

Universitatea din Bucureşti
Facultatea de Matematică și Informatică

GESTIUNEA UNUI SISTEM BANCAR

(Proiect Sisteme de Gestiune a Bazelor de Date)

Student: Ghinea Dragoș Dumitru
Specializarea: Informatică
Grupa: 232
Anul II

Profesor Coordonator: Lect. Univ. Dr. Gabriela Mihai

Profesor Laborator: Ioana Delia Gherghe

Bucureşti
2023

Cuprins

1. Prezentare pe scurt a bazei de date (utilitatea ei).....	3
1.1. Descrierea modelului real.....	3
1.2. Alte restricții impuse asupra modelului.....	4
2. Diagrama entitate-relație (ERD).....	5
3. Diagrama conceptuală.....	6
3.1 Grafic.....	6
3.2 Scheme relaționale.....	7
3.3. Descrierea entităților.....	7
3.4. Descrierea relațiilor.....	9
3.5. Descrierea atributelor.....	11
4. Implementarea diagramei conceptuale.....	17
5. Inserare de date.....	26
6. Problemă cu tipuri de colecții.....	48
7. Problemă cu tipuri de cursoare.....	52
8. Problemă cu funcție, excepții, trei tabele într-o comanda SQL.....	56
9. Problemă cu procedură, excepții, cinci tabele într-o comanda SQL.....	62
10. Trigger LMD la nivel de comandă.....	67
11. Trigger LMD la nivel de linie.....	68
12. Trigger LDD.....	74
13. Un pachet care să conțină toate obiectele definite.....	75
14. Pachet de utilități.....	86

1. Prezentați pe scurt baza de date (utilitatea ei)

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

Modelul de date va gestiona informații legate de funcționarea unui sistem bancar. În cadrul acestui sistem există mai multe persoane care pot fi clienți sau angajați. Clienții sunt înregistrați prin intermediul unui contract de utilizare care se realizează în prezență/cu acordul unui angajat.

Banca în cauză are mai multe sedii la care lucrează angajați, însă există și angajați care lucrează remote. Sistemul oferă clienților mai multe conturi care au asociat automat câte un cont de economii, care oferă un mic bonus persoanelor care contribuie cu depozite pe termen lung. Un depozit în contul de economii se realizează în funcție de dobânzile disponibile. După trecerea perioadei oferite de dobândă, clientul poate alege să își retragă banii înapoi în contul principal, cu adaosul corespunzător, oricând dorește. Întrucât procentul dobânzii poate fluctua, clientul poate aștepta un procent mai favorabil, dacă simte că acesta poate să apară.

Angajații pot fi clasificați după job-ul lor, aceștia putând fi moderatori, administratori, casieri, etc. Un contract de utilizator se realizează fie la un sediu, în prezență unui casier, fie online, urmând să fie aprobat de un moderator sau administrator.

Tranzacțiile realizate la bancomat sau între conturi sunt stocate într-un istoric asociat conturilor în cauză. Intervențiile administrative sau depozitele/retragerele realizate prin intermediul angajaților figurează în același istoric de tranzacții menționat anterior. O tranzacție externă are mai mult rol informativ, ea nu fiind o tranzacție propriu-zisă, ci o notificare din partea sistemului referitoare la modificări realizate de personalul băncii.

Restricții de funcționare respectate de acest model de date:

- Dacă o persoană are mai multe domicilii, se va lua în considerare doar unul, ales de persoana în cauză. Dacă domiciliul lipsește din diverse motive (persoana se află în mijlocul unei mutări, are un domiciliu privat) acesta va fi setat la NULL.

- Mai multe persoane nu pot folosi aceeași adresă de email.

- Un depozit sau o retragere se realizează neapărat la un bancomat folosind un card. Depozite/retrageri realizate la sediu sunt considerate tranzacții externe.

- Un cont are mereu asociat un cont de economii, și numai unul. Tot un cont are un istoric de tranzacții asociat și doar unul.

- Un cont de economii nu poate exista fără un cont deoarece clientul nu ar avea unde să-și stocheze suma blocată și dobânda câștigată la momentul revendicării depozitului de economii.

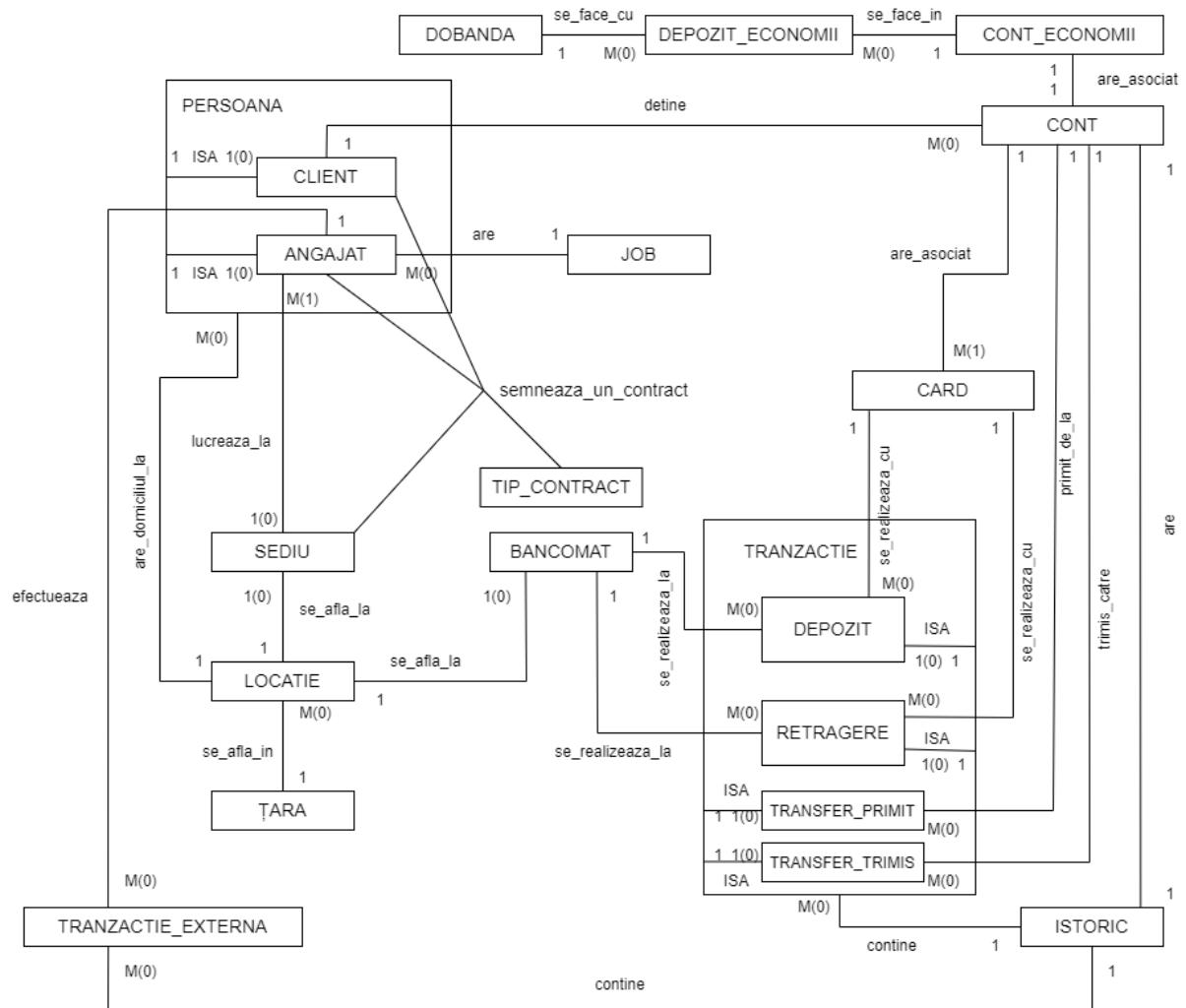
- Un transfer bancar se realizează între două conturi, care pot să aparțină aceluiași client și este stocat sub forma a două entități. Se presupune că există un cooldown de 3 secunde la nivel de aplicație între tranzacțiile efectuate de un cont.

- Un depozit de economii la un moment în timp este fie revendicat sau nerevendicat.

1.2. Alte restricții impuse asupra modelului

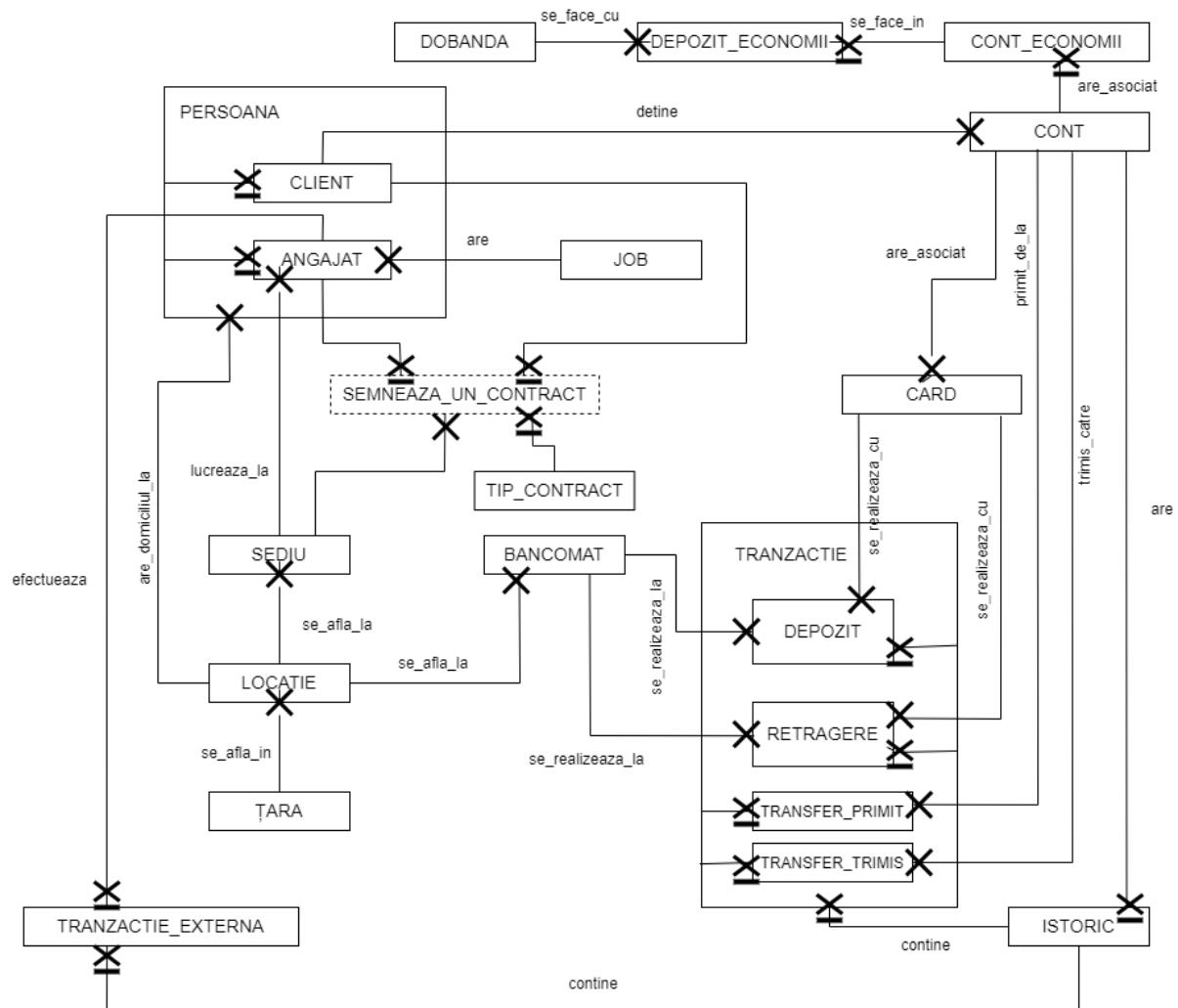
- Un client poate să aibă mai multe conturi, dar un cont aparține doar unui client. Sistemul nu lucrează cu persoane juridice.
- Un angajat poate avea un singur job.
- La un sediu lucrează minim un angajat, dar un angajat nu este obligat să lucreze la un sediu, decât dacă este casier.
- La o locație nu se pot afla două sedii, nici două bancomate.
- Un cont poate avea asociat mai multe carduri dar un card aparține doar unui cont, implicit doar unui client.
- Un depozit de economii se face în funcție de o singură dobândă și este asociat unui singur cont de economii. Un cont de economii poate avea mai multe depozite.
- Un contract de utilizator se realizează la înregistrarea clientului, cu aprobatarea unui angajat, fie online fie la unul din sedii. Alte contracte pot fi semnate pe parcursul desfășurării activității de utilizator.
- Contractele legate de angajați nu vor fi stocate în acest model.
- Un sold bancar poate să aibă valori negative, semnificând că persoana în cauză are datorii.
- Acest model de date nu stochează informații arhive precum job_history, card_history, contract_history, cont_history, etc_history.
- Depozitele de economii revendicate rămân salvate până la arhivarea acestora, moment în care s-ar muta în depozite_de_economii_history, transfer care nu este responsabilitatea acestui model.

2. Realizați diagrama entitate-relație (ERD)



3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.

3.1. Diagrama conceptuală



3.2 Scheme relaționale

PERSOANA(cod_persoana#, cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
CLIENT(cod_persoana#, tag_utilizator)
ANGAJAT(cod_persoana#, salariu, cod_sediu, cod_job)
DOBANDA(cod_dobanda#, procent, durata_luni)
CONT(cod_cont#, cod_persoana, sold, IBAN, data_creare)
CONT_ECONOMII(cod_cont#, depozite_contor)
DEPOZIT_ECONOMII(cod_cont#, cod_depozit_economii#, cod_dobanda, valoare, data_start, data_sfarsit, revendicat)
JOB(cod_job#, denumire_job, salariu_minim, salariu_maxim)
SEDIU(cod_sediu#, cod_locatie)
BANCOMAT(cod_bancomat#, cod_locatie)
LOCATIE(cod_locatie#, cod_postal, adresa, oras, cod_tara)
TARA(cod_tara#, nume_tara)
ISTORIC(cod_cont#, tranzactii_contor)
TRANZACTIE_EXTERNA(cod_persoana#, cod_cont#, data_tranzactie#, mesaj_administrativ)
CARD(cod_card#, pin, tip, card_number, cod_cont)
TRANZACTIE(cod_tranzactie#, cod_cont#, valoare, data_tranzactie)
DEPOZIT(cod_tranzactie#, cod_cont#, cod_bancomat, cod_card)
RETRAGERE(cod_tranzactie#, cod_cont#, cod_bancomat, cod_card)
TRANSFER_PRIMIT(cod_tranzactie#, cod_cont#, cod_cont_sursa)
TRANSFER_TRIMIS(cod_tranzactie#, cod_cont#, cod_cont_destinatie)
TIP_CONTRACT(tip_contract#, titlu, continut)
SEMNEAZA_UN_CONTRACT(cod_angajat#, cod_client#, tip_contract#, cod_sediu, data_contract)

3.3. Descrierea entităților, incluzând precizarea cheii primare.

Pentru modelul de date referitor la sistemul bancar, structurile PERSOANA, CLIENT, ANGAJAT, JOB, DOBANDA, DEPOZIT_ECONOMII, CONT_ECONOMII, CONT, CARD, ISTORIC, SEDIU, BANCOMAT, LOCATIE, TARA, TRANZACTIE, DEPOZIT, RETRAGERE, TRANSFER_TRIMIS, TRANSFER_PRIMIT, TRANZACTIE_EXTERNA reprezintă entități.

PERSOANA = Persoană fizică, om considerat ca subiect cu drepturi și cu obligații și care participă în această calitate la raporturile juridice civile. Cheia primară a entității este *cod_persoana*.

CLIENT = Subentitate a entității PERSOANA, care conține informații specifice clienților. Cheia primară a entității este *cod_persoana*.

ANGAJAT = Subentitate a entității PERSOANA, care conține informații specifice angajaților. Cheia primară a entității este *cod_angajat*.

TIP_CONTRACT = Un şablon de contract care conține informații generale despre regulile și prevederile băncii. Cheia primară a entității este *tip_contract*.

JOB = Serviciu pe care îl prestează un angajat. Cheia primară a entității este *cod_job*.

DOBANDA = În general, reprezintă o sumă de bani pe care un client o poate primi pentru contribuția sa la activitatea bancară, realizată prin blocarea unei cantități de bani în contul de economii. În acest model de date, ea reprezintă un procent care poate varia în timp, însotit de o perioadă care exprimă timpul după care procentul menționat anterior poate fi revendicat. Cheia primară a entității este *cod_dobanda*.

CONT = Instrument de stocare a datelor contabile. Cheia primară a entității este *cod_cont*.

CONT_ECONOMII = Entitate dependentă de CONT, care stochează banii blocați în vederea obținerii unei dobânzi. Cheia primară a entității este *cod_cont*.

DEPOZIT_ECONOMII = O sumă de bani blocată de bunăvoie de către client pe o perioadă aleasă în vederea obținerii unui procent extra de bani. Entitatea este dependentă de *CONT_ECONOMII*. Cheia primară a entității este compusă din *cod_cont* și *cod_depozit_economii*.

CARD = Instrument de plată pe suport magnetic, emis de o bancă sau alte instituții financiare autorizate, prin care deținătorul poate efectua diferite tranzacții. Cheia primară a entității este *cod_card*.

ISTORIC = Instrument de stocare a informațiilor referitoare la tranzacțiile efectuate. Entitatea este dependentă de *CONT* și are cheia primară *cod_cont*.

SEDIU = Spațiu de desfășurare al activității angajaților băncii. De asemenea, locul în care se pot înregistra clienții dacă nu optează pentru varianta online. Cheia primară a entității este *cod_sediu*.

BANCOMAT = Automat bancar care eliberează numerar pe baza cardului. Cheia primară a entității este *cod_bancomat*.

LOCATIE = Entitate care identifică o locație în care fie se află un sediu fie un bancomat sau care este asociată unei persoane, semnificând domiciliul acesteia. Cheia primară a entității este *cod_locatie*.

TARA = Entitate care identifică o țară. Cheia primară a entității este *cod_tara*.

TRANZACTIE_EXTERNA = Entitate care se ocupă cu stocarea notificărilor oferite de angajați, la intervenția administrativă a acestora asupra unui cont. Entitatea este dependentă de angajat cât și de istoric. Cheia primară a acestei entități este compusă din *cod_persoana*, *cod_cont*, *data_tranzactie*.

TRANZACTIE = Entitate dependentă de ISTORIC, care stochează o modificare asupra soldului contului realizată direct de utilizator. Cheia primară a acestei entități este compusă din *cod_cont* și *cod_tranzactie*.

DEPOZIT = Subentitate a entității TRANZACTIE care stochează o creștere a soldului unui cont. Cheia primară este compusă din *cod_cont*, *cod_tranzactie*.

RETRAGERE = Subentitate a entității TRANZACTIE care stochează o scădere a soldului unui cont. Cheia primară este compusă din *cod_cont*, *cod_tranzactie*.

TRANSFER_TRIMIS = Subentitate a entității TRANZACTIE care stochează jumătate dintr-un transfer a unei sume de bani dintr-un cont în altul, mai exact în istoricul contului care trimite, stochează contul către care se trimite. Cheia primară a acestei entități este compusă din *cod_cont*, *cod_tranzactie*.

TRANSFER_PRIMIT = Subentitate a entității TRANZACTIE care stochează jumătate dintr-un transfer a unei sume de bani dintr-un cont în altul, mai exact în istoricul contului care primește, stochează contul de la care se primește. Cheia primară a acestei entități este compusă din *cod_cont*, *cod_tranzactie*.

3.4. Descrierea relațiilor, incluzând precizarea cardinalității acestora.

CLIENT_detine_CONT = relație dintre entitățile CLIENT și CONT (ce conturi deține un client). Relația are cardinalitatea minimă 1:0 (un client nu trebuie să aibă conturi dar un cont trebuie să aibă un client.) și cardinalitatea maximă 1:n (un client poate avea mai multe conturi, iar un cont poate avea doar un client.).

ANGAJAT_are_JOB = relație dintre entitățile ANGAJAT și JOB (care este jobul fiecărui angajat). Relația are cardinalitatea minimă 0:1 (un angajat trebuie să aibă un job, un job nu trebuie să aibă un angajat) și cardinalitatea maximă n:1 (un angajat poate să aibă doar un job, iar un job poate să aibă mai mulți angajați).

ANGAJAT_lucreaza_la_SEDIU = relația dintre entitățile ANGAJAT și SEDIU (unde lucrează angajații). Relația are cardinalitatea minimă 1:0 (un angajat nu trebuie să lucreze la un sediu, dar la un sediu trebuie să lucreze un angajat) și cardinalitatea maximă n:1 (la un sediu pot lucra mai mulți angajați dar un angajat poate lucra la maxim un sediu).

SEDIU_se_afla_la_LOCATIE = relația dintre entitățile SEDIU și LOCATIE (la ce locații se află sediile). Relația are cardinalitatea minimă 0:1 (un sediu trebuie să se afle la o locație dar la o locație nu trebuie să se afle un sediu) și cardinalitatea maximă 1:1 (un sediu poate să se afle doar la o locație și la o locație poate să se afle doar un sediu).

BANCOMAT_se_afla_la_LOCATIE = relația dintre entitățile BANCOMAT și LOCATIE (la ce locații se află bancomatele). Relația are cardinalitatea minimă 0:1 (un bancomat trebuie să se afle la o locație dar la o locație nu trebuie să se afle un bancomat) și cardinalitatea maximă 1:1 (un bancomat poate să se afle doar la o locație și la o locație poate să se afle doar un bancomat).

LOCATIE_se_afla_in_TARA = relația dintre entitățile LOCATIE și TARA (în ce țară se află o locație). Relația are cardinalitatea minimă 1:0 și cardinalitatea maximă 1:n.

PERSOANA_are_domiciliul_la_LOCATIE = relația dintre entitățile PERSOANA și LOCAȚIE (care este domiciliul unei persoane). Relația are cardinalitatea minimă 0:1 și cardinalitatea maximă n:1.

CONT_are_asociat_CONT_ECONOMII = relația dintre entitățile CONT și CONT_ECONOMII (care este contul de economii asociat unui cont). Relația are cardinalitatea minimă cât și maximă 1:1, un cont trebuie și poate să aibă doar un cont de economii și vice versa.

DEPOZIT_ECONOMII_se_face_in_CONT_ECONOMII = relația dintre entitățile DEPOZIT_ECONOMII și CONT_ECONOMII (care sunt depozitele de bani făcute în vederea obținerii unei dobânzi). Cardinalitatea minimă este 0:1 iar cea maximă este n:1.

DEPOZIT_ECONOMII_se_face_cu_DOBANDA = relația dintre DEPOZIT_ECONOMII și DOBANDA (care sunt dobânzile asociate depozitelor de economii). Cardinalitatea minimă este 0:1 iar cea maximă este n:1.

CONT_are_asociat_CARD = relația dintre CONT și CARD (care sunt cardurile asociate conturilor). Cardinalitatea minimă este 1:1 și cardinalitatea maximă 1:n.

CONT_are_ISTORIC = relația dintre CONT și ISTORIC (leagă contul propriu zis de istoricul de tranzacții pe care acesta îl are). Cardinalitatea minimă cât și maximă este 1:1.

ANGAJAT_efectueaza_TRANZACTIE_EXTERNA = relația dintre ANGAJAT și TRANZACTIE_EXTERNA (ce angajați efectuează tranzacțiile externe administrative). Cardinalitatea minimă este 1:0 iar cea maximă este 1:n.

ISTORIC_contine_TRANZACTIE_EXTERNA = relația dintre ISTORIC și TRANZACTIE_EXTERNA (de ce istoric, implicit cont căruia i se aplică, sunt conținute tranzacțiile externe). Relația are cardinalitatea minimă 0:1 și cardinalitatea maximă n:1.

ISTORIC_contine_TRANZACTIE = relația dintre ISTORIC și TRANZACTIE (care sunt tranzacțiile efectuate în contul de care aparține istoricul). Relația are cardinalitatea minimă 1:0 și cardinalitatea maximă 1:n.

TRANSFER_PRIMIT_primit_de_la_CONT = relația dintre TRANSFER_PRIMIT și CONT (identificarea transferurilor primite și a conturilor de unde provin banii). Relația are cardinalitatea minimă 0:1 și cardinalitatea maximă n:1.

TRANSFER_TRIMIS Trimis_catre_CONT = relația dintre TRANSFER_TRIMIS și CONT (identificarea transferurilor trimise și a conturilor catre care au fost trimiși banii). Relația are cardinalitatea minimă 0:1 și cardinalitatea maximă n:1.

DEPOZIT_se_realizeaza_cu_CARD = relația dintre DEPOZIT și CARD (care carduri realizează depozitele). Relația are cardinalitatea minimă 0:1 și cardinalitatea maximă n:1.

RETRAGERE_se_realizeaza_cu_CARD = relația dintre RETRAGERE și CARD (care carduri realizează retragerile). Relația are cardinalitatea minimă 0:1 și cardinalitatea maximă n:1.

DEPOZIT_se_realizeaza_la_BANCOMAT = relația dintre DEPOZIT și BANCOMAT (la care bancomante se realizează depozitele). Relația are cardinalitatea minimă 0:1 și cardinalitatea maximă n:1.

RETRAGERE_se_realizeaza_la_BANCOMAT = relația dintre RETRAGERE și BANCOMAT (la care bancomante se realizează retragerile). Relația are cardinalitatea minimă 0:1 și cardinalitatea maximă n:1.

CLIENT_semneaza_un_contract_TIP_CONTRACT_la_SEDIU_cu_un_ANGAJAT = relație de tip 3 ce leagă CLIENT, SEDIU, TIP_CONTRACT și ANGAJAT, reflectând care client a semnat contractul, ce tip de contract a semnat, unde l-a semnat și care angajat a aprobat contractul. Denumirea acestei relații va fi semneaza_un_contract.

3.5. Descrierea atributelor, incluzând tipul de date și eventualele constrângeri.

Entitatea independentă PERSOANA are ca atrbute:

cod_persoana = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare în sistem al unei persoane.

cnp = variabilă de tip caracter, de lungime maximă 20, care reprezintă codul numeric personal.

nume = variabilă de tip caracter, de lungime maximă 25, care reprezintă numele persoanei.

prenume = variabilă de tip caracter, de lungime maximă 25, care reprezintă prenumele persoanei.

email = variabilă de tip caracter, de lungime maximă 30, care reprezintă email-ul de contact al persoanei.

data_nastere = variabilă de tip dată calendaristică, care reprezintă data nașterii persoanei.

gen = variabilă de tip caracter, luând valorile m sau f, de lungime 1, care reprezintă genul persoanei, dacă persoana dorește să se abțină din a comunica această informație, genul rămâne NULL.

cod_locatie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul localizării persoanei (domiciliul). Atributul trebuie să corespundă la o valoare a cheii primare din tabelul LOCATIE.

Subentitatea CLIENT are ca atribute:

cod_persoana = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare în sistem al clientului.

tag_utilizator = variabilă de tip caracter, de lungime maximă 25, care reprezintă o poreclă ce poate fi folosită pentru identificarea unui client (exemple: dragos12, mihaiA33, etc).

Subentitatea ANGAJAT are ca atribute:

cod_persoana = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare în sistem al angajatului.

salariu = variabilă de tip numeric (real) de lungime maximă 15 dintre care 2 zecimale, care reprezintă salariul lunar al angajatului.

cod_sediu = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare al sediului la care lucrează angajatul, dacă angajatul lucrează remote, aceasta va fi NULL. Atributul, dacă nu este NULL, trebuie să corespundă la o valoare a cheii primare din tabelul SEDIU.

cod_job = variabilă de tip caracter de lungime maximă 25, care reprezintă codul jobului pe care angajatul îl are. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul JOB.

Entitatea independentă DOBANDA are ca atribute:

cod_dobanda = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare al unei dobânzi.

procent = variabilă de tip numeric (real) de lungime maximă 3, dintre care 2 zecimale, care reprezintă procentul de bani din suma depozitată pe care îl obții la revendicarea depozitului.

durata_luni = variabilă de tip întreg de lungime maximă 3, care reprezintă numărul de luni în care banii depozitați vor fi blocați.

Entitatea independentă CONT are ca atribute:

cod_cont = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare al unui cont.

sold = variabilă de tip numeric (real) de lungime maximă 15 dintre care 2 zecimale, care

reprezintă suma de bani aflată în cont.

cod_persoana = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul clientului de care aparține contul. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CLIENT.

IBAN = variabilă de tip caracter, de lungime maximă 25, care reprezintă un cod implementat pentru facilitarea plășilor, folosit la identificarea beneficiarului unei plăști.

data_creare = variabilă de tip dată calendaristică, care reprezintă data la care a fost creat contul.

Entitatea dependentă CONT_ECONOMII de CONT are ca atribute:

cod_cont = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare al unui cont. Se folosește același cod care este cheie primară al entității CONT deoarece este garantat că există un unic cont asociat unui cont de economii.

depozite_contor = variabilă de tip numeric (real) de lungime maximă 15, care reprezintă numărul depozitelor create într-un cont de economii. Folosit ca generator de id-uri pentru depozite, care sunt dependente de contul de economii de care aparțin.

Entitatea dependentă DEPOZIT_ECONOMII de CONT_ECONOMII are ca atribute:

cod_depozit_economii = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare al unui depozit de economii.

cod_cont = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare al contului de care aparține depozitul. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CONT_ECONOMII.

cod_dobanda = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare al dobânzii cu care s-a făcut depozitul. În cazul în care dobânda este eliminată și acest atribut devine NULL, banii pot fi retrăși oricând. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul DOBANDA.

data_start = variabilă de tip dată calendaristică, care reprezintă data la care s-a făcut depozitul

data_sfarsit = variabilă de tip dată calendaristică, care reprezintă data de la care depozitul, împreună cu procentul de dobândă, poate fi revendicat. Deși acest atribut este redundant deoarece poate fi calculat folosindu-ne de data_start și dobânda asociată, acesta reprezintă o redundanță utilă, deoarece acesta este un atribut accesat frecvent, care nu ar trebui să fie calculat de fiecare dată.

valoare = variabilă de tip numeric (real) de lungime maximă 15 dintre care 2 zecimale, care reprezintă suma de bani care a fost depozitată. Are o limită inferioară de 100.

revendicat = variabilă de tip numeric (real) de lungime 1, care poate lua doar valorile 0 și 1. Dacă revendicat este 0 înseamnă că acest depozit nu a fost revendicat încă, altfel, depozitul a fost deja revendicat.

Entitatea independentă JOB are ca atribute:

cod_job = variabilă de tip caracter de lungime maximă 25, care reprezintă codul de identificare al unui job.

denumire_job = variabilă de tip caracter de lungime maximă 25, care reprezintă denumirea jobului.

salariu_minim = variabilă de tip numeric (real) de lungime maximă 15 dintre care 2 zecimale, care reprezintă suma de bani minimă pe care trebuie să o aibă un salar de angajat care are acest job.

salariu_maxim = variabilă de tip numeric (real) de lungime maximă 15 dintre care 2 zecimale, care reprezintă suma de bani maximă pe care o poate avea un salar de angajat care are acest job.

Entitatea independentă SEDIU are ca atribute:

cod_sediu = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare al sediului.

cod_locatie = variabilă de tip întreg de lungime maximă 5, care reprezintă codul locației la care se află sediul. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul LOCATIE.

Entitatea independentă BANCOMAT are ca atribute:

cod_bancomat = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare al bancomatului.

cod_locatie = variabilă de tip întreg de lungime maximă 5, care reprezintă codul locației la care se află bancomatul. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul LOCATIE.

Entitatea independentă LOCATIE are ca atribute:

cod_locatie = variabilă de tip întreg de lungime maximă 5, care reprezintă codul de identificare al locației.

cod_postal = variabilă de tip caracter de lungime maximă 15, care reprezintă codul poștal al locației.

adresa = variabilă de tip caracter de lungime maximă 40, care este alcătuită din numele străzii, numărul, (eventual și blocul, scara, etajul și apartamentul).

oras = variabilă de tip caracter de lungime maximă 30, care reprezintă orașul în care se află locația.

cod_tara = variabilă de tip număr întreg de lungime maximă 5, care reprezintă id-ul țării în care se află locația. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul TARA.

Entitatea independentă TARA are ca atribute:

cod_tara = variabilă de tip număr întreg de lungime maximă 5, care reprezintă id-ul țării.

nume_tara = variabilă de tip caracter de lungime maximă 25, care reprezintă denumirea țării.

Entitatea dependentă ISTORIC de CONT are ca atribute:

cod_cont = variabilă de tip număr întreg de lungime maximă 5. Se folosește același cod care este cheie primară al entității CONT deoarece este garantat că există un unic cont asociat unui istoric.

tranzactii_contor = variabilă de tip numeric (real) de lungime maximă 15, care reprezintă numărul tranzacțiilor efectuate într-un cont. Folosit ca generator de id-uri pentru tranzacții, care sunt dependente de contul de care aparțin.

Entitatea dependenta TRANZACTIE_EXTERNA de ANGAJAT și ISTORIC are ca atribute:

cod_persoana = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul angajatului care efectuează tranzacția externă.

cod_cont = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul contului în al cărui istoric este adăugată tranzacția externă.

valoare = variabilă de tip numeric (real) de lungime maximă 15 dintre care 2 zecimale, care reprezintă suma de bani care a fost tranzacționată.

data_tranzactie = variabilă de tip dată calendaristică, care reprezintă data la care a avut loc tranzacția.

mesaj_administrativ = variabilă de tip TEXT care conține un mesaj din partea angajatului care a efectuat tranzacția, NULL în cazul în care nu există un mesaj suplimentar.

Entitatea independentă CARD are ca atribute:

cod_card = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul de identificare al cardului.

pin = variabilă de tip număr întreg de lungime 4, care reprezintă un cod special folosit pentru aprobarea unor tranzacții.

tip = variabilă de tip caracter cu lungime maximă de 10, care reprezintă tipul cardului și poate avea valorile ‘visa’ sau ‘mastercard’.

card_number = variabilă de tip caracter de lungime maximă 20, care reprezintă un număr de identificare al cardului accesibil clientului.

cod_cont = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul contului la care este asociat. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CONT.

Entitatea dependentă TRANZACTIE de ISTORIC are ca atribute:

cod_tranzactie = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul de identificare al tranzacției.

cod_cont = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul contului de care aparține istoricul din care face parte tranzacția.

valoare = variabilă de tip numeric (real) de lungime maximă 15 dintre care 2 zecimale, care reprezintă suma de bani care a fost tranzacționată.

data_tranzactie = variabilă de tip dată calendaristică, care reprezintă data la care a avut loc tranzacția.

tip_tranzactie = variabilă de tip caracter de lungime maximă 20, care reprezintă tipul tranzacției. Acest atribut poate lua valori precum ‘depozit’, ‘retragere’, ‘transfer_primit’, ‘transfer Trimis’.

Subentitatea DEPOZIT are ca atribute:

cod_tranzactie = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul de identificare al depozitului.

cod_cont = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul contului de care aparține istoricul din care face parte depozitul.

cod_bancomat = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul bancomatului la care s-a realizat depozitul.

cod_card = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul codul cardului cu care s-a realizat depozitul.

Subentitatea RETRAGERE are ca atribute:

cod_tranzactie = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul de identificare al retragerii.

cod_cont = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul contului de care aparține istoricul din care face parte retragerea.

cod_bancomat = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul bancomatului la care s-a realizat retragerea.

cod_card = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul codul cardului cu care s-a realizat retragerea.

Subentitatea TRANSFER_TRIMIS are ca atribute:

cod_tranzactie = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul de identificare al retragerii.

cod_cont = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul contului sursă, din care se iau banii pentru transfer.

cod_cont_destinatie = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul contului în care ajung banii din transfer.

Subentitatea TRANSFER_PRIMIT are ca atribute:

cod_tranzactie = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul de identificare al retragerii.

cod_cont_sursa = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul contului sursă, din care se iau banii pentru transfer.

cod_cont = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul contului în care ajung banii din transfer.

Entitatea independentă TIP_CONTRACT are ca atribute:

tip_contract = variabilă de tip caracter de lungime maximă 20, care reprezintă tipul contractului.

titlu_contract = variabilă de tip caracter de lungime maximă 40, care reprezintă titlul contractului.

continut = variabilă de tip TEXT, care reprezintă conținutul contractului (reguli, prevederi, abateri, responsabilități, etc).

Relația SEMNEAZA_UN_CONTRACT are ca atribute:

cod_angajat = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul de identificare al angajatului care aproba contractul. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul ANGAJAT.

cod_client = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul de identificare al clientului care semnează contractul de utilizator. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CLIENT.

tip_contract = variabilă de tip caracter de lungime maximă 20, care reprezintă tipul contractului semnat. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul TIP_CONTRACT.

cod_sediu = variabilă de tip număr întreg de lungime maximă 5, care reprezintă codul de identificare al sediului la care a fost semnat contractul, NULL în cazul în care acesta a fost completat online. Atributul, dacă nu este NULL, trebuie să corespundă la o valoare a cheii primare din tabelul SEDIU.

data_contract = variabilă de tip dată calendaristică, care reprezintă data la care a fost semnat și aprobat contractul.

4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).

--4. Implementati in Oracle diagrama conceptuala realizata:

- definiți toate tabelele, implementand toate constrângerile de integritate
- necesare (chei primare, chei externe etc).

```
DROP SEQUENCE generator_cod_persoana;
DROP SEQUENCE generator_cod_cont;
DROP SEQUENCE generator_cod_sediu;
DROP SEQUENCE generator_cod_locatie;
DROP SEQUENCE generator_cod_tara;
DROP SEQUENCE generator_cod_dobanda;
DROP SEQUENCE generator_cod_bancomat;
DROP SEQUENCE generator_cod_card;
DROP SEQUENCE generator_card_default_pin;
```

```
DROP TABLE TARA CASCADE CONSTRAINTS;
DROP TABLE LOCATIE CASCADE CONSTRAINTS;
DROP TABLE SEDIU CASCADE CONSTRAINTS;
DROP TABLE JOB CASCADE CONSTRAINTS;
DROP TABLE TIP_CONTRACT CASCADE CONSTRAINTS;
DROP TABLE DOBANDA CASCADE CONSTRAINTS;
DROP TABLE BANCOMAT CASCADE CONSTRAINTS;
DROP TABLE PERSOANA CASCADE CONSTRAINTS;
```

```

DROP TABLE CLIENT CASCADE CONSTRAINTS;
DROP TABLE ANGAJAT CASCADE CONSTRAINTS;
DROP TABLE CONT CASCADE CONSTRAINTS;
DROP TABLE CONT_ECONOMII CASCADE CONSTRAINTS;
DROP TABLE DEPOZIT_ECONOMII CASCADE CONSTRAINTS;
DROP TABLE CARD CASCADE CONSTRAINTS;
DROP TABLE ISTORIC CASCADE CONSTRAINTS;
DROP TABLE TRANZACTIE_EXTERNA CASCADE CONSTRAINTS;
DROP TABLE TRANZACTIE CASCADE CONSTRAINTS;
DROP TABLE SEMNEAZA_UN_CONTRACT CASCADE CONSTRAINTS;
DROP TABLE TRANSFER_TRIMIS CASCADE CONSTRAINTS;
DROP TABLE TRANSFER_PRIMIT CASCADE CONSTRAINTS;
DROP TABLE DEPOZIT CASCADE CONSTRAINTS;
DROP TABLE RETRAGERE CASCADE CONSTRAINTS;

CREATE SEQUENCE generator_cod_persoana NOCACHE;
CREATE SEQUENCE generator_cod_cont NOCACHE;
CREATE SEQUENCE generator_cod_sediu NOCACHE;
CREATE SEQUENCE generator_cod_locatie NOCACHE;
CREATE SEQUENCE generator_cod_tara NOCACHE;
CREATE SEQUENCE generator_cod_dobanda NOCACHE;
CREATE SEQUENCE generator_cod_bancomat NOCACHE;
CREATE SEQUENCE generator_cod_card NOCACHE;
CREATE SEQUENCE generator_card_default_pin MINVALUE 1000 MAXVALUE 9999
CYCLE NOCACHE;

CREATE TABLE TARA(
    cod_tara NUMBER(5) PRIMARY KEY,
    nume_tara VARCHAR(25) NOT NULL UNIQUE
);

CREATE TABLE LOCATIE(
    cod_locatie NUMBER(5) PRIMARY KEY,
    cod_postal VARCHAR(15) NOT NULL,
    adresa VARCHAR(40) NOT NULL,
    oras VARCHAR(30) NOT NULL,
    cod_tara NUMBER(5) REFERENCES TARA(cod_tara) ON DELETE SET NULL
);

CREATE TABLE SEDIU(
    cod_sediu NUMBER(5) PRIMARY KEY,
    cod_locatie NUMBER(5) REFERENCES LOCATIE(cod_locatie) ON DELETE SET NULL
);

CREATE TABLE JOB(
    cod_job VARCHAR(25) PRIMARY KEY,
    denumire_job VARCHAR(25) NOT NULL,
    salariu_minim NUMBER(15, 2) NOT NULL CHECK(salariu_minim>0),
);

```

```

        salariu_maxim NUMBER(15, 2) NOT NULL CHECK(salariu_maxim>0)
);

CREATE TABLE TIP_CONTRACT(
    tip_contract VARCHAR(20) PRIMARY KEY,
    titlu_contract VARCHAR(40) NOT NULL,
    continut CLOB NOT NULL
);

CREATE TABLE DOBANDA(
    cod_dobanda NUMBER(5) PRIMARY KEY,
    procent NUMBER(3,2) CHECK (procent>0 AND procent<=1) NOT NULL,
    durata_luni NUMBER(3) NOT NULL
);

CREATE TABLE BANCOMAT(
    cod_bancomat NUMBER(5) PRIMARY KEY,
    cod_locatie NUMBER(5) REFERENCES LOCATIE(cod_locatie) ON DELETE SET
NULL
);

CREATE TABLE PERSOANA(
    cod_persoana NUMBER(5) PRIMARY KEY,
    cnp VARCHAR(20) UNIQUE,
    nume VARCHAR(25) NOT NULL,
    prenume VARCHAR(25) NOT NULL,
    email VARCHAR(30) UNIQUE,
    data_nastere DATE NOT NULL,
    gen CHAR(1),
    cod_locatie NUMBER(5) REFERENCES LOCATIE(cod_locatie) ON DELETE SET
NULL
);

CREATE TABLE CLIENT(
    cod_persoana NUMBER(5) PRIMARY KEY REFERENCES
PERSOANA(cod_persoana),
    tag_utilizator VARCHAR(25) NOT NULL UNIQUE
);

CREATE TABLE ANGAJAT(
    cod_persoana NUMBER(5) PRIMARY KEY REFERENCES
PERSOANA(cod_persoana),
    salariu NUMBER(15, 2) NOT NULL CHECK (salariu>0),
    cod_sediu NUMBER(5) REFERENCES SEDIU(cod_sediu) ON DELETE SET NULL,
    cod_job VARCHAR(25) REFERENCES JOB(cod_job) ON DELETE SET NULL
);

```

);

```
CREATE TABLE CONT(
    cod_cont NUMBER(5) PRIMARY KEY,
    cod_persoana NUMBER(5) NOT NULL REFERENCES CLIENT(cod_persoana) ON
DELETE CASCADE,
    sold NUMBER(15, 2) DEFAULT 0 NOT NULL,
    IBAN VARCHAR(25) NOT NULL UNIQUE
);
```

```
CREATE TABLE CONT_ECONOMII(
    cod_cont NUMBER(5) PRIMARY KEY REFERENCES CONT(cod_cont) ON
DELETE CASCADE,
    depozite_contor NUMBER(15) DEFAULT 0 NOT NULL
);
```

```
CREATE TABLE DEPOZIT_ECONOMII(
    cod_depozit_economii NUMBER(5),
    cod_cont NUMBER(5) REFERENCES CONT_ECONOMII(cod_cont) ON DELETE
CASCADE,
    cod_dobanda NUMBER(5) REFERENCES DOBANDA(cod_dobanda) NOT NULL,
    data_start DATE NOT NULL,
    data_sfarsit DATE NOT NULL,
    valoare NUMBER(15, 2) CHECK (valoare >= 100),
    revendicat NUMBER(1) DEFAULT 0 CHECK (revendicat = 0 or revendicat = 1),
    PRIMARY KEY(cod_depozit_economii, cod_cont)
);
```

```
CREATE TABLE CARD(
    cod_card NUMBER(5) PRIMARY KEY,
    pin NUMBER(4) CHECK (pin>999) NOT NULL,
    tip VARCHAR(10) NOT NULL,
    card_number VARCHAR(20) NOT NULL UNIQUE,
    cod_cont NUMBER(5) REFERENCES CONT(cod_cont) ON DELETE SET NULL
);
```

```
CREATE TABLE ISTORIC(
    cod_cont NUMBER(5) PRIMARY KEY REFERENCES CONT(cod_cont) ON
DELETE CASCADE,
    tranzactii_contor NUMBER(15) DEFAULT 0 NOT NULL
);
```

```
CREATE TABLE TRANZACTIE_EXTERNA(
    cod_persoana NUMBER(5) REFERENCES PERSOANA(cod_persoana),
    cod_cont NUMBER(5) REFERENCES ISTORIC(cod_cont) ON DELETE CASCADE,
```

```
    valoare NUMBER(15, 2) NOT NULL,  
    data_tranzactie DATE,  
    mesaj_administrativ VARCHAR2(4000),  
    PRIMARY KEY(cod_persoana, cod_cont, data_tranzactie)  
);
```

```
CREATE TABLE TRANZACTIE(  
    cod_tranzactie NUMBER(5),  
    cod_cont NUMBER(5) REFERENCES ISTORIC(cod_cont) ON DELETE CASCADE,  
    valoare NUMBER(15, 2) NOT NULL,  
    data_tranzactie DATE NOT NULL,  
    tip_tranzactie VARCHAR(20) NOT NULL,  
    PRIMARY KEY(cod_tranzactie, cod_cont)  
);
```

```
CREATE TABLE DEPOZIT(  
    cod_tranzactie NUMBER(5),  
    cod_cont NUMBER(5),  
    cod_bancomat NUMBER(5) REFERENCES BANCOMAT(cod_bancomat) ON  
DELETE SET NULL,  
    cod_card NUMBER(5) REFERENCES CARD(cod_card) ON DELETE SET NULL,  
    PRIMARY KEY(cod_tranzactie, cod_cont),  
    FOREIGN KEY(cod_tranzactie, cod_cont) REFERENCES  
TRANZACTIE(cod_tranzactie, cod_cont) ON DELETE CASCADE  
);
```

```
CREATE TABLE RETRAGERE(  
    cod_tranzactie NUMBER(5),  
    cod_cont NUMBER(5),  
    cod_bancomat NUMBER(5) REFERENCES BANCOMAT(cod_bancomat) ON  
DELETE SET NULL,  
    cod_card NUMBER(5) REFERENCES CARD(cod_card) ON DELETE SET NULL,  
    PRIMARY KEY(cod_tranzactie, cod_cont),  
    FOREIGN KEY(cod_tranzactie, cod_cont) REFERENCES  
TRANZACTIE(cod_tranzactie, cod_cont) ON DELETE CASCADE  
);
```

```
CREATE TABLE TRANSFER_PRIMIT(  
    cod_tranzactie NUMBER(5),  
    cod_cont NUMBER(5),  
    cod_cont_sursa REFERENCES CONT(cod_cont) ON DELETE SET NULL,  
    PRIMARY KEY(cod_tranzactie, cod_cont),  
    FOREIGN KEY(cod_tranzactie, cod_cont) REFERENCES  
TRANZACTIE(cod_tranzactie, cod_cont) ON DELETE CASCADE  
);
```

```

CREATE TABLE TRANSFER_TRIMIS(
    cod_tranzactie NUMBER(5),
    cod_cont NUMBER(5),
    cod_cont_destinatie REFERENCES CONT(cod_cont) ON DELETE SET NULL,
    PRIMARY KEY(cod_tranzactie, cod_cont),
    FOREIGN KEY(cod_tranzactie, cod_cont) REFERENCES
    TRANZACTIE(cod_tranzactie, cod_cont) ON DELETE CASCADE
);

```

```

CREATE TABLE SEMNEAZA_UN_CONTRACT(
    cod_angajat NUMBER(5) REFERENCES ANGAJAT(cod_persoana),
    cod_client NUMBER(5) REFERENCES CLIENT(cod_persoana),
    tip_contract VARCHAR(20) REFERENCES TIP_CONTRACT(tip_contract),
    cod_sediu NUMBER(5),
    PRIMARY KEY(cod_angajat, cod_client, tip_contract)
);

```

--cativa triggeri pentru a automatiza generarea cheilor primare

```

CREATE OR REPLACE TRIGGER genereaza_pk_tara
BEFORE INSERT ON tara
FOR EACH ROW
BEGIN
    IF :NEW.cod_tara IS NULL THEN
        :NEW.cod_tara := generator_cod_tara.nextval;
    END IF;
END;
/

```

```

CREATE OR REPLACE TRIGGER genereaza_pk_locatie
BEFORE INSERT ON locatie
FOR EACH ROW
BEGIN
    IF :NEW.cod_locatie IS NULL THEN
        :NEW.cod_locatie := generator_cod_locatie.nextval;
    END IF;
END;
/

```

```

CREATE OR REPLACE TRIGGER genereaza_pk_sediu
BEFORE INSERT ON sediu
FOR EACH ROW
BEGIN
    IF :NEW.cod_sediu IS NULL THEN
        :NEW.cod_sediu := generator_cod_sediu.nextval;
    END IF;
END;
/

```

```

CREATE OR REPLACE TRIGGER genereaza_pk_dobanda

```

```

BEFORE INSERT ON dobanda
FOR EACH ROW
BEGIN
    IF :NEW.cod_dobanda IS NULL THEN
        :NEW.cod_dobanda := generator_cod_dobanda.nextval;
    END IF;
END;
/

CREATE OR REPLACE TRIGGER genereaza_pk_bancomat
BEFORE INSERT ON bancomat
FOR EACH ROW
BEGIN
    IF :NEW.cod_bancomat IS NULL THEN
        :NEW.cod_bancomat := generator_cod_bancomat.nextval;
    END IF;
END;
/

CREATE OR REPLACE TRIGGER genereaza_pk_persoana
BEFORE INSERT ON persoana
FOR EACH ROW
BEGIN
    IF :NEW.cod_persoana IS NULL THEN
        :NEW.cod_persoana := generator_cod_persoana.nextval;
    END IF;
END;
/

CREATE OR REPLACE TRIGGER genereaza_pk_cont
BEFORE INSERT ON cont
FOR EACH ROW
BEGIN
    IF :NEW.cod_cont IS NULL THEN
        :NEW.cod_cont := generator_cod_cont.nextval;
    END IF;
END;
/

CREATE OR REPLACE TRIGGER genereaza_pk_card
BEFORE INSERT ON card
FOR EACH ROW
BEGIN
    IF :NEW.cod_card IS NULL THEN
        :NEW.cod_card := generator_cod_card.nextval;
    END IF;
END;
/

```

```

CREATE OR REPLACE TRIGGER genereaza_pk_depozit_economii
BEFORE INSERT ON depozit_economii
FOR EACH ROW
BEGIN
    IF :NEW.cod_cont IS NULL THEN RAISE_APPLICATION_ERROR(-20001, 'O
inserare de depozit de economii necesita un cont de economii in care sa se realizeze.');
    END IF;

    SELECT depozite_contor+1 INTO :NEW.cod_depozit_economii FROM
CONT_ECONOMII c WHERE :NEW.cod_cont = c.cod_cont;
    UPDATE CONT_ECONOMII c SET depozite_contor = depozite_contor + 1 WHERE
c.cod_cont = :NEW.cod_cont;
EXCEPTION
    WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR(-20002, 'Nu se
poate insera un depozit pentru un cont inexistent');
    --nu poate da TOO_MANY_ROWS, se cauta dupa cheia primara
    WHEN OTHERS THEN
        RAISE;
END;
/

CREATE OR REPLACE TRIGGER actualizeaza_tranzactii_contor
AFTER INSERT ON tranzactie
FOR EACH ROW
BEGIN
    UPDATE ISTORIC i SET tranzactii_contor = tranzactii_contor + 1 WHERE i.cod_cont
= :NEW.cod_cont;
END;
/

CREATE OR REPLACE TRIGGER insereaza_istoric_la_cont
AFTER INSERT ON cont
FOR EACH ROW
BEGIN
    INSERT INTO istoric(cod_cont) VALUES(:NEW.cod_cont);
END;
/

CREATE OR REPLACE TRIGGER insereaza_economii_la_cont
AFTER INSERT ON cont
FOR EACH ROW
BEGIN
    INSERT INTO cont_economii(cod_cont) VALUES(:NEW.cod_cont);
END;
/

CREATE OR REPLACE TRIGGER genereaza_pin_card
BEFORE INSERT ON card
FOR EACH ROW

```

```
BEGIN
  IF :NEW.pin IS NULL THEN
    :NEW.pin := generator_card_default_pin.nextval;
  END IF;
END;
/
```

```
328 | AFTER INSERT ON cont
329 | FOR EACH ROW
330 | BEGIN
331 |   INSERT INTO istoric(cod_cont) VALUES(:NEW.cod_cont);
332 | END;
333 |
334 |
335 | CREATE OR REPLACE TRIGGER insereaza_economii_la_cont
336 | AFTER INSERT ON cont
337 | FOR EACH ROW
338 | BEGIN
339 |   INSERT INTO cont_economii(cod_cont) VALUES(:NEW.cod_cont);
340 | END;
341 |
342 |
343 | CREATE OR REPLACE TRIGGER genereaza_pin_card
344 | BEFORE INSERT ON card
345 | FOR EACH ROW
346 | BEGIN
347 |   IF :NEW.pin IS NULL THEN
348 |     :NEW.pin := generator_card_default_pin.nextval;
349 |   END IF;
350 | END;
351 |
352 | /
```

Script Output      | Task completed in 0.79 seconds

Trigger INSEREAZA_ECONOMII_LA_CONT compiled

Trigger GENEREAZA_PIN_CARD compiled

5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

- 5. Adaugati informatii coerente in tabelele create
-- (minim 5 înregistrari pentru fiecare entitate independenta;
-- minim 10 înregistrari pentru tabela asociativa).

INSERT ALL

```
INTO TARA(nume_tara) VALUES('Romania')
INTO TARA(nume_tara) VALUES('Olanda')
INTO TARA(nume_tara) VALUES('Bulgaria')
INTO TARA(nume_tara) VALUES('Spania')
INTO TARA(nume_tara) VALUES('Rusia')
INTO TARA(nume_tara) VALUES('Elvetia')
INTO TARA(nume_tara) VALUES('Finlanda')
INTO TARA(nume_tara) VALUES('Suedia')
INTO TARA(nume_tara) VALUES('Grecia')
INTO TARA(nume_tara) VALUES('Japonia')
SELECT 1 FROM DUAL;
```

INSERT ALL

```
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('2270', 'Str Artar Nr 12',
'Bucuresti', 1)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('11199', 'Str Macin Nr
22', 'Cluj', 1)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('22222', 'Str Green Nr 1',
'Amsterdam', 2)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('93123', 'Str Goodfellow
Nr 5', 'Haga', 2)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('6677', 'Str Bujha Nr 9',
'Sofia', 3)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('9221', 'Str Ehia Nr 29',
'Vidin', 3)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('1122', 'Str Ernesto Nr 8
Bl A2 Sc 1 Et 3 Ap 10', 'Madrid', 4)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('8890', 'Str Ruski Nr 10
Bl F3 Sc 2 Et 1 Ap 4', 'Moscova', 5)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('1000', 'Str Kijrsi Nr 9 Bl
C2 Sc 1 Et 2 Ap 7', 'Moscova', 5)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('3216', 'Str Etiop Nr 12
Bl A1 Sc 4 Et 5 Ap 32', 'Geneva', 6)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('2839', 'Str Yaht Nr 89',
'Atena', 9)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('1909', 'Str Kohli Nr 54',
'Corfu', 9)
INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('5750', 'Str Juim Nr 72 Bl
```

```

D3 Sc 2 Et 1 Ap 2', 'Tokyo', 10)
    INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('2111', 'Str Fukushi Nr 78
B1 B3 Sc 3 Et 4 Ap 22', 'Hokkaido', 10)
    INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('233111', 'Str Iancului Nr
122', 'Suceava', 1)
    INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('28765', 'Str Armeni Nr
36', 'Drobeta-Turnu-Severin', 1)
    INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('8955', 'Str Maresal Nr
81', 'Targu-Jiu', 1)
    INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('29464', 'Str Stoica Nr
77', 'Sibiu', 1)
    INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('6894', 'Str Plopilor Nr
90', 'Brasov', 1)
    INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('46948', 'Str Cicero Nr
25', 'Drobeta-Turnu-Severin', 1)
    INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('44766', 'Str Titeica Nr
17', 'Sibiu', 1)
    INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('40099', 'Str Sadoveanu
Nr 89', 'Bucuresti', 1)
    INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('10200', 'Str Alias Nr 98',
'Cluj', 1)
    INTO LOCATIE(cod_postal, adresa, oras, cod_tara) VALUES('100229', 'Str Mesterului
Nr 4', 'Bucuresti', 1)
SELECT 1 FROM DUAL;

```

INSERT ALL

```

    INTO SEDIU(cod_locatie) VALUES(1)
    INTO SEDIU(cod_locatie) VALUES(2)
    INTO SEDIU(cod_locatie) VALUES(4)
    INTO SEDIU(cod_locatie) VALUES(5)
    INTO SEDIU(cod_locatie) VALUES(6)
SELECT 1 FROM DUAL;

```

INSERT ALL

```

    INTO JOB VALUES('MANAGER', 'Manager', 4000, 12000)
    INTO JOB VALUES('ADMIN', 'Administrator', 2000, 8000)
    INTO JOB VALUES('MOD', 'Moderator', 1500, 4000)
    INTO JOB VALUES('CASIER', 'Casier', 1300, 2000)
    INTO JOB VALUES('OP_CALL', 'Operator CallCenter', 1500, 3000)
SELECT 1 FROM DUAL;

```

INSERT ALL

```

    INTO TIP_CONTRACT VALUES('TERMS', 'Termini si Conditii', 'Sa fii cuminte, sa nu
faci evaziune fiscală, sa nu faci tranzactii dubioase, sa respecti regulile de bun simt si
angajatii cu care interactionezi... etc etc')

```

```

    INTO TIP_CONTRACT VALUES('PRIVACY', 'Politica de Confidentialitate', 'Nu
spunem, nu facem, nu transmitem decat daca te prinde ANAF-ul, ai tag de utilizator,
numele nu se comunica public, totu privat si sigur... etc etc')

```

```

    INTO TIP_CONTRACT VALUES('ETHIC', 'Politica de Etica BUSINESS', 'Daca vinzi

```

produse fa-o legal, daca te da cineva in judecata noi nu ne bagam, birocratie birocratie etc etc')

INTO TIP_CONTRACT VALUES('ECONOMII', 'Politica de Economii', 'Ajuta sistemul sa aiba o cantitate constanta de bani si primeste o dobanda pentru buna ta vointa, etc etc...')
SELECT 1 FROM DUAL;

INSERT ALL

INTO DOBANDA(procent, durata_luni) VALUES(0.1, 3)
INTO DOBANDA(procent, durata_luni) VALUES(0.3, 6)
INTO DOBANDA(procent, durata_luni) VALUES(0.7, 12)
INTO DOBANDA(procent, durata_luni) VALUES(0.15, 24)
INTO DOBANDA(procent, durata_luni) VALUES(0.35, 48)

SELECT 1 FROM DUAL;

INSERT ALL

INTO BANCOMAT(cod_locatie) VALUES(1)
INTO BANCOMAT(cod_locatie) VALUES(2)
INTO BANCOMAT(cod_locatie) VALUES(4)
INTO BANCOMAT(cod_locatie) VALUES(5)
INTO BANCOMAT(cod_locatie) VALUES(6)
INTO BANCOMAT(cod_locatie) VALUES(11)
INTO BANCOMAT(cod_locatie) VALUES(12)

SELECT 1 FROM DUAL;

INSERT ALL

INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('2870816526255', 'Pintenaru', 'Melissa', 'melissa.p@gmail.com',
TO_DATE('16-08-1987', 'DD-MM-YYYY'), 'f', 7)

INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('1990426180155', 'Iliecu', 'Gabriel-Bogdan', 'bogdan.i@gmail.com',
TO_DATE('26-04-1999', 'DD-MM-YYYY'), 'm', 8)

INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('2940315119504', 'Zahar', 'Andreea', 'andreea23@yahoo.ro',
TO_DATE('15-03-1994', 'DD-MM-YYYY'), 'f', 9)

INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('1990721328881', 'Grigorescu', 'Ciprian-Eduard',
'grigorescu.ciprian@gmail.com', TO_DATE('21-07-1999', 'DD-MM-YYYY'), 'm', 15)

INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('2870212037355', 'Funar', 'Diana', 'diana1.funar@yahoo.com',
TO_DATE('12-02-1987', 'DD-MM-YYYY'), 'f', 16)

INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('2911002010206', 'Sala', 'Veronica', 'veronica.sala3@gmail.com',
TO_DATE('02-03-1991', 'DD-MM-YYYY'), 'f', 17)

INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('1951217161961', 'Stelian', 'Albert', 'albert.steliian@yahoo.com',
TO_DATE('17-12-1995', 'DD-MM-YYYY'), 'm', 18)

INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('1930309011068', 'Aurelian', 'Octavian-Dumitru',
'd-octavian.aurelian@gmail.com', TO_DATE('09-03-1993', 'DD-MM-YYYY'), 'm', 19)

```

        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('5020916348569', 'Dacian', 'Mihai', 'dacian.mihai@yahoo.ro',
TO_DATE('16-09-2002', 'DD-MM-YYYY'), 'm', 20)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('1960513234172', 'Dalca', 'Costache', 'costache_dalca@gmail.com',
TO_DATE('13-05-1996', 'DD-MM-YYYY'), 'm', 21)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('5031014179484', 'Balan', 'Alex', 'a.balan8@yahoo.com',
TO_DATE('14-10-2003', 'DD-MM-YYYY'), 'm', 22)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('2881202027415', 'Cojocaru', 'Mihaela', 'miha.cojo@yahoo.com',
TO_DATE('02-12-1988', 'DD-MM-YYYY'), 'f', 23)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('1900527165080', 'Lupu', 'Emil', 'emilll.lup@gmail.com',
TO_DATE('27-05-1990', 'DD-MM-YYYY'), 'm', 24)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('2941012172179', 'Ilieșcu', 'Roxana', 'roxi.ilieșcu@yahoo.com',
TO_DATE('12-10-1994', 'DD-MM-YYYY'), 'f', 8)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('1890528306731', 'Grigorescu', 'Marian', 'grigorescu.marian@yahoo.com',
TO_DATE('28-05-1989', 'DD-MM-YYYY'), 'm', 24)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('1950122298018', 'Adam', 'Ioan', 'adam.ioan6@gmail.com',
TO_DATE('22-01-1995', 'DD-MM-YYYY'), 'm', 16)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('1900414346266', 'Balan', 'Adelin', 'balan.adelin@yahoo.com',
TO_DATE('14-04-1990', 'DD-MM-YYYY'), 'm', 10)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('5020413517471', 'Dionisie', 'David', 'david.dion@gmail.com',
TO_DATE('13-04-2002', 'DD-MM-YYYY'), 'm', 13)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('5010911340845', 'Simion', 'Marian-Eduard', 'marian_ed.simion@gmail.com',
TO_DATE('11-09-2001', 'DD-MM-YYYY'), 'm', 14)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('2870426288591', 'Mosida', 'Anisoara', 'ani.mosida0@yahoo.ro',
TO_DATE('26-04-2001', 'DD-MM-YYYY'), 'f', 9)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('2910903121802', 'Alexandrescu', 'Andra-Maria',
'andra.alexandrescu4@gmail.com', TO_DATE('03-09-1991', 'DD-MM-YYYY'), 'f', 15)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('5001217466275', 'Anastasescu', 'Radu', 'radu.anastas7@yahoo.com',
TO_DATE('17-12-2000', 'DD-MM-YYYY'), 'm', 21)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('1880608200140', 'Ungur', 'Beniamin', 'beniamin.ungur@gmail.com',
TO_DATE('08-06-1988', 'DD-MM-YYYY'), 'm', 20)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)
VALUES('1940509282540', 'Petran', 'Bogdan-Teodor', 'bogdan.t0@yahoo.ro',
TO_DATE('09-05-1994', 'DD-MM-YYYY'), 'm', 18)
        INTO PERSOANA(cnp, nume, prenume, email, data_nastere, gen, cod_locatie)

```

```
VALUES('5021224054511', 'Ionescu', 'Ion', 'ion.ion8@yahoo.com',
TO_DATE('24-12-2002', 'DD-MM-YYYY'), 'm', 17)
SELECT 1 FROM DUAL;
```

```
INSERT ALL
  INTO CLIENT VALUES (1, 'melis2_')
  INTO CLIENT VALUES (2, 'bogdiG')
  INTO CLIENT VALUES (3, 'deeutza')
  INTO CLIENT VALUES (4, 'ediCipri')
  INTO CLIENT VALUES (5, 'dianaF_')
  INTO CLIENT VALUES (6, 'veronica.v')
  INTO CLIENT VALUES (7, 'albertoo')
  INTO CLIENT VALUES (8, 'octiDumitru')
  INTO CLIENT VALUES (9, 'mihaita')
  INTO CLIENT VALUES (10, 'Costache')
  INTO CLIENT VALUES (11, 'aleX_')
  INTO CLIENT VALUES (12, 'Miha24')
  INTO CLIENT VALUES (13, 'EmilLu')
  INTO CLIENT VALUES (14, 'RoxiI')
  INTO CLIENT VALUES (15, 'MariG_')
SELECT 1 FROM DUAL;
```

```
INSERT ALL
  INTO ANGAJAT VALUES(16, 100000, NULL, 'MANAGER')
  INTO ANGAJAT VALUES(17, 6000, 1, 'ADMIN')
  INTO ANGAJAT VALUES(18, 7500, 2, 'ADMIN')
  INTO ANGAJAT VALUES(19, 2900, 1, 'MOD')
  INTO ANGAJAT VALUES(20, 3000, 3, 'MOD')
  INTO ANGAJAT VALUES(21, 1500, 1, 'CASIER')
  INTO ANGAJAT VALUES(22, 1700, 2, 'CASIER')
  INTO ANGAJAT VALUES(23, 1620, 3, 'CASIER')
  INTO ANGAJAT VALUES(24, 2000, NULL, 'OP_CALL')
  INTO ANGAJAT VALUES(25, 2100, NULL, 'OP_CALL')
SELECT 1 FROM DUAL;
```

```
INSERT ALL
  INTO SEMNEAZA_UN_CONTRACT VALUES(17, 1, 'TERMS', NULL)
  INTO SEMNEAZA_UN_CONTRACT VALUES(17, 2, 'TERMS', 1)
  INTO SEMNEAZA_UN_CONTRACT VALUES(18, 3, 'TERMS', 1)
  INTO SEMNEAZA_UN_CONTRACT VALUES(16, 4, 'TERMS', NULL)
  INTO SEMNEAZA_UN_CONTRACT VALUES(18, 5, 'TERMS', 2)
  INTO SEMNEAZA_UN_CONTRACT VALUES(16, 6, 'TERMS', NULL)
  INTO SEMNEAZA_UN_CONTRACT VALUES(19, 7, 'TERMS', 2)
  INTO SEMNEAZA_UN_CONTRACT VALUES(17, 8, 'TERMS', NULL)
  INTO SEMNEAZA_UN_CONTRACT VALUES(16, 9, 'TERMS', 3)
  INTO SEMNEAZA_UN_CONTRACT VALUES(19, 10, 'TERMS', 4)
  INTO SEMNEAZA_UN_CONTRACT VALUES(18, 11, 'TERMS', NULL)
  INTO SEMNEAZA_UN_CONTRACT VALUES(16, 12, 'TERMS', 5)
  INTO SEMNEAZA_UN_CONTRACT VALUES(16, 13, 'TERMS', 5)
```

```

INTO SEMNEAZA_UN_CONTRACT VALUES(18, 14, 'TERMS', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(17, 15, 'TERMS', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(17, 1, 'PRIVACY', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(17, 2, 'PRIVACY', 1)
INTO SEMNEAZA_UN_CONTRACT VALUES(18, 3, 'PRIVACY', 1)
INTO SEMNEAZA_UN_CONTRACT VALUES(16, 4, 'PRIVACY', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(18, 5, 'PRIVACY', 2)
INTO SEMNEAZA_UN_CONTRACT VALUES(16, 6, 'PRIVACY', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(19, 7, 'PRIVACY', 2)
INTO SEMNEAZA_UN_CONTRACT VALUES(17, 8, 'PRIVACY', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(16, 9, 'PRIVACY', 3)
INTO SEMNEAZA_UN_CONTRACT VALUES(19, 10, 'PRIVACY', 4)
INTO SEMNEAZA_UN_CONTRACT VALUES(18, 11, 'PRIVACY', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(16, 12, 'PRIVACY', 5)
INTO SEMNEAZA_UN_CONTRACT VALUES(16, 13, 'PRIVACY', 5)
INTO SEMNEAZA_UN_CONTRACT VALUES(18, 14, 'PRIVACY', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(17, 15, 'PRIVACY', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(18, 1, 'ETHIC', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(17, 2, 'ETHIC', 1)
INTO SEMNEAZA_UN_CONTRACT VALUES(18, 4, 'ETHIC', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(19, 9, 'ETHIC', 3)
INTO SEMNEAZA_UN_CONTRACT VALUES(18, 10, 'ETHIC', 4)
INTO SEMNEAZA_UN_CONTRACT VALUES(19, 12, 'ETHIC', 5)
INTO SEMNEAZA_UN_CONTRACT VALUES(19, 15, 'ETHIC', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(19, 2, 'ECONOMII', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(18, 3, 'ECONOMII', 1)
INTO SEMNEAZA_UN_CONTRACT VALUES(17, 4, 'ECONOMII', NULL)
INTO SEMNEAZA_UN_CONTRACT VALUES(17, 9, 'ECONOMII', 3)
INTO SEMNEAZA_UN_CONTRACT VALUES(19, 10, 'ECONOMII', 4)
INTO SEMNEAZA_UN_CONTRACT VALUES(19, 11, 'ECONOMII', 5)
INTO SEMNEAZA_UN_CONTRACT VALUES(18, 14, 'ECONOMII', NULL)
SELECT 1 FROM DUAL;

```

```

INSERT ALL
  INTO CONT(cod_persoana, sold, iban) VALUES(1, 3200.20,
'RO80RZBR884596838338826')
    INTO CONT(cod_persoana, sold, iban) VALUES(2, 1000.2,
'RO93PORL7529645415765354')
      INTO CONT(cod_persoana, sold, iban) VALUES(3, 20.32,
'RO43PORL9543945186522245')
        INTO CONT(cod_persoana, sold, iban) VALUES(1, 100200.2,
'RO48PORL5146774785878989')
          INTO CONT(cod_persoana, sold, iban) VALUES(5, 39580.01,
'RO42RZBR7724779358767381')
            INTO CONT(cod_persoana, sold, iban) VALUES(6, 3339,
'RO37PORL3841381829398764')
              INTO CONT(cod_persoana, sold, iban) VALUES(7, 2981.44,
'RO31RZBR1288422343628368')
                INTO CONT(cod_persoana, sold, iban) VALUES(8, 1050.67,

```

```

'RO13PORL2734191267583753')
    INTO CONT(cod_persoana, sold, iban) VALUES(9, 18900.3,
'RO58PORL5263537759453949')
    INTO CONT(cod_persoana, sold, iban) VALUES(10, 36711.2,
'RO26RZBR6125587281719456')
SELECT 1 FROM DUAL;

INSERT ALL
    INTO CARD(tip, card_number, cod_cont) VALUES('visa', '4532962746090018', 1)
    INTO CARD(tip, card_number, cod_cont) VALUES('visa', '4801769871971639', 1)
    INTO CARD(tip, card_number, cod_cont) VALUES('visa', '4929249181139042', 1)
    INTO CARD(tip, card_number, cod_cont) VALUES('visa', '4539763856109249', 3)
    INTO CARD(tip, card_number, cod_cont) VALUES('visa', '4202142181423458', 3)
    INTO CARD(tip, card_number, cod_cont) VALUES('visa', '4929647287465379', 5)
    INTO CARD(tip, card_number, cod_cont) VALUES('visa', '4292894190135140', 6)
    INTO CARD(tip, card_number, cod_cont) VALUES('visa', '4556357158099097', 7)
    INTO CARD(tip, card_number, cod_cont) VALUES('mastercard', '5233771702998802',
2)
    INTO CARD(tip, card_number, cod_cont) VALUES('mastercard', '5196701739892160',
4)
    INTO CARD(tip, card_number, cod_cont) VALUES('mastercard', '5395006293149810',
8)
    INTO CARD(tip, card_number, cod_cont) VALUES('mastercard', '5336340664664640',
8)
    INTO CARD(tip, card_number, cod_cont) VALUES('mastercard', '5273043537498908',
9)
    INTO CARD(tip, card_number, cod_cont) VALUES('mastercard', '5528612483101600',
10)
    INTO CARD(tip, card_number, cod_cont) VALUES('mastercard', '5491696102956307',
10)
SELECT 1 FROM DUAL;

INSERT ALL
    INTO TRANZACTIE_EXTERNA VALUES(21, 2, 200, TO_DATE('03-01-2022',
'DD-MM-YYYY'), 'Depozit realizat la sediu.')
    INTO TRANZACTIE_EXTERNA VALUES(21, 3, 300, TO_DATE('12-04-2022',
'DD-MM-YYYY'), 'Depozit realizat la sediu.')
    INTO TRANZACTIE_EXTERNA VALUES(22, 1, -200, TO_DATE('06-05-2022',
'DD-MM-YYYY'), 'Retragere realizata la sediu.')
    INTO TRANZACTIE_EXTERNA VALUES(18, 1, 100, TO_DATE('09-02-2022',
'DD-MM-YYYY'), 'Reparare tranzactie fantoma.')
    INTO TRANZACTIE_EXTERNA VALUES(23, 4, 900, TO_DATE('23-03-2022',
'DD-MM-YYYY'), 'Depozit realizat la sediu.')
    INTO TRANZACTIE_EXTERNA VALUES(17, 6, 2000, TO_DATE('21-03-2022',
'DD-MM-YYYY'), 'Transfer de bani din contul sters.')
    INTO TRANZACTIE_EXTERNA VALUES(18, 7, -3200, TO_DATE('11-07-2022',
'DD-MM-YYYY'), 'Repararea unei erori de sistem.')
    INTO TRANZACTIE_EXTERNA VALUES(23, 7, -100, TO_DATE('14-06-2022',
'DD-MM-YYYY'), 'Retragere realizata la sediu.')

```

```

INTO TRANZACTIE_EXTERNA VALUES(21, 8, 400, TO_DATE('02-08-2022',
'DD-MM-YYYY'), 'Repararea unei erori de sistem.')
INTO TRANZACTIE_EXTERNA VALUES(18, 10, 600, TO_DATE('01-01-2022',
'DD-MM-YYYY'), 'Repararea unei erori de sistem.')
SELECT 1 FROM DUAL;

INSERT ALL
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(1, 1, TO_DATE('19-01-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('19-01-2022', 'DD-MM-YYYY'),3), 300, 1)
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(1, 1, TO_DATE('08-05-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('08-05-2022', 'DD-MM-YYYY'),3), 210, 0)
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(1, 2, TO_DATE('25-03-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('25-03-2022', 'DD-MM-YYYY'),6), 1000, 0)
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(4, 1, TO_DATE('16-05-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('16-05-2022', 'DD-MM-YYYY'),3), 520, 0)
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(4, 3, TO_DATE('21-02-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('21-02-2022', 'DD-MM-YYYY'),12), 3000, 0)
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(4, 5, TO_DATE('14-04-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('14-04-2022', 'DD-MM-YYYY'),48), 400, 0)
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(5, 4, TO_DATE('02-03-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('02-03-2022', 'DD-MM-YYYY'),24), 1200, 0)
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(6, 2, TO_DATE('19-03-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('19-03-2022', 'DD-MM-YYYY'),6), 1000, 0)
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(7, 1, TO_DATE('24-01-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('24-01-2022', 'DD-MM-YYYY'),3), 600, 1)
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(8, 1, TO_DATE('07-02-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('07-02-2022', 'DD-MM-YYYY'),3), 700, 1)
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(9, 5, TO_DATE('05-04-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('05-04-2022', 'DD-MM-YYYY'),48), 3100, 0)
  INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, data_start, data_sfarsit, valoare,
revendicat) VALUES(10, 1, TO_DATE('18-04-2022', 'DD-MM-YYYY'),
ADD_MONTHS(TO_DATE('18-04-2022', 'DD-MM-YYYY'),3), 2200, 0)
SELECT 1 FROM DUAL;

```

--inserarea tranzactiilor manual, as fi folosit functiile predifinite (care apar la cerinta 14) de depozit_bani,retragere_bani,transfer_bani
--dar am vrut sa aiba date diferite
--un trigger va creste automat tranzactii_contor din ISTORIC

```

INSERT ALL
  INTO TRANZACTIE VALUES(1, 1, 300, TO_DATE('03-05-2022', 'DD-MM-YYYY'),
'depozit')
    INTO DEPOZIT VALUES(1, 1, 1, 1)

    INTO TRANZACTIE VALUES(2, 1, 500, TO_DATE('12-01-2022', 'DD-MM-YYYY'),
'depozit')
      INTO DEPOZIT VALUES(2, 1, 4, 2)

    INTO TRANZACTIE VALUES(3, 1, 200, TO_DATE('24-04-2022', 'DD-MM-YYYY'),
'depozit')
      INTO DEPOZIT VALUES(3, 1, 3, 2)

    INTO TRANZACTIE VALUES(1, 2, 467, TO_DATE('15-02-2022', 'DD-MM-YYYY'),
'depozit')
      INTO DEPOZIT VALUES(1, 2, 3, 9)

    INTO TRANZACTIE VALUES(1, 3, 1200, TO_DATE('07-08-2022',
'DD-MM-YYYY'), 'depozit')
      INTO DEPOZIT VALUES(1, 3, 2, 4)

    INTO TRANZACTIE VALUES(1, 4, 1800, TO_DATE('09-03-2022',
'DD-MM-YYYY'), 'depozit')
      INTO DEPOZIT VALUES(1, 4, 4, 10)

    INTO TRANZACTIE VALUES(1, 5, 1200, TO_DATE('09-01-2022',
'DD-MM-YYYY'), 'depozit')
      INTO DEPOZIT VALUES(1, 5, 5, 6)

    INTO TRANZACTIE VALUES(1, 6, 800, TO_DATE('16-04-2022', 'DD-MM-YYYY'),
'depozit')
      INTO DEPOZIT VALUES(1, 6, 2, 7)

    INTO TRANZACTIE VALUES(2, 5, 1200, TO_DATE('09-01-2022',
'DD-MM-YYYY'), 'depozit')
      INTO DEPOZIT VALUES(2, 5, 5, 6)

    INTO TRANZACTIE VALUES(1, 8, 1100, TO_DATE('15-02-2022',
'DD-MM-YYYY'), 'depozit')
      INTO DEPOZIT VALUES(1, 8, 6, 12)

    INTO TRANZACTIE VALUES(1, 9, 2100, TO_DATE('09-01-2022',
'DD-MM-YYYY'), 'depozit')
      INTO DEPOZIT VALUES(1, 9, 1, 13)

    INTO TRANZACTIE VALUES(2, 9, 100, TO_DATE('09-01-2022', 'DD-MM-YYYY'),
'depozit')
      INTO DEPOZIT VALUES(2, 9, 2, 13)

```

```
INTO TRANZACTIE VALUES(4, 1, 200, TO_DATE('18-05-2022', 'DD-MM-YYYY'),  
'retragere')  
INTO RETRAGERE VALUES(4, 1, 2, 1)  
  
INTO TRANZACTIE VALUES(5, 1, 400, TO_DATE('17-01-2022', 'DD-MM-YYYY'),  
'retragere')  
INTO RETRAGERE VALUES(5, 1, 1, 2)  
  
INTO TRANZACTIE VALUES(6, 1, 150, TO_DATE('28-04-2022', 'DD-MM-YYYY'),  
'retragere')  
INTO RETRAGERE VALUES(6, 1, 4, 2)  
  
INTO TRANZACTIE VALUES(2, 2, 200, TO_DATE('11-09-2022', 'DD-MM-YYYY'),  
'retragere')  
INTO RETRAGERE VALUES(2, 2, 5, 9)  
  
INTO TRANZACTIE VALUES(2, 3, 10, TO_DATE('09-05-2022', 'DD-MM-YYYY'),  
'retragere')  
INTO RETRAGERE VALUES(2, 3, 7, 4)  
  
INTO TRANZACTIE VALUES(2, 4, 30, TO_DATE('11-07-2022', 'DD-MM-YYYY'),  
'retragere')  
INTO RETRAGERE VALUES(2, 4, 3, 10)  
  
INTO TRANZACTIE VALUES(3, 5, 70, TO_DATE('05-06-2022', 'DD-MM-YYYY'),  
'retragere')  
INTO RETRAGERE VALUES(3, 5, 2, 6)  
  
INTO TRANZACTIE VALUES(2, 6, 800, TO_DATE('11-03-2022', 'DD-MM-YYYY'),  
'retragere')  
INTO RETRAGERE VALUES(2, 6, 6, 7)  
  
INTO TRANZACTIE VALUES(4, 5, 1200, TO_DATE('27-01-2022',  
'DD-MM-YYYY'), 'retragere')  
INTO RETRAGERE VALUES(4, 5, 5, 6)  
  
INTO TRANZACTIE VALUES(2, 8, 60, TO_DATE('24-02-2022', 'DD-MM-YYYY'),  
'retragere')  
INTO RETRAGERE VALUES(2, 8, 2, 12)  
  
INTO TRANZACTIE VALUES(3, 9, 1300, TO_DATE('07-02-2022',  
'DD-MM-YYYY'), 'retragere')  
INTO RETRAGERE VALUES(3, 9, 1, 13)  
  
INTO TRANZACTIE VALUES(4, 9, 50, TO_DATE('11-03-2022', 'DD-MM-YYYY'),  
'retragere')  
INTO RETRAGERE VALUES(4, 9, 2, 13)
```

```

    INTO TRANZACTIE VALUES(7, 1, 50, TO_DATE('16-03-2022', 'DD-MM-YYYY'),
'transfer_triminis')
    INTO TRANSFER_TRIMIS VALUES(7, 1, 2)
    INTO TRANZACTIE VALUES(3, 2, 50, TO_DATE('16-03-2022', 'DD-MM-YYYY'),
'transfer_primit')
    INTO TRANSFER_PRIMIT VALUES(3, 2, 1)

    INTO TRANZACTIE VALUES(3, 4, 6000, TO_DATE('21-02-2022',
'DD-MM-YYYY'), 'transfer_triminis')
    INTO TRANSFER_TRIMIS VALUES(3, 4, 3)
    INTO TRANZACTIE VALUES(3, 3, 6000, TO_DATE('21-02-2022',
'DD-MM-YYYY'), 'transfer_primit')
    INTO TRANSFER_PRIMIT VALUES(3, 3, 4)

    INTO TRANZACTIE VALUES(1, 10, 2100, TO_DATE('01-04-2022',
'DD-MM-YYYY'), 'transfer_triminis')
    INTO TRANSFER_TRIMIS VALUES(1, 10, 3)
    INTO TRANZACTIE VALUES(4, 3, 2100, TO_DATE('01-04-2022',
'DD-MM-YYYY'), 'transfer_primit')
    INTO TRANSFER_PRIMIT VALUES(4, 3, 10)

    INTO TRANZACTIE VALUES(3, 6, 900, TO_DATE('09-05-2022', 'DD-MM-YYYY'),
'transfer_triminis')
    INTO TRANSFER_TRIMIS VALUES(3, 6, 7)
    INTO TRANZACTIE VALUES(1, 7, 900, TO_DATE('09-05-2022', 'DD-MM-YYYY'),
'transfer_primit')
    INTO TRANSFER_PRIMIT VALUES(1, 7, 6)

    INTO TRANZACTIE VALUES(4, 4, 250, TO_DATE('09-03-2022', 'DD-MM-YYYY'),
'transfer_triminis')
    INTO TRANSFER_TRIMIS VALUES(4, 4, 8)
    INTO TRANZACTIE VALUES(3, 8, 250, TO_DATE('09-03-2022', 'DD-MM-YYYY'),
'transfer_primit')
    INTO TRANSFER_PRIMIT VALUES(3, 8, 4)
SELECT 1 FROM DUAL;

```

Worksheet | Query Builder

```
887 INTO TRANSFER_TRIMIS VALUES(3, 4, 3)
888 INTO TRANZACTIE VALUES(3, 3, 6000, TO_DATE('21-02-2022', 'DD-MM-YYYY'), 'transfer_primit')
889 INTO TRANSFER_PRIMIT VALUES(3, 3, 4)
890
891 INTO TRANZACTIE VALUES(1, 10, 2100, TO_DATE('01-04-2022', 'DD-MM-YYYY'), 'transfer_trimis')
892 INTO TRANSFER_TRIMIS VALUES(1, 10, 3)
893 INTO TRANZACTIE VALUES(4, 3, 2100, TO_DATE('01-04-2022', 'DD-MM-YYYY'), 'transfer_primit')
894 INTO TRANSFER_PRIMIT VALUES(4, 3, 10)
895
896 INTO TRANZACTIE VALUES(3, 6, 900, TO_DATE('09-05-2022', 'DD-MM-YYYY'), 'transfer_trimis')
897 INTO TRANSFER_TRIMIS VALUES(3, 6, 7)
898 INTO TRANZACTIE VALUES(1, 7, 900, TO_DATE('09-05-2022', 'DD-MM-YYYY'), 'transfer_primit')
899 INTO TRANSFER_PRIMIT VALUES(1, 7, 6)
900
901 INTO TRANZACTIE VALUES(4, 4, 250, TO_DATE('09-03-2022', 'DD-MM-YYYY'), 'transfer_trimis')
902 INTO TRANSFER_TRIMIS VALUES(4, 4, 8)
903 INTO TRANZACTIE VALUES(3, 8, 250, TO_DATE('09-03-2022', 'DD-MM-YYYY'), 'transfer_primit')
904 INTO TRANSFER_PRIMIT VALUES(3, 8, 4)
905 SELECT 1 FROM DUAL;
```

Script Output | Query Result | Task completed in 0.148 seconds

10 rows inserted.

44 rows inserted.

10 rows inserted.

15 rows inserted.

10 rows inserted.

12 rows inserted.

68 rows inserted.

Tabelele după inserare:

907 | SELECT * FROM PERSOANA;

	COD_PERSOANA	CNP	NUME	PRENUME	EMAIL	DATA_NASTERE	GEN	COD_LOCATIE
1	1 2870816526255	Pintenaru	Melissa		melissa.p@gmail.com	16-AUG-87	f	7
2	2 1990426180155	Ilieșcu	Gabriel-Bogdan		bogdan.i@gmail.com	26-APR-99	m	8
3	3 2940315119504	Zahar	Andreea		andreea23@yahoo.ro	15-MAR-94	f	9
4	4 1990721328881	Grigorescu	Ciprian-Eduard		grigorescu.ciprian@gmail.com	21-JUL-99	m	15
5	5 2870212037355	Funar	Diana		diana.funar@yahoo.com	12-FEB-87	f	16
6	6 2911002010206	Sala	Veronica		veronica.sala3@gmail.com	02-MAR-91	f	17
7	7 1951217161961	Stelian	Albert		albert.stelian@yahoo.com	17-DEC-95	m	18
8	8 1930309011068	Aurelian	Octavian-Dumitru		d-octavian.aurelian@gmail.com	09-MAR-93	m	19
9	9 5020916348569	Dacian	Mihai		dacian.mihai@yahoo.ro	16-SEP-02	m	20
10	10 1960513234172	Dalca	Costache		costache_dalca@gmail.com	13-MAY-96	m	21
11	11 5031014179484	Balan	Alex		a.balan8@yahoo.com	14-OCT-03	m	22
12	12 2881202027415	Cojocaru	Mihaela		miha.cojo@yahoo.com	02-DEC-88	f	23
13	13 1900527165080	Lupu	Emil		emill1.lup@gmail.com	27-MAY-90	m	24
14	14 2941012172179	Ilieșcu	Roxana		roxi.ilieșcu@yahoo.com	12-OCT-94	f	8
15	15 1890528306731	Grigorescu	Marian		grigorescu.marian@yahoo.com	28-MAY-89	m	24
16	16 1950122298018	Adam	Ioan		adam.ioan6@gmail.com	22-JAN-95	m	16
17	17 1900414346266	Balan	Adelin		balan.adelin@yahoo.com	14-APR-90	m	10
18	18 5020413517471	Dionisie	David		david.dion@gmail.com	13-APR-02	m	13
19	19 5010911340845	Simion	Marian-Eduard		marijan_ed.simion@gmail.com	11-SEP-01	m	14
20	20 2870426288591	Mosida	Anisoara		ani.mosida0@yahoo.ro	26-APR-01	f	9
21	21 2910903121802	Alexandrescu	Andra-Maria		andra.alexandrescu4@gmail.com	03-SEP-91	f	15
22	22 5001217466275	Anastasescu	Radu		radu.anastas7@yahoo.com	17-DEC-00	m	21
23	23 1880608200140	Ungur	Beniamin		beniamin.ungur@gmail.com	08-JUN-88	m	20
24	24 1940509282540	Petran	Bogdan-Teodor		bogdan.t0@yahoo.ro	09-MAY-94	m	18
25	25 5021224054511	Ionescu	Ion		ion.ion8@yahoo.com	24-DEC-02	m	17

907 | SELECT * FROM ANGAJAT;

	COD_PERSOANA	SALARIU	COD_SEDIU	COD_JOB
1	16	100000	(null)	MANAGER
2	17	6000		1 ADMIN
3	18	7500		2 ADMIN
4	19	2900		1 MOD
5	20	3000		3 MOD
6	21	1500		1 CASIER
7	22	1700		2 CASIER
8	23	1620		3 CASIER
9	24	2000	(null)	OP_CALL
10	25	2100	(null)	OP_CALL

```
907 | SELECT * FROM CLIENT;
```

Script Output x

Query Result x

SQL | All Rows Fetched: 15 in 0.007 seconds

	COD_PERSOANA	TAG_UTILIZATOR
1		1 melis2_
2		2 bogdiG
3		3 deeutza
4		4 ediCipri
5		5 dianaF_
6		6 veronica.v
7		7 albertoo
8		8 octiDumitru
9		9 mihaita
10		10 Costache
11		11 aleX_
12		12 Miha24
13		13 EmilLu
14		14 RoxiI
15		15 MariG_

```
907 | SELECT * FROM JOB;
```

Script Output x

Query Result x

SQL | All Rows Fetched: 5 in 0.003 seconds

	COD_JOB	DENUMIRE_JOB	SALARIU_MINIM	SALARIU_MAXIM
1	MANAGER	Manager	4000	12000
2	ADMIN	Administrator	2000	8000
3	MOD	Moderator	1500	4000
4	CASIER	Casier	1300	2000
5	OP_CALL	Operator CallCenter	1500	3000

```
907 | SELECT * FROM DOBANDA;
```

Script Output x Query Result x

SQL | All Rows Fetched: 5 in 0.002 seconds

COD_DOBANDA	PROCENT	DURATA_LUNI
1	1	0.1
2	2	0.3
3	3	0.7
4	4	0.15
5	5	0.35

```
907 | SELECT * FROM SEDIU;
```

Script Output x Query Result x

SQL | All Rows Fetched: 5 in 0.002 seconds

COD_SEDIU	COD_LOCATIE
1	1
2	2
3	3
4	4
5	5

```
907 | SELECT * FROM TRANZACTIE_EXTERNA;
```

Script Output x Query Result x

SQL | All Rows Fetched: 10 in 0.003 seconds

COD_PERSOANA	COD_CONT	VALOARE	DATA_TRANZACTIE	MESAJ_ADMINISTRATIV
1	21	2	200 03-JAN-22	Depozit realizat la sediu.
2	21	3	300 12-APR-22	Depozit realizat la sediu.
3	22	1	-200 06-MAY-22	Retragere realizata la sediu.
4	18	1	100 09-FEB-22	Reparare tranzactie fantoma.
5	23	4	900 23-MAR-22	Depozit realizat la sediu.
6	17	6	2000 21-MAR-22	Transfer de bani din contul sters.
7	18	7	-3200 11-JUL-22	Repararea unei erori de sistem.
8	23	7	-100 14-JUN-22	Retragere realizata la sediu.
9	21	8	400 02-AUG-22	Repararea unei erori de sistem.
10	18	10	600 01-JAN-22	Repararea unei erori de sistem.

907 | SELECT * FROM TARA;

Script Output x Query Result x

SQL | All Rows Fetched: 10 in 0.003 seconds

	COD_TARA	NUME_TARA
1		1 Romania
2		2 Olanda
3		3 Bulgaria
4		4 Spania
5		5 Rusia
6		6 Elvetia
7		7 Finlanda
8		8 Suedia
9		9 Grecia
10		10 Japonia

907 | SELECT * FROM LOCATIE;

Script Output x Query Result x

SQL | All Rows Fetched: 24 in 0.003 seconds

	COD_LOCATIE	COD_POSTAL	ADRESA	ORAS	COD_TARA
1	1 2270		Str Artar Nr 12	Bucuresti	1
2	2 11199		Str Macin Nr 22	Cluj	1
3	3 22222		Str Green Nr 1	Amsterdam	2
4	4 93123		Str Goodfellow Nr 5	Haga	2
5	5 6677		Str Bujha Nr 9	Sofia	3
6	6 9221		Str Ehia Nr 29	Vidin	3
7	7 1122		Str Ernesto Nr 8 Bl A2 Sc 1 Et 3 Ap 10	Madrid	4
8	8 8890		Str Ruski Nr 10 Bl F3 Sc 2 Et 1 Ap 4	Moscova	5
9	9 1000		Str Kijrsi Nr 9 Bl C2 Sc 1 Et 2 Ap 7	Moscova	5
10	10 3216		Str Etiop Nr 12 Bl A1 Sc 4 Et 5 Ap 32	Geneva	6
11	11 2839		Str Yaht Nr 89	Atena	9
12	12 1909		Str Kohli Nr 54	Corfu	9
13	13 5750		Str Juim Nr 72 Bl D3 Sc 2 Et 1 Ap 2	Tokyo	10
14	14 2111		Str Fukushi Nr 78 Bl B3 Sc 3 Et 4 Ap 22	Hokkaido	10
15	15 233111		Str Iancului Nr 122	Suceava	1
16	16 28765		Str Armeni Nr 36	Drobeta-Turnu-Severin	1
17	17 8955		Str Maresal Nr 81	Targu-Jiu	1
18	18 29464		Str Stoica Nr 77	Sibiu	1
19	19 6894		Str Plopilor Nr 90	Brasov	1
20	20 46948		Str Cicero Nr 25	Drobeta-Turnu-Severin	1
21	21 44766		Str Titeica Nr 17	Sibiu	1
22	22 40099		Str Sadoveanu Nr 89	Bucuresti	1
23	23 10200		Str Alias Nr 98	Cluj	1
24	24 100229		Str Mesterului Nr 4	Bucurestii	1

```
907 | SELECT * FROM BANCOMAT;
```

Script Output x Query Result x

SQL | All Rows Fetched: 7 in 0.002 seconds

COD_BANCOMAT	COD_LOCATIE
1	1
2	2
3	3
4	4
5	5
6	6
7	11
	12

```
985 | SELECT * FROM CONT;
```

Script Output x Query Result x Query Result 1 x

SQL | All Rows Fetched: 10 in 0.001 seconds

COD_CONT	COD_PERSOANA	SOLD	IBAN
1	1	1	3200.2 RO80RZBR8845968383338826
2	2	2	1000.2 RO93PORL7529645415765354
3	3	3	20.32 RO43PORL9543945186522245
4	4	1	100200.2 RO48PORL5146774785878989
5	5	5	39580.01 RO42RZBR7724779358767381
6	6	6	3339 RO37PORL3841381829398764
7	7	7	2981.44 RO31RZBR1288422343628368
8	8	8	1050.67 RO13PORL2734191267583753
9	9	9	18900.3 RO58PORL5263537759453949
10	10	10	36711.2 RO26RZBR6125587281719456

```
907 | SELECT * FROM CONT_ECONOMII;
```

Script Output x Query Result x

SQL | All Rows Fetched: 10 in 0.002 seconds

COD_CONT	DEPOZITE_CONTOR
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

```
907 | SELECT * FROM DEPOZIT_ECONOMII;
```

Script Output x Query Result x

SQL | All Rows Fetched: 12 in 0.002 seconds

COD_DEPOZIT_ECONOMII	COD_CONT	COD_DOBANDA	DATA_START	DATA_SFARSIT	VALOARE	REVENDICAT
1	1	1	1 19-JAN-22	19-APR-22	300	1
2	2	1	1 08-MAY-22	08-AUG-22	210	0
3	3	1	2 25-MAR-22	25-SEP-22	1000	0
4	1	4	1 16-MAY-22	16-AUG-22	520	0
5	2	4	3 21-FEB-22	21-FEB-23	3000	0
6	3	4	5 14-APR-22	14-APR-26	400	0
7	1	5	4 02-MAR-22	02-MAR-24	1200	0
8	1	6	2 19-MAR-22	19-SEP-22	1000	0
9	1	7	1 24-JAN-22	24-APR-22	600	1
10	1	8	1 07-FEB-22	07-MAY-22	700	1
11	1	9	5 05-APR-22	05-APR-26	3100	0
12	1	10	1 18-APR-22	18-JUL-22	2200	0

```
907 | SELECT * FROM ISTORIC;
```

Script Output x Query Result x

SQL | All Rows Fetched: 10 in 0.003 seconds

	COD_CONT	TRANZACTII_CONTOR
1	1	7
2	2	3
3	3	4
4	4	4
5	5	4
6	6	3
7	7	1
8	8	3
9	9	4
10	10	1

```
918 | SELECT * FROM CARD;
```

Script Output x Query Result x

SQL | All Rows Fetched: 15 in 0.002 seconds

	COD_CARD	PIN	TIP	CARD_NUMBER	COD_CONT
1	1	1000	visa	4532962746090018	1
2	2	1001	visa	4801769871971639	1
3	3	1002	visa	4929249181139042	1
4	4	1003	visa	4539763856109249	3
5	5	1004	visa	4202142181423458	3
6	6	1005	visa	4929647287465379	5
7	7	1006	visa	4292894190135140	6
8	8	1007	visa	4556357158099097	7
9	9	1008	mast...	5233771702998802	2
10	10	1009	mast...	5196701739892160	4
11	11	1010	mast...	5395006293149810	8
12	12	1011	mast...	5336340664664640	8
13	13	1012	mast...	5273043537498908	9
14	14	1013	mast...	5528612483101600	10
15	15	1014	mast...	5491696102956307	10

```
918 | SELECT * FROM TRANZACTIE;
```

Script Output x Query Result x

All Rows Fetched: 34 in 0.003 seconds

COD_TRA...	COD_CONT	VALOARE	DATA_TRANZACTIE	TIP_TRANZACTIE
1	1	1	300 03-MAY-22	depozit
2	2	1	500 12-JAN-22	depozit
3	3	1	200 24-APR-22	depozit
4	1	2	467 15-FEB-22	depozit
5	1	3	1200 07-AUG-22	depozit
6	1	4	1800 09-MAR-22	depozit
7	1	5	1200 09-JAN-22	depozit
8	1	6	800 16-APR-22	depozit
9	2	5	1200 09-JAN-22	depozit
10	1	8	1100 15-FEB-22	depozit
11	1	9	2100 09-JAN-22	depozit
12	2	9	100 09-JAN-22	depozit
13	4	1	200 18-MAY-22	retragere
14	5	1	400 17-JAN-22	retragere
15	6	1	150 28-APR-22	retragere
16	2	2	200 11-SEP-22	retragere
17	2	3	10 09-MAY-22	retragere
18	2	4	30 11-JUL-22	retragere
19	3	5	70 05-JUN-22	retragere
20	2	6	800 11-MAR-22	retragere
21	4	5	1200 27-JAN-22	retragere
22	2	8	60 24-FEB-22	retragere
23	3	9	1300 07-FEB-22	retragere
24	4	9	50 11-MAR-22	retragere
25	7	1	50 16-MAR-22	transfer Trimis
26	3	2	50 16-MAR-22	transfer Primit
27	3	4	6000 21-FEB-22	transfer Trimis
28	3	3	6000 21-FEB-22	transfer Primit
29	1	10	2100 01-APR-22	transfer Trimis
30	4	3	2100 01-APR-22	transfer Primit

```
918 | SELECT * FROM DEPOZIT;
```

Script Output x Query Result x

SQL | All Rows Fetched: 12 in 0.002 seconds

	COD_TRAZACTIE	COD_CONT	COD_BANCOMAT	COD_CARD
1	1	1		1
2	2	1		2
3	3	1		2
4	1	2		9
5	1	3		4
6	1	4		10
7	1	5		6
8	1	6		7
9	2	5		6
10	1	8		12
11	1	9		13
12	2	9		13

```
918 | SELECT * FROM RETRAGERE;
```

Script Output x Query Result x

SQL | All Rows Fetched: 12 in 0.002 seconds

	COD_TRANZACTIE	COD_CONT	COD_BANCOMAT	COD_CARD
1		4	1	2
2		5	1	1
3		6	1	4
4		2	2	5
5		2	3	7
6		2	4	3
7		3	5	2
8		2	6	6
9		4	5	5
10		2	8	2
11		3	9	1
12		4	9	2

```
918 | SELECT * FROM TRANSFER_PRIMIT;
```

Script Output x Query Result x

SQL | All Rows Fetched: 5 in 0.002 seconds

	COD_TRANZACTIE	COD_CONT	COD_CONT_SURSA
1		3	2
2		3	3
3		4	3
4		1	7
5		3	8

```
918 | SELECT * FROM TRANSFER_TRIMIS;
```

Script Output x Query Result x

SQL | All Rows Fetched: 5 in 0.003 seconds

	COD_TRANZACTIE	COD_CONT	COD_CONT_DESTINATIE
1		7	1
2		3	4
3		1	10
4		3	6
5		4	4

```
918 | SELECT * FROM TIP_CONTRACT;
```

Script Output x Query Result x

SQL | All Rows Fetched: 4 in 0.029 seconds

TIP_CONTRACT	TITLU_CONTRACT	CONTINUT
1 TERMS	Termini si Conditii	Sa fii cuminte, sa nu faci evaziune fiscala, sa nu faci tranzactii dubioase, sa respecti regulile de bun ...
2 PRIVACY	Politica de Confidentialitate	Nu spunem, nu facem, nu transmitem decat daca te prinde ANAF-ul, ai tag de utilizator, numele nu se comun...
3 ETHIC	Politica de Etica BUSINESS	Daca vinzi produse fa-o legal, daca te da cineva in judecata noi nu ne bagam, birocratie birocratie etc etc
4 ECONOMII	Politica de Economii	Ajuta sistemul sa aiba o cantitate constanta de bani si primeste o dobanda pentru buna ta vointa, etc etc...

6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze două tipuri diferite de colecții studiate. Apelați subprogramul.

--6. Formulati in limbaj natural o problema pe care sa o rezolvati folosind
-- un subprogram stocat independent care sa utilizeze doua tipuri diferite
-- de colectii studiate. Apelati subprogramul.

--Problema:

--Manager-ul vrea sa incerce o noua strategie de marketing.
--El vrea sa motiveze persoanele inactive sa foloseasca serviciile bancii mai mult asa ca
--acesta propune trimiterea de mail-uri cu oferte in functie de activitatea clientilor.
--Activitatea utilizatorului va fi un total de puncte, calculat astfel:
-- - 1 punct pentru fiecare depozit/retragere/transfer Trimis
-- - 5 puncte pentru fiecare card detinut
-- - 2 puncte * numarul de luni pe care este facut un depozit de economii
--Faceti o procedura si afisati email-ul, username-ul si punctajul clientilor pentru a
--oferi manager-ului informatii despre activitatea utilizatorilor.
--Procedura poate sa primeasca ca parametru un numar pozitiv reprezentand un prag
superior,
--urmand sa se afiseze doar clientii cu punctajul mai mic sau egal cu cel dat ca parametru.

```
CREATE OR REPLACE PROCEDURE activitate_utilizatori(activ_max PLS_INTEGER  
DEFAULT -1)
```

AS

```
TYPE t_info_user IS RECORD(  
    email PERSOANA.email%TYPE,  
    username CLIENT.tag_utilizator%TYPE,  
    punctaj PLS_INTEGER DEFAULT 0  
)
```

```
TYPE t_idx IS TABLE OF t_info_user INDEX BY PLS_INTEGER;  
v_useri t_idx;  
v_index_user PLS_INTEGER;
```

```
TYPE t_punctaj IS RECORD(  
    cod_persoana PERSOANA.cod_persoana%TYPE,  
    punctaj PLS_INTEGER  
)
```

```
TYPE t_imb_punctaj IS TABLE OF t_punctaj;  
v_adaos_punctaj t_imb_punctaj; --nu are rost sa initializam, folosim bulk collect
```

```
TYPE t_info_user_cu_cod IS RECORD(  
    cod_persoana PERSOANA.cod_persoana%TYPE,  
    email PERSOANA.email%TYPE,
```

```

        username CLIENT.tag_utilizator%TYPE
);

        TYPE t_vector_info_user IS VARRAY(2000) OF t_info_user_cu_cod; --suntem o banca
mica, ne ajunge maxim 2000, marim la nevoie
        v_aux_1 t_vector_info_user; --nu are rost sa initializam, folosim bulk collect
BEGIN
        --ne abtinem din a folosi un ciclu cursor, doar de dragul de a folosi un vector
        SELECT p.cod_persoana, email, tag_utilizator BULK COLLECT INTO v_aux_1
FROM PERSOANA p, CLIENT c WHERE p.cod_persoana = c.cod_persoana;

        IF v_aux_1.count = 0 THEN --nu exista utilizatori, nu are rost sa facem calcule
            dbms_output.put_line('Nu exista utilizatori care sa aiba activitate.');
            RETURN;
        END IF;

        FOR i IN 1..v_aux_1.count LOOP
            v_useri(v_aux_1(i).cod_persoana).email := v_aux_1(i).email;
            v_useri(v_aux_1(i).cod_persoana).username := v_aux_1(i).username;
        END LOOP;

        --calculam punctele din depozite/retrageri/transferuri trimise si stocam in
v_adaos_punctaj
        WITH
            tranzactiiPerCont AS (SELECT cod_cont, COUNT(cod_tranzactie) tranzactii
                FROM TRANZACTIE WHERE DECODE(lower(tip_tranzactie),
'depozit', 1, 'retragere', 1, 'transfer Trimis', 1, 0) = 1
                GROUP BY cod_cont)
            SELECT c.cod_persoana, SUM(tranzactii) BULK COLLECT INTO v_adaos_punctaj
            FROM CONT c, tranzactiiPerCont
            WHERE c.cod_cont = tranzactiiPerCont.cod_cont GROUP BY c.cod_persoana;

        --fiind inserat cu bulk collect, stiu ca este dens
        FOR i IN 1..v_adaos_punctaj.count LOOP
            v_useri(v_adaos_punctaj(i).cod_persoana).punctaj :=
v_useri(v_adaos_punctaj(i).cod_persoana).punctaj + v_adaos_punctaj(i).punctaj;
            IF activ_max != -1 AND
v_useri(v_adaos_punctaj(i).cod_persoana).punctaj > activ_max THEN
                v_useri.delete(v_adaos_punctaj(i).cod_persoana); --pragul superior s-a deposit, nu
ne mai intereseaza utilizatorul
            END IF;
        END LOOP;

        --calculam punctele pentru carduri detinute
        WITH
            carduriPerCont AS (SELECT cod_cont, COUNT(cod_card) carduri FROM CARD
            GROUP BY cod_cont)

```

```

SELECT c.cod_persoana, SUM(carduri)*5 BULK COLLECT INTO v_adaos_punctaj
FROM CONT c, carduriPerCont
WHERE c.cod_cont = carduriPerCont.cod_cont GROUP BY c.cod_persoana;

--acum va trebui sa verificam si daca exista persoana in lista
--daca nu exista inseamna ca a fost scoasa pentru ca limita a fost depasita
--la un pas anterior
FOR i IN 1..v_adaos_punctaj.count LOOP
    IF NOT v_useri.exists(v_adaos_punctaj(i).cod_persoana) THEN
        CONTINUE;
    END IF;

    v_useri(v_adaos_punctaj(i).cod_persoana).punctaj :=
    v_useri(v_adaos_punctaj(i).cod_persoana).punctaj + v_adaos_punctaj(i).punctaj;
    IF activ_max != -1 AND
    v_useri(v_adaos_punctaj(i).cod_persoana).punctaj>activ_max THEN
        v_useri.delete(v_adaos_punctaj(i).cod_persoana); --pragul superior s-a depasit, nu
ne mai intereseaza utilizatorul
    END IF;
END LOOP;

--schimbam din with in join pe 3 tabele
SELECT c.cod_persoana, SUM(durata_luni)*2 BULK COLLECT INTO
v_adaos_punctaj
FROM DEPOZIT_ECONOMII dE, DOBANDA d, CONT c WHERE c.cod_cont =
dE.cod_cont AND dE.cod_dobanda = d.cod_dobanda GROUP BY c.cod_persoana;

--neschimbat fata de for-ul anterior
FOR i IN 1..v_adaos_punctaj.count LOOP
    IF NOT v_useri.exists(v_adaos_punctaj(i).cod_persoana) THEN
        CONTINUE;
    END IF;

    v_useri(v_adaos_punctaj(i).cod_persoana).punctaj :=
    v_useri(v_adaos_punctaj(i).cod_persoana).punctaj + v_adaos_punctaj(i).punctaj;
    IF activ_max != -1 AND
    v_useri(v_adaos_punctaj(i).cod_persoana).punctaj>activ_max THEN
        v_useri.delete(v_adaos_punctaj(i).cod_persoana); --pragul superior s-a depasit, nu
ne mai intereseaza utilizatorul
    END IF;
END LOOP;

IF v_useri.first IS NULL THEN
    dbms_output.put_line('Nu exista utilizatori care sa aiba activitate.');
    RETURN;
END IF;

dbms_output.put_line('===== Puncte de activitate =====');
v_index_user := v_useri.first;

```

```

WHILE v_index_user IS NOT NULL LOOP
    dbms_output.put_line('> ' || v_useri(v_index_user).username || '(' ||
v_useri(v_index_user).email || ') - ' || v_useri(v_index_user).punctaj);
    v_index_user := v_useri.next(v_index_user);
END LOOP;
END;
/

EXECUTE activitate_utilizatori;
--lipsesc melis2_si mihaita
EXECUTE activitate_utilizatori(100);

```

```

823 | IF v_useri.first IS NULL THEN
824 |   dbms_output.put_line('Nu exista utilizatori care sa aiba activitate.');
825 |   RETURN;
826 | END IF;
827 |
828 |
829 | dbms_output.put_line('===== Puncte de activitate =====');
830 | v_index_user := v_useri.first;
831 | WHILE v_index_user IS NOT NULL LOOP
832 |   dbms_output.put_line('> ' || v_useri(v_index_user).username || ' (' ||
833 |   v_index_user := v_useri.next(v_index_user);
834 | END LOOP;
835 | END;
836 |
837 |
838 EXECUTE activitate_utilizatori;
839 --lipsesc melis2_si mihaita
840 EXECUTE activitate_utilizatori(100);

```

Script Output x | Task completed in 0.036 seconds

PL/SQL procedure successfully completed.

Dbms Output

+ | Buffer Size: 20000 | localhost x

```

===== Puncte de activitate =====
> melis2_ (melissa.p@gmail.com) - 181
> bogdanG (bogdan.i@gmail.com) - 7
> deeutza (andreea23@yahoo.ro) - 12
> ediCipri (grigorescu.ciprian@gmail.com) - 0
> dianaf_ (diana1.funar@yahoo.com) - 57
> veronica.v (veronica.sala3@gmail.com) - 20
> albertoo (albert.stelian@yahoo.com) - 11
> octidumitru (d-octavian.aurelian@gmail.com) - 18
> mihaita (dacian.mihai@yahoo.ro) - 105
> Costache (costache_dalca@gmail.com) - 17
> aleX_ (a.balan@yahoo.com) - 0
> Miha24 (miha.cojo@yahoo.com) - 0
> Emilll_ (emilll.lup@gmail.com) - 0
> RoxiII (roxii.iliescu@yahoo.com) - 0
> MariG_ (grigorescu.marian@yahoo.com) - 0

```

```

824 IF v_useri.first IS NULL THEN
825   dbms_output.put_line('Nu exista utilizatori care sa aiba activitate.');
826   RETURN;
827 END IF;
828
829 dbms_output.put_line('===== Puncte de activitate =====');
830 v_index_user := v_useri.first;
831 WHILE v_index_user IS NOT NULL LOOP
832   dbms_output.put_line('> ' || v_useri(v_index_user).username || ' (' || v_useri(v_index_user).email || ') - ' || v_useri(v_index_user).punctaj);
833   v_index_user := v_useri.next(v_index_user);
834 END LOOP;
835 END;
836 /
837
838 EXECUTE activitate_utilizatori;
839 --lipsesc melis2_si mihaita
840 EXECUTE activitate_utilizatori(100);
841

```

Script Output | Task completed in 0.045 seconds
PL/SQL procedure successfully completed.

Dbsn Output | Buffer Size: 20000 | localhost

```

===== Puncte de activitate =====
> bogdanG (bogdan.i@gmail.com) - 7
> deeutza (andreea23@yahoo.ro) - 12
> ediCipri (grigorescu.ciprian@gmail.com) - 0
> dianaF_ (diana.funar@yahoo.com) - 57
> veronica.v (veronica.sala3@gmail.com) - 20
> albertoo (albert.stelian@yahoo.com) - 11
> octiDumitru (d-octavian.aurelian@gmail.com) - 18
> Costache (costache_daica@gmail.com) - 17
> aleX_ (a.balan@yahoo.com) - 0
> Miha24 (miha.cojo@yahoo.com) - 0
> Emillu (emillu.lup@gmail.com) - 0
> Roxii (roxi.iliescu@yahoo.com) - 0
> MariG_ (grigorescu.marian@yahoo.com) - 0

```

7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat. Apelați subprogramul.

--7. Formulati in limbaj natural o problema pe care sa o rezolvati
-- folosind un subprogram stocat independent care sa utilizeze
-- 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind
--cursor parametrizat. Apelati subprogramul.

--Problema:

--Dorim sa afisam informatii despre un client anume, al carui tag de utilizator se da ca parametru:

--Informatii personale: nume, prenume, email, gen

--Informatii despre conturile asociate persoanei: iban, sold, carduri daca exista

--Pentru carduri se va afisa tipul si numarul si toate tranzactiile efectuate cu acel card (data, tip_tranzactie, suma)

CREATE OR REPLACE PROCEDURE info_client(tag_utilizator
CLIENT.tag_utilizator%TYPE)

AS

CURSOR personal_client(p_tag_utilizator CLIENT.tag_utilizator%TYPE) IS SELECT

```

p.cod_persoana, nume, prenume, email, gen
    FROM PERSOANA p, CLIENT c WHERE p.cod_persoana = c.cod_persoana AND
c.tag_utilizator = p_tag_utilizator;

    CURSOR conturi(cod_client PERSOANA.cod_persoana%TYPE) IS SELECT
cod_cont, iban, sold,
        CURSOR(SELECT cod_card, tip, card_number FROM CARD WHERE
CARD.cod_cont = CONT.cod_cont)
        FROM CONT WHERE cod_persoana = cod_client;

    CURSOR tranzactii_cu_card(p_cod_card CARD.cod_card%TYPE, p_cod_cont
CONT.cod_cont%TYPE) IS
        WITH tranzactii AS (
            (SELECT cod_tranzactie FROM DEPOZIT d WHERE d.cod_cont = cod_cont
AND d.cod_card = p_cod_card)
            UNION ALL
            (SELECT cod_tranzactie FROM RETRAGERE r WHERE r.cod_cont =
cod_cont AND r.cod_card = p_cod_card)
        )
        SELECT data_tranzactie, tip_tranzactie, valoare FROM TRANZACTIE t
        WHERE t.cod_cont = p_cod_cont AND t.cod_tranzactie IN (SELECT * FROM
tranzactii);

TYPE t_personal_info IS RECORD(
    cod_persoana PERSOANA.cod_persoana%TYPE,
    nume PERSOANA.nume%TYPE,
    prenume PERSOANA.prenume%TYPE,
    email PERSOANA.email%TYPE,
    gen PERSOANA.gen%TYPE
);
    v_personal_info t_personal_info;
    v_cod_cont CONT.cod_cont%TYPE;
    v_ibан CONT.iban%TYPE;
    v_sold CONT.sold%TYPE;
    carduri SYS_REFCURSOR;
    v_cod_card CARD.cod_card%TYPE;
    v_tip CARD.tip%TYPE;
    v_card_number CARD.card_number%TYPE;
    v_verifica BOOLEAN;
BEGIN
    OPEN personal_client(tag_utilizator);
    FETCH personal_client INTO v_personal_info;
    IF personal_client%ROWCOUNT = 0 THEN
        CLOSE personal_client;
        RAISE_APPLICATION_ERROR(-20008, 'Nu s-a gasit persoana cu tag-ul ' ||
```

```

tag_utilizator);
END IF;
CLOSE personal_client;

--a trecut de primul cursor, avem date in v_personal_info
dbms_output.put_line('===== Info Client =====');
dbms_output.put_line('> Nume: ' || v_personal_info.nume);
dbms_output.put_line('> Prenume: ' || v_personal_info.prenume);
dbms_output.put_line('> Email: ' || v_personal_info.email);
dbms_output.put_line('> Gen: ' || UPPER(v_personal_info.gen));
dbms_output.new_line;
OPEN conturi(v_personal_info.cod_persoana);
LOOP
  FETCH conturi INTO v_cod_cont, v_iban, v_sold, carduri;
  EXIT WHEN conturi%NOTFOUND;

  dbms_output.put_line('==== Cont Info ====');
  dbms_output.put_line('>> IBAN: ' || v_iban);
  dbms_output.put_line('>> SOLD: ' || v_sold);
  dbms_output.new_line;

  LOOP
    FETCH carduri INTO v_cod_card, v_tip, v_card_number;
    EXIT WHEN carduri%NOTFOUND;
    dbms_output.put_line('---- Card Info ----');
    dbms_output.put_line('>>> Card Number: ' || v_card_number);
    dbms_output.put_line('>>> Tip: ' || v_tip);
    dbms_output.put_line('>>> Tranzactii: ');
    v_verifica := FALSE;
    FOR v_tranzactie_info IN tranzactii_cu_card(v_cod_card, v_cod_cont) LOOP
      dbms_output.put_line(' - Tranzactie (' || v_tranzactie_info.tip_tranzactie || ')
efectuata pe ' || v_tranzactie_info.data_tranzactie || ' in valoare de ' ||
v_tranzactie_info.valoare);
      v_verifica := TRUE;
    END LOOP;
    IF NOT v_verifica THEN
      dbms_output.put_line('>>> Nu are tranzactii efectuate cu acest card');
    END IF;
    dbms_output.new_line;
  END LOOP;
  IF carduri%ROWCOUNT = 0 THEN
    dbms_output.put_line('Clientul nu are carduri.');
  END IF;
  CLOSE carduri;
END LOOP;

IF conturi%ROWCOUNT = 0 THEN
  dbms_output.put_line('Clientul nu are conturi.');
END IF;

```

```

CLOSE conturi;
END;
/

EXECUTE info_client('UnUserInexistent_');
EXECUTE info_client('melis2_');

```

Worksheet

```

1129:     dbms_output.put_line('>>> Tranzactii: ');
1130:     v_verifica := FALSE;
1131:     FOR v_tranzactie_info IN tranzactii_cu_card(v_cod_card, v_cod_cont) LOOP
1132:         dbms_output.put_line(' - Tranzactie (' || v_tranzactie_info.data_tranzactie || ') efectuata pe ' || v_tranzactie_info.data_tranzactie || ' in valoare de ' || v_tranzactie_info.
1133:         v_verifica := TRUE;
1134:     END LOOP;
1135:     IF NOT v_verifica THEN
1136:         dbms_output.put_line('>>> Nu are tranzactii efectuate cu acest card');
1137:     END IF;
1138:     dbms_output.new_line;
1139: END LOOP;
1140: IF carduri%ROWCOUNT = 0 THEN
1141:     dbms_output.put_line('Clientul nu are carduri.');
1142: END IF;
1143: CLOSE carduri;
1144: END LOOP;
1145:
1146: IF conturi%ROWCOUNT = 0 THEN
1147:     dbms_output.put_line('Clientul nu are conturi.');
1148: END IF;
1149: CLOSE conturi;
1150: /
1151:
1152: EXECUTE info_client('UnUserInexistent_');
1153: EXECUTE info_client('melis2_');
1154:

```

Script Output

```

Procedure INFO_CLIENT compiled

Error starting at line : 1,153 in command -
BEGIN info_client('UnUserInexistent_'); END;
Error report -
ORA-20008: Nu s-a gasit persoana cu tag-ul UnUserInexistent_
ORA-06512: at "DRAGOS.INFO_CLIENT", line 51
ORA-06512: at line 1

```

Worksheet

```

1147:     dbms_output.put_line('Clientul nu are conturi.');
1148: END IF;
1149: CLOSE conturi;
1150: /
1151:
1152: EXECUTE info_client('UnUserInexistent_');
1153: EXECUTE info_client('melis2_');
1154:

```

PL/SQL Output

```

PL/SQL procedure successfully completed.

Dbsa Output
localhost x
=====
Info Client =====
> Name: Pintenaru
> Prenume: Melissa
> Email: melissa.p@gmail.com
> Gen: F

----- Cont Info -----
>> IBAN: RO80RZBR88459683338826
>> SOLD: 3200.2

----- Card Info -----
>> Card Number: 4532962746090018
>> Tip: visa
>> Tranzactii:
- Tranzactie (deposit) efectuata pe 03-MAY-22 in valoare de 300
- Tranzactie (retragere) efectuata pe 18-MAY-22 in valoare de 200

----- Card Info -----
>> Card Number: 4801769871971639
>> Tip: visa
>> Tranzactii:
- Tranzactie (deposit) efectuata pe 12-JAN-22 in valoare de 500
- Tranzactie (deposit) efectuata pe 24-APR-22 in valoare de 200
- Tranzactie (retragere) efectuata pe 17-JAN-22 in valoare de 400
- Tranzactie (retragere) efectuata pe 28-APR-22 in valoare de 150

```

Continuare screenshot anterior pentru a completa output-ul:

```

1147      dbms_output.put_line('Clientul nu are conturi.');
1148  END IF;
1149  CLOSE conturi;
1150 END;
1151 /
1152
1153 EXECUTE info_client('UnUserInexistent_');
1154 EXECUTE info_client('meliis2_');
1155
4

```

Script Output: Task completed in 0.04 seconds

PL/SQL procedure successfully completed.

Dbms Output

Card Info
>>> Card Number: 4801769871971639 >>> Tip: visa >>> Tranzactii: - Tranzactie (depozit) efectuata pe 12-JAN-22 in valoare de 500 - Tranzactie (depozit) efectuata pe 24-APR-22 in valoare de 200 - Tranzactie (retragere) efectuata pe 17-JAN-22 in valoare de 400 - Tranzactie (retragere) efectuata pe 28-APR-22 in valoare de 150
>>> Card Info >>> Card Number: 4929249181139042 >>> Tip: visa >>> Tranzactii: >>> Nu are tranzactii efectuate cu acest card
--<-- Conn_Info --<-- >> IBAN: RO48PORNLS1467747485878989 >> SOLD: 100200.2
--<-- Card Info --<-- >> Card Number: 5196701739892160 >> Tip: mastercard >> Tranzactii: - Tranzactie (depozit) efectuata pe 09-MAR-22 in valoare de 1800 - Tranzactie (retragere) efectuata pe 11-JUL-22 in valoare de 30

8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

--8. Formulati in limbaj natural o problema pe care sa o rezolvati folosind un
-- subprogram stocat independent de tip functie care sa utilizeze intr-o singura
-- comanda SQL 3 dintre tabelele definite. Definiti minim 2 exceptii.
-- Apelati subprogramul astfel incat sa evidentiati toate cazurile tratate.

--Problema:

--PIN-urile cardurilor se uita constant, manager-ul ne-a cerut o metoda de a le recupera
--dupa mai multe criterii.

--1. PIN-ul poate fi cerut de un client, care isi va confirma
--identitatea cu CNP-ul (in mod normal am folosi hash-ul parolei, dar nu avem in model
--deci ne adaptam si presupunem ca CNP-urile se afla la fel de greu ca parbolele)
--2. PIN-ul poate fi cerut de un angajat, care se identifica tot cu CNP-ul sau, doar daca

```

--este admin sau manager.
--Din motive de securitate, daca angajatul lucreaza online ii permitem sa ceara pin-ul
--doar daca are domiciliul in Romania, in caz contrar, se considera ca nu are privilegii.

CREATE OR REPLACE FUNCTION cere_pin(cine_cere
PERSOANA.cod_persoana%TYPE, p_card_number CARD.card_number%TYPE, p_cnp
PERSOANA.cnp%TYPE)
RETURN CARD.pin%TYPE
AS
    cnp_gresit_angajat EXCEPTION;
    cnp_gresit_client EXCEPTION;
    nu_detine_cont EXCEPTION; --are cnp-ul, are cardul, dar nu e detinatorul
    fara_privilegii EXCEPTION; --este angajat, dar nu este admin sau manager
    card_anonim EXCEPTION; --cardul exista dar nu e asociat unui cont, doar angajatii pot
face rost de pin
    card_negasit EXCEPTION;
PRAGMA EXCEPTION_INIT(card_negasit, -20010);

    v_cod_tara TARA.cod_tara%TYPE;
    v_cod_tara2 TARA.cod_tara%TYPE;
    v_cod_persoana PERSOANA.cod_persoana%TYPE;
    v_cod_cont CONT.cod_cont%TYPE;
    v_pin CARD.pin%TYPE;

    v_cod_sediu SEDIU.cod_sediu%TYPE;
    v_cnp PERSOANA.cnp%TYPE;
    v_aux PLS_INTEGER;
    v_aux2 PLS_INTEGER;
BEGIN
    GOTO inceput;
    <<verificare_angajat>>
    --folosim count ca sa nu dea NO_DATA_FOUND
    --COUNT() poate da doar 0 sau 1 in acest caz pentru ca avem cine_cere cheie primara
    SELECT MAX(a.cod_sediu), MAX(l.cod_tara), MAX(p.cnp),
    COUNT(a.cod_persoana), MAX(DECODE(UPPER(a.cod_job), 'MANAGER', 1,
    'ADMIN', 1, 0))
    INTO v_cod_sediu, v_cod_tara2, v_cnp, v_aux, v_aux2
    FROM ANGAJAT a, PERSOANA p, LOCATIE l
    WHERE p.cod_persoana = cine_cere AND a.cod_persoana = cine_cere AND
    p.cod_locatie = l.cod_locatie;

    IF v_aux = 0 THEN --nu exista angajatul
        IF v_cod_cont IS NULL THEN --cardul este anonim
            RAISE card_anonim;
        ELSE --angajatul nu exista dar un proprietar are daca a ajuns aici
            RAISE nu_detine_cont;
        END IF;
    ELSIF v_aux2 = 0 OR (v_cod_sediu IS NULL AND v_cod_tara2 != v_cod_tara) THEN
    --angajatul exista, dar nu este admin sau manager valid

```

```

        RAISE fara_privilegii;
ELSIF v_cnp != p_cnp THEN --e angajat cu privilegii, dar n-a trecut validarea identitatii
        RAISE cnp_gresit_angajat;
ELSE --in sfarsit a trecut si el toate validarile
        RETURN v_pin;
END IF;

<<inceput>>
SELECT cod_tara INTO v_cod_tara FROM TARA WHERE lower(nume_tara) LIKE
'romania';

BEGIN
    SELECT cod_cont, pin INTO v_cod_cont, v_pin FROM CARD WHERE
card_number LIKE p_card_number;
    IF v_cod_cont IS NULL THEN
        --este un card anonim, verificam daca este angajat valid, altfel erori
        --acolo va da fie eroare fie return
        GOTO verificare_angajat;
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20010, 'Cardul introdus nu exista.');
END;

--daca a ajuns aici, card-ul are un cont asociat
BEGIN
    --luam persoana care detine contul
    SELECT p.cod_persoana, cnp INTO v_cod_persoana, v_cnp FROM PERSOANA p,
CONT c WHERE p.cod_persoana = c.cod_persoana AND c.cod_cont = v_cod_cont;

    IF v_cod_persoana != cine_cere THEN
        GOTO verificare_angajat;
    ELSIF v_cnp != p_cnp THEN
        RAISE cnp_gresit_client;
    END IF;
EXCEPTION
    --nu ar trebui sa se intampla datorita cheilor externe, stiind ca v_cod_cont este nenul.
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20011, 'Detinatorul cardului este definit dar
cumva nu exista');
    END;
    RETURN v_pin;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20012, 'Romania nu se afla in lista tarilor,
reparati.');
    WHEN cnp_gresit_angajat THEN
        RAISE_APPLICATION_ERROR(-20013, 'Validarea identificarii angajatului a

```

```

esuat.');
WHEN cnp_gresit_client THEN
    RAISE_APPLICATION_ERROR(-20014, 'Validarea identificarii clientului a esuat.');
WHEN nu_detine_cont THEN
    RAISE_APPLICATION_ERROR(-20015, 'Persoana nu este angajat si nici nu detine
contul asociat cardului introdus.');
WHEN fara_privilegii THEN
    RAISE_APPLICATION_ERROR(-20016, 'Angajatul nu are privilegiile necesare de a
obtine PIN-ul acestui card.');
WHEN card_anonim THEN
    RAISE_APPLICATION_ERROR(-20017, 'Cardul este anonim, iar utilizatorul nu este
un angajat.');
WHEN card_negasit THEN
    RAISE_APPLICATION_ERROR(-20018, 'Numarul introdus nu corespunde niciunui
card.');

WHEN OTHERS THEN
    RAISE;
END;
/

--clientul 1 cere pin-ul pentru cardul sau, folosind cnp-ul corect si numarul de card corect
SELECT cere_pin(1, '4532962746090018', '2870816526255') pin FROM DUAL;

--clientul 1 cere pentru cardul sau, dar cnp-ul este gresit
SELECT cere_pin(1, '4532962746090018', '2870816526256') pin FROM DUAL;

--clientul 1 cere pentru cardul sau, dar numarul de card este gresit
SELECT cere_pin(1, '4532962746090019', '2870816526255') pin FROM DUAL;

--clientul 1 cere pentru cardul altui client
SELECT cere_pin(1, '5336340664664640', '2870816526255') pin FROM DUAL;

--managerul (id 16) cere pentru cardul clientul 1, cu toate datele corecte
SELECT cere_pin(16, '4532962746090018', '1950122298018') pin FROM DUAL;

--managerul (id 16) cere pentru cardul clientul 1, cu cnp-ul gresit
SELECT cere_pin(16, '4532962746090018', '1950122298011') pin FROM DUAL;

--casierul cu id-ul 21 incearca sa obtina pin-ul cardului clientului 1
SELECT cere_pin(21, '4532962746090018', '1950122298018') pin FROM DUAL;

COMMIT;

--pentru a testa card_anonim, facem un card anonim, cel cu cod_number-ul:
5491696102956307
UPDATE CARD SET cod_cont = NULL WHERE card_number LIKE
'5491696102956307';

```

```
--clientul apeleaza cardul anonim
SELECT cere_pin(1, '5491696102956307', '2870816526255') pin FROM DUAL;

--managerul apeleaza cardul anonim
SELECT cere_pin(16, '5491696102956307', '1950122298018') pin FROM DUAL;

--sa testam si eroarea care "invalideaza" intreaga functie, inexistentă Romaniei
UPDATE TARA SET nume_tara = 'Romania2' WHERE lower(nume_tara) = 'romania';

--clientul 1, cardul sau, date corecte
SELECT cere_pin(1, '4532962746090018', '2870816526255') pin FROM DUAL;

ROLLBACK;
```

The screenshot shows three separate query executions in Oracle SQL Developer:

- Query 1:** SELECT cere_pin(1, '4532962746090018', '2870816526255') pin FROM DUAL; Result: PIN 1 1000
- Query 2:** SELECT cere_pin(1, '4532962746090018', '2870816526256') pin FROM DUAL; Error: ORA-20014: Validarea identificarii clientului a esuat.
ORA-06512: at "DRAGOS.CERE_PIN", line 84
- Query 3:** SELECT cere_pin(1, '4532962746090019', '2870816526255') pin FROM DUAL; Error: ORA-20018: Numarul introdus nu corespunde niciunui card.
ORA-06512: at "DRAGOS.CERE_PIN", line 92

```
1279 | --clientul 1 cere pentru cardul altui client
1280 | SELECT cere_pin(1, '5336340664664640', '2870816526255') pin FROM DUAL;
1281 |
```

Script Output x Query Result x
SQL | Executing:SELECT cere_pin(1, '5336340664664640', '2870816526255') pin FROM DUAL in 0 seconds
ORA-20015: Persoana nu este angajat si nici nu detine contul asociat cardului introdus.
ORA-06512: at "DRAGOS.CERE_PIN", line 86

```
1282 | --managerul (id 16) cere pentru cardul clientul 1, cu toate datele corecte
1283 | SELECT cere_pin(16, '4532962746090018', '1950122298018') pin FROM DUAL;
1284 |
```

Script Output x Query Result x
SQL | All Rows Fetched: 1 in 0.002 seconds

PIN
1 1000

```
1285 | --managerul (id 16) cere pentru cardul clientul 1, cu cnp-ul gresit
1286 | SELECT cere_pin(16, '4532962746090018', '1950122298011') pin FROM DUAL;
1287 |
```

Script Output x Query Result x
SQL | Executing:SELECT cere_pin(16, '4532962746090018', '1950122298011') pin FROM DUAL in 0 seconds
ORA-20013: Validarea identificarii angajatului a esuat.
ORA-06512: at "DRAGOS.CERE_PIN", line 82

```
1288 | --casierul cu id-ul 21 incercă sa obtina pin-ul cardului clientului 1
1289 | SELECT cere_pin(21, '4532962746090018', '1950122298018') pin FROM DUAL;
1290 |
```

Script Output x Query Result x
SQL | Executing:SELECT cere_pin(21, '4532962746090018', '1950122298018') pin FROM DUAL in 0 seconds
ORA-20016: Angajatul nu are privilegiile necesare de a obtine PIN-ul acestui card.
ORA-06512: at "DRAGOS.CERE_PIN", line 88

```
1291 | COMMIT;
1292 |
1293 | --pentru a testa card_anonim, facem un card anonim, cel cu cod_number-ul: 5491696102956307
1294 | UPDATE CARD SET cod_cont = NULL WHERE card_number LIKE '5491696102956307';
1295 |
1296 | --clientul apeleaza cardul anonim
1297 | SELECT cere_pin(1, '5491696102956307', '2870816526255') pin FROM DUAL;
1298 |
1299 | --managerul apeleaza cardul anonim
```

Script Output x Query Result x
SQL | Executing:SELECT cere_pin(1, '5491696102956307', '2870816526255') pin FROM DUAL in 0 seconds
ORA-20017: Cardul este anonim, iar utilizatorul nu este un angajat.
ORA-06512: at "DRAGOS.CERE_PIN", line 90

```

1299 | --managerul apeleaza cardul anonim
1300 | SELECT cere_pin(16, '5491696102956307', '1950122298018') pin FROM DUAL;
1301 |

```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0 seconds

	PIN
1	1014

```

1302 | --sa testam si eroarea care "invalideaza" intreaga functie, inexistentă României
1303 | UPDATE TARA SET nume_tara = 'Romania2' WHERE lower(nume_tara) = 'romania';
1304 |
1305 | --clientul 1, cardul sau, date corecte
1306 | SELECT cere_pin(1, '4532962746090018', '2870816526255') pin FROM DUAL;
1307 |

```

Script Output x Query Result x

SQL | Executing:SELECT cere_pin(1, '4532962746090018', '2870816526255') pin FROM DUAL in 0 seconds

ORA-20012: Romania nu se afla in lista tarilor, reparati.
ORA-06512: at "DRAGOS.CERE_PIN", line 80

9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, inclusiv excepțiile NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

--9. Formulati in limbaj natural o problema pe care sa o rezolvati folosind
-- un subprogram stocat independent de tip procedura care sa utilizeze
-- intr-o singura comanda SQL 5 dintre tabelele definite. Tratati toate
-- exceptiile care pot aparea, inclusand exceptiile NO_DATA_FOUND si
-- TOO_MANY_ROWS. Apelati subprogramul astfel incat sa evidentiati toate
-- cazurile tratate.

--Problema:

--Managerul doreste sa faca iar modificari...
--De data asta are nevoie de informatii despre sediile bancii.
--Dandu-se un nume de tara dorim sa obtinem statistici.
--Daca nu avem sediu in tara respectiva, vrem sa stim cate persoane inregistrate
--locuiesc acolo si diverse detalii despre acestea.
--Daca avem un sediu, vrem sa stim detalii despre contractele semnate la acel sediu.
--Daca avem mai multe sedii, vrem sa stim detalii despre angajatii
--care lucreaza fizic in tara respectiva.

```

--alternativa ex9 select 1
--SELECT nume, prenume, a.cod_job, cl.tag_utilizator, COUNT(c.cod_persoana)
nrConturi, l.oras
--FROM PERSOANA p, LOCATIE l, ANGAJAT a, CLIENT cl, CONT c
--WHERE p.cod_persoana = a.cod_persoana(+) AND cl.cod_persoana(+) =
p.cod_persoana
--AND c.cod_persoana(+) = p.cod_persoana AND p.cod_locatie = l.cod_locatie AND
l.cod_tara = 1
--GROUP BY p.cod_persoana, nume, prenume, a.cod_job, cl.tag_utilizator, l.oras;

CREATE OR REPLACE PROCEDURE statistici_sedii(p_numetara
TARA.nume_tara%TYPE)
AS
    v_cod_tara TARA.cod_tara%TYPE;
    tara_inexistenta EXCEPTION;
    v_cod_sediu SEDIU.cod_sediu%TYPE;

    v_aux PLS_INTEGER;
BEGIN
    BEGIN
        SELECT cod_tara INTO v_cod_tara FROM TARA WHERE nume_tara =
p_numetara;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE tara_inexistenta;
        --nu poate da TOO_MANY_ROWS, am pus UNIQUE pe nume_tara
    END;

    SELECT cod_sediu INTO v_cod_sediu FROM LOCATIE l WHERE
s.cod_locatie = l.cod_locatie AND l.cod_tara = v_cod_tara;
    dbms_output.put_line('Un sediu gasit in tara ' || INITCAP(p_numetara));
    dbms_output.put_line('La sediul din aceasta tara au fost semnate urmatoarele contracte:');
    FOR v_detalii_contract IN (SELECT p1.nume_angajat_nume, p1.prenume
angajat_prenume, a.cod_job, p2.nume_client_nume, p2.prenume_client_prenume,
c.tag_utilizator tag_client, c2.titlu_contract
        FROM SEMNEAZA_UN_CONTRACT s, ANGAJAT a, CLIENT c,
TIP_CONTRACT c2, PERSOANA p1, PERSOANA p2
        WHERE s.cod_client = c.cod_persoana AND s.cod_angajat = a.cod_persoana AND
s.tip_contract = c2.tip_contract
        AND p1.cod_persoana = a.cod_persoana AND p2.cod_persoana = c.cod_persoana
        AND s.cod_sediu = v_cod_sediu)
    LOOP
        dbms_output.put_line('> Contract de "' || v_detalii_contract.titlu_contract || " pentru
clientul ' || v_detalii_contract.client_nume || ' ' || v_detalii_contract.client_prenume
        || '(' || v_detalii_contract.tag_client || ') semnat de angajatul ' ||
v_detalii_contract.angajat_nume || ' ' || v_detalii_contract.angajat_prenume
        || '(' || v_detalii_contract.cod_job || ')'
    );

```

```

END LOOP;

EXCEPTION
  WHEN tara_inexistenta THEN
    dbms_output.put_line('Oops, tara introdusa nu se afla in baza de date.');
  WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('> Nu s-a gasit niciun sediu in tara ' || INITCAP(p_numar_tara));
    dbms_output.put_line('In aceasta tara avem inregistrati utilizatori: ');
    --daca nu considerati ca aceasta este o singura instructiune
    --care contine 5 tabele pentru ca sunt subcereri
    --aveti <<alternativa ex9 select 1>> mai sus care e doar join-uri
    --si returneaza acelasi lucru
    v_aux := 0;
    FOR v_detalii_persoana IN (SELECT nume, prenume,
      (SELECT cod_job FROM ANGAJAT a WHERE a.cod_persoana(+) =
      p.cod_persoana) cod_job,
      (SELECT tag_utilizator FROM CLIENT cl WHERE cl.cod_persoana(+) =
      p.cod_persoana) tag_utilizator,
      (SELECT COUNT(cod_cont) FROM CONT c WHERE c.cod_persoana =
      p.cod_persoana) nrConturi,
      l.oras
      FROM PERSOANA p, LOCATIE l WHERE p.cod_locatie = l.cod_locatie AND
      l.cod_tara = v_cod_tara)
    LOOP
      dbms_output.put('>> ' || v_detalii_persoana.nume || ' ' || v_detalii_persoana.prenume
      || ');
      IF v_detalii_persoana.tag_utilizator IS NOT NULL THEN
        dbms_output.put('este client (cu tag-ul "' || v_detalii_persoana.tag_utilizator || "'),
      );
      END IF;
      IF v_detalii_persoana.cod_job IS NOT NULL THEN
        dbms_output.put('este angajat (avand job-ul cu codul "' ||
      v_detalii_persoana.cod_job || "), ');
      END IF;
      dbms_output.put('are ' || v_detalii_persoana.nrConturi || ' conturi si este din orasul '
      || v_detalii_persoana.oras);
      dbms_output.new_line;
      v_aux := v_aux + 1;
    END LOOP;
    IF v_aux = 0 THEN
      dbms_output.put_line('Nu exista utilizatori inregistrati in aceasta tara.');
    ELSE
      dbms_output.put_line('Un total de ' || v_aux || ' utilizatori inregistrati in aceasta
tara.');
    END IF;
  WHEN TOO_MANY_ROWS THEN
    dbms_output.put_line('> In tara ' || INITCAP(p_numar_tara) || ' exista mai multe
sedii.');
    dbms_output.put_line('Activitatea angajatilor care lucreaza fizic in aceasta tara: ');

```

```

v_aux := 0;
FOR v_detalii_angajat IN (
    SELECT nume, prenume, denumire_job,
        (SELECT COUNT(cod_persoana) FROM TRANZACTIE_EXTERNA t
    WHERE t.cod_persoana = a.cod_persoana) nrTranzactiiExterne,
        (SELECT COUNT(cod_angajat) FROM SEMNEAZA_UN_CONTRACT sc
    WHERE sc.cod_angajat = a.cod_persoana) nrContracte
    FROM ANGAJAT a, PERSOANA p, JOB j, SEDIU s, LOCATIE l
    WHERE p.cod_persoana = a.cod_persoana AND j.cod_job = a.cod_job
        AND a.cod_sediu = s.cod_sediu AND s.cod_locatie = l.cod_locatie
        AND l.cod_tara = v_cod_tara)
LOOP
    dbms_output.put_line('> Angajatul ' || v_detalii_angajat.nume || '' ||
v_detalii_angajat.prenume || ' este ' || v_detalii_angajat.denumire_job
    || ', a efectuat ' || v_detalii_angajat.nrTranzactiiExterne || ' tranzactii externe si a
semnat ' || v_detalii_angajat.nrContracte || ' contracte.'
);
    v_aux := 1;
END LOOP;
IF v_aux = 0 THEN
    dbms_output.put_line('Nu exista angajati in aceasta tara.');
END IF;
END;
/
EXECUTE statistici_sedii('TaraInexistenta');
EXECUTE statistici_sedii('Japonia');
EXECUTE statistici_sedii('Olanda');
EXECUTE statistici_sedii('Romania');

```

```

1400      (SELECT COUNT(cod_persoana) FROM TRANZACTIE_EXTERNA t WHERE t.cod_persoana = a.cod_persoana) nrTranzactiiExterne,
1401      (SELECT COUNT(cod_angajat) FROM SEMNEAZA_UN_CONTRACT sc WHERE sc.cod_angajat = a.cod_persoana) nrContracte
1402  FROM ANGAJAT a, PERSOANA p, JOB j, SEDIU s, LOCATIE l
1403  WHERE p.cod_persoana = a.cod_persoana AND j.cod_job = a.cod_job
1404      AND a.cod_sediu = s.cod_sediu AND s.cod_locatie = l.cod_locatie
1405      AND l.cod_tara = v_cod_tara)
1406
1407  dbms_output.put_line('> Angajatul ' || v_detalii_angajat.nume || '' ||
1408      || v_detalii_angajat.prenume || ' este ' || v_detalii_angajat.denumire_job
1409      || ', a efectuat ' || v_detalii_angajat.nrTranzactiiExterne || ' tranzactii externe si a semnat ' || v_detalii_angajat.nrContracte || ' contracte.'
1410 );
1411  v_aux := 1;
1412 END LOOP;
1413 IF v_aux = 0 THEN
1414     dbms_output.put_line('Nu exista angajati in aceasta tara.');
1415 END IF;
1416
1417 EXECUTE statistici_sedii('TaraInexistenta');
1418

```

Script Output | Query Result | Task completed in 0.044 seconds

Procedure STATISTICI_SEDII compiled

PL/SQL procedure successfully completed.

Dbms Output | Buffer Size: 20000 | localhost

Oups, tara introdusa nu se afla in baza de date.

```

1419 | EXECUTE statistici_sedii('Japonia');
      |
      +-----+
      | Script Output x | Query Result x
      | Task completed in 0.039 seconds
      |
      PL/SQL procedure successfully completed.

      PL/SQL procedure successfully completed.

      |
      +-----+
      Dbsms Output
      +-----+
      | Buffer Size: 20000 | | localhost x
      |
      > Nu s-a gasit niciun sediu in tara Japonia
      In aceasta tara avem inregistrati utilizatorii:
      >> Dionisie David este angajat (avand job-ul cu codul 'ADMIN'), are 0 conturi si este din orasul Tokyo
      >> Simion Marian-Eduard este angajat (avand job-ul cu codul 'MOD'), are 0 conturi si este din orasul Hokkaido
      Un total de 2 utilizatori inregistrati in aceasta tara.

      |
      +-----+
      1420 | EXECUTE statistici_sedii('Olanda');
      +-----+
      | Script Output x | Query Result x
      | Task completed in 0.026 seconds
      |
      PL/SQL procedure successfully completed.

      PL/SQL procedure successfully completed.

      PL/SQL procedure successfully completed.

      |
      +-----+
      Dbsms Output
      +-----+
      | Buffer Size: 20000 | | localhost x
      |
      Un sediu gasit in tara Olanda
      La sediul din aceasta tara au fost semnate urmatoarele contracte:
      > Contract de 'Politica de Etica BUSINESS' pentru clientul Dacian Mihai(mihaita) semnat de angajatul Simion Marian-Eduard(MOD)
      > Contract de 'Politica de Economii' pentru clientul Dacian Mihai(mihaita) semnat de angajatul Balan Adelin(ADMIN)
      > Contract de 'Termini si Conditi' pentru clientul Dacian Mihai(mihaita) semnat de angajatul Adam Ioan(MANAGER)
      > Contract de 'Politica de Confidentialitate' pentru clientul Dacian Mihai(mihaita) semnat de angajatul Adam Ioan(MANAGER)

      |
      +-----+
      1421 | EXECUTE statistici_sedii('Romania');
      1422 | +-----+
      | Script Output x | Query Result x
      | Task completed in 0.032 seconds
      |
      PL/SQL procedure successfully completed.

      PL/SQL procedure successfully completed.

      PL/SQL procedure successfully completed.

      |
      +-----+
      Dbsms Output
      +-----+
      | Buffer Size: 20000 | | localhost x
      |
      > In tara Romania exista mai multe sedii.
      Activitatea angajatilor care lucreaza fizic in aceasta tara:
      > Angajatul Balan Adelin este Administrator, a efectuat 1 tranzactii externe si a semnat 11 contracte.
      > Angajatul Dionisie David este Administrator, a efectuat 3 tranzactii externe si a semnat 13 contracte.
      > Angajatul Simion Marian-Eduard este Moderator, a efectuat 0 tranzactii externe si a semnat 10 contracte.
      > Angajatul Alexandrescu Andra-Maria este Casier, a efectuat 3 tranzactii externe si a semnat 0 contracte.
      > Angajatul Anastasescu Radu este Casier, a efectuat 1 tranzactii externe si a semnat 0 contracte.

```

10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

--10. Definiți un trigger de tip LMD la nivel de comandă. Declansati trigger-ul.

--Problema: Din motive de securitate contractele semnate nu pot fi updateate
--si nici sterse. Exceptie de la regula se face in perioada de administratie (ziua de luni)
--sau daca utilizatorul este 'SYS', acesta avand permisiunea sa modifice oricand.

```
CREATE OR REPLACE TRIGGER securitate_contracte
BEFORE UPDATE OR DELETE ON SEMNEAZA_UN_CONTRACT
BEGIN
    IF UPPER(USER) = 'SYS' THEN
        RETURN;
    ELSIF UPPER(TRIM(TO_CHAR(SYSDATE, 'DAY', 'NLS_DATE_LANGUAGE = ROMANIAN'))) = 'LUNI' THEN
        RETURN;
    END IF;
    --nu a trecut validarile, oprim actiunea
    RAISE_APPLICATION_ERROR(-20020, 'Actiune blocata din motive de securitate,
contactati administratorul bazei de date.');
END;
/
```

```
DELETE FROM SEMNEAZA_UN_CONTRACT;
DROP TRIGGER securitate_contracte;
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, the code for the trigger is displayed:

```
1430 CREATE OR REPLACE TRIGGER securitate_contracte
1431 BEFORE UPDATE OR DELETE ON SEMNEAZA_UN_CONTRACT
1432 BEGIN
1433     IF UPPER(USER) = 'SYS' THEN
1434         RETURN;
1435     ELSIF UPPER(TRIM(TO_CHAR(SYSDATE, 'DAY', 'NLS_DATE_LANGUAGE = ROMANIAN'))) = 'LUNI' THEN
1436         RETURN;
1437     END IF;
1438     --nu a trecut validarile, oprim actiunea
1439     RAISE_APPLICATION_ERROR(-20020, 'Actiune blocata din motive de securitate,
1440 contactati administratorul bazei de date.');
1441
1442
1443 DELETE FROM SEMNEAZA_UN_CONTRACT;
1444 ROLLBACK;
```

In the bottom-left pane, the 'Script Output' tab shows the trigger was compiled successfully:

```
Trigger SECURITATE_CONTRACTE compiled
```

In the bottom-right pane, an error message is shown:

```
Error starting at line : 1,443 in command -
DELETE FROM SEMNEAZA_UN_CONTRACT
Error report -
ORA-20020: Actiune blocata din motive de securitate, contactati administratorul bazei de date.
ORA-06512: at "DRAGOS.SECURITATE_CONTRACTE", line 8
ORA-04088: error during execution of trigger 'DRAGOS.SECURITATE_CONTRACTE'
```

To the right of the error message is a calendar for December 2022, highlighting December 30th.

11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

```
--11. Definiti un trigger LMD la nivel de linie. Declansati trigger-ul.

--Problema: Vrem sa automatizam cat de cat depozitele de economii.
--INSERT
--Cand un depozit de economii este inserat, daca nu contine data_start, se va considera a fi
SYSDATE.
--Daca nu contine nici data_sfarsit, aceasta va fi calculata automat pe baza dobanzii.
--UPDATE
--Cand un depozit este revendicat (coloana revendicat e updatata de la 0 la 1) atunci vom
adauga
--in contul asociat valoarea depozitului + profitul obtinut. Daca se incerca o revendicare
--inainte de data_sfarsit, se va arunca o eroare. Daca se va incerca trecerea de la revendicat
--la nerevendicat se va arunca alta eroare. Daca se schimba codul dobanzii, data_start si
data_sfarsit
--vor fi reactualizate
--DELETE
--Daca un depozit nerevendicat este sters, valoarea acestuia va fi adaugata in contul asociat

--cerinta propusa are nevoie si de before si de after, ma doare sufletu
--sa o schimb asa ca puteti lua in considerare doar triggerul de before
--dar pentru utilitate le voi implementa pe ambele

CREATE OR REPLACE TRIGGER auto_depozite_economii_before
BEFORE INSERT OR UPDATE ON DEPOZIT_ECONOMII
FOR EACH ROW
DECLARE
    aux_luni DOBANDA.durata_luni%TYPE;
BEGIN
    IF INSERTING THEN
        IF :NEW.data_start IS NULL THEN
            :NEW.data_start := SYSDATE;
        END IF;

        IF :NEW.data_sfarsit IS NULL THEN
            BEGIN
                SELECT durata_luni INTO aux_luni FROM DOBANDA WHERE
DOBANDA.cod_dobanda = :NEW.cod_dobanda;
                :NEW.data_sfarsit := ADD_MONTHS(SYSDATE, aux_luni);
            EXCEPTION
                WHEN NO_DATA_FOUND THEN --inseamna ca nu exista dobanda, se va
ocupa foreign key-ul
                    dbms_output.put_line('[DEBUG TRIGGER auto_depozite_economii_before]');
                    A intrat pe NO_DATA_FOUND (1);
                END;
            END IF;
        ELSE
            IF :OLD.revendicat = 1 AND :NEW.revendicat = 0 THEN
```

```

        RAISE_APPLICATION_ERROR(-20023, 'Nu se poate trece de la revendicat la
nerevendicat.');
      END IF;

      IF :OLD.cod_dobanda != :NEW.cod_dobanda THEN
        BEGIN

          SELECT durata_luni INTO aux_luni FROM DOBANDA WHERE
DOBANDA.cod_dobanda = :NEW.cod_dobanda;
          :NEW.data_start := SYSDATE;
          :NEW.data_sfarsit := ADD_MONTHS(SYSDATE, aux_luni);
        EXCEPTION
          WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('[DEBUG TRIGGER auto_depozite_economii_before]
A intrat pe NO_DATA_FOUND (2)');
            RETURN; --nu exista dobanda, nu are rost sa facem verificari extra, se ocupa
foreign key-ul
          END;
        END IF;
        IF :OLD.revendicat = 0 AND :NEW.revendicat = 1 AND SYSDATE <
:NEW.data_sfarsit THEN
          RAISE_APPLICATION_ERROR(-20022, 'Nu se poate revendica inainte de
termen.');
        END IF;
      END IF;
    END;
  /

```

CREATE OR REPLACE TRIGGER auto_depozite_economii_after
AFTER DELETE OR UPDATE ON DEPOZIT_ECONOMII
FOR EACH ROW
BEGIN
 IF UPDATING THEN
 IF :OLD.revendicat = 0 AND :NEW.revendicat = 1 THEN
 UPDATE CONT SET sold = sold + :NEW.valoare + :NEW.valoare*(SELECT
procent FROM DOBANDA WHERE DOBANDA.cod_dobanda = :NEW.cod_dobanda);
 END IF;
 ELSE
 IF :OLD.revendicat = 0 THEN
 UPDATE CONT SET sold = sold + :OLD.valoare;
 END IF;
 END IF;
 END IF;
END;
/

--cod_dobanda nu exista
INSERT INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, valoare) VALUES(1, -1,
100);

```

--inserare normala
INSERT INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, valoare) VALUES(1, 1, 100);
--dupa un insert, ultimu element va avea acelasi cod_depozit_economii cu numarul de
depozite create in total
SELECT * FROM DEPOZIT_ECONOMII WHERE cod_cont = 1 AND
cod_depozit_economii = (SELECT depozite_contor FROM CONT_ECONOMII WHERE
cod_cont = 1);

--update de revendicare inainte de termen pe insertul facut anterior
UPDATE DEPOZIT_ECONOMII SET revendicat = 1 WHERE cod_cont = 1 AND
cod_depozit_economii = (SELECT depozite_contor FROM CONT_ECONOMII WHERE
cod_cont = 1);

--update de la revendicat la nerevendicat
SELECT * FROM DEPOZIT_ECONOMII WHERE cod_cont = 1;
UPDATE DEPOZIT_ECONOMII SET revendicat = 0 WHERE cod_cont = 1 AND
cod_depozit_economii = 1;

--revendicare bani + profit
SELECT * FROM CONT WHERE cod_cont = 1; --3200.2 sold
UPDATE DEPOZIT_ECONOMII SET revendicat = 1 WHERE cod_cont = 1 AND
cod_depozit_economii = 3; --depozit de 1000, sold de 4500.2 dupa

--revendicare bani fara profit
DELETE FROM DEPOZIT_ECONOMII WHERE cod_cont = 1 AND
cod_depozit_economii = 3;
SELECT * FROM CONT WHERE cod_cont = 1; --sold de 4200.2 de la 3200

ROLLBACK;
DROP TRIGGER auto_depozite_economii_before;
DROP TRIGGER auto_depozite_economii_after;

```

Worksheet | Query Builder

```

1501     END IF;
1502     IF :OLD.revendicat = 0 AND :NEW.revendicat = 1 AND SYSDATE < :NEW.data_sfarsit THEN
1503         RAISE_APPLICATION_ERROR(-20022, 'Nu se poate revindica inainte de termen.');
1504     END IF;
1505     END IF;
1506 END;
1507 /
1509
1510 CREATE OR REPLACE TRIGGER auto_depozite_economii_after
1511 AFTER DELETE OR UPDATE ON DEPOZIT_ECONOMII
1512 FOR EACH ROW
1513 BEGIN
1514 IF UPDATING THEN
1515     IF :OLD.revendicat = 0 AND :NEW.revendicat = 1 THEN
1516         UPDATE CONT SET sold = sold + :NEW.valoare + (:NEW.valoare * (SELECT procent FROM DOBANDA WHERE DOBANDA.cod_dobanda = :NEW.cod_dobanda));
1517     END IF;
1518     ELSE
1519         IF :OLD.revendicat = 0 THEN
1520             UPDATE CONT SET sold = sold + :OLD.valoare;
1521         END IF;
1522     END IF;
1523 END;
1524
1525

```

Script Output | Query Result | Task completed in 0.05 seconds

Trigger AUTO_DEPOZITE_ECONOMII_BEFORE compiled

Trigger AUTO_DEPOZITE_ECONOMII_AFTER compiled

```

1526 --cod_dobanda nu exista
1527 INSERT INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, valoare) VALUES(1, -1, 100);
1528

```

Script Output | Query Result | Task completed in 0.027 seconds

Trigger AUTO_DEPOZITE_ECONOMII_BEFORE compiled

Trigger AUTO_DEPOZITE_ECONOMII_AFTER compiled

Error starting at line : 1,527 in command -

```

INSERT INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, valoare) VALUES(1, -1, 100)
Error report -
ORA-01400: cannot insert NULL into ("DRAGOS"."DEPOZIT_ECONOMII"."DATA_SFARSIT")

```

Dbms Output

+ | Buffer Size: 20000 |

localhost

[DEBUG TRIGGER auto_depozite_economii_before] A intrat pe NO_DATA_FOUND (1)

```

1529 --inserare normala
1530 INSERT INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, valoare) VALUES(1, 1, 100);
1531 --dupa un insert, ultimu element va avea acelasi cod_depozit_economii cu numarul de depozite create in total
1532 SELECT * FROM DEPOZIT_ECONOMII WHERE cod_cont = 1 AND cod_depozit_economii = (SELECT depozite_contor FROM CONT_ECONOMII WHERE cod_cont = 1);
1533

```

Script Output | Query Result | Task completed in 0.024 seconds

1 row inserted.

```

1529 | --inserare normala
1530 | INSERT INTO DEPOZIT_ECONOMII(cod_cont, cod_dobanda, valoare) VALUES(1, 1, 100);
1531 | --dupa un insert, ultimul element va avea acelasi cod_depozit_economii cu numarul de depozite create in total
1532 | SELECT * FROM DEPOZIT_ECONOMII WHERE cod_cont = 1 AND cod_depozit_economii = (SELECT depozite_contor FROM CONT_ECONOMII WHERE cod_cont = 1);
1533 |

```

Script Output | Query Result | SQL | All Rows Fetched: 1 in 0 seconds

COD_DEPOZIT_ECONOMII	COD_CONT	COD_DOBANDA	DATA_START	DATA_SFARSIT	VALOARE	REVENDICAT
1	4	1	1 30-DEC-22	30-MAR-23	100	0


```

1534 | --update de revendicare inainte de termen pe insertul facut anterior
1535 | UPDATE DEPOZIT_ECONOMII SET revendicat = 1 WHERE cod_cont = 1 AND cod_depozit_economii = (SELECT depozite_contor FROM CONT_ECONOMII WHERE cod_cont = 1);
1536 |

```

Script Output | Query Result | SQL | Task completed in 0.037 seconds

1 row inserted.


```

Error starting at line : 1,535 in command -
UPDATE DEPOZIT_ECONOMII SET revendicat = 1 WHERE cod_cont = 1 AND cod_depozit_economii = (SELECT depozite_contor FROM CONT_ECONOMII WHERE cod_cont = 1)
Error report -
ORA-20022: Nu se poate revendica inainte de termen.
ORA-06512: at "DRAGOS.AUTO_DEPOZITE_ECONOMII_BEFORE", line 36
ORA-04088: error during execution of trigger 'DRAGOS.AUTO_DEPOZITE_ECONOMII BEFORE'

```



```

1537 | --update de la revendicat la nerevendicat
1538 | SELECT * FROM DEPOZIT_ECONOMII WHERE cod_cont = 1;
1539 | UPDATE DEPOZIT_ECONOMII SET revendicat = 0 WHERE cod_cont = 1 AND cod_depozit_economii = 1;

```

Script Output | Query Result | SQL | All Rows Fetched: 4 in 0.002 seconds

COD_DEPOZIT_ECONOMII	COD_CONT	COD_DOBANDA	DATA_START	DATA_SFARSIT	VALOARE	REVENDICAT
1	1	1	1 19-JAN-22	19-APR-22	300	1
2	2	1	1 08-MAY-22	08-AUG-22	210	0
3	3	1	2 25-MAR-22	25-SEP-22	1000	0
4	4	1	1 30-DEC-22	30-MAR-23	100	0


```

1537 | --update de la revendicat la nerevendicat
1538 | SELECT * FROM DEPOZIT_ECONOMII WHERE cod_cont = 1;
1539 | UPDATE DEPOZIT_ECONOMII SET revendicat = 0 WHERE cod_cont = 1 AND cod_depozit_economii = 1;
1540 |

```

Script Output | Query Result | SQL | Task completed in 0.035 seconds


```

Error starting at line : 1,539 in command -
UPDATE DEPOZIT_ECONOMII SET revendicat = 0 WHERE cod_cont = 1 AND cod_depozit_economii = 1
Error report -
ORA-20023: Nu se poate trece de la revendicat la nerevendicat.
ORA-06512: at "DRAGOS.AUTO_DEPOZITE_ECONOMII BEFORE", line 20
ORA-04088: error during execution of trigger 'DRAGOS.AUTO_DEPOZITE_ECONOMII BEFORE'

```



```

1541 | --revendicare bani + profit
1542 | SELECT * FROM CONT WHERE cod_cont = 1; --3200.2 sold
1543 | UPDATE DEPOZIT_ECONOMII SET revendicat = 1 WHERE cod_cont = 1 AND cod_depozit_economii = 3; --depozit de 1000, sold de 4500.2 dupa

```

Script Output | Query Result | SQL | All Rows Fetched: 1 in 0 seconds

COD_CONT	COD_PERSOANA	SOLD	IBAN
1	1	1 3200.2 R080RZBR884596538338826	

```

1541 | --revendicare bani + profit
1542 | SELECT * FROM CONT WHERE cod_cont = 1; --3200.2 sold
1543 | UPDATE DEPOZIT_ECONOMII SET revendicat = 1 WHERE cod_cont = 1 AND cod_depozit_economii = 3; --depozit de 1000, sold de 4500.2 dupa
1544 |
1545 | Script Output x | Query Result x
1546 | Task completed in 0.035 seconds
1547 |
1548 | 1 row updated.

1541 | --revendicare bani + profit
1542 | SELECT * FROM CONT WHERE cod_cont = 1; --3200.2 sold
1543 | UPDATE DEPOZIT_ECONOMII SET revendicat = 1 WHERE cod_cont = 1 AND cod_depozit_economii = 3; --depozit de 1000, sold de 4500.2 dupa
1544 |
1545 | Script Output x | Query Result x
1546 | SQL | All Rows Fetched: 1 in 0 seconds
1547 | COD_CONT COD_PERSOANA SOLD IBAN
1548 | 1 1 4500.2 R080RZBR8845968383338826
1549 |

1545 | --revendicare bani fara profit
1546 | DELETE FROM DEPOZIT_ECONOMII WHERE cod_cont = 1 AND cod_depozit_economii = 3;
1547 | SELECT * FROM CONT WHERE cod_cont = 1; --sold de 4200.2 de la 3200
1548 |
1549 | ROLLBACK;

1545 | Script Output x | Query Result x
1546 | Task completed in 0.024 seconds
1547 |
1548 | 1 row updated.

1545 | Rollback complete.

1545 | 1 row deleted.

1545 | --revendicare bani fara profit
1546 | DELETE FROM DEPOZIT_ECONOMII WHERE cod_cont = 1 AND cod_depozit_economii = 3;
1547 | SELECT * FROM CONT WHERE cod_cont = 1; --sold de 4200.2 de la 3200
1548 |
1549 | ROLLBACK;

1545 | Script Output x | Query Result x
1546 | SQL | All Rows Fetched: 1 in 0.001 seconds
1547 | COD_CONT COD_PERSOANA SOLD IBAN
1548 | 1 1 4200.2 R080RZBR8845968383338826

```

12. Definiți un trigger de tip LDD. Declanșați trigger-ul.

--12. Definiti un trigger de tip LDD. Declansati trigger-ul.

--Problema: Faceti un trigger care sa protejeze schema conceputa.

--Daca utilizatorul este 'SYS', acesta poate sa faca ce modificari vrea.

--Aceasta regula se aplica doar pentru tabelele si sevenetele create

--la exercitiul 4

CREATE OR REPLACE TRIGGER protejeaza_structura

BEFORE ALTER OR DROP OR CREATE ON SCHEMA

BEGIN

 IF UPPER(USER) = 'SYS' THEN

 RETURN;

 END IF;

 IF UPPER(SYS.DICTIONARY_OBJ_TYPE) = 'SEQUENCE' THEN

 IF UPPER(SYS.DICTIONARY_OBJ_NAME) NOT IN

('GENERATOR_COD_CONT', 'GENERATOR_COD_SEDIU',

'GENERATOR_COD_LOCATIE', 'GENERATOR_COD_TARA',

 'GENERATOR_CODDOBANDA',

'GENERATOR_COD_BANCOMAT', 'GENERATOR_COD_CARD',

'GENERATOR_COD_DEFAULT_PIN')

 THEN

 RETURN;

 END IF;

 ELSIF UPPER(SYS.DICTIONARY_OBJ_TYPE) = 'TABLE' THEN

 IF UPPER(SYS.DICTIONARY_OBJ_NAME) NOT IN ('TARA', 'LOCATIE',

'SEDIU', 'JOB', 'TIP_CONTRACT', 'DOBANDA', 'BANCOMAT', 'PERSOANA',

'CLIENT', 'ANGAJAT', 'CONT',

 'CONT_ECONOMII', 'DEPOZIT_ECONOMII', 'CARD',

'ISTORIC', 'TRANZACTIE_EXTERNA', 'TRANZACTIE',

'SEMNEAZA_UN_CONTRACT',

 'TRANSFER_TRIMIS', 'TRANSFER_PRIMIT',

'DEPOZIT', 'RETRAGERE')

 THEN

 RETURN;

 END IF;

 ELSE

 RETURN;

 END IF;

 RAISE_APPLICATION_ERROR(-20030, 'Nu puteti face modificari structurale, contactati administratorul.');

 END;

/

DROP TABLE TARA;

DROP SEQUENCE generator_cod_cont;

CREATE TABLE TEST(id NUMBER(1));

DROP TABLE TEST;

DROP TRIGGER protejeaza_structura;

```

1574      END IF;
1575      ELSIF UPPER(SYS.DICTIONARY_OBJ_TYPE) = 'TABLE' THEN
1576          IF UPPER(SYS.DICTIONARY_OBJ_NAME) NOT IN ('TARA', 'LOCATIE', 'SEDIU', 'JOB', 'TIP_CONTRACT', 'DOBANDA', 'BANCOMAT', 'PERSOANA', 'CLIENT', 'ANGAJAT', 'CONT',
1577              'CONT_ECONOMII', 'DEPOZIT_ECONOMII', 'CARD', 'istoric', 'TRANZACTIE_EXTERNA', 'TRANZACTIE', 'SEMNEAZA_UN_CONTRACT',
1578              'TRANSFER_TRIMIS', 'TRANSFER_PRIMIT', 'DEPOZIT', 'RETRAGERE')
1579          THEN
1580              RETURN;
1581          END IF;
1582      ELSE
1583          RETURN;
1584      END IF;
1585      RAISE_APPLICATION_ERROR(-20030, 'Nu puteti face modificarile structurale, contactati administratorul.');
1586  END;
1587 /
1588
1589 DROP TABLE TARA;
1590 DROP SEQUENCE generator_cod_cont;
1591 CREATE TABLE TEST(id NUMBER(1));
1592

```

Script Output | Query Result | Task completed in 0.031 seconds

Trigger PROTEJEAZA_STRUCTURA compiled

Error starting at line : 1,589 in command -
DROP TABLE TARA
Error report -
ORA-00604: error occurred at recursive SQL level 1
ORA-20030: Nu puteti face modificarile structurale, contactati administratorul.
ORA-06512: at line 22
00604. 00000 - "error occurred at recursive SQL level %s"
*Cause: An error occurred while processing a recursive SQL statement
(a statement applying to internal dictionary tables).
*Action: If the situation described in the next error on the stack

13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.

```

CREATE OR REPLACE PACKAGE cerinte_proiect
AS
    --ex6
    PROCEDURE activitate_utilizatori(activ_max PLS_INTEGER DEFAULT -1);
    --ex7
    PROCEDURE info_client(tag_utilizator CLIENT.tag_utilizator%TYPE);
    --ex8
    FUNCTION cere_pin(cine_cere PERSOANA.cod_persoana%TYPE, p_card_number
    CARD.card_number%TYPE, p_cnp PERSOANA.cnp%TYPE) RETURN
    CARD.pin%TYPE;
    --ex9
    PROCEDURE statistici_sedii(p_nume_tara TARA.nume_tara%TYPE);
END cerinte_proiect;
/

```

```

CREATE OR REPLACE PACKAGE BODY cerinte_proiect
AS
    PROCEDURE activitate_utilizatori(activ_max PLS_INTEGER DEFAULT -1)
    AS
        TYPE t_info_user IS RECORD(

```

```

email PERSOANA.email%TYPE,
username CLIENT.tag_utilizator%TYPE,
punctaj PLS_INTEGER DEFAULT 0
);

TYPE t_idx IS TABLE OF t_info_user INDEX BY PLS_INTEGER;
v_useri t_idx;
v_index_user PLS_INTEGER;

TYPE t_punctaj IS RECORD(
cod_persoana PERSOANA.cod_persoana%TYPE,
punctaj PLS_INTEGER
);

TYPE t_imb_punctaj IS TABLE OF t_punctaj;
v_adaos_punctaj t_imb_punctaj; --nu are rost sa initializam, folosim bulk collect

TYPE t_info_user_cu_cod IS RECORD(
cod_persoana PERSOANA.cod_persoana%TYPE,
email PERSOANA.email%TYPE,
username CLIENT.tag_utilizator%TYPE
);

TYPE t_vector_info_user IS VARRAY(2000) OF t_info_user_cu_cod; --suntem o
banca mica, ne ajunge maxim 2000, marim la nevoie
v_aux_1 t_vector_info_user; --nu are rost sa initializam, folosim bulk collect
BEGIN
--ne abtinem din a folosi un ciclu cursor, doar de dragul de a folosi un vector
SELECT p.cod_persoana, email, tag_utilizator BULK COLLECT INTO v_aux_1
FROM PERSOANA p, CLIENT c WHERE p.cod_persoana = c.cod_persoana;

IF v_aux_1.count = 0 THEN --nu exista utilizatori, nu are rost sa facem calcule
dbms_output.put_line('Nu exista utilizatori care sa aiba activitate.');
RETURN;
END IF;

FOR i IN 1..v_aux_1.count LOOP
v_useri(v_aux_1(i).cod_persoana).email := v_aux_1(i).email;
v_useri(v_aux_1(i).cod_persoana).username := v_aux_1(i).username;
END LOOP;

--calculam punctele din depozite/retragere/transferuri trimise si stocam in
v_adaos_punctaj
WITH
tranzactiiPerCont AS (SELECT cod_cont, COUNT(cod_tranzactie) tranzactii
FROM TRANZACTIE WHERE DECODE(lower(tip_tranzactie),
'depozit', 1, 'retragere', 1, 'transfer Trimis', 1, 0) = 1
GROUP BY cod_cont)
SELECT c.cod_persoana, SUM(tranzactii) BULK COLLECT INTO v_adaos_punctaj

```

```

FROM CONT c, tranzactiiPerCont
WHERE c.cod_cont = tranzactiiPerCont.cod_cont GROUP BY c.cod_persoana;

--fiind inserat cu bulk collect, stiu ca este dens
FOR i IN 1..v_adaos_punctaj.count LOOP
    v_useri(v_adaos_punctaj(i).cod_persoana).punctaj :=
v_useri(v_adaos_punctaj(i).cod_persoana).punctaj + v_adaos_punctaj(i).punctaj;
    IF activ_max != -1 AND
v_useri(v_adaos_punctaj(i).cod_persoana).punctaj>activ_max THEN
        v_useri.delete(v_adaos_punctaj(i).cod_persoana); --pragul superior s-a depasit,
nu ne mai intereseaza utilizatorul
    END IF;
END LOOP;

--calculam punctele pentru carduri detinute
WITH
carduriPerCont AS (SELECT cod_cont, COUNT(cod_card) carduri FROM CARD
GROUP BY cod_cont)
    SELECT c.cod_persoana, SUM(carduri)*5 BULK COLLECT INTO v_adaos_punctaj
FROM CONT c, carduriPerCont
WHERE c.cod_cont = carduriPerCont.cod_cont GROUP BY c.cod_persoana;

--acum va trebui sa verificam si daca exista persoana in lista
--daca nu exista inseamna ca a fost scoasa pentru ca limita a fost depasita
--la un pas anterior
FOR i IN 1..v_adaos_punctaj.count LOOP
    IF NOT v_useri.exists(v_adaos_punctaj(i).cod_persoana) THEN
        CONTINUE;
    END IF;

    v_useri(v_adaos_punctaj(i).cod_persoana).punctaj :=
v_useri(v_adaos_punctaj(i).cod_persoana).punctaj + v_adaos_punctaj(i).punctaj;
    IF activ_max != -1 AND
v_useri(v_adaos_punctaj(i).cod_persoana).punctaj>activ_max THEN
        v_useri.delete(v_adaos_punctaj(i).cod_persoana); --pragul superior s-a depasit,
nu ne mai intereseaza utilizatorul
    END IF;
END LOOP;

--schimbam din with in join pe 3 tabele
    SELECT c.cod_persoana, SUM(durata_luni)*2 BULK COLLECT INTO
v_adaos_punctaj
    FROM DEPOZIT_ECONOMII dE, DOBANDA d, CONT c WHERE c.cod_cont =
dE.cod_cont AND dE.cod_dobanda = d.cod_dobanda GROUP BY c.cod_persoana;

--neschimbat fata de for-ul anterior
FOR i IN 1..v_adaos_punctaj.count LOOP

```

```

        IF NOT v_useri.exists(v_adaos_punctaj(i).cod_persoana) THEN
            CONTINUE;
        END IF;

            v_useri(v_adaos_punctaj(i).cod_persoana).punctaj :=
v_useri(v_adaos_punctaj(i).cod_persoana).punctaj + v_adaos_punctaj(i).punctaj;
            IF activ_max != -1 AND
v_useri(v_adaos_punctaj(i).cod_persoana).punctaj > activ_max THEN
                v_useri.delete(v_adaos_punctaj(i).cod_persoana); --pragul superior s-a depasit,
nu ne mai intereseaza utilizatorul
            END IF;
        END LOOP;

        IF v_useri.first IS NULL THEN
            dbms_output.put_line('Nu exista utilizatori care sa aiba activitate.');
            RETURN;
        END IF;

        dbms_output.put_line('===== Puncte de activitate =====');
        v_index_user := v_useri.first;
        WHILE v_index_user IS NOT NULL LOOP
            dbms_output.put_line('> ' || v_useri(v_index_user).username || '(' ||
v_useri(v_index_user).email || ') - ' || v_useri(v_index_user).punctaj);
            v_index_user := v_useri.next(v_index_user);
        END LOOP;
    END;

    PROCEDURE info_client(tag_utilizator CLIENT.tag_utilizator%TYPE)
    AS
        CURSOR personal_client(p_tag_utilizator CLIENT.tag_utilizator%TYPE) IS
SELECT p.cod_persoana, nume, prenume, email, gen
        FROM PERSOANA p, CLIENT c WHERE p.cod_persoana = c.cod_persoana
AND c.tag_utilizator = p_tag_utilizator;

        CURSOR conturi(cod_client PERSOANA.cod_persoana%TYPE) IS SELECT
cod_cont, iban, sold,
        CURSOR(SELECT cod_card, tip, card_number FROM CARD WHERE
CARD.cod_cont = CONT.cod_cont)
        FROM CONT WHERE cod_persoana = cod_client;

        CURSOR tranzactii_cu_card(p_cod_card CARD.cod_card%TYPE, p_cod_cont
CONT.cod_cont%TYPE) IS
        WITH tranzactii AS (
        (SELECT cod_tranzactie FROM DEPOZIT d WHERE d.cod_cont = cod_cont
AND d.cod_card = p_cod_card)
        UNION ALL
        (SELECT cod_tranzactie FROM RETRAGERE r WHERE r.cod_cont =
cod_cont AND r.cod_card = p_cod_card)
        )

```

```

SELECT data_tranzactie, tip_tranzactie, valoare FROM TRANZACTIE t
WHERE t.cod_cont = p_cod_cont AND t.cod_tranzactie IN (SELECT * FROM
tranzactii);

TYPE t_personal_info IS RECORD(
cod_persoana PERSOANA.cod_persoana%TYPE,
nume PERSOANA.nume%TYPE,
prenume PERSOANA.prenume%TYPE,
email PERSOANA.email%TYPE,
gen PERSOANA.gen%TYPE
);

v_personal_info t_personal_info;

v_cod_cont CONT.cod_cont%TYPE;
v_ibан CONT.iban%TYPE;
v_sold CONT.sold%TYPE;
carduri SYS_REFCURSOR;

v_cod_card CARD.cod_card%TYPE;
v_tip CARD.tip%TYPE;
v_card_number CARD.card_number%TYPE;

v_verifica BOOLEAN;

BEGIN
OPEN personal_client(tag_utilizator);
FETCH personal_client INTO v_personal_info;
IF personal_client%ROWCOUNT = 0 THEN
CLOSE personal_client;
RAISE_APPLICATION_ERROR(-20008, 'Nu s-a gasit persoana cu tag-ul ' ||
tag_utilizator);
END IF;
CLOSE personal_client;

--a trecut de primul cursor, avem date in v_personal_info
dbms_output.put_line('===== Info Client =====');
dbms_output.put_line('> Nume: ' || v_personal_info.nume);
dbms_output.put_line('> Prenume: ' || v_personal_info.prenume);
dbms_output.put_line('> Email: ' || v_personal_info.email);
dbms_output.put_line('> Gen: ' || UPPER(v_personal_info.gen));
dbms_output.new_line;
OPEN conturi(v_personal_info.cod_persoana);
LOOP
FETCH conturi INTO v_cod_cont, v_ibан, v_sold, carduri;
EXIT WHEN conturi%NOTFOUND;

dbms_output.put_line('---- Cont Info ----');
dbms_output.put_line('>> IBAN: ' || v_ibан);
dbms_output.put_line('>> SOLD: ' || v_sold);

```

```

dbms_output.new_line;

LOOP
    FETCH carduri INTO v_cod_card, v_tip, v_card_number;
    EXIT WHEN carduri%NOTFOUND;
    dbms_output.put_line('----- Card Info -----');
    dbms_output.put_line('>>> Card Number: ' || v_card_number);
    dbms_output.put_line('>>> Tip: ' || v_tip);
    dbms_output.put_line('>>> Tranzactii: ');
    v_verifica := FALSE;
    FOR v_tranzactie_info IN tranzactii_cu_card(v_cod_card, v_cod_cont) LOOP
        dbms_output.put_line(' - Tranzactie (' || v_tranzactie_info.tip_tranzactie || ')
efectuata pe ' || v_tranzactie_info.data_tranzactie || ' in valoare de ' ||
v_tranzactie_info.valoare);
        v_verifica := TRUE;
    END LOOP;
    IF NOT v_verifica THEN
        dbms_output.put_line('>>>> Nu are tranzactii efectuate cu acest card');
    END IF;
    dbms_output.new_line;
END LOOP;
IF carduri%ROWCOUNT = 0 THEN
    dbms_output.put_line('Clientul nu are carduri.');
END IF;
CLOSE carduri;
END LOOP;

IF conturi%ROWCOUNT = 0 THEN
    dbms_output.put_line('Clientul nu are conturi.');
END IF;
CLOSE conturi;
END;

FUNCTION cere_pin(cine_cere PERSOANA.cod_persoana%TYPE, p_card_number
CARD.card_number%TYPE, p_cnp PERSOANA.cnp%TYPE) RETURN
CARD.pin%TYPE
AS
cnp_gresit_angajat EXCEPTION;
cnp_gresit_client EXCEPTION;
nu_detine_cont EXCEPTION; --are cnp-ul, are cardul, dar nu e detinatorul
fara_privilegii EXCEPTION; --este angajat, dar nu este admin sau manager
card_anonim EXCEPTION; --cardul exista dar nu e asociat unui cont, doar angajatii
pot face rost de pin
card_negasit EXCEPTION;
PRAGMA EXCEPTION_INIT(card_negasit, -20010);

v_cod_tara TARA.cod_tara%TYPE;
v_cod_tara2 TARA.cod_tara%TYPE;
v_cod_persoana PERSOANA.cod_persoana%TYPE;

```

```

v_cod_cont CONT.cod_cont%TYPE;
v_pin CARD.pin%TYPE;

v_cod_sediu SEDIU.cod_sediu%TYPE;
v_cnp PERSOANA.cnp%TYPE;
v_aux PLS_INTEGER;
v_aux2 PLS_INTEGER;

BEGIN
    GOTO inceput;
    <<verificare_angajat>>
    --folosim count ca sa nu dea NO_DATA_FOUND
    --COUNT() poate da doar 0 sau 1 in acest caz pentru ca avem cine_cere cheie primara
    SELECT MAX(a.cod_sediu), MAX(l.cod_tara), MAX(p.cnp),
    COUNT(a.cod_persoana), MAX(DECODE(UPPER(a.cod_job), 'MANAGER', 1,
    'ADMIN', 1, 0))
    INTO v_cod_sediu, v_cod_tara2, v_cnp, v_aux, v_aux2
    FROM ANGAJAT a, PERSOANA p, LOCATIE l
    WHERE p.cod_persoana = cine_cere AND a.cod_persoana = cine_cere AND
    p.cod_locatie = l.cod_locatie;

    IF v_aux = 0 THEN --nu exista angajatul
        IF v_cod_cont IS NULL THEN --cardul este anonim
            RAISE card_anonim;
        ELSE --angajatul nu exista dar un proprietar are daca a ajuns aici
            RAISE nu_detine_cont;
        END IF;
    ELSIF v_aux2 = 0 OR (v_cod_sediu IS NULL AND v_cod_tara2 != v_cod_tara)
    THEN --angajatul exista, dar nu este admin sau manager valid
        RAISE fara_privilegii;
    ELSIF v_cnp != p_cnp THEN --e angajat cu privilegii, dar n-a trecut validarea
    identitatii
        RAISE cnp_gresit_angajat;
    ELSE --in sfarsit a trecut si el toate validarile
        RETURN v_pin;
    END IF;

    <<inceput>>
    SELECT cod_tara INTO v_cod_tara FROM TARA WHERE lower(nume_tara) LIKE
    'romania';

    BEGIN
        SELECT cod_cont, pin INTO v_cod_cont, v_pin FROM CARD WHERE
        card_number LIKE p_card_number;
        IF v_cod_cont IS NULL THEN
            --este un card anonim, verificam daca este angajat valid, altfel erori
            --acolo va da fie eroare fie return
            GOTO verificare_angajat;
        END IF;

```

```

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20010, 'Cardul introdus nu exista.');
  END;

--daca a ajuns aici, card-ul are un cont asociat
BEGIN
  --luam persoana care detine contul
  SELECT p.cod_persoana, cnp INTO v_cod_persoana, v_cnp FROM PERSOANA
  p, CONT c WHERE p.cod_persoana = c.cod_persoana AND c.cod_cont = v_cod_cont;

  IF v_cod_persoana != cine_cere THEN
    GOTO verificare_angajat;
  ELSIF v_cnp != p_cnp THEN
    RAISE cnp_gresit_client;
  END IF;
EXCEPTION
  --nu ar trebui sa se intample datorita cheilor externe, stiind ca v_cod_cont este
  nenul.
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20011, 'Detinatorul cardului este definit dar
cumva nu exista');
  END;
  RETURN v_pin;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20012, 'Romania nu se afla in lista tarilor,
reparati.');
  WHEN cnp_gresit_angajat THEN
    RAISE_APPLICATION_ERROR(-20013, 'Validarea identificarii angajatului a
esuat.');
  WHEN cnp_gresit_client THEN
    RAISE_APPLICATION_ERROR(-20014, 'Validarea identificarii clientului a
esuat.');
  WHEN nu_detine_cont THEN
    RAISE_APPLICATION_ERROR(-20015, 'Persoana nu este angajat si nici nu
detine contul asociat cardului introdus.');
  WHEN fara_privilegii THEN
    RAISE_APPLICATION_ERROR(-20016, 'Angajatul nu are privilegiile necesare
de a obtine PIN-ul acestui card.');
  WHEN card_anonim THEN
    RAISE_APPLICATION_ERROR(-20017, 'Cardul este anonim, iar utilizatorul nu
este un angajat.');
  WHEN card_negasit THEN
    RAISE_APPLICATION_ERROR(-20018, 'Numarul introdus nu corespunde
niciunui card.');
  WHEN OTHERS THEN
    RAISE;

```

```

END;

PROCEDURE statistici_sedii(p_numere_tara TARA.nume_tara%TYPE)
AS
    v_cod_tara TARA.cod_tara%TYPE;
    tara_inexistenta EXCEPTION;
    v_cod_sediu SEDIU.cod_sediu%TYPE;

    v_aux PLS_INTEGER;
BEGIN
    BEGIN
        SELECT cod_tara INTO v_cod_tara FROM TARA WHERE nume_tara =
p_numere_tara;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE tara_inexistenta;
        --nu poate da TOO_MANY_ROWS, am pus UNIQUE pe nume_tara
    END;

    SELECT cod_sediu INTO v_cod_sediu FROM SEDIU s, LOCATIE l WHERE
s.cod_locatie = l.cod_locatie AND l.cod_tara = v_cod_tara;
    dbms_output.put_line('Un sediu gasit in tara ' || INITCAP(p_numere_tara));
    dbms_output.put_line('La sediul din aceasta tara au fost semnate urmatoarele
contracte: ');
    FOR v_detalii_contract IN (SELECT p1.nume_angajat_nume, p1.prenume_
angajat_prenume, a.cod_job, p2.nume_client_nume, p2.prenume_client_prenume,
c.tag_utilizator_tag_client, c2.titlu_contract
        FROM SEMNEAZA_UN_CONTRACT s, ANGAJAT a, CLIENT c,
TIP_CONTRACT c2, PERSOANA p1, PERSOANA p2
        WHERE s.cod_client = c.cod_persoana AND s.cod_angajat = a.cod_persoana AND
s.tip_contract = c2.tip_contract
        AND p1.cod_persoana = a.cod_persoana AND p2.cod_persoana = c.cod_persoana
        AND s.cod_sediu = v_cod_sediu)
    LOOP
        dbms_output.put_line('> Contract de "' || v_detalii_contract.titlu_contract || '" pentru
clientul ' || v_detalii_contract.client_nume || ' ' || v_detalii_contract.client_prenume
        || '(' || v_detalii_contract.tag_client || ') semnat de angajatul ' ||
v_detalii_contract.angajat_nume || ' ' || v_detalii_contract.angajat_prenume
        || '(' || v_detalii_contract.cod_job || ')'
    );
    END LOOP;

EXCEPTION
    WHEN tara_inexistenta THEN
        dbms_output.put_line('Oops, tara introdusa nu se afla in baza de date.');
    WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('> Nu s-a gasit niciun sediu in tara ' ||
INITCAP(p_numere_tara));
        dbms_output.put_line('In aceasta tara avem inregistrati utilizatorii: ');

```

```

--daca nu considerati ca aceasta este o singura instructiune
--care contine 5 table pentru ca sunt subcereri
--aveti <<alternativa ex9 select 1>> mai sus care e doar join-uri
--si returneaza acelasi lucru
v_aux := 0;
FOR v_detalii_persoana IN (SELECT nume, prenume,
                               (SELECT cod_job FROM ANGAJAT a WHERE a.cod_persoana(+) =
p.cod_persoana) cod_job,
                               (SELECT tag_utilizator FROM CLIENT cl WHERE cl.cod_persoana(+) =
p.cod_persoana) tag_utilizator,
                               (SELECT COUNT(cod_cont) FROM CONT c WHERE c.cod_persoana =
p.cod_persoana) nrConturi,
                               l.oras
                         FROM PERSOANA p, LOCATIE l WHERE p.cod_locatie = l.cod_locatie AND
l.cod_tara = v_cod_tara)
LOOP
    dbms_output.put('>> ' || v_detalii_persoana.nume || ' ' ||
v_detalii_persoana.prenume || ' ');
    IF v_detalii_persoana.tag_utilizator IS NOT NULL THEN
        dbms_output.put('este client (cu tag-ul "' || v_detalii_persoana.tag_utilizator || '''), ');
    END IF;
    IF v_detalii_persoana.cod_job IS NOT NULL THEN
        dbms_output.put('este angajat (avand job-ul cu codul "' || v_detalii_persoana.cod_job || '''), ');
    END IF;
    dbms_output.put('are ' || v_detalii_persoana.nrConturi || ' conturi si este din orasul
' || v_detalii_persoana.oras);
    dbms_output.new_line;
    v_aux := v_aux + 1;
END LOOP;
IF v_aux = 0 THEN
    dbms_output.put_line('Nu exista utilizatori inregistrati in aceasta tara.');
ELSE
    dbms_output.put_line('Un total de ' || v_aux || ' utilizatori inregistrati in aceasta
tara.');
END IF;
WHEN TOO_MANY_ROWS THEN
    dbms_output.put_line('> In tara ' || INITCAP(p_nume_tara) || ' exista mai multe
sedii.');
    dbms_output.put_line('Activitatea angajatilor care lucreaza fizic in aceasta tara: ');
    v_aux := 0;
    FOR v_detalii_angajat IN (
        SELECT nume, prenume, denumire_job,
               (SELECT COUNT(cod_persoana) FROM TRANZACTIE_EXTERNA t
WHERE t.cod_persoana = a.cod_persoana) nrTranzactiiExterne,
               (SELECT COUNT(cod_angajat) FROM SEMNEAZA_UN_CONTRACT sc
WHERE sc.cod_angajat = a.cod_persoana) nrContracte
        FROM ANGAJAT a, PERSOANA p, JOB j, SEDIU s, LOCATIE l
)
LOOP
    dbms_output.put('>> ' || v_detalii_angajat.nume || ' ' ||
v_detalii_angajat.prenume || ' ');
    IF v_detalii_angajat.denumire_job IS NOT NULL THEN
        dbms_output.put('este angajat (' || v_detalii_angajat.denumire_job || ')');
    ELSE
        dbms_output.put('este client (' || v_detalii_angajat.denumire_job || ')');
    END IF;
    dbms_output.put(' cu ' || v_detalii_angajat.nrTranzactiiExterne || ' tranzactii
externe si ' || v_detalii_angajat.nrContracte || ' contracte');
    dbms_output.new_line;
    v_aux := v_aux + 1;
END LOOP;

```

```

        WHERE p.cod_persoana = a.cod_persoana AND j.cod_job = a.cod_job
        AND a.cod_sediu = s.cod_sediu AND s.cod_locatie = l.cod_locatie
        AND l.cod_tara = v_cod_tara)
    LOOP
        dbms_output.put_line('> Angajatul ' || v_detalii_angajat.nume || ' ' ||
v_detalii_angajat.prenume || ' este ' || v_detalii_angajat.denumire_job
        || ', a efectuat ' || v_detalii_angajat.nrTranzactiiExterne || ' tranzactii externe si a
semnat ' || v_detalii_angajat.nrContracte || ' contracte.'
        );
        v_aux := 1;
    END LOOP;
    IF v_aux = 0 THEN
        dbms_output.put_line('Nu exista angajati in aceasta tara.');
    END IF;
END;

END cerinte_proiect;
/

EXECUTE cerinte_proiect.activitate_utilizatori;
--lipsesc melis2_ si mihaita
EXECUTE cerinte_proiect.activitate_utilizatori(100);

EXECUTE cerinte_proiect.info_client('UnUserInexistent_');
EXECUTE cerinte_proiect.info_client('melis2_');

--clientul 1 cere pin-ul pentru cardul sau, folosind cnp-ul corect si numarul de card corect
SELECT cerinte_proiect.cere_pin(1, '4532962746090018', '2870816526255') pin FROM
DUAL;
--clientul 1 cere pentru cardul sau, dar cnp-ul este gresit
SELECT cerinte_proiect.cere_pin(1, '4532962746090018', '2870816526256') pin FROM
DUAL;
--managerul (id 16) cere pentru cardul clientul 1, cu toate datele corecte
SELECT cerinte_proiect.cere_pin(16, '4532962746090018', '1950122298018') pin FROM
DUAL;

EXECUTE cerinte_proiect.statistici_sedii('TaraInexistenta');
EXECUTE cerinte_proiect.statistici_sedii('Japonia');
EXECUTE cerinte_proiect.statistici_sedii('Olanda');
EXECUTE cerinte_proiect.statistici_sedii('Romania');

```

```

1403 CREATE OR REPLACE PACKAGE cerinte_proiect
1404 AS
1405   --ex6
1406   PROCEDURE activitate_utilizatori(activ_max PLS_INTEGER DEFAULT -1);
1407   --ex7
1408   PROCEDURE info_client(tag_utilizator CLIENT.tag_utilizator%TYPE);
1409   --ex8
1410   FUNCTION cere_pin(cine_cere PERSOANA.cod_persoana%TYPE, p_card_number CARD.card_number%TYPE, p_cnp PERSOANA.cnp%TYPE) RETURN CARD.pin%TYPE;
1411   --ex9
1412   PROCEDURE statistici_sedii(p_numar_tara TARA.numar_tara%TYPE);
1413 END cerinte_proiect;
1414 /
1415
1416
1417 CREATE OR REPLACE PACKAGE BODY cerinte_proiect
1418 AS
1419   PROCEDURE activitate_utilizatori(activ_max PLS_INTEGER DEFAULT -1)... .
1527
1528   PROCEDURE info_client(tag_utilizator CLIENT.tag_utilizator%TYPE)... .
1620
1621   FUNCTION cere_pin(cine_cere PERSOANA.cod_persoana%TYPE, p_card_number CARD.card_number%TYPE, p_cnp PERSOANA.cnp%TYPE) RETURN CAR...| .
1716
1717   PROCEDURE statistici_sedii(p_numar_tara TARA.numar_tara%TYPE)... .
1803
1804 END cerinte_proiect;
1805 /
...

```

Script Output X | Task completed in 0.039 seconds

Package CERINTE_PROIECT compiled

Package Body CERINTE_PROIECT compiled

14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

--14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare
-- unui flux de acțiuni integrate, specifice bazei de date definite (minim 2
-- tipuri de date, minim 2 funcții, minim 2 proceduri)

CREATE OR REPLACE PACKAGE utilitati_bancare
AS

card_negasit EXCEPTION;
cont_negasit EXCEPTION;
persoana_negasita EXCEPTION;
insuficienti_bani EXCEPTION;
ambiguitate EXCEPTION;

PRAGMA EXCEPTION_INIT(card_negasit, -20101);
PRAGMA EXCEPTION_INIT(cont_negasit, -20102);
PRAGMA EXCEPTION_INIT(persoana_negasita, -20103);
PRAGMA EXCEPTION_INIT(insuficienti_bani, -20104);
PRAGMA EXCEPTION_INIT(ambiguitate, -20105);

TYPE info_locatie IS RECORD(
cod_postal LOCATIE.cod_postal%TYPE,
adresa LOCATIE.adresa%TYPE,
oras LOCATIE.oras%TYPE,
numar_tara TARA.numar_tara%TYPE

```

);

TYPE info_personal IS RECORD(
    nume PERSOANA.nume%TYPE,
    prenume PERSOANA.prenume%TYPE,
    email PERSOANA.email%TYPE,
    gen PERSOANA.gen%TYPE,
    locatie info_locatie
);

TYPE info_card IS RECORD(
    card_number CARD.card_number%TYPE,
    tip CARD.tip%TYPE
);

TYPE lista_carduri IS TABLE OF info_card;

TYPE info_cont IS RECORD(
    iban CONT.iban%TYPE,
    sold CONT.sold%TYPE,
    carduri lista_carduri
);

TYPE lista_conturi IS TABLE OF info_cont;

TYPE info_client IS RECORD(
    tag_utilizator CLIENT.tag_utilizator%TYPE,
    personal_data info_personal,
    conturi lista_conturi
);

TYPE info_contract IS RECORD(
    titlu_contract TIP_CONTRACT.titlu_contract%TYPE,
    client_data info_personal,
    angajat_data info_personal,
    locatie_semnat info_locatie
);

TYPE lista_contracte IS TABLE OF info_contract;

TYPE info_angajat IS RECORD(
    job_titlu JOB.denumire_job%TYPE,
    personal_data info_personal,
    salariu ANGAJAT.salariu%TYPE,
    contracte_semnate lista_contracte
);

PROCEDURE depozit_bani(
    p_ibан CONT.iban%TYPE,

```

```

    valoare TRANZACTIE.valoare%TYPE,
    p_cod_bancomat DEPOZIT.cod_bancomat%TYPE,
    p_card_number CARD.card_number%TYPE
);

PROCEDURE retragere_bani(
    p_ibanc CONT.iban%TYPE,
    valoare TRANZACTIE.valoare%TYPE,
    p_cod_bancomat DEPOZIT.cod_bancomat%TYPE,
    p_card_number CARD.card_number%TYPE
);

PROCEDURE transfer_bani(
    p_ibanc_sursa CONT.iban%TYPE,
    p_ibanc_destinatie CONT.iban%TYPE,
    valoare TRANZACTIE.valoare%TYPE
);

FUNCTION get_client_info(tag_utilizator CLIENT.tag_utilizator%TYPE) RETURN
info_client;

FUNCTION get_angajat_info(p_num PERSOANA.num%TYPE, p_prenume
PERSOANA.prenume%TYPE) RETURN info_angajat;
END utilitati_bancare;
/

```



```

CREATE OR REPLACE PACKAGE BODY utilitati_bancare
AS
    FUNCTION get_cod_persoana(p_num PERSOANA.num%TYPE, p_prenume
PERSOANA.prenume%TYPE) RETURN PERSOANA.cod_persoana%TYPE
    AS
        v_cod PERSOANA.cod_persoana%TYPE;
    BEGIN
        SELECT cod_persoana INTO v_cod FROM PERSOANA WHERE nume = p_num
        AND p_prenume = prenume;
        RETURN v_cod;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20103, 'Persoana cautata nu a fost gasita.');
        WHEN TOO_MANY_ROWS THEN
            RAISE_APPLICATION_ERROR(-20105, 'Ambiguitate, sunt mai multe persoane
cu numele si prenumele ' || p_num || ' ' || p_prenume);
    END;

    FUNCTION get_cod_client(p_tag_utilizator CLIENT.tag_utilizator%TYPE) RETURN
CLIENT.cod_persoana%TYPE
    AS

```

```

    v_cod CLIENT.cod_persoana%TYPE;
BEGIN
    SELECT cod_persoana INTO v_cod FROM CLIENT WHERE tag_utilizator =
p_tag_utilizator;
    RETURN v_cod;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20103, 'Clientul cautat nu a fost gasit');
END;

FUNCTION get_cod_card(p_card_number CARD.card_number%TYPE) RETURN
CARD.cod_card%TYPE
AS
    v_cod CARD.cod_card%TYPE;
BEGIN
    SELECT cod_card INTO v_cod FROM CARD WHERE card_number =
p_card_number;
    RETURN v_cod;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20101, 'Cardul cautat nu a fost gasit');
END;

FUNCTION get_cod_cont(p_ibан CONT.iban%TYPE) RETURN
CONT.cod_cont%TYPE
AS
    v_cod CONT.cod_cont%TYPE;
BEGIN
    SELECT cod_cont INTO v_cod FROM CONT WHERE iban = p_ibан;
    RETURN v_cod;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20102, 'Contul cautat nu a fost gasit');
END;

PROCEDURE depozit_bani(
    p_ibан CONT.iban%TYPE,
    valoare TRANZACTIE.valoare%TYPE,
    p_cod_bancomat DEPOZIT.cod_bancomat%TYPE,
    p_card_number CARD.card_number%TYPE
)
AS
    tranzactie_id TRANZACTIE.cod_tranzactie%TYPE;
    v_cod_cont CONT.cod_cont%TYPE;
    v_cod_card CARD.cod_card%TYPE;
BEGIN
    v_cod_cont := get_cod_cont(p_ibан);
    v_cod_card := get_cod_card(p_card_number);

```

```

        SELECT tranzactii_contor+1 INTO tranzactie_id FROM ISTORIC i WHERE
i.cod_cont = v_cod_cont;

        INSERT INTO TRANZACTIE VALUES(tranzactie_id, v_cod_cont, valoare,
SYSDATE, 'depozit');
        INSERT INTO DEPOZIT VALUES(tranzactie_id, v_cod_cont, p_cod_bancomat,
v_cod_card);
        UPDATE CONT c SET sold = sold + valoare WHERE c.cod_cont = v_cod_cont;
        dbms_output.put_line('Depozit realizat cu succes!');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20056, 'Contul exista dar nu are un istoric
asociat.');
        --nu poate da too_many_rows, verificam dupa cheia primara
END;

PROCEDURE retragere_bani(
    p_ibан CONT.iban%TYPE,
    valoare TRANZACTIE.valoare%TYPE,
    p_cod_bancomat DEPOZIT.cod_bancomat%TYPE,
    p_card_number CARD.card_number%TYPE
)
AS
    tranzactie_id TRANZACTIE.cod_tranzactie%TYPE;
    v_sold CONT.sold%TYPE;
    v_cod_cont CONT.cod_cont%TYPE;
    v_cod_card CARD.cod_cont%TYPE;
BEGIN
    v_cod_cont := get_cod_cont(p_ibан);
    v_cod_card := get_cod_card(p_card_number);
    SELECT tranzactii_contor+1 INTO tranzactie_id FROM ISTORIC i WHERE
i.cod_cont = v_cod_cont;
    --daca trece de select-ul anterior, sigur exista si contul fiind conectate prin foreign key
    SELECT sold INTO v_sold FROM CONT c WHERE c.cod_cont = v_cod_cont;
    IF v_sold < valoare THEN
        RAISE_APPLICATION_ERROR(-20104, 'Contul nu are suficienti bani pentru a
efectua retragerea: ' || v_sold || '<' || valoare);
    END IF;

    INSERT INTO TRANZACTIE VALUES(tranzactie_id, v_cod_cont, valoare,
SYSDATE, 'retragere');
    INSERT INTO RETRAGERE VALUES(tranzactie_id, v_cod_cont, p_cod_bancomat,
v_cod_card);
    UPDATE CONT c SET sold = sold - valoare WHERE c.cod_cont = v_cod_cont;
    dbms_output.put_line('Retragere realizata cu succes!');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20056, 'Contul exista dar nu are un istoric
asociat.');
        --nu poate da too_many_rows, verificam dupa cheia primara
END;

```

```

asociat.');
--nu poate da too_many_rows, verificam dupa cheia primara
END;

PROCEDURE transfer_bani(
    p_ibан_sursa CONT.iban%TYPE,
    p_ibан_destinatie CONT.iban%TYPE,
    valoare TRANZACTIE.valoare%TYPE
)
AS
    tranzactie_id TRANZACTIE.cod_tranzactie%TYPE;
    v_sold CONT.sold%TYPE;
    v_data DATE := SYSDATE;
    v_cod_cont_sursa CONT.cod_cont%TYPE;
    v_cod_cont_destinatie CONT.cod_cont%TYPE;
BEGIN
    IF p_ibан_sursa = p_ibан_destinatie THEN
        RAISE_APPLICATION_ERROR(-20057, 'Nu se pot trimite bani dintr-un cont in
acelasi cont.');
    END IF;

    BEGIN
        v_cod_cont_sursa := get_cod_cont(p_ibан_sursa);
    EXCEPTION
        WHEN cont_negasit THEN
            RAISE_APPLICATION_ERROR(-20102, 'Contul sursa nu a fost gasit.');
    END;

    BEGIN
        v_cod_cont_destinatie := get_cod_cont(p_ibан_destinatie);
    EXCEPTION
        WHEN cont_negasit THEN
            RAISE_APPLICATION_ERROR(-20102, 'Contul destinatie nu a fost gasit.');
    END;

    SAVEPOINT inceput_transfer;

    --inserare transfer trimis
    BEGIN
        SELECT tranzactii_contor+1 INTO tranzactie_id FROM ISTORIC i WHERE
        i.cod_cont = v_cod_cont_sursa;
        --daca trece de select-ul anterior, sigur exista si contul fiind conectate prin foreign
        key
        SELECT sold INTO v_sold FROM CONT c WHERE c.cod_cont =
        v_cod_cont_sursa;
        IF v_sold < valoare THEN
            RAISE_APPLICATION_ERROR(-20005, 'Contul nu are suficienti bani pentru a
            efectua transferul: ' || v_sold || '<' || valoare);
    END;

```

```

        END IF;

        INSERT INTO TRANZACTIE VALUES(tranzactie_id, v_cod_cont_sursa, valoare,
v_data, 'transfer Trimis');
        INSERT INTO TRANSFER_TRIMIS VALUES(tranzactie_id, v_cod_cont_sursa,
v_cod_cont_destinatie);
        UPDATE CONT c SET sold = sold - valoare WHERE c.cod_cont =
v_cod_cont_sursa;
        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                RAISE_APPLICATION_ERROR(-20005, 'Nu se poate efectua un transfer
pentru un cont cu istoric inexistent. (1)');
            END;

--inserare transfer primit
BEGIN
    SELECT tranzactii_contor+1 INTO tranzactie_id FROM ISTORIC i WHERE
i.cod_cont = v_cod_cont_destinatie;

    INSERT INTO TRANZACTIE VALUES(tranzactie_id, v_cod_cont_destinatie,
valoare, v_data, 'transfer_primit');
    INSERT INTO TRANSFER_PRIMIT VALUES(tranzactie_id,
v_cod_cont_destinatie, v_cod_cont_sursa);
    UPDATE CONT c SET sold = sold + valoare WHERE c.cod_cont =
v_cod_cont_destinatie;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20006, 'Nu se poate efectua un transfer
pentru un cont cu istoric inexistent. (2)');
        END;
        dbms_output.put_line('Transfer realizat cu succes!');
    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK TO inceput_transfer; --daca orice parte a transferului esueaza, nu
vrem sa ramana fragmente de transfer in baza de date
            RAISE;
    END;

FUNCTION get_client_info(tag_utilizator CLIENT.tag_utilizator%TYPE) RETURN
info_client
AS
    deReturnat info_client;
    v_cod_client CLIENT.cod_persoana%TYPE;
    v_i PLS_INTEGER;
BEGIN
    v_cod_client := get_cod_client(tag_utilizator);
    FOR v_record IN (
        SELECT cod_postal, adresa, oras, nume_tara, nume, prenume, tag_utilizator, email,
gen

```

```

        FROM TARA t, LOCATIE l, PERSOANA p, CLIENT c WHERE
        t.cod_tara = l.cod_tara AND l.cod_locatie = p.cod_locatie AND
        p.cod_persoana = c.cod_persoana AND c.cod_persoana = v_cod_client
    )
    LOOP
        deReturnat.tag_utilizator := v_record.tag_utilizator;
        deReturnat.personal_data.nume := v_record.nume;
        deReturnat.personal_data.prenume := v_record.prenume;
        deReturnat.personal_data.email := v_record.email;
        deReturnat.personal_data.gen := v_record.gen;
        deReturnat.personal_data.locatie.cod_postal := v_record.cod_postal;
        deReturnat.personal_data.locatie.adresa := v_record.adresa;
        deReturnat.personal_data.locatie.oras := v_record.oras;
        deReturnat.personal_data.locatie.nume_tara := v_record.nume_tara;
        deReturnat.conturi := lista_conturi();

        v_i := 0;
        FOR v_record_conturi IN (SELECT cod_cont, sold, iban FROM CONT WHERE
cod_persoana = v_cod_client) LOOP
            deReturnat.conturi.extend;
            v_i := v_i + 1;
            deReturnat.conturi(v_i).sold := v_record_conturi.sold;
            deReturnat.conturi(v_i).iban := v_record_conturi.iban;

            SELECT card_number, tip BULK COLLECT INTO
deReturnat.conturi(v_i).carduri FROM CARD WHERE cod_cont =
v_record_conturi.cod_cont;

        END LOOP;

        END LOOP;

        return deReturnat;
    END;

    FUNCTION get_angajat_info(p_nume PERSOANA.nume%TYPE, p_prenume
PERSOANA.prenume%TYPE) RETURN info_angajat
    AS
        deReturnat.info_angajat;
        v_cod PERSOANA.cod_persoana%TYPE;
        v_i PLS_INTEGER;
    BEGIN
        v_cod := get_cod_persoana(p_nume, p_prenume);
        deReturnat.personal_data.nume := p_nume;
        deReturnat.personal_data.prenume := p_prenume;
        FOR v_record IN (
            SELECT email, gen, cod_postal, adresa, oras, nume_tara, denumire_job, salariu
            FROM PERSOANA p, LOCATIE l, TARA t, ANGAJAT a, JOB j
            WHERE p.cod_persoana = a.cod_persoana AND l.cod_locatie = p.cod_locatie

```

```

        AND t.cod_tara = l.cod_tara AND a.cod_job = j.cod_job AND p.cod_persoana =
v_cod
    )
LOOP
    deReturnat.job_titlu := v_record.denumire_job;
    deReturnat.salariu := v_record.salariu;
    deReturnat.personal_data.email := v_record.email;
    deReturnat.personal_data.gen := v_record.gen;
    deReturnat.personal_data.locatie.cod_postal := v_record.cod_postal;
    deReturnat.personal_data.locatie.adresa := v_record.adresa;
    deReturnat.personal_data.locatie.oras := v_record.oras;
    deReturnat.personal_data.locatie.nume_tara := v_record.nume_tara;

    deReturnat.contracte_semnate := lista_contracte();

    v_i := 0;

    FOR v_record_contract_cu_sedi IN(
        SELECT nume, prenume, email, gen, l.cod_postal, l.adresa, l.oras, t.nume_tara,
titlu_contract,
        ls.cod_postal sediu_cod_postal, ls.adresa sediu_adresa, ls.oras sediu_oras,
ts.nume_tara sediu_nume_tara
        FROM PERSOANA p, CLIENT c, LOCATIE l,
SEMNEAZA_UN_CONTRACT sc, TIP_CONTRACT tc, SEDIU s, LOCATIE ls, TARA
t, TARA ts
        WHERE p.cod_persoana = c.cod_persoana AND l.cod_locatie = p.cod_locatie
AND sc.cod_client = p.cod_persoana
            AND t.cod_tara = l.cod_tara AND ls.cod_tara = ts.cod_tara
            AND tc.tip_contract = sc.tip_contract
            AND s.cod_sedi = sc.cod_sedi AND ls.cod_locatie = s.cod_locatie AND
sc.cod_angajat = v_cod
    )
    LOOP
        deReturnat.contracte_semnate.extend;
        v_i := v_i + 1;
        deReturnat.contracte_semnate(v_i).titlu_contract :=
v_record_contract_cu_sedi.titlu_contract;
        deReturnat.contracte_semnate(v_i).angajat_data := deReturnat.personal_data;
        deReturnat.contracte_semnate(v_i).client_data.nume :=
v_record_contract_cu_sedi.nume;
        deReturnat.contracte_semnate(v_i).client_data.prenume :=
v_record_contract_cu_sedi.prenume;
        deReturnat.contracte_semnate(v_i).client_data.email :=
v_record_contract_cu_sedi.email;
        deReturnat.contracte_semnate(v_i).client_data.gen :=
v_record_contract_cu_sedi.gen;
        deReturnat.contracte_semnate(v_i).client_data.locatie.cod_postal :=
v_record_contract_cu_sedi.cod_postal;
        deReturnat.contracte_semnate(v_i).client_data.locatie.adresa :=

```

```

v_record_contract_cu_sediu.adresa;
    deReturnat.contracte_semnate(v_i).client_data.locatie.oras :=
v_record_contract_cu_sediu.oras;
    deReturnat.contracte_semnate(v_i).client_data.locatie.nume_tara :=
v_record_contract_cu_sediu.nume_tara;

    deReturnat.contracte_semnate(v_i).locatie_semnat.cod_postal :=
v_record_contract_cu_sediu.sediu_cod_postal;
    deReturnat.contracte_semnate(v_i).locatie_semnat.adresa :=
v_record_contract_cu_sediu.sediu_adresa;
    deReturnat.contracte_semnate(v_i).locatie_semnat.oras :=
v_record_contract_cu_sediu.sediu_oras;
    deReturnat.contracte_semnate(v_i).locatie_semnat.nume_tara :=
v_record_contract_cu_sediu.sediu_nume_tara;
    END LOOP;

FOR v_record_contract IN(
    SELECT nume, prenume, email, gen, l.cod_postal, l.adresa, l.oras, t.nume_tara,
titlu_contract
        FROM PERSOANA p, CLIENT c, LOCATIE l,
SEMNEAZA_UN_CONTRACT sc, TIP_CONTRACT tc, TARA t
        WHERE p.cod_persoana = c.cod_persoana AND l.cod_locatie = p.cod_locatie
AND sc.cod_client = p.cod_persoana
        AND t.cod_tara = l.cod_tara
        AND tc.tip_contract = sc.tip_contract
        AND sc.cod_sediu IS NULL AND sc.cod_angajat = v_cod
)
LOOP
    deReturnat.contracte_semnate.extend;
    v_i := v_i + 1;
    deReturnat.contracte_semnate(v_i).titlu_contract :=
v_record_contract.titlu_contract;
    deReturnat.contracte_semnate(v_i).angajat_data := deReturnat.personal_data;
    deReturnat.contracte_semnate(v_i).client_data.nume := v_record_contract.nume;
    deReturnat.contracte_semnate(v_i).client_data.prenume :=
v_record_contract.prenume;
    deReturnat.contracte_semnate(v_i).client_data.email := v_record_contract.email;
    deReturnat.contracte_semnate(v_i).client_data.gen := v_record_contract.gen;
    deReturnat.contracte_semnate(v_i).client_data.locatie.cod_postal :=
v_record_contract.cod_postal;
    deReturnat.contracte_semnate(v_i).client_data.locatie.adresa :=
v_record_contract.adresa;
    deReturnat.contracte_semnate(v_i).client_data.locatie.oras :=
v_record_contract.oras;
    deReturnat.contracte_semnate(v_i).client_data.locatie.nume_tara :=
v_record_contract.nume_tara;
    END LOOP;
END LOOP;
return deReturnat;

```

```

        END;
END utilitati_bancare;
/

SELECT * FROM CONT;
EXECUTE utilitati_bancare.depozit_bani('RO43PORL9543945186522245', 122, 1,
'4539763856109249');
EXECUTE utilitati_bancare.retragere_bani('RO43PORL9543945186522245', 122, 1,
'4539763856109249');
EXECUTE utilitati_bancare.transfer_bani('RO80RZBR8845968383338826',
'RO93PORL7529645415765354', 1000);

DECLARE
    v_client utilitati_bancare.info_client;
BEGIN
    v_client := utilitati_bancare.get_client_info('UserInexistent');
EXCEPTION
    WHEN utilitati_bancare.persoana_negasita THEN
        dbms_output.put_line('A intrat aici.');
END;
/


DECLARE
    v_client utilitati_bancare.info_client;
BEGIN
    v_client := utilitati_bancare.get_client_info('melis2_');
    dbms_output.put_line('==== Client Info ====');
    dbms_output.put_line('> Nume: ' || v_client.personal_data.nume);
    dbms_output.put_line('> Prenume: ' || v_client.personal_data.prenume);
    dbms_output.put_line('> Email: ' || v_client.personal_data.email);
    dbms_output.put_line('> Gen: ' || UPPER(v_client.personal_data.gen));
    dbms_output.put_line('> Locatie: ');
    dbms_output.put_line('>> Cod Postal: ' || v_client.personal_data.locatie.cod_postal);
    dbms_output.put_line('>> Adresa: ' || v_client.personal_data.locatie.adresa);
    dbms_output.put_line('>> Oras: ' || v_client.personal_data.locatie.oras);
    dbms_output.put_line('>> Tara: ' || v_client.personal_data.locatie.nume_tara);

    IF v_client.conturi.count = 0 THEN
        dbms_output.put_line('> Clientul nu are conturi');
        RETURN;
    END IF;

    FOR i IN 1..v_client.conturi.count LOOP
        dbms_output.put_line('--- Cont Info ---');
        dbms_output.put_line('> IBAN: ' || v_client.conturi(i).iban);
        dbms_output.put_line('> Sold: ' || v_client.conturi(i).sold);

        IF v_client.conturi(i).carduri.count = 0 THEN

```

```

        dbms_output.put_line('> Contul nu are carduri asociate.');
        CONTINUE;
    END IF;

    dbms_output.put_line('> Carduri:');
    FOR j IN 1..v_client.conturi(i).carduri.count LOOP
        dbms_output.put_line(' - ' || v_client.conturi(i).carduri(j).card_number || '(' ||
v_client.conturi(i).carduri(j).tip || ')');
    END LOOP;
    END LOOP;
EXCEPTION
    WHEN utilitati_bancare.persoana_negasita THEN
        dbms_output.put_line('A intrat aici.');
END;
/

```

DECLARE

```

    v_angajat utilitati_bancare.info_angajat;
BEGIN
    v_angajat := utilitati_bancare.get_angajat_info('Adam', 'Ioan');

    dbms_output.put_line('==== Angajat Info ====');
    dbms_output.put_line('> Nume: ' || v_angajat.personal_data.nume);
    dbms_output.put_line('> Prenume: ' || v_angajat.personal_data.prenume);
    dbms_output.put_line('> Email: ' || v_angajat.personal_data.email);
    dbms_output.put_line('> Gen: ' || UPPER(v_angajat.personal_data.gen));
    dbms_output.put_line('> Locatie: ');
    dbms_output.put_line('>> Cod Postal: ' || v_angajat.personal_data.locatie.cod_postal);
    dbms_output.put_line('>> Adresa: ' || v_angajat.personal_data.locatie.adresa);
    dbms_output.put_line('>> Oras: ' || v_angajat.personal_data.locatie.oras);
    dbms_output.put_line('>> Tara: ' || v_angajat.personal_data.locatie.num_tara);
    dbms_output.put_line('> Job: ' || v_angajat.job_titlu);
    dbms_output.put_line('> Salariu: ' || v_angajat.salariu);

    IF v_angajat.contracte_semnate.count = 0 THEN
        dbms_output.put_line('> Angajatul nu a semnat niciun contract.');
        RETURN;
    END IF;

    FOR i IN 1..v_angajat.contracte_semnate.count LOOP
        dbms_output.put_line('--- Contract ' || v_angajat.contracte_semnate(i).titlu_contract || ' ---');
        dbms_output.put_line('> Client: ' || v_angajat.contracte_semnate(i).client_data.nume ||
' ' || v_angajat.contracte_semnate(i).client_data.prenume);
        dbms_output.put('> A fost semnat ');
        IF v_angajat.contracte_semnate(i).locatie_semnat.adresa IS NULL THEN
            dbms_output.put('REMOTE');
        ELSE

```

```

        dbms_output.put('la ' || v_angajat.contracte_semnate(i).locatie_semnat.adresa || '' ||
v_angajat.contracte_semnate(i).locatie_semnat.oras || '' ||
        v_angajat.contracte_semnate(i).locatie_semnat.nume_tara);
    END IF;
    dbms_output.new_line;
END LOOP;
END;
/

```

```

94 END utilitati_bancare;
95 /
96
97
98
99 CREATE OR REPLACE PACKAGE BODY utilitati_bancare
100 AS
101     FUNCTION get_cod_persoana(p_nume PERSOANA.nume%TYPE, p_prenume PERSOANA.prenume%TYPE) RETURN PERSOANA.cod_persoana%TYPE...
113
114     FUNCTION get_cod_client(p_tag_utilizator CLIENT.tag_utilizator%TYPE) RETURN CLIENT.cod_persoana%TYPE...
124
125     FUNCTION get_cod_card(p_card_number CARD.card_number%TYPE) RETURN CARD.cod_card%TYPE...
135
136     FUNCTION get_cod_cont(p_iban CONT.iban%TYPE) RETURN CONT.cod_cont%TYPE...
146
147
148     PROCEDURE depozit_bani...
172
173
174     PROCEDURE retragere_bani...
204
205
206     PROCEDURE transfer_bani...
272
273     FUNCTION get_client_info(tag_utilizator CLIENT.tag_utilizator%TYPE) RETURN info_client...
313
314     FUNCTION get_angajat_info(p_nume PERSOANA.nume%TYPE, p_prenume PERSOANA.prenume%TYPE) RETURN info_angajat...
397 END utilitati_bancare;
398 /

```

Script Output | Task completed in 0.042 seconds

Package UTILITATI_BANCARE compiled

Package Body UTILITATI_BANCARE compiled

Initial:

COD_CONT	COD_PERSOANA	SOLD	IBAN
1	1	1	3200.2 R080RZBR8845968383338826
2	2	2	1000.2 R093PORL7529645415765354
3	3	3	20.32 R043PORL9543945186522245

```

401 | EXECUTE utilitati_bancare.depozit_bani('RO43PORL9543945186522245', 122, 1, '4539763856109249');
402 | EXECUTE utilitati_bancare.retragere_bani('RO43PORL9543945186522245', 122, 1, '4539763856109249');
403 | EXECUTE utilitati_bancare.transfer_bani('RO80RZBR8845968383338826', 'RO93PORL7529645415765354', 3000);

```

Script Output x Query Result x
✖ ✚ ✖ ✖ Task completed in 0.028 seconds

PL/SQL procedure successfully completed.

Dbms Output

+ - ✚ ✖ ✖ ✖ Buffer Size: 20000

localhost x

Depozit realizat cu succes!

COD_CONT	COD_PERSOANA	SOLD	IBAN
1	1	1	3200.2 RO80RZBR8845968383338826
2	2	2	1000.2 RO93PORL7529645415765354
3	3	3	142.32 RO43PORL9543945186522245

```

401 | EXECUTE utilitati_bancare.depozit_bani('RO43PORL9543945186522245', 122, 1, '4539763856109249');
402 | EXECUTE utilitati_bancare.retragere_bani('RO43PORL9543945186522245', 1122, 1, '4539763856109249');
403 | EXECUTE utilitati_bancare.retragere_bani('RO43PORL9543945186522245', 122, 1, '4539763856109249');

```

Script Output x Query Result x
✖ ✚ ✖ ✖ Task completed in 0.035 seconds

ORA-20104: Contul nu are suficiente bani pentru a efectua retragerea: 142.32 < 1122
ORA-06512: at "DRAGOS.UTILITATI_BANCARE", line 95
ORA-06512: at line 1

```

403 | EXECUTE utilitati_bancare.retragere_bani('RO43PORL9543945186522245', 122, 1, '4539763856109249');
404 | EXECUTE utilitati_bancare.transfer_bani('RO80RZBR8845968383338826', 'RO93PORL7529645415765354', 3000);
405 |
406 |DECLARE
407 |

```

Script Output x Query Result x
✖ ✚ ✖ ✖ Task completed in 0.041 seconds

PL/SQL procedure successfully completed.

Dbms Output

+ - ✚ ✖ ✖ ✖ Buffer Size: 20000

localhost x

Retragere realizata cu succes!

```
404 | EXECUTE utilitati_bancare.transfer_bani('RO80RZBR884596838338826', 'RO93PORL7529645415765354', 1000);
405 |
406 |DECLARE
```

Script Output x Query Result x
Task completed in 0.039 seconds

PL/SQL procedure successfully completed.

Dbsms Output

+ | Buffer Size: 20000 |

localhost x

Transfer realizat cu succes!

	COD_CONT	COD_PERSOANA	SOLD	IBAN
1	1	1	1	2200.2 RO80RZBR884596838338826
2	2	2	2	2000.2 RO93PORL7529645415765354
3	3	3	3	20.32 RO43PORL9543945186522245

```
406 |DECLARE
407 |    v_client utilitati_bancare.info_client;
408 |BEGIN
409 |    v_client := utilitati_bancare.get_client_info('UserInexistent');
410 |EXCEPTION
411 |    WHEN utilitati_bancare.persoana_negasita THEN
412 |        dbms_output.put_line('A intrat aici.');
413 |END;
414 |/
415 |
416 |
417 |DECLARE
418 |    v_client utilitati_bancare.info_client;
419 |END;
```

Script Output x Query Result x
Task completed in 0.039 seconds

PL/SQL procedure successfully completed.

Dbsms Output

+ | Buffer Size: 20000 |

localhost x

A intrat aici.

```

417  DECLARE
418      v_client utilitati_bancare.info_client;
419  BEGIN
420      v_client := utilitati_bancare.get_client_info('melis2_');
421      dbms_output.put_line('==== Client Info ====');
422      dbms_output.put_line('> Nume: ' || v_client.personal_data.nume);
423      dbms_output.put_line('> Prenume: ' || v_client.personal_data.prenume);
424      dbms_output.put_line('> Email: ' || v_client.personal_data.email);
425      dbms_output.put_line('> Gen: ' || UPPER(v_client.personal_data.gen));
426      dbms_output.put_line('> Locatie: ');
427      "

```

Script Output x Query Result x

Task completed in 0.027 seconds

PL/SQL procedure successfully completed.

Dbm Output

Buffer Size: 20000 |

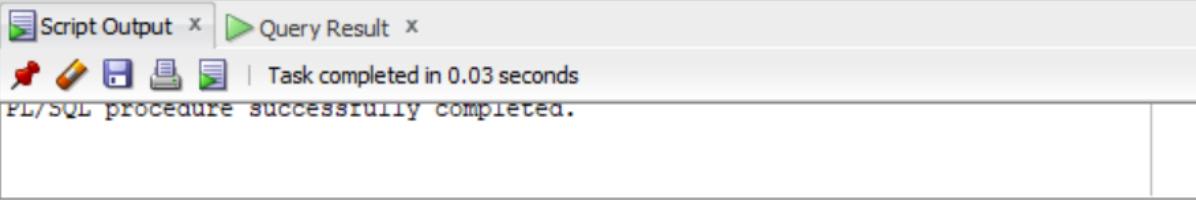
localhost x

```

===== Client Info =====
> Nume: Pintenaru
> Prenume: Melissa
> Email: melissa.p@gmail.com
> Gen: F
> Locatie:
>> Cod Postal: 1122
>> Adresa: Str Ernesto Nr 8 Bl A2 Sc 1 Et 3 Ap 10
>> Oras: Madrid
>> Tara: Spania
--- Cont Info ---
> IBAN: RO80RZBR8845968383338826
> Sold: 2200.2
> Carduri:
- 4532962746090018 (visa)
- 4801769871971639 (visa)
- 4929249181139042 (visa)
--- Cont Info ---
> IBAN: RO48PORL5146774785878989
> Sold: 100200.2
> Carduri:
- 5196701739892160 (mastercard)

```

```
457
458
459 DECLARE
460     v_angajat utilitati_bancare.info_angajat;
461 BEGIN
462     v_angajat := utilitati_bancare.get_angajat_info('Adam', 'Ioan');
463
464     dbms_output.put_line('==== Angajat Info ====');
465     dbms_output.put_line('> Nume: ' || v_angajat.personal_data.nume);
466     dbms_output.put_line('> Prenume: ' || v_angajat.personal_data.prenume);
467     dbms_output.put_line('> Email: ' || v_angajat.personal_data.email);
```



The screenshot shows the Oracle SQL Developer interface with the DBMS Output window open. The title bar says "localhost x". The window displays the output of the PL/SQL procedure, which includes personal information for an employee named Adam Ioan, along with various address and contract details. The output is formatted with indentation and line breaks to represent the original code's structure.

```
==== Angajat Info ====
> Nume: Adam
> Prenume: Ioan
> Email: adam.ioan6@gmail.com
> Gen: M
> Locatie:
>> Cod Postal: 28765
>> Adresa: Str Armeni Nr 36
>> Oras: Drobeta-Turnu-Severin
>> Tara: Romania
> Job: Manager
> Salariu: 100000
--- Contract Politica de Confidentialitate ---
> Client: Dacian Mihai
> A fost semnat la Str Goodfellow Nr 5 Haga Olanda
--- Contract Termini si Conditii ---
> Client: Dacian Mihai
> A fost semnat la Str Goodfellow Nr 5 Haga Olanda
--- Contract Politica de Confidentialitate ---
> Client: Lupu Emil
> A fost semnat la Str Ehia Nr 29 Vidin Bulgaria
--- Contract Termini si Conditii ---
> Client: Lupu Emil
> A fost semnat la Str Ehia Nr 29 Vidin Bulgaria
--- Contract Politica de Confidentialitate ---
```