**Baza de date a unui lant de magazine de tip retail**

**-proiect SGBD-**

**Ștefănescu Mircea Valentin**

**Grupa 1151 ID**

**An III**

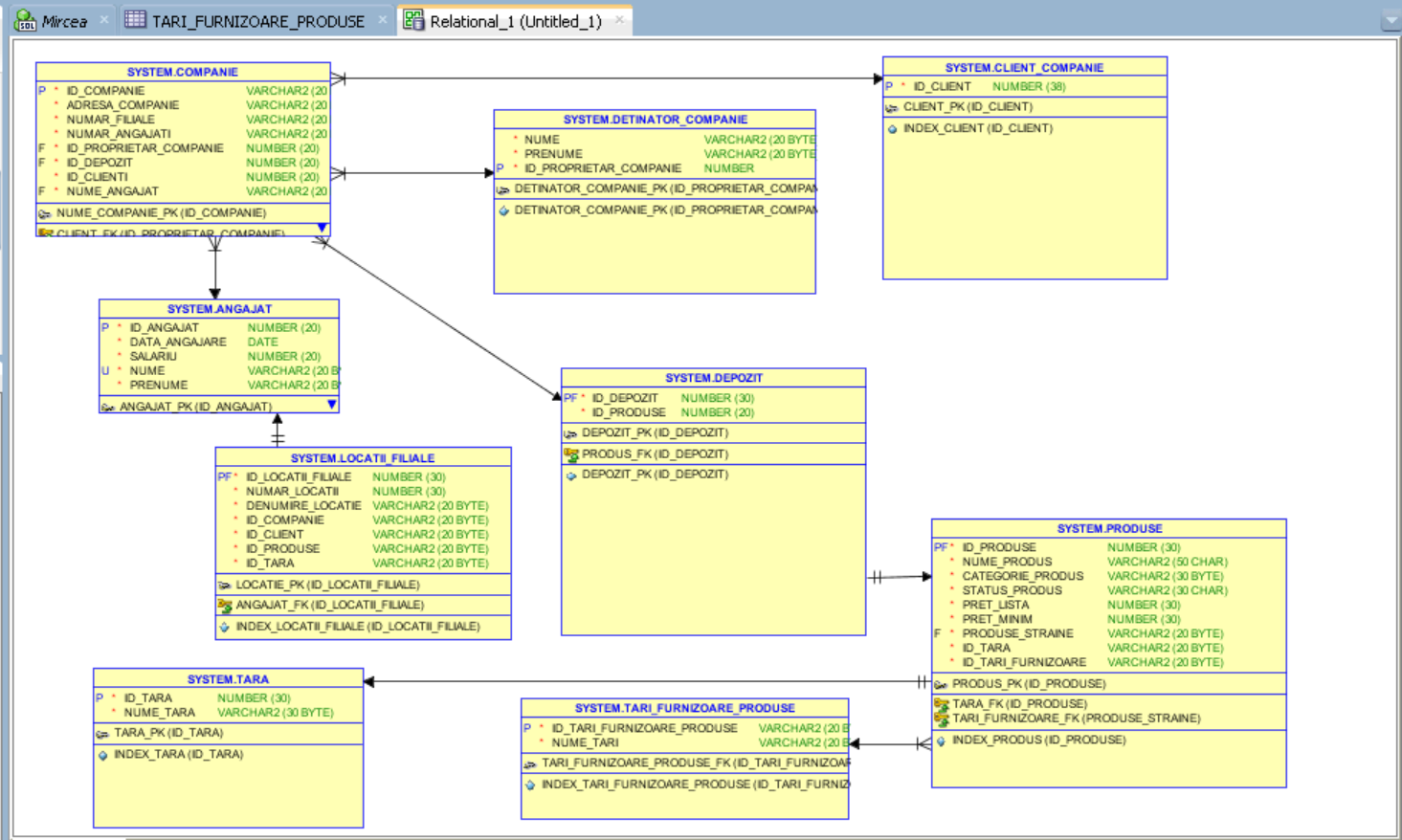
**Cuprins**

1. **Descrierea bazei de date**
2. **Schema conceptuala a bazei de date semestrul I**
3. **Interacțiunea cu serverul Oracle prin intermediul comenzilor SQL (LDD şi LMD): folosirea lui execute immediate, particularități cu privire la folosirea comenzii select precum și a funților la nivel de rând respectiv de grup.**
4. **Structuri alternative și repetitive.**
5. **Tratarea excepțiilor. (minim 3 implicite, 2 explicite)**
6. **Gestionarea cursorilor: impliciți și expliciți (cu și fără parametrii)**
7. **Funcții, proceduri, includerea acestora în pachete. (minim 3 funcții, 2 proceduri și un pachet)**
8. **Declanșatori la nivel de instrucțiune și de rând (minim 2 din fiecare)**

***Descrierea bazei de date***

Se realizeaza o aplicatie cu baze de date pentru evidenta desfasurarii activitatii la nivelul unui lant de magazine de tip retail dintr-o anumita tara.

In baza de date sunt inregistrate date referitoare la detinatorul companiei, numele si adresa companiei care detine lantul de magazine din tara, angajati si client ai filialelor, produsele prezente in fiecare filiala, tara si locatia depozitului in care isi desfasoara compania afacerea cat si tarile de unde isi procura unele produse.

***Schema conceptuala a bazei de date***

**Schema neschimbata din semestrul I.**

**Interacțiunea cu serverul Oracle prin intermediul comenzilor SQL (LDD şi LMD): folosirea lui execute immediate, particularități cu privire la folosirea comenzii select precum și a funților la nivel de rând respectiv de grup.**

**Comanda SELECT:**

Sa se afiseze toate inregistrarile din tabela PRODUSE:

*SELECT \* FROM PRODUSE*

**Comanda UPDATE:**

Se mărește cu 10 procente pretul produselor minime din tabela PRODUSE care au în prezent pretul mai mic decât o anumită valoare:

*DECLARE*

*PROCENTE NUMBER := 0.1;*

*PRAG PRODUSE.PRET\_MINIM%TYPE := 10;*

*BEGIN*

*UPDATE PRODUSE*

*SET*

*PRET\_MINIM = PRET\_MINIM \* (1 + PROCENTE)*

*WHERE PRET\_MINIM < PRAG;*

*END;*

*/*

*SELECT \**

*FROM PRODUSE;*

**Comanda DELETE**

Se şterge produsul cu denumirea “CIOCOLATA” din tabela PRODUSE:

*DECLARE BEGIN*

*DELETE FROM PRODUSE*

*WHERE NUME\_PRODUS LIKE '%CIOCOLATA%';*

*ROLLBACK;*

*END;*

*/*

*SELECT \**

*FROM PRODUSE;*

**Comanda ALTER:**

Se adaugă o nouă coloană STOC în tabela PRODUSE:

*DECLARE*

*NUME VARCHAR2 (200);*

*BEGIN*

*NUME := 'ALTER TABLE PRODUSE ADD (STOC NUMBER (10))';*

*DBMS\_OUTPUT.PUT\_LINE (NUME);*

*EXECUTE IMMEDIATE NUME;*

*END;*

*/*

*SELECT \**

*FROM PRODUSE;*

**Structuri alternative și repetitive.**

**Instrucțiune condițională CASE:**

Sa se afiseze din tabela “ANGAJAT” un mesaj corespunzator cu valoarea salariului angajatului. Daca un angajat are un salariu mai mic de 2000 sa se afiseze mesajul “Low”; daca un angajat are salariul intre 2000 si 3000 sa se afiseze mesajul “Mediu” iar daca salariul este mai mare de 3000 sa se afiseze mesajul “High”.

*SET SERVEROUTPUT ON SIZE 1000000;*

*DECLARE*

*SAL ANGAJAT.SALARIU%TYPE;*

*ID\_ANG ANGAJAT.ID\_ANGAJAT%TYPE := 5;*

*MESAJ VARCHAR (20);*

*BEGIN*

*SELECT SALARIU*

*INTO SAL*

*FROM ANGAJAT*

*WHERE ID\_ANGAJAT = ID\_ANG;*

*CASE*

*WHEN SAL < 2000 THEN*

*MESAJ := 'Low';*

*WHEN*

*SAL >= 2000 AND SAL <= 3000*

*THEN*

*MESAJ := 'Fair';*

*WHEN SAL >= 3000 THEN*

*MESAJ := 'High';*

*END CASE;*

*DBMS\_OUTPUT.PUT\_LINE (MESAJ);*

*END;*

*/*

**Structura WHILE…..LOOP….END LOOP:**

Se afişează din tabela “ANGAJAT” în ordine angajaţii cu id-urile în intervalul 25-50 atât timp cât salariul acestora este mai mic decât media:

*DECLARE*

*SAL ANGAJAT.SALARIU%TYPE;*

*MEDIU SAL%TYPE;*

*I NUMBER (4) := 25;*

*BEGIN*

*SELECT AVG (SALARIU)*

*INTO MEDIU*

*FROM ANGAJAT;*

*DBMS\_OUTPUT.PUT\_LINE ('Salariul mediu este: ' || MEDIU);*

*WHILE I <= 50 LOOP*

*SELECT SALARIU*

*INTO SAL*

*FROM ANGAJAT*

*WHERE ID\_ANGAJAT = I;*

*DBMS\_OUTPUT.PUT\_LINE ('Salariatul cu codul ' || I || ' are salariul: ' || SAL);*

*I := I + 1;*

*EXIT WHEN SAL < MEDIU;*

*END LOOP;*

*END;*

*/*

**Tratarea excepțiilor. (minim 3 implicite, 2 explicite)**

**NO\_DATA\_FOUND (Implicita):**

Sa se afiseze angajatul cu id-ul 10050 din tabela ANGAJAT.

*SET SERVEROUTPUT ON*

*DECLARE*

*NUME\_ANGAJAT VARCHAR2 (20);*

*BEGIN*

*SELECT NUME*

*INTO NUME\_ANGAJAT*

*FROM ANGAJAT*

*WHERE ID\_ANGAJAT = 10050;*

*DBMS\_OUTPUT.PUT\_LINE (NUME\_ANGAJAT);*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*DBMS\_OUTPUT.PUT\_LINE ('Nu exista angajatul cu acest ID!');*

*END;*

*/*

**TOO\_MANY\_ROWS(Implicita):**

Să se afişeze salariul angajatului cu prenumele “Dorel” din tabela ANGAJAT. Să se trateze eroare apărută în cazul în care există mai mulţi angajaţi cu acelaşi nume (interogarea SQL din bloc întoarce mai multe înregistrări).

*SET SERVEROUTPUT ON*

*DECLARE*

*SAL ANGAJAT.SALARIU%TYPE;*

*BEGIN*

*SELECT SALARIU*

*INTO SAL*

*FROM ANGAJAT*

*WHERE PRENUME = 'Dorel';*

*DBMS\_OUTPUT.PUT\_LINE ('Dorel are salariul de: ' || SAL);*

*EXCEPTION*

*WHEN TOO\_MANY\_ROWS THEN*

*DBMS\_OUTPUT.PUT\_LINE ('Exista mai multi salariati cu numele Dorel! Utilizati un cursor pentru selectie!');*

*END;*

*/*

**CURSOR\_ALREADY\_OPEN(Implicita):**

Se deschide un cursor folosind OPEN și mai apoi se încearcă parcurgerea sa folosind FOR. Instrucțiunea FOR încearcă să deschidă din nou cursorul rezultând excepția CURSOR\_ALREADY\_OPEN:

*DECLARE*

*CURSOR C IS*

*SELECT NUME, PRENUME*

*FROM ANGAJAT*

*ORDER BY SALARIU DESC;*

*BEGIN*

*OPEN C;*

*FOR R IN C LOOP*

*EXIT WHEN C%ROWCOUNT > 15;*

*DBMS\_OUTPUT.PUT\_LINE (R.NUME || ' ' || R.PRENUME);*

*END LOOP;*

*EXCEPTION*

*WHEN CURSOR\_ALREADY\_OPEN THEN*

*DBMS\_OUTPUT.PUT\_LINE ('Cursorul este deja deschis');*

*END;*

*/*

**Tratarea excepțiilor explicite**

* **RAISE:**

Să se modifice denumirea produsului cu id-ul 5 din tabela “PRODUSE”. Dacă nu se produce nici o actualizare atunci să se declanşeze o excepţie prin care să fie avertizat utilizatorul:

*DECLARE*

*PRODUS\_INVALID EXCEPTION;*

*BEGIN*

*UPDATE PRODUSE*

*SET*

*NUME\_PRODUS = 'Morcov'*

*WHERE ID\_PRODUS = 5;*

*IF SQL%NOTFOUND THEN*

*RAISE NO\_DATA\_FOUND;*

*END IF;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*DBMS\_OUTPUT.PUT\_LINE (SQLERRM);*

*END;*

*/*

* **RAISE\_APPLICATION\_ERROR:**

Aceeasi cerinta ca mai sus

*SET SERVEROUTPUT ON*

*DECLARE*

*COD NUMBER (7);*

*MESAJ VARCHAR2 (255);*

*PRODUS\_INVALID EXCEPTION;*

*PRAGMA EXCEPTION\_INIT (INVALID\_PROD, -456);*

*BEGIN*

*UPDATE PRODUSE*

*SET*

*NUME\_PRODUS = 'Morcov'*

*WHERE ID\_PRODUS = 5;*

*IF SQL%NOTFOUND THEN*

*RAISE\_APPLICATION\_ERROR (-456, 'Cod produs invalid!');*

*END IF;*

*EXCEPTION*

*WHEN PRODUS\_INVALID THEN*

*DBMS\_OUTPUT.PUT\_LINE ('Nu exista produsul cu acest ID');*

*COD := SQLCODE;*

*MESAJ := SQLERRM;*

*INSERT INTO ERORI VALUES (*

*USER, SYSDATE, COD, MESAJ*

*);*

*END;*

*/*

*SELECT \**

*FROM ERORI;*

**Gestionarea cursorilor: impliciți și expliciți (cu și fără parametrii)**

* **Cursorul Implicit:**

Se șterg produsele din categoria *“CATEGORIE\_PRODUS”* care nu au fost comandate. Se afișează numărul de rânduri şterse.

*SET SERVEROUTPUT ON*

*BEGIN*

*DELETE FROM PRODUSE P*

*WHERE CATEGORIE = 'hardware3' AND NOT EXISTS (*

*SELECT 1*

*FROM CATEGORIE\_PRODUS R*

*WHERE P.ID\_PRODUSE = R.ID\_PRODUSE*

*);*

*DBMS\_OUTPUT.PUT\_LINE (SQL%ROWCOUNT || ' randuri sterse');*

*ROLLBACK;*

*DBMS\_OUTPUT.PUT\_LINE (SQL%ROWCOUNT || ' randuri afectate');*

*END;*

*/*

După ROLLBACK, atributul SQL%ROWCOUNT devine 0. Rezultatul rulării este:

*4 randuri sterse*

*0 randuri afectate*

* **Cursorul Explicit**

Să se afişeze primele 2 produse al carui status de produs este “PREZENT”. În acest caz înregistrările vor fi ordonate descrescător în funcţie de numărul produselor prezente.

*SET SERVEROUTPUT ON*

*DECLARE*

*CURSOR C\_PREZENT IS*

*SELECT P.STATUS\_PRODUS, COUNT (R.ID\_PRODUS) NUMAR*

*FROM PRODUSE P, RAND\_PRODUSE R*

*WHERE P.STATUS\_PRODUS = R.STATUS\_PRODUS*

*GROUP BY P.STATUS\_PRODUS*

*ORDER BY COUNT (R.ID\_PRODUS) DESC;*

*REC\_PROD C\_PROD%ROWTYPE;*

*BEGIN*

*DBMS\_OUTPUT.PUT\_LINE ('Numarul de produse pentru fiecare comanda:');*

*IF NOT C\_PREZENT%ISOPEN THEN*

*OPEN C\_PREZENT;*

*END IF;*

*LOOP*

*FETCH C\_PREZENT INTO REC\_PROD;*

*EXIT WHEN C\_PREZENT%NOTFOUND OR C\_PREZENT%ROWCOUNT > 2;*

*DBMS\_OUTPUT.PUT\_LINE ('Comanda ' || REC\_PROD.STATUS\_PRODUS || ' are: '*

*|| REC\_PROD.NUMAR || ' produse');*

*END LOOP;*

*CLOSE C\_PREZENT;*

*END;*

*/*

**Funcții, proceduri, includerea acestora în pachete. (minim 3 funcții, 2 proceduri și un pachet)**

**Functia 1:**

Sa se creeze o functie ce returneaza numarul total de clienti din tabela “CLIENT\_COMPANIE”

*CREATE OR REPLACE FUNCTION CLIENTI\_TOTALI*

*RETURN number IS*

*total number(2) := 0;*

*BEGIN*

*SELECT count(\*) into total*

*FROM CLIENT\_COMPANIE;*

*RETURN total;*

*END;*

*/*

Apelare functie:

*DECLARE*

*c number(2);*

*BEGIN*

*c :=CLIENTI\_TOTALI ();*

*dbms\_output.put\_line('Numarul total de clienti: ' || c);*

*END;*

*/*

**Functia 2:**

Sa se creeze o functie care returneaza “TRUE/FALSE” daca are mai mare/mic sau egal cu salariul mediu si NULL daca salariul nu exista

*CREATE OR REPLACE FUNCTION VERIFICARE\_SALARIU (*

*P\_ID\_ANGAJAT IN ANGAJAT.ID\_ANGAJAT%TYPE, P\_SALARIU\_MEDIU IN NUMBER*

*) RETURN BOOLEAN IS*

*VALOARE\_SALARIU ANGAJAT.SALARIU%TYPE;*

*BEGIN*

*SELECT SALARIU*

*INTO VALOARE\_SALARIU*

*FROM ANGAJAT*

*WHERE ID\_ANGAJAT = P\_ID\_ANGAJAT;*

*IF VALOARE\_SALARIU > P\_SALARIU\_MEDIU THEN*

*RETURN TRUE;*

*ELSE*

*RETURN FALSE;*

*END IF;*

*EXCEPTION*

*WHEN NO\_DATA\_FOUND THEN*

*RETURN NULL;*

*END;*

*/*

*SHOW ERRORS*

**Functia 3:**

Sa se creeze o functie ce calculeaza si returneaza valoarea maxima dintre 2 numere

*DECLARE*

*A NUMBER;*

*B NUMBER;*

*C NUMBER;*

*FUNCTION FINDMAX (*

*X IN NUMBER, Y IN NUMBER*

*) RETURN NUMBER IS*

*Z NUMBER;*

*BEGIN*

*IF X > Y THEN*

*Z := X;*

*ELSE*

*Z := Y;*

*END IF;*

*RETURN Z;*

*END;*

*BEGIN*

*A := 15;*

*B := 34;*

*C := FINDMAX (A, B);*

*DBMS\_OUTPUT.PUT\_LINE (' Maximum of (23,45): ' || C);*

*END;*

*/*

**Procedura 1:**

Sa se creeze o procedura ce numele si prenumele din tabela ANGAJAT.

*CREATE OR REPLACE PROCEDURE INFORMATII (*

*IN\_ID\_ANGAJAT NUMBER*

*) IS*

*R\_INFO ANGAJAT%ROWTYPE;*

*BEGIN*

*SELECT \**

*INTO R\_INFO*

*FROM ANGAJAT*

*WHERE ID\_ANGAJAT = P\_ID\_ANGAJAT;*

*DBMS\_OUTPUT.PUT\_LINE (R\_INFO.NUME || ' ' || R\_PRENUME);*

*EXCEPTION*

*WHEN OTHERS THEN*

*DBMS\_OUTPUT.PUT\_LINE (SQLERRM);*

*END;*

Executare procedura:

*EXECUTE INFORMATII (15);*

**Procedura 2:**

Sa se creeze o procedura ce afiseaza un mesaj

*CREATE OR REPLACE PROCEDURE MES AS*

*BEGIN*

*DBMS\_OUTPUT.PUT\_LINE ('Hello World!');*

*END;*

*/*

**Pachet:**

Sa se creeze un pachet ce afiseaza salariul unui angajat pe baza id-ului din tabela ANGAJAT.

*CREATE OR REPLACE PACKAGE BODY ANGAJAT\_SALARIU AS*

*PROCEDURE AFISEAZA\_SALARIU (*

*A\_SAL ANGAJAT.ID\_ANGAJAT%TYPE*

*) IS*

*A\_SAL CUSTOMERS.SALARY%TYPE;*

*BEGIN*

*SELECT SALARY*

*INTO A\_SAL*

*FROM SALARIU*

*WHERE ID\_ANGAJAT = A\_SAL;*

*DBMS\_OUTPUT.PUT\_LINE ('Salary: ' || A\_SAL);*

*END AFISEAZA\_SALARIU;*

*END ANGAJAT\_SALARIU;*

*/*

*Package body created.*

**Folosire pachet:**

*DECLARE*

*CODE ANGAJAT.ID\_ANGAJAT%TYPE := &A\_ID;*

*BEGIN*

*ANGAJAT\_SALARIU.AFISEAZA\_SALARIU (CODE);*

*END;*

*/*

*Enter value for A\_ID: 5*

*Salary: 2546*

*PL/SQL procedure successfully completed.*

**Declanșatori la nivel de instrucțiune și de rând (minim 2 din fiecare)**

**La nivel de instrucțiune:**

Sa se defineasca un trigger care restrictioneaza userul de la a modifica salariul pentru angajatii care au fost angajati intre ziua 10 si 20 a fiecarei luni pentru tabela “ANGAJAT”

*CREATE OR REPLACE TRIGGER ANGAJAT\_TRIGGER BEFORE*

*UPDATE OF SALARIU ON ANGAJAT*

*DECLARE*

*L\_DAY\_OF\_MONTH NUMBER;*

*BEGIN*

*L\_DAY\_OF\_MONTH := EXTRACT (DAY FROM SYSDATE);*

*IF L\_DAY\_OF\_MONTH BETWEEN 10 AND 20 THEN*

*RAISE\_APPLICATION\_ERROR (-20100, 'NU SE POATE MODIFICA SALARIUL PENTRU ANGAJATII CARE AU FOST ANGAJATI INTRE ZIUA 10 SI 20');*

*END IF;*

*END;*

Sa se defineasca un trigger care restrictioneaza userul de la a modifica numele produsului pentru produsele care au preturile de lista intre 15 si 20 pentru tabela “PRODUS”

*CREATE OR REPLACE TRIGGER PRODUSE\_TRIGGER BEFORE*

*UPDATE OF NUME\_PRODUS ON PRODUSE*

*DECLARE*

*PRET\_LISTA NUMBER;*

*BEGIN*

*IF PRET\_LISTA BETWEEN 15 AND 25 THEN*

*RAISE\_APPLICATION\_ERROR (-20100, 'NU SE POT MODIFICA NUMELE PRODUSELOR CARE AU PRETUL INTRE 15 SI 25');*

*END IF;*

*END;*

**La nivel de rand:**

Sa se defineasca un trigger care restrictioneaza userul de la a modifica salariul daca salariul este mai mare decat dublu pentru tabela “ANGAJAT”

*CREATE OR REPLACE TRIGGER TRIGGER\_MODIFICARE\_SALARIU BEFORE*

*UPDATE OF SALARIU ON CUSTOMERS*

*FOR EACH ROW*

*WHEN (NEW.SALARIU > 0)*

*BEGIN*

*IF :NEW.SALARIU >= 2 \* :OLD.SALARIU THEN*

*RAISE\_APPLICATION\_ERROR*

*(-20101, 'Noul salariu ' || :NEW.SALARIU || ' nu mai poate fii modificat mai mult de dublu ' || :*

*OLD.CREDIT\_LIMIT);*

*END IF;*

*END;*

Sa se defineasca un trigger care restrictioneaza userul de la a modifica numele produsului pentru produsele care au preturile de lista mai mari de 25 pentru tabela “PRODUS”

*CREATE OR REPLACE TRIGGER TRIGGER\_MODIFICARE\_PRET\_PRODUS BEFORE*

*UPDATE OF PRET\_LISTA ON PRODUS*

*FOR EACH ROW*

*WHEN (NEW.PRET\_LISTA > 0)*

*BEGIN*

*IF :NEW.PRET\_LISTA >= 25 THEN*

*RAISE\_APPLICATION\_ERROR (-20101, 'Noul pret ' || :NEW.PRET\_LISTA || ' nu poate fii modificat daca este mai mare sau egal cu 25 ' || :*

*OLD.CREDIT\_LIMIT);*

*END IF;*

*END;*

