

Sisteme de Gestiune a Bazelor de Date

-

School Schedule

Ana-Maria Comorașu

December, 2020

## Contents

<b>1</b>	<b>Prezentarea bazei de date</b>	<b>3</b>
1.1	Tehnologii folosite . . . . .	3
1.2	Scopul proiectului . . . . .	3
<b>2</b>	<b>Diagrama Entitate-Relație</b>	<b>4</b>
<b>3</b>	<b>Diagrama Conceptuală</b>	<b>5</b>
<b>4</b>	<b>Implementarea bazei de date în Oracle</b>	<b>6</b>
4.1	Crearea bazei de date locală . . . . .	6
4.2	Adăugarea tabelelor și a constrângerilor de integritate . . . . .	6
4.2.1	Entități simple . . . . .	6
4.2.2	Entități care au în compoziție chei externe . . . . .	6
4.2.3	Tabele asociative . . . . .	7
4.3	Tabelele rezultat din baza de date . . . . .	7
<b>5</b>	<b>Popularea bazei de date cu informații</b>	<b>8</b>
5.1	Metoda Excel . . . . .	8
5.2	Popularea tabelor prin rezolvarea cerințelor . . . . .	8
<b>6</b>	<b>Subprogram ce utilizează un tip de colecție</b>	<b>8</b>
6.1	Enunț . . . . .	8
6.2	Codul PL/SQL . . . . .	8
6.3	Rularea codului . . . . .	9
<b>7</b>	<b>Subprogram ce utilizează un tip de cursor</b>	<b>9</b>
7.1	Enunț . . . . .	9
7.2	Codul PL/SQL . . . . .	9
7.3	Rularea codului . . . . .	10
<b>8</b>	<b>Subprogram de tip funcție care utilizează 3 tabele</b>	<b>10</b>
8.1	Enunț . . . . .	11
8.2	Codul PL/SQL . . . . .	11
8.3	Rularea codului . . . . .	12
<b>9</b>	<b>Subprogram de tip procedură care utilizează 5 tabele</b>	<b>12</b>
9.1	Enunț . . . . .	12
9.2	Codul PL/SQL . . . . .	12
9.3	Rularea codului . . . . .	13
<b>10</b>	<b>Trigger LMD la nivel de comandă</b>	<b>13</b>
10.1	Enunț . . . . .	13
10.2	Codul PL/SQL . . . . .	14
10.3	Rularea codului . . . . .	14
<b>11</b>	<b>Trigger LMD la nivel de linie</b>	<b>15</b>
11.1	Enunț . . . . .	15
11.2	Codul PL/SQL . . . . .	15
11.3	Rularea codului . . . . .	15

<b>12 Trigger LDD</b>	<b>16</b>
12.1 Enunț . . . . .	16
12.2 Codul PL/SQL . . . . .	16
12.3 Rularea codului . . . . .	16
<b>13 Pachet ce conține obiectele definite în cadrul proiectului</b>	<b>16</b>
<b>14 Pachet care include tipuri de date complexe și obiecte necesare pentru acțiuni integrate</b>	<b>20</b>
14.1 Scopul pachetului . . . . .	20
14.2 Codul PL/SQL . . . . .	20
14.3 Rularea pachetului . . . . .	22

# 1 Prezentarea bazei de date

## 1.1 Tehnologii folosite

Pentru proiectul din cadrul materiei Sisteme de Gestiune a Bazelor de date, voi folosi versiunea **o11g** a *Oracle Database*.

Aplicații folosite sunt:

- *Oracle SQL Developer*
- *Oracle Database Express Edition*

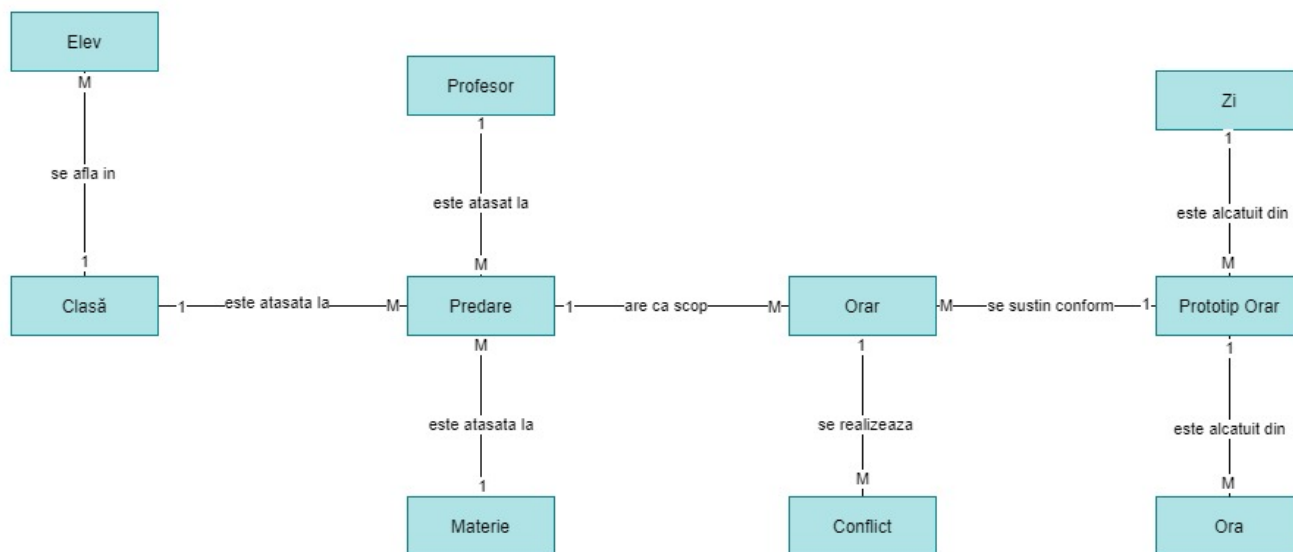
Am detaliat mai multe despre *Oracle Database Express Edition*, crearea bazei de date, oferirea drepturilor în primul referat pentru curs, intitulat *Utilizarea Oracle Express pentru crearea unei Baze de Date*.

## 1.2 Scopul proiectului

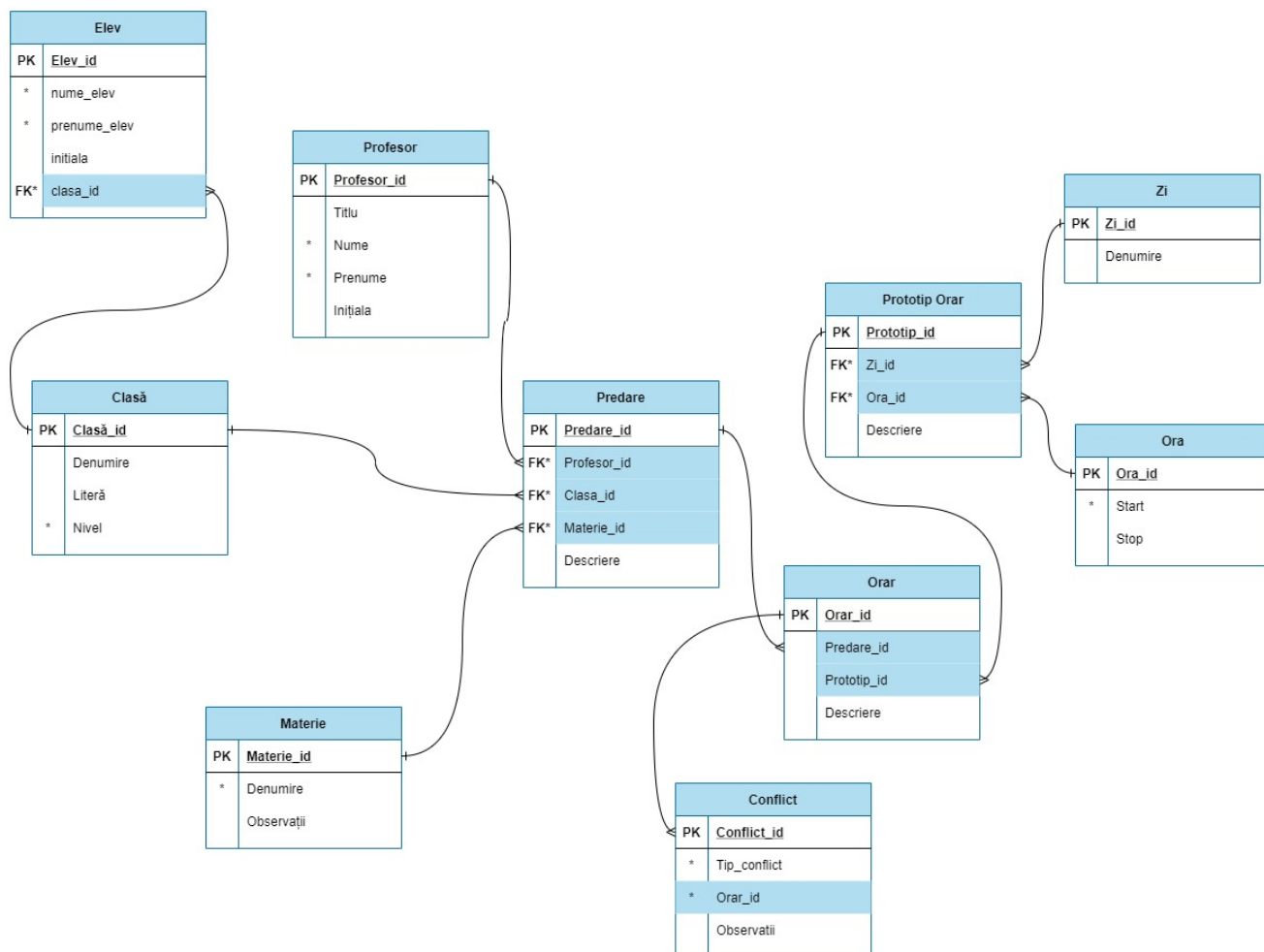
Ideea proiectului este realizarea/generarea unui orar al unui liceu. Aceasta cuprinde următoarele entități:

- Entitatea **Profesor**, care va fi alcătuită din lista cu profesori, persoane care predau una sau mai multe materii la una sau mai multe clase, din cadrul aceleiași școli.
- Entitatea **Elev**, alcătuită din lista cu elevii întregii școli, persoane care fac parte dintr-o singură clasă.
- **Clasa**, entitate care grupează mai mulți elevi, în funcție de nivel. Pe un singur nivel se pot afla mai multe clase, dacă numărul total de elevi din cadrul aceluiași nivel depășește o anumită limită (în general, 20-30).
- **Materie**, care poate fi predată mai multor nivele.
- De partea cealaltă se află entitățile **Zi** și **Oră**. Prima entitate se referă la ziua din săptămână în care se va desfășura programul, iar orele vor fi un model pentru fiecare zi din săptămână. Spre exemplu, ora 1 va începe la 8:15, ora 2 la 9:15 etc.
- De la cele 2 entități menționate mai sus, se va crea o tabelă asociativă numită **Prototip Orar**, prin care se va crea programul disponibil de lucru/învățat pentru fiecare zi din săptămână.
- De partea cealaltă, se află tabela asociativă **Predare**, care va fi constrânsă de 3 chei externe, pentru fiecare dintre entitățile Profesor, Clasă și Materie. Am realizat această tabelă asociativă din cele 3 entități, întrucât un profesor poate predă la mai multe clase, respectiv poate predă mai multe materii, iar o clasă studiază mai multe materii, cu profesori diferiți.
- Cele 2 tabele asociative vor fi, de asemenea, unite în tabela **Orar**, care va fi principalul scop al proiectului, anume crearea unui orar omogen al unui liceu.
- Întrucât tabela orar nu poate constrânge compatibilitatea tuturor orelor, pentru a putea crea un orar omogen, am decis adăugarea entității **Conflict**, care va reține întotdeauna conflictele dintre două ore din orar, precum și tipul lor (conflict la clasă, conflict la profesor). Predarea trebuie să fie unică între profesor și clasă, astfel încât în orarul corespondent unei ore din săptămână, un profesor să aibă oră la o singură clasă, iar o clasă să aibă asignată o singură oră.
- Dacă orarul este perfect generat, la final nu trebuie să existe conflicte.

## 2 Diagrama Entitate-Relație



### 3 Diagrama Conceptuală



## 4 Implementarea bazei de date în Oracle

### 4.1 Crearea bazei de date locală

Am creat o bază de date local folosind **Oracle Database Express Edition**, despre care am detaliat mai mult în primul meu referat pentru curs, unde am oferit și privilegiile aferente.

### 4.2 Adăugarea tabelelor și a constrângerilor de integritate

#### 4.2.1 Entități simple

Acestea sunt entitățile fără chei externe.

```
1  -- PROFESOR --
2  CREATE TABLE profesor(
3      profesor_id NUMBER(10) PRIMARY KEY,
4      titlu VARCHAR2(10),
5      nume VARCHAR2(20) NOT NULL,
6      prenume VARCHAR2(20) NOT NULL,
7      initiala VARCHAR2(3)
8  );
9
10 -- MATERIE --
11 CREATE TABLE materie(
12     materie_id VARCHAR2(4) PRIMARY KEY,
13     denumire VARCHAR2(20) NOT NULL,
14     observatii VARCHAR2(20)
15 );
16
17 -- CLASA --
18 CREATE TABLE clasa (
19     clasa_id NUMBER(10) PRIMARY KEY,
20     denumire VARCHAR2(20),
21     litera VARCHAR2(1),
22     nivel NUMBER(10) NOT NULL
23 );
24
25 -- ZI --
26 CREATE TABLE zi(
27     zi_id NUMBER(10) PRIMARY KEY,
28     nume VARCHAR2(20)
29 );
30
31 -- ORA --
32 CREATE TABLE ora(
33     ora_id NUMBER(10) PRIMARY KEY,
34     ora_start VARCHAR2(20) NOT NULL,
35     ora_stop VARCHAR2(20)
36 );
```

#### 4.2.2 Entități care au în compoziție chei externe

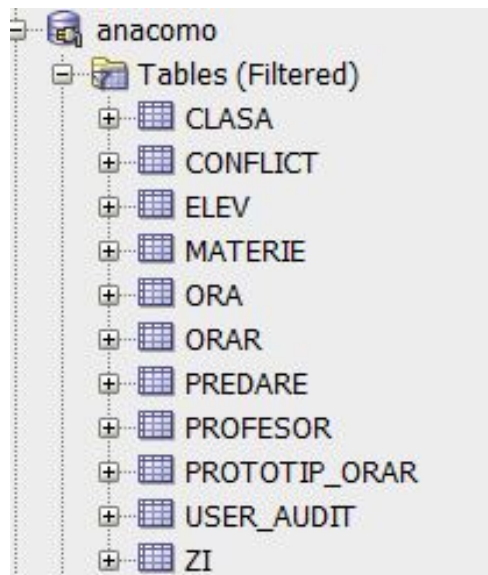
Entitatea **Elev** conține o referință către id-ul clasei.

```
1  -- ELEV --
2  CREATE TABLE elev(
3      elev_id NUMBER(10) PRIMARY KEY,
4      nume VARCHAR2(20) NOT NULL,
5      prenume VARCHAR2(20) NOT NULL,
6      initiala VARCHAR2(3),
7      clasa_id NUMBER(10) REFERENCES clasa(clasa_id) ON DELETE SET NULL
8  );
```

### 4.2.3 Tabele asociative

```
1  -- PROTOTIP ORAR --
2  CREATE TABLE prototip_orar(
3      prototip_id NUMBER(10) PRIMARY KEY,
4      zi_id NUMBER(10) REFERENCES zi(zi_id) ON DELETE SET NULL,
5      ora_id NUMBER(10) REFERENCES ora(ora_id) ON DELETE SET NULL,
6      descriere VARCHAR2(20)
7  );
8
9  -- PREDARE --
10 CREATE TABLE predare(
11     predare_id VARCHAR2(20) PRIMARY KEY,
12     profesor_id NUMBER(10) REFERENCES profesor(profesor_id) ON DELETE SET NULL,
13     clasa_id NUMBER(10) REFERENCES clasa(clasa_id) ON DELETE SET NULL,
14     materie_id VARCHAR2(4) REFERENCES materie(materie_id) ON DELETE SET NULL,
15     descriere VARCHAR2(20)
16 );
17
18 -- ORAR --
19 CREATE TABLE orar(
20     orar_id VARCHAR(20) PRIMARY KEY,
21     predare_id VARCHAR2(20) NOT NULL
22     REFERENCES predare(predare_id) ON DELETE SET NULL,
23     prototip_id NUMBER(10) NOT NULL
24     REFERENCES prototip_orar(prototip_id) ON DELETE SET NULL,
25     descriere VARCHAR(20)
26 );
27
28 -- CONFLICT --
29 CREATE TABLE conflict(
30     conflict_id VARCHAR(20) PRIMARY KEY,
31     conflict_tip VARCHAR(20) NOT NULL,
32     orar_id VARCHAR(20) REFERENCES orar(orar_id) ON DELETE SET NULL,
33     observatii VARCHAR(20)
34 );
```

### 4.3 Tabelele rezultat din baza de date





## 5 Popularea bazei de date cu informații

### 5.1 Metoda Excel

Pentru popularea bazei de date cu înregistrări, am folosit metoda importării datelor dintr-un fișier Excel, datorită funcțiilor accesibile, precum și ușurința introducerii în tabel, deoarece Excel este o aplicație care se bazează pe WYSIWYG (What You See Is What You Get). Am detaliat mai multe despre acest subiect în cel de-al doilea referat de la curs.

### 5.2 Popularea tabelelor prin rezolvarea cerințelor

Înainte de rezolvarea cerințelor, toate entitățile au fost completate cu informații, mai puțin tabelele **Orar** și **Conflict**, în care intenționez să introduc înregistrări prin cerințele proiectului.

## 6 Subprogram ce utilizează un tip de colecție

### 6.1 Enunț

După ce s-au generat în tabela **Orar** înregistrări, să se creeze un subprogram care introduce în tabela **Conflict** toate conflictele ce se regăsesc în tabela **Orar** (fie conflicte la clasă, fie la profesor).

### 6.2 Codul PL/SQL

```
1 SET SERVEROUTPUT ON;
2 /
3 CREATE OR REPLACE TYPE confl_type AS OBJECT(
4     pid     NUMBER(10),
5     oid     VARCHAR2(20),
6     pfid    NUMBER(10),
7     cid     NUMBER(10),
8     rnum    VARCHAR2(20)
9 );
10 /
11 CREATE OR REPLACE PROCEDURE adauga_conflict
12 AS
13     TYPE v_confl IS TABLE OF confl_type INDEX BY PLS_INTEGER;
14     t     v_confl;
15     CURSOR c IS
16         SELECT prototip_id as pid,
17                orar_id as oid,
18                profesor_id as pfid,
19                clasa_id as cid,
20                to_number(row_number() over ( order by prototip_id )) as rnum
21     FROM orar
22     JOIN predare USING (predare_id)
23     JOIN prototip_orar USING (prototip_id)
24     ORDER BY clasa_id;
25     cnt NUMBER := 1;
26     nr   NUMBER := 1000;
27 BEGIN
28     FOR it IN c LOOP
29         t(cnt) := confl_type(it.pid, it.oid, it.pfid, it.cid, it.rnum);
30         cnt := cnt + 1;
31     END LOOP;
32     DELETE FROM conflict;
33     cnt := cnt - 1;
34     FOR i IN 1..(cnt-1) LOOP
35         FOR j IN (i+1)..cnt LOOP
36             IF t(i).pid = t(j).pid and t(i).cid = t(j).cid THEN
37                 INSERT INTO conflict (conflict_id, conflict_tip, orar_id, observatii)
38                 VALUES('OC' || to_char(nr), 'clasa', t(i).oid, null);
39                 DBMS_OUTPUT.PUT_LINE('OC' || to_char(nr) || ' clasa ' || t(i).oid);
40                 nr := nr + 1;
41                 INSERT INTO conflict (conflict_id, conflict_tip, orar_id, observatii)
```

```

42         VALUES('OC' || to_char(nr), 'clasa', t(j).oid, null);
43         DBMS_OUTPUT.PUT_LINE('OC' || to_char(nr) || ' clasa ' || t(j).oid);
44         nr := nr + 1;
45     ELSIF t(i).pid = t(j).pid and t(i).pfid = t(j).pfid THEN
46         INSERT INTO conflict (conflict_id, conflict_tip, orar_id, observatii)
47         VALUES('OC' || to_char(nr), 'profesor', t(i).oid, null);
48         DBMS_OUTPUT.PUT_LINE('OC' || to_char(nr) || ' profesor ' || t(i).oid);
49         nr := nr + 1;
50         INSERT INTO conflict (conflict_id, conflict_tip, orar_id, observatii)
51         VALUES('OC' || to_char(nr), 'profesor', t(j).oid, null);
52         DBMS_OUTPUT.PUT_LINE('OC' || to_char(nr) || ' profesor ' || t(j).oid);
53         nr := nr + 1;
54     END IF;
55 END LOOP;
56 END LOOP;
57 END;
58 /
59 EXECUTE ADAUGA_CONFLICT;
60 ROLLBACK;

```

### 6.3 Rularea codului

```

PL/SQL procedure successfully completed.

Procedure ADAUGA_CONFLICT compiled

```

## 7 Subprogram ce utilizează un tip de cursor

Definiți un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

### 7.1 Enunț

În tabela **Ora**, au fost inserate 7 înregistrări, reprezentând maximul de ore pe care orarul le are la dispoziție în fiecare zi. Conform acestuia, elevii încep programul de la 08:15 și au la dispoziție 45 de minute într-o oră de curs, cu 10 minute de pauză.

Odată cu venirea pandemiei și închiderea școlilor, părinții au sesizat probleme în ceea ce privește programul copiilor de la școală. Aceștia nu au destul de mult timp pentru a se ”despărți de computer” și solicită conducerii școlii să prelungească pauza cu 5 minute.

Scrieți un subprogram de tip procedură în care să se prelucreze tabela **Ora**, conform cerințelor conducerii liceului.

### 7.2 Codul PL/SQL

```

1 CREATE OR REPLACE PROCEDURE modificare_orar
2 IS
3     c_id ora.ora_id%TYPE;
4     c_start ora.ora_start%TYPE;
5     c_stop ora.ora_stop%TYPE;
6     CURSOR c IS
7         SELECT *
8         FROM ora
9         FOR UPDATE OF ora_start, ora_stop NOWAIT;
10    min5 NUMBER;
11    i NUMBER;

```

```

12     h VARCHAR(10);
13 BEGIN
14     min5 := 1/288;
15     i := 0;
16     OPEN c;
17     LOOP
18         FETCH c INTO c_id, c_start, c_stop;
19         EXIT WHEN c%NOTFOUND;
20         UPDATE ora
21         SET ora_start = to_char(to_date(c_start, 'HH24:MI') + i*1/288, 'HH24:MI'),
22             ora_stop = to_char(to_date(c_stop, 'HH24:MI') + i*1/288, 'HH24:MI')
23         WHERE CURRENT OF c;
24         i := i + 1;
25     END LOOP;
26     CLOSE c;
27 END;
28 /
29 EXECUTE modificare_orar;
30 SELECT * FROM ora;
31 ROLLBACK;

```

### 7.3 Rularea codului

În prima stângă a figurii este tabela Ora înainte de executarea procedurii, iar în partea dreaptă este aceeași tabelă, modificată după executarea procedurii.

60	ora_stop = to_cha	60	ora_stop = to_char
61	WHERE CURRENT OF c;	61	WHERE CURRENT OF c;
62	i := i + 1;	62	i := i + 1;
63	END LOOP;	63	END LOOP;
64	CLOSE c;	64	CLOSE c;
65	END;	65	END;
66	/	66	/
67	EXECUTE modificare_orar;	67	EXECUTE modificare_orar;
68	select * from ora;	68	select * from ora;
69	rollback;	69	rollback;

ORA_ID	ORA_START	ORA_STOP
1	1 08:15	09:00
2	2 09:10	09:55
3	3 10:05	10:50
4	4 11:00	11:45
5	5 11:55	12:40
6	6 12:50	13:35
7	7 13:45	14:30

ORA_ID	ORA_START	ORA_STOP
1	1 08:15	09:00
2	2 09:15	10:00
3	3 10:15	11:00
4	4 11:15	12:00
5	5 12:15	13:00
6	6 13:15	14:00
7	7 14:15	15:00

## 8 Subprogram de tip funcție care utilizează 3 tabele

Definiți un subprogram stocat de tip funcție care să utilizeze 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să tratați toate cazurile tratate.

## 8.1 Enunț

La școala *Hogwarts*, o familie are mulți copii înscriși. Săptămâna viitoare, familia Weasley va fi plecată în Transilvania să îmbânzească dragoni, așa că doamna Molly Weasley trebuie să trimită scrisori personale către profesori prin care să îi învoiască pe copii de la școală (folosind bufnița mesager, bineînțeles). Să se creeze o funcție stocată cu un parametru (acesta fiind id-ul unuia dintre copii) care să returneze numărul de profesori la care trebuie trimise scrisori. Bineînțeles, dacă doi sau mai mulți copii vor face cu același profesor, se va trimite o singură scrisoare profesorului (se va face și economie de hârtie, iar familia Weasley oricum nu e așa bine înstărită). De asemenea, să se afișeze și copiii care susțin orele cu fiecare profesor.

## 8.2 Codul PL/SQL

```
1 SET SERVEROUTPUT ON;
2 /
3 CREATE OR REPLACE FUNCTION suna_profesori(id NUMBER)
4     RETURN VARCHAR2
5 IS
6     CURSOR c IS
7         SELECT DISTINCT pf.titlu tit, pf.prenume ppn, pf.nume pn, e.nume en, e.prenume ep
8         FROM elev e
9         JOIN CLASA c1 USING (CLASA_ID)
10        JOIN PREDARE p USING (CLASA_ID)
11        JOIN PROFESOR pf USING (PROFESOR_ID)
12        WHERE e.nume IN (SELECT nume
13                        FROM elev
14                        WHERE elev_id = id)
15        ORDER BY pf.NUME;
16    prec    VARCHAR2(20) := 'null';
17    nr      NUMBER(10) := 0;
18    text    VARCHAR2(32767) := '';
19 BEGIN
20     FOR i IN c LOOP
21         IF prec = 'null' THEN
22             DBMS_OUTPUT.NEW_LINE();
23             DBMS_OUTPUT.PUT(i.tit || ' ' || i.ppn || ' ' || i.pn || ' sustine ore cu ' || i.ep);
24             nr := nr + 1;
25         ELSIF i.pn = prec THEN
26             DBMS_OUTPUT.PUT(', ' || i.ep);
27         ELSE
28             DBMS_OUTPUT.NEW_LINE();
29             DBMS_OUTPUT.PUT(i.tit || ' ' || i.ppn || ' ' || i.pn || ' sustine ore cu ' || i.ep);
30             nr := nr + 1;
31         END IF;
32         prec := i.pn;
33     END LOOP;
34     DBMS_OUTPUT.PUT_LINE(nr-1);
35     RETURN 'Aveti de sunat ' || to_char(nr-1) || ' profesori';
36 EXCEPTION
37     WHEN OTHERS THEN
38         DBMS_OUTPUT.PUT_LINE (SQLCODE || ' - ' || SQLERRM);
39 END suna_profesori;
40 /
41 BEGIN
42     DBMS_OUTPUT.PUT_LINE(suna_profesori(114));
43 END;
44 /
45 BEGIN
46     DBMS_OUTPUT.PUT_LINE(suna_profesori(NULL));
47 END;
```

## 8.3 Rularea codului

```
73 BEGIN
74     DBMS_OUTPUT.PUT_LINE(suna_profesori(114));
75 END;
```

Script Output x Query Result x

Task completed in 0.049 seconds

MISS AURORA SINISTRA sustine ore cu CHARLIE, FRED, GEORGE, GINNY, RONALD  
MR HORACE SLUGHORN sustine ore cu CHARLIE, FRED, GEORGE, GINNY, RONALD  
MR SEVERUS SNAPE sustine ore cu FRED, GEORGE, RONALD  
MISS DOLORES UMBRIDGE sustine ore cu CHARLIE, FRED, GEORGE, GINNY, RONALD  
MISS SEPTIMA VECTOR sustine ore cu CHARLIE17  
Aveti de sunat 17 profesori

PL/SQL procedure successfully completed.

## 9 Subprogram de tip procedură care utilizează 5 tabele

### 9.1 Enunț

Să se creeze un subprogram de tip procedură care generează înregistrări pentru tabela **Orar**, având drept argument id-ul clasei la care dorim generarea orarului. Se va ține cont de următoarele aspecte:

- Nu vor putea exista mai mult de 2 conflicte de același tip la același profesor.
- Se va genera un anume procent de ore față de cele din descriere. Spre exemplu, în norma de profesor, doar aproximativ 60% din orele lui vor fi de predare efectivă, pentru a lăsa loc de conflicte.

### 9.2 Codul PL/SQL

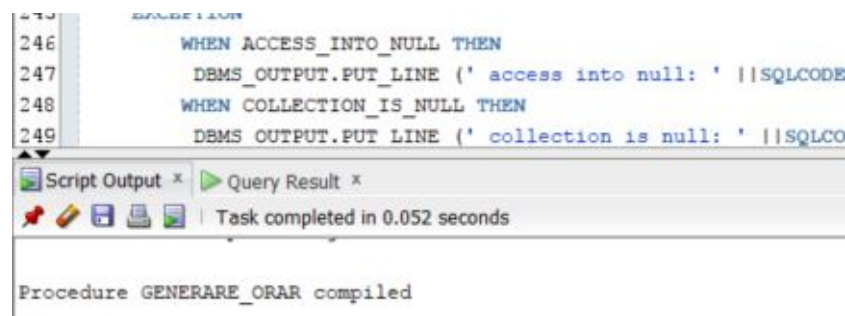
```
1 CREATE OR REPLACE PROCEDURE generare_orar(cls_id NUMBER, procentaj NUMBER)
2 AS
3     TYPE tablou_orar IS TABLE OF orar_type INDEX BY PLS_INTEGER;
4     t tablou_orar;
5     TYPE vect IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
6     fr vect;
7     CURSOR c IS
8         SELECT predare_id pid, ROUND(descriere * procentaj/100) nr, denumire, titlu,
9             prenume, nume
10        FROM predare
11       JOIN profesor USING (profesor_id)
12       JOIN materie USING (materie_id)
13      WHERE clasa_id = cls_id;
14     CURSOR p IS
15         SELECT prototip_id pidp, nume, ora_start, ora_stop
16        FROM prototip_orar
17       JOIN zi using (zi_id)
18       JOIN ora using (ora_id)
19      ORDER BY DBMS_RANDOM.RANDOM;
20     numar_ore NUMBER;
21     k NUMBER := 1;
22     it NUMBER := 1;
23 BEGIN
24     DELETE FROM orar
25    WHERE predare_id IN
26        (SELECT predare_id FROM predare WHERE clasa_id = cls_id);
```

```

26 SELECT COUNT(*)
27 INTO numar_ore
28 FROM prototip_orar;
29 FOR cnt IN 1..numar_ore LOOP
30     fr(cnt) := 0;
31 END LOOP;
32 FOR i IN c LOOP
33     k := 0;
34     FOR j IN p LOOP
35         IF k >= i.nr THEN
36             EXIT;
37         END IF;
38         IF fr(j.pidp) < 2 THEN
39             fr(j.pidp) := fr(j.pidp) + 1;
40             t(it) := orar_type(to_char(j.pidp)||i.pid, i.pid, j.pidp, null);
41             DBMS_OUTPUT.PUT_LINE(i.denumire||' cu '||i.titlu||' '||i.prenume||' '||i.
nume||', '||j.nume||' de la '||j.ora_start||' la '||j.ora_stop);
42             INSERT INTO orar(orar_id, predare_id, prototip_id, descriere)
43             VALUES (to_char(j.pidp)||i.pid, i.pid, j.pidp, null);
44             it := it + 1;
45             k := k + 1;
46         END IF;
47     END LOOP;
48 END LOOP;
49 EXCEPTION
50 WHEN OTHERS THEN
51     DBMS_OUTPUT.PUT_LINE (' others: ' ||SQLCODE || '- ' || SQLERRM);
52 END;
53 /
54 EXECUTE generare_orar(91, 60);
55 SELECT * from orar
56 JOIN predare using (predare_id)
57 WHERE clasa_id = 91;
58 COMMIT;

```

### 9.3 Rularea codului



## 10 Trigger LMD la nivel de comandă

### 10.1 Enunț

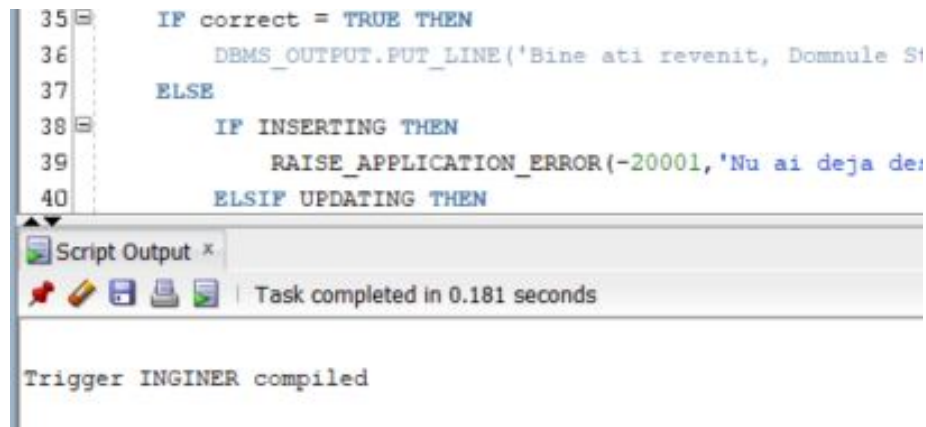
Liceul are un angajat care se ocupă de baza de date a liceului. Acesta are biroul pe același hol cu clasele de la profilul mate-info, care au învățat la opționalul de informatică să lucreze cu baze de date. Deoarece biroul inginerului nu are cheie, iar elevii de liceu sunt mereu puși pe obraznicii în pauze, domnul inginer Ștefan dorește să se asigure că niciun elev nu intră în pauză în cabinetul lui și face modificări la baza de date (de exemplu, elevii să își pună în orar mai puține ore de chimie).

Ajutați-l pe domnul Ștefan să își securizeze foarte bine baza de date, creând un declanșator, astfel încât să se poată realiza comenzi de *Insert*, *Update* sau *Delete* numai în programul de lucru, anume zilele săptămânii din tabela **Zi** și intervalele orare din tabela **Ora**.

## 10.2 Codul PL/SQL

```
1 SET SERVEROUTPUT ON;
2 CREATE OR REPLACE TRIGGER inginer
3 BEFORE INSERT OR UPDATE OR DELETE ON orar
4 DECLARE
5     correct          BOOLEAN := FALSE;
6     nume_zi          zi.nume%TYPE;
7     ora_inc          ora.ora_start%TYPE;
8     ora_fin          ora.ora_stop%TYPE;
9     nume_azi         VARCHAR (20);
10    CURSOR weekly_prog IS
11        SELECT nume, ora_start, ora_stop
12        FROM prototip_orar
13        JOIN zi USING (zi_id)
14        JOIN ora USING (ora_id);
15 BEGIN
16     SELECT TO_CHAR(SYSDATE, 'DAY', 'NLS_DATE_LANGUAGE = ROMANIAN')
17     INTO nume_azi
18     FROM dual;
19     OPEN weekly_prog;
20     LOOP
21         FETCH weekly_prog INTO nume_zi, ora_inc, ora_fin;
22         EXIT WHEN weekly_prog%NOTFOUND;
23         IF UPPER(nume_azi) = UPPER(nume_zi)
24         AND TO_DATE(ora_inc, 'HH24:MI') <= TO_DATE(TO_CHAR(SYSDATE, 'HH24:MI'), 'HH24:MI')
25         AND TO_DATE(ora_fin, 'HH24:MI') > TO_DATE(TO_CHAR(SYSDATE, 'HH24:MI'), 'HH24:MI')
26         THEN
27             correct := TRUE;
28         END IF;
29     END LOOP;
30     CLOSE weekly_prog;
31     IF correct = TRUE THEN
32         DBMS_OUTPUT.PUT_LINE('Bine ati revenit, Domnule Stefan!');
33     ELSE
34         IF INSERTING THEN
35             RAISE_APPLICATION_ERROR(-20001, 'Nu ai deja destule ore? Vrei mai multe?');
36         ELSIF UPDATING THEN
37             RAISE_APPLICATION_ERROR(-20002, 'Las-o balta, nu te las sa schimbi orele de
38 chimie cu sport');
39         ELSIF DELETING THEN
40             RAISE_APPLICATION_ERROR(-20003, 'Not on my watch!');
41         END IF;
42     END IF;
43 END;
44 /
45 DELETE FROM orar;
```

## 10.3 Rularea codului



```
35 IF correct = TRUE THEN
36     DBMS_OUTPUT.PUT_LINE('Bine ati revenit, Domnule S
37 ELSE
38     IF INSERTING THEN
39         RAISE_APPLICATION_ERROR(-20001, 'Nu ai deja de
40     ELSIF UPDATING THEN
```

Script Output x

Task completed in 0.181 seconds

Trigger INGINER compiled



## 11 Trigger LMD la nivel de linie

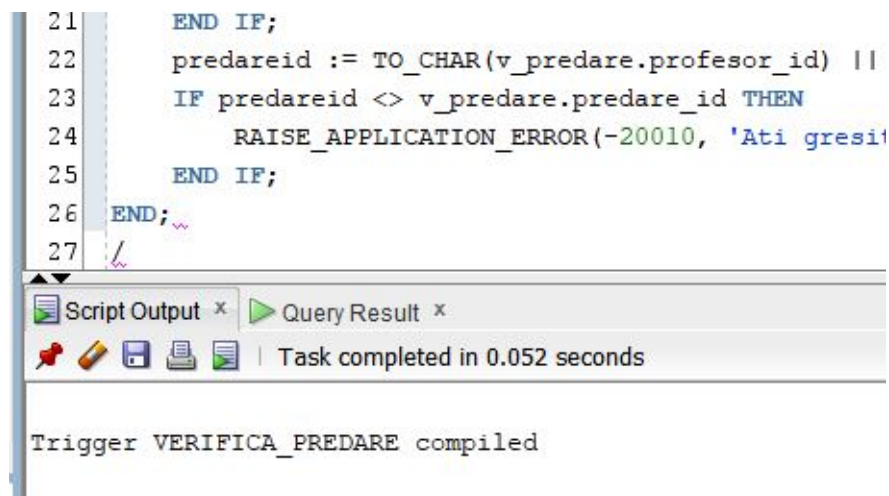
### 11.1 Enunț

Să se creeze un trigger la nivel de linie care ridică o eroare la aplicație de fiecare dată când domnul inginer Ștefan inserează greșit un rând în tabela predare. Regula inserării se poate observa în fișierul Excel (concatenare de id-uri chei externe, dar id-ul clasei are obligatoriu 3 caractere).

### 11.2 Codul PL/SQL

```
1 CREATE OR REPLACE TRIGGER verifica_predare
2 BEFORE INSERT OR UPDATE ON predare
3 FOR EACH ROW
4 DECLARE
5     v_predare    predare%ROWTYPE;
6     predareid    VARCHAR2(20);
7     cid          VARCHAR2(3);
8 BEGIN
9     v_predare.predare_id := :NEW.predare_id;
10    v_predare.profesor_id := :NEW.profesor_id;
11    v_predare.clasa_id := :NEW.clasa_id;
12    v_predare.materie_id := :NEW.materie_id;
13    v_predare.descriere := :NEW.descriere;
14    IF :NEW.clasa_id < 100 THEN
15        cid := '0' || TO_CHAR(:NEW.clasa_id);
16    ELSE
17        cid := TO_CHAR(:NEW.clasa_id);
18    END IF;
19    predareid := TO_CHAR(v_predare.profesor_id) || cid || v_predare.materie_id;
20    IF predareid <> v_predare.predare_id THEN
21        RAISE_APPLICATION_ERROR(-20010, 'Ati gresit id-ul, va rugam sa corectati id-ul cu
22        ' || predareid);
23    END IF;
24 END;
25 /
26 INSERT INTO predare(predare_id, profesor_id, clasa_id, materie_id, descriere)
27 VALUES ('AAAAAA', 27, 91, 'MAT', NULL);
28 SELECT * from predare
29 WHERE predare_id = 'AAAAAA';
30 ROLLBACK;
```

### 11.3 Rularea codului





## 12 Trigger LDD

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

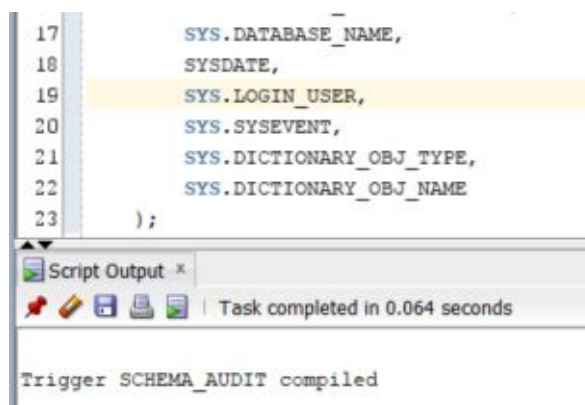
### 12.1 Enunț

Pentru că știm acum că baza de date pe care o administrează domnul inginer Ștefan poate fi modificată doar în timpul programului orelor de curs, mai sunt niște probleme. Elevii ar putea intra în biroul lui și în timpul orelor, atunci când domnul Ștefan este în celălalt corp al liceului, unde se află laboratoarele de informatică. Domnul Ștefan ar vrea să creeze o tabelă în care să aibă o evidență completă pentru toate operațiile care se fac pe baza de date, iar un trigger va insera date după fiecare astfel de procedură (dacă numărul de ore din descriere nu este null).

### 12.2 Codul PL/SQL

```
1 CREATE TABLE user_audit(  
2     bd_num     VARCHAR2(50),  
3     data       DATE,  
4     nume_user  VARCHAR2(30),  
5     operation  VARCHAR2(20),  
6     tip_obiect VARCHAR2(30),  
7     nume_obiect VARCHAR2(30)  
8 );  
9  
10 CREATE OR REPLACE TRIGGER schema_audit  
11 AFTER CREATE OR ALTER OR DROP ON SCHEMA  
12 BEGIN  
13     INSERT INTO user_audit VALUES (  
14         SYS.DATABASE_NAME,  
15         SYSDATE,  
16         SYS.LOGIN_USER,  
17         SYS.SYSEVENT,  
18         SYS.DICTIONARY_OBJ_TYPE,  
19         SYS.DICTIONARY_OBJ_NAME  
20     );  
21 END;
```

### 12.3 Rularea codului



## 13 Pachet ce conține obiectele definite în cadrul proiectului

### Codul PL/SQL

```
1 CREATE OR REPLACE PACKAGE pachet_anacomo AS  
2     PROCEDURE adauga_conflict;  
3     PROCEDURE modificare_orar;
```

```

4      FUNCTION suna_profesori(id NUMBER)
5          RETURN VARCHAR2;
6      PROCEDURE generare_orar(cls_id NUMBER, procentaj NUMBER);
7  END pachet_anacomo;
8  /
9  CREATE OR REPLACE PACKAGE BODY pachet_anacomo AS
10 ----- PRIMA PROCEDURA
11      PROCEDURE adauga_conflict
12  AS
13          TYPE v_confl IS TABLE OF confl_type INDEX BY PLS_INTEGER;
14          t v_confl;
15          CURSOR c IS
16              SELECT prototip_id as pid,
17                     orar_id as oid,
18                     profesor_id as pfid,
19                     clasa_id as cid,
20                     to_number(row_number() over ( order by prototip_id )) as rnum
21          FROM orar
22          JOIN predare USING (predare_id)
23          JOIN prototip_orar USING (prototip_id)
24          ORDER BY clasa_id;
25          cnt NUMBER := 1;
26          nr NUMBER := 1000;
27  BEGIN
28      FOR it IN c LOOP
29          t(cnt) := confl_type(it.pid, it.oid, it.pfid, it.cid, it.rnum);
30          cnt := cnt + 1;
31      END LOOP;
32      DELETE FROM conflict;
33      cnt := cnt - 1;
34      FOR i IN 1..(cnt-1) LOOP
35          FOR j IN (i+1)..cnt LOOP
36              IF t(i).pid = t(j).pid and t(i).cid = t(j).cid THEN
37                  INSERT INTO conflict (conflict_id, conflict_tip, orar_id, observatii)
38                  VALUES('OC' || to_char(nr), 'clasa', t(i).oid, null);
39                  DBMS_OUTPUT.PUT_LINE('OC' || to_char(nr) || ' clasa ' || t(i).oid);
40                  nr := nr + 1;
41                  INSERT INTO conflict (conflict_id, conflict_tip, orar_id, observatii)
42                  VALUES('OC' || to_char(nr), 'clasa', t(j).oid, null);
43                  DBMS_OUTPUT.PUT_LINE('OC' || to_char(nr) || ' clasa ' || t(j).oid);
44                  nr := nr + 1;
45              ELSIF t(i).pid = t(j).pid and t(i).pfid = t(j).pfid THEN
46                  INSERT INTO conflict (conflict_id, conflict_tip, orar_id, observatii)
47                  VALUES('OC' || to_char(nr), 'profesor', t(i).oid, null);
48                  DBMS_OUTPUT.PUT_LINE('OC' || to_char(nr) || ' profesor ' || t(i).oid);
49                  nr := nr + 1;
50                  INSERT INTO conflict (conflict_id, conflict_tip, orar_id, observatii)
51                  VALUES('OC' || to_char(nr), 'profesor', t(j).oid, null);
52                  DBMS_OUTPUT.PUT_LINE('OC' || to_char(nr) || ' profesor ' || t(j).oid);
53                  nr := nr + 1;
54              END IF;
55          END LOOP;
56      END LOOP;
57  END;
58 ----- A DOUA PROCEDURA
59  PROCEDURE modificare_orar
60  IS
61      c_id ora.ora_id%TYPE;
62      c_start ora.ora_start%TYPE;
63      c_stop ora.ora_stop%TYPE;
64      CURSOR c IS
65          SELECT *
66          FROM ora
67          FOR UPDATE OF ora_start, ora_stop NOWAIT;
68          min5 NUMBER;
69          i NUMBER;
70          h VARCHAR(10);
71  BEGIN
72      min5 := 1/288;
73

```

```

74     i := 0;
75     OPEN c;
76     LOOP
77         FETCH c INTO c_id, c_start, c_stop;
78         EXIT WHEN c%NOTFOUND;
79         UPDATE ora
80         SET ora_start = to_char(to_date(c_start, 'HH24:MI') + i*1/288, 'HH24:MI'),
81             ora_stop = to_char(to_date(c_stop, 'HH24:MI') + i*1/288, 'HH24:MI')
82         WHERE CURRENT OF c;
83         i := i + 1;
84     END LOOP;
85     CLOSE c;
86 END;
87 ----- a TREIA
88 FUNCTION suna_profesori(id NUMBER)
89     RETURN VARCHAR2
90 IS
91     CURSOR c IS
92         SELECT DISTINCT pf.titlu tit, pf.prenume ppn, pf.numa pn, e.numa en, e.prenume ep
93         FROM elev e
94         JOIN CLASA cl USING (CLASA_ID)
95         JOIN PREDARE p USING (CLASA_ID)
96         JOIN PROFESOR pf USING (PROFESOR_ID)
97         WHERE e.numa IN (SELECT nume
98                         FROM elev
99                         WHERE elev_id = id)
100        ORDER BY pf.NUMA;
101     prec VARCHAR2(20) := 'null';
102     nr    NUMBER(10) := 0;
103     text  VARCHAR2(32767) := '';
104 BEGIN
105     FOR i IN c LOOP
106         IF prec = 'null' THEN
107             DBMS_OUTPUT.NEW_LINE();
108             DBMS_OUTPUT.PUT(i.tit || ' ' || i.ppn || ' ' || i.pn || ' sustine ore cu ' || i.ep);
109             nr := nr + 1;
110         ELSIF i.pn = prec THEN
111             DBMS_OUTPUT.PUT(', ' || i.ep);
112         ELSE
113             DBMS_OUTPUT.NEW_LINE();
114             DBMS_OUTPUT.PUT(i.tit || ' ' || i.ppn || ' ' || i.pn || ' sustine ore cu ' || i.ep);
115             nr := nr + 1;
116         END IF;
117         prec := i.pn;
118     END LOOP;
119     DBMS_OUTPUT.PUT_LINE(nr-1);
120     RETURN 'Aveti de sunat ' || to_char(nr-1) || ' profesori';
121 EXCEPTION
122     WHEN OTHERS THEN
123         DBMS_OUTPUT.PUT_LINE (SQLCODE || ' - ' || SQLERRM);
124 END suna_profesori;
125 ----- A PATRA
126 PROCEDURE generare_orar(cls_id NUMBER, procentaj NUMBER)
127 AS
128     TYPE tablou_orar IS TABLE OF orar_type INDEX BY PLS_INTEGER;
129     t tablou_orar;
130     TYPE vect IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
131     fr vect;
132     CURSOR c IS
133         SELECT predare_id pid, ROUND(descriere * procentaj/100) nr, denumire, titlu,
134         prenume, nume
135         FROM predare
136         JOIN profesor USING (profesor_id)
137         JOIN materie USING (materie_id)
138         WHERE clasa_id = cls_id;
139     CURSOR p IS
140         SELECT prototip_id pidp, nume, ora_start, ora_stop
141         FROM prototip_orar
142         JOIN zi using (zi_id)
143         JOIN ora using (ora_id)

```

```

143      ORDER BY DBMS_RANDOM.VALUE;
144      numar_ore NUMBER;
145      k NUMBER := 1;
146      it NUMBER := 1;
147  BEGIN
148      DELETE FROM orar
149      WHERE predare_id IN
150      (SELECT predare_id FROM predare WHERE clasa_id = cls_id);
151      SELECT COUNT(*)
152      INTO numar_ore
153      FROM prototip_orar;
154      FOR cnt IN 1..numar_ore LOOP
155          fr(cnt) := 0;
156      END LOOP;
157      FOR i IN c LOOP
158          k := 0;
159          FOR j IN p LOOP
160              IF k >= i.nr THEN
161                  EXIT;
162              END IF;
163              IF fr(j.pidp) < 2 THEN
164                  fr(j.pidp) := fr(j.pidp) + 1;
165                  t(it) := orar_type(to_char(j.pidp)||i.pid, i.pid, j.pidp, null);
166                  DBMS_OUTPUT.PUT_LINE(i.denumire||' cu '||i.titlu||' '||i.prenume||' '||i.
nume||', '||j.nume||' de la '||j.ora_start||' la '||j.ora_stop);
167                  INSERT INTO orar(orar_id, predare_id, prototip_id, descriere)
168                  VALUES (to_char(j.pidp)||i.pid, i.pid, j.pidp, null);
169                  it := it + 1;
170                  k := k + 1;
171              END IF;
172          END LOOP;
173      END LOOP;
174      EXCEPTION
175      WHEN OTHERS THEN
176          DBMS_OUTPUT.PUT_LINE (' others: ' ||SQLCODE || '- ' || SQLERRM);
177  END;
178  END PACHET_ANACOMO;
179  /
180  EXECUTE pachet_anacomo.generare_orar(91, 260);

```

## Rularea pachetului

```

33      cnt := cnt + 1;
34  END LOOP;
35  DELETE FROM conflict;

```

Script Output x

Task completed in 0.076 seconds

Errors: Check compiler log

Package PACHET\_ANACOMO compiled

Package Body PACHET\_ANACOMO compiled

## 14 Pachet care include tipuri de date complexe și obiecte necesare pentru acțiuni integrate

### 14.1 Scopul pachetului

Domnul inginer Ștefan vrea să creeze o aplicație pentru elevi, care le oferă niște informații despre programul lor la școală, colegi, profesori, orar. Să se creeze un pachet care să conțină următoarele proprietăți:

1. Se poate introduce id-ul elevului.
2. Se poate afișa datele elevului.
3. Se pot afișa numele colegilor de clasă.
4. Se pot afișa numele profesorilor de la clasă.
5. Se poate afișa orarul clasei.

### 14.2 Codul PL/SQL

```
1 CREATE OR REPLACE PACKAGE pachet2_anacomo
2 AS
3     TYPE elevi IS VARRAY(2) OF NUMBER;
4     date_elevi ELEV := ELEV();
5     PROCEDURE set_my_id(my_id NUMBER);
6     PROCEDURE get_my_id;
7     PROCEDURE afiseaza_colegi;
8     PROCEDURE afiseaza_profesori;
9     PROCEDURE afiseaza_orar;
10 END pachet2_anacomo;
11 /
12 CREATE OR REPLACE PACKAGE BODY pachet2_anacomo AS
13 -- set my id
14     PROCEDURE set_my_id(my_id NUMBER)
15     AS
16         cid     NUMBER;
17         cnt     NUMBER;
18     BEGIN
19         SELECT COUNT (*)
20         INTO cnt
21         FROM elev
22         WHERE elev_id = my_id;
23         IF cnt = 1 THEN
24             pachet2_anacomo.date_elevi.extend();
25             pachet2_anacomo.date_elevi(1) := my_id;
26             SELECT elev.clasa_id
27             INTO cid
28             FROM elev
29             WHERE elev.elev_id = pachet2_anacomo.date_elevi(1);
30             pachet2_anacomo.date_elevi.extend();
31             pachet2_anacomo.date_elevi(2) := cid;
32             dbms_output.put_line('Id-ul dvs este '||pachet2_anacomo.date_elevi(1));
33         ELSE
34             dbms_output.put_line('va rugam sa introduceti un id valid!');
35         END IF;
36     END;
37 -- get my id
38     PROCEDURE get_my_id
39     AS
40     BEGIN
41         dbms_output.put_line('Id-ul dvs este '|| pachet2_anacomo.date_elevi(1));
42     END;
43 -- afiseaza-mi colegii de clasa
44     PROCEDURE afiseaza_colegi
45     IS
46     BEGIN
47         DBMS_OUTPUT.PUT_LINE('Colegii dumneavoastra sunt:');
```

```

48     FOR i IN (SELECT nume, prenume, initiala
49                FROM elev e
50                WHERE e.clasa_id = pachet2_anacomo.date_elevi(2)) LOOP
51         DBMS_OUTPUT.PUT_LINE(i.prenume || ' ' || i.initiala || ' ' || i.prenume);
52     END LOOP;
53     EXCEPTION
54         WHEN NO_DATA_FOUND THEN
55             DBMS_OUTPUT.PUT_LINE('No data found');
56         WHEN OTHERS THEN
57             DBMS_OUTPUT.PUT_LINE('Others');
58     END;
59 -- afiseaza-mi profesorii
60     PROCEDURE afiseaza_profesori
61     IS
62     BEGIN
63         DBMS_OUTPUT.PUT_LINE('Profesorii dumneavoastra sunt:');
64         FOR i IN (SELECT DISTINCT profesor.titlu pt, profesor.nume pn, profesor.prenume
65                    FROM clasa
66                    JOIN predare USING (clasa_id)
67                    JOIN profesor USING (profesor_id)
68                    WHERE clasa_id = pachet2_anacomo.date_elevi(2)) LOOP
69             DBMS_OUTPUT.PUT_LINE(i.pt || ' ' || i.pn || ' ' || i.ppn);
70         END LOOP;
71     EXCEPTION
72         WHEN NO_DATA_FOUND THEN
73             DBMS_OUTPUT.PUT_LINE('No data found');
74         WHEN OTHERS THEN
75             DBMS_OUTPUT.PUT_LINE('Others');
76     END;
77 -- afiseaza-mi orarul
78     PROCEDURE afiseaza_orar
79     IS
80     BEGIN
81         DBMS_OUTPUT.PUT_LINE('Orarul dumneavoastra este:');
82         FOR i IN (SELECT zi.nume zn, ora.ora_start ost, ora.ora_stop ostop, materie.
83                    denumire den
84                    FROM CLASA
85                    JOIN PREDARE USING (clasa_id)
86                    JOIN ORAR USING (predare_id)
87                    JOIN MATERIE USING (materie_id)
88                    JOIN PROTOTIP_ORAR on orar.prototip_id = prototip_orar.prototip_id
89                    JOIN ZI USING (zi_id)
90                    JOIN ORA USING (ora_id)
91                    WHERE clasa_id = pachet2_anacomo.date_elevi(2)
92                    ORDER BY orar.prototip_id) LOOP
93             DBMS_OUTPUT.PUT_LINE(i.zn || ' de la ' || i.ost || ' la ' || i.ostp || '
94             aveti ' || i.den);
95         END LOOP;
96     EXCEPTION
97         WHEN NO_DATA_FOUND THEN
98             DBMS_OUTPUT.PUT_LINE('No data found');
99         WHEN OTHERS THEN
100             DBMS_OUTPUT.PUT_LINE('Others');
101     END;
102 /
103 EXECUTE pachet2_anacomo.set_my_id(120);
104 EXECUTE pachet2_anacomo.get_my_id;
105 EXECUTE pachet2_anacomo.afiseaza_colegi;
106 EXECUTE pachet2_anacomo.afiseaza_profesori;
107 EXECUTE pachet2_anacomo.afiseaza_orar;
108 /

```

### 14.3 Rularea pachetului

```
68      INTO cid
69      FROM elev
70      WHERE elev.elev_id = pachet2_anacomo.my_actual_id;
71      DBMS_OUTPUT.PUT_LINE('Profesorii dumneavoastra sunt:');

```

Script Output x Query Result x

Task completed in 0.045 seconds

PL/SQL procedure successfully completed.

Package Body PACHET2\_ANACOMO compiled