Nume: Mocanu Radu
Grupa: 244

Proiect SGBD

Gestionarea şi administrarea Turneelor de Grand Slam și a jucătorilor participanţi

1. Prezentați pe scurt baza de date și utilitatea ei:

Pentru acest proiect, am ales realizarea bazei de date a unor turnee de Grand Slam, precum ,,Australian_Open”, ,,US_Open” sau ,,Wimbledon”.

Baza de date contine informatii cu privire la orasul si tara in care se desfasoara turneul, cum sunt impartite rundele acestuia, inregistrandu-se rezultatele participantilor, detaliile despre complexul de terenuri unde se va desfasura turneul, precum si despre fiecare teren in parte.

Fiecare turneu va avea o cupa, careia i se va asocia o valoare a premiului, iar aceasta cupa poate fi castigata de un singur participant, ce va fi campionul turneului.

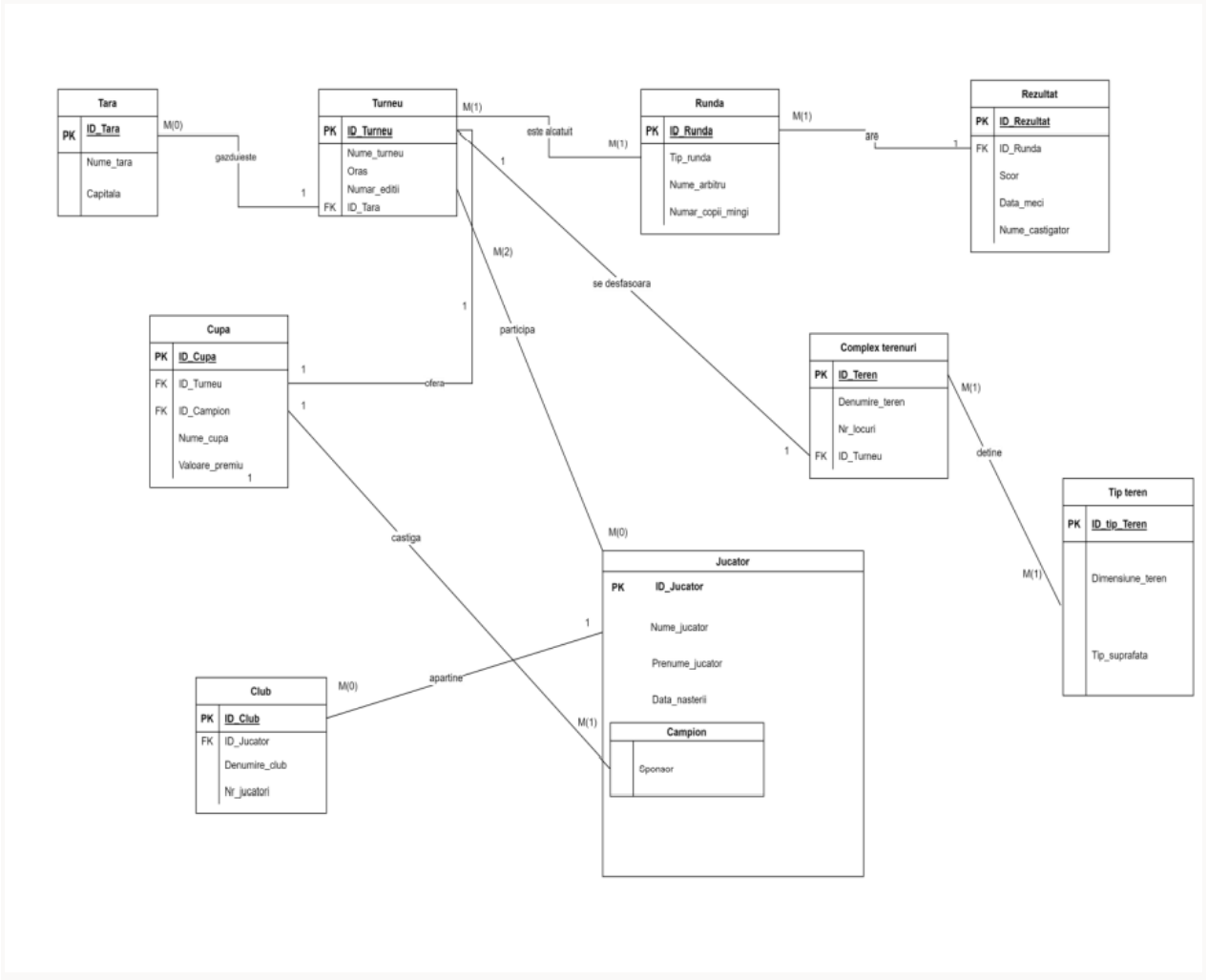
Baza de date retine pentru fiecare jucator in parte, daca este campion sau nu, numarul de campionate castigate si clubul unde se antreneaza.

Scopul crearii acestei baze de date este de a putea tine atat evidenta jucatorilor si a turneelor, precum si a altor detalii relevante administrative: numarul maxim de spectatori, tipul terenurilor de care dispune un complex, numarul copiiilor de mingi din cadrul fiecarei runde si asa mai departe.

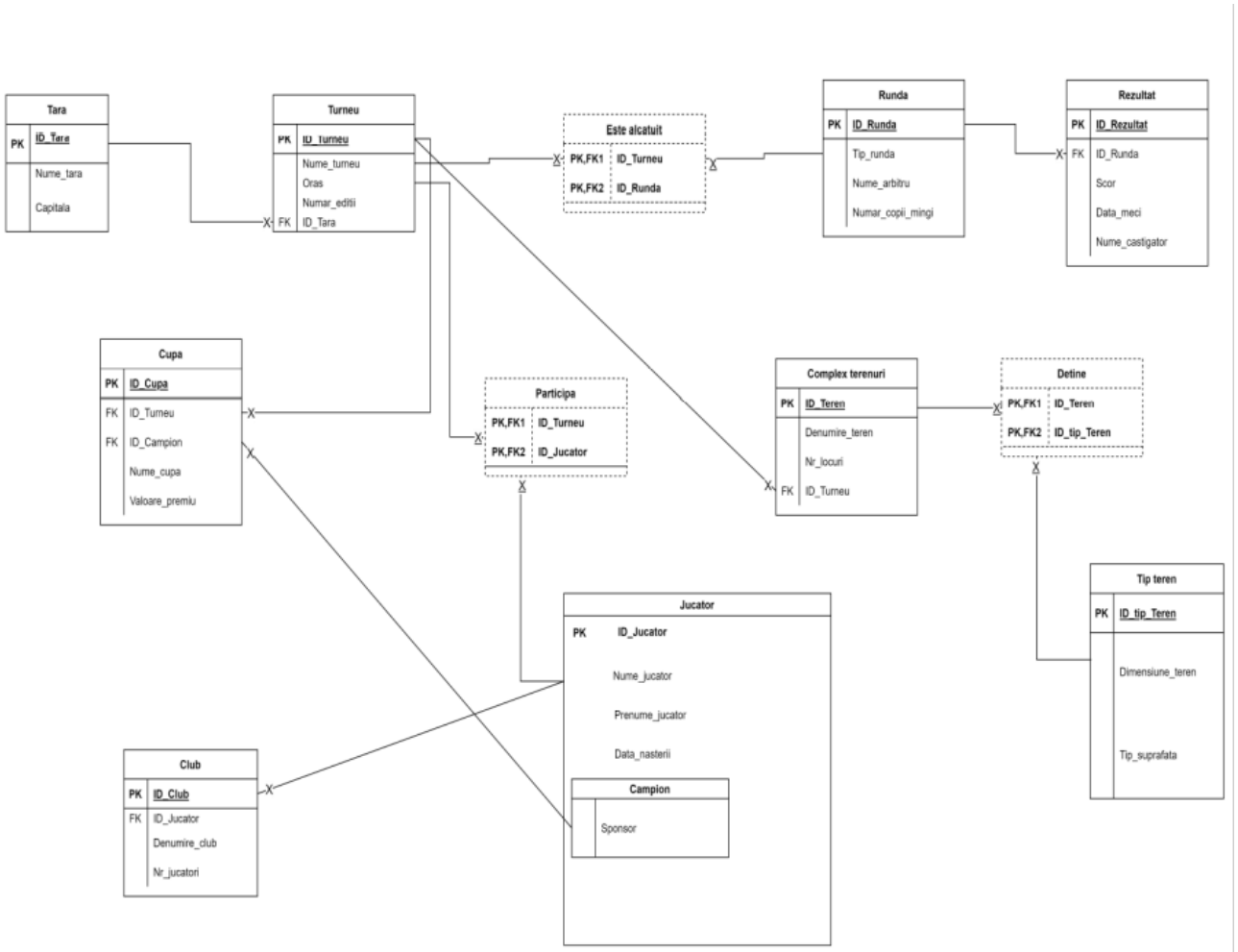
Fiecare turneu se desfasoara intr-o tara, este format din runde, fiecare runda avand un rezultat unic si un castigator.

De asemenea, fiecare turneu are o cupa. Cupa va fi primita de campionul turneului ce este un jucator care face parte dintr-un club

2. Realizitați diagrama entitae-relație (ERD):



3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare:



4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe, etc.):

Name	Type	Created
CLUB	Table	116 seconds ago
COMPLEX_TERENURI	Table	3 minutes ago
CUPA	Table	73 seconds ago
DETINE	Table	41 seconds ago
ESTE_ALCATUIT	Table	73 seconds ago
JUCATOR	Table	116 seconds ago
PARTICIPA	Table	41 seconds ago
REZULTAT	Table	3 minutes ago
RUNDA	Table	3 minutes ago
TARA	Table	3 minutes ago
TIP_TEREN	Table	116 seconds ago
TURNEU	Table	3 minutes ago

```
CREATE TABLE Tara(
ID_Tara int not null primary key,
Nume_tara varchar2(50),
Capitala varchar2(50)
);
CREATE TABLE Turneu(
ID_Turneu int not null primary key,
Nume_turneu varchar2(50),
Oras varchar2(50),
Numar_editii int,
ID_Tara not null,
foreign key(ID_Tara) references Tara(ID_Tara)
);

CREATE TABLE Runda(
```

```
ID_Runda int not null primary key,
Tip_runda varchar2(50),
Nume_arbitru varchar2(50),
Numar_copii_mingii int
);
CREATE TABLE Rezultat(
ID_Rezultat int not null primary key,
Scor varchar2(50),
Data_meci varchar2(50),
Nume_castigator varchar(50),
ID_Runda not null,
foreign key(ID_Runda) references Runda(ID_Runda)
);

CREATE TABLE Complex_terenuri(
ID_Teren int not null primary key,
Denumire_teren varchar2(50),
Numar_locuri varchar2(50),
ID_Turneu not null,
foreign key(ID_Turneu) references Turneu(ID_Turneu)
);

CREATE TABLE Tip_teren(
ID_Tip_teren int not null primary key,
Dimensiune_teren varchar2(50),
Tip_suprafata varchar2(50)
);
CREATE TABLE Jucator(
ID_Jucator int not null primary key,
Nume_jucator varchar2(50),
Prenume_jucator varchar2(50),
Data_nasterii varchar(50),
Tip varchar2(20) not null,
Sponsor varchar2(50)
);
CREATE TABLE Club(
ID_Club int not null primary key,
Denumire_club varchar2(50),
Numar_jucatori varchar2(50),
ID_Jucator not null,
foreign key(ID_Jucator) references Jucator(ID_Jucator)
);

CREATE TABLE Cupa(
ID_Cupa int not null primary key,
Nume_cupa varchar2(50),
Valoare_premiu varchar2(50),
ID_Turneu not null,
```

```
foreign key(ID_Turneu) references Turneu(ID_Turneu),
ID_Campion not null,
foreign key(ID_Campion) references Jucator(ID_Jucator)
);
CREATE TABLE Este_alcatuit(
ID_Turneu int not null,
ID_Runda int not null,
constraint pk_este_alcatuit primary key(ID_Turneu, ID_Runda)
);

CREATE TABLE Participa(
ID_Turneu int not null,
ID_Jucator int not null,
constraint pk_participa primary key(ID_Turneu, ID_Jucator)
);
CREATE TABLE Detine(
ID_Teren int not null,
ID_Tip_Teren int not null,
constraint pk_Detine primary key(ID_Teren, ID_Tip_Teren)
);
```

5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă):

```
insert into Tara values(1,'Austalia','Camberra');
insert into Tara values(2,'Franta','Paris');
insert into Tara values(3,'Romania','Bucuresti');
insert into Tara values(4,'SUA','Washington');
insert into Tara values(5,'UK','Londra');

insert into Turneu values(11,'Australian_Open','Melbourne','44',1);
insert into Turneu values(12,'Roland_Garros','Paris','65',2);
insert into Turneu values(13,'Bucharest_Open','Bucuresti','12',3);
insert into Turneu values(14,'US_Open','New_York','54',4);
insert into Turneu values(15,'Wimbledon','Londra','52',5);
insert into Runda values(21,'Optime','John_W','5');
insert into Runda values(22,'Sfert','Andrew_S','6');
insert into Runda values(23,'Semi_finala','Filip_P','4');
insert into Runda values(24,'Finala','Jo_S','8');
insert into Runda values(25,'Calificari','Simona_H','13');

insert into Rezultat values(31,'3-1','1.02.2022','Andrei',21);
insert into Rezultat values(32,'2-3','3.02.2022','Alex',22);
insert into Rezultat values(33,'3-0','4.02.2022','Horia',23);
insert into Rezultat values(34,'3-5','6.02.2022','Halep',24);
insert into Rezultat values(35,'3-1','26.01.2022','Vinchin',25);
insert into Complex_terenuri values(41,'Bernabeu','45000',11);
insert into Complex_terenuri values(42,'Camp_Nou','87000',12);
insert into Complex_terenuri values(43,'Omblemenco','32000',13);
insert into Complex_terenuri values(44,'Hinjust','45000',14);
insert into Complex_terenuri values(45,'National_Arena','94000',15);

insert into Tip_teren values(51,'80','zgura');
insert into Tip_teren values(52,'110','gazon');
insert into Tip_teren values(53,'130','hard');
insert into Tip_teren values(54,'45','zgura');
insert into Tip_teren values(55,'39','gazon');
CREATE SEQUENCE secventa
start with 56
```

```
increment by 1
minvalue 56
cache 60;
insert into Tip_teren values(secventa.nextval,'78','iarba');
insert into Tip_teren values(secventa.nextval,'65','hard');
insert into Tip_teren values(secventa.nextval,'57','iarba');
insert into Tip_teren values(secventa.nextval,'90','zgura');

insert into Jucator values(61,'Ionescu','Alex','2.02.1987','negativ',null);
insert into Jucator values(62,'Popescu','Andrei','12.03.1997','negativ',null);
insert into Jucator values(63,'Ionescu','Horia','23.05.1992','negativ',null);
insert into Jucator values(64,'Ionescu','Halep','3.11.2001','pozitiv','rexona');
insert into Jucator values(65,'Ionescu','Vinchin','5.07.2002','negativ',null);
insert into Club values(71,'FC_Invingatori',35,61);
insert into Club values(72,'Neomasters',43,62);
insert into Club values(73,'Ghindril',57,63);
insert into Club values(74,'CSS6',22,64);
insert into Club values(75,'Best',76,65);

insert into Cupa values(81,'Shining',10000,11,64);
insert into Cupa values(82,'Winners_Cup',40000,12,64);
insert into Cupa values(83,'Friendship_Cup',9300,13,64);
insert into Cupa values(84,'Juniors_Cup',10000,14,64);
insert into Cupa values(85,'Seniors_Cup',10000,15,64);
insert into Este_alcatuit values(11,21);
insert into Este_alcatuit values(11,22);
insert into Este_alcatuit values(11,23);
insert into Este_alcatuit values(11,24);
insert into Este_alcatuit values(11,25);

insert into Este_alcatuit values(12,21);
insert into Este_alcatuit values(12,22);
insert into Este_alcatuit values(12,23);
insert into Este_alcatuit values(12,24);
insert into Este_alcatuit values(12,25);
insert into Participa values(11,62);
insert into Participa values(11,63);
insert into Participa values(11,61);
insert into Participa values(11,64);
insert into Participa values(11,65);
insert into Participa values(12,61);
insert into Participa values(12,62);
insert into Participa values(12,63);
insert into Participa values(12,64);
insert into Participa values(12,65);

insert into Detine values(41,51);
insert into Detine values(41,52);
```

```
insert into Detine values(41,53);
insert into Detine values(41,54);
insert into Detine values(41,55);
insert into Detine values(42,51);
insert into Detine values(42,52);
insert into Detine values(42,53);
insert into Detine values(42,54);
insert into Detine values(42,55);
```

Selecturi cu tabelele populate:

```
---
609
610    select * from jucator;
```

ID_JUCATOR	NUME_JUCATOR	PRENUME_JUCATOR	DATA_NASTERII	TIP	SPONSOR
61	Ionescu	Alex	2.02.1987	negativ	-
62	Popescu	Andrei	12.03.1997	negativ	-
63	Ionescu	Horia	23.05.1992	negativ	-
64	Ionescu	Halep	3.11.2001	pozitiv	rexona
65	Ionescu	Vinchin	5.07.2002	negativ	-

```
612
613
614    select * from club;
```

ID_CLUB	DENUMIRE_CLUB	NUMAR_JUCATORI	ID_JUCATOR
71	FC_Invingatori	35	61
72	Neomasters	43	62
73	Ghindril	57	63
74	CSS6	22	64
75	Best	76	65

```
615
616
617    select * from tara;
```

ID_TARA	NUME_TARA	CAPITALA
1	Austalia	Camberra
2	Franta	Paris
3	Romania	Bucuresti
4	SUA	Washington
5	UK	Londra

```
618
619    select * from turneu;
```

ID_TURNEU	NUME_TURNEU	ORAS	NUMAR_EDITII	ID_TARA
11	Australian_Open	Melbourne	44	1
12	Roland_Garros	Paris	65	2
13	Bucharest_Open	Bucuresti	12	3
14	US_Open	New_York	54	4
15	Wimbledon	Londra	52	5

```
624
625    select * from complex_terenuri;
```

ID_TEREN	DENUMIRE_TEREN	NUMAR_LOCURI	ID_TURNEU
41	Bernabeu	45000	11
42	Camp_Nou	87000	12
43	Omblemenco	32000	13
44	Hinjust	45000	14
45	National_Arena	94000	15

```
619    select * from turneu;
620
621    select * from detine;
```

ID_TEREN	ID_TIP_TEREN
41	51
41	52
41	53
41	54
41	55
42	51
42	52
42	53
42	54
42	55

```
626
627    select * from cupa;
```

ID_CUPA	NUME_CUPA	VALOARE_PREMIU	ID_TURNEU	ID_CAMPION
81	Shining	10000	11	64
82	Winners_Cup	40000	12	64
83	Friendship_Cup	9300	13	64
84	Juniors_Cup	10000	14	64
85	Seniors_Cup	10000	15	64

```
628
629    select * from este_alcatuit;
```

ID_TURNEU	ID_RUNDA
11	21
11	22
11	23
11	24
11	25
12	21
12	22
12	23
12	24
12	25

```
630
631    select * from participa;
```

ID_TURNEU	ID_JUCATOR
11	61
11	62
11	63
11	64
11	65
12	61
12	62
12	63
12	64
12	65
13	61

```
632
633    select * from rezultat;
```

ID_REZULTAT	SCOR	DATA_MECI	NUME_CASTIGATOR	ID_RUNDA
31	3-1	1.02.2022	Andrei	21
32	2-3	3.02.2022	Alex	22
33	3-0	4.02.2022	Horia	23
34	3-5	6.02.2022	Halep	24
35	3-1	26.01.2022	Vinchin	25

```
633    select * from rezultat;
634
635    select * from runda;
```

ID_RUNDA	TIP_RUNDA	NUME_ARBITRU	NUMAR_COPII_MINGII
21	Optime	John_W	5
22	Sfert	Andrew_S	6
23	Semi_finala	Filip_P	4
24	Finala	Jo_S	8
25	Calificari	Simona_H	13

ID_TIP_TEREN	DIMENSIUNE_TEREN	TIP_SUPRAFATA
51	80	zgura
52	110	gazon
53	130	hard
54	45	zgura
55	39	gazon
56	78	iarba
57	65	hard
58	57	iarba
59	90	zgura

6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de colecții studiate. Apelați subprogramul.

Cerința (în limbaj natural):

Să se afișeze țările în care s-au desfășurat cel puțin 5 turnee de Grand Slam. Pentru fiecare dintre acestea să se specifice orașul în care s-au desfășurat. Tratați cazul în care nu există nicio astfel de țară. (Să se folosească un tablou indexat pentru informațiile necesare din tabela turneu, precum și o colecție de tip varray pentru memorarea și afișarea țărilor găsite)

```
create or replace procedure printCountriesWithMoreThan5Tournaments
is
 type informatii_turneu is table of turneu%rowtype index by binary_integer;
 type nume_tari is varray(40) of tara.nume_tara%type;
 info_turneu informatii_turneu;
 tari nume_tari := nume_tari();
 index_tara number(20);
begin
    select * bulk collect into info_turneu from turneu;
    index_tara := 1;
    for i in info_turneu.first..info_turneu.last loop
        if info_turneu(i).numar_editii > 5 then
            dbms_output.put_line(info_turneu(i).oras);
            tari.extend();
            select t.nume_tara into tari(index_tara) from tara t where info_turneu(i).id_tara = t.id_tara;
            index_tara := index_tara + 1;
        end if;
    end loop;

    if index_tara = 0 then
        dbms_output.put_line('Nu exista nicio astfel de țară');
    end if;
    if index_tara > 0 then
        dbms_output.put_line('Țările care au găzduit turneele din orașe de mai sus sunt:');
        for i in 1..index_tara - 1 loop
            dbms_output.put_line(tari(i));
        end loop;
    end if;
end;

execute printCountriesWithMoreThan5Tournaments;
```

create or replace procedure printCountriesWithMoreThan5Tournaments
is
type informatii_turneu is table of turneu%rowtype index by binary_integer;
type nume_tari is varray(40) of tara.nume_tara%type;
info_turneu informatii_turneu;
tari nume_tari := nume_tari();
index_tara number(20);

```
begin
    select * bulk collect into info_turneu from turneu;
    index_tara := 1;
    for i in info_turneu.first..info_turneu.last loop
        if info_turneu(i).numar_editii > 5 then
            dbms_output.put_line(info_turneu(i).oras);
            tari.extend();
            select t.nume_tara into tari(index_tara) from tara t where info_turneu(i).id_tara =
t.id_tara;
            index_tara := index_tara + 1;
        end if;
    end loop;

    if index_tara = 0 then
        dbms_output.put_line('Nu exista nicio astfel de tara');
    end if;
    if index_tara > 0 then
        dbms_output.put_line('Tarile care au gazduit turneele din orasele de mai sus sunt: ');
        for i in 1..index_tara - 1 loop
            dbms_output.put_line(tari(i));
        end loop;
    end if;
end;


execute printCountriesWithMoreThan5Tournaments;
```

Output:

```
Statement processed.
Melbourne
Paris
Bucuresti
New_York
Londra
Tarile care au gazduit turneele din orasele de mai sus sunt:
Austalia
Franta
Romania
SUA
UK
```

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

Cerința (în limbaj natural):

Să se afișeze numărul total de locuri al tuturor terenurilor pe care se desfășoară un sezon al turneelor de Grand Slam (indiferent de complexul din care acestea fac parte), care au ,,zgură" ca tip de suprafață. De asemenea, să se afișeze numele complexurilor de terenuri care le detin și să se rețină numele terenului cu cea mai mare capacitate (pentru a stabili unde se va desfășura finala). Tratați cazul în care nu există niciun astfel de teren.

```
create or replace procedure printCapacityOfSlagPlayingFields
is

   --cursor
    cursor c_tip_teren(tip tip_teren.tip_suprafata%type) is
    select id_tip_teren id_tip
    from tip_teren
    where tip_suprafata = tip;

    --ciclu cursor
    cursor c_complex_terenuri is
    select denumire_teren nume, numar_locuri numar, d.id_tip_teren id_detine
    from complex_terenuri co, detine d
    where co.id_teren = d.id_teren;
    type nume_complexe is varray(40) of complex_terenuri.denumire_teren%type;
    num_complexe nume_complexe := nume_complexe(30);
    nr_max number(30);
    nr_total number(30);
    nr_curent number(30);
    i number(20);
    nume_max complex_terenuri.denumire_teren%type;
    id_tip tip_teren.id_tip_teren%type;
    gasit number(2);

begin
   nr_max := 0;
   nr_total := 0;
```

```
    i := 0;
    open c_tip_teren('zgura');
    loop
        fetch c_tip_teren into id_tip;
        exit when c_tip_teren%NOTFOUND;
        for complex in c_complex_terenuri loop
            gasit := 0;
            if complex.id_detine = id_tip then
                for num in 1..i loop
                    if num_complexe(num) = complex.nume then
                        gasit := 1;
                    end if;
                end loop;
                if gasit = 0  then
                    dbms_output.put_line(complex.nume);
                    i := i+1;
                    num_complexe.extend();
                    num_complexe(i) := complex.nume;
                end if;
                nr_curent := complex.numar;
                if nr_max < nr_curent then
                    nr_max := nr_curent;
                    nume_max := complex.nume;
                end if;
                nr_total := nr_total + nr_curent;

            end if;
        end loop;
    end loop;
    close c_tip_teren;
    if nr_max = 0 then
        dbms_output.put_line('Nu exista terenuri care au zgura ca suprafata!');
    else
        dbms_output.put_line('Numarul total de spectatori ce pot participa la meciurile de pe
zgura pe toata durata sezonului este de ' || nr_total );
        dbms_output.put_line('Complexul cu cea mai mare capacitate este ' || nume_max);
    end if;

end;

execute printCapacityOfSlagPlayingFields();
```

```
    cursor c_complex_terenuri is
    select denumire_teren nume, numar_locuri numar, d.id_tip_teren id_detine
    from complex_terenuri co, detine d
    where co.id_teren = d.id_teren;
    type nume_complexe is varray(40) of complex_terenuri.denumire_teren%type;
    num_complexe nume_complexe := nume_complexe(30);
    nr_max number(30);
    nr_total number(30);
    nr_curent number(30);
    i number(20);
    nume_max complex_terenuri.denumire_teren%type;
    id_tip tip_teren.id_tip_teren%type;
    gasit number(2);

begin
    nr_max := 0;
    nr_total := 0;
    i := 0;
    open c_tip_teren('zgura');
    loop
        fetch c_tip_teren into id_tip;
        exit when c_tip_teren%NOTFOUND;
        for complex in c_complex_terenuri loop
            gasit := 0;
            if complex.id_detine = id_tip then
                for num in 1..i loop
                    if num_complexe(num) = complex.nume then
                        gasit := 1;
                    end if;
                end loop;
                if gasit = 0  then
                    dbms_output.put_line(complex.nume);
                    i := i+1;
                    num_complexe.extend();
                    num_complexe(i) := complex.nume;
                end if;
                nr_curent := complex.numar;
                if nr_max < nr_curent then
                    nr_max := nr_curent;
                    nume_max := complex.nume;
                end if;
                nr_total := nr_total + nr_curent;

            end if;
        end loop;
    end loop;
    close c_tip_teren;
    if nr_max = 0 then
        dbms_output.put_line('Nu exista terenuri care au zgura ca suprafata!');
    else
        dbms_output.put_line('Numarul total de spectatori ce pot participa la meciurile de pe zgura pe toata durata sezonului este de ' || nr_total );
        dbms_output.put_line('Complexul cu cea mai mare capacitate este ' || nume_max);
    end if;

end;

execute printCapacityOfSlagPlayingFields();
```

Output:

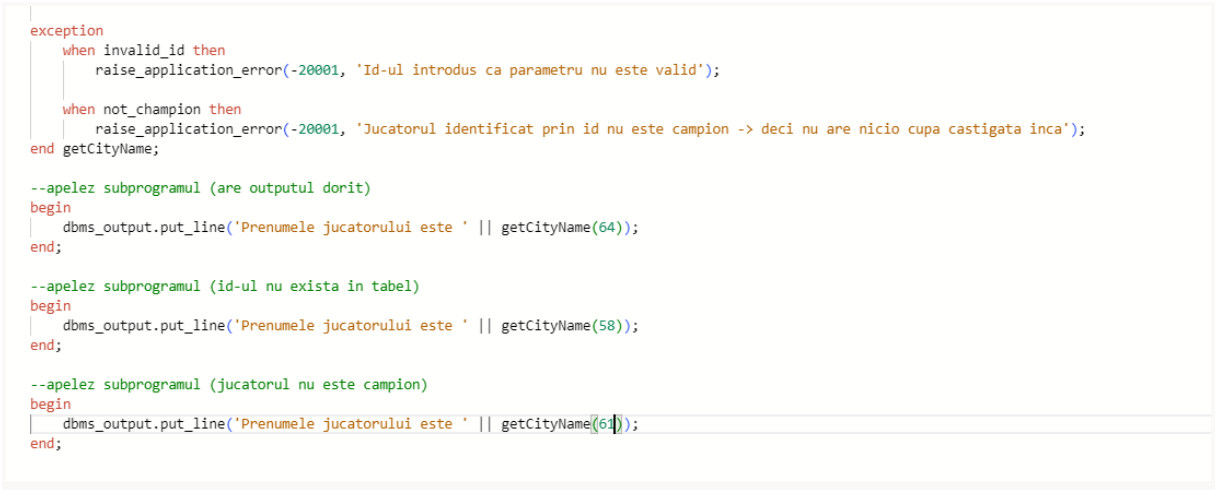
```
Statement processed.
Bernabeu
Camp_Nou
Numarul total de spectatori ce pot participa la meciurile de pe zgura pe toata durata sezonului este de 264000
Complexul cu cea mai mare capacitate este Camp_Nou
```

8. Formulați în limbaj natural o problemă pe care să o rezolvaţi folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Cerința (în limbaj natural):
Dându-se id-ul unui jucător să se afișeze orașele în care s-au desfășurat turneele ale căror cupe le-a câștigat. Să se afișeze, de asemena, numele campionului. Să se trateze următoarele excepții:

- Id-ul introdus nu este valabil
- Jucătorul nu a fost campion (deci nu are nicio cupă câștigată)

```
create or replace function getCityName(id_j jucator.id_jucator%type)
    return varchar2 is
    valid_id number(2);
    este_campion jucator.tip%type;
    nume_oras turneu.oras%type;
    p_jucator jucator.prenume_jucator%type;
    invalid_id exception;
    not_champion exception;
    competitie_amicala exception;
begin
    valid_id := 0 ;
    select count(*) into valid_id from jucator j where j.id_jucator = id_j;
    if valid_id = 0 then
        raise invalid_id;
    end if;
    select tip into este_campion from jucator j where j.id_jucator = id_j;
     if este_campion = 'negativ' then
        raise not_champion;
    end if;
    -- join pe 3 tabele diferite
    dbms_output.put_line('Orasele in care a castigat turnee sunt: ' || p_jucator);
    for oras in (select t.oras o from jucator j join cupa cu on cu.id_campion = id_jucator and j.id_jucator = id_j
    join turneu t on t.id_turneu = cu.id_turneu) loop
        dbms_output.put_line(oras.o);
    end loop;


    select prenume_jucator into p_jucator from jucator where id_jucator = id_j;
    return p_jucator;

exception
    when invalid_id then
        raise_application_error(-20001, 'Id-ul introdus ca parametru nu este valid');
```



```
exception
    when invalid_id then
        raise_application_error(-20001, 'Id-ul introdus ca parametru nu este valid');

    when not_champion then
        raise_application_error(-20001, 'Jucatorul identificat prin id nu este campion -> deci nu are nicio cupa castigata inca');
end getCityName;

--apelez subprogramul (are outputul dorit)
begin
    dbms_output.put_line('Prenumele jucatorului este ' || getCityName(64));
end;

--apelez subprogramul (id-ul nu exista in tabel)
begin
    dbms_output.put_line('Prenumele jucatorului este ' || getCityName(58));
end;

--apelez subprogramul (jucatorul nu este campion)
begin
    dbms_output.put_line('Prenumele jucatorului este ' || getCityName(61));
end;
```

```
create or replace function getCityName(id_j jucator.id_jucator%type)
    return varchar2 is
    valid_id number(2);
    este_campion jucator.tip%type;
    nume_oras turneu.oras%type;
    p_jucator jucator.prenume_jucator%type;
    invalid_id exception;
    not_champion exception;
    competitie_amicala exception;
begin
    valid_id := 0 ;
    select count(*) into valid_id from jucator j where j.id_jucator = id_j;
    if valid_id = 0 then
        raise invalid_id;
    end if;
    select tip into este_campion from jucator j where j.id_jucator = id_j;
     if este_campion = 'negativ' then
        raise not_champion;
    end if;
    -- join pe 3 tabele diferite
    dbms_output.put_line('Orasele in care a castigat turnee sunt: ' || p_jucator);
    for oras in (select t.oras o from jucator j join cupa cu on cu.id_campion = id_jucator
and j.id_jucator = id_j
    join turneu t on t.id_turneu = cu.id_turneu) loop
        dbms_output.put_line(oras.o);
    end loop;
```

```
    select prenume_jucator into p_jucator from jucator where id_jucator = id_j;
    return p_jucator;

exception
    when invalid_id then
        raise_application_error(-20001, 'Id-ul introdus ca parametru nu este valid');

    when not_champion then
        raise_application_error(-20001, 'Jucatorul identificat prin id nu este campion ->
deci nu are nicio cupa castigata inca');
end getCityName;

--apelez subprogramul (are outputul dorit)
begin
    dbms_output.put_line('Prenumele jucatorului este ' || getCityName(64));
end;

--apelez subprogramul (id-ul nu exista in tabel)
begin
    dbms_output.put_line('Prenumele jucatorului este ' || getCityName(58));
end;

--apelez subprogramul (jucatorul nu este campion)
begin
    dbms_output.put_line('Prenumele jucatorului este ' || getCityName(61));
end;
```

Outputurile:

```
Statement processed.
Orasele in care a castigat turnee sunt:
Melbourne
Paris
Bucuresti
New_York
Londra
Prenumele jucatorului este Halep
```

```
ORA-20001: Id-ul introdus ca parametru nu este valid ORA-06512: at "SQL_DGVNHGKXZLRQLCDXROJHDGUPH.GETCITYNAME", line 33
ORA-06512: at line 2
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

```
ORA-20001: Jucatorul identificat prin id nu este campion -> deci nu are nicio cupa castigata inca ORA-06512: at
"SQL_DGVNHGKXZLRQLCDXROJHDGUPH.GETCITYNAME", line 36
ORA-06512: at line 2
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

9. 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 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS.

Cerința (în limbaj natural):

Având în vedere situația pandemică în care ne aflăm, este nevoie de o evidență a cluburilor participante la nivel de turneu. Astfel, pentru orice țară dată ca parametru, să se afișeze atât numele clubului, ce are cel puțin un jucător care urmează să participe la minim un turneu desfășurat în acea țară, cât și numărul de jucători al acestuia. Astfel, colegii săi de club vor putea fi ținuți sub observație pentru a facilita procesul de împiedicare a răspândirii pandemiei. Se vor trata următoarele excepții:

- Nu există niciun astfel de club
- Există mai multe înregistrări care corespund acestei descrieri (sunt mai multe cluburi care corespund descrierii)
- Țara introdusă nu se află pe lista de țări gazdă

```
--inserted a row for testing all cases
insert into participa values (13,61);

create or replace procedure getCLubNameAndNumberOfPlayers
    (nume tara.nume_tara%type)
is
    gasit number(2);
    country_not_found exception;
    number_of_players club.numar_jucatori%type;
    name_of_club club.denumire_club%type;
begin
    select count(*) into gasit from tara where nume_tara = nume;
    if gasit = 0 then
        raise country_not_found;
    end if;
    --join pe 5 tabele
    select cl.denumire_club, cl.numar_jucatori into name_of_club, number_of_players from tara t join
    turneu tu on t.id_tara = tu.id_tara join
    participa p on p.id_turneu = tu.id_turneu join
    jucator j on j.id_jucator = p.id_jucator join
    club cl on cl.id_jucator = p.id_jucator
    where t.nume_tara = nume
    order by cl.nr_jucatori desc;
    dbms_output.put_line('Clubul este ' || name_of_club);
    dbms_output.put_line('Clubul este ' || number_of_players);
exception
    when country_not_found then
        raise_application_error(-20001, 'Tara specificata nu se afla in lista tarilor gazda');
    when TOO_MANY_ROWS then
        raise_application_error(-20001, 'Sunt mai multe cluburi în situația precizată');
    when NO_DATA_FOUND then
        raise_application_error(-20001, 'Nu exista niciun astfel de club');
    when others then
        raise_application_error(-20001, 'Eroare necunoscuta');
```

```
        raise_application_error(-20001, 'Eroare nec

end getCLubNameAndNumberOfPlayers;

--procedura are outputul dorit
execute getCLubNameAndNumberOfPlayers('Romania');

--exceptie TOO_MANY_ROWS
execute getCLubNameAndNumberOfPlayers('Franta');
--exceptie NO_DATA_FOUND
execute getCLubNameAndNumberOfPlayers('SUA');
--exceptie country_not_found
execute getCLubNameAndNumberOfPlayers('Bulgaria');
```

--inserted a row for testing all cases
insert into participa values (13,61);

create or replace procedure getCLubNameAndNumberOfPlayers

```
    (nume tara.nume_tara%type)
  is
    gasit number(2);
    country_not_found exception;
    number_of_players club.numar_jucatori%type;
    name_of_club club.denumire_club%type;
  begin
    select count(*) into gasit from tara where nume_tara = nume;
    if gasit = 0 then
      raise country_not_found;
    end if;
    --join pe 5 tabele
    select cl.denumire_club, cl.numar_jucatori into name_of_club, number_of_players from
tara t join
    turneu tu on t.id_tara = tu.id_tara join
    participa p on p.id_turneu = tu.id_turneu join
    jucator j on j.id_jucator = p.id_jucator join
    club cl on cl.id_jucator = p.id_jucator
    where t.nume_tara = nume
    order by cl.numar_jucatori desc;
    dbms_output.put_line('Clubul este ' || name_of_club);
    dbms_output.put_line('Clubul are ' || number_of_players || 'jucatori legitimati');
  exception
    when country_not_found then
      raise_application_error(-20001, 'Tara specificata nu se afla in lista tarilor gazda');
    when TOO_MANY_ROWS then
      raise_application_error(-20001, 'Sunt mai multe cluburi în situația precizată');
    when NO_DATA_FOUND then
      raise_application_error(-20001, 'Nu exista niciun astfel de club');
    when others then
      raise_application_error(-20001, 'Eroare necunoscuta');


  end getCLubNameAndNumberOfPlayers;

--procedura are outputul dorit
execute getCLubNameAndNumberOfPlayers('Romania');

--exceptie TOO_MANY_ROWS
execute getCLubNameAndNumberOfPlayers('Franta');
--exceptie NO_DATA_FOUND
execute getCLubNameAndNumberOfPlayers('SUA');
--exceptie country_not_found
execute getCLubNameAndNumberOfPlayers('Bulgaria');
```

Outputurile:

```
Statement processed.
Clubul este FC_Invingatori
Clubul are 35jucatori legitimati
```

```
ORA-20001: Sunt mai multe cluburi în situația precizată ORA-06512: at
"SQL_DGVNHGKXZLRQLCDXROJHDGUPH.GETCLUBNAMEANDNUMBEROFPLAYERS", line 27
ORA-06512: at line 1
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

```
ORA-20001: Nu exista niciun astfel de club ORA-06512: at "SQL_DGVNHGKXZLRQLCDXROJHDGUPH.GETCLUBNAMEANDNUMBEROFPLAYERS", line 29
ORA-06512: at line 1
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

```
ORA-20001: Tara specificata nu se afla in lista tarilor gazda ORA-06512: at
"SQL_DGVNHGKXZLRQLCDXROJHDGUPH.GETCLUBNAMEANDNUMBEROFPLAYERS", line 25
ORA-06512: at line 1
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

10. Definiți un trigger LMD la nivel de comandă.

În intervalul orar 01:00 - 04:00 serverul de baze de date este supus verificarilor de rulare optima + eventualelor interventii de mentenanta, astfel modificarile asupra datelor nu ar trebui sa fie acceptate. (trebuie definit un trigger separat pentru fiecare tabel; am atasat triggerul pentru tabelul club)

```
create or replace trigger maitenanceBreak
    before insert or update or delete on club
begin
    if to_char(sysdate, 'HH24') between 1 and 4 then
        raise_application_error(-20001, 'Maintenance Break! please come back later');
    end if;
end;

select * from club;
delete from club where id_jucator = 63;
```

create or replace trigger maitenanceBreak
    before insert or update or delete on club
begin
    if to_char(sysdate, 'HH24') between 1 and 4 then
        raise_application_error(-20001, 'Maintenance Break! please come back later');
    end if;
end;

select * from club;
delete from club where id_jucator = 63;

11. Definiți un trigger LMD la nivel de linie.

Când un club este eliminat din baza de date să se șteargă toți jucătorii legitimați la acel club. Să se creeze o nouă tabelă care să stocheze jucătorii șterși.

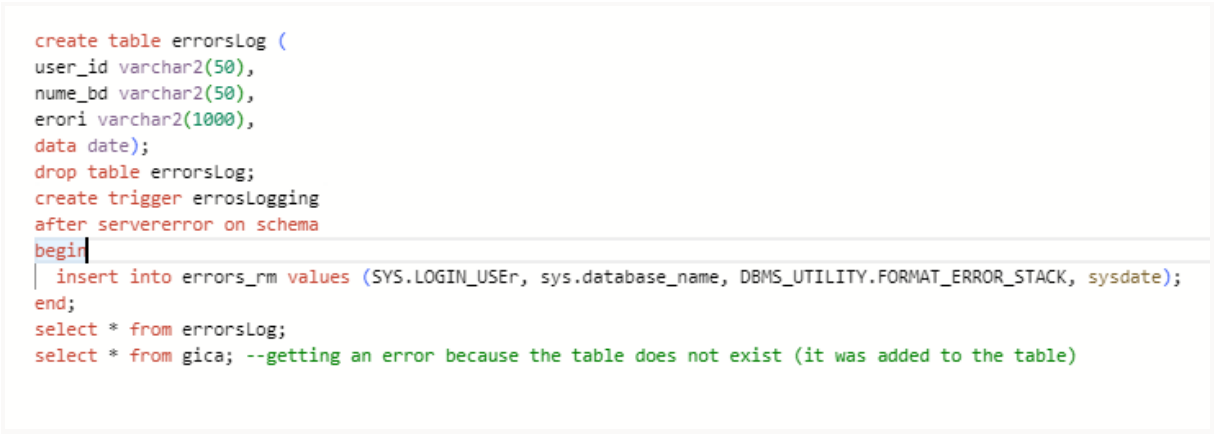
```
create or replace trigger deletePlayersFromDeletedClub
    after delete on club
    for each row
begin
    delete from jucator j where j.id_jucator = :old.id_jucator;
    insert into jucatori_stersi values( :old.nume_jucator, :old.prenume_jucator, :old.data_nasterii, :old.sponsor);
end;

select * from jucator;
delete from club where id_jucator = 63;
```

create or replace trigger deletePlayersFromDeletedClub
    after delete on club
    for each row
begin
    delete from jucator j where j.id_jucator = :old.id_jucator;
    insert into jucatori_stersi values(:old.nume_jucator, :old.prenume_jucator, :old.data_nasterii, :old.sponsor);
end;

select * from jucator;
delete from club where id_jucator = 63;

12. Definiti un trigger LDD.

Să se memoreze toate erorile apărute într-un tabel de erori.



```
create table errorsLog (
user_id varchar2(50),
nume_bd varchar2(50),
erori varchar2(1000),
data date);
drop table errorsLog;
create trigger errosLogging
after servererror on schema
begin
  insert into errors_rm values (SYS.LOGIN_USEr, sys.database_name, DBMS_UTILITY.FORMAT_ERROR_STACK, sysdate);
end;
select * from errorsLog;
select * from gica; --getting an error because the table does not exist (it was added to the table)
```

create table errorsLog (
user_id varchar2(50),
nume_bd varchar2(50),
erori varchar2(1000),
data date);
drop table errorsLog;
create trigger errosLogging
after servererror on schema
begin
  insert into errors_rm values (SYS.LOGIN_USEr, sys.database_name, DBMS_UTILITY.FORMAT_ERROR_STACK, sysdate);
end;
select * from errorsLog;
select * from gica; --getting an error because the table does not exist (it was added to the table)

13. Definiti un pachet care sa contina toate obiectele definite in cadrul proiectului.

```
create or replace package tenisTournamentsManagement as
    -- ex 6
    procedure printCountriesWithMoreThan5Tournaments;
    -- ex 7
    procedure printCapacityOfSlagPlayingFields;
    -- ex 8
     function getCityName(id_j jucator.id_jucator%type)
    return varchar2;
    -- ex 9
    procedure getCLubNameAndNumberOfPlayers
        (nume tara.nume_tara%type);
end tenisTournamentsManagement;


create or replace package body tenisTournamentsManagement as

-- 6

procedure printCountriesWithMoreThan5Tournaments
is
type informatii_turneu is table of turneu%rowtype index by binary_integer;
type nume_tari is varray(40) of tara.nume_tara%type;
info_turneu informatii_turneu;
tari nume_tari := nume_tari();
index_tara number(20);
begin
    select * bulk collect into info_turneu from turneu;
    index_tara := 1;
    for i in info_turneu.first..info_turneu.last loop
        if info_turneu(i).numar_editii > 5 then
            dbms_output.put_line(info_turneu(i).oras);
            tari.extend();
            select t.nume_tara into tari(index_tara) from tara t where info_turneu(i).id_tara =
t.id_tara;
            index_tara := index_tara + 1;
        end if;
    end loop;

    if index_tara = 0 then
        dbms_output.put_line('Nu exista nicio astfel de tara');
```

```
    end if;
    if index_tara > 0 then
        dbms_output.put_line('Tarile care au gazduit turneele din orasele de mai sus sunt: ');
        for i in 1..index_tara - 1 loop
            dbms_output.put_line(tari(i));
        end loop;
    end if;
end;




-- 7
procedure printCapacityOfSlagPlayingFields
is

    --cursor
     cursor c_tip_teren(tip tip_teren.tip_suprafata%type) is
     select id_tip_teren id_tip
     from tip_teren
     where tip_suprafata = tip;

     --ciclu cursor
     cursor c_complex_terenuri is
     select denumire_teren nume, numar_locuri numar, d.id_tip_teren id_detine
     from complex_terenuri co, detine d
     where co.id_teren = d.id_teren;
     type nume_complexe is varray(40) of complex_terenuri.denumire_teren%type;
     num_complexe nume_complexe := nume_complexe(30);
     nr_max number(30);
     nr_total number(30);
     nr_curent number(30);
     i number(20);
     nume_max complex_terenuri.denumire_teren%type;
     id_tip tip_teren.id_tip_teren%type;
     gasit number(2);

begin
    nr_max := 0;
    nr_total := 0;
    i := 0;
    open c_tip_teren('zgura');
    loop
        fetch c_tip_teren into id_tip;
        exit when c_tip_teren%NOTFOUND;
        for complex in c_complex_terenuri loop
            gasit := 0;
```

```
            if complex.id_detine = id_tip then
                for num in 1..i loop
                    if num_complexe(num) = complex.nume then
                        gasit := 1;
                    end if;
                end loop;
                if gasit = 0  then
                    dbms_output.put_line(complex.nume);
                    i := i+1;
                    num_complexe.extend();
                    num_complexe(i) := complex.nume;
                end if;
                nr_curent := complex.numar;
                if nr_max < nr_curent then
                    nr_max := nr_curent;
                    nume_max := complex.nume;
                end if;
                nr_total := nr_total + nr_curent;

            end if;
        end loop;
    end loop;
    close c_tip_teren;
    if nr_max = 0 then
        dbms_output.put_line('Nu exista terenuri care au zgura ca suprafata!');
    else
        dbms_output.put_line('Numarul total de spectatori ce pot participa la meciurile de pe
zgura pe toata durata sezonului este de ' || nr_total );
        dbms_output.put_line('Complexul cu cea mai mare capacitate este ' || nume_max);
    end if;

end;




-- 8

function getCityName(id_j jucator.id_jucator%type)
    return varchar2 is
    valid_id number(2);
    este_campion jucator.tip%type;
    nume_oras turneu.oras%type;
    p_jucator jucator.prenume_jucator%type;
    invalid_id exception;
    not_champion exception;
    competitie_amicala exception;
begin
```

```
    valid_id := 0 ;
    select count(*) into valid_id from jucator j where j.id_jucator = id_j;
    if valid_id = 0 then
        raise invalid_id;
    end if;
    select tip into este_campion from jucator j where j.id_jucator = id_j;
     if este_campion = 'negativ' then
        raise not_champion;
    end if;
    -- join pe 3 tabele diferite
    dbms_output.put_line('Orasele in care a castigat turnee sunt: ' || p_jucator);
    for oras in (select t.oras o from jucator j join cupa cu on cu.id_campion = id_jucator and
j.id_jucator = id_j
    join turneu t on t.id_turneu = cu.id_turneu) loop
        dbms_output.put_line(oras.o);
    end loop;


    select prenume_jucator into p_jucator from jucator where id_jucator = id_j;
    return p_jucator;

exception
    when invalid_id then
        raise_application_error(-20001, 'Id-ul introdus ca parametru nu este valid');

    when not_champion then
        raise_application_error(-20001, 'Jucatorul identificat prin id nu este campion -> deci nu
are nicio cupa castigata inca');
end getCityName;




-- 9
procedure getCLubNameAndNumberOfPlayers
        (nume tara.nume_tara%type)
    is
        gasit number(2);
        country_not_found exception;
        number_of_players club.numar_jucatori%type;
        name_of_club club.denumire_club%type;
    begin
        select count(*) into gasit from tara where nume_tara = nume;
        if gasit = 0 then
            raise country_not_found;
        end if;
        --join pe 5 tabele
```

```
        select cl.denumire_club, cl.numar_jucatori into name_of_club, number_of_players from
tara t join
        turneu tu on t.id_tara = tu.id_tara join
        participa p on p.id_turneu = tu.id_turneu join
        jucator j on j.id_jucator = p.id_jucator join
        club cl on cl.id_jucator = p.id_jucator
        where t.nume_tara = nume
        order by cl.numar_jucatori desc;
        dbms_output.put_line('Clubul este ' || name_of_club);
        dbms_output.put_line('Clubul are ' || number_of_players || 'jucatori legitimati');
    exception
        when country_not_found then
            raise_application_error(-20001, 'Tara specificata nu se afla in lista tarilor gazda');
        when TOO_MANY_ROWS then
            raise_application_error(-20001, 'Sunt mai multe cluburi în situația precizată');
        when NO_DATA_FOUND then
            raise_application_error(-20001, 'Nu exista niciun astfel de club');
        when others then
            raise_application_error(-20001, 'Eroare necunoscuta');


    end getCLubNameAndNumberOfPlayers;
end tenisTournamentsManagement;
```

```
create or replace package tenisTournamentsManagement as
    -- ex 6
    procedure printCountriesWithMoreThan5Tournaments;
    -- ex 7
    procedure printCapacityOfSlagPlayingFields;
    -- ex 8
    function getCityName(id_j jucator.id_jucator%type)
    return varchar2;
    -- ex 9
    procedure getCLubNameAndNumberOfPlayers
        (nume tara.nume_tara%type);
end tenisTournamentsManagement;


create or replace package body tenisTournamentsManagement as

-- 6

procedure printCountriesWithMoreThan5Tournaments
is
type informatii_turneu is table of turneu%rowtype index by binary_integer;
type nume_tari is varray(40) of tara.nume_tara%type;
info_turneu informatii_turneu;
tari nume_tari := nume_tari();
index_tara number(20);
begin
    select * bulk collect into info_turneu from turneu;
    index_tara := 1;
    for i in info_turneu.first..info_turneu.last loop
        if info_turneu(i).numar_editii > 5 then
            dbms_output.put_line(info_turneu(i).oras);
            tari.extend();
            select t.nume_tara into tari(index_tara) from tara t where info_turneu(i).id_tara = t.id_tara;
            index_tara := index_tara + 1;
        end if;
    end loop;

    if index_tara = 0 then
        dbms_output.put_line('Nu exista nicio astfel de tara');
    end if;
```

```
    end if;
    if index_tara > 0 then
        dbms_output.put_line('Tarile care au gazduit turneele din orasele de mai sus sunt: ');
        for i in 1..index_tara - 1 loop
            dbms_output.put_line(tari(i));
        end loop;
    end if;
end;



-- 7
procedure printCapacityOfSlagPlayingFields
is

    --cursor
    cursor c_tip_teren(tip tip_teren.tip_suprafata%type) is
    select id_tip_teren id_tip
    from tip_teren
    where tip_suprafata = tip;

    --ciclu cursor
    cursor c_complex_terenuri is
    select denumire_teren nume, numar_locuri numar, d.id_tip_teren id_detine
    from complex_terenuri co, detine d
    where co.id_teren = d.id_teren;
    type nume_complexe is varray(40) of complex_terenuri.denumire_teren%type;
    num_complexe nume_complexe := nume_complexe(30);
    nr_max number(30);
    nr_total number(30);
    nr_curent number(30);
    i number(20);
    nume_max complex_terenuri.denumire_teren%type;
    id_tip tip_teren.id_tip_teren%type;
    gasit number(2);

begin
```

```
    nr_max := 0;
    nr_total := 0;
    i := 0;
    open c_tip_teren('zgura');
    loop
        fetch c_tip_teren into id_tip;
        exit when c_tip_teren%NOTFOUND;
        for complex in c_complex_terenuri loop
            gasit := 0;
            if complex.id_detine = id_tip then
                for num in 1..i loop
                    if num_complexe(num) = complex.nume then
                        gasit := 1;
                    end if;
                end loop;
                if gasit = 0  then
                    dbms_output.put_line(complex.nume);
                    i := i+1;
                    num_complexe.extend();
                    num_complexe(i) := complex.nume;
                end if;
                nr_curent := complex.numar;
                if nr_max < nr_curent then
                    nr_max := nr_curent;
                    nume_max := complex.nume;
                end if;
                nr_total := nr_total + nr_curent;

            end if;
        end loop;
    end loop;
    close c_tip_teren;
    if nr_max = 0 then
        dbms_output.put_line('Nu exista terenuri care au zgura ca suprafata!');
    else
        dbms_output.put_line('Numarul total de spectatori ce pot participa la meciurile de pe zgura pe toata durata sezonului este de ' || nr_total );
        dbms_output.put_line('Complexul cu cea mai mare capacitate este ' || nume_max);
    end if;

end;
```

```
-- 8

function getCityName(id_j jucator.id_jucator%type)
    return varchar2 is
    valid_id number(2);
    este_campion jucator.tip%type;
    nume_oras turneu.oras%type;
    p_jucator jucator.prenume_jucator%type;
    invalid_id exception;
    not_champion exception;
    competitie_amicala exception;
begin
    valid_id := 0 ;
    select count(*) into valid_id from jucator j where j.id_jucator = id_j;
    if valid_id = 0 then
        raise invalid_id;
    end if;
    select tip into este_campion from jucator j where j.id_jucator = id_j;
     if este_campion = 'negativ' then
        raise not_champion;
    end if;
    -- join pe 3 tabele diferite
    dbms_output.put_line('Orasele in care a castigat turnee sunt: ' || p_jucator);
    for oras in (select t.oras o from jucator j join cupa cu on cu.id_campion = id_jucator and j.id_jucator = id_j
    join turneu t on t.id_turneu = cu.id_turneu) loop
        dbms_output.put_line(oras.o);
    end loop;


    select prenume_jucator into p_jucator from jucator where id_jucator = id_j;
    return p_jucator;

exception
    when invalid_id then
        raise_application_error(-20001, 'Id-ul introdus ca parametru nu este valid');

    when not_champion then
        raise_application_error(-20001, 'Jucatorul identificat prin id nu este campion -> deci nu are nicio cupa castigata inca');
end getCityName;
```

```
-- 9
procedure getCLubNameAndNumberOfPlayers
        (nume tara.nume_tara%type)
    is
        gasit number(2);
        country_not_found exception;
        number_of_players club.numar_jucatori%type;
        name_of_club club.denumire_club%type;
    begin
        select count(*) into gasit from tara where nume_tara = nume;
        if gasit = 0 then
            raise country_not_found;
        end if;
        --join pe 5 tabele
        select cl.denumire_club, cl.numar_jucatori into name_of_club, number_of_players from tara t join
        turneu tu on t.id_tara = tu.id_tara join
        participa p on p.id_turneu = tu.id_turneu join
        jucator j on j.id_jucator = p.id_jucator join
        club cl on cl.id_jucator = p.id_jucator
        where t.nume_tara = nume
        order by cl.numar_jucatori desc;
        dbms_output.put_line('Clubul este ' || name_of_club);
        dbms_output.put_line('Clubul are ' || number_of_players || 'jucatori legitimati');
    exception
        when country_not_found then
            raise_application_error(-20001, 'Tara specificata nu se afla in lista tarilor gazda');
        when TOO_MANY_ROWS then
            raise_application_error(-20001, 'Sunt mai multe cluburi în situația precizată');
        when NO_DATA_FOUND then
            raise_application_error(-20001, 'Nu exista niciun astfel de club');
        when others then
            raise_application_error(-20001, 'Eroare necunoscuta');


    end getCLubNameAndNumberOfPlayers;
end tenisTournamentsManagement;

execute tenisTournamentsManagement.printCountriesWithMoreThan5Tournaments;
```

Cateva exemple de apelari ale pachetului + outputurile lor:

```
598    execute tenisTournamentsManagement.printCountriesWithMoreThan5Tournaments;
```

```
Statement processed.
Melbourne
Paris
Bucuresti
New_York
Londra
Tarile care au gazduit turneele din orasele de mai sus sunt:
Austalia
Franta
Romania
SUA
UK
```

```
602
603    execute tenisTournamentsManagement.printCapacityOfSlagPlayingFields();
```

```
Statement processed.
Bernabeu
Camp_Nou
Numarul total de spectatori ce pot participa la meciurile de pe zgura pe toata durata sezonului este de 264000
Complexul cu cea mai mare capacitate este Camp_Nou
```

```
602
603    execute tenisTournamentsManagement.printCapacityOfSlagPlayingFields();
604
605    begin
606        dbms_output.put_line('Prenumele jucatorului este ' || tenisTournamentsManagement.getCityName(64));
607    end;
```

```
Statement processed.
Orasele in care a castigat turnee sunt:
Melbourne
Paris
Bucuresti
New_York
Londra
Prenumele jucatorului este Halep
```

14. Definiti un pachet care sa includa tipuri de date complexe si obiecte necesare unui flux de actiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 functii, minim 2 proceduri).

Voi defini pachetul *TournamentsManagementOperations* ce are ca scop definirea unor metode utile in administrarea turneelor de Grand Slam si afisarea rapida a datelor vitale ale acestuia.

Pachetul va contine tipuri de date complexe, necesare unui flux de actiuni complex:

- info_round (de tip record ce va mapa informatiile utile din tabelele runda + rezultat)

- tournament_round (de tip vector de info_record ce va memora pentru fiecare turneu in parte informatiile mentionate mai sus, legate de una din rundele sale)

- tournament_all_details (o matrice de dimensiune n- linii (unde n este numarul turneelor) si m- coloane (unde m este numarul maxim de runde ale unui sigur turneu), ce va retine pentru fiecare id al turneelor, toate tournament_round-urile mentionate mai sus)

Procedurile definite in acest pachet:

- init_package (va popula tipurile de date complexe conform descrierilor de mai sus cu informatiile din baza de date)
- print_info_about_tournament (va afisa toate informatiile unui turneu dat ca parametru intr-un mod lizibil si usor interpretabil + va trata eventualele erori ce pot aparea logand mesaje corespunzatoare)

Functiile definite in acest pachet:

-print_copii_mingi (va return toti copiii de mingi participanti la un turneu al carui id va fi dat ca parametru + va trata eventualele erori ce pot aparea logand mesaje corespunzatoare)

-get_arbitru_finala (va returna numele arbitrului ce a arbitrat in finala turneului cu id-ul dat ca parametru)

```
create or replace package TournamentsManagementOperations as
    -- tipurile complexe de date

    type info_round_type is record
        (
            tip_runda varchar2(50),
            nume_castigator varchar2(50),
            nume_arbitru varchar2(50),
            numar_copii_mingii number(20)
        );

    type tournament_round_type is varray(50) of info_round_type;


    type tournament_all_details_type is varray(50) of tournament_round_type;

    -- procedurile
    procedure init_package;
    procedure print_info_about_tournament(index_turneu turneu.id_turneu%type);

    --functiile
    function print_copii_mingi(index_turneu turneu.id_turneu%type) return number;
    function get_arbitru_finala(index_turneu turneu.id_turneu%type) return varchar2;
end TournamentsManagementOperations;

create or replace package body TournamentsManagementOperations as
    -- definirea variabilelor de tipuri complexe
```

```
create or replace package body TournamentsManagementOperations as
    -- definirea variabilelor de tipuri complexe


    info_round info_round_type;

    tournament_round tournament_round_type := tournament_round_type();

    tournament_all_details tournament_all_details_type := tournament_all_details_type();

    -- variabila suplimentara ce retine id -ul turneului in functie de pozitia pe care se afla in matrice + nr de linii din matrice + nr de coloane

    type tournaments_id__position_by_index_type is varray(50) of number;
    tournaments_id_position_by_index tournaments_id__position_by_index_type := tournaments_id__position_by_index_type();
    no_of_rows number(2);
    no_of_columns number(2) := 0;


    --procedurile

    procedure init_package
    is

        tip_runda runda.tip_runda%type;
        nume_castigator rezultat.nume_castigator%type;
        nume_arbitru runda.nume_arbitru%type;
        index_coloana number(20);
        index_linie number(20);
        modified number(2);
    begin
        index_linie := 1;
        index_coloana := 0;
        tournament_all_details.extend();
        tournament_all_details(index_linie) := tournament_round_type();
        -- creating tournament_all_details
        for turneu in (select id_turneu id_turneu from turneu) loop
            -- appending row into tournament_all_details
            modified := 0;
            for iduri in (select e.id_runda runda from este_alcatuit e where turneu.id_turneu = e.id_turneu) loop
                for info in (select ru.numar_copii_mingii copii, ru.tip_runda runda, ru.nume_arbitru arbitru, re.nume_castigator castigator from runda ru, rezultat re where iduri.runda = ru.id_runda and ru.id_runda = re.id_runda) loop
                    -- creating info_round record
                    info_round.tip_runda := info.runda;
```

```
    -- creating tournament_all_details
    for turneu in (select id_turneu id_turneu from turneu) loop
        -- appending row into tournament_all_details
        modified := 0;
        for iduri in (select e.id_runda runda from este_alcatuit e where turneu.id_turneu = e.id_turneu) loop
            for info in (select ru.numar_copii_mingii copii, ru.tip_runda runda, ru.nume_arbitru arbitru, re.nume_castigator castigator from runda ru, rezultat re where iduri.runda = ru.id_runda and ru.id_runda = re.id_runda) loop
                -- creating info_round record
                info_round.tip_runda := info.runda;
                info_round.nume_castigator := info.castigator;
                info_round.nume_arbitru := info.arbitru;
                info_round.numar_copii_mingii := info.copii;
                -- appending tournament round
                index_coloana := index_coloana + 1;
                tournament_all_details(index_linie).extend();
                tournament_all_details(index_linie)(index_coloana) := info_round;
            end loop;
            modified := 1;
        end loop;
        if index_coloana > no_of_columns then
            no_of_columns := index_coloana;
        end if;
        index_coloana := 0;
        if modified = 1 then
            tournaments_id_position_by_index.extend();
            tournaments_id_position_by_index(index_linie) := turneu.id_turneu;
            index_linie := index_linie + 1;
            tournament_all_details.extend();
            tournament_all_details(index_linie) := tournament_round_type();
        end if;

    end loop;

    no_of_rows := tournament_all_details.last-1;

end init_package;
```

```
    row_num number(10);
    invalid_tournament exception;
begin
    init_package();
    row_num := 0;
    for i in tournaments_id_position_by_index.first..tournaments_id_position_by_index.last loop
        if tournaments_id_position_by_index(i) = index_turneu then
            row_num := i;
        end if;
    end loop;
    if row_num = 0 then
        raise invalid_tournament;
    end if;
    dbms_output.put_line('Runda   Arbitrul     Castigatorul');

    for j in 1..no_of_columns loop
        dbms_output.put_line(tournament_all_details(row_num)(j).tip_runda || '     ' ||tournament_all_details(row_num)(j).nume_castigator || '     ' || tournament_all_details(row_num)(j).nume_arbitru || '     ' || tournament_all_de
    end loop;
exception
    when invalid_tournament then
        raise_application_error(-20001, 'Id-ul specificat nu a fost validat corespunzator requestului dvs');
end print_info_about_tournament;

--functiile

function print_copii_mingi(index_turneu turneu.id_turneu%type) return number
is
    suma_copii number(20);
    row_num number(20);
    invalid_tournament exception;
begin
    init_package();
    suma_copii := 0;
    row_num := 0;
    for i in tournaments_id_position_by_index.first..tournaments_id_position_by_index.last loop
        if tournaments_id_position_by_index(i) = index_turneu then
            row_num := i;
```

```
        row_num := 0;
        for i in tournaments_id_position_by_index.first..tournaments_id_position_by_index.last loop
            if tournaments_id_position_by_index(i) = index_turneu then
                row_num := i;
            end if;
        end loop;
        if row_num = 0 then
            raise invalid_tournament;
        end if;
        for j in 1..no_of_columns loop
            suma_copii  := suma_copii + tournament_all_details(row_num)(j).numar_copii_mingii;
        end loop;
        return suma_copii;

    exception
        when invalid_tournament then
            raise_application_error(-20001, 'Id-ul specificat nu a fost validat corespunzator requestului dvs');
    end print_copii_mingi;

    function get_arbitru_finala(index_turneu turneu.id_turneu%type) return varchar2
    is

        row_num number(20);
        invalid_tournament exception;
    begin
        init_package();
        row_num := 0;
        for i in tournaments_id_position_by_index.first..tournaments_id_position_by_index.last loop
            if tournaments_id_position_by_index(i) = index_turneu then
                row_num := i;
            end if;
        end loop;
        if row_num = 0 then
            raise invalid_tournament;
        end if;
        -- deoarece finala este mereu ultima partida desfasurata
        return tournament_all_details(row_num)(no_of_columns).nume_arbitru;
```

```
    exception
        when invalid_tournament then
            raise_application_error(-20001, 'Id-ul specificat nu a fost validat corespunzator requestului dvs');
    end;

end TournamentsManagementOperations;


execute TournamentsManagementOperations.init_package;

execute TournamentsManagementOperations. print_info_about_tournament(11);

begin
    dbms_output.put_line('Numarul total al copiiilor de mingi din turneu este '|| TournamentsManagementOperations.print_copii_mingi(12));
end;

begin
    dbms_output.put_line('Arbitrul ce a arbitrat in finala este '|| TournamentsManagementOperations.get_arbitru_finala(11));
end;
```

Exemple de outputuri:

```
362
363     execute TournamentsManagementOperations. print_info_about_tournament(11);
364
```

```
Statement processed.
Runda   Arbitrul     Castigatorul
Optime      Andrei      John_W      5
Sfert     Alex      Andrew_S      6
Semi_finala     Horia      Filip_P      4
Finala      Halep      Jo_S      8
Calificari      Vinchin      Simona_H      13
```

```
364
365  begin
366      dbms_output.put_line('Numarul total al copiiilor de mingi din turneu este '|| TournamentsManagementOperations.print_copii_mingi(12));
367  end;
```

```
Statement processed.
Numarul total al copiiilor de mingi din turneu este 36
```

```
368
369  begin
370      dbms_output.put_line('Arbitrul ce a arbitrat in finala este '|| TournamentsManagementOperations.get_arbitru_finala(11));
371  end;
372
373
```

```
Statement processed.
Arbitrul ce a arbitrat in finala este Simona_H
```

```
368
369  begin
370      dbms_output.put_line('Arbitrul ce a arbitrat in finala este '|| TournamentsManagementOperations.get_arbitru_finala(16));
371  end;
372
373
```

```
ORA-20001: Id-ul specificat nu a fost validat corespunzator requestului dvs ORA-06512: at "SQL_UCWHEMZSRQVBSAECOAAYIZQXV.TOURNAMENTSMANAGEMENTOPERATIONS", line 148
ORA-06512: at line 2
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

```
create or replace package TournamentsManagementOperations as
    -- tipurile complexe de date


    type info_round_type is record
        (
            tip_runda varchar2(50),
            nume_castigator varchar2(50),
            nume_arbitru varchar2(50),
            numar_copii_mingii number(20)
        );

    type tournament_round_type is varray(50) of info_round_type;


    type tournament_all_details_type is varray(50) of tournament_round_type;

    -- procedurile
    procedure init_package;
    procedure print_info_about_tournament(index_turneu turneu.id_turneu%type);

    --functiile
    function print_copii_mingi(index_turneu turneu.id_turneu%type) return number;
    function get_arbitru_finala(index_turneu turneu.id_turneu%type) return varchar2;
end TournamentsManagementOperations;

create or replace package body TournamentsManagementOperations as
    -- definirea variabilelor de tipuri complexe
```

```
    info_round info_round_type;

    tournament_round tournament_round_type := tournament_round_type();

    tournament_all_details tournament_all_details_type := tournament_all_details_type();

   -- variabila suplimentara ce retine id -ul turneului in functie de pozitia pe care se afla in
matrice + nr de linii din matrice + nr de coloane

   type tournaments_id__position_by_index_type is varray(50) of number;
   tournaments_id_position_by_index tournaments_id__position_by_index_type :=
tournaments_id__position_by_index_type();
   no_of_rows number(2);
   no_of_columns number(2) := 0;


   --procedurile

   procedure init_package
   is

      tip_runda runda.tip_runda%type;
      nume_castigator rezultat.nume_castigator%type;
      nume_arbitru runda.nume_arbitru%type;
      index_coloana number(20);
      index_linie number(20);
      modified number(2);
   begin
      index_linie := 1;
      index_coloana := 0;
      tournament_all_details.extend();
      tournament_all_details(index_linie) := tournament_round_type();
      -- creating tournament_all_details
      for turneu in (select id_turneu id_turneu from turneu) loop
         -- appending row into tournament_all_details
         modified := 0;
         for iduri in (select e.id_runda runda from este_alcatuit e where turneu.id_turneu =
e.id_turneu) loop
            for info in (select ru.numar_copii_mingii copii, ru.tip_runda runda, ru.nume_arbitru
arbitru, re.nume_castigator castigator from runda ru, rezultat re where iduri.runda =
ru.id_runda and ru.id_runda = re.id_runda) loop
               -- creating info_round record
               info_round.tip_runda := info.runda;
               info_round.nume_castigator := info.castigator;
               info_round.nume_arbitru := info.arbitru;
               info_round.numar_copii_mingii := info.copii;
```

```
                -- appending tournament round
                index_coloana := index_coloana + 1;
                tournament_all_details(index_linie).extend();
                tournament_all_details(index_linie)(index_coloana) := info_round;
            end loop;
            modified := 1;
        end loop;
        if index_coloana > no_of_columns then
            no_of_columns := index_coloana;
        end if;
        index_coloana := 0;
        if modified = 1 then
            tournaments_id_position_by_index.extend();
            tournaments_id_position_by_index(index_linie) := turneu.id_turneu;
            index_linie := index_linie + 1;
            tournament_all_details.extend();
            tournament_all_details(index_linie) := tournament_round_type();
        end if;

    end loop;

    no_of_rows := tournament_all_details.last-1;

end init_package;


procedure print_info_about_tournament(index_turneu turneu.id_turneu%type)
is
    row_num number(10);
    invalid_tournament exception;
begin
    init_package();
    row_num := 0;
    for i in tournaments_id_position_by_index.first..tournaments_id_position_by_index.last
loop
        if tournaments_id_position_by_index(i) = index_turneu then
            row_num := i;
        end if;
    end loop;
    if row_num = 0 then
        raise invalid_tournament;
    end if;
     dbms_output.put_line('Runda   Arbitrul     Castigatorul');

    for j in 1..no_of_columns loop
        dbms_output.put_line(tournament_all_details(row_num)(j).tip_runda || '     '
||tournament_all_details(row_num)(j).nume_castigator || '     ' ||
```

```
tournament_all_details(row_num)(j).nume_arbitru || '     ' ||
tournament_all_details(row_num)(j).numar_copii_mingii);
        end loop;
    exception
        when invalid_tournament then
            raise_application_error(-20001, 'Id-ul specificat nu a fost validat corespunzator
requestului dvs');
    end print_info_about_tournament;

    --functiile

    function print_copii_mingi(index_turneu turneu.id_turneu%type) return number
    is
        suma_copii number(20);
        row_num number(20);
        invalid_tournament exception;
    begin
        init_package();
        suma_copii := 0;
        row_num := 0;
        for i in tournaments_id_position_by_index.first..tournaments_id_position_by_index.last
loop
            if tournaments_id_position_by_index(i) = index_turneu then
                row_num := i;
            end if;
        end loop;
        if row_num = 0 then
            raise invalid_tournament;
        end if;
        for j in 1..no_of_columns loop
             suma_copii  := suma_copii +
tournament_all_details(row_num)(j).numar_copii_mingii;
        end loop;
        return suma_copii;

    exception
        when invalid_tournament then
            raise_application_error(-20001, 'Id-ul specificat nu a fost validat corespunzator
requestului dvs');
    end print_copii_mingi;

    function get_arbitru_finala(index_turneu turneu.id_turneu%type) return varchar2
    is

        row_num number(20);
        invalid_tournament exception;
    begin
        init_package();
```

```
        row_num := 0;
        for i in tournaments_id_position_by_index.first..tournaments_id_position_by_index.last
loop
            if tournaments_id_position_by_index(i) = index_turneu then
                row_num := i;
            end if;
        end loop;
        if row_num = 0 then
            raise invalid_tournament;
        end if;
        -- deoarece finala este mereu ultima partida desfasurata
        return tournament_all_details(row_num)(no_of_columns).nume_arbitru;


    exception
        when invalid_tournament then
            raise_application_error(-20001, 'Id-ul specificat nu a fost validat corespunzator
requestului dvs');
    end;

end TournamentsManagementOperations;



execute TournamentsManagementOperations.init_package;

execute TournamentsManagementOperations. print_info_about_tournament(11);

begin
    dbms_output.put_line('Numarul total al copiiilor de mingi din turneu este '||
TournamentsManagementOperations.print_copii_mingi(12));
end;

begin
    dbms_output.put_line('Arbitrul ce a arbitrat in finala este '||
TournamentsManagementOperations.get_arbitru_finala(11));
end;
```