**PROIECT BAZE DE DATE**

**-Magazin de haine online-**

Fîrțală Maria-Doina

Cuprins

[Descrierea modelului real, a utilității acestuia și a regulilor de funcționare. 3](#_Toc167193069)

[Descrierea entităților, incluzând precizarea cheii primare. 5](#_Toc167193070)

[Descrierea relațiilor, incluzând precizarea cardinalității acestora. 7](#_Toc167193071)

[Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor. 10](#_Toc167193072)

[Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5. 15](#_Toc167193073)

[Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectate la punctul 6. Diagrama conceptuală obținută trebuie să conțină minimum 7 tabele (fără considerarea subentităților), dintre care cel puțin un tabel asociativ 16](#_Toc167193074)

[Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7. 17](#_Toc167193075)

[Realizarea normalizării până la forma normală 3 (FN1-FN3). 18](#_Toc167193076)

[Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele (punctul 11). 21](#_Toc167193077)

[Crearea tabelelor în SQL și inserarea de date coerente în fiecare dintre acestea (minimum 5 înregistrări în fiecare tabel neasociativ; minimum 10 înregistrări în tabelele asociative; maxim 30 de înregistrări în fiecare tabel). 22](#_Toc167193078)

[Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente: 53](#_Toc167193079)

[Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri. 57](#_Toc167193080)

[Crearea unei vizualizări complexe. Dați un exemplu de operație LMD permisă pe vizualizarea respectivă și un exemplu de operație LMD nepermisă. 62](#_Toc167193081)

[Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outer-join pe minimum 4 tabele, o cerere ce utilizează operația division și o cerere care implementează analiza top-n. 65](#_Toc167193082)

[Optimizarea unei cereri, aplicând regulile de optimizare ce derivă din proprietățile operatorilor algebrei relaționale. Cererea va fi exprimată prin expresie algebrică, arbore algebric și limbaj (SQL), atât anterior cât și ulterior optimizării. 65](#_Toc167193083)

[a. Realizarea normalizării BCNF, FN4, FN5. 66](#_Toc167193084)

[b. Aplicarea denormalizării, justificând necesitatea acesteia. 69](#_Toc167193085)

**CUPRINS:**

**1**. Descrierea modelului real, a utilității acestuia și a regulilor de funcționare.

**2**. Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului.

**3**. Descrierea entităților, incluzând precizarea cheii primare.

**4**. Descrierea relațiilor, incluzând precizarea cardinalității acestora.

**5**. Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor.

**6**. Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5.

**7**. Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectate la punctul 6. Diagrama conceptuală obținută trebuie să conțină minimum 7 tabele (fără considerarea subentităților), dintre care cel puțin un tabel asociativ.

**8**. Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7.

**9**. Realizarea normalizării până la forma normală 3 (FN1-FN3).

**10**. Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele (punctul 11).

**11**. Crearea tabelelor în SQL și inserarea de date coerente în fiecare dintre acestea (minimum 5 înregistrări în fiecare tabel neasociativ; minimum 10 înregistrări în tabelele asociative; maxim 30 de înregistrări în fiecare tabel).

**12**. Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente:

a) subcereri sincronizate în care intervin cel puțin 3 tabele

b) subcereri nesincronizate în clauza FROM

c) grupări de date, funcții grup, filtrare la nivel de grupuri cu subcereri nesincronizate (în clauza de HAVING) în care intervin cel puțin 3 tabele (in cadrul aceleiași cereri)

d) ordonări și utilizarea funcțiilor NVL și DECODE (în cadrul aceleiași cereri)

e) utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 funcții pe date calendaristice, a cel puțin unei expresii CASE

f) utilizarea a cel puțin 1 bloc de cerere (clauza WITH)

Observație: Într-o cerere se vor regăsi mai multe elemente dintre cele enumerate mai sus, astfel încât cele 5 cereri să le cuprindă pe toate.

**13**. Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri.

**14**. Crearea unei vizualizări complexe. Dați un exemplu de operație LMD permisă pe vizualizarea respectivă și un exemplu de operație LMD nepermisă.

**15**. Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outer-join pe minimum 4 tabele, o cerere ce utilizează operația division și o cerere care implementează analiza top-n.

Observație: Cele 3 cereri sunt diferite de cererile de la exercițiul 12.

**16**. Optimizarea unei cereri, aplicând regulile de optimizare ce derivă din proprietățile operatorilor algebrei relaționale. Cererea va fi exprimată prin expresie algebrică, arbore algebric și limbaj (SQL), atât anterior cât și ulterior optimizării.

**17**. a. Realizarea normalizării BCNF, FN4, FN5.

b. Aplicarea denormalizării, justificând necesitatea acesteia.

# Descrierea modelului real, a utilității acestuia și a regulilor de funcționare.

O studentă la Facultatea de Matematică și Informatică dorește să dezvolte o platformă pentru achiziționarea hainelor. În cadrul magazinului online, clienții au posibilitatea să-și creeze conturi personalizate și să se autentifice pe site. Odată autentificați, ei pot gestiona detaliile contului lor, inclusiv adresele de livrare. Clienții pot să stocheze mai multe adrese de livrare și să selecteze una dintre ele ca adresă implicită, care va fi folosită automat pentru comenzile lor viitoare.

Utilizatorii au posibilitatea să verifice statusul comenzilor lor în orice moment. Această funcționalitate le oferă transparență și control, permițându-le să urmărească progresul comenzilor lor de la plasare până la livrare. Astfel, ei pot fi la curent cu starea fiecărei comenzi și să se asigure că vor primi produsele dorite în timp util.

Reguli de funcționare:

* **Autentificare și conturi utilizatori**:
  + Utilizatorii trebuie să-și creeze conturi personalizate pentru a accesa funcționalitățile complete ale magazinului online.
  + Fiecare utilizator trebuie să ofere informații de autentificare valide (cum ar fi adresă de email și parolă) pentru a accesa contul său.
* **Gestionarea contului utilizatorului**:
  + Utilizatorii au responsabilitatea să-și mențină informațiile de cont actualizate și să furnizeze date precise pentru livrare.
  + Ei pot să-și modifice sau să-și șteargă conturile, dar trebuie să respecte procedurile și restricțiile stabilite de platformă.
* **Gestionarea adreselor de livrare**:
  + Utilizatorii pot să stocheze mai multe adrese de livrare și să selecteze una dintre ele ca adresă implicită. Acest lucru este util daca clientul doreste ca livrarea sa se realizeze acasa, la locul de munca etc.
  + Este responsabilitatea lor să se asigure că adresele furnizate sunt valide și precise pentru a evita problemele de livrare.
* **Plasarea și urmărirea comenzilor**:
  + Utilizatorii trebuie să respecte procedurile de plasare a comenzilor și să furnizeze informații complete pentru fiecare comandă (cum ar fi adresa de livrare, modalitatea de plată etc.).
  + Ei pot verifica statusul comenzilor lor în orice moment și să contacteze echipa de suport pentru orice întrebări sau nelămuriri.
  + Utilizatorii pot vizualiza istoricul comenzilor lor
* **Gama larga de produse**:
* Site-ul ofera o gama variata de produse, organizate pe categorii precum incaltaminte, jachete, rochii, pantaloni etc.
* Fiecare produs este prezentat cu detalii precum descrierea, instrucțiunile de ingrijire și imaginile produsului.
* Utilizatorii pot vizualiza variatele opțiuni disponibile pentru fiecare produs, inclusiv culorile și mărimile disponibile.
* **Promotii:**
* Site-ul ofera periodic promotii pentru anumite categorii de produse.
* Fiecare promoție este asociată cu o perioadă de valabilitate și un discount specific, care se aplică automat produselor eligibile.
* **Gestionarea stocului si depozite:**
  + Produsele sunt gestionate în depozite, fiecare depozit având o capacitate maximă de stocare.
  + Sistemul actualizează automat stocurile disponibile pentru fiecare produs, afișând informații în timp real pentru utilizatori.

1. **Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului.**

* Fiecare entitate trebuie să aibă un identificator unic (cheie primară).
* Relațiile dintre entități trebuie să fie definite și menționate în mod corespunzător.
* Cardinalitățile relațiilor trebuie specificate în funcție de logica modelului, indicând numărul minim și maxim de legături între entități (ex: 1(0) la M(0), M(0) la M(0)).
* Atributele fiecărei entități trebuie definite, specificând tipul de date, constrângerile și valorile implicite (dacă există).
* Parolele vor fii stocate direct criptate.
* Informațiile personale furnizate de utilizatori (cum ar fi adresele de livrare) trebuie să fie valide și să respecte formatele acceptate.
* Utilizatorii trebuie să respecte procedurile și termenii de plată atunci când plasează o comandă.
* Comenzile trebuie să fie plasate conform disponibilității produselor și a politicilor de livrare ale magazinului online.
* Produsele disponibile pentru achiziție trebuie să fie actualizate și să reflecte stocul real al magazinului online.
* O promotie este activa doar pentru o anumita perioada de timp.
* Promoțiile pot fi aplicate doar pentru anumite categorii, iar aceste categorii vor avea doar cate o promotie pana aceasta se finalizeaza.
* Clientul nu poate plasa o comanda pentru haine care nu mai sunt in stoc.
* Comenzile se livreaza pana in max 14 zile.

# Descrierea entităților, incluzând precizarea cheii primare.

Entitatile sunt:

1. PROMOTIE,
2. PRODUS\_CATEGORIE,
3. PRODUS,
4. MARIMI\_CATEGORIE,
5. PRODUS\_ITEM,
6. OPTIUNE\_MARIME,
7. DEPOZIT,
8. STATUS\_COMANDA,
9. COMANDA,
10. USER,
11. ADRESA,
12. TARA,
13. RECENZIE,
14. IMAGINE\_PRODUS
15. CULOARE

USER = un client inregistrat pe site care poate plasa comenzi, cheia primara este id\_user#

ADRESA = o lista cu toate adresele din sistem, care sunt folosite pentru a livra comenzi. Cheia primara este id\_adresa#

TARA = o tabela pentru tarile care sunt folosite la adrese(am creat aceasta entitate pentru o gestionare mai usoara a datelor). Cheia primara este id\_tara#

COMANDA = evidenta unei comenzi care a fost plasata de catre un user pentru una sau mai multe produse. Cheia primara este id\_comanda#

STATUS\_COMANDA = tine evidenta statusului unei comenzi (comanda a fost plasata, in tranzit, livrata, anulata). Cheia primara este id\_status#

PRODUS\_CATEGORIE = o lista de categorii la care apartin produsele. Cheia primara este produs\_cat\_id#

PROMOTIE = promotiile oferite de companie. Cheia prima este promotie\_id#

PRODUS = reprezentarea unui produs. Cheia primara este id\_produs# (ex: tricou)

PRODUS\_ITEM = instanta unui produs cu toate variantele lui. Cheia primara este id\_produs\_item#. (ex: tricou rosu, verde, negru)

RECENZIE = reprezinta recenzia pe care o da user-ul unui produs din magazin. Cheia primara este id\_recenzie#

IMAGINE\_PRODUS = reprezinta diferitele imagini pe care le poate avea un produs. Cheia primara este id\_imagine#

MARIMI\_CATEGORIE = reprezinta diferitele marimi pe care le pot avea hainele (Incaltaminte, Rochii, Pantaloni). Cheia primara este id\_marimi\_cat#

OPTIUNE\_MARIME = reprezinta valoarea din tabelul marimi\_categorie. (Incaltamintea are altfel de marimi fata de haine, M, 40 etc). Cheia primara id\_optiune\_marime#

DEPOZIT = reprezinta locul unde se proceseaza comenzile care pleaca spre useri. Cheia primara este id\_depozit#

CULOARE = reprezinta diferitele culori pe care un item le poate avea. Cheia primara este id\_culoare#

# Descrierea relațiilor, incluzând precizarea cardinalității acestora.

**USER are ADRESA**

Relatia are cardinalitatea minima 1:1 :

Cati useri trebuie sa aiba o adresa? Cel putin 1

Cate adrese trebuie sa aiba un user? Macar 1

Relatia are cardinalitatea maxima M:M :

Cati useri pot avea aceeasi adresa? Mai multi => M

Cate adrese poate avea un user? Mai multe => M

Aceasta relatie arata ca un utilizator poate sa aiba mai multe adrese (loc de munca, acasa), dar trebuie sa aiba cel putin una, iar o adresa poate avea mai multi utilizatori (fiecare membru al familiei comanda), dar cel putin un utilizator deoarece e inutil sa retinem o adresa daca aceasta nu are un utilizator.

**ADRESA are o TARA**

Relatia are cardinalitatea minima 0:1 :

Cate tari trebuie sa aiba o adresa? Cel putin 1

Cate adrese trebuie sa aiba o tara? Cel putin 1

Relatia are cardinalitatea maxima M:1 :

Cate tari poate sa aiba o adresa? 1

Cate adrese poate sa aiba o tara? Mai multe => M

**COMANDA are STATUS\_COMANDA**

Relatia are cardinalitate minima 1:1 :

Cate comenzi trebuie sa aiba un status? Macar 1

Cate statusuri trebuie sa aiba o comanda? Macar 1

Relatia are cardinalitate maxima M:1 :

Cate comenzi pot avea un status? Mai multe => M

Cate statusuri poate avea o comanda? 1

Aceasta relatie arata ca fiecare comanda trebuie sa aiba cel putin un status asociat, iar fiecare comanda este asociata cu exact un status.

**MARIMI\_CATEGORIE are OPTIUNE\_MARIME**

Relatia are cardinalitate minima 1:1 :

Cate marimi trebuie sa aiba o optiune? Macar 1

Cate optiuni trebuie sa aiba o marime? Macar 1

Relatia are cardinalitate maxima 1:M :

Cate marimi poate avea o optiune? 1

Cate optiuni poatea avea o marime? M

Aceasta relatie arata ca fiecare varietate trebuie sa fie asociata cu cel putin o optiune (Marime are macar M) si fiecare optiune trebuie sa fie asociata cu cel putin o varietate (S apartine Marimilor). Optiunea apartine cel mult unei varietati (S nu poate sa fie la culori), dar o varietate poate sa aiba mai multe optiuni (S, M, L…)

**PRODUS\_CATEGORIE are MARIME\_CATEGORIE**

Relatia are cardinalitate minima 1:1 :

Cate produse\_cat trebuie sa aiba o marime\_cat? Macar 1

Cate marimi\_cat trebuie sa aiba un produs\_categorie? Macar 1

Relatia are cardinalitate maxima M:M :

Cate categorii poate sa aiba o marime? M

Cate marimi poate sa aiba o categorie? M

Aceasta relatie arata ca o marime trebuia sa apartina unei categorii si invers (nu are sens sa definim o marime care nu apartine unei categorii, sau o categorie fara cel putin o marime). De asemenea, diferite marimi pot apartine unui produs (US, EU), iar diferite produse pot apartine unei marimi (Tricouri femei, Tricouri barbati).

**PRODUS\_CATEGORIE are PROMOTIE**

Relatia are cardinalitate minima 1:0 :

Cate categorii trebuie sa aiba o promotie? 0

Cate promotii trebuie sa aiba o categorie? 1

Relatia are cardinalitate maxima M:1 :

Cate categorii poate avea o promotie? M

Cate promotii poate avea o categorie? 1

Aceasta relatie arata ca fiecare promotie trebuie sa fie asociata cu cel putin o categorie, dar nu toate categoriile trebuie sa aiba promotii. De asemenea, mai multe categorii pot avea aceeasi promotie, iar o promotie poate fi asociata cu o singura categorie.

**PRODUS\_CATEGORIE are PRODUS**

Relatia are cardinalitate minima 1:1 :

Cate categorii trebuie sa aiba un produs? 1

Cate produse trebuia sa aiba o categorie? 1

Relatia are cardinalitate maxima 1:M :

Cate categorii poate sa aiba un produs? 1

Cate produse poate sa aiba o categorie? M

Aceasta relatie ne arata ca nu exista produse fara categorie sau nu exista categorii goale. Un produs apartine unei singure categorii, iar o categorie contine mai multe produse.

**PRODUS are PRODUS\_ITEM**

Relatia are cardinalitate minima 1:1 :

Cate produse trebuie sa aiba un produs item? Cel putin 1

Cate produse-item trebuie sa aiba un produs? Cel putin 1

Relatia are cardinalitate maxima 1:M:

Cate produse poate sa aiba un produs-item? 1

Cate produse-item poate sa aiba un produs? Mai multe => M

Aceasta relatie indica faptul ca un produs\_item trebuie sa fie asociat cu cel putin si maxim un produs (variatiile unui produs apartin acelui produs), iar un produs poate avea mai multe produs-item-uri asociate, deoarece un produs poate avea mai multe variante sau configurații.

**PRODUS\_ITEM** **are RECENZIE**

Relatia are cardinalitate minima 0:0 :

Cate produse trebuie sa aiba o recenzie? 0 (nu e obligatoriu ca userul sa dea o recenzie)

Cate recenzii trebuie sa aiba un produs? 0 (inca nu are recenzie)

Relatia are cardinalitate maxima 1:M :

Cate produse poate sa aiba o recenzie? 1

Cate recenzii poate sa aiba un produs? M

Aceasta relatie ne arata ca un produs nu trebuie sa aiba o recenzie, ia

**PRODUS\_ALES are COMANDA**

Relatia are cardinalitate minima 1:0 :

Cate produse\_alese trebuie sa aiba o comanda? 1

Cate comenzi trebuie sa aiba un produs\_ales? 0

Relatia are cardinalitate maxima M:M :

Cate produse\_alese poate sa aiba o comanda? M

Cate comenzi poate sa aiba un produs\_ales? M

Aceasta relatie ne arata ca fiecare comanda trebuie sa aiba un produs\_ales. Asta inseamna ca o comanda nu poate exista fara un produs. Un produs\_ales nu trebuie sa faca parte dintr-o comanda neaparat (de ex userul a selectat un produs dar nu a continuat cu plata sau inca nu a plasat comanda). O comanda poate avea un numar nelimitat de produse, iar un produs se afla in mai multe comenzi, ceea ce sugereaza ca acelasi produs poate fi achizitionat de mai multi clienti in comenzi diferite.

**PRODUS\_ITEM are IMAGINE\_PRODUS**

Relatia are cardinalitate minima 1:1 :

Cate produse trebuie sa aiba o imagine? 1

Care imagini trebuie sa aiba un produs? 1

Relatia are cardinalitate maxima 1:M :

Cate produse poate sa aiba o imagine? 1

Cate imagini poate sa aiba un produs? M

**PRODUS\_ITEM are CULOARE**

Relatia are cardinalitate minima 1:1 :

Cate produse trebuie sa aiba o culoara? 1

Cate culori trebuie sa aiba un produs? 1

Relatia are cardinalitate maxima M:1 :

Cate produse poate sa aiba o culoare? M

Cate culori poate sa aiba un produs? 1

Aceasta relatie ne arata ca fiecare produs\_item are mai multe culori, dar o culoare apartine unui singur produs\_item.

**PRODUS are DEPOZIT**

Relatia are cardinalitate minima 1:1 :

Cate produse trebuie sa aiba un depozit? 1

Cate depozituri trebuie sa aiba un produs? 1

Relatia are cardinalitate maxima :

Cate produse poate sa aiba un depozit? M

Cate depozituri poate sa aiba un produs? M

Aceasta relatie ne arata ca un produs apartine mai multor depozite, iar un depozit contine mai multe produse.

**RELATIA CUMPARA:**

Un user poate cumpara mai multe comenzi, iar aceste comenzi vor fi procesate de catre depozit => cardinalitatea la comanda este M(0) (userul poate sa nu comande nimic)

O comanda care este procesata intr-un depozit este cumparata de un singur user => cardinalitatea este 1(1) la user.

Un user poate plasa mai multe comenzi, pe care le va cumpara ulterior dintr-un depozit => cardinalitatea la depozit este M(0) (un depozit poate procesa mai multe comenzi, dar de asemenea pot sa nu existe comenzi ce trebuiesc procesate)

# Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor.

USER are urmatoarele atribute:

* id\_user – variabila de tip intreg, e cheie primara (NOT NULL si UNIQUE). Reprezinta id-ul userului
* adresa\_email – variabila de tip string (varchar), e unica. Reprezinta adresa de email a utilizatorului aceasta fiind unica pentru identificarea corecta a userului.
* nr\_telefon – variabila de tip string (varchar, unele nr pot avea – sau paranteze, in plus e posibil sa fie nevoie sa identificam codul tarii (ex: +40 070000000)), unic. Ca si constrangere avem faptul ca un nr de telefon trebuie sa respecte un format specific (10 cifre iar primele doua incep cu 07). Stochează numărul de telefon al unui utilizator
* parola – variabila de tip string (varchar), not null. Constrangeri: lungime minima, complexitate (, securitate). Stochează parola asociată cu contul utilizatorului în baza de date.

ADRESA are urm atribute:

* id\_adresa – variabila de tip intreg, e cheie primara. Reprezinta id-ul adresei
* cod\_postal – variabila de tip varchar. Costrangerea ar fii lungimea si not null. Stochează codul poștal asociat unei adrese în baza de date
* oras – variabila de tip varchar, not null. Stochează numele orașului asociat unei adrese în baza de date
* adresa\_linie1 – variabila de tip varchar, not null. Stochează prima linie a adresei asociate unei locații în baza de date.
* adresa\_linie2 – variabila de tip varchar (poate fi null daca incape tot in prima). Stochează a doua linie a adresei asociate unei locații în baza de date.
* id\_tara - variabila de tip intreg, foreign key. Valori posibile: aceleasi cu cele din entitatea tara. Acest atribut este utilizat pentru a identifica țara asociată unei anumite locații sau adrese.

TARA are urm atribute:

* id\_tara - variabila de tip intreg, cheie primara. Reprezinta id-ul tarii
* nume\_tara – variabila de tip varchar, unique. Acest atribut stochează numele unei țări și este utilizat pentru a identifica țările în cadrul sistemului sau aplicației. Unicitatea acestui atribut este importantă pentru a evita duplicarea numelui țări.

COMANDA are urm atribute:

* id\_comanda - variabila de tip intreg, cheie primara. Reprezinta id-ul comenzii
* data\_comanda– variabila de tip data(time). Valoarea implicita: cand e calculata sa ajunga comanda(peste 7 zile de la plasarea comenzii). Acest atribut stochează data la care este programată livrarea comenzii.
* total\_plata – variabila de tip intreg, valoare default 0.00, not null. Reprezinta cat are de platit user-ul dupa ce face comanda
* metoda\_plata - variabila de tip varchar, not null. Valori posibile: “Plata cu cardul”, “ plata curier”. Acest atribut stochează metoda de plată utilizată pentru a efectua plata comenzii.
* status\_comanda – variabila de tip intreg, foreign key. Reprezinta statusul comenzii. Valori posibile: aceleasi cu cele din entitatea status\_comanda.

STATUS\_COMANDA are urm atribute:

* id\_status – variabila de tip intreg, cheie primara. Reprezinta statusul comenzii.
* status - variabila de tip varchar. **Valori posibile**  "În așteptare", "În procesare", "Expediată", "Livrare în curs", "Livrare finalizată", etc. Acest atribut stochează statusul actual al comenzii, indicând în ce stadiu se află procesul de îndeplinire a comenzii.

PRODUS\_ITEM are urm atribute:

* id\_produs\_item - variabila de tip intreg, cheie primara. Reprezinta id-ul produsului
* id\_produs - variabila de tip intreg, foreign key. Arata spre tabela produs care ne arata carui produs ii apartine.
* pret\_original – variabila de tip decimal, not null, pozitiv. Reprezinta pretul original al produsului (poate diferi la produse de acelasi fel dar alta culoare)
* pret\_dupa\_reducere – variabila de tip decimal, pozitiv. Reprezinta pretul dupa reducere al produsului
* culoare – variabila de tip varchar. Reprezinta culoarea produsului
* cantitate\_in\_stoc – variabila de tip int. Reprezinta cantitatea acelui produs si daca se afla in stoc. Daca valoarea e 0 se va afisa mesajul “nu se afla in stoc”.

OPTIUNE\_MARIME are urm atribute:

* id\_optiune\_marime - variabila de tip intreg, cheie primara. Reprezinta id-ul optiunii alese.
* id\_categorie\_marime – variabila de tip intreg, foreign key. Valori posibile: aceleasi cu cele din entitatea varietate.
* marime\_nume – variabila de tip varchar. Reprezinta valoarea aleasa de catre user (S sau L sau M; rosu sau verde sau galben; bumbac sau sintetic)
* ordonare – variabila de tip int. Reprezinta felul in care vrem sa afisam marimile (ordonarea alfabetica nu arata bine)

MARIME\_CATEGORIE are urm atribute:

* id\_marimi\_cat - variabila de tip intreg, cheie primara. Reprezinta id-ul categoriei.
* nume – variabila de tip varchar, unica(sa nu am categorii duplicate. Stocheaza numele categoriei

PRODUS are urm atribute:

* id\_produs - variabila de tip intreg, cheie primara. Reprezinta id-ul produsului
* id\_categorie - variabila de tip intreg, foreign key. Arata catre tabela produs\_categorie pentru a stii carei categorii ii apartine produsul.
* nume – variabila de tip varchar. Stocheaza numele produsului.
* instructiuni\_ingrijire – variabila de tip varchar. Stocheaza informatii despre ingrijirea hainei
* descriere – variabila de tip varchar. Stocheaza o mica descriere despre produs (va fii aceeasi pentru acelasi item dar cu diferite culori)

PRODUS\_CATEGORIE are urm atribute:

* produs\_cat\_id – variabila de tip intreg, cheie primara. Reprezinta id-ul categoriei de produse
* nume\_cat – variabila de tip varchar, not null. Stocheaza numele categoriei (Incaltaminte, Jachete etc)
* imagine\_cat - variabila de tip varchar, not null. Stocheaza URL-ul catre o imagine sugestiva, specifica categoriei.
* parinte\_cat\_id - variabila de tip intreg, foreign key. Reprezinta un id catre entitatea produs\_categorie, pentru a sublinia ca avem categorii care apartin altor categorii (FEMEI -> PANTOFI FEMEI)
* promotie\_id – variabila de tip int, cheie straina. Reprezinta id\_ul catre entitatea promotie, ca sa stim caror categorii li se aplica promotiile.

PROMOTIE are urm atribute:

* promotie\_id - variabila de tip intreg, cheie primara. Reprezinta id-ul promotiei
* nume\_promotie - variabila de tip varchar. Stocheaza numele promotiei disponibile in magazinul online.
* descriere - variabila de tip varchar. Reprezinta o scurta descriere despre promotie
* discount – variabila de tip int. Constrangeri: combinatie de 2 nr care se termina in %. Reprezinta discountul care poate fi adaugat unui produs.
* data\_incepere – variabila de tip date (time). Reprezinta data de incepere a promotiei
* data\_incheiere – variabila de tip date (time). Reprezinta data de incheiere a promotiei

RECENZIE are urm atribute:

* id\_recenzie – variabila de tip int, cheie primara. Reprezinta id-ul recenziei
* id\_produs\_item – variabila de tip int, cheie straina. Reprezinta legatura catre tabelul PRODUS\_ITEM pt a arata ca un produs are mai multe recenzii
* rating – variabila de tip int. Reprezinta valoarea pe care a lasat-o userul unui produs, poate fi intre 1 si 5.
* comentariu – variabila de tip varchar. Reprezinta un comentariu pe care userul la lasat unui produs.

IMAGINE\_PRODUS are urm atribute:

* id\_imagine – variabila de tip int, cheie primara. Reprezinta id-ul imaginii
* id\_produs\_item – variabila de tip int, foreign key. Reprezinta id-ul produsului careia ii apartine imaginea.
* nume\_fisier – variabila de tip caracter. Reprezinta locatia imaginii de pe site.

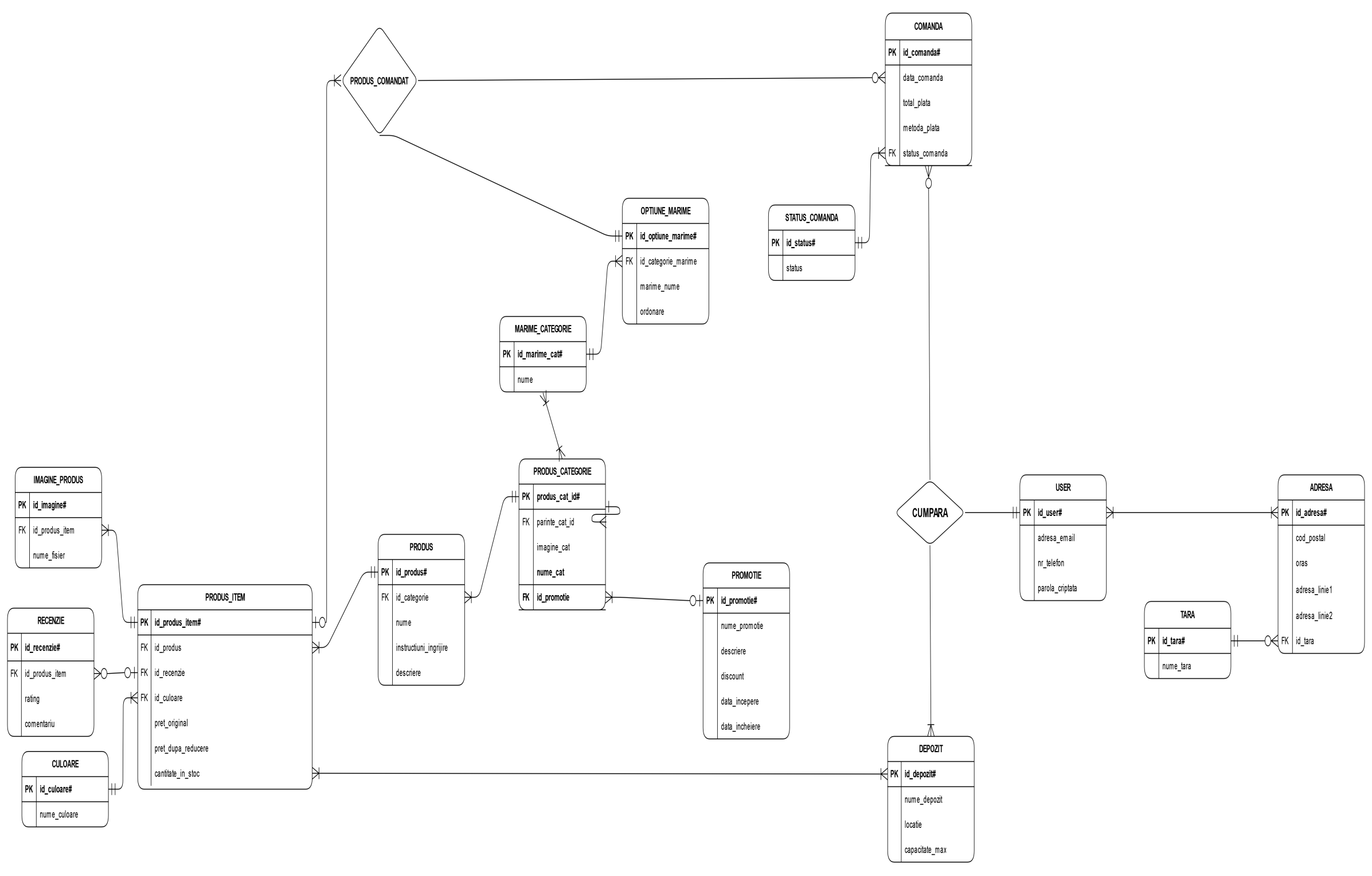
DEPOZIT are urm atribute:

* id\_depozit – variabila de tip int, cheie primara. Reprezinta id-ul depozitului
* nume\_depozit – variabila varchar, constrangere: are doar o litera si o cifra in nume (ex: T1, H2 etc). Reprezinta numele depozitului
* locatie – variabila de tip varchar, unique, not null. Reprezinta locatia la care se afla depozitul (Bucuresti, Iasi, Brasov, Paris etc)
* capacitate\_max – variabila de tip int, are o limita superioara. Reprezinta capacitatea de stocare a depozitului.

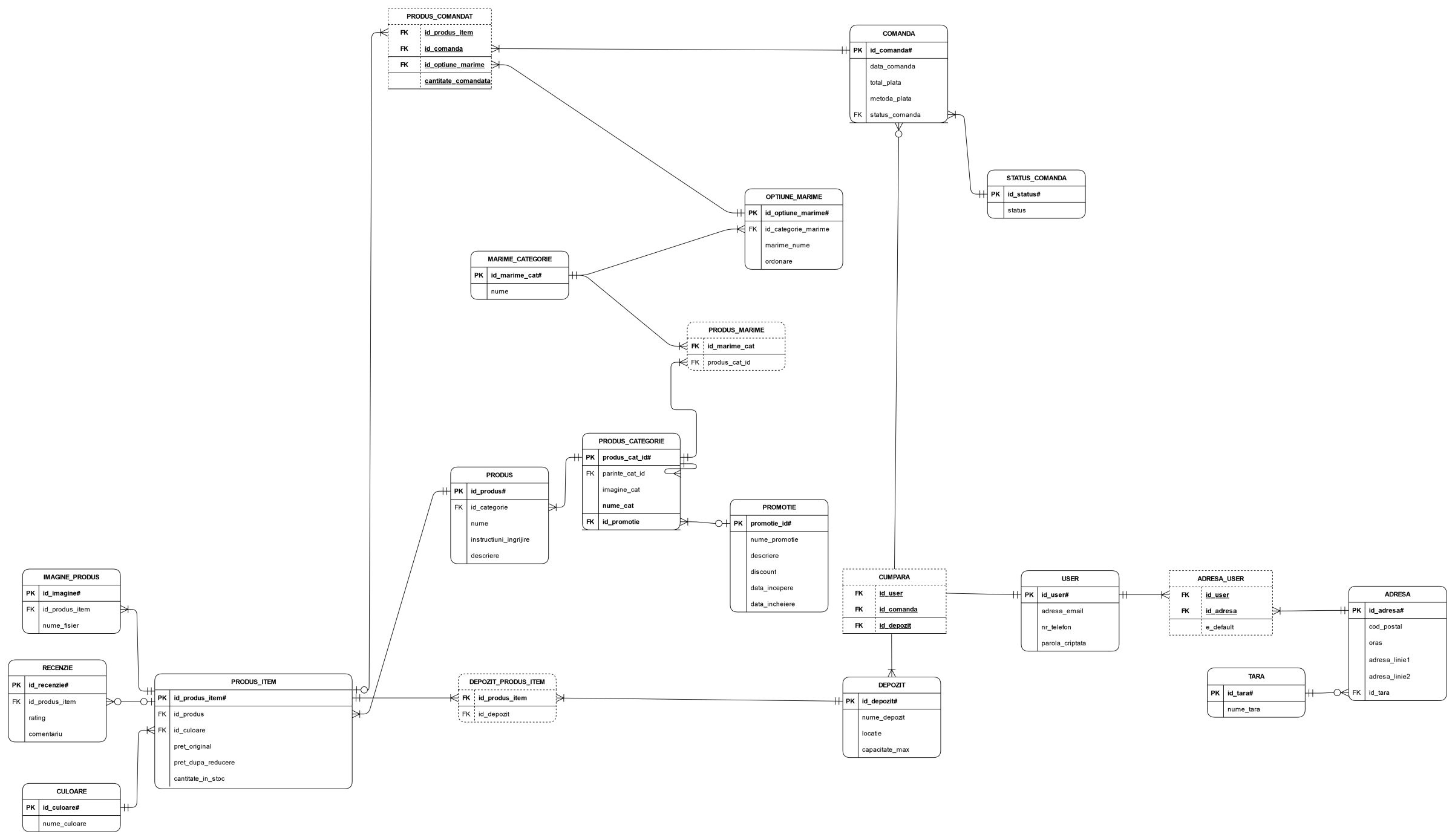
CULOARE are urm atribute:

* id\_culoare – variabila de tip int, cheie primara. Reprezinta id-ul culorii
* nume\_culoare – variabila de tip caracter, unica. Reprezinta numele culorii pe care un item il poate avea

# Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5.

****

# Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectate la punctul 6. Diagrama conceptuală obținută trebuie să conțină minimum 7 tabele (fără considerarea subentităților), dintre care cel puțin un tabel asociativ

****

# Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7.

Schemele relationale sunt:

PROMOTIE(id\_promotie#, nume\_promotie, descriere, discount, data\_incepere, data\_incheiere),

MARIME\_CATEGORIE(id\_marime\_cat#, nume),

PRODUS\_CATEGORIE(id\_produs\_cat#, id\_parinte\_cat(FK), promotie\_id, imagine\_cat, nume\_cat),

PRODUS\_MARIME(id\_marime\_cat(FK), id\_produs\_cat(FK))

PRODUS(id\_produs#, id\_categorie(FK), nume, instructiuni\_ingrijire, descriere),

OPTIUNE\_MARIME(id\_optiune\_marime#, id\_categorie\_marime(FK),marime\_nume, ordonare),

IMAGINE\_PRODUS(id\_imagine#, id\_produs\_item(FK), nume\_fisier),

RECENZIE(id\_recenzie#, id\_user, comentariu, rating),

CULOARE(id\_culoare#, nume\_culoare),

PRODUS\_ITEM(id\_produs\_item#, id\_produs(FK), id\_recenzie(FK), culoare, pret\_original, pret\_dupa\_reducere),

TARA(id\_tara#, nume\_tara),

ADRESA(id\_adresa#, cod\_postal, oras, adresa\_linie1, adresa\_linie2, id\_tara(FK)),

USER(id\_user#, adresa\_email, nr\_telefon, parola\_criptata),

ADRESA\_USER(id\_user(FK), id\_adresa(FK), e\_default),

DEPOZIT(id\_depozit#, nume\_depozit, locatie, capacitate\_max),

DEPOZIT\_PRODUS\_ITEM(id\_produs, id\_depozit),

COMANDA(id\_comanda#, data\_estimata\_livrare, total\_plata, metoda\_plata, status\_plata(FK)),

STATUS\_COMANDA(id\_status#, status),

CUMPARA(id\_user#, id\_comanda#, id\_depozit#, data\_achizitie),

PRODUS\_COMANDAT(id\_produs\_item(FK), id\_comanda(FK), id.optiune\_marime(FK), cantitate\_comandata).

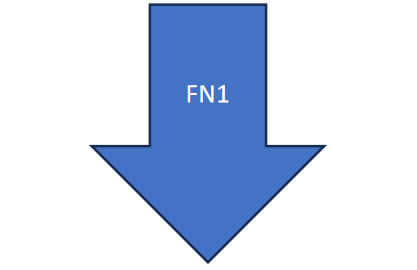
# Realizarea normalizării până la forma normală 3 (FN1-FN3).

Exemplu pentru FN1:

In tabelul de mai jos avem informatie despre un produs, si anume, id-ul si culoarea acestuia.

In coloana culoare observam ca avem mai multe valori deoarece un produs poate sa aiba mai multe culori. Asadar nu respecta FN1. Vom transforma tabelul in felul urmator

|  |  |
| --- | --- |
| id\_produs\_item# | culoare |
| 1 | negru, albastru |
| 2 | rosu, alb, verde |

****

|  |  |
| --- | --- |
| id\_produs\_item# | culoare |
| 1 | negru |
| 1 | albastru |
| 2 | rosu |
| 2 | alb |
| 2 | verde |

Exemplu pt FN2:

Tabelul se va afla in FN2 daca mai intai se afla in FN1 si mai apoi nu va contine dependente partiale.

Tabelul contine id\_user, id\_comanda, id\_depozit si data\_achizitie.

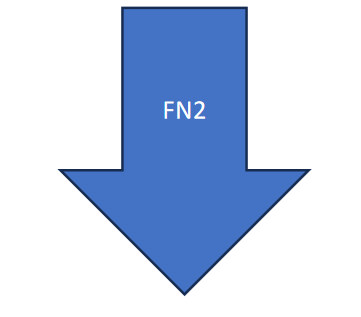
Atributul data\_achizitie depinde doar de id\_comanda

Pentru a aduce tabelul in FN2 vom descompune urmatorul tabel in 2 tabele astfel:

CUMPARA1(id\_user, id\_comanda, id\_depozit)

CUMPARA2(id\_comanda, data\_achizitiei)

|  |  |  |  |
| --- | --- | --- | --- |
| id\_user | id\_comanda | id\_depozit | data\_achizitie |
| 1 | 1 | 1 | 2024-10-17 |
| 2 | 2 | 2 | 2024-7-29 |



|  |  |  |
| --- | --- | --- |
| id\_user | id\_comanda | id\_depozit |
| 1 | 1 | 1 |
| 2 | 2 | 2 |

|  |  |
| --- | --- |
| id\_comanda | data\_achizitie |
| 1 | 2024-10-17 |
| 2 | 2024-7-29 |

Exemplu pt FN3:

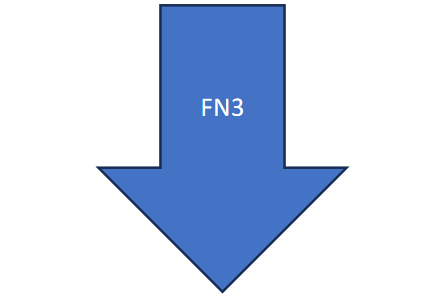
Ca tabelul sa se afle in FN3 trebuie mai intai sa se afle in FN2, iar a doua conditie este ca tabelul contine o dependenta tranzitiva.

Tabelul contine urm atribute: id\_comanda, status\_comanda, data\_achizitie, depozit, expediere\_colet.

depozit -> reprezinta depozitul de unde se va trimite comanda

Dupa cum putem observa avem dependenta tranzitiva deoarece expediere\_colet nu depinde neaparat de comanda ci de depozitul de unde aceasta va fi trimisa. Asadar despartim tabelul astfel:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id\_comanda | status\_comanda | data\_achizitie | depozit | expediere\_colet |
| 1 | expediat | 2024-10-22 | R1 | 2024-11-5 |



|  |  |  |  |
| --- | --- | --- | --- |
| id\_comanda | status\_comanda | data\_achizitie | id\_depozit |
| 1 | expediat | 2024-10-22 | 1 |

|  |  |  |
| --- | --- | --- |
| id\_depozit | nume\_depozit | expediere\_colet |
| 1 | R1 | 2024-11-5 |

# Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele (punctul 11).

CREATE SEQUENCE promotie\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE produs\_cat\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE produs\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE marime\_cat\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE optiune\_marime\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE imag\_prod\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE recenzie\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE produs\_item\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE tara\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE adresa\_seq START WITH 1 INCREMENT BY 1;

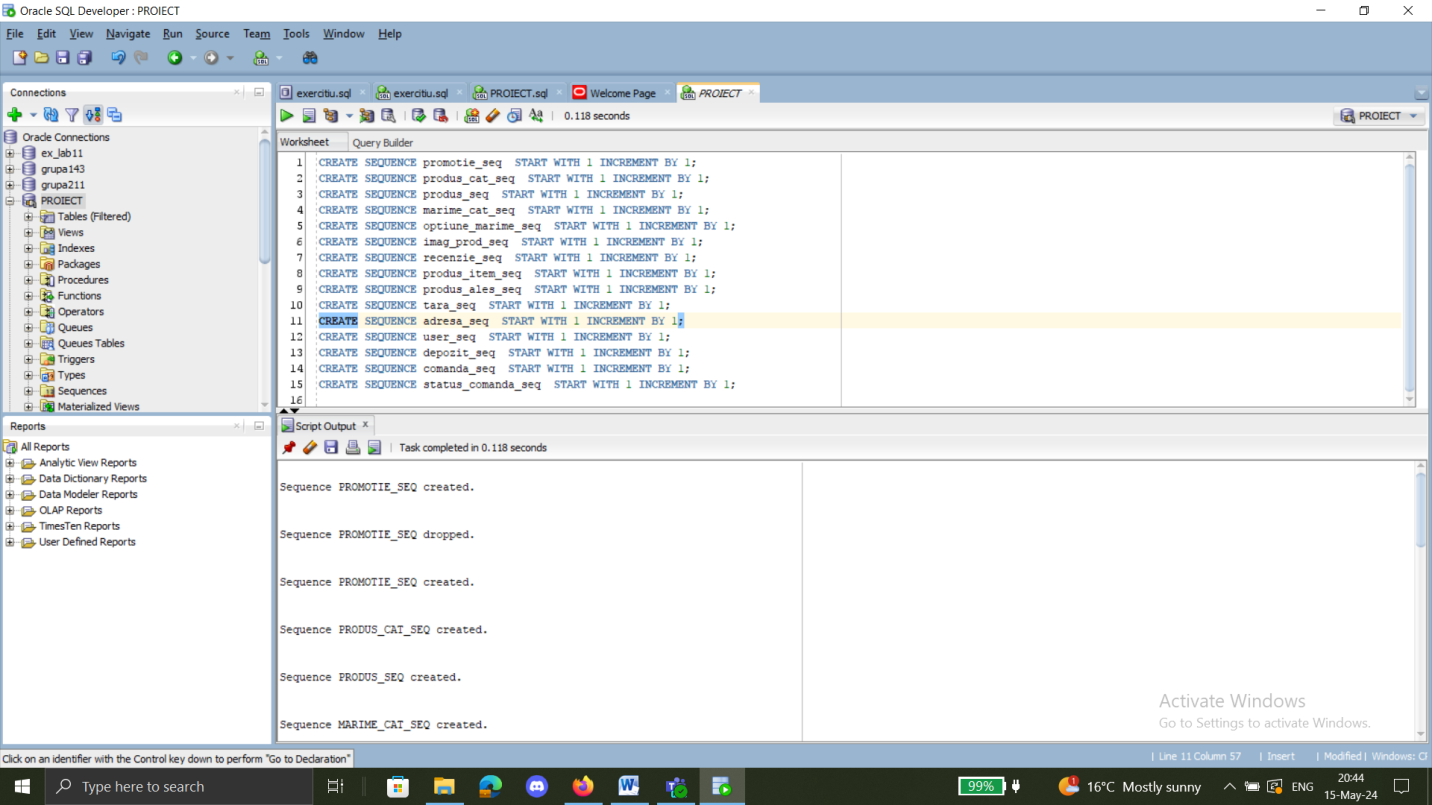
CREATE SEQUENCE user\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE depozit\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE comanda\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE status\_comanda\_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE culoare\_seq START WITH 1 INCREMENT BY 1;



# Crearea tabelelor în SQL și inserarea de date coerente în fiecare dintre acestea (minimum 5 înregistrări în fiecare tabel neasociativ; minimum 10 înregistrări în tabelele asociative; maxim 30 de înregistrări în fiecare tabel).

--creez tabelul promotie

CREATE TABLE PROMOTIE (

promotie\_id INT DEFAULT promotie\_seq.NEXTVAL PRIMARY KEY,

nume\_promotie VARCHAR(255),

descriere VARCHAR(255),

discount VARCHAR(5) CHECK (REGEXP\_LIKE(discount, '^\d{2}%$')),

data\_incepere DATE,

data\_incheiere DATE

);

INSERT INTO PROMOTIE (promotie\_id, nume\_promotie, descriere, discount, data\_incepere, data\_incheiere)

VALUES (1, 'Reducere de Iarna', 'Reducere de sezon', '10%', TO\_DATE('2024-11-01', 'YYYY-MM-DD'), TO\_DATE('2024-12-30', 'YYYY-MM-DD'));

INSERT INTO PROMOTIE (promotie\_id, nume\_promotie, descriere, discount, data\_incepere, data\_incheiere)

VALUES (2, 'Black Friday', 'Reduceri de Black Friday', '25%', TO\_DATE('2024-11-29', 'YYYY-MM-DD'), TO\_DATE('2024-11-30', 'YYYY-MM-DD'));

INSERT INTO PROMOTIE (promotie\_id, nume\_promotie, descriere, discount, data\_incepere, data\_incheiere)

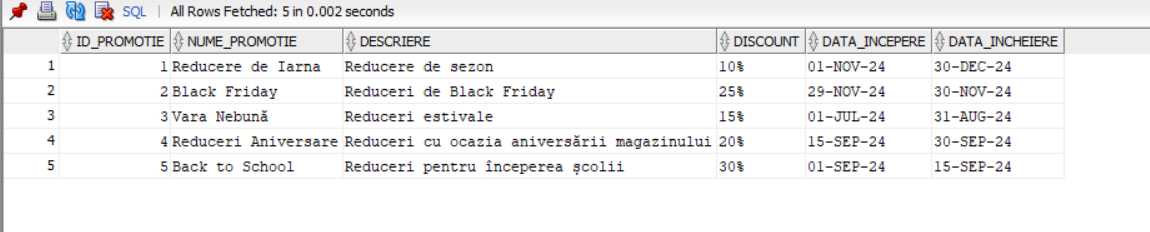
VALUES (3, 'Vara Nebună', 'Reduceri estivale', '15%', TO\_DATE( '2024-07-01', 'YYYY-MM-DD'), TO\_DATE('2024-08-31', 'YYYY-MM-DD'));

INSERT INTO PROMOTIE (promotie\_id, nume\_promotie, descriere, discount, data\_incepere, data\_incheiere)

VALUES (4, 'Reduceri Aniversare', 'Reduceri cu ocazia aniversării magazinului', '20%', TO\_DATE('2024-09-15', 'YYYY-MM-DD'), TO\_DATE('2024-09-30', 'YYYY-MM-DD'));

INSERT INTO PROMOTIE (promotie\_id, nume\_promotie, descriere, discount, data\_incepere, data\_incheiere)

VALUES (5, 'Back to School', 'Reduceri pentru începerea școlii', '30%', TO\_DATE('2024-09-01', 'YYYY-MM-DD'), TO\_DATE('2024-09-15', 'YYYY-MM-DD'));

SELECT \* FROM promotie;

--creez tabelul marime\_categorie

CREATE TABLE MARIME\_CATEGORIE (

marime\_cat\_id INT DEFAULT marime\_cat\_seq.NEXTVAL PRIMARY KEY,

nume VARCHAR(100) UNIQUE

);

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Incaltaminte femei (EU)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Incaltaminte femei (US)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Incaltaminte barbati (EU)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Incaltaminte barbati (US)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Rochii (EU)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Rochii (US)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Tricouri (EU)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Tricouri (US)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Pantaloni (US)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Pantaloni (EU)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Jachete, Paltoane, Cardigane (EU)');

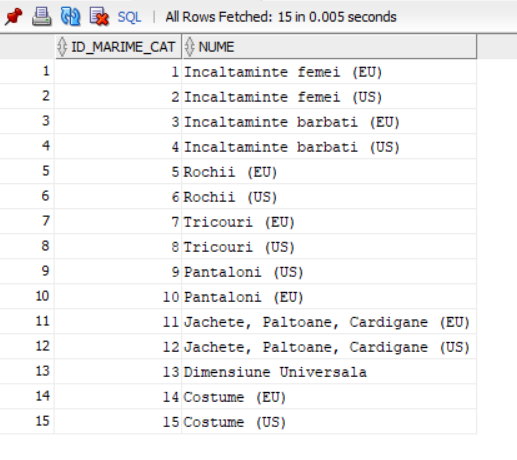
INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Jachete, Paltoane, Cardigane (US)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Dimensiune Universala');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Costume (EU)');

INSERT INTO MARIME\_CATEGORIE (nume) VALUES ('Costume (US)');

SELECT \* FROM marime\_categorie;



--creez tabelul produs\_categorie

CREATE TABLE PRODUS\_CATEGORIE (

produs\_cat\_id INT DEFAULT produs\_cat\_seq.NEXTVAL PRIMARY KEY,

nume\_cat VARCHAR(255) NOT NULL,

imagine\_cat VARCHAR(255) NOT NULL,

parinte\_cat\_id INT,

promotie\_id INT,

CONSTRAINT fk\_parinte\_cat FOREIGN KEY (parinte\_cat\_id) REFERENCES produs\_categorie(produs\_cat\_id),

CONSTRAINT fk\_promotie FOREIGN KEY (promotie\_id) REFERENCES PROMOTIE(promotie\_id)

);

INSERT INTO produs\_categorie(nume\_cat, imagine\_cat, parinte\_cat\_id)

VALUES ('Femei', '/images/femei.png', NULL);

INSERT INTO produs\_categorie(nume\_cat, imagine\_cat, parinte\_cat\_id)

VALUES ('Bărbați', '/images/barbati.png', NULL);

INSERT INTO produs\_categorie(nume\_cat, imagine\_cat, parinte\_cat\_id)

VALUES ('Sport', '/images/haine\_sport\_femei.png', 1);

INSERT INTO produs\_categorie(nume\_cat, imagine\_cat, parinte\_cat\_id, promotie\_id)

VALUES ('Echipament fitness', '/images/echipament\_fitness\_femei.png', 3, 5);

INSERT INTO produs\_categorie(nume\_cat, imagine\_cat, parinte\_cat\_id)

VALUES ('Rochii', '/images/rochie\_femei.png', 1);

INSERT INTO produs\_categorie(nume\_cat, imagine\_cat, parinte\_cat\_id, promotie\_id)

VALUES ('Incaltaminte Femei', '/images/incaltaminte\_femei.png', 1, 4);

INSERT INTO produs\_categorie(nume\_cat, imagine\_cat, parinte\_cat\_id, promotie\_id)

VALUES ('Jachete', '/images/jachete\_barbati.png', 2, 1);

INSERT INTO produs\_categorie(nume\_cat, imagine\_cat, parinte\_cat\_id, promotie\_id)

VALUES ('Costume', '/images/costume\_barbati.png', 2, 2);

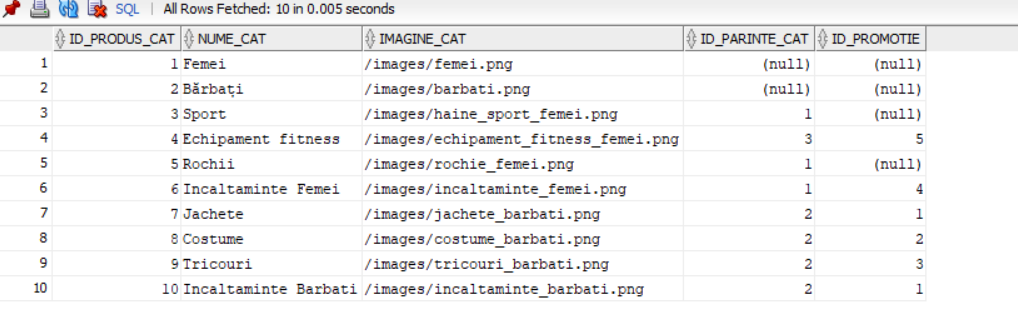
INSERT INTO produs\_categorie(nume\_cat, imagine\_cat, parinte\_cat\_id, promotie\_id)

VALUES ('Tricouri', '/images/tricouri\_barbati.png', 2, 3);

INSERT INTO produs\_categorie(nume\_cat, imagine\_cat, parinte\_cat\_id, promotie\_id)

VALUES ('Incaltaminte Barbati', '/images/incaltaminte\_barbati.png', 2, 1);

SELECT \* FROM produs\_categorie;



--creez tabelul produs\_marime

CREATE TABLE PRODUS\_MARIME(

produs\_cat\_id INT,

marime\_cat\_id INT,

PRIMARY KEY (produs\_cat\_id, marime\_cat\_id),

FOREIGN KEY (produs\_cat\_id) REFERENCES PRODUS\_CATEGORIE(produs\_cat\_id),

FOREIGN KEY (marime\_cat\_id) REFERENCES MARIME\_CATEGORIE(marime\_cat\_id)

);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(4, 7);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(4, 8);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(4, 9);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(4, 10);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(5, 5);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(5, 6);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(6, 1);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(6, 2);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(7, 11);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(7, 12);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(3, 13);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(8, 14);

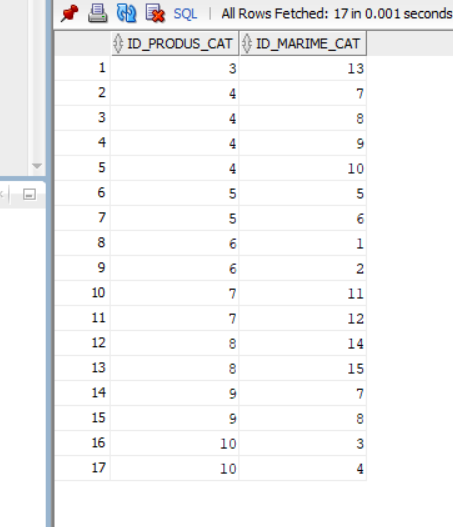
INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(8, 14);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(9, 7);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(9, 8);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(10, 3);

INSERT INTO PRODUS\_MARIME(produs\_cat\_id, marime\_cat\_id) VALUES(10, 4);

SELECT \* FROM produs\_marime;

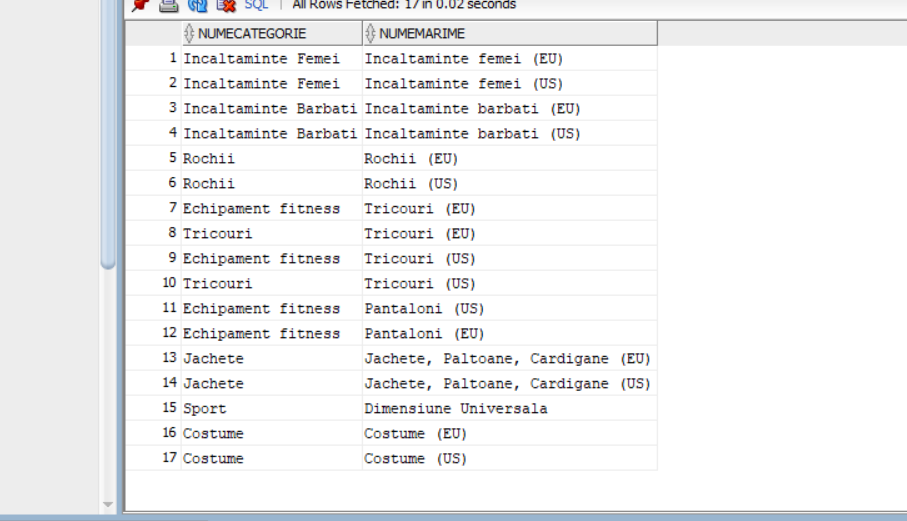
In query-ul de mai jos putem vedea ca hainele au diferite marimi.

SELECT pc.nume\_cat AS NumeCategorie, mc.nume AS NumeMarime

FROM PRODUS\_CATEGORIE pc

JOIN PRODUS\_MARIME pm ON pc.id\_produs\_cat = pm.id\_produs\_cat

JOIN MARIME\_CATEGORIE mc ON pm.id\_marime\_cat = mc.id\_marime\_cat;



-- creez tabelul produs

CREATE TABLE PRODUS (

id\_produs INT DEFAULT produs\_seq.NEXTVAL PRIMARY KEY,

id\_categorie INT,

nume VARCHAR(500) NOT NULL,

instructiuni\_ingrijire VARCHAR(200),

descriere VARCHAR(200),

FOREIGN KEY (id\_categorie) REFERENCES produs\_categorie(produs\_cat\_id)

);

INSERT INTO PRODUS (id\_categorie, nume, instructiuni\_ingrijire, descriere)

VALUES (6, 'Balerini cu vârful ascuțit', 'Ștergere cu o lavetă umedă. A se feri de umiditate și temperaturi extreme.', 'Balerinii cu vârf ascuțit reprezintă o alegere elegantă și sofisticată pentru orice garderobă feminină.');

INSERT INTO PRODUS (id\_categorie, nume, instructiuni\_ingrijire, descriere)

VALUES (10, 'Bocanci de iarnă pentru bărbați', 'Ștergere cu o lavetă umedă. A se feri de umiditate și temperaturi extreme.', 'Bocanci rezistenți și călduroși, potriviți pentru iarna grea. Design practic și materiale durabile.');

INSERT INTO PRODUS (id\_categorie, nume, instructiuni\_ingrijire, descriere)

VALUES (5, 'Rochie de vară cu imprimeu floral', 'Se recomandă spălarea manuală în apă rece. A se usca la umbră.', 'Rochie lejeră perfectă pentru zilele călduroase de vară. Material confortabil și ușor de purtat.');

INSERT INTO PRODUS (id\_categorie, nume, instructiuni\_ingrijire, descriere)

VALUES (4, 'Pantaloni sport', 'Spălare la mașină cu apă rece. A se călca la temperatură scăzută.', 'Pantalonii sport sunt ideali pentru activitățile fizice, oferind confort și libertate de mișcare.');

INSERT INTO PRODUS (id\_categorie, nume, instructiuni\_ingrijire, descriere)

VALUES (3, 'Benzi de rezistență', 'Poți folosi o lavetă umedă sau șervețele de curățare pentru a șterge suprafața benzii.', 'Benzi de rezistență sunt accesorii versatile pentru antrenamentul fizic, potrivite pentru o varietate de exerciții și activități sportive.');

INSERT INTO PRODUS (id\_categorie, nume, instructiuni\_ingrijire, descriere)

VALUES (9, 'Tricou casual pentru bărbați', 'Spălare la mașină cu apă rece. A se usca la umbră.', 'Tricou confortabil și versatil, ideal pentru outfiturile de zi cu zi. Design clasic și materiale rezistente.');

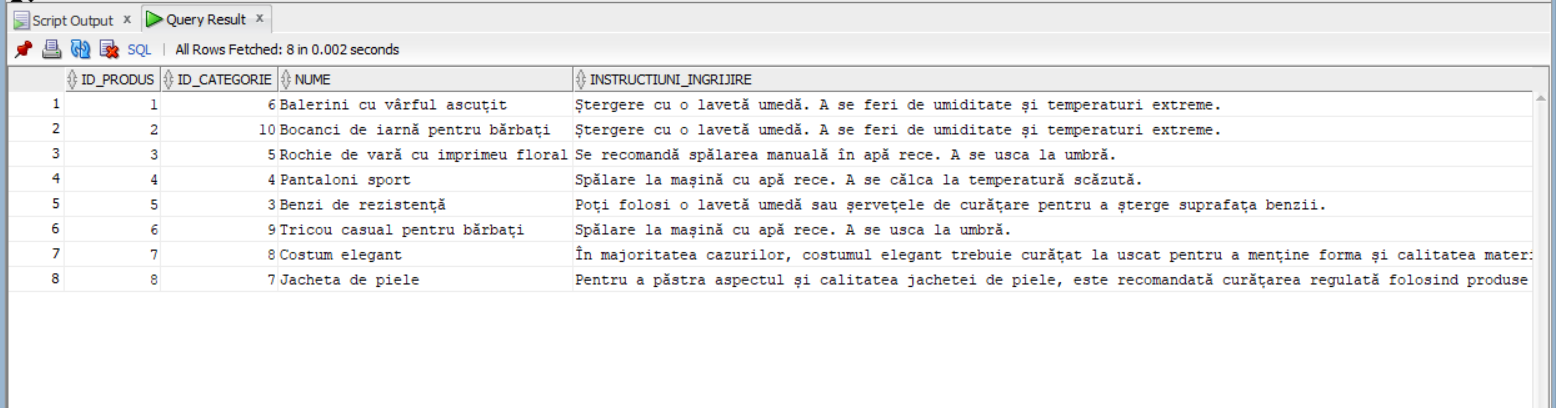
INSERT INTO PRODUS (id\_categorie, nume, instructiuni\_ingrijire, descriere)

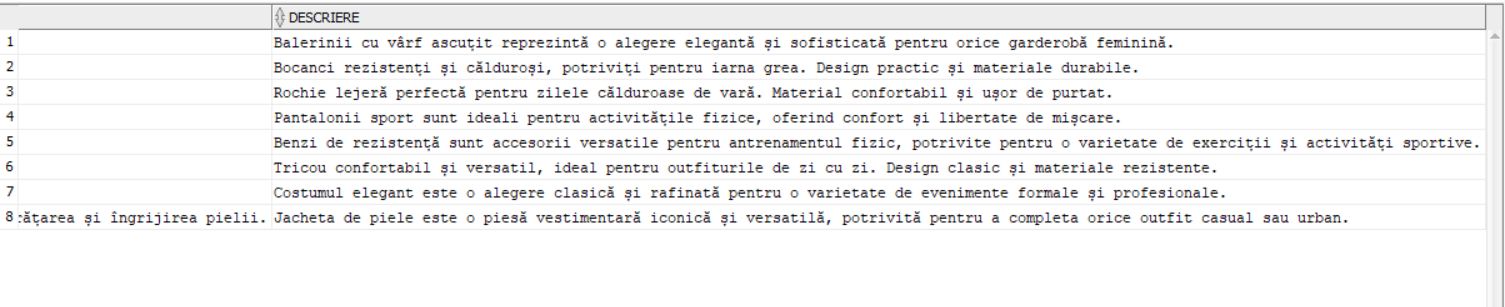
VALUES (8, 'Costum elegant', ' În majoritatea cazurilor, costumul elegant trebuie curățat la uscat pentru a menține forma și calitatea materialului.', 'Costumul elegant este o alegere clasică și rafinată pentru o varietate de evenimente formale și profesionale.');

INSERT INTO PRODUS (id\_categorie, nume, instructiuni\_ingrijire, descriere)

VALUES (7, 'Jacheta de piele', 'Pentru a păstra aspectul și calitatea jachetei de piele, este recomandată curățarea regulată folosind produse specializate pentru curățarea și îngrijirea pielii.', 'Jacheta de piele este o piesă vestimentară iconică și versatilă, potrivită pentru a completa orice outfit casual sau urban.');

SELECT \* FROM PRODUS;





--creez tabelul optiune\_marime

CREATE TABLE OPTIUNE\_MARIME (

id\_optiune\_marime INT DEFAULT optiune\_marime\_seq.NEXTVAL PRIMARY KEY,

id\_categorie\_marime INT,

marime\_nume VARCHAR(50),

ordonare INT,

CONSTRAINT fk\_categorie\_marime FOREIGN KEY (id\_categorie\_marime) REFERENCES MARIME\_CATEGORIE(id\_marime\_cat)

);

--Tricouri

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (7, '40', 1);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (7, '44', 3)

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (8, '30', 1);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (8, '34', 3);

--Rochii

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (5, '32', 1);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (5, '36', 2);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (6, '2', 1);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (6, '6', 2);

-- Pentru categorie "Incaltaminte barbati"

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (3, '38', 1);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (3, '40', 2);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (4, '8', 1);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (4, '10', 2);

-- Pentru categorie "Incaltaminte femei"

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (1, '36', 1);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (1, '38', 2);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (2, '6', 1);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (2, '8', 2);

--Pentru pantaloni

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (9, 'S', 1);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (9, 'M', 2);

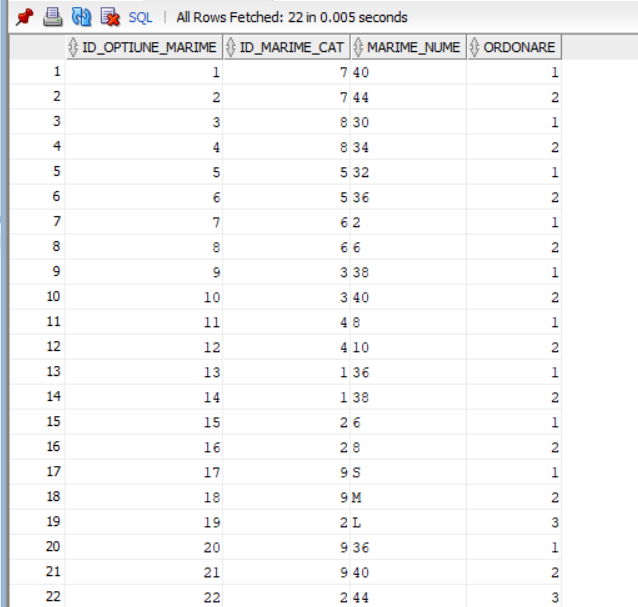
INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (2, 'L', 1);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (9, '36', 1);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (9, '40', 2);

INSERT INTO OPTIUNE\_MARIME (id\_categorie\_marime, marime\_nume, ordonare) VALUES (2, '44', 1);

SELECT \* FROM optiune\_marime;



--creez tabelul tara

CREATE TABLE tara(

id\_tara INT DEFAULT tara\_seq.NEXTVAL PRIMARY KEY,

nume\_tara VARCHAR(50) UNIQUE

);

INSERT INTO tara (nume\_tara) VALUES ('România');

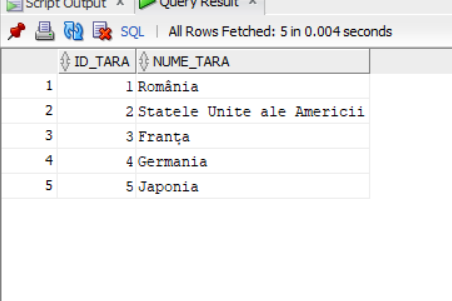
INSERT INTO tara (nume\_tara) VALUES ('Statele Unite ale Americii');

INSERT INTO tara (nume\_tara) VALUES ('Franța');

INSERT INTO tara (nume\_tara) VALUES ('Germania');

INSERT INTO tara (nume\_tara) VALUES ('Japonia');

SELECT \* FROM tara;



-- creez tabelul adresa

CREATE TABLE adresa (

id\_adresa INT DEFAULT adresa\_seq.NEXTVAL PRIMARY KEY,

cod\_postal VARCHAR(10) NOT NULL,

oras VARCHAR(50) NOT NULL,

adresa\_linie1 VARCHAR(100) NOT NULL,

adresa\_linie2 VARCHAR(255),

id\_tara INT,

CONSTRAINT fk\_tara FOREIGN KEY (id\_tara) REFERENCES tara(id\_tara)

);

INSERT INTO adresa (cod\_postal, oras, adresa\_linie1, adresa\_linie2, id\_tara)

VALUES ('061622', 'București', 'Sector 6, strada Orhideea', 'bloc 24, scara C', 1);

INSERT INTO adresa (cod\_postal, oras, adresa\_linie1, adresa\_linie2, id\_tara)

VALUES ('12601', 'New York', '5 Taylor Street Poughkeepsie', NULL, 2);

INSERT INTO adresa (cod\_postal, oras, adresa\_linie1, adresa\_linie2, id\_tara)

VALUES ('75010', 'Paris', '174 Quai de Jemmapes', NULL, 3);

INSERT INTO adresa (cod\_postal, oras, adresa\_linie1, adresa\_linie2, id\_tara)

VALUES ('13629', 'Berlin', '95 Nonnendammallee', 'Spandau', 4);

INSERT INTO adresa (cod\_postal, oras, adresa\_linie1, adresa\_linie2, id\_tara)

VALUES ('179-0072', 'Tokyo', 'Tagara High School, 1 Toshimaen-dori', NULL, 5);

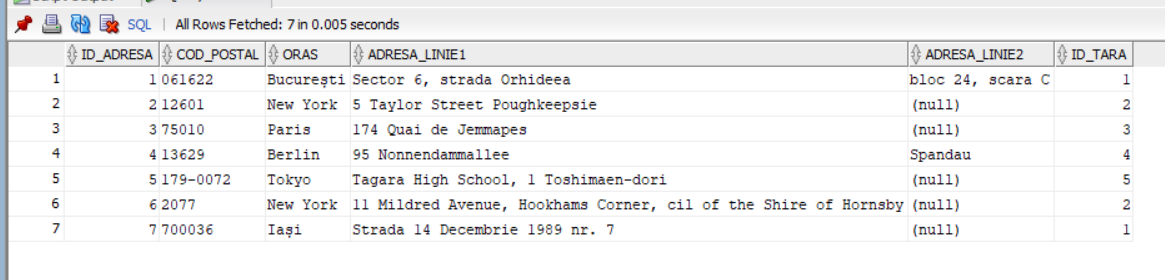
INSERT INTO adresa (cod\_postal, oras, adresa\_linie1, adresa\_linie2, id\_tara)

VALUES ('2077', 'New York', '11 Mildred Avenue, Hookhams Corner, cil of the Shire of Hornsby', NULL, 2);

INSERT INTO adresa (cod\_postal, oras, adresa\_linie1, adresa\_linie2, id\_tara)

VALUES ('700036', 'Iași', 'Strada 14 Decembrie 1989 nr. 7', NULL, 1);

SELECT \* FROM adresa;



--creez tabelul utilizator

CREATE TABLE utilizator (

id\_user INT DEFAULT user\_seq.NEXTVAL PRIMARY KEY,

adresa\_email VARCHAR(255) UNIQUE,

nr\_telefon VARCHAR(20) UNIQUE,

parola\_hash VARCHAR(128) NOT NULL,

CONSTRAINT check\_nr\_telefon\_format CHECK (

REGEXP\_LIKE(nr\_telefon, '^\+?[0-9]{1,3}[-. ]?[0-9]{1,4}([- .]?[0-9]{1,4}){1,4}$')

)

);

INSERT INTO utilizator (adresa\_email, nr\_telefon, parola\_hash)

VALUES ('john.doe@example.com', '+40700000000', '5f4dcc3b5aa765d61d8327deb882cf99');

INSERT INTO utilizator (adresa\_email, nr\_telefon, parola\_hash)

VALUES ('jane\_doe123@example.com', '+1 202-555-0165', '098f6bcd4621d373cade4e832627b4f6');

INSERT INTO utilizator (adresa\_email, nr\_telefon, parola\_hash)

VALUES ('maria@gmail.com', '+44 20 7123 4567', 'c4ca4238a0b923820dcc509a6f75849b');

INSERT INTO utilizator (adresa\_email, nr\_telefon, parola\_hash)

VALUES ('doina123@gmail.com', '+33 1 23 45 67 89', 'eccbc87e4b5ce2fe28308fd9f2a7baf3');

INSERT INTO utilizator (adresa\_email, nr\_telefon, parola\_hash)

VALUES ('ELena.234@yahoo.com', '+49 30 12345678', 'a87ff679a2f3e71d9181a67b7542122c');

INSERT INTO utilizator (adresa\_email, nr\_telefon, parola\_hash)

VALUES ('florina.alex@outlook.com', '+7 495 123-45-67', 'e4da3b7fbbce2345d7772b0674a318d5');

INSERT INTO utilizator (adresa\_email, nr\_telefon, parola\_hash)

VALUES ('mihai.alex@yahoo.com', '+52 55 1234 5678', '1679091c5a880faf6fb5e6087eb1b2dc');

INSERT INTO utilizator (adresa\_email, nr\_telefon, parola\_hash)

VALUES ('andrei234@outlook.com', '+61-2-12345678', '8f14e45fceea167a5a36dedd4bea2543');

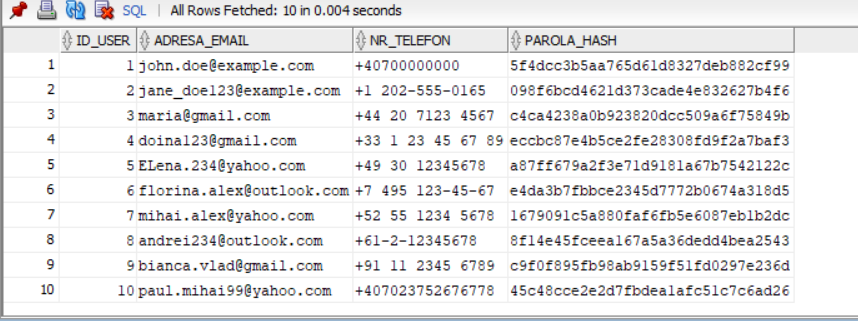
INSERT INTO utilizator (adresa\_email, nr\_telefon, parola\_hash)

VALUES ('bianca.vlad@gmail.com', '+91 11 2345 6789', 'c9f0f895fb98ab9159f51fd0297e236d');

INSERT INTO utilizator (adresa\_email, nr\_telefon, parola\_hash)

VALUES ('paul.mihai99@yahoo.com', '+407023752676778', '45c48cce2e2d7fbdea1afc51c7c6ad26');

SELECT \* FROM utilizator;



--creez tabelul adresa\_user

CREATE TABLE ADRESA\_USER (

id\_user INT,

id\_adresa INT,

e\_default CHAR(1),

PRIMARY KEY (id\_user, id\_adresa),

CONSTRAINT fk\_user FOREIGN KEY (id\_user) REFERENCES utilizator(id\_user),

CONSTRAINT fk\_adresa FOREIGN KEY (id\_adresa) REFERENCES adresa(id\_adresa),

CONSTRAINT chk\_default CHECK (e\_default IN ('Y', 'N'))

);

INSERT INTO ADRESA\_USER(id\_user, id\_adresa, e\_default)

SELECT u.id\_user, a.id\_adresa, 'N'

FROM utilizator u

JOIN adresa a ON u.id\_user = a.id\_adresa;

INSERT INTO ADRESA\_USER VALUES (8, 3, 'N');

INSERT INTO ADRESA\_USER VALUES (9, 5, 'N');

INSERT INTO ADRESA\_USER VALUES (10, 1, 'N');

UPDATE ADRESA\_USER

SET e\_default = 'Y'

WHERE id\_user = 5;

UPDATE ADRESA\_USER

SET e\_default = 'Y'

WHERE id\_user = 6;

UPDATE ADRESA\_USER

SET e\_default = 'Y'

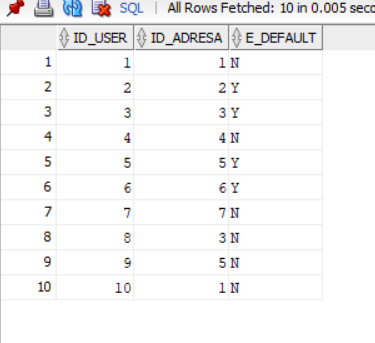
WHERE id\_user = 2;

UPDATE ADRESA\_USER

SET e\_default = 'Y'

WHERE id\_user = 3;

SELECT \* FROM ADRESA\_USER;



--creez tabelul recenzie

CREATE TABLE RECENZIE (

id\_recenzie INT DEFAULT recenzie\_seq.NEXTVAL PRIMARY KEY,

id\_user INT,

rating INT CHECK(rating BETWEEN 1 AND 5),

comentariu VARCHAR2(255),

CONSTRAINT fk\_id\_user FOREIGN KEY (id\_user) REFERENCES utilizator(id\_user)

);

INSERT INTO RECENZIE (rating, comentariu, id\_user)

VALUES (5, 'Produsul este excelent, sunt foarte mulțumit de calitate și design.', 2);

INSERT INTO RECENZIE (rating, comentariu, id\_user)

VALUES (4, 'Foarte mulțumită de produs, livrare rapidă și calitate excelentă.', 3);

INSERT INTO RECENZIE (rating, comentariu, id\_user)

VALUES (3, 'Produsul a fost ok, dar mă așteptam la ceva mai mult în acest preț.', 10);

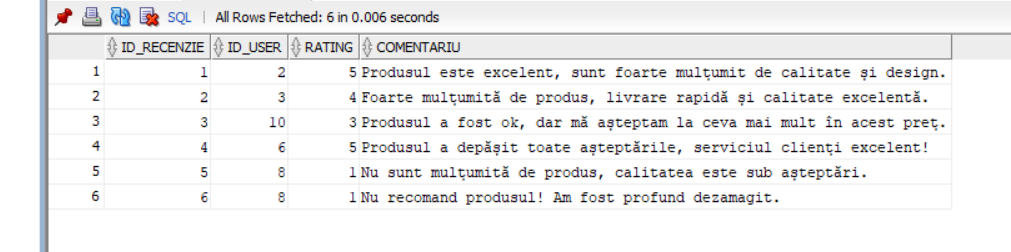
INSERT INTO RECENZIE (rating, comentariu, id\_user)

VALUES (5, 'Produsul a depășit toate așteptările, serviciul clienți excelent!', 6);

INSERT INTO RECENZIE (rating, comentariu, id\_user)

VALUES (1, 'Nu sunt mulțumită de produs, calitatea este sub așteptări.', 8);

SELECT \* FROM recenzie;



--creez tabelul culoare

CREATE TABLE CULOARE (

id\_culoare INT PRIMARY KEY,

nume\_culoare VARCHAR(50) UNIQUE

);

INSERT INTO CULOARE (id\_culoare, nume\_culoare) VALUES (1, 'Negru');

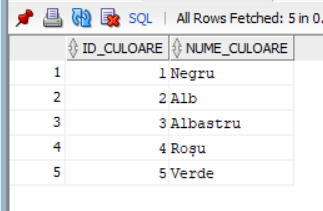
INSERT INTO CULOARE (id\_culoare, nume\_culoare) VALUES (2, 'Alb');

INSERT INTO CULOARE (id\_culoare, nume\_culoare) VALUES (3, 'Albastru');

INSERT INTO CULOARE (id\_culoare, nume\_culoare) VALUES (4, 'Roșu');

INSERT INTO CULOARE (id\_culoare, nume\_culoare) VALUES (5, 'Verde');

SELECT \* FROM CULOARE;



-- creez tabelul produs\_item

CREATE TABLE PRODUS\_ITEM (

id\_produs\_item INT DEFAULT produs\_item\_seq.NEXTVAL PRIMARY KEY,

id\_produs INT,

cantitate\_in\_stoc INT CHECK (cantitate\_in\_stoc >= 0),

pret\_original DECIMAL(10, 2) NOT NULL CHECK (pret\_original >= 0),

pret\_dupa\_reducere DECIMAL(10, 2) CHECK (pret\_dupa\_reducere >= 0),

id\_recenzie INT UNIQUE,

id\_culoare INT NOT NULL,

CONSTRAINT fk\_produs FOREIGN KEY (id\_produs) REFERENCES PRODUS(id\_produs),

CONSTRAINT fk\_recenzie FOREIGN KEY (id\_recenzie) REFERENCES RECENZIE(id\_recenzie),

CONSTRAINT fk\_culoare FOREIGN KEY (id\_culoare) REFERENCES CULOARE(id\_culoare)

);

INSERT INTO PRODUS\_ITEM (id\_produs, cantitate\_in\_stoc, pret\_original, id\_recenzie, id\_culoare)

--Balerini cu varf ascutit, 50 in stoc, pret original: 99.99, pret redus: 79.99, recenzia cu nr 2

VALUES (1, 50, 99.99, 2, 1);

INSERT INTO PRODUS\_ITEM (id\_produs, cantitate\_in\_stoc, pret\_original, id\_culoare)

-- Bocanci de iarna, 30 in stoc, nu are recenzie, albastrii

VALUES (2, 30, 129.99, 3);

INSERT INTO PRODUS\_ITEM (id\_produs, cantitate\_in\_stoc, pret\_original, id\_recenzie, id\_culoare)

--Rochie cu imprimeu floral, 20 in stoc, recenzia cu nr 3

VALUES (3, 20, 149.99, 3, 2);

INSERT INTO PRODUS\_ITEM (id\_produs, cantitate\_in\_stoc, pret\_original, id\_recenzie, id\_culoare)

--Pantaloni sport, 20 in stoc, recenzia cu nr 1

VALUES (4, 40, 59.99, 1, 4);

INSERT INTO PRODUS\_ITEM (id\_produs, cantitate\_in\_stoc, pret\_original, id\_recenzie, id\_culoare)

--Tricou casual, 40 in stoc, verde

VALUES (6, 40, 99.99, 5, 5);

INSERT INTO PRODUS\_ITEM (id\_produs, cantitate\_in\_stoc, pret\_original, id\_recenzie, id\_culoare)

--Tricou casual, 20 in stoc, rosu

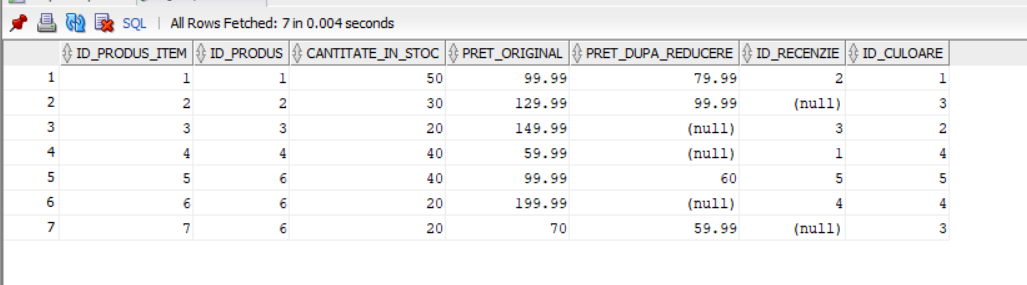
VALUES (6, 20, 199.99, 4, 4);

INSERT INTO PRODUS\_ITEM (id\_produs, cantitate\_in\_stoc, pret\_original, id\_culoare)

--Tricou casual, 20 in stoc, albastru

VALUES (6, 20, 70, 3);

SELECT \* FROM produs\_item;



Fac update la tabel ca sa imi returneze pret-dupa\_reducere unde este cazul

UPDATE PRODUS\_ITEM pi

SET pret\_dupa\_reducere = pret\_original - (

pret\_original \* (

SELECT CAST(SUBSTR(MAX(p.discount), 1, LENGTH(MAX(p.discount)) - 1) AS DECIMAL(4, 2)) / 100

FROM PROMOTIE p

JOIN PRODUS pr ON pi.id\_produs = pr.id\_produs

JOIN PRODUS\_CATEGORIE pc ON pr.id\_categorie = pc.id\_produs\_cat

WHERE pc.id\_promotie = p.id\_promotie

)

);

--creez tabela imagine\_produs

CREATE TABLE IMAGINE\_PRODUS (

id\_imagine INT DEFAULT imag\_prod\_seq.NEXTVAL PRIMARY KEY,

id\_produs\_item INT,

nume\_fisier VARCHAR(255),

FOREIGN KEY (id\_produs\_item) REFERENCES PRODUS\_ITEM(id\_produs\_item)

);

INSERT INTO IMAGINE\_PRODUS (id\_produs\_item, nume\_fisier)

VALUES (1, '/images/balerini\_cu\_varf\_ascutit.jpg');

INSERT INTO IMAGINE\_PRODUS (id\_produs\_item, nume\_fisier)

VALUES (2, '/images/bocanci\_barbati.jpg');

INSERT INTO IMAGINE\_PRODUS (id\_produs\_item, nume\_fisier)

VALUES (3, '/images/rochie\_florala.jpg');

INSERT INTO IMAGINE\_PRODUS (id\_produs\_item, nume\_fisier)

VALUES (4, '/images/pantaloni\_sport.jpg');

INSERT INTO IMAGINE\_PRODUS (id\_produs\_item, nume\_fisier)

VALUES (6, '/images/tricou\_casual\_rosu.jpg');

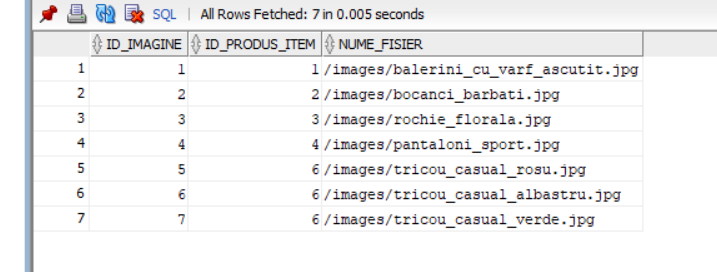
INSERT INTO IMAGINE\_PRODUS (id\_produs\_item, nume\_fisier)

VALUES (6, '/images/tricou\_casual\_albastru.jpg');

INSERT INTO IMAGINE\_PRODUS (id\_produs\_item, nume\_fisier)

VALUES (6, '/images/tricou\_casual\_verde.jpg');

SELECT \* FROM IMAGINE\_PRODUS;



--creez tabelul depozit

CREATE TABLE DEPOZIT (

id\_depozit INT DEFAULT depozit\_seq.NEXTVAL PRIMARY KEY,

nume\_depozit VARCHAR(2),

locatie VARCHAR(50) UNIQUE NOT NULL,

capacitate\_max INT,

CONSTRAINT nume\_depozit\_check CHECK (REGEXP\_LIKE(nume\_depozit, '^[A-Z]{1}[0-9]{1}$')),

CONSTRAINT capacitate\_check CHECK (capacitate\_max > 0 AND capacitate\_max <= 5000)

);

INSERT INTO DEPOZIT (nume\_depozit, locatie, capacitate\_max)

VALUES ('T1', 'Bucuresti', 7000);

INSERT INTO DEPOZIT (nume\_depozit, locatie, capacitate\_max)

VALUES ('H2', 'New York', 4000);

INSERT INTO DEPOZIT (nume\_depozit, locatie, capacitate\_max)

VALUES ('F4', 'Paris', 2500);

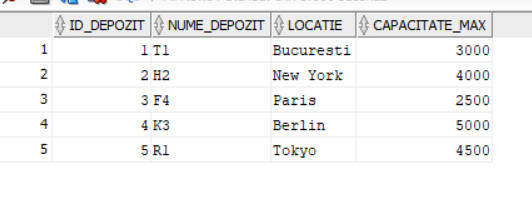
INSERT INTO DEPOZIT (nume\_depozit, locatie, capacitate\_max)

VALUES ('K3', 'Berlin', 5000);

INSERT INTO DEPOZIT (nume\_depozit, locatie, capacitate\_max)

VALUES ('R1', 'Tokyo', 4500);

SELECT \* FROM DEPOZIT;



--creez tabelul DEPOZIT\_PRODUS

CREATE TABLE DEPOZIT\_PRODUS\_ITEM(

id\_produs\_item INT,

id\_depozit INT,

PRIMARY KEY (id\_produs\_item, id\_depozit),

CONSTRAINT fk\_id\_produs\_item FOREIGN KEY (id\_produs\_item) REFERENCES PRODUS\_ITEM(id\_produs\_item),

CONSTRAINT fk\_depozit FOREIGN KEY (id\_depozit) REFERENCES DEPOZIT(id\_depozit)

);

INSERT INTO DEPOZIT\_PRODUS\_ITEM (id\_produs\_item, id\_depozit)

SELECT p.id\_produs\_item, d.id\_depozit

FROM PRODUS\_ITEM p

CROSS JOIN DEPOZIT d

WHERE d.id\_depozit = 1

AND p.id\_produs\_item IN (1, 2, 3);

INSERT INTO DEPOZIT\_PRODUS\_ITEM (id\_produs\_item, id\_depozit)

SELECT p.id\_produs\_item, d.id\_depozit

FROM PRODUS\_ITEM p

CROSS JOIN DEPOZIT d

WHERE d.id\_depozit = 2

AND p.id\_produs\_item IN (1, 2, 3, 4, 5, 6, 7);

INSERT INTO DEPOZIT\_PRODUS\_ITEM (id\_produs\_item, id\_depozit)

SELECT p.id\_produs\_item, d.id\_depozit

FROM PRODUS\_ITEM p

CROSS JOIN DEPOZIT d

WHERE d.id\_depozit = 3

AND p.id\_produs\_item IN (1, 4, 5, 6);

INSERT INTO DEPOZIT\_PRODUS\_ITEM (id\_produs\_item, id\_depozit)

SELECT p.id\_produs\_item, d.id\_depozit

FROM PRODUS\_ITEM p

CROSS JOIN DEPOZIT d

WHERE d.id\_depozit = 4

AND p.id\_produs\_item IN (1, 7, 8);

INSERT INTO DEPOZIT\_PRODUS\_ITEM (id\_produs\_item, id\_depozit)

SELECT p.id\_produs\_item, d.id\_depozit

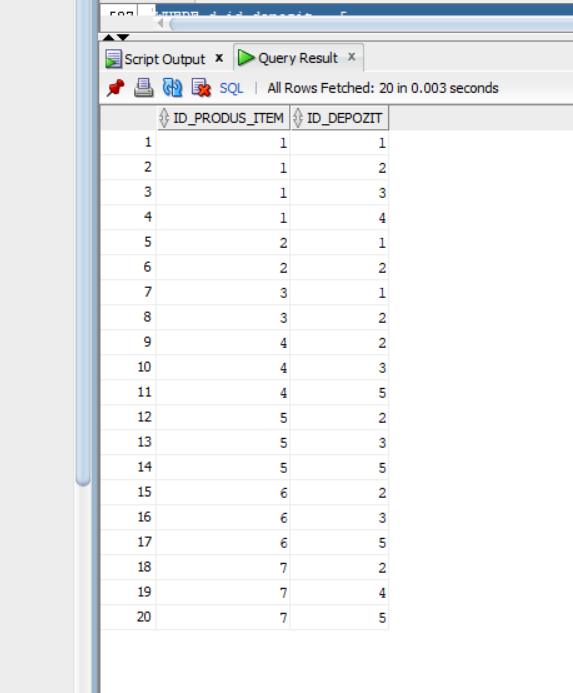
FROM PRODUS\_ITEM p

CROSS JOIN DEPOZIT d

WHERE d.id\_depozit = 5

AND p.id\_produs\_item IN (4, 5, 6, 7);

SELECT \* FROM DEPOZIT\_PRODUS\_ITEM;



--creez tabelul status\_comanda

CREATE TABLE status\_comanda (

id\_status INT DEFAULT status\_comanda\_seq.NEXTVAL PRIMARY KEY,

status VARCHAR(50)

);

INSERT INTO status\_comanda (id\_status, status)

VALUES (1, 'În așteptare');

INSERT INTO status\_comanda (id\_status, status)

VALUES (2, 'În procesare');

INSERT INTO status\_comanda (id\_status, status)

VALUES (3, 'Expediată');

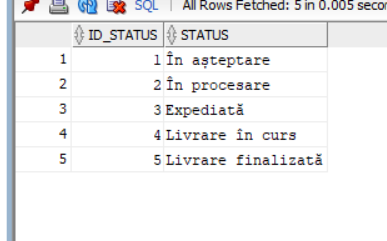
INSERT INTO status\_comanda (id\_status, status)

VALUES (4, 'Livrare în curs');

INSERT INTO status\_comanda (id\_status, status)

VALUES (5, 'Livrare finalizată');

SELECT \* FROM status\_comanda;



--creez tabelul comanda

CREATE TABLE comanda (

id\_comanda INT DEFAULT comanda\_seq.NEXTVAL PRIMARY KEY,

data\_comanda TIMESTAMP DEFAULT (SYSTIMESTAMP + INTERVAL '7' DAY),

total\_plata DECIMAL(10, 2) DEFAULT 0.00 NOT NULL,

metoda\_plata VARCHAR(50) NOT NULL,

status\_comanda INT,

CONSTRAINT fk\_status FOREIGN KEY (status\_comanda) REFERENCES status\_comanda(id\_status),

CONSTRAINT total\_plata\_check CHECK (total\_plata >= 0)

);

INSERT INTO comanda (data\_comanda, metoda\_plata, status\_comanda)

VALUES (TIMESTAMP '2024-05-20 10:00:00', 'Plata cu cardul', 1);

INSERT INTO comanda (metoda\_plata, status\_comanda)

VALUES ('Plata curier', 2);

INSERT INTO comanda (data\_comanda, metoda\_plata, status\_comanda)

VALUES (TIMESTAMP '2024-10-17 17:00:00', 'Plata cu cardul', 3);

INSERT INTO comanda (metoda\_plata, status\_comanda)

VALUES ('Plata curier', 1);

INSERT INTO comanda (data\_comanda, metoda\_plata, status\_comanda)

VALUES (TIMESTAMP '2024-9-29 18:30:00', 'Plata cu cardul', 4);

INSERT INTO comanda (data\_comanda, metoda\_plata, status\_comanda)

VALUES (TIMESTAMP '2024-1-10 10:45:00', 'Plata curier', 3);

UPDATE comanda

SET total\_plata = (

SELECT

SUM(

CASE

WHEN PI.pret\_dupa\_reducere IS NOT NULL THEN PA.cantitate\_comandata \* PI.pret\_dupa\_reducere

ELSE PA.cantitate\_comandata \* PI.pret\_original

END

) AS total\_plata

FROM

PRODUS\_ALES PA

JOIN

PRODUS\_ITEM PI ON PA.id\_produs\_item = PI.id\_produs\_item

WHERE

PA.id\_produs\_ales = comanda.id\_comanda

GROUP BY

PA.id\_produs\_ales

)

WHERE

id\_comanda IN (1, 2, 3, 4, 5, 6);



--creez tabelul PRODUS\_COMANDAT

CREATE TABLE PRODUS\_COMANDAT(

id\_produs\_item INT,

id\_comanda INT,

id\_optiune\_marime INT,

cantitate\_comandata INT,

PRIMARY KEY(id\_produs\_item, id\_comanda, id\_optiune\_marime),

FOREIGN KEY (id\_produs\_item) REFERENCES PRODUS\_ITEM(id\_produs\_item),

FOREIGN KEY (id\_comanda) REFERENCES COMANDA(id\_comanda),

FOREIGN KEY (id\_optiune\_marime) REFERENCES OPTIUNE\_MARIME(id\_optiune\_marime)

);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--cumpara balerini femei

VALUES(1, 14, 1, 3);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--cumpara bocanci barbati

VALUES(2, 9, 1, 5);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--rochie de vara

VALUES(3, 7, 1, 4);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--tricou barbati

VALUES(6, 7, 2, 1);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--tricou barbati

VALUES(5, 7, 3, 2);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--tricou barbati

VALUES(6, 7, 4, 7);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--rochie de vara

VALUES(3, 7, 4, 4);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--tricou barbati

VALUES(6, 7, 5, 1);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--cumpara balerini femei

VALUES(1, 14, 5, 1);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--cumpara bocanci barbati

VALUES(2, 9, 5, 3);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--cumpara balerini femei

VALUES(1, 14, 6, 1);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--cumpara balerini femei

VALUES(1, 14, 7, 10);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--cumpara bocanci barbati

VALUES(2, 9, 8, 1);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--tricou barbati

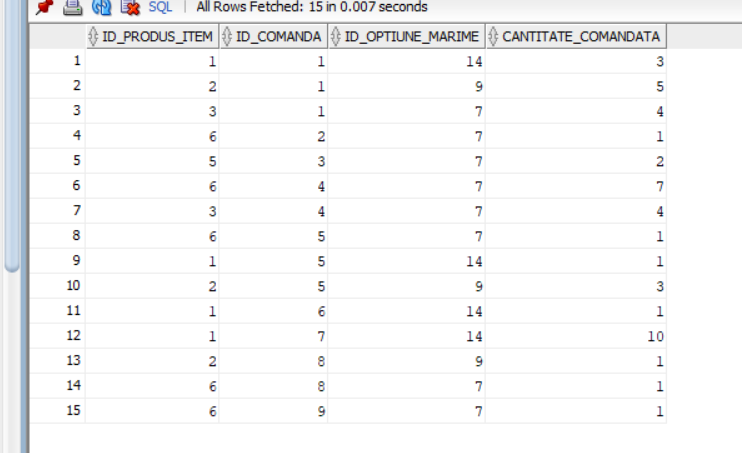
VALUES(6, 7, 8, 1);

INSERT INTO PRODUS\_COMANDAT(id\_produs\_item, id\_optiune\_marime, id\_comanda, cantitate\_comandata)

--tricou barbati

VALUES(6, 7, 9, 1);

SELECT \* FROM PRODUS\_COMANDAT;



--creez tabelul cumpara

CREATE TABLE CUMPARA(

id\_user INT,

id\_comanda INT,

id\_depozit INT,

PRIMARY KEY(id\_user, id\_comanda, id\_depozit),

FOREIGN KEY (id\_user) REFERENCES UTILIZATOR(id\_user),

FOREIGN KEY (id\_comanda) REFERENCES COMANDA(id\_comanda),

FOREIGN KEY (id\_depozit) REFERENCES DEPOZIT(id\_depozit)

);

INSERT INTO CUMPARA(id\_user, id\_comanda, id\_depozit) VALUES(1, 1, 1);

INSERT INTO CUMPARA(id\_user, id\_comanda, id\_depozit) VALUES(1, 2, 1);

INSERT INTO CUMPARA(id\_user, id\_comanda, id\_depozit) VALUES(2, 3, 2);

INSERT INTO CUMPARA(id\_user, id\_comanda, id\_depozit) VALUES(3, 2, 3);

INSERT INTO CUMPARA(id\_user, id\_comanda, id\_depozit) VALUES(4, 1, 4);

INSERT INTO CUMPARA(id\_user, id\_comanda, id\_depozit) VALUES(5, 4, 5);

INSERT INTO CUMPARA(id\_user, id\_comanda, id\_depozit) VALUES(6, 5, 5);

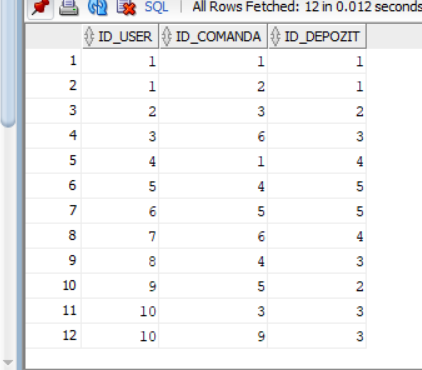
INSERT INTO CUMPARA(id\_user, id\_comanda, id\_depozit) VALUES(7, 2, 4);

INSERT INTO CUMPARA(id\_user, id\_comanda, id\_depozit) VALUES(8, 4, 3);

INSERT INTO CUMPARA(id\_user, id\_comanda, id\_depozit) VALUES(9, 5, 2);

INSERT INTO CUMPARA(id\_user, id\_comanda, id\_depozit) VALUES(10, 3, 3);

SELECT \* FROM CUMPARA;



# Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente:

subcereri sincronizate în care intervin cel puțin 3 tabele

--interogarea are ca scop obtinerea detaliilor despre utilizatorii care au facut achizitii in care suma totala platita(total\_plata) este mai mare decat media de plata total pt toate comenzile

--de asemenea, ne arata si numele produselor care au fost achizitionate si care au o valoare mai mare decat media produselor (tricou casual o sa fie deoarece e singurul care indeplineste (suma produselor e 729 impartim la 7 ne da 103))

SELECT u.adresa\_email AS utilizator, c.total\_plata AS plata, LISTAGG(p.nume, ', ') WITHIN GROUP (ORDER BY p.nume) AS produse

FROM utilizator u

JOIN cumpara cu ON u.id\_user = cu.id\_user

JOIN comanda c ON cu.id\_comanda = c.id\_comanda

JOIN produs\_comandat pc ON c.id\_comanda = pc.id\_comanda

JOIN produs\_item pi ON pc.id\_produs\_item = pi.id\_produs\_item

JOIN produs p ON pi.id\_produs = p.id\_produs

WHERE

c.total\_plata > (

SELECT AVG(total\_plata)

FROM comanda)

AND (pc.cantitate\_comandata \* pi.pret\_dupa\_reducere) > (

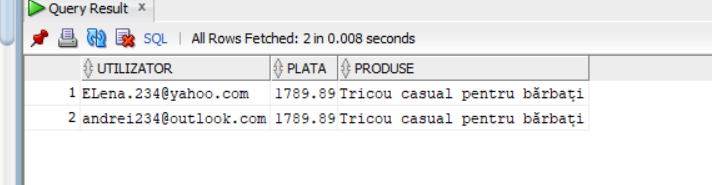
SELECT AVG(pc\_inner.cantitate\_comandata \* pi\_inner.pret\_dupa\_reducere)

FROM produs\_comandat pc\_inner

JOIN produs\_item pi\_inner ON pc\_inner.id\_produs\_item = pi\_inner.id\_produs\_item)

GROUP BY u.adresa\_email, c.total\_plata

ORDER BY u.adresa\_email;



subcereri nesincronizate în clauza FROM

--fiind cerere nesincronizata inseamna ca in subquery nu fac nicio referinta la tabelul din query

--aceasta subcerere afiseaza id-ul depozitelor, numele depozitelor si castigul realizat de acestea (adun c.total\_plata de la comenzi ca sa vad fiecare depozit cati bani a facut)

SELECT

d.id\_depozit, d.nume\_depozit, sub.castig

FROM

DEPOZIT d,

(

SELECT cp.id\_depozit, SUM(c.total\_plata) AS castig

FROM COMANDA c

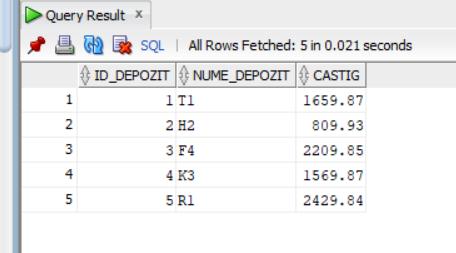
JOIN CUMPARA cp ON c.id\_comanda = cp.id\_comanda

GROUP BY cp.id\_depozit

) sub

WHERE

d.id\_depozit = sub.id\_depozit;



grupări de date, funcții grup, filtrare la nivel de grupuri cu subcereri nesincronizate (în clauza de HAVING) în care intervin cel puțin 3 tabele (in cadrul aceleiași cereri)

SELECT

u.adresa\_email AS Utilizator,

a.adresa\_linie1 AS Adresa\_Utilizator,

c.id\_comanda AS ID\_Comanda,

d.nume\_depozit AS Depozit\_Plecare,

c.total\_plata AS Total\_Plata

FROM

utilizator u

JOIN

adresa\_user au ON u.id\_user = au.id\_user

JOIN

adresa a ON au.id\_adresa = a.id\_adresa

JOIN

cumpara cmp ON u.id\_user = cmp.id\_user

JOIN

comanda c ON cmp.id\_comanda = c.id\_comanda

JOIN

depozit d ON cmp.id\_depozit = d.id\_depozit

JOIN

produs\_comandat pc ON c.id\_comanda = pc.id\_comanda

WHERE

pc.cantitate\_comandata > 5;

SELECT u.adresa\_email AS utilizator, c.id\_comanda, LISTAGG(p.nume, ', ') WITHIN GROUP (ORDER BY p.nume) AS produse,

LISTAGG(pc.cantitate\_comandata, ', ') WITHIN GROUP (ORDER BY p.nume) AS cantitati, SUM(pc.cantitate\_comandata \* pi.pret\_dupa\_reducere) AS total\_plata

FROM utilizator u

JOIN cumpara cu ON u.id\_user = cu.id\_user

JOIN comanda c ON cu.id\_comanda = c.id\_comanda

JOIN produs\_comandat pc ON c.id\_comanda = pc.id\_comanda

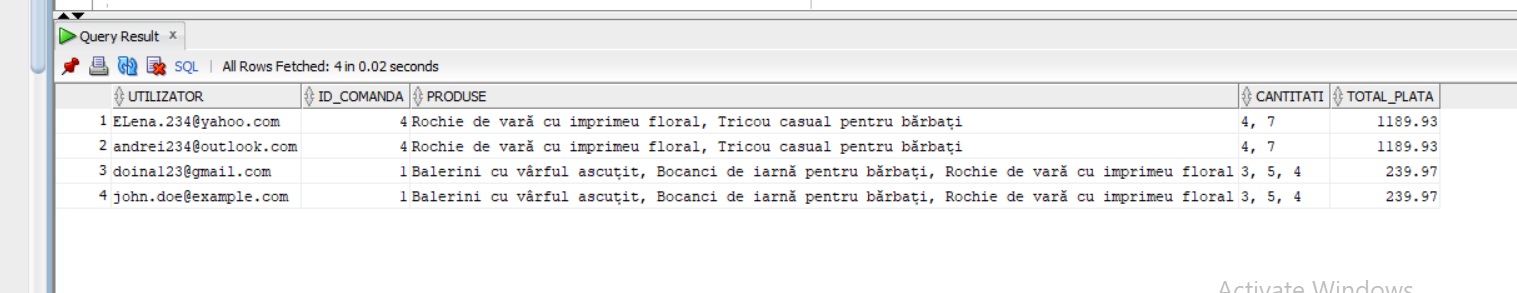
JOIN produs\_item pi ON pc.id\_produs\_item = pi.id\_produs\_item

JOIN produs p ON pi.id\_produs = p.id\_produs

GROUP BY u.adresa\_email, c.id\_comanda

HAVING SUM(pc.cantitate\_comandata) > 5

ORDER BY u.adresa\_email, c.id\_comanda;



ordonări și utilizarea funcțiilor NVL și DECODE (în cadrul aceleiași cereri)+

utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 funcții pe date calendaristice, a cel puțin unei expresii CASE

SELECT

id\_comanda,

NVL(data\_comanda, CURRENT\_TIMESTAMP) AS data\_comanda,

total\_plata,

NVL(metoda\_plata, 'Plata se va realiza la curier') AS metoda\_plata,

LOWER(

CASE

WHEN status\_comanda = 1 THEN 'In procesare'

WHEN status\_comanda = 2 THEN 'Livrare in curs'

WHEN status\_comanda = 3 THEN 'Livrata'

WHEN status\_comanda = 4 THEN 'Anulata'

ELSE 'Status necunoscut'

END

) AS status\_comanda,

UPPER(metoda\_plata) AS metoda\_plata\_mare,

SUBSTR(metoda\_plata, 1, 5) AS metoda\_plata\_partiala,

EXTRACT(YEAR FROM NVL(data\_comanda, CURRENT\_TIMESTAMP)) AS anul\_comenzii,

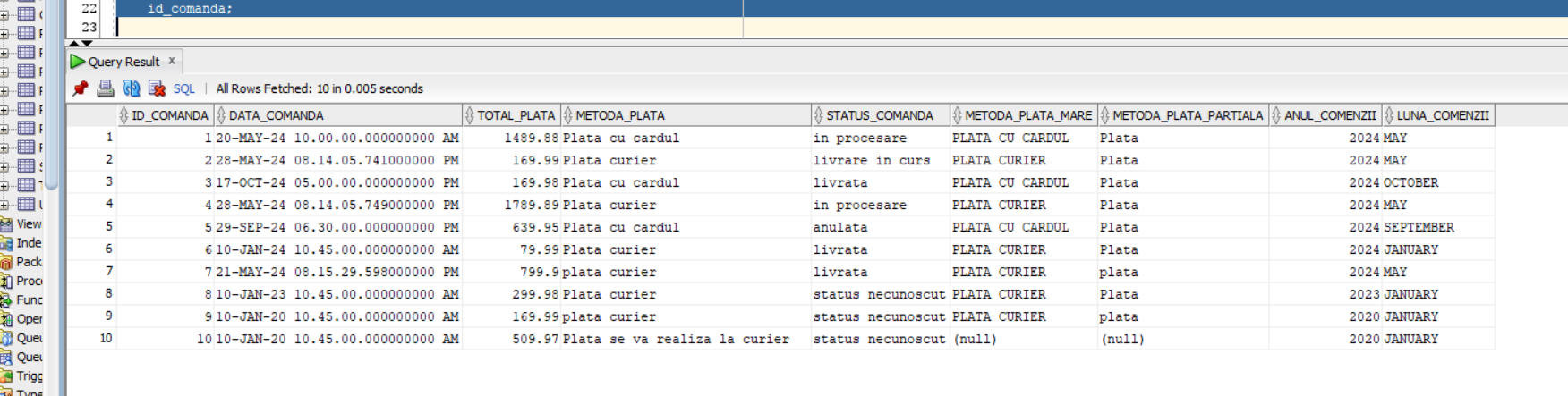
TO\_CHAR(NVL(data\_comanda, CURRENT\_TIMESTAMP), 'MONTH') AS luna\_comenzii

FROM

comanda

ORDER BY

id\_comanda;



utilizarea a cel puțin 1 bloc de cerere (clauza WITH)

WITH SumaTotalaPlati AS (

SELECT

id\_comanda,

SUM(total\_plata) AS suma\_totala

FROM

comanda

GROUP BY

id\_comanda

)

SELECT

c.id\_comanda,

NVL(c.data\_comanda, CURRENT\_TIMESTAMP) AS data\_comanda,

NVL(c.metoda\_plata, 'Plata curier') AS metoda\_plata,

LOWER(

CASE

WHEN c.status\_comanda = 1 THEN 'In procesare'

WHEN c.status\_comanda = 2 THEN 'Livrare in curs'

WHEN c.status\_comanda = 3 THEN 'Livrata'

WHEN c.status\_comanda = 4 THEN 'Anulata'

ELSE 'Status necunoscut'

END

) AS status\_comanda,

stp.suma\_totala AS suma\_totala\_de\_platit

FROM

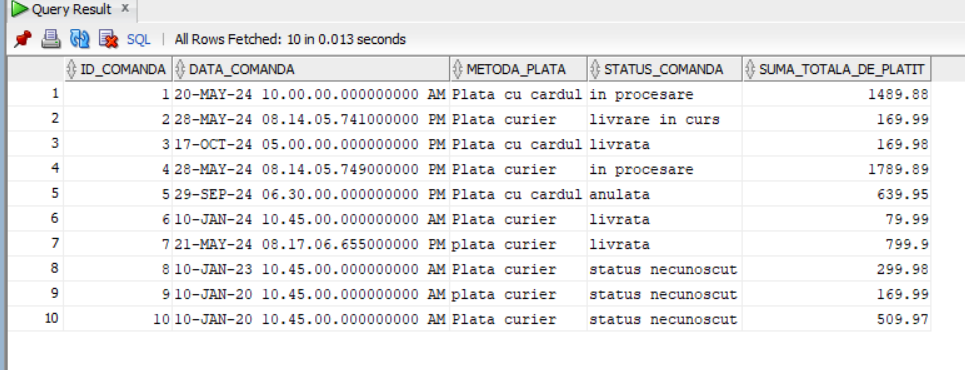
comanda c

LEFT JOIN

SumaTotalaPlati stp ON c.id\_comanda = stp.id\_comanda

ORDER BY

c.id\_comanda;



# Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri.

--aceasta comanda actualizeaza tabelul comanda dupa ce se adauga in produs\_comandat id-ul catre un produs\_ales si id\_comanda. Aceasta operatie updateaza coloana total\_plata in tabelul COMANDA. Inserarea comenzilor se face inaintea (asta inseamna ca total\_plata va fii 0) inserarilor in tabelul PRODUS\_COMANDAT deoarece PRODUS\_COMANDAT este un tabel de legatura. Astfel in momentul in care stim ce produse se afla in comanda noastra putem actualiza tabelul COMANDA si pret\_total sa fie afisat corect

UPDATE comanda c

SET total\_plata = (

SELECT SUM(

CASE

WHEN PI.pret\_dupa\_reducere IS NOT NULL THEN PA.cantitate\_comandata \* PI.pret\_dupa\_reducere

ELSE PA.cantitate\_comandata \* PI.pret\_original

END

)

FROM produs\_comandat pc

JOIN produs\_ales PA ON pc.id\_produs\_ales = PA.id\_produs\_ales

JOIN produs\_item PI ON PA.id\_produs\_item = PI.id\_produs\_item

WHERE pc.id\_comanda = c.id\_comanda

GROUP BY pc.id\_comanda

)

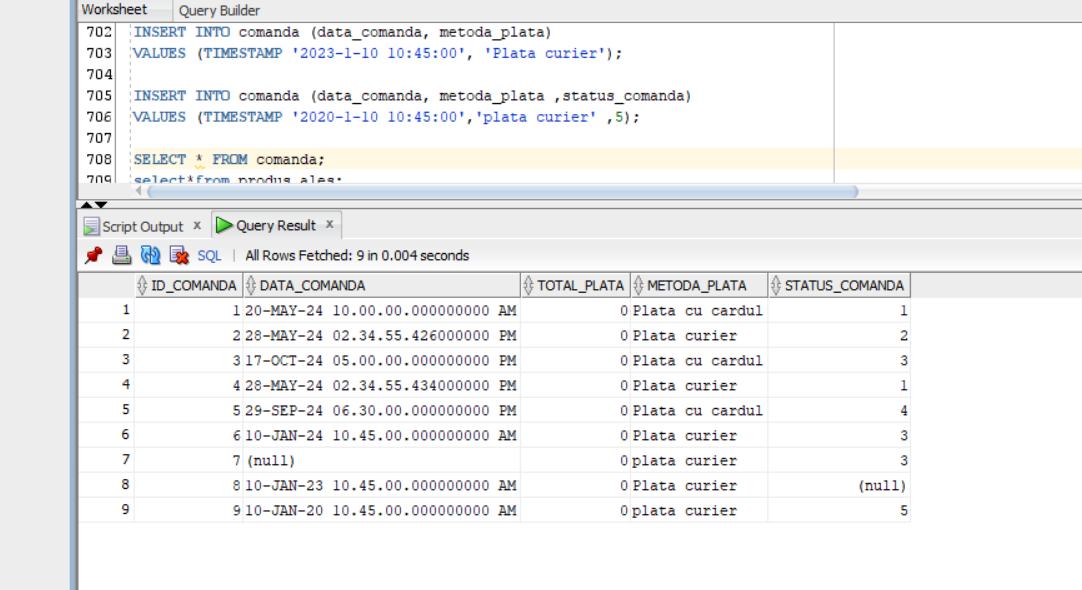
WHERE EXISTS (

SELECT 1

FROM produs\_comandat pc

WHERE pc.id\_comanda = c.id\_comanda

);

inainte de update:

Introduc elemente in tabelul PRODUS\_COMANDAT si dau update la tabelul COMANDA

Dupa update:



--acest update calculeaza pret\_dupa\_reducere pentru fiecare inregistrare din tabel

UPDATE PRODUS\_ITEM pi

SET pret\_dupa\_reducere = pret\_original - (

pret\_original \* (

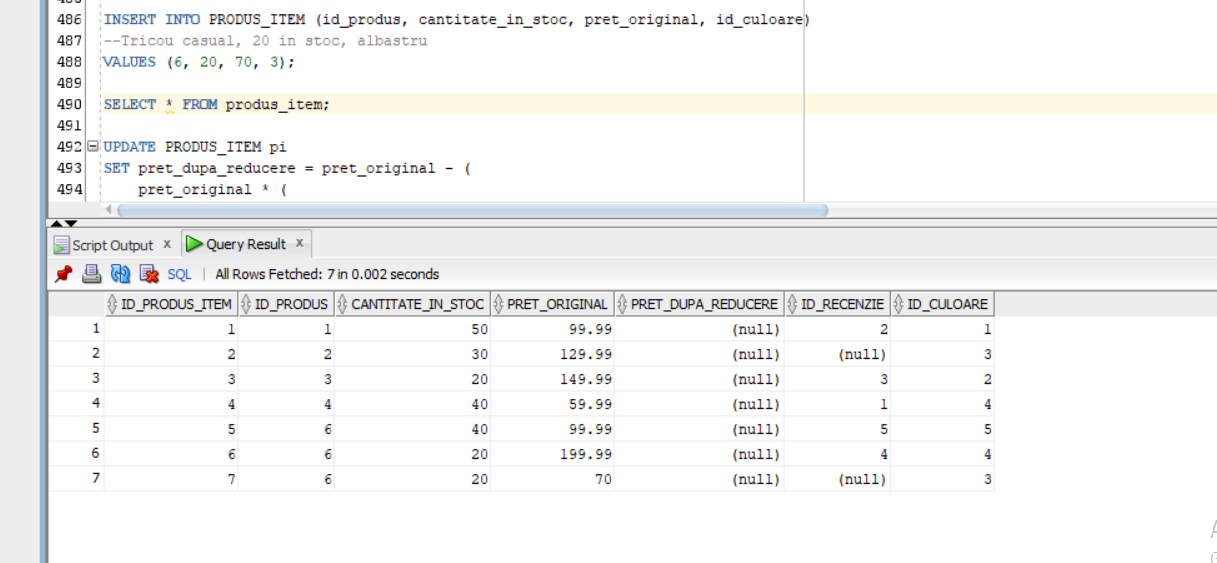
SELECT CAST(SUBSTR(p.discount, 1, LENGTH(p.discount) - 1) AS DECIMAL(4, 2)) / 100

FROM PROMOTIE p

JOIN PRODUS pr ON pi.id\_produs = pr.id\_produs

JOIN PRODUS\_CATEGORIE pc ON pr.id\_categorie = pc.id\_produs\_cat

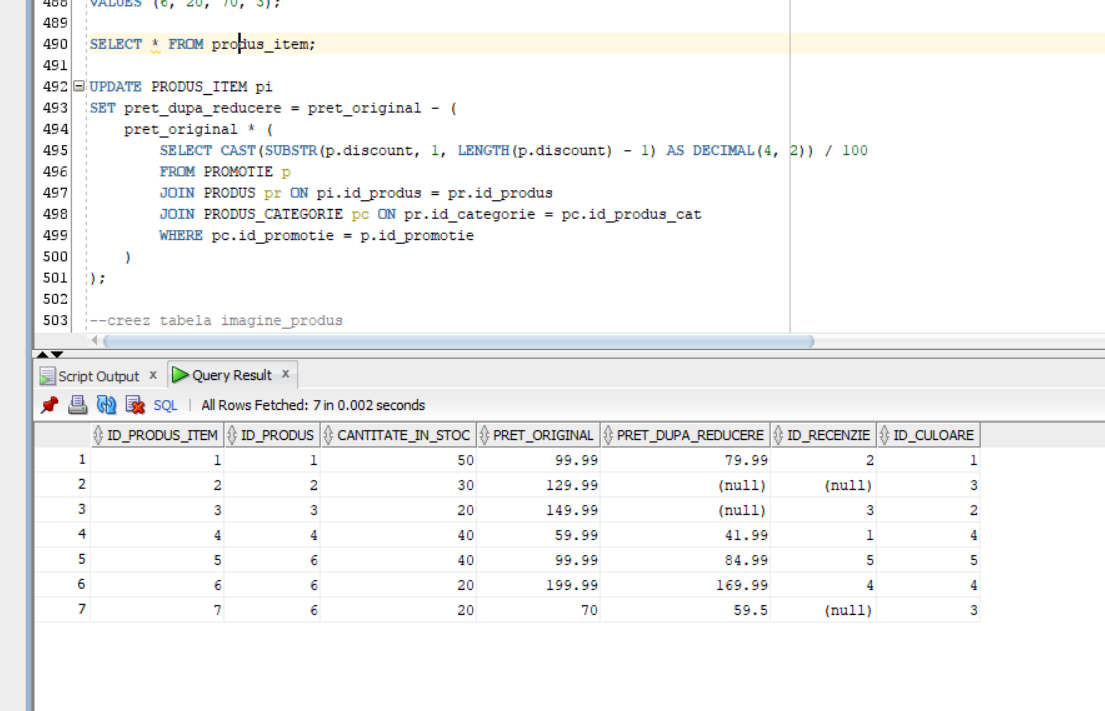
WHERE pc.id\_promotie = p.id\_promotie

 )

);

Inainte de update:

Dupa update:



-ne arata utilizatorii cu comenzi finalizate

SELECT u.adresa\_email, c.id\_comanda, sc.status

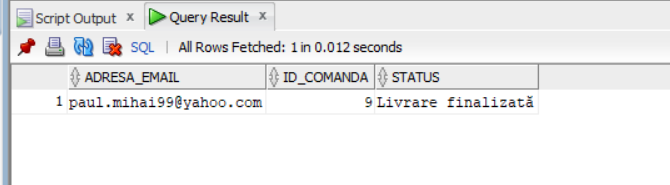
FROM CUMPARA cp

JOIN COMANDA c ON cp.id\_comanda = c.id\_comanda

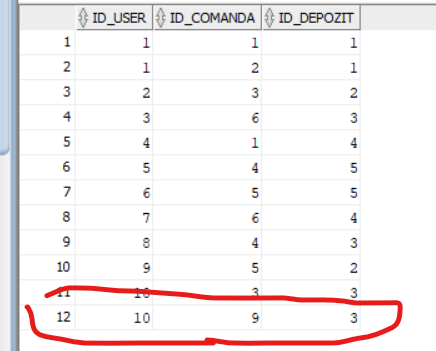
JOIN UTILIZATOR u ON cp.id\_user = u.id\_user

JOIN status\_comanda sc ON c.status\_comanda = sc.id\_status

WHERE c.status\_comanda = 5;



In tabelul cumpara vedem comanda 9



--stergem din baza de date acele comenzi care s-au finalizat deja

DELETE FROM CUMPARA cmp

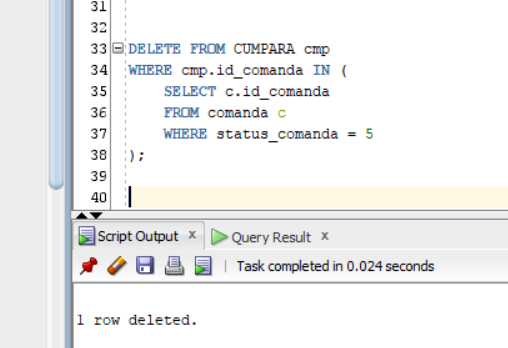
WHERE cmp.id\_comanda IN (

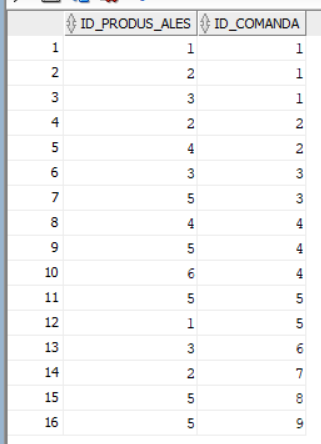
SELECT c.id\_comanda

FROM comanda c

WHERE status\_comanda = 5

);





Stergem si inregistrarile din produs\_comandat(before->)

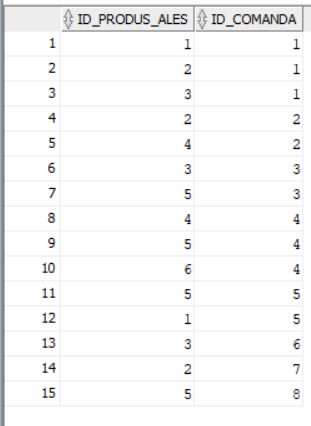
DELETE FROM PRODUS\_COMANDAT

WHERE id\_comanda IN (

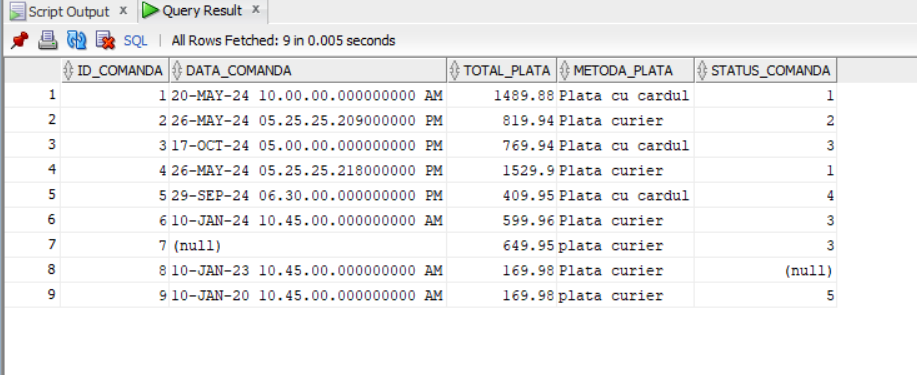
SELECT id\_comanda

FROM COMANDA

WHERE status\_comanda = 5

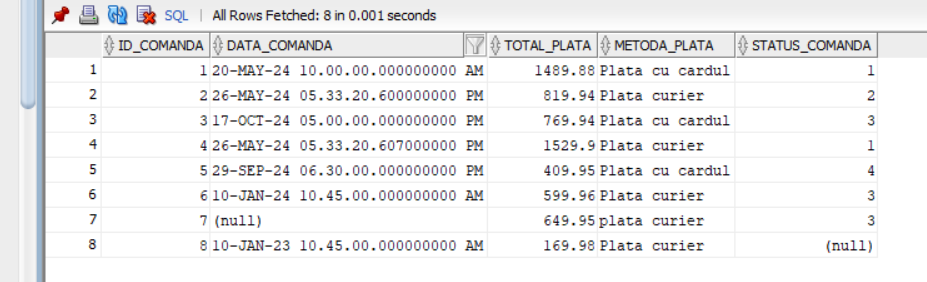
);

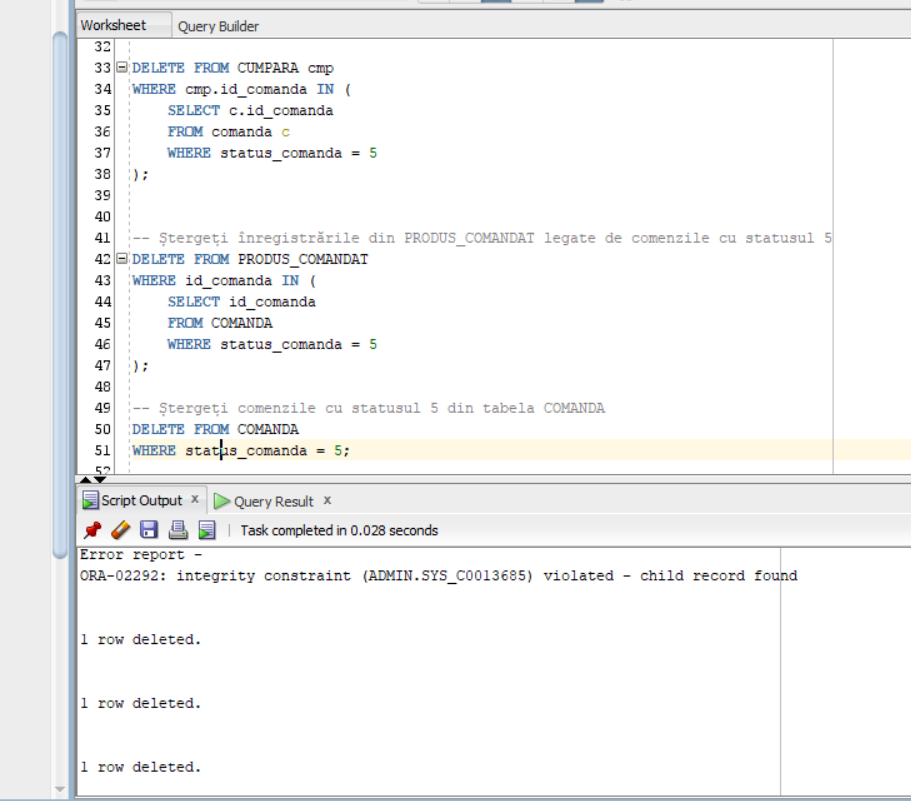
Stergem si inregistrarile din comanda(before)



DELETE FROM COMANDA

WHERE status\_comanda = 5;

****

****

Daca dorim sa facem aceste modificari intr-o singura tranzactie putem folosi urmatorul cod:

BEGIN

--Stergem din PRODUS\_COMANDAT

DELETE FROM PRODUS\_COMANDAT

WHERE id\_comanda IN (

SELECT id\_comanda

FROM COMANDA

WHERE status\_comanda = 5

);

-- Stergem din CUMPARA

DELETE FROM CUMPARA

WHERE id\_comanda IN (

SELECT id\_comanda

FROM COMANDA

WHERE status\_comanda = 5

);

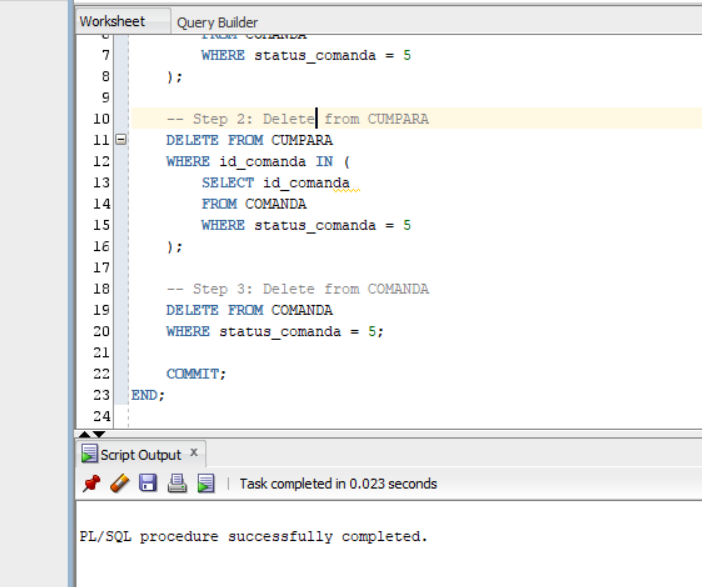
-- Stergem din COMANDA

DELETE FROM COMANDA

WHERE status\_comanda = 5;

COMMIT;

END;



# Crearea unei vizualizări complexe. Dați un exemplu de operație LMD permisă pe vizualizarea respectivă și un exemplu de operație LMD nepermisă.

--creez o vizualizare complexa pentru a obtine o imagine completa asupra comenzilor si produselor comandate de pe site

CREATE VIEW VIZUALIZARE\_COMENZI AS

SELECT c.id\_comanda, c.data\_comanda, c.total\_plata, sc.status AS status\_comanda, u.adresa\_email AS utilizator,

LISTAGG(p.nume, ', ') WITHIN GROUP (ORDER BY p.nume) AS nume\_articole

FROM COMANDA c

JOIN STATUS\_COMANDA sc ON c.status\_comanda = sc.id\_status

JOIN CUMPARA cmp ON c.id\_comanda = cmp.id\_comanda

JOIN UTILIZATOR u ON cmp.id\_user = u.id\_user

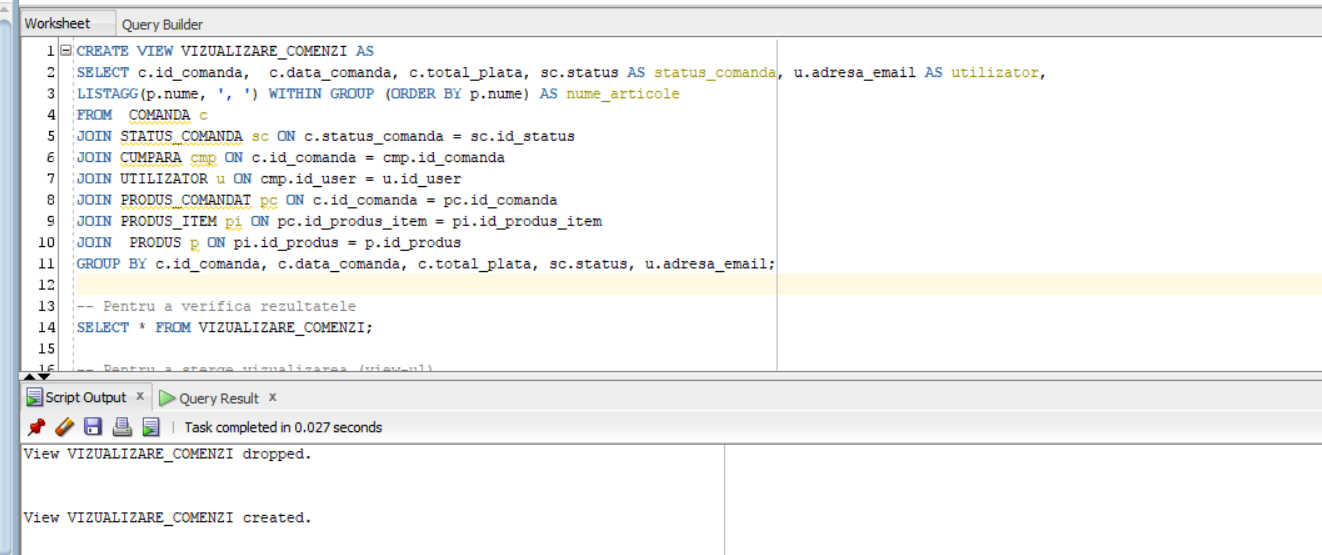
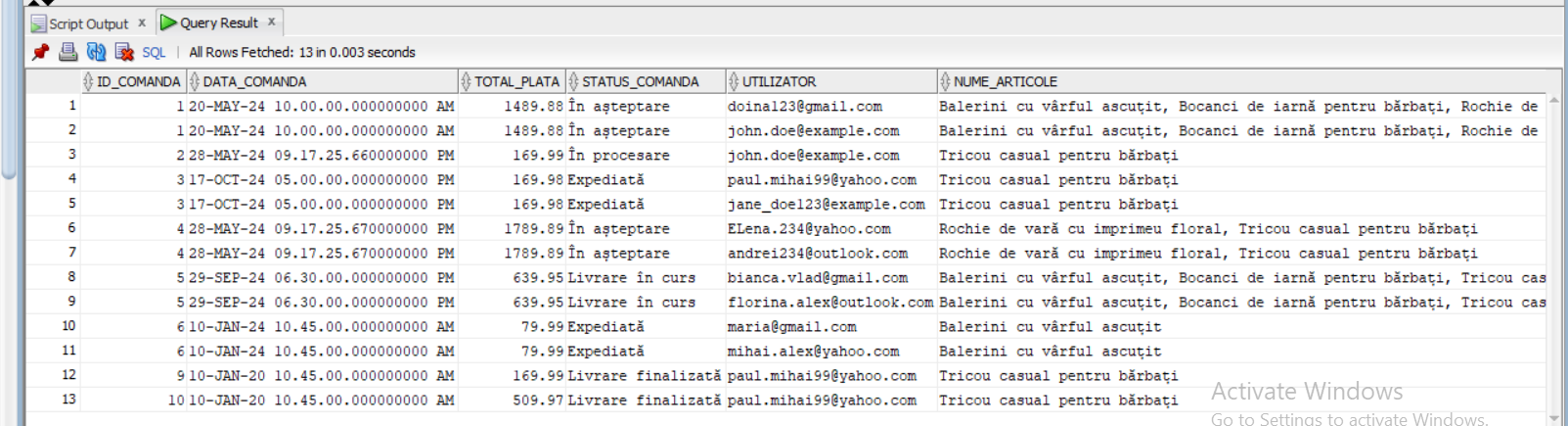
JOIN PRODUS\_COMANDAT pc ON c.id\_comanda = pc.id\_comanda

JOIN PRODUS\_ITEM pi ON pc.id\_produs\_item = pi.id\_produs\_item

JOIN PRODUS p ON pi.id\_produs = p.id\_produs

GROUP BY c.id\_comanda, c.data\_comanda, c.total\_plata, sc.status, u.adresa\_email;

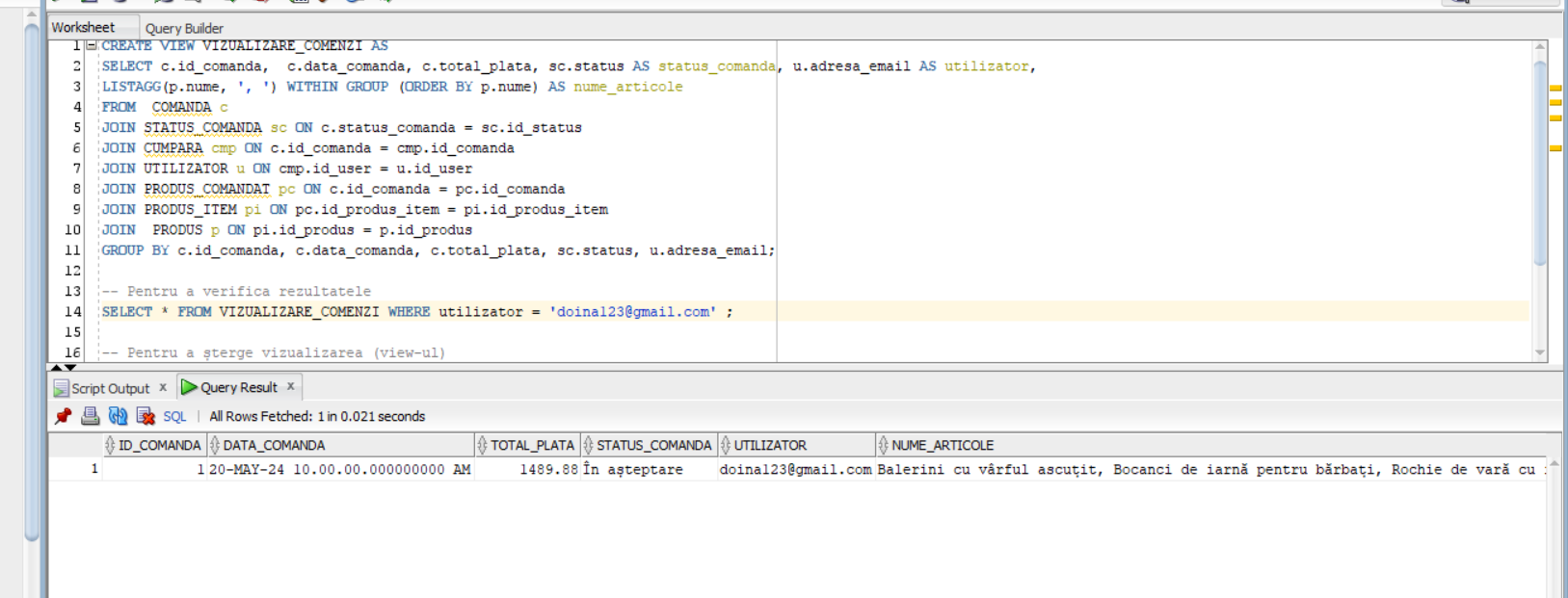
SELECT \* FROM VIZUALIZARE\_COMENZI;

****

O operatie LMD permisa este cea de SELECT

SELECT \* FROM VIZUALIZARE\_COMENZI WHERE utilizator = 'doina123@gmail.com' ;

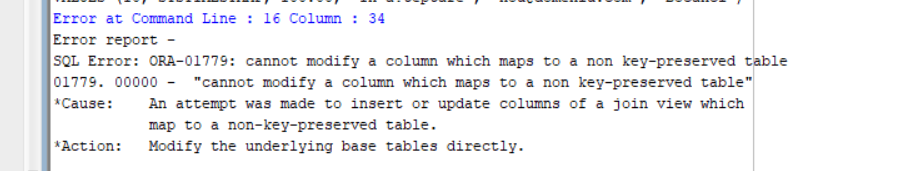
(De asemenea putem permite operatii de UPDATE daca utilizam INSTEAD OF triggers pentru insert, delete, update)



O operatie LMD nepermisa este cea de insert

INSERT INTO VIZUALIZARE\_COMENZI (id\_comanda, data\_comanda, total\_plata, status\_comanda, utilizator, nume\_articole)

VALUES (10, SYSTIMESTAMP, 100.00, 'În așteptare', 'nou@domeniu.com', 'Bocanci');



# Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outer-join pe minimum 4 tabele, o cerere ce utilizează operația division și o cerere care implementează analiza top-n.

--folosesc operatia outer join ca sa afisez inregistrarile din toate tabelele, chiar si daca nu exista corespondente intre ele

SELECT U.id\_user, U.adresa\_email AS Utilizator, D.id\_depozit, D.nume\_depozit, C.id\_comanda, PC.id\_produs\_item, PC.cantitate\_comandata

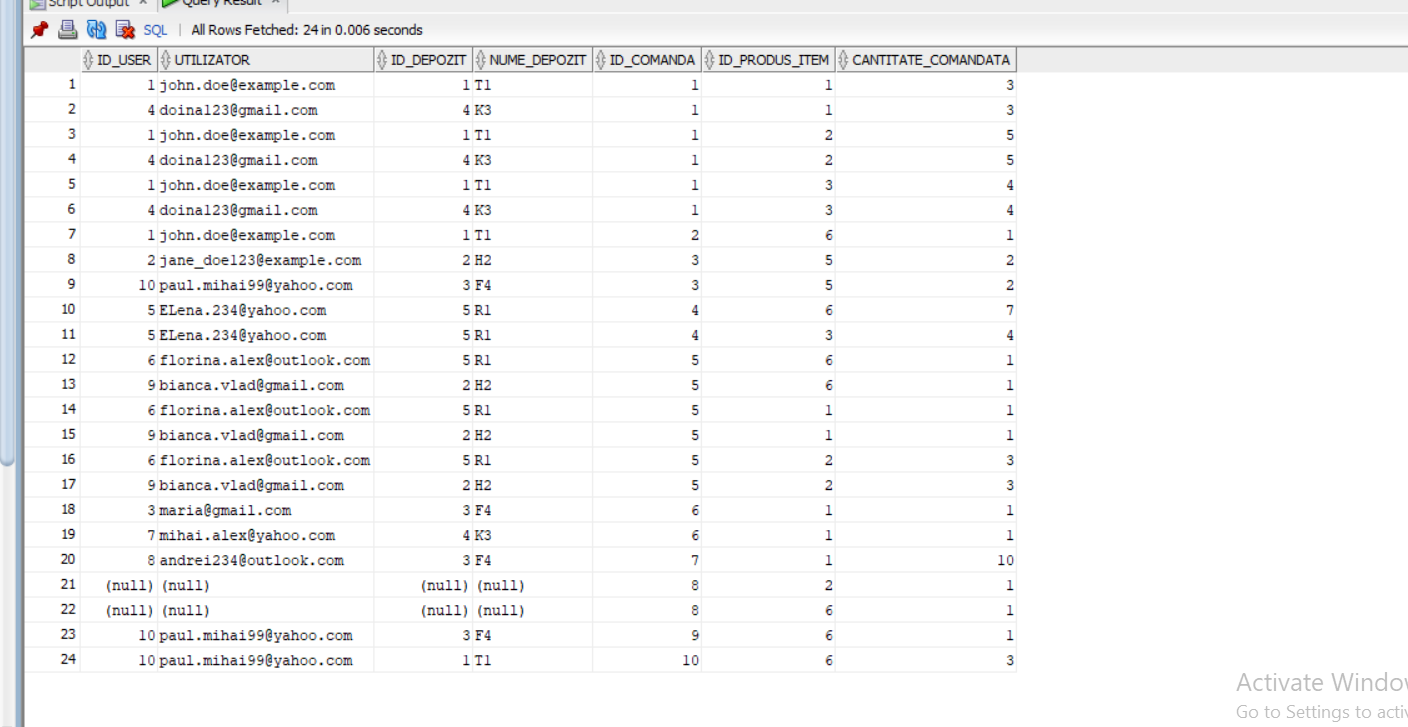
FROM utilizator U

FULL OUTER JOIN cumpara CMP ON U.id\_user = CMP.id\_user

FULL OUTER JOIN comanda C ON CMP.id\_comanda = C.id\_comanda

FULL OUTER JOIN depozit D ON CMP.id\_depozit = D.id\_depozit

FULL OUTER JOIN produs\_comandat PC ON C.id\_comanda = PC.id\_comanda;



--folosesc operatia division ca sa afisez utilizatorii care au comandat o comanda, dar acea comanda nu contine id\_produs\_item cu id-ul 1

SELECT u.id\_user, u.adresa\_email

FROM utilizator u

WHERE EXISTS (

SELECT 1

FROM comanda c

JOIN cumpara cmp ON c.id\_comanda = cmp.id\_comanda

WHERE cmp.id\_user = u.id\_user

AND NOT EXISTS (

SELECT 1

FROM produs\_comandat pc

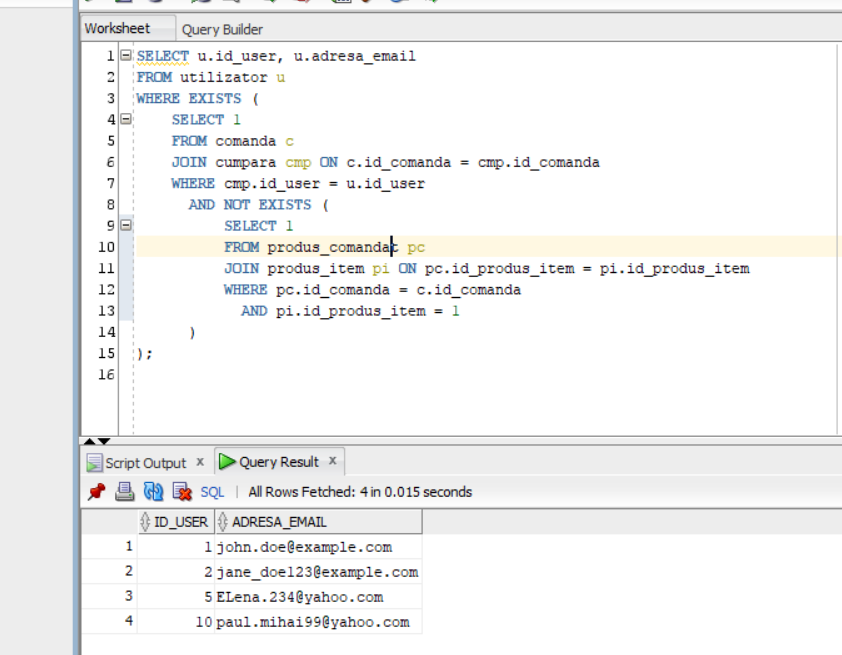
JOIN produs\_item pi ON pc.id\_produs\_item = pi.id\_produs\_item

WHERE pc.id\_comanda = c.id\_comanda

AND pi.id\_produs\_item = 1

)

);



--doresc sa obtin id-ul produselor care au fost cel mai des cumparate din comenzi. Asadar urmatorul query afiseaza id\_produs\_item si numar\_comenzi ceea ce reprezinta in cate comenzi s-a aflat acel produs.

SELECT id\_produs\_item, numar\_comenzi

FROM (

SELECT id\_produs\_item, COUNT(\*) AS numar\_comenzi

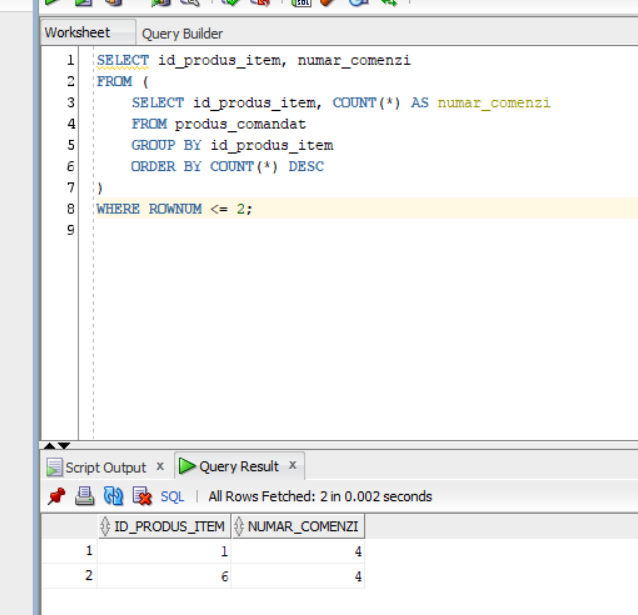
FROM produs\_comandat

GROUP BY id\_produs\_item

ORDER BY COUNT(\*) DESC

)

WHERE ROWNUM <= 2;



# Optimizarea unei cereri, aplicând regulile de optimizare ce derivă din proprietățile operatorilor algebrei relaționale. Cererea va fi exprimată prin expresie algebrică, arbore algebric și limbaj (SQL), atât anterior cât și ulterior optimizării.

ANTERIOR OPTIMIZARII

SELECT comanda.data\_comanda, utilizator.adresa\_email, adresa.cod\_postal, adresa.oras, tara.nume\_tara, adresa\_user.e\_default

FROM comanda

JOIN cumpara ON comanda.id\_comanda = cumpara.id\_comanda

JOIN utilizator ON cumpara.id\_user = utilizator.id\_user

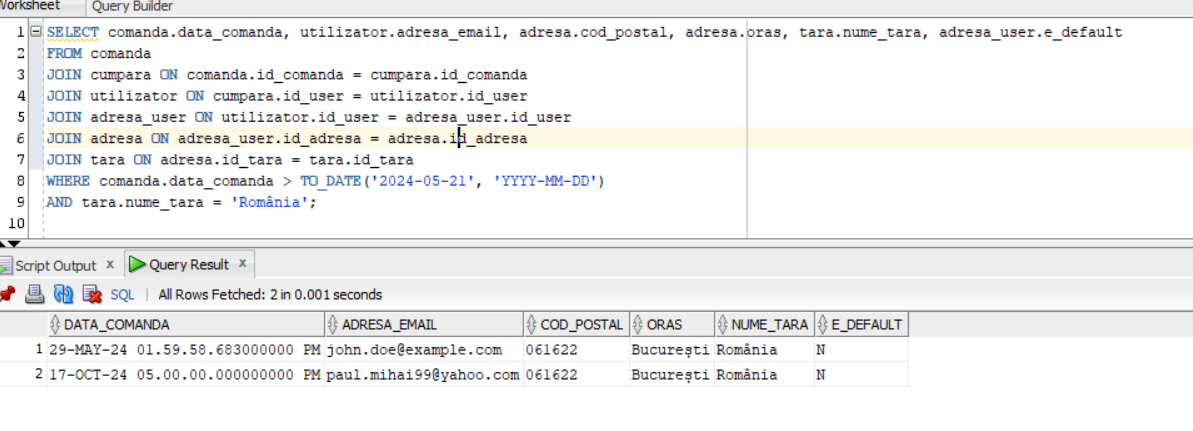
JOIN adresa\_user ON utilizator.id\_user = adresa\_user.id\_user

JOIN adresa ON adresa\_user.id\_adresa = adresa.id\_adresa

JOIN tara ON adresa.id\_tara = tara.id\_tara

WHERE comanda.data\_comanda > TO\_DATE('2024-05-21', 'YYYY-MM-DD')

AND tara.nume\_tara = 'România';



Arborele algebric:

ULTERIOR OPTIMIZARII

--afiseaza datele pentru utilizatorii careu au facut comenzi dupa data de 21 mai 2024 si care au domiciliul in Romania

SELECT c.data\_comanda, u.adresa\_email, a.cod\_postal, a.oras, t.nume\_tara, au.e\_default

FROM(SELECT id\_comanda, data\_comanda

FROM comanda

WHERE data\_comanda > TO\_DATE('2024-05-21', 'YYYY-MM-DD')) c

JOIN (SELECT id\_user, id\_comanda

FROM cumpara)cmp ON c.id\_comanda = cmp.id\_comanda

JOIN(SELECT id\_user, adresa\_email

FROM utilizator)u ON cmp.id\_user = u.id\_user

JOIN(SELECT id\_user, id\_adresa, e\_default

FROM adresa\_user)au ON u.id\_user = au.id\_user

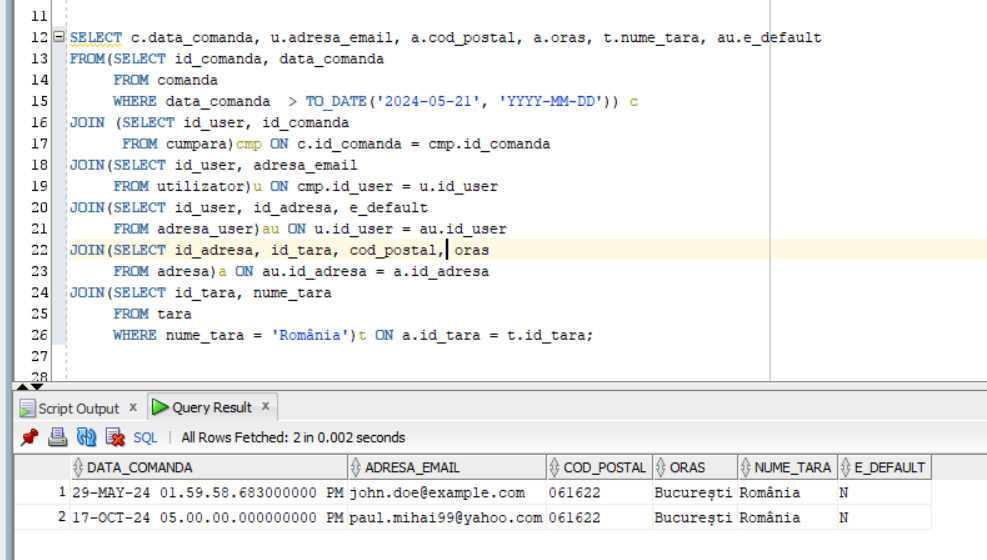
JOIN(SELECT id\_adresa, id\_tara, cod\_postal, oras

FROM adresa)a ON au.id\_adresa = a.id\_adresa

JOIN(SELECT id\_tara, nume\_tara

FROM tara

WHERE nume\_tara = 'România')t ON a.id\_tara = t.id\_tara;



# a. Realizarea normalizării BCNF, FN4, FN5.

BCNF:

Ca sa ne aflam in BCNF trebuie mai intai sa avem FN3

Consideram urmatorul tabel:

COMANDA(id\_comanda, id\_adresa, depozit)

Avem conditiile:

1. fiecare comanda poate sa aiba mai multe adrese
2. Mai multe depozite pot trimite o anumita comanda
3. pt fiecare adresa, i se asociaza o comanda si un depozit

|  |  |  |
| --- | --- | --- |
| **id\_comanda** | **id\_adresa** | **depozit** |
| 1 | 1 | R1 |
| 2 | 2 | T1 |
| 3 | 3 | H1 |

In tabelul de mai sus, id\_comanda si id\_adresa formeaza cheia primara pentru ca acestea determina fiecare coloana din tabel.

Un depozit trimite o comanda, dar o comanda poate fi trimisa de mai multe depozite. Ceea ce inseamna ca avem o dependenta intre depozit si comanda

FN1: e satisfacuta, toate valorile sunt atomice, iar coloanele unice

FN2: e satisfacuta, nu avem dependente partiale

FN3: e satisfacuta, nu avem dependente tranzitive

Tabelul urmareste toate formele normale mai putin BCNF

Deoarece id\_comanda si id\_adresa formeaza cheia primara, ceea ce inseamna ca atributele din id\_adresa sunt atribute prime.

Dar avem dependenta depozit -> id\_comanda. Asta inseamnca ca depozit nu e atribut prim.

Asta inseamna ca transformam tabelul asa:

|  |  |
| --- | --- |
| id\_adresa | id\_livrare |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

Si

|  |  |  |
| --- | --- | --- |
| id\_livrare | id\_comanda | depozit |
| 1 | 1 | R1 |
| 2 | 2 | T1 |
| 3 | 3 | H1 |

Asadar BCNF e completa.

FN4:

Consideram urmatorul tabel:

(id\_user, id\_comanda, preferinte)

Un user poate aveam mai multe comenzi si mai multe preferinte.

|  |  |  |
| --- | --- | --- |
| id\_user | id\_comanda | preferinte |
| 1 | 1 | elegant |
| 1 | 3324 | casual |
| 3 | 45 | sportiv |

Tabelul se afla in FN3, dar id\_comanda si preferinte sunt entitati independente. Asadar nu avem o relatie intre ele

In tabelul de mai sus avem id\_user cu id-ul 1 care are doua comenzi si doua preferinte. Asta inseamna ca avem dependenta multivalorica pt id\_user ceea ce duce la repetarea inutila a datelor.

Pt a respecta FN4 despartim in felul urmator:

USER\_COMANDA

|  |  |
| --- | --- |
| id\_user | id\_comanda |
| 1 | 1 |
| 1 | 3324 |
| 3 | 45 |

Si

PREFERINTE\_USER

|  |  |
| --- | --- |
| id\_user | preferinte |
| 1 | elegant |
| 1 | casual |
| 3 | sportiv |

FN5:

Consideram urmatorul tabel: VANZARE(asistent\_vanzari, promotie, produse )

|  |  |  |
| --- | --- | --- |
| asistent\_vanzari | brand | produse |
| Andrei | Nike | incaltaminte |
| Andrei | Nike | imbracaminte |
| Maria | Chanel | imbracaminte |
| Maria | Chanel | parfum |
| Elena | Nike | incaltaminte |
| Elena | Chanel | imbracaminte |
| Elena | Levi’s | imbracaminte |

Un asistent\_vanzari are mai multe branduri de la care poate sa vanda produse intr-un magazin.

Cheia primara o reprezinta compunerea tuturor coloanelor.

Tabelul se afla in NF4 deoarece nu avem dependente multivalorice.

Presupunem urmatoriul scenariu: Un asistent de vanzari are anumite branduri si anumite produse in repertoriul sau. Daca brandul B1 si brandul B2 se afla in repertoriul sau, si produsul P se afla in repertoriul sau atunci (presupunand ca B1 si B2 fac impreuna produsul P), asistentul de vanzari trebuie sa ofere produse de tipul P, facute de B1 si cele facute de B2.

Asadar putem desparti tabelul astfel:

T1(Produse ale asistentului)

|  |  |
| --- | --- |
| asistent\_vanzari | produse |
| Andrei | incaltaminte |
| Andrei | imbracaminte |
| Maria | imbracaminte |
| Maria | parfum |
| Elena | incaltaminte |
| Elena | imbracaminte |
| Elena | imbracaminte |

T2(Branduri ale asistentului)

|  |  |
| --- | --- |
| asistent\_vanzari | brand |
| Andrei | Nike |
| Maria | Chanel |
| Elena | Nike |
| Elena | Chanel |
| Elena | Levi’s |

T3(Produsele brandului)

|  |  |
| --- | --- |
| brand | produse |
| Nike | incaltaminte |
| Nike | imbracaminte |
| Chanel | imbracaminte |
| Chanel | parfum |
| Levi’s | imbracaminte |

Asadar: VANZARE ≠ JOIN(T1, T2)

VANZARE ≠ JOIN(T1, T3)

VANZARE ≠ JOIN(T2, T3)

VANZARE = JOIN(JOIN(T1, T2), T3)

# b**. Aplicarea denormalizării, justificând necesitatea acesteia.**

Avem urmatorul tabel:

COMANDA(id\_user, id\_comanda, recenzie)

Scopul nostru este de a obtine informatii mai rapid, sa reducem timpul de rulare al interogarilor de selectie, asadar vom adauga date redundante in tabela COMANDA si anume adaugam atributele email\_user, data\_comanda, total\_plata, status\_comanda

Tabelul denormalizat va arata astfel:

COMANDA\_DENORM(id\_user, id\_comanda, email\_user, data\_comanda, total\_plata, status\_comanda, recenzie)

Deoarece adaugam atribute redundante, putem evita necesitatea de a efectua operatii de join care s-ar putea prelungi pe mai multe tabele (ex. daca trebuie sa facem 10 join-uri pentru a returna numele clientului si produsele vandute acestuia. Unele tabele de-a lungul caii ar putea contine cantitati mari de date, asadar e mai usor sa adaugam un atribut id\_client in tabela produse\_vandute)

Cu toate acestea este important sa mentionam ca pot exista si dezavantaje la denormalizare, precum:

* spatiul disk-ului: deoarece avem duplicare de date
* anomalii de date: datele trebuie acum schimbate in mai multe locuri. Trebuie sa fim atenti ca acestea sa se afle in concordanta
* incetinirea altor operatii: vom incetini operatiile de inserare, modificare, stergere, deoarece avem mai multe locuri in care datele sunt stocate si actualizate