Nafornita Adrian-Valentin
Gr 243

Gestiunea unui campionat

Modelul real

Baza de date cuprinde informatii referitoare la desfasurarea unor campionate tipice de ciclism. In acest scop se vor stoca date despre campionate, despre stagiile pe care le vor tine, disciplinele la care ciclistii vor putea concura, despre sponsori si legaturile lor de finantare. Ciclistii vor putea participa la o superpromotie pe fiecare traseu in urma carora invingatorilor li se acorda puncta bonus.

Scopul bazei se rezuma la a putea organiza in decursul unei zile distributia ciclistilor la evenimentele stabilite de catre organizator.

DB-ul respecta o serie de restrictii firesti. Asadar, o grupa va fi alcatuita din oricat de multi ciclisti. Un ciclist se poate afla in oricate grupe, asa cum orice grupa poate participa la oricate discipline din cadrul unui campionat. Disciplinele pe de alta parte vor fi sustinute singular de catre fiecare competitie in parte, ceea ce inseamna ca odata ce o disciplina a fost organizata de un campionat, ea nu va mai putea exista si in urmatorul.

Totodata, sponsorii lucreaza atat cu campionatele, oferind finantare, cat si cu ciclisti in scopuri comerciale. Ei pot fi sponzoritati de oricat de multe companii se ofera.

In continuare, un ciclist care castiga o superpromotie, cu alte cuvinte o cursa specifica fiecarui traseu si care se parcurge o singura data, va primi punctaj bonus.

Impreuna cu medicii, grupa concurenta se asociaza unui traseu si se va forma orarul competitiv.

Reguli de functionare

Modelul de date prezinta urmatoarele restrictii de functionare:

1. O disciplina nu poate fi sustinuta decat de un campionat
2. O grupa are voie sa participe la oricate discipline doreste, precum ciclistii pot fi alocati in infinit de multe grupe
3. Un sponsor poate finanta oricat de multe campionate sau ciclisti simultan.
4. Un ciclist poate concura in mai multe superpromotii, dar numai o singura data.
5. O grupa are un singur medic responsabil per cursa.

Entitati

Urmatoarele structuri in modelul de date se numesc entitati: CAMPIONAT, DISCIPLINA, GRUPA, CICLIST, SPONSOR, MEDIC, TRASEU.

In cele ce urmeaza, voi descrie complet entitatile, precizant cheia primara. Entitati dependente sunt ALOCAT, FINANTARE, SPONSORIZARE, ORAR.

CAMPIONAT = eveniment unde pasionatii de ciclism se intrec pentru a demonstra performanta individuala. Cheia primara este id_camp.

DISCIPLINA = categorie de ciclism, unde participa persoane special antrenate si cu echipament specific. Cheia primara este id_disciplina.

GRUPA = asociatie de persoane, cu scop comun de regula stabilita in functie de performantele indivizilor ce o alcatuiesc. Cheia primara este id_grupa.

CICLIST = individ pasionat si antrenat pentru competitiile sportive de ciclism, care doreste sa arate performanta sa intr-un eveniment organizat. Cheia primara este id_ciclist.

SPONSOR = companie cu posibilitati financiare care doreste sa isi promoveze imaginea in urma dotarii participantilor sau a unui campionat cu materiale sau bani. Cheia primara este id_sponsor.

MEDIC = o persoana educata sa ajute persoanele vatamate in urma ciclismului, sport deosebit de periculos cand este dus la extrem. Cheia primara este id_medic.

TRASEU = circuit, locatie pusa la dispozitie de catre autoritati pentru sportivi sa concureze. Cheia primara este id_traseu.

ALOCAT = modalitatea unui ciclist de a fi repartizat intr-o grupa. Cheia primara compusa este formata din id_ciclist si id_grupa.

FINANTARE = tine evidenta companiilor care ajuta direct campionatul. Cheia primara este compusa dn id_camp si id_sponsor.

SPONSORIZARE = mijlocul prin care o companie sustine unul sau mai multi concurent cu particularitati comune. Cheia primara compusa este formata din id_ciclist si id_sponsor.

ORAR = programul unui campionat, sustinand repartitia traseelor, a medicilor si a grupelor pentru a desfasura o singura cursa, de fiecare data. Cheia primara compusa are id_grupa, id_medic, id_traseu.

Relatii

DISCIPLINA_sustinuta_de_GRUPA = relatie tip many-to-many care leaga entitatile GRUPA si DISCIPLINA, denuntand legatura dintre acestea. Cardinalitatea minima este 1:0 (o disciplina trebuie sa fie sustinuta de o grupa) si cardinalitatea maxima m:n (o disciplina poate fi sustinuta de indiferent de multe grupe).

GRUPA_formata_din_CICLIST = relatie de tip one to many; cardinalitatea minima este 1:1 (grupa trebuie sa aiba minim un ciclist) si cardinalitatea maxima m:n (oricate grupe pot avea oricati ciclisti).

SPONSOR_promoveaza_CICLIST = many-to-many; cardinalitatea minima 1:0 (nu este necesar ca un ciclist sa fie sponsorizat), iar cardinalitatea maxima m:n (un ciclist poate avea oricat de multi spnosori).

SPONSOR_finanteaza_CAMPIONAT = many-to-many; cardinalitatea minima este 1:0 si cardinalitatea maxima m:n.

CICLIST_superpromo_TRASEU = relatie one-to-many, care leaga CICLIST de TRASEU pentru a tine evidenta superpromotiilor. Cardinalitatea minima este 1:0 si cardinalitatea maxima este 1:m (un ciclist poate participa pe fiecare traseu la eveniment).

GRUPA_ajutat_MEDIC_la_loc_TRASEU = relatie tip 3 ce leaga grupa de medic si traseu. Denumirea este ORAR.

Atribute

Campionat

1. id_camp = integer, codul unui campionat PK
2. nume = sir caractere de lungime mai mica decat 30, numele unui campionat NOT NULL
3. an = integer, NOT NULL
4. mail = sir caractere de lungime mai mica decat 30

Disciplina

1. id_disciplina = integer PK, codul disciplinei in sistem
2. in_champ = integer FK, codul trebuie sa corespunda unei chei primare din campionat
3. nume = sir caractere de lungime mai mica decat 20, NOT NULL

Grupa

1. id_grupa = integer PK, codul unei grupe
2. in_discipline = integer FK, codul trebuie sa corespunda unei chei primare din disciplina.

Alocat

1. ciclist_ = integer PK, FK, codul trebuie sa corespunda unei chei primare din ciclist.
2. grupa_ = integer PK, FK, codul trebuie sa corespunda unei chei primare din grupa

Ciclist

1. id_ciclist = integer PK

2. nume = sir caractere < 15 NOT NULL
3. prenume = sir caractere < 15 NOT NULL
4. sex = integer, DEFAULT NULL
5. data_nastere = variabila tip data calendaristica, NOT NULL
6. telefon = sir caractere < 16, NOT NULL
7. punctaj = integer, DEFAULT 0

Sponsor

1. id_sponsor = integer PK,
2. nume = sir caractere < 25 NOT NULL
3. mainly_distributes = sir caractere < 15, NOT NULL, ofera informatie legat de principalul produs pe care firma il comercializeaza
4. website = sir caractere < 40 NOT NULL

Finantare

1. sponsor = integer, PK, FK codul trebuie sa corespunda cheii primare din sponsor
2. campionat = integer, PK, FK codul trebuie sa corespunda cheii primare din campionat

Promovari

1. ciclist = integer, PK, FK, codul trebuie sa corespunda cheii primare din ciclist
2. sponsor = integer PK, FK codul trebuie sa corespunda cheii primare din sponsor
3. data_contract = data calendaristica, NOT NULL

Medici

1. id_medic = integer, cheie primara
2. nume = sir de caractere <50, numele complet not null
3. varsta = integer, NOT NULL
4. exp_anterioara = BIT, default 0, memoreaza daca medicul a mai activat in campionate de ciclism sau este prima oara

Traseu

1. id_traseu = integer, PK
2. lungime = float, NOT NULL
3. tip = CHAR, clasificarea traseului dupa standardele internationale

Superpromotie

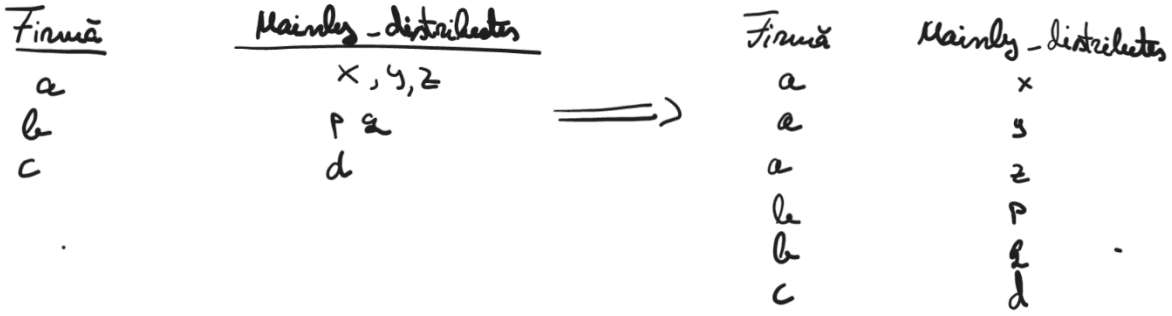
1. ciclist = integer, PK, FK, codul trebuie sa corespunda cheii primare din ciclist
2. traseu = integer, PK, FK, codul trebuie sa corespunda cheii primare din traseu
3. timp = integer, NOT NULL timpul pe care ciclistul l-a scos
4. castiga = BIT, DEFAULT 0, memoreaza daca superpromotia a fost castigata

Orar

1. grupa, PK, FK codul trebuie sa corespunda cheii primare din grupa
2. traseu, PK, FK codul trebuie sa corespunda cheii primare din traseu
3. medic, PK, FK codul trebuie sa corespunda cheii primare din medic

Normalizare

First normal form (1NF) is a property of a relation in a relational database. A relation is in first normal form if and only if no attribute domain has relations as elements.

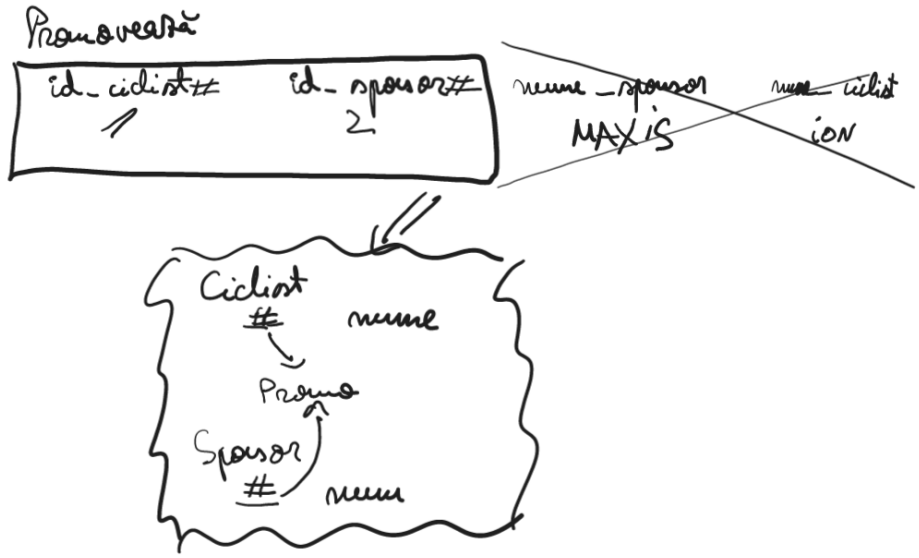


A relation is in the 2NF form if it fulfills the following two requirements:

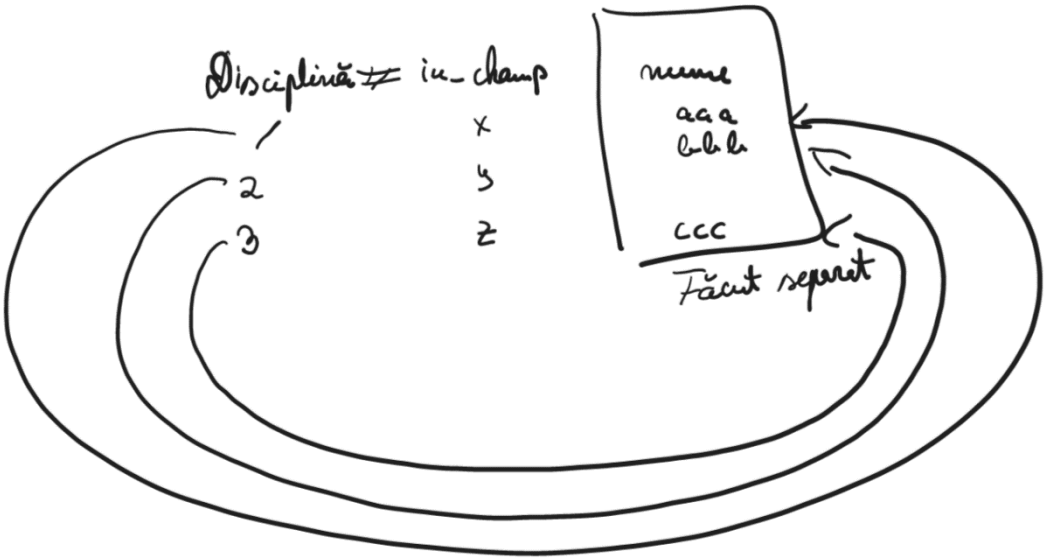
#It is in first normal form.

#It does not have any non-prime attribute that is functionally dependent on any proper subset of any candidate key of the relation. A non-prime attribute of a relation is an attribute that is

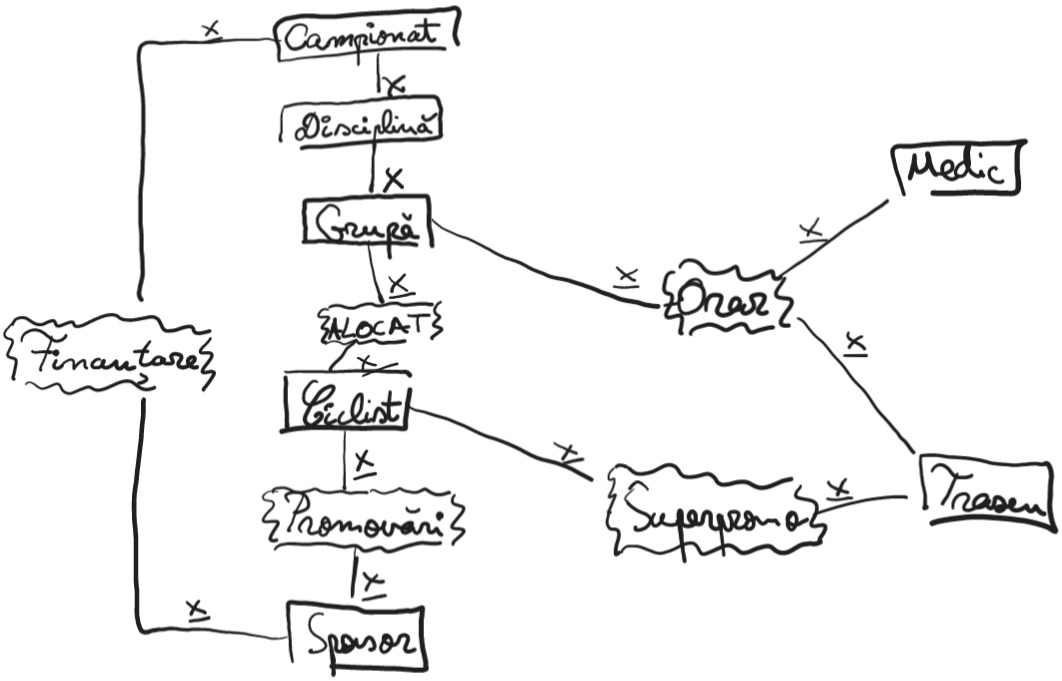
not a part of any candidate key of the relation.

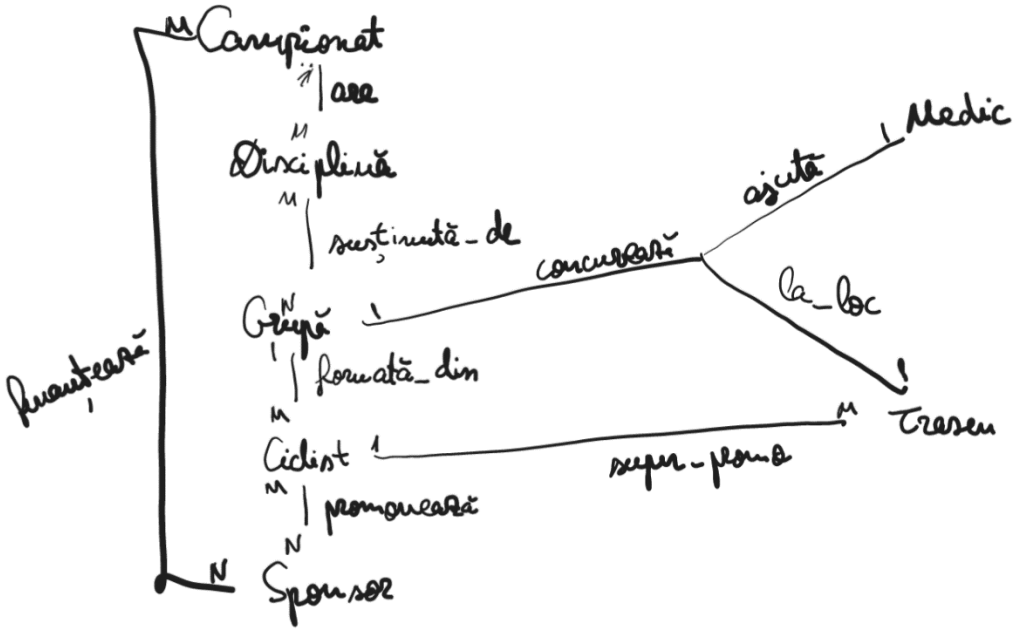


3NF: Codd defined this as a relation in second normal form where all non-prime attributes depend only on the candidate keys and do not have a transitive dependency on another key



Conceptual & ER/D





Exercitiile 5-14. Rezultatele fiecarei proceduri/functii pot fi consultate la finalul documentului.

```
CREATE TABLE Campionat
(
   id_camp INT PRIMARY KEY,
   nume    VARCHAR(30) NOT NULL,
   an      INT         NOT NULL,
   mail    VARCHAR(30)
);
SELECT *
FROM Campionat;
INSERT INTO Campionat
VALUES (1, 'C.N. Downhill', 2022, 'cheilegradistei@office.com');
INSERT INTO Campionat
VALUES (2, 'Downhill Word Championship', 2022,
'worldcycling@managment.com');
INSERT INTO Campionat
VALUES (3, 'Redbull Rampage', 2023, 'redbull@tv.com');
INSERT INTO Campionat
```

```
VALUES (4, 'Tour de France', 2023, 'francecycliste@gmail.com');
INSERT INTO Campionat
VALUES (5, 'Whistler Closing Week Showdown', 2022, 'whistler@office.com');
INSERT INTO Campionat
VALUES (6, 'BikeXpert Challenge', 2022, 'bikexpert@team.ro');

CREATE TABLE Disciplina
(
   id_disciplina INT PRIMARY KEY,
   in_champ      INT,
   nume          VARCHAR(20) NOT NULL,
   CONSTRAINT fk_Disciplina_Campionat FOREIGN KEY (in_champ) REFERENCES
CAMPIONAT (id_camp)
);
SELECT *
FROM DISCIPLINA;

-- Lucrez pe campionatul 2
INSERT INTO Disciplina
VALUES (132, 1, 'Downhill');
INSERT INTO Disciplina
VALUES (133, 4, 'Road');
INSERT INTO Disciplina
VALUES (134, 2, 'Gravity Downhill');
INSERT INTO Disciplina
VALUES (135, 2, 'Pump Track');
INSERT INTO Disciplina
VALUES (136, 2, 'FreeStyle');



CREATE TABLE Grupa
(
   id_grupa      INT PRIMARY KEY,
   in_discipline INT,
   CONSTRAINT fk_Grupa_Disciplina
       FOREIGN KEY (in_discipline) REFERENCES Disciplina (id_disciplina)
);

SELECT *
FROM GRUPA;
INSERT INTO Grupa
VALUES (11, 134);
```

```sql
INSERT INTO Grupa
VALUES (12, 134);
INSERT INTO Grupa
VALUES (13, 134);
INSERT INTO Grupa
VALUES (21, 135);
INSERT INTO Grupa
VALUES (31, 136);
INSERT INTO Grupa
VALUES (49, 132);
INSERT INTO Grupa
VALUES (50, 132);


DROP TABLE Alocat;
CREATE TABLE Alocat
(
   ciclist_ INT,
   grupa_   INT,
   CONSTRAINT pk_Alocat PRIMARY KEY (ciclist_, grupa_),
   CONSTRAINT fk_Alocat_Ciclist
       FOREIGN KEY (ciclist_) REFERENCES Ciclist (id_ciclist),
   CONSTRAINT fk_Alocat_Grupa
       FOREIGN KEY (grupa_) REFERENCES Grupa (id_grupa)
);

SELECT *
FROM ALOCAT;
INSERT INTO Alocat
VALUES (480, 11);
INSERT INTO Alocat
VALUES (500, 11);
INSERT INTO Alocat
VALUES (470, 11);
INSERT INTO Alocat
VALUES (510, 11);
INSERT INTO Alocat
VALUES (570, 12);
-- INSERT INTO Alocat VALUES(580,12);
INSERT INTO Alocat
VALUES (560, 12);
INSERT INTO Alocat
VALUES (550, 12);
```

```sql
INSERT INTO Alocat
VALUES (490, 13);
INSERT INTO Alocat
VALUES (540, 13);
INSERT INTO Alocat
VALUES (520, 13);
INSERT INTO Alocat
VALUES (530, 13);

CREATE TABLE Ciclist
(
   id_ciclist   INT PRIMARY KEY,
   nume         VARCHAR(15) NOT NULL,
   prenume      VARCHAR(15) NOT NULL,
   sex          INT DEFAULT NULL,
   data_nastere DATE        NOT NULL,
   telefon      VARCHAR(16) NOT NULL,
   punctaj      INT DEFAULT 0
);

DROP TABLE Ciclist;
SELECT *
FROM Ciclist;
INSERT INTO Ciclist (id_ciclist, nume, prenume, sex, data_nastere, telefon,
punctaj)
VALUES (SEQ_CICL.NEXTVAL, 'Remy', 'Metailler', 1, DATE '1984-07-12',
'+812382189', 0);
INSERT INTO Ciclist (id_ciclist, nume, prenume, sex, data_nastere, telefon,
punctaj)
VALUES (SEQ_CICL.NEXTVAL, 'Vali', 'Holl', 0, DATE '1995-02-12',
'+1312321431', 0);
INSERT INTO Ciclist (id_ciclist, nume, prenume, sex, data_nastere, telefon,
punctaj)
VALUES (SEQ_CICL.NEXTVAL, 'Troy', 'Brosnan', 1, DATE '1989-01-02',
'+21321321421', 0);
INSERT INTO Ciclist (id_ciclist, nume, prenume, sex, data_nastere, telefon,
punctaj)
VALUES (SEQ_CICL.NEXTVAL, 'Jordie', 'Lynn', 1, DATE '1987-11-11',
'+213129321', 0);
INSERT INTO Ciclist (id_ciclist, nume, prenume, sex, data_nastere, telefon,
punctaj)
VALUES (SEQ_CICL.NEXTVAL, 'Corinne', 'Sutter', 0, DATE '2000-03-07',
'+312242141', 0);
```

```
INSERT INTO Ciclist (id_ciclist, nume, prenume, sex, data_nastere, telefon,
punctaj)
VALUES (SEQ_CICL.NEXTVAL, 'Nicole', 'Schimdhofer', 0, DATE '2001-09-11',
'+421893417', 0);
INSERT INTO Ciclist (id_ciclist, nume, prenume, sex, data_nastere, telefon,
punctaj)
VALUES (SEQ_CICL.NEXTVAL, 'Mihaela', 'Shiffrin', 0, DATE '1998-08-27',
'+4019238129', 0);
INSERT INTO Ciclist (id_ciclist, nume, prenume, sex, data_nastere, telefon,
punctaj)
VALUES (SEQ_CICL.NEXTVAL, 'Brandon', 'Semenuk', 1, DATE '2001-01-16',
'+918213123', 0);
INSERT INTO Ciclist (id_ciclist, nume, prenume, sex, data_nastere, telefon,
punctaj)
VALUES (SEQ_CICL.NEXTVAL, 'Brett', 'Rheeder', 1, DATE '1995-12-29',
'+94738475', 0);
INSERT INTO Ciclist (id_ciclist, nume, prenume, sex, data_nastere, telefon,
punctaj)
VALUES (SEQ_CICL.NEXTVAL, 'Tom', 'van Steenbergen', 1, DATE '1989-11-01',
'+238218219', 0);
INSERT INTO Ciclist (id_ciclist, nume, prenume, sex, data_nastere, telefon,
punctaj)
VALUES (SEQ_CICL.NEXTVAL, 'Brendan', 'Fairclough', 1, DATE '1999-02-12',
'+752978897', 0);


CREATE SEQUENCE SEQ_CICL
   START WITH 470
   INCREMENT BY 10
   MAXVALUE 1350;

DROP SEQUENCE SEQ_CICL;



CREATE TABLE Sponsor
(
   id_sponsor         INT PRIMARY KEY,
   nume               VARCHAR(25) NOT NULL,
   mainly_distributes VARCHAR(15) NOT NULL,
   website            VARCHAR(40) NOT NULL
);
```

```sql
INSERT INTO Sponsor
VALUES (13, 'NUKEPROOF', 'Pedale', 'nukeproof.com');
INSERT INTO Sponsor
VALUES (14, 'MAXXIS', 'Anvelope', 'maxxis.com');
INSERT INTO Sponsor
VALUES (15, 'Mucoff', 'Lubrifianti', 'mucoff.com');
INSERT INTO Sponsor
VALUES (16, 'KMC', 'Lanturi', 'kmc-chains.org');
INSERT INTO Sponsor
VALUES (17, 'WTB', 'Amnvelope', 'wtbtyres.com');
INSERT INTO Sponsor
VALUES (18, 'Cube', 'Cadre', 'cube.de');
INSERT INTO Sponsor
VALUES (19, 'Shimano', 'Deraioare', 'shimano.eu');
INSERT INTO Sponsor
VALUES (20, 'SRAM', 'Deraioare', 'sram.us');



CREATE TABLE Finantare
(
   sponsor   INT,
   campionat INT,
   PRIMARY KEY (sponsor, campionat),
   FOREIGN KEY (sponsor) REFERENCES Sponsor (id_sponsor),
   FOREIGN KEY (campionat) REFERENCES Campionat (id_camp)
);

INSERT INTO Finantare
VALUES (15, 2);
INSERT INTO Finantare
VALUES (19, 2);
INSERT INTO Finantare
VALUES (14, 2);
INSERT INTO Finantare
VALUES (20, 2);
INSERT INTO Finantare
VALUES (15, 1);
CREATE TABLE Promovari
(
   ciclist       INT,
   sponsor       INT,
   data_contract DATE NOT NULL,
```

```
   PRIMARY KEY (ciclist, sponsor),
   FOREIGN KEY (ciclist) REFERENCES Ciclist (id_ciclist) ON DELETE CASCADE,
   FOREIGN KEY (sponsor) REFERENCES Sponsor (id_sponsor) ON DELETE CASCADE
);

SELECT *
FROM PROMOVARI;

INSERT INTO Promovari
VALUES (480, 13, DATE '2022-09-01');
INSERT INTO Promovari
VALUES (490, 14, DATE '2022-09-03');
INSERT INTO Promovari
VALUES (490, 16, DATE '2022-09-02');
INSERT INTO Promovari
VALUES (500, 17, DATE '2022-09-03');
INSERT INTO Promovari
VALUES (500, 19, DATE '2022-09-06');
INSERT INTO Promovari
VALUES (520, 13, DATE '2022-09-07');
INSERT INTO Promovari
VALUES (530, 20, DATE '2022-09-02');
INSERT INTO Promovari
VALUES (530, 17, DATE '2022-09-03');
INSERT INTO Promovari
VALUES (540, 18, DATE '2022-09-12');
INSERT INTO Promovari
VALUES (540, 19, DATE '2022-09-11');
INSERT INTO Promovari
VALUES (540, 13, DATE '2022-09-21');

CREATE TABLE Medici
(
   id_medic       INT PRIMARY KEY,
   nume           VARCHAR(50) NOT NULL,
   varsta         INT         NOT NULL,
   exp_anterioara NUMBER(1) DEFAULT 0
);

SELECT *
FROM MEDICI;
INSERT INTO Medici
VALUES (112, 'Ion Popescu', 29, 1);
```

```sql
INSERT INTO Medici
VALUES (911, 'John Doe', 41, 1);
INSERT INTO Medici
VALUES (113, 'Elena Andronie', 32, 0);
INSERT INTO Medici
VALUES (114, 'Melinte Mihai', 51, 1);
INSERT INTO Medici
VALUES (912, 'John Deer', 54, 0);

CREATE TABLE Traseu
(
   id_traseu INT PRIMARY KEY,
   lungime   FLOAT NOT NULL,
   tip       CHAR
);

INSERT INTO Traseu
VALUES (1, 2.19, 'A');
INSERT INTO Traseu
VALUES (2, 3.32, 'B');
INSERT INTO Traseu
VALUES (3, 1.74, 'C');
INSERT INTO Traseu
VALUES (4, 4.02, 'D');

CREATE TABLE Superpromotie
(
   ciclist INT,
   traseu  INT,
   timp    INT NOT NULL,
   castiga NUMBER(1) DEFAULT 0,
   PRIMARY KEY (ciclist, traseu),
   FOREIGN KEY (ciclist) REFERENCES Ciclist (id_ciclist) ON DELETE CASCADE,
   FOREIGN KEY (traseu) REFERENCES Traseu (id_traseu)
);



INSERT INTO Superpromotie
VALUES (480, 1, 367, 0);
INSERT INTO Superpromotie
VALUES (490, 1, 421, 0);
INSERT INTO Superpromotie
```

```
VALUES (520, 2, 567, 0);
INSERT INTO Superpromotie
VALUES (570, 2, 556, 0);
INSERT INTO Superpromotie
VALUES (550, 3, 423, 0);
INSERT INTO Superpromotie
VALUES (500, 3, 367, 0);
INSERT INTO Superpromotie
VALUES (560, 3, 381, 0);


CREATE TABLE Orar
(
   grupa  INT,
   traseu INT,
   medic  INT,
   PRIMARY KEY (grupa, traseu, medic),
   FOREIGN KEY (grupa) REFERENCES Grupa (id_grupa) ON DELETE CASCADE,
   FOREIGN KEY (traseu) REFERENCES Traseu (id_traseu) ON DELETE CASCADE,
   FOREIGN KEY (medic) REFERENCES Medici (id_medic) ON DELETE CASCADE
);



INSERT INTO Orar
VALUES (11, 1, 112);
INSERT INTO Orar
VALUES (11, 3, 113);
INSERT INTO Orar
VALUES (12, 2, 912);
INSERT INTO Orar
VALUES (12, 4, 114);
INSERT INTO Orar
VALUES (13, 1, 113);
INSERT INTO Orar
VALUES (13, 4, 911);

-- Fiind cunoscut numele unui Campionat
-- sa se afiseze toate grupele
-- si concurentii participanti sau []
-- pentru oricare dintre trasee
-- ca JSON
```

```
CREATE OR REPLACE PROCEDURE ex6(numeTurneu CAMPIONAT.nume%type)
AS
   TYPE proto_table IS TABLE OF TRASEU%rowtype INDEX BY PLS_INTEGER;
   tracks    proto_table;
   TYPE proto2_table IS TABLE OF VARCHAR(80) INDEX BY PLS_INTEGER;
   persoane  proto2_table;
   TYPE proto3_nest IS TABLE OF GRUPA%rowtype;
   grupe     proto3_nest := proto3_nest();
   cantitate NUMBER(5);
BEGIN
   -- preiau traseele
   SELECT * BULK COLLECT
   INTO tracks
   FROM TRASEU;

   --numar grupele care apar in campionat
   SELECT COUNT(*)
   INTO cantitate
   FROM GRUPA g,
        DISCIPLINA d,
        CAMPIONAT c
   WHERE g.in_discipline = d.ID_DISCIPLINA
     AND d.IN_CHAMP = c.ID_CAMP
     AND LOWER(c.NUME) LIKE LOWER(numeTurneu);

   grupe.extend(cantitate + 1);

   -- iau grupele care apar in campionat
   SELECT g.ID_GRUPA,
          g.IN_DISCIPLINE BULK COLLECT
   INTO grupe
   FROM GRUPA g,
        DISCIPLINA d,
        CAMPIONAT c
   WHERE g.in_discipline = d.ID_DISCIPLINA
     AND d.IN_CHAMP = c.ID_CAMP
     AND LOWER(c.NUME) LIKE LOWER(numeTurneu);

   DBMS_OUTPUT.PUT_LINE('{');
   FOR i in tracks.first..tracks.LAST
       LOOP
           DBMS_OUTPUT.PUT_LINE('"Traseul' || tracks(i).tip ||
tracks(i).id_traseu || '":{');
```

```
            --

            FOR j in grupe.first..grupe.LAST
                LOOP
                    --                      DBMS_OUTPUT.PUT_LINE('>---<');
                    DBMS_OUTPUT.PUT_LINE('"Grupa' || grupe(j).id_grupa ||
'":');


                    SELECT c.nume || c.prenume BULK COLLECT
                    INTO persoane
                    FROM Ciclist c,
                         ORAR o,
                         alocat a
                    WHERE a.ciclist_ = c.id_ciclist
                      AND a.grupa_ = grupe(j).id_grupa
                      AND o.grupa = grupe(j).id_grupa
                      AND tracks(i).id_traseu = o.traseu;

                    IF persoane.COUNT > 0 THEN
                        DBMS_OUTPUT.PUT_LINE('[');
                        FOR k in persoane.first..persoane.LAST
                            LOOP
                                DBMS_OUTPUT.PUT_LINE('"' || persoane(k) ||
'",');
                            end loop;
                        DBMS_OUTPUT.PUT_LINE('],');
                    end if;
                    IF persoane.COUNT = 0 THEN
                        DBMS_OUTPUT.PUT_LINE('[],');
                    end if;

                end loop;
            DBMS_OUTPUT.PUT_LINE('}');
        end loop;
    DBMS_OUTPUT.PUT_LINE('}');
END;


BEGIN
    ex6('Downhill Word Championship');
end;
```

```
-- Ex 7
-- Fiind cunoscuta numele unei discipline (disciplinele sunt "singleton" si
se pot tine de un singur campionat)
-- Sa se arate numele sponsorilor
-- cu prezenta in grupe

CREATE OR REPLACE PROCEDURE ex7(name_of_discipline DISCIPLINA.NUME%TYPE)
AS
   CURSOR discipline(name_of_discipline DISCIPLINA.NUME%TYPE) IS
       SELECT d.id_disciplina
       FROM DISCIPLINA d
       WHERE LOWER(d.NUME) LIKE LOWER(name_of_discipline);
   CURSOR grupe(id_of_discipline DISCIPLINA.id_disciplina%TYPE) IS
       SELECT g.id_grupa
       FROM GRUPA g
       WHERE g.IN_DISCIPLINE = id_of_discipline;
   CURSOR ciclisti(id_of_group GRUPA.id_grupa%TYPE) IS
       SELECT a.ciclist_
       FROM ALOCAT a
       WHERE a.grupa_ = id_of_group;
   CURSOR sponsori(id_of_cyclist CICLIST.id_ciclist%TYPE) IS
       SELECT p.sponsor
       FROM PROMOVARI p
       WHERE p.ciclist = id_of_cyclist;
   v_id_grupa        GRUPA.id_grupa%TYPE;
   v_id_ciclist      CICLIST.id_ciclist%TYPE;
   v_id_sponsori     SPONSOR.id_sponsor%TYPE;
   name_of_a_sponsor SPONSOR.nume%TYPE;
BEGIN
   FOR disciplina in discipline(name_of_discipline)
       LOOP
           DBMS_OUTPUT.PUT_LINE(UPPER(name_of_discipline));
           OPEN grupe(disciplina.id_disciplina);
           LOOP
               FETCH grupe INTO v_id_grupa;
               EXIT WHEN grupe%NOTFOUND;
               DBMS_OUTPUT.PUT_LINE(' ');
               DBMS_OUTPUT.PUT_LINE('  Grupa ' || v_id_grupa || ':');
--

               OPEN ciclisti(v_id_grupa);
--                 DBMS_OUTPUT.PUT_LINE('Ciclisti:');
               LOOP
```

```
                    FETCH ciclisti INTO v_id_ciclist;
                    EXIT WHEN ciclisti%NOTFOUND;
--                      DBMS_OUTPUT.PUT_LINE(v_id_ciclist);

                    OPEN sponsori(v_id_ciclist);
--                      DBMS_OUTPUT.PUT_LINE('Sponsori:');
                    LOOP
                        FETCH sponsori INTO v_id_sponsori;
                        EXIT WHEN sponsori%NOTFOUND;
--
                        SELECT s.nume
                        INTO name_of_a_sponsor
                        FROM SPONSOR s
                        WHERE s.ID_SPONSOR = v_id_sponsori;
                        DBMS_OUTPUT.PUT_LINE(name_of_a_sponsor);
                    end loop;
--                          DBMS_OUTPUT.PUT_LINE('--');
                    CLOSE sponsori;

                end loop;

                CLOSE ciclisti;


            end loop;
            CLOSE grupe;
        end loop;
end;

BEGIN
   ex7('Gravity Downhill');
end;
-- begin
--     dbms_output.put_line('test');
-- end;

-- Federatia de Ciclism va cere sa
-- identificati dintr-o disciplina la alegere
-- numarul de participanti neconformi de gen
-- in scopul evaluarii calitatilor fizice
-- a.i sa nu existe inegalitati intre participanti
```

```
CREATE OR REPLACE FUNCTION ex8(name_of_discipline Disciplina.nume%type,
gender char)
   RETURN number
   IS
   checker_disciplina   NUMBER(10);
   nr_pers_identificate NUMBER(10);
   wrong_discipline EXCEPTION;
   wrong_gender EXCEPTION;
BEGIN

   SELECT COUNT(*) INTO checker_disciplina FROM Disciplina d WHERE d.nume =
name_of_discipline;

   IF checker_disciplina = 0 THEN
       RAISE wrong_discipline;
   end if;
   IF gender <> 'X' THEN
       RAISE wrong_gender;
   end if;


   SELECT COUNT(*)
   INTO nr_pers_identificate
   FROM (SELECT C2.sex
         FROM DISCIPLINA D
                  JOIN Grupa G ON D.id_disciplina = G.in_discipline
                  JOIN Alocat A2 ON G.id_grupa = A2.grupa_
                  JOIN Ciclist C2 ON C2.id_ciclist = A2.ciclist_
         WHERE D.nume = 'Gravity Downhill')
   WHERE sex = 2;


   RETURN nr_pers_identificate;

EXCEPTION
   WHEN wrong_discipline THEN
       DBMS_OUTPUT.PUT_LINE('Disciplina nu exista!');
       RETURN -1;
   WHEN wrong_gender THEN
       DBMS_OUTPUT.PUT_LINE('Necesar sa trimiteti ca parametru litera X');
       RETURN -1;
end;
```

```
BEGIN
   DBMS_OUTPUT.PUT_LINE(ex8('Gravity Downhill', 'X'));
end;

-- Sponsorii propun asistenta medicala terta-parte
-- asa ca se adreseaza dvs. pentru a consulta
-- relatia dintre grupele la care fac parte concurentii lor
-- si medicii disponibili


-- tabel global temporar trebuie rulat inainte de ex9 se va termina cand se
inchide sesiunea
CREATE GLOBAL TEMPORARY TABLE my_temp_table
(
   nume1 VARCHAR2(50),
   nume2 VARCHAR2(50)
) ON COMMIT PRESERVE ROWS;

CREATE OR REPLACE PROCEDURE ex9(name_of_sponsor Sponsor.nume%TYPE)
AS
   TYPE for_sponsors_proto IS TABLE OF Sponsor%ROWTYPE;
   for_sponsor for_sponsors_proto;
   CURSOR my_cursor IS
       SELECT nume1, nume2
       FROM my_temp_table;

   TOO_MANY_ROWS EXCEPTION;
   NO_DATA_FOUND EXCEPTION;

BEGIN
   DELETE FROM my_temp_table;

   SELECT * BULK COLLECT
   INTO for_sponsor
   FROM Sponsor
   WHERE nume = name_of_sponsor;
   IF for_sponsor.COUNT > 1 THEN
       RAISE TOO_MANY_ROWS;
   end if;
   IF for_sponsor.COUNT = 0 THEN
       RAISE NO_DATA_FOUND;
   end if;
```

```
   INSERT INTO my_temp_table (nume1, nume2)
   SELECT m.nume, C2.nume
   FROM MEDICI m
            JOIN Orar O on m.id_medic = O.medic
            JOIN Grupa G on G.id_grupa = O.grupa
            JOIN Alocat A2 on G.id_grupa = A2.grupa_
            JOIN Ciclist C2 on C2.id_ciclist = A2.ciclist_
            JOIN Promovari P on C2.id_ciclist = P.ciclist
            JOIN Sponsor S on S.id_sponsor = P.sponsor
   WHERE S.nume = name_of_sponsor;

   FOR row IN my_cursor
       LOOP
           dbms_output.put_line('HP: ' || row.nume1 || '/' || row.nume2);
       END LOOP;

EXCEPTION
   WHEN TOO_MANY_ROWS THEN
       DBMS_OUTPUT.PUT_LINE('Problema cu entry in sponsori: DUPLICATE');
   WHEN NO_DATA_FOUND THEN
       DBMS_OUTPUT.PUT_LINE('Nu exista sponsor ' || name_of_sponsor);
END;


-- pentru test am dublat linia de sponsor SRAM
BEGIN
   ex9('SRAM');
end;

-- Vacantele sunt momente in care
-- studentul trebuie sa isi adune puterea pentru sesiune.
-- Opreste lucrul la SGBD in timpul vacantei FMI.


CREATE OR
   REPLACE TRIGGER ex10
   BEFORE
       INSERT OR UPDATE OR
       DELETE
   ON ORAR
DECLARE
   current_date DATE;
BEGIN
```

```
   current_date := SYSDATE;
   IF current_date BETWEEN to_date('23-DEC-2022', 'DD-MON-YYYY') AND
to_date('09-JAN-2023', 'DD-MON-YYYY') THEN
       DBMS_OUTPUT.PUT_LINE('Ia o pauza! Ne vedem dupa sarbatori.');
   END IF;
END;

-- INSERT INTO ORAR VALUES (11, 1, 911);

-- Implementati varsta de pensionare a medicilor

CREATE OR REPLACE TRIGGER ex11
AFTER INSERT OR UPDATE
ON MEDICI
FOR EACH ROW
BEGIN
 IF :NEW.VARSTA > 67 THEN
   RAISE_APPLICATION_ERROR(-20000, 'Age cannot be higher than 67');
 END IF;
END;

-- Log-uiti orice operatiune facuta
CREATE OR REPLACE TRIGGER ex12
   AFTER CREATE OR ALTER OR DROP ON SCHEMA
BEGIN
   DBMS_OUTPUT.PUT_LINE(sys.LOGIN_USER || ' '|| sys.sysevent ||' '||
sys.database_name);
END;

DROP TABLE TEST;
CREATE TABLE Test(id INT PRIMARY KEY );

CREATE OR REPLACE PACKAGE sbgd_NAFORNITA_ADRIAN_VALENTIN AS
   CREATE OR REPLACE PROCEDURE ex6(numeTurneu CAMPIONAT.nume%type);
   CREATE OR REPLACE PROCEDURE ex7(name_of_discipline
DISCIPLINA.NUME%TYPE);
   CREATE OR REPLACE FUNCTION  ex8(name_of_discipline Disciplina.nume%type,
gender char);
   CREATE OR REPLACE PROCEDURE ex9(name_of_sponsor Sponsor.nume%TYPE);
END sbgd_NAFORNITA_ADRIAN_VALENTIN;

CREATE OR REPLACE PACKAGE BODY sgbd_NAFORNITA_ADRIAN_VALENTIN
   AS
```

```
   CREATE OR REPLACE PROCEDURE ex6(numeTurneu CAMPIONAT.nume%type)
AS
   TYPE proto_table IS TABLE OF TRASEU%rowtype INDEX BY PLS_INTEGER;
   tracks    proto_table;
   TYPE proto2_table IS TABLE OF VARCHAR(80) INDEX BY PLS_INTEGER;
   persoane  proto2_table;
   TYPE proto3_nest IS TABLE OF GRUPA%rowtype;
   grupe     proto3_nest := proto3_nest();
   cantitate NUMBER(5);
BEGIN
   -- preiau traseele
   SELECT * BULK COLLECT
   INTO tracks
   FROM TRASEU;

   --numar grupele care apar in campionat
   SELECT COUNT(*)
   INTO cantitate
   FROM GRUPA g,
        DISCIPLINA d,
        CAMPIONAT c
   WHERE g.in_discipline = d.ID_DISCIPLINA
     AND d.IN_CHAMP = c.ID_CAMP
     AND LOWER(c.NUME) LIKE LOWER(numeTurneu);

   grupe.extend(cantitate + 1);

   -- iau grupele care apar in campionat
   SELECT g.ID_GRUPA,
          g.IN_DISCIPLINE BULK COLLECT
   INTO grupe
   FROM GRUPA g,
        DISCIPLINA d,
        CAMPIONAT c
   WHERE g.in_discipline = d.ID_DISCIPLINA
     AND d.IN_CHAMP = c.ID_CAMP
     AND LOWER(c.NUME) LIKE LOWER(numeTurneu);

   DBMS_OUTPUT.PUT_LINE('{');
   FOR i in tracks.first..tracks.LAST
       LOOP
           DBMS_OUTPUT.PUT_LINE('"Traseul' || tracks(i).tip ||
tracks(i).id_traseu || '":{');
```

```
            --

            FOR j in grupe.first..grupe.LAST
                LOOP
                    --                      DBMS_OUTPUT.PUT_LINE('>---<');
                    DBMS_OUTPUT.PUT_LINE('"Grupa' || grupe(j).id_grupa ||
'":');


                    SELECT c.nume || c.prenume BULK COLLECT
                    INTO persoane
                    FROM Ciclist c,
                         ORAR o,
                         alocat a
                    WHERE a.ciclist_ = c.id_ciclist
                      AND a.grupa_ = grupe(j).id_grupa
                      AND o.grupa = grupe(j).id_grupa
                      AND tracks(i).id_traseu = o.traseu;

                    IF persoane.COUNT > 0 THEN
                        DBMS_OUTPUT.PUT_LINE('[');
                        FOR k in persoane.first..persoane.LAST
                            LOOP
                                DBMS_OUTPUT.PUT_LINE('"' || persoane(k) ||
'",');
                            end loop;
                        DBMS_OUTPUT.PUT_LINE('],');
                    end if;
                    IF persoane.COUNT = 0 THEN
                        DBMS_OUTPUT.PUT_LINE('[],');
                    end if;

                end loop;
            DBMS_OUTPUT.PUT_LINE('}');
        end loop;
    DBMS_OUTPUT.PUT_LINE('}');
END ex6;
        CREATE OR REPLACE PROCEDURE ex7(name_of_discipline
DISCIPLINA.NUME%TYPE)
AS
    CURSOR discipline(name_of_discipline DISCIPLINA.NUME%TYPE) IS
        SELECT d.id_disciplina
        FROM DISCIPLINA d
```

```
        WHERE LOWER(d.NUME) LIKE LOWER(name_of_discipline);
    CURSOR grupe(id_of_discipline DISCIPLINA.id_disciplina%TYPE) IS
        SELECT g.id_grupa
        FROM GRUPA g
        WHERE g.IN_DISCIPLINE = id_of_discipline;
    CURSOR ciclisti(id_of_group GRUPA.id_grupa%TYPE) IS
        SELECT a.ciclist_
        FROM ALOCAT a
        WHERE a.grupa_ = id_of_group;
    CURSOR sponsori(id_of_cyclist CICLIST.id_ciclist%TYPE) IS
        SELECT p.sponsor
        FROM PROMOVARI p
        WHERE p.ciclist = id_of_cyclist;
    v_id_grupa        GRUPA.id_grupa%TYPE;
    v_id_ciclist      CICLIST.id_ciclist%TYPE;
    v_id_sponsori     SPONSOR.id_sponsor%TYPE;
    name_of_a_sponsor SPONSOR.nume%TYPE;
BEGIN
    FOR disciplina in discipline(name_of_discipline)
        LOOP
            DBMS_OUTPUT.PUT_LINE(UPPER(name_of_discipline));
            OPEN grupe(disciplina.id_disciplina);
            LOOP
                FETCH grupe INTO v_id_grupa;
                EXIT WHEN grupe%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE(' ');
                DBMS_OUTPUT.PUT_LINE('  Grupa ' || v_id_grupa || ':');
--

                OPEN ciclisti(v_id_grupa);
--                  DBMS_OUTPUT.PUT_LINE('Ciclisti:');
                LOOP
                    FETCH ciclisti INTO v_id_ciclist;
                    EXIT WHEN ciclisti%NOTFOUND;
--                      DBMS_OUTPUT.PUT_LINE(v_id_ciclist);

                    OPEN sponsori(v_id_ciclist);
--                      DBMS_OUTPUT.PUT_LINE('Sponsori:');
                    LOOP
                        FETCH sponsori INTO v_id_sponsori;
                        EXIT WHEN sponsori%NOTFOUND;
--
                        SELECT s.nume
```

```
                        INTO name_of_a_sponsor
                        FROM SPONSOR s
                        WHERE s.ID_SPONSOR = v_id_sponsori;
                        DBMS_OUTPUT.PUT_LINE(name_of_a_sponsor);
                    end loop;
--                      DBMS_OUTPUT.PUT_LINE('--');
                    CLOSE sponsori;

                end loop;

                CLOSE ciclisti;


            end loop;
            CLOSE grupe;
        end loop;
end ex7;
            CREATE OR REPLACE FUNCTION ex8(name_of_discipline
Disciplina.nume%type, gender char)
    RETURN number
    IS
    checker_disciplina   NUMBER(10);
    nr_pers_identificate NUMBER(10);
    wrong_discipline EXCEPTION;
    wrong_gender EXCEPTION;
BEGIN

    SELECT COUNT(*) INTO checker_disciplina FROM Disciplina d WHERE d.nume =
name_of_discipline;

    IF checker_disciplina = 0 THEN
        RAISE wrong_discipline;
    end if;
    IF gender <> 'X' THEN
        RAISE wrong_gender;
    end if;


    SELECT COUNT(*)
    INTO nr_pers_identificate
    FROM (SELECT C2.sex
          FROM DISCIPLINA D
                   JOIN Grupa G ON D.id_disciplina = G.in_discipline
```

```
                JOIN Alocat A2 ON G.id_grupa = A2.grupa_
                JOIN Ciclist C2 ON C2.id_ciclist = A2.ciclist_
        WHERE D.nume = 'Gravity Downhill')
   WHERE sex = 2;


   RETURN nr_pers_identificate;

EXCEPTION
   WHEN wrong_discipline THEN
       DBMS_OUTPUT.PUT_LINE('Disciplina nu exista!');
       RETURN -1;
   WHEN wrong_gender THEN
       DBMS_OUTPUT.PUT_LINE('Necesar sa trimiteti ca parametru litera X');
       RETURN -1;
end ex8;
              CREATE OR REPLACE PROCEDURE ex9(name_of_sponsor
Sponsor.nume%TYPE)
AS
   TYPE for_sponsors_proto IS TABLE OF Sponsor%ROWTYPE;
   for_sponsor for_sponsors_proto;
   CURSOR my_cursor IS
       SELECT nume1, nume2
       FROM my_temp_table;

   TOO_MANY_ROWS EXCEPTION;
   NO_DATA_FOUND EXCEPTION;

BEGIN
   DELETE FROM my_temp_table;

   SELECT * BULK COLLECT
   INTO for_sponsor
   FROM Sponsor
   WHERE nume = name_of_sponsor;
   IF for_sponsor.COUNT > 1 THEN
       RAISE TOO_MANY_ROWS;
   end if;
   IF for_sponsor.COUNT = 0 THEN
       RAISE NO_DATA_FOUND;
   end if;

   INSERT INTO my_temp_table (nume1, nume2)
```

```
   SELECT m.nume, C2.nume
   FROM MEDICI m
            JOIN Orar O on m.id_medic = O.medic
            JOIN Grupa G on G.id_grupa = O.grupa
            JOIN Alocat A2 on G.id_grupa = A2.grupa_
            JOIN Ciclist C2 on C2.id_ciclist = A2.ciclist_
            JOIN Promovari P on C2.id_ciclist = P.ciclist
            JOIN Sponsor S on S.id_sponsor = P.sponsor
   WHERE S.nume = name_of_sponsor;

   FOR row IN my_cursor
       LOOP
           dbms_output.put_line('HP: ' || row.nume1 || '/' || row.nume2);
       END LOOP;

EXCEPTION
   WHEN TOO_MANY_ROWS THEN
       DBMS_OUTPUT.PUT_LINE('Problema cu entry in sponsori: DUPLICATE');
   WHEN NO_DATA_FOUND THEN
       DBMS_OUTPUT.PUT_LINE('Nu exista sponsor ' || name_of_sponsor);
END ex9;
END sgbd_NAFORNITA_ADRIAN_VALENTIN;


INSERT INTO MEDICI VALUES(1,'XYZ', 70,1);
-- SELECT varsta
-- FROM MEDICI;

-- SELECT column_name
-- FROM all_tab_columns
-- WHERE table_name = 'MEDICI';
```

EX6

```
ORACLE> BEGIN
            ex6('Downhill Word Championship');
        end;
[2023-01-13 22:26:08] completed in 54 ms
{
"TraseulA1":{
"Grupa11":
[
"RemyMetailler",
"ValiHoll",
"JordieLynn",
"CorinneSutter",
"RemyMetailler",
"ValiHoll",
"JordieLynn",
"CorinneSutter",
],
"Grupa12":
[],
"Grupa13":
[
```

EX7

```
        end;
[2023-01-13 22:27:15] completed in 17 ms
GRAVITY DOWNHILL

  Grupa 11:
NUKEPROOF
WTB
Shimano

  Grupa 12:

  Grupa 13:
MAXXIS
KMC
NUKEPROOF
WTB
SRAM
NUKEPROOF
```

EX8

```
ORACLE> BEGIN
            DBMS_OUTPUT.PUT_LINE(ex8('Gravity Downhill', 'X'));
        end;
[2023-01-13 22:27:44] completed in 22 ms
1
```

```
1
ORACLE> BEGIN
            DBMS_OUTPUT.PUT_LINE(ex8('Gravity Downhill', 'Y'));
        end;
[2023-01-13 22:28:03] completed in 23 ms
Necesar sa trimiteti ca parametru litera X
-1
```

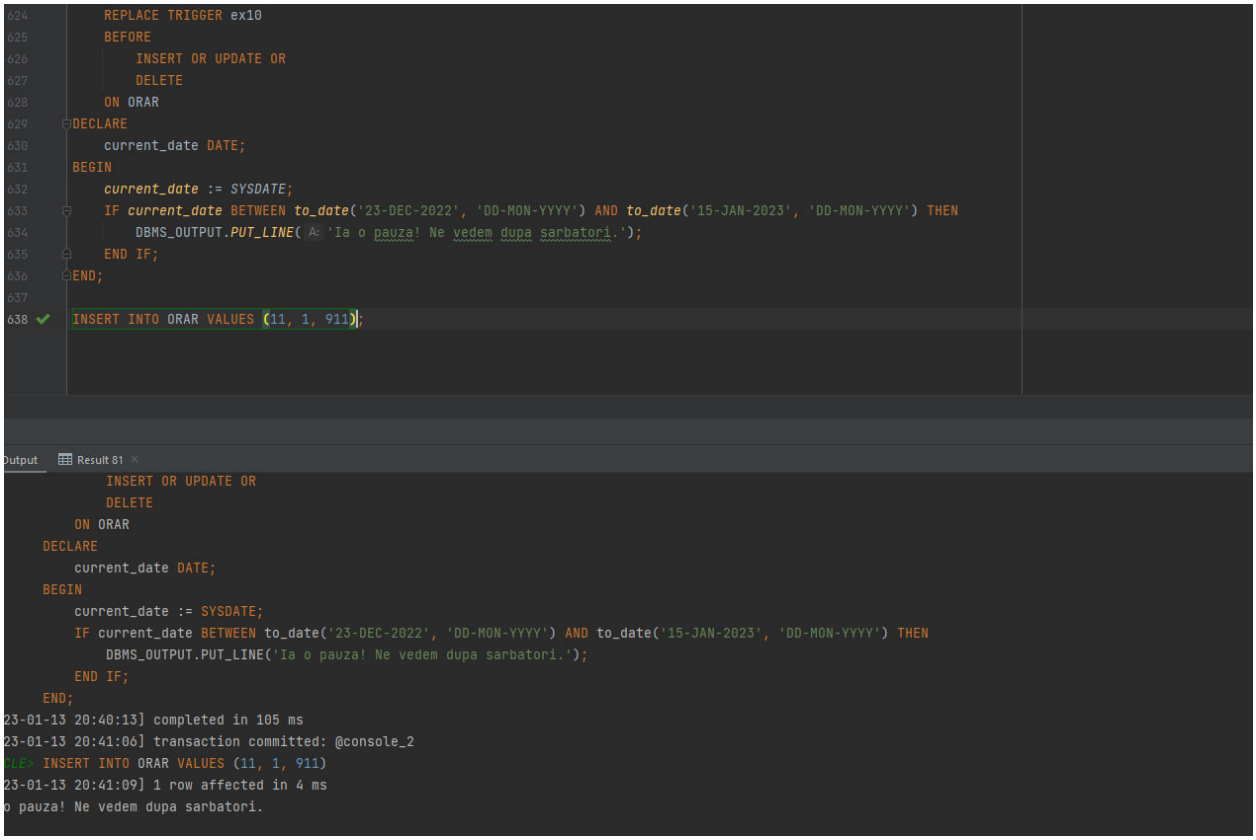
```
-1
ORACLE> BEGIN
            DBMS_OUTPUT.PUT_LINE(ex8('Gravity Downhil', 'Y'));
        end;
[2023-01-13 22:28:22] completed in 5 ms
Disciplina nu exista!
-1
```

EX9

```
ORACLE> BEGIN
            ex9('SRAM');
        end;
[2023-01-13 22:29:39] completed in 26 ms
HP: Elena Andronie/Mihaela
HP: John Doe/Mihaela
```

```
ORACLE> BEGIN
            ex9('MAXIS');
        end;
[2023-01-13 22:30:03] completed in 5 ms
Nu exista sponsor MAXIS
```

TRIGGER A



```
624     REPLACE TRIGGER ex10
625     BEFORE
626         INSERT OR UPDATE OR
627         DELETE
628     ON ORAR
629 DECLARE
630     current_date DATE;
631 BEGIN
632     current_date := SYSDATE;
633     IF current_date BETWEEN to_date('23-DEC-2022', 'DD-MON-YYYY') AND to_date('15-JAN-2023', 'DD-MON-YYYY') THEN
634         DBMS_OUTPUT.PUT_LINE( A: 'Ia o pauza! Ne vedem dupa sarbatori.');
635     END IF;
636 END;
637
638 INSERT INTO ORAR VALUES (11, 1, 911);
```

```
Output   Result 81
        INSERT OR UPDATE OR
        DELETE
    ON ORAR
DECLARE
    current_date DATE;
BEGIN
    current_date := SYSDATE;
    IF current_date BETWEEN to_date('23-DEC-2022', 'DD-MON-YYYY') AND to_date('15-JAN-2023', 'DD-MON-YYYY') THEN
        DBMS_OUTPUT.PUT_LINE('Ia o pauza! Ne vedem dupa sarbatori.');
    END IF;
END;
23-01-13 20:40:13] completed in 105 ms
23-01-13 20:41:06] transaction committed: @console_2
TLE> INSERT INTO ORAR VALUES (11, 1, 911)
23-01-13 20:41:09] 1 row affected in 4 ms
o pauza! Ne vedem dupa sarbatori.
```

TRIGGER B

```
CREATE OR REPLACE TRIGGER ex11
 AFTER INSERT OR UPDATE
 ON MEDICI
 FOR EACH ROW
BEGIN
  IF :NEW.VARSTA > 67 THEN
    RAISE_APPLICATION_ERROR(-20000, 'Age cannot be higher than 67');
  END IF;
END;


INSERT INTO MEDICI VALUES(1,'XYZ', 70,1);
SELECT varsta
FROM MEDICI;

SELECT column_name
```

```
[72000][20000]
ORA-20000: Age cannot be higher than 67
ORA-06512: at "ORACLE.MEDIC_TRG", line 3
ORA-04088: error during execution of trigger 'ORACLE.MEDIC_TRG'
Position: 12
```

utput ORACLE.MEDICI

```
    IF :NEW.VARSTA > 67 THEN
      RAISE_APPLICATION_ERROR(-20000, 'Age cannot be higher than 67');
    END IF;
  END;
3-01-13 21:16:09] completed in 27 ms
LE> INSERT INTO MEDICI VALUES(1,'XYZ', 70,1)
3-01-13 21:16:14] [72000][20000]
3-01-13 21:16:14]   ORA-20000: Age cannot be higher than 67
3-01-13 21:16:14]   ORA-06512: at "ORACLE.MEDIC_TRG", line 3
3-01-13 21:16:14]   ORA-04088: error during execution of trigger 'ORACLE.MEDIC_TRG'
3-01-13 21:16:14] Position: 12
```

TRIGGER C

```
651
652   -- Log-uiti orice operatiune facuta
653   CREATE OR REPLACE TRIGGER ex12
654       AFTER CREATE OR ALTER OR DROP ON SCHEMA
655   BEGIN
656       DBMS_OUTPUT.PUT_LINE( A: sys.LOGIN_USER || ' '|| sys.sysevent ||' '|| sys.database_name
657   END;
658
659   DROP TABLE TEST;
660   CREATE TABLE Test(id INT PRIMARY KEY );
661
662
663   INSERT INTO MEDICI VALUES(1,'XYZ', 70,1);
664   -- SELECT varsta
665   -- FROM MEDICI;
666
667   -- SELECT column_name
668   -- FROM all_tab_columns
```

Test

```
ORACLE> CREATE TABLE Test(id INT PRIMARY KEY )
[2023-01-13 21:52:30] completed in 201 ms
ORACLE CREATE XE
ORACLE CREATE XE
```