**PROIECT SGBD**

Realizat de Stoica Elias-Valeriu

Facultatea de Matematica și Informatica Bucuresti

Grupa 251

1. **Descrierea modelului real, a utilității acestuia si a regulilor de funcționare**

Ligile din Europa pun la start echipe din diferite colțuri ale țărilor. Fiecare echipa are un stadion în orașul de provenienta, unde disputa meciurile de pe propriul teren. Stadioanele se clasifica după capacitate, dupa numarul de stele oferit de UEFA, depinzand de cat de modern este. O echipa este alcatuita dintr-un lot de jucători. Acestea sunt pregatiti de un antrenor principal.

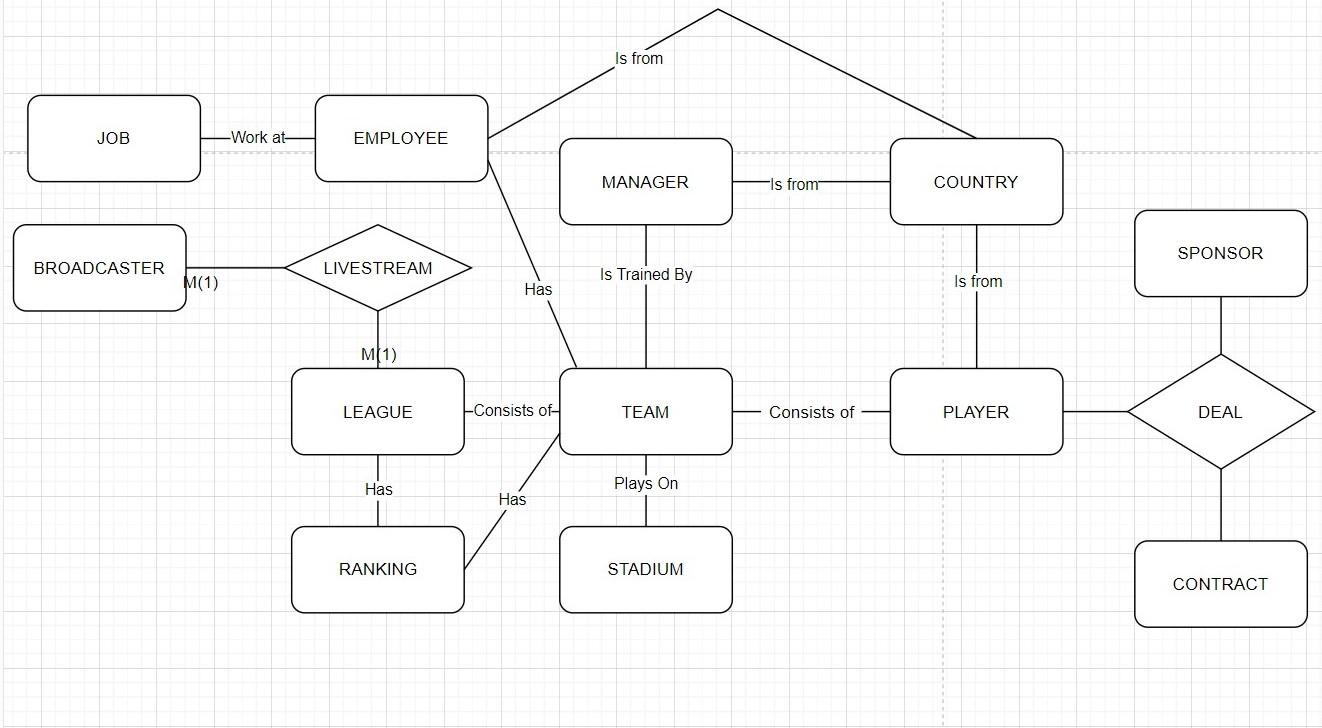
Fiecare echipa are în spate un număr mare de angajați. Aceștia primesc un salariu atractiv, în funcție de job. Pentru fiecare membru din club se cunoaște țara de proveniență.

Jucătorii sunt sponsorizati de anumite companii. Acestia au un contract bine stabilit si nu pot avea mai multe sponsorizări în același timp. Sponsorizarile constau în produse de marca, adidasi, echipament etc. Acest contract poate avea și o clauza de reziliere, in cazul in care exista, jucătorul poate renunța oricând la sponsorul sau.

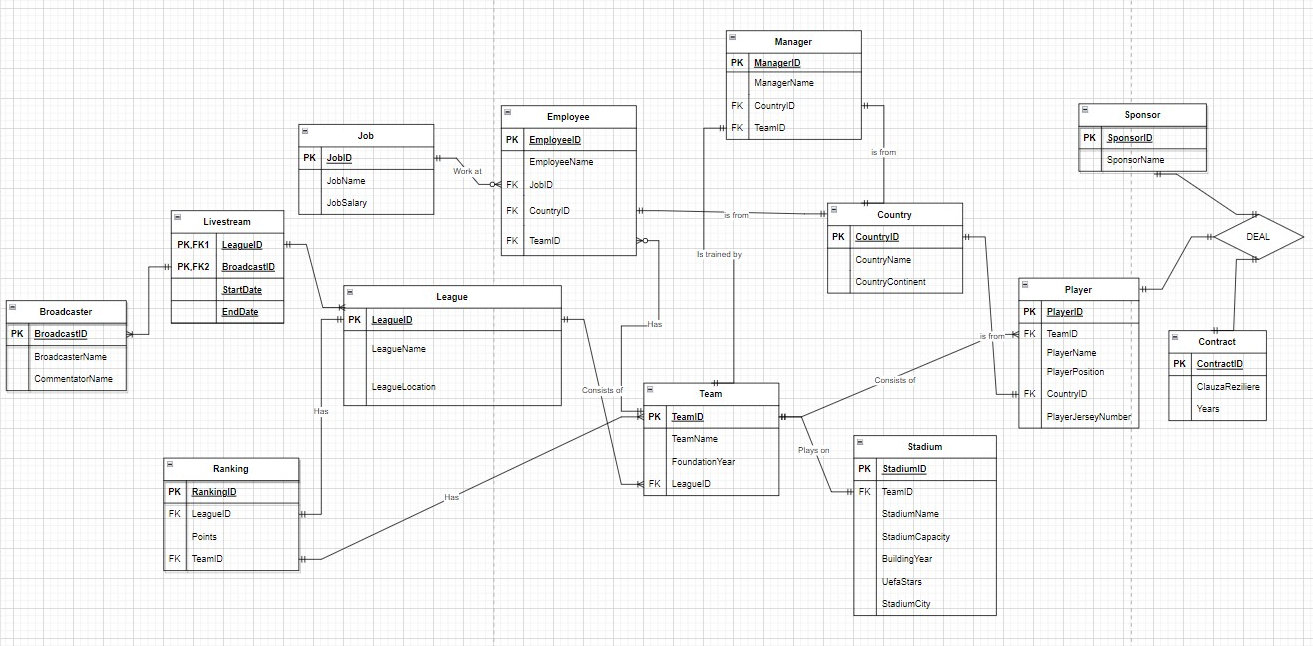
Campionatul de fotbal are mai multe contracte de livestreaming cu diferite televiziuni. Acestea sunt făcute pe o perioada bine definită.

La finalul sezonului, echipele își află locul în clasament. În funcție de cate puncte acumulează în timpul sezonului, se pot bate pentru competitiile europene.

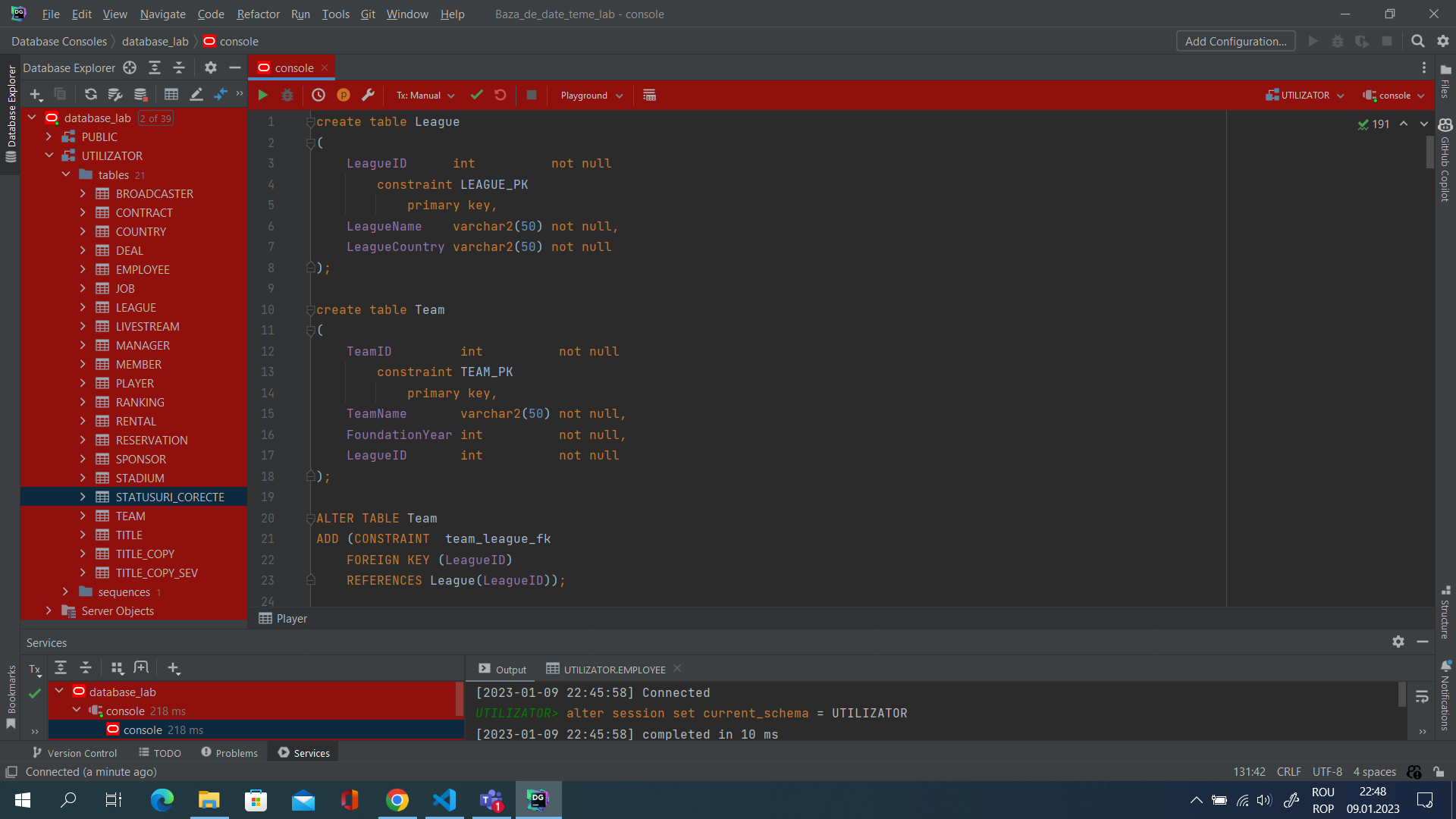
1. **Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5.**



1. **Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectate la punctul 6.**



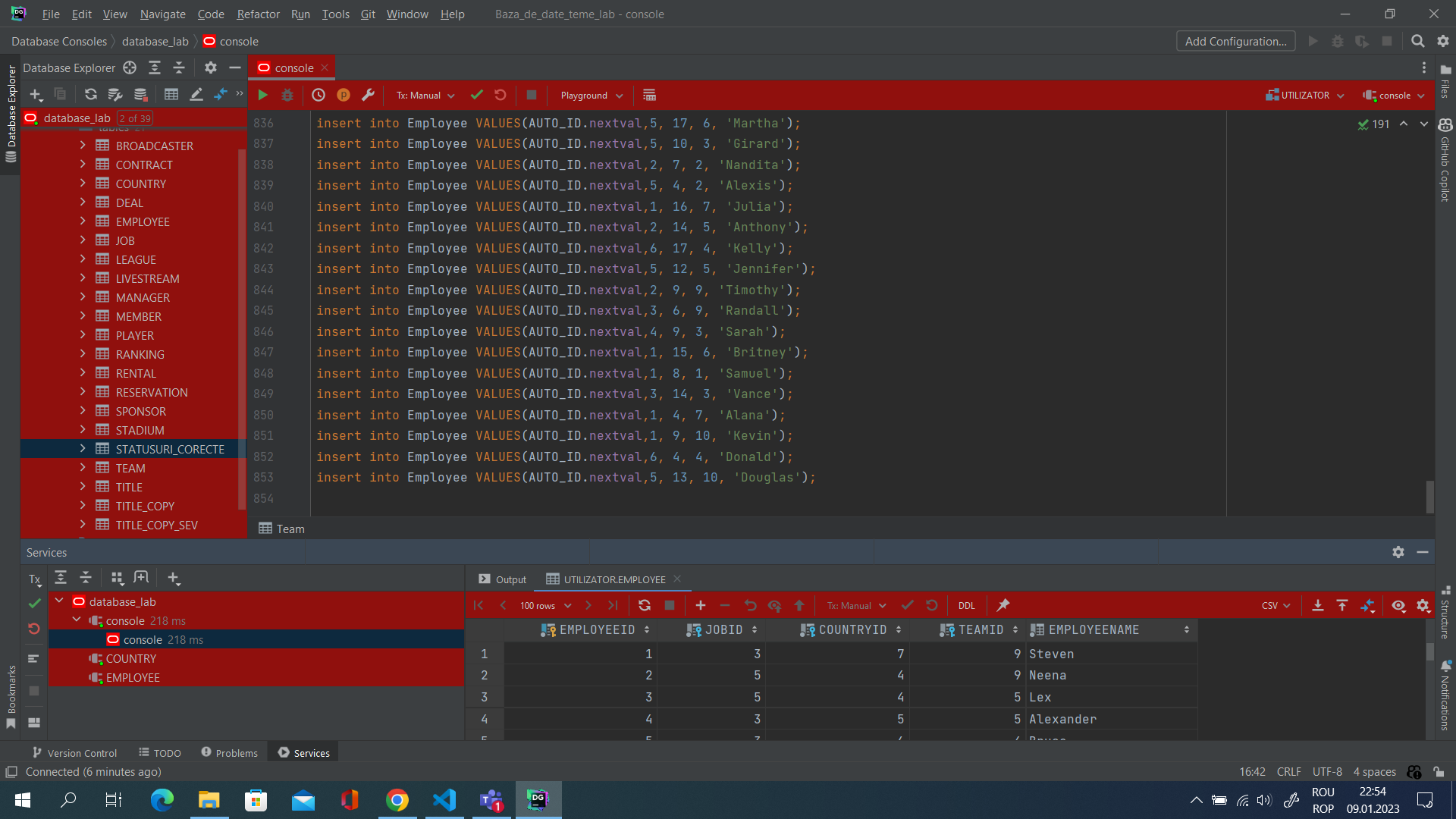
1. **Implementați în Oracle diagrama conceptuală realizată. Definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, chei externe etc.).**



Fișierele de creare sunt într-un repository pe github, la adresa următoare:

[**https://github.com/wood11nho/Proiect-SGBD**](https://github.com/wood11nho/Proiect-SGBD)

1. **Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independență; minim 10 înregistrări pentru tabela asociativă).**



Fișierele cu comenzile de inserare se găsesc pe următorul repository de pe github:

[**https://github.com/wood11nho/Proiect-SGBD**](https://github.com/wood11nho/Proiect-SGBD)

1. **Formulați în limbaj natural o problema pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze două tipuri de colecție studiate. Apelați subprogramul.**

**--Pentru un jucător de fotbal, căruia i se va da ID-ul de la tastatură, salvați și afișați numele tuturor coechipierilor săi, iar pentru fiecare coechipier să se salveze și sponsorul acestuia.**

create or replace procedure *afisare\_coechipieri*

(id\_jucator\_tast Player.PlayerID%type)

AS

TYPE tablou\_indexat is TABLE OF Player.PlayerName%type INDEX BY PLS\_INTEGER;

nume\_coechipieri tablou\_indexat;

TYPE tablou\_imbricat is TABLE OF Sponsor.SponsorName%type;

sponsor\_jucatori tablou\_imbricat := tablou\_imbricat();

i NUMBER;

id\_echipa\_jucator Player.TeamID%type;

id\_echipa\_coechiper Player.TeamID%type;

nume\_jucator Player.PlayerName%type;

id\_coechipier Player.PlayerID%type;

nume\_coechipier Player.PlayerName%type;

nume\_sponsor Sponsor.SponsorName%type;

numar\_jucatori NUMBER;

BEGIN

i := 0;

select TeamID, PlayerName into id\_echipa\_jucator, nume\_jucator from Player where PlayerID = id\_jucator\_tast;

select *count*(\*) into numar\_jucatori from Player;

for id\_jucator in 1..numar\_jucatori

loop

select TeamID into id\_echipa\_coechiper from Player where PlayerID = id\_jucator;

if id\_echipa\_coechiper = id\_echipa\_jucator and id\_jucator\_tast <> id\_jucator then

i := i + 1;

select PlayerName, PlayerID into nume\_coechipier, id\_coechipier from Player where PlayerID = id\_jucator;

nume\_coechipieri(i) := nume\_coechipier;

sponsor\_jucatori.extend;

select SponsorName into nume\_sponsor from Sponsor where SponsorID = (select SponsorID from Deal where PlayerID = id\_coechipier);

sponsor\_jucatori(i) := nume\_sponsor;

end if;

end loop;

DBMS\_OUTPUT.*PUT\_LINE*('Jucatorul ' || nume\_jucator || ' are urmatorii coechipieri:');

FOR j IN 1..nume\_coechipieri.COUNT LOOP

DBMS\_OUTPUT.*PUT\_LINE*(nume\_coechipieri(j) || ' - ' || sponsor\_jucatori(j));

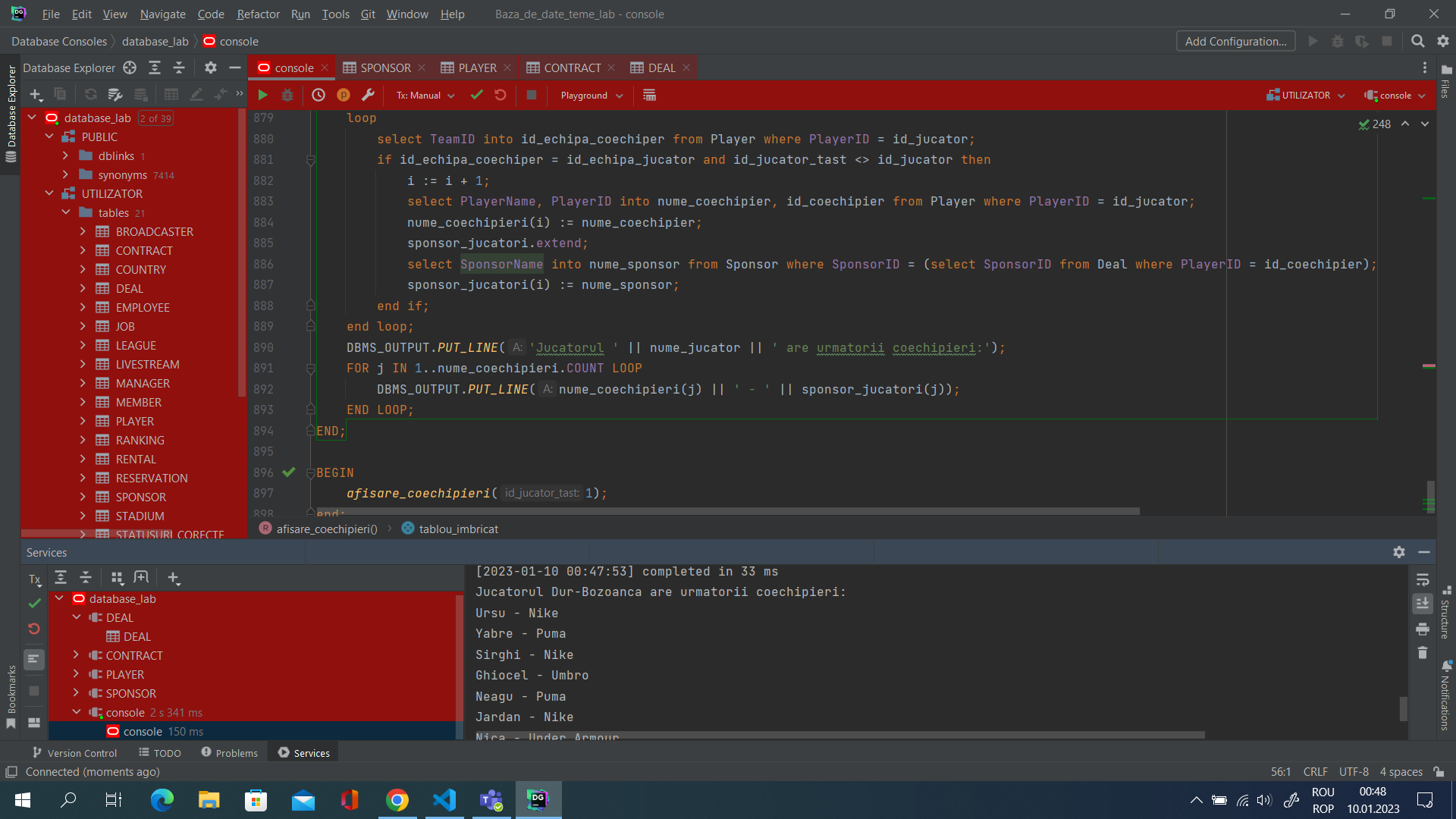
END LOOP;

END;

BEGIN

*AFISARE\_COECHIPIERI*(1);

end;



1. **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri de cursoare studiate, unul dintre acestea fiind parametrizat. Apelați subprogramul.**

**--Să se afișeze pentru fiecare echipă numărul de jucători. În cazul în care echipa nu are jucători înscriși, se va afișa un mesaj corespunzător. Pentru fiecare jucător cu numărul de pe tricou mai mare decât un parametru dat, să se afișeze poziția, numărul de pe tricou și țara de proveniență.**

CREATE OR REPLACE FUNCTION *filtrare\_jucatori\_dupa\_numar*

(numar\_echipa Player.TeamID%TYPE, numar\_tricou Player.JerseyNumber%TYPE)

RETURN NUMBER is

nr\_jucatori\_filtrati NUMBER;

BEGIN

select *count*(\*) into nr\_jucatori\_filtrati

from Player

where JerseyNumber > numar\_tricou and TeamID = numar\_echipa;

return nr\_jucatori\_filtrati;

end;

CREATE OR REPLACE PROCEDURE *afisare\_detalii\_echipe* AS

CURSOR c1 is

SELECT t.TeamID id\_echipa, TeamName nume\_echipa, *COUNT*(PlayerID) numar\_jucatori

FROM Team t, Player p

WHERE t.TeamID = p.TeamID(+)

GROUP BY t.TeamID, TeamName;

CURSOR c2(parametru NUMBER, echipa\_id Team.TeamID%TYPE) is

SELECT PlayerName nume\_jucator, PlayerPosition pozitie\_jucator, CountryName tara\_jucator, JerseyNumber numar\_jucator

FROM Player p, Country c

WHERE p.CountryID = c.CountryID(+) and JerseyNumber > parametru and TeamID = echipa\_id;

TYPE tip\_detalii\_echipa IS RECORD(

id\_echipa Team.TeamID%TYPE,

nume\_echipa Team.TeamName%TYPE,

numar\_jucatori NUMBER

);

detalii\_echipa tip\_detalii\_echipa;

count\_echipe NUMBER;

count\_jucatori NUMBER;

numar\_jucator\_tastatura NUMBER := &numar\_tast;

nr\_jucatori\_filtrati NUMBER;

BEGIN

DBMS\_OUTPUT.*PUT\_LINE*('Echipe si jucatori: ');

count\_echipe := 0;

OPEN c1;

LOOP

FETCH c1 INTO detalii\_echipa;

count\_echipe := count\_echipe + 1;

EXIT WHEN c1%NOTFOUND;

IF detalii\_echipa.numar\_jucatori = 0 THEN

DBMS\_OUTPUT.*PUT\_LINE*(count\_echipe || '. ' || detalii\_echipa.nume\_echipa || ' nu are niciun jucator semnat.');

ELSE

nr\_jucatori\_filtrati:=*filtrare\_jucatori\_dupa\_numar*(detalii\_echipa.id\_echipa,numar\_jucator\_tastatura);

DBMS\_OUTPUT.*PUT\_LINE*(count\_echipe || '. ' || detalii\_echipa.nume\_echipa || ' are ' || detalii\_echipa.numar\_jucatori || ' jucatori semnati iar cei ' || nr\_jucatori\_filtrati || ' cu numarul mai mare decat ' || numar\_jucator\_tastatura || ' sunt: ');

count\_jucatori := 0;

FOR i in c2(numar\_jucator\_tastatura, detalii\_echipa.id\_echipa) LOOP

count\_jucatori := count\_jucatori + 1;

DBMS\_OUTPUT.*PUT\_LINE*(count\_jucatori || '. ' || i.nume\_jucator || ', numarul ' || i.numar\_jucator || ', joaca pe pozitia ' || i.pozitie\_jucator || ' si este din ' || i.tara\_jucator);

end loop;

END IF;

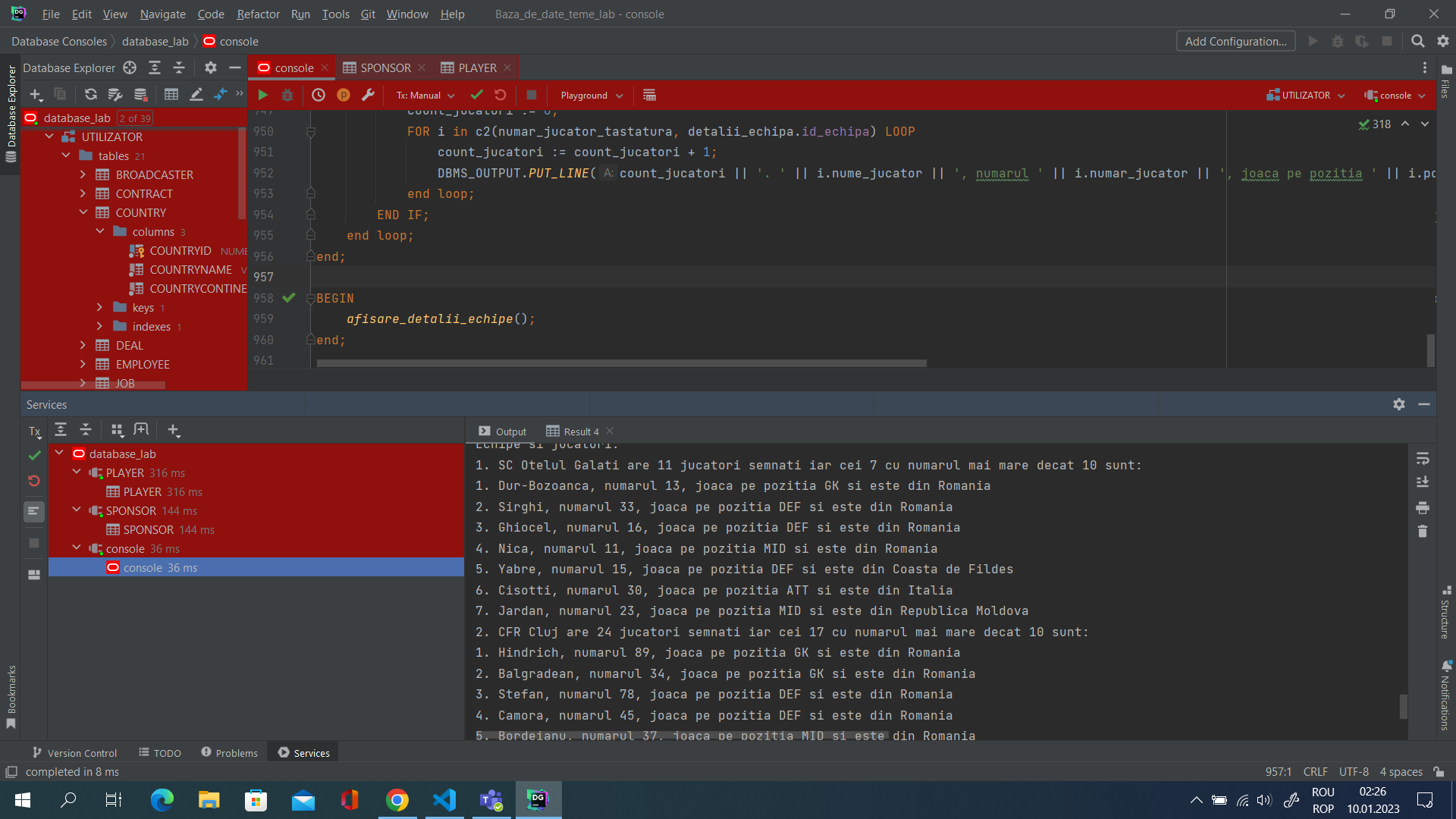
end loop;

end;

BEGIN

*afisare\_detalii\_echipe*();

end;



1. **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singura comanda SQL 3 tabele diferite. Definiți minim 2 excepții.**

**--Scrie o funcție care va afișa stadionul pe care evoluează echipa unui jucător, al cărui nume va fi dat de la tastatura.**

CREATE OR REPLACE FUNCTION *stadion\_jucator*

(nume\_jucator Player.PlayerName%TYPE)

RETURN Stadium.StadiumName%TYPE IS

id\_jucator Player.PlayerID%TYPE;

nume\_stadion Stadium.StadiumName%TYPE;

BEGIN

select PlayerID

into id\_jucator

from Player

where PlayerName = nume\_jucator;

select StadiumName

into nume\_stadion

from Stadium s

join Team t on t.TeamID = s.TeamID

join Player p on p.TeamID = t.TeamID

where p.PlayerID = id\_jucator;

return nume\_stadion;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

*RAISE\_APPLICATION\_ERROR*(-20000, 'Nu exista jucator cu numele ' || nume\_jucator);

WHEN OTHERS THEN

*RAISE\_APPLICATION\_ERROR*(-20002, 'Alta eroare');

end;

BEGIN

DBMS\_OUTPUT.*PUT\_LINE*('Jucatorul cu numele Sefer evolueaza pe ' || *stadion\_jucator*('Sefer'));

--Raspuns:

--[2023-01-10 03:17:42] completed in 15 ms

--Jucatorul cu numele Sefer evolueaza pe stadionul Stadionul Giulesti

DBMS\_OUTPUT.*PUT\_LINE*('Jucatorul cu numele Elias evolueaza pe ' || *stadion\_jucator*('Elias'));

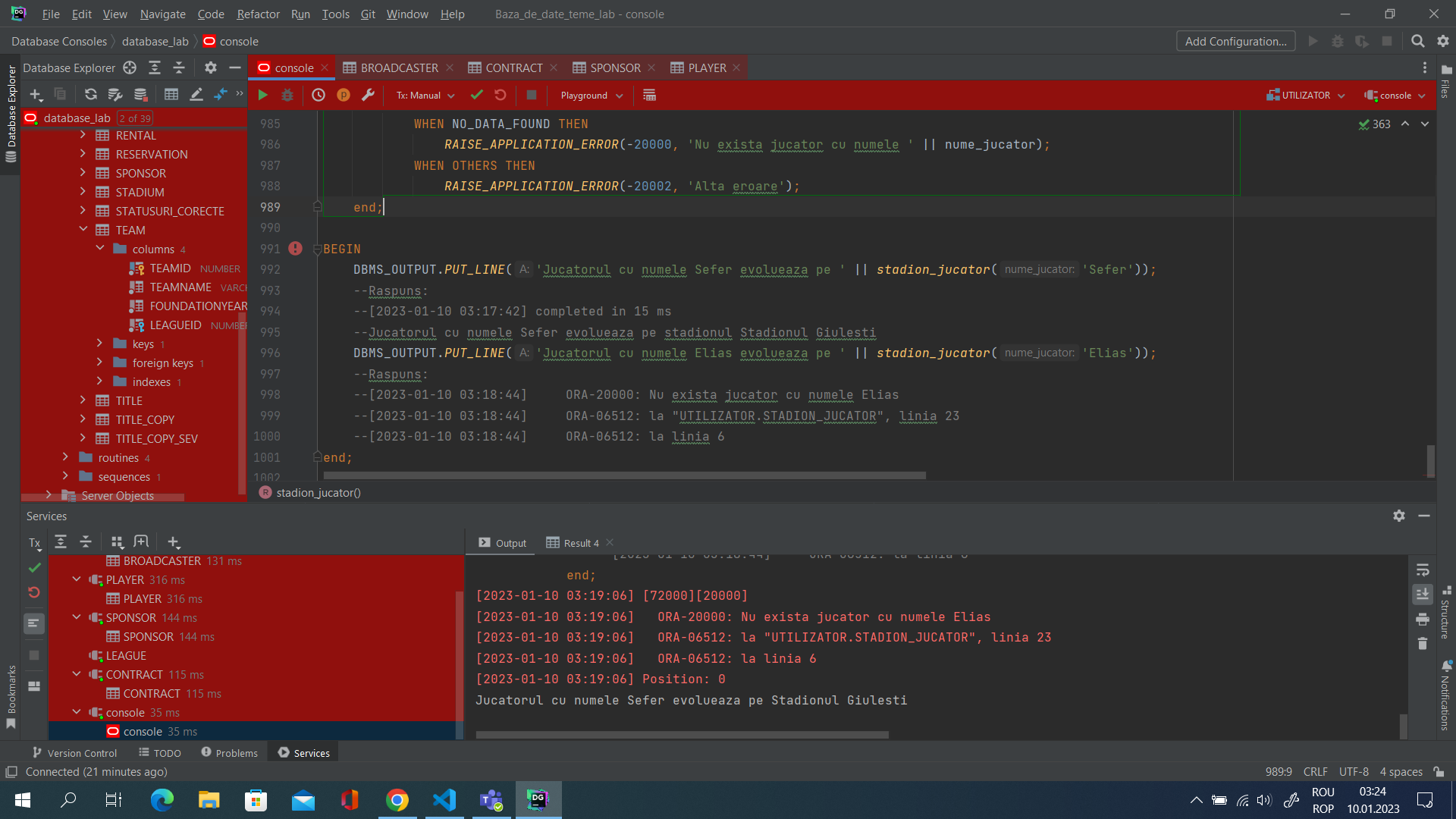
--Raspuns:

--[2023-01-10 03:18:44] ORA-20000: Nu exista jucator cu numele Elias

--[2023-01-10 03:18:44] ORA-06512: la "UTILIZATOR.STADION\_JUCATOR", linia 23

--[2023-01-10 03:18:44] ORA-06512: la linia 6

end;



1. **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 tabele diferite. Tratați toate excepțiile care pot apărea.**

**--Să se afișeze numele, echipa, numărul de pe tricou și sponsorul unui jucător dat ca parametru, doar dacă acesta a terminat pe locul 1 în campionatul din România și dacă are o clauză de reziliere activă în contractul cu sponsorii săi.**

CREATE OR REPLACE PROCEDURE *afisare\_jucator\_exc*

(nume\_jucator Player.PlayerName%TYPE)

IS

CURSOR c1 is

select t.TeamID id\_echipa, t.TeamName nume\_echipa, Points puncte

from Team t, Ranking r

where t.TeamID = r.TeamID;

TYPE tip\_detalii\_echipa IS RECORD (

id\_echipa Team.TeamID%TYPE,

nume\_echipa Team.TeamName%TYPE,

puncte\_echipa Ranking.Points%TYPE );

TYPE tip\_detalii\_jucator IS RECORD(

id\_jucator Player.PlayerID%TYPE,

nume\_jucator Player.PlayerName%TYPE,

echipa\_jucator Team.TeamName%TYPE,

numar\_jucator Player.JerseyNumber%TYPE,

sponsor\_jucator Sponsor.SponsorName%TYPE,

clauza\_reziliere Contract.ClauzaReziliere%TYPE);

jucator\_par tip\_detalii\_jucator;

echipa\_castigatoare tip\_detalii\_echipa;

nr\_maxim\_puncte Ranking.Points%TYPE;

exceptie\_clauza EXCEPTION;

exceptie\_echipa EXCEPTION;

BEGIN

select *MAX*(Points)

into nr\_maxim\_puncte

from Ranking;

open c1;

LOOP

FETCH c1 into echipa\_castigatoare;

EXIT WHEN echipa\_castigatoare.puncte\_echipa = nr\_maxim\_puncte;

end loop;

select p.PlayerID, p.PlayerName, t.TeamName, p.JerseyNumber, s.SponsorName, c.ClauzaReziliere

into jucator\_par

from Player p

join Team t on (t.TeamID = p.TeamID)

join Deal d on (d.PlayerID = p.PlayerID)

join Contract c on (c.ContractID = d.ContractID)

join Sponsor s on (s.SponsorID = d.SponsorID)

where p.PlayerName = nume\_jucator;

if jucator\_par.clauza\_reziliere = 0 then

raise exceptie\_clauza;

end if;

if jucator\_par.echipa\_jucator <> echipa\_castigatoare.nume\_echipa then

raise exceptie\_echipa;

end if;

DBMS\_OUTPUT.*PUT\_LINE*('Jucatorul ' || jucator\_par.nume\_jucator || ' evolueaza pentru echipa ' || jucator\_par.echipa\_jucator || ', campioana Romaniei, poarta numarul ' || jucator\_par.numar\_jucator || ' si este sponsorizat de ' || jucator\_par.sponsor\_jucator);

EXCEPTION

WHEN exceptie\_clauza THEN

*RAISE\_APPLICATION\_ERROR*(-20003, 'Jucatorul ' || nume\_jucator || ' nu are o clauza activa.');

WHEN exceptie\_echipa THEN

*RAISE\_APPLICATION\_ERROR*(-20003, 'Jucatorul ' || nume\_jucator || ' nu joaca pentru echipa castigatoare din Romania.');

WHEN NO\_DATA\_FOUND THEN

*RAISE\_APPLICATION\_ERROR*(-20000, 'Nu exista jucator cu numele ' || nume\_jucator);

WHEN TOO\_MANY\_ROWS THEN

*RAISE\_APPLICATION\_ERROR*(-20001, 'Sunt mai multi jucatori cu numele ' || nume\_jucator);

WHEN OTHERS THEN

*RAISE\_APPLICATION\_ERROR*(-20002, 'Alta eroare');

end;

BEGIN

*AFISARE\_JUCATOR\_EXC*('Susic');

--[2023-01-10 04:37:24] ORA-20003: Jucatorul Susic nu are o clauza activa.

--[2023-01-10 04:37:24] ORA-06512: la "UTILIZATOR.AFISARE\_JUCATOR\_EXC", linia 58

*AFISARE\_JUCATOR\_EXC*('Manea');

--Jucatorul Manea evolueaza pentru echipa CFR Cluj, campioana Romaniei, poarta numarul 4 si este sponsorizat de Umbro

*AFISARE\_JUCATOR\_EXC*('Elias');

--[2023-01-10 04:37:54] ORA-20000: Nu exista jucator cu numele Elias

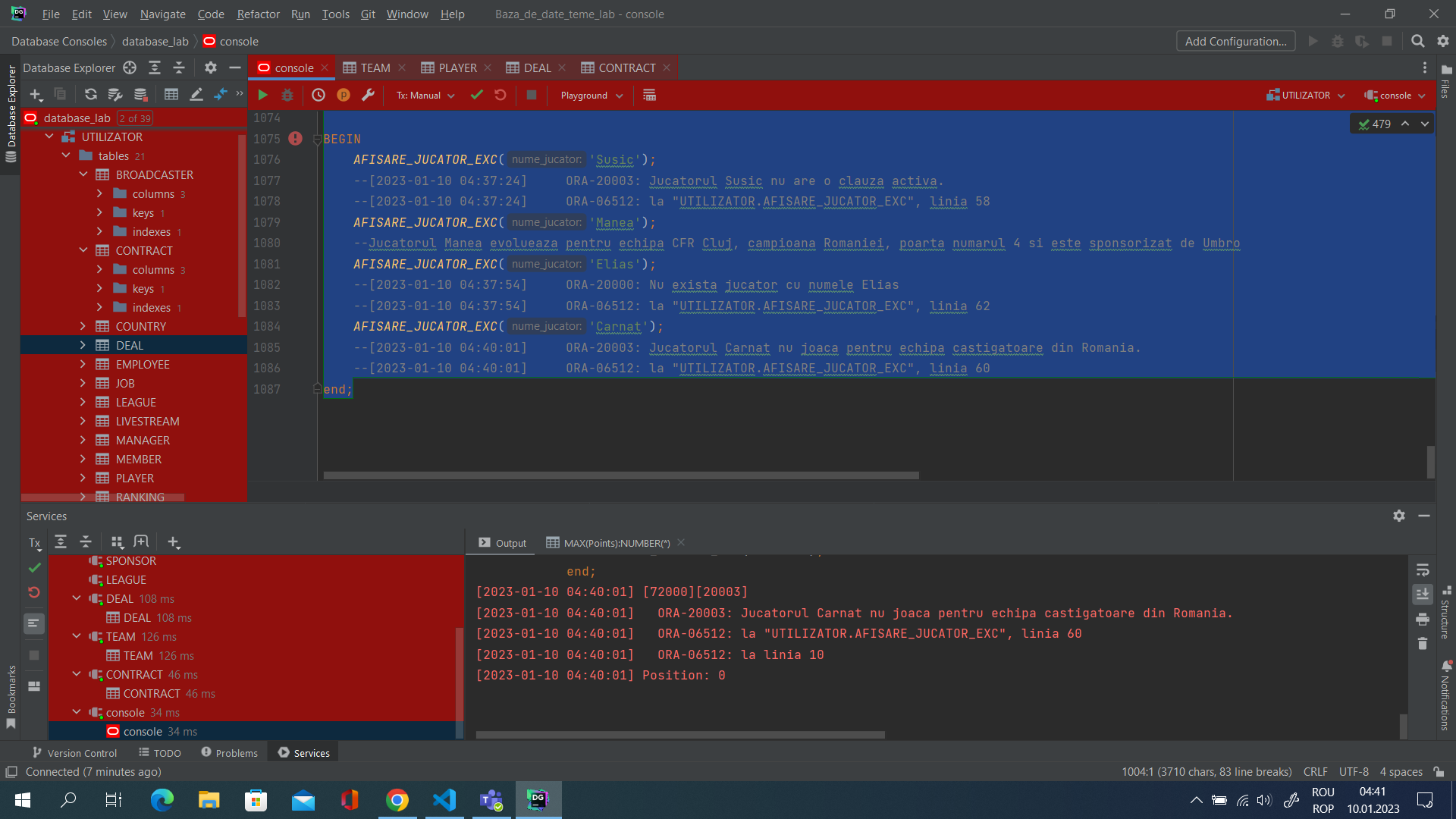
--[2023-01-10 04:37:54] ORA-06512: la "UTILIZATOR.AFISARE\_JUCATOR\_EXC", linia 62

*AFISARE\_JUCATOR\_EXC*('Carnat');

--[2023-01-10 04:40:01] ORA-20003: Jucatorul Carnat nu joaca pentru echipa castigatoare din Romania.

--[2023-01-10 04:40:01] ORA-06512: la "UTILIZATOR.AFISARE\_JUCATOR\_EXC", linia 60

end;

****

1. **Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.**

**--Presupunem ca fiecare ligă poate fi transmisă pe 2 canale de televiziune diferite(concurente). Creati un trigger de tip LMD la nivel de comanda care sa nu permita inserarea a mai multe canale de televiziune decât aceasta limita permisă. Declansati trigger-ul.**

CREATE OR REPLACE TRIGGER trig\_canale\_tv

BEFORE INSERT ON Broadcaster

DECLARE

nr NUMBER;

nr\_ligi NUMBER;

BEGIN

SELECT *COUNT*(\*) INTO nr FROM Broadcaster;

SELECT *COUNT*(\*) INTO nr\_ligi from League;

nr\_ligi := nr\_ligi \* 2;

IF nr >= nr\_ligi THEN

*RAISE\_APPLICATION\_ERROR*(-20001, 'Nu se pot insera mai mult de ' || nr\_ligi || ' canale de televiziune.');

END IF;

END;

BEGIN

DELETE FROM Broadcaster WHERE BroadcasterID >= 7 and BroadcasterID <= 11;

INSERT INTO Broadcaster VALUES(7, 'GSP TV', 'Teodor Raducanu');

INSERT INTO Broadcaster VALUES(8, 'Antena 1', 'Mihai Gadea');

INSERT INTO Broadcaster VALUES(9, 'Dolce Sport', 'Pavel Pascu');

INSERT INTO Broadcaster VALUES(10, 'Sport.ro', 'Ioana Cozma');

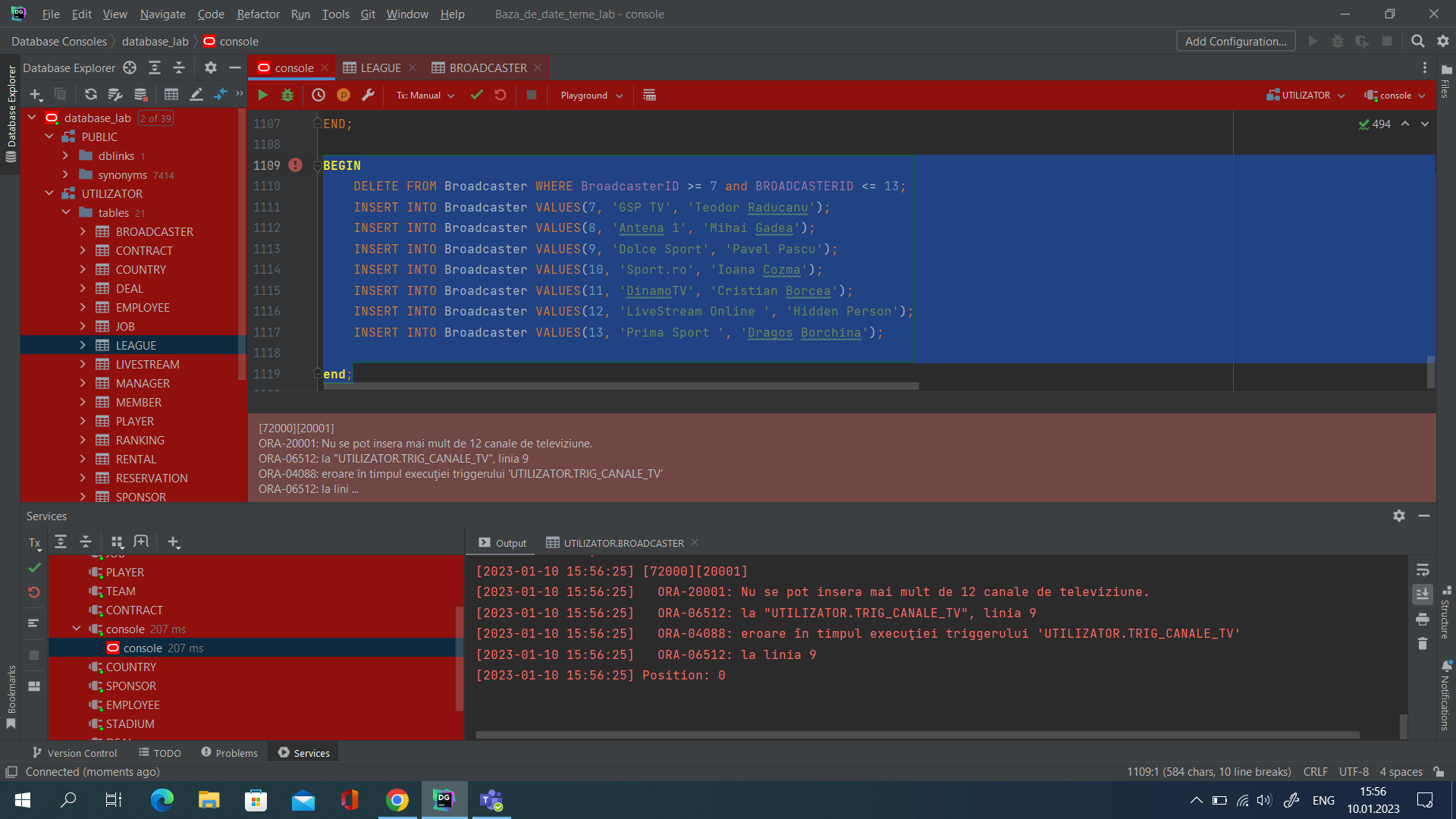
INSERT INTO Broadcaster VALUES(11, 'DinamoTV', 'Cristian Borcea');

INSERT INTO Broadcaster VALUES(12, 'LiveStream Online ', 'Hidden Person');

INSERT INTO Broadcaster VALUES(13, 'Prima Sport ', 'Dragos Borchina');

--[2023-01-10 15:37:16] ORA-20001: Nu se pot insera mai mult de 12 canale de televiziune.

end;



1. **Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.**

**--A început perioada de transferuri. Creati un trigger de tip LMD la nivel de linie care să verifice și să actualizeze detaliile jucătorilor care au fost transferați, ținând cont de toate cazurile posibile (insert, update, delete). Declanșați trigger-ul. Pentru a evita erori de tip Mutating Table, creați o vizualizare unde veți face toate modificările.**

CREATE OR REPLACE VIEW view\_jucatori AS

select p.PlayerID, p.TeamID, p.CountryID, p.PlayerName, p.PlayerPosition, p.JerseyNumber, d.SponsorID, d.ContractID

from Player p, Deal d

where p.PlayerID = d.PlayerID;

CREATE OR REPLACE TRIGGER transfer\_jucator

INSTEAD OF INSERT OR UPDATE OR DELETE ON view\_jucatori

FOR EACH ROW

DECLARE

exista\_deja\_numar NUMBER;

exceptie\_numar EXCEPTION;

exceptie\_numar\_invalid EXCEPTION;

exceptie\_prea\_multi\_jucatori EXCEPTION;

exceptie\_nu\_exista EXCEPTION;

nr\_jucatori NUMBER;

nume\_echipa\_noua Team.TeamName%TYPE;

exista\_jucator NUMBER;

BEGIN

IF INSERTING or UPDATING THEN

if :NEW.JerseyNumber > 99 or :NEW.JerseyNumber < 1 then

raise exceptie\_numar\_invalid;

end if;

--verificare numar jucatori echipa noua

select *count*(\*) into nr\_jucatori from view\_jucatori where TeamID = :NEW.TeamID;

select TeamName into nume\_echipa\_noua from Team where TeamID = :NEW.TeamID;

if nr\_jucatori >= 27 then

raise exceptie\_prea\_multi\_jucatori;

end if;

exista\_deja\_numar := 0;

select *count*(\*) into exista\_deja\_numar

from view\_jucatori where TeamID = :NEW.TeamID and JerseyNumber = :NEW.JerseyNumber;

if exista\_deja\_numar > 0 then

raise exceptie\_numar;

end if;

if INSERTING THEN

INSERT INTO Player VALUES(:NEW.PlayerID, :NEW.TeamID, :NEW.CountryID, :NEW.PlayerName, :NEW.PlayerPosition, :NEW.JerseyNumber);

INSERT INTO Deal VALUES(:NEW.PlayerID, :NEW.SponsorID, :NEW.ContractID);

elsif UPDATING THEN

UPDATE Player SET TeamID = :NEW.TeamID

where PlayerID = :NEW.PlayerID;

end if;

elsif DELETING then

select *count*(\*) into exista\_jucator from view\_jucatori where PlayerID = :OLD.PlayerID;

if exista\_jucator = 0 then

raise exceptie\_nu\_exista;

end if;

DELETE FROM Deal WHERE PlayerID = :OLD.PlayerID;

DELETE FROM Player WHERE PlayerID = :OLD.PlayerID;

end if;

EXCEPTION

WHEN exceptie\_numar THEN

*RAISE\_APPLICATION\_ERROR*(-20003, 'Jucatorul ' || :NEW.PlayerName || ' nu poate purta numarul ' || :NEW.JerseyNumber || ' pentru ca echipa ' || nume\_echipa\_noua || ' are deja un jucator cu acest numar.');

WHEN exceptie\_numar\_invalid THEN

*RAISE\_APPLICATION\_ERROR*(-20003, 'Jucatorul ' || :NEW.PlayerName || ' nu are un numar de tricou valid.');

WHEN exceptie\_prea\_multi\_jucatori THEN

*RAISE\_APPLICATION\_ERROR*(-20003, 'Echipa ' || nume\_echipa\_noua || ' are deja 27 de jucatori.');

WHEN exceptie\_nu\_exista THEN

*RAISE\_APPLICATION\_ERROR*(-20003, 'Jucatorul cu ID-ul ' || :OLD.PlayerID || ' nu exista.');

WHEN OTHERS THEN

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

end;

--Declansare trigger:

--Se transfera un jucator care vrea sa poarte un numar deja existent la CFR CLUJ.

insert into view\_jucatori values (139, 1, 20, 'Jefte', 'ATT', 9, 1, 1);

--[2023-01-10 16:32:30] ORA-20003: Jucatorul Jefte nu poate purta numarul 9 pentru ca echipa CFR Cluj are deja un jucator cu acest numar.

--Se transfera un jucator cu un numar invalid

insert into view\_jucatori values (139, 1, 20, 'Jefte', 'ATT', 122, 1, 1);

--[2023-01-10 16:33:11] ORA-20003: Jucatorul Jefte nu are un numar de tricou valid.

--Se transfera un jucator, dar echipa are deja 27 de jucatori

insert into view\_jucatori values (139, 4, 20, 'Iglesias', 'MID', 88, 1, 1);

--[2023-01-10 16:37:21] ORA-20003: Echipa FC Dinamo Bucuresti are deja 27 de jucatori.

--Se transfera un jucator nou

insert into view\_jucatori values (139, 3, 20, 'Iglesias', 'MID', 88, 1, 1);

--Stergere jucator

delete from view\_jucatori where PlayerID = 139;

--UTILIZATOR> delete from view\_jucatori where PlayerID = 139

--[2023-01-11 17:50:56] 1 row affected in 12 ms

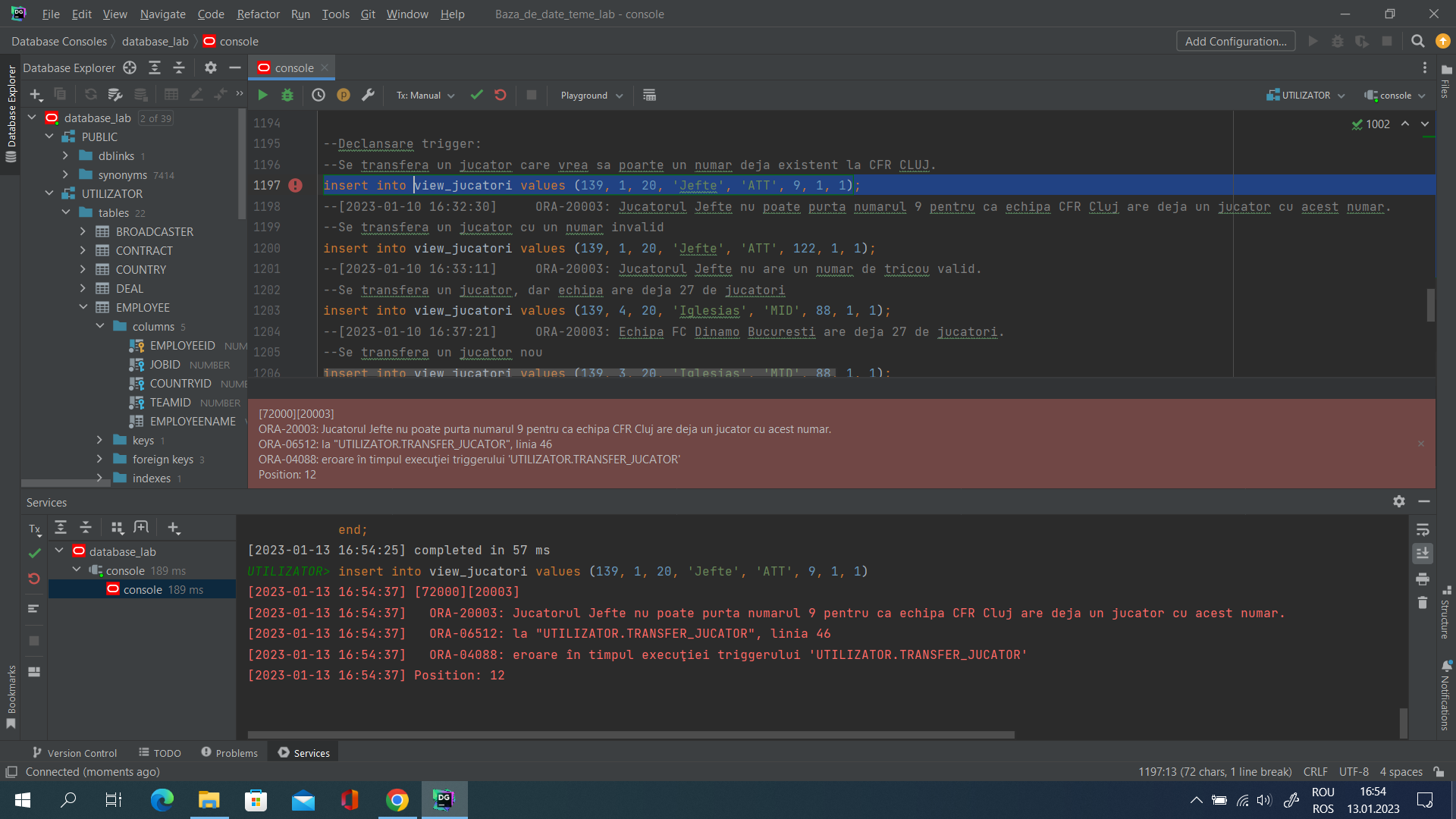
--Un jucator deja existent se transfera la alta echipa

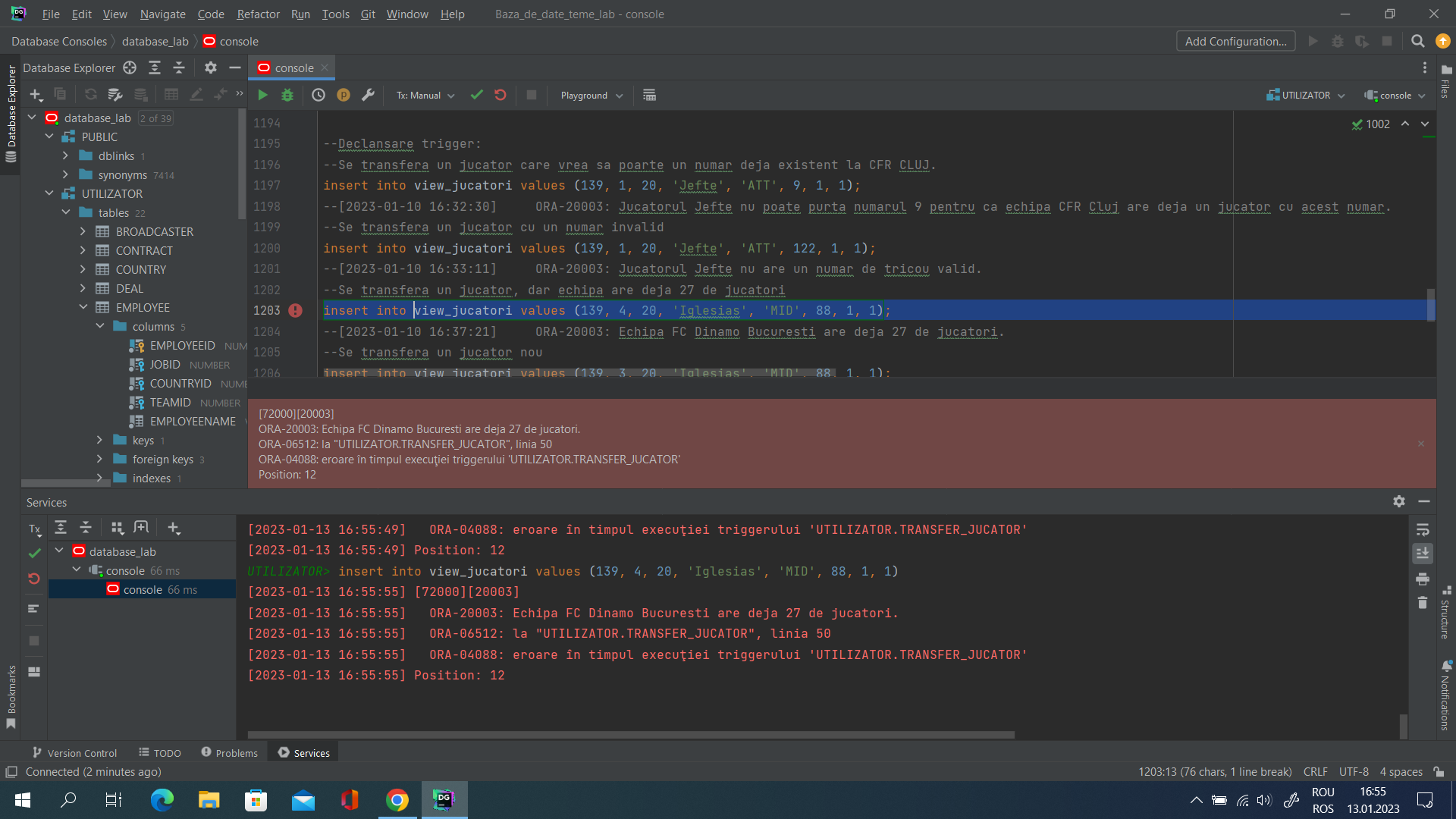
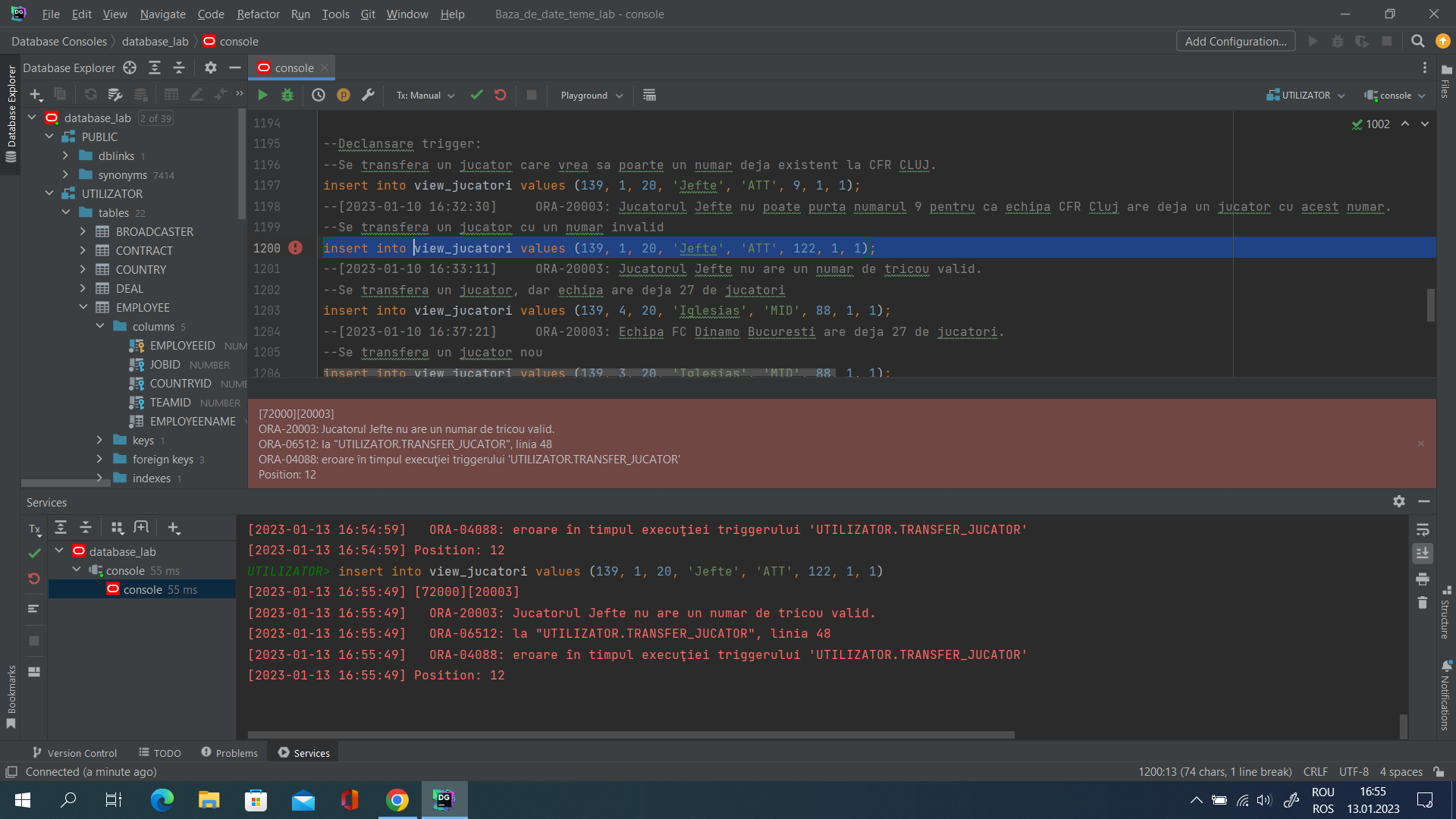
update view\_jucatori set TeamID = 1 where PlayerID = 139;

--UTILIZATOR> update view\_jucatori set TeamID = 1 where PlayerID = 139

--[2023-01-11 18:02:26] 1 row affected in 18 ms

****

****

****

1. **Definiți un trigger de tip LDD. Declanșați trigger-ul.**

**--Definiți un trigger de tip LDD care să permită modificarea bazei de date doar dacă utilizatorul curent este administratorul bazei de date(UTILIZATOR). Salvați modificările făcute într-o tabelă nouă.**

CREATE TABLE info\_admin

(

username VARCHAR2(30),

nume\_bd VARCHAR2(50),

data\_modificare TIMESTAMP(6),

comanda VARCHAR2(1000),

eroare VARCHAR2(1000)

);

CREATE OR REPLACE TRIGGER trigger\_admin

BEFORE CREATE OR ALTER OR DROP ON SCHEMA

DECLARE

comanda\_aux VARCHAR2(1000);

BEGIN

--in comanda\_aux vreau sa am comanda care a declansat trigger-ul

comanda\_aux := 'alter table Player drop column Salary;';

if *USER* != 'UTILIZATOR' then

insert into info\_admin values (*USER*, *SYS\_CONTEXT*('USERENV', 'DB\_NAME'), *SYSDATE*, comanda\_aux, 'Nu aveti drepturi de administrator.');

*raise\_application\_error*(-20000, 'Nu aveti drepturi de administrator.');

end if;

insert into info\_admin values (*USER*, *SYS\_CONTEXT*('USERENV', 'DB\_NAME'), *SYSDATE*, comanda\_aux, 'OK');

end;

--Declansare trigger:

alter table Player add Salary number(10,2);

--Acum vreau sa sterg coloana asta adaugata

alter table Player drop column Salary;



1. **Definiți un pachet care sa conțină toate obiectele definite în cadrul proiectului.**

CREATE OR REPLACE PACKAGE proiect\_sgbd\_evs AS

PROCEDURE *afisare\_coechipieri*(id\_jucator\_tast Player.PlayerID%TYPE);

FUNCTION *filtrare\_jucatori\_dupa\_numar*(numar\_echipa Player.TeamID%TYPE, numar\_tricou Player.JerseyNumber%TYPE) RETURN NUMBER;

PROCEDURE *afisare\_detalii\_echipe*;

FUNCTION *stadion\_jucator*(nume\_jucator Player.PlayerName%TYPE) RETURN Stadium.StadiumName%TYPE;

PROCEDURE *afisare\_jucator\_exc*(nume\_jucator Player.PlayerName%TYPE);

END proiect\_sgbd\_evs;

CREATE OR REPLACE PACKAGE BODY proiect\_sgbd\_evs AS

--Exercitiul 6

--Pentru un jucator de fotbal, caruia i se va da ID-ul de la tastatura, salvati si afisati

--numele tuturor coechipierilor sai, iar pentru fiecare coechipier sa se salveze si sponsorul acestuia.

procedure *afisare\_coechipieri*

(id\_jucator\_tast Player.PlayerID%type)

AS

TYPE tablou\_indexat is TABLE OF Player.PlayerName%type INDEX BY PLS\_INTEGER;

nume\_coechipieri tablou\_indexat;

TYPE tablou\_imbricat is TABLE OF Sponsor.SponsorName%type;

sponsor\_jucatori tablou\_imbricat := tablou\_imbricat();

i NUMBER;

id\_echipa\_jucator Player.TeamID%type;

id\_echipa\_coechiper Player.TeamID%type;

nume\_jucator Player.PlayerName%type;

id\_coechipier Player.PlayerID%type;

nume\_coechipier Player.PlayerName%type;

nume\_sponsor Sponsor.SponsorName%type;

numar\_jucatori NUMBER;

id\_jucator Player.PlayerID%type;

BEGIN

i := 0;

select TeamID, PlayerName into id\_echipa\_jucator, nume\_jucator from Player where PlayerID = id\_jucator\_tast;

select *count*(\*) into numar\_jucatori from Player;

id\_jucator := 1;

while id\_jucator <= numar\_jucatori

loop

select TeamID into id\_echipa\_coechiper from Player where PlayerID = id\_jucator;

if id\_echipa\_coechiper = id\_echipa\_jucator and id\_jucator\_tast <> id\_jucator then

i := i + 1;

select PlayerName, PlayerID into nume\_coechipier, id\_coechipier from Player where PlayerID = id\_jucator;

nume\_coechipieri(i) := nume\_coechipier;

sponsor\_jucatori.extend;

select SponsorName into nume\_sponsor from Sponsor where SponsorID = (select SponsorID from Deal where PlayerID = id\_coechipier);

sponsor\_jucatori(i) := nume\_sponsor;

end if;

id\_jucator := id\_jucator + 1;

end loop;

DBMS\_OUTPUT.*PUT\_LINE*('Jucatorul ' || nume\_jucator || ' are urmatorii coechipieri:');

FOR j IN 1..nume\_coechipieri.COUNT LOOP

DBMS\_OUTPUT.*PUT\_LINE*(nume\_coechipieri(j) || ' - ' || sponsor\_jucatori(j));

END LOOP;

END;

--Exercitiul 7

--Sa se afiseze pentru fiecare echipa numarul de jucatori. In cazul in

--care echipa nu are jucatori inscrisi, se va afisa un mesaj corespunzator.

--Pentru fiecare jucator cu numarul de pe tricou mai mare decat un parametru dat,

--sa se afiseze pozitia, numarul de pe tricou si tara de provenienta.

FUNCTION *filtrare\_jucatori\_dupa\_numar*

(numar\_echipa Player.TeamID%TYPE, numar\_tricou Player.JerseyNumber%TYPE)

RETURN NUMBER is

nr\_jucatori\_filtrati NUMBER;

BEGIN

select *count*(\*) into nr\_jucatori\_filtrati

from Player

where JerseyNumber > numar\_tricou and TeamID = numar\_echipa;

return nr\_jucatori\_filtrati;

end;

PROCEDURE *afisare\_detalii\_echipe* AS

CURSOR c1 is

SELECT t.TeamID id\_echipa, TeamName nume\_echipa, *COUNT*(PlayerID) numar\_jucatori

FROM Team t, Player p

WHERE t.TeamID = p.TeamID(+)

GROUP BY t.TeamID, TeamName;

CURSOR c2(parametru NUMBER, echipa\_id Team.TeamID%TYPE) is

SELECT PlayerName nume\_jucator, PlayerPosition pozitie\_jucator, CountryName tara\_jucator, JerseyNumber numar\_jucator

FROM Player p, Country c

WHERE p.CountryID = c.CountryID(+) and JerseyNumber > parametru and TeamID = echipa\_id;

TYPE tip\_detalii\_echipa IS RECORD(

id\_echipa Team.TeamID%TYPE,

nume\_echipa Team.TeamName%TYPE,

numar\_jucatori NUMBER

);

detalii\_echipa tip\_detalii\_echipa;

count\_echipe NUMBER;

count\_jucatori NUMBER;

numar\_jucator\_tastatura NUMBER := &numar\_tricou;

nr\_jucatori\_filtrati NUMBER;

BEGIN

DBMS\_OUTPUT.*PUT\_LINE*('Echipe si jucatori: ');

count\_echipe := 0;

OPEN c1;

LOOP

FETCH c1 INTO detalii\_echipa;

count\_echipe := count\_echipe + 1;

EXIT WHEN c1%NOTFOUND;

IF detalii\_echipa.numar\_jucatori = 0 THEN

DBMS\_OUTPUT.*PUT\_LINE*(count\_echipe || '. ' || detalii\_echipa.nume\_echipa || ' nu are niciun jucator semnat.');

ELSE

nr\_jucatori\_filtrati := *filtrare\_jucatori\_dupa\_numar*(detalii\_echipa.id\_echipa, numar\_jucator\_tastatura);

DBMS\_OUTPUT.*PUT\_LINE*(count\_echipe || '. ' || detalii\_echipa.nume\_echipa || ' are ' || detalii\_echipa.numar\_jucatori || ' jucatori semnati iar cei ' || nr\_jucatori\_filtrati || ' cu numarul mai mare decat ' || numar\_jucator\_tastatura || ' sunt: ');

count\_jucatori := 0;

FOR i in c2(numar\_jucator\_tastatura, detalii\_echipa.id\_echipa) LOOP

count\_jucatori := count\_jucatori + 1;

DBMS\_OUTPUT.*PUT\_LINE*(count\_jucatori || '. ' || i.nume\_jucator || ', numarul ' || i.numar\_jucator || ', joaca pe pozitia ' || i.pozitie\_jucator || ' si este din ' || i.tara\_jucator);

end loop;

END IF;

end loop;

end;

--Exercitiul 8

--Scrie o functie care va afisa stadionul pe care evolueaza echipa unui jucator, al carui nume va fi dat de la tastatura.

FUNCTION *stadion\_jucator*

(nume\_jucator Player.PlayerName%TYPE)

RETURN Stadium.StadiumName%TYPE IS

id\_jucator Player.PlayerID%TYPE;

nume\_stadion Stadium.StadiumName%TYPE;

BEGIN

select PlayerID

into id\_jucator

from Player

where PlayerName = nume\_jucator;

select StadiumName

into nume\_stadion

from Stadium s

join Team t on t.TeamID = s.TeamID

join Player p on p.TeamID = t.TeamID

where p.PlayerID = id\_jucator;

return nume\_stadion;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

*RAISE\_APPLICATION\_ERROR*(-20000, 'Nu exista jucator cu numele ' || nume\_jucator);

WHEN OTHERS THEN

*RAISE\_APPLICATION\_ERROR*(-20002, 'Alta eroare');

end;

--Exercitiul 9

--Sa se afiseze numele, echipa, numarul de pe tricou si sponsorul unui jucator dat ca parametru, doar daca

--acesta a terminat pe locul 1 in campionatul din Romania si daca are o clauza de reziliere activa in contractul cu sponsorii sai.

PROCEDURE *afisare\_jucator\_exc*

(nume\_jucator Player.PlayerName%TYPE)

IS

CURSOR c1 is

select t.TeamID id\_echipa, t.TeamName nume\_echipa, Points puncte

from Team t, Ranking r

where t.TeamID = r.TeamID;

TYPE tip\_detalii\_echipa IS RECORD (

id\_echipa Team.TeamID%TYPE,

nume\_echipa Team.TeamName%TYPE,

puncte\_echipa Ranking.Points%TYPE );

TYPE tip\_detalii\_jucator IS RECORD(

id\_jucator Player.PlayerID%TYPE,

nume\_jucator Player.PlayerName%TYPE,

echipa\_jucator Team.TeamName%TYPE,

numar\_jucator Player.JerseyNumber%TYPE,

sponsor\_jucator Sponsor.SponsorName%TYPE,

clauza\_reziliere Contract.ClauzaReziliere%TYPE);

jucator\_par tip\_detalii\_jucator;

echipa\_castigatoare tip\_detalii\_echipa;

nr\_maxim\_puncte Ranking.Points%TYPE;

exceptie\_clauza EXCEPTION;

exceptie\_echipa EXCEPTION;

BEGIN

select *MAX*(Points)

into nr\_maxim\_puncte

from Ranking;

open c1;

LOOP

FETCH c1 into echipa\_castigatoare;

EXIT WHEN echipa\_castigatoare.puncte\_echipa = nr\_maxim\_puncte;

end loop;

select p.PlayerID, p.PlayerName, t.TeamName, p.JerseyNumber, s.SponsorName, c.ClauzaReziliere

into jucator\_par

from Player p

join Team t on (t.TeamID = p.TeamID)

join Deal d on (d.PlayerID = p.PlayerID)

join Contract c on (c.ContractID = d.ContractID)

join Sponsor s on (s.SponsorID = d.SponsorID)

where p.PlayerName = nume\_jucator;

if jucator\_par.clauza\_reziliere = 0 then

raise exceptie\_clauza;

end if;

if jucator\_par.echipa\_jucator <> echipa\_castigatoare.nume\_echipa then

raise exceptie\_echipa;

end if;

DBMS\_OUTPUT.*PUT\_LINE*('Jucatorul ' || jucator\_par.nume\_jucator || ' evolueaza pentru echipa ' || jucator\_par.echipa\_jucator || ', campioana Romaniei, poarta numarul ' || jucator\_par.numar\_jucator || ' si este sponsorizat de ' || jucator\_par.sponsor\_jucator);

EXCEPTION

WHEN exceptie\_clauza THEN

*RAISE\_APPLICATION\_ERROR*(-20003, 'Jucatorul ' || nume\_jucator || ' nu are o clauza activa.');

WHEN exceptie\_echipa THEN

*RAISE\_APPLICATION\_ERROR*(-20003, 'Jucatorul ' || nume\_jucator || ' nu joaca pentru echipa castigatoare din Romania.');

WHEN NO\_DATA\_FOUND THEN

*RAISE\_APPLICATION\_ERROR*(-20000, 'Nu exista jucator cu numele ' || nume\_jucator);

WHEN TOO\_MANY\_ROWS THEN

*RAISE\_APPLICATION\_ERROR*(-20001, 'Sunt mai multi jucatori cu numele ' || nume\_jucator);

WHEN OTHERS THEN

*RAISE\_APPLICATION\_ERROR*(-20002, 'Alta eroare');

end;

END proiect\_sgbd\_evs;

**--Testare pachet**

**begin**

**--Exercitiul 6**

**proiect\_sgbd\_evs.*afisare\_coechipieri*(1);**

**--Exercitiul 7**

**proiect\_sgbd\_evs.*afisare\_detalii\_echipe*();**

**--Exercitiul 8**

**DBMS\_OUTPUT.*PUT\_LINE*('Jucatorul cu numele Sefer evolueaza pe ' || proiect\_sgbd\_evs.*stadion\_jucator*('Sefer'));**

**--[2023-01-10 03:17:42] completed in 15 ms**

**--Jucatorul cu numele Sefer evolueaza pe stadionul Stadionul Giulesti**

**DBMS\_OUTPUT.*PUT\_LINE*('Jucatorul cu numele Elias evolueaza pe ' || proiect\_sgbd\_evs.*stadion\_jucator*('Elias'));**

**--[2023-01-10 03:18:44] ORA-20000: Nu exista jucator cu numele Elias**

**--[2023-01-10 03:18:44] ORA-06512: la "UTILIZATOR.STADION\_JUCATOR", linia 23**

**--Exercitiul 9**

**proiect\_sgbd\_evs.*AFISARE\_JUCATOR\_EXC*('Susic');**

**--[2023-01-10 04:37:24] ORA-20003: Jucatorul Susic nu are o clauza activa.**

**--[2023-01-10 04:37:24] ORA-06512: la "UTILIZATOR.AFISARE\_JUCATOR\_EXC", linia 58**

**proiect\_sgbd\_evs.*AFISARE\_JUCATOR\_EXC*('Manea');**

**--Jucatorul Manea evolueaza pentru echipa CFR Cluj, campioana Romaniei, poarta numarul 4 si este sponsorizat de Umbro**

**proiect\_sgbd\_evs.*AFISARE\_JUCATOR\_EXC*('Elias');**

**--[2023-01-10 04:37:54] ORA-20000: Nu exista jucator cu numele Elias**

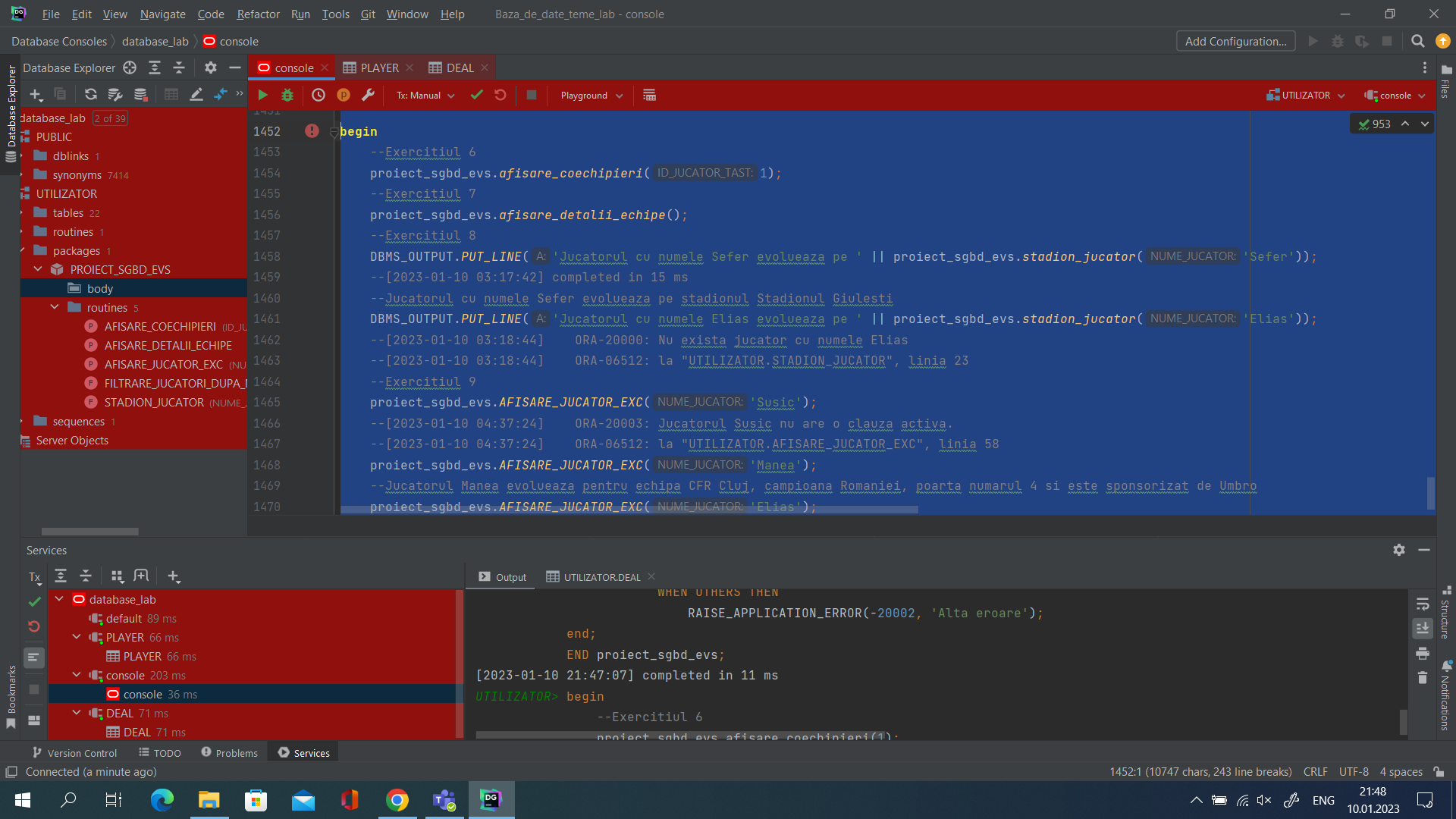
**--[2023-01-10 04:37:54] ORA-06512: la "UTILIZATOR.AFISARE\_JUCATOR\_EXC", linia 62**

**proiect\_sgbd\_evs.*AFISARE\_JUCATOR\_EXC*('Carnat');**

**--[2023-01-10 04:40:01] ORA-20003: Jucatorul Carnat nu joaca pentru echipa castigatoare din Romania.**

**--[2023-01-10 04:40:01] ORA-06512: la "UTILIZATOR.AFISARE\_JUCATOR\_EXC", linia 60**

**end;**

****

1. **Definiți un pachet care sa includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite.**

**--Pentru fiecare echipă, obțineți numele precum și lista numelor angajaților care își desfășoară activitatea în cadrul acesteia, și calculați salariul mediu pentru fiecare echipă. Pentru fiecare angajat se va afișa și funcția acestuia și cati colegi din același departament și aceeași echipă are.**

CREATE OR REPLACE PACKAGE proiect\_sgbd\_evs2 AS

TYPE tablou\_imbricat is TABLE OF VARCHAR2(100);

TYPE echipa\_record IS RECORD (

id\_echipa Team.TeamID%TYPE,

nume\_echipa Team.TeamName%TYPE);

CURSOR c\_echipa RETURN echipa\_record;

FUNCTION *get\_salariu*(id\_angajat Employee.EmployeeID%TYPE) RETURN NUMBER;

FUNCTION *functii\_angajati* RETURN tablou\_imbricat;

FUNCTION *salariu\_mediu\_echipa*(nume\_echipa Team.TeamName%TYPE) RETURN NUMBER;

FUNCTION *afisare\_nr\_colegi\_angajat*(id\_angajat Employee.EmployeeID%TYPE) RETURN NUMBER;

PROCEDURE *afisare\_detalii\_echipe*;

END proiect\_sgbd\_evs2;

CREATE OR REPLACE PACKAGE BODY proiect\_sgbd\_evs2 AS

CURSOR c\_echipa RETURN echipa\_record IS

SELECT Team.TeamID, Team.TeamName

FROM Team;

FUNCTION *get\_salariu*(id\_angajat Employee.EmployeeID%TYPE) RETURN NUMBER IS

salariu NUMBER;

BEGIN

select JobSalary

into salariu

from Job

where JobID = (select JobID

from Employee

where EmployeeID = id\_angajat);

RETURN salariu;

END *get\_salariu*;

FUNCTION *functii\_angajati* RETURN tablou\_imbricat IS

v\_tablou tablou\_imbricat := tablou\_imbricat();

nr\_angajati NUMBER;

BEGIN

SELECT *COUNT*(\*)

INTO nr\_angajati

FROM Employee;

FOR id\_angajat IN 1..nr\_angajati

LOOP

v\_tablou.extend;

SELECT Job.JobName

INTO v\_tablou(id\_angajat)

FROM Employee, Job

WHERE Employee.JobID = Job.JobID AND Employee.EmployeeID = id\_angajat;

END LOOP;

RETURN v\_tablou;

END *functii\_angajati*;

FUNCTION *salariu\_mediu\_echipa*(nume\_echipa Team.TeamName%TYPE)

RETURN NUMBER IS

salariu\_mediu NUMBER;

BEGIN

SELECT *Round*(*AVG*(JobSalary), 2)

INTO salariu\_mediu

FROM Employee, Job, Team

WHERE Employee.JobID = Job.JobID AND Employee.TeamID = Team.TeamID AND TeamName = nume\_echipa;

RETURN salariu\_mediu;

END *salariu\_mediu\_echipa*;

FUNCTION *afisare\_nr\_colegi\_angajat*(id\_angajat Employee.EmployeeID%TYPE)

RETURN NUMBER IS

nr\_colegi NUMBER;

TYPE angajat\_record is RECORD(

id\_angajat Employee.EmployeeName%TYPE,

id\_job\_angajat Job.JobID%TYPE,

id\_echipa\_angajat Team.TeamID%TYPE

);

angajatul\_nostru angajat\_record;

BEGIN

SELECT Employee.EmployeeName, Employee.JobID, Employee.TeamID

INTO angajatul\_nostru

FROM Employee

WHERE Employee.EmployeeID = id\_angajat;

SELECT *COUNT*(\*)

INTO nr\_colegi

FROM Employee

WHERE Employee.JobID = angajatul\_nostru.id\_job\_angajat AND Employee.TeamID = angajatul\_nostru.id\_echipa\_angajat;

RETURN nr\_colegi;

END *afisare\_nr\_colegi\_angajat*;

PROCEDURE *afisare\_detalii\_echipe* IS

nr\_angajati NUMBER;

i NUMBER;

j NUMBER;

vector\_functii\_angajati tablou\_imbricat := tablou\_imbricat();

record\_echipa echipa\_record;

BEGIN

vector\_functii\_angajati := *functii\_angajati*;

OPEN c\_echipa;

i := 0;

LOOP

i := i + 1;

FETCH c\_echipa INTO record\_echipa;

EXIT WHEN c\_echipa%NOTFOUND;

SELECT *COUNT*(\*) INTO nr\_angajati FROM Employee WHERE TeamID = record\_echipa.id\_echipa;

DBMS\_OUTPUT.*PUT\_LINE*('----------------------------------------------------------------------');

if nr\_angajati > 0 then

DBMS\_OUTPUT.*PUT\_LINE*(i || '. Echipa ' || record\_echipa.nume\_echipa || ' are ' || nr\_angajati || ' angajati si salariul mediu este ' || *salariu\_mediu\_echipa*(record\_echipa.nume\_echipa));

--AFISARE ANGAJATI

j := 0;

FOR angajat in (SELECT EmployeeID, EmployeeName FROM Employee WHERE TeamID = record\_echipa.id\_echipa)

LOOP

j := j + 1;

DBMS\_OUTPUT.*PUT\_LINE*(j || '. ' || angajat.EmployeeName || ' este ' || vector\_functii\_angajati(angajat.EmployeeID) || ', castiga ' || *get\_salariu*(angajat.EmployeeID) || ' si are ' || *afisare\_nr\_colegi\_angajat*(angajat.EmployeeID) || ' colegi din acelasi departament.');

END LOOP;

else

DBMS\_OUTPUT.*PUT\_LINE*(i || '. Echipa ' || record\_echipa.nume\_echipa || ' nu are angajati');

end if;

end loop;

END *afisare\_detalii\_echipe*;

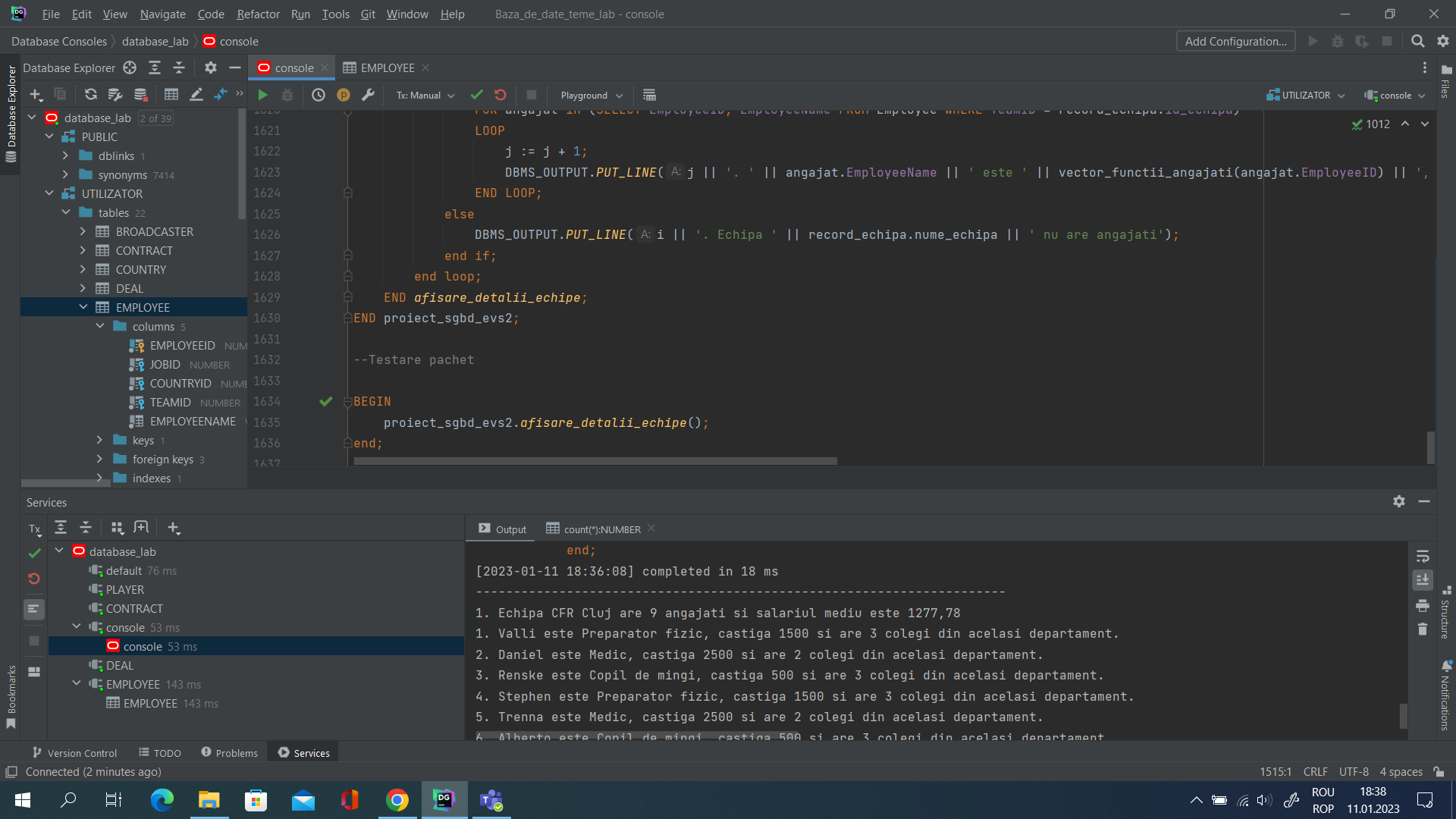
END proiect\_sgbd\_evs2;

--Testare pachet

BEGIN

proiect\_sgbd\_evs2.*afisare\_detalii\_echipe*();

end;

****