Proiect SGBD Restaurant

MINEA ȘTEFAN TEODOR

SERIA 4

GRUPA 241

AN 2025-2026

Cuprins

[Introducere 1](#_Toc221641991)

[Cerinta 1 2](#_Toc221641992)

[Cerinta 2 3](#_Toc221641993)

[Cerinta 3 4](#_Toc221641994)

[Cerinta 4 5](#_Toc221641995)

[Script creare tabele: 5](#_Toc221641996)

[Script stergere tabele: 11](#_Toc221641997)

[Cerinta 5 13](#_Toc221641998)

[Print-screen: 22](#_Toc221641999)

[Cerinta 6 27](#_Toc221642000)

[Cerinta: 27](#_Toc221642001)

[Rezolvare: 27](#_Toc221642002)

[Print-screen: 38](#_Toc221642003)

[Cerinta 7 44](#_Toc221642004)

[Cerinta: 44](#_Toc221642005)

[Rezolvare: 44](#_Toc221642006)

[Print-screen: 52](#_Toc221642007)

[Cerinta 8 61](#_Toc221642008)

[Cerinta: 61](#_Toc221642009)

[Rezolvare: 61](#_Toc221642010)

[Print-screen: 68](#_Toc221642011)

[Cerinta 9 70](#_Toc221642012)

[Cerinta: 70](#_Toc221642013)

[Rezolvare: 70](#_Toc221642014)

[Print-screen: 82](#_Toc221642015)

[Cerinta 10 86](#_Toc221642016)

[Cerinta: 86](#_Toc221642017)

[Rezolvare: 86](#_Toc221642018)

[Print-screen: 91](#_Toc221642019)

[Cerinta 11 92](#_Toc221642020)

[Cerinta: 92](#_Toc221642021)

[Rezolvare: 92](#_Toc221642022)

[Print-screen: 97](#_Toc221642023)

[Cerinta 12 98](#_Toc221642024)

[Cerinta: 98](#_Toc221642025)

[Rezolvare: 98](#_Toc221642026)

[Print-screen: 110](#_Toc221642027)

[Cerinta 13 112](#_Toc221642028)

[Cerinta: 112](#_Toc221642029)

[Rezolvare: 112](#_Toc221642030)

[Print-screen: 133](#_Toc221642031)

# Introducere

Am realizat acest proiect în Oracle Database 21c Express Edition Release 21.0.0.0.0 – Production. SBGD-ul rulează pe un sistem de operare Windows 11, în mașină virtuală creată cu Vmware.

# Cerinta 1

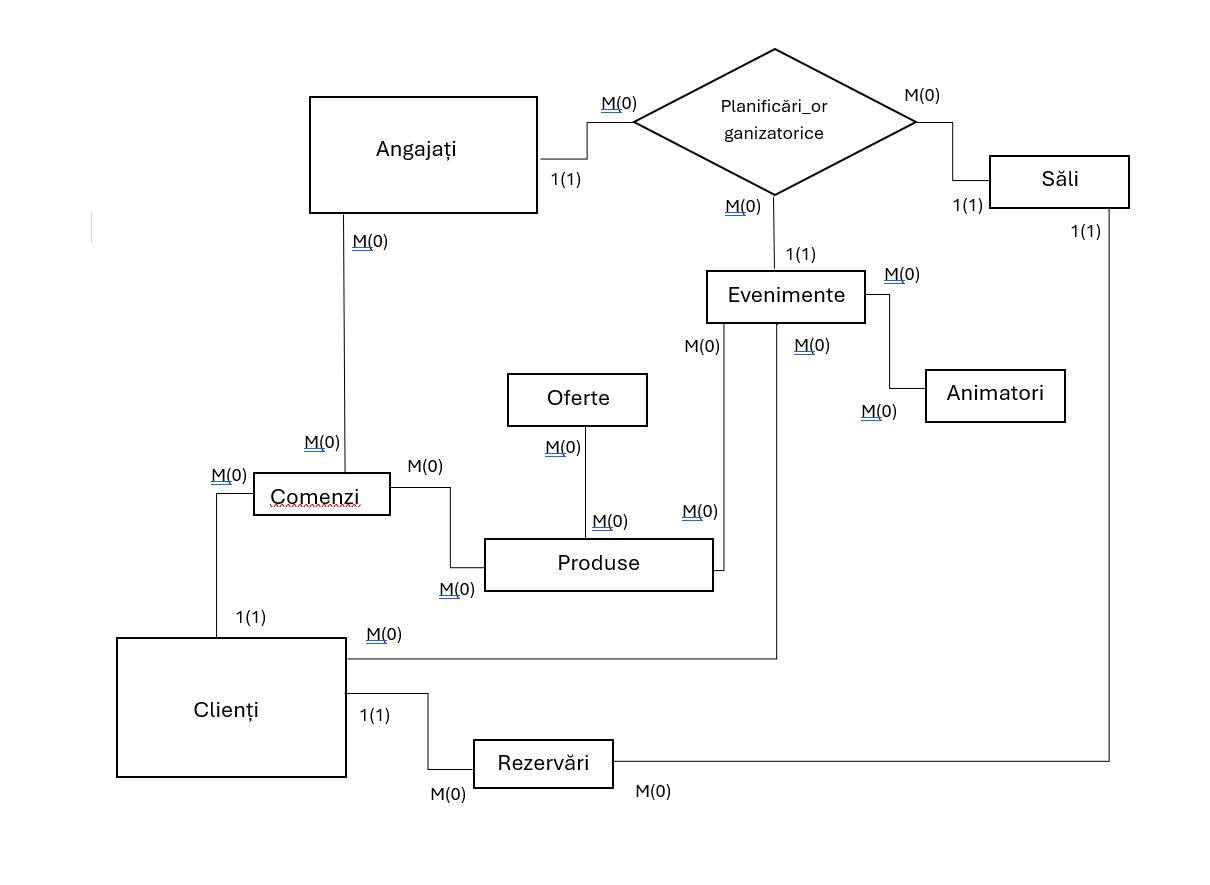
Baza de date modelează activitatea unui **restaurant / locație de evenimente**, având ca scop administrarea integrată a interacțiunilor cu clienții, a rezervărilor, a comenzilor și a organizării evenimentelor. Sistemul permite urmărirea atât a operațiunilor zilnice (rezervări și comenzi), cât și a activităților planificate (evenimente, colaboratori, produse necesare și implicarea personalului).

Prin această bază de date se pot gestiona, într-un mod coerent și centralizat:

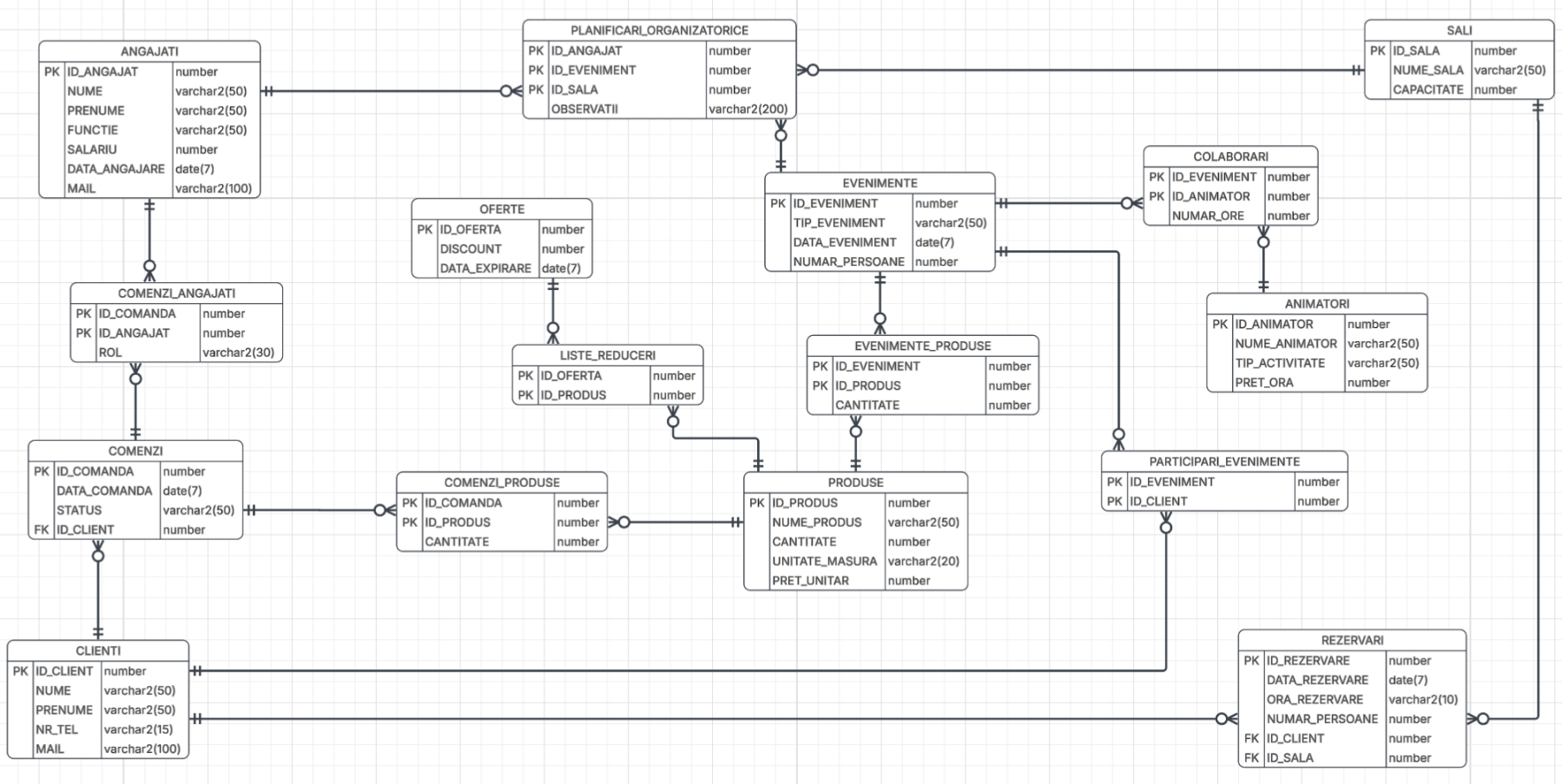
* **Clienții** și istoricul lor (rezervări efectuate, comenzi plasate, participări la evenimente);
* **Rezervările** pe săli, cu evidențierea capacității și a disponibilității;
* **Comenzile** și detalierea lor pe produse (cantități, comenzi asociate unui client, personal implicat);
* **Produsele** disponibile și utilizarea lor în comenzi sau evenimente;
* **Evenimentele** organizate (tip, dată, număr de persoane), împreună cu:
  + produsele necesare evenimentului,
  + animatorii / colaboratorii implicați,
  + planificarea personalului pe săli (cine lucrează la ce eveniment și în ce sală);
* **Ofertele / reducerile**, prin asocierea unor produse la promoții cu valabilitate.

În ansamblu, baza de date oferă suport pentru decizii operaționale (programare săli, evitare conflicte, capacitate) și pentru analize (activitate client, consum produse, implicarea angajaților, organizarea evenimentelor).

# Cerinta 2

ERD:

# Cerinta 3

Diagrama conceptuala:

# Cerinta 4

## Script creare tabele:

CREATE TABLE Angajati (

id\_angajat NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

nume VARCHAR2(50) NOT NULL,

prenume VARCHAR2(50) NOT NULL,

functie VARCHAR2(50) NOT NULL,

salariu NUMBER(10,2) NOT NULL CHECK (salariu > 0),

data\_angajare DATE DEFAULT SYSDATE NOT NULL,

mail VARCHAR2(100) NOT NULL

-- CONSTRAINT uq\_angajati\_mail UNIQUE (mail)

);

CREATE TABLE Clienti (

id\_client NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

nume VARCHAR2(50) NOT NULL,

prenume VARCHAR2(50) NOT NULL,

nr\_tel VARCHAR2(15) NOT NULL,

mail VARCHAR2(100) NOT NULL

-- CONSTRAINT uq\_clienti\_mail UNIQUE (mail)

);

CREATE TABLE Sali (

id\_sala NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

nume\_sala VARCHAR2(50) NOT NULL,

capacitate NUMBER NOT NULL CHECK (capacitate > 0)

);

CREATE TABLE Evenimente (

id\_eveniment NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

tip\_eveniment VARCHAR2(50) NOT NULL,

data\_eveniment DATE NOT NULL,

numar\_persoane NUMBER NOT NULL CHECK (numar\_persoane > 0)

);

CREATE TABLE Animatori (

id\_animator NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

nume\_animator VARCHAR2(50) NOT NULL,

tip\_activitate VARCHAR2(50) NOT NULL,

pret\_ora NUMBER(10,2) NOT NULL CHECK (pret\_ora > 0)

);

CREATE TABLE Oferte (

id\_oferta NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

discount NUMBER(5,2) NOT NULL CHECK (discount >= 0 AND discount <= 100),

data\_expirare DATE NOT NULL

);

CREATE TABLE Produse (

id\_produs NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

nume\_produs VARCHAR2(50) NOT NULL,

cantitate NUMBER NOT NULL CHECK (cantitate >= 0),

unitate\_masura VARCHAR2(20) NOT NULL CHECK (unitate\_masura IN ('buc', 'portie', 'sticla', 'pahar')),

pret\_unitar NUMBER(10,2) NOT NULL CHECK (pret\_unitar > 0)

);

CREATE TABLE Rezervari (

id\_rezervare NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

data\_rezervare DATE NOT NULL,

ora\_rezervare VARCHAR2(10) NOT NULL

CHECK ( REGEXP\_LIKE(ora\_rezervare, '^([0-1][0-9]|2[0-3]):[0-5][0-9]$') ),

numar\_persoane NUMBER NOT NULL CHECK (numar\_persoane > 0),

id\_client NUMBER NOT NULL,

id\_sala NUMBER NOT NULL,

CONSTRAINT fk\_rez\_client FOREIGN KEY (id\_client) REFERENCES Clienti(id\_client),

CONSTRAINT fk\_rez\_sala FOREIGN KEY (id\_sala) REFERENCES Sali(id\_sala)

);

CREATE TABLE Comenzi (

id\_comanda NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

data\_comanda DATE DEFAULT SYSDATE NOT NULL,

status VARCHAR2(50) NOT NULL

CHECK (status IN ('in redactare', 'plasata', 'in curs de preparare', 'platita', 'anulata')),

-- pret\_total NUMBER(10,2) NOT NULL CHECK (pret\_total >= 0),

id\_client NUMBER NOT NULL,

CONSTRAINT fk\_com\_client FOREIGN KEY (id\_client) REFERENCES Clienti(id\_client)

);

-- =========================

-- CREATE (relatii / tabele asociative)

-- =========================

-- Comenzi M:N Produse (o comanda contine produse)

CREATE TABLE Comenzi\_Produse (

id\_comanda NUMBER NOT NULL,

id\_produs NUMBER NOT NULL,

cantitate NUMBER NOT NULL CHECK (cantitate > 0),

PRIMARY KEY (id\_comanda, id\_produs),

CONSTRAINT fk\_cp\_comanda FOREIGN KEY (id\_comanda) REFERENCES Comenzi(id\_comanda),

CONSTRAINT fk\_cp\_produs FOREIGN KEY (id\_produs) REFERENCES Produse(id\_produs)

);

-- Comenzi M:N Angajati (o comanda este preluata/gestionata de angajati)

CREATE TABLE Comenzi\_Angajati (

id\_comanda NUMBER NOT NULL,

id\_angajat NUMBER NOT NULL,

rol VARCHAR2(30),

PRIMARY KEY (id\_comanda, id\_angajat),

CONSTRAINT fk\_ca\_comanda FOREIGN KEY (id\_comanda) REFERENCES Comenzi(id\_comanda),

CONSTRAINT fk\_ca\_angajat FOREIGN KEY (id\_angajat) REFERENCES Angajati(id\_angajat)

);

-- Oferte M:N Produse (produse incluse intr-o oferta)

CREATE TABLE Liste\_reduceri (

id\_oferta NUMBER NOT NULL,

id\_produs NUMBER NOT NULL,

PRIMARY KEY (id\_oferta, id\_produs),

CONSTRAINT fk\_lr\_oferta FOREIGN KEY (id\_oferta) REFERENCES Oferte(id\_oferta),

CONSTRAINT fk\_lr\_produs FOREIGN KEY (id\_produs) REFERENCES Produse(id\_produs)

);

-- Evenimente M:N Produse (produse necesare la un eveniment)

CREATE TABLE Evenimente\_Produse (

id\_eveniment NUMBER NOT NULL,

id\_produs NUMBER NOT NULL,

cantitate NUMBER NOT NULL CHECK (cantitate > 0),

PRIMARY KEY (id\_eveniment, id\_produs),

CONSTRAINT fk\_ep\_eveniment FOREIGN KEY (id\_eveniment) REFERENCES Evenimente(id\_eveniment),

CONSTRAINT fk\_ep\_produs FOREIGN KEY (id\_produs) REFERENCES Produse(id\_produs)

);

-- Evenimente M:N Animatori (colaborari)

CREATE TABLE Colaborari (

id\_eveniment NUMBER NOT NULL,

id\_animator NUMBER NOT NULL,

numar\_ore NUMBER NOT NULL,

PRIMARY KEY (id\_eveniment, id\_animator),

CONSTRAINT fk\_col\_eveniment FOREIGN KEY (id\_eveniment) REFERENCES Evenimente(id\_eveniment),

CONSTRAINT fk\_col\_animator FOREIGN KEY (id\_animator) REFERENCES Animatori(id\_animator)

);

-- Clienti M:N Evenimente (participari)

CREATE TABLE Participari\_evenimente (

id\_eveniment NUMBER NOT NULL,

id\_client NUMBER NOT NULL,

PRIMARY KEY (id\_eveniment, id\_client),

CONSTRAINT fk\_pe\_eveniment FOREIGN KEY (id\_eveniment) REFERENCES Evenimente(id\_eveniment),

CONSTRAINT fk\_pe\_client FOREIGN KEY (id\_client) REFERENCES Clienti(id\_client)

);

-- Relatie ternara: Angajati x Evenimente x Sali (planificari organizatorice)

CREATE TABLE Planificari\_organizatorice (

id\_angajat NUMBER NOT NULL,

id\_eveniment NUMBER NOT NULL,

id\_sala NUMBER NOT NULL,

observatii VARCHAR2(200),

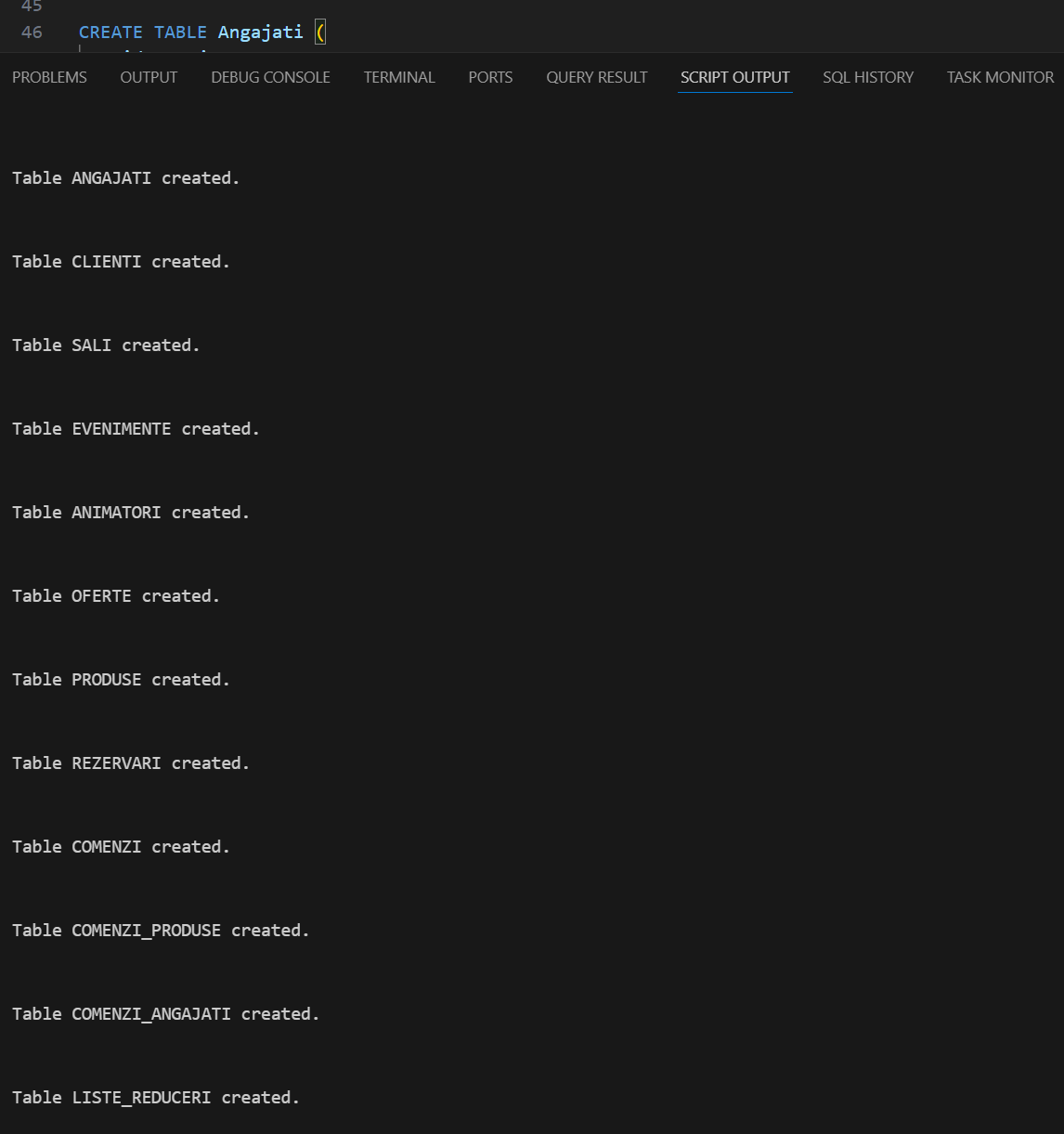
PRIMARY KEY (id\_angajat, id\_eveniment, id\_sala),

CONSTRAINT fk\_po\_angajat FOREIGN KEY (id\_angajat) REFERENCES Angajati(id\_angajat),

CONSTRAINT fk\_po\_eveniment FOREIGN KEY (id\_eveniment) REFERENCES Evenimente(id\_eveniment),

CONSTRAINT fk\_po\_sala FOREIGN KEY (id\_sala) REFERENCES Sali(id\_sala)

);





## Script stergere tabele:

DROP TABLE Planificari\_organizatorice CASCADE CONSTRAINTS;

DROP TABLE Participari\_evenimente CASCADE CONSTRAINTS;

DROP TABLE Colaborari CASCADE CONSTRAINTS;

DROP TABLE Evenimente\_Produse CASCADE CONSTRAINTS;

DROP TABLE Liste\_reduceri CASCADE CONSTRAINTS;

DROP TABLE Comenzi\_Angajati CASCADE CONSTRAINTS;

DROP TABLE Comenzi\_Produse CASCADE CONSTRAINTS;

DROP TABLE Rezervari CASCADE CONSTRAINTS;

DROP TABLE Comenzi CASCADE CONSTRAINTS;

DROP TABLE Animatori CASCADE CONSTRAINTS;

DROP TABLE Oferte CASCADE CONSTRAINTS;

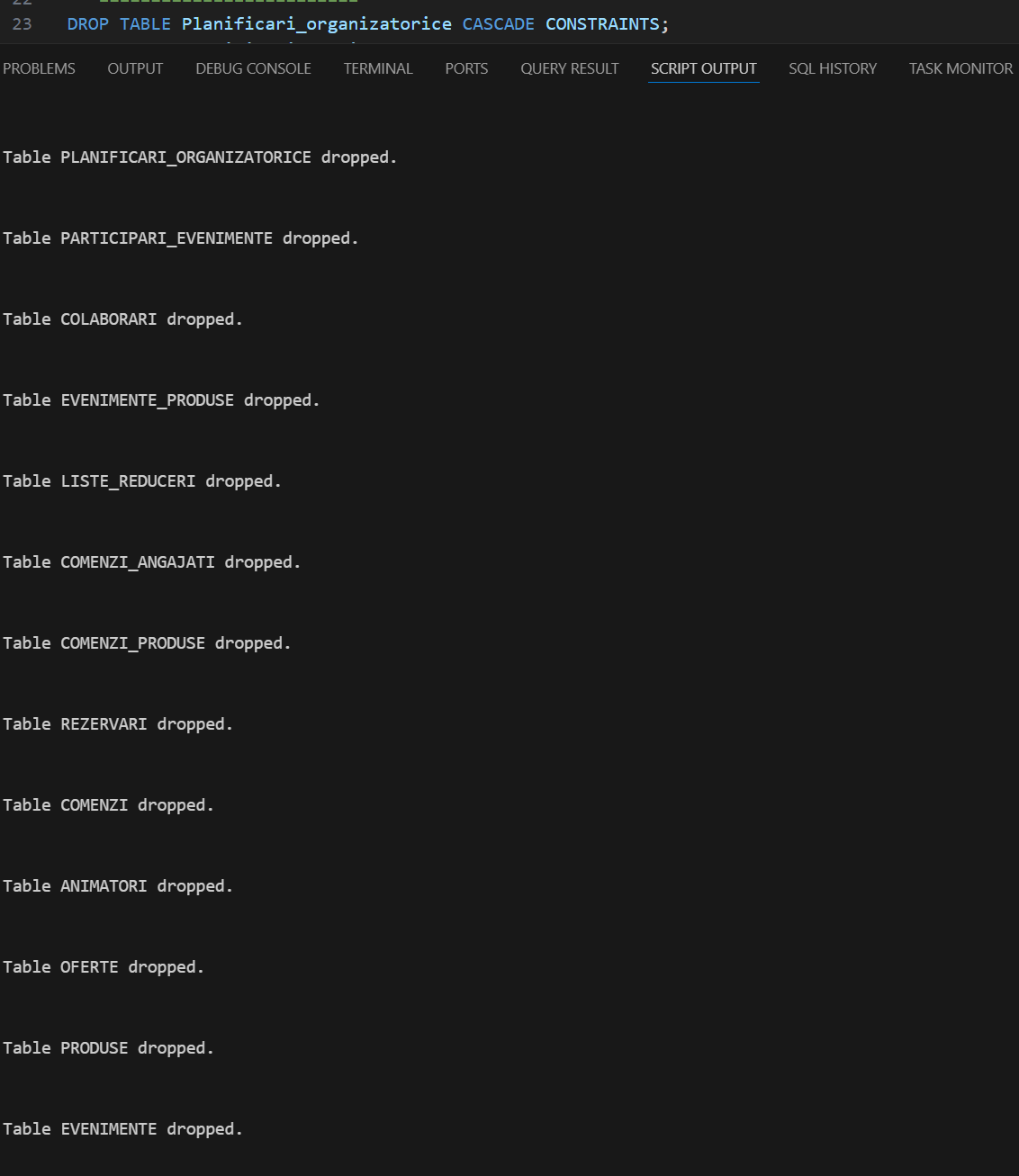
DROP TABLE Produse CASCADE CONSTRAINTS;

DROP TABLE Evenimente CASCADE CONSTRAINTS;

DROP TABLE Sali CASCADE CONSTRAINTS;

DROP TABLE Clienti CASCADE CONSTRAINTS;

DROP TABLE Angajati CASCADE CONSTRAINTS;



# Cerinta 5

-- ============================================================

-- Inserare date coerente - schema ERD simplificata (Oracle)

-- 7 randuri pentru fiecare tabela de baza; 12 pentru asociative

-- ============================================================

SET DEFINE OFF;

-- =========================

-- ANGAJATI (7)

-- =========================

INSERT INTO Angajati (nume, prenume, functie, salariu, data\_angajare, mail) VALUES ('Popescu', 'Andrei', 'Manager', 7500, DATE '2024-03-15', 'andrei.popescu@BellaIta.ro');

INSERT INTO Angajati (nume, prenume, functie, salariu, data\_angajare, mail) VALUES ('Ionescu', 'Maria', 'Chelner', 4200, DATE '2024-06-01', 'maria.ionescu@BellaIta.ro');

INSERT INTO Angajati (nume, prenume, functie, salariu, data\_angajare, mail) VALUES ('Dumitru', 'Vlad', 'Bucatar', 5200, DATE '2023-11-10', 'vlad.dumitru@BellaIta.ro');

INSERT INTO Angajati (nume, prenume, functie, salariu, data\_angajare, mail) VALUES ('Stan', 'Ioana', 'Barman', 4500, DATE '2024-02-20', 'ioana.stan@BellaIta.ro');

INSERT INTO Angajati (nume, prenume, functie, salariu, data\_angajare, mail) VALUES ('Georgescu', 'Radu', 'Host', 3800, DATE '2024-09-05', 'radu.georgescu@BellaIta.ro');

INSERT INTO Angajati (nume, prenume, functie, salariu, data\_angajare, mail) VALUES ('Marin', 'Elena', 'Coordonator evenimente', 6000, DATE '2023-07-12', 'elena.marin@BellaIta.ro');

INSERT INTO Angajati (nume, prenume, functie, salariu, data\_angajare, mail) VALUES ('Toma', 'Mihai', 'Curier', 3600, DATE '2024-01-08', 'mihai.toma@BellaIta.ro');

-- =========================

-- CLIENTI (7)

-- =========================

INSERT INTO Clienti (nume, prenume, nr\_tel, mail) VALUES ('Popa', 'Ana', '0711001100', 'ana.popa@gmail.com');

INSERT INTO Clienti (nume, prenume, nr\_tel, mail) VALUES ('Rusu', 'Bogdan', '0722002200', 'bogdan.rusu@gmail.com');

INSERT INTO Clienti (nume, prenume, nr\_tel, mail) VALUES ('Munteanu', 'Ioana', '0733003300', 'ioana.munteanu@gmail.com');

INSERT INTO Clienti (nume, prenume, nr\_tel, mail) VALUES ('Ilie', 'Cristian', '0744004400', 'cristian.ilie@gmail.com');

INSERT INTO Clienti (nume, prenume, nr\_tel, mail) VALUES ('Sandu', 'Daria', '0755005500', 'daria.sandu@gmail.com');

INSERT INTO Clienti (nume, prenume, nr\_tel, mail) VALUES ('Lazar', 'Paul', '0766006600', 'paul.lazar@gmail.com');

INSERT INTO Clienti (nume, prenume, nr\_tel, mail) VALUES ('Neagu', 'Teodora', '0777007700', 'teodora.neagu@gmail.com');

-- =========================

-- SALI (7)

-- =========================

INSERT INTO Sali (nume\_sala, capacitate) VALUES ('Sala Mare', 200);

INSERT INTO Sali (nume\_sala, capacitate) VALUES ('Sala VIP', 60);

INSERT INTO Sali (nume\_sala, capacitate) VALUES ('Terasa', 120);

INSERT INTO Sali (nume\_sala, capacitate) VALUES ('Salon Rustic', 80);

INSERT INTO Sali (nume\_sala, capacitate) VALUES ('Sala Conferinte', 100);

INSERT INTO Sali (nume\_sala, capacitate) VALUES ('Salon Family', 40);

INSERT INTO Sali (nume\_sala, capacitate) VALUES ('Gradina', 150);

-- =========================

-- EVENIMENTE (7)

-- =========================

INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane) VALUES ('nunta', DATE '2026-02-14', 150);

INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane) VALUES ('botez', DATE '2026-01-25', 60);

INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane) VALUES ('corporate', DATE '2026-03-05', 90);

INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane) VALUES ('aniversare', DATE '2026-01-18', 30);

INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane) VALUES ('petrecere copii', DATE '2026-02-01', 45);

INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane) VALUES ('majorat', DATE '2026-02-21', 70);

INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane) VALUES ('team building', DATE '2026-03-20', 110);

-- =========================

-- ANIMATORI (7)

-- =========================

INSERT INTO Animatori (nume\_animator, tip\_activitate, pret\_ora) VALUES ('Dinu Ion', 'DJ', 300);

INSERT INTO Animatori (nume\_animator, tip\_activitate, pret\_ora) VALUES ('Popescu Marcel', 'Magician', 450);

INSERT INTO Animatori (nume\_animator, tip\_activitate, pret\_ora) VALUES ('Mocanu Alina', 'Animatoare copii', 250);

INSERT INTO Animatori (nume\_animator, tip\_activitate, pret\_ora) VALUES ('Aleandru Tudor', 'Prezentatoare', 350);

INSERT INTO Animatori (nume\_animator, tip\_activitate, pret\_ora) VALUES ('Ionescu Roxana', 'Lider echipa dansatori', 500);

INSERT INTO Animatori (nume\_animator, tip\_activitate, pret\_ora) VALUES ('Ciubotaru Vlad', 'Fotograf', 400);

INSERT INTO Animatori (nume\_animator, tip\_activitate, pret\_ora) VALUES ('Lungu Sorin', 'Lider trupa live', 650);

-- =========================

-- OFERTE (7)

-- =========================

INSERT INTO Oferte (discount, data\_expirare) VALUES (10, DATE '2026-03-31');

INSERT INTO Oferte (discount, data\_expirare) VALUES (15, DATE '2026-02-28');

INSERT INTO Oferte (discount, data\_expirare) VALUES (20, DATE '2026-01-31');

INSERT INTO Oferte (discount, data\_expirare) VALUES (5, DATE '2026-04-30');

INSERT INTO Oferte (discount, data\_expirare) VALUES (12.5, DATE '2026-03-15');

INSERT INTO Oferte (discount, data\_expirare) VALUES (25, DATE '2026-02-15');

INSERT INTO Oferte (discount, data\_expirare) VALUES (8, DATE '2026-05-31');

-- =========================

-- PRODUSE (7)

-- =========================

INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar) VALUES ('Pizza Margherita', 400, 'portie', 32.5);

INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar) VALUES ('Paste Carbonara', 200, 'portie', 38.0);

INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar) VALUES ('Apa plata 0.5L', 500, 'sticla', 8.5);

INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar) VALUES ('Suc natural', 250, 'pahar', 12.0);

INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar) VALUES ('Bere draft', 400, 'pahar', 10.0);

INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar) VALUES ('Desert Tiramisu', 150, 'portie', 22.0);

INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar) VALUES ('Cafea espresso', 40, 'buc', 9.0);

-- =========================

-- REZERVARI (7)

-- =========================

INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala) VALUES (DATE '2026-01-10', '00:00', 4, 1, 6);

INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala) VALUES (DATE '2026-01-11', '20:00', 2, 2, 2);

INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala) VALUES (DATE '2026-01-12', '19:15', 6, 3, 1);

INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala) VALUES (DATE '2026-01-13', '13:00', 3, 4, 4);

INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala) VALUES (DATE '2026-01-14', '21:45', 5, 5, 3);

INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala) VALUES (DATE '2026-01-15', '01:03', 2, 6, 2);

INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala) VALUES (DATE '2026-01-16', '12:30', 8, 7, 7);

-- =========================

-- COMENZI (7)

-- =========================

INSERT INTO Comenzi (data\_comanda, status, id\_client) VALUES (DATE '2026-01-10', 'plasata', 1);

INSERT INTO Comenzi (data\_comanda, status, id\_client) VALUES (DATE '2026-01-10', 'in curs de preparare', 2);

INSERT INTO Comenzi (data\_comanda, status, id\_client) VALUES (DATE '2026-01-11', 'platita', 3);

INSERT INTO Comenzi (data\_comanda, status, id\_client) VALUES (DATE '2026-01-11', 'plasata', 4);

INSERT INTO Comenzi (data\_comanda, status, id\_client) VALUES (DATE '2026-01-12', 'anulata', 5);

INSERT INTO Comenzi (data\_comanda, status, id\_client) VALUES (DATE '2026-01-12', 'platita', 6);

INSERT INTO Comenzi (data\_comanda, status, id\_client) VALUES (DATE '2026-01-13', 'plasata', 7);

-- =========================

-- COMENZI\_PRODUSE (12)

-- =========================

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (1, 1, 2);

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (1, 3, 2);

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (2, 2, 1);

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (2, 5, 2);

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (3, 6, 2);

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (3, 7, 2);

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (4, 1, 1);

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (4, 4, 2);

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (5, 2, 1);

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (5, 3, 1);

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (6, 1, 3);

INSERT INTO Comenzi\_Produse (id\_comanda, id\_produs, cantitate) VALUES (7, 7, 2);

-- =========================

-- COMENZI\_ANGAJATI (12)

-- =========================

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (1, 2);

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (1, 3);

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (2, 2);

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (2, 4);

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (3, 2);

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (3, 3);

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (4, 5);

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (4, 3);

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (5, 2);

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (6, 4);

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (6, 2);

INSERT INTO Comenzi\_Angajati (id\_comanda, id\_angajat) VALUES (7, 7);

-- =========================

-- LISTE\_REDUCERI (12)

-- =========================

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (1, 1);

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (1, 2);

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (2, 3);

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (2, 4);

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (3, 5);

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (3, 6);

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (4, 7);

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (5, 1);

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (5, 6);

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (6, 2);

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (6, 5);

INSERT INTO Liste\_reduceri (id\_oferta, id\_produs) VALUES (7, 3);

-- =========================

-- EVENIMENTE\_PRODUSE (12)

-- =========================

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (1, 3, 150);

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (1, 5, 120);

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (2, 3, 60);

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (2, 6, 40);

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (3, 3, 90);

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (3, 4, 80);

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (4, 1, 20);

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (4, 7, 25);

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (5, 4, 60);

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (6, 5, 90);

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (6, 6, 50);

INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (7, 3, 110);

-- =========================

-- COLABORARI (12)

-- =========================

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (1, 1, 6);

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (1, 4, 6);

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (2, 6, 4);

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (3, 7, 5);

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (4, 2, 2);

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (5, 3, 4);

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (6, 1, 5);

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (6, 5, 3);

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (7, 4, 4);

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (3, 6, 4);

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (2, 3, 3);

INSERT INTO Colaborari (id\_eveniment, id\_animator, numar\_ore) VALUES (7, 7, 6);

-- =========================

-- PARTICIPARI\_EVENIMENTE (12)

-- =========================

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (1, 1);

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (1, 2);

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (2, 3);

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (2, 4);

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (3, 5);

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (3, 6);

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (4, 7);

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (5, 1);

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (5, 3);

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (6, 2);

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (6, 4);

INSERT INTO Participari\_evenimente (id\_eveniment, id\_client) VALUES (7, 6);

-- =========================

-- PLANIFICARI\_ORGANIZATORICE (12)

-- =========================

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (2, 1, 1, 'servire sala mare');

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (3, 1, 1, 'coordonare meniu');

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (6, 1, 1, 'organizare generala');

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (5, 2, 2, 'primire invitati');

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (2, 3, 5, 'servire coffee break');

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (6, 3, 5, 'logistica eveniment');

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (4, 4, 4, 'bar eveniment');

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (3, 5, 6, 'meniu copii');

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (6, 5, 6, 'management activitati');

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (2, 6, 3, 'servire terasa');

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (4, 6, 3, 'bar terasa');

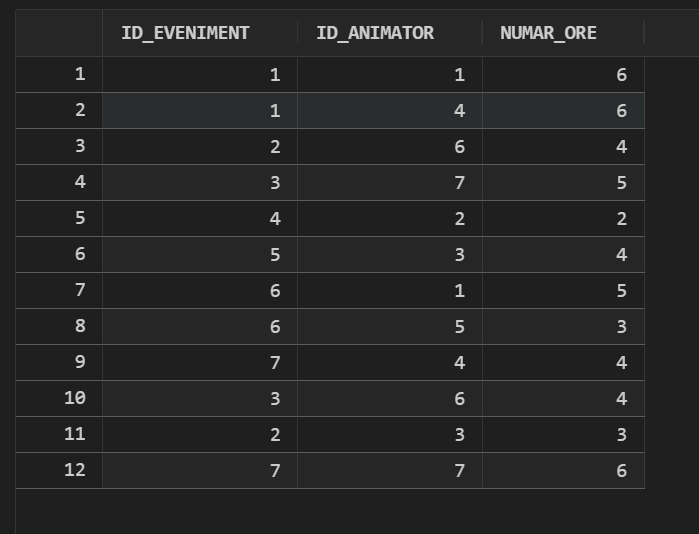
INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (6, 7, 7, 'organizare outdoor');

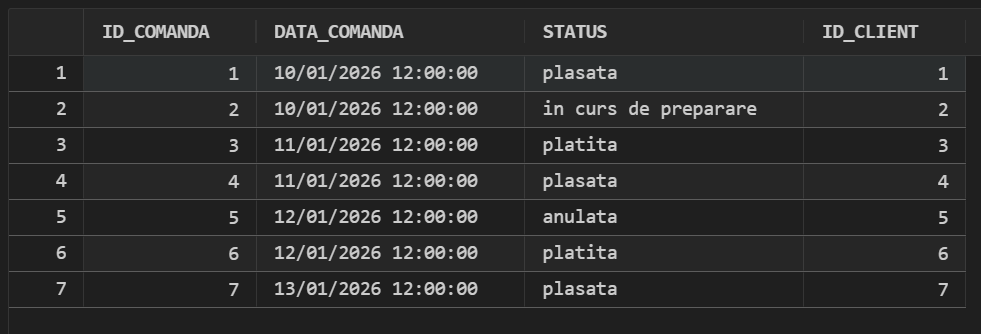
COMMIT;

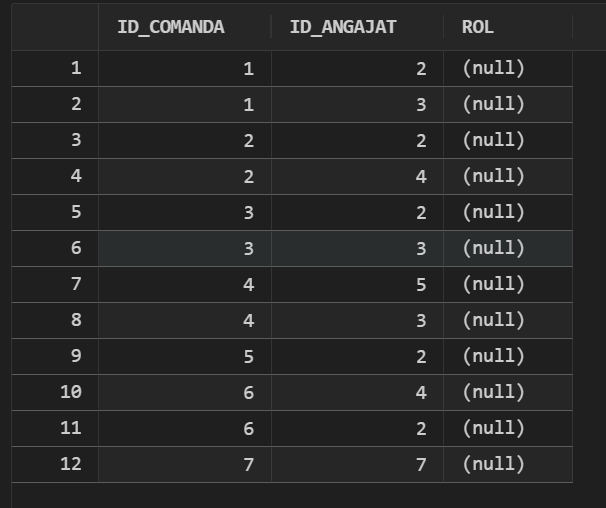
## Print-screen:

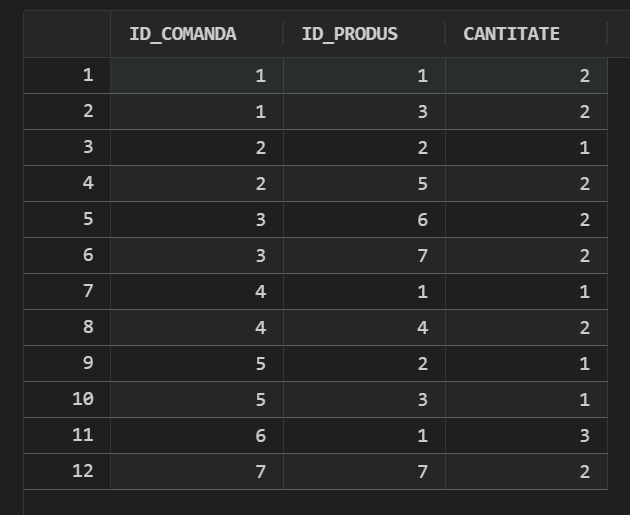




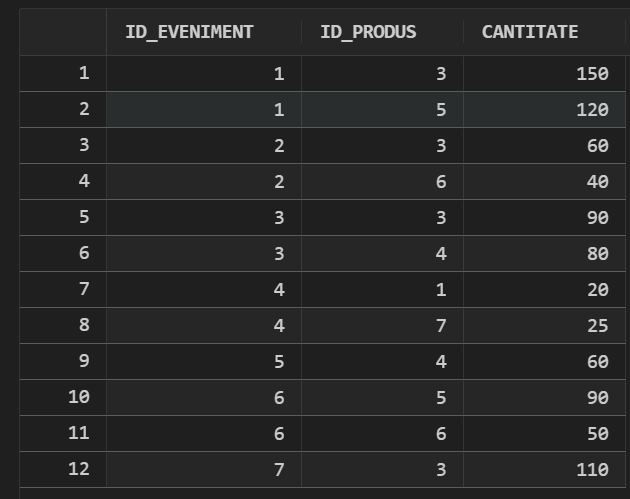


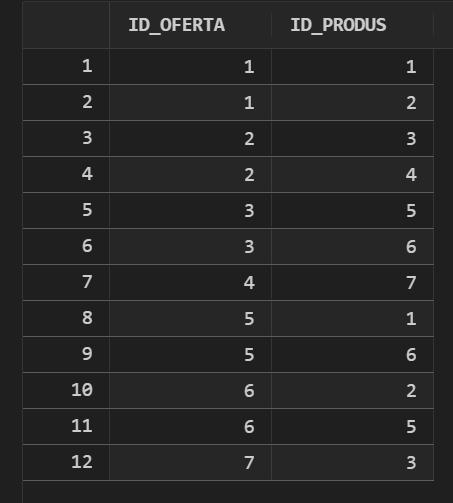






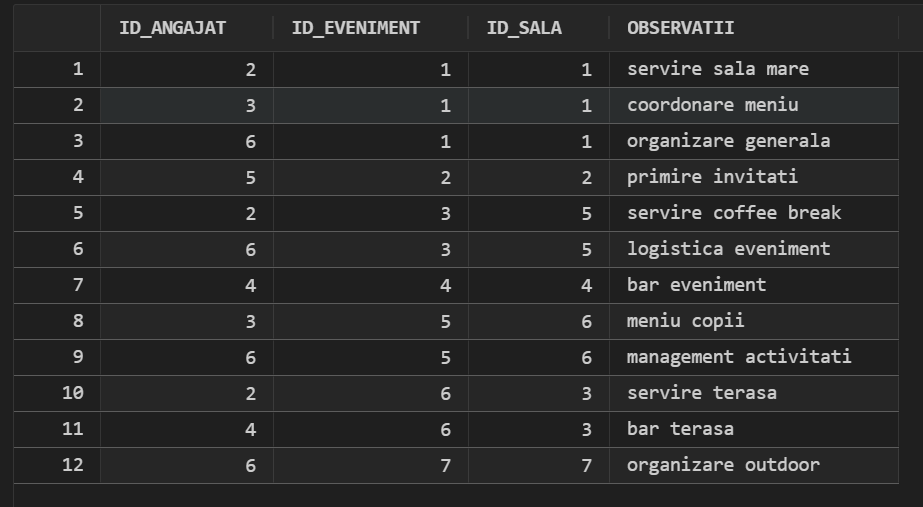


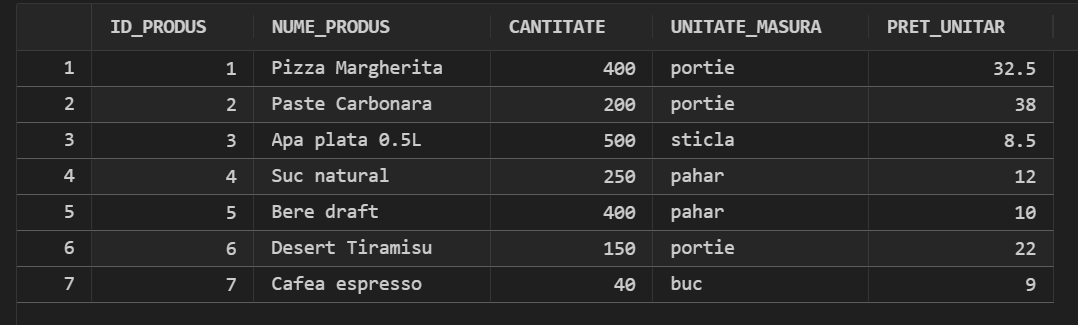














# Cerinta 6

## Cerinta:

CERINTA 6 (Proiect SGBD 2025-2026)

Subprogram stocat independent care utilizeaza toate cele 3 tipuri de colectii:

  (1) VARRAY

  (2) NESTED TABLE

  (3) ASSOCIATIVE ARRAY (INDEX BY)

Pentru un eveniment dat (id\_eveniment), sa se genereze un raport logistic care:

 - afiseaza detalii despre eveniment (tip, data, numar\_persoane);

 - calculeaza capacitatea totala a salilor alocate evenimentului;

 - listeaza produsele necesare evenimentului (cantitati) si verifica daca stocul este suficient;

- calculeaza costul estimat pentru produsele necesare;

- afiseaza alerte (ex: stoc insuficient, capacitate insuficienta, lipsa planificare).

## Rezolvare:

SET SERVEROUTPUT ON;

-- DROP TYPE t\_alerte FORCE

-- DROP TYPE t\_lista\_produse FORCE

-- (1) VARRAY: lista fixa de alerte

CREATE OR REPLACE TYPE t\_alerte IS VARRAY(10) OF VARCHAR2(200);

/

-- (2) NESTED TABLE: lista de id-uri de produse pentru eveniment

CREATE OR REPLACE TYPE t\_lista\_produse IS TABLE OF NUMBER;

/

CREATE OR REPLACE PROCEDURE raport\_logistic\_eveniment(

    p\_id\_eveniment IN NUMBER

) AS

    -- (3) ASSOCIATIVE ARRAY: mapare id\_produs -> cantitate necesara

    TYPE t\_necesar\_map IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

    v\_alerte        t\_alerte := t\_alerte();

    v\_id\_produse    t\_lista\_produse := t\_lista\_produse();

    v\_necesar       t\_necesar\_map;

    v\_tip\_eveniment  Evenimente.tip\_eveniment%TYPE;

    v\_data\_eveniment Evenimente.data\_eveniment%TYPE;

    v\_nr\_pers        Evenimente.numar\_persoane%TYPE;

    v\_cap\_totala     NUMBER := 0;

    v\_participanti   NUMBER := 0;

    v\_cost\_produse   NUMBER := 0;

    -- v\_cost\_animatori NUMBER := 0;

    PROCEDURE add\_alert(p\_msg IN VARCHAR2) IS

    BEGIN

        IF v\_alerte.COUNT < v\_alerte.LIMIT THEN

            v\_alerte.EXTEND;

            v\_alerte(v\_alerte.COUNT) := p\_msg;

        END IF;

    END;

BEGIN

    -- 1) Detalii eveniment

    BEGIN

        SELECT tip\_eveniment, data\_eveniment, numar\_persoane

        INTO   v\_tip\_eveniment, v\_data\_eveniment, v\_nr\_pers

        FROM   Evenimente

        WHERE  id\_eveniment = p\_id\_eveniment;

    EXCEPTION

        WHEN NO\_DATA\_FOUND THEN

            DBMS\_OUTPUT.PUT\_LINE('Nu exista eveniment cu id\_eveniment = ' || p\_id\_eveniment);

            RETURN;

    END;

    DBMS\_OUTPUT.PUT\_LINE('============================================================');

    DBMS\_OUTPUT.PUT\_LINE('RAPORT LOGISTIC - Eveniment #' || p\_id\_eveniment);

    DBMS\_OUTPUT.PUT\_LINE('Tip: ' || v\_tip\_eveniment || ' | Data: ' || TO\_CHAR(v\_data\_eveniment, 'YYYY-MM-DD') ||

                         ' | Nr. persoane: ' || v\_nr\_pers);

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

    -- 2) Capacitate totala a salilor alocate (distinct)

    SELECT NVL(SUM(s.capacitate), 0)

    INTO   v\_cap\_totala

    FROM   Sali s

    WHERE  s.id\_sala IN (

        SELECT DISTINCT po.id\_sala

        FROM Planificari\_organizatorice po

        WHERE po.id\_eveniment = p\_id\_eveniment

    );

    IF v\_cap\_totala = 0 THEN

        add\_alert('Nu exista sali alocate (Planificari\_organizatorice) pentru acest eveniment.');

    ELSIF v\_nr\_pers > v\_cap\_totala THEN

        add\_alert('Capacitate insuficienta: ' || v\_cap\_totala || ' locuri pentru ' || v\_nr\_pers || ' persoane.');

    END IF;

    -- 3) Numar participanti inscrisi (din participari)

    SELECT COUNT(\*)

    INTO   v\_participanti

    FROM   Participari\_evenimente

    WHERE  id\_eveniment = p\_id\_eveniment;

    DBMS\_OUTPUT.PUT\_LINE('Capacitate sali (total): ' || v\_cap\_totala);

    DBMS\_OUTPUT.PUT\_LINE('Participanti inregistrati: ' || v\_participanti);

    -- 5) Colectare produse necesare (NESTED TABLE) + mapare cantitati (ASSOCIATIVE)

    SELECT ep.id\_produs

    BULK COLLECT INTO v\_id\_produse

    FROM Evenimente\_Produse ep

    WHERE ep.id\_eveniment = p\_id\_eveniment

    ORDER BY ep.id\_produs;

    FOR ep IN (

        SELECT id\_produs, cantitate

        FROM Evenimente\_Produse

        WHERE id\_eveniment = p\_id\_eveniment

    ) LOOP

        v\_necesar(ep.id\_produs) := ep.cantitate;

    END LOOP;

    IF v\_id\_produse.COUNT = 0 THEN

        add\_alert('Nu exista produse asociate evenimentului (Evenimente\_Produse).');

    END IF;

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

    DBMS\_OUTPUT.PUT\_LINE('Produse necesare (necesar vs stoc):');

    -- Folosim NESTED TABLE in SQL: TABLE(v\_id\_produse)

    FOR p IN (

        SELECT p2.id\_produs, p2.nume\_produs, p2.cantitate AS stoc,

               p2.unitate\_masura, p2.pret\_unitar

        FROM Produse p2

        WHERE p2.id\_produs IN (SELECT COLUMN\_VALUE FROM TABLE(v\_id\_produse))

        ORDER BY p2.id\_produs

    ) LOOP

        DECLARE

            v\_req NUMBER;

        BEGIN

            v\_req := NVL(v\_necesar(p.id\_produs), 0);

            v\_cost\_produse := v\_cost\_produse + (v\_req \* p.pret\_unitar);

            DBMS\_OUTPUT.PUT\_LINE(' - #' || p.id\_produs || ' ' || RPAD(p.nume\_produs, 20) ||

                                 ' necesar=' || TO\_CHAR(v\_req) || ' ' || p.unitate\_masura ||

                                 ' | stoc=' || TO\_CHAR(p.stoc) || ' ' || p.unitate\_masura ||

                                 ' | pret\_unitar=' || TO\_CHAR(p.pret\_unitar));

            IF v\_req > p.stoc THEN

                add\_alert('Stoc insuficient pentru produsul "' || p.nume\_produs || '": necesar ' || v\_req ||

                          ', stoc ' || p.stoc || '.');

            END IF;

        END;

    END LOOP;

    DBMS\_OUTPUT.PUT\_LINE('Cost estimat produse (necesar \* pret\_unitar): ' || TO\_CHAR(v\_cost\_produse, '9999990D00'));

    -- 7) Afisare alerte (VARRAY)

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

    IF v\_alerte.COUNT = 0 THEN

        DBMS\_OUTPUT.PUT\_LINE('Alerte: (niciuna)');

    ELSE

        DBMS\_OUTPUT.PUT\_LINE('Alerte (' || v\_alerte.COUNT || '):');

        FOR i IN 1 .. v\_alerte.COUNT LOOP

            DBMS\_OUTPUT.PUT\_LINE(' ! ' || v\_alerte(i));

        END LOOP;

    END IF;

    DBMS\_OUTPUT.PUT\_LINE('============================================================');

END;

/

-- ============================================================

-- APEL (exemplu) - foloseste id\_eveniment = 1 (conform setului de date 7/12)

-- ============================================================

BEGIN

    raport\_logistic\_eveniment(7);

END;

/

SET SERVEROUTPUT ON;

DECLARE

  v\_evt\_fara\_tot     NUMBER;

  v\_evt\_cap\_stoc     NUMBER;

  v\_sala\_mica        NUMBER;

  v\_ang              NUMBER;

  v\_prod             NUMBER;

  v\_mail             VARCHAR2(200);

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('================= TEST PUNCTUL 6 =================');

  SAVEPOINT sp6;

  -- Caz 1: eveniment inexistent (NO\_DATA\_FOUND tratat in procedura)

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 1: eveniment inexistent ---');

  raport\_logistic\_eveniment(-1);

  raport\_logistic\_eveniment(NULL);

  -- Caz 2: eveniment existent, dar fara sali + fara produse (alerte)

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 2: fara sali + fara produse ---');

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES ('TEST\_P6\_FARA', DATE '2099-01-01', 10)

  RETURNING id\_eveniment INTO v\_evt\_fara\_tot;

  raport\_logistic\_eveniment(v\_evt\_fara\_tot);

  -- Caz 3: capacitate insuficienta + stoc insuficient (alerte)

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 3: capacitate insuficienta + stoc insuficient ---');

  INSERT INTO Sali (nume\_sala, capacitate)

  VALUES ('SALA\_TEST\_P6', 5)

  RETURNING id\_sala INTO v\_sala\_mica;

  v\_mail := 'p6\_' || TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3') || '@ex.com';

  INSERT INTO Angajati (nume, prenume, functie, salariu, mail)

  VALUES ('Test', 'P6', 'Chelner', 3000, v\_mail)

  RETURNING id\_angajat INTO v\_ang;

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES ('TEST\_P6\_CAP\_STOC', DATE '2099-01-02', 50)

  RETURNING id\_eveniment INTO v\_evt\_cap\_stoc;

  -- (daca la tine PK la planificari e (id\_angajat,id\_eveniment) e ok; daca e pe 3 coloane, e tot ok)

  INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii)

  VALUES (v\_ang, v\_evt\_cap\_stoc, v\_sala\_mica, 'test capacitate');

  INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar)

  VALUES ('PROD\_TEST\_P6', 1, 'buc', 10)

  RETURNING id\_produs INTO v\_prod;

  INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate)

  VALUES (v\_evt\_cap\_stoc, v\_prod, 10);

  raport\_logistic\_eveniment(v\_evt\_cap\_stoc);

  ROLLBACK TO sp6;

  DBMS\_OUTPUT.PUT\_LINE('===================================================');

END;

/

SET SERVEROUTPUT ON;

DECLARE

  v\_evt\_test NUMBER;

  v\_tip      Evenimente.tip\_eveniment%TYPE;

  PROCEDURE safe\_exec(p\_sql VARCHAR2) IS

  BEGIN

    EXECUTE IMMEDIATE p\_sql;

  EXCEPTION

    WHEN OTHERS THEN

      DBMS\_OUTPUT.PUT\_LINE('   [WARN] ' || p\_sql || ' -> ' || SQLERRM);

  END;

  PROCEDURE safe\_count(p\_table VARCHAR2) IS

    v\_cnt NUMBER;

  BEGIN

    EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM ' || p\_table INTO v\_cnt;

    DBMS\_OUTPUT.PUT\_LINE('   COUNT(' || p\_table || ') = ' || v\_cnt);

  EXCEPTION

    WHEN OTHERS THEN

      DBMS\_OUTPUT.PUT\_LINE('   COUNT(' || p\_table || ') = ? -> ' || SQLERRM);

  END;

  PROCEDURE try\_p6(p\_evt NUMBER) IS

  BEGIN

    DBMS\_OUTPUT.PUT\_LINE('   Apel: raport\_logistic\_eveniment(' || p\_evt || ')');

    raport\_logistic\_eveniment(p\_evt);

    DBMS\_OUTPUT.PUT\_LINE('   -> OK (nu a aruncat exceptie ne-prinsa)');

  EXCEPTION

    WHEN OTHERS THEN

      DBMS\_OUTPUT.PUT\_LINE('   -> EROARE NEPRINSA: ' || SQLERRM);

  END;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('================= TEST P6: tabele goale =================');

  SAVEPOINT sp6;

  -- eveniment de test (ca sa putem apela in majoritatea subtestelor)

  BEGIN

    SELECT tip\_eveniment INTO v\_tip FROM Evenimente WHERE ROWNUM = 1;

  EXCEPTION WHEN NO\_DATA\_FOUND THEN

    v\_tip := 'TEST';

  END;

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES (v\_tip, SYSDATE + 10, 10)

  RETURNING id\_eveniment INTO v\_evt\_test;

  -- (A) EVENIMENTE gol

  DBMS\_OUTPUT.PUT\_LINE('--- (A) EVENIMENTE gol ---');

  SAVEPOINT tA;

  safe\_exec('DELETE FROM Colaborari');

  safe\_exec('DELETE FROM Evenimente\_Produse');

  safe\_exec('DELETE FROM Participari\_evenimente');

  safe\_exec('DELETE FROM Planificari\_organizatorice');

  safe\_exec('DELETE FROM Evenimente');

  safe\_count('Evenimente');

  try\_p6(1);

  ROLLBACK TO tA;

  -- (B) PLANIFICARI\_ORGANIZATORICE gol

  DBMS\_OUTPUT.PUT\_LINE('--- (B) PLANIFICARI\_ORGANIZATORICE gol ---');

  SAVEPOINT tB;

  safe\_exec('DELETE FROM Planificari\_organizatorice');

  safe\_count('Planificari\_organizatorice');

  try\_p6(v\_evt\_test);

  ROLLBACK TO tB;

  -- (C) SALI gol (sterge intai dependente uzuale)

  DBMS\_OUTPUT.PUT\_LINE('--- (C) SALI gol ---');

  SAVEPOINT tC;

  safe\_exec('DELETE FROM Planificari\_organizatorice');

  safe\_exec('DELETE FROM Rezervari'); -- ca sa poti goli SALI daca Rezervari refera SALI

  safe\_exec('DELETE FROM Sali');

  safe\_count('Sali');

  try\_p6(v\_evt\_test);

  ROLLBACK TO tC;

  -- (D) PARTICIPARI\_EVENIMENTE gol

  DBMS\_OUTPUT.PUT\_LINE('--- (D) PARTICIPARI\_EVENIMENTE gol ---');

  SAVEPOINT tD;

  safe\_exec('DELETE FROM Participari\_evenimente');

  safe\_count('Participari\_evenimente');

  try\_p6(v\_evt\_test);

  ROLLBACK TO tD;

  -- (E) EVENIMENTE\_PRODUSE gol

  DBMS\_OUTPUT.PUT\_LINE('--- (E) EVENIMENTE\_PRODUSE gol ---');

  SAVEPOINT tE;

  safe\_exec('DELETE FROM Evenimente\_Produse');

  safe\_count('Evenimente\_Produse');

  try\_p6(v\_evt\_test);

  ROLLBACK TO tE;

  -- (F) PRODUSE gol (trebuie golit Evenimente\_Produse intai) - daca e null

  DBMS\_OUTPUT.PUT\_LINE('--- (F) PRODUSE gol ---');

  SAVEPOINT tF;

  safe\_exec('DELETE FROM Evenimente\_Produse');

  safe\_exec('DELETE FROM Liste\_reduceri');

  safe\_exec('DELETE FROM Comenzi\_Produse');

  safe\_exec('DELETE FROM Produse');

  safe\_count('Produse');

  try\_p6(v\_evt\_test);

  ROLLBACK TO tF;

  -- (G) ANGAJATI gol (trebuie golit Planificari\_organizatorice intai)

  DBMS\_OUTPUT.PUT\_LINE('--- (G) ANGAJATI gol ---');

  SAVEPOINT tG;

  safe\_exec('DELETE FROM Planificari\_organizatorice');

  safe\_exec('DELETE FROM Comenzi\_Angajati'); -- daca exista in schema ta

  safe\_exec('DELETE FROM Angajati');

  safe\_count('Angajati');

  try\_p6(v\_evt\_test);

  ROLLBACK TO tG;

  ROLLBACK TO sp6;

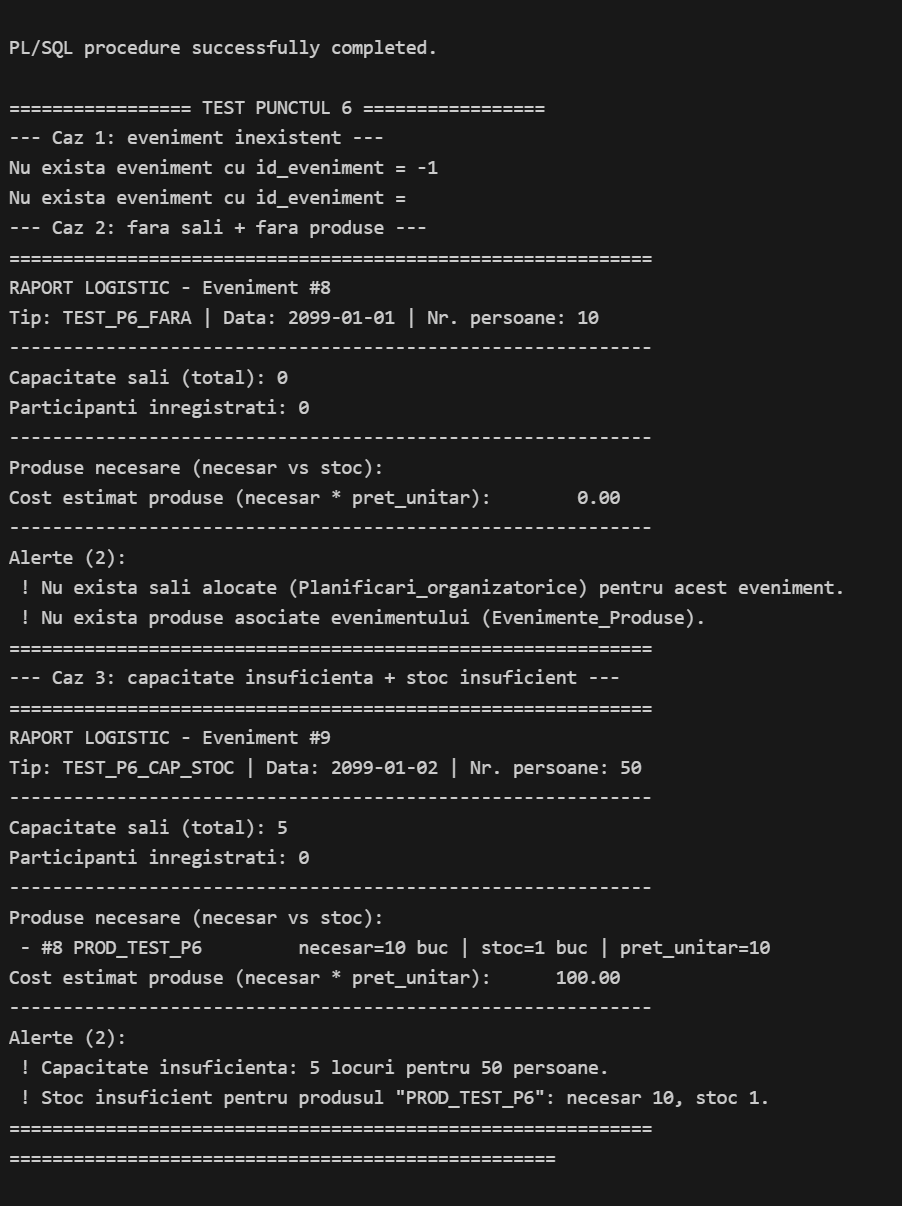
  DBMS\_OUTPUT.PUT\_LINE('===========================================================');

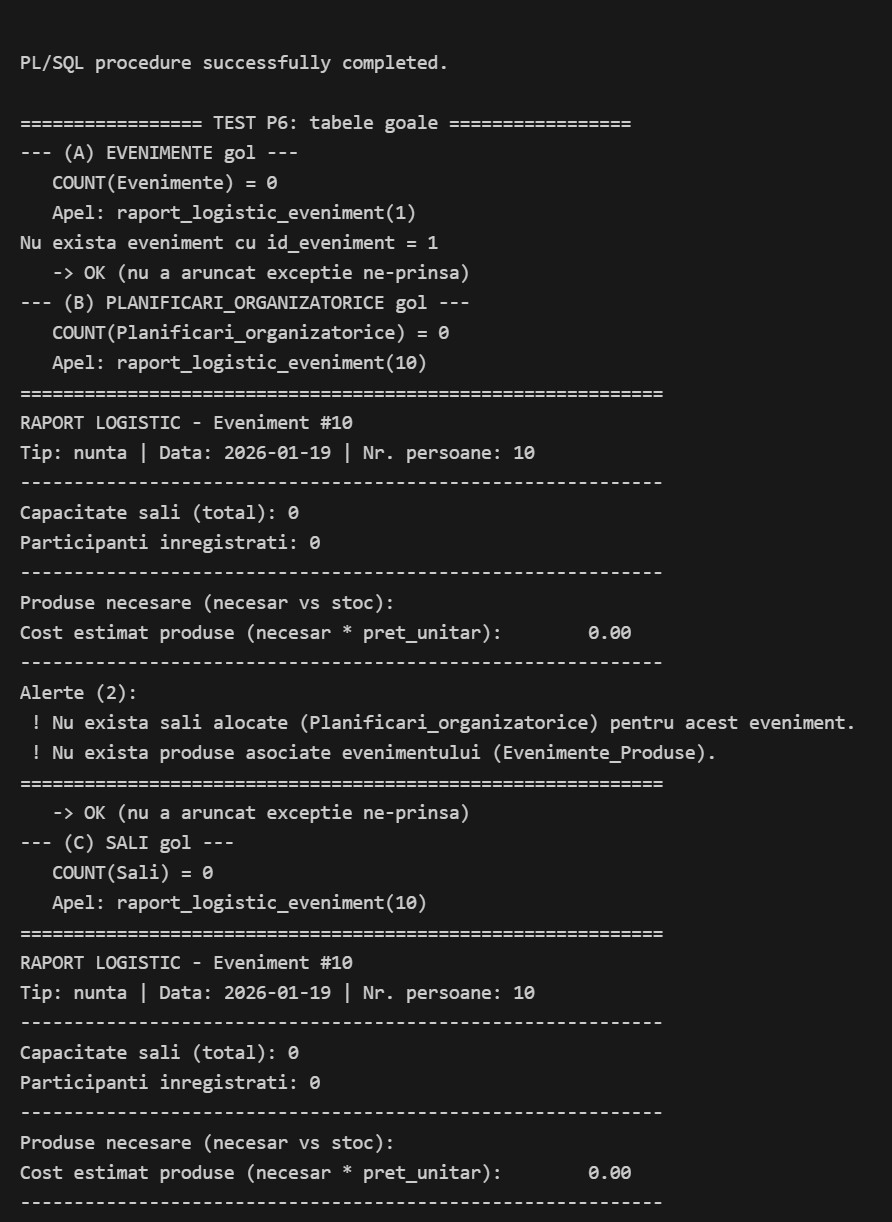
END;

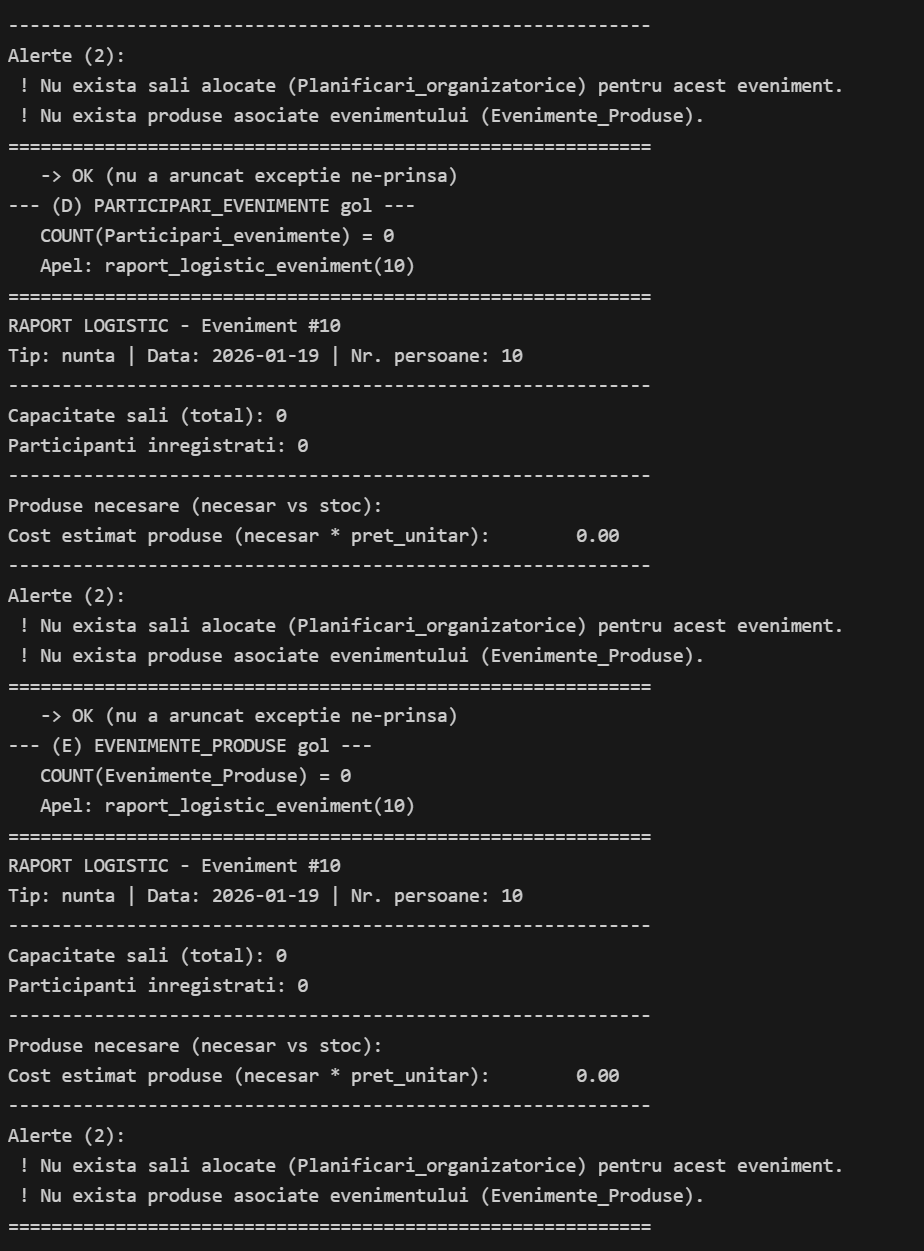
/

## Print-screen:

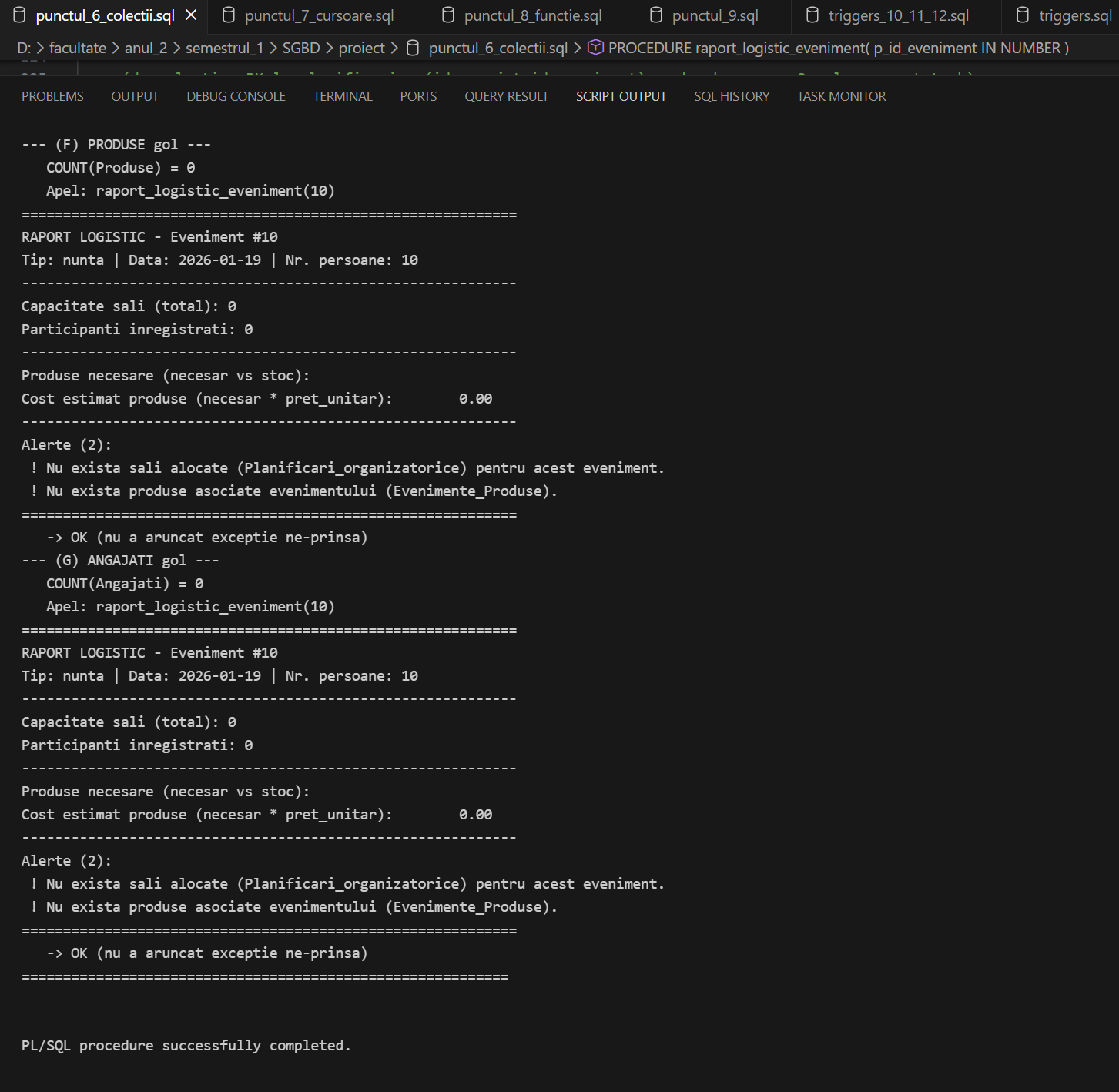












# Cerinta 7

## Cerinta:

CERINTA 7 (Proiect SGBD 2025-2026)

Pentru o perioada data, afisam lista evenimentelor si, pentru fiecare eveniment, salile alocate si numarul de angajati planificati in fiecare sala.

Se folosesc 2 tipuri de cursoare:

    (1) cursor explicit cu OPEN/FETCH/CLOSE (pentru evenimente)

  (2) cursor parametrizat, dependent de primul (pentru salile evenimentului), parcurs cu cursor FOR loop.

## **Rezolvare:**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE raport\_evenimente\_sali(

    p\_data\_min IN DATE,

    p\_data\_max IN DATE

) AS

    -- Cursor 1: explicit (OPEN/FETCH/CLOSE)

    CURSOR c\_evenimente IS

        SELECT id\_eveniment, tip\_eveniment, data\_eveniment, numar\_persoane

        FROM   Evenimente

        WHERE  data\_eveniment BETWEEN p\_data\_min AND p\_data\_max

        ORDER  BY data\_eveniment, id\_eveniment;

    v\_evt c\_evenimente%ROWTYPE;

    v\_gasit BOOLEAN := FALSE;

    -- Cursor 2: parametrizat (dependent de cursorul 1)

    CURSOR c\_sali(p\_id\_eveniment NUMBER) IS

        SELECT s.id\_sala,

               s.nume\_sala,

               s.capacitate,

               COUNT(DISTINCT po.id\_angajat) AS nr\_angajati

        FROM   Planificari\_organizatorice po

               JOIN Sali s ON s.id\_sala = po.id\_sala

        WHERE  po.id\_eveniment = p\_id\_eveniment

        GROUP  BY s.id\_sala, s.nume\_sala, s.capacitate

        ORDER  BY s.nume\_sala;

    v\_participanti NUMBER;

    v\_cap\_total    NUMBER;

    v\_are\_sali     BOOLEAN;

BEGIN

    DBMS\_OUTPUT.PUT\_LINE('============================================================');

    DBMS\_OUTPUT.PUT\_LINE('RAPORT EVENIMENTE -> SALI + ANGAJATI');

    DBMS\_OUTPUT.PUT\_LINE('Perioada: ' || TO\_CHAR(p\_data\_min,'YYYY-MM-DD') || ' .. ' || TO\_CHAR(p\_data\_max,'YYYY-MM-DD'));

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

    OPEN c\_evenimente;

    LOOP

        FETCH c\_evenimente INTO v\_evt;

        EXIT WHEN c\_evenimente%NOTFOUND;

        v\_gasit := TRUE;

        -- participanti inscrisi (0 daca nu exista)

        SELECT COUNT(\*)

        INTO   v\_participanti

        FROM   Participari\_evenimente

        WHERE  id\_eveniment = v\_evt.id\_eveniment;

        DBMS\_OUTPUT.PUT\_LINE('Eveniment #' || v\_evt.id\_eveniment ||

                             ' | ' || v\_evt.tip\_eveniment ||

                             ' | ' || TO\_CHAR(v\_evt.data\_eveniment,'YYYY-MM-DD') ||

                             ' | persoane=' || v\_evt.numar\_persoane ||

                             ' | participanti=' || v\_participanti);

        DBMS\_OUTPUT.PUT\_LINE('  Sali alocate:');

        v\_cap\_total := 0;

        v\_are\_sali  := FALSE;

        -- Cursor parametrizat dependent de evenimentul curent

        FOR r IN c\_sali(v\_evt.id\_eveniment) LOOP

            v\_are\_sali  := TRUE;

            v\_cap\_total := v\_cap\_total + r.capacitate;

            DBMS\_OUTPUT.PUT\_LINE('   - ' || RPAD(r.nume\_sala, 18) ||

                                 ' (cap=' || r.capacitate || ')' ||

                                 ' | angajati\_planificati=' || r.nr\_angajati);

        END LOOP;

        IF NOT v\_are\_sali THEN

            DBMS\_OUTPUT.PUT\_LINE('   (nu exista sali planificate pentru acest eveniment)');

        ELSE

            DBMS\_OUTPUT.PUT\_LINE('  Capacitate totala sali: ' || v\_cap\_total);

            IF v\_evt.numar\_persoane > v\_cap\_total THEN

                DBMS\_OUTPUT.PUT\_LINE('  ATENTIE: capacitate insuficienta pentru numarul de persoane!');

            END IF;

        END IF;

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

    END LOOP;

    CLOSE c\_evenimente;

    IF NOT v\_gasit THEN

        DBMS\_OUTPUT.PUT\_LINE('Nu exista evenimente in perioada selectata.');

    END IF;

    DBMS\_OUTPUT.PUT\_LINE('============================================================');

END;

/

SHOW ERRORS;

-- ============================================================

-- APEL (exemplu): interval foarte larg ca sa prinda toate datele inserate

-- ============================================================

BEGIN

    raport\_evenimente\_sali(DATE '1900-01-01', DATE '2999-12-31');

END;

/

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (5, 1, 2, 'servire sala mare');

INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii) VALUES (4, 1, 2, 'servire sala mare');

COMMIT;

-- DELETE FROM PLANIFICARI\_ORGANIZATORICE po

-- WHERE po.ID\_EVENIMENT = 1 AND (po.ID\_ANGAJAT = 5 OR po.ID\_ANGAJAT = 4);

SET SERVEROUTPUT ON;

DECLARE

  v\_evt   NUMBER;

  v\_sala  NUMBER;

  v\_ang   NUMBER;

  v\_mail  VARCHAR2(200);

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('================= TEST PUNCTUL 7 =================');

  SAVEPOINT sp7;

  -- Caz 1: perioada fara evenimente (mesajul "Nu exista evenimente...")

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 1: perioada fara evenimente ---');

  raport\_evenimente\_sali(DATE '1800-01-01', DATE '1800-12-31');

  raport\_evenimente\_sali(NULL, NULL);

  -- Caz 2: perioada cu evenimente (datele existente)

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 2: perioada cu evenimente ---');

  raport\_evenimente\_sali(DATE '1900-01-01', DATE '2999-12-31');

  -- Caz 3: avertizare "capacitate insuficienta"

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 3: capacitate insuficienta (eveniment de test) ---');

  INSERT INTO Sali (nume\_sala, capacitate)

  VALUES ('SALA\_TEST\_P7', 5)

  RETURNING id\_sala INTO v\_sala;

  v\_mail := 'p7\_' || TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3') || '@ex.com';

  INSERT INTO Angajati (nume, prenume, functie, salariu, mail)

  VALUES ('Test', 'P7', 'Chelner', 3000, v\_mail)

  RETURNING id\_angajat INTO v\_ang;

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES ('TEST\_P7', DATE '2099-02-01', 50)

  RETURNING id\_eveniment INTO v\_evt;

  INSERT INTO Planificari\_organizatorice (id\_angajat, id\_eveniment, id\_sala, observatii)

  VALUES (v\_ang, v\_evt, v\_sala, 'test capacitate');

  raport\_evenimente\_sali(DATE '2099-02-01', DATE '2099-02-01');

  ROLLBACK TO sp7;

  DBMS\_OUTPUT.PUT\_LINE('===================================================');

END;

/

SET SERVEROUTPUT ON;

DECLARE

  PROCEDURE safe\_exec(p\_sql VARCHAR2) IS

  BEGIN

    EXECUTE IMMEDIATE p\_sql;

  EXCEPTION WHEN OTHERS THEN

    DBMS\_OUTPUT.PUT\_LINE('   [WARN] ' || p\_sql || ' -> ' || SQLERRM);

  END;

  PROCEDURE safe\_count(p\_table VARCHAR2) IS

    v\_cnt NUMBER;

  BEGIN

    EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM ' || p\_table INTO v\_cnt;

    DBMS\_OUTPUT.PUT\_LINE('   COUNT(' || p\_table || ') = ' || v\_cnt);

  EXCEPTION WHEN OTHERS THEN

    DBMS\_OUTPUT.PUT\_LINE('   COUNT(' || p\_table || ') = ? -> ' || SQLERRM);

  END;

  PROCEDURE try\_p7 IS

  BEGIN

    DBMS\_OUTPUT.PUT\_LINE('   Apel: raport\_evenimente\_sali(1900-01-01, 2999-12-31)');

    raport\_evenimente\_sali(DATE '1900-01-01', DATE '2999-12-31');

    DBMS\_OUTPUT.PUT\_LINE('   -> OK (nu a aruncat exceptie ne-prinsa)');

  EXCEPTION WHEN OTHERS THEN

    DBMS\_OUTPUT.PUT\_LINE('   -> EROARE NEPRINSA: ' || SQLERRM);

  END;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('================= TEST P7: tabele goale =================');

  SAVEPOINT sp7;

  -- (A) EVENIMENTE gol (stergem dependente uzuale)

  DBMS\_OUTPUT.PUT\_LINE('--- (A) EVENIMENTE gol ---');

  SAVEPOINT tA;

  safe\_exec('DELETE FROM Participari\_evenimente');

  safe\_exec('DELETE FROM Planificari\_organizatorice');

  safe\_exec('DELETE FROM Colaborari');

  safe\_exec('DELETE FROM Evenimente\_produse');

  safe\_exec('DELETE FROM Evenimente');

  safe\_count('Evenimente');

  try\_p7;

  ROLLBACK TO tA;

  -- (B) PARTICIPARI\_EVENIMENTE gol

  DBMS\_OUTPUT.PUT\_LINE('--- (B) PARTICIPARI\_EVENIMENTE gol ---');

  SAVEPOINT tB;

  safe\_exec('DELETE FROM Participari\_evenimente');

  safe\_count('Participari\_evenimente');

  try\_p7;

  ROLLBACK TO tB;

  -- (C) PLANIFICARI\_ORGANIZATORICE gol

  DBMS\_OUTPUT.PUT\_LINE('--- (C) PLANIFICARI\_ORGANIZATORICE gol ---');

  SAVEPOINT tC;

  safe\_exec('DELETE FROM Planificari\_organizatorice');

  safe\_count('Planificari\_organizatorice');

  try\_p7;

  ROLLBACK TO tC;

  -- (D) SALI gol (necesita sa golesti tabele care refera SALI)

  DBMS\_OUTPUT.PUT\_LINE('--- (D) SALI gol ---');

  SAVEPOINT tD;

  safe\_exec('DELETE FROM Planificari\_organizatorice');

  safe\_exec('DELETE FROM Rezervari'); -- daca Rezervari refera SALI

  safe\_exec('DELETE FROM Sali');

  safe\_count('Sali');

  try\_p7;

  ROLLBACK TO tD;

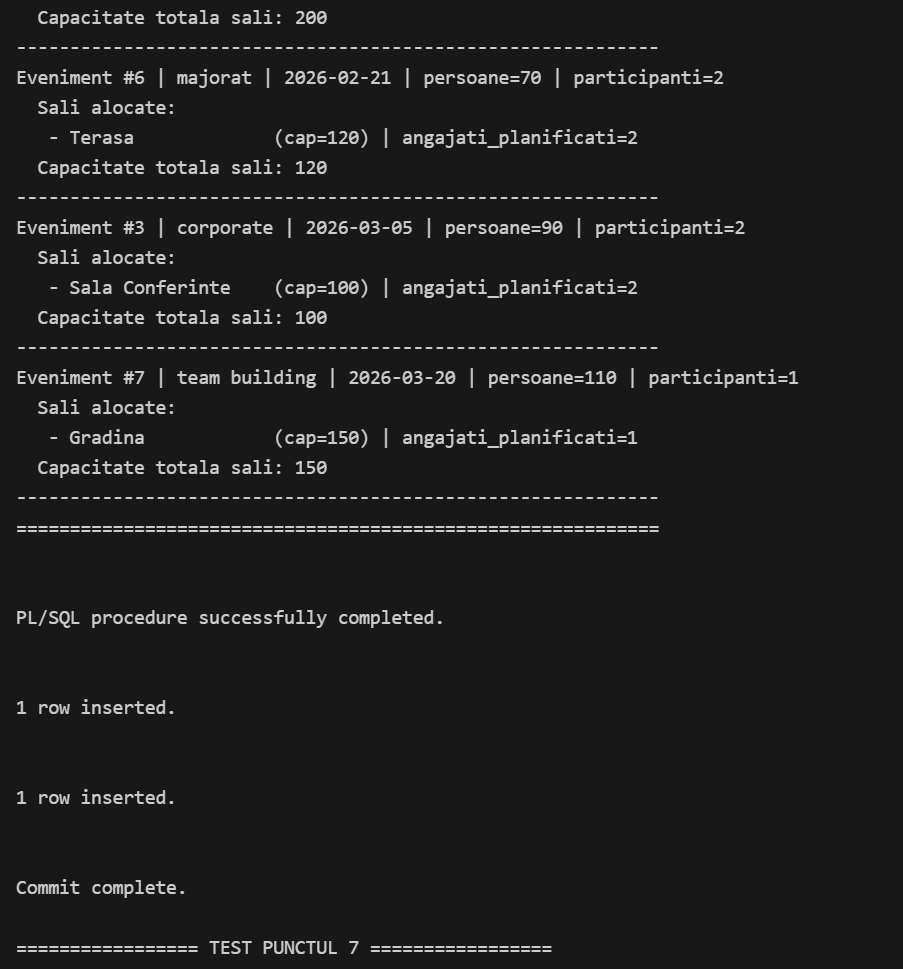
  ROLLBACK TO sp7;

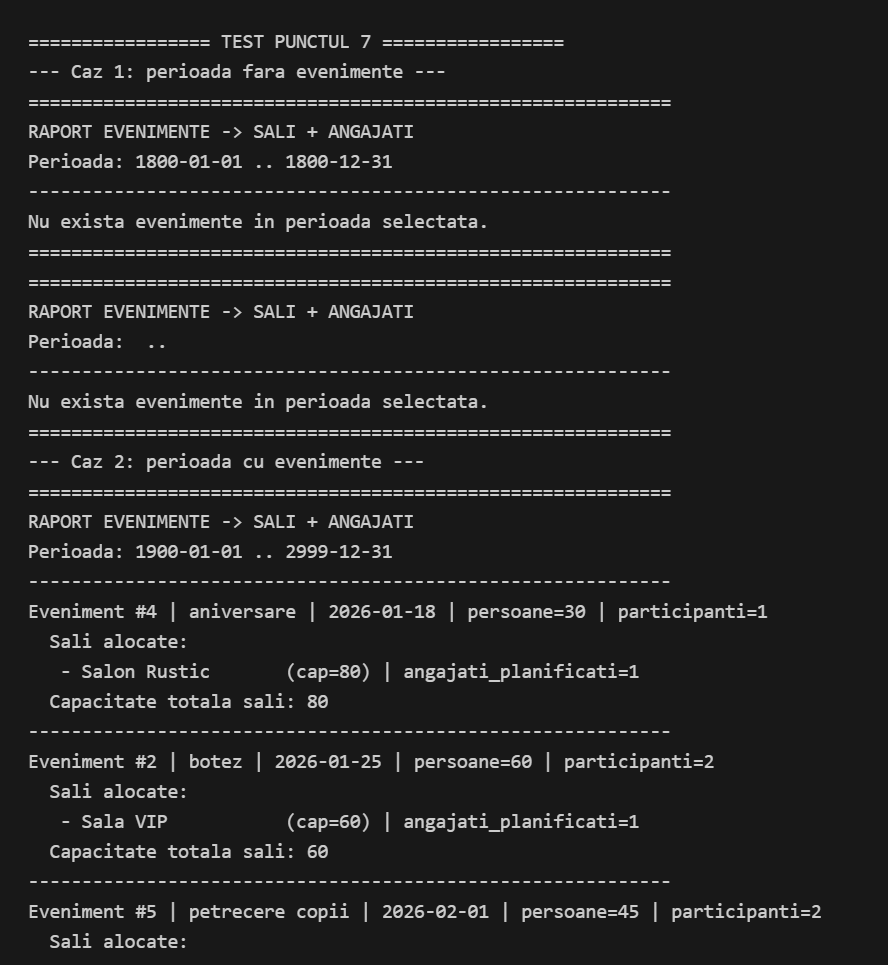
  DBMS\_OUTPUT.PUT\_LINE('===========================================================');

END;

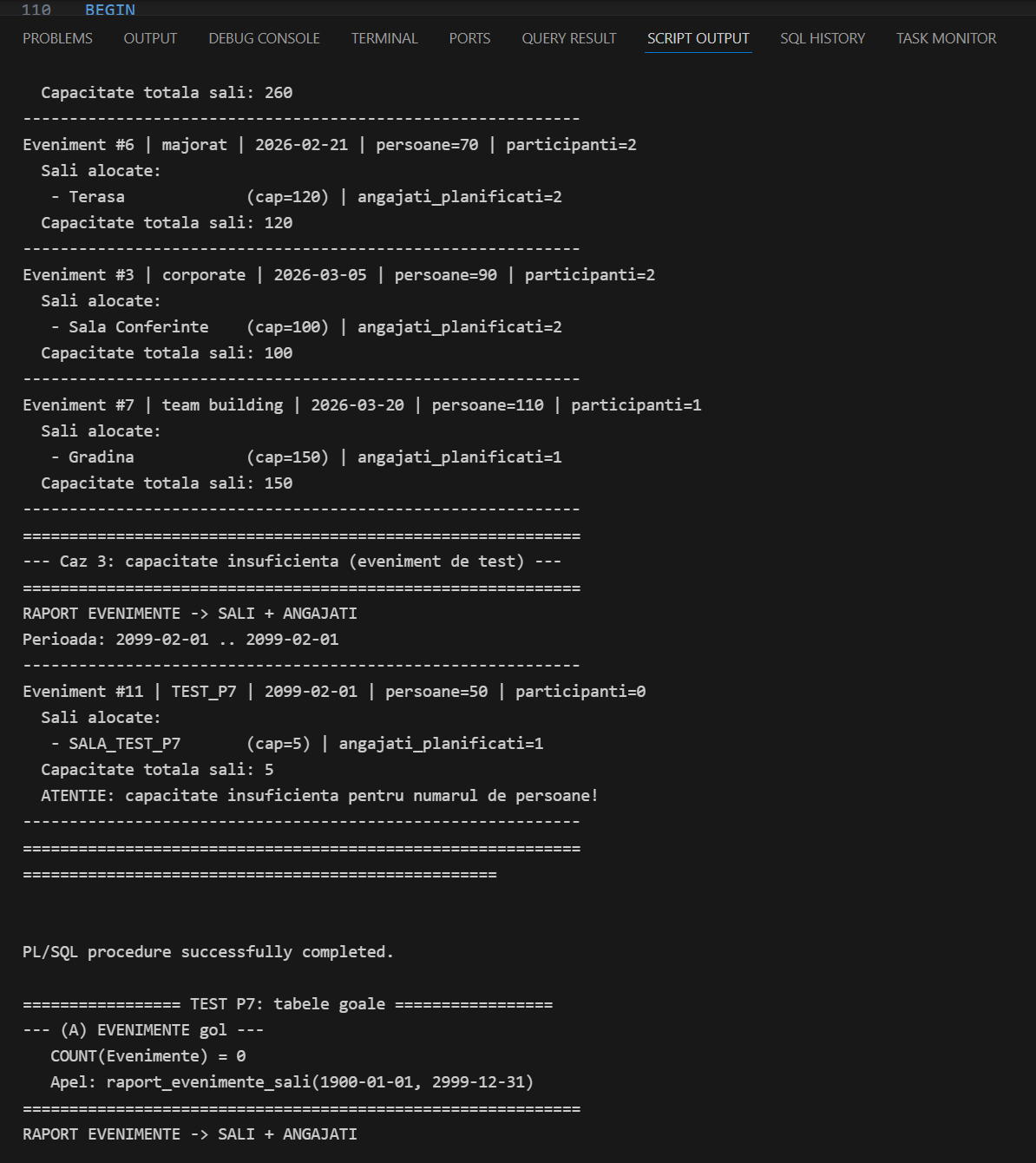
/

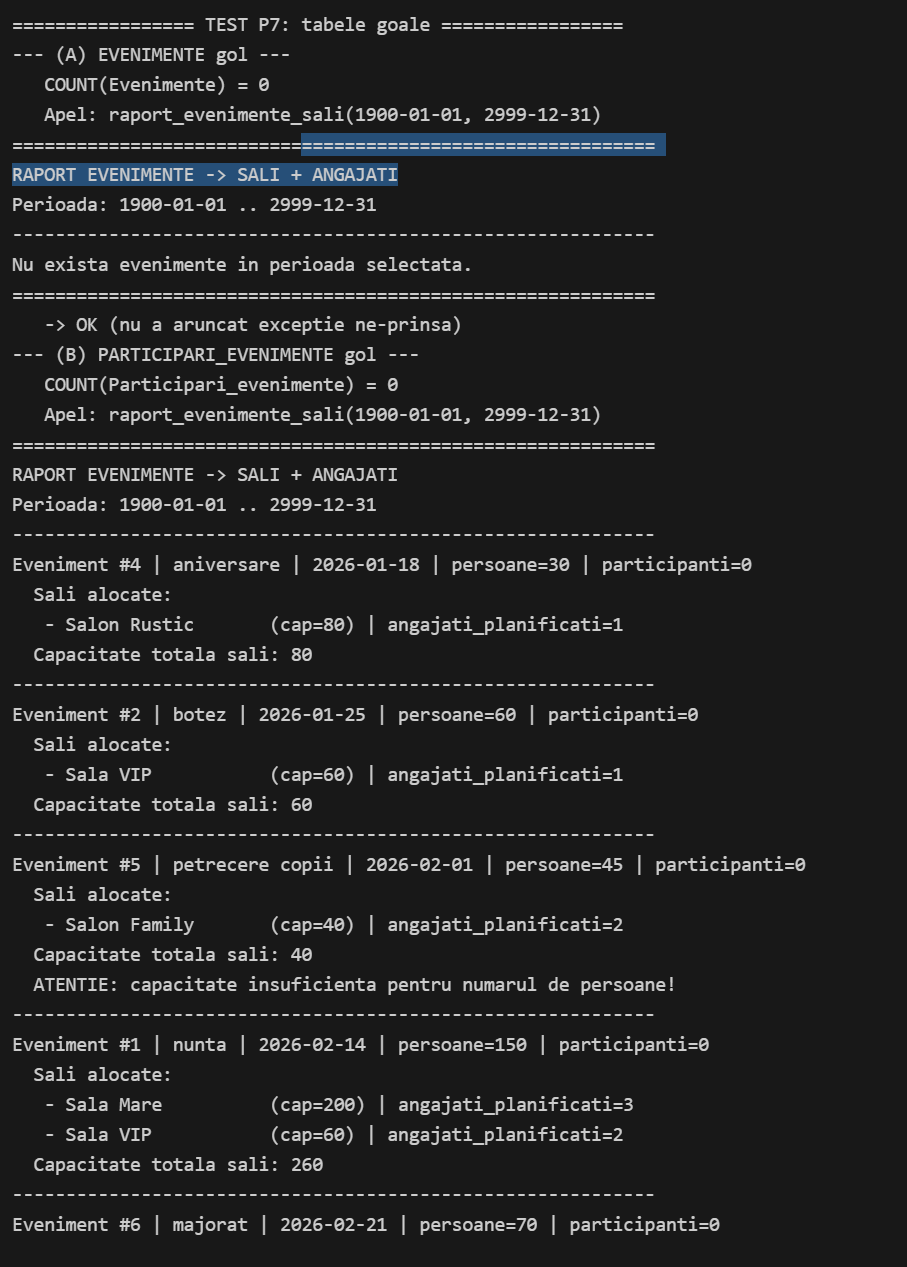
## **Print-screen:**



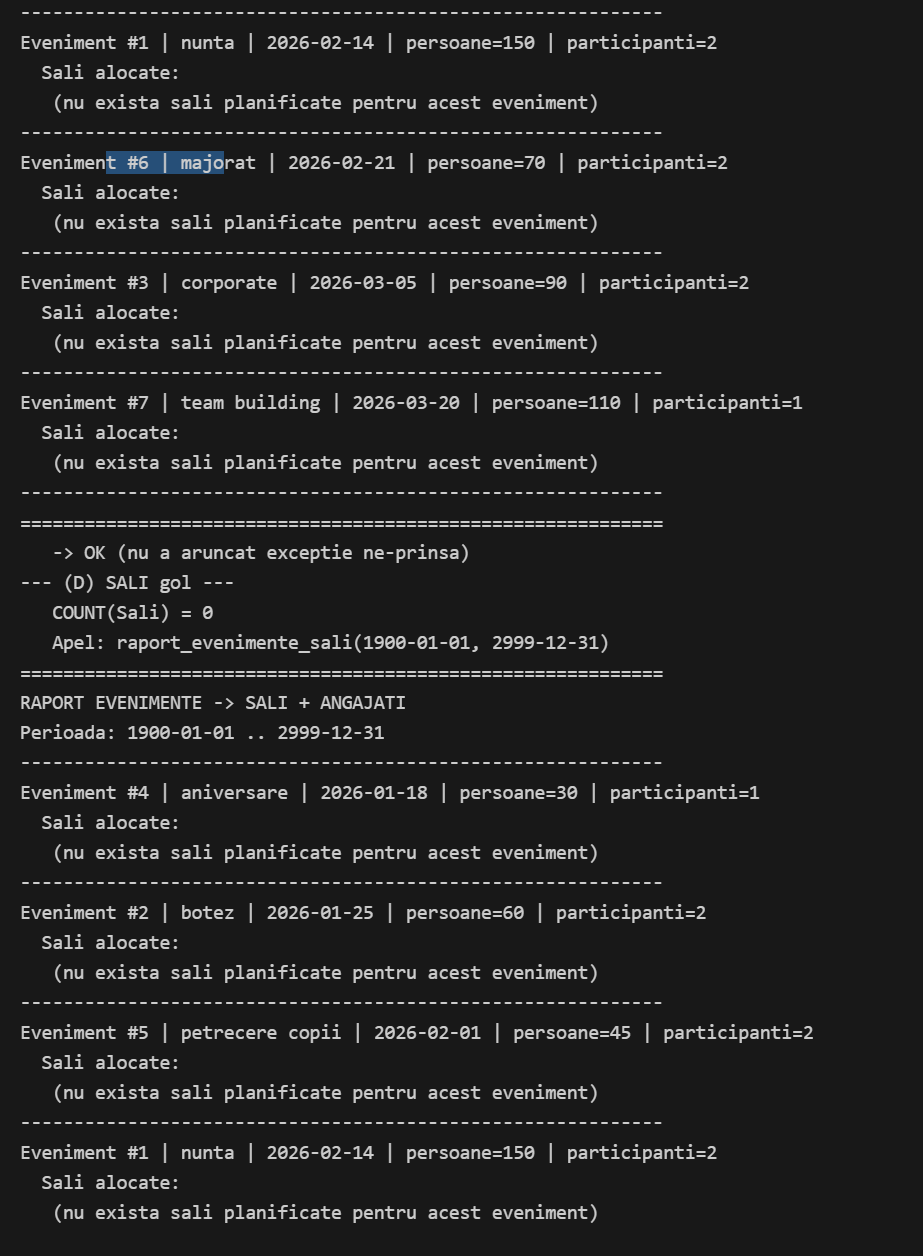














# Cerinta 8

## Cerinta:

CERINTA 8 (Proiect SGBD 2025-2026)

Pentru un client si o zi data, determina sala (si detalii despre rezervare) in care clientul are rezervare. Daca nu exista rezervare avem NO\_DATA\_FOUND.

  Daca exista mai multe rezervari in aceeasi zi pentru acel client avem TOO\_MANY\_ROWS."

Cele 3 tabele: Clienti, Rezervari, Sali

## Rezolvare:

SET SERVEROUTPUT ON;

-- optional curatare

BEGIN

  EXECUTE IMMEDIATE 'DROP FUNCTION f\_detalii\_rezervare\_client\_zi';

EXCEPTION

  WHEN OTHERS THEN NULL;

END;

/

SHOW ERRORS;

CREATE OR REPLACE FUNCTION f\_detalii\_rezervare\_client\_zi(

    p\_id\_client IN NUMBER,

    p\_data      IN DATE

) RETURN VARCHAR2

AS

    -- exceptie proprie pentru parametri invalizi

    e\_param\_invalizi EXCEPTION;

    v\_rezultat VARCHAR2(4000);

BEGIN

    IF p\_id\_client IS NULL OR p\_data IS NULL OR p\_id\_client <> TRUNC(p\_id\_client) THEN

        RAISE e\_param\_invalizi;

    END IF;

    -- O singura comanda SQL care foloseste 3 tabele: Clienti, Rezervari, Sali

    SELECT

        'Client: ' || c.nume || ' ' || c.prenume ||

        ' | Sala: '  || s.nume\_sala ||

        ' | Data: '  || TO\_CHAR(r.data\_rezervare, 'YYYY-MM-DD') ||

        ' | Ora: '   || r.ora\_rezervare ||

        ' | Persoane: ' || r.numar\_persoane

    INTO v\_rezultat

    FROM Clienti c

    JOIN Rezervari r ON r.id\_client = c.id\_client

    JOIN Sali s      ON s.id\_sala   = r.id\_sala

    WHERE c.id\_client = p\_id\_client

      AND TRUNC(r.data\_rezervare) = TRUNC(p\_data);

    RETURN v\_rezultat;

EXCEPTION

    WHEN e\_param\_invalizi THEN

        DBMS\_OUTPUT.PUT\_LINE('[PARAMETRI INVALIZI] id\_client sau data nu sunt corecte.');

        RETURN 'PARAMETRI INVALIZI';

    WHEN NO\_DATA\_FOUND THEN

        DBMS\_OUTPUT.PUT\_LINE('[NO\_DATA\_FOUND] Nu exista rezervare pentru clientul ' || p\_id\_client ||

                             ' la data ' || TO\_CHAR(p\_data, 'YYYY-MM-DD') || '.');

        RETURN 'NU EXISTA REZERVARE';

    WHEN TOO\_MANY\_ROWS THEN

        DBMS\_OUTPUT.PUT\_LINE('[TOO\_MANY\_ROWS] Exista mai multe rezervari pentru clientul ' || p\_id\_client ||

                             ' la data ' || TO\_CHAR(p\_data, 'YYYY-MM-DD') || '.');

        DBMS\_OUTPUT.PUT\_LINE('Lista rezervarilor din acea zi:');

        FOR rr IN (

            SELECT r.id\_rezervare,

                   TO\_CHAR(r.data\_rezervare, 'YYYY-MM-DD') AS data\_rez,

                   r.ora\_rezervare,

                   r.numar\_persoane,

                   s.nume\_sala

            FROM Rezervari r

            JOIN Sali s ON s.id\_sala = r.id\_sala

            WHERE r.id\_client = p\_id\_client

              AND TRUNC(r.data\_rezervare) = TRUNC(p\_data)

            ORDER BY r.ora\_rezervare, r.id\_rezervare

        ) LOOP

            DBMS\_OUTPUT.PUT\_LINE(' - Rez#' || rr.id\_rezervare ||

                                 ' | ' || rr.data\_rez ||

                                 ' ' || rr.ora\_rezervare ||

                                 ' | sala=' || rr.nume\_sala ||

                                 ' | persoane=' || rr.numar\_persoane);

        END LOOP;

        RETURN 'PREA MULTE REZERVARI';

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('[ALTA EROARE] ' || SQLERRM);

        RETURN 'EROARE: ' || SQLERRM;

END;

/

SHOW ERRORS;

-- ============================================================

-- APELURI DE TEST

-- ============================================================

DECLARE

    v\_client\_id NUMBER := 1;  -- presupunere standard in setul de date 7/12 (ID-uri identity de la 1)

    v\_sala\_id   NUMBER := 1;

    v\_data\_ok   DATE   := DATE '2099-12-30';

BEGIN

    DBMS\_OUTPUT.PUT\_LINE('================ TEST PUNCTUL 8 ================');

    -- Folosim SAVEPOINT ca sa nu "murdarim" baza de date dupa demonstratie

    SAVEPOINT sp\_test\_8;

    -- 1) Caz NORMAL: exact o rezervare in acea zi

    INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala)

    VALUES (v\_data\_ok, '12:00', 4, v\_client\_id, v\_sala\_id);

    DBMS\_OUTPUT.PUT\_LINE('--- Caz normal (1 rand) ---');

    DBMS\_OUTPUT.PUT\_LINE(f\_detalii\_rezervare\_client\_zi(v\_client\_id, v\_data\_ok));

    -- 2) Caz NO\_DATA\_FOUND: zi fara rezervare

    DBMS\_OUTPUT.PUT\_LINE('--- Caz NO\_DATA\_FOUND ---');

    DBMS\_OUTPUT.PUT\_LINE(f\_detalii\_rezervare\_client\_zi(v\_client\_id, DATE '2099-12-29'));

    DBMS\_OUTPUT.PUT\_LINE(f\_detalii\_rezervare\_client\_zi(v\_client\_id, NULL));

    -- 3) Caz TOO\_MANY\_ROWS: introducem a doua rezervare in aceeasi zi pentru acelasi client

    INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala)

    VALUES (v\_data\_ok, '14:00', 2, v\_client\_id, v\_sala\_id);

    DBMS\_OUTPUT.PUT\_LINE('--- Caz TOO\_MANY\_ROWS ---');

    DBMS\_OUTPUT.PUT\_LINE(f\_detalii\_rezervare\_client\_zi(v\_client\_id, v\_data\_ok));

    -- 4) Caz parametri invalizi (exceptie proprie)

    DBMS\_OUTPUT.PUT\_LINE('--- Caz parametri invalizi ---');

    DBMS\_OUTPUT.PUT\_LINE(f\_detalii\_rezervare\_client\_zi(v\_client\_id, NULL));

    DBMS\_OUTPUT.PUT\_LINE('--- Caz parametri invalizi 2---');

    DBMS\_OUTPUT.PUT\_LINE(f\_detalii\_rezervare\_client\_zi(1.5, DATE '2099-12-29'));

    -- revenim la starea initiala

    ROLLBACK TO sp\_test\_8;

    DBMS\_OUTPUT.PUT\_LINE('================================================');

END;

/

-- SET SERVEROUTPUT ON;

DECLARE

  v\_ret VARCHAR2(4000);

  PROCEDURE safe\_exec(p\_sql VARCHAR2) IS

  BEGIN

    EXECUTE IMMEDIATE p\_sql;

  EXCEPTION WHEN OTHERS THEN

    DBMS\_OUTPUT.PUT\_LINE('   [WARN] ' || p\_sql || ' -> ' || SQLERRM);

  END;

  PROCEDURE safe\_count(p\_table VARCHAR2) IS

    v\_cnt NUMBER;

  BEGIN

    EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM ' || p\_table INTO v\_cnt;

    DBMS\_OUTPUT.PUT\_LINE('   COUNT(' || p\_table || ') = ' || v\_cnt);

  EXCEPTION WHEN OTHERS THEN

    DBMS\_OUTPUT.PUT\_LINE('   COUNT(' || p\_table || ') = ? -> ' || SQLERRM);

  END;

  PROCEDURE try\_p8 IS

  BEGIN

    v\_ret := f\_detalii\_rezervare\_client\_zi(1, DATE '2099-01-01');

    DBMS\_OUTPUT.PUT\_LINE('   Return: ' || v\_ret);

    DBMS\_OUTPUT.PUT\_LINE('   -> OK (nu a aruncat exceptie ne-prinsa)');

  EXCEPTION WHEN OTHERS THEN

    DBMS\_OUTPUT.PUT\_LINE('   -> EROARE NEPRINSA: ' || SQLERRM);

  END;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('================= TEST P8: tabele goale =================');

  SAVEPOINT sp8;

  -- (A) CLIENTI gol (stergem intai Rezervari)

  DBMS\_OUTPUT.PUT\_LINE('--- (A) CLIENTI gol ---');

  SAVEPOINT tA;

  safe\_exec('DELETE FROM Rezervari');

  safe\_exec('DELETE FROM Participari\_evenimente');

  safe\_exec('DELETE FROM Comenzi\_produse');

  safe\_exec('DELETE FROM Comenzi\_angajati');

  safe\_exec('DELETE FROM Comenzi');

  safe\_exec('DELETE FROM Clienti');

  safe\_count('Clienti');

  try\_p8;

  ROLLBACK TO tA;

  -- (B) REZERVARI gol

  DBMS\_OUTPUT.PUT\_LINE('--- (B) REZERVARI gol ---');

  SAVEPOINT tB;

  safe\_exec('DELETE FROM Rezervari');

  safe\_count('Rezervari');

  try\_p8;

  ROLLBACK TO tB;

  -- (C) SALI gol (necesita sa golesti tabele care refera SALI)

  DBMS\_OUTPUT.PUT\_LINE('--- (C) SALI gol ---');

  SAVEPOINT tC;

  safe\_exec('DELETE FROM Rezervari');

  safe\_exec('DELETE FROM Planificari\_organizatorice');

  safe\_exec('DELETE FROM Sali');

  safe\_count('Sali');

  try\_p8;

  ROLLBACK TO tC;

  ROLLBACK TO sp8;

  DBMS\_OUTPUT.PUT\_LINE('===========================================================');

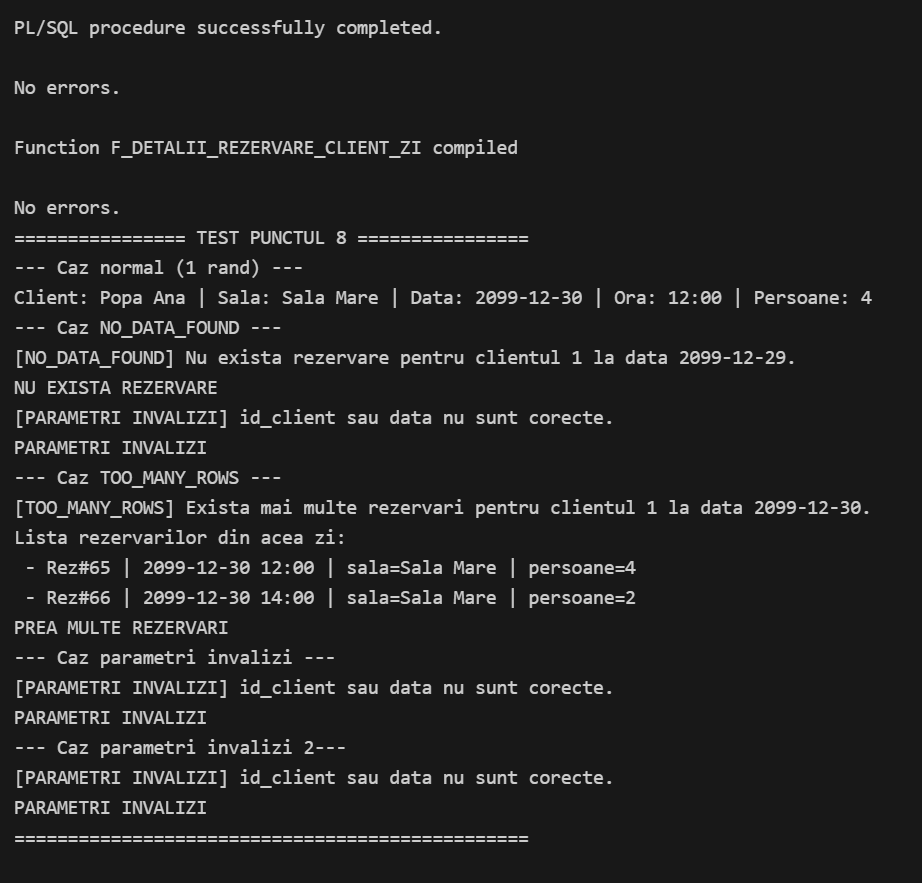
END;

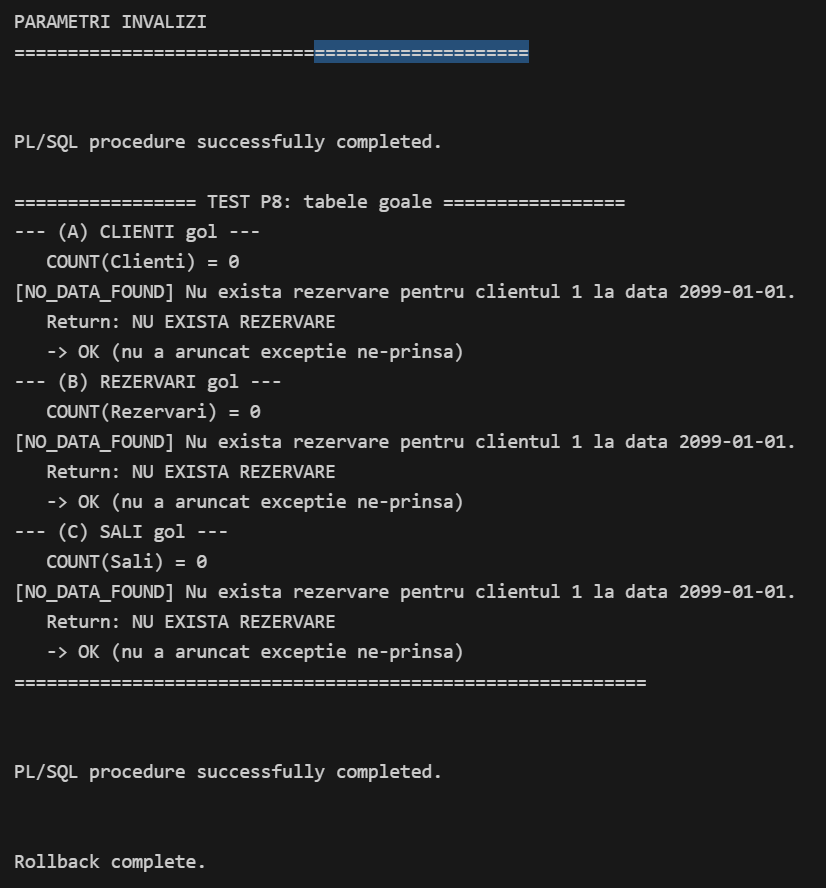
/

ROLLBACK;

/

## Print-screen:





# Cerinta 9

## Cerinta:

CERINTA 9 (Proiect SGBD 2025-2026)

Pentru un client dat, afiseaza ultimele N evenimente la care a participat (N = p\_max\_evenimente: 1 -> ultimul, 2 -> ultimele doua, etc.).

  Pentru fiecare eveniment, afiseaza raportul de participare:

  - detalii eveniment, total participanti inscrisi, angajati planificati,

  - sali alocate, capacitate totala

**- o lista cu ceilalti participanti la eveniment inregistrati (in vederea posibil identificarii unui network)**

## Rezolvare:

SET SERVEROUTPUT ON;

BEGIN

  EXECUTE IMMEDIATE 'DROP PROCEDURE p9\_raport\_istoric\_client';

EXCEPTION

  WHEN OTHERS THEN NULL;

END;

/

SHOW ERRORS;

CREATE OR REPLACE PROCEDURE p9\_raport\_istoric\_client(

    p\_id\_client      IN NUMBER,

    p\_max\_evenimente IN NUMBER

)

AS

    -- Exceptii proprii (minim 2)

    e\_parametri\_invalizi EXCEPTION;

    e\_fara\_evenimente    EXCEPTION;

    v\_nume\_client   VARCHAR2(200);

    v\_afisate       PLS\_INTEGER := 0;

    -- Cursor = O singura comanda SQL care foloseste 5 tabele distincte:

    --   Clienti, Evenimente, Participari\_evenimente, Planificari\_organizatorice, Sali

    CURSOR c\_raport(p\_cid NUMBER, p\_lim NUMBER) IS

      SELECT \*

      FROM (

        SELECT

          e.id\_eveniment,

          e.tip\_eveniment,

          e.data\_eveniment,

          e.numar\_persoane,

          c.nume || ' ' || c.prenume AS nume\_client,

          COUNT(DISTINCT pe\_all.id\_client) AS total\_participanti\_inregistrati,

          COUNT(DISTINCT po.id\_angajat)    AS angajati\_planificati,

          COUNT(DISTINCT po.id\_sala)       AS nr\_sali,

          -- capacitate totala (sum pe sali distincte) via subquery

          NVL((

              SELECT SUM(s2.capacitate)

              FROM Sali s2

              WHERE s2.id\_sala IN (

                SELECT DISTINCT po2.id\_sala

                FROM Planificari\_organizatorice po2

                WHERE po2.id\_eveniment = e.id\_eveniment

              )

          ), 0) AS capacitate\_totala,

          ROW\_NUMBER() OVER (ORDER BY e.data\_eveniment DESC, e.id\_eveniment DESC) AS rn

        FROM Evenimente e

        JOIN Clienti c

          ON c.id\_client = p\_cid

        LEFT JOIN Participari\_evenimente pe\_all

          ON pe\_all.id\_eveniment = e.id\_eveniment

        LEFT JOIN Planificari\_organizatorice po

          ON po.id\_eveniment = e.id\_eveniment

        WHERE EXISTS (

          SELECT 1

          FROM Participari\_evenimente pe

          WHERE pe.id\_eveniment = e.id\_eveniment

            AND pe.id\_client = p\_cid

        )

        GROUP BY

          e.id\_eveniment, e.tip\_eveniment, e.data\_eveniment, e.numar\_persoane,

          c.nume, c.prenume

      )

      WHERE rn <= p\_lim

      ORDER BY data\_eveniment DESC, id\_eveniment DESC;

BEGIN

    -- 0) Validare parametri

    IF p\_id\_client IS NULL OR p\_max\_evenimente IS NULL OR p\_max\_evenimente < 1 OR p\_id\_client <> TRUNC(p\_id\_client) OR p\_max\_evenimente <> TRUNC(p\_max\_evenimente) THEN

      RAISE e\_parametri\_invalizi;

    END IF;

    -- 1) Preluam numele clientului (poate arunca NO\_DATA\_FOUND)

    SELECT nume || ' ' || prenume

    INTO   v\_nume\_client

    FROM   Clienti

    WHERE  id\_client = p\_id\_client;

    DBMS\_OUTPUT.PUT\_LINE('============================================================');

    DBMS\_OUTPUT.PUT\_LINE('ISTORIC EVENIMENTE CLIENT (P9)');

    DBMS\_OUTPUT.PUT\_LINE('Client #' || p\_id\_client || ': ' || v\_nume\_client);

    DBMS\_OUTPUT.PUT\_LINE('Ultimele N evenimente (N=' || p\_max\_evenimente || '), incepand cu cel mai recent.');

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

    -- 2) Afisam raportul pentru fiecare eveniment selectat

    FOR r IN c\_raport(p\_id\_client, p\_max\_evenimente) LOOP

      v\_afisate := v\_afisate + 1;

      DBMS\_OUTPUT.PUT\_LINE('Eveniment #' || r.id\_eveniment ||

                             ' | ' || r.tip\_eveniment ||

                             ' | Data: ' || TO\_CHAR(r.data\_eveniment, 'YYYY-MM-DD') ||

                             ' | Persoane(plan): ' || r.numar\_persoane);

      DBMS\_OUTPUT.PUT\_LINE('  Participanti inscrisi: ' || r.total\_participanti\_inregistrati ||

                             ' | Angajati planificati: ' || r.angajati\_planificati);

      DBMS\_OUTPUT.PUT\_LINE('  Sali: ' || r.nr\_sali ||

                             ' | Capacitate totala: ' || r.capacitate\_totala);

      -- Lista altor clienti inscrisi la eveniment (cu colectie + exceptii specifice colectiilor)

      DECLARE

        TYPE t\_alti\_clienti\_nt IS TABLE OF VARCHAR2(200); -- NESTED TABLE (PL/SQL)

        v\_alti\_clienti t\_alti\_clienti\_nt;                 -- va fi populata prin BULK COLLECT

      BEGIN

        SELECT 'Client #' || c2.id\_client || ': ' || c2.nume || ' ' || c2.prenume

        BULK COLLECT INTO v\_alti\_clienti

        FROM   Participari\_evenimente pe2

        JOIN   Clienti c2 ON c2.id\_client = pe2.id\_client

        WHERE  pe2.id\_eveniment = r.id\_eveniment

          AND  pe2.id\_client <> p\_id\_client

        ORDER BY c2.id\_client;

        -- Afisare: incercam sa afisam primul element; daca nu exista -> SUBSCRIPT\_BEYOND\_COUNT

        BEGIN

          DBMS\_OUTPUT.PUT\_LINE('  Alti clienti inregistrati:');

          DBMS\_OUTPUT.PUT\_LINE('    - ' || v\_alti\_clienti(1)); -- daca lista e goala -> exceptie

          IF v\_alti\_clienti.COUNT > 1 THEN

            FOR i IN 2 .. v\_alti\_clienti.COUNT LOOP

              DBMS\_OUTPUT.PUT\_LINE('    - ' || v\_alti\_clienti(i));

            END LOOP;

          END IF;

          DBMS\_OUTPUT.PUT\_LINE('    (Total alti clienti: ' || v\_alti\_clienti.COUNT || ')');

        EXCEPTION

          WHEN SUBSCRIPT\_BEYOND\_COUNT THEN

            DBMS\_OUTPUT.PUT\_LINE('  Alti clienti inregistrati: (niciunul) -> clientul este singurul inregistrat.');

          WHEN COLLECTION\_IS\_NULL THEN

            DBMS\_OUTPUT.PUT\_LINE('  [EROARE COLECTIE] Colectia este NULL (COLLECTION\_IS\_NULL).');

          WHEN OTHERS THEN

            DBMS\_OUTPUT.PUT\_LINE('  [EROARE COLECTIE] ' || SQLERRM);

        END;

      END;

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

    END LOOP;

    -- 3) Daca nu exista evenimente pentru client

    IF v\_afisate = 0 THEN

      RAISE e\_fara\_evenimente;

    END IF;

    DBMS\_OUTPUT.PUT\_LINE('Total evenimente afisate: ' || v\_afisate);

    DBMS\_OUTPUT.PUT\_LINE('============================================================');

EXCEPTION

    WHEN e\_parametri\_invalizi THEN

      DBMS\_OUTPUT.PUT\_LINE('[P9] PARAMETRI INVALIDI: p\_id\_client si p\_max\_evenimente trebuie sa fie nenule, iar p\_max\_evenimente >=1 si ambele sunt numere intregi.');

    WHEN e\_fara\_evenimente THEN

      DBMS\_OUTPUT.PUT\_LINE('[P9] NU EXISTA EVENIMENTE: clientul #' || p\_id\_client ||

                           ' nu are participari inregistrate.');

    WHEN NO\_DATA\_FOUND THEN

      DBMS\_OUTPUT.PUT\_LINE('[P9] NO\_DATA\_FOUND: nu exista client cu id\_client=' || p\_id\_client);

    WHEN TOO\_MANY\_ROWS THEN

      DBMS\_OUTPUT.PUT\_LINE('[P9] TOO\_MANY\_ROWS: o interogare SELECT INTO a returnat mai multe randuri.');

    WHEN OTHERS THEN

      DBMS\_OUTPUT.PUT\_LINE('[P9] ALTA EROARE: ' || SQLERRM);

END;

/

SHOW ERRORS;

-- ============================================================

-- TESTARE

-- ============================================================

DECLARE

  v\_client\_ok     NUMBER;

  v\_client\_fara   NUMBER;

  v\_tip\_existent Evenimente.tip\_eveniment%TYPE;

  v\_evt\_singur  NUMBER;  -- id\_eveniment pentru cazul clientul e singurul

  v\_evt\_multi   NUMBER;  -- id\_eveniment pentru cazul mai multi alti clienti

  v\_c2          NUMBER;  -- id\_client pentru primul client extra

  v\_c3          NUMBER;  -- id\_client pentru al doilea client extra

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('================= TEST PUNCTUL 9 =================');

  SAVEPOINT sp9;

  -- A) Alegem un client existent (sau cream unul daca tabela e goala)

  BEGIN

    SELECT MIN(id\_client) INTO v\_client\_ok FROM Clienti;

    IF v\_client\_ok IS NULL THEN

      INSERT INTO Clienti (nume, prenume, nr\_tel, mail)

      VALUES ('Client', 'Default', '0712345678',

              'p9\_' || TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3') || '@ex.com')

      RETURNING id\_client INTO v\_client\_ok;

    END IF;

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

      INSERT INTO Clienti (nume, prenume, nr\_tel, mail)

      VALUES ('Client', 'Default', '0712345678',

              'p9\_' || TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3') || '@ex.com')

      RETURNING id\_client INTO v\_client\_ok;

  END;

  -- Tip eveniment existent (ca sa treaca orice CHECK posibil)

  BEGIN

    SELECT tip\_eveniment INTO v\_tip\_existent FROM Evenimente WHERE ROWNUM = 1;

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

      v\_tip\_existent := 'TEST';

  END;

  -- 1) Parametri invalizi (exceptie proprie)

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 1: parametri invalizi ---');

  p9\_raport\_istoric\_client(v\_client\_ok, 0);

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 1: parametri invalizi 2---');

  p9\_raport\_istoric\_client(v\_client\_ok, 1.5);

  -- 2) Client inexistent (NO\_DATA\_FOUND)

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 2: client inexistent ---');

  p9\_raport\_istoric\_client(-999999, 1);

  -- 3) Client fara participari (exceptie proprie)

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 3: client fara participari ---');

  INSERT INTO Clienti (nume, prenume, nr\_tel, mail)

  VALUES ('Client', 'FaraParticipari', '0799999999',

          'p9\_nop\_' || TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3') || '@ex.com')

  RETURNING id\_client INTO v\_client\_fara;

  p9\_raport\_istoric\_client(v\_client\_fara, 2);

  -- 4) Eveniment unde clientul e singurul inregistrat (SUBSCRIPT\_BEYOND\_COUNT pe colectie)

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 4: singurul client (lista alti clienti vida -> SUBSCRIPT\_BEYOND\_COUNT) ---');

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES (v\_tip\_existent, DATE '2099-12-30', 10)

  RETURNING id\_eveniment INTO v\_evt\_singur;

  INSERT INTO Participari\_evenimente (id\_eveniment, id\_client)

  VALUES (v\_evt\_singur, v\_client\_ok);

  p9\_raport\_istoric\_client(v\_client\_ok, 1);

  -- 5) Eveniment cu mai multi alti clienti (colectia are elemente -> listare)

  DBMS\_OUTPUT.PUT\_LINE('--- Caz 5: mai multi clienti (colectia are elemente -> listare completa) ---');

  INSERT INTO Clienti (nume, prenume, nr\_tel, mail)

  VALUES ('Client', 'Extra1', '0711111111',

          'p9\_c2\_' || TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3') || '@ex.com')

  RETURNING id\_client INTO v\_c2;

  INSERT INTO Clienti (nume, prenume, nr\_tel, mail)

  VALUES ('Client', 'Extra2', '0722222222',

          'p9\_c3\_' || TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3') || '@ex.com')

  RETURNING id\_client INTO v\_c3;

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES (v\_tip\_existent, DATE '2100-01-01', 20)

  RETURNING id\_eveniment INTO v\_evt\_multi;

  INSERT INTO Participari\_evenimente (id\_eveniment, id\_client)

  VALUES (v\_evt\_multi, v\_client\_ok);

  INSERT INTO Participari\_evenimente (id\_eveniment, id\_client)

  VALUES (v\_evt\_multi, v\_c2);

  INSERT INTO Participari\_evenimente (id\_eveniment, id\_client)

  VALUES (v\_evt\_multi, v\_c3);

  p9\_raport\_istoric\_client(v\_client\_ok, 1);

  ROLLBACK TO sp9;

  DBMS\_OUTPUT.PUT\_LINE('===================================================');

END;

/

-- SET SERVEROUTPUT ON;

DECLARE

  PROCEDURE safe\_exec(p\_sql VARCHAR2) IS

  BEGIN

    EXECUTE IMMEDIATE p\_sql;

  EXCEPTION WHEN OTHERS THEN

    DBMS\_OUTPUT.PUT\_LINE('   [WARN] ' || p\_sql || ' -> ' || SQLERRM);

  END;

  PROCEDURE safe\_count(p\_table VARCHAR2) IS

    v\_cnt NUMBER;

  BEGIN

    EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM ' || p\_table INTO v\_cnt;

    DBMS\_OUTPUT.PUT\_LINE('   COUNT(' || p\_table || ') = ' || v\_cnt);

  EXCEPTION WHEN OTHERS THEN

    DBMS\_OUTPUT.PUT\_LINE('   COUNT(' || p\_table || ') = ? -> ' || SQLERRM);

  END;

  PROCEDURE try\_p9 IS

  BEGIN

    DBMS\_OUTPUT.PUT\_LINE('   Apel: p9\_raport\_istoric\_client(1, 2)');

    p9\_raport\_istoric\_client(1, 2);

    DBMS\_OUTPUT.PUT\_LINE('   -> OK (nu a aruncat exceptie ne-prinsa)');

  EXCEPTION WHEN OTHERS THEN

    DBMS\_OUTPUT.PUT\_LINE('   -> EROARE NEPRINSA: ' || SQLERRM);

  END;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('================= TEST P9: tabele goale =================');

  SAVEPOINT sp9;

  -- (A) CLIENTI gol (stergem dependente uzuale)

  DBMS\_OUTPUT.PUT\_LINE('--- (A) CLIENTI gol ---');

  SAVEPOINT tA;

  safe\_exec('DELETE FROM Rezervari');

  safe\_exec('DELETE FROM Participari\_evenimente');

  safe\_exec('DELETE FROM Comenzi\_produse');

  safe\_exec('DELETE FROM Comenzi\_angajati');

  safe\_exec('DELETE FROM Comenzi');

  safe\_exec('DELETE FROM Clienti');

  safe\_count('Clienti');

  try\_p9;

  ROLLBACK TO tA;

  -- (B) EVENIMENTE gol

  DBMS\_OUTPUT.PUT\_LINE('--- (B) EVENIMENTE gol ---');

  SAVEPOINT tB;

  safe\_exec('DELETE FROM Participari\_evenimente');

  safe\_exec('DELETE FROM Planificari\_organizatorice');

  safe\_exec('DELETE FROM Colaborari');

  safe\_exec('DELETE FROM Evenimente\_produse');

  safe\_exec('DELETE FROM Evenimente');

  safe\_count('Evenimente');

  try\_p9;

  ROLLBACK TO tB;

  -- (C) PARTICIPARI\_EVENIMENTE gol

  DBMS\_OUTPUT.PUT\_LINE('--- (C) PARTICIPARI\_EVENIMENTE gol ---');

  SAVEPOINT tC;

  safe\_exec('DELETE FROM Participari\_evenimente');

  safe\_count('Participari\_evenimente');

  try\_p9;

  ROLLBACK TO tC;

  -- (D) PLANIFICARI\_ORGANIZATORICE gol

  DBMS\_OUTPUT.PUT\_LINE('--- (D) PLANIFICARI\_ORGANIZATORICE gol ---');

  SAVEPOINT tD;

  safe\_exec('DELETE FROM Planificari\_organizatorice');

  safe\_count('Planificari\_organizatorice');

  try\_p9;

  ROLLBACK TO tD;

  -- (E) SALI gol (necesita sa golesti tabele care refera SALI)

  DBMS\_OUTPUT.PUT\_LINE('--- (E) SALI gol ---');

  SAVEPOINT tE;

  safe\_exec('DELETE FROM Rezervari');

  safe\_exec('DELETE FROM Planificari\_organizatorice');

  safe\_exec('DELETE FROM Sali');

  safe\_count('Sali');

  try\_p9;

  ROLLBACK TO tE;

  ROLLBACK TO sp9;

  DBMS\_OUTPUT.PUT\_LINE('===========================================================');

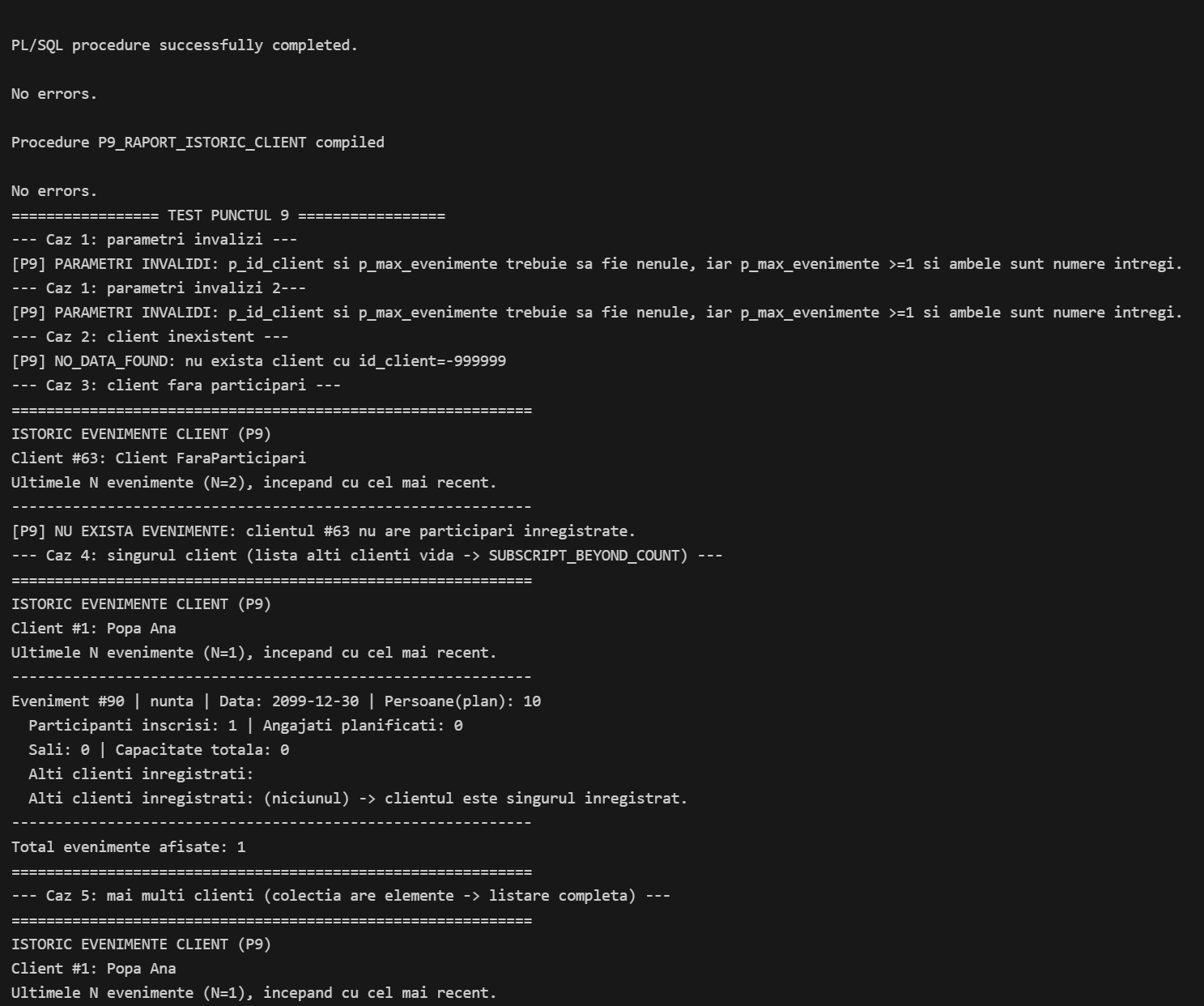
END;

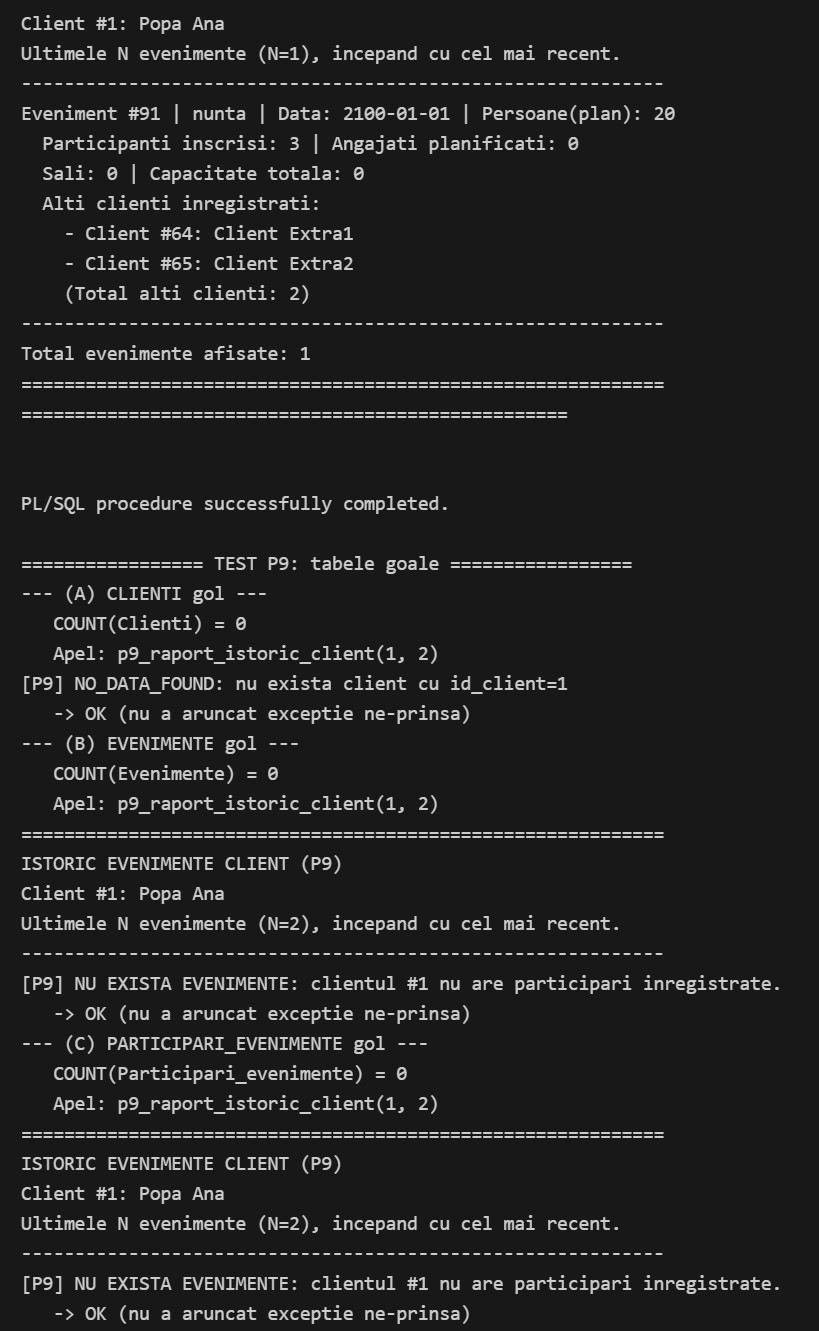
/

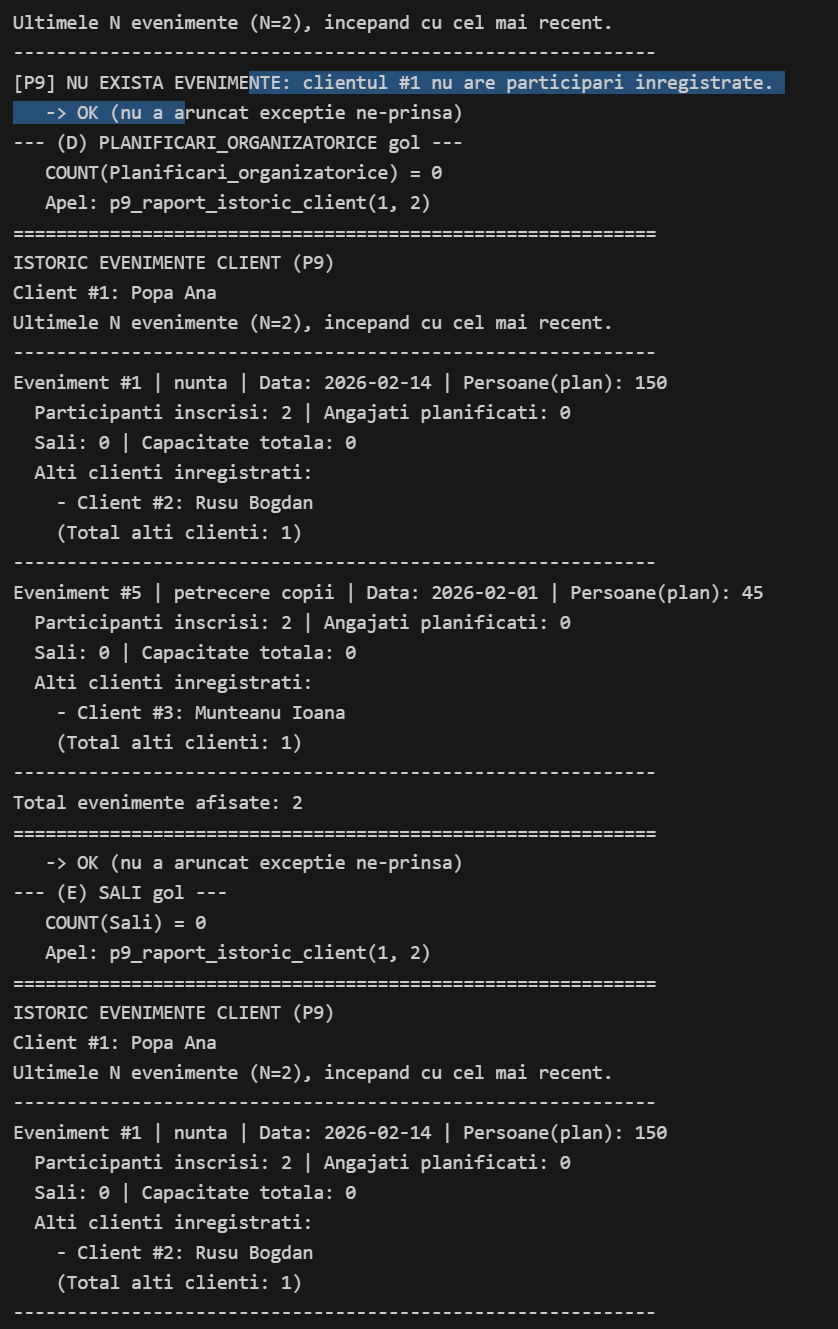
ROLLBACK;

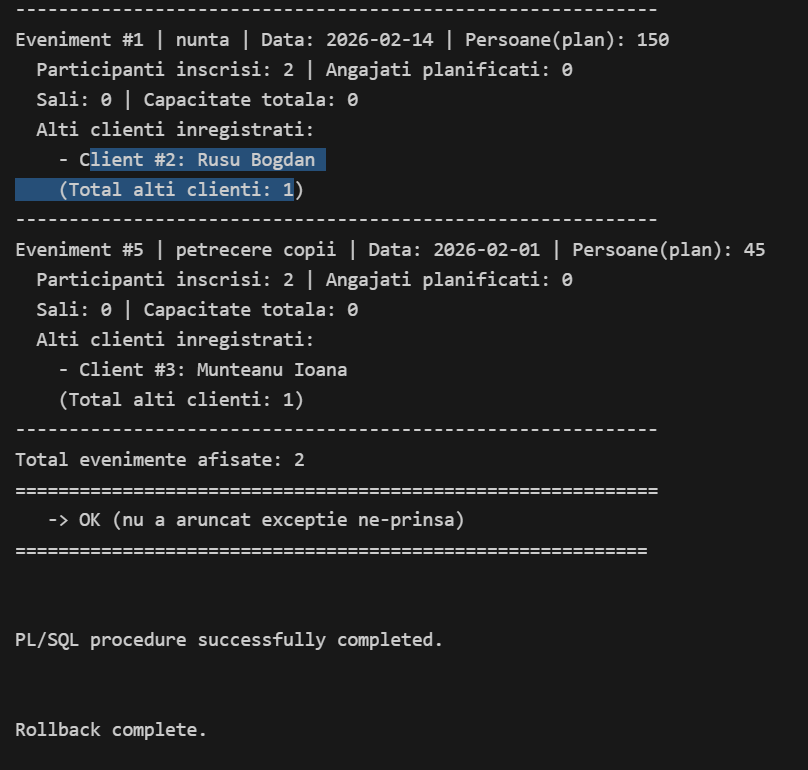
/

## Print-screen:









# Cerinta 10

## Cerinta:

CERINTA 10 (Proiect SGBD 2025-2026)

  - Dupa orice INSERT/UPDATE/DELETE in PLANIFICARI\_ORGANIZATORICE, se verifica:

      (1) Nu exista sala cu 2 evenimente diferite in aceeasi zi.

      (2) Nu exista sala in care, in aceeasi zi, exista si eveniment si rezervare.

  - Daca apare conflict, se respinge comanda si se afiseaza un mesaj util.

  - Daca, pentru data conflictului, nu mai exista nicio sala libera (toate sunt ocupate

    prin evenimente sau rezervari), mesajul cere mutarea datei evenimentului.

## Rezolvare:

BEGIN

  EXECUTE IMMEDIATE 'DROP TRIGGER trg\_po\_conflicte\_stmt';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE OR REPLACE TRIGGER trg\_po\_conflicte\_stmt

AFTER INSERT OR UPDATE ON Planificari\_organizatorice

DECLARE

  v\_sala NUMBER;

  v\_data DATE;

BEGIN

  -- 1) Conflict: aceeasi sala + aceeasi zi -> 2+ evenimente diferite

  BEGIN

    SELECT x.id\_sala, x.d

    INTO   v\_sala, v\_data

    FROM (

      SELECT po.id\_sala AS id\_sala,

             TRUNC(e.data\_eveniment) AS d

      FROM   Planificari\_organizatorice po

      JOIN   Evenimente e ON e.id\_eveniment = po.id\_eveniment

      GROUP  BY po.id\_sala, TRUNC(e.data\_eveniment)

      HAVING COUNT(DISTINCT po.id\_eveniment) > 1

    ) x

    WHERE ROWNUM = 1;

    RAISE\_APPLICATION\_ERROR(

      -20010,

      'Planificare respinsa: sala ' || v\_sala ||

      ' este alocata la mai multe evenimente in data ' || TO\_CHAR(v\_data,'YYYY-MM-DD') ||

      '. Alege alta sala sau muta data.'

    );

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

      NULL;

  END;

  -- 2) Conflict: exista rezervari in aceeasi sala + aceeasi zi cu un eveniment planificat

  BEGIN

    SELECT x.id\_sala, x.d

    INTO   v\_sala, v\_data

    FROM (

      SELECT DISTINCT po.id\_sala AS id\_sala,

             TRUNC(e.data\_eveniment) AS d

      FROM   Planificari\_organizatorice po

      JOIN   Evenimente e ON e.id\_eveniment = po.id\_eveniment

      WHERE  EXISTS (

        SELECT 1

        FROM   Rezervari r

        WHERE  r.id\_sala = po.id\_sala

          AND  TRUNC(r.data\_rezervare) = TRUNC(e.data\_eveniment)

      )

    ) x

    WHERE ROWNUM = 1;

    RAISE\_APPLICATION\_ERROR(

      -20011,

      'Planificare respinsa: exista rezervari in sala ' || v\_sala ||

      ' la data ' || TO\_CHAR(v\_data,'YYYY-MM-DD') ||

      '. Reprogrameaza rezervarile sau muta data evenimentului.'

    );

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

      NULL;

  END;

END;

/

SET SERVEROUTPUT ON;

DECLARE

  v\_sala   NUMBER;

  v\_ang    NUMBER;

  v\_cli    NUMBER;

  v\_evt\_A  NUMBER; -- pentru conflict cu rezervari

  v\_evt\_B1 NUMBER; -- conflict eveniment-eveniment

  v\_evt\_B2 NUMBER;

  PROCEDURE ok(p\_msg VARCHAR2) IS BEGIN DBMS\_OUTPUT.PUT\_LINE('  [OK] '||p\_msg); END;

  PROCEDURE fail(p\_msg VARCHAR2) IS BEGIN DBMS\_OUTPUT.PUT\_LINE('  [FAIL] '||p\_msg); END;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('================= TEST TRIGGER 10 =================');

  SAVEPOINT sp10;

  -- setup minim

  INSERT INTO Sali (nume\_sala, capacitate)

  VALUES ('Sala\_T10', 20)

  RETURNING id\_sala INTO v\_sala;

  INSERT INTO Angajati (nume, prenume, functie, salariu, mail)

  VALUES ('Test', 'Ang\_T10', 'Chelner', 3000, 't10\_ang\_'||TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3')||'@ex.com')

  RETURNING id\_angajat INTO v\_ang;

  INSERT INTO Clienti (nume, prenume, nr\_tel, mail)

  VALUES ('Test', 'Client\_T10', '0700000000', 't10\_c\_'||TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3')||'@ex.com')

  RETURNING id\_client INTO v\_cli;

  -- (1) conflict planificare vs rezervari (astept ORA-20011)

  DBMS\_OUTPUT.PUT\_LINE('--- (1) Conflict cu REZERVARI (astept ORA-20011) ---');

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES ('TEST', DATE '2026-02-10', 5)

  RETURNING id\_eveniment INTO v\_evt\_A;

  INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala)

  VALUES (DATE '2026-02-10', '18:00', 2, v\_cli, v\_sala);

  BEGIN

    INSERT INTO Planificari\_organizatorice (id\_eveniment, id\_sala, id\_angajat, observatii)

    VALUES (v\_evt\_A, v\_sala, v\_ang, 'conflict rezervari');

    fail('NU a respins (ar fi trebuit ORA-20011).');

  EXCEPTION

    WHEN OTHERS THEN

      IF SQLCODE = -20011 THEN ok('Respins corect: '||SQLERRM);

      ELSE fail('Eroare diferita: '||SQLERRM);

      END IF;

  END;

  -- (2) conflict planificare vs alt eveniment (astept ORA-20010)

  DBMS\_OUTPUT.PUT\_LINE('--- (2) Conflict cu ALT EVENIMENT (astept ORA-20010) ---');

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES ('TEST', DATE '2026-02-12', 5)

  RETURNING id\_eveniment INTO v\_evt\_B1;

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES ('TEST', DATE '2026-02-12', 5)

  RETURNING id\_eveniment INTO v\_evt\_B2;

  -- prima planificare OK

  INSERT INTO Planificari\_organizatorice (id\_eveniment, id\_sala, id\_angajat, observatii)

  VALUES (v\_evt\_B1, v\_sala, v\_ang, 'primul eveniment');

  -- a doua planificare trebuie respinsa

  BEGIN

    INSERT INTO Planificari\_organizatorice (id\_eveniment, id\_sala, id\_angajat, observatii)

    VALUES (v\_evt\_B2, v\_sala, v\_ang, 'al doilea eveniment');

    fail('NU a respins (ar fi trebuit ORA-20010).');

  EXCEPTION

    WHEN OTHERS THEN

      IF SQLCODE = -20010 THEN ok('Respins corect: '||SQLERRM);

      ELSE fail('Eroare diferita: '||SQLERRM);

      END IF;

  END;

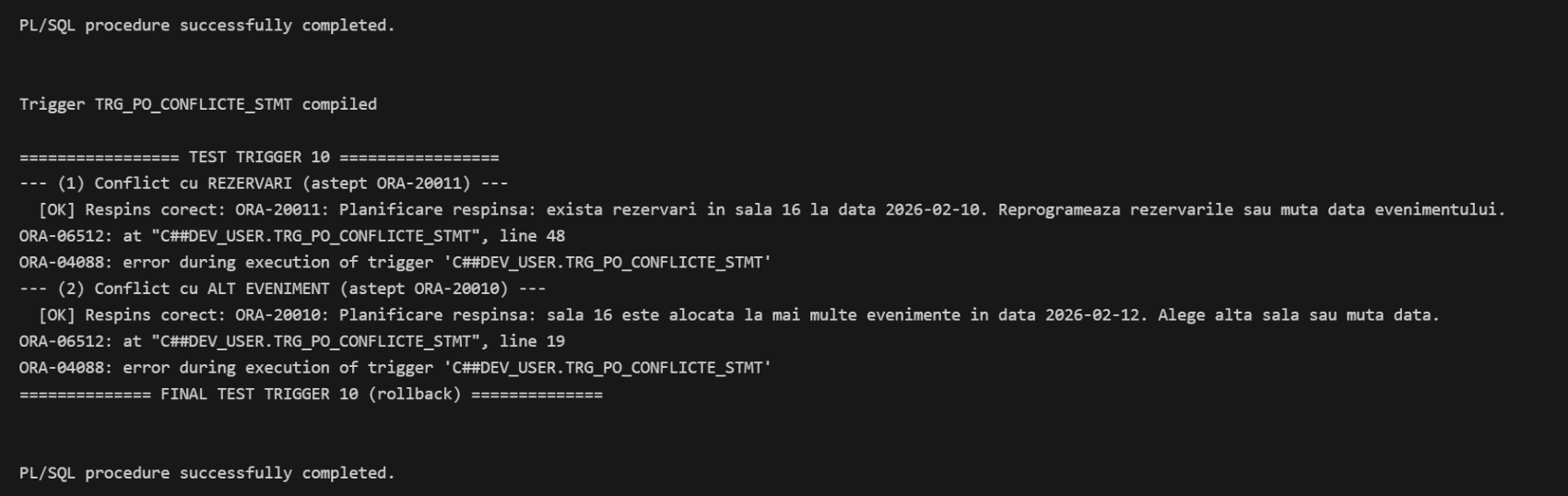
  ROLLBACK TO sp10;

  DBMS\_OUTPUT.PUT\_LINE('============== FINAL TEST TRIGGER 10 (rollback) ==============');

END;

/

## Print-screen:



# Cerinta 11

## Cerinta:

CERINTA 11 (Proiect SGBD 2025-2026)

Pentru fiecare rezervare inserata/modificata, se verifica daca in sala aleasa exista un eveniment programat in aceeasi zi. Daca exista, se respinge rezervarea.

## Rezolvare:

BEGIN

EXECUTE IMMEDIATE 'DROP TRIGGER trg\_rezervari\_capacitate';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE OR REPLACE TRIGGER trg\_rezervari\_capacitate

BEFORE INSERT OR UPDATE ON Rezervari

FOR EACH ROW

DECLARE

v\_dummy NUMBER;

BEGIN

-- Sala exista?

BEGIN

SELECT 1 INTO v\_dummy

FROM Sali

WHERE id\_sala = :NEW.id\_sala;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20020, 'Sala inexistentă (id\_sala='||:NEW.id\_sala||').');

END;

-- Exista eveniment in aceeasi sala + aceeasi zi?

BEGIN

SELECT 1 INTO v\_dummy

FROM dual

WHERE EXISTS (

SELECT 1

FROM Planificari\_organizatorice po

JOIN Evenimente e

ON e.id\_eveniment = po.id\_eveniment

WHERE po.id\_sala = :NEW.id\_sala

AND TRUNC(e.data\_eveniment) = TRUNC(:NEW.data\_rezervare)

);

-- daca EXISTS a gasit ceva => respingem

RAISE\_APPLICATION\_ERROR(

-20021,

'Rezervare respinsa: exista eveniment in sala '||:NEW.id\_sala||

' la data '||TO\_CHAR(:NEW.data\_rezervare,'YYYY-MM-DD')||'.'

);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

NULL; -- nu exista eveniment

END;

END;

/

SET SERVEROUTPUT ON;

DECLARE

v\_sala NUMBER;

v\_ang NUMBER;

v\_cli1 NUMBER;

v\_cli2 NUMBER;

v\_evt NUMBER;

v\_rez\_ok NUMBER;

PROCEDURE ok(p\_msg VARCHAR2) IS BEGIN DBMS\_OUTPUT.PUT\_LINE(' [OK] '||p\_msg); END;

PROCEDURE fail(p\_msg VARCHAR2) IS BEGIN DBMS\_OUTPUT.PUT\_LINE(' [FAIL] '||p\_msg); END;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('================= TEST TRIGGER 11 (ROW - conflict eveniment) =================');

SAVEPOINT sp11;

-- setup minim

INSERT INTO Sali (nume\_sala, capacitate)

VALUES ('Sala\_T11', 10)

RETURNING id\_sala INTO v\_sala;

INSERT INTO Angajati (nume, prenume, functie, salariu, mail)

VALUES ('Test', 'Ang\_T11', 'Chelner', 3000, 't11\_ang\_'||TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3')||'@ex.com')

RETURNING id\_angajat INTO v\_ang;

INSERT INTO Clienti (nume, prenume, nr\_tel, mail)

VALUES ('Test', 'Client1\_T11', '0700000000', 't11\_c1\_'||TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3')||'@ex.com')

RETURNING id\_client INTO v\_cli1;

INSERT INTO Clienti (nume, prenume, nr\_tel, mail)

VALUES ('Test', 'Client2\_T11', '0711111111', 't11\_c2\_'||TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3')||'@ex.com')

RETURNING id\_client INTO v\_cli2;

-- Cream un eveniment pe 2026-02-15 in sala v\_sala

INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

VALUES ('TEST', DATE '2026-02-15', 8)

RETURNING id\_eveniment INTO v\_evt;

INSERT INTO Planificari\_organizatorice (id\_eveniment, id\_sala, id\_angajat, observatii)

VALUES (v\_evt, v\_sala, v\_ang, 'blocare rezervari');

-- (1) rezervare respinsa din cauza evenimentului (astept ORA-20021)

DBMS\_OUTPUT.PUT\_LINE('--- (1) Conflict cu EVENIMENT (astept ORA-20021) ---');

BEGIN

INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala)

VALUES (DATE '2026-02-15', '19:00', 2, v\_cli1, v\_sala);

fail('NU a respins (ar fi trebuit ORA-20021).');

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE = -20021 THEN ok('Respins corect: '||SQLERRM);

ELSE fail('Eroare diferita: '||SQLERRM);

END IF;

END;

-- (2) rezervare OK intr-o zi fara eveniment

DBMS\_OUTPUT.PUT\_LINE('--- (2) Rezervare OK (fara eveniment) ---');

BEGIN

INSERT INTO Rezervari (data\_rezervare, ora\_rezervare, numar\_persoane, id\_client, id\_sala)

VALUES (DATE '2026-02-20', '18:00', 3, v\_cli1, v\_sala)

RETURNING id\_rezervare INTO v\_rez\_ok;

ok('Inserare reusita (id\_rezervare='||v\_rez\_ok||').');

EXCEPTION

WHEN OTHERS THEN

fail('NU ar fi trebuit sa dea eroare: '||SQLERRM);

END;

-- (3) UPDATE catre o zi cu eveniment (astept ORA-20021)

DBMS\_OUTPUT.PUT\_LINE('--- (3) UPDATE catre zi cu eveniment (astept ORA-20021) ---');

BEGIN

UPDATE Rezervari

SET data\_rezervare = DATE '2026-02-15'

WHERE id\_rezervare = v\_rez\_ok;

fail('NU a respins UPDATE-ul (ar fi trebuit ORA-20021).');

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE = -20021 THEN ok('Respins corect la UPDATE: '||SQLERRM);

ELSE fail('Eroare diferita: '||SQLERRM);

END IF;

END;

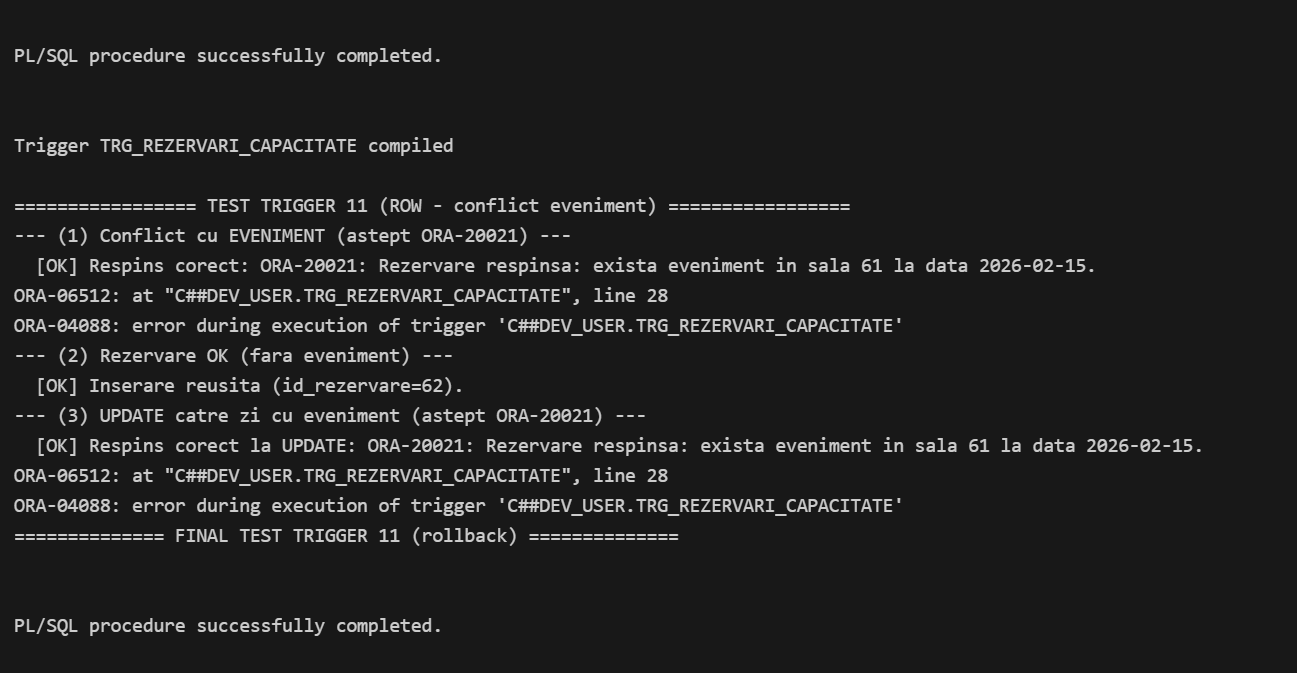
ROLLBACK TO sp11;

DBMS\_OUTPUT.PUT\_LINE('============== FINAL TEST TRIGGER 11 (rollback) ==============');

END;

/

## Print-screen:



# Cerinta 12

## Cerinta:

CERINTA 12 (Proiect SGBD 2025-2026)

Pentru fiecare actiune de adaugare, modificare sau stergere a unui tabel in baza de date sa se creeze o intrare intr-un tabel de monitorizare, ca un track-record si sa se blocheze actiuni nepermise pe entitati notate intr-un tabel special creat in acest scop.

## Rezolvare:

-- ALTER TRIGGER trg\_audit\_ddl DISABLE;

-- DROP TABLE audit\_ddl;

-- DROP TABLE guard\_ldd;

CREATE TABLE audit\_ddl (

  id           NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

  event\_time   TIMESTAMP DEFAULT SYSTIMESTAMP NOT NULL,

  db\_user      VARCHAR2(30),

  os\_user      VARCHAR2(128),

  host         VARCHAR2(128),

  ip\_address   VARCHAR2(64),

  ddl\_event    VARCHAR2(30),

  object\_type  VARCHAR2(30),

  object\_name  VARCHAR2(128)

);

CREATE TABLE guard\_ldd (

  obj\_type       VARCHAR2(30)  NOT NULL,

  obj\_name       VARCHAR2(128) NOT NULL,

  active         CHAR(1) DEFAULT 'Y' NOT NULL CHECK (active IN ('Y','N')),

  block\_drop     CHAR(1) DEFAULT 'N' NOT NULL CHECK (block\_drop IN ('Y','N')),

  block\_truncate CHAR(1) DEFAULT 'N' NOT NULL CHECK (block\_truncate IN ('Y','N')),

  block\_alter    CHAR(1) DEFAULT 'N' NOT NULL CHECK (block\_alter IN ('Y','N')),

  reason         VARCHAR2(4000),

  CONSTRAINT pk\_guard\_ldd PRIMARY KEY (obj\_type, obj\_name),

  -- majuscule by default pt comparatii

  CONSTRAINT ck\_guard\_upper CHECK (obj\_type = UPPER(obj\_type) AND obj\_name = UPPER(obj\_name)),

  -- interzise tabelele folosite de trigger in lista de guard

  CONSTRAINT ck\_guard\_no\_infra CHECK (obj\_name NOT IN ('AUDIT\_DDL','AUDIT\_LDD','GUARD\_LDD','GUARD\_DDL'))

);

CREATE OR REPLACE TRIGGER trg\_audit\_ddl

BEFORE DDL ON SCHEMA

DECLARE

  PRAGMA AUTONOMOUS\_TRANSACTION;

  v\_event    VARCHAR2(30)  := ORA\_SYSEVENT;

  v\_type     VARCHAR2(30)  := ORA\_DICT\_OBJ\_TYPE;

  v\_name     VARCHAR2(128) := ORA\_DICT\_OBJ\_NAME;

  v\_bd CHAR(1);

  v\_bt CHAR(1);

  v\_ba CHAR(1);

BEGIN

  -- 1) AUDIT (nu vrem ca auditarea sa blocheze DDL-ul daca apar probleme)

  BEGIN

    INSERT INTO audit\_ddl (db\_user, os\_user, host, ip\_address, ddl\_event, object\_type, object\_name)

    VALUES (

      SYS\_CONTEXT('USERENV','SESSION\_USER'),

      SYS\_CONTEXT('USERENV','OS\_USER'),

      SYS\_CONTEXT('USERENV','HOST'),

      SYS\_CONTEXT('USERENV','IP\_ADDRESS'),

      v\_event,

      v\_type,

      v\_name

    );

    COMMIT;

  EXCEPTION

    WHEN OTHERS THEN

      -- nu blocam DDL din cauza auditului

      NULL;

  END;

  -- 2) GUARD: nu protejam infrastructura (si nici nu vrem sa consultam guard pe ea)

  IF UPPER(v\_name) IN ('AUDIT\_DDL','AUDIT\_LDD','GUARD\_LDD','GUARD\_DDL') THEN

    RETURN;

  END IF;

  -- 3) Citim regulile de guard (daca exista)

  BEGIN

    SELECT block\_drop, block\_truncate, block\_alter

    INTO   v\_bd,       v\_bt,          v\_ba

    FROM guard\_ldd

    WHERE active   = 'Y'

      AND obj\_type = UPPER(v\_type)

      AND obj\_name = UPPER(v\_name);

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

      RETURN; -- nu e protejat

    WHEN OTHERS THEN

      RETURN; -- daca guard-ul are probleme, nu blocam DDL

  END;

  -- 4) Aplicam blocarea doar pe evenimentul cerut

  IF v\_event = 'DROP' AND v\_bd = 'Y' THEN

    RAISE\_APPLICATION\_ERROR(-20050, 'DDL blocat de GUARD\_LDD: DROP pe '||v\_type||' '||v\_name);

  ELSIF v\_event = 'TRUNCATE' AND v\_bt = 'Y' THEN

    RAISE\_APPLICATION\_ERROR(-20051, 'DDL blocat de GUARD\_LDD: TRUNCATE pe '||v\_type||' '||v\_name);

  ELSIF v\_event = 'ALTER' AND v\_ba = 'Y' THEN

    RAISE\_APPLICATION\_ERROR(-20052, 'DDL blocat de GUARD\_LDD: ALTER pe '||v\_type||' '||v\_name);

  END IF;

END;

/

SET SERVEROUTPUT ON;

DECLARE

  PROCEDURE ok(p VARCHAR2)   IS BEGIN DBMS\_OUTPUT.PUT\_LINE('  [OK] '||p); END;

  PROCEDURE fail(p VARCHAR2) IS BEGIN DBMS\_OUTPUT.PUT\_LINE('  [FAIL] '||p); END;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('================= TEST CERINTA 12 (AUDIT + GUARD) =================');

  DBMS\_OUTPUT.PUT\_LINE('--- (0) NU ai voie sa adaugi AUDIT\_DDL in GUARD\_LDD (astept ORA-02290) ---');

  BEGIN

    INSERT INTO guard\_ldd(obj\_type, obj\_name, block\_drop) VALUES ('TABLE','AUDIT\_DDL','Y');

    fail('A permis inserarea AUDIT\_DDL in GUARD\_LDD.');

  EXCEPTION

    WHEN OTHERS THEN

      IF SQLCODE = -2290 THEN ok('Blocare corecta (check constraint): '||SQLERRM);

      ELSE fail('Eroare diferita: '||SQLERRM);

      END IF;

  END;

  ROLLBACK;

  DBMS\_OUTPUT.PUT\_LINE('--- (0b) NU ai voie sa adaugi GUARD\_LDD in GUARD\_LDD (astept ORA-02290) ---');

  BEGIN

    INSERT INTO guard\_ldd(obj\_type, obj\_name, block\_drop) VALUES ('TABLE','GUARD\_LDD','Y');

    fail('A permis inserarea GUARD\_LDD in GUARD\_LDD.');

  EXCEPTION

    WHEN OTHERS THEN

      IF SQLCODE = -2290 THEN ok('Blocare corecta (check constraint): '||SQLERRM);

      ELSE fail('Eroare diferita: '||SQLERRM);

      END IF;

  END;

  ROLLBACK;

  -- Curatam (daca exista) tabela de test

  BEGIN

    EXECUTE IMMEDIATE 'DROP TABLE t\_guard\_test PURGE';

  EXCEPTION WHEN OTHERS THEN NULL;

  END;

  DBMS\_OUTPUT.PUT\_LINE('--- (1) CREATE TABLE t\_guard\_test (trebuie sa mearga) ---');

  BEGIN

    EXECUTE IMMEDIATE 'CREATE TABLE t\_guard\_test (id NUMBER)';

    ok('CREATE TABLE reusit.');

  EXCEPTION

    WHEN OTHERS THEN fail('Nu trebuia eroare: '||SQLERRM);

  END;

  DBMS\_OUTPUT.PUT\_LINE('--- (2) Protejam tabela (block\_drop=Y) ---');

  INSERT INTO guard\_ldd(obj\_type, obj\_name, active, block\_drop, block\_truncate, block\_alter, reason)

  VALUES ('TABLE','T\_GUARD\_TEST','Y','Y','N','N','Test protectie DROP');

  COMMIT;

  ok('Regula GUARD\_LDD inserata.');

  DBMS\_OUTPUT.PUT\_LINE('--- (3) DROP TABLE protejat (astept ORA-20050) ---');

  BEGIN

    EXECUTE IMMEDIATE 'DROP TABLE t\_guard\_test PURGE';

    fail('NU trebuia sa permita DROP (era protejata).');

  EXCEPTION

    WHEN OTHERS THEN

      IF SQLCODE = -20050 THEN ok('Blocare corecta: '||SQLERRM);

      ELSE

        IF INSTR(DBMS\_UTILITY.FORMAT\_ERROR\_STACK, 'ORA-20050') > 0 THEN

        ok('Blocare corecta: ORA-20050 apare in stack (e wrapped in alte exceptii).');

        ELSE

          fail('Eroare diferita: '||SQLERRM);

          DBMS\_OUTPUT.PUT\_LINE(DBMS\_UTILITY.FORMAT\_ERROR\_STACK);

        END IF;

      END IF;

  END;

  DBMS\_OUTPUT.PUT\_LINE('--- (4) Dezactivam protectia si incercam din nou DROP (trebuie sa mearga) ---');

  UPDATE guard\_ldd

  SET active='N'

  WHERE obj\_type='TABLE' AND obj\_name='T\_GUARD\_TEST';

  COMMIT;

  BEGIN

    EXECUTE IMMEDIATE 'DROP TABLE t\_guard\_test PURGE';

    ok('DROP TABLE reusit dupa dezactivarea guard.');

  EXCEPTION

    WHEN OTHERS THEN fail('Nu trebuia eroare: '||SQLERRM);

  END;

  DBMS\_OUTPUT.PUT\_LINE('--- (5) Ultimele intrari din AUDIT\_DDL (top 10) ---');

  FOR r IN (

    SELECT id, event\_time, ddl\_event, object\_type, object\_name, db\_user

    FROM audit\_ddl

    ORDER BY id DESC

    FETCH FIRST 10 ROWS ONLY

  ) LOOP

    DBMS\_OUTPUT.PUT\_LINE('   #'||r.id||' | '||

      TO\_CHAR(r.event\_time,'YYYY-MM-DD HH24:MI:SS')||' | '||

      r.ddl\_event||' | '||r.object\_type||' '||r.object\_name||' | '||r.db\_user);

  END LOOP;

  DBMS\_OUTPUT.PUT\_LINE('--- (6) SEQUENCE + DROP ---');

  BEGIN EXECUTE IMMEDIATE 'DROP SEQUENCE seq\_guard\_test'; EXCEPTION WHEN OTHERS THEN NULL; END;

  DELETE FROM guard\_ldd WHERE obj\_type='SEQUENCE' AND obj\_name='SEQ\_GUARD\_TEST';

  EXECUTE IMMEDIATE 'CREATE SEQUENCE seq\_guard\_test START WITH 1 INCREMENT BY 1';

  INSERT INTO guard\_ldd(obj\_type, obj\_name, active, block\_drop, block\_truncate, block\_alter, reason)

  VALUES ('SEQUENCE','SEQ\_GUARD\_TEST','Y','Y','N','N','Seq DROP blocat');

  COMMIT;

  BEGIN

    EXECUTE IMMEDIATE 'DROP SEQUENCE seq\_guard\_test';

    fail('(6.a) NU trebuia sa permita DROP SEQUENCE.');

  EXCEPTION

    WHEN OTHERS THEN

      IF INSTR(DBMS\_UTILITY.FORMAT\_ERROR\_STACK,'ORA-20050')>0 THEN

        ok('(6.a) Blocare corecta (ORA-20050 in stack).');

      ELSE

        fail('(6.a) Eroare diferita: '||SQLERRM);

        DBMS\_OUTPUT.PUT\_LINE(DBMS\_UTILITY.FORMAT\_ERROR\_STACK);

      END IF;

  END;

  UPDATE guard\_ldd SET active='N'

  WHERE obj\_type='SEQUENCE' AND obj\_name='SEQ\_GUARD\_TEST';

  COMMIT;

  BEGIN

    EXECUTE IMMEDIATE 'DROP SEQUENCE seq\_guard\_test';

    ok('(6.b) DROP permis corect (active=N).');

  EXCEPTION

    WHEN OTHERS THEN fail('(6.b) DROP trebuia sa mearga: '||SQLERRM);

  END;

  EXECUTE IMMEDIATE 'CREATE SEQUENCE seq\_guard\_test START WITH 1 INCREMENT BY 1';

  UPDATE guard\_ldd SET active='Y', block\_drop='N'

  WHERE obj\_type='SEQUENCE' AND obj\_name='SEQ\_GUARD\_TEST';

  COMMIT;

  BEGIN

    EXECUTE IMMEDIATE 'DROP SEQUENCE seq\_guard\_test';

    ok('(6.c) DROP permis corect (block\_drop=N).');

  EXCEPTION

    WHEN OTHERS THEN fail('(6.c) DROP trebuia sa mearga: '||SQLERRM);

  END;

  DELETE FROM guard\_ldd WHERE obj\_type='SEQUENCE' AND obj\_name='SEQ\_GUARD\_TEST';

  COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('--- (7) SEQUENCE + ALTER ---');

  BEGIN EXECUTE IMMEDIATE 'DROP SEQUENCE seq\_guard\_alter'; EXCEPTION WHEN OTHERS THEN NULL; END;

  DELETE FROM guard\_ldd WHERE obj\_type='SEQUENCE' AND obj\_name='SEQ\_GUARD\_ALTER';

  EXECUTE IMMEDIATE 'CREATE SEQUENCE seq\_guard\_alter START WITH 1 INCREMENT BY 1';

  INSERT INTO guard\_ldd(obj\_type, obj\_name, active, block\_drop, block\_truncate, block\_alter, reason)

  VALUES ('SEQUENCE','SEQ\_GUARD\_ALTER','Y','N','N','Y','Seq ALTER blocat');

  COMMIT;

  BEGIN

    EXECUTE IMMEDIATE 'ALTER SEQUENCE seq\_guard\_alter INCREMENT BY 5';

    fail('(7.a) NU trebuia sa permita ALTER SEQUENCE.');

  EXCEPTION

    WHEN OTHERS THEN

      IF INSTR(DBMS\_UTILITY.FORMAT\_ERROR\_STACK,'ORA-20052')>0 THEN

        ok('(7.a) Blocare corecta (ORA-20052 in stack).');

      ELSE

        fail('(7.a) Eroare diferita: '||SQLERRM);

        DBMS\_OUTPUT.PUT\_LINE(DBMS\_UTILITY.FORMAT\_ERROR\_STACK);

      END IF;

  END;

  UPDATE guard\_ldd SET active='N'

  WHERE obj\_type='SEQUENCE' AND obj\_name='SEQ\_GUARD\_ALTER';

  COMMIT;

  BEGIN

    EXECUTE IMMEDIATE 'ALTER SEQUENCE seq\_guard\_alter INCREMENT BY 5';

    ok('(7.b) ALTER permis corect (active=N).');

  EXCEPTION

    WHEN OTHERS THEN fail('(7.b) ALTER trebuia sa mearga: '||SQLERRM);

  END;

  UPDATE guard\_ldd SET active='Y', block\_alter='N'

  WHERE obj\_type='SEQUENCE' AND obj\_name='SEQ\_GUARD\_ALTER';

  COMMIT;

  BEGIN

    EXECUTE IMMEDIATE 'ALTER SEQUENCE seq\_guard\_alter INCREMENT BY 2';

    ok('(7.c) ALTER permis corect (block\_alter=N).');

  EXCEPTION

    WHEN OTHERS THEN fail('(7.c) ALTER trebuia sa mearga: '||SQLERRM);

  END;

  BEGIN EXECUTE IMMEDIATE 'DROP SEQUENCE seq\_guard\_alter'; EXCEPTION WHEN OTHERS THEN NULL; END;

  DELETE FROM guard\_ldd WHERE obj\_type='SEQUENCE' AND obj\_name='SEQ\_GUARD\_ALTER';

  COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('--- (8) VIEW + DROP ---');

  BEGIN EXECUTE IMMEDIATE 'DROP VIEW v\_guard\_test'; EXCEPTION WHEN OTHERS THEN NULL; END;

  DELETE FROM guard\_ldd WHERE obj\_type='VIEW' AND obj\_name='V\_GUARD\_TEST';

  EXECUTE IMMEDIATE 'CREATE OR REPLACE VIEW v\_guard\_test AS SELECT 1 AS x FROM dual';

  INSERT INTO guard\_ldd(obj\_type, obj\_name, active, block\_drop, block\_truncate, block\_alter, reason)

  VALUES ('VIEW','V\_GUARD\_TEST','Y','Y','N','N','View DROP blocat');

  COMMIT;

  BEGIN

    EXECUTE IMMEDIATE 'DROP VIEW v\_guard\_test';

    fail('(8.a) NU trebuia sa permita DROP VIEW.');

  EXCEPTION

    WHEN OTHERS THEN

      IF INSTR(DBMS\_UTILITY.FORMAT\_ERROR\_STACK,'ORA-20050')>0 THEN

        ok('(8.a) Blocare corecta (ORA-20050 in stack).');

      ELSE

        fail('(8.a) Eroare diferita: '||SQLERRM);

        DBMS\_OUTPUT.PUT\_LINE(DBMS\_UTILITY.FORMAT\_ERROR\_STACK);

      END IF;

  END;

  UPDATE guard\_ldd SET active='N'

  WHERE obj\_type='VIEW' AND obj\_name='V\_GUARD\_TEST';

  COMMIT;

  BEGIN

    EXECUTE IMMEDIATE 'DROP VIEW v\_guard\_test';

    ok('(8.b) DROP permis corect (active=N).');

  EXCEPTION

    WHEN OTHERS THEN fail('(8.b) DROP trebuia sa mearga: '||SQLERRM);

  END;

  EXECUTE IMMEDIATE 'CREATE OR REPLACE VIEW v\_guard\_test AS SELECT 1 AS x FROM dual';

  UPDATE guard\_ldd SET active='Y', block\_drop='N'

  WHERE obj\_type='VIEW' AND obj\_name='V\_GUARD\_TEST';

  COMMIT;

  BEGIN

    EXECUTE IMMEDIATE 'DROP VIEW v\_guard\_test';

    ok('(8.c) DROP permis corect (block\_drop=N).');

  EXCEPTION

    WHEN OTHERS THEN fail('(8.c) DROP trebuia sa mearga: '||SQLERRM);

  END;

  DELETE FROM guard\_ldd WHERE obj\_type='VIEW' AND obj\_name='V\_GUARD\_TEST';

  COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('--- (5) Ultimele intrari din AUDIT\_DDL (top 20) ---');

  FOR r IN (

    SELECT id, event\_time, ddl\_event, object\_type, object\_name, db\_user

    FROM audit\_ddl

    ORDER BY id DESC

    FETCH FIRST 20 ROWS ONLY

  ) LOOP

    DBMS\_OUTPUT.PUT\_LINE('   #'||r.id||' | '||

      TO\_CHAR(r.event\_time,'YYYY-MM-DD HH24:MI:SS')||' | '||

      r.ddl\_event||' | '||r.object\_type||' '||r.object\_name||' | '||r.db\_user);

  END LOOP;

  DBMS\_OUTPUT.PUT\_LINE('================= FINAL TEST CERINTA 12 =================');

END;

/

## Print-screen:



# Cerinta 13

## **Cerinta:**

CERINTA 13 (Proiect SGBD 2025-2026)

Formulați în limbaj natural o problemă pe care să o rezolvați folosind 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).

Formulare limbaj natural:

Sa se creeze o functionalitate prin care angajatii sau oamenii care gestioneaza baza de date a restaurantului sa poata crea si receptiona comenzi in functie de produsele necesare pentru evenimente. Planificarea se face raportandu-ne la un eveniment, un orizont de timp si un buffer. Practic se analizeaza cererea la evenimentul de reper a produsului respectiv si cererea sa pe un orizont de zile dupa eveniment. Pentru siguranta logisitica, se poate mentiona un buffer, reprezentand procentul suplimentar de produse care se comanda pentru a nu fi la limita cu produsele. Se permite o singura comanda/produs, care se poate anula sau receptiona. Alte comenzi la produsul respectiv sunt conditionate de receptionarea sau anularea comenzii precedente pentru limitarea comenzilor necontrolate. Comanda face update pe stocul produselor si nu modifica nimic daca sunt destule produse.

## Rezolvare:

SET SERVEROUTPUT ON;

--------------------------------------------------------------------------------

-- PACHET: PKG\_GESTIUNE\_STOC

-- Flux: analiza -> comanda (in memorie) -> receptie (update Produse) -> raport

--------------------------------------------------------------------------------

BEGIN

  EXECUTE IMMEDIATE 'DROP PACKAGE pkg\_gestiune\_stoc';

EXCEPTION WHEN OTHERS THEN NULL;

END;

/

CREATE OR REPLACE PACKAGE pkg\_gestiune\_stoc AS

  ------------------------------------------------------------------------------

  -- Tipuri complexe: record + nested table + associative array (map)

  ------------------------------------------------------------------------------

  TYPE t\_linie\_analiza IS RECORD(

    id\_produs         Produse.id\_produs%TYPE,

    nume\_produs       Produse.nume\_produs%TYPE,

    unitate\_masura    Produse.unitate\_masura%TYPE,

    stoc\_curent       Produse.cantitate%TYPE,

    necesar\_eveniment NUMBER,

    cerere\_orizont    NUMBER,

    tinta\_stoc        NUMBER,

    deficit           NUMBER,

    recomandare       VARCHAR2(20) -- 'COMANDA' / 'NU\_COMANDA'

  );

  TYPE t\_lista\_analiza IS TABLE OF t\_linie\_analiza;

  -- o "comandă" in memorie pentru un produs

  TYPE t\_comanda\_produs IS RECORD(

    id\_produs      Produse.id\_produs%TYPE,

    nume\_produs    Produse.nume\_produs%TYPE,

    unitate\_masura Produse.unitate\_masura%TYPE,

    cantitate      NUMBER, -- cantitatea ce va fi adaugata la stoc la receptie

    id\_eveniment   Evenimente.id\_eveniment%TYPE,

    data\_evt       DATE,

    orizont\_zile   NUMBER,

    buffer\_pct     NUMBER,

    created\_at     TIMESTAMP

  );

  -- map: id\_produs -> comanda pending (1 per produs)

  TYPE t\_comenzi\_map IS TABLE OF t\_comanda\_produs INDEX BY PLS\_INTEGER;

  ------------------------------------------------------------------------------

  -- Functii

  ------------------------------------------------------------------------------

  FUNCTION f\_data\_eveniment(p\_id\_eveniment IN NUMBER) RETURN DATE;

  FUNCTION f\_cerere\_produs\_orizont(

    p\_id\_produs  IN NUMBER,

    p\_data\_start IN DATE,

    p\_data\_end   IN DATE

  ) RETURN NUMBER;

  FUNCTION f\_analiza\_eveniment(

    p\_id\_eveniment IN NUMBER,

    p\_orizont\_zile IN NUMBER DEFAULT 0,

    p\_buffer\_pct   IN NUMBER DEFAULT 0

  ) RETURN t\_lista\_analiza;

  ------------------------------------------------------------------------------

  -- Proceduri (flux)

  ------------------------------------------------------------------------------

  PROCEDURE p\_genereaza\_comenzi(

    p\_id\_eveniment IN NUMBER,

    p\_orizont\_zile IN NUMBER DEFAULT 0,

    p\_buffer\_pct   IN NUMBER DEFAULT 0

  );

  PROCEDURE p\_raport\_eveniment(

    p\_id\_eveniment IN NUMBER,

    p\_orizont\_zile IN NUMBER DEFAULT 0,

    p\_buffer\_pct   IN NUMBER DEFAULT 0

  );

  PROCEDURE p\_raport\_comenzi\_curente;

  -- receptie/anulare per produs

  PROCEDURE p\_receptioneaza\_comanda\_produs(p\_id\_produs IN NUMBER);

  PROCEDURE p\_anuleaza\_comanda\_produs(p\_id\_produs IN NUMBER);

  -- utilitare: receptie/anulare pentru toate comenzile pending

  PROCEDURE p\_receptioneaza\_toate;

  PROCEDURE p\_anuleaza\_toate;

END pkg\_gestiune\_stoc;

/

SHOW ERRORS;

CREATE OR REPLACE PACKAGE BODY pkg\_gestiune\_stoc AS

  -- stare in memorie (pe sesiune): comenzi multiple, cheie = id\_produs

  g\_comenzi t\_comenzi\_map;

  PROCEDURE validate\_params(p\_id\_eveniment NUMBER, p\_orizont\_zile NUMBER, p\_buffer\_pct NUMBER) IS

  BEGIN

    IF p\_id\_eveniment IS NULL OR p\_orizont\_zile IS NULL OR p\_buffer\_pct IS NULL

       OR p\_orizont\_zile < 0 OR p\_buffer\_pct < 0 OR p\_buffer\_pct > 100 THEN

      RAISE\_APPLICATION\_ERROR(-20102, 'Parametri invalizi (orizont>=0, buffer 0..100).');

    END IF;

  END;

  FUNCTION f\_data\_eveniment(p\_id\_eveniment IN NUMBER) RETURN DATE IS

    v\_d DATE;

  BEGIN

    SELECT data\_eveniment INTO v\_d

    FROM Evenimente

    WHERE id\_eveniment = p\_id\_eveniment;

    RETURN v\_d;

  EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

      RAISE\_APPLICATION\_ERROR(-20100, 'Eveniment inexistent (id\_eveniment='||p\_id\_eveniment||').');

  END;

  FUNCTION f\_cerere\_produs\_orizont(

    p\_id\_produs  IN NUMBER,

    p\_data\_start IN DATE,

    p\_data\_end   IN DATE

  ) RETURN NUMBER IS

    v\_sum NUMBER;

  BEGIN

    SELECT NVL(SUM(ep.cantitate), 0)

    INTO   v\_sum

    FROM   Evenimente\_Produse ep

    JOIN   Evenimente e ON e.id\_eveniment = ep.id\_eveniment

    WHERE  ep.id\_produs = p\_id\_produs

      AND  TRUNC(e.data\_eveniment) >= TRUNC(p\_data\_start)

      AND  TRUNC(e.data\_eveniment) <= TRUNC(p\_data\_end);

    RETURN v\_sum;

  END;

  FUNCTION f\_analiza\_eveniment(

    p\_id\_eveniment IN NUMBER,

    p\_orizont\_zile IN NUMBER DEFAULT 0,

    p\_buffer\_pct   IN NUMBER DEFAULT 0

  ) RETURN t\_lista\_analiza IS

    v\_data\_evt DATE;

    v\_list t\_lista\_analiza := t\_lista\_analiza();

    v\_cnt NUMBER;

  BEGIN

    validate\_params(p\_id\_eveniment, p\_orizont\_zile, p\_buffer\_pct);

    v\_data\_evt := f\_data\_eveniment(p\_id\_eveniment);

    SELECT COUNT(\*) INTO v\_cnt

    FROM Evenimente\_Produse

    WHERE id\_eveniment = p\_id\_eveniment;

    IF v\_cnt = 0 THEN

      RAISE\_APPLICATION\_ERROR(-20101, 'Eveniment fara produse asociate (Evenimente\_Produse).');

    END IF;

    SELECT

      p.id\_produs,

      p.nume\_produs,

      p.unitate\_masura,

      p.cantitate AS stoc\_curent,

      ep.cantitate AS necesar\_eveniment,

      f\_cerere\_produs\_orizont(p.id\_produs, v\_data\_evt, v\_data\_evt + p\_orizont\_zile) AS cerere\_orizont,

      CAST(NULL AS NUMBER) AS tinta\_stoc,

      CAST(NULL AS NUMBER) AS deficit,

      CAST(NULL AS VARCHAR2(20)) AS recomandare

    BULK COLLECT INTO v\_list

    FROM Evenimente\_Produse ep

    JOIN Produse p ON p.id\_produs = ep.id\_produs

    WHERE ep.id\_eveniment = p\_id\_eveniment

    ORDER BY p.id\_produs;

    FOR i IN 1 .. v\_list.COUNT LOOP

      v\_list(i).tinta\_stoc :=

        CEIL(

          GREATEST(v\_list(i).necesar\_eveniment, v\_list(i).cerere\_orizont)

          \* (1 + p\_buffer\_pct/100)

        );

      v\_list(i).deficit := v\_list(i).tinta\_stoc - v\_list(i).stoc\_curent;

      IF v\_list(i).deficit > 0 THEN

        v\_list(i).recomandare := 'COMANDA';

      ELSE

        v\_list(i).deficit := 0;

        v\_list(i).recomandare := 'NU\_COMANDA';

      END IF;

    END LOOP;

    RETURN v\_list;

  END;

  PROCEDURE p\_raport\_eveniment(

    p\_id\_eveniment IN NUMBER,

    p\_orizont\_zile IN NUMBER DEFAULT 0,

    p\_buffer\_pct   IN NUMBER DEFAULT 0

  ) IS

    v\_list t\_lista\_analiza;

    v\_data DATE;

  BEGIN

    v\_data := f\_data\_eveniment(p\_id\_eveniment);

    v\_list := f\_analiza\_eveniment(p\_id\_eveniment, p\_orizont\_zile, p\_buffer\_pct);

    DBMS\_OUTPUT.PUT\_LINE('============================================================');

    DBMS\_OUTPUT.PUT\_LINE('RAPORT STOC - Eveniment #'||p\_id\_eveniment||

                         ' | Data='||TO\_CHAR(v\_data,'YYYY-MM-DD')||

                         ' | Orizont='||p\_orizont\_zile||' | Buffer='||p\_buffer\_pct||'%');

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

    FOR i IN 1 .. v\_list.COUNT LOOP

      DBMS\_OUTPUT.PUT\_LINE(

        'Produs #'||v\_list(i).id\_produs||' '||RPAD(v\_list(i).nume\_produs, 20)||

        ' stoc='||TO\_CHAR(v\_list(i).stoc\_curent)||

        ' necesar\_evt='||TO\_CHAR(v\_list(i).necesar\_eveniment)||

        ' cerere\_oriz='||TO\_CHAR(v\_list(i).cerere\_orizont)||

        ' tinta='||TO\_CHAR(v\_list(i).tinta\_stoc)||

        ' deficit='||TO\_CHAR(v\_list(i).deficit)||

        ' -> '||v\_list(i).recomandare

      );

    END LOOP;

    DBMS\_OUTPUT.PUT\_LINE('============================================================');

  EXCEPTION

    WHEN OTHERS THEN

      DBMS\_OUTPUT.PUT\_LINE('[RAPORT] '||SQLERRM);

  END;

  PROCEDURE p\_genereaza\_comenzi(

    p\_id\_eveniment IN NUMBER,

    p\_orizont\_zile IN NUMBER DEFAULT 0,

    p\_buffer\_pct   IN NUMBER DEFAULT 0

  ) IS

    v\_list t\_lista\_analiza;

    v\_data\_evt DATE;

    v\_has\_deficit BOOLEAN := FALSE;

    v\_nr\_create PLS\_INTEGER := 0;

    v\_nr\_ignorate PLS\_INTEGER := 0;

  BEGIN

    validate\_params(p\_id\_eveniment, p\_orizont\_zile, p\_buffer\_pct);

    v\_data\_evt := f\_data\_eveniment(p\_id\_eveniment);

    v\_list := f\_analiza\_eveniment(p\_id\_eveniment, p\_orizont\_zile, p\_buffer\_pct);

    -- daca nu exista niciun deficit deloc -> comanda inutila

    FOR i IN 1 .. v\_list.COUNT LOOP

      IF v\_list(i).deficit > 0 THEN

        v\_has\_deficit := TRUE;

        EXIT;

      END IF;

    END LOOP;

    IF NOT v\_has\_deficit THEN

      RAISE\_APPLICATION\_ERROR(-20104, 'Stoc suficient raportat la necesar/orizont. Comanda inutila.');

    END IF;

    -- Adaugam comenzi doar pentru produsele fara comanda pending; restul se ignora (mesaj)

    FOR i IN 1 .. v\_list.COUNT LOOP

      IF v\_list(i).deficit > 0 THEN

        IF g\_comenzi.EXISTS(v\_list(i).id\_produs) THEN

          v\_nr\_ignorate := v\_nr\_ignorate + 1;

          DBMS\_OUTPUT.PUT\_LINE(

            '[INFO] Produs #'||v\_list(i).id\_produs||' ('||v\_list(i).nume\_produs||') '||

            'are deja comanda pending. Pe analiza curenta (orizont='||p\_orizont\_zile||

            ', buffer='||p\_buffer\_pct||'%) deficit='||v\_list(i).deficit||' -> IGNORAT.'

          );

        ELSE

          g\_comenzi(v\_list(i).id\_produs).id\_produs      := v\_list(i).id\_produs;

          g\_comenzi(v\_list(i).id\_produs).nume\_produs    := v\_list(i).nume\_produs;

          g\_comenzi(v\_list(i).id\_produs).unitate\_masura := v\_list(i).unitate\_masura;

          g\_comenzi(v\_list(i).id\_produs).cantitate      := v\_list(i).deficit;

          g\_comenzi(v\_list(i).id\_produs).id\_eveniment   := p\_id\_eveniment;

          g\_comenzi(v\_list(i).id\_produs).data\_evt       := v\_data\_evt;

          g\_comenzi(v\_list(i).id\_produs).orizont\_zile   := p\_orizont\_zile;

          g\_comenzi(v\_list(i).id\_produs).buffer\_pct     := p\_buffer\_pct;

          g\_comenzi(v\_list(i).id\_produs).created\_at     := SYSTIMESTAMP;

          v\_nr\_create := v\_nr\_create + 1;

        END IF;

      END IF;

    END LOOP;

    IF v\_nr\_create = 0 THEN

      DBMS\_OUTPUT.PUT\_LINE(

        '[COMENZI] Nu s-a adaugat nicio comanda noua pentru eveniment #'||p\_id\_eveniment||

        '. Produse cu deficit ignorate (aveau deja comanda): '||v\_nr\_ignorate||'.'

      );

      -- fluxul e ok, doar nu e nimic de adaugat

      RETURN;

    END IF;

    DBMS\_OUTPUT.PUT\_LINE(

      '[COMENZI] Adaugate '||v\_nr\_create||' comenzi noi in memorie pentru eveniment #'||p\_id\_eveniment||

      ' (orizont='||p\_orizont\_zile||', buffer='||p\_buffer\_pct||'%). '||

      'Ignorate (deja pending): '||v\_nr\_ignorate||'.'

    );

    p\_raport\_comenzi\_curente;

  END;

  PROCEDURE p\_raport\_comenzi\_curente IS

    k PLS\_INTEGER;

    v\_cnt PLS\_INTEGER := 0;

  BEGIN

    IF g\_comenzi.COUNT = 0 THEN

      DBMS\_OUTPUT.PUT\_LINE('[COMENZI] (nu exista comenzi pending in memorie)');

      RETURN;

    END IF;

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

    DBMS\_OUTPUT.PUT\_LINE('COMENZI PENDING (1 / produs): '||g\_comenzi.COUNT);

    k := g\_comenzi.FIRST;

    WHILE k IS NOT NULL LOOP

      v\_cnt := v\_cnt + 1;

      DBMS\_OUTPUT.PUT\_LINE(

        '  ['||v\_cnt||'] Produs #'||g\_comenzi(k).id\_produs||' '||RPAD(g\_comenzi(k).nume\_produs, 20)||

        ' cant='||TO\_CHAR(g\_comenzi(k).cantitate)||' '||g\_comenzi(k).unitate\_masura||

        ' | eveniment='||g\_comenzi(k).id\_eveniment||

        ' | data\_evt='||TO\_CHAR(g\_comenzi(k).data\_evt,'YYYY-MM-DD')||

        ' | orizont='||g\_comenzi(k).orizont\_zile||' | buffer='||g\_comenzi(k).buffer\_pct||'%'

      );

      k := g\_comenzi.NEXT(k);

    END LOOP;

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

  END;

  PROCEDURE p\_receptioneaza\_comanda\_produs(p\_id\_produs IN NUMBER) IS

    v\_q NUMBER;

    v\_rows NUMBER;

  BEGIN

    IF p\_id\_produs IS NULL THEN

      RAISE\_APPLICATION\_ERROR(-20102, 'Parametru invalid: id\_produs NULL.');

    END IF;

    IF NOT g\_comenzi.EXISTS(p\_id\_produs) THEN

      RAISE\_APPLICATION\_ERROR(-20105, 'Nu exista comanda pending in memorie pentru produs #'||p\_id\_produs||'.');

    END IF;

    v\_q := g\_comenzi(p\_id\_produs).cantitate;

    UPDATE Produse

    SET cantitate = cantitate + v\_q

    WHERE id\_produs = p\_id\_produs;

    v\_rows := SQL%ROWCOUNT;

    IF v\_rows = 0 THEN

      RAISE\_APPLICATION\_ERROR(-20110, 'Produs inexistent in tabela Produse (id\_produs='||p\_id\_produs||').');

    END IF;

    DBMS\_OUTPUT.PUT\_LINE('[RECEPTIE] Produs #'||p\_id\_produs||' + '||v\_q||

                         ' (eveniment='||g\_comenzi(p\_id\_produs).id\_eveniment||'). Stoc actualizat.');

    g\_comenzi.DELETE(p\_id\_produs);

  END;

  PROCEDURE p\_anuleaza\_comanda\_produs(p\_id\_produs IN NUMBER) IS

  BEGIN

    IF p\_id\_produs IS NULL THEN

      RAISE\_APPLICATION\_ERROR(-20102, 'Parametru invalid: id\_produs NULL.');

    END IF;

    IF g\_comenzi.EXISTS(p\_id\_produs) THEN

      DBMS\_OUTPUT.PUT\_LINE('[ANULARE] S-a anulat comanda pending pentru produs #'||p\_id\_produs||

                           ' (eveniment='||g\_comenzi(p\_id\_produs).id\_eveniment||').');

      g\_comenzi.DELETE(p\_id\_produs);

    ELSE

      DBMS\_OUTPUT.PUT\_LINE('[ANULARE] Nu exista comanda pending pentru produs #'||p\_id\_produs||'.');

    END IF;

  END;

  PROCEDURE p\_receptioneaza\_toate IS

    k PLS\_INTEGER;

  BEGIN

    IF g\_comenzi.COUNT = 0 THEN

      DBMS\_OUTPUT.PUT\_LINE('[RECEPTIE] Nu exista comenzi pending.');

      RETURN;

    END IF;

    k := g\_comenzi.FIRST;

    WHILE k IS NOT NULL LOOP

      -- salvam NEXT inainte de delete

      DECLARE nxt PLS\_INTEGER;

      BEGIN

        nxt := g\_comenzi.NEXT(k);

        p\_receptioneaza\_comanda\_produs(k);

        k := nxt;

      END;

    END LOOP;

  END;

  PROCEDURE p\_anuleaza\_toate IS

  BEGIN

    g\_comenzi.DELETE;

    DBMS\_OUTPUT.PUT\_LINE('[ANULARE] Toate comenzile pending au fost anulate.');

  END;

END pkg\_gestiune\_stoc;

/

SHOW ERRORS;

----------------

-- TESTE

----------------

SET SERVEROUTPUT ON;

DECLARE

  v\_tip Evenimente.tip\_eveniment%TYPE;

  -- Produse

  v\_pA NUMBER; v\_pB NUMBER; v\_pC NUMBER; v\_pD NUMBER; v\_pE NUMBER;

  -- Evenimente

  v\_e1 NUMBER; v\_e2 NUMBER; v\_e3 NUMBER; v\_e4 NUMBER; v\_e5 NUMBER; v\_e6 NUMBER;

  v\_tag VARCHAR2(50) := TO\_CHAR(SYSTIMESTAMP,'YYYYMMDDHH24MISSFF3');

  PROCEDURE ok(p VARCHAR2) IS BEGIN DBMS\_OUTPUT.PUT\_LINE('  [OK] '||p); END;

  PROCEDURE fail(p VARCHAR2) IS BEGIN DBMS\_OUTPUT.PUT\_LINE('  [FAIL] '||p); END;

  FUNCTION stoc(p\_id NUMBER) RETURN NUMBER IS v NUMBER;

  BEGIN SELECT cantitate INTO v FROM Produse WHERE id\_produs = p\_id; RETURN v; END;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('===================== TESTE =====================');

  SAVEPOINT sp\_all;

  -- tip\_eveniment existent

  BEGIN

    SELECT tip\_eveniment INTO v\_tip FROM Evenimente WHERE ROWNUM = 1;

  EXCEPTION WHEN NO\_DATA\_FOUND THEN

    v\_tip := 'TEST';

  END;

  -- Produse de test

  INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar)

  VALUES ('T\_Prod\_A\_'||v\_tag, 100, 'buc', 5) RETURNING id\_produs INTO v\_pA;

  INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar)

  VALUES ('T\_Prod\_B\_'||v\_tag, 2, 'buc', 10) RETURNING id\_produs INTO v\_pB;

  INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar)

  VALUES ('T\_Prod\_C\_'||v\_tag, 0, 'buc', 7) RETURNING id\_produs INTO v\_pC;

  ok('Produse create: A='||v\_pA||', B='||v\_pB||', C='||v\_pC);

  -------------------------------------------------------------------

  -- (1) Stoc suficient -> -20104

  -------------------------------------------------------------------

  DBMS\_OUTPUT.PUT\_LINE('--- (1) Stoc suficient => astept -20104 ---');

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES (v\_tip, DATE '2032-01-10', 10) RETURNING id\_eveniment INTO v\_e1;

  INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate)

  VALUES (v\_e1, v\_pA, 10);

  BEGIN

    pkg\_gestiune\_stoc.p\_genereaza\_comenzi(v\_e1, 0, 0);

    fail('Nu a blocat comanda desi stoc suficient.');

  EXCEPTION

    WHEN OTHERS THEN

      IF SQLCODE = -20104 THEN ok('Blocare corecta: '||SQLERRM);

      ELSE fail('Alta eroare: '||SQLERRM);

      END IF;

  END;

  -------------------------------------------------------------------

  -- (2) Deficit produs B -> creeaza comanda pending pentru B

  -------------------------------------------------------------------

  DBMS\_OUTPUT.PUT\_LINE('--- (2) Deficit B => creeaza comanda pending B ---');

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES (v\_tip, DATE '2032-01-12', 10) RETURNING id\_eveniment INTO v\_e2;

  INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate)

  VALUES (v\_e2, v\_pB, 10);

  pkg\_gestiune\_stoc.p\_genereaza\_comenzi(v\_e2, 0, 0);

  ok('B pending creat. Stoc B inainte='||stoc(v\_pB));

  -------------------------------------------------------------------

  -- (3) Cu B pending, deficit C -> permite comanda C in paralel

  -------------------------------------------------------------------

  DBMS\_OUTPUT.PUT\_LINE('--- (3) B pending + deficit C => creeaza si C ---');

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES (v\_tip, DATE '2032-01-13', 10) RETURNING id\_eveniment INTO v\_e3;

  INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate)

  VALUES (v\_e3, v\_pC, 5);

  pkg\_gestiune\_stoc.p\_genereaza\_comenzi(v\_e3, 0, 0);

  ok('C pending creat in paralel cu B.');

  pkg\_gestiune\_stoc.p\_raport\_comenzi\_curente;

  -------------------------------------------------------------------

  -- (4) daca produsul B are deja comanda, IGNORA B si creeaza comenzi pentru alte produse din eveniment.

  --     Cream produs D deficit + eveniment cu (B + D).

  -------------------------------------------------------------------

  DBMS\_OUTPUT.PUT\_LINE('--- (4) Eveniment cu (B deja pending) + D deficit => IGNORA B, creeaza D ---');

  INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar)

  VALUES ('T\_Prod\_D\_'||v\_tag, 0, 'buc', 3) RETURNING id\_produs INTO v\_pD;

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES (v\_tip, DATE '2032-01-14', 10) RETURNING id\_eveniment INTO v\_e4;

  INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (v\_e4, v\_pB, 3);

  INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (v\_e4, v\_pD, 4);

  -- AICI trebuie sa NU dea eroare

  BEGIN

    pkg\_gestiune\_stoc.p\_genereaza\_comenzi(v\_e4, 0, 0);

    ok('A rulat fara eroare: B ignorat, D adaugat.');

  EXCEPTION

    WHEN OTHERS THEN

      fail('Nu trebuia sa dea eroare: '||SQLERRM);

  END;

  pkg\_gestiune\_stoc.p\_raport\_comenzi\_curente;

  -------------------------------------------------------------------

  -- (5) Eveniment doar cu B (B pending, deficit) => NU creeaza nimic nou

  -------------------------------------------------------------------

  DBMS\_OUTPUT.PUT\_LINE('--- (5) Eveniment doar cu B pending => nu creeaza nimic nou ---');

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES (v\_tip, DATE '2032-01-15', 10) RETURNING id\_eveniment INTO v\_e5;

  INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (v\_e5, v\_pB, 4);

  BEGIN

    pkg\_gestiune\_stoc.p\_genereaza\_comenzi(v\_e5, 0, 0);

    ok('A rulat: B ignorat, nicio comanda noua (corect).');

  EXCEPTION

    WHEN OTHERS THEN

      fail('Nu trebuia eroare aici: '||SQLERRM);

  END;

  -------------------------------------------------------------------

  -- (6) Receptie pe toate si verificare stocuri

  -------------------------------------------------------------------

  DBMS\_OUTPUT.PUT\_LINE('--- (6) Receptie pe toate (B, C, D) + verificare stoc ---');

  DBMS\_OUTPUT.PUT\_LINE('  Stoc inainte: B='||stoc(v\_pB)||', C='||stoc(v\_pC)||', D='||stoc(v\_pD));

  pkg\_gestiune\_stoc.p\_receptioneaza\_toate;

  DBMS\_OUTPUT.PUT\_LINE('  Stoc dupa:    B='||stoc(v\_pB)||', C='||stoc(v\_pC)||', D='||stoc(v\_pD));

  pkg\_gestiune\_stoc.p\_raport\_comenzi\_curente;

  -------------------------------------------------------------------

  -- (7) TEST 7: 2 comenzi pe 2 produse diferite + buffering

  --     - una pe orizont+buffer

  --     - una pe eveniment: initial fara buffer, anulata, refacuta cu buffer

  --     - apoi receptie pe toate

  -------------------------------------------------------------------

  DBMS\_OUTPUT.PUT\_LINE('--- (7) 2 comenzi pe 2 produse diferite + buffering + receptie toate ---');

  -- curatam memoria

  pkg\_gestiune\_stoc.p\_anuleaza\_toate;

  -- Produs A2 (orizont): stoc 50, cerere orizont 100, buffer 10% => tinta 110 => deficit 60

  INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar)

  VALUES ('T7\_Prod\_A2\_'||v\_tag, 50, 'buc', 5) RETURNING id\_produs INTO v\_pA;

  -- Produs E (eveniment): stoc 0, necesar 10, refacut cu buffer 20% => tinta 12 => deficit 12

  INSERT INTO Produse (nume\_produs, cantitate, unitate\_masura, pret\_unitar)

  VALUES ('T7\_Prod\_E\_'||v\_tag, 0, 'buc', 9) RETURNING id\_produs INTO v\_pE;

  -- Eveniment initiator orizont

  INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

  VALUES (v\_tip, DATE '2033-02-01', 10) RETURNING id\_eveniment INTO v\_e6;

  INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (v\_e6, v\_pA, 40);

  -- Eveniment suplimentar in orizont (+14 zile)

  DECLARE v\_e6b NUMBER;

  BEGIN

    INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

    VALUES (v\_tip, DATE '2033-02-15', 10) RETURNING id\_eveniment INTO v\_e6b;

    INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (v\_e6b, v\_pA, 60);

  END;

  DBMS\_OUTPUT.PUT\_LINE('  (7.1) Generez ORIZONT+BUFFER (orizont=30, buffer=10) pentru A2');

  pkg\_gestiune\_stoc.p\_genereaza\_comenzi(v\_e6, 30, 10);

  -- Eveniment pentru produs E

  DECLARE v\_eE NUMBER;

  BEGIN

    INSERT INTO Evenimente (tip\_eveniment, data\_eveniment, numar\_persoane)

    VALUES (v\_tip, DATE '2033-03-01', 10) RETURNING id\_eveniment INTO v\_eE;

    INSERT INTO Evenimente\_Produse (id\_eveniment, id\_produs, cantitate) VALUES (v\_eE, v\_pE, 10);

    DBMS\_OUTPUT.PUT\_LINE('  (7.2) Generez EVENIMENT (buffer=0) pentru E');

    pkg\_gestiune\_stoc.p\_genereaza\_comenzi(v\_eE, 0, 0);

    DBMS\_OUTPUT.PUT\_LINE('  (7.3) Anulez comanda E si regenerez cu buffer=20');

    pkg\_gestiune\_stoc.p\_anuleaza\_comanda\_produs(v\_pE);

    pkg\_gestiune\_stoc.p\_genereaza\_comenzi(v\_eE, 0, 20);

  END;

  DBMS\_OUTPUT.PUT\_LINE('  (7.4) Comenzi pending (ambele trebuie sa arate buffer in detalii):');

  pkg\_gestiune\_stoc.p\_raport\_comenzi\_curente;

  DBMS\_OUTPUT.PUT\_LINE('  (7.5) Receptie pe toate separat + verificare stoc:');

  DBMS\_OUTPUT.PUT\_LINE('    Stoc inainte: A2='||stoc(v\_pA)||', E='||stoc(v\_pE));

  pkg\_gestiune\_stoc.p\_receptioneaza\_comanda\_produs(v\_pE);

  pkg\_gestiune\_stoc.p\_receptioneaza\_comanda\_produs(v\_pA);

  DBMS\_OUTPUT.PUT\_LINE('    Stoc dupa:    A2='||stoc(v\_pA)||', E='||stoc(v\_pE));

  pkg\_gestiune\_stoc.p\_raport\_comenzi\_curente;

  -- curatenie

  pkg\_gestiune\_stoc.p\_anuleaza\_toate;

  ROLLBACK TO sp\_all;

  DBMS\_OUTPUT.PUT\_LINE('===================== FINAL TESTE (rollback) =====================');

EXCEPTION

  WHEN OTHERS THEN

    fail('Eroare neasteptata in teste: '||SQLERRM);

    BEGIN pkg\_gestiune\_stoc.p\_anuleaza\_toate; EXCEPTION WHEN OTHERS THEN NULL; END;

    ROLLBACK TO sp\_all;

END;

/

## Print-screen:

