

# Arthur's PlayTime

Ion Alexandra - grupa 242

## CUPRINS !

- **Descrierea bazei de date (pag 1 - 3)**
- **Creare si inserare (pag 4 - 6)**
- **Diagramele E/R si Conceptuala (pag 7 - 8)**
- **Exercițiile 6 – 12 (pag 9 - 26 )**
- **Exercițiile 13, 14 (pag 27 - 39 )**

Această bază de date gestionează informațiile firmei Arthur's PlayTime S.R.L., ținând o evidență clară asupra produselor, angajaților și clienților. Arthur's PlayTime este o afacere care se ocupă cu comercializarea de produse printate 3D. Fiecare produs vândut este marca proprie și necesita proiectarea, iar apoi printarea cu ajutorul imprimantei 3D.

Fiecare produs are nevoie de un proiect 3D înainte de a fi printat. Acest model se realizează într-o aplicație specială, de către un proiectant, urmând să fie trimis după aceea către o imprimanta 3D compatibilă cu tipul de filament de care este nevoie pentru acel produs. Un produs poate avea orice culoare este disponibilă pentru tipul său de filament. Un produs are mai multe caracteristici precum nume, pret, cost de producere, gramaj, dimensiune, greutate și poate fi personalizabil. Filamentul este materialul folosit pentru printarea 3D, fiind un plastic special, topit de imprimante și apoi turnat în straturi subțiri, pe baza modelului proiectat anterior. Fiecare tip de filament este compatibil doar cu anumite imprimante deținute de firmă. Acest material poate avea diferite culori, gramaje și puncte de topire. Filamentul este achiziționat de la diverși furnizori. Același tip de filament se poate achiziționa de la mai mulți furnizori. De asemenea, imprimantele au dimensiunea patului de printare diferită, acesta fiind un aspect important atunci când se stabilește compatibilitatea între imprimantă, filament și produs. Angajatul care se ocupă de printarea și finisarea produsului este executantul. Fiecare produs este încadrat într-o categorie. În concluzie, un produs este realizat de doi angajați, cu job-urile de proiectant și executant.

Firma are mai mulți clienți care comunică cu agenții de la relații publice. Fiecare client înregistrat în baza de date a dat cel puțin o comandă. O comandă este formată din cel puțin un produs. Mai mult, fiecare client poate da o recenzie produselor. Așadar, un produs poate avea mai multe recenzii, sub formă de număr de stele, de la diferiți clienți. Un produs poate să nu fi fost vândut niciodată, așadar nu are recenzie, dar în același timp, se poate ca niciun client care l-a achiziționat să nu fi acordat stele.

Baza de date este utilă pentru gestionarea tuturor datelor firmei, păstrând evidența tuturor produselor și detaliile tehnice ale acestora, clienților, angajaților și furnizorilor. Cu ajutorul acestui model se fluidizează gestionarea afacerii și buna desfășurare a activității firmei.

### **Reguli de funcționare ale modelului:**

- Un produs face parte dintr-o categorie, iar dintr-o categorie pot face parte atât mai multe produse, cât și niciunul.
- Un produs este făcut dintr-un singur tip de filament, dar din acel tip de filament pot fi făcute mai multe produse. Dintr-un tip de filament trebuie să existe cel puțin un produs.
- Un tip de filament are un sortiment de culori, dar minimul este de o culoare. Mai multe filamente pot avea aceeași culoare, dar pot exista culori care nu aparțin niciunui tip de filament.
- Un tip de filament este achiziționat de la mai mulți furnizori, dar de la cel puțin unul. În același timp, de la un furnizor se pot achiziționa mai multe tipuri de filament, dar minim unul.
- O imprimantă este compatibilă cu mai multe tipuri de filament, dar minim unul. Un tip de filament are minim o imprimantă cu care este compatibil, dar poate avea și mai multe.
- Un client poate avea mai multe comenzi, dar trebuie să dea minim o comandă pentru a exista în baza de date. O comandă aparține unui singur client.
- Dintr-o comandă pot face parte mai multe comenzi, dar trebuie să existe minim una. Un produs poate exista în mai multe comenzi, dar poate și să nu fi fost vândut niciodată.
- Un client poate da mai multe recenzii sau niciuna, dar o recenzie aparține unui singur client.
- O recenzie este a unui singur produs, dar un produs poate avea și mai multe recenzii, și niciuna.
- Un produs este realizat de unul sau mai mulți proiectanți în parteneriat cu unul sau mai mulți executanți.
- Proiectanții și executanții sunt angajați și fiecare trebuie să realizeze minim un produs.
- Mai mulți clienți comunica cu un angajat de tipul agent relații publice, dar nu sunt obligați să comunice cu cineva neapărat. Agenții pot comunica cu mai mulți clienți, dar și cu niciunul.

**Schemele relaționale** corespunzătoare diagramei conceptuale sunt următoarele:

1. FURNIZORI (id\_furnizor#, denumirea, telefon, email)
2. RECENTIE (id\_recenzie#, id\_produs#, id\_client#, nr\_stelute)
3. CLIENTI (id\_client#, nume, prenume, telefon, email)
4. IMPRIMANTE (id\_imprimanta#, nume, dimensiune\_pat)
5. FILAMENT (id\_filament#, tip\_filament, temperatura\_topire, gramaj)
6. CULORI (id\_culoare#, nume)
7. CATEGORIE (id\_categorie#, nume)
8. PRODUSE (id\_produs#, id\_categorie#, id\_filament#, nume, pret\_vanzare, cost\_producere, cantitate\_filament, dimensiune, greutate, personalizabil)
9. ANGAJATI (id\_angajat#, nume, prenume, telefon, salariu, data\_angajarii, job)
10. PROIECTANT (id\_angajat#, aplicatie)
11. EXECUTANT (id\_angajat#)
12. AGENT\_RELATII\_PUBLICE (id\_angajat#)
13. COMENZI (id\_comanda#, id\_client#, data, valoarea)
14. ACHIZITIE (id\_furnizor#, id\_filament#, data#)
15. COMPATIBILITATE (id\_filament#, id\_imprimanta#)
16. CULOARE\_FILAMENT (id\_filament#, id\_culoare#)
17. REALIZARE (id\_produs#, id\_proiectant, id\_executant#, durata)
18. CUPRINS\_COMENZI (id\_comanda#, id\_produs#, bucati)
19. COMUNICARE (id\_client#, id\_agent#)

## Ex 4 si 5: Crearea tabelelor în SQL și inserarea de date

ID_FURNIZOR	DENUMIREA	TELEFON	EMAIL
1	101 EMAG	0728282106	contact@emag.ro
2	102 OptimusDigital	0728282107	contact@optidigi.ro
3	103 HobbyMarket	0721516130	contact@hobby.ro
4	104 Printam3D	0721216130	contact@3D.ro
5	105 CEL	0372245334	contact@cel.ro
6	106 ArduShop	0727387467	contact@ardu.ro

ID_FILAMENT	TIP	TEMPERATURA_TOPIRE	GRAMAJ
1	101 PLA	200	500
2	102 TPU	250	500
3	103 PLA	230	1000
4	104 ABS	180	500
5	105 PETG	330	250
6	106 PLA	(null)	500

ID_CULOARE	NUME
1	2 Alb
2	3 Verde primavara
3	4 Negru
4	5 Rosu caramiziu
5	6 Rosu foc
6	7 Auriu

ID_CATEGORIE	NUME
1	101 Home
2	102 Lithophane
3	103 Suport Telefon
4	104 Produse Caini
5	105 Puzzle
6	106 Licheni

ID_IMPRIMANTA	NUME	DIMENSIUNE_PAT
1	20 Odysseuss	625
2	30 TevoTornado	900
3	40 Anycubic	600
4	50 Odysseuss	1000
5	60 Prusa	900
6	70 MakerBot	2500

ID_ANGAJAT	NUME	PRENUME	JOB	TELEFON	SALARIU	DATA_ANGAJARII	APLICATIE
1	101 Ion	Loredana	executant	0245222162	1500	10-MAY-21	(null)
2	102 Andreescu	Daiana	proiectant	0745222163	1800	09-JAN-10	Fusion360
3	103 Ramond	Liliana	agent	0745228182	3000	14-JUN-19	(null)
4	104 Flo	Luis	proiectant	0245552168	4500	25-MAY-18	Creo
5	105 Dan	Cornelia	proiectant	0243722175	1500	11-AUG-19	Fusion360
6	106 Remus	Denis	executant	0245722196	6500	10-MAY-15	(null)
7	107 Ion	Gabriel	agent	0712125866	3000	14-JUN-20	(null)

ID_COMANDA	ID_CLIENT	DATA	VALOAREA
1	10	101 10-MAY-21	150
2	11	102 15-APR-21	30
3	12	104 10-MAY-21	120
4	13	103 25-MAR-20	240
5	14	105 12-DEC-20	120
6	15	101 30-JAN-21	12
7	16	106 30-MAY-21	24
8	17	106 30-JUN-21	170

ID_RECENZIE	ID_PRODUS	ID_CLIENT	NR_STELUTE
1	101	119	101 *****
2	102	122	103 ***
3	103	127	101 *
4	104	127	104 *****
5	105	126	106 ***
6	106	120	106 ****

ID_CLIENT	NUME	PRENUME	TELEFON	EMAIL
1	101 Ion	Alexandra	0728282106	alexion200123@yahoo.com
2	102 Dinu	Cristian	0724702608	dinuc02@yahoo.com
3	103 Vulpe	Catalin	0785212833	vulpea69@gmail.com
4	104 Dinca	Ruxandra	0721283488	ruxi23@yahoo.com
5	105 Rotaru	Cristina	0700569254	cristina_rotaru32@yahoo.com
6	106 Iacob	Andreea	0721385065	icob_andreea@yahoo.com

ID_PRODUS	ID_CATEGORIE	ID_FILAMENT	NUME	PRET_VANZARE	COST_PRODUCERE	CANTITATE_FILAMENT	DIMENSIUNE	GREUTATE	PERSONALIZABIL
1	119	102	101 Lampa	120	50	30	25	80	1
2	120	101	101 Suport chei	50	(null)	15	25	80	1
3	121	102	103 Cub	100	(null)	30	900	(null)	0
4	122	103	106 Suport cu incarcator	12	(null)	30	25	50	0
5	123	106	102 Tablou licheni	40	(null)	30	800	(null)	0
6	124	106	105 Terariu	70	50	30	40	(null)	0
7	125	104	102 Mingie	10	3	30	9	30	0
8	126	101	104 Oaie-suport	30	5	10	10	(null)	0
9	127	105	101 Cutie labirint	24	10	5	9	10	0

### TABELE ASOCIATIVE

DATA	ID_FURNIZOR	ID_FILAMENT
1 03-JAN-20	102	101
2 12-MAY-20	102	101
3 02-AUG-19	104	102
4 24-MAR-21	106	102
5 25-JUN-21	102	103
6 01-MAY-21	106	104
7 12-MAY-20	101	105
8 12-DEC-20	101	105
9 12-DEC-20	105	105
10 12-APR-19	103	106
11 11-MAY-21	103	106

ID_IMPRIMANTA	ID_FILAMENT
1 30	101
2 40	102
3 50	102
4 60	102
5 50	103
6 70	103
7 20	104
8 70	104
9 30	105
10 30	106
11 60	106

ID_FILAMENT	ID_CULOARE
1 101	2
2 102	3
3 102	7
4 103	4
5 103	5
6 104	3
7 104	4
8 104	5
9 105	2
10 105	5
11 106	2
12 106	7

ID_PRODUS	ID_PROIECTANT	ID_EXECUTANT	DURATA
1 119	102	106	850
2 120	104	101	200
3 121	105	101	45
4 122	105	106	80
5 123	104	106	96
6 124	104	106	1500
7 125	104	106	450
8 126	105	101	200
9 127	102	101	15
10 121	102	101	45
11 124	104	101	1500

ID_CLIENT	ID_AGENT
1 101	103
2 102	103
3 103	103
4 103	107
5 104	103
6 104	107
7 105	103
8 105	107
9 106	103
10 106	107

ID_COMANDA	ID_PRODUS	BUCATI
1 10	120	1
2 10	121	1
3 11	126	1
4 12	119	1
5 13	119	2
6 14	119	1
7 15	122	1
8 16	127	1
9 17	121	1
10 17	124	1

S-au folosit secvente pentru inserarea datelor. Acestea au fost utilizate in generarea id-urilor.

### EXAMPLE

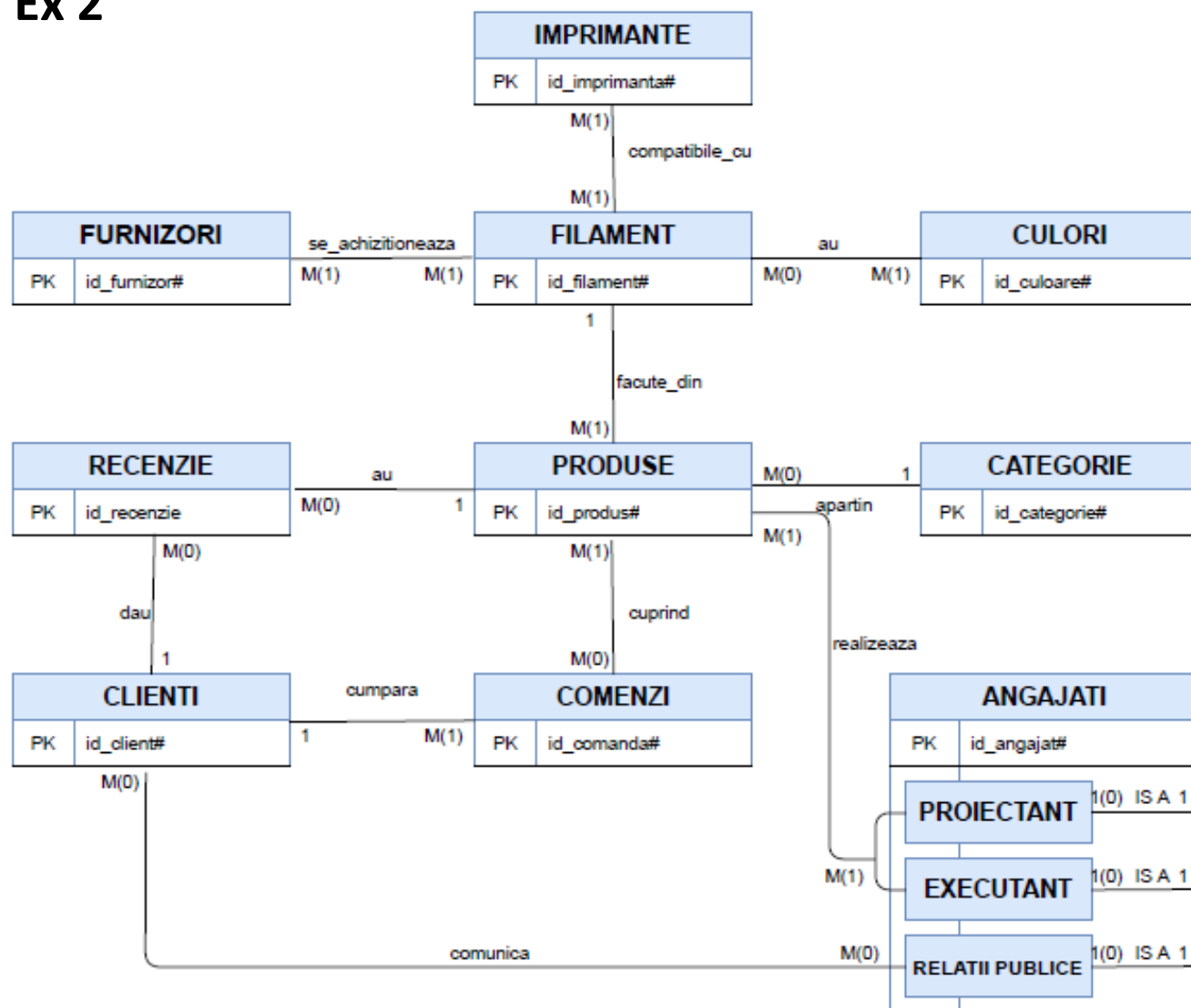
```
CREATE SEQUENCE NEXT_ID_FURNIZOR  
INCREMENT by 1  
START WITH 100  
MAXVALUE 99999  
NOCYCLE;
```

```
INSERT INTO FURNIZOR VALUES (NEXT_ID_FURNIZOR.NEXTVAL, 'EMAG', '0728282106', 'contact@emag.ro');  
INSERT INTO FURNIZOR VALUES (NEXT_ID_FURNIZOR.NEXTVAL, 'OptimusDigital', '0728282107', 'contact@optidigi.ro');  
INSERT INTO FURNIZOR VALUES (NEXT_ID_FURNIZOR.NEXTVAL, 'HobbyMarket', '0721516130', 'contact@hobby.ro');  
INSERT INTO FURNIZOR VALUES (NEXT_ID_FURNIZOR.NEXTVAL, 'Printam3D', '0721216130', 'contact@3D.ro');  
INSERT INTO FURNIZOR VALUES (NEXT_ID_FURNIZOR.NEXTVAL, 'CEL', '0372245334', 'contact@cel.ro');  
INSERT INTO FURNIZOR VALUES (NEXT_ID_FURNIZOR.NEXTVAL, 'ArduShop', '0727387467', 'contact@ardu.ro');
```

```
CREATE SEQUENCE NEXT_ID_IMPRIMANTE  
INCREMENT by 10  
START WITH 10  
MAXVALUE 99999  
NOCYCLE;
```

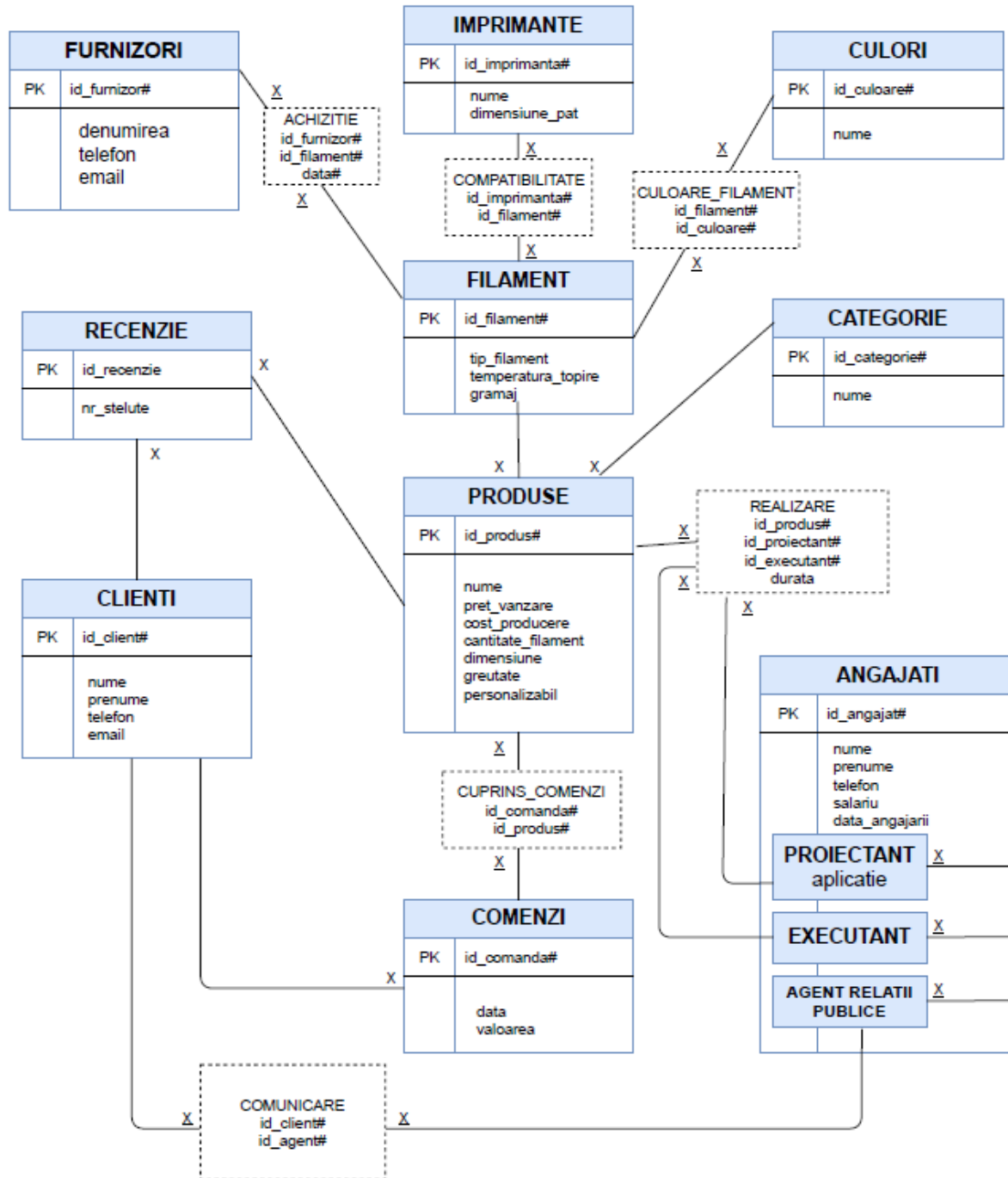
```
INSERT INTO IMPRIMANTE VALUES (NEXT_ID_IMPRIMANTE.NEXTVAL, 'Odysseuss', 625);  
INSERT INTO IMPRIMANTE VALUES (NEXT_ID_IMPRIMANTE.NEXTVAL, 'TevoTornado', 900);  
INSERT INTO IMPRIMANTE VALUES (NEXT_ID_IMPRIMANTE.NEXTVAL, 'Anycubic', 600);  
INSERT INTO IMPRIMANTE VALUES (NEXT_ID_IMPRIMANTE.NEXTVAL, 'Odysseuss', 1000);  
INSERT INTO IMPRIMANTE VALUES (NEXT_ID_IMPRIMANTE.NEXTVAL, 'Prusa', 900);  
INSERT INTO IMPRIMANTE VALUES (NEXT_ID_IMPRIMANTE.NEXTVAL, 'MakerBot', 2500);
```

-DIAGRAMA E/R-





## Ex 3



-DIAGRAMA CONCEPTUALA-



6. Folosind un subprogram stocat (o procedura) si 2 colectii, creati un tabel ‘info\_produce\_realizate’ cu urmatoarea structura: id\_angajat, lista produselor realizate (id), valoarea totala a acestora (pret total) si lista colegilor cu care au colaborat vreodata in realizarea unui produs (id). Pentru angajatii cu job-ul ‘agent’ se va completa cu NULL.

```

84      info_angajat(103); --NULL (este agent)
85      info_angajat(200);--No_data_found
86  END;
87
88  select * from info_produce_realizate;
89

```

ID_ANGAJAT	PRODUSE	PRET_TOTAL	COLABORARE
1	101 ALEXION2001.PRODUSE REALIZATE(127, 124, 126, 120, 121)	274	ALEXION2001.COLEGI('104 ', '102 ', '105 '
2	103 (null)	(null)	(null)

```

F;
_e=TRUE THEN colegi_aux.exter
colegi_aux(colegi_a
F;

```

Nu exista acest angajat

```
CREATE OR REPLACE TYPE colegi AS VARRAY(100) OF CHAR(5);
```

```
CREATE OR REPLACE TYPE produse_realizate IS TABLE OF NUMBER(5);
```

```
CREATE TABLE info_produce_realizate(id_angajat NUMBER(5),
```

```
    produse produse_realizate,
```

```
    pret_total NUMBER,
```

```
    colaborare colegi)
```

```
NESTED TABLE produse STORE AS produse_r;
```

```
-----
```

```
CREATE OR REPLACE PROCEDURE info_angajat
```

```
    (v_cod angajati.id_angajat%TYPE)
```

IS

```
pret_total number(5) :=0;
produse_aux produse_realizate := produse_realizate();
colegi_aux colegi := colegi();
job_aux angajati.job%TYPE;
ok_p BOOLEAN;
ok_e BOOLEAN;
cnt NUMBER(2);
```

BEGIN

```
SELECT job INTO job_aux
FROM angajati
WHERE id_angajat = v_cod;
```

```
IF job_aux = 'agent' THEN INSERT INTO info_produse_realizate VALUES (v_cod,
NULL,NULL,NULL);
```

ELSE

```
    pret_total :=0;
    FOR i IN (SELECT DISTINCT p.id_produs, p.pret_vanzare, a.id_angajat
              FROM PRODUSE p JOIN REALIZARE r ON (p.id_produs=r.id_produs)
              JOIN ANGAJATI a ON (a.id_angajat=r.id_executant OR
a.id_angajat=r.id_proiectant)
              WHERE id_angajat=v_cod) LOOP
```

```
        pret_total := pret_total + i.pret_vanzare;
```

```
        produse_aux.extend;
        produse_aux(produse_aux.last) := i.id_produs;
```

END LOOP;

FOR i IN (SELECT id\_produs  
FROM realize  
WHERE id\_executant=v\_cod OR id\_proiectant=v\_cod) LOOP

FOR j IN (SELECT id\_proiectant,id\_executant  
FROM REALIZARE  
WHERE id\_produs = i.id\_produs) LOOP

ok\_p := TRUE;

ok\_e := TRUE;

IF j.id\_proiectant = v\_cod THEN ok\_p :=FALSE;

ELSIF j.id\_executant = v\_cod THEN ok\_e :=FALSE;

END IF;

cnt := colegi\_aux.COUNT;

FOR k IN 1..cnt LOOP

IF j.id\_proiectant = colegi\_aux(k) THEN ok\_p :=FALSE;

ELSIF j.id\_executant = colegi\_aux(k) THEN ok\_e :=FALSE;

END IF;

END LOOP;

IF ok\_p=TRUE THEN colegi\_aux.extend;

```
        colegi_aux(colegi_aux.last):= j.id_proiectant;
    END IF;
    IF ok_e=TRUE THEN colegi_aux.extend;
        colegi_aux(colegi_aux.last):= j.id_executant;
    END IF;
END LOOP;

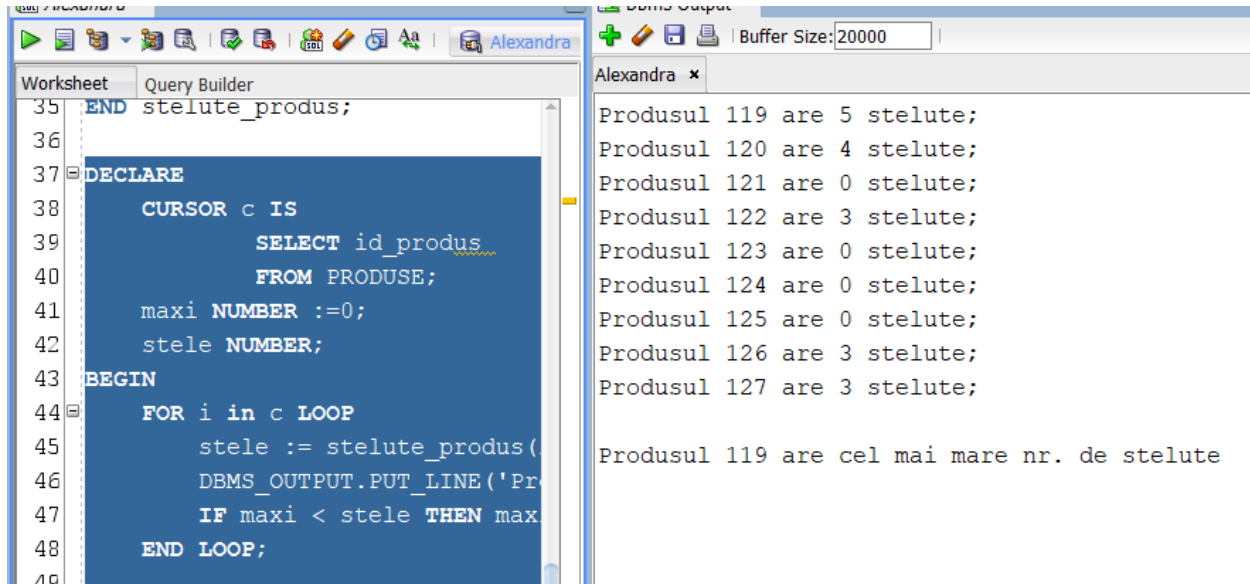
END LOOP;

INSERT INTO info_produse_realizate VALUES (v_cod, produse_aux, pret_total,
colegi_aux);
END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Nu exista acest angajat');
END info_angajat;

BEGIN
    info_angajat(101);
    info_angajat(103); --NULL (este agent)
    info_angajat(200);--No_data_found
END;

select * from info_produse_realizate;
```

7. Folosind un subprogram stocat (functie) si un cursor, afisati numarul total de stelute al produselor (0 daca nu au recenzii) si cele mai bine cotate produse (cu cele mai multe stelute). Numarul de stele ale unui produs este dat de media aritmetica dintre numarul de stelute lasate de catre clienti in recenzii.



The screenshot shows a database IDE with two panes. The left pane, titled 'Query Builder', contains the following PL/SQL code:

```

35 END stelute_produș;
36
37 DECLARE
38     CURSOR c IS
39         SELECT id_produș
40         FROM PRODUSE;
41     maxi NUMBER :=0;
42     stele NUMBER;
43 BEGIN
44     FOR i in c LOOP
45         stele := stelute_produș(
46             DBMS_OUTPUT.PUT_LINE('Pr
47             IF maxi < stele THEN maxi
48     END LOOP;
49

```

The right pane, titled 'Demo Output', shows the output of the function for products 119 through 127:

```

Produsul 119 are 5 stelute;
Produsul 120 are 4 stelute;
Produsul 121 are 0 stelute;
Produsul 122 are 3 stelute;
Produsul 123 are 0 stelute;
Produsul 124 are 0 stelute;
Produsul 125 are 0 stelute;
Produsul 126 are 3 stelute;
Produsul 127 are 3 stelute;

Produsul 119 are cel mai mare nr. de stelute

```

CREATE OR REPLACE FUNCTION stelute\_produș --subprogramul calculeaza numarul de stele total al unui produs

(v\_cod produse.id\_produș%TYPE)

RETURN NUMBER IS

stele\_total NUMBER:=0;

stele recenzie.nr\_stelute%TYPE;

nr NUMBER;

CURSOR c IS

SELECT r.nr\_stelute, CASE

WHEN r.id\_produș IN (SELECT id\_produș FROM RECENZIE) THEN  
(SELECT COUNT(\*) FROM RECENZIE re WHERE re.id\_produș = r.id\_produș)

ELSE 0

END As nr

FROM RECENZIE r FULL OUTER JOIN PRODUSE p ON (p.id\_produș = r.id\_produș)

WHERE r.id\_produș = v\_cod;

BEGIN

OPEN c;

LOOP FETCH c INTO stele, nr;

EXIT WHEN c%NOTFOUND;

IF stele = '\*' THEN stele\_total := stele\_total + 1;

ELSIF stele = '\*\*' THEN stele\_total := stele\_total + 2;

ELSIF stele = '\*\*\*' THEN stele\_total := stele\_total + 3;

ELSIF stele = '\*\*\*\*' THEN stele\_total := stele\_total + 4;

ELSE stele\_total := stele\_total + 5;

END IF;

END LOOP;

CLOSE c;

IF nr>0 THEN stele\_total := stele\_total/nr;

ELSE stele\_total := 0;

END IF;

RETURN stele\_total;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN DBMS\_OUTPUT.PUT\_LINE('Nu exista acest produs');

END stelute\_produs;

DECLARE

CURSOR c IS

SELECT id\_produs

FROM PRODUSE;

maxi NUMBER :=0;

```
    stele NUMBER;  
  
BEGIN  
  
    FOR i in c LOOP  
  
        stele := stelute_proodus(i.id_proodus);  
  
        DBMS_OUTPUT.PUT_LINE('Produsul '||i.id_proodus||' are '||stele||' stelute;');  
  
        IF maxi < stele THEN maxi :=stele; END IF;  
  
    END LOOP;  
  
  
    DBMS_OUTPUT.NEW_LINE;  
  
  
    FOR i in c LOOP  
  
        stele := stelute_proodus(i.id_proodus);  
  
        IF maxi = stele THEN DBMS_OUTPUT.PUT_LINE('Produsul '||i.id_proodus||' are cel mai  
mare nr. de stelute'); END IF;  
  
    END LOOP;  
  
END;
```

**8. Se da numele unei categorii si se cere sa se afiseze (utilizand o functie) codul si lista cu toate imprimantele complet compatibile si sa se returneze (si afiseze) numarul acestora. O imprimanta este complet compatibila daca cel putin un produs din acea categorie se poate printa pe imprimanta, luandu-se in considerare compatibilitatea filamentului si a dimensiunii produsului fata de cea a patului imprmantei.**

**Sa se ia in considerare cazurile in care nu exista nicio imprimanta compatibila sau nu se gaseste categoria.**

Obs. Nu se poate arunca exceptia TOO\_MANY\_ROWS, deoarece constrangerile tabelului CATEGORIE nu permit dublarea numelui sau a codului.



```

UT.PUT_LINE('Nu exista imp
imp.first..imp.last LOOP
PUT.PUT_LINE(imp(j));

imante;

```

```

Pentru categoria Home, cu codul 101, sunt complet compatibile urmatoarele imprimante:
30
20
70
Nr de imprimante complet compatibile este 3
Pentru categoria Diverse, cu codul 107, sunt complet compatibile urmatoarele imprimante:
Nu exista imprimante complet compatibile
Nr de imprimante complet compatibile este 0

```

```

50
51 BEGIN
52     DBMS_OUTPUT.PUT_LINE('Nr de imprimante complet compatibile este '|| imprima
53     DBMS_OUTPUT.PUT_LINE('Nr de imprimante complet compatibile este '|| imprima

```

Script Output x Query Result x

Task completed in 0.092 seconds

Error report -

```

ORA-20000: Nu exista o categorie cu acest nume!
ORA-06512: at "ALEXION2001.IMPRIMANTE_CATEGORIE", line 45
ORA-06512: at line 4
20000. 00000 - "%s"
*Cause:      The stored procedure 'raise_application_error'
              was called which causes this error to be generated.
*Action:     Correct the problem as described in the error message or contact
              the application administrator or DBA for more information.

```

CREATE OR REPLACE FUNCTION imprimante\_categorie

(v\_nume categorie.nume%TYPE)

RETURN NUMBER IS

numar\_imprimante NUMBER;

cod NUMBER;

TYPE imprimante IS TABLE OF NUMBER;

imp imprimante := imprimante();

CURSOR c IS

SELECT c.id\_categorie,c.nume, i.id\_imprimanta, i.dimensiune\_pat, p.dimensiune

FROM CATEGORIE c JOIN PRODUSE p ON (c.id\_categorie = p.id\_categorie)

JOIN FILAMENT f ON (f.id\_filament = p.id\_filament)

JOIN COMPATIBILITATE co ON (f.id\_filament = co.id\_filament)

```
        JOIN IMPRIMANTE i ON (i.id_imprimanta = co.id_imprimanta)

        WHERE c.num = v_num;

BEGIN

    numar_imprimante := 0;

    SELECT id_categorie INTO cod
    FROM CATEGORIE
    WHERE v_num = num;

    DBMS_OUTPUT.PUT_LINE('Pentru categoria ' || v_num || ', cu codul ' || cod || ', sunt complet
    compatibile urmatoarele imprimante:');

    FOR i IN c LOOP

        IF i.dimensiune < i.dimensiune_pat THEN

            imp.EXTEND();

            imp(imp.last):=i.id_imprimanta;

        END IF;

    END LOOP;

    numar_imprimante:= imp.count;

    IF numar_imprimante = 0 THEN

        DBMS_OUTPUT.PUT_LINE('Nu exista imprimante complet compatibile');

    ELSE

        FOR j IN imp.first..imp.last LOOP

            DBMS_OUTPUT.PUT_LINE(imp(j));

        END LOOP;

    END IF;
```

```
RETURN numar_imprimante;
```

```
EXCEPTION
```

```
WHEN NO_DATA_FOUND THEN
```

```
    RAISE_APPLICATION_ERROR(-20000, 'Nu exista o categorie cu acest nume!');
```

```
WHEN OTHERS THEN
```

```
    RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');
```

```
END imprimante_categorie;
```

```
-----
```

```
INSERT INTO CATEGORIE VALUES (107, 'Diverse');
```

```
BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE('Nr de imprimante complet compatibile este ' ||  
imprimante_categorie('Home')); -- corect. nr > 0
```

```
    DBMS_OUTPUT.PUT_LINE('Nr de imprimante complet compatibile este ' ||  
imprimante_categorie('Diverse')); --corect, nr = 0
```

```
    DBMS_OUTPUT.PUT_LINE('Nr de imprimante complet compatibile este ' ||  
imprimante_categorie('Food')); -- NO_DATA_FOUND
```

```
END;
```

**9. Se da o data si se cere sa se afiseze produsele cu o recenzie de minim 2 stelute, care pot fi produse din tipul de filament achizitionat la acea data. Se vor lua in considerare doar recenziile care au fost lasate de clienti care au comunicat macar o data cu un agent de vanzari. Sa se afiseze data si numele furnizorului de la care s-a cumparat in acea data filament. Sa se trateze cazurile TOO\_MANY\_ROWS, NO\_DATA\_FOUND si lista de produse goala.**

<pre> 1 BEGIN 2   data_produce('3-JAN-2020'); -- corect, produse &gt;0 3   data_produce('15-MAY-21'); -- corect, produse &lt;0 4   data_produce('30-JAN-2020'); -- NO_DATA_FOUND 5   data_produce('12-MAY-2020'); -- TOO_MANY_ROWS 6 7 END;</pre>	<pre> In data 03-JAN-20 s-a achizitionat filament de la furnizorul OptimusDigital Cutie labirint Suport chei Lampa  In data 15-MAY-21 s-a achizitionat filament de la furnizorul ArduShop Nu exista produse</pre>
---	---

```

6 data_produce('15-MAY-21'); -- corect, produse <0
7 data_produce('30-JAN-2020'); -- NO_DATA_FOUND
8 data_produce('12-MAY-2020'); -- TOO_MANY_ROWS
9
0 END;
1

```

Script Output x Query Result x

Task completed in 0.039 seconds

```

D;
ror report -
A-20000: Nu exista aceasta data!
A-06512: at "ALEXION2001.DATA_PRODUCE", line 41
A-06512: at line 4
000. 00000 - "%s"
ause: The stored procedure 'raise_application_error'
       was called which causes this error to be generated.

```

```

16 data_produce('15-MAY-21'); -- corect, produse <0
17 --data_produce('30-JAN-2020'); -- NO_DATA_FOUND
18 data_produce('12-MAY-2020'); -- TOO_MANY_ROWS
19
20 END;
21

```

Script Output x Query Result x

Task completed in 0.052 seconds

```

VD;
ror report -
A-20000: Exista mai multe date la fel!
A-06512: at "ALEXION2001.DATA_PRODUCE", line 43
A-06512: at line 5
0000. 00000 - "%s"
ause: The stored procedure 'raise_application_error'
       was called which causes this error to be generated.

```

CREATE OR REPLACE PROCEDURE data\_produce

(v\_data achizitie.data%TYPE)

IS

v\_furnizor furnizor.denumirea%TYPE;

TYPE produse IS TABLE OF CHAR(25);

p produse := produse();

CURSOR c IS

SELECT fu.denumirea,a.data, p.num

FROM PRODUCE p JOIN FILAMENT f ON (f.id\_filament = p.id\_filament)

JOIN ACHIZITIE a ON (a.id\_filament = f.id\_filament)

JOIN FURNIZOR fu ON (fu.id\_furnizor = a.id\_furnizor)

JOIN RECENZIE r ON (r.id\_produs = p.id\_produs)

JOIN CLIENTI c ON (r.id\_client = c.id\_client)

WHERE r.nr\_stelute <> '\*' AND c.id\_client IN (SELECT id\_client FROM  
COMUNICARE) AND a.data = v\_data;

BEGIN

```
SELECT fu.denumirea INTO v_furnizor
FROM ACHIZITIE a JOIN FURNIZOR fu ON (fu.id_furnizor = a.id_furnizor)
WHERE v_data = a.data;

DBMS_OUTPUT.PUT_LINE('In data ||v_data|| s-a achizitionat filament de la furnizorul
'||v_furnizor);

FOR i IN c LOOP
    p.EXTEND();
    p(p.last):=i.nume;
END LOOP;

IF p.COUNT = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Nu exista produse');
ELSE
    FOR j IN p.first..p.last LOOP
        DBMS_OUTPUT.PUT_LINE(p(j));
    END LOOP;
    DBMS_OUTPUT.NEW_LINE;
END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista aceasta data!');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20000, 'Exista mai multe date la fel!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');

END data_produce;
```

-----

```
INSERT INTO FILAMENT VALUES (107, 'TPU', NULL, NULL);
```

```
INSERT INTO ACHIZITIE VALUES ('15-MAY-21',106,107);
```

```
BEGIN
```

```
    data_produce('3-JAN-2020'); -- corect, produse >0
```

```
    data_produce('15-MAY-21'); -- corect, produse <0
```

```
    data_produce('30-JAN-2020'); -- NO_DATA_FOUND
```

```
    data_produce('12-MAY-2020'); -- TOO_MANY_ROWS
```

```
END;
```

**10. Definiți un trigger de tip LMD la nivel de comanda care sa se declanseze la fiecare inserare sau update asupra tabelul ACHIZITIE. La fiecare inserare să apeleze subprogramul data\_produce Se va afisa de fiecare data utilizatorul care a facut operatia si numarul actual de achizitii facute la fiecare furnizor.**

Query Builder	Alexandra x
<pre> OPEN c_numar; LOOP FETCH c_numar INTO v_nume, v_cursor; EXIT WHEN c_numar%NOTFOUND; DBMS_OUTPUT.PUT_LINE ('Furnizorul '  v_nume); LOOP FETCH v_cursor INTO v_nr; EXIT WHEN v_cursor%NOTFOUND; DBMS_OUTPUT.PUT_LINE (v_nr); END LOOP; END LOOP; CLOSE c_numar; END; / -- declansare trigger UPDATE ACHIZITIE SET id_furnizor = 105 WHERE id_furnizor = '106'; INSERT INTO ACHIZITIE VALUES(SYSDATE, 101, 106 ); </pre>	<pre> ALEXION2001 a adaugat o noua achizitie ! In data 02-JAN-22 s-a achizitionat filament de la furnizorul EMAG Suport cu incarcator  Numar achizitii: Furnizorul EMAG 3 Furnizorul OptimusDigital 3 Furnizorul HobbyMarket 2 Furnizorul Printam3D 1 Furnizorul CEL 1 Furnizorul ArduShop 2 </pre>

Query Builder	Alexandra x
<pre> 33 OPEN c_numar; 34 LOOP FETCH c_numar INTO v_nume, v_cursor; 35 EXIT WHEN c_numar%NOTFOUND; 36 DBMS_OUTPUT.PUT_LINE ('Furnizorul '  v_nume); 37 LOOP FETCH v_cursor INTO v_nr; 38 EXIT WHEN v_cursor%NOTFOUND; 39 DBMS_OUTPUT.PUT_LINE (v_nr); 40 END LOOP; 41 END LOOP; 42 CLOSE c_numar; 43 END; 44 / 45 -- declansare trigger 46 INSERT INTO ACHIZITIE VALUES(SYSDATE, 101, 106 ); 47 UPDATE ACHIZITIE SET id_furnizor = 105 WHERE id_furnizor = '106'; 48 </pre>	<pre> ALEXION2001 a modificat(update) tabelul achizitie! Numar achizitii: Furnizorul EMAG 3 Furnizorul OptimusDigital 3 Furnizorul HobbyMarket 2 Furnizorul Printam3D 1 Furnizorul CEL 3 Furnizorul ArduShop 0 </pre>

```
CREATE OR REPLACE TRIGGER trig_achizitie
  AFTER INSERT OR UPDATE ON ACHIZITIE
DECLARE
  TYPE refcursor IS REF CURSOR;
  CURSOR c_numar IS
    SELECT denumirea,
           CURSOR (SELECT COUNT(*)
                   FROM achizitie a
                   WHERE a.id_furnizor = f.id_furnizor)
    FROM furnizor f;
  v_nume furnizor.denumirea%TYPE;
  v_cursor refcursor;
  v_nr NUMBER;
BEGIN
  IF INSERTING THEN
    DBMS_OUTPUT.PUT_LINE(USER|| ' a adaugat o noua achizitie !');
    data_produce(SYSDATE);
  ELSE
    DBMS_OUTPUT.PUT_LINE(USER|| ' a modificat(update) tabelul achizitie!');
  END IF;
  DBMS_OUTPUT.PUT_LINE('Numar achizitii:');

  OPEN c_numar;
  LOOP FETCH c_numar INTO v_nume, v_cursor;
  EXIT WHEN c_numar%NOTFOUND;

  DBMS_OUTPUT.PUT_LINE ('Furnizorul '||v_nume);
  LOOP FETCH v_cursor INTO v_nr;
  EXIT WHEN v_cursor%NOTFOUND;
```



```

DBMS_OUTPUT.PUT_LINE (v_nr);

END LOOP;

END LOOP;

CLOSE c_numar;

END;

/

-- declansare trigger

INSERT INTO ACHIZITIE VALUES(SYSDATE, 101, 106 );

UPDATE ACHIZITIE SET id_furnizor = 105 WHERE id_furnizor = '106';

```

**11. Definiți un trigger de tip LMD la nivel de linie care sa se declanseze la fiecare inserare sau update asupra tabelul ANGAJATI. La fiecare inserare a unui agent să se adauge angajatul in tabelul info\_produce\_realizate. Daca se face o marire de salariu pentru un angajat cu job-ul proiectant sau executant, acesta trebuie sa fi contribuit la minim un produs cu rating maxim (5 stelute – se apeleaza subprogramul stelute\_produ).**

```

31 UPDATE angajati SET salariu = salariu;
32 UPDATE angajati SET salariu = salariu;
33
34 select * from info_produce_realizate v
35 rollback;

```

ID_ANGAJAT	PRODUCE	PRET_TOTAL	COLABORARE
1	200	(null)	(null)

```

31 UPDATE angajati SET sa
32 UPDATE angajati SET sa

```

ID_ANGAJAT	SALARIU
1	102
	1800



```

28 /
29
30 INSERT INTO ANGAJATI VALUES (200,
31 UPDATE angajati SET salariu = sal
32 UPDATE angajati SET salariu = sal

```

ID_ANGAJAT	SALARIU
1	102
	1980

```

24         END IF;
25     END IF;
26 END IF;
27 END;
28 /

```

Script Output x Query Result x

Task completed in 0.04 seconds

Error starting at line : 32 in command -  
UPDATE angajati SET salariu = salariu\*1.1 WHERE id\_angajat = '105'  
Error report -  
ORA-20001: Nu se poate marii salariul - niciun produs nu are rating maxim  
ORA-06512: at "ALEXION2001.TRIG\_ANGAJATI", line 20  
ORA-04088: error during execution of trigger 'ALEXION2001.TRIG\_ANGAJATI'

CREATE OR REPLACE TRIGGER trig\_angajati

BEFORE UPDATE OR INSERT ON ANGAJATI

FOR EACH ROW

DECLARE

ok number :=0;

BEGIN

IF INSERTING THEN

IF (:NEW.job = 'agent') THEN

INSERT INTO info\_produce\_realizate VALUES (:NEW.id\_angajat,NULL,NULL,NULL);

END IF;

ELSIF UPDATING THEN

IF (:NEW.job <> 'agent') AND (:NEW.salariu > :OLD.salariu) THEN

FOR i IN (SELECT DISTINCT r.id\_produ

FROM REALIZARE r JOIN PRODUSE p ON (r.id\_produ

JOIN RECENZIE rec ON (rec.id\_produ

WHERE r.id\_proiectant = :OLD.id\_angajat OR r.id\_executant = :OLD.id\_angajat)

LOOP

IF stelute\_produ

END IF;

```

END LOOP;

IF ok=0 THEN

    RAISE_APPLICATION_ERROR(-20001,'Nu se poate marii salariul - niciun produs nu
are rating maxim');

    END IF;

END IF;

END IF;

END;

/

```

```

INSERT INTO ANGAJATI VALUES (200,
'Ton','Ton','agent','0760606060',3000,SYSDATE,NULL);--merge

UPDATE angajati SET salariu = salariu*1.1 WHERE id_angajat = '102'; --merge

UPDATE angajati SET salariu = salariu*1.1 WHERE id_angajat = '105'; --eroare din trigger

```

**12. Definiți un trigger LDD care sa se declanseze la orice comanda la nivel de schema (DELETE, ALTER, CREATE). Modificarile se pot face doar de catre admini (utilizatorul SYS), Toate actiunile facute se vor stoca in tabelul admin\_baza (data la care a fost facuta comanda, tabelul modificat si tipul comenzii).**

```

19 CREATE TABLE tabel_test (numar NUMBER);
20 ALTER TABLE tabel_test ADD nume VARCHAR(10);
21 DROP TABLE tabel_test;
22

```

	DATA	TABEL_AFECTAT	TIPUL_COMENZII
1	03-JAN-22 12.13.24.440000000 AM	TABEL TEST	CREATE
2	03-JAN-22 12.13.32.092000000 AM	TABEL TEST	ALTER
3	03-JAN-22 12.14.03.205000000 AM	TABEL TEST	DROP

```

19 CREATE TABLE tabel_test(numar NUMBER);
20 ALTER TABLE tabel_test ADD aux VARCHAR(10);

```

Script Output x Query Result x

Task completed in 0.04 seconds

CREATE TABLE tabel\_test (numar NUMBER);

Error report -

ORA-00604: error occurred at recursive SQL level 1

ORA-20010: Utilizatorul ALEXION2001 nu este admin !

```

CREATE TABLE admin_baza(data TIMESTAMP(3),
                           tabel_afectat VARCHAR2(50),
                           tipul_comenzii VARCHAR2(30));

```

```

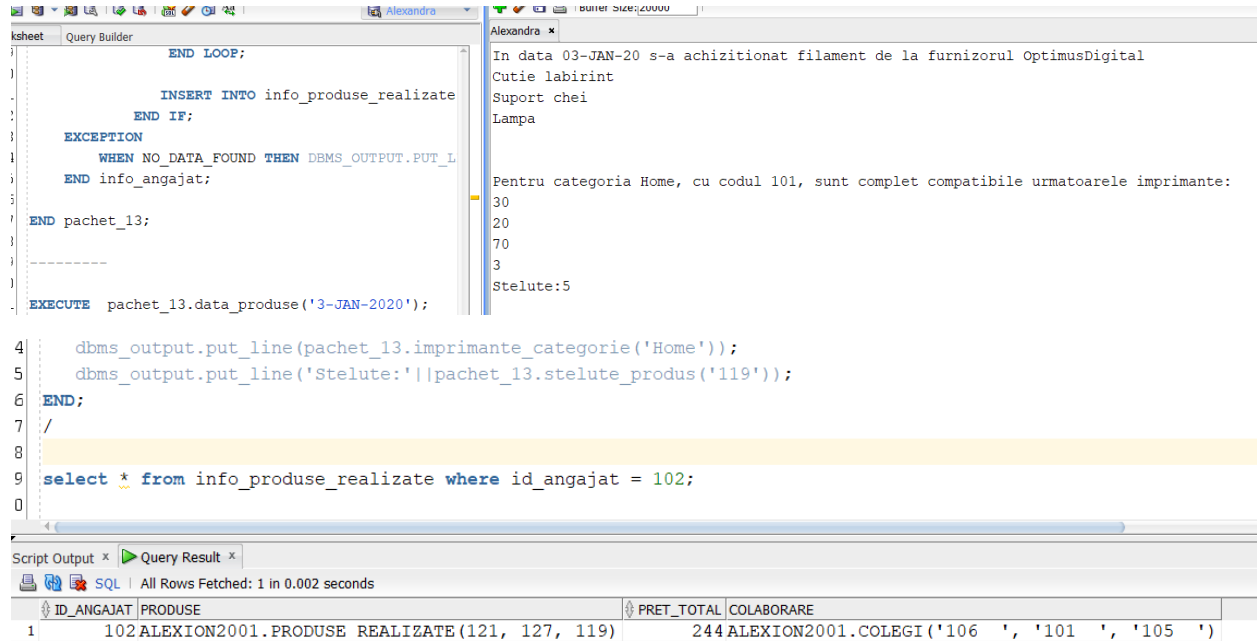
CREATE OR REPLACE TRIGGER trig_admin --trigger LDD
AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
    IF USER <> 'ALEXION2001' THEN
        RAISE_APPLICATION_ERROR(-20010, 'Nu sunteti admin!');
    ELSE
        INSERT INTO admin_baza VALUES
        (SYSTIMESTAMP(3),SYS.DICTIONARY_OBJ_NAME ,SYS.SYSEVENT);
    END IF;
END;
/

CREATE TABLE tabel_test(numar NUMBER);
ALTER TABLE tabel_test ADD nume VARCHAR(10);
DROP TABLE tabel_test;

select * from admin_baza;

```

### 13. Pachetul definit contine cele 4 subprograme stocate de la ex 6-9



The screenshot shows a SQL Developer window with a PL/SQL package definition in the 'Query Builder' tab. The package is named 'pachet\_13' and contains four subprograms: 'imprimante\_categorie', 'stelute\_produs', 'data\_produse', and 'info\_angajat'. The package is executed with the command 'EXECUTE pachet\_13.data\_produse('3-JAN-2020');'. The 'Query Result' tab shows the output of the execution, which is a table with columns 'ID\_ANGAJAT', 'PRODUSE', 'PRET\_TOTAL', and 'COLABORARE'. The table contains one row of data for employee 102.

```

END LOOP;

        INSERT INTO info_produce_realizate
        END IF;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_L
    END info_angajat;
END pachet_13;

EXECUTE pachet_13.data_produse('3-JAN-2020');

4      dbms_output.put_line(pachet_13.imprimante_categorie('Home'));
5      dbms_output.put_line('Stelute:'||pachet_13.stelute_produs('119'));
6  END;
7  /

9  select * from info_produce_realizate where id_angajat = 102;
10

```

ID_ANGAJAT	PRODUSE	PRET_TOTAL	COLABORARE
102	ALEXION2001.PRODUSE REALIZATE (121, 127, 119)	244	ALEXION2001.COLEGI ('106 ', '101 ', '105 ')

CREATE OR REPLACE PACKAGE pachet\_13 AS

FUNCTION imprimante\_categorie (v\_nume categorie.nume%TYPE)

RETURN NUMBER;

FUNCTION stelute\_produs (v\_cod produse.id\_produs%TYPE)

RETURN NUMBER;

PROCEDURE data\_produse(v\_data achizitie.data%TYPE);

PROCEDURE info\_angajat(v\_cod angajati.id\_angajat%TYPE);

End pachet\_13;

/

CREATE OR REPLACE PACKAGE BODY pachet\_13 AS

FUNCTION imprimante\_categorie

(v\_nume categorie.nume%TYPE)

RETURN NUMBER IS

numar\_imprimante NUMBER;

```
cod NUMBER;
```

```
TYPE imprimante IS TABLE OF NUMBER;
```

```
imp imprimante := imprimante();
```

```
CURSOR c IS
```

```
    SELECT c.id_categorie,c.numa, i.id_imprimanta, i.dimensiune_pat, p.dimensiune
```

```
    FROM CATEGORIE c JOIN PRODUSE p ON (c.id_categorie = p.id_categorie)
```

```
            JOIN FILAMENT f ON (f.id_filament = p.id_filament)
```

```
            JOIN COMPATIBILITATE co ON (f.id_filament = co.id_filament)
```

```
            JOIN IMPRIMANTE i ON (i.id_imprimanta = co.id_imprimanta)
```

```
    WHERE c.numa = v_numa;
```

```
BEGIN
```

```
    numar_imprimante := 0;
```

```
    SELECT id_categorie INTO cod
```

```
    FROM CATEGORIE
```

```
    WHERE v_numa = numa;
```

```
    DBMS_OUTPUT.PUT_LINE('Pentru categoria '||v_numa||', cu codul '||cod||', sunt complet  
compatibile urmatoarele imprimante:');
```

```
    FOR i IN c LOOP
```

```
        IF i.dimensiune < i.dimensiune_pat THEN
```

```
            imp.EXTEND();
```

```
            imp(imp.last):=i.id_imprimanta;
```

```
        END IF;
```

```
    END LOOP;
```

```
    numar_imprimante:= imp.count;
```

```
IF numar_imprimante = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Nu exista imprimante complet compatibile');
ELSE
    FOR j IN imp.first..imp.last LOOP
        DBMS_OUTPUT.PUT_LINE(imp(j));
    END LOOP;
END IF;

RETURN numar_imprimante;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista o categorie cu acest
nume!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');
END imprimante_categorie;

FUNCTION stelute_produs
(v_cod produse.id_produs%TYPE)
RETURN NUMBER IS
    stele_total NUMBER:=0;
    stele_recenzie.nr_stelute%TYPE;
    nr NUMBER;
    CURSOR c IS
        SELECT r.nr_stelute, CASE
            WHEN r.id_produs IN (SELECT id_produs FROM RECENZIE)
            THEN (SELECT COUNT(*) FROM RECENZIE re WHERE re.id_produs = r.id_produs)
            ELSE 0
```



```
END As nr

FROM RECENZIE r FULL OUTER JOIN PRODUSE p ON (p.id_produs =
r.id_produs)

WHERE r.id_produs = v_cod;

BEGIN

OPEN c;

LOOP FETCH c INTO stele, nr;

EXIT WHEN c%NOTFOUND;


IF stele = '*' THEN stele_total := stele_total + 1;
ELSIF stele = '**' THEN stele_total := stele_total + 2;
ELSIF stele = '***' THEN stele_total := stele_total + 3;
ELSIF stele = '****' THEN stele_total := stele_total + 4;
ELSE stele_total := stele_total + 5;
END IF;


END LOOP;

CLOSE c;

IF nr>0 THEN stele_total := stele_total/nr;
ELSE stele_total := 0;
END IF;


RETURN stele_total;

EXCEPTION

WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Nu exista acest
produs');

END stelute_produs;


PROCEDURE data_produce
```

```
(v_data achizitie.data%TYPE)
```

```
IS
```

```
v_furnizor furnizor.denumirea%TYPE;
```

```
TYPE produse IS TABLE OF CHAR(25);
```

```
p produse := produse();
```

```
CURSOR c IS
```

```
SELECT fu.denumirea,a.data, p.num
```

```
FROM PRODUSE p JOIN FILAMENT f ON (f.id_filament = p.id_filament)
```

```
JOIN ACHIZITIE a ON (a.id_filament = f.id_filament)
```

```
JOIN FURNIZOR fu ON (fu.id_furnizor = a.id_furnizor)
```

```
JOIN RECENZIE r ON (r.id_produs = p.id_produs)
```

```
JOIN CLIENTI c ON (r.id_client = c.id_client)
```

```
WHERE r.nr_stelute <> '*' AND c.id_client IN (SELECT id_client FROM  
COMUNICARE) AND a.data = v_data;
```

```
BEGIN
```

```
SELECT fu.denumirea INTO v_furnizor
```

```
FROM ACHIZITIE a JOIN FURNIZOR fu ON (fu.id_furnizor = a.id_furnizor)
```

```
WHERE v_data = a.data;
```

```
DBMS_OUTPUT.PUT_LINE('In data '||v_data||' s-a achizitionat filament de la furnizorul  
'||v_furnizor);
```

```
FOR i IN c LOOP
```

```
p.EXTEND();
```

```
p(p.last):=i.num;
```

```
END LOOP;
```

```
IF p.COUNT = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Nu exista produse');
ELSE
    FOR j IN p.first..p.last LOOP
        DBMS_OUTPUT.PUT_LINE(p(j));
    END LOOP;
    DBMS_OUTPUT.NEW_LINE;
END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu exista aceasta data!');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20000, 'Exista mai multe date la fel!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');
END data_produce;

PROCEDURE info_angajat
(v_cod angajati.id_angajat%TYPE)
IS
    pret_total number(5) :=0;
    produse_aux produse_realizate := produse_realizate();
    colegi_aux colegi := colegi();
    job_aux angajati.job%TYPE;
    ok_p BOOLEAN;
    ok_e BOOLEAN;
```

```
cnt NUMBER(2);

BEGIN

SELECT job INTO job_aux
FROM angajati
WHERE id_angajat = v_cod;

IF job_aux = 'agent' THEN INSERT INTO info_produce_realizate VALUES (v_cod,
NULL,NULL,NULL);
ELSE

    pret_total :=0;
    FOR i IN (SELECT DISTINCT p.id_produc, p.pret_vanzare, a.id_angajat
              FROM PRODUSE p JOIN REALIZARE r ON (p.id_produc=r.id_produc)
              JOIN ANGAJATI a ON (a.id_angajat=r.id_executant OR
a.id_angajat=r.id_proiectant)
              WHERE id_angajat=v_cod) LOOP

        pret_total := pret_total + i.pret_vanzare;

        produse_aux.extend;
        produse_aux(produse_aux.last) := i.id_produc;

    END LOOP;

    FOR i IN (SELECT id_produc
              FROM realizare
              WHERE id_executant=v_cod OR id_proiectant=v_cod) LOOP
```

```
FOR j IN (SELECT id_proiectant,id_executant
          FROM REALIZARE
          WHERE id_produs = i.id_produs) LOOP

    ok_p := TRUE;
    ok_e := TRUE;

    IF j.id_proiectant = v_cod THEN ok_p :=FALSE;
    ELSIF j.id_executant = v_cod THEN ok_e :=FALSE;
    END IF;

    cnt := colegi_aux.COUNT;
    FOR k IN 1..cnt LOOP
        IF j.id_proiectant = colegi_aux(k) THEN ok_p :=FALSE;
        ELSIF j.id_executant = colegi_aux(k) THEN ok_e :=FALSE;
        END IF;
    END LOOP;

    IF ok_p=TRUE THEN colegi_aux.extend;
        colegi_aux(colegi_aux.last):= j.id_proiectant;
    END IF;
    IF ok_e=TRUE THEN colegi_aux.extend;
        colegi_aux(colegi_aux.last):= j.id_executant;
    END IF;
END LOOP;
```

```

END LOOP;

INSERT INTO info_produce_realizate VALUES (v_cod, produse_aux, pret_total,
colegi_aux);

END IF;

EXCEPTION

WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Nu exista acest
angajat');

END info_angajat;

END pachet_13;

-----

EXECUTE pachet_13.data_produce('3-JAN-2020');

EXECUTE pachet_13.info_angajat('102');

BEGIN

dbms_output.put_line(pachet_13.imprimante_categorie('Home'));

dbms_output.put_line('Stelute:'||pachet_13.stelute_produs('119'));

END;

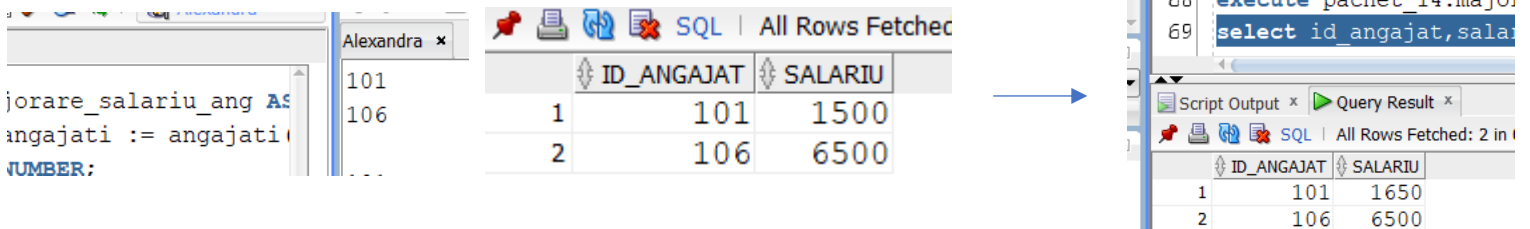
/

select * from info_produce_realizate where id_angajat = 102;

```

**14. Sa se defineasca un pachet care sa contina urmatoarele obiecte:**

- O functie care sa returneze lista cu codurile angajatilor care au muncit cel mai mult (au cel mai mare numar de produse realizate)
- O procedura care sa apeleze functia anterioara si sa mareasca cu 10% salariul acestor angajati daca acestia au salariul minim
- O functie care sa returneze lista cu codurile clientilor care au cumparat mai mult de 2 produse
- O procedura care sa apeleze functia anterioara si sa emita un voucher in valoare de 15% din valoarea totala a comenzilor plasate de acesti clienti.



The screenshot shows a SQL Developer window with a query result table and a script execution window. The query result table has columns ID\_ANGAJAT and SALARIU, with rows (1, 1500) and (2, 6500). The script execution window shows the execution of a script that updates the salary of employees with ID 101 and 106 by 10%.

ID_ANGAJAT	SALARIU
1	1500
2	6500

Script Output:

```

68 execute pachet_14.majorare_salariu_angajati(101);
69 select id_angajat, salariu from info_angajati where id_angajat = 101;

```

Query Result:

ID_ANGAJAT	SALARIU
1	1650
2	6500

Package Body	Alexandra x
<pre> clienti := clienti(); = NUMBER(5); </pre>	<pre> Clientul 101 a primit un voucher in valoare de 24.3 RON Clientul 106 a primit un voucher in valoare de 29.1 RON </pre>

```

CREATE OR REPLACE PACKAGE pachet_14 IS
    TYPE clienti IS TABLE OF NUMBER(4);
    TYPE angajati IS VARRAY(100) OF NUMBER(4);

    FUNCTION ang_max_produce RETURN angajati;
    PROCEDURE majorare_salariu_ang;

    FUNCTION clienti_max_produce RETURN clienti;
    PROCEDURE voucher;

END pachet_14;

```

```

CREATE OR REPLACE PACKAGE BODY pachet_14 IS

```

```

    FUNCTION ang_max_produce RETURN angajati IS
        id_ang angajati := angajati();
        maxim NUMBER;
        CURSOR c_max IS
            SELECT a.id_angajat,COUNT(DISTINCT r.id_produ) As nr
            FROM REALIZARE r JOIN ANGAJATI a ON (a.id_angajat=r.id_executant OR
            a.id_angajat=r.id_proiectant)
            GROUP BY a.id_angajat;
    BEGIN
        SELECT MAX(nr) INTO maxim
        FROM (SELECT a.id_angajat, COUNT(DISTINCT r.id_produ) As nr

```



```
FROM REALIZARE r JOIN ANGAJATI a ON (a.id_angajat=r.id_executant OR  
a.id_angajat=r.id_proiectant)
```

```
GROUP BY a.id_angajat);
```

```
FOR i in c_max LOOP
```

```
  IF i.nr = maxim THEN
```

```
    id_ang.EXTEND;
```

```
    id_ang(id_ang.LAST):=i.id_angajat;
```

```
  END IF;
```

```
END LOOP;
```

```
return id_ang;
```

```
END ang_max_produce;
```

```
PROCEDURE majorare_salariu_ang AS
```

```
  id_ang angajati := angajati();
```

```
  minim NUMBER;
```

```
  sal NUMBER(5);
```

```
BEGIN
```

```
  id_ang := ang_max_produce;
```

```
  SELECT MIN(salariu) INTO minim
```

```
  FROM angajati;
```

```
  FOR i In id_ang.FIRST..id_ang.LAST LOOP
```

```
    SELECT salariu INTO sal
```

```
    FROM angajati
```

```
    WHERE id_angajat = id_ang(i);
```

```
IF (minim = sal) THEN
    UPDATE angajati
    SET salariu = salariu * 1.1
    WHERE id_angajat = id_ang(i);
END IF;

END LOOP;

END;

FUNCTION clienti_max_produce RETURN clienti IS
    id_cl clienti := clienti();
    CURSOR c IS
        SELECT c.id_client, COUNT(cc.id_produs) As nr
        FROM clienti c JOIN Comenzi co ON (c.id_client=co.id_client)
            JOIN Cuprins_Comenzi cc ON (co.id_comanda=cc.id_comanda)
        GROUP BY c.id_client;
    BEGIN
        FOR i in c LOOP
            IF i.nr > 2 THEN
                id_cl.EXTEND;
                id_cl(id_cl.LAST):=i.id_client;
            END IF;
        END LOOP;
        return id_cl;
    END clienti_max_produce;

PROCEDURE voucher AS
    id_cl clienti := clienti();
```

```
valoare NUMBER(5);

BEGIN

    id_cl := clienti_max_produce;

    FOR i In id_cl.FIRST..id_cl.LAST LOOP

        SELECT SUM(valoarea) INTO valoare

        FROM Comenzi

        WHERE id_client = id_cl(i);

        dbms_output.put_line('Clientul '||id_cl(i)|| ' a primit un voucher in valoare de '||
        valoare*0.15||' RON');

    END LOOP;

END;

END pachet_14;

execute pachet_14.majorare_salariu_ang(); --a marit pt angajatul 101
select id_angajat,salariu from angajati where id_angajat=101 OR id_angajat=106;
execute pachet_14.voucher();
```