

Proiect

Sisteme de

Gestiune a

Bazelor de Date

Anul 2 Semestrul I

2020-2021

A. Descrierea modelului real, a utilității acestuia, a regulilor de funcționare și a constrângerilor/restricțiilor.

Acest proiect reprezintă o bază de date pentru o aplicație, care ar putea ajuta atât sportivii de performanță, cât și cluburile sportive să primească în timp real detalii despre viitoarele competiții sportive.

Există Federații care se ocupă de buna desfășurare a competițiilor sportive. O competiție sportivă poate fi sponsorizată de unele persoane.

Sportivii pot participa la competiții, dacă se simt pregătiți și se antrenează în săli, iar aceste săli pot fi oricâte. Sălile de sport au angajați (cleaner, paznic sau administrator) și se află în locații, locații unde pot fi mai multe săli.

Sportivii pot face parte din mai multe federații sportive și doar dintr-un singur club de sport. Aceștia sunt antrenați de antrenorii de la club, care pot fi mai mulți.

Această aplicație este limitată doar în țara noastră, dar se poate extinde și la nivel global.

Modelul de date respectă următoarele reguli de funcționare și restricții :

- a. Angajații sălii de sport pot fi de tip: cleaner, paznic sau administrator.
- b. Angajații trebuie să lucreze la o sală de sport.
- c. Într-o sală de sport pot lucra mai mulți angajați sau niciunul (abia a fost deschisă).
- d. Sala de sport se află într-o locație.
- e. Într-o locație se pot afla mai multe săli de sport sau niciuna (abia a fost deschisă spre închiriere locația respectivă).
- f. Un sportiv se poate antrena la mai multe săli de sport, sau niciuna (se antrenează în parc sau acasă).
- g. Un sportiv poate face parte din mai multe federații de sport (ex: helterofil și powerlifter și bodybuilder) sau niciuna (abia și-a obținut legitimația de sportiv de performanță).
- h. Sala de sport poate găzdui mai mulți sportivi, sau niciunul (este abia deschisă).
- i. Sala de sport aparține mai multor federații sau niciuna.
- j. O federație poate avea mai mulți sportivi sau niciunul (abia s-a înființat Pe viitor poate apar și alte sporturi).
- k. O federație își poate practica activitatea în mai multe săli de sport sau niciuna.
- l. O federație organizează mai multe competiții sau niciuna (încă nu s-a deschis sezonul competitiv).

- m. O competitie apartine unei singure federatii.
- n. Un sponsor sponsorizeaza mai multe competitii, sau niciuna.
- o. O competitie este sponsorizata de mai multi sponsori sau niciunul.
- p. La o competitie participa mai multi sportivi, sau niciunul (adica competitia abia a fost introdusa in baza de date si nimeni nu s-a inscris inca).
- q. Un sportiv participa la mai multe competitii sau niciuna(nu se simte pregatiti).
- r. Un sportiv apartine unui club neaparat.
- s. Un club are mai multi sportivi sau niciunul (abia infiintat).
- t. Un sportiv a fost in trecut la mai multe cluburi sau niciunul.
- u. Un club apare intr-un istoric de mai multe ori sau niciodata.
- v. Un club are mai multi antrenori sau niciunul(s-au pensionat cu totii).
- w. Un antrenor face parte dintr-un club neaparat.

B. Descrierea entităților, incluzând precizarea cheii primare.

Pentru modelul ales de mine, referitor la competițiile sportive, ANGAJATI_SALA, SALA_SPORT, LOCATIE, SPORTIV, FEDERATIE_DE_SPORT, COMPETITIE, CLUB, SPONSORI, CLUB_HISTORY, ANTRENORI reprezinta entitati.

Vom prezenta entitățile modelului de date, dând o descriere completă a fiecăreia. De asemenea, pentru fiecare entitate se va preciza cheia primară.

ANGAJATI SALA = persoane care lucreaza in sala de sport. Acestea pot fi de tip cleaner, paznic sau administrator. Cheia primara a entitatii este **cod_angajat**.

SALA SPORT = locatie unde se fac antrenamentele sportive si/sau competitii sportive. Ele sunt dotate standart cu aparate specifice care permit desfasurarea oricarui antrenament doreste sa faca sportivul. Atributul **cod_sala** reprezinta cheia primara a acestei entitati.

LOCATIE = locul unde este situate sala de sport si/sau competitia sportive si/sau clubul din care fac parte sportivii. Cheia primara este **cod_locatie**.

SPORTIV = persoana care practica sport de performanta(este acreditat de una dintre federatiile de sport) si participa la competitii. El face parte dintr-un club sportiv. Cheia primara o reprezinta atributul **cod_sportiv**.

FEDERATIE DE SPORT = sunt structuri sportive de interes național, constituite prin asocierea cluburilor sportive pe ramuri de sport. Organizeaza competitii si are cheia primara atributul ***cod_federatie***.

COMPETITIE = concurs ce este organizat de o federatie sportiva, la care participa sportive. Ea este sponsorizata. Atributul ***cod_competitie*** reprezinta cheia primara a acestei entitati.

CLUB = organizație care are un scop comun. Noi ne referim la cluburile sportive. Acestea au antrenori si sportivi. Cheia primară este ***cod_club***.

SPONSORI = sunt persoanele care sponsorizeaza competitii sportive caritabil sau pentru promovare. Cheia primară este ***cod_sponsor***.

CLUB HISTORY = în această entitate este stocat istoricul unui sportiv. Adică este reținută perioada în care un sportiv a făcut parte dintr-un club. Atributele ***cod_sportiv*** si ***data_start*** formează cheia primară compusă.

ANTRENORI = sunt persoane care prepare sportivii pentru competiții în cadrul unui club. Cheia primară este atributul ***cod_antrenor***.

CLEANER, PAZNIC, ADMINISTRATOR = subentități ale entității ANGAJAȚI_SALA.

C. Descrierea relațiilor, incluzând precizarea cardinalității acestora.

Vom prezenta relațiile modelului de date, dând o descriere completă a fiecăreia. De fapt, denumirile acestor legături sunt sugestive, reflectând conținutul acestora și entitățile pe care le leagă. Pentru fiecare relație se va preciza cardinalitatea minimă și maximă.

ANGAJATI_SALA_lucreaza_SALA_SPORT = relație ce leagă entitățile ANGAJAȚI_SALA și SALA_SPORT , reflectând legătura dintre acestea (cine lucrează la o sala de sport). Ea are cardinalitatea minima 0:1 (un angajat trebuie să lucreze la o sală de sport și o sala poate să nu aibă niciun angajat) și cardinalitatea maximă n:1 (mai mulți angajați pot lucra la o singura sală de sport, iar o sală de sport are mai mulți angajați).

SALA_SPORT_se afla LOCATIE = relație ce leagă entitățile SALA_SPORT și LOCATIE , reflectând legătura dintre acestea (unde se află o sală de sport). Ea are cardinalitatea minima 0:1 (o sală de sport trebuie să se afle

la o locație și într-o locație poate să nu fie nicio sală de sport) și cardinalitatea maximă $n:1$ (mai multe săli de sport se află într-o locație, iar o locație are mai multe săli de sport).

LOCATIE_se_afla_CLUB = relație ce leagă entitățile LOCATIE și CLUB , reflectând legătura dintre acestea (unde se află un club). Ea are cardinalitatea minimă $1:0$ (intr-o locație poate să nu fie niciun club și un club trebuie să se afle la o locație) și cardinalitatea maximă $1:n$ (o locație are mai multe cluburi sportive, iar mai multe cluburi se află într-o locație).

LOCATIE_se_afla_FEDERATIE_DE_SPORT = relație ce leagă entitățile LOCATIE și FEDERATIE_DE_SPORT , reflectând legătura dintre acestea (unde se află o federație). Ea are cardinalitatea minimă $1:0$ (intr-o locație poate să nu fie nicio federație și o federație trebuie să se afle la o locație) și cardinalitatea maximă $1:n$ (o locație are mai multe federatii, iar mai multe federatii se află într-o locație).

LOCATIE_se_afla_COMPETITIE = relație ce leagă entitățile LOCATIE și COMPETITIE , reflectând legătura dintre acestea (unde se află/se desfășoară o competiție). Ea are cardinalitatea minimă $1:0$ (intr-o locație poate să nu fie nicio competiție și o competiție trebuie să se afle la o locație) și cardinalitatea maximă $1:n$ (într-o locație se desfășoară mai multe competiții sportive, iar mai multe competiții se află într-o locație).

SPORTIV_apartine_SALA_SPORT_apartine_FEDERATIE_DE_SPORT = relație de tip 3 ce leagă entitățile SPORTIV, SALA_SPORT și FEDERATIE_DE_SPORT, reflectând sălile de sport pe care sportivul le frecventează și federațiile în cadrul cărora se antrenează. Denumirea acestei relații va fi apartine.

FEDERATIE_DE_SPORT_organizeaza_COMPETITIE = relație ce leagă entitățile FEDERATIE_DE_SPORT și COMPETITIE , reflectând legătura dintre acestea (ce federație organizează o competiție). Ea are cardinalitatea minimă $1:0$ (o federație poate să nu organizeze nicio competiție și o competiție trebuie să fie organizată de o federație) și cardinalitatea maximă $1:n$ (o federație poate să organizeze mai multe competiții și o competiție trebuie să fie organizată de o federație).

SPONSORI_sponsorizeaza_COMPETITIE = relație de tip *many-to-many* dintre entitățile SPONSOR și COMPETITIE, reflectând legătura dintre acestea (ce sponsori finanțează competițiile). Relația are cardinalitatea minimă 0:0 (un sponsor poate să nu sponsorizeze nicio competiție și o competiție poate fi finanțată fara sponsori) și cardinalitatea maximă *m:n*.

SPORTIV_participa_COMPETITIE = relație de tip *many-to-many* dintre entitățile SPORTIV și COMPETITIE, reflectând legătura dintre acestea (la ce competitii participa un sportiv). Relația are cardinalitatea minimă 0:0 (un sportiv poate să nu participe la nicio competiție și o competiție poate fi fara participanți (abia a fost introdusă in baza de date)) și cardinalitatea maximă *m:n*.

CLUB_fac_parte_SPORTIV = relație ce leagă entitățile CLUB și SPORTIV, reflectând legătura dintre acestea (dintr-un club fac parte sportivi). Ea are cardinalitatea minima 1:0 (un club poate să nu aiba niciun sportiv și un sportiv trebuie să aparțină unui club) și cardinalitatea maximă 1:n (un club poate avea mai mulți sportivi și un sportiv trebuie să aparțină unui club).

ANTRENORI_antreneaza_CLUB = relație ce leagă entitățile ANTRENORI și CLUB , reflectând legătura dintre acestea (ce antrenori antreneaza un club de sportivi). Ea are cardinalitatea minima 0:1 (un antrenor trebuie să fie intr-un club și un club poate să nu aibă antrenori (abia s-a deschis clubul)) și cardinalitatea maximă n:1 (mai mulți antrenori se află intr-un club, iar un club are mai mulți antrenori).

SPORTIV_are_CLUB_HISTORY = relație ce leagă entitățile SPORTIV și CLUB_HISTORY, reflectând legătura dintre acestea (un sportive are istoricul cluburilor unde a fost). Ea are cardinalitatea minima 1:0 (un sportiv poate să nu aibă niciun istoric și un istoric trebuie să aparțină unui sportiv) și cardinalitatea maximă 1:n (un sportiv poate avea mai multe istorice ale cluburilor unde a fost și un istoric trebuie să aparțină unui sportiv).

CLUB_HISTORY_apare_in_CLUB = relație ce leagă entitățile CLUB_HISTORY și CLUB , reflectând legătura dintre acestea (ce club apare in istoric). Ea are cardinalitatea minima 0:1 (un club nu e necesar sa aparțină unui istoric, dar un istoric trebuie sa aiba un club) și cardinalitatea maximă n:1 (un istoric trebuie sa aiba un club si un club poate apare in mai multe istoricuri).

D. Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor.

- Entitatea ANGAJATI_SALA are ca atribute:

cod_angajat = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui angajat temporar. Trebuie să fie not null și unic.

cod_sala = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul sălii angajatului respectiv. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul SALA_SPORT. Nu trebuie să fie null.

nume = variabilă de tip caracter, de lungime maximă 25, care reprezintă numele angajatului. Nu trebuie să fie null.

prenume = variabilă de tip caracter, de lungime maximă 25, care reprezintă prenumele angajatului. Nu trebuie să fie null.

data_nastere = variabilă de tip dată calendaristică, care reprezintă data nașterii angajatului respectiv. Nu trebuie să fie null.

sex = variabilă de tip caracter, luând valorile m sau f, de lungime 1, care reprezintă sexul angajatului.

data_angajarii = variabilă de tip dată calendaristică, care reprezintă data angajarii angajatului respectiv. Este default sysdate.

salariu = variabilă de tip întreg, de lungime maximă 5, care reprezintă salariul lunar al angajatului.

job = variabilă de tip caracter, de lungime 15, care reprezintă funcția pentru care a fost angajată persoana respectivă. De exemplu, poate să fie cleaner, paznic sau administrator.

- Entitatea SALA_SPORT are atributele :

cod_sala = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul sălii respective. Trebuie să fie not null și unic.

cod_locatie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul locației unde este amplasată sala de sport. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul LOCATIE. Nu trebuie să fie null.

an_infiintare = variabilă de tip întreg, de lungime maximă 5, care reprezintă anul înființării sălii de sport, adică anul deschiderii. Este default anul curent.

denumire = variabilă de tip caracter, de lungime 20, care reprezintă denumirea sălii de sport. Nu trebuie să fie null.

- Entitatea FEDERATIE_DE_SPORT are atributele :

cod_federatie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul federației respective. Trebuie să fie not null și unic.

denumire = variabilă de tip caracter, de lungime 20, care reprezintă denumirea sălii de sport. Nu trebuie să fie null.

an_infiintare = variabilă de tip întreg, de lungime maximă 5, care reprezintă anul înființării sălii de sport, adică anul deschiderii. Este default anul curent.

sport_practicat = variabilă de tip caracter, de lungime 15, care reprezintă sportul care este practicat în această federație. Nu trebuie să fie null.

cod_locatie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul locației unde este amplasată sala de sport. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul LOCATIE. Nu trebuie să fie null.

- Entitatea COMPETITIE are attributele :

cod_competitie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul competiției respective. Trebuie să fie not null și unic.

cod_federatie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul federației care organizează competiția. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul FEDERATIE_DE_SPORT. Nu trebuie să fie null.

cod_locatie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul locației unde este amplasată sala de sport. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul LOCATIE. Nu trebuie să fie null.

data_competitie = variabilă de tip dată calendaristică, care reprezintă data competiției respective. Nu trebuie să fie null.

valoare_totala_premii = variabilă de tip întreg, de lungime maximă 5, care reprezintă valoarea premiilor.

- Entitatea SPORTIV are attributele :

cod_sportiv = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul sportivului respectiv. Trebuie să fie not null și unic.

cod_club = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul federației care organizează competiția. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul FEDERATIE_DE_SPORT. Nu trebuie să fie null.

nume = variabilă de tip caracter, de lungime maximă 15, care reprezintă numele sportivului. Nu trebuie să fie null.

prenume = variabilă de tip caracter, de lungime maximă 25, care reprezintă prenumele sportivului. Nu trebuie să fie null.

data_nastere = variabilă de tip dată calendaristică, care reprezintă data nașterii sportivului respectiv. Nu trebuie să fie null.

nrtelefon = variabilă de tip caracter, de lungime maximă 10, care reprezintă numărul de telefon al sportivului.

- Entitatea SPONSORI are attributele :

cod_sponsor = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul sponsorului respectiv. Trebuie să fie not null și unic.

denumire_sponsor = variabilă de tip caracter, de lungime maximă 25, care reprezintă numele complet al sponsorului respectiv. Nu trebuie să fie null.

adresa_email = variabilă de tip caracter, de lungime maximă 25, care reprezintă adresa de email al sponsorului.

- Entitatea CLUB are attributele :

cod_club = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul clubului sportiv respectiv. Trebuie să fie not null și unic.

cod_locatie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul locației unde este amplasată sala de sport. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul LOCATIE. Nu trebuie să fie null.

denumire = variabilă de tip caracter, de lungime 25, care reprezintă denumirea clubului sportiv. Nu trebuie să fie null.

data_infiintare_club = variabilă de tip dată calendaristică, care reprezintă data înființării clubului respectiv. Este default sysdate.

nume_admin = variabilă de tip caracter, de lungime maximă 25, care reprezintă numele administratorului clubului respectiv. Nu trebuie sa fie null.

prenume_admin = variabilă de tip caracter, de lungime maximă 25, care reprezintă prenumele administratorului clubului respectiv. Nu trebuie sa fie null.

- Entitatea ANTRENORI are attributele :

cod_antrenor = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul antrenorului respectiv. Trebuie sa fie not null si unic.

cod_club = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul clubului unde lucrează antrenorul respectiv. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CLUB. Nu trebuie sa fie null.

nume = variabilă de tip caracter, de lungime maximă 15, care reprezintă numele antrenorului. Nu trebuie sa fie null.

prenume = variabilă de tip caracter, de lungime maximă 25, care reprezintă prenumele antrenorului. Nu trebuie sa fie null.

data_nastere = variabilă de tip dată calendaristică, care reprezintă data nașterii antrenorului respectiv. Nu trebuie sa fie null.

nrtелефon = variabila tip caracter, de lungime maxima 10, care reprezinta numarul de telefon al antrenorului.

data_inscriere_club = variabilă de tip dată calendaristică, care reprezintă data în care antrenorul s-a inscris la clubul unde antrenează. Este default sysdate.

- Entitatea LOCATIE are attributele :

cod_locatie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul locației respective. Trebuie sa fie not null si unic.

judet = variabilă de tip caracter, de lungime maximă 25, care reprezintă judetul locatiei. Nu trebuie sa fie null.

oras = variabilă de tip caracter, de lungime maximă 15, care reprezintă orasul locatiei. Nu trebuie sa fie null.

strada = variabilă de tip caracter, de lungime maximă 20, care reprezintă strada locatiei. Nu trebuie sa fie null.

numar = variabilă de tip caracter, de lungime maximă 8, care reprezintă numarul locatiei. Nu trebuie sa fie null. Am pus acest atribut de tip caracter deoarece pot exista adrese care au ca și număr "14A".

- Entitatea CLUB_HISTORY are attributele :

cod_sportiv = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul sportivului al carui istoric il inserăm. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul SPORTIV. Trebuie sa fie not null.

data_start = variabilă de tip dată calendaristică, care reprezintă data de start al inscrierii sportivului la un club din trecut. Nu trebuie sa fie null.

cod_club = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul federației care organizează competiția. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul FEDERATIE_DE_SPORT. Nu trebuie sa fie null.

data_final = variabilă de tip dată calendaristică, care reprezintă data de final al plecării sportivului la un club la care a fost in trecut. Nu trebuie sa fie null.

- Relația

SPORTIV_apartine_SALA_SPORT_apartine_FEDERATIE_DE_SPORT
are ca attribute :

cod_sportiv = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul sportivului care face antrenamentele la sala de sport in cadrul federatiei respective. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul SPORTIV. Trebuie sa fie not null.

cod_federatie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul federației care organizează competiția. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul FEDERATIE_DE_SPORT. Nu trebuie sa fie null.

cod_sala = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul sălii angajatului respectiv. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul SALA_SPORT. Nu trebuie sa fie null.

- Relația *SPORTIV_participa_COMPETITIE* are ca si attribute :

cod_sportiv = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul sportivului care face antrenamentele la sala de sport in cadrul federatiei respective. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul SPORTIV. Trebuie sa fie not null.

cod_competitie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul competiției la care s-a inscris sportivul. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul COMPETITIE. Trebuie sa fie not null.

data_inscriere = variabilă de tip dată calendaristică, care reprezintă data la care sportivul s-a inscris la competitie. Este default sysdate.

- Relația *SPONSORI_sponsorizeaza_COMPETITIE* are ca și attribute :

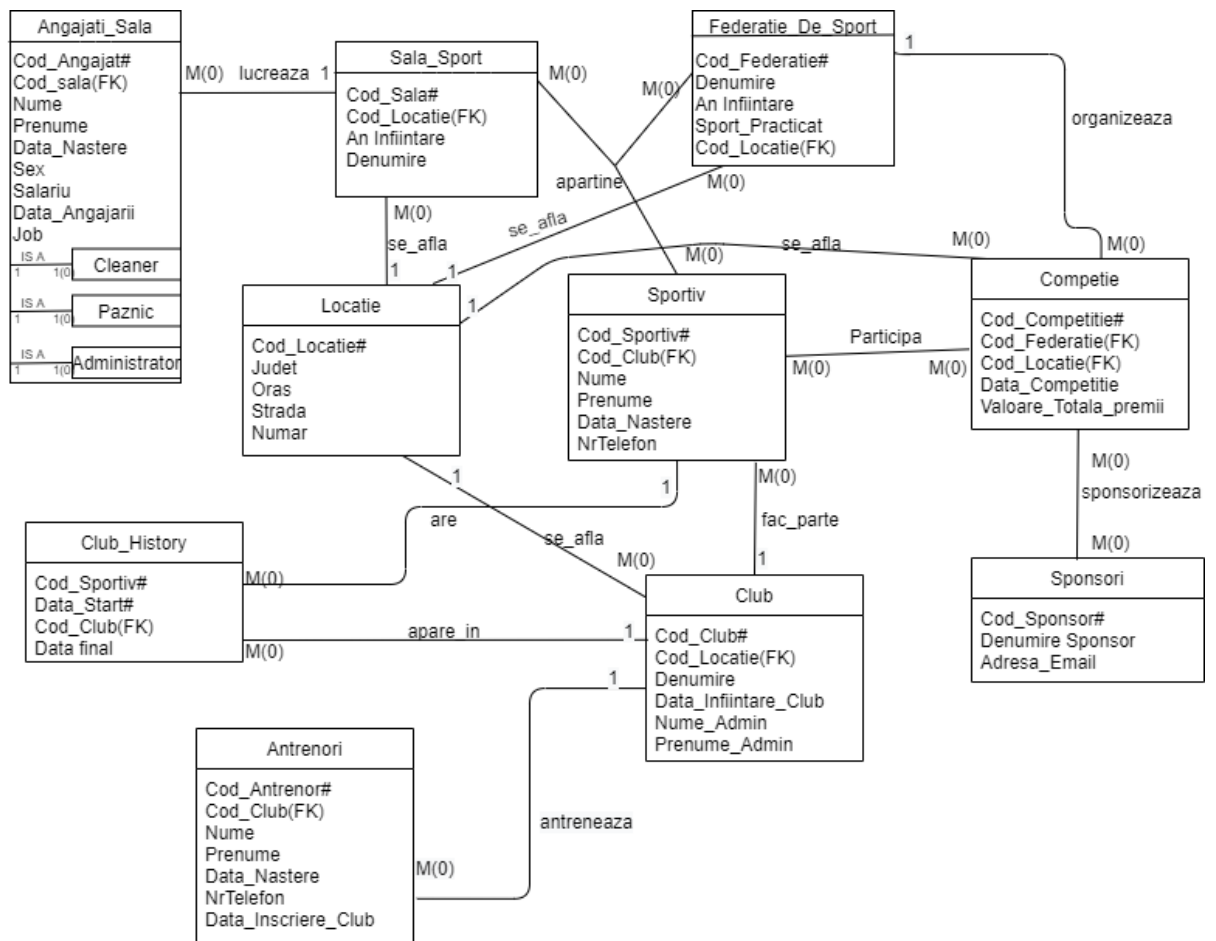
cod_sponsor = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul sponsorului care a oferit sponsorizarea. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul SPONSOR. Trebuie sa fie not null.

cod_competitie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul competiției la care s-a inscris sportivul. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul COMPETITIE. Trebuie sa fie not null.

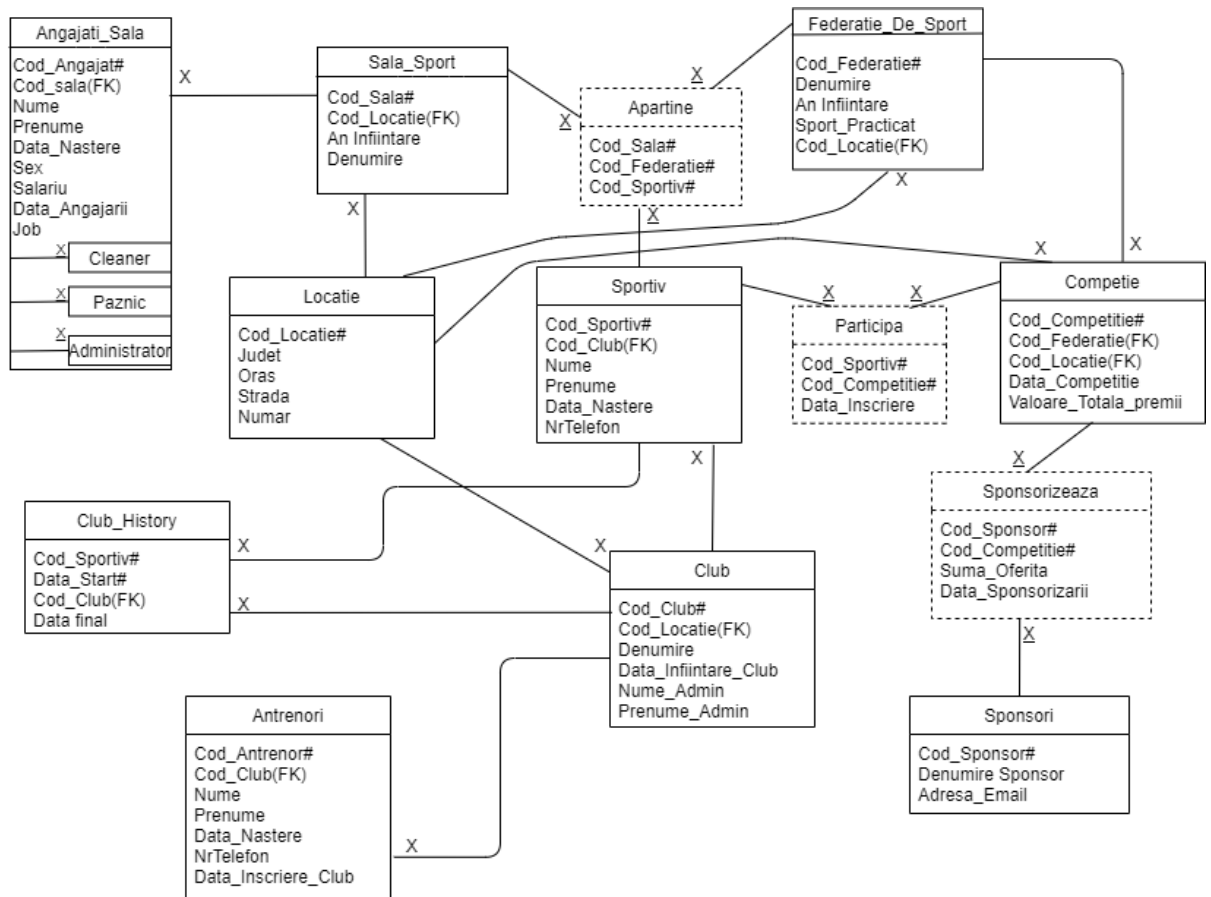
suma_oferita = variabilă de tip întreg, de lungime maximă 5, care reprezintă suma (in lei) pe care a oferit-o sponsorul pentru desfasurarea competiției. Nu trebuie sa fie null.

data_sponsorizarii = variabilă de tip dată calendaristică, care reprezintă data cand s-a facut sponsorizarea. Este default sysdate.

E. Realizarea diagramei entitate-relație corespunzătoare descrierii anterioare.



F. Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectată la punctul anterior.



G. Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectata la punctul 6.

ANGAJATI_SALA(cod_angajat#, cod_sala, nume, prenume, data_nastere, sex, salariu, data_angajarii, job)

SALA SPORT(cod sala#, cod locatie, an infiintare, denumire)

FEDERATIE_DE_SPORT(cod_federatie#, denumire, an_infiintare, sport_practicat, adresa_sediu_central)

LOCATIE(cod_locatie#, judet, oras, strada, numar)

SPORTIV(cod_sportiv#, cod_club, nume , prenume, data_nastere, nrtelefon)

COMPETITIE(cod_competitie#, cod_federatie, cod_locatie,
data_competitie, valoare_totala_premii)

CLUB HISTORY(cod sportiv#, data start#, cod club, data final)

```

CLUB(cod_club#, cod_locatie, denumire, data_infiintare_club,
nume_admin)

```

ANTRENORI(cod_antrenor#, cod_club, nume, prenume, data_nastere, nrtelefon, data_inscriere_club)

SPONSORI(cod sponsor#, nume sponsor, adresa email)

APARTINE(cod_sala#,cod_federatie#,cod_sportiv#)

PARTICIPA(cod_sportiv#, cod_competitie#, data_inscriere)
SPONSORIZEAZA(cod_sponsor#, cod_competitie#, suma_oferita,
data_sponsorizarii)

H. Crearea tabelelor în SQL și inserarea de date coerente în fiecare dintre acestea (minimum 5 înregistrări în fiecare tabel neasociativ; minimum 10 înregistrări în tabelele asociative).

create table LOCATIE

(cod_locatie number(5) constraint locatie_pk primary key,
judet varchar2(25) constraint locatie_judet not null,
oras varchar2(15) constraint locatie_oras not null,
strada varchar2(20) constraint locatie_strada not null,
numar varchar2(8) constraint locatie_numar not null
);

create table SPONSORI

(cod_sponsor number(5) constraint sponsor_pk primary key,
denumire_sponsor varchar2(25) constraint sponsor_denumire not null,
adresa_email varchar2(25)
);

create table SALA_SPORT

(cod_sala number(5) constraint sala_pk primary key,
cod_locatie number(5) constraint sala_locatie not null,
an_infiintare number(5) default to_number(to_char(sysdate,'yyyy')),
denumire varchar2(20) constraint sala_denumire not null,
constraint sala_locatie_fk foreign key(cod_locatie) references
locatie(cod_locatie)
);

create table ANGAJATI_SALA

(cod_angajat number(5) constraint angajati_pk primary key,
cod_sala number(5) constraint angajati_sala not null,
nume varchar2(25) constraint angajati_nume not null,
prenume varchar2(25) constraint angajati_prenume not null,
data_nastere date constraint angajati_datanastere not null,
sex varchar2(1),
data_angajarii date default sysdate,
salariu number(5) constraint angajati_salariu not null,
job varchar2(15),
constraint angajati_sala_fk foreign key(cod_sala) references
sala_sport(cod_sala),
constraint angajati_job_check check(lower(job)='cleaner' or lower(job)='paznic'
or lower(job)='administrator' or job is null),

```
constraint angajati_sex_check check(lower(sex)='m' or lower(sex)='f' or sex is
null)
);
```

```
create table CLUB
```

```
(cod_club number(5) constraint club_pk primary key,
cod_locatie number(5) constraint club_locatie not null,
denumire varchar2(25) constraint club_denumire not null,
data_infiintare_club date default sysdate,
nume_admin varchar2(25) constraint club_admin_nume not null,
prenume_admin varchar2(25) constraint club_admin_prenume not null,
constraint club_locatie_fk foreign key(cod_locatie) references
locatie(cod_locatie)
);
```

```
create table ANTRENORI
```

```
(cod_antrenor number(5) constraint antrenori_pk primary key,
cod_club number(5) constraint antrenori_club not null,
nume varchar2(15) constraint antrenori_nume not null,
prenume varchar2(25) constraint antrenori_prenume not null,
data_nastere date constraint antrenori_datanastere not null,
nrtelefon varchar2(10),
data_inscriere_club date default sysdate,
constraint antrenori_club_fk foreign key(cod_club) references club(cod_club)
);
```

```
create table SPORTIV
```

```
(cod_sportiv number(5) constraint sportiv_pk primary key,
cod_club number(5) constraint sportiv_club not null,
nume varchar2(15) constraint sportiv_nume not null,
prenume varchar2(25) constraint sportiv_prenume not null,
data_nastere date constraint sportiv_datanastere not null,
nrtelefon varchar2(10),
constraint sportiv_club_fk foreign key(cod_club) references club(cod_club)
);
```

```
create table FEDERATIE_DE_SPORT
```

```
(cod_federatie number(5) constraint federatie_pk primary key,
denumire varchar2(20) constraint federatie_denumire not null,
an_infiintare number(5) default to_number(to_char(sysdate,'yyyy')),
sport_practicat varchar2(15) constraint federatie_sport not null,
cod_locatie number(5) constraint federatie_locatie not null,
constraint federatie_loc_fk foreign key(cod_locatie) references
locatie(cod_locatie)
);
```

```
create table CLUB_HISTORY
```

```
(cod_sportiv number(5) constraint club_h_sportiv_fk references
sportiv(cod_sportiv),
data_start date constraint club_h_data_s not null,
cod_club number(5) constraint club_h_club not null,
data_final date,
constraint club_h_pk_compus primary key(cod_sportiv,data_start),
constraint club_h_data_f_check check(data_final > data_start),
constraint club_h_club_fk foreign key(cod_club) references club(cod_club)
);
```

```
create table COMPETITIE
```

```
(cod_competitie number(5) constraint competitie_pk primary key,
cod_federatie number(5) constraint competitie_fed not null,
cod_locatie number(5) constraint competitie_loc not null,
data_competitie date constraint competitie_data not null,
valoare_totala_premii number(5),
constraint competitie_fed_fk foreign key(cod_federatie) references
federatie_de_sport(cod_federatie),
constraint competitie_loc_fk foreign key(cod_locatie) references
locatie(cod_locatie)
);
```

```
--urmeaza tabelele asociative--
```

```
create table APARTINE
```

```
(cod_sala number(5) constraint apartine_sala not null,
cod_federatie number(5) constraint apartine_fed not null,
cod_sportiv number(5) constraint apartine_sportiv not null,
constraint apartine_sala_fk foreign key(cod_sala) references
sala_sport(cod_sala),
constraint apartine_fed_fk foreign key(cod_federatie) references
federatie_de_sport(cod_federatie),
constraint apartine_sportiv_fk foreign key(cod_sportiv) references
sportiv(cod_sportiv),
constraint apartine_pk_compus primary
key(cod_sala,cod_federatie,cod_sportiv)
);
```

```
create table PARTICIPA
```

```
(cod_sportiv number(5) constraint participa_sportiv not null,
cod_competitie number(5) constraint participa_competitie not null,
data_inscriere date default sysdate,
constraint participa_sportiv_fk foreign key(cod_sportiv) references
sportiv(cod_sportiv),
constraint participa_competitie_fk foreign key(cod_competitie) references
competitie(cod_competitie),
constraint participa_pk_compus primary key(cod_sportiv,cod_competitie)
```

```
);

create table SPONSORIZEAZA
(cod_sponsor number(5) constraint sponsorizeaza_sponsor not null,
cod_competitie number(5) constraint sponsorizeaza_competitie not null,
suma_oferita number(5) constraint sponsorizeaza_suma not null,
data_sponsorizarii date default sysdate,
constraint sponsorizeaza_sponsor_fk foreign key(cod_sponsor) references
sponsori(cod_sponsor),
constraint sponsorizeaza_competitie_fk foreign key(cod_competitie) references
competitie(cod_competitie),
constraint sponsorizeaza_pk_compus primary key(cod_sponsor,cod_competitie)
);
```

```
--Insert in locatie--
INSERT INTO LOCATIE
VALUES(1,'Neamt','Roman','BD.Roman-Musat','120A');
INSERT INTO LOCATIE
VALUES(2,'Bucuresti','Bucuresti','Calea Soarelui','451');
INSERT INTO LOCATIE
VALUES(3,'Bacau','Onesti','Mihai Viteazul','1B');
INSERT INTO LOCATIE
VALUES(4,'Bucuresti','Bucuresti','BD.Independentei','220K');
INSERT INTO LOCATIE
VALUES(5,'Brasov','Brasov','Vasile Alecsandi','98');
INSERT INTO LOCATIE
VALUES(6,'Sibiu','Sibiu','Al.I.Cuza','666');
INSERT INTO LOCATIE
VALUES(7,'Neamt','Roman','Tineretului','17C');
```

```
--insert in Sponsori
INSERT INTO SPONSORI
VALUES(10,'Facebook','mark@facebook.com');
INSERT INTO SPONSORI
VALUES(20,'KFC','office@kfc.com');
INSERT INTO SPONSORI
VALUES(30,'TonusGym','tonusgym@gmail.com');
INSERT INTO SPONSORI
VALUES(40,'Dani Baciu','dani@fitness.com');
INSERT INTO SPONSORI
VALUES(50,'McDonalds','office@mcdonalds.com');
```

```
--insert in sala_sport
INSERT INTO SALA_SPORT
VALUES(100,1,default,'Tonus Gym');
INSERT INTO SALA_SPORT
```



```
VALUES(200,3,2014,'StayFit');
INSERT INTO SALA_SPORT
VALUES(300,3,default,'OzerGolden');
INSERT INTO SALA_SPORT
VALUES(400,2,2010,'WorldClass');
INSERT INTO SALA_SPORT
VALUES(500,6,2019,'Fitness.EU');
INSERT INTO SALA_SPORT
VALUES(600,7,2016,'Fitness2000');
INSERT INTO SALA_SPORT
VALUES(700,2,default,'WellnessClub');

--insert in angajati
CREATE SEQUENCE SEQ_ANGAJATI_SALA
INCREMENT by 10
START WITH 110
MAXVALUE 1000
NOCYCLE;

INSERT INTO ANGAJATI_SALA
VALUES(SEQ_ANGAJATI_SALA.NEXTVAL,100,'Baciu','Daniel',to_date('18/09/2001','dd/mm/yyyy'),'m',default,15000,'administrator');
INSERT INTO ANGAJATI_SALA
VALUES(SEQ_ANGAJATI_SALA.NEXTVAL,100,'Ungureanu','Alex',to_date('10/01/1995','dd/mm/yyyy'),'m',to_date('10/11/2018','dd/mm/yyyy'),2100,'cleaner');
INSERT INTO ANGAJATI_SALA
VALUES(SEQ_ANGAJATI_SALA.NEXTVAL,300,'Adi','Bianca',to_date('08/10/1989','dd/mm/yyyy'),'f',to_date('08/10/1989','dd/mm/yyyy'),2500,'paznic');
INSERT INTO ANGAJATI_SALA
VALUES(SEQ_ANGAJATI_SALA.NEXTVAL,200,'Tudor','Gabriel',to_date('09/03/2000','dd/mm/yyyy'),'m',default,15000,'administrator');
INSERT INTO ANGAJATI_SALA
VALUES(SEQ_ANGAJATI_SALA.NEXTVAL,200,'Gabriel','Daniela',to_date('11/05/1999','dd/mm/yyyy'),'f',default,1500,'cleaner');
INSERT INTO ANGAJATI_SALA
VALUES(SEQ_ANGAJATI_SALA.NEXTVAL,500,'Ungureanu','Vlad',to_date('10/09/2002','dd/mm/yyyy'),'m',default,9000,'administrator');
INSERT INTO ANGAJATI_SALA
VALUES(SEQ_ANGAJATI_SALA.NEXTVAL,400,'Gabriela','Dancila',to_date('10/05/1976','dd/mm/yyyy'),'f',to_date('02/07/1995','dd/mm/yyyy'),2840,'paznic');

--insert in federatie
select * from federatie_de_sport;
INSERT INTO FEDERATIE_DE_SPORT
VALUES(10,'FederatiaCulturism',1977,'culturism',2);
INSERT INTO FEDERATIE_DE_SPORT
VALUES(20,'FederatiaFitness',1963,'fitness',3);
```

```
INSERT INTO FEDERATIE_DE_SPORT
VALUES(30,'Aruncarea Ciocanului',1954,'aruncare ciocan',2);
INSERT INTO FEDERATIE_DE_SPORT
VALUES(40,'FederatiaHaltere',1977,'haltere',2);
INSERT INTO FEDERATIE_DE_SPORT
VALUES(50,'FederatiaAlergare',1989,'alergare',1);
INSERT INTO FEDERATIE_DE_SPORT
VALUES(60,'FederatiaSaritCoarda',2002,'sarit coarda',5);

--insert in club
INSERT INTO CLUB
VALUES(1000,2,'Energia',default,'Gheorghe','Ionel');
INSERT INTO CLUB
VALUES(2000,1,'Otelu Galati',to_date('17/01/2007','dd/mm/yyyy'),'Baciu','David');
INSERT INTO CLUB
VALUES(3000,3,'Petrolul',to_date('19/09/1999','dd/mm/yyyy'),'Andrei','Petru');
INSERT INTO CLUB
VALUES(4000,4,'Steaua',to_date('08/11/2001','dd/mm/yyyy'),'Becali','Gigi');
INSERT INTO CLUB
VALUES(5000,2,'Dinamo',default,'Gheorghe','Mihai');
INSERT INTO CLUB
VALUES(6000,3,'Victoria',to_date('29/06/1985','dd/mm/yyyy'),'Ungureanu','Gigel');

--insert antrenori
INSERT INTO ANTRENORI
VALUES(1,2000,'Petrescu','Andrei',to_date('12/03/1965','dd/mm/yyyy'),'0712345678',
'default');
INSERT INTO ANTRENORI
VALUES(2,1000,'Baciu','Daniel',to_date('17/09/2001','dd/mm/yyyy'),'0756119760',to_
_date('17/05/2017','dd/mm/yyyy'));
INSERT INTO ANTRENORI
VALUES(3,2000,'Frone','Gigel',to_date('12/11/1977','dd/mm/yyyy'),'0747812563',to_
date('10/07/2007','dd/mm/yyyy'));
INSERT INTO ANTRENORI
VALUES(4,3000,'Andrei','Alex',to_date('24/07/1990','dd/mm/yyyy'),'0767238145',to_
date('07/11/2020','dd/mm/yyyy'));
INSERT INTO ANTRENORI
VALUES(5,4000,'Klaus','Werner',to_date('10/12/2000','dd/mm/yyyy'),'0756347128',t
o_date('01/01/2010','dd/mm/yyyy'));

--insert in sportiv

CREATE SEQUENCE SEQ_SPORTIV
INCREMENT by 100
START WITH 3000
MAXVALUE 10000
NOCYCLE;
```

```
INSERT INTO SPORTIV
VALUES(SEQ_SPORTIV.NEXTVAL,
1000,'Nastase','Petru',to_date('25/05/1981','dd/mm/yyyy'),'0711211212');
INSERT INTO SPORTIV
VALUES(SEQ_SPORTIV.NEXTVAL,
2000,'Cosnete','Benone',to_date('20/07/1967','dd/mm/yyyy'),'0755555555');
INSERT INTO SPORTIV
VALUES(SEQ_SPORTIV.NEXTVAL, 1000,'Baciu','Daniel-
Mihai',to_date('18/09/2001','dd/mm/yyyy'),'0744444444');
INSERT INTO SPORTIV
VALUES(SEQ_SPORTIV.NEXTVAL,
3000,'Arici','Tudor',to_date('09/12/2000','dd/mm/yyyy'),'0777777777');
INSERT INTO SPORTIV
VALUES(SEQ_SPORTIV.NEXTVAL,
2000,'Ungureanu','Amanda',to_date('25/09/1979','dd/mm/yyyy'),'0799999999');
INSERT INTO SPORTIV
VALUES(SEQ_SPORTIV.NEXTVAL,
5000,'Ungureanu','Florin',to_date('05/11/1988','dd/mm/yyyy'),'0700000000');
INSERT INTO SPORTIV
VALUES(SEQ_SPORTIV.NEXTVAL,
3000,'Baciu','Felicia',to_date('13/03/1994','dd/mm/yyyy'),'0733333333');

--insert in club_history

INSERT INTO CLUB_HISTORY
VALUES(3000,to_date('17/06/2007','dd/mm/yyyy'),2000,to_date('18/07/2010','dd/m
m/yyyy'));
INSERT INTO CLUB_HISTORY
VALUES(3000,to_date('19/07/2010','dd/mm/yyyy'),3000,to_date('20/03/2018','dd/m
m/yyyy'));
INSERT INTO CLUB_HISTORY
VALUES(3300,to_date('10/09/2006','dd/mm/yyyy'),1000,to_date('18/07/2010','dd/m
m/yyyy'));
INSERT INTO CLUB_HISTORY
VALUES(3200,to_date('17/06/2007','dd/mm/yyyy'),2000,to_date('18/07/2010','dd/m
m/yyyy'));
INSERT INTO CLUB_HISTORY
VALUES(3100,to_date('01/01/2001','dd/mm/yyyy'),4000,to_date('04/09/2017','dd/m
m/yyyy'));
INSERT INTO CLUB_HISTORY
VALUES(3100,to_date('05/09/2017','dd/mm/yyyy'),3000,to_date('25/11/2019','dd/m
m/yyyy'));
INSERT INTO club_history
VALUES(3100,to_date('30/12/2019','dd/mm/yyyy'),1000,sysdate);
```

```
--insert in competitii
INSERT INTO COMPETITIE
VALUES(10,10,2,to_date('08/10/2022','dd/mm/yyyy'),10000);
INSERT INTO COMPETITIE
VALUES(20,10,5,to_date('27/05/2023','dd/mm/yyyy'),5000);
INSERT INTO COMPETITIE
VALUES(30,30,4,to_date('18/04/2022','dd/mm/yyyy'),1000);
INSERT INTO COMPETITIE
VALUES(40,20,2,to_date('20/07/2023','dd/mm/yyyy'),7000);
INSERT INTO COMPETITIE
VALUES(50,50,5,to_date('02/12/2025','dd/mm/yyyy'),40000);
INSERT INTO COMPETITIE
VALUES(60,40,3,to_date('12/08/2021','dd/mm/yyyy'),15000);
```

--inserari in tabele asociative

```
INSERT INTO PARTICIPA
VALUES(3000,40,default);
INSERT INTO PARTICIPA
VALUES(3000,30,default);
INSERT INTO PARTICIPA
VALUES(3100,50,to_date('19/04/2021','dd/mm/yyyy'));
INSERT INTO PARTICIPA
VALUES(3200,10,to_date('09/01/2021','dd/mm/yyyy'));
INSERT INTO PARTICIPA
VALUES(3000,50,default);
INSERT INTO PARTICIPA
VALUES(3500,10,to_date('27/09/2020','dd/mm/yyyy'));
INSERT INTO PARTICIPA
VALUES(3400,60,default);
INSERT INTO PARTICIPA
VALUES(3500,20,default);
INSERT INTO PARTICIPA
VALUES(3600,60,to_date('10/12/2020','dd/mm/yyyy'));
INSERT INTO PARTICIPA
VALUES(3300,50,default);
INSERT INTO PARTICIPA
VALUES(3200,20,to_date('14/11/2019','dd/mm/yyyy'));
INSERT INTO PARTICIPA
VALUES(3600,30,default);
```

```
INSERT INTO APARTINE
VALUES(100,10,3000);
INSERT INTO APARTINE
VALUES(100,10,3100);
```

```
INSERT INTO APARTINE  
VALUES(400,10,3200);  
INSERT INTO APARTINE  
VALUES(100,20,3000);  
INSERT INTO APARTINE  
VALUES(200,40,3000);  
INSERT INTO APARTINE  
VALUES(400,50,3400);  
INSERT INTO APARTINE  
VALUES(300,40,3100);  
INSERT INTO APARTINE  
VALUES(500,50,3300);  
INSERT INTO APARTINE  
VALUES(600,60,3600);  
INSERT INTO APARTINE  
VALUES(700,30,3500);  
INSERT INTO APARTINE  
VALUES(500,30,3600);
```

```
INSERT INTO SPONSORIZEAZA  
VALUES(20,10,5000,default);  
INSERT INTO SPONSORIZEAZA  
VALUES(50,10,5000,default);  
INSERT INTO SPONSORIZEAZA  
VALUES(10,40,2700,to_date('09/12/2020','dd/mm/yyyy'));  
INSERT INTO SPONSORIZEAZA  
VALUES(50,40,4300,to_date('12/02/2021','dd/mm/yyyy'));  
INSERT INTO SPONSORIZEAZA  
VALUES(10,60,1000,to_date('19/10/2020','dd/mm/yyyy'));  
INSERT INTO SPONSORIZEAZA  
VALUES(50,60,5000,to_date('01/04/2021','dd/mm/yyyy'));  
INSERT INTO SPONSORIZEAZA  
VALUES(30,60,9000,to_date('19/11/2020','dd/mm/yyyy'));  
INSERT INTO SPONSORIZEAZA  
VALUES(50,20,2000,to_date('23/06/2020','dd/mm/yyyy'));  
INSERT INTO SPONSORIZEAZA  
VALUES(20,30,200,to_date('07/05/2021','dd/mm/yyyy'));  
INSERT INTO SPONSORIZEAZA  
VALUES(10,30,300,to_date('31/12/2020','dd/mm/yyyy'));
```

a. Angajati_sala

COD_ANGAJAT	COD_SALA	NUME	PRENUME	DATA_NASTERE	SEX	DATA_ANGAJARII	SALARIU	JOB
110	100	Baciu	Daniel	18-SEP-01	m	30-DEC-21	15000	administrator
120	100	Ungureanu	Alex	10-JAN-95	m	10-NOV-18	2100	cleaner
130	300	Adi	Bianca	08-OCT-89	f	08-OCT-89	2500	paznic
140	200	Tudor	Gabriel	09-MAR-00	m	30-DEC-21	15000	administrator
150	200	Gabriel	Daniela	11-MAY-99	f	30-DEC-21	1500	cleaner
160	500	Ungureanu	Vlad	10-SEP-02	m	30-DEC-21	9000	administrator
170	400	Gabriela	Dancila	10-MAY-76	f	02-JUL-95	2840	paznic

b. Sala_sport

COD_SALA	COD_LOCATIE	AN_INFIINTARE	DENUMIRE
100	1	2021	Tonus Gym
200	3	2014	StayFit
300	3	2021	OzerGolden
400	2	2010	WorldClass
500	6	2019	Fitness.EU
600	7	2016	Fitness2000
700	2	2021	WellnessClub

c. Sponsori

COD_SPONSOR	DENUMIRE_SPONSOR	ADRESA_EMAIL
1	10 Facebook	mark@facebook.com
2	20 KFC	office@kfc.com
3	30 TonusGym	tonusgym@gmail.com
4	40 Dani Baciu	dani@fitness.com
5	50 McDonalds	office@mcdonalds.com

d. Competitie

COD_COMPETITIE	COD_FEDERATIE	COD_LOCATIE	DATA_COMPETITIE	VALOARE_TOTALA_PREMII
10	10	2	08-OCT-22	10000
20	10	5	27-MAY-23	5000
30	30	4	18-APR-22	1000
40	20	2	20-JUL-23	7000
50	50	5	02-DEC-25	40000
60	40	3	12-AUG-21	15000

e. Locatie

COD_LOCATIE	JUDET	ORAS	STRADA	NUMAR
1	Neamt	Roman	BD.Roman-Musat	120A
2	Bucuresti	Bucuresti	Calea Soarelui	451
3	Bacau	Onesti	Mihai Viteazul	1B
4	Bucuresti	Bucuresti	BD.Independentei	220K
5	Brasov	Brasov	Vasile Alecsandi	98
6	Sibiu	Sibiu	Al.I.Cuza	666
7	Neamt	Roman	Tineretului	17C

f. Club

COD_CLUB	COD_LOCATIE	DENUMIRE	DATA_INFIINTARE_CLUB	NUME_ADMIN	PRENUME_ADMIN
1000	2	Energia	25-MAY-21	Gheorghe	Ionel
2000	1	Otelu Galati	17-JAN-07	Baciu	David
3000	3	Petrolul	19-SEP-99	Andrei	Petru
4000	4	Steaua	08-NOV-01	Becali	Gigi
5000	2	Dinamo	25-MAY-21	Gheorghe	Mihai
6000	3	Victoria	29-JUN-85	Ungureanu	Gigel

g. Antrenori

COD_ANTRENOR	COD_CLUB	NUME	PRENUME	DATA_NASTERE	NRTELEFON	DATA_INSCRIERE_CLUB
1	2000	Petrescu	Andrei	12-MAR-65	0712345678	25-MAY-21
2	1000	Baciu	Daniel	17-SEP-01	0756119760	17-MAY-17
3	2000	Frone	Gigel	12-NOV-77	0747812563	10-JUL-07
4	3000	Andrei	Alex	24-JUL-90	0767238145	07-NOV-20
5	4000	Klaus	Werner	10-DEC-00	0756347128	01-JAN-10

h. Club_history

COD_SPORTIV	DATA_START	COD_CLUB	DATA_FINAL
3000	17-JUN-07	2000	18-JUL-10
3000	19-JUL-10	3000	20-MAR-18
3300	10-SEP-06	1000	18-JUL-10
3200	17-JUN-07	2000	18-JUL-10
3100	01-JAN-01	4000	04-SEP-17
3100	05-SEP-17	3000	25-NOV-19
3100	30-DEC-19	1000	30-DEC-21

i. Federatie_de_sport

COD_FEDERATIE	DENUMIRE	AN_INFIINTARE	SPORT_PRACTICAT	COD_LOCATIE
10	FederatiaCulturism	1977	culturism	2
20	FederatiaFitness	1963	fitness	3
30	Aruncarea Ciocanului	1954	aruncare ciocan	2
40	FederatiaHaltere	1977	haltere	2
50	FederatiaAlergare	1989	alergare	1
60	FederatiaSaritCoarda	2002	sarit coarda	5

j. Sportiv

COD_SPORTIV	COD_CLUB	NUME	PRENUME	DATA_NASTERE	NRTELEFON
3000	1000	Nastase	Petru	25-MAY-81	0711211212
3100	2000	Cosnete	Benone	20-JUL-67	0755555555
3200	1000	Baciu	Daniel-Mihai	18-SEP-01	0744444444
3300	3000	Arici	Tudor	09-DEC-00	0777777777
3400	2000	Ungureanu	Amanda	25-SEP-79	0799999999
3500	5000	Ungureanu	Florin	05-NOV-88	0700000000
3600	3000	Baciu	Felicia	13-MAR-94	0733333333

k. Apartine

COD_SALA	COD_FEDERATIE	COD_SPORTIV
100	10	3000
100	10	3100
100	20	3000
200	40	3000
300	40	3100
400	10	3200
400	50	3400
500	30	3600
500	50	3300
600	60	3600
700	30	3500

l. Participa

COD_SPORTIV	COD_COMPETITIE	DATA_INSCRIERE
3000	40	30-DEC-21
3000	30	30-DEC-21
3100	50	19-APR-21
3200	10	09-JAN-21
3000	50	30-DEC-21
3500	10	27-SEP-20
3400	60	30-DEC-21
3500	20	30-DEC-21
3600	60	10-DEC-20
3300	50	30-DEC-21
3200	20	14-NOV-19
3600	30	30-DEC-21

m.Sponsorizeaza

COD_SPONSOR	COD_COMPETITIE	SUMA_OFERITA	DATA_SPONSORIZARII
20	10	5000	25-MAY-21
50	10	5000	25-MAY-21
10	40	2700	09-DEC-20
50	40	4300	12-FEB-21
10	60	1000	19-OCT-20
50	60	5000	01-APR-21
30	60	9000	19-NOV-20
50	20	2000	23-JUN-20
20	30	200	07-MAY-21
10	30	300	31-DEC-20

- I. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze două tipuri de colecție studiate. Apelați subprogramul.

--pentru o locatie primita ca parametru, sa se afiseze suma totala obtinuta din sponsorizari

--impreuna cu numele sponsorilor si data la care au contribuit la aceasta sponsorizare

--pentru primele 3 competitii dupa data competitiei

```
CREATE OR REPLACE PROCEDURE prog_sponsorizari( cod_loc
competitie.cod_locatie%type )
```

```
IS
```

```
TYPE rec_num_data IS RECORD
```

```
(nume sponsori.denumire_sponsor%type,
```

```
data_sponsorizarii sponsorizeaza.data_sponsorizarii%type);
```

```
TYPE tablou_sponsorizare IS TABLE OF rec_num_data;
```

```
TYPE vector_coduri_comp IS VARRAY(3) OF competitie.cod_competitie%type;
```

```
tab_sponsori tablou_sponsorizare;
```

```
vec_comp vector_coduri_comp;
```



```
    suma_totala NUMBER := 0;
BEGIN
    SELECT cod_competitie
    BULK COLLECT INTO vec_comp
    FROM competitie where cod_locatie = cod_loc AND rownum <= 3
    ORDER BY data_competitie;

    FOR i IN vec_comp.FIRST..vec_comp.LAST LOOP
        IF vec_comp.exists(i) THEN
            SELECT sum(suma_oferita)
            INTO suma_totala
            FROM SPONSORIZEAZA
            WHERE COD_COMPETITIE = vec_comp(i);

            DBMS_OUTPUT.PUT_LINE('DETALII DESPRE COMPETITIA ' || vec_comp(i) ||
':');
            DBMS_OUTPUT.PUT_LINE('Suma totala : ' || suma_totala);

            SELECT denumire_sponsor, data_sponsorizarii
            BULK COLLECT INTO tab_sponsori
            FROM SPONSORI S JOIN SPONSORIZEAZA SP ON (SP.COD_SPONSOR =
S.COD_SPONSOR)
            WHERE SP.COD_COMPETITIE = vec_comp(i);

            FOR i IN tab_sponsori.FIRST..tab_sponsori.LAST LOOP
                IF tab_sponsori.exists(i) THEN
                    DBMS_OUTPUT.PUT_LINE('Sponsorul ' || i || ' este ' ||
tab_sponsori(i).nume || ' si a donat in data de - ' ||
tab_sponsori(i).data_sponsorizarii);
                END IF;
            END LOOP;
            tab_sponsori.delete();
        END IF;
    END LOOP;

END PROG_SPONSORIZARI;
/

DECLARE
    cod NUMBER := '&cod_locatie';
BEGIN
    prog_sponsorizari(cod);
END;
/
```

```

--pentru o locatie primita ca parametru, sa se afiseze suma totala obtinuta din sponsorizari
--impreuna cu numele sponsorilor si data la care au contribuit la aceasta sponsorizare
--pentru primele 3 competitii dupa data competitiei
CREATE OR REPLACE PROCEDURE prog_sponsorizari( cod_loc competitie.cod_locatie%type )
IS
    TYPE rec_nume_data IS RECORD
    (
        nume sponsori.denumire_sponsor%type,
        data_sponsorizarii sponsorizeaza.data_sponsorizarii%type);
    TYPE tablou_sponsorizare IS TABLE OF rec_nume_data;
    TYPE vector_coduri_comp IS VARRAY(3) OF competitie.cod_competitie%type;

    --sum[cod_competitie] += suma_oferita
    tab_sponsori tablou_sponsorizare;
    vec_comp vector_coduri_comp;
    suma_totala NUMBER := 0;

BEGIN
    SELECT cod_competitie
    BULK COLLECT INTO vec_comp
    FROM competitie where cod_locatie = cod_loc AND rownum <= 3
    ORDER BY data_competitie;

    FOR i IN vec_comp.FIRST..vec_comp.LAST LOOP
        IF vec_comp.exists(i) THEN
            SELECT sum(suma_oferita)
            INTO suma_totala
            FROM SPONSORIZEAZA
            WHERE COD_COMPETITIE = vec_comp(i);

            DBMS_OUTPUT.PUT_LINE('DETALII DESPRE COMPETITIA ' || vec_comp(i) || ':');
            DBMS_OUTPUT.PUT_LINE('Suma totala : ' || suma_totala);

        SELECT denumire_sponsor, data_sponsorizarii

```

```

DETALII DESPRE COMPETITIA 10:
Suma totala : 10000
Sponsorul 1 este KFC si a donat in data de - 02-JAN-2
Sponsorul 2 este McDonalds si a donat in data de - 02
DETALII DESPRE COMPETITIA 40:
Suma totala : 7000
Sponsorul 1 este Facebook si a donat in data de - 09-
Sponsorul 2 este McDonalds si a donat in data de - 12

```

J. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

--sa se scrie codul si numele federatiilor, codul si numele sportivilor care participa la competitii la un cod de locatie dat

```

CREATE OR REPLACE PROCEDURE prog_sportivi_fed(cod club.cod_locatie%type)
IS
    cod_f federatie_de_sport.cod_federatie%type;
    den federatie_de_sport.denumire%type;
    c_sportivi SYS_REFCURSOR;

    TYPE sportiv_record IS RECORD
    (
        cod_s sportiv.cod_sportiv%type,
        nume_s sportiv.nume%type,
        prenume_s sportiv.prenume%type
    );
    sportiv_detalii sportiv_record;
    CURSOR c_fest IS SELECT f.cod_federatie, f.denumire,
        CURSOR (
            SELECT s.cod_sportiv, s.nume, s.prenume
            FROM sportiv s JOIN apartine a ON (a.cod_sportiv =
s.cod_sportiv)
            WHERE a.cod_federatie = f.cod_federatie)
        FROM FEDERATIE_DE_SPORT f JOIN COMPETITIE c ON (f.cod_federatie
= c.cod_federatie)
        WHERE c.cod_locatie = cod;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Codul locatiei este : ' || cod);

```

```

OPEN c_fest;

LOOP
    FETCH c_fest INTO cod_f, den, c_sportivi;
    EXIT WHEN c_fest%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Cod federatie : ' || cod_f || '. Nume
federatie : ' || den || ' are sportivii :');

    LOOP
        FETCH c_sportivi INTO sportiv_detalii;
        EXIT WHEN c_sportivi%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('Cod sportiv (' || sportiv_detalii.cod_s || ') pe care il
cheama ' || sportiv_detalii.num_s || ' ' || sportiv_detalii.prenume_s || '.');

    END LOOP;
END LOOP;

CLOSE c_fest;

END prog_sportivi_fed;

/

DECLARE
    cod club.cod_locatie%type := '&cod_locatie';
BEGIN
    prog_sportivi_fed(cod);
END;
/

```

```

--sa se scrie codul si numele federatiilor, codul si numele
--sportivilor care participa la competitii la un cod de locatie dat
CREATE OR REPLACE PROCEDURE prog_sportivi_fed(cod club.cod_locatie%type)
IS
    cod_f federatie_de_sport.cod_federatie%type;
    den federatie_de_sport.denumire%type;
    c_sportivi SYS_REFCURSOR;

    TYPE sportiv_record IS RECORD
    (
        cod_s sportiv.cod_sportiv%type,
        nume_s sportiv.num_s%type,
        prenume_s sportiv.prenume%type
    );
    sportiv_detalii sportiv_record;
    CURSOR c_fest IS SELECT f.cod_federatie, f.denumire,
        CURSOR (
            SELECT s.cod_sportiv, s.num_s, s.prenume
            FROM sportiv s JOIN apartina a ON (a.cod_sportiv = s.cod_sp
            WHERE a.cod_federatie = f.cod_federatie)
        FROM FEDERATIE_DE_SPORT f JOIN COMPETITIE c ON (f.cod_federatie = c.cod_federatie)
        WHERE c.cod_locatie = cod;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Codul locatiei este : ' || cod);

    OPEN c_fest;

    LOOP
        FETCH c_fest INTO cod_f, den, c_sportivi;
        EXIT WHEN c_fest%NOTFOUND;

```

Codul locatiei este : 2
 Cod federatie : 10. Nume federatie : FederatiaCultur
 Cod sportiv (3000) pe care il cheama Nastase Petru.
 Cod sportiv (3100) pe care il cheama Cosneta Benone
 Cod sportiv (3200) pe care il cheama Baciu Daniel-M
 Cod federatie : 20. Nume federatie : FederatiaFitness
 Cod sportiv (3000) pe care il cheama Nastase Petru.

Task completed in 1.634 seconds

PL/SQL procedure successfully completed.

K. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate

--Sa se afiseze primul club (luat in ordine crescatoare dupa cod) la care sportivii au varsta < 30 ani si care

--au participat la competitia din anul dat ca parametru sponsorizata de anumit sponsor

```
CREATE OR REPLACE FUNCTION fct_cluburi_an_sponsor( an number,
                                                    nume_spons sponsori.denumire_sponsor%type)
RETURN club%rowtype
IS
    informatii club%rowtype;

BEGIN
    SELECT c.cod_club, c.cod_locatie, c.denumire, c.data_infiintare_club,
    c.num_e_admin, c.prenume_admin
    INTO informatii
    FROM club c JOIN sportiv s ON (c.cod_club=s.cod_club)
    JOIN participa p ON (s.cod_sportiv=p.cod_sportiv) JOIN competitie co ON
    (co.cod_competitie=p.cod_competitie)
    JOIN sponsorizeaza a ON (co.cod_competitie=a.cod_competitie)
    JOIN sponsori b ON (a.cod_sponsor=b.cod_sponsor)
    WHERE months_between(sysdate,s.data_nastere) < 30*12 AND
    to_number(to_char(co.data_competitie,'yyyy'))= an AND
    lower(b.denumire_sponsor) = nume_spons AND rownum <= 1
    ORDER BY c.cod_club;

    RETURN informatii;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu s-au gasit cluburi care sa indeplineasca
    cerintele');
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Mai multe cluburi indeplinesc cerintele!');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');
```

```

END fct_cluburi_an_sponsor;
/
DECLARE
    an number := '&an';
    numele_sponsorului sponsori.denumire_sponsor%type := '&nume_sponsor';
    info club%rowtype;
BEGIN
    info := fct_cluburi_an_sponsor(an, numele_sponsorului);
    DBMS_OUTPUT.PUT_LINE('Date intrare : an - ' || an || ' si nume - '
    || numele_sponsorului);
    DBMS_OUTPUT.PUT_LINE('Cod club (' || info.cod_club || '), cunoscut si sub numele
de ');
    DBMS_OUTPUT.PUT_LINE(info.denumire || ' este infiintat in data de
' || info.data_infiintare_club || ' si este administrat de ');
    DBMS_OUTPUT.PUT_LINE(info.num_admin || ' ' || info.preume_admin || ' si
indeplineste conditiile cerute');
    --am pus mai multe dbms ca se se vada mai frumos, sa nu apara o linie lunga
END;
/

```

The screenshot shows the Oracle SQL Developer interface. The main window displays a PL/SQL procedure named `fct_cluburi_an_sponsor`. The procedure takes an `an` (year) and a `nume_sponsor` (sponsor name) as input. It uses a complex SQL query to find clubs that match the criteria: the club's location is the same as the sponsor's, the club's name contains the sponsor's name, and the club's founding date is within 30 months of the current date. The query also filters for clubs that are currently active and have a non-zero number of rows. The procedure returns the club's details as a row type.

The right-hand pane shows the output of the procedure execution. The output is as follows:

```

Date intrare : an - 2022 si nume - kfc
Cod club (1000), cunoscut si sub numele de
Energia este infiintat in data de 30-DEC-21 si este adm
Gheorghe Ionel si indeplineste conditiile cerute

```

The bottom status bar indicates that the PL/SQL procedure was successfully completed.

```

Worksheet | Query Builder
--
EXCEPTION
WHEN NO_DATA_FOUND THEN
  RAISE_APPLICATION_ERROR(-20000, 'Nu s-au gasit cluburi care sa indeplineasca cerintele');
WHEN TOO_MANY_ROWS THEN
  RAISE_APPLICATION_ERROR(-20001, 'Mai multe cluburi indeplinesc cerintele!');
WHEN OTHERS THEN
  RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');

END fct_cluburi_an_sponsor;
/

DECLARE
  an number := '&an';
  numele_sponsorului sponsori.denumire_sponsor%type := '&nume_sponsor';
  info club%rowtype;
BEGIN
  info := fct_cluburi_an_sponsor(an, numele_sponsorului);
  DBMS_OUTPUT.PUT_LINE('Date intrare : an - '|| an || ' si nume - '|| numele_sponsorului);
  DBMS_OUTPUT.PUT_LINE('Cod club ('|| info.cod_club || '), cunoscut si sub numele de ');
  DBMS_OUTPUT.PUT_LINE(info.denumire || ' este infiintat in data de '|| info.data_infiintare_club || ' si este administrat de ');
  DBMS_OUTPUT.PUT_LINE(info.nume_admin || ' '|| info.prenume_admin || ' si indeplineste conditiile cerute');
  --am pus mai multe dbms ca se se vada mai frumos, sa nu apara o linie lunga
END;
/

Script Output
--
Task completed in 7.699 seconds

END;
Error report -
ORA-20000: Nu s-au gasit cluburi care sa indeplineasca cerintele
ORA-06512: at "DANI.FCT_CLUBURI_AN_SPONSOR", line 23
ORA-06512: at line 6
20000. 00000 - "$s"
*Cause:      The stored procedure 'raise_application_error'
              was called which causes this error to be generated.

```

L. Formulati în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat 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**. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

--Afisati sportivul care a fost la cele mai multe competitii de sport sponsorizate de o anumita firma,
 --si nr de competitii. (aici a trebuit sa mai adaug un sportiv, insertul il gasiti pe ultimul rand)

```

CREATE OR REPLACE PROCEDURE p_9_sport(nume_spons
sponsori.denumire_sponsor%type)
IS
  CURSOR c IS select s.cod_sportiv cod, initcap(s.nume), initcap(s.prenume), count(*)
  numar

```

```

    from sportiv s JOIN participa p on (s.cod_sportiv = p.cod_sportiv)
    JOIN competitie c on (p.cod_competitie = c.cod_competitie)
    JOIN sponsorizeaza s1 on (s1.cod_competitie = c.cod_competitie)
    JOIN sponsori s2 on (s1.cod_sponsor = s2.cod_sponsor)
    where initcap(nume_spons) = initcap(denumire_sponsor)
    group by s.cod_sportiv, initcap(s.nume), initcap(s.prenume);

```

```

TYPE rec_inreg IS RECORD
  (cod_s sportiv.cod_sportiv%type,
  nume sportiv.nume%type,
  prenume sportiv.prenume%type,
  nr NUMBER);
inreg rec_inreg;

```

```
    maxim NUMBER := 0;
    frecv NUMBER :=0;
BEGIN
    FOR inr IN c LOOP
        IF inr.numar > maxim THEN
            maxim := inr.numar;
            inreg := inr;
            frecv := 1;
        ELSIF inr.numar = maxim THEN
            frecv := frecv + 1;
        END IF;
    END LOOP;

    IF maxim = 0 THEN
        RAISE NO_DATA_FOUND;
    END IF;

    IF frecv > 1 THEN
        RAISE TOO_MANY_ROWS;
    END IF;

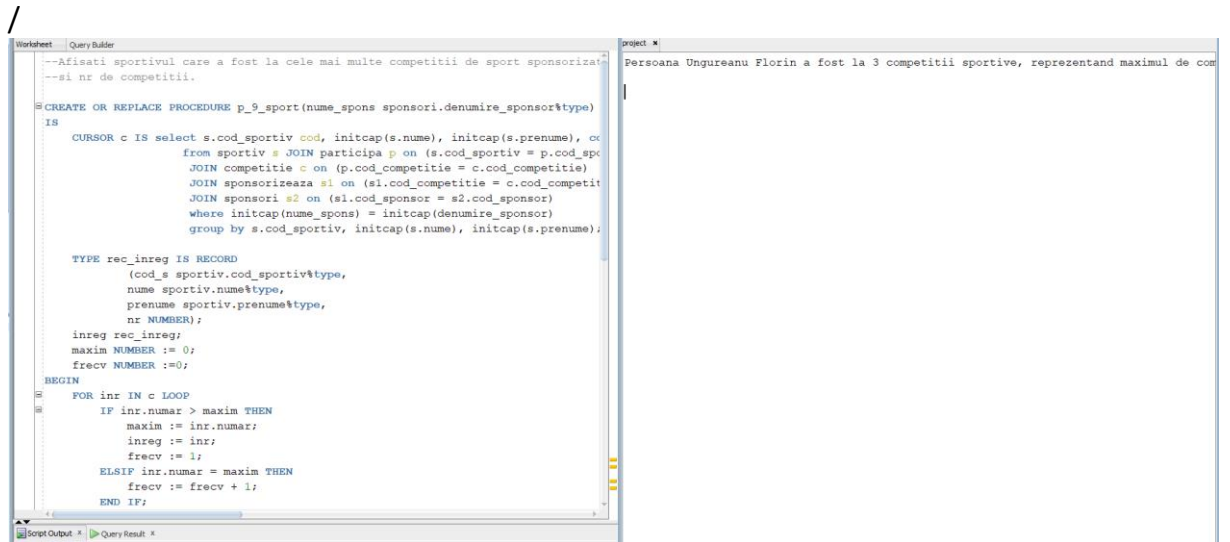
    DBMS_OUTPUT.PUT_LINE('Persoana ' || inreg.numa || ' ' || inreg.prenume || ' a
fost la ' || inreg.nr || ' competitii sportive, reprezentand maximul de competitii la care
a fost o pers dintre competiile sponsorizate de ' || nume_spons);

EXCEPTION
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Mai multi sportivi indeplinesc cerinta.');
```

WHEN NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR(-20000, 'Nu este indeplinita cerinta');

WHEN OTHERS THEN
RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!!!!');

```
END p_9_sport;
/
DECLARE
    nume_spons sponsori.denumire_sponsor%type := '&denumire_sponsor';
BEGIN
    p_9_sport(nume_spons);
END;
/
select * from sponsori;
select * from sponsorizeaza
order by cod_sponsor;
select * from competitie;
select * from participa;
/
insert into participa values (3500, 40, sysdate);
```



The screenshot displays a SQL Query Builder window with two panes. The left pane, titled 'Worksheet', contains a PL/SQL procedure named 'p_9_sport' and a cursor 'c'. The procedure is designed to find the sport with the most sponsors. The cursor 'c' is defined as a SELECT statement joining 'sportiv', 'participa', 'competitie', 'sponsorizeaza', and 'sponsor' tables. The right pane, titled 'Project', shows the query result for the procedure, indicating that 'Persoana Ungureanu Florin' has participated in 3 sports, representing the maximum number of sports.

```
--Afisati sportivul care a fost la cele mai multe competitii de sport sponsorizat
--si nr de competitii.

CREATE OR REPLACE PROCEDURE p_9_sport(nume_spons sponsor.denumire_sponsor%type)
IS
    CURSOR c IS select s.cod_sportiv cod, initcap(s.nume), initcap(s.prenume), c
    from sportiv s JOIN participa p on (s.cod_sportiv = p.cod_sportiv)
    JOIN competitie c on (p.cod_competitie = c.cod_competitie)
    JOIN sponsorizeaza s1 on (s1.cod_competitie = c.cod_competitie)
    JOIN sponsor s2 on (s1.cod_sponsor = s2.cod_sponsor)
    where initcap(nume_spons) = initcap(denumire_sponsor)
    group by s.cod_sportiv, initcap(s.nume), initcap(s.prenume);

    TYPE rec_inreg IS RECORD
    (cod_sportiv cod_sportiv%type,
    nume sportiv.nume%type,
    prenume sportiv.prenume%type,
    nr NUMBER);

    inreg rec_inreg;
    maxim NUMBER := 0;
    frecv NUMBER := 0;
BEGIN
    FOR inr IN c LOOP
        IF inr.nr > maxim THEN
            maxim := inr.nr;
            inreg := inr;
            frecv := 1;
        ELSIF inr.nr = maxim THEN
            frecv := frecv + 1;
        END IF;
    END LOOP;
END;
```

Persoana Ungureanu Florin a fost la 3 competitii sportive, reprezentand maximul de com

The screenshot displays the Oracle SQL Developer environment. The top pane shows a PL/SQL script in the 'Query Builder' window. The script includes an IF statement for 'frecv' and an EXCEPTION block with three handlers: 'TOO_MANY_ROWS', 'NO_DATA_FOUND', and 'OTHERS'. The 'NO_DATA_FOUND' handler is highlighted with a red box. Below the script, the 'Script Output' window shows the execution results. The first execution, with input 'DaniBaciu', results in an error 'ORA-20000: Nu este indeplinita cerinta'. The second execution, with input 'kfc', results in an error 'ORA-20001: Mai multi sportivi indeplinesc cerinta'.

```

Worksheet      Query Builder
IF frecv > 1 THEN
    RAISE TOO_MANY_ROWS;
END IF;

DBMS_OUTPUT.PUT_LINE('Persoana '||inreg.nume||' '||inreg.prenume||' a fost la '||inreg.nr||' competitii sporti');

EXCEPTION
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Mai multi sportivi indeplinesc cerinta.');
```

```

-- WHEN NO_DATA_FOUND THEN
--     RAISE_APPLICATION_ERROR(-20000, 'Nu este indeplinita cerinta');
```

```

    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!!!!');
END p_9_sport;
/
DECLARE
    nume_spons sponsori.denumire_sponsor%type := '&denumire_sponsor';
BEGIN
    p_9_sport(nume_spons);
END;
/
select * from sponsori;
```

Script Output x Query Result x
Task completed in 5.181 seconds

Error starting at line : 49 in command -
DECLARE
nume_spons sponsori.denumire_sponsor%type := '&denumire_sponsor';
BEGIN
p_9_sport(nume_spons);
END;
Error report -
ORA-20000: Nu este indeplinita cerinta
ORA-06512: at "DANI.P_9_SPORT", line 44
ORA-06512: at line 4

```

IF maxim = 0 THEN
    RAISE NO_DATA_FOUND;
END IF;

IF frecv > 1 THEN
    RAISE TOO_MANY_ROWS;
END IF;

DBMS_OUTPUT.PUT_LINE('Persoana '||inreg.nume||' '||inreg.prenume||' a fost la '||inreg.nr||' competitii sporti');

EXCEPTION
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Mai multi sportivi indeplinesc cerinta.');
```

```

    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu este indeplinita cerinta');
```

```

    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!!!!');
END p_9_sport;
/
DECLARE
    nume_spons sponsori.denumire_sponsor%type := '&denumire_sponsor';
BEGIN
```

Script Output x
Task completed in 3.641 seconds

```

nume_spons sponsori.denumire_sponsor%type := '&denumire_sponsor';
BEGIN
    p_9_sport(nume_spons);
END;
Error report -
ORA-20001: Mai multi sportivi indeplinesc cerinta.
ORA-06512: at "DANI.P_9_SPORT", line 42
ORA-06512: at line 4
```

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

–Creați un trigger la nivel de comandă care actualizează valoarea premiilor la –competiții

CREATE OR REPLACE TRIGGER t_10

AFTER INSERT OR DELETE OR UPDATE ON sponsorizeaza

DECLARE

```

    CURSOR curs IS select cod_competitie, sum(suma_oferita) suma from
    sponsorizeaza group by cod_competitie;
BEGIN
    FOR a IN curs LOOP
        UPDATE competitie
        SET valoare_totala_premii = a.suma
        WHERE cod_competitie = a.cod_competitie;
    END LOOP;
END t_10;
/
BEGIN
    INSERT INTO sponsorizeaza
    VALUES(40, 20, 10000, sysdate);
END;
/

```

–before

The screenshot shows the SQL Developer interface. The top pane displays a PL/SQL script for a trigger named t_10. The script is designed to update the 'valoare_totala_premii' column in the 'competitie' table whenever a new row is inserted into the 'sponsorizeaza' table. The trigger uses a cursor to calculate the sum of 'suma_oferita' for each 'cod_competitie'.

```

CREATE OR REPLACE TRIGGER t_10
AFTER INSERT OR DELETE OR UPDATE ON sponsorizeaza
DECLARE
    CURSOR curs IS select cod_competitie, sum(suma_oferita) suma from sponsorizeaza gr
BEGIN
    FOR a IN curs LOOP
        UPDATE competitie
        SET valoare_totala_premii = a.suma
        WHERE cod_competitie = a.cod_competitie;
    END LOOP;
END t_10;
/
BEGIN
    INSERT INTO sponsorizeaza
    VALUES (40, 20, 10000, sysdate);
END;
/
select * from competitie;

```

The bottom pane shows the 'Query Result' tab, which displays the data from the 'competitie' table after the trigger has been executed. The table has six columns: COD_COMPETITIE, COD_FEDERATIE, COD_LOCATIE, DATA_COMPETITIE, and VALOARE_TOTALA_PREMII. The results show six rows of data.

	COD_COMPETITIE	COD_FEDERATIE	COD_LOCATIE	DATA_COMPETITIE	VALOARE_TOTALA_PREMII
1	10	10	2	08-OCT-22	20000
2	20	10	5	27-MAY-23	5000
3	30	30	4	18-APR-22	1000
4	40	20	2	20-JUL-23	7000
5	50	50	5	02-DEC-25	40000
6	60	40	3	12-AUG-21	15000

–after

The screenshot shows a SQL Query Builder window with a 'Worksheet' tab. The SQL code defines a trigger named 't_10' that fires after INSERT, DELETE, or UPDATE on the 'sponsorizeaza' table. The trigger declares a cursor 'curs' and a loop to update the 'competitie' table. It also includes an INSERT statement and a SELECT statement.

```

CREATE OR REPLACE TRIGGER t_10
AFTER INSERT OR DELETE OR UPDATE ON sponsorizeaza
DECLARE
    CURSOR curs IS select cod_competitie, sum(suma_oferita) suma_from sponsorizeaza gr
BEGIN
    FOR a IN curs LOOP
        UPDATE competitie
        SET valoare_totala_premii = a.suma
        WHERE cod_competitie = a.cod_competitie;
    END LOOP;
END t_10;
/
BEGIN
    INSERT INTO sponsorizeaza
    VALUES(40, 20, 10000, sysdate);
END;
/
select * from competitie;

```

Below the code, the 'Query Result' tab shows the execution output. It indicates that all rows were fetched in 0.001 seconds. The result is a table with 6 rows and 5 columns: COD_COMPETITIE, COD_FEDERATIE, COD_LOCATIE, DATA_COMPETITIE, and VALOARE_TOTALA_PREMII.

	COD_COMPETITIE	COD_FEDERATIE	COD_LOCATIE	DATA_COMPETITIE	VALOARE_TOTALA_PREMII
1	10	10	2	08-OCT-22	20000
2	20	10	5	27-MAY-23	20000
3	30	30	4	18-APR-22	20000
4	40	20	2	20-JUL-23	20000
5	50	50	5	02-DEC-25	40000
6	60	40	3	12-AUG-21	20000

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

--Creați un trigger la nivel de linie care, la update-ul valorii totale a premiilor a competiției,

--sa alocă jumătate din valoarea premiilor maxime găsite în tabel dacă noua valoare a premiilor

--e mai mic decât acest maxim.

--varianta urmatoare este gresita, deoarece produce eroarea MUTATING TABLE

```
CREATE OR REPLACE TRIGGER trig_11_gresit
```

```
AFTER INSERT OR UPDATE ON competitie
```

```
FOR EACH ROW
```

```
DECLARE
```

```
    p_min competitie.valoare_totala_premii%TYPE;
```

```
BEGIN
```

```
    -- gasesc jumatatea pretului maxim
```

```
    SELECT MAX (valoare_totala_premii) / 2
```

```
    INTO p_min
```

```
    FROM competitie;
```

```

-- verific cu noua valoare
IF p_min > :NEW.valoare_totala_premii THEN
    UPDATE competitie
    SET valoare_totala_premii = p_min
    WHERE cod_competitie = :NEW.cod_competitie;
END IF;
END;
/

UPDATE competitie
SET valoare_totala_premii = 120
WHERE cod_competitie = 20;
/
drop trigger trig_11_gresit;

```

The screenshot shows a SQL IDE interface. The top window is the 'Query Builder' showing a trigger definition. The bottom window is the 'Script Output' showing the execution of the trigger and an error report.

Query Builder:

```

--varianta urmatoare este gresita, deoarece produce eroarea MUTATING TABLE
CREATE OR REPLACE TRIGGER trig_11_gresit
AFTER INSERT OR UPDATE ON competitie
FOR EACH ROW
DECLARE
    p_min competitie.valoare_totala_premii%TYPE;
BEGIN
    -- gasesc jumatatea pretului maxim
    SELECT MAX (valoare_totala_premii) / 2
    INTO p_min
    FROM competitie;

    -- verific cu noua valoare
    IF p_min > :NEW.valoare_totala_premii THEN
        UPDATE competitie
        SET valoare_totala_premii = p_min
        WHERE cod_competitie = :NEW.cod_competitie;
    END IF;
END;
/

UPDATE competitie
SET valoare_totala_premii = 120
WHERE cod_competitie = 20;
/
select * from competitie

```

Script Output:

```

SET valoare_totala_premii = 120
WHERE cod_competitie = 20
Error report -
ORA-04091: table DANI.COMPETITIE is mutating, trigger/function may not see it
ORA-06512: at "DANI.TRIG_11_GRESIT", line 5
ORA-04088: error during execution of trigger 'DANI.TRIG_11_GRESIT'

```

--folosim compound trigger

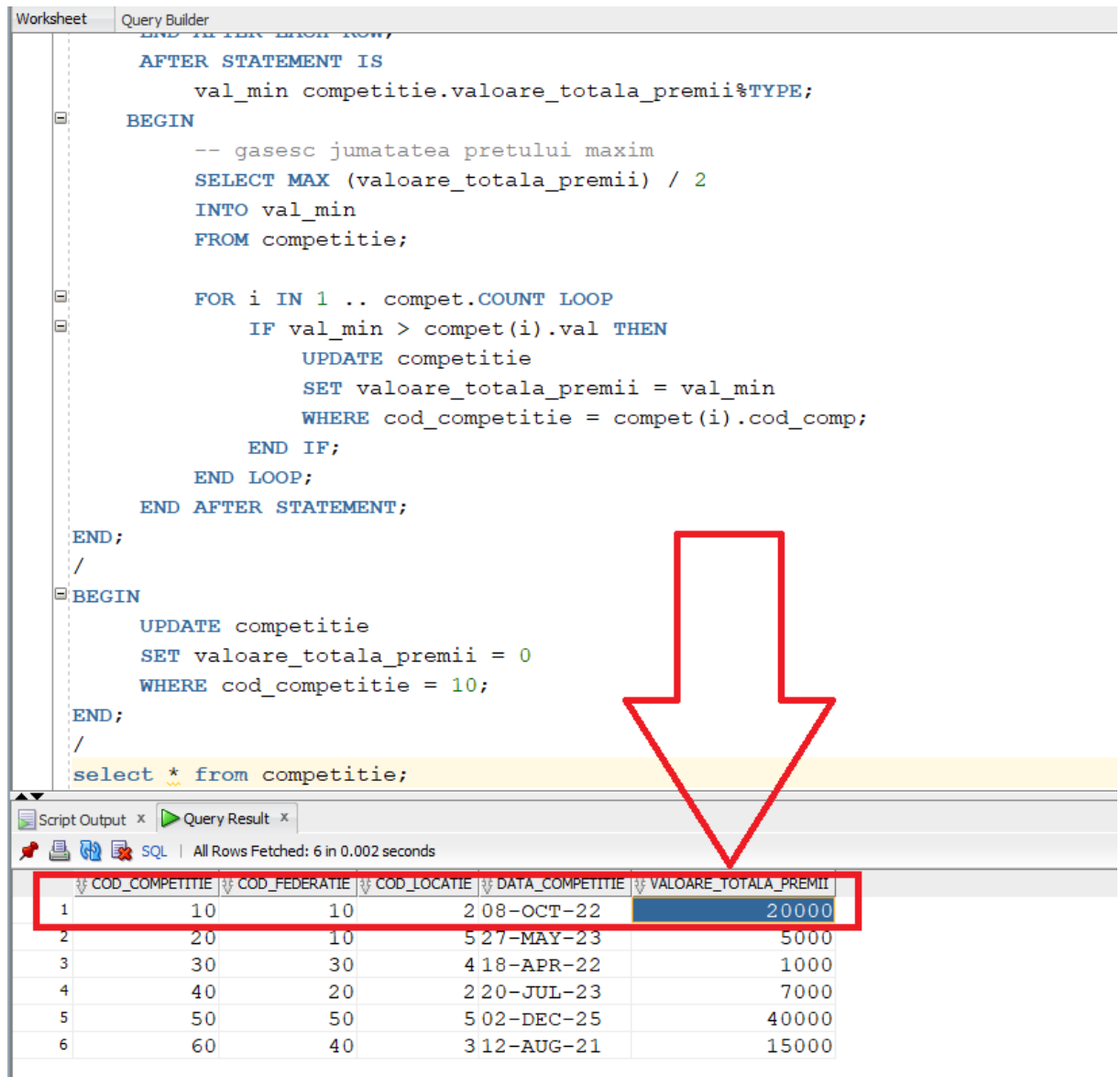
```

/
CREATE OR REPLACE TRIGGER tr_ex11_corect

```

```
FOR UPDATE OR INSERT ON competitie
COMPOUND TRIGGER
  TYPE r_comp IS RECORD (
    cod_comp competitie.cod_comp%TYPE,
    val competitie.valoare_totala_premii%TYPE);
  TYPE t_comp IS TABLE OF r_comp INDEX BY PLS_INTEGER;
  compet t_comp;
  AFTER EACH ROW IS
  BEGIN
    compet(compet.COUNT + 1).cod_comp := :NEW.cod_comp;
    compet(compet.COUNT).val := :NEW.valoare_totala_premii;
  END AFTER EACH ROW;
  AFTER STATEMENT IS
    val_min competitie.valoare_totala_premii%TYPE;
  BEGIN
    -- gasesc jumatatea pretului maxim
    SELECT MAX (valoare_totala_premii) / 2
    INTO val_min
    FROM competitie;

    FOR i IN 1 .. compet.COUNT LOOP
      IF val_min > compet(i).val THEN
        UPDATE competitie
        SET valoare_totala_premii = val_min
        WHERE cod_comp = compet(i).cod_comp;
      END IF;
    END LOOP;
  END AFTER STATEMENT;
END;
/
BEGIN
  UPDATE competitie
  SET valoare_totala_premii = 0
  WHERE cod_comp = 10;
END;
/
select * from competitie;
```



The screenshot shows a SQL Query Builder window with a PL/SQL script in the main editor. The script defines a trigger and a procedure. The procedure, named 'PROCEDURE', calculates the maximum prize value from the 'competitie' table and updates the 'valoare_totala_premii' column for the competition with the lowest value. The script is executed, and the results are shown in the 'Query Result' tab. The results table has 6 rows, with the first row highlighted in blue and a red box around it. A large red arrow points from the script to the result table.

```

END AFTER EACH ROW;

AFTER STATEMENT IS
    val_min competitie.valoare_totala_premii%TYPE;
BEGIN
    -- gasesc jumatatea pretului maxim
    SELECT MAX (valoare_totala_premii) / 2
    INTO val_min
    FROM competitie;

    FOR i IN 1 .. compet.COUNT LOOP
        IF val_min > compet(i).val THEN
            UPDATE competitie
            SET valoare_totala_premii = val_min
            WHERE cod_competitie = compet(i).cod_comp;
        END IF;
    END LOOP;
END AFTER STATEMENT;

END;
/

BEGIN
    UPDATE competitie
    SET valoare_totala_premii = 0
    WHERE cod_competitie = 10;
END;
/

select * from competitie;

```

COD_COMPETITIE	COD_FEDERATIE	COD_LOCATIE	DATA_COMPETITIE	VALOARE_TOTALA_PREMII
1	10	10	2 08-OCT-22	20000
2	20	10	5 27-MAY-23	5000
3	30	30	4 18-APR-22	1000
4	40	20	2 20-JUL-23	7000
5	50	50	5 02-DEC-25	40000
6	60	40	3 12-AUG-21	15000

O. Definiți un trigger de tip LDD. Declanșați trigger-ul.

```

create table logs
(
    utilizator VARCHAR2(50),
    nume_database VARCHAR2(70),
    eveniment VARCHAR2(20),
    nume_tabel VARCHAR2(40),
    data_log date
);
/
commit;
/
CREATE OR REPLACE TRIGGER tr_log
AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
    INSERT INTO logs

```

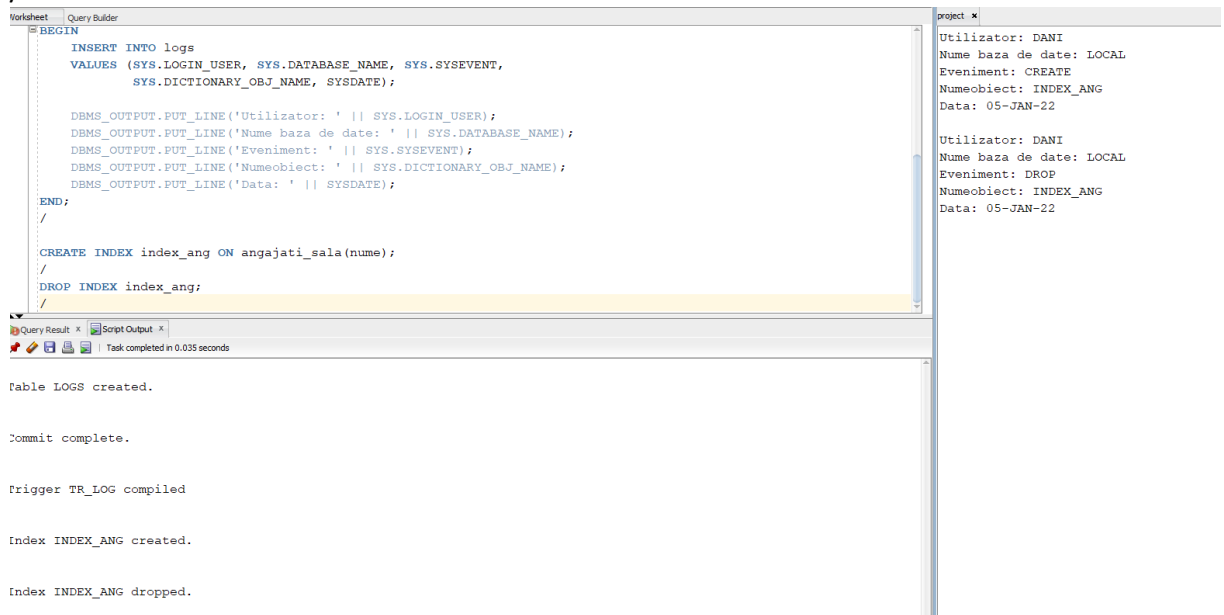
```

VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT,
        SYS.DICTIONARY_OBJ_NAME, SYSDATE);

DBMS_OUTPUT.PUT_LINE('Utilizator: ' || SYS.LOGIN_USER);
DBMS_OUTPUT.PUT_LINE('Nume baza de date: ' || SYS.DATABASE_NAME);
DBMS_OUTPUT.PUT_LINE('Eveniment: ' || SYS.SYSEVENT);
DBMS_OUTPUT.PUT_LINE('Numeobiect: ' || SYS.DICTIONARY_OBJ_NAME);
DBMS_OUTPUT.PUT_LINE('Data: ' || SYSDATE);
END;
/

CREATE INDEX index_ang ON angajati_sala(num);
/
DROP INDEX index_ang;
/

```



```

CREATE OR REPLACE TRIGGER trig_extra
  BEFORE INSERT OR DELETE OR UPDATE on participa
BEGIN
  IF (TO_CHAR(SYSDATE,'D') = 1) OR (TO_CHAR(SYSDATE,'HH24') NOT BETWEEN 8
  AND 16) THEN
    RAISE_APPLICATION_ERROR(-20001, 'Sportivii se pot inscrie la competitii doar in
    intervalul orar de lucru!');
  END IF;
END;
/
Insert into participa values (3000, 10, sysdate);
/
drop trigger trig_extra;

```

The screenshot shows the Oracle SQL Developer interface. The top pane displays a SQL script for creating a trigger named 'trig_extra' on the 'participa' table. The trigger is a BEFORE INSERT OR DELETE OR UPDATE trigger. The script includes a BEGIN block with an IF statement that checks if the day of the month is 1 or if the time is not between 8 and 16. If either condition is true, it raises an application error with message 'Sportivii se pot inscrie la competitii doar in intervalul orar de lucru!'. The script ends with an END block, followed by an INSERT statement and a ROLLBACK statement.

The bottom pane shows the 'Script Output' tab with the following error message:

```

Error starting at line : 9 in command -
Insert into participa values (3000, 10, sysdate)
Error report -
ORA-20001: Sportivii se pot inscrie la competitii doar in intervalul orar de lucru!
ORA-06512: at "DANI.TRIG_EXTRA", line 3
ORA-04088: error during execution of trigger 'DANI.TRIG_EXTRA'

```

P. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului

CREATE OR REPLACE PACKAGE pack13 AS

 PROCEDURE prog_sponsorizari(cod_loc competitie.cod_locatie%type);

 PROCEDURE prog_sportivi_fed(cod club.cod_locatie%type);

 FUNCTION fct_cluburi_an_sponsor(an number, nume_spons
sponsori.denumire_sponsor%type) RETURN club%rowtype;

 PROCEDURE p_9_sport(nume_spons sponsori.denumire_sponsor%type);

END pack13;

/

CREATE OR REPLACE PACKAGE BODY pack13 AS

 --ex6

 --pentru o locatie primita ca parametru, sa se afiseze suma totala obtinuta din
sponsorizari

 --impreuna cu numele sponsorilor si data la care au contribuit la aceasta
sponsorizare

 --pentru primele 3 competitii dupa data competitiei

 PROCEDURE prog_sponsorizari(cod_loc competitie.cod_locatie%type)

 IS

 TYPE rec_num_data IS RECORD

 (nume sponsori.denumire_sponsor%type,

 data_sponsorizarii sponsorizeaza.data_sponsorizarii%type);

 TYPE tablou_sponsorizare IS TABLE OF rec_num_data;

 TYPE vector_coduri_comp IS VARRAY(3) OF competitie.cod_competitie%type;

 tab_sponsori tablou_sponsorizare;

 vec_comp vector_coduri_comp;


```

        suma_totala NUMBER := 0;
BEGIN
    SELECT cod_competitie
    BULK COLLECT INTO vec_comp
    FROM competitie where cod_locatie = cod_loc AND rownum <= 3
    ORDER BY data_competitie;

    FOR i IN vec_comp.FIRST..vec_comp.LAST LOOP
        IF vec_comp.exists(i) THEN
            SELECT sum(suma_oferita)
            INTO suma_totala
            FROM SPONSORIZEAZA
            WHERE COD_COMPETITIE = vec_comp(i);

            DBMS_OUTPUT.PUT_LINE('DETALII DESPRE COMPETITIA ' || vec_comp(i) ||
':');

            DBMS_OUTPUT.PUT_LINE('Suma totala : ' || suma_totala);

            SELECT denumire_sponsor, data_sponsorizarii
            BULK COLLECT INTO tab_sponsori
            FROM SPONSORI S JOIN SPONSORIZEAZA SP ON (SP.COD_SPONSOR =
S.COD_SPONSOR)
            WHERE SP.COD_COMPETITIE = vec_comp(i);

            FOR i IN tab_sponsori.FIRST..tab_sponsori.LAST LOOP
                IF tab_sponsori.exists(i) THEN
                    DBMS_OUTPUT.PUT_LINE('Sponsorul ' || i || ' este ' ||
tab_sponsori(i).nume || ' si a donat in data de - ' ||
tab_sponsori(i).data_sponsorizarii);
                END IF;
            END LOOP;
            tab_sponsori.delete();
        END IF;
    END LOOP;

END PROG_SPONSORIZARI;

-- ex 7
--sa se scrie codul si numele federatiilor, codul si numele
--sportivilor care participa la competitii la un cod de locatie dat
PROCEDURE prog_sportivi_fed(cod_club.cod_locatie%type)
IS
    cod_f federatie_de_sport.cod_federatie%type;
    den_federatie_de_sport.denumire%type;
    c_sportivi SYS_REFCURSOR;

```

```

TYPE sportiv_record IS RECORD
    ( cod_s sportiv.cod_sportiv%type,
      nume_s sportiv.nume%type,
      prenume_s sportiv.prenume%type
    );
sportiv_detalii sportiv_record;
CURSOR c_fest IS SELECT f.cod_federatie, f.denumire,
                      CURSOR (
                        SELECT s.cod_sportiv, s.nume, s.prenume
                        FROM sportiv s JOIN apartine a ON (a.cod_sportiv =
s.cod_sportiv)
                        WHERE a.cod_federatie = f.cod_federatie)
                      FROM FEDERATIE_DE_SPORT f JOIN COMPETITIE c ON
(f.cod_federatie = c.cod_federatie)
                      WHERE c.cod_locatie = cod;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Codul locatiei este : ' || cod);

    OPEN c_fest;

    LOOP
        FETCH c_fest INTO cod_f, den, c_sportivi;
        EXIT WHEN c_fest%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Cod federatie : ' || cod_f || '. Nume
federatie : ' || den || ' are sportivii :');

        LOOP
            FETCH c_sportivi INTO sportiv_detalii;
            EXIT WHEN c_sportivi%NOTFOUND;

            DBMS_OUTPUT.PUT_LINE('Cod sportiv (' || sportiv_detalii.cod_s || ') pe care
il cheama ' || sportiv_detalii.nume_s || ' ' || sportiv_detalii.prenume_s || '.');

        END LOOP;
    END LOOP;

    CLOSE c_fest;

END prog_sportivi_fed;

```

```

--ex8
--Sa se afiseze primul club (luat in ordine crescatoare dupa cod) la care sportivii au
varsta < 30 ani si care
--au participat la competitia din anul dat ca parametru sponsorizata de anumit
sponsor
FUNCTION fct_cluburi_an_sponsor( an number,

```

```

                nume_spons sponsori.denumire_sponsor%type)
            RETURN club%rowtype
        IS
            informatii club%rowtype;

        BEGIN
            SELECT c.cod_club, c.cod_locatie, c.denumire, c.data_infiintare_club,
            c.num_e_admin, c.prenume_admin
            INTO informatii
            FROM club c JOIN sportiv s ON (c.cod_club=s.cod_club)
            JOIN participa p ON (s.cod_sportiv=p.cod_sportiv) JOIN competitie co ON
            (co.cod_competitie=p.cod_competitie)
            JOIN sponsorizeaza a ON (co.cod_competitie=a.cod_competitie)
            JOIN sponsori b ON (a.cod_sponsor=b.cod_sponsor)
            WHERE months_between(sysdate,s.data_nastere) < 30*12 AND
            to_number(to_char(co.data_competitie,'yyyy'))= an AND
            lower(b.denumire_sponsor) = nume_spons AND rownum <= 1
            ORDER BY c.cod_club;

            RETURN informatii;

        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                RAISE_APPLICATION_ERROR(-20000, 'Nu s-au gasit cluburi care sa
            indeplineasca cerintele');
            WHEN TOO_MANY_ROWS THEN
                RAISE_APPLICATION_ERROR(-20001, 'Mai multe cluburi indeplinesc
            cerintele!');
            WHEN OTHERS THEN
                RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!');

        END fct_cluburi_an_sponsor;

--ex9
--Afisati sportivul care a fost la cele mai multe competitii de sport sponsorizate de
o anumita firma,
--si nr de competitii. (aici a trebuit sa mai adaug un sportiv, insertul il gasiti pe
ultimul rand)
        PROCEDURE p_9_sport(nume_spons sponsori.denumire_sponsor%type)
        IS
            CURSOR c IS select s.cod_sportiv cod, initcap(s.num_e), initcap(s.prenume),
            count(*) numar
                from sportiv s JOIN participa p on (s.cod_sportiv = p.cod_sportiv)
                JOIN competitie c on (p.cod_competitie = c.cod_competitie)
                JOIN sponsorizeaza s1 on (s1.cod_competitie = c.cod_competitie)
                JOIN sponsori s2 on (s1.cod_sponsor = s2.cod_sponsor)

```

```
        where initcap(ume_spons) = initcap(denumire_sponsor)
        group by s.cod_sportiv, initcap(s.ume), initcap(s.prenume);

TYPE rec_inreg IS RECORD
    (cod_s sportiv.cod_sportiv%type,
     ume sportiv.ume%type,
     prenume sportiv.prenume%type,
     nr NUMBER);
inreg rec_inreg;
maxim NUMBER := 0;
frecv NUMBER :=0;
BEGIN
    FOR inr IN c LOOP
        IF inr.numar > maxim THEN
            maxim := inr.numar;
            inreg := inr;
            frecv := 1;
        ELSIF inr.numar = maxim THEN
            frecv := frecv + 1;
        END IF;
    END LOOP;

    IF maxim = 0 THEN
        RAISE NO_DATA_FOUND;
    END IF;

    IF frecv > 1 THEN
        RAISE TOO_MANY_ROWS;
    END IF;

    DBMS_OUTPUT.PUT_LINE('Persoana ' || inreg.ume || ' ' || inreg.prenume || ' a
fost la ' || inreg.nr || ' competitii sportive, reprezentand maximul de competitii la care
a fost o pers dintre competiile sponsorizate de ' || ume_spons);

EXCEPTION
    WHEN TOO_MANY_ROWS THEN
        RAISE_APPLICATION_ERROR(-20001, 'Mai multi sportivi indeplinesc
cerinta.');
```

cerinta.');

```
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20000, 'Nu este indeplinita cerinta');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Alta eroare!!!!');
    END p_9_sport;
END pack13;
/
```

```

CREATE OR REPLACE PACKAGE pack13 AS
    PROCEDURE prog_sponsorizari( cod_loc competitie.cod_locatie%type );
    PROCEDURE prog_sportivi_fed(cod club.cod_locatie%type);
    FUNCTION fct_cluburi_an_sponsor( an number, nume_spons sponsori.denumire_sponsor%type) RETURN club%rowtype;
    PROCEDURE p_9_sport(nume_spons sponsori.denumire_sponsor%type);
END pack13;
/

CREATE OR REPLACE PACKAGE BODY pack13 AS

    --ex6
    --pentru o locatie primita ca parametru, sa se afiseze suma totala obtinuta din sponsorizari
    --impreuna cu numele sponsorilor si data la care au contribuit la aceasta sponsorizare
    --pentru primele 3 competitii dupa data competitiei

    PROCEDURE prog_sponsorizari( cod_loc competitie.cod_locatie%type )
    IS
        TYPE rec_nume_data IS RECORD
            (nume sponsori.denumire_sponsor%type,
             data_sponsorizarii sponsorizeaza.data_sponsorizarii%type);
        TYPE tablou_sponsorizare IS TABLE OF rec_nume_data;
        TYPE vector_coduri_comp IS VARRAY(3) OF competitie.cod_competitie%type;
        tab_sponsori tablou_sponsorizare;
        vec_comp vector_coduri_comp;
        suma_totala NUMBER := 0;
    BEGIN
        SELECT cod_competitie

```

Script Output x Query Result x

Task completed in 0.109 seconds

Package PACK13 compiled

Package Body PACK13 compiled

—deoarece am testat fiecare functie in parte la fiecare exercitiu, am considerat ca este de ajuns doar sa le pun in pachet, fara a le mai testa o data.

Q. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

```

CREATE OR REPLACE PACKAGE pack14 AS
    TYPE rec IS RECORD
        (cod_a antrenori.cod_antrenor%type,
         nume antrenori.nume%type,
         prenume antrenori.prenume%type,
         data_n antrenori.data_nastere%type,
         nrtel antrenori.nrtelefon%type);
    TYPE tab_rec IS VARRAY(10) OF rec;
    vect tab_rec;

    PROCEDURE p_14_1(fed federatie_de_sport.denumire%type);

    FUNCTION p_14_2 RETURN federatie_de_sport.denumire%type;

    PROCEDURE p_14_3;

    FUNCTION p_14_4(an NUMBER) RETURN tab_rec;

END pack14;
/

```

```
CREATE OR REPLACE PACKAGE BODY pack14 AS
```

```
--Sa se afiseze angajatii care au salariu > media salariilor angajatilor
--ce lucreaza intr-o sala din cadrul federatiei de fitness sau unei federatii primita ca parametru
```

```
PROCEDURE p_14_1(fed federatie_de_sport.denumire%type)
IS
    CURSOR c IS SELECT cod_angajat, nume, prenume, data_nastere, sex, salariu
                FROM angajati_sala e
                WHERE e.salariu > ( SELECT avg(salariu)
                                FROM angajati_sala a JOIN apartine c ON
(a.cod_sala=c.cod_sala)
                                JOIN federatie_de_sport d ON
(c.cod_federatie=d.cod_federatie)
                                WHERE lower(d.denumire) = lower(fed) or
lower(d.denumire) = 'federatiadefitness');
    BEGIN
        FOR x IN c LOOP
            dbms_output.put('Angajatul ' || x.nume || ' ' || x.prenume || ', care este nascut
pe data de ' || x.data_nastere || ' are salariul ' || x.salariu || ' si este ');
            IF x.sex = 'm' THEN DBMS_OUTPUT.PUT_LINE('barbat.');
```

```
ELSE DBMS_OUTPUT.PUT_LINE('femeie.');
```

```
END IF;
        END LOOP;
    END p_14_1;

-- sa se selecteze federatia cu cei mai multi sportivi, daca sunt mai multe, sa se ia
prima ordonata dupa cod
FUNCTION p_14_2 RETURN federatie_de_sport.denumire%type IS
    nume_f federatie_de_sport.denumire%type := 'dani';
    TYPE rec_f IS RECORD
        (cod_f federatie_De_sport.cod_federatie%type,
        nr number);
    vect_rec rec_f;
    begin
        with abc as (select cod_federatie, count(*) nr_sportivi from apartine group by
cod_federatie)
        select * into vect_rec
        from abc
        where nr_sportivi = (select max(nr_sportivi) from abc) and rownum <= 1
        order by cod_federatie;

        select denumire into nume_f
        from federatie_de_sport
        where cod_federatie = vect_rec.cod_f;
```

```

    return nume_f;
end p_14_2;

```

---sa se scrie o procedura ce afiseaza sportivii impreuna cu nr de zile si numele clubului la

--care au stat in trecut cele mai multe zile si sa se specifice daca au
 --stat sub 1000 zile, sub 5000 zile sau peste 5000zile.

```

PROCEDURE p_14_3 IS
  TYPE rec_14_3 IS RECORD
    (cod_s sportiv.cod_sportiv%type,
     nume sportiv.nume%type,
     prenume sportiv.prenume%type,
     nrtel sportiv.nrtelefon%type,
     den_club club.denumire%type,
     nr_zile number);
  TYPE tablou_sport IS TABLE OF rec_14_3;
  tab_sportivi tablou_sport;
BEGIN
  WITH istoric AS ( SELECT cod_sportiv,cod_club,sum(data_final-data_start) as
"NUMAR_ZILE"
                    FROM club_history
                    GROUP BY cod_sportiv,cod_club),
  istoric_cu_maxim AS ( SELECT * FROM istoric i
                        WHERE i.numar_zile = ( select max(numar_zile) from istoric
                        where i.cod_sportiv=cod_sportiv))
  select s.cod_sportiv ,s.nume, s.prenume, s.nrtelefon, c.denumire, numar_zile
  BULK COLLECT INTO tab_sportivi
  FROM sportiv s join istoric_cu_maxim ist on (s.cod_sportiv=ist.cod_sportiv)
  join club c on (ist.cod_club=c.cod_club)
  ORDER BY numar_zile;

  FOR i IN tab_sportivi.FIRST..tab_sportivi.LAST LOOP
    IF tab_sportivi.exists(i) THEN
      DBMS_OUTPUT.PUT('Sportivul ' || tab_sportivi(i).nume || '
' || tab_sportivi(i).prenume || ' (cod ' || tab_sportivi(i).cod_s || ') il poti suna la
' || tab_sportivi(i).nrtel || ' si a facut parte din clubul ' || tab_sportivi(i).den_club || '
pentru o perioada aproximativa de ');
      CASE
        WHEN tab_sportivi(i).nr_zile < 1000 THEN DBMS_OUTPUT.PUT_LINE(' SUB
1000 ZILE');
        WHEN tab_sportivi(i).nr_zile < 5000 THEN DBMS_OUTPUT.PUT_LINE('
INTRE 1000 SI 5000');
        ELSE DBMS_OUTPUT.PUT_LINE('PESTE 5000 ZILE');
      END CASE;
    END IF;
  END LOOP;

```

```

END p_14_3;

--Sa se afiseze primii 10 antrenori ai cluburilor care se afla in bucuresti sau
--judetul neamt si au sportivi nascuti inainte de un an trimis ca parametru.Daca
numarul
--de telefon este null, atunci nu se va afisa.
FUNCTION p_14_4(an NUMBER) RETURN tab_rec
IS
    vect_ret tab_rec;
    cursor a is (SELECT DISTINCT
cod_antrenor,a.ume,a.prenume,a.data_nastere,a.nrtelefon
FROM antrenori a JOIN club b ON (a.cod_club=b.cod_club)
JOIN locatie c ON(b.cod_locatie=c.cod_locatie)
JOIN sportiv d ON(b.cod_club=d.cod_club)
WHERE (lower(judet)='bucuresti' OR lower(judet)='neamt') AND
to_number(to_char(d.data_nastere,'yyyy')) <= an AND rownum <= 10);
BEGIN
    open a;
    fetch a bulk collect into vect_ret;
    close a;
    return vect_ret;
END p_14_4;
END pack14;
/
begin
    dbms_output.put_line('-----');
    declare
        an number := '&an_limita_pentru_procedura_14_4';
    begin
        dbms_output.put_line('incepe executia p_14_4');
        --testare p_14_4
        pack14.vect := pack14.p_14_4(an);
        FOR i IN pack14.vect.FIRST..pack14.vect.LAST LOOP
            IF pack14.vect.exists(i) THEN
                DBMS_OUTPUT.PUT('codul ' || pack14.vect(i).cod_a || ' apartine
antrenorului ' || pack14.vect(i).ume || ' ' || pack14.vect(i).prenume || ' si este nascut
in data de ' || pack14.vect(i).data_n);
                IF pack14.vect(i).nrtele IS NOT NULL THEN DBMS_OUTPUT.PUT_LINE(' si il
puteti contacta la numarul ' || pack14.vect(i).nrtele);
                ELSE dbms_output.put_line('.');
                END IF;
            END IF;
        END LOOP;
    end;
end;

```