Proiect Sisteme de gestiune a bazelor de date

Tema: Administrarea bazei de date a unei firme din domeniul IT

Diță Alexandru

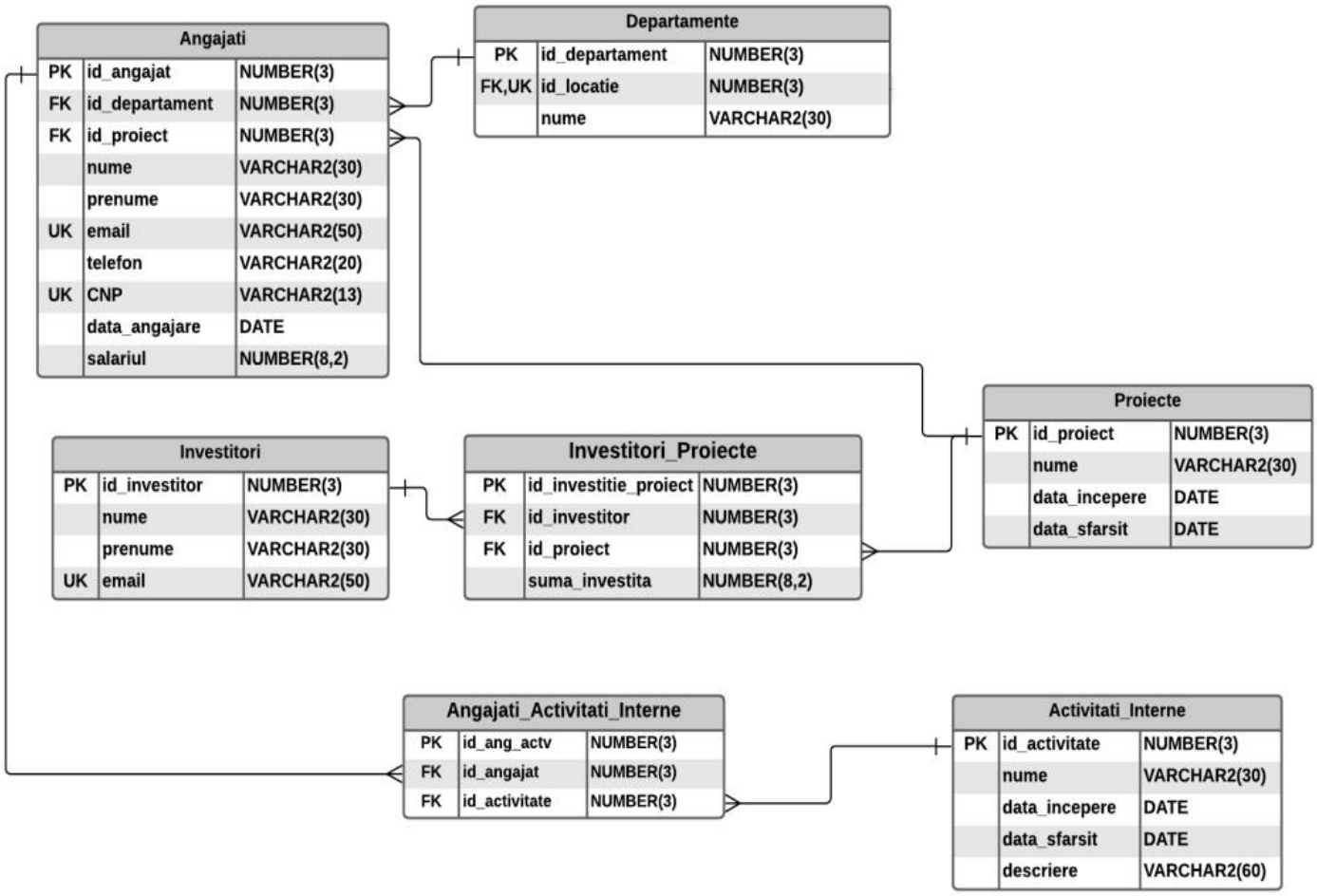
Seria C

Grupa 1051

Descrierea temei + Schema Conceptuală

Cum tehnologia avansează din ce în ce mai rapid și vine în sprijinul activităților economice, folosirea bazelor de date informatice de tip Oracle, MySQL, PostgreSQL, MariaDB, a devenit din ce în ce mai indispensabilă prin facilitarea operațiilor cu datele. Pentru vizualizarea datelor într-un mod cât mai facil, am folosit diferite functionalități pentru sortare, calculare de date numerice și diferite interogări ce ajută în acest sens.

Prin urmare, avem schema bazei de date și legăturile dintre tabele.



Legături de tip 1:M

• Legătura dintre tabela **Angajați** și **Departamente** se face prin intermediul coloanei **id_departament** (cheie primară în tabela **Departamente**), angajații făcând parte dintr-un singur departament.

• Legătura dintre tabela **Angajați** și **Proiecte** se face prin intermediul coloanei **id_proiect** (cheie primară în tabela **Proiecte**), angajații punându-și amprenta asupra unui singur proiect.

Legături de tip M:M

• Legătura dintre tabela **Investitori** și **Proiecte** se face cu ajutorul unei noi tabele **Investitori_Proiecte**. Astfel, cheile primare din cele două tabele (**id_investitor** și **id_proiect**) le adaugăm în tabela **Investitori_Proiecte**, de asemenea ele vor fi foreign keys.

• Legătura dintre tabela **Angajați** și **Activități_Interne** se face cu ajutorul unei noi tabele **Angajați_Activități_Interne**. Astfel, cheile primare din cele două tabele **(id_angajat** și **id_activitate** le adaugăm în tabela **Angajați_Activități_Interne**, de asemenea ele vor fi foreign keys.

Interacțiunea cu serverul Oracle prin intermediul comenzilor SQL (LDD SI LMD)

```
--1. Sa se mareasca salariul angajatului care are email-ul: "alex.dita2207@gmail.com", cu 10%
set serveroutput on
DECLARE
    V_SALARIUL ANGAJATI.SALARIUL%TYPE;
    V_MARIRE ANGAJATI.SALARIUL%TYPE;
    V_EMAIL ANGAJATI.EMAIL%TYPE;
BEGIN
    V_EMAIL := 'alex.dita2207@gmail.com';
SELECT SALARIUL INTO V_SALARIUL FROM ANGAJATI
WHERE email = 'alex.dita2207@gmail.com';
V_MARIRE := V_SALARIUL * 0.1;
EXECUTE IMMEDIATE
 'UPDATE ANGAJATI SET SALARIUL = SALARIUL + ' || TO_CHAR(V_MARIRE) || ' WHERE email = ''V_EMAIL''';
DBMS_OUTPUT.PUT_LINE('Salariul angajatului cu emailu alex.dita2207@gmail.com a fost marit cu ' || V_MARIRE || ' lei');
END;
/
```

```
-- 2. Sa se stearga proiectul cu id-ul 5 (Minds) deoarece acesta s-a amanat!
BEGIN
    EXECUTE IMMEDIATE
    'DELETE FROM PROIECTE WHERE ID_PROIECT = 5';
    DBMS_OUTPUT.PUT_LINE('Proiectul cu id-ul 5 a fost sters');
END;
/
```

```
-- 3. Sa se afiseze media salariilor angajatilor folosind PL/SQL
SET SERVEROUTPUT ON
DECLARE
    V_MEDIE ANGAJATI.SALARIUL%TYPE;
BEGIN
    SELECT AVG(SALARIUL) INTO V_MEDIE
    FROM ANGAJATI;
    DBMS_OUTPUT.PUT_LINE('Media salariul angajatilor este de ' || V_MEDIE || ' lei');
END;
/
```

```
-- 4. Sa se schimbe tipul coloanei ADRESA din tabela DEPARTAMENTE din varchar2(40)
-- in varchar2(50)

BEGIN
    EXECUTE IMMEDIATE
    'ALTER TABLE DEPARTAMENTE MODIFY ADRESA VARCHAR2(50)';
    DBMS_OUTPUT.PUT_LINE('S-a actualizat cu succes');
END;
/
```

```
--5. Sa se afiseze data pana la care ar fi valabil Proiectul cu ID 4 daca
-- acesta se va prelungi cu 6 luni
DECLARE
    V_PROIECT PROIECTE%ROWTYPE;
BEGIN
    SELECT * INTO V_PROIECT    FROM PROIECTE WHERE ID_PROIECT = 4;
    DBMS_OUTPUT.PUT_LINE ('Proiectul cu id-ul ' || TO_CHAR(V_PROIECT.ID_PROIECT)
    || ' se va termina pe data de ' || TO_CHAR(ADD_MONTHS(V_PROIECT.DATA_SFARSIT, 6))
    || ' daca se va prelungi cu 6 luni.');
END;
/
```

Structuri alternative și repetitive

```
--6. Sa se redica salariului angajatului cu id
-- 4 dupa urmatoarea schema: daca salariul este intre 1000 si 5000 de lei, se va
-- reduce cu 5%
-- daca salariul este intre 5000 si 10000 de lei se va reduce cu 10%, iar daca
-- este mai mare de 10000 lei se va reduce cu 15%

DECLARE
    V_SALARIUL NUMBER;
    V_REDUCERE VARCHAR2(10);
BEGIN
    SELECT SALARIUL INTO V_SALARIUL FROM ANGAJATI WHERE ID_ANGAJAT = 4;
    DBMS_OUTPUT.PUT_LINE('Salariul angajatului inainte de reducere = ' || V_SALARIUL);

    IF V_SALARIUL BETWEEN 0 AND 5000 THEN
        V_SALARIUL := V_SALARIUL * 0.95;
        V_REDUCERE := '5%';
    ELSIF V_SALARIUL BETWEEN 5000 AND 10000 THEN
        V_SALARIUL := V_SALARIUL * 0.9;
        V_REDUCERE := '10%';
    ELSE
        V_SALARIUL := V_SALARIUL *0.85;
        V_REDUCERE := '15%';
    END IF;
    DBMS_OUTPUT.PUT_LINE('Salariul angajatului dupa o reducere de ' || V_REDUCERE ||
    ' = ' || V_SALARIUL);
END;
/
```

```
--7. Sa se rezolve exercitiul anterior cu CASE

DECLARE
    V_SALARIUL NUMBER;
    V_REDUCERE VARCHAR2(10);
BEGIN
    SELECT SALARIUL INTO V_SALARIUL FROM ANGAJATI WHERE ID_ANGAJAT = 4;
    DBMS_OUTPUT.PUT_LINE('Salariul angajatului inainte de reducere = ' || V_SALARIUL);

    CASE WHEN V_SALARIUL BETWEEN 0 AND 5000 THEN
        V_SALARIUL := V_SALARIUL * 0.95;
        V_REDUCERE := '5%';
    WHEN V_SALARIUL BETWEEN 5000 AND 10000 THEN
        V_SALARIUL := V_SALARIUL * 0.9;
        V_REDUCERE := '10%';
    ELSE
        V_SALARIUL := V_SALARIUL * 0.85;
        V_REDUCERE := '15%';
    END CASE;
     DBMS_OUTPUT.PUT_LINE('Salariul angajatului dupa o reducere de ' || V_REDUCERE ||
    ' = ' || V_SALARIUL);
END;
/
```

```
--8. Sa se afiseze lista angajatilor cu id-ul cuprins intre 2 si 5, urmand ca apoi
-- sa se afiseze cei cu id-ul cuprins intre 4 si 7
DECLARE
    V_ANGAJAT ANGAJATI%ROWTYPE;
BEGIN
    FOR i IN 1..2 LOOP
        IF i = 1 THEN
            DBMS_OUTPUT.PUT_LINE('Angajatii cu id-ul cuprins intre 2 si 5');
            FOR i IN 2..5 LOOP
                SELECT ID_ANGAJAT, NUME, PRENUME INTO V_ANGAJAT.ID_ANGAJAT, V_ANGAJAT.NUME,
                V_ANGAJAT.PRENUME FROM ANGAJATI WHERE ID_ANGAJAT = i;
                DBMS_OUTPUT.PUT_LINE(V_ANGAJAT.ID_ANGAJAT || ' ' || V_ANGAJAT.NUME
                || ' ' || V_ANGAJAT.PRENUME);
                END LOOP;
        ELSIF i = 2 THEN
            DBMS_OUTPUT.PUT_LINE('Angajatii cu id-ul cuprins intre 4 si 7');
            FOR i IN 4..7 LOOP
                SELECT ID_ANGAJAT, NUME, PRENUME INTO V_ANGAJAT.ID_ANGAJAT, V_ANGAJAT.NUME,
                V_ANGAJAT.PRENUME FROM ANGAJATI WHERE ID_ANGAJAT = i;
                DBMS_OUTPUT.PUT_LINE(V_ANGAJAT.ID_ANGAJAT || ' ' || V_ANGAJAT.NUME
                || ' ' || V_ANGAJAT.PRENUME);
                END LOOP;
            END IF;
        END LOOP;
    END;
    /
```

```
--9. Sa se prelungeasca cu o luna deadline-ul tuturor proiectelor
DECLARE
    V_COUNT NUMBER;
    i NUMBER;
BEGIN
    SELECT COUNT(*) INTO V_COUNT FROM PROIECTE;
    i := 1;
    WHILE (i <= V_COUNT) LOOP
        UPDATE PROIECTE SET DATA_SFARSIT = ADD_MONTHS (DATA_SFARSIT, 1)
        WHERE ID_PROIECT = i;
        DBMS_OUTPUT.PUT_LINE('S-a prelungit cu o luna deadline-ul proiectului cu id-ul '
        || TO_CHAR(i));
        i := i + 1;
    END LOOP;
END;
/

--Exceptii
```

Tratarea excepțiilor

```
-- 10. Sa se trateze exceptia atunci cand se incearca impartirea la 0 a salariului
-- unui angajat
DECLARE
    A_SALARIUL ANGAJATI.SALARIUL%TYPE;
BEGIN
    UPDATE ANGAJATI SET SALARIUL = SALARIUL/0 WHERE id_angajat = 4;
EXCEPTION
    WHEN ZERO_DIVIDE THEN
    DBMS_OUTPUT.PUT_LINE('Exceptie! Impartirea la 0 nu se poate realiza!');
END;
```

```
-- 11. Sa se afiseze salariul angajatului cu emailu "alex.dita2207@gmail.com".
-- Daca acesta nu exista, se va tratata exceptia
DECLARE
    A_SALARIUL ANGAJATI.SALARIUL%TYPE;
BEGIN
    SELECT SALARIUL INTO A_SALARIUL FROM ANGAJATI WHERE email = 'alex.dita2207@gmail.com';
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Exceptie! Acest email nu exista in baza de date!');
END;
```

```
-- 12. Sa se mareasca cu 10% salariul angajatului cu CNP-ul 1991124015954.
-- Daca acest CNP nu exista, sa se trateze exceptia
DECLARE
    EX_FUNCTIE EXCEPTION;
BEGIN
    UPDATE ANGAJATI SET SALARIUL = SALARIUL * 0.1 WHERE CNP = '1991124015954';
    IF SQL%NOTFOUND THEN
        RAISE EX_FUNCTIE;
    END IF;
EXCEPTION
    WHEN EX_FUNCTIE THEN
        DBMS_OUTPUT.PUT_LINE('CNP-ul nu exista!');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Exceptie!');
END;
/
```

```
-- 13. Sa se incerce modificarea numarului de telefon al angajatului cu id 2.
-- Sa se trateze exceptia in care numarul de telefon nu respecta numarul de cifre
SET SERVEROUTPUT ON
DECLARE
    A_TELEFON NUMBER;
    EX_INVALID_TELEFON EXCEPTION;
    PRAGMA EXCEPTION_INIT(EX_INVALID_TELEFON, -20010);
BEGIN
    A_TELEFON := '07255239871231';
    IF(LENGTH(A_TELEFON) <> 10) THEN
        RAISE_APPLICATION_ERROR(-20010, 'Numar de telefon invalid!');
    ELSE
        UPDATE ANGAJATI SET TELEFON = A_TELEFON WHERE id_angajat = 2;
    END IF;
EXCEPTION
    WHEN EX_INVALID_TELEFON THEN
        DBMS_OUTPUT.PUT_LINE('Numar de telefon invalid');
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        DBMS_OUTPUT.PUT_LINE(SQLCODE);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Exceptie!');
END;
```

Gestionarea cursorilor

```
--14. Sa se mareasca salariul angajatului cu id-ul introdus de la tastatura.
--Daca update-ul nu se va realiza, sa se afisze un anunt in acest sens
ACCEPT a_id PROMPT 'Introduceti id-ul angajatului:'
DECLARE
    A_ID ANGAJATI.ID_ANGAJAT%TYPE := &a_id;
BEGIN
    UPDATE ANGAJATI SET SALARIUL = SALARIUL * 0.1 WHERE ID_ANGAJAT = A_ID;
    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista angajatul cu acest cod');
    END IF;
END;
```

```
--15. Sa se micsoreze cu 5% salariile peste 10000 lei si sa se afiseze
-- numarul salariilor micsorate
BEGIN
UPDATE ANGAJATI SET SALARIUL = SALARIUL * 0.95 WHERE SALARIUL > 10000;
DBMS_OUTPUT.PUT_LINE('S-au micsorat ' || TO_CHAR(SQL%ROWCOUNT) || ' salarii.');
END;
```

```
-- 16. Sa se afiseze angajatii care au salariul peste 10.000 de lei folosind un
-- cursor explicit
DECLARE
    CURSOR ANG_TOP_CURSOR IS SELECT ID_ANGAJAT, NUME, SALARIUL FROM ANGAJATI
WHERE SALARIUL > 10000;
    ANG angajati%ROWTYPE;

BEGIN
    DBMS_OUTPUT.PUT_LINE('Lista angajatilor cu salariul de peste 10.000 de lei');
    OPEN ANG_TOP_CURSOR;
    LOOP
        FETCH ANG_TOP_CURSOR INTO ANG.ID_ANGAJAT, ANG.NUME, ANG.SALARIUL;
        EXIT WHEN ANG_TOP_CURSOR%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Angajatul: ' || ANG.ID_ANGAJAT || ' ' || ANG.NUME ||
        ' Salariul ' || ANG.SALARIUL);
    END LOOP;
    CLOSE ANG_TOP_CURSOR;
END;
/
```

```
--17. Sa se afiseze datele INVESTITORULUI cu id-ul transmis prin parametru
-- printr-un cursor explicit
DECLARE
    CURSOR C_INVESTITOR (P_ID NUMBER) IS SELECT NUME, PRENUME, EMAIL
    FROM INVESTITORI WHERE ID_INVESTITOR = P_ID;
    REC_INVESTITOR C_INVESTITOR%ROWTYPE;
BEGIN
IF NOT C_INVESTITOR%ISOPEN THEN
    OPEN C_INVESTITOR(1);
END IF;

FETCH C_INVESTITOR INTO REC_INVESTITOR;
DBMS_OUTPUT.PUT_LINE('Investitorul gasit pentru id-ul dat ' || REC_INVESTITOR.NUME
|| ' ' || REC_INVESTITOR.PRENUME || ' ' || REC_INVESTITOR.EMAIL);
CLOSE C_INVESTITOR;
END;
/
```

Funcții, proceduri, includerea acestora în pachete

```
-- 18. Creati o functie care returneaza valoarea salariului unui angajat conform
-- id-ului acestuia primit ca parametru
CREATE OR REPLACE FUNCTION GET_SAL(P_ID ANGAJATI.ID_ANGAJAT%TYPE)
RETURN NUMBER IS
    V_SAL ANGAJATI.SALARIUL%TYPE := 0;
BEGIN
    SELECT SALARIUL INTO V_SAL FROM ANGAJATI WHERE ID_ANGAJAT = P_ID;
    RETURN V_SAL;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END GET_SAL;

DECLARE
    V_SAL ANGAJATI.SALARIUL%TYPE;
BEGIN
    V_SAL := GET_SAL(1);
    DBMS_OUTPUT.PUT_LINE('Salariul corespunzator angajatului cu id-ul specificat
    este : ' || V_SAL);
END;
```

```
-- 19. Sa se creeze o functie care verifica daca exista un anumit departament
-- in tabela cu DEPARTAMENTE primind drept parametru numele acestuia
CREATE OR REPLACE FUNCTION VALID_DEPARTAMENT (P_DEPARTAMENT DEPARTAMENTE.NUME%TYPE)
RETURN BOOLEAN IS
    V_ID DEPARTAMENTE.ID_DEPARTAMENT%TYPE;
BEGIN
    SELECT ID_DEPARTAMENT INTO V_ID FROM DEPARTAMENTE
    WHERE NUME = P_DEPARTAMENT;
    RETURN TRUE;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    RETURN FALSE;
END;

DECLARE
    V_VALID BOOLEAN;
    V_NUME_DEPARTAMENT DEPARTAMENTE.NUME%TYPE;
BEGIN
    V_NUME_DEPARTAMENT := UPPER('It');
    IF VALID_DEPARTAMENT (UPPER(V_NUME_DEPARTAMENT)) THEN
        DBMS_OUTPUT.PUT_LINE('Sponsorul ' || V_NUME_DEPARTAMENT || ' exista in
        tabela cu sponsori');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Sponsorul ' || V_NUME_DEPARTAMENT || ' nu exista in
        tabela cu departamente');
    END IF;
END;
```

```sql
-- 20. Sa se creeze o functie care returneaza suma totala a investitiilor
-- din tabela INVESTITORI_PROIECTE
CREATE OR REPLACE FUNCTION GET_SUMA_TOTALA_INVESTITII
RETURN NUMBER IS
    V_SUMA INVESTITORI_PROIECTE.SUMA_INVESTITA%TYPE;
BEGIN
    SELECT SUM(SUMA_INVESTITA) INTO V_SUMA FROM INVESTITORI_PROIECTE;
    RETURN V_SUMA;
END;

DECLARE
    V_SUMA INVESTITORI_PROIECTE.SUMA_INVESTITA%TYPE;
BEGIN
    V_SUMA := GET_SUMA_TOTALA_INVESTITII;
    DBMS_OUTPUT.PUT_LINE('Suma totala a investitiilor este de ' || V_SUMA || ' lei');
END;
```

```sql
-- 21. Creeaza o procedura care face acelasi lucru ca si functia de mai sus
CREATE OR REPLACE PROCEDURE SUMA_INVESTITII_TOTALA (P_SUMA OUT NUMBER) AS
BEGIN
    SELECT SUM(SUMA_INVESTITA) INTO P_SUMA FROM INVESTITORI_PROIECTE;
END;

DECLARE
    V_SUMA INVESTITORI_PROIECTE.SUMA_INVESTITA%TYPE;
BEGIN
    SUMA_INVESTITII_TOTALA(V_SUMA);
    DBMS_OUTPUT.PUT_LINE('Suma totala a investitiilor este de ' || V_SUMA || ' lei.');
END;
```

```sql
-- 22. Sa se creeze o procedura ce permite reducerea salariului unui anumit angajat
-- cu un anumit procent trimis ca si parametru
CREATE OR REPLACE PROCEDURE REDUCERE_SALARIU (P_ID IN ANGAJATI.ID_ANGAJAT%TYPE,
P_PROCENT IN FLOAT) AS
BEGIN
    IF P_PROCENT < 1 THEN
        UPDATE ANGAJATI SET SALARIUL = SALARIUL - (SALARIUL * P_PROCENT) WHERE
        ID_ANGAJAT = P_ID;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Procent invalid!');
    END IF;
END;

BEGIN
    REDUCERE_SALARIU(1, 0.1);
END;
```

```sql
-- 23. Sa se creeze un pachet care contine functiile si procedurile specifice
-- angajatilor
CREATE OR REPLACE PACKAGE ANG_PACKAGE AS
    FUNCTION GET_SAL(P_ID ANGAJATI.ID_ANGAJAT%TYPE) RETURN NUMBER;
    PROCEDURE REDUCERE_SALARIU(P_ID ANGAJATI.ID_ANGAJAT%TYPE, P_PROCENT FLOAT);
END ANG_PACKAGE;

CREATE OR REPLACE PACKAGE BODY ANG_PACKAGE AS
    FUNCTION GET_SAL(P_ID ANGAJATI.ID_ANGAJAT%TYPE)
    RETURN NUMBER IS
        V_SAL ANGAJATI.SALARIUL%TYPE := 0;
    BEGIN
        SELECT SALARIUL INTO V_SAL FROM ANGAJATI WHERE ID_ANGAJAT = P_ID;
        RETURN V_SAL;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN NULL;
    END;

    PROCEDURE REDUCERE_SALARIU(P_ID IN ANGAJATI.ID_ANGAJAT%TYPE, P_PROCENT IN FLOAT) A
        BEGIN
            IF P_PROCENT < 1 THEN
                UPDATE ANGAJATI SET SALARIUL = SALARIUL - (SALARIUL * P_PROCENT)
                WHERE ID_ANGAJAT = P_ID;
            ELSE
                DBMS_OUTPUT.PUT_LINE('Procent invalid!');
            END IF;
        END;
    END ANG_PACKAGE;

    DECLARE
        V_SAL ANGAJATI.SALARIUL%TYPE;
    BEGIN
        V_SAL := ANG_PACKAGE.GET_SAL(1);
        DBMS_OUTPUT.PUT_LINE('Salariul corespunzator angajatului cu id-ul specificat
        este: ' || V_SAL);
        ANG_PACKAGE.REDUCERE_SALARIU(1, 0.1);
    END;
```

Declanșatori

```
--24. Sa se creeze un trigger care afiseaza un status de fiecare data cand
--una din comenzile INSERT, DELETE, UPDATE este rulata in tabela PROIECTE
CREATE OR REPLACE TRIGGER DISPLAY_OPERATION_TRIGGER
AFTER INSERT OR UPDATE OR DELETE ON PROIECTE
BEGIN
    IF INSERTING THEN
        DBMS_OUTPUT.PUT_LINE('Comanda de inserare in tabela PROIECTE a fost rulat cu succes');
    ELSIF UPDATING THEN
        DBMS_OUTPUT.PUT_LINE('Comanda de actualizare in tabela PROIECTE a fost rulata cu succes');
    ELSIF DELETING THEN
        DBMS_OUTPUT.PUT_LINE('Comanda de stergere in tabela PROIECTE a fost rulata cu succes');
    END IF;
END;

--verificare
UPDATE PROIECTE SET NUME = 'nume_schimbat' WHERE ID_PROIECT = 1;
```

```
--25. Sa se creeze un trigger care se declanseaza inaintea fiecarui update al salariului
-- unui angajat si aduaga userul si data la care s-a facut operatiunea
CREATE TABLE LOG_TABLE
(USER_ID VARCHAR(50),
LOGON_DATE DATE);

CREATE OR REPLACE TRIGGER LOG_SAL_UPDATE
BEFORE UPDATE OF SALARIUL ON ANGAJATI
BEGIN
    INSERT INTO LOG_TABLE(USER_ID, LOGON_DATE)
    VALUES(USER,SYSDATE);
END;

--verificare
UPDATE ANGAJATI SET SALARIUL = SALARIUL * 1.3 WHERE ID_ANGAJAT = 3;
SELECT * FROM LOG_TABLE;
```

```
--26. Sa se creeze un trigger care arunca o eroare daca valoarea ce se doreste
--sa modifice salariul sau sa fie introdusa in acest camp este mai mica decat
--minimul pe economie

CREATE OR REPLACE TRIGGER TOO_SMALL_SALARY_TRIGGER
    BEFORE INSERT OR UPDATE OF SALARIUL ON ANGAJATI FOR EACH ROW
BEGIN
    IF :NEW.SALARIUL < 2300 THEN
        RAISE_APPLICATION_ERROR(-20204, 'Angajatul trebuie sa aiba asigurat macar minimul pe economie!');
    END IF;
END;

UPDATE ANGAJATI SET SALARIUL = 1000 WHERE ID_ANGAJAT IN (1, 2, 3);
```