Sisteme de Gestiune a Bazelor de Date

Proiect

Sistem de Gestiune a Restaurării Patrimoniului Cultural

Student: Voaideș-Negustor Robert-Ionuț
Grupa: 251
Seria: 25
Anul Universitar: 2025-2026

Cuprins

Introducere

Infrastructura utilizata

- **Sistem de operare:** Fedora Linux
- **Containerizare:** Docker
- **SGBD:** Oracle Database XE 21c (container `oracle-xe`, imagine `gvenzl/oracle-xe:21-slim`)
- **Mediu de lucru:** Visual Studio Code și DataGrip

1. Descrierea Bazei de Date

1.1 Scenariul Real Modelat

Această bază de date a fost proiectată pentru a gestiona activitatea unei instituții sau companii specializate în conservarea și restaurarea patrimoniului istoric. Scenariul real modelat urmărește ciclul de viață complet al unui proiect de restaurare, de la identificarea obiectivului (monumentul) și asigurarea surselor de finanțare, până la execuția efectivă și monitorizarea calității lucrărilor.

Baza de date surprinde interacțiunile complexe dintre diversele entități implicate într-un șantier de restaurare: monumentele istorice, echipele multidisciplinare de experți, resursele materiale necesare, instituțiile statului care avizează lucrările și sursele de finanțare.

1.2 Utilitatea Bazei de Date

Implementarea acestui sistem informatic răspunde nevoii de a avea o evidență clară, transparentă și centralizată asupra modului în care este conservat patrimoniul. Principalele funcționalități și beneficii sunt:

- **Trasabilitatea intervențiilor:** Sistemul permite crearea unui istoric detaliat al tuturor restaurărilor efectuate asupra unui monument. Se poate verifica oricând ce materiale s-au utilizat și ce experți au participat, informații vitale pentru viitoarele lucrări de întreținere.
- **Gestionarea resurselor:** Baza de date rezolvă problema alocării eficiente a resurselor. Deoarece experții și stocurile de materiale pot fi partajate între mai multe șantiere simultan, sistemul evidențiază clar distribuția acestora.
- **Monitorizarea financiară și legală:** Permite urmărirea bugetelor alocate din diverse surse de finanțare, precum și starea avizelor necesare de la autoritățile competente (ex: Ministerul Culturii, Primărie) și rezultatele inspecțiilor de șantier.

2. Diagrama Entitate-Relație (ERD)

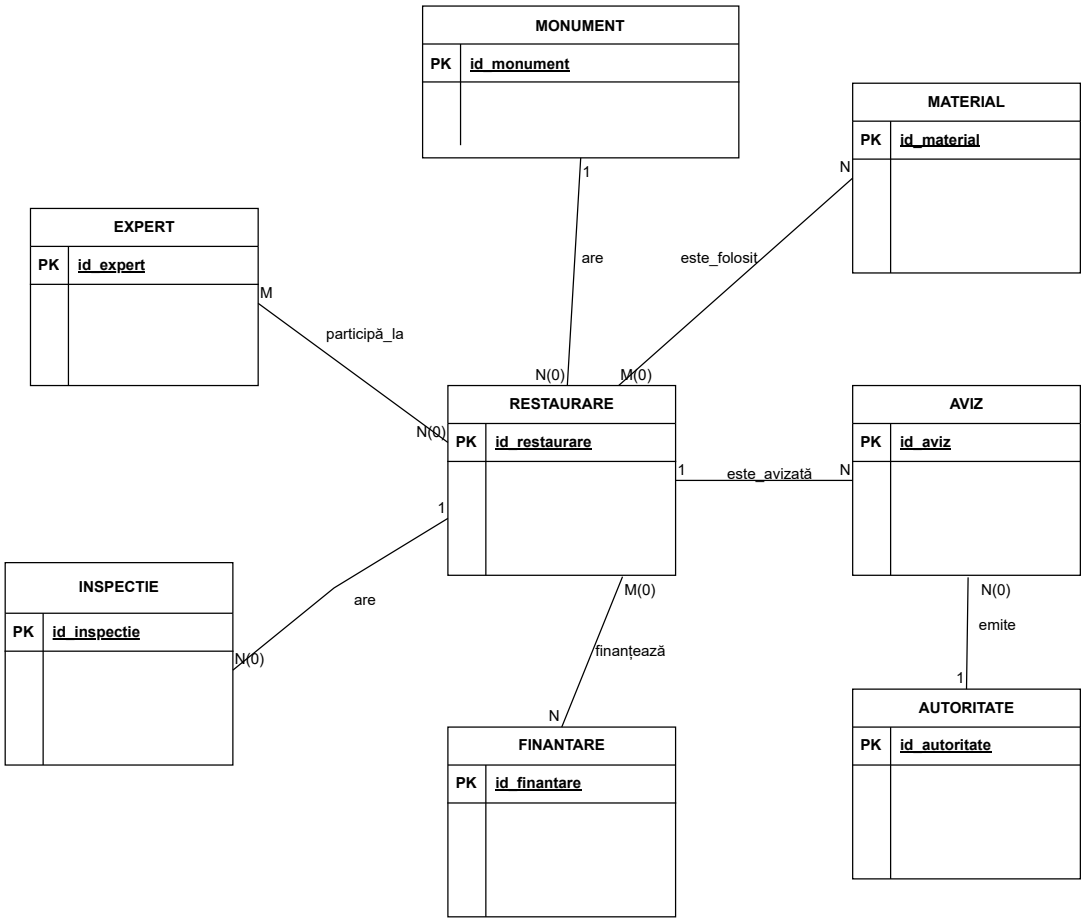

Figura 2.1: Diagrama ERD a bazei de date

3. Diagrama Conceptuală

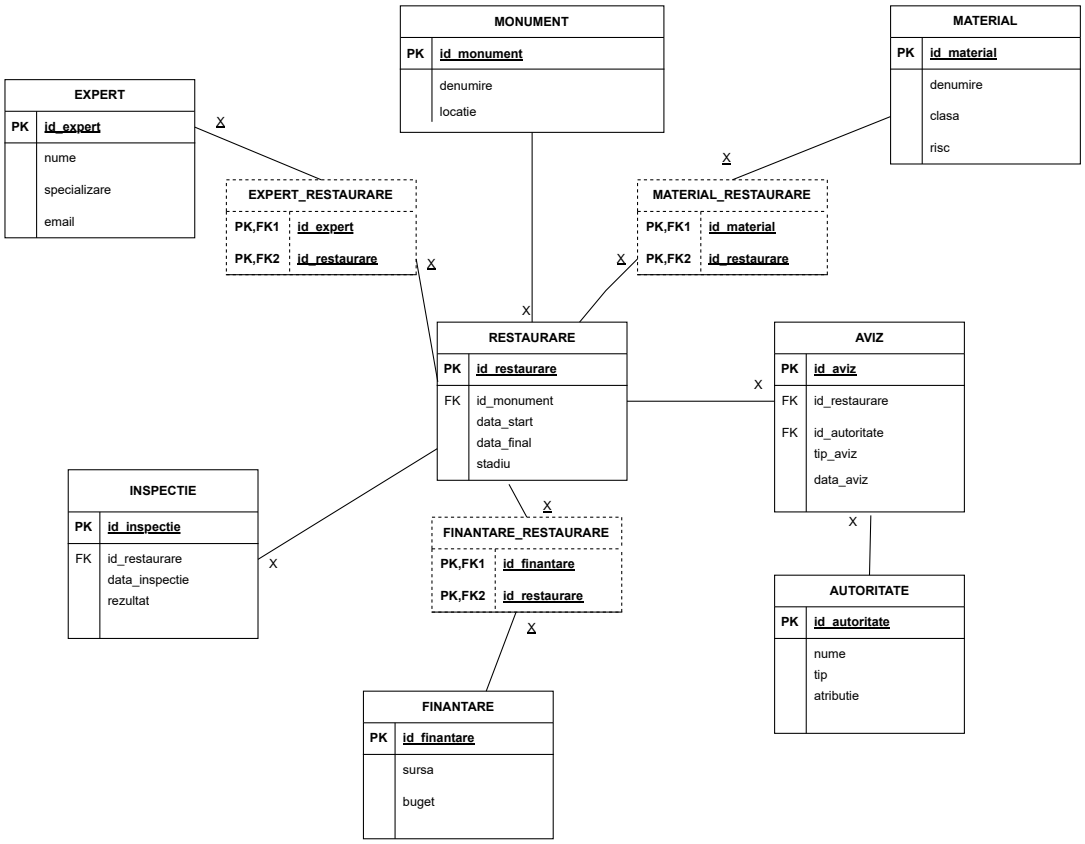

Figura 3.1: Diagrama Conceptuală

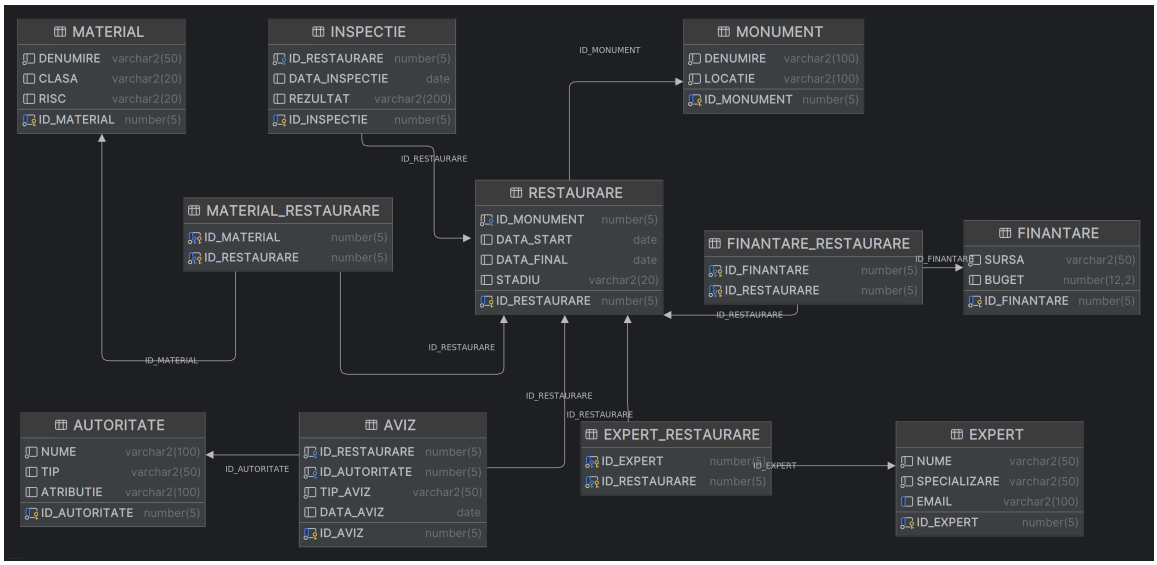

Figura 3.2: Diagrama Conceptuală generată de DataGrip

4. Implementarea Structurii

Mai jos sunt prezentate instrucțiunile de creare a tabelelor și constrângerile de integritate.

```
CREATE TABLE monument (
    id_monument NUMBER(5) CONSTRAINT pk_monument PRIMARY KEY,
    denumire    VARCHAR2(100) NOT NULL,
    locatie     VARCHAR2(100) NOT NULL
);


CREATE TABLE expert (
    id_expert    NUMBER(5) CONSTRAINT pk_expert PRIMARY KEY,
    nume         VARCHAR2(50) NOT NULL,
    specializare VARCHAR2(50) NOT NULL,
    email        VARCHAR2(100) CONSTRAINT uq_expert_email UNIQUE
);


CREATE TABLE material (
    id_material NUMBER(5) CONSTRAINT pk_material PRIMARY KEY,
    denumire    VARCHAR2(50) NOT NULL,
    clasa       VARCHAR2(20),
    risc        VARCHAR2(20)
);


CREATE TABLE finantare (
    id_finantare NUMBER(5) CONSTRAINT pk_finantare PRIMARY KEY,
    sursa        VARCHAR2(50) NOT NULL,
    buget        NUMBER(12, 2) CONSTRAINT ck_buget_pozitiv CHECK (buget
   > 0)
);


CREATE TABLE autoritate (
    id_autoritate NUMBER(5) CONSTRAINT pk_autoritate PRIMARY KEY,
    nume          VARCHAR2(100) NOT NULL,
    tip           VARCHAR2(50),
    atributie     VARCHAR2(100)
);

CREATE TABLE restaurare (
    id_restaurare NUMBER(5) CONSTRAINT pk_restaurare PRIMARY KEY,
    id_monument   NUMBER(5) NOT NULL,
    data_start    DATE DEFAULT SYSDATE,
    data_final    DATE,
    stadiu        VARCHAR2(20) CONSTRAINT ck_stadiu CHECK (stadiu IN ('
   Planificat', 'In executie', 'Finalizat', 'Suspendat')),
    CONSTRAINT fk_restaurare_monument FOREIGN KEY (id_monument)
   REFERENCES monument(id_monument),
    CONSTRAINT ck_date_valide CHECK (data_final >= data_start)
);


CREATE TABLE aviz (
    id_aviz       NUMBER(5) CONSTRAINT pk_aviz PRIMARY KEY,
    id_restaurare NUMBER(5) NOT NULL,
```

```
52      id_autoritate NUMBER(5) NOT NULL,
53      tip_aviz      VARCHAR2(50) NOT NULL,
54      data_aviz     DATE DEFAULT SYSDATE,
55      CONSTRAINT fk_aviz_restaurare FOREIGN KEY (id_restaurare) REFERENCES
        restaurare(id_restaurare),
56      CONSTRAINT fk_aviz_autoritate FOREIGN KEY (id_autoritate) REFERENCES
        autoritate(id_autoritate)
57  );
58
59
60  CREATE TABLE inspectie (
61      id_inspectie   NUMBER(5) CONSTRAINT pk_inspectie PRIMARY KEY,
62      id_restaurare  NUMBER(5) NOT NULL,
63      data_inspectie DATE DEFAULT SYSDATE,
64      rezultat       VARCHAR2(200),
65      CONSTRAINT fk_inspectie_restaurare FOREIGN KEY (id_restaurare)
       REFERENCES restaurare(id_restaurare)
66  );
67
68
69  CREATE TABLE expert_restaurare (
70      id_expert     NUMBER(5),
71      id_restaurare NUMBER(5),
72      CONSTRAINT pk_expert_restaurare PRIMARY KEY (id_expert,
       id_restaurare),
73      CONSTRAINT fk_er_expert FOREIGN KEY (id_expert) REFERENCES expert(
       id_expert),
74      CONSTRAINT fk_er_restaurare FOREIGN KEY (id_restaurare) REFERENCES
       restaurare(id_restaurare)
75  );
76
77
78  CREATE TABLE material_restaurare (
79      id_material   NUMBER(5),
80      id_restaurare NUMBER(5),
81      CONSTRAINT pk_material_restaurare PRIMARY KEY (id_material,
       id_restaurare),
82      CONSTRAINT fk_mr_material FOREIGN KEY (id_material) REFERENCES
       material(id_material),
83      CONSTRAINT fk_mr_restaurare FOREIGN KEY (id_restaurare) REFERENCES
       restaurare(id_restaurare)
84  );
85
86
87  CREATE TABLE finantare_restaurare (
88      id_finantare  NUMBER(5),
89      id_restaurare NUMBER(5),
90      CONSTRAINT pk_finantare_restaurare PRIMARY KEY (id_finantare,
       id_restaurare),
91      CONSTRAINT fk_fr_finantare FOREIGN KEY (id_finantare) REFERENCES
       finantare(id_finantare),
92      CONSTRAINT fk_fr_restaurare FOREIGN KEY (id_restaurare) REFERENCES
       restaurare(id_restaurare)
93  );
```

Listing 4.1: Crearea tabelelor

Dovada rulării în Oracle:

```
Table MONUMENT created.

Table EXPERT created.

Table MATERIAL created.

Table FINANTARE created.

Table AUTORITATE created.

Table RESTAURARE created.

Table AVIZ created.

Table INSPECTIE created.

Table EXPERT_RESTAURARE created.

Table MATERIAL_RESTAURARE created.

Table FINANTARE_RESTAURARE created.
```

Figura 4.1: Rularea comenzilor CREATE TABLE

5. Popularea Bazei de Date

Au fost inserate minim 5 înregistrări pentru entitățile independente și 10 pentru cele asociative.

```
INSERT INTO monument VALUES (1, 'Castelul Peles', 'Sinaia');
INSERT INTO monument VALUES (2, 'Biserica Neagra', 'Brasov');
INSERT INTO monument VALUES (3, 'Cetatea de Scaun', 'Suceava');
INSERT INTO monument VALUES (4, 'Manastirea Voronet', 'Gura Humorului');
INSERT INTO monument VALUES (5, 'Cazinoul', 'Constanta');

INSERT INTO expert VALUES (101, 'Popescu Ion', 'Arhitect', 'popescu.
    i@expert.ro');
INSERT INTO expert VALUES (102, 'Ionescu Maria', 'Inginer Structurist',
    'maria.i@expert.ro');
INSERT INTO expert VALUES (103, 'Georgescu Vlad', 'Restaurator Pictura',
     'vlad.g@expert.ro');
INSERT INTO expert VALUES (104, 'Dumitru Ana', 'Arheolog', 'ana.d@expert
    .ro');
INSERT INTO expert VALUES (105, 'Stanescu Dan', 'Manager Proiect', 'dan.
    s@expert.ro');

INSERT INTO material VALUES (201, 'Piatra de rau', 'Natural', 'Scazut');
INSERT INTO material VALUES (202, 'Mortar hidraulic', 'Sintetic', 'Mediu
    ');
INSERT INTO material VALUES (203, 'Lemn de stejar tratat', 'Natural', '
    Ridicat');
INSERT INTO material VALUES (204, 'Pigment mineral', 'Chimic', 'Mediu');
INSERT INTO material VALUES (205, 'Caramida arsa', 'Compozit', 'Scazut')
    ;

INSERT INTO finantare VALUES (301, 'Ministerul Culturii', 5000000);
INSERT INTO finantare VALUES (302, 'Fonduri Europene REGIO', 12000000);
INSERT INTO finantare VALUES (303, 'Buget Local Sinaia', 200000);
INSERT INTO finantare VALUES (304, 'Donatii Private ONG', 50000);
INSERT INTO finantare VALUES (305, 'Granturi Norvegiene', 3500000);

INSERT INTO autoritate VALUES (401, 'Ministerul Culturii', '
    Guvernamental', 'Avizare patrimoniu');
INSERT INTO autoritate VALUES (402, 'Primaria Sinaia', 'Local', '
    Certificat Urbanism');
INSERT INTO autoritate VALUES (403, 'ISC Brasov', 'Inspectie', 'Control
    Calitate');
INSERT INTO autoritate VALUES (404, 'Directia Judeteana Cultura', '
    Judetean', 'Monitorizare');
INSERT INTO autoritate VALUES (405, 'Primaria Constanta', 'Local', '
    Autorizatie Constructie');

INSERT INTO restaurare VALUES (1001, 1, TO_DATE('01-03-2024','DD-MM-YYYY
    '), TO_DATE('01-12-2025','DD-MM-YYYY'), 'In executie');
INSERT INTO restaurare VALUES (1002, 2, TO_DATE('15-05-2023','DD-MM-YYYY
    '), TO_DATE('15-05-2024','DD-MM-YYYY'), 'Finalizat');
INSERT INTO restaurare VALUES (1003, 5, TO_DATE('01-01-2020','DD-MM-YYYY
    '), TO_DATE('01-01-2026','DD-MM-YYYY'), 'In executie');
INSERT INTO restaurare VALUES (1004, 3, TO_DATE('10-10-2025','DD-MM-YYYY
    '), NULL, 'Planificat');
```

```
35 INSERT INTO restaurare VALUES (1005, 4, TO_DATE('01-06-2024','DD-MM-YYYY
   '), TO_DATE('01-09-2024','DD-MM-YYYY'), 'Suspendat');
36
37 INSERT INTO aviz VALUES (1, 1001, 401, 'Aviz Favorabil', TO_DATE('
   20-02-2024','DD-MM-YYYY'));
38 INSERT INTO aviz VALUES (2, 1001, 402, 'Autorizatie Constructie',
   TO_DATE('25-02-2024','DD-MM-YYYY'));
39 INSERT INTO aviz VALUES (3, 1003, 405, 'Prelungire Autorizatie', TO_DATE
   ('10-01-2024','DD-MM-YYYY'));
40
41
42 INSERT INTO inspectie VALUES (1, 1001, TO_DATE('01-04-2024','DD-MM-YYYY'
   ), 'Conform cu proiectul');
43 INSERT INTO inspectie VALUES (2, 1003, TO_DATE('15-04-2024','DD-MM-YYYY'
   ), 'Degradari neprevazute la fatada');
44 INSERT INTO inspectie VALUES (3, 1002, TO_DATE('20-03-2024','DD-MM-YYYY'
   ), 'Inspectie finala, lucrari conforme');
45 INSERT INTO inspectie VALUES (4, 1004, TO_DATE('12-11-2025','DD-MM-YYYY'
   ), 'Verificare preliminara a santierului');
46 INSERT INTO inspectie VALUES (5, 1005, TO_DATE('25-07-2024','DD-MM-YYYY'
   ), 'Necesita remedieri la acoperis');
47
48 INSERT INTO expert_restaurare VALUES (101, 1001);
49 INSERT INTO expert_restaurare VALUES (102, 1001);
50 INSERT INTO expert_restaurare VALUES (105, 1001);
51 INSERT INTO expert_restaurare VALUES (101, 1003);
52 INSERT INTO expert_restaurare VALUES (102, 1003);
53 INSERT INTO expert_restaurare VALUES (103, 1005);
54 INSERT INTO expert_restaurare VALUES (104, 1004);
55 INSERT INTO expert_restaurare VALUES (105, 1002);
56 INSERT INTO expert_restaurare VALUES (102, 1002);
57 INSERT INTO expert_restaurare VALUES (101, 1002);
58
59 INSERT INTO material_restaurare VALUES (203, 1001);
60 INSERT INTO material_restaurare VALUES (205, 1001);
61 INSERT INTO material_restaurare VALUES (201, 1002);
62 INSERT INTO material_restaurare VALUES (202, 1002);
63 INSERT INTO material_restaurare VALUES (201, 1003);
64 INSERT INTO material_restaurare VALUES (202, 1003);
65 INSERT INTO material_restaurare VALUES (204, 1003);
66 INSERT INTO material_restaurare VALUES (201, 1004);
67 INSERT INTO material_restaurare VALUES (204, 1005);
68 INSERT INTO material_restaurare VALUES (202, 1005);
69
70 INSERT INTO finantare_restaurare VALUES (301, 1001);
71 INSERT INTO finantare_restaurare VALUES (303, 1001);
72 INSERT INTO finantare_restaurare VALUES (304, 1002);
73 INSERT INTO finantare_restaurare VALUES (305, 1002);
74 INSERT INTO finantare_restaurare VALUES (302, 1003);
75 INSERT INTO finantare_restaurare VALUES (301, 1003);
76 INSERT INTO finantare_restaurare VALUES (305, 1003);
77 INSERT INTO finantare_restaurare VALUES (302, 1004);
78 INSERT INTO finantare_restaurare VALUES (301, 1005);
79 INSERT INTO finantare_restaurare VALUES (304, 1005);
80
81 COMMIT;
```

Listing 5.1: Inserarea datelor

Dovada rulării în Oracle:

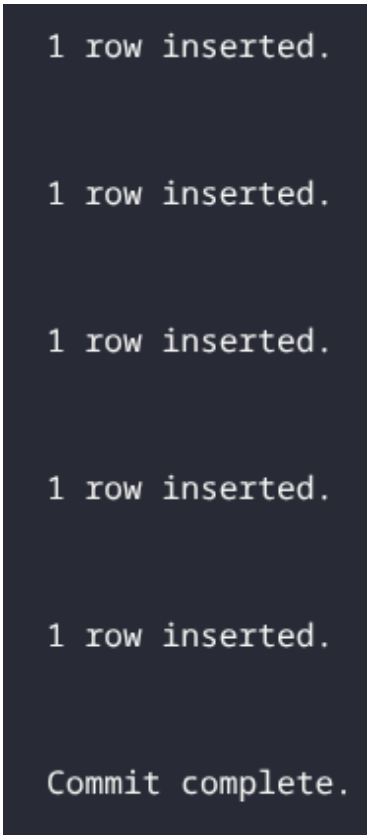

Figura 5.1: Rezultatul inserării datelor (nu sunt afișate toate insert-urile)

6. Utilizarea Colecțiilor (Tablouri/-Vectori)

Enunțul Problemei

Să se scrie o procedură stocată numită `analiza_complexa_proiect` care primește ca parametru ID-ul unui proiect de restaurare. Procedura trebuie să realizeze următoarele acțiuni folosind colecții:

Să identifice primele 5 materiale folosite în proiect și să le stocheze într-un Varray.

Să colecteze numele tuturor experților care lucrează la acest proiect într-un Nested Table.

Să calculeze un bonus simbolic pentru fiecare expert (bazat pe ID-ul expertului) și să stocheze aceste valori într-un Associative Array , având ID-ul expertului drept cheie. La final, să afișeze un raport sintetic.

Rezolvare PL/SQL

```
CREATE OR REPLACE PROCEDURE analiza_complexa_proiect (
    p_id_restaurare IN restaurare.id_restaurare%TYPE
) IS

    TYPE t_materiale_va IS VARRAY(5) OF VARCHAR2(50);
    v_lista_materiale t_materiale_va;

    TYPE t_experti_nt IS TABLE OF VARCHAR2(50);
    v_lista_experti t_experti_nt := t_experti_nt();

    TYPE t_bonusuri_aa IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
    v_map_bonusuri t_bonusuri_aa;

    v_id_expert expert.id_expert%TYPE;

BEGIN
    DBMS_OUTPUT.PUT_LINE('Analiza Proiectului ID: ' || p_id_restaurare);

    SELECT m.denumire
    BULK COLLECT INTO v_lista_materiale
    FROM material m
    JOIN material_restaurare mr ON m.id_material = mr.id_material
    WHERE mr.id_restaurare = p_id_restaurare
    AND ROWNUM <= 5;

    DBMS_OUTPUT.PUT_LINE('> Materiale principale (Varray):');
    IF v_lista_materiale.COUNT > 0 THEN
        FOR i IN 1..v_lista_materiale.COUNT LOOP
            DBMS_OUTPUT.PUT_LINE('  ' || i || '. ' || v_lista_materiale(
   i));
        END LOOP;
    ELSE
        DBMS_OUTPUT.PUT_LINE('  Niciun material gasit.');
    END IF;
```

```
34
35      FOR r IN (
36          SELECT e.id_expert, e.nume
37          FROM expert e
38          JOIN expert_restaurare er ON e.id_expert = er.id_expert
39          WHERE er.id_restaurare = p_id_restaurare
40      ) LOOP
41
42          v_lista_experti.EXTEND;
43          v_lista_experti(v_lista_experti.LAST) := r.nume;
44
45          v_map_bonusuri(r.id_expert) := 1500;
46      END LOOP;
47
48      DBMS_OUTPUT.PUT_LINE('> Lista Experti (Nested Table):');
49      IF v_lista_experti.COUNT > 0 THEN
50          FOR i IN 1..v_lista_experti.COUNT LOOP
51              DBMS_OUTPUT.PUT_LINE('  - ' || v_lista_experti(i));
52          END LOOP;
53      ELSE
54          DBMS_OUTPUT.PUT_LINE('  Niciun expert alocat.');
55      END IF;
56
57      DBMS_OUTPUT.PUT_LINE('> Calcul Bonusuri (Associative Array):');
58      v_id_expert := v_map_bonusuri.FIRST;
59      WHILE v_id_expert IS NOT NULL LOOP
60          DBMS_OUTPUT.PUT_LINE('  Expert ID ' || v_id_expert || ' primeste
        bonus: ' || v_map_bonusuri(v_id_expert) || ' RON');
61          v_id_expert := v_map_bonusuri.NEXT(v_id_expert);
62      END LOOP;
63
64  EXCEPTION
65      WHEN NO_DATA_FOUND THEN
66          DBMS_OUTPUT.PUT_LINE('Nu s-au gasit date pentru acest proiect.')
       ;
67      WHEN OTHERS THEN
68          DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
69  END;
70  /
```

Listing 6.1: Subprogram cu colectii

Apel și Rezultat

```
1  SET SERVEROUTPUT ON;
2  BEGIN
3      analiza_complexa_proiect(1002);
4  END;
```

```
Procedure ANALIZA_COMPLEXA_PROIECT compiled

No errors.
Analiza Proiectului ID: 1002
> Materiale principale (Varray):
  1. Piatra de rau
  2. Mortar hidraulic
> Lista Experti:
  - Popescu Ion
  - Ionescu Maria
  - Stanescu Dan
> Calcul Bonusuri:
  Expert ID 101 primeste bonus: 1500 RON
  Expert ID 102 primeste bonus: 1500 RON
  Expert ID 105 primeste bonus: 1500 RON


PL/SQL procedure successfully completed.
```

Figura 6.1: Execuția cerinței 6

7. Utilizarea Cursoarelor

Enunțul Problemei

Să se scrie o procedură stocată numită `raport_financiar_locatie` care primește ca parametru numele unei locații (oraș). Procedura va afișa monumentele din acea locație și sursele lor de finanțare, folosind două cursoare:

Cursorul Principal (Independent): Va itera prin toate monumentele care se află în locația specificată.

Cursorul Secundar (Parametrizat/Dependent): Pentru fiecare monument găsit de primul cursor, acest al doilea cursor va căuta toate sursele de finanțare ale restaurărilor asociate acelui monument, primind ca parametru ID-ul monumentului.

Rezolvare PL/SQL

```
CREATE OR REPLACE PROCEDURE raport_financiar_locatie (
    p_locatie IN monument.locatie%TYPE
) IS
    CURSOR c_monumente IS
        SELECT id_monument, denumire
        FROM monument
        WHERE UPPER(locatie) = UPPER(p_locatie);

    CURSOR c_finantari (p_id_mon NUMBER) IS
        SELECT f.sursa, f.buget, r.stadiu
        FROM finantare f
        JOIN finantare_restaurare fr ON f.id_finantare = fr.id_finantare
        JOIN restaurare r ON fr.id_restaurare = r.id_restaurare
        WHERE r.id_monument = p_id_mon;

    v_total_monument NUMBER;

BEGIN
    DBMS_OUTPUT.PUT_LINE('Raport Financiar pentru: ' || p_locatie);

    FOR r_mon IN c_monumente LOOP
        DBMS_OUTPUT.PUT_LINE('Monument: ' || r_mon.denumire || ' (ID: '
   || r_mon.id_monument || ')');

        v_total_monument := 0;

        FOR r_fin IN c_finantari(r_mon.id_monument) LOOP
            DBMS_OUTPUT.PUT_LINE('   > Sursa: ' || r_fin.sursa ||
                                 ' | Buget: ' || r_fin.buget ||
                                 ' | Stadiu: ' || r_fin.stadiu);
            v_total_monument := v_total_monument + r_fin.buget;
        END LOOP;

        IF v_total_monument = 0 THEN
             DBMS_OUTPUT.PUT_LINE('   > Nu exista finantari inregistrate
   .');
        ELSE
```

```
36              DBMS_OUTPUT.PUT_LINE('    > TOTAL MONUMENT: ' ||
    v_total_monument || ' RON');
37          END IF;
38
39      END LOOP;
40  END;
41  /
```

Listing 7.1: Subprogram cu cursoare

Apel și Rezultat

```
1      SET SERVEROUTPUT ON;
2   BEGIN
3      raport_financiar_locatie('Sinaia');
4      DBMS_OUTPUT.PUT_LINE('');
5      raport_financiar_locatie('Constanta');
6   END;
7   /
```

```
Procedure RAPORT_FINANCIAR_LOCATIE compiled

No errors.
Raport Financiar pentru: Sinaia
Monument: Castelul Peles (ID: 1)
   > Sursa: Ministerul Culturii | Buget: 5000000 | Stadiu: In executie
   > Sursa: Buget Local Sinaia | Buget: 200000 | Stadiu: In executie
   > TOTAL MONUMENT: 5200000 RON

Raport Financiar pentru: Constanta
Monument: Cazinoul (ID: 5)
   > Sursa: Ministerul Culturii | Buget: 5000000 | Stadiu: In executie
   > Sursa: Fonduri Europene REGIO | Buget: 12000000 | Stadiu: In execut
   > Sursa: Granturi Norvegiene | Buget: 3500000 | Stadiu: In executie
   > TOTAL MONUMENT: 20500000 RON


PL/SQL procedure successfully completed.
```

Figura 7.1: Execuția cerinței 7

8. Funcție Stocată

Enunțul Problemei

Să se scrie o funcție stocată numită `get_material_unic` care primește ca parametru ID-ul unei restaurări și returnează denumirea materialului utilizat în cadrul acesteia. Funcția va interoga trei tabele (MATERIAL, MATERIAL_RESTAURARE, RESTAURARE) pentru a obține rezultatul. Trebuie tratate cazurile în care restaurarea nu există sau nu are materiale alocate (`NO_DATA_FOUND`) și cazul în care restaurarea utilizează mai multe materiale simultan (`TOO_MANY_ROWS`), returnând mesaje corespunzătoare.

Rezolvare PL/SQL

```
CREATE OR REPLACE FUNCTION get_material_unic(p_id_restaurare NUMBER)
RETURN VARCHAR2 IS
    v_denumire_material VARCHAR2(50);
BEGIN
    SELECT m.denumire
    INTO v_denumire_material
    FROM material m
    JOIN material_restaurare mr ON m.id_material = mr.id_material
    JOIN restaurare r ON mr.id_restaurare = r.id_restaurare
    WHERE r.id_restaurare = p_id_restaurare;

    RETURN 'Material identificat: ' || v_denumire_material;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'Eroare: Nu exista materiale sau ID invalid.';
    WHEN TOO_MANY_ROWS THEN
        RETURN 'Eroare: Proiectul utilizeaza mai multe materiale.';
    WHEN OTHERS THEN
        RETURN 'Alta eroare: ' || SQLERRM;
END;
/
```

Listing 8.1: Functia PL/SQL

Apel și Rezultat (Tratare Excepții)

```
SET SERVEROUTPUT ON;
BEGIN
    DBMS_OUTPUT.PUT_LINE('CAZ 1: O singura inregistrare (Succes)');
    DBMS_OUTPUT.PUT_LINE(get_material_unic(1004));

    DBMS_OUTPUT.PUT_LINE('CAZ 2: Mai multe inregistrari (TOO_MANY_ROWS)'
    );
    DBMS_OUTPUT.PUT_LINE(get_material_unic(1001));

    DBMS_OUTPUT.PUT_LINE('CAZ 3: Nicio inregistrare (NO_DATA_FOUND)');
    DBMS_OUTPUT.PUT_LINE(get_material_unic(9999));
```

```
11 END;
12 /
```

```
Function GET_MATERIAL_UNIC compiled

No errors.
CAZ 1: O singura inregistrare (Succes)
Material identificat: Piatra de rau
CAZ 2: Mai multe inregistrari (TOO_MANY_ROWS)
Eroare: Proiectul utilizeaza mai multe materiale
CAZ 3: Nicio inregistrare (NO_DATA_FOUND)
Eroare: Nu exista materiale sau ID invalid.


PL/SQL procedure successfully completed.
```

Figura 8.1: Apelul funcției și tratarea excepțiilor

9. Procedură Stocată

Enunțul Problemei

Să se scrie o procedură stocată numită `audit_complex_monument` care primește doi parametri: ID-ul unui monument și un prag minim financiar. Procedura va calcula bugetul total alocat și numărul de experți implicați în restaurările acelui monument, utilizând o singură interogare ce unește 5 tabele (`MONUMENT`, `RESTAURARE`, `FINANTARE_RESTAURARE`, `FINANTARE`, `EXPERT_RESTAURARE`). Să se definească și să se trateze două excepții proprii:

`exc_fonduri_insuficiente`: Se declanșează dacă bugetul total este sub pragul minim furnizat ca parametru.

`exc_expertiza_limitata`: Se declanșează dacă la proiect participă mai puțin de 2 experți. De asemenea, să se trateze excepția de sistem `NO_DATA_FOUND`.

Rezolvare PL/SQL

```
CREATE OR REPLACE PROCEDURE audit_complex_monument (
    p_id_monument IN NUMBER,
    p_prag_financiar IN NUMBER
) IS
    v_total_buget NUMBER;
    v_nr_experti NUMBER;

    exc_fonduri_insuficiente EXCEPTION;
    exc_expertiza_limitata EXCEPTION;
BEGIN
    SELECT SUM(f.buget), COUNT(DISTINCT er.id_expert)
    INTO v_total_buget, v_nr_experti
    FROM monument m
    JOIN restaurare r ON m.id_monument = r.id_monument
    JOIN finantare_restaurare fr ON r.id_restaurare = fr.id_restaurare
    JOIN finantare f ON fr.id_finantare = f.id_finantare
    JOIN expert_restaurare er ON r.id_restaurare = er.id_restaurare
    WHERE m.id_monument = p_id_monument
    GROUP BY m.id_monument;

    IF v_total_buget < p_prag_financiar THEN
        RAISE exc_fonduri_insuficiente;
    END IF;

    IF v_nr_experti < 2 THEN
        RAISE exc_expertiza_limitata;
    END IF;

    DBMS_OUTPUT.PUT_LINE('Audit trecut cu succes. Buget: ' ||
   v_total_buget || ' RON, Experti: ' || v_nr_experti);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Eroare: Monumentul nu are restaurari
   active, finantari sau experti alocati.');
    WHEN exc_fonduri_insuficiente THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Alerta: Fonduri sub pragul minim (' ||
    v_total_buget || ' < ' || p_prag_financiar || ').');
    WHEN exc_expertiza_limitata THEN
        DBMS_OUTPUT.PUT_LINE('Alerta: Echipa tehnica insuficienta (' ||
    v_nr_experti || ' experti).');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Eroare necunoscuta: ' || SQLERRM);
END;
/
```

Listing 9.1: Procedura PL/SQL

Apel și Rezultat

```
SET SERVEROUTPUT ON;
BEGIN
    DBMS_OUTPUT.PUT_LINE(' CAZ 1: Succes (Castelul Peles, Buget mic
    cerut) ');
    audit_complex_monument(1, 1000);

    DBMS_OUTPUT.PUT_LINE(' CAZ 2: Exceptie Proprie 1 - Fonduri
    Insuficiente ');
    audit_complex_monument(1, 999999999);

    DBMS_OUTPUT.PUT_LINE(' CAZ 3: Exceptie Proprie 2 - Expertiza
    Limitata (Voronet) ');
    audit_complex_monument(4, 100);

    DBMS_OUTPUT.PUT_LINE(' CAZ 4: Exceptie Sistem - NO_DATA_FOUND ');
    % audit_complex_monument(99, 1000);
END;
/
```

```
Procedure AUDIT_COMPLEX_MONUMENT compiled

No errors.
 CAZ 1: Succes (Castelul Peles, Buget mic cerut)
Audit trecut cu succes. Buget: 15600000 RON, Experti: 3
 CAZ 2: Excepție Proprie 1 - Fonduri Insuficiente
Alerta: Fonduri sub pragul minim (15600000 < 999999999).
 CAZ 3: Excepție Proprie 2 - Expertiza Limitata (Voronet)
Alerta: Echipa tehnica insuficienta (1 experti).
 CAZ 4: Excepție Sistem - NO_DATA_FOUND
Eroare: Monumentul nu are restaurari active, finantari sau experti alocati.


PL/SQL procedure successfully completed.
```

Figura 9.1: Execuția procedurii

10. Trigger LMD la Nivel de Comandă

Enunțul Problemei

Trigger LMD la nivel de comandă (statement-level) care securizează tabela `MONUMENT`. Trigger-ul se declanșează înainte de orice instrucțiune `DELETE` și oprește execuția, afișând un mesaj de eroare personalizat, deoarece ștergerea monumentelor istorice din baza de date este strict interzisă prin politica instituției.

Definire Trigger

```
1 CREATE OR REPLACE TRIGGER trg_stop_stergere_monument
2 BEFORE DELETE ON monument
3 BEGIN
4     RAISE_APPLICATION_ERROR(-20005, 'Securitate: Stergerea
     inregistrarilor din tabela MONUMENT este interzisa!');
5 END;
6 /
```

Declanșare

```
1 DELETE FROM monument WHERE id_monument = 1;
```

```
Trigger TRG_STOP_STERGERE_MONUMENT compiled

No errors.

Error starting at line : 1 in command -
DELETE FROM monument WHERE id_monument = 1
Error at Command Line : 1 Column : 13
Error report -
SQL Error: ORA-20005: Securitate: Stergerea inregistrarilor din tabela MONUMENT este interzisa!
ORA-06512: at "SYSTEM.TRG_STOP_STERGERE_MONUMENT", line 2
ORA-04088: error during execution of trigger 'SYSTEM.TRG_STOP_STERGERE_MONUMENT'

https://docs.oracle.com/error-help/db/ora-20005/

More Details :
https://docs.oracle.com/error-help/db/ora-20005/
https://docs.oracle.com/error-help/db/ora-06512/
https://docs.oracle.com/error-help/db/ora-04088/
```

Figura 10.1: Demonstrație declanșare trigger

11. Trigger LMD la Nivel de Linie

Enuntul Problemei

Un trigger LMD la nivel de linie (row-level) pe tabelul `RESTAURARE`. Trigger-ul va interzice modificarea stadiului unui proiect inapoi in `"In executie"` sau `"Planificat"` daca acesta a fost deja marcat anterior ca `"Finalizat"` (se interzice redeschiderea proiectelor inchise).

Definire Trigger

```
CREATE OR REPLACE TRIGGER trg_blocare_redeschidere
BEFORE UPDATE ON restaurare
FOR EACH ROW
BEGIN
    IF :OLD.stadiu = 'Finalizat'
       AND :NEW.stadiu IN ('In executie', 'Planificat') THEN
        RAISE_APPLICATION_ERROR(
            -20020,
            'Eroare: Un proiect finalizat nu poate fi redeschis!'
        );
    END IF;
END;
/
```

Declanșare

```
UPDATE restaurare
SET stadiu = 'In executie'
WHERE id_restaurare = 1002;
```

```
Trigger TRG_BLOCARE_REDESCHIDERE compiled

No errors.

Error starting at line : 1 in command -
UPDATE restaurare
SET stadiu = 'In executie'
WHERE id_restaurare = 1002
Error at Command Line : 1 Column : 8
Error report -
SQL Error: ORA-20020: Eroare: Un proiect finalizat nu mai poate fi modificat!
ORA-06512: at "SYSTEM.TRG_BLOCARE_REDESCHIDERE", line 3
ORA-04088: error during execution of trigger 'SYSTEM.TRG_BLOCARE_REDESCHIDERE'

https://docs.oracle.com/error-help/db/ora-20020/

More Details :
https://docs.oracle.com/error-help/db/ora-20020/
https://docs.oracle.com/error-help/db/ora-06512/
https://docs.oracle.com/error-help/db/ora-04088/
```

Figura 11.1: Demonstrație declanșare trigger

12. Trigger LDD

Enunțul Problemei

Se definește un tabel de log numit `LOG_LDD` și un trigger LDD la nivel de schemă. Trigger-ul se declanșează automat după orice instrucțiune de creare (`CREATE`), modificare (`ALTER`) sau ștergere (`DROP`) a unui obiect din baza de date și înregistrează detaliile acțiunii în tabelul de log.

```
CREATE TABLE log_ldd (
    utilizator    VARCHAR2(30),
    data_actiune  DATE,
    tip_eveniment VARCHAR2(30),
    nume_obiect   VARCHAR2(50)
);

CREATE OR REPLACE TRIGGER trg_audit_ldd
AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
    INSERT INTO log_ldd (utilizator, data_actiune, tip_eveniment,
   nume_obiect)
    VALUES (USER, SYSDATE, ORA_SYSEVENT, ORA_DICT_OBJ_NAME);
END;
/
```

Declanșare

```
CREATE TABLE test_temp (id NUMBER);
DROP TABLE test_temp;

SELECT * FROM log_ldd;
```

	UTILIZATOR	DATA_ACTIUNE	TIP_EVENIMENT	NUME_OBIECT
1	SYSTEM	09/01/2026 04:52:33	CREATE	TEST_TEMP
2	SYSTEM	09/01/2026 04:52:36	DROP	TEST_TEMP

Figura 12.1: Demonstrație declanșare trigger LDD

13. Pachet PL/SQL

Enunțul Problemei

Să se creeze un pachet numit `pkg_audit_rapid` pentru verificarea experților. Pachetul va conține două tipuri (o listă de ID-uri și un tip record pentru statistici), două funcții (una verifică dacă un expert există, alta numără proiectele lui) și două proceduri (una procesează o listă de experți, cealaltă afișează un raport scurt).

Specificația Pachetului

```
CREATE OR REPLACE PACKAGE pkg_audit_rapid AS
    TYPE t_lista_id IS TABLE OF NUMBER;
    TYPE t_stats   IS RECORD (nr_proiecte NUMBER);

    FUNCTION exista_expert(p_id NUMBER) RETURN BOOLEAN;
    FUNCTION nr_proiecte(p_id NUMBER) RETURN NUMBER;

    PROCEDURE verifica_lista(p_lista t_lista_id);
    PROCEDURE raport_scurt(p_id NUMBER);
END;
/
```

Corpul Pachetului

```
CREATE OR REPLACE PACKAGE BODY pkg_audit_rapid AS

    FUNCTION exista_expert(p_id NUMBER) RETURN BOOLEAN IS
        v_chk NUMBER;
    BEGIN
        SELECT COUNT(*) INTO v_chk FROM expert WHERE id_expert = p_id;
        RETURN v_chk > 0;
    END;

    FUNCTION nr_proiecte(p_id NUMBER) RETURN NUMBER IS
        v_count NUMBER;
    BEGIN
        SELECT COUNT(*) INTO v_count FROM expert_restaurare WHERE
   id_expert = p_id;
        RETURN v_count;
    END;

    PROCEDURE verifica_lista(p_lista t_lista_id) IS
    BEGIN
        FOR i IN 1..p_lista.COUNT LOOP
            IF exista_expert(p_lista(i)) THEN
                DBMS_OUTPUT.PUT_LINE('ID ' || p_lista(i) || ' valid.');
            ELSE
                DBMS_OUTPUT.PUT_LINE('ID ' || p_lista(i) || ' INEXISTENT
   .');
            END IF;
```

```
25          END LOOP;
26      END;
27
28      PROCEDURE raport_scurt(p_id NUMBER) IS
29          v_stat t_stats;
30      BEGIN
31          v_stat.nr_proiecte := nr_proiecte(p_id);
32          DBMS_OUTPUT.PUT_LINE('Expertul ' || p_id || ' are ' || v_stat.
    nr_proiecte || ' proiecte.');
33      END;
34
35  END;
36  /
```

Apel Funcționalități

```
1  SET SERVEROUTPUT ON;
2  DECLARE
3      v_lista pkg_audit_rapid.t_lista_id := pkg_audit_rapid.t_lista_id
    (101, 999);
4  BEGIN
5      pkg_audit_rapid.verifica_lista(v_lista);
6      pkg_audit_rapid.raport_scurt(101);
7  END;
8  /
```

```
Package PKG_AUDIT_RAPID compiled

No errors.

Package Body PKG_AUDIT_RAPID compiled

No errors.
ID 101 valid.
ID 999 INEXISTENT.
Expertul 101 are 3 proiecte.


PL/SQL procedure successfully completed.
```

Figura 13.1: Execuția elementelor din pachet