proiectului: Gestionarea unui magazin cu instrumente muzicale

Descrierea non-tehnică:

În cadrul acestui proiect se definește o bază de date relațională pentru gestionarea eficientă a stocurilor, vânzărilor și relațiilor cu clienții într-un magazin de instrumente muzicale. Instrumentele muzicale sunt împărțite pe categorii (instrumente de suflat/ cu coarde/ cu clape / tradiționale, percuție, boxe, microfoane, lumini, etc.), iar un instrument poate aparține unei singure categorii. Stocul fiecărui produs este actualizat manual printr-o instrucțiune *update* după fiecare achiziție făcută sau la adăugarea unui nou produs.

Baza de date se ocupă doar de vânzarea articolelor și actualizarea stocurilor și nu tratează problema achizițiilor și a populării magazinului cu produse de la furnizori. De asemenea, nu este tratată modalitatea de achiziție a produselor (online sau fizic) și nici modalitatea de plată a comenzii (cash, card, în rate). O vânzare poate fi inițiată doar dacă stocul tuturor produselor din respectiva comandă este nenul. În cazul în care un produs nu mai este disponibil, stocul acestuia este 0 și nu se mai pot realiza vânzări ale respectivului produs. Fiecare vânzare este inițiată la o anumită dată.

Clienții pot iniția vânzări care includ produse de diverse tipuri, dar la o singură vânzare poate fi achiziționat un singur produs de un anumit tip. Fiecare client este identificat printr-un id unic, nume, prenume și date personale: telefon, email, adresă și data la care acesta a fost înregistrat în sistem.

În urma analizei cerințelor au rezultat 5 entități: CATEGORII, PRODUSE, CLIENTI, DETALII_CLIENTI, VANZARI și o entitatea intermediară, PRODUSE_VANZARI, entitate ce rezolvă relația dintre PRODUSE și VANZARI. Sunt definite relații între tabele prin chei primare și chei străine pentru a permite vizualizarea datelor corelate între entități. Majoritatea atributelor ce aparțin fiecărei entități dispun de ce cel puțin o constrângere (Primary Key, Foreign Key, Not Null, Check) pentru a asigura și menține integritatea datelor, precum și pentru validarea lungimii sau a formatului atributelor (ex: email, telefon, preț pozitiv, stoc >= 0). De asemenea, sunt definite triggere pentru validarea corectitudinii datelor calendaristice introduse.

Structura și inter-relaționarea tabelelor

Entități:

1. Entitatea **CATEGORIE (id_categorie, denumire):**
    - id_categorie: NUMBER(3)
    - denumire: VARCHAR2(20)

2. Entitatea **PRODUSE (id_produs, nume, pret, stoc):**
    - id_produs: NUMBER(5)
    - nume: VARCHAR(30)
    - pret: NUMBER(5)
    - stoc: NUMBER(5)

3. Entitatea **CLIENTI (id_client, nume, prenume):**
    - id_client: NUMBER(5)
    - nume: NUMBER(25)
    - prenume : NUMBER(25)

4. Entitatea **DETALII_CLIENTI (telefon, email, adresa, data_inregistrarii):**
    - telefon: VARCHAR2(10)
    - email: VARCHAR2(25)
    - adresa: VARCHAR2(40)
    - data_inregistrarii: DATE

5. Entitatea **VANZARI (id_vanzare, data_vanzare):**
    - id_vanzare: NUMBER(5)
    - data_vanzare: DATE

6. Entitatea **PRODUSE_VANZARI (id_produs, id_vanzare):**
    - id_produs: NUMBER(5)
    - id_vanzare: NUMBER(5)

Datele sunt normalizate până la a treia formă normală (3NF), deoarece sunt respectate atât prima formă normală, precum și cea de a doua formă normală și nu există dependențe tranzitive între atributele non-cheie.

Prima formă normală (1NF) este respectată deoarece: fiecare tabel are un set de atribute nedivizibile sau atomice, nu există grupuri repetitive de atribute și fiecare coloană conține valori de același tip. De exemplu, în tabela PRODUSE, coloanele *nume*, *pret*, *stoc* conțin informații atomice și nu există atribute care să repete aceste informații într-un rând.

A doua formă normală (2NF) este respectată deoarece toate atributele non-cheie depind complet de cheia primară și astfel nu există dependențe parțiale. Folosind ca exemplu aceeași tabelă, PRODUSE, toate coloanele (*nume, pret, stoc*) depind complet de cheia primară *id_produs*.

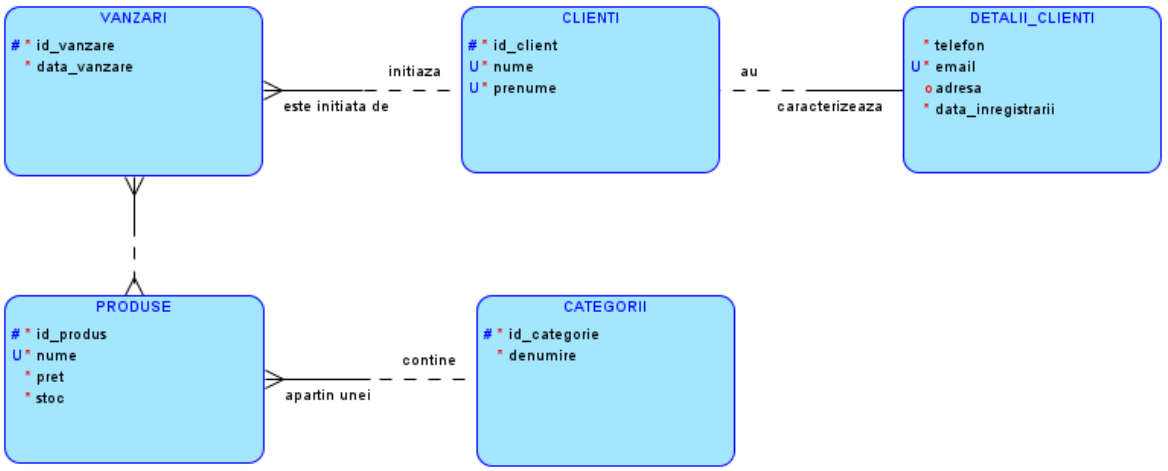
Pentru cea de a treia formă normală (3NF), observăm că sunt respectate primele două forme normale, iar în entitățile create nu există dependențe tranzitive sau atribute non-cheie care să depindă de alte atribute non-cheie. De exemplu, în tabela DETALII_CLIENTI, coloana *email* depinde direct de cheia primară *id_client* și nu de alte coloane cum ar fi *telefon* sau *adresa*.

Normalizarea datelor până la forma normală 3 asigură eliminarea redundanțelor, creșterea consistenței informațiilor și ușurința întreținerii. Astfel, datele sunt separate astfel încât fiecare informație să fie stocată o singură dată, modificările structurale sunt mai ușor de realizat deoarece datele sunt organizate logic pe entități independente, iar în cazul modificării unei informații, modificarea se realizează într-un singur loc, fără riscul de inconsecvență.

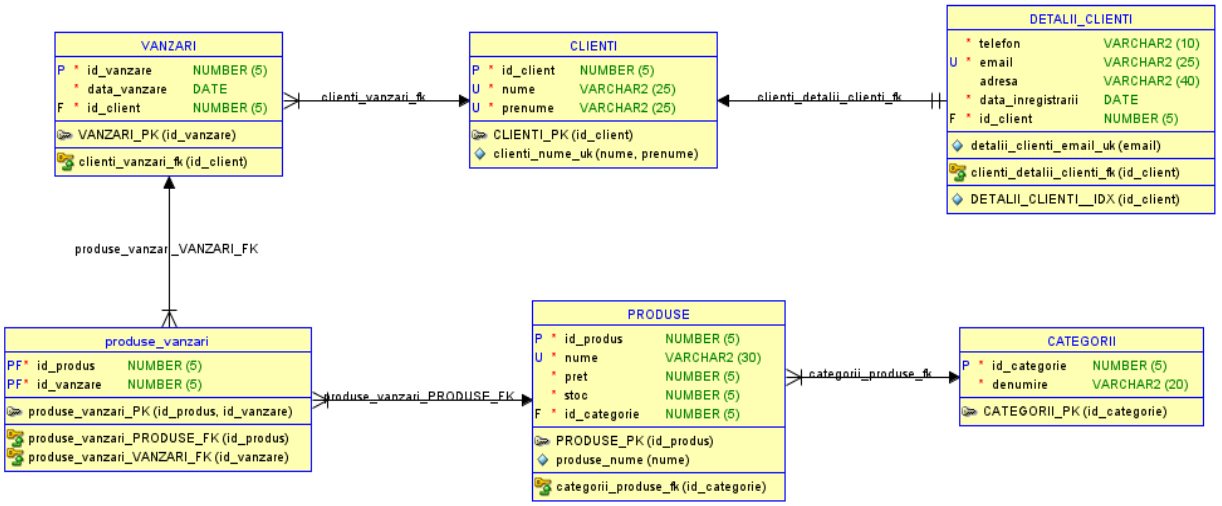
Relațiile dintre entități:

- **CATEGORII** și **PRODUSE -** relație One to Many, o categorie poate conține mai multe produse, iar un produs poate aparține unei singure categorii
- **CLIENTI** și **DETALII_CLIENTI -** relație One to One, un client este caracterizat de un singur set de date personale, iar un set de date poate caracteriza un singur client
- **CLIENȚI** și **VANZARI -** relație One to Many, un client poate inița mai multe vânzări, dar o vânzare aparține și este inițiată de un singur client
- **PRODUSE** și **VANZARI -** relație Many to Many, un produs poate să apară în mai multe vânzări, iar o vânzare poate conține mai multe produse

Modelul logic:



Modelul relațional:



Descrierea coloanelor din tabele

1. Tabelul **CATEGORII**
   - *id_categorie* - Valoare numerică de maxim 5 cifre utilizată pentru identificarea unică a fiecărei categorii. Dimensiunea maximă de 5 cifre permite gestionarea până la 99.999 de categorii, gestionarea automată a valorilor reducând erorile umane.
   - *denumire* - Valoare de tip șir de caractere de dimensiune maximă egală cu 20 de caractere ce conține numele categoriei. Dimensiunea maximă de 20 de caractere permite înregistrarea de denumiri descriptive, dar concise.

2. Tabelul **CLIENTI**
    - *id_client* - Valoare numerică de maxim 5 cifre utilizată pentru identificarea unică a fiecărui client. Aceasta oferă un spațiu suficient pentru stocarea până la 99.999 de clienți
    - *nume* - Valoare de tip șir de caractere de lungime maximă de 25 de caractere, reprezintă numele de familie al clientului.
    - *prenume* - Valoare de tip șir de caractere de lungime maximă de 25 de caractere, reprezintă prenumele clientului.

3. Tabelul **DETALII_CLIENTI**
    - *telefon* - Valoare de tip șir de caractere de lungime maximă de 10 caractere ce stochează numărul de telefon al clientului. Stocarea în format șir de caractere oferă flexibilitate în cazul numerelor de telefon ce conțin prefixuri.
    - *email* - Valoare de tip șir de caractere de lungime maximă 25 de caractere, reprezintă adresa de email a clientului. Dimensiunea a fost aleasă astfel încât să fie suficientă pentru majoritatea formatelor uzuale de adrese, dar să nu fie prea mare, pentru a evita risipa de spațiu.
    - *adresa* - Valoare de tip șir de caractere de lungime maximă de 40 de caractere, reprezintă adresa clientului incluzând numele străzii, număr și alte detalii relevante. Lungimea de 40 de caractere este adecvată pentru majoritatea adreselor.
    - *data_inregistrarii* - Valoare de tip dată ce stochează data înregistrării clientului în sistem.
    - *id_client* - Valoare numerică de maxim 5 caractere, este cheia străină care face legătura cu tabelul **CLIENTI**

4. Tabelul **PRODUSE**
    - *id_produs* - Valoare numerică de dimensiune maximă de 5 caractere ce identifică în mod unic fiecare produs. Dimensiunea permite stocarea până la 99.999 de produse.
    - *nume* - Valoare de tip șir de caractere de dimensiune maximă de 30 de caractere, permite înregistrarea denumirii produselor ce pot conține descrieri.
    - *preț* - Valoare numerică de maxim 5 caractere ce stochează prețul produsului. Această dimensiune permite înregistrarea de prețuri cu valoarea de până la 99.999 unități monetare.
    - *stoc* - Valoare numerică de dimensiune maximă de 5 caractere ce reprezintă numărul de produse disponibile în magazin.
    - *id_categorie* - Valoare numerică ce reprezintă cheia străină care face legătura cu tabelul **CATEGORII**

5. Tabelul **VANZARI**
    - *id_vanzare* - Valoare numerică de dimensiune maximă de 5 caractere ce identifică în mod unic fiecare vânzare inițiată.
    - *data_vanzare* - Valoare de tip date ce înregistrează data inițierii unei vânzări

- *id_client* - Valoare numerică de dimensiune maximă de 5 caractere, reprezintă cheia străină care face legătura cu tabelul **CLIENTI**

6. Tabelul **PRODUSE_VANZARI**
   - *id_produs* - Valoare numerică de maxim 5 caractere, reprezintă cheia străină care face legătura cu tabelul **PRODUSE**
   - *Id_vanzare* - Valoare numerică de maxim 5 caractere, reprezintă cheia străină care face legătura cu tabelul **VANZARI**

Constrângerile folosite în tabele

1. Entitatea **CATEGORII:**
   - categorii_denumire_ck (CHECK) - verifică dacă lungimea denumirii este mai mare de un caracter pentru a evita introducerea de denumiri invalide sau incomplete
   - categorii_pk (PRIMARY KEY) - asigură unicitatea fiecărei categorii prin atributul id_categorie

2. Entitatea **CLIENTI:**
   - clienti_nume_ck (CHECK) - asigură că lungimea numelui este mai mare de un caracter și că acesta conține doar litere și cratime, prevenind astfel introducerea de nume invalide sau care conțin alte simboluri speciale
   - clienti_prenume_ck (CHECK) - aceeași constrângere aplicată atributului *nume*
   - clienti_pk (PRIMARY KEY) - asigură unicitatea fiecărui client prin atributul id_client
   - clienti_nume_uk (UNIQUE) - creează o cheie compusă din atributele *nume* și *prenume*, asigurând astfel unicitatea numelui clientului și prevenind introducerea de nume duplicate ce pot crea confuzii

3. Entitatea **DETALII_CLIENTI:**
   - detalii_clienti_telefon_ck (CHECK) - asigură ca lungimea numerelor de telefon să fie de exact 10 caractere precum și faptul că acest atribut conține doar caractere numerice validând astfel numerele de telefon ale clienților
   - detalii_clienti_email_ck (CHECK) - validează formatul adreselor de email folosind o expresie regulată
   - detalii_clienti_email_uk (UNIQUE) - impune unicitatea numerelor de telefon prevenind duplicatele

4. Entitatea **PRODUSE:**
   - produse_nume_ck (CHECK) - validează lungimea numelui produsului evitând astfel introducerea de nume incomplete sau generice
   - produse_pret_ck (CHECK) - asigură că prețul produsului este un număr pozitiv nenul
   - produse_stoc_ck (CHECK) - asigură ca stocul unui produs să fie mai mare sau egal cu 0, prevenind astfel ca stocul să devină negativ, fapt ce ar fi ilogic

- produse_pk (PRIMARY KEY) - definește atributul *id_produs* ca fiind cheie primară garantând astfel unicitatea fiecărui produs
- produse_nume_uk (UNIQUE) - asigură unicitatea numelor produselor, evitând introducerea de produse duplicat cu id-uri diferite

5. Entitatea **PRODUSE_VANZARI**:
    - produse_vanzari_pk (PRIMARY KEY) - definește combinația dintre *id_produs* și *id_vanzare* drept cheie primară garantând unicitatea fiecărei asocieri produs-vânzare

6. Entitatea **VANZARI**:
    - vanzari_pk (PRIMARY KEY) - definește *id_vanzare* drept cheie primară și asigură unicitatea fiecărei vânzări

Constrângeri de tip **FOREIGN KEY:**

1. categorii_produse_fk (în entitatea **PRODUSE**) - leagă *id_categorie* din **PRODUSE** de *id_categorie* din **CATEGORII** și garantează că produsele sunt asociate unor categorii valide

2. clienti_detalii_clienti_fk (în entitatea **DETALII_CLIENTI**) - leagă *id_client* din **DETALII_CLIENTI** de *id_client* din **CLIENTI** și garantează că detaliile sunt asociate unor clienți existenți

3. clienti_vanzari_fk (în entitatea **VANZARI**) - leagă *id_client* din **VANZARI** de *id_client* din **CLIENTI** și asigură că fiecare vânzare este asociată unui client valid

4. produse_vanzari_produse_fk (în entitatea **PRODUSE_VANZARI**) - leagă *id_produs* din **PRODUSE_VANZARI** de *id_produs* din **PRODUSE** și garantează că produsele dintr-o vânzare sunt valide

5. produse_vanzari_vanzari_fk (în entitatea **PRODUSE_VANZARI**) - leagă *id_vanzare* din **PRODUSE_VANZARI** de *id_vanzare* din **VANZARI** și garantează că vânzările asociate sunt valide

Triggerele implementate:

1. *trg_detalii_clientii_BRIU* - verifică dacă data_inregistrarii este mai mică sau egală cu data curentă, prevenind înregistrarea clienților la date viitoare invalide
2. *trg_vanzari_BRIU* - verifică dacă data_vanzare este mai mică sau egală cu data curentă, prevenind înregistrarea vânzărilor cu date viitoare

Autoincrement:

Autoincrementurile au fost create folosind clauza GENERATED BY DEFAULT AS IDENTITY, care permite generarea automată a valorilor pentru câmpurile cheie primare. Această clauză este specifică bazelor de date Oracle și permite setarea unor valori inițiale și

incrementale pentru coloanele care vor acționa ca identificate (ID-uri). Următoarele coloane sunt actualizate automat folosind autoincrementul:

- Tabela **CATEGORII** - *id_categorie* - valoare de start este 100 și este incrementată cu 2 la fiecare nouă inserare
- Tabela **CLIENTI** - *id_client* - valoare de start este 500 și este incrementată cu 1 la fiecare nouă inserare
- Tabela **PRODUSE** - *id_produs* - valoare de start este 200 și este incrementată cu 1 la fiecare nouă inserare
- Tabela **VANZARI** - *id_vanzare* - valoare de start este 1000 și este incrementată cu 5 la fiecare nouă inserare

Descrierea use-case-urilor

1. Introducerea unui produs nou: Acest scenariu presupune existența unui administrator al bazei de date ce poate fi chiar managerul de magazin sau un alt administrator. Acesta adaugă un nou produs în sistem, specificând detalii precum numele, prețul, stocul și categoria din care face parte acesta. Sistemul validează că prețul și stocul sunt constituite din valori valide și că numele produsului nu este deja folosit, iar dacă aceste condiții sunt îndeplinite, datele sunt actualizate printr-o comandă COMMIT.
2. Înregistrarea unui client nou: Această acțiune poate fi realizată de agentul de vânzări care completează informațiile relevante (nume, prenume, telefon, email). Sistemul validează formatul emailului și al telefonului, asigură unicitatea clientului, iar dacă aceste condiții sunt îndeplinite, datele sunt actualizate printr-o comandă COMMIT.
3. Înregistrarea unei vânzări: Agentul de vânzări înregistrează vânzarea în sistem, asociind-o cu clientul și produsele vândute. Sistemul actualizează stocurile și salvează detalii precum data vânzării, produsele implicate și clientul care a inițiat vânzarea respectivă.
4. Vizualizarea istoricului vânzărilor unui client: Un agent de vânzări dorește să vizualizeze istoricul vânzărilor pentru un anumit client, pentru a verifica produsele achiziționate anterior sau pentru a analiza comportamentul de cumpărare.

Tranzacții

În cadrul acestui proiect am folosit tranzacții la inserarea datelor de test în tabele CLIENTI și DETALII_CLIENTI, precum și în tabelele VANZARI si PRODUSE_VANZARI. De asemenea am mai folosit tranzacții în scripturile ce testează funcționalitățile use-case-urilor în care am folosit ROLLBACK-uri pentru a putea reveni la starea anterioară a bazei de date în cazul apariției erorilor și COMMIT-uri pentru a valida schimbările și pentru a încheia tranzacția.