**Proiect Sisteme de gestiune a bazelor de date**

**Tema: Administrarea unei echipe de football**



**Matei Adrian**

**Seria D Grupa 1055**

**Cuprins**

[**Descrierea temei** 3](#_Toc72606131)

[**Schema conceptuală pentru modelarea bazei de date** 4](#_Toc72606132)

[**Interacțiunea cu serverul Oracle prin intermediul comenzilor SQL (LDD şi LMD)** 5](#_Toc72606133)

[**Structuri alternative și repetitive** 7](#_Toc72606134)

[**Tratarea excepțiilor** 9](#_Toc72606135)

[**Gestionarea cursorilor** 12](#_Toc72606136)

[**Funcții, proceduri, includerea acestora în pachete** 13](#_Toc72606137)

[**Declanșatori** 16](#_Toc72606138)

[**Aplicație APEX** 19](#_Toc72606139)

# **Descrierea temei**

Tema pe care am ales-o este “Administrarea unei echipe de football”. Am ales această temă pentru că evidența jucătorilor și a angajaților dintr-o echipă de football trebuie ținută obligatoriu, cât și contractele cu jucătorii și sponsorii echipei, de cele mai multe ori în joc fiind vorba de sume de bani destul de mari.

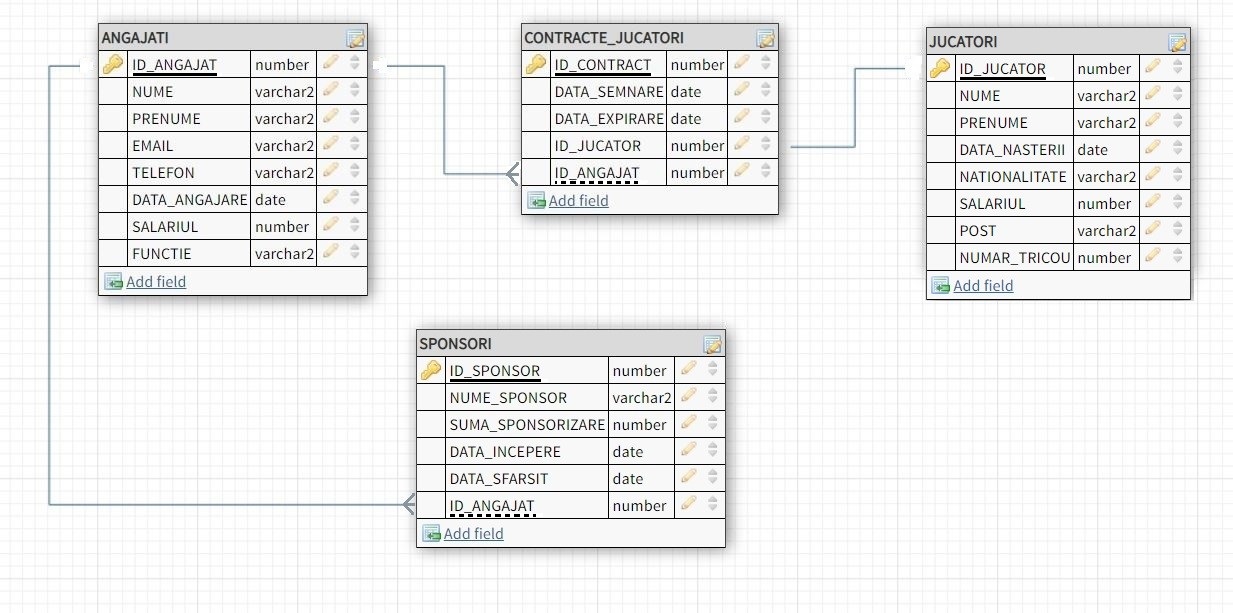
Prima tabelă este cea a angajaților, a oamenilor care se ocupă de buna funcționare a echipei (manager, director sportiv/marketing, îngrijitor teren, doctor, antrenor, etc). Am ales să pun coloana FUNCTIE tot în această tabelă (fără a mai face o tabela separată pentru funcții) deoarece echipa la care voi face referire este una fictiva, aceasta având un număr restrâns de angajați, respectiv un angajat/funcție. Restul coloanelor reprezintă date de identificare (ID\_ANGAJAT, NUME, PRENUME, EMAIL, TELEFON), dar și 2 coloane pentru a putea ține evidența angajaților (DATA\_ANGAJARE, SALARIUL).

A doua tabelă este cea a jucătorilor, având de asemenea câteva coloane pentru identificare (ID\_JUCATOR, NUME, PRENUME, DATA\_NASTERII, NATIONALITATE), dar și o coloană cu postul (coloană POST) pe care aceștia joacă, salariul (coloana SALARIUL) pe care aceștia îl iau lunar, dar și numărul de pe tricou (coloana NUMAR\_TRICOU).

A treia tabelă este cea în care sunt stocate datele despre contractele jucătorilor, data semnării și data expirării acestora (coloanele DATA\_SEMNARII, DATA\_EXPIRARII). În această tabelă se află și coloanele ID\_JUCATOR și ID\_ANGAJAT, acestea fiind chei externe. În football jucătorii au contracte încheiate pe perioade determinate de timp, semnătura fiind dată de directorul sportiv al clubului sau de o alta persoană suplinitoare în caz de acesta nu este disponibil.

A patra tabela este tabela sponsorilor, aceasta fiind o tabela importantă în ceea ce privesc finanțele clubului, aici regăsindu-se coloana cu numele sponsorilor, suma de bani cu care aceștia sponsorizează clubul, dar și data începerii și încheierii perioadei de sponsorizare. Coloana ID\_ANGAJAT reprezintă angajatul care s-a ocupat de parteneriatul respectiv, de obicei acesta fiind directorului departamentului de marketing, în situații speciale acesta fiind inlocuit de manager.

# **Schema conceptuală pentru modelarea bazei de date**



**Legăturile sunt de tip 1:M:**

Legătura dintre tabela ANGAJATI și tabela SPONSORI se face prin intermediul coloanei ID\_ANGAJAT (cheie primară în tabela ANGAJATI), contractele încheiate cu sponsorii putând fi semnate de anumiți angajați, fiecare sponsor interacționând doar cu un singur angajat;

Legatura dintre tabela ANGAJATI și tabela CONTRACTE\_JUCATORI se face prin intermediul coloanei ID\_ANGAJAT (cheie primară în tabela ANGAJATI), contractele jucătorilor putând fii semnate de anumiți angajați, fiecare contract fiind semnat doar de un singur angajat.

**Legătura de tip 1:1:**

Legătura dintre tabela JUCATORI și tabela CONTRACTE\_JUCATORI se face prin intermediul coloanei ID\_JUCATOR (cheie primară în tabela JUCATORI), fiecare jucator având un singur contract, iar fiecare contract aparține unui singur jucător.

Schema a ramas aceeasi ca pe semestrul 1.

# **Interacțiunea cu serverul Oracle prin intermediul comenzilor SQL (LDD şi LMD)**

-- 1. Sa se mareasca salariul jucatorului cu numarul tricoului 10 cu 10%

DECLARE

    V\_SALARIUL JUCATORI.SALARIUL%TYPE;

    V\_MARIRE JUCATORI.SALARIUL%TYPE;

BEGIN

    SELECT SALARIUL INTO V\_SALARIUL

    FROM JUCATORI

    WHERE NUMAR\_TRICOU = 10;

    V\_MARIRE := V\_SALARIUL \* 0.1;

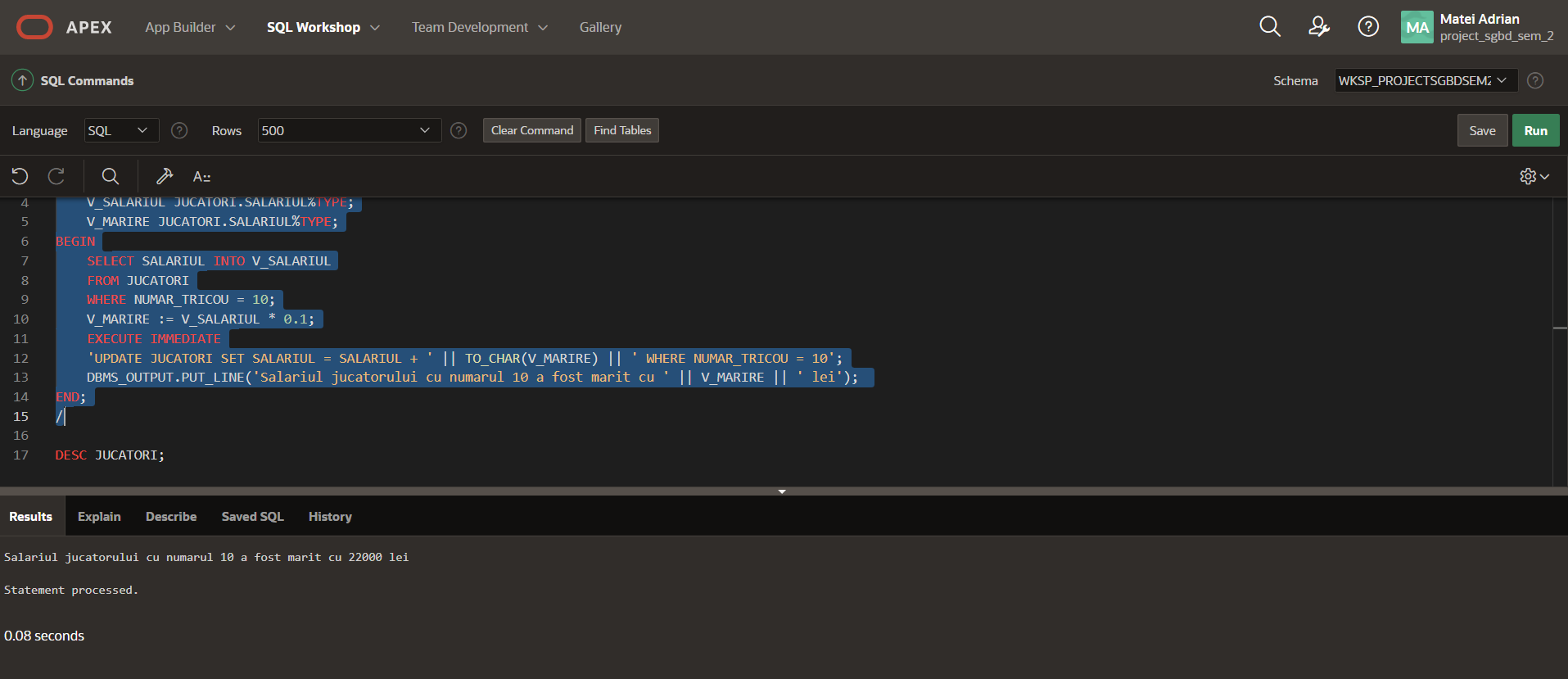
    EXECUTE IMMEDIATE

    'UPDATE JUCATORI SET SALARIUL = SALARIUL + ' || TO\_CHAR(V\_MARIRE) || ' WHERE NUMAR\_TRICOU = 10';

    DBMS\_OUTPUT.PUT\_LINE('Salariul jucatorului cu numarul 10 a fost marit cu ' || V\_MARIRE || ' lei');

END;

/



- 2. Sa se stearga angajatul cu functia de ASISTENT MEDICAL, deoarece a fost

concediat

DECLARE

    V\_FUNCTIE ANGAJATI.FUNCTIE%TYPE;

    V\_COMANDA VARCHAR2(100);

BEGIN

    V\_FUNCTIE := 'ASISTENT\_MEDICAL';

    V\_COMANDA := 'DELETE FROM ANGAJATI WHERE FUNCTIE = ''V\_FUNCTIE''';

    EXECUTE IMMEDIATE V\_COMANDA;

END;

/

-- 3. Sa se afiseze media salariilor angajatilor folosind PL/SQL

DECLARE

    V\_MEDIE ANGAJATI.SALARIUL%TYPE;

BEGIN

    SELECT AVG(SALARIUL) INTO V\_MEDIE

    FROM ANGAJATI;

    DBMS\_OUTPUT.PUT\_LINE('Media salariului angajatilor este de '||V\_MEDIE||' lei');

END;

/

- 4. Sa se schimbe tipul coloanei POST din tabela JUCATORI din vachar2(50) in varchar2(25)

BEGIN

    EXECUTE IMMEDIATE

    'ALTER TABLE JUCATORI MODIFY POST VARCHAR2(25)';

    DBMS\_OUTPUT.PUT\_LINE('S-a acutualizat cu succes');

END;

/

- 5. Sa se afiseze data pana la care ar fi valabil contractul cu ID 4 daca acesta se va prelungi cu 6 luni

DECLARE

    V\_CONTRACT CONTRACTE\_JUCATORI%ROWTYPE;

BEGIN

    SELECT \* INTO V\_CONTRACT

    FROM CONTRACTE\_JUCATORI WHERE ID\_CONTRACT=4;

    DBMS\_OUTPUT.PUT\_LINE('Contractul cu id-ul ' || TO\_CHAR(V\_CONTRACT.ID\_CONTRACT) || ' va expira pe data de ' || TO\_CHAR(ADD\_MONTHS(V\_CONTRACT.DATA\_EXPIRARE, 6)) ||

    ' daca se va prelungi cu 6 luni.');

END;

/

# **Structuri alternative și repetitive**

- 6. Sa se reduca salariul 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 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 afiseze lista angajatilor cu idul cuprins inre 2 si 5 odata, urmand ca apoi sa se afiseze cei cu idul cuprins intre 4 si 7 odata

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;

/

- 8. Sa se studieze care este salariul angajatului cu id 4 daca este redus 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 se va reduce cu 15%. Se va folosi 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;

/

-- 9. Sa se prelungeasca cu o luna toate contractele jucatorilor

DECLARE

    V\_COUNT NUMBER;

    i NUMBER;

BEGIN

    SELECT COUNT(\*) INTO V\_COUNT FROM CONTRACTE\_JUCATORI;

    i := 1;

    WHILE(i < V\_COUNT) LOOP

        UPDATE CONTRACTE\_JUCATORI SET DATA\_EXPIRARE = ADD\_MONTHS(DATA\_EXPIRARE, 1) WHERE ID\_CONTRACT = i;

        DBMS\_OUTPUT.PUT\_LINE('S-a prelungit cu o luna contractul cu id-ul ' || TO\_CHAR(i));

        i := i + 1;

    END LOOP;

END;

# **Tratarea excepțiilor**

- 10.Sa se afiseze contractul jucatorului cu data semnarii 03/04/2020. Sa se trateze eroare aparuta in

-- cazul in care exista mai multe contracte cu aceeasi data de semnare.

DECLARE

    V\_DATA CONTRACTE\_JUCATORI.DATA\_SEMNARE%TYPE;

    V\_ID CONTRACTE\_JUCATORI.ID\_CONTRACT%TYPE;

BEGIN

    SELECT ID\_CONTRACT, DATA\_SEMNARE INTO V\_ID, V\_DATA FROM CONTRACTE\_JUCATORI WHERE DATA\_SEMNARE=TO\_DATE('03/04/2020','MM/DD/YYYY');

    DBMS\_OUTPUT.PUT\_LINE('ID-ul contractului semnat pe 03/04/2020 este ' || V\_ID);

EXCEPTION

    WHEN TOO\_MANY\_ROWS THEN

        DBMS\_OUTPUT.PUT\_LINE('Exista mai multe contracte semnate la aceasta data!');

END;

/

- 11.Sa se trateze exceptia atunci cand se incearca impartirea la 0 a salariului unui angajat

DECLARE

    V\_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;

/

- 12. Sa se afiseze salariul jucatorului cu numarul tricoului 15. Daca acesta nu exista, se fa trata exceptia

DECLARE

    V\_SALARIUL JUCATORI.SALARIUL%TYPE;

BEGIN

    SELECT SALARIUL INTO V\_SALARIUL FROM JUCATORI WHERE NUMAR\_TRICOU = 15;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

    DBMS\_OUTPUT.PUT\_LINE('Exceptie! Acest jucator nu exista in baza de date!');

END;

/

- 13.Sa se afiseze contractul jucatorului cu data semnarii 03/04/2020. Sa se trateze eroare aparuta in

-- cazul in care exista mai multe contracte cu aceeasi data de semnare. In cazul introducerii unei date gresite, sa se trateze exceptia la mod general.

DECLARE

    V\_DATA CONTRACTE\_JUCATORI.DATA\_SEMNARE%TYPE;

    V\_ID CONTRACTE\_JUCATORI.ID\_CONTRACT%TYPE;

BEGIN

    SELECT ID\_CONTRACT, DATA\_SEMNARE INTO V\_ID, V\_DATA FROM CONTRACTE\_JUCATORI WHERE DATA\_SEMNARE=TO\_DATE('03/0dsd/2020','MM/DD/YYYY');

    DBMS\_OUTPUT.PUT\_LINE('ID-ul contractului semnat pe 03/04/2020 este ' || V\_ID);

EXCEPTION

    WHEN TOO\_MANY\_ROWS THEN

        DBMS\_OUTPUT.PUT\_LINE('Exista mai multe contracte semnate la aceasta data!');

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Exceptie!');

END;

/

- 14. Sa se mareasca cu 10% salariul angajatului cu functia de Traducator. Daca aceasta functie nu exista, sa se trateze exceptia

DECLARE

    EX\_FUNCTIE EXCEPTION;

BEGIN

    UPDATE ANGAJATI SET SALARIUL = SALARIUL \* 0.1 WHERE FUNCTIE = 'Traducator';

    IF SQL%NOTFOUND THEN

        RAISE EX\_FUNCTIE;

    END IF;

EXCEPTION

   WHEN EX\_FUNCTIE THEN

        DBMS\_OUTPUT.PUT\_LINE('Functia nu exista!');

   WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Exceptie!');

END;

/

- 15.Sa se incerce modificare numarului de telefon al angajatului cu id 2. Sa se trateze exceptia in care numarul de telefon nu respecta numarul de cifre

DECLARE

    V\_TELEFON NUMBER;

    EX\_INVALID\_TELEFON EXCEPTION;

    PRAGMA EXCEPTION\_INIT(EX\_INVALID\_TELEFON,-20010);

BEGIN

    V\_TELEFON := '0742423168254';

    IF(LENGTH(V\_TELEFON) <> 10) THEN

         RAISE\_APPLICATION\_ERROR (-20010,'Numar de telefon invalid!');

    ELSE

        UPDATE ANGAJATI SET TELEFON = V\_TELEFON WHERE ID\_ANGAJAT = 2;

    END IF;

EXCEPTION

    WHEN EX\_INVALID\_TELEFON THEN

        DBMS\_OUTPUT.PUT\_LINE('Numar de telefon invalid');

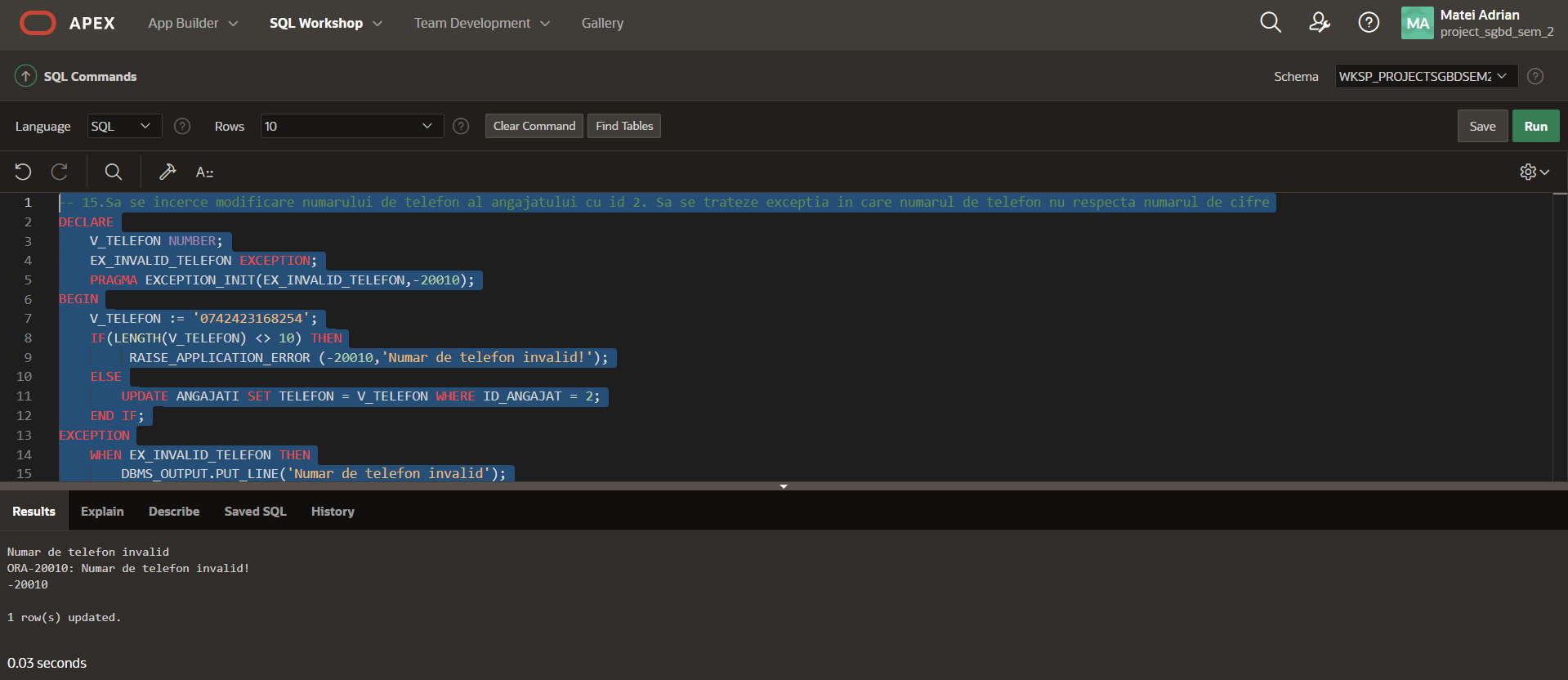
        DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

        DBMS\_OUTPUT.PUT\_LINE(SQLCODE);

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Exceptie!');

END;



# **Gestionarea cursorilor**

- 16. Sa se mareasca salariul angajatului cu idul introdus de la tastatura. Daca update-ul nu se va realiza, sa se afiseze un anunt in acest sens

ACCEPT a\_id PROMPT 'Introduceti id-ul angajatului:'

DECLARE

    V\_ID ANGAJATI.ID\_ANGAJAT%TYPE := &a\_id;

BEGIN

    UPDATE ANGAJATI SET SALARIUL = SALARIUL \* 0.1 WHERE ID\_ANGAJAT = V\_ID;

    IF SQL%NOTFOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('Nu exista angajatul cu acest cod');

    END IF;

END;

/

17.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;

/

- 18. 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 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;

/

-- 19. Sa se afiseze datele jucatorului cu id-ul transmis prin parametru intr-un cursor explicit

DECLARE

    CURSOR C\_JUCATOR (P\_ID NUMBER) IS SELECT NUME, POST, SALARIUL

    FROM JUCATORI WHERE ID\_JUCATOR = P\_ID;

    REC\_JUCATOR C\_JUCATOR%ROWTYPE;

BEGIN

IF NOT C\_JUCATOR%ISOPEN THEN

    OPEN C\_JUCATOR(1);

END IF;

FETCH C\_JUCATOR INTO REC\_JUCATOR;

DBMS\_OUTPUT.PUT\_LINE('Jucatorul gasit pentru id-ul dat: ' || REC\_JUCATOR.NUME || ' ' || REC\_JUCATOR.POST || ' ' || REC\_JUCATOR.SALARIUL);

CLOSE C\_JUCATOR;

END;

/

# **Funcții, proceduri, includerea acestora în pachete**

- 20.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;

- 21. Sa se creeze o functie care verifica daca exista un anumit sponsor in tabela cu sponsori primind drept parametru numele acestuia

CREATE OR REPLACE FUNCTION VALID\_SPONSOR (P\_SPONSOR SPONSORI.NUME\_SPONSOR%TYPE)

RETURN BOOLEAN IS

    V\_ID SPONSORI.ID\_SPONSOR%TYPE;

BEGIN

    SELECT ID\_SPONSOR INTO V\_ID FROM SPONSORI

    WHERE NUME\_SPONSOR = P\_SPONSOR;

    RETURN TRUE;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        RETURN FALSE;

END;

DECLARE

    V\_VALID BOOLEAN;

    V\_NUME\_SPONSOR SPONSORI.NUME\_SPONSOR%TYPE;

BEGIN

    V\_NUME\_SPONSOR := UPPER('DAcsIA');

    IF VALID\_SPONSOR(UPPER(V\_NUME\_SPONSOR)) THEN

        DBMS\_OUTPUT.PUT\_LINE('Sponsorul ' || V\_NUME\_SPONSOR || ' exista in tabela cu sponsori');

    ELSE

        DBMS\_OUTPUT.PUT\_LINE('Sponsorul ' || V\_NUME\_SPONSOR || ' nu exista in tabela cu sponsori');

    END IF;

END;

- 22. Sa se creeze o functie care returneaza suma totala a sponsorizarilor din tabela cu sponsori

CREATE OR REPLACE FUNCTION GET\_SUMA\_TOTALA\_SPONSORIZARI

RETURN NUMBER IS

    V\_SUMA SPONSORI.SUMA\_SPONSORIZARE%TYPE;

BEGIN

    SELECT SUM(SUMA\_SPONSORIZARE) INTO V\_SUMA FROM SPONSORI;

    RETURN V\_SUMA;

END;

DECLARE

    V\_SUMA SPONSORI.SUMA\_SPONSORIZARE%TYPE;

BEGIN

    V\_SUMA := GET\_SUMA\_TOTALA\_SPONSORIZARI;

    DBMS\_OUTPUT.PUT\_LINE('Suma totala a sponsorizarilor este de ' || V\_SUMA || ' lei.');

END;

-- 23. Creeaza o procedura care face acelasi lucru ca si functia de mai sus

CREATE OR REPLACE PROCEDURE SUMA\_SPONSORIZARI\_TOTALA (P\_SUMA OUT NUMBER) AS

BEGIN

    SELECT SUM(SUMA\_SPONSORIZARE) INTO P\_SUMA FROM SPONSORI;

END;

DECLARE

    V\_SUMA SPONSORI.SUMA\_SPONSORIZARE%TYPE;

BEGIN

    SUMA\_SPONSORIZARI\_TOTALA(V\_SUMA);

    DBMS\_OUTPUT.PUT\_LINE('Suma totala a sponsorizarilor este de ' || V\_SUMA || ' lei.');

END;

- 24. 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;

- 25. 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) 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;

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**

- 26. Sa se creeze un trigger care afiseaza un status de fiecare data cand una din comenzile INSERT, DELETE, UPDATE este rulata in tabela JUCATORI

CREATE OR REPLACE TRIGGER DISPLAY\_OPERATION\_TRIGGER

AFTER INSERT OR UPDATE OR DELETE ON JUCATORI

BEGIN

    IF INSERTING THEN

        DBMS\_OUTPUT.PUT\_LINE('Comanda de inserare in tabela JUCATORI a fost rulata cu succes!');

    ELSIF UPDATING THEN

        DBMS\_OUTPUT.PUT\_LINE('Comanda de actualizare in tabela JUCATORI a fost rulata cu succes!');

    ELSIF DELETING THEN

        DBMS\_OUTPUT.PUT\_LINE('Comanda de stergere in tabela JUCATORI a fost rulata cu succes!');

    END IF;

END;

UPDATE JUCATORI SET SALARIUL = SALARIUL \* 1.1 WHERE ID\_JUCATOR = 13133;

- 27. Sa se creeze un trigger care se declanseaza inaintea fiecarui update al salariului unui angajat si adauga 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;

- 28. 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);

- 29. Sa se creeze un trigger care arunca o exceptie daca la incercarea modificarii id-ului angajatului cu o functie ce permite semnarea contractelor se introduce un id al unui angajat fara functie ce permite semnarea contractelor

CREATE OR REPLACE TRIGGER FUNCTIE\_INVALIDA\_TRIGGER

BEFORE UPDATE OF ID\_ANGAJAT ON CONTRACTE\_JUCATORI FOR EACH ROW

DECLARE

    V\_ID\_ANGAJAT ANGAJATI.ID\_ANGAJAT%TYPE;

    FUNCTIE\_INVALIDA EXCEPTION;

    PRAGMA EXCEPTION\_INIT(FUNCTIE\_INVALIDA, -20404);

BEGIN

    IF :NEW.ID\_ANGAJAT NOT IN (1, 2) THEN

        RAISE\_APPLICATION\_ERROR(-20404, 'Doar managerul si directorul sportiv au drept de semnare!');

    END IF;

EXCEPTION

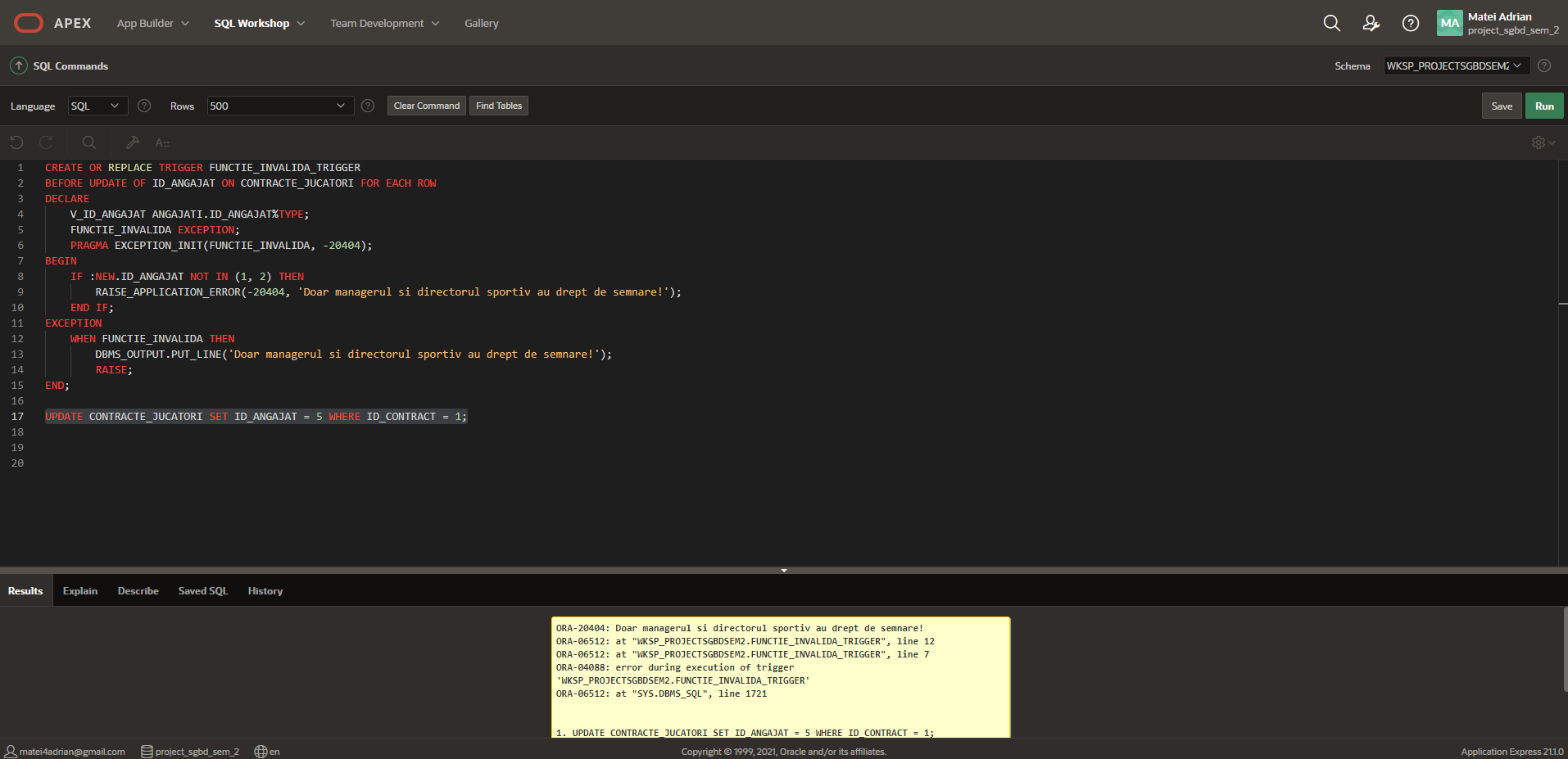
    WHEN FUNCTIE\_INVALIDA THEN

        DBMS\_OUTPUT.PUT\_LINE('Doar managerul si directorul sportiv au drept de semnare!');

        RAISE;

END;

UPDATE CONTRACTE\_JUCATORI SET ID\_ANGAJAT = 5 WHERE ID\_CONTRACT = 1;



# **Aplicație APEX**

Link:[**Proiect\_Matei\_Adrian\_1055 (oracle.com)**](https://apex.oracle.com/pls/apex/project_sgbd_sem_2/r/proiect-matei-adrian-1055/home?session=111614897155490)

