

Sisteme de Gestiune a Bazelor de Date

Proiect

Sistem de Gestiune a Restaurării Patrimoniului Cultural

Student: Voaides-Negustor Robert-Ionuț

Grupa: 251

Seria: 25

Anul Universitar: 2025-2026

Cuprins

1 Descrierea Bazei de Date	3
1.1 Scenariul Real Modelat	3
1.2 Utilitatea Bazei de Date	3
2 Diagrama Entitate-Relație (ERD)	4
3 Diagrama Conceptuală	5
4 Implementarea Structurii	6
5 Popularea Bazei de Date	9
6 Utilizarea Colecțiilor (Tablouri/Vectori)	12
7 Utilizarea Cursoarelor	15
8 Funcție Stocată	17
9 Procedură Stocată	19
10 Trigger LMD la Nivel de Comandă	21
11 Trigger LMD la Nivel de Linie	22
12 Trigger LDD	23
13 Pachet PL/SQL	24

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 finanțiară 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)

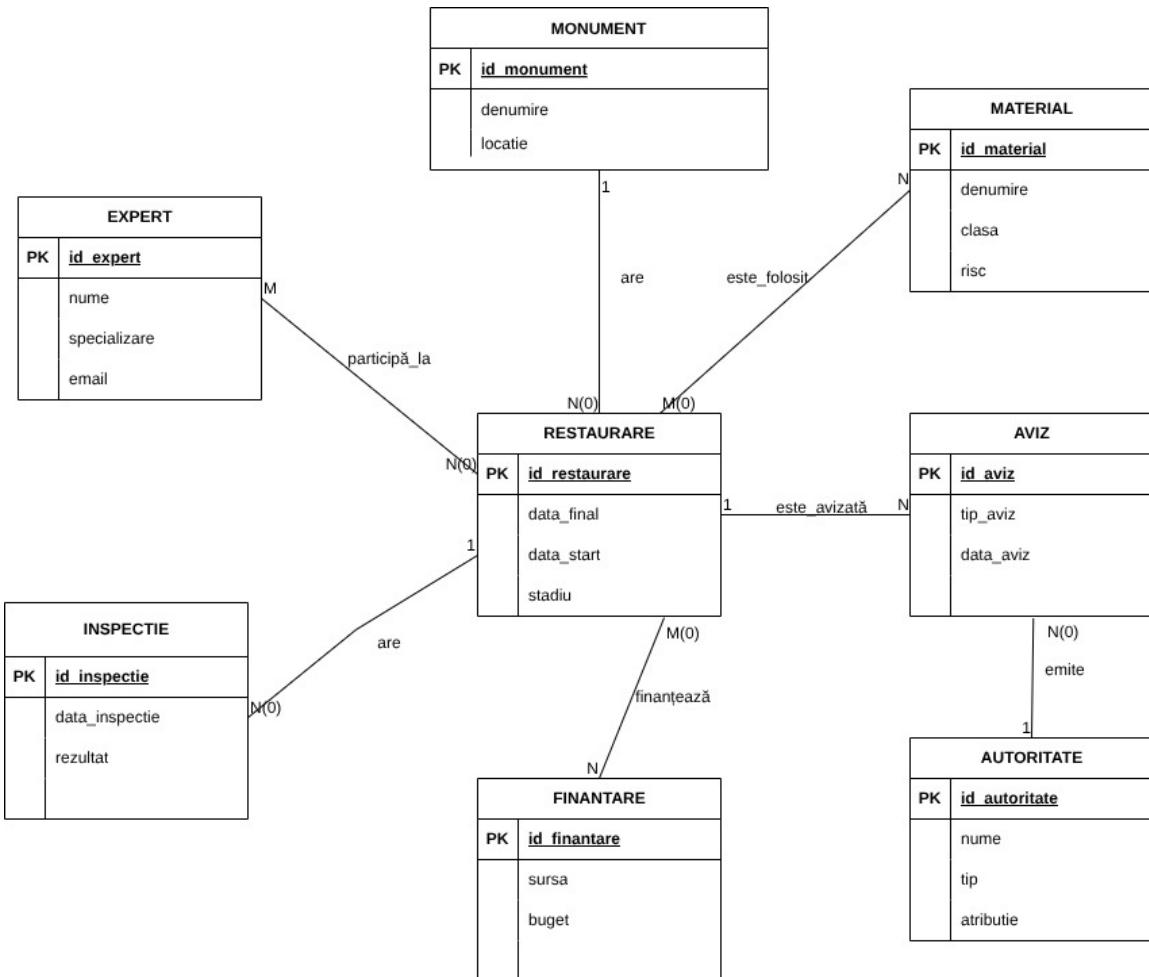


Figura 2.1: Diagrama ERD a bazei de date

3. Diagrama Conceptuală

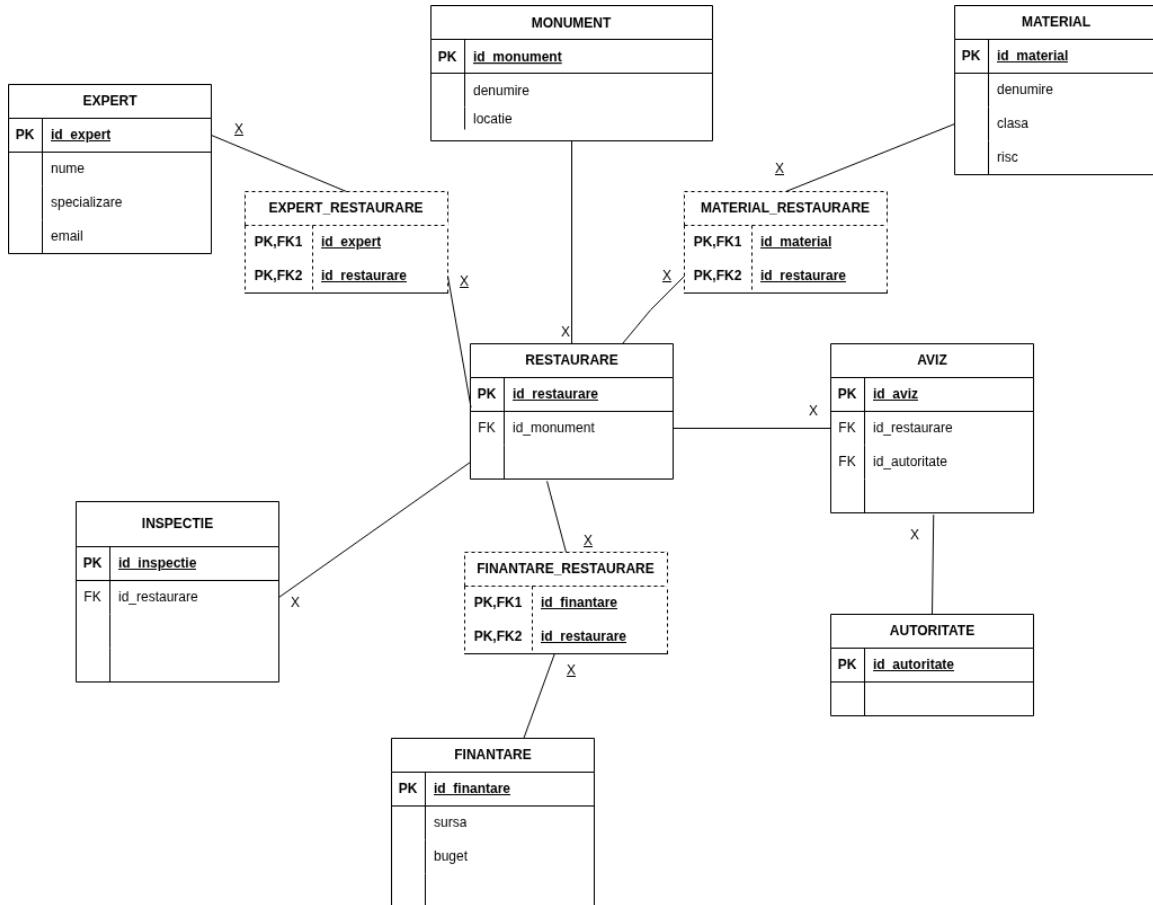


Figura 3.1: Diagrama Conceptuală

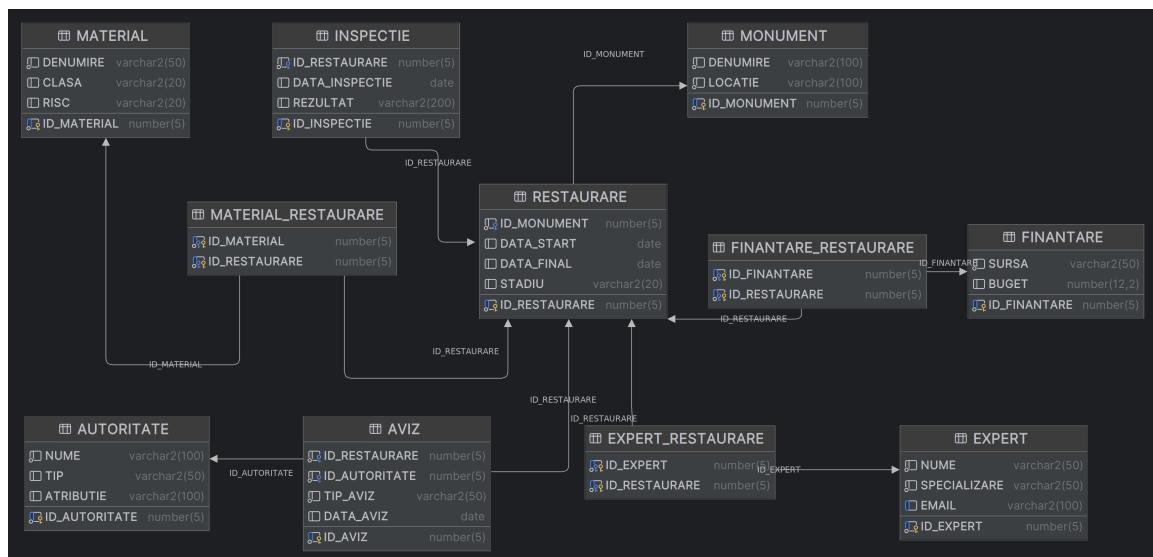


Figura 3.2: Diagrama Conceptuală generată de DataGrip

4. Implementarea Structurii

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

```
1 CREATE TABLE monument (
2     id_monument NUMBER(5) CONSTRAINT pk_monument PRIMARY KEY,
3     denumire     VARCHAR2(100) NOT NULL,
4     locatie      VARCHAR2(100) NOT NULL
5 );
6
7
8 CREATE TABLE expert (
9     id_expert      NUMBER(5) CONSTRAINT pk_expert PRIMARY KEY,
10    nume           VARCHAR2(50) NOT NULL,
11    specializare   VARCHAR2(50) NOT NULL,
12    email          VARCHAR2(100) CONSTRAINT uq_expert_email UNIQUE
13 );
14
15
16 CREATE TABLE material (
17     id_material    NUMBER(5) CONSTRAINT pk_material PRIMARY KEY,
18     denumire       VARCHAR2(50) NOT NULL,
19     clasa          VARCHAR2(20),
20     risc            VARCHAR2(20)
21 );
22
23
24 CREATE TABLE finantare (
25     id_finantare  NUMBER(5) CONSTRAINT pk_finantare PRIMARY KEY,
26     sursa          VARCHAR2(50) NOT NULL,
27     buget          NUMBER(12, 2) CONSTRAINT ck_buget_pozitiv CHECK (buget
> 0)
28 );
29
30
31 CREATE TABLE autoritate (
32     id_autoritate NUMBER(5) CONSTRAINT pk_autoritate PRIMARY KEY,
33     nume          VARCHAR2(100) NOT NULL,
34     tip            VARCHAR2(50),
35     atributie     VARCHAR2(100)
36 );
37
38 CREATE TABLE restaurare (
39     id_restaurare NUMBER(5) CONSTRAINT pk_restaurare PRIMARY KEY,
40     id_monument   NUMBER(5) NOT NULL,
41     data_start     DATE DEFAULT SYSDATE,
42     data_final    DATE,
43     stadiu        VARCHAR2(20) CONSTRAINT ck_stadiu CHECK (stadiu IN ('
Planificat', 'In executie', 'Finalizat', 'Suspendat')),
44     CONSTRAINT fk_restaurare_monument FOREIGN KEY (id_monument)
45     REFERENCES monument(id_monument),
46     CONSTRAINT ck_date_valide CHECK (data_final >= data_start)
47 );
48
49 CREATE TABLE aviz (
50     id_aviz         NUMBER(5) CONSTRAINT pk_aviz PRIMARY KEY,
51     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
56     restaurare(id_restaurare),
57     CONSTRAINT fk_aviz_autoritate FOREIGN KEY (id_autoritate) REFERENCES
58     autoritate(id_autoritate)
59   );
60
61
62
63
64
65
66
67
68
69 CREATE TABLE inspectie (
70     id_inspectie    NUMBER(5) CONSTRAINT pk_inspectie PRIMARY KEY ,
71     id_restaurare  NUMBER(5) NOT NULL ,
72     data_inspectie DATE DEFAULT SYSDATE ,
73     rezultat        VARCHAR2(200) ,
74     CONSTRAINT fk_inspectie_restaurare FOREIGN KEY (id_restaurare)
75     REFERENCES restaurare(id_restaurare)
76   );
77
78
79
80
81
82
83
84
85
86
87 CREATE TABLE material_restaurare (
88     id_material     NUMBER(5) ,
89     id_restaurare  NUMBER(5) ,
90     CONSTRAINT pk_material_restaurare PRIMARY KEY (id_material ,
91     id_restaurare),
92     CONSTRAINT fk_mr_material FOREIGN KEY (id_material) REFERENCES
93     material(id_material),
94     CONSTRAINT fk_mr_restaurare FOREIGN KEY (id_restaurare) REFERENCES
95     restaurare(id_restaurare)
96   );
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193

```

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.

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

6
7 INSERT INTO expert VALUES (101, 'Popescu Ion', 'Arhitect', 'popescu.
     i@expert.ro');
8 INSERT INTO expert VALUES (102, 'Ionescu Maria', 'Inginer Structurist',
     'maria.i@expert.ro');
9 INSERT INTO expert VALUES (103, 'Georgescu Vlad', 'Restaurator Pictura',
     'vlad.g@expert.ro');
10 INSERT INTO expert VALUES (104, 'Dumitru Ana', 'Arheolog', 'ana.d@expert
     .ro');
11 INSERT INTO expert VALUES (105, 'Stanescu Dan', 'Manager Proiect', 'dan.
     s@expert.ro');

12
13 INSERT INTO material VALUES (201, 'Piatra de rau', 'Natural', 'Scuzat');
14 INSERT INTO material VALUES (202, 'Mortar hidraulic', 'Sintetic', 'Mediu
     ');
15 INSERT INTO material VALUES (203, 'Lemn de stejar tratat', 'Natural', 'Ridicat');
16 INSERT INTO material VALUES (204, 'Pigment mineral', 'Chimic', 'Mediu');
17 INSERT INTO material VALUES (205, 'Caramida arsa', 'Compozit', 'Scuzat')
     ;

18
19 INSERT INTO finantare VALUES (301, 'Ministerul Culturii', 5000000);
20 INSERT INTO finantare VALUES (302, 'Fonduri Europene REGIO', 12000000);
21 INSERT INTO finantare VALUES (303, 'Buget Local Sinaia', 200000);
22 INSERT INTO finantare VALUES (304, 'Donatii Private ONG', 50000);
23 INSERT INTO finantare VALUES (305, 'Granturi Norvegiene', 3500000);

24
25 INSERT INTO autoritate VALUES (401, 'Ministerul Culturii', 'Guvernamental',
     'Avizare patrimoniu');
26 INSERT INTO autoritate VALUES (402, 'Primaria Sinaia', 'Local', 'Certificat Urbanism');
27 INSERT INTO autoritate VALUES (403, 'ISC Brasov', 'Inspectie', 'Control Calitate');
28 INSERT INTO autoritate VALUES (404, 'Directia Judeteana Cultura', 'Judetean',
     'Monitorizare');
29 INSERT INTO autoritate VALUES (405, 'Primaria Constanta', 'Local', 'Autorizatie Constructie');

30
31 INSERT INTO restaurare VALUES (1001, 1, TO_DATE('01-03-2024', 'DD-MM-YYYY'),
     TO_DATE('01-12-2025', 'DD-MM-YYYY'), 'In executie');
32 INSERT INTO restaurare VALUES (1002, 2, TO_DATE('15-05-2023', 'DD-MM-YYYY'),
     TO_DATE('15-05-2024', 'DD-MM-YYYY'), 'Finalizat');
33 INSERT INTO restaurare VALUES (1003, 5, TO_DATE('01-01-2020', 'DD-MM-YYYY'),
     TO_DATE('01-01-2026', 'DD-MM-YYYY'), 'In executie');
34 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_restaureare VALUES (101, 1001);
49 INSERT INTO expert_restaureare VALUES (102, 1001);
50 INSERT INTO expert_restaureare VALUES (105, 1001);
51 INSERT INTO expert_restaureare VALUES (101, 1003);
52 INSERT INTO expert_restaureare VALUES (102, 1003);
53 INSERT INTO expert_restaureare VALUES (103, 1005);
54 INSERT INTO expert_restaureare VALUES (104, 1004);
55 INSERT INTO expert_restaureare VALUES (105, 1002);
56 INSERT INTO expert_restaureare VALUES (102, 1002);
57 INSERT INTO expert_restaureare VALUES (101, 1002);
58
59 INSERT INTO material_restaureare VALUES (203, 1001);
60 INSERT INTO material_restaureare VALUES (205, 1001);
61 INSERT INTO material_restaureare VALUES (201, 1002);
62 INSERT INTO material_restaureare VALUES (202, 1002);
63 INSERT INTO material_restaureare VALUES (201, 1003);
64 INSERT INTO material_restaureare VALUES (202, 1003);
65 INSERT INTO material_restaureare VALUES (204, 1003);
66 INSERT INTO material_restaureare VALUES (201, 1004);
67 INSERT INTO material_restaureare VALUES (204, 1005);
68 INSERT INTO material_restaureare VALUES (202, 1005);
69
70 INSERT INTO finantare_restaureare VALUES (301, 1001);
71 INSERT INTO finantare_restaureare VALUES (303, 1001);
72 INSERT INTO finantare_restaureare VALUES (304, 1002);
73 INSERT INTO finantare_restaureare VALUES (305, 1002);
74 INSERT INTO finantare_restaureare VALUES (302, 1003);
75 INSERT INTO finantare_restaureare VALUES (301, 1003);
76 INSERT INTO finantare_restaureare VALUES (305, 1003);
77 INSERT INTO finantare_restaureare VALUES (302, 1004);
78 INSERT INTO finantare_restaureare VALUES (301, 1005);
79 INSERT INTO finantare_restaureare VALUES (304, 1005);
80
81 COMMIT;

```

Listing 5.1: Inserarea datelor

Dovada rulării în Oracle:

```
1 row inserted.  
1 row inserted.  
1 row inserted.  
1 row inserted.  
Commit complete.
```

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 expertilor 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

```
1 CREATE OR REPLACE PROCEDURE analiza_complexa_proiect (
2     p_id_restaurare IN restaurare.id_restaurare%TYPE
3 ) IS
4
5     TYPE t_materiale_va IS VARRAY(5) OF VARCHAR2(50);
6     v_lista_materiale t_materiale_va;
7
8     TYPE t_experti_nt IS TABLE OF VARCHAR2(50);
9     v_lista_experti t_experti_nt := t_experti_nt();
10
11    TYPE t_bonusuri_aa IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
12    v_map_bonusuri t_bonusuri_aa;
13
14    v_id_expert expert.id_expert%TYPE;
15
16 BEGIN
17     DBMS_OUTPUT.PUT_LINE('Analiza Proiectului ID: ' || p_id_restaurare);
18
19     SELECT m.denumire
20     BULK COLLECT INTO v_lista_materiale
21     FROM material m
22     JOIN material_restaurare mr ON m.id_material = mr.id_material
23     WHERE mr.id_restaurare = p_id_restaurare
24     AND ROWNUM <= 5;
25
26     DBMS_OUTPUT.PUT_LINE('> Materiale principale (Varray):');
27     IF v_lista_materiale.COUNT > 0 THEN
28         FOR i IN 1..v_lista_materiale.COUNT LOOP
29             DBMS_OUTPUT.PUT_LINE(' ' || i || '. ' || v_lista_materiale(
30             i));
31         END LOOP;
32     ELSE
33         DBMS_OUTPUT.PUT_LINE(' Niciun material gasit.');
34     END IF;
```

```

34
35 FOR r IN (
36     SELECT e.id_expert, e.nume
37     FROM expert e
38     JOIN expert_restaureare er ON e.id_expert = er.id_expert
39     WHERE er.id_restaureare = p_id_restaureare
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
61 bonus: ' || v_map_bonusuri(v_id_expert) || ' RON');
62     v_id_expert := v_map_bonusuri.NEXT(v_id_expert);
63 END LOOP;
64
65 EXCEPTION
66     WHEN NO_DATA_FOUND THEN
67         DBMS_OUTPUT.PUT_LINE('Nu s-au gasit date pentru acest proiect.')
68 ;
69     WHEN OTHERS THEN
70         DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLERRM);
71 END;
72 /

```

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 aceluui monument, primind ca parametru ID-ul monumentului.

Rezolvare PL/SQL

```
1 CREATE OR REPLACE PROCEDURE raport_financiar_locatie (
2     p_locatie IN monument.locatie%TYPE
3 ) IS
4     CURSOR c_monumente IS
5         SELECT id_monument, denumire
6         FROM monument
7         WHERE UPPER(locatie) = UPPER(p_locatie);
8
9     CURSOR c_finantari (p_id_mon NUMBER) IS
10        SELECT f.sursa, f.buget, r.stadiu
11        FROM finantare f
12        JOIN finantare_restaurare fr ON f.id_finantare = fr.id_finantare
13        JOIN restaurare r ON fr.id_restaurare = r.id_restaurare
14        WHERE r.id_monument = p_id_mon;
15
16     v_total_monument NUMBER;
17
18 BEGIN
19     DBMS_OUTPUT.PUT_LINE('Raport Financiar pentru: ' || p_locatie);
20
21     FOR r_mon IN c_monumente LOOP
22         DBMS_OUTPUT.PUT_LINE('Monument: ' || r_mon.denumire || ' (ID: '
|| r_mon.id_monument || ')');
23
24         v_total_monument := 0;
25
26         FOR r_fin IN c_finantari(r_mon.id_monument) LOOP
27             DBMS_OUTPUT.PUT_LINE('    > Sursa: ' || r_fin.sursa ||
28                                ' | Buget: ' || r_fin.buget ||
29                                ' | Stadiu: ' || r_fin.stadiu);
30             v_total_monument := v_total_monument + r_fin.buget;
31         END LOOP;
32
33         IF v_total_monument = 0 THEN
34             DBMS_OUTPUT.PUT_LINE('    > Nu există finanțări înregistrate
..');
35         ELSE
```

```

36      DBMS_OUTPUT.PUT_LINE('    > TOTAL MONUMENT: ' || 
37      v_total_monument || ' RON');
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 executie
    > 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

```
1 CREATE OR REPLACE FUNCTION get_material_unic(p_id_restaurare NUMBER)
2 RETURN VARCHAR2 IS
3     v_denumire_material VARCHAR2(50);
4 BEGIN
5     SELECT m.denumire
6     INTO v_denumire_material
7     FROM material m
8     JOIN material_restaurare mr ON m.id_material = mr.id_material
9     JOIN restaurare r ON mr.id_restaurare = r.id_restaurare
10    WHERE r.id_restaurare = p_id_restaurare;
11
12    RETURN 'Material identificat: ' || v_denumire_material;
13
14 EXCEPTION
15    WHEN NO_DATA_FOUND THEN
16        RETURN 'Eroare: Nu există materiale sau ID invalid.';
17    WHEN TOO_MANY_ROWS THEN
18        RETURN 'Eroare: Proiectul utilizează mai multe materiale./';
19    WHEN OTHERS THEN
20        RETURN 'Alta eroare: ' || SQLERRM;
21 END;
22 /
```

Listing 8.1: Functia PL/SQL

Apel și Rezultat (Tratare Excepții)

```
1 SET SERVEROUTPUT ON;
2 BEGIN
3     DBMS_OUTPUT.PUT_LINE('CAZ 1: O singura înregistrare (Success)');
4     DBMS_OUTPUT.PUT_LINE(get_material_unic(1004));
5
6     DBMS_OUTPUT.PUT_LINE('CAZ 2: Mai multe înregistrări (TOO_MANY_ROWS)');
7     DBMS_OUTPUT.PUT_LINE(get_material_unic(1001));
8
9     DBMS_OUTPUT.PUT_LINE('CAZ 3: Nicio înregistrare (NO_DATA_FOUND)');
10    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 aceluia 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

```
1 CREATE OR REPLACE PROCEDURE audit_complex_monument (
2     p_id_monument IN NUMBER,
3     p_prag_financiar IN NUMBER
4 ) IS
5     v_total_buget NUMBER;
6     v_nr_experți NUMBER;
7
8     exc_fonduri_insuficiente EXCEPTION;
9     exc_expertiza_limitata EXCEPTION;
10
11    BEGIN
12        SELECT SUM(f.buget), COUNT(DISTINCT er.id_expert)
13        INTO v_total_buget, v_nr_experți
14        FROM monument m
15        JOIN restaurare r ON m.id_monument = r.id_monument
16        JOIN finantare_restaurare fr ON r.id_restaurare = fr.id_restaurare
17        JOIN finantare f ON fr.id_finantare = f.id_finantare
18        JOIN expert_restaurare er ON r.id_restaurare = er.id_restaurare
19        WHERE m.id_monument = p_id_monument
20        GROUP BY m.id_monument;
21
22        IF v_total_buget < p_prag_financiar THEN
23            RAISE exc_fonduri_insuficiente;
24        END IF;
25
26        IF v_nr_experți < 2 THEN
27            RAISE exc_expertiza_limitata;
28        END IF;
29
30        DBMS_OUTPUT.PUT_LINE('Audit trecut cu succes. Buget: ' ||
31        v_total_buget || ' RON, Experti: ' || v_nr_experți);
32
33    EXCEPTION
34        WHEN NO_DATA_FOUND THEN
35            DBMS_OUTPUT.PUT_LINE('Eroare: Monumentul nu are restaurari
active, finantari sau experti alocati.');
36        WHEN exc_fonduri_insuficiente THEN
```

```

35     DBMS_OUTPUT.PUT_LINE('Alerta: Fonduri sub pragul minim ( ' || 
36     v_total_buget || ' < ' || p_prag_financiar || ') .');
37     WHEN exc_expertiza_limitata THEN
38         DBMS_OUTPUT.PUT_LINE('Alerta: Echipa tehnica insuficienta (' || 
39         v_nr_experti || ' experti).');
40     WHEN OTHERS THEN
41         DBMS_OUTPUT.PUT_LINE('Eroare necunoscuta: ' || SQLERRM);
42 END;
43 /

```

Listing 9.1: Procedura PL/SQL

Apel și Rezultat

```

1 SET SERVEROUTPUT ON;
2 BEGIN
3     DBMS_OUTPUT.PUT_LINE(' CAZ 1: Succes (Castelul Peles, Buget mic
cerut)');
4     audit_complex_monument(1, 1000);
5
6     DBMS_OUTPUT.PUT_LINE(' CAZ 2: Exceptie Proprie 1 - Fonduri
Insuficiente');
7     audit_complex_monument(1, 999999999);
8
9     DBMS_OUTPUT.PUT_LINE(' CAZ 3: Exceptie Proprie 2 - Expertiza
Limitata (Voronet)');
10    audit_complex_monument(4, 100);
11
12    DBMS_OUTPUT.PUT_LINE(' CAZ 4: Exceptie Sistem - NO_DATA_FOUND ');
13    % audit_complex_monument(99, 1000);
14 END;
15 /

```

```

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: Exceptie Proprie 1 - Fonduri Insuficiente
Alerta: Fonduri sub pragul minim (15600000 < 999999999).
CAZ 3: Exceptie Proprie 2 - Expertiza Limitata (Voronet)
Alerta: Echipa tehnica insuficienta (1 experti).
CAZ 4: Exceptie 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 /
```

Declansare

```
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 declansare 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

```
1 CREATE OR REPLACE TRIGGER trg_blocare_redeschidere
2 BEFORE UPDATE ON restaurare
3 FOR EACH ROW
4 BEGIN
5     IF :OLD.stadiu = 'Finalizat'
6         AND :NEW.stadiu IN ('In executie', 'Planificat') THEN
7         RAISE_APPLICATION_ERROR(
8             -20020,
9             'Eroare: Un proiect finalizat nu poate fi redeschis!')
10    );
11 END IF;
12 END;
13 /
```

Declansare

```
1 UPDATE restaurare
2 SET stadiu = 'In executie'
3 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 declansare trigger

12. Trigger LDD

Enunțul Problemei

Se definește un tabel de log numit LOG_LDD și un trigger LDD la nivel de schemă. Triggerul 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.

```
1 CREATE TABLE log_ldd (
2     utilizator      VARCHAR2(30),
3     data_actiune    DATE,
4     tip_eveniment   VARCHAR2(30),
5     nume_object    VARCHAR2(50)
6 );
7
8 CREATE OR REPLACE TRIGGER trg_audit_ldd
9 AFTER CREATE OR DROP OR ALTER ON SCHEMA
10 BEGIN
11     INSERT INTO log_ldd (utilizator, data_actiune, tip_eveniment,
12                         nume_object)
13         VALUES (USER, SYSDATE, ORA_SYSEVENT, ORA_DICT_OBJ_NAME);
14 END;
15 /
```

Declansare

```
1 CREATE TABLE test_temp (id NUMBER);
2 DROP TABLE test_temp;
3
4 SELECT * FROM log_ldd;
```

	UTILIZATOR	DATA_ACTIUNE	TIP_EVENIMENT	NUME_OBJECT
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 declansare trigger LDD

13. Pachet PL/SQL

Enunțul Problemei

Să se creeze un pachet numit `pkg_audit_rapid` pentru verificarea expertilor. 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 experti, cealaltă afișează un raport scurt).

Specificația Pachetului

```
1 CREATE OR REPLACE PACKAGE pkg_audit_rapid AS
2   TYPE t_lista_id IS TABLE OF NUMBER;
3   TYPE t_stats    IS RECORD (nr_proiecte NUMBER);
4
5   FUNCTION există_expert(p_id NUMBER) RETURN BOOLEAN;
6   FUNCTION nr_proiecte(p_id NUMBER) RETURN NUMBER;
7
8   PROCEDURE verifica_lista(p_lista t_lista_id);
9   PROCEDURE raport_scurt(p_id NUMBER);
10  END;
11 /
```

Corful Pachetului

```
1 CREATE OR REPLACE PACKAGE BODY pkg_audit_rapid AS
2
3   FUNCTION există_expert(p_id NUMBER) RETURN BOOLEAN IS
4     v_chk NUMBER;
5   BEGIN
6     SELECT COUNT(*) INTO v_chk FROM expert WHERE id_expert = p_id;
7     RETURN v_chk > 0;
8   END;
9
10  FUNCTION nr_proiecte(p_id NUMBER) RETURN NUMBER IS
11    v_count NUMBER;
12  BEGIN
13    SELECT COUNT(*) INTO v_count FROM expert_restaureare WHERE
14      id_expert = p_id;
15    RETURN v_count;
16  END;
17
18  PROCEDURE verifica_lista(p_lista t_lista_id) IS
19  BEGIN
20    FOR i IN 1..p_lista.COUNT LOOP
21      IF există_expert(p_lista(i)) THEN
22        DBMS_OUTPUT.PUT_LINE('ID ' || p_lista(i) || ' valid.');
23      ELSE
24        DBMS_OUTPUT.PUT_LINE('ID ' || p_lista(i) || ' INEXISTENT
. ');
25      END IF;
```

```

25     END LOOP;
26
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.
33     nr_proiecte || ' proiecte.');
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
4     (101, 999);
5 BEGIN
6     pkg_audit_rapid.verifica_lista(v_lista);
7     pkg_audit_rapid.raport_scurt(101);
8 END;
9 /

```

```

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