**Gestionarea stocului unei firme**

* **Semestrul 2-**

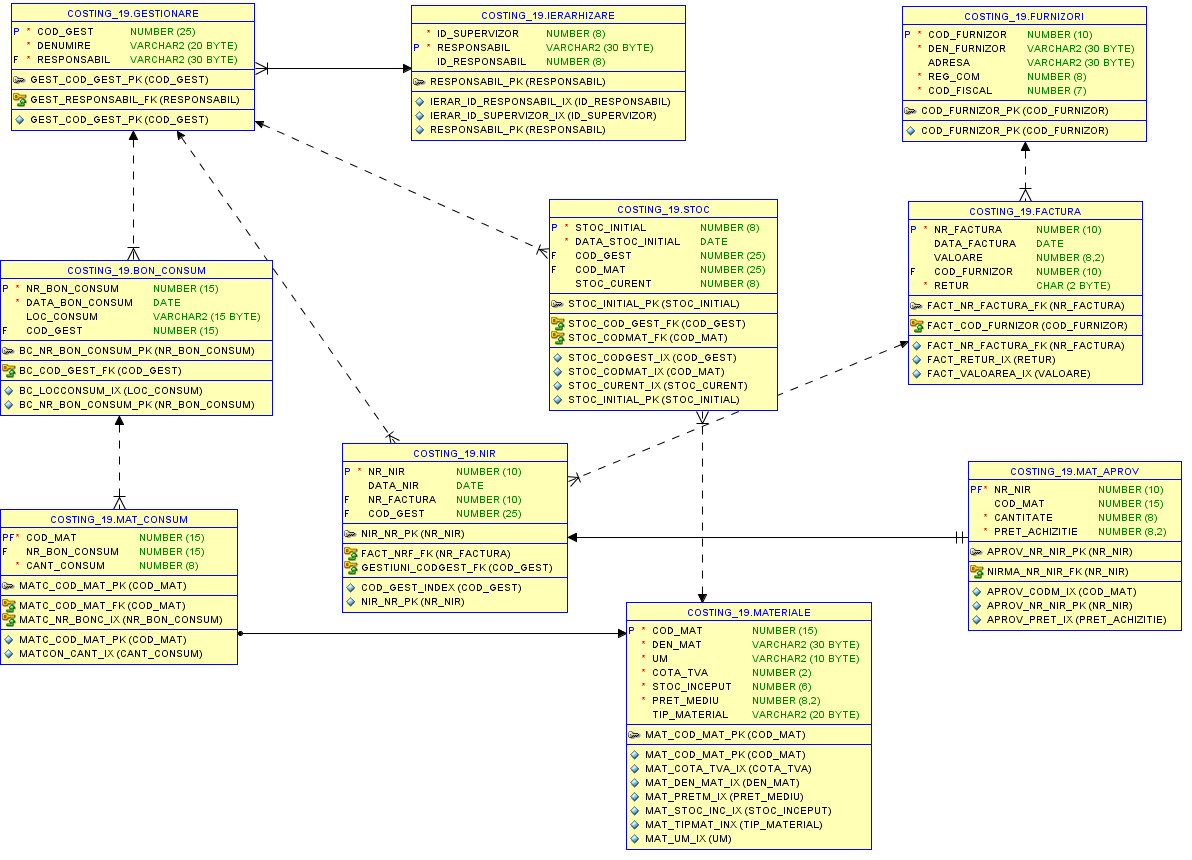
Glejaru Costin-Cosmin

Grupa 1049, Seria C

2020-2021

**Gestiunea stocului unei firme**

Schema următoare descrie situația unei firme care dorește informatizarea gestiunii stocului.



Pentru furnizorii firmei, se cunosc: un număr unic de identificare ,denumirea ,adresa, numărul de la Registrul  Comerțului, si un cod fiscal. Materiile prime sunt achiziționate de la furnizori prin intermediul facturilor ce conțin un număr unic de identificare ,data , valoare si codul furnizorilor. Materiile prime sunt caracterizate prin cod, denumire, unitate de măsură, , tva, stoc la începutul anului, dar si un preț mediu. Pentru fiecare factura, se efectuează o Nota de Intrare-Recepție.(cred ca m am complicat aici puțin, dar am realizat ca e destul de importanta), care conține un număr unic, data întocmirii intrării si recepție, numărul facturii si codul de gestiune.

După fiecare aprovizionare se calculează prețul mediu. Materiile prime necorespunzătoare se returnează furnizorului, primindu-se pentru aceasta o factura de retur, astfel încât, pentru fiecare factura primita, valoarea acesteia sa fie egala cu valoarea materiilor prime aprovizionate si recepționate, plus valoarea facturii de retur, daca aceasta exista. Factura de retur conține un nr unic, data si valoarea pentru fiecare produs returnat.

Fiecare gestiune se caracterizează printr-un cod, denumire si responsabil.

In momentul consumului se întocmește un bon de consum de către gestiunea eminenta, caracterizată printr-un număr, data, locul de consum. Tabela ierarhizarea ne ajută să ținem ușor evidența persoanelor care se ocupă de gestionarea mărfii.

Din punct de vedere economic, această bază de date ne ajută să eficientizăm toate procesele din interiorul gestiunii. De exemplu, putem ține evidența facturilor care au avut parte de retur sau nu, având parte de evidența plăților făcute către furnizori, economisind astfel resurse precum timpul cât și financiare.

De asemenea, aplicația ne ajută să combatem problemele de gestionare, fiecare responsabil având parte de o gestiune proprie și un supervizor.

**Operații cu structuri repetitive.**

SET SERVEROUTPUT ON;

1.

--Un block Pl care sa afiseze denumirea furnizorilor,registrul comertului si

--codul fiscal cu cod ul furnizorului intre 100-120

DECLARE

v\_COD FURNIZORI.COD\_FURNIZOR%TYPE;

V\_DEN FURNIZORI.DEN\_FURNIZOR%TYPE;

V\_REG FURNIZORI.REG\_COM%TYPE;

V\_CODF FURNIZORI.COD\_FISCAL%TYPE;

I NUMBER(4):=101;

BEGIN

WHILE I<=120 LOOP

SELECT COD\_FURNIZOR,DEN\_FURNIZOR,REG\_COM,COD\_FISCAL INTO

V\_COD,V\_DEN,V\_REG,V\_CODF FROM FURNIZORI WHERE

COD\_FURNIZOR=I;

DBMS\_OUTPUT.PUT\_LINE(V\_DEN||' '||V\_REG||' '||V\_CODF||' '||I);

I:=I+1;

END LOOP;

END;

/

-Metoda 2

DECLARE

v\_COD FURNIZORI.COD\_FURNIZOR%TYPE;

V\_DEN FURNIZORI.DEN\_FURNIZOR%TYPE;

V\_REG FURNIZORI.REG\_COM%TYPE;

V\_CODF FURNIZORI.COD\_FISCAL%TYPE;

BEGIN

FOR I IN 101..120 LOOP

SELECT COD\_FURNIZOR,DEN\_FURNIZOR,REG\_COM,COD\_FISCAL INTO

V\_COD,V\_DEN,V\_REG,V\_CODF FROM FURNIZORI WHERE

COD\_FURNIZOR=I;

DBMS\_OUTPUT.PUT\_LINE(V\_DEN||' '||V\_REG||' '||V\_CODF||' '||I);

END LOOP;

END;

/

Metoda 3-

loop…exit

DECLARE

v\_COD FURNIZORI.COD\_FURNIZOR%TYPE;

V\_DEN FURNIZORI.DEN\_FURNIZOR%TYPE;

V\_REG FURNIZORI.REG\_COM%TYPE;

V\_CODF FURNIZORI.COD\_FISCAL%TYPE;

I NUMBER:=100;

BEGIN

LOOP

SELECT COD\_FURNIZOR,DEN\_FURNIZOR,REG\_COM,COD\_FISCAL INTO

V\_COD,V\_DEN,V\_REG,V\_CODF FROM FURNIZORI WHERE

COD\_FURNIZOR=I;

DBMS\_OUTPUT.PUT\_LINE(V\_DEN||' '||V\_REG||' '||V\_CODF||' '||I);

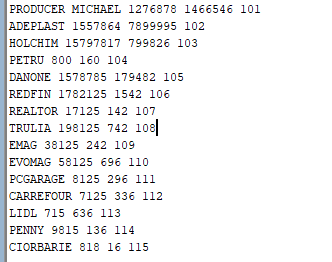
I:=I+1;

EXIT WHEN I>120;

END LOOP;

END;

/



2.

--printr un bloc pl/sql sa se atribuie PRODUSELOR in functie de UM PE CARE O AU REDUCERI

--PRODUSELE cu UM="BUC" vor avea REDUCERE intre 10%

-- UM="LITRI" REDUCERE 15%

-- UM="KG" REDUCERE 5%

--restul vor avea REDUCERE nemodificat

BEGIN

UPDATE MATERIALE

SET REDUCERE=CASE WHEN UM='BUC' THEN 0.15

WHEN UM='LITRI' THEN 0.1

WHEN UM='KG' THEN 0.05

ELSE REDUCERE END;

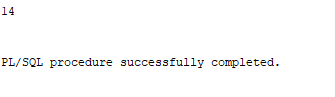
IF SQL%ROWCOUNT =0 THEN DBMS\_OUTPUT.PUT\_LINE('Nu exista modificari');

ELSE DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT);

END IF;

END;

/



**Operații cu cursori**

3.

-- SA SE SCRIE UN BLOC PL/SQL

--Sa se afiseze toate elementele de stoc care au codul materialului intre 10001 si 10012

DECLARE

CURSOR C IS

SELECT \* FROM STOC

WHERE COD\_MAT IN

(SELECT COD\_MAT FROM STOC WHERE COD\_MAT BETWEEN 10001 AND 100012);

V\_NR NUMBER;

BEGIN

OPEN C;

LOOP

FETCH C INTO V\_NR;

EXIT WHEN C%ROWCOUNT>3;

DBMS\_OUTPUT.PUT\_LINE(V\_NR.COD\_GEST||' '||V\_NR.COD\_MAT);

END LOOP;

CLOSE C;

END;

/

4 .

--Construiți un bloc PL/SQL prin care să se afișeze informații despre stocul initial,stocul curent si den mat

DECLARE

CURSOR C IS

SELECT S.STOC\_INITIAL,S.STOC\_CURENT,M.DEN\_MAT,M.COD\_MAT

FROM STOC S JOIN MATERIALE M

ON S.COD\_MAT=M.COD\_MAT

ORDER BY S.STOC\_CURENT DESC;

BEGIN

FOR VAR IN C LOOP

DBMS\_OUTPUT.PUT\_LINE(VAR.COD\_MAT||' '||VAR.DEN\_MAT||VAR.STOC\_CURENT||' '||VAR.STOC\_INITIAL||' ');

END LOOP;

5.

--SA SE AFISEZE DENUMIREA FURNIZORULUI, CODUL FURNIZORULUI PENTRU CARE S-A FACUT RETUR

END;

DECLARE

CURSOR FACT\_CURSOR IS SELECT DEN\_FURNIZOR,COD\_FURNIZOR

FROM FURNIZORI F INNER JOIN FACTURA USING (COD\_FURNIZOR)

WHERE RETUR='DA';

V\_DEN furnizori.den\_furnizor%TYPE;

V\_COD furnizori.cod\_furnizor%type;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('furnizorilor pentru care s a facut retur');

open fact\_cursor;

loop

FETCH FACT\_CURSOR INTO V\_DEN,V\_COD;

EXIT WHEN FACT\_CURSOR%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('NUMELE FURNIZORULUI ESTE'||V\_DEN||'SI ARE CODUL'||V\_COD);

END LOOP;

CLOSE FACT\_CURSOR;

END;

/

6.

-- SA SE AFISEZE GESTIONARI, SI COD DE GESTIUNE CARE AU ID\_RESPONSABBIL=3

DECLARE

CURSOR GEST\_CURSOR IS SELECT DENUMIRE,COD\_GEST

FROM GESTIONARE

WHERE responsabil=3;

V\_DEN gestionare.denumire%type;

v\_cod gestionare.cod\_gest%type;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('LISTA GESTIONARILOR CARE AU SUPERVIZOR=3');

OPEN GEST\_CURSOR;

LOOP

FETCH GEST\_CURSOR INTO V\_DEN,V\_COD;

EXIT WHEN GEST\_CURSOR%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('NUMELE GESTIONARULUI'||V\_DEN||'SI CODUL DE GESTIUNE'||V\_COD);

END LOOP;

CLOSE GEST\_CURSOR;

END;

**Excepții**

7

--.Creati un bloc PL/SQL pentru a selecta numarul facturii si data acestora pentru un an introdus de la tastura

DECLARE

V\_AN NUMBER := :AN;

V\_NR FACTURA.NR\_FACTURA%TYPE;

V\_DATA FACTURA.DATA\_FACTURA%TYPE;

BEGIN

SELECT F.NR\_FACTURA,F.DATA\_FACTURA

INTO V\_NR,V\_DATA

FROM FACTURA F

WHERE EXTRACT(YEAR FROM V\_DATA)=V\_AN;

DBMS\_OUTPUT.PUT\_LINE('Factura cu numarul'||V\_NR||'DIN DATA'||V\_DATA);

EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('PREA MULTE FACTURI IN ANUL INTRODUS');

WHEN NO\_DATA\_FOUND THEN DBMS\_OUTPUT.PUT\_LINE('NICI O FACTURA IN ANUL CITIT');

END;

/

8

-- Creati un bloc PL/SQL prin care dubleaza pretul(pret\_mediu) al carui cod este citit de la tastatura

DECLARE

V\_COD NUMBER := :COD;

NU\_EXISTA EXCEPTION;

V\_NR NUMBER;

BEGIN

SELECT COUNT(COD\_MAT) INTO V\_NR

FROM MATERIALE

WHERE ID\_PRODUS=V\_COD;

IF V\_NR=0 THEN

RAISE NU\_EXISTA;

ELSE

UPDATE MATERIALE

SET PRET\_MEDIU=PRET\_MEDIU\*2

WHERE ID\_PRODUS=V\_COD;

DBMS\_OUTPUT.PUT\_LINE('S A DUBLAT PRETUL MEDIU AL MATERIALULUI CU COD UL'||V\_COD);

END IF;

EXCEPTION

WHEN NU\_EXISTA THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXISTA');

END;

/

9.

-- . Într-un bloc PL/SQL citiți de la tastatură identificatorul unui materiale. Afișați denumirea materialului care are acel cod.

-- De asemenea, afisati cantitatea din acel material.

DECLARE

V\_ID NUMBER := :ID;

V\_DEN MATERIALE.DEN\_MAT%TYPE;

V\_CANT NUMBER;

V\_NR NUMBER;

NU\_A\_FOST\_GASIT NUMBER;

BEGIN

SELECT DEN\_MAT INTO V\_DEN FROM MATERIALE WHERE COD\_MAT=V\_ID;

SELECT COUNT(COD\_MAT) INTO V\_NR FROM MATERIALE WHERE COD\_MAT=V\_ID;

IF V\_NR>0 THEN

SELECT STOC\_CURENT INTO V\_CANT FROM STOC WHERE COD\_MAT=V\_ID;

DBMS\_OUTPUT.PUT\_LINE('materialul cu cod ul'||v\_id||'are denumirea'||v\_den||'are un stoc in valoare de'||v\_cant);

ELSE

RAISE NU\_A\_FOST\_GASIT;

END IF;

EXCEPTION

WHEN NU\_A\_FOST\_GASIT THEN DBMS\_OUTPUT.PUT\_LINE('PRODUSUL NU ARE STOC');

WHEN NO\_DATA\_FOUND THEN DBMS\_OUTPUT.PUT\_LINE('produsul nu exista');

END;

/

10.

--. Intr-un bloc PL/SQL sa se afiseze valoarea totala(cantitate\*pret\_achizitie) si codul materialului

DECLARE

V\_COD NUMBER := :V\_ID;

V\_VAL NUMBER;

V\_MAT MAT\_APROV.COD\_MAT%TYPE;

V\_DEN MATERIALE.DEN\_MAT%TYPE;

NU\_EXISTA\_MAT EXCEPTION;

V\_NR NUMBER;

BEGIN

SELECT COUNT(COD\_MAT)INTO V\_NR FROM MAT\_APROV WHERE COD\_MAT=V\_COD;

IF V\_NR=0 THEN

RAISE NU\_EXISTA\_MAT;

ELSE

SELECT COD\_MAT,(PRET\_ACHIZITIE\*CANTITATE) INTO V\_MAT,V\_VAL

FROM mat\_aprov

WHERE COD\_MAT=V\_COD;

SELECT DEN\_MAT INTO V\_DEN FROM MATERIALE WHERE COD\_MAT=V\_COD;

DBMS\_OUTPUT.PUT\_LINE('PRODUSUL CU DENUMIREA'||V\_DEN||'SI CU CODUL'||V\_MAT||'ARE O VALOAREA TOTALA'||V\_VAL);

END IF;

EXCEPTION

WHEN NU\_EXISTA\_MAT THEN

DBMS\_OUTPUT.PUT\_LINE('NU EXISTA MATERIALUL'||V\_COD);

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NU AU FOST GASITE DATE');

--WHEN TOO\_MANY\_ROWS THEN DBMS\_OUTPUT.PUT\_LINE('MAI MULTE PRODUSE CU ACELASI COD');

WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE('A APARUT EROAREA'||SQLERRM);

END;

/

**Proceduri/funcții**

11.

--SA SE CONSTRUIASCA O PROCEDURA CARE SA DUBLEZE PRETUL MEDIU MATERIALELOR CARE AU PARTE DE REDUCERE

-- SI AU STOC IN ANUL PRIMIT CA PARAMETRU

SET SERVEROUT ON;

CREATE OR REPLACE PROCEDURE DUBLEAZA\_PRET(V\_AN IN NUMBER)

IS

NU\_AVEM\_MATERIAL EXCEPTION;

BEGIN

UPDATE MATERIALE M

SET PRET\_MEDIU=PRET\_MEDIU\*2

WHERE REDUCERE IS NOT NULL

AND (SELECT COUNT(COD\_MAT) FROM STOC WHERE EXTRACT(YEAR FROM DATA\_STOC\_INITIAL)=V\_AN AND COD\_MAT=M.COD\_MAT)>0;

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('AU FOST MODIFICARI' || SQL%ROWCOUNT);

ELSE

RAISE NU\_AVEM\_MATERIAL;

END IF;

EXCEPTION

WHEN NU\_AVEM\_MATERIAL THEN DBMS\_OUTPUT.PUT\_LINE('NU AVEM MATERIALUL');

END;

/

--APEL

BEGIN

DUBLEAZA\_PRET(2020);

END;

12.

-- SA SE CREEZE O PROCEDURA CARE PRIMESTE UN PARAMETRU DE TIP IN CODUL MATERIALULUI SI RETURNEAZA

-- PRIN PARAMETRII DE TIP OUT DENUMIREA SI TIPUL MATERIALULUI

CREATE OR REPLACE PROCEDURE CAUTA\_MATERIAL

(COD IN MATERIALE.COD\_MAT%TYPE,

P\_DEN OUT MATERIALE.DEN\_MAT%TYPE,

P\_TIP OUT MATERIALE.DEN\_MAT%TYPE

)

IS

BEGIN

SELECT DEN\_MAT,TIP\_MATERIAL INTO P\_DEN,P\_TIP FROM MATERIALE

WHERE COD\_MAT=COD;

DBMS\_OUTPUT.PUT\_LINE('PRODUSUL ARE DENUMIREA'||P\_DEN||'SI ESTE UN'||P\_TIP);

END;

/

DECLARE

V\_DEN MATERIALE.DEN\_MAT%TYPE;

V\_TIP MATERIALE.TIP\_MATERIAL%TYPE;

BEGIN

CAUTA\_MATERIAL(10001,V\_DEN,V\_TIP);

END;

/

13.

-- SA SE REALIZEZE O PROCEDURA CARE ARE DOI PARAMETRI: P\_COD\_MAT SI RED PRIN CARE SE REDUCE PRETUL\_MEDIU.

CREATE OR REPLACE PROCEDURE MODIFICA\_PRET\_MEDIU

(P\_COD IN MATERIALE.COD\_MAT%TYPE,

RED IN NUMBER)

IS

V\_PRET MATERIALE.PRET\_MEDIU%TYPE;

BEGIN

SELECT PRET\_MEDIU INTO V\_PRET FROM MATERIALE WHERE COD\_MAT=P\_COD;

DBMS\_OUTPUT.PUT\_LINE('PRODUSUL ARE PRETUL'||V\_PRET);

UPDATE MATERIALE

SET PRET\_MEDIU=PRET\_MEDIU\*(1-RED/100)

WHERE COD\_MAT=P\_COD;

SELECT PRET\_MEDIU INTO V\_PRET FROM MATERIALE WHERE COD\_MAT=P\_COD;

DBMS\_OUTPUT.PUT\_LINE('PRODUSUL ARE PRETUL ACUM'||V\_PRET);

END;

/

BEGIN

MODIFICA\_PRET\_MEDIU(10012,10);

END;

/

14.

-- SA SE CREEZE FUNCTIA VERIFICA\_PRET CARE RETURNEAZA TRUE/FALSE DACA PRETUL MEDIU ESTE MAI MIC

-- DECAT PRET ACHIZITIE.

CREATE OR REPLACE FUNCTION VERIFICA\_PRET (

P\_COD IN MATERIALE.COD\_MAT%TYPE

)

RETURN BOOLEAN

IS

V\_PM MATERIALE.PRET\_MEDIU%TYPE;

V\_PA MAT\_APROV.PRET\_ACHIZITIE%TYPE;

BEGIN

SELECT PRET\_MEDIU INTO V\_PM FROM MATERIALE WHERE COD\_MAT=P\_COD;

SELECT PRET\_ACHIZITIE INTO V\_PA FROM MAT\_APROV WHERE COD\_MAT=P\_COD;

IF V\_PM<V\_PA THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN NULL;

END;

/

CALL VERIFICA\_PRET;

15.-- SA SE CREEZE O FUNCTIE CARE AFISEAZA DENUMIREA, ADRESA AL UNUI FURNIZOR

-- DAT CA PARAMETRU

CREATE OR REPLACE FUNCTION AFISEAZA\_INFORMATII(P\_COD IN FURNIZORI.COD\_FURNIZOR%TYPE)

RETURN VARCHAR2

IS

V\_INF FURNIZORI.DEN\_FURNIZOR%TYPE;

BEGIN

SELECT DEN\_FURNIZOR||' '||ADRESA INTO V\_INF

FROM FURNIZORI

WHERE COD\_FURNIZOR=P\_COD;

RETURN V\_INF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN NULL;

END;

/

--APEL—

--APEL

SELECT COD\_FURNIZOR,AFISEAZA\_INFORMATII(COD\_FURNIZOR)FROM FURNIZORI;

--APEL

DECLARE

V\_NUME VARCHAR2(50);

BEGIN

V\_NUME:=afiseaza\_informatii(101);

IF V\_NUME IS NULL

THEN DBMS\_OUTPUT.PUT\_LINE('NU EXISTA FURNIZORU');

ELSE

DBMS\_OUTPUT.PUT\_LINE(V\_NUME);

END IF;

END;

/

16.

-- SA SE AFISEZE VECHIMEA PRODUSELOR CARE FAC PARTE DIN STOCUL INITIAL

CREATE OR REPLACE FUNCTION VECHIME\_STOC(P\_COD STOC.COD\_MAT%TYPE)

RETURN NUMBER

IS

V\_VEC NUMBER;

BEGIN

SELECT EXTRACT(YEAR FROM DATA\_STOC\_INITIAL)INTO V\_VEC

FROM STOC

WHERE COD\_MAT=P\_COD;

RETURN EXTRACT(YEAR FROM SYSDATE)-V\_VEC;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN NULL;

END;

/

-- APEL

DECLARE

VEC NUMBER;

BEGIN

VEC:=vechime\_stoc(10009);

IF VEC IS NULL THEN DBMS\_OUTPUT.PUT\_LINE('NU EXISTA PRODUSUL');

ELSE

DBMS\_OUTPUT.PUT\_LINE('ANGAJATUL ARE O VECHIME DE'||VEC||'ANI');

END IF;

END;

/

17.

--SA SE AFISEZE DATELE DESPRE FURNIZORI UTILIZAND CODUL FURNIZORLUI

CREATE OR REPLACE FUNCTION DATE\_FURN(COD IN FURNIZORI.COD\_FURNIZOR%TYPE)

RETURN FURNIZORI%ROWTYPE

IS

REC\_FURN FURNIZORI%ROWTYPE;

BEGIN

SELECT \* INTO REC\_FURN

FROM FURNIZORI

WHERE COD\_FURNIZOR=COD;

RETURN REC\_FURN;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20000, 'NU EXISTA FURNIZORUL');

END;

/

--APEL

DECLARE

V\_FUR FURNIZORI%ROWTYPE;

BEGIN

v\_FUR := DATE\_FURN(105);

DBMS\_OUTPUT.PUT\_LINE(V\_FUR.DEN\_FURNIZOR);

END;

/

18.

Creează o procedura prin care sa se modifice cu x stocul curent al materialelor achiziționate în ultimul an (x este parametrul procedurii).

CREATE OR REPLACE PROCEDURE MODIFICA\_STOC(X IN NUMBER)

IS

EXCEPTIE EXCEPTION;

BEGIN

UPDATE STOC

SET STOC\_CURENT=STOC\_CURENT + X

WHERE STOC\_INITIAL IN (SELECT STOC\_INITIAL FROM STOC WHERE (SYSDATE-DATA\_STOC\_INITIAL)/365<1);

IF SQL%NOTFOUND THEN RAISE EXCEPTIE;

END IF;

EXCEPTION

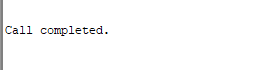
WHEN EXCEPTIE THEN DBMS\_OUTPUT.PUT\_LINE('NU EXISTA MATERIALE CARE SA NU RESPECTE CRITERIUL');

END;

/



CALL MODIFICA\_STOC(200);



19.

--. Creează o procedura prin care sa se afiseze furnizorii de la care s-au achiziționat

--materiale la un preț de achiziție mai mare decât prețul de achiziție mediu pentru orașul furnizorului.

CREATE OR REPLACE PROCEDURE AFISEAZA AS

V\_MEDIU NUMBER;

V\_MEDIU:=(SELECT AVG(M.PRET\_ACHIZITIE) FROM MAT\_APROV M, FURNIZORI F, FACTURA FC, NIR N WHERE F.COD\_FURNIZOR=FC.COD\_FURNIZOR AND FC.NR\_FACTURA=N.NR\_FACTURA AND M.NR\_NIR=N.NR\_NIR AND ADRESA=F.ADRESA )

V\_COD FURNIZORI.COD\_FURNIZOR%TYPE;

CURSOR C IS SELECT F.COD\_FURNIZOR

FROM FURNIZORI F, FACTURA FC, MAT\_APROV M, NIR N

WHERE F.COD\_FURNIZOR=FC.COD\_FURNIZOR AND M.NR\_NIR=N.NR\_NIR AND FC.NR\_FACTURA=N.NR\_FACTURA AND PRET\_ACHIZITIE> V\_MEDIU

BEGIN

OPEN C;

FETCH C INTO V\_COD;

DBMS\_OUTPUT.PUT\_LINE(V\_COD);

END LOOP;

CLOSE C;

END;

--APEL

CALL AFISEAZA;

**Declanșatori**

20. -- Realizati un declansator cantitate pozitiva care sa nu permita in tabela materiale aprovizionate

-- valori negative ale cantitati

CREATE OR REPLACE TRIGGER CANTITATE\_POZ

BEFORE INSERT OR UPDATE ON MAT\_APROV

FOR EACH ROW

BEGIN

IF (:NEW.CANTITATE<=0)

THEN

RAISE\_APPLICATION\_ERROR(-20000,'CANTITATEA NU POATE FI INTRODUSA');

END IF;

END;

/

UPDATE MAT\_APROV

SET CANTITATE=-1

WHERE COD\_MAT=10001;

21. -- Sa se realizeze un trigger care asigura unicitatea codului un material folosind valorile generate de o secvența

CREATE SEQUENCE MATERIALE\_SECV

START WITH 10000

INCREMENT BY 1

MAXVALUE 10200

NOCYCLE;

CREATE OR REPLACE TRIGGER GENERARE\_COD\_MAT

BEFORE INSERT ON MATERIALE

FOR EACH ROW

BEGIN

SELECT PRODUSE\_SECV.NEXTVAL INTO :NEW.COD\_MAT FROM DUAL;

END;

22.

-- Sa se realizeze UN TRIGGER care sa limiteze introducerea de preturi de achizitie intre doua limite min si max

CREATE OR REPLACE TRIGGER VERIFICA\_LIMITA

BEFORE UPDATE OF PRET\_ACHIZITIE ON MAT\_APROV

FOR EACH ROW

DECLARE

V\_MIN MAT\_APROV.PRET\_ACHIZITIE%TYPE;

V\_MAX MAT\_APROV.PRET\_ACHIZITIE%TYPE;

E\_INVALID EXCEPTION;

BEGIN

SELECT MIN(PRET\_ACHIZITIE),MAX(ACHIZITIE)

INTO V\_MIN,V\_MAX

FROM MAT\_APROV

WHERE COD\_MAT=:COD\_MAT;

IF(V\_MIN<:NEW.PRET\_ACHIZITIE)OR

(V\_MAX>:NEW.PRET\_ACHIZITIE) THEN

RAISE E\_INVALID;

END IF;

EXCEPTION

WHEN E\_INVALID THEN

RAISE\_APPLICATION\_ERROR(-20567,'NU POATE FI INTRODUS DIN CAUZA INTERVALULUI DE PRET');

END;

UPDATE MAT\_APROV

SET PRET\_ACHIZITIE=750000

WHERE COD\_MAT BETWEEN(10001,100015);

23. -- Sa se creeze un declansator care:

--a.daca este eliminata o factura, va sterge toate actele care contin aceea factura

--b. daca se schimba numarul unei facturi, atunci va modifica pentru fiecare act acel numar

CREATE OR REPLACE TRIGGER DECL\_FACT

BEFORE DELETE OR UPDATE OF NR\_FACTURA ON FACTURA

FOR EACH ROW

BEGIN

IF DELETING THEN

DELETE FROM FACTURA

WHERE NR\_FACTURA=:OLD.NR\_FACTURA;

END IF;

IF UPDATING AND NR\_FACTURA!=:NEW.NR\_FACTURA THEN

UPDATE FACTURA

SET NR\_FACTURA=:NEW.NR\_FACTURA

WHERE NR\_FACTURA=:OLD.NR\_FACTURA;

END IF;

END;

DELETE FROM FACTURA

WHERE NR\_FACTURA=773;

UPDATE FACTURA

SET NR\_FACTURA=777

WHERE NR\_FACTURA=212;

**Pachete de subprograme**

24. -- sa se creeze un pachet care sa creeze o adauge o factura,

--sa adauge o factura, modifica factura, si pentru a sterge o factura.

CREATE OR REPLACE PACKAGE PACHET1 AS

PROCEDURE ADAUGA\_FACTURA(NR NUMBER,DATA DATE,VAL NUMBER,COD NUMBER,RET CHAR);

PROCEDURE MODIFICA\_FACTURA(NR NUMBER,VAL NUMBER);

PROCEDURE STERGE\_FUNCTIE(NR NUMBER);

END PACHET1;

CREATE OR REPLACE PACKAGE BODY PACHET1 AS

PROCEDURE ADAUGA\_FACTURA(NR NUMBER,DATA DATE,VAL NUMBER,COD NUMBER,RET CHAR) AS

V\_NR NUMBER;

EX EXCEPTION;

BEGIN

SELECT COUNT(NR\_FACTURA) INTO V\_NR FROM FACTURA WHERE NR\_FACTURA=NR;

IF(V\_NR>0)THEN RAISE EX;

ELSE

INSERT INTO FACTURA VALUES(NR,DATA,VAL,COD,RET);

END IF;

EXCEPTION

WHEN EX THEN DBMS\_OUTPUT.PUT\_LINE('EXISTA O FACTURA CU ACEL NUMAR');

END;

PROCEDURE MODIFICA\_FACTURA(NR NUMBER,VAL NUMBER) IS

NUMAR NUMBER;

EX EXCEPTION;

BEGIN

SELECT COUNT(NR)INTO NUMAR FROM FACTURA WHERE NR\_FACTURA=NR;

IF(NUMAR=0)THEN

RAISE EX;

ELSE

UPDATE FACTURA

SET VALOARE=VAL

WHERE NR\_FACTURA=NR;

END IF;

EXCEPTION

WHEN EX THEN DBMS\_OUTPUT.PUT\_LINE('NU EXISTA FACTURA CU ACEL NUMAR');

END;

PROCEDURE STERGE\_FACTURA(NR NUMBER)IS

NUMAR NUMBER;

EX EXCEPTION;

BEGIN

SELECT COUNT(NR)INTO NUMAR FROM FACTURA WHERE NR\_FACTURA=NR;

IF(NUMAR=0)THEN

RAISE EX;

ELSE

DELETE FROM FACTURA WHERE NR\_FACTURA=NR;

END IF;

EXCEPTION

WHEN EX THEN DBMS\_OUTPUT.PUT\_LINE('NU EXISTA FACTURA CU ACEL NUMAR');

END;

END PACKET1;

/

EXECUTE PACHET1. ADAUGA\_FACTURA(215,SYSDATE,7000,112, 'DA');

EXECUTE PACHET1.MODIFICA\_FACTURA(215,70150);

EXECUTE PACHET1.STERGE\_FACTURA(215);

25. -- sa se scrie un pachet de subprograme care sa modifica un material

CREATE OR REPLACE PACKAGE ACTUALIZARE\_MAT IS

PROCEDURE MODIFICA\_MATERIAL

(P\_COD MATERIALE.COD\_MAT%TYPE,

P\_DEN MATERIALE.DEN\_MAT%TYPE,

P\_UM MATERIALE.UM%TYPE,

P\_TIP MATERIALE.TIP\_MATERIAL%TYPE);

PROCEDURE MODIFICA\_MATERIAL

(P\_COD MATERIALE.COD\_MAT%TYPE,

P\_DEN MATERIALE.DEN\_MAT%TYPE);

END;

-- sa se scrie un pachet de subprograme care sa modifica un material

CREATE OR REPLACE PACKAGE BODY ACTUALIZARE\_MAT IS

EXCEPTIE EXCEPTION;

PROCEDURE MODIFICA\_MATERIAL

(P\_COD MATERIALE.COD\_MAT%TYPE,

P\_DEN MATERIALE.DEN\_MAT%TYPE,

P\_UM MATERIALE.UM%TYPE,

P\_TIP MATERIALE.TIP\_MATERIAL%TYPE)

IS BEGIN

IF EXISTA\_COD(P\_COD) THEN

UPDATE MATERIALE

SET DEN\_MAT=P\_DEN, UM=P\_UM, TIP\_MATERIAL=P\_TIP

WHERE COD\_MAT=P\_COD;

ELSE

RAISE EXCEPTIE;

END IF;

EXCEPTION

WHEN EXCEPTIE THEN

DBMS\_OUTPUT.PUT\_LINE('PRODUSUL CU ACEST COD NU EXISTA');

END;

PROCEDURE MODIFICA\_MATERIAL

(P\_COD MATERIALE.COD\_MAT%TYPE,

P\_PRET MATERIALE.PRET\_MEDIU%TYPE)

IS BEGIN

IF EXISTA\_COD(P\_COD) THEN

UPDATE MATERIALE

SET PRET\_MEDIU=P\_PRET

WHERE COD\_MAT=P\_COD;

ELSE

RAISE EXCEPTION;

END IF;

EXCEPTION

WHEN EXCEPTION

THEN DBMS\_OUTPUT.PUT\_LINE('PRODUSUL CU ACEST COD NU EXISTA');

END;

FUNCTION EXISTA\_COD(P\_COD MATERIALE.COD\_MAT%TYPE)

RETURN BOOLEAN

IS

VUNU NUMBER;

BEGIN

SELECT 1 INTO VUNU

FROM MATERIALE

WHERE COD\_MAT=P\_COD;

RETURN TRUE;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN RETURN FALSE;

END;

END ACTUALIZARE\_MAT;

/

--APEL

EXECUTE ACTUALIZARE\_MAT.MODIFICA\_PRODUS(100017, 'CAFEA',30, 'HRANA');

SELECT \* FROM MATERIALE WHERE COD\_MAT=100017;

EXECUTE ACTUALIZARE\_MAT.MODIFICA\_PRODUS(100017,50);

SELECT \* FROM MATERIALE WHERE COD\_MAT=100017;

**Operații LDD**

26.

-- Sa se afiseze toate materialele care au un anumit cod

CREATE FUNCTION GET\_MATERIALE(COD NUMBER)

RETURN MATERIALE%ROWTYPE IS

V\_MAT VARCHAR(300);

MAT\_REC MATERIALE%ROWTYPE;

BEEGIN

V\_MAT :=

'SELECT \* FROM MATERIALE WHERE COD\_MAT = :COD';

EXECUTE IMMEDIATE V\_MAT INTO MAT\_REC USING COD;

RETURN MAT\_REC;

END;

DECLARE

MAT\_REC MATERIALE%ROWTYPE := GET\_MATERIALE(10001);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('MATERIALUL: '|| MAT\_REC.COD\_MAT);

END;

/