

Proiect SGBD

- Atelier Arhitectura -

CUPRINS

1. Prezentarea pe scurt a bazei de date (utilitatea ei)	2
2. Diagrama entitate-relatie (ERD) – in FN3	3
3. Diagrama conceptuala.....	3
4. Implementarea diagramei conceptuale in Oracle.....	4
5. Popularea tabelor cu inregistrari	17
6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze două tipuri de colecție studiate. Apelați subprogramul.	24
7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.	26
8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.....	28
9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.	31
10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.	33
11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.	34
12. Definiți un trigger de tip LDD. Declanșați trigger-ul.....	37
13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.	38

1. Prezentarea pe scurt a bazei de date (utilitatea ei)

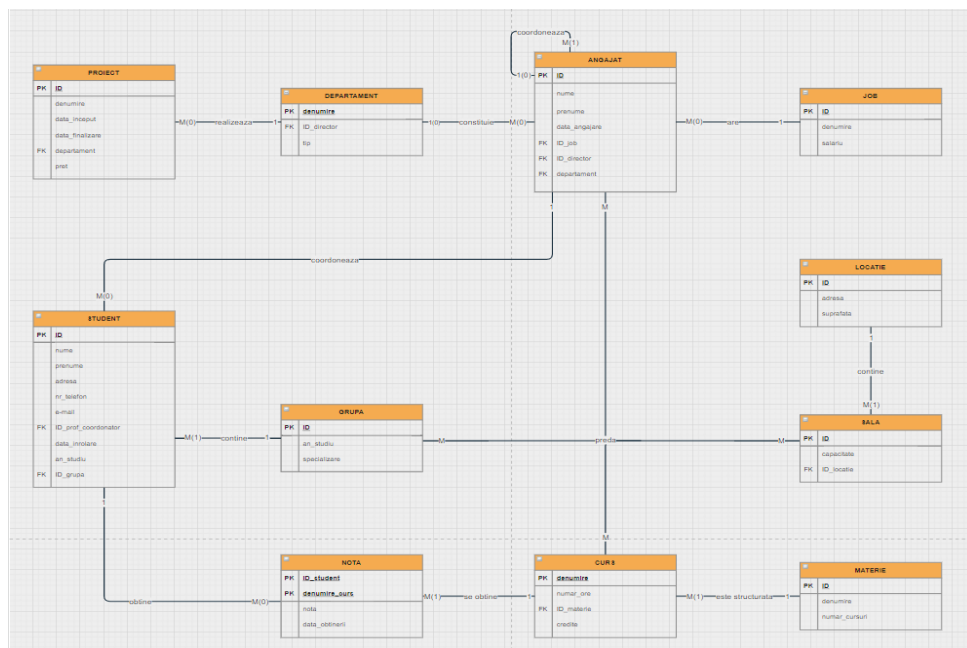
În prezentarea aceasta va voi introduce în cadrul “*Atelierului de Arhitectura*”, unde putem găsi specialiști în domeniul Arhitecturii, al Design-ului Interior și al Urbanismului, angajați în diverse Proiecte inovative, împreună cu Profesorii care se ocupă de formarea viitorilor Arhitecți, prin predarea secretelor meseriei și prin îndrumarea acestora către succes în această carieră esențială în societate, bazată pe creativitate, stăpânirea conceptelor teoretice și multă practică.

Scopul gestiunii acestei bazei de date este de a facilita operarea cu informații și date legate de toate aspectele ce implică un Atelier de Arhitectura: realizarea proiectelor de către angajați, activitățile de predare ale profesorilor și cele de învățare ale studenților etc.

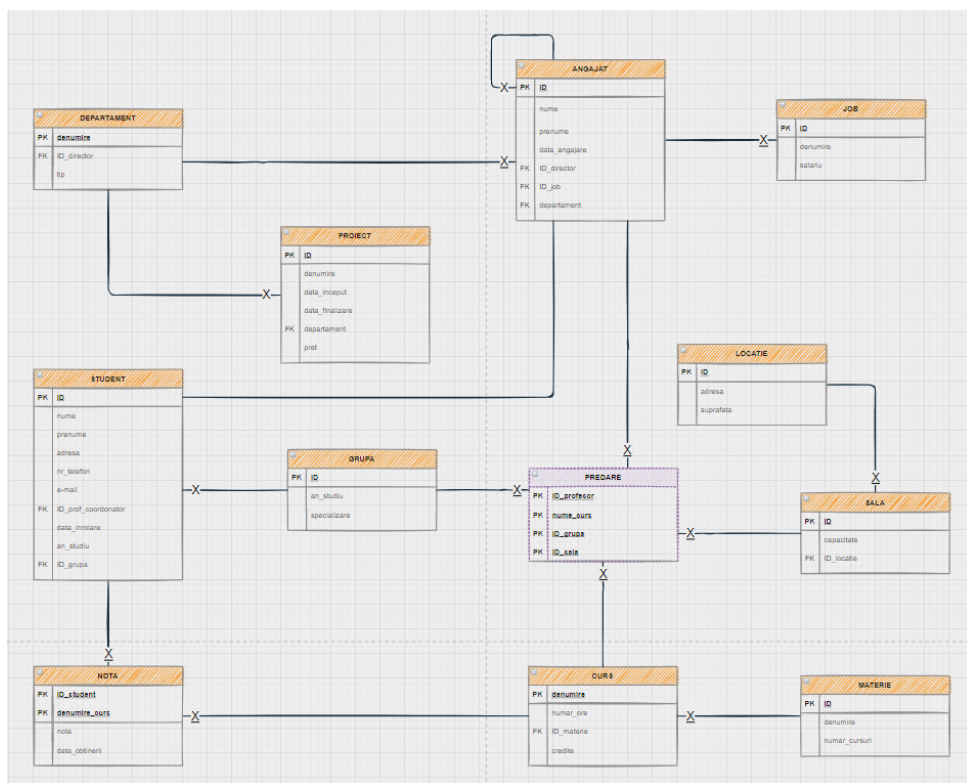
Prezentarea, pe scurt, a entităților și constrângerilor ce definesc baza de date:

- 12 tabele, dintre care 2 asociative (NOTA și PREDARE)
- fiecare angajat are un manager (sau null, în cazul în care acesta este managerul atelierului)
- fiecare angajat face parte dintr-un singur departament
- fiecare angajat primește salariul în funcție de job-ul în care activează
- un proiect poate fi realizat de un singur departament

2. Diagrama entitate-relatie (ERD) – in FN3

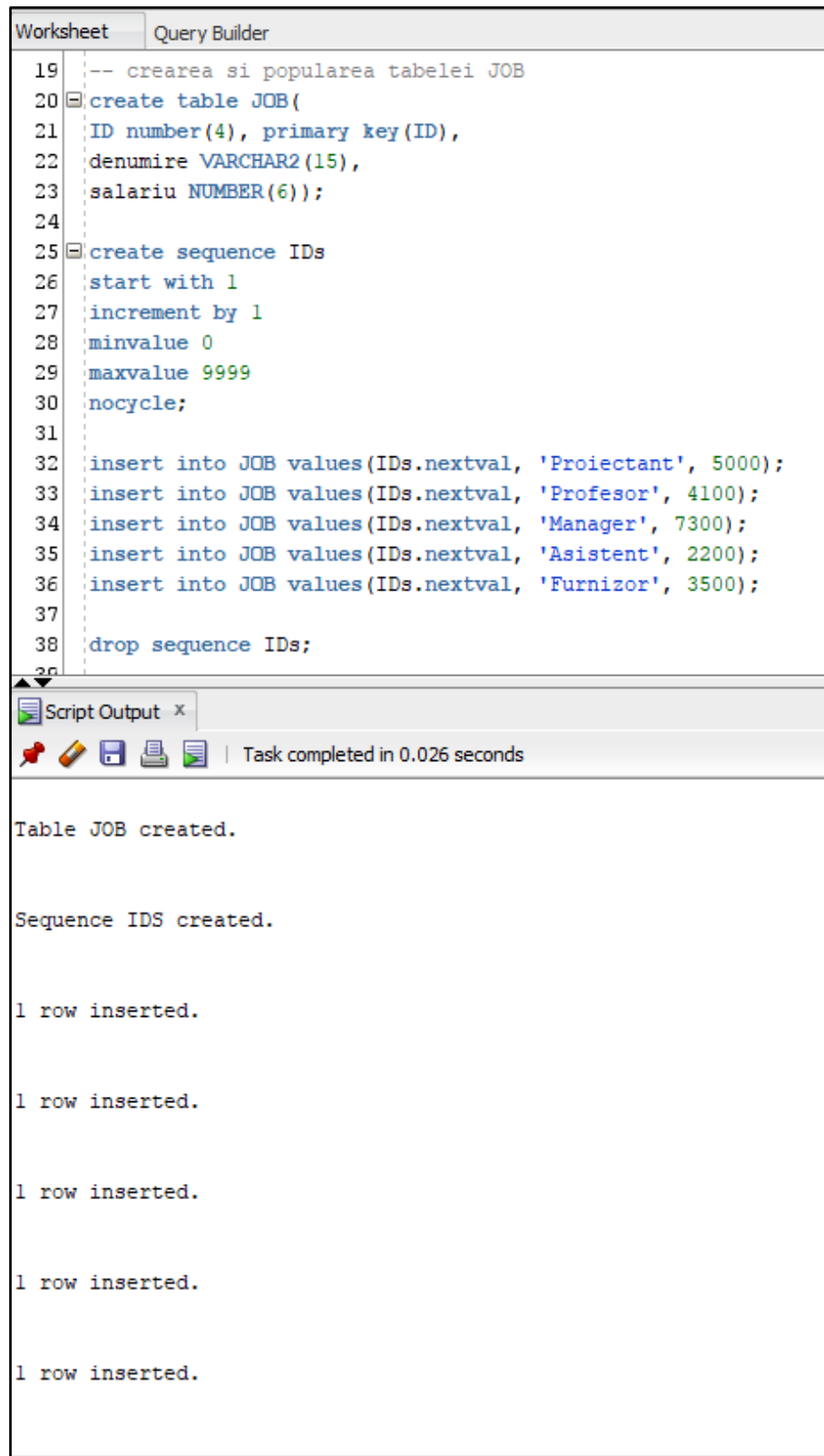


3. Diagrama conceptuala



4. Implementarea diagramei conceptuale in Oracle

1) JOB



The screenshot displays the Oracle SQL Developer interface. The top pane, titled 'Worksheet', contains a SQL script for creating and populating a table named 'JOB'. The script includes comments, table creation with a primary key, sequence creation, and five insert statements. The bottom pane, titled 'Script Output', shows the successful execution of the script, with messages for table creation, sequence creation, and five individual row insertions. A status bar at the bottom of the output pane indicates the task was completed in 0.026 seconds.

```
19  -- crearea si popularea tabelii JOB
20  create table JOB(
21    ID number(4), primary key(ID),
22    denumire VARCHAR2(15),
23    salariu NUMBER(6));
24
25  create sequence IDS
26    start with 1
27    increment by 1
28    minvalue 0
29    maxvalue 9999
30    nocycle;
31
32  insert into JOB values(IDs.nextval, 'Proiectant', 5000);
33  insert into JOB values(IDs.nextval, 'Profesor', 4100);
34  insert into JOB values(IDs.nextval, 'Manager', 7300);
35  insert into JOB values(IDs.nextval, 'Asistent', 2200);
36  insert into JOB values(IDs.nextval, 'Furnizor', 3500);
37
38  drop sequence IDS;
```

Script Output x

Task completed in 0.026 seconds

Table JOB created.

Sequence IDS created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

2) DEPARTAMENT

Worksheet

Query Builder

41

-- crearea si popularea tabelii DEPARTAMENT

42

create table DEPARTAMENT(

43

denumire varchar2(15), primary key(denumire),

44

ID_director number(4),

45

tip varchar2(10));

46

47

insert into DEPARTAMENT values('Confectii', 5, 'Productie');

48

insert into DEPARTAMENT values('Machete', 4, 'Productie');

49

insert into DEPARTAMENT values('Planse', 7, 'Cercetare');

50






insert into DEPARTAMENT values('Software', 1, 'Cercetare');

51

insert into DEPARTAMENT values('Asezari', 2, 'Cercetare');

52

Script Output x



Task completed in 0.027 seconds

1 row inserted.

1 row inserted.

1 row inserted.

Sequence IDS dropped.

Table DEPARTAMENT created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

3) PROIECT

Worksheet

Query Builder

53

-- crearea si popularea tabelii PROIECT

54

create table PROIECT(

55

ID number(6), primary key(ID),

56

denumire varchar2(30),

57

data_inceput date,

58

data_final date,

59

departament varchar2(15),

60

pret float(2),

61

foreign key(departament) references DEPARTAMENT(denumire));

62

63

create sequence IDS

64

start with 1

65

increment by 1

66

minvalue 0

67

maxvalue 999999

68

nocycle;

69

70

insert into PROIECT values(IDs.nextval, 'Restaurare Castel', '10-03-2019', '18-03-2019', 'Planse', 450.00);

71

insert into PROIECT values(IDs.nextval, 'Macheta Residence', '21-05-2021', null, 'Machete', 1800.00);

72

insert into PROIECT values(IDs.nextval, 'Colaj Ateneul Roman', '07-07-2020', '02-03-2021', 'Asezari', 300.00);

73

insert into PROIECT values(IDs.nextval, 'Casa lui Mos-Martin', '01-04-2021', null, 'Machete', 2500.00);

74

insert into PROIECT values(IDs.nextval, 'Eficientizare AutoCad', '13-04-2021', '14-04-2021', 'Software', 135.00);

75

76

drop sequence IDS;

77

Script Output x

Task completed in 0.047 seconds

1 row inserted.

1 row inserted.



1 row inserted.





1 row inserted.

1 row inserted.

Sequence IDS dropped.

4) ANGAJAT

Worksheet	Query Builder
79	-- crearea si popularea tabelii ANGAJAT
80	 create table ANGAJAT(
81	ID number(4), primary key(ID),
82	nume varchar2(15),
83	prenume varchar2(15),
84	data_angajare date,
85	ID_director number(4),
86	departament varchar2(15),
87	ID_job number(4),
88	foreign key(ID_director) references ANGAJAT(ID),
89	foreign key(departament) references DEPARTAMENT(denumire),
90	foreign key(ID_job) references JOB(ID));
91	
92	 create sequence IDs
93	start with 1
94	increment by 1
95	minvalue 0
96	maxvalue 9999
97	nocycle;
98	
99	insert into ANGAJAT values(IDs.nextval, 'Neagu', 'Alexandra-Ioana', '22-12-2015', null, 'Asezari', 3);
100	insert into ANGAJAT values(IDs.nextval, 'Anton', 'Mihai-Cosmin', '17-11-2014', 1, 'Software', 1);
101	insert into ANGAJAT values(IDs.nextval, 'Dima', 'Carol-Valentin', '13-01-2016', 1, null, 2);

Script Output x
    Task completed in 0.044 seconds
1 row inserted.
1 row inserted.
1 row inserted.
1 row inserted.
1 row inserted.
Sequence IDS dropped.

5) GRUPA

Worksheet

Query Builder

```
110 -- crearea si popularea tabelii GRUPA
111 create table GRUPA(
112   ID number(4), primary key(ID),
113   an_studiu number(1),
114   specializare varchar2(15));
115
116 create sequence IDs
117   start with 1
118   increment by 1
119   minvalue 0
120   maxvalue 9999
121   nocycle;
122
123 insert into GRUPA values(IDs.nextval, 3, 'Urbanism');
124 insert into GRUPA values(IDs.nextval, 1, 'Arhitectura');
125 insert into GRUPA values(IDs.nextval, 2, 'Arhitectura');
126 insert into GRUPA values(IDs.nextval, 2, 'Interior');
127 insert into GRUPA values(IDs.nextval, 1, 'Urbanism');
128 insert into GRUPA values(IDs.nextval, 3, 'Arhitectura');
129
130 drop sequence IDs;
```

Script Output x






     | Task completed in 0.06 seconds

Table GRUPA created.

Sequence IDS created.

1 row inserted.

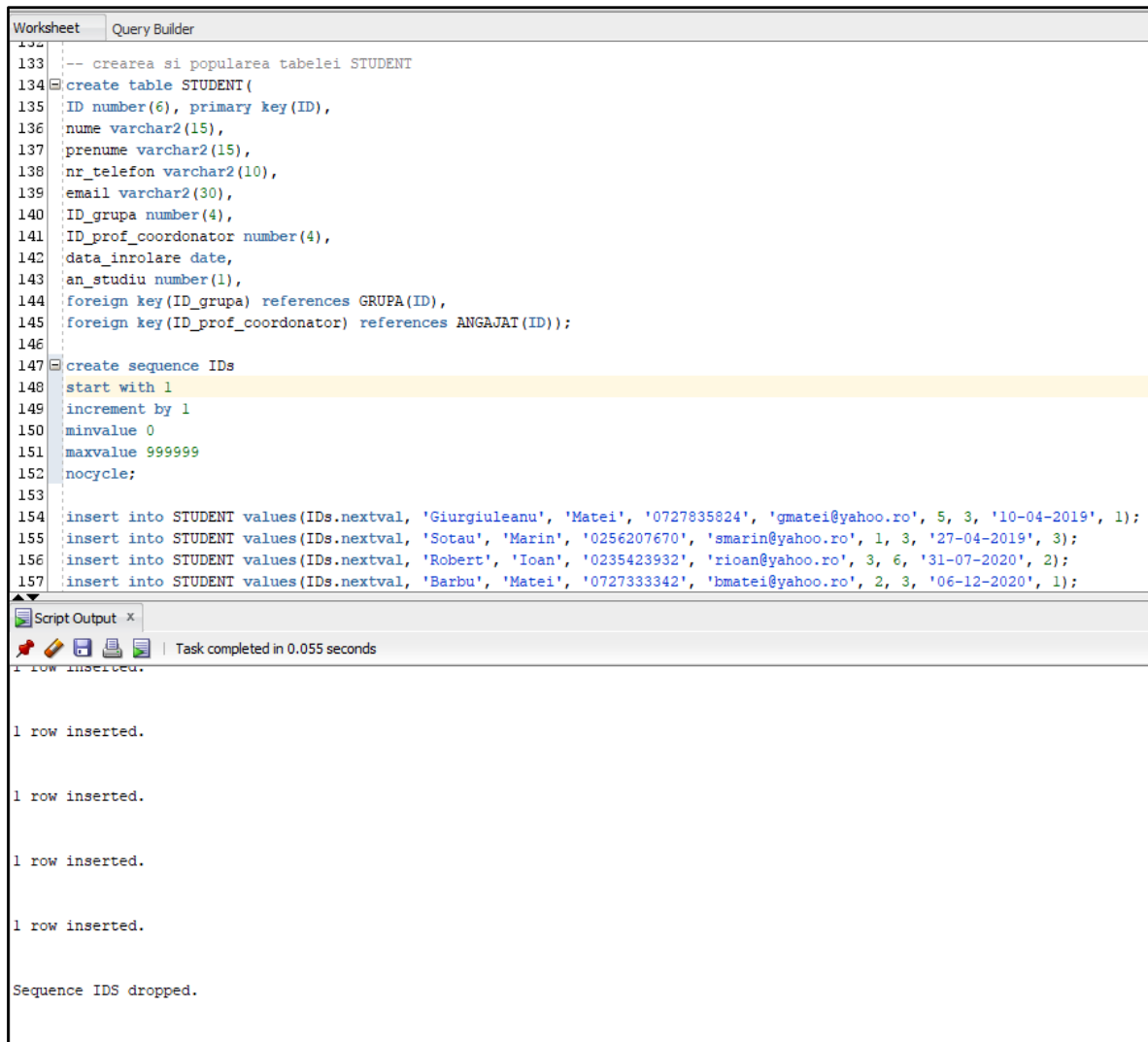
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

6) STUDENT



```
133 -- crearea si popularea tabelii STUDENT
134 create table STUDENT(
135     ID number(6), primary key(ID),
136     nume varchar2(15),
137     prenume varchar2(15),
138     nr_telefon varchar2(10),
139     email varchar2(30),
140     ID_grupa number(4),
141     ID_prof_coordonator number(4),
142     data_inrolare date,
143     an_studiu number(1),
144     foreign key(ID_grupa) references GRUPA(ID),
145     foreign key(ID_prof_coordonator) references ANGAJAT(ID));
146
147 create sequence IDS
148     start with 1
149     increment by 1
150     minvalue 0
151     maxvalue 999999
152     nocycle;
153
154 insert into STUDENT values(IDs.nextval, 'Giurgiuleanu', 'Matei', '0727835824', 'gmatei@yahoo.ro', 5, 3, '10-04-2019', 1);
155 insert into STUDENT values(IDs.nextval, 'Sotau', 'Marin', '0256207670', 'smarin@yahoo.ro', 1, 3, '27-04-2019', 3);
156 insert into STUDENT values(IDs.nextval, 'Robert', 'Ioan', '0235423932', 'rioan@yahoo.ro', 3, 6, '31-07-2020', 2);
157 insert into STUDENT values(IDs.nextval, 'Barbu', 'Matei', '0727333342', 'bmatei@yahoo.ro', 2, 3, '06-12-2020', 1);
```

Script Output x

Task completed in 0.055 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Sequence IDS dropped.

7) MATERIE

Worksheet

Query Builder

```
168 -- crearea si popularea tabelii MATERIE
169 create table MATERIE(
170     ID number(4), primary key(ID),
171     denumire varchar2(30),
172     numar_cursuri number(2));
173
174 create sequence IDs
175     start with 1
176     increment by 1
177     minvalue 0
178     maxvalue 9999
179     nocycle;
180
181 insert into MATERIE values(IDs.nextval, 'Studiul Formei', 8);
182 insert into MATERIE values(IDs.nextval, 'Istoria Asezarilor in Europa', 8);
183 insert into MATERIE values(IDs.nextval, 'Geometrie Descriptiva', 16);
184 insert into MATERIE values(IDs.nextval, 'Introducere in Arh. Cont.', 12);
185 insert into MATERIE values(IDs.nextval, 'Perspectiva', 16);
186 insert into MATERIE values(IDs.nextval, 'Limba Engleza', 4);
187
188 drop sequence IDs;
189
```

Script Output x






     | Task completed in 0.052 seconds

Table MATERIE created.

Sequence IDS created.

1 row inserted.

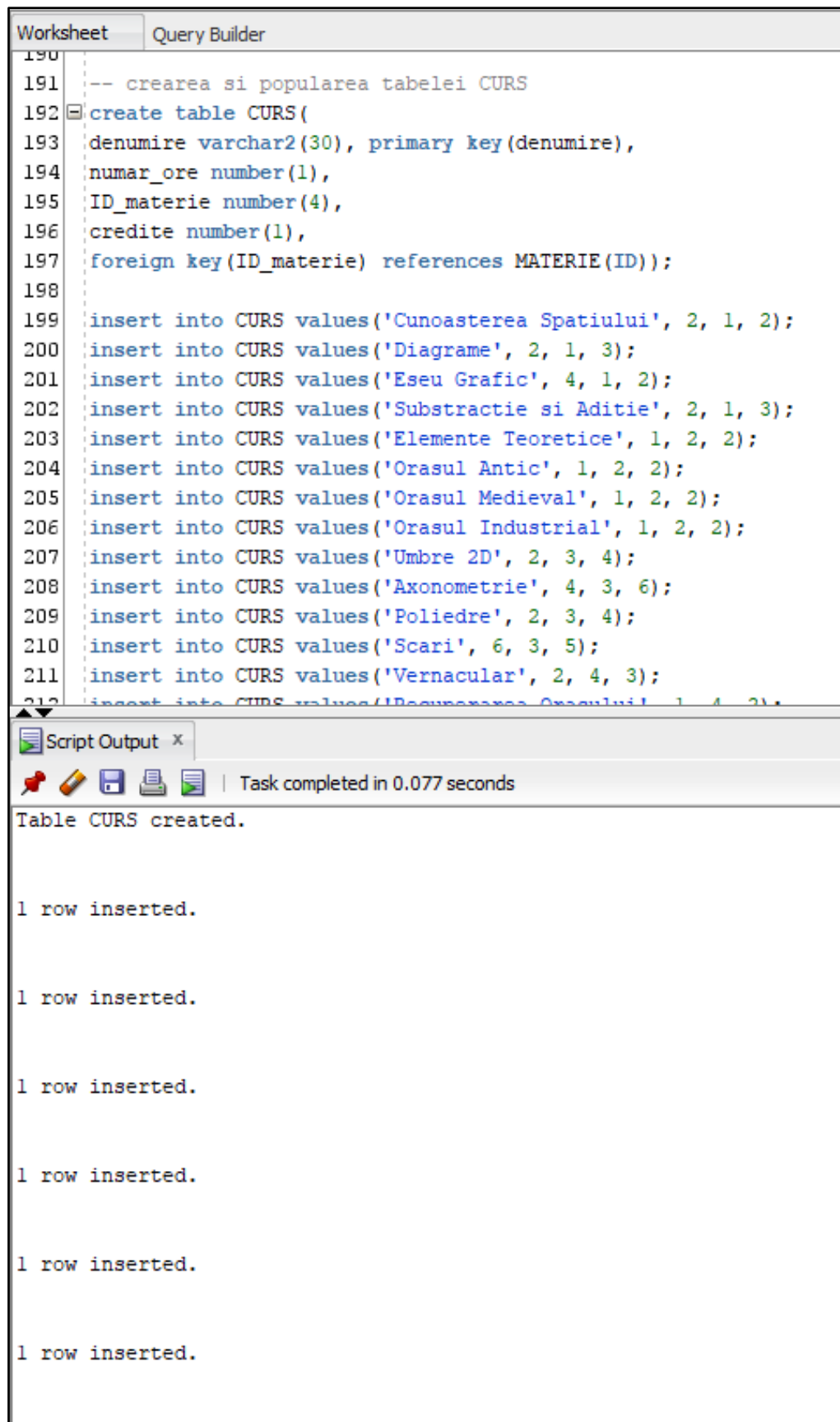
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

8) CURS



The screenshot shows a SQL query editor window with a 'Worksheet' tab and a 'Query Builder' tab. The 'Worksheet' tab is active, displaying a SQL script. The script starts with a comment '-- crearea si popularea tabelii CURS' followed by a 'create table CURS' statement. The table has four columns: 'denumire' (varchar2(30)), 'numar_ore' (number(1)), 'ID_materie' (number(4)), and 'credite' (number(1)). There is a primary key on 'denumire' and a foreign key on 'ID_materie' that references the 'ID' column of the 'MATERIE' table. After the table creation, there are 12 'insert into CURS' statements, each adding a row with specific values for the four columns. The script ends with a semi-colon on line 212.

Below the query editor is a 'Script Output' window. It shows the execution results of the SQL script. The first line is 'Table CURS created.' followed by 12 lines, each stating '1 row inserted.'.

```
190
191 -- crearea si popularea tabelii CURS
192 create table CURS(
193     denumire varchar2(30), primary key(denumire),
194     numar_ore number(1),
195     ID_materie number(4),
196     credite number(1),
197     foreign key(ID_materie) references MATERIE(ID));
198
199 insert into CURS values('Cunoasterea Spatiului', 2, 1, 2);
200 insert into CURS values('Diagrame', 2, 1, 3);
201 insert into CURS values('Eseu Grafic', 4, 1, 2);
202 insert into CURS values('Subtractie si Aditie', 2, 1, 3);
203 insert into CURS values('Elemente Teoretice', 1, 2, 2);
204 insert into CURS values('Orasul Antic', 1, 2, 2);
205 insert into CURS values('Orasul Medieval', 1, 2, 2);
206 insert into CURS values('Orasul Industrial', 1, 2, 2);
207 insert into CURS values('Umbre 2D', 2, 3, 4);
208 insert into CURS values('Axonometrie', 4, 3, 6);
209 insert into CURS values('Poliedre', 2, 3, 4);
210 insert into CURS values('Scari', 6, 3, 5);
211 insert into CURS values('Vernacular', 2, 4, 3);
212 insert into CURS values('Requereror Orasului', 1, 4, 2);
```

Script Output x

Task completed in 0.077 seconds

Table CURS created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

9) LOCATIE

Worksheet

Query Builder

224

-- crearea si popularea tabelii LOCATIE

225

create table LOCATIE(

226

ID number(2), primary key(ID),

227

adresa varchar2(60),

228

suprafata number(6));

229

230

create sequence IDS

231

start with 1

232

increment by 1

233

minvalue 0

234

maxvalue 99

235

nocycle;

236

237

insert into LOCATIE values(IDs.nextval, 'str. Belindo, nr. 6', 12500);

238

insert into LOCATIE values(IDs.nextval, 'str. Aurie, nr. 7A', 1500);

239

insert into LOCATIE values(IDs.nextval, 'str. Teilor, nr. 4', 20300);

240

insert into LOCATIE values(IDs.nextval, 'str. Suspendata, nr. 2B', 780);

241

insert into LOCATIE values(IDs.nextval, 'str. Nikolas Vinz, nr. 11', 321400);

242

243

drop sequence IDS;

244

Script Output x

Task completed in 0.044 seconds

Table LOCATIE created.

Sequence IDS created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Sequence IDS dropped.

10) SALA

Worksheet

Query Builder

246

-- crearea si popularea tabelii SALA

247

create table SALA(

248

ID number(4), primary key(ID),

249

capacitate number(2),

250

ID_locatie number(2),

251

foreign key(ID_locatie) references LOCATIE(ID));

252

253

create sequence IDS

254

start with 1

255

increment by 1

256

minvalue 0

257

maxvalue 9999

258

nocycle;

259

260

insert into SALA values(IDs.nextval, 18, 1);

261

insert into SALA values(IDs.nextval, 10, 2);

262

insert into SALA values(IDs.nextval, 28, 4);

263

insert into SALA values(IDs.nextval, 70, 3);

264

insert into SALA values(IDs.nextval, 20, 5);

265

insert into SALA values(IDs.nextval, 80, 1);

266

insert into SALA values(IDs.nextval, 60, 4);

Script Output x






     | Task completed in 0.065 seconds

Table SALA created.

Sequence IDS created.

1 row inserted.

1 row inserted.






1 row inserted.

1 row inserted.

1 row inserted.





11) NOTA

Worksheet	Query Builder
271	-- crearea si popularea tabelii NOTA
272	create table NOTA(
273	ID_student number(6),
274	denumire_curs varchar2(30),
275	primary key(ID_student, denumire_curs),
276	nota float(8),
277	data_obtinerii date,
278	foreign key(ID_student) references STUDENT(ID),
279	foreign key(denumire_curs) references CURS(denumire));
280	
281	insert into NOTA values(1, 'Scari', 7.50, '12-06-2020');
282	insert into NOTA values(1, 'Axonometrie', 8.50, '13-07-2020');
283	insert into NOTA values(2, 'Poliedre', 9.50, '11-05-2019');
284	insert into NOTA values(2, 'Axonometrie', 9, '13-07-2019');
285	insert into NOTA values(3, 'Poliedre', 7, '13-07-2020');
286	insert into NOTA values(4, 'Poliedre', 6, '12-06-2020');
287	insert into NOTA values(4, 'Umbre 2D', 8.50, '11-05-2019');
288	insert into NOTA values(4, 'Orasul Traditional', 4.30, '11-05-2019');
289	insert into NOTA values(4, 'Arhitectul', 8.10, '13-07-2020');
290	insert into NOTA values(5, 'Poliedre', 7.20, '11-05-2019');
291	insert into NOTA values(5, 'Umbre 2D', 8.90, '12-06-2020');

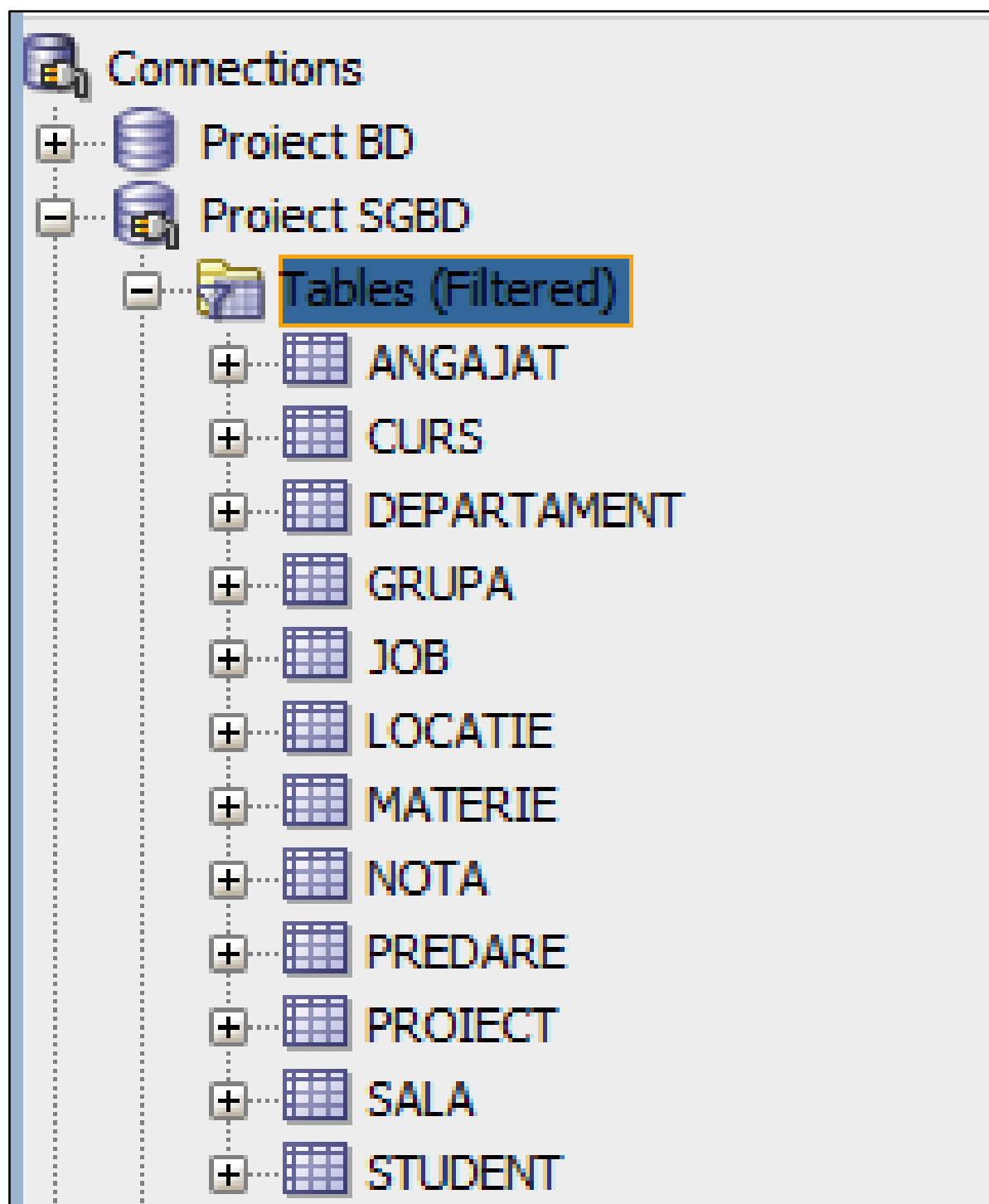
Script Output x
     Task completed in 0.059 seconds
Table NOTA created.
1 row inserted.
1 row inserted.
1 row inserted.
1 row inserted.
1 row inserted.
1 row inserted.

12) PREDARE

Worksheet	Query Builder
306	-- crearea si popularea tabelii PREDARE
307	<code>create table PREDARE(</code>
308	<code> ID_profesor number(4),</code>
309	<code> nume_curs varchar2(30),</code>
310	<code> ID_grupa number(4),</code>
311	<code> ID_sala number(4),</code>
312	<code> primary key(ID_profesor, nume_curs, ID_grupa, ID_sala),</code>
313	<code> foreign key(ID_profesor) references ANGAJAT(ID),</code>
314	<code> foreign key(nume_curs) references CURS(denumire),</code>
315	<code> foreign key(ID_grupa) references GRUPA(ID),</code>
316	<code> foreign key(ID_sala) references SALA(ID));</code>
317	
318	<code>insert into PREDARE values(3, 'Poliedre', 1, 1);</code>
319	<code>insert into PREDARE values(3, 'Axonometrie', 1, 1);</code>
320	<code>insert into PREDARE values(3, 'Poliedre', 2, 2);</code>
321	<code>insert into PREDARE values(3, 'Scari', 5, 3);</code>
322	<code>insert into PREDARE values(3, 'Umbre 2D', 2, 3);</code>
323	<code>insert into PREDARE values(3, 'Poliedre', 3, 4);</code>
324	<code>insert into PREDARE values(3, 'Axonometrie', 5, 5);</code>
325	<code>insert into PREDARE values(6, 'Arhitectul', 4, 1);</code>
326	<code>insert into PREDARE values(6, 'Arhitectul', 6, 6);</code>

Script Output x
    Task completed in 0.056 seconds
Table PREDARE created.
1 row inserted.
1 row inserted.
1 row inserted.
1 row inserted.
1 row inserted.
1 row inserted.

13) VEDERE de ANSAMBLU



5. Popularea tabelelor cu inregistrari in Oracle:

```
1. drop table predare;
2. drop table nota;
3. drop table sala;
4. drop table locatie;
5. drop table curs;
6. drop table materie;
7. drop table student;
8. drop table grupa;
9. drop table angajat;
10. drop table proiect;
11. drop table departament;
12. drop table job;
13.
14. ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YYYY';
15.
16.
17. -- crearea si popularea tabelii JOB
18. create table JOB(
19. ID number(4), primary key(ID),
20. denumire varchar2(15),
21. salariu number(6));
22.
23. create sequence IDs
24. start with 1
25. increment by 1
26. minvalue 0
27. maxvalue 9999
28. nocycle;
29.
30. insert into JOB values(IDs.nextval, 'Proiectant', 5000);
31. insert into JOB values(IDs.nextval, 'Profesor', 4100);
32. insert into JOB values(IDs.nextval, 'Manager', 7300);
33. insert into JOB values(IDs.nextval, 'Asistent', 2200);
34. insert into JOB values(IDs.nextval, 'Furnizor', 3500);
35.
36. drop sequence IDs;
37. commit;
38.
39.
40. -- crearea si popularea tabelii DEPARTAMENT
41. create table DEPARTAMENT(
42. denumire varchar2(15), primary key(denumire),
43. ID_director number(4),
44. tip varchar2(10));
45.
46. insert into DEPARTAMENT values('Confectii', 5, 'Productie');
47. insert into DEPARTAMENT values('Machete', 4, 'Productie');
48. insert into DEPARTAMENT values('Planse', 7, 'Cercetare');
49. insert into DEPARTAMENT values('Software', 1, 'Cercetare');
50. insert into DEPARTAMENT values('Asezari', 2, 'Cercetare');
51.
52. -- crearea si popularea tabelii PROIECT
53. create table PROIECT(
54. ID number(6), primary key(ID),
```

```

55.denumire varchar2(30),
56.data_inceput date,
57.data_final date,
58.departament varchar2(15),
59.pret float(2),
60.foreign key(departament) references DEPARTAMENT(denumire));
61.
62.create sequence IDs
63.start with 1
64.increment by 1
65.minvalue 0
66.maxvalue 999999
67.nocycle;
68.
69.insert into PROIECT values(IDs.nextval, 'Restaurare Castel', '10-03-2019',
    '18-03-2019', 'Planse', 450.00);
70.insert into PROIECT values(IDs.nextval, 'Macheta Residence', '21-05-2021',
    null, 'Machete', 1800.00);
71.insert into PROIECT values(IDs.nextval, 'Colaj Ateneul Roman', '07-07-2020',
    '02-03-2021', 'Asezari', 300.00);
72.insert into PROIECT values(IDs.nextval, 'Casa lui Mos-Martin', '01-04-2021',
    null, 'Machete', 2500.00);
73.insert into PROIECT values(IDs.nextval, 'Eficientizare AutoCad', '13-04-2021',
    '14-04-2021', 'Software', 135.00);
74.
75.drop sequence IDs;
76.commit;
77.
78.
79.-- crearea si popularea tabelii ANGAJAT
80.create table ANGAJAT(
81.ID number(4), primary key(ID),
82.nume varchar2(15),
83.prenume varchar2(15),
84.data_angajare date,
85.ID_director number(4),
86.departament varchar2(15),
87.ID_job number(4),
88.foreign key(ID_director) references ANGAJAT(ID),
89.foreign key(departament) references DEPARTAMENT(denumire),
90.foreign key(ID_job) references JOB(ID));
91.
92.create sequence IDs
93.start with 1
94.increment by 1
95.minvalue 0
96.maxvalue 9999
97.nocycle;
98.
99.insert into ANGAJAT values(IDs.nextval, 'Neagu', 'Alexandra-Ioana', '22-12-
    2015', null, 'Asezari', 3);
100.insert into ANGAJAT values(IDs.nextval, 'Anton', 'Mihai-Cosmin', '17-11-
    2014', 1, 'Software', 1);
101.insert into ANGAJAT values(IDs.nextval, 'Dima', 'Carol-Valentin', '13-01-
    2016', 1, null, 2);
102.insert into ANGAJAT values(IDs.nextval, 'Florea', 'Irina', '26-12-2019', 1,
    'Confectii', 4);
103.insert into ANGAJAT values(IDs.nextval, 'Benescu', 'Ioan', '02-02-2020', 1,
    'Planse', 4);

```

```

104. insert into ANGAJAT values(IDs.nextval, 'Gruia', 'Gabriel', '27-06-2019', 1,
    null, 2);
105. insert into ANGAJAT values(IDs.nextval, 'Dragulescu', 'Raluca', '02-08-
    2016', 4, 'Machete', 5);
106.
107. drop sequence IDs;
108. commit;
109.
110.
111. -- crearea si popularea tabelii GRUPA
112. create table GRUPA(
113. ID number(4), primary key(ID),
114. an_studiu number(1),
115. specializare varchar2(15));
116.
117. create sequence IDs
118. start with 1
119. increment by 1
120. minvalue 0
121. maxvalue 9999
122. nocycle;
123.
124. insert into GRUPA values(IDs.nextval, 3, 'Urbanism');
125. insert into GRUPA values(IDs.nextval, 1, 'Arhitectura');
126. insert into GRUPA values(IDs.nextval, 2, 'Arhitectura');
127. insert into GRUPA values(IDs.nextval, 2, 'Interior');
128. insert into GRUPA values(IDs.nextval, 1, 'Urbanism');
129. insert into GRUPA values(IDs.nextval, 3, 'Arhitectura');
130. drop sequence IDs;
131. commit;
132.
133. -- crearea si popularea tabelii STUDENT
134. create table STUDENT(
135. ID number(6), primary key(ID),
136. nume varchar2(15),
137. prenume varchar2(15),
138. nr_telefon varchar2(10),
139. email varchar2(30),
140. ID_grupa number(4),
141. ID_prof_coordonator number(4),
142. data_inrolare date,
143. an_studiu number(1),
144. foreign key(ID_grupa) references GRUPA(ID),
145. foreign key(ID_prof_coordonator) references ANGAJAT(ID));
146.
147. create sequence IDs
148. start with 1
149. increment by 1
150. minvalue 0
151. maxvalue 999999
152. nocycle;
153.
154. insert into STUDENT values(IDs.nextval, 'Giurgiuleanu', 'Matei',
    '0727835824', 'gmatei@yahoo.ro', 5, 3, '10-04-2019', 1);
155. insert into STUDENT values(IDs.nextval, 'Sotau', 'Marin', '0256207670',
    'smarin@yahoo.ro', 1, 3, '27-04-2019', 3);
156. insert into STUDENT values(IDs.nextval, 'Robert', 'Ioan', '0235423932',
    'rioan@yahoo.ro', 3, 6, '31-07-2020', 2);

```

```

157. insert into STUDENT values(IDs.nextval, 'Barbu', 'Matei', '0727333342',
    'bmatei@yahoo.ro', 2, 3, '06-12-2020', 1);
158. insert into STUDENT values(IDs.nextval, 'Stefanescu', 'Bob', '0269559228',
    'sbob@yahoo.ro', 2, 6, '28-09-2021', 1);
159. insert into STUDENT values(IDs.nextval, 'Paun', 'Cristinel', '0212520355',
    'pcristinel@yahoo.ro', 6, 6, '06-11-2021', 3);
160. insert into STUDENT values(IDs.nextval, 'Bitulescu', 'Carol', '0214082820',
    'bcarol@yahoo.ro', 1, 3, '20-08-2021', 3);
161. insert into STUDENT values(IDs.nextval, 'Popos', 'Florentin', '0740142399',
    'pflorentin@yahoo.ro', 4, 3, '13-02-2021', 1);
162. insert into STUDENT values(IDs.nextval, 'Dobre', 'Marcel', '0721328241',
    'dmarcel@yahoo.ro', 4, 3, '24-03-2020', 2);
163. insert into STUDENT values(IDs.nextval, 'Rosevilici', 'Teodor',
    '0744555788', 'rteodor@yahoo.ro', 2, 6, '19-07-2019', 1);
164.
165. drop sequence IDs;
166. commit;
167.
168.
169. -- crearea si popularea tabelii MATERIE
170. create table MATERIE(
171. ID number(4), primary key(ID),
172. denumire varchar2(30),
173. numar_cursuri number(2));
174.
175. create sequence IDs
176. start with 1
177. increment by 1
178. minvalue 0
179. maxvalue 9999
180. nocycle;
181.
182. insert into MATERIE values(IDs.nextval, 'Studiul Formei', 8);
183. insert into MATERIE values(IDs.nextval, 'Istoria Asezarilor in Europa', 8);
184. insert into MATERIE values(IDs.nextval, 'Geometrie Descriptiva', 16);
185. insert into MATERIE values(IDs.nextval, 'Introducere in Arh. Cont.', 12);
186. insert into MATERIE values(IDs.nextval, 'Perspectiva', 16);
187. insert into MATERIE values(IDs.nextval, 'Limba Engleza', 4);
188.
189. drop sequence IDs;
190. commit;
191.
192.
193. -- crearea si popularea tabelii CURS
194. create table CURS(
195. denumire varchar2(30), primary key(denumire),
196. numar_ore number(1),
197. ID_materie number(4),
198. credite number(1),
199. foreign key(ID_materie) references MATERIE(ID));
200.
201. insert into CURS values('Cunoasterea Spatiului', 2, 1, 2);
202. insert into CURS values('Diagrame', 2, 1, 3);
203. insert into CURS values('Eseu Grafic', 4, 1, 2);
204. insert into CURS values('Subtractie si Aditie', 2, 1, 3);
205. insert into CURS values('Elemente Teoretice', 1, 2, 2);
206. insert into CURS values('Orasul Antic', 1, 2, 2);
207. insert into CURS values('Orasul Medieval', 1, 2, 2);
208. insert into CURS values('Orasul Industrial', 1, 2, 2);

```

```

209. insert into CURS values('Umbre 2D', 2, 3, 4);
210. insert into CURS values('Axonometrie', 4, 3, 6);
211. insert into CURS values('Poliedre', 2, 3, 4);
212. insert into CURS values('Scari', 6, 3, 5);
213. insert into CURS values('Vernacular', 2, 4, 3);
214. insert into CURS values('Recuperarea Orasului', 1, 4, 2);
215. insert into CURS values('Arhitectul', 1, 4, 2);
216. insert into CURS values('Orasul Traditional', 1, 4, 2);
217. insert into CURS values('Sisteme de Proiectie', 4, 5, 5);
218. insert into CURS values('Perspectiva la Fuga', 2, 5, 4);
219. insert into CURS values('Trasarea Umbrelor', 4, 5, 6);
220. insert into CURS values('Perspectiva la Calculator', 2, 5, 4);
221. insert into CURS values('Past Tenses', 1, 6, 2);
222. insert into CURS values('Architectural Vocabulary', 2, 6, 3);
223. insert into CURS values('Useful Phrases', 2, 6, 2);
224.
225. commit;
226.
227.
228. -- crearea si popularea tabelii LOCATIE
229. create table LOCATIE(
230. ID number(2), primary key(ID),
231. adresa varchar2(60),
232. suprafata number(6));
233.
234. create sequence IDs
235. start with 1
236. increment by 1
237. minvalue 0
238. maxvalue 99
239. nocycle;
240.
241. insert into LOCATIE values(IDs.nextval, 'str. Belindo, nr. 6', 12500);
242. insert into LOCATIE values(IDs.nextval, 'str. Aurie, nr. 7A', 1500);
243. insert into LOCATIE values(IDs.nextval, 'str. Teilor, nr. 4', 20300);
244. insert into LOCATIE values(IDs.nextval, 'str. Suspendata, nr. 2B', 780);
245. insert into LOCATIE values(IDs.nextval, 'str. Nikolas Vinz, nr. 11',
321400);
246.
247. drop sequence IDs;
248. commit;
249.
250.
251. -- crearea si popularea tabelii SALA
252. create table SALA(
253. ID number(4), primary key(ID),
254. capacitate number(2),
255. ID_locatie number(2),
256. foreign key(ID_locatie) references LOCATIE(ID));
257.
258. create sequence IDs
259. start with 1
260. increment by 1
261. minvalue 0
262. maxvalue 9999
263. nocycle;
264.
265. insert into SALA values(IDs.nextval, 18, 1);
266. insert into SALA values(IDs.nextval, 10, 2);

```

```

267. insert into SALA values(IDs.nextval, 28, 4);
268. insert into SALA values(IDs.nextval, 70, 3);
269. insert into SALA values(IDs.nextval, 20, 5);
270. insert into SALA values(IDs.nextval, 80, 1);
271. insert into SALA values(IDs.nextval, 60, 4);
272.
273. drop sequence IDs;
274. commit;
275.
276.
277. -- crearea si popularea tablei NOTA
278. create table NOTA(
279. ID_student number(6),
280. denumire_curs varchar2(30),
281. primary key(ID_student, denumire_curs),
282. nota float(8),
283. data_obtinerii date,
284. foreign key(ID_student) references STUDENT(ID),
285. foreign key(denumire_curs) references CURS(denumire));
286.
287. insert into NOTA values(1, 'Scari', 7.50, '12-06-2020');
288. insert into NOTA values(1, 'Axonometrie', 8.50, '13-07-2020');
289. insert into NOTA values(2, 'Poliedre', 9.50, '11-05-2019');
290. insert into NOTA values(2, 'Axonometrie', 9, '13-07-2019');
291. insert into NOTA values(3, 'Poliedre', 7, '13-07-2020');
292. insert into NOTA values(4, 'Poliedre', 6, '12-06-2020');
293. insert into NOTA values(4, 'Umbre 2D', 8.50, '11-05-2019');
294. insert into NOTA values(4, 'Orasul Traditional', 4.30, '11-05-2019');
295. insert into NOTA values(4, 'Arhitectul', 8.10, '13-07-2020');
296. insert into NOTA values(5, 'Poliedre', 7.20, '11-05-2019');
297. insert into NOTA values(5, 'Umbre 2D', 8.90, '12-06-2020');
298. insert into NOTA values(5, 'Orasul Traditional', 8.50, '11-05-2019');
299. insert into NOTA values(5, 'Arhitectul', 10, '13-07-2020');
300. insert into NOTA values(6, 'Arhitectul', 9, '11-05-2019');
301. insert into NOTA values(6, 'Vernacular', 6.50, '13-07-2020');
302. insert into NOTA values(7, 'Poliedre', 8.50, '11-05-2019');
303. insert into NOTA values(7, 'Axonometrie', 7.10, '12-06-2020');
304. insert into NOTA values(8, 'Arhitectul', 3.20, '12-06-2020');
305. insert into NOTA values(9, 'Arhitectul', 9.10, '11-05-2019');
306. insert into NOTA values(10, 'Poliedre', 5, '13-07-2020');
307. insert into NOTA values(10, 'Umbre 2D', 5, '11-05-2019');
308. insert into NOTA values(10, 'Orasul Traditional', 4, '12-06-2020');
309. insert into NOTA values(10, 'Arhitectul', 6, '13-07-2020');
310.
311. commit;
312.
313.
314. -- crearea si popularea tablei PREDARE
315. create table PREDARE(
316. ID_profesor number(4),
317. nume_curs varchar2(30),
318. ID_grupa number(4),
319. ID_sala number(4),
320. primary key(ID_profesor, nume_curs, ID_grupa, ID_sala),
321. foreign key(ID_profesor) references ANGAJAT(ID),
322. foreign key(nume_curs) references CURS(denumire),
323. foreign key(ID_grupa) references GRUPA(ID),
324. foreign key(ID_sala) references SALA(ID));
325.

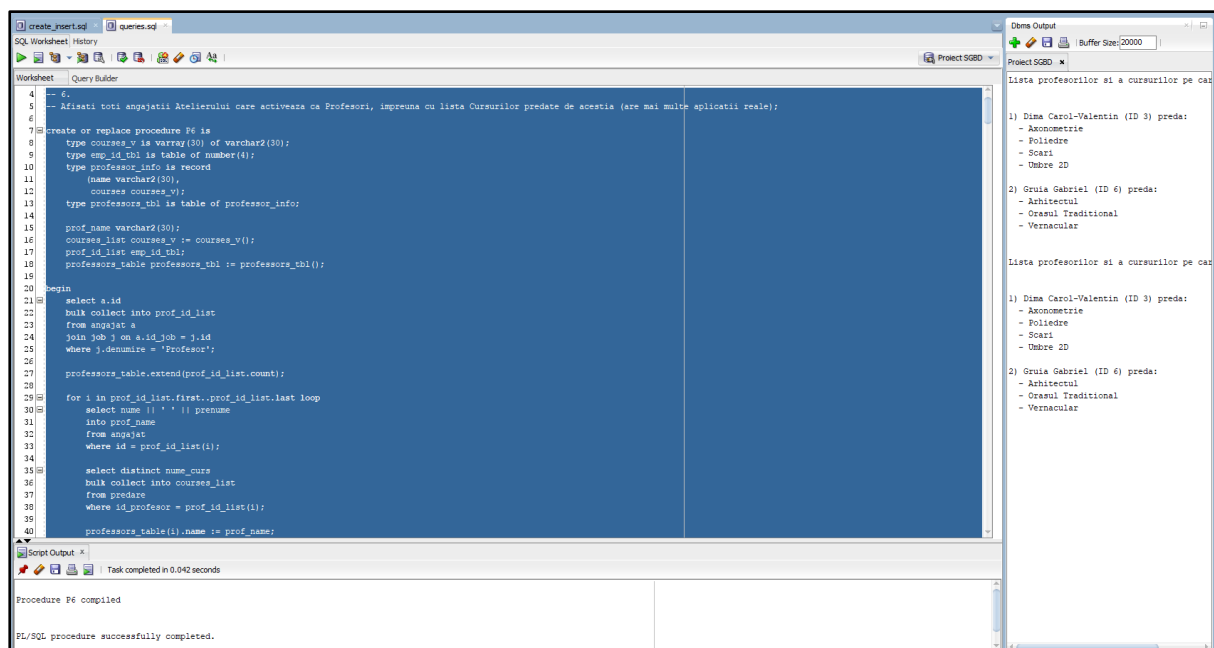
```

```
326. insert into PREDARE values(3, 'Poliedre', 1, 1);
327. insert into PREDARE values(3, 'Axonometrie', 1, 1);
328. insert into PREDARE values(3, 'Poliedre', 2, 2);
329. insert into PREDARE values(3, 'Scari', 5, 3);
330. insert into PREDARE values(3, 'Umbre 2D', 2, 3);
331. insert into PREDARE values(3, 'Poliedre', 3, 4);
332. insert into PREDARE values(3, 'Axonometrie', 5, 5);
333. insert into PREDARE values(6, 'Arhitectul', 4, 1);
334. insert into PREDARE values(6, 'Arhitectul', 6, 6);
335. insert into PREDARE values(6, 'Vernacular', 6, 3);
336. insert into PREDARE values(6, 'Orasul Traditional', 2, 6);
337. insert into PREDARE values(6, 'Arhitectul', 2, 2);
338.
339. commit;
```

6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze două tipuri de colecție studiate. Apelați subprogramul.

“Afișați totii angajații Atelierului care activează ca Profesori, împreună cu lista Cursurilor predate de acestia (are mai multe aplicații reale)”.

Retinem într-un nested table, ID-urile angajaților care activează ca profesori în cadrul atelierului apoi, pentru fiecare dintre acestia, obținem și afișăm lista cursurilor (din tabela “PREDARE”) pe care acestia le-au susținut.



The screenshot displays the Oracle SQL Developer environment. The main window shows a PL/SQL procedure named 'P6' in the 'Script' editor. The procedure is designed to process employee data and course information using nested tables. It includes comments in Romanian and SQL code for creating variables, tables, and performing bulk operations. The 'Diagnostics' pane at the bottom indicates that the procedure compiled successfully and executed without errors.

```
4 6.
5 -- Afișati totii angajații Atelierului care activează ca Profesori, împreună cu lista Cursurilor predate de acestia (are mai multe aplicații reale);
6
7 create or replace procedure P6 is
8   type course_v is varray(50) of varchar2(30);
9   type emp_id_tbl is table of number(4);
10  type professor_info is record
11    (name varchar2(50),
12     courses course_v);
13  type professors_tbl is table of professor_info;
14
15  prof_name varchar2(50);
16  courses_list course_v := course_v();
17  prof_id_list emp_id_tbl;
18  professors_table professors_tbl := professors_tbl();
19
20 begin
21  select a.id
22    bulk collect into prof_id_list
23  from angajat a
24  join job j on a.id_job = j.id
25  where j.denumire = 'Professor';
26
27  professors_table.extend(prof_id_list.count);
28
29  for i in prof_id_list.first..prof_id_list.last loop
30    select name || ' ' || prenume
31      into prof_name
32    from angajat
33    where id = prof_id_list(i);
34
35    select distinct nume_curs
36      bulk collect into courses_list
37    from predare
38    where id_profesor = prof_id_list(i);
39
40    professors_table(i).name := prof_name;
```

Diagnostics

Task completed in 0.042 seconds

Procedure P6 compiled

PL/SQL procedure successfully completed.

Diagnostics Output

Lista profesorilor si a cursurilor pe care

1) Dima Carol-Valentin (ID 3) preda:

- Axonometrie
- Poliedre
- Soari
- Umbre 2D

2) Grăia Gabriel (ID 6) preda:

- Arhitectul
- Orașul Traditional
- Vernacular

Lista profesorilor si a cursurilor pe care

1) Dima Carol-Valentin (ID 3) preda:

- Axonometrie
- Poliedre
- Soari
- Umbre 2D

2) Grăia Gabriel (ID 6) preda:

- Arhitectul
- Orașul Traditional
- Vernacular


```

1. -- 6.
2. -- Afisati toti angajatii Atelierului care activeaza ca Profesori, impreuna cu
   lista Cursurilor predate de acestia (are mai multe aplicatii reale);
3.
4. create or replace procedure P6 is
5.     type courses_v is varray(30) of varchar2(30);
6.     type emp_id_tbl is table of number(4);
7.     type professor_info is record
8.         (name varchar2(30),
9.          courses courses_v);
10.    type professors_tbl is table of professor_info;
11.
12.    prof_name varchar2(30);
13.    courses_list courses_v := courses_v();
14.    prof_id_list emp_id_tbl;
15.    professors_table professors_tbl := professors_tbl();
16.
17. begin
18.     select a.id
19.     bulk collect into prof_id_list
20.     from angajat a
21.     join job j on a.id_job = j.id
22.     where j.denumire = 'Profesor';
23.
24.     professors_table.extend(prof_id_list.count);
25.
26.     for i in prof_id_list.first..prof_id_list.last loop
27.         select nume || ' ' || prenume
28.         into prof_name
29.         from angajat
30.         where id = prof_id_list(i);
31.
32.         select distinct nume_curs
33.         bulk collect into courses_list
34.         from predare
35.         where id_profesor = prof_id_list(i);
36.
37.         professors_table(i).name := prof_name;
38.         professors_table(i).courses := courses_list;
39.     end loop;
40.
41.     dbms_output.put_line('Lista profesorilor si a cursurilor pe care le
        sustin:');
42.     dbms_output.new_line;
43.     dbms_output.new_line;
44.
45.     for i in professors_table.first..professors_table.last loop
46.         dbms_output.put_line(i || ') ' || professors_table(i).name || ' (ID '
            || prof_id_list(i) || ') preda: ');
47.         for j in
            professors_table(i).courses.first..professors_table(i).courses.last loop
48.             dbms_output.put_line(' - ' || professors_table(i).courses(j));
49.         end loop;
50.         dbms_output.new_line;
51.     end loop;
52. end P6;
53. /
54. execute P6;
55. /

```

7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

“Afisati clasamentul studentilor (in functie de nota obtinuta) de la fiecare curs predat, impreuna cu media notelor (pentru a-i trimite la olimpiade si concursuri)”.

Retinem intr-un cursor parametrizat, pentru fiecare curs: id-ul, numele si nota fiecarui student ce a parcurs acest curs.

Pentru fiecare curs din cursor, realizam clasamentul studentilor in functie de nota, iar daca mai multi studenti au aceeasi nota, ii vom clasa pe aceeasi pozitie.

In final, afisam media notelor de la curs.

```
61 -- 7.
62 -- Afisati clasamentul studentilor (in functie de nota obtinuta) de la fiecare curs predat, impreuna cu media notelor (pentru a-i trimite la olimpiade si concursuri):
63
64 create or replace procedure P7 is
65     type courses_tbl is table of varchar2(30);
66     type student_id_tbl is table of number(4, 0);
67
68     course_list courses_tbl := courses_tbl();
69     student_id_list student_id_tbl := student_id_tbl();
70
71     cursor c(course_name varchar2) is
72         select n.denumire_curs as course, s.id as id, s.nume || ' ' || s.prenume as name, n.nota as mark
73         from nota n
74         join student s on n.id_student = s.id
75         where denumire_curs = course_name
76         order by denumire_curs asc, nota desc;
77
78     nr_students number(4);
79     sum_marks_course float;
80     previous_mark float;
81     rank number(4);
82
83     begin
84         select distinct denumire_curs
85         bulk collect into course_list
86         from nota
87         order by denumire_curs;
88
89         for i in course_list.first..course_list.last loop
90             nr_students := 0;
91             sum_marks_course := 0;
92             previous_mark := 0;
93             rank := 0;
94             dbms_output.put_line(course_list(i) || ' ');
95             for std in c(course_list(i)) loop
96                 if std.mark <= previous_mark then
97                     rank := rank;
98                 else
99                     rank := nr_students + 1;
100                 end if;
101                 nr_students := nr_students + 1;
102                 sum_marks_course := sum_marks_course + std.mark;
103             end loop;
104             previous_mark := std.mark;
105             dbms_output.put_line('Media notelor: ' || (sum_marks_course / nr_students));
106         end loop;
107     end;
```

Task completed in 0.055 seconds

Procedure P7 compiled

PL/SQL procedure successfully completed.

Arhitectură:

1. Stefanescu Bob - 10
2. Dobre Marcel - 9.1
3. Paun Cristinel - 9
4. Barbu Matei - 8.1
5. Rosevilici Teodor - 6
6. Popos Florentina - 3.2

Media notelor: 7.56

Axonometrie:

1. Sotau Marin - 9
2. Giurguleanu Matei - 8.5
3. Bitulescu Carol - 7.1

Media notelor: 8.2

Orașul Traditional:

1. Stefanescu Bob - 8.5
2. Barbu Matei - 4.3
3. Rosevilici Teodor - 4

Media notelor: 5.6

Poliedre:

1. Sotau Marin - 9.5
2. Bitulescu Carol - 8.5
3. Stefanescu Bob - 7.2
4. Robert Ioan - 7
5. Barbu Matei - 6
6. Rosevilici Teodor - 5

Media notelor: 7.2

Scări:

1. Giurguleanu Matei - 7.5

Media notelor: 7.5

Umbre 2D:

1. Stefanescu Bob - 8.9
2. Barbu Matei - 8.5
3. Rosevilici Teodor - 6

```

1. -- 7.
2. -- Afisati clasamentul studentilor (in functie de nota obtinuta) de la fiecare
   curs predat, impreuna cu media notelor (pentru a-i trimite la olimpiade si
   concursuri);
3.
4. create or replace procedure P7 is
5.     type courses_tbl is table of varchar2(30);
6.     type student_id_tbl is table of number(6, 0);
7.
8.     course_list courses_tbl := courses_tbl();
9.     student_id_list student_id_tbl := student_id_tbl();
10.
11.     cursor c(course_name varchar2) is
12.         select n.denumire_curs as course, s.id as id, s.nume || ' ' ||
           s.prenume as name, n.nota as mark
13.         from nota n
14.         join student s on n.id_student = s.id
15.         where denumire_curs = course_name
16.         order by denumire_curs asc, nota desc;
17.
18.     nr_students number(4);
19.     sum_marks_course float;
20.     previous_mark float;
21.     rank number(4);
22.
23. begin
24.     select distinct denumire_curs
25.     bulk collect into course_list
26.     from nota
27.     order by denumire_curs;
28.
29.     for i in course_list.first..course_list.last loop
30.         nr_students := 0;
31.         sum_marks_course := 0;
32.         previous_mark := 0;
33.         rank := 0;
34.         dbms_output.put_line(course_list(i) || ':');
35.
36.         for std in c(course_list(i)) loop
37.             if std.mark != previous_mark then
38.                 rank := rank + 1;
39.                 previous_mark := std.mark;
40.             end if;
41.             dbms_output.put_line(' ' || rank || '. ' || std.name || ' - ' ||
               std.mark);
42.             nr_students := nr_students + 1;
43.             sum_marks_course := sum_marks_course + std.mark;
44.         end loop;
45.         dbms_output.new_line;
46.         dbms_output.put_line('Media notelor: ' || trunc(sum_marks_course /
               nr_students, 2));
47.         dbms_output.put_line('-----');
48.         dbms_output.new_line;
49.     end loop;
50. end P7;
51. /
52. execute P7;
53. /

```

8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

“Creați o Funcție care sa intoarca lista Angajatilor unui Departament, care au lucrat la un Proiect trimis ca parametru (pentru ca proiectul a generat o suma mare de bani)”.

Retinem intr-o colectie de tip nested table, numele angajatilor care au fost angajati inainte de data de inceput a proiectului si care au terminat acel proiect aferent unui departament.

Funcția primește ca parametru numele proiectului.

EXCEPTIILE tratate:

- **NO_PROJECT_FOUND** cand nu exista niciun proiect cu numele dat in evidenta atelierului;
- **NO_EMPLOYEES_FOUND** cand proiectul a fost realizat de angajati care nu mai lucreaza in prezent in acel departament.

The screenshot displays the Oracle SQL Developer environment. The main window shows a PL/SQL script for a function named 'P8'. The script is designed to return a nested table of employee names who worked on a specific project before its start date. It includes exception handling for 'NO_PROJECT_FOUND' and 'NO_EMPLOYEES_FOUND'. The script is as follows:

```
113 -- 8.
114 -- Creați o Funcție care sa intoarca lista Angajatilor unui Departament, care au lucrat la un Proiect trimis ca parametru (pentru ca proiectul
115 -- a generat o suma mare de bani)
116
117 create or replace type emp_tbl as table of varchar2(30);
118 /
119
120
121 create or replace function P8 (project_name project.denumire%type) return varchar2 is
122     employees_list emp_tbl := emp_tbl();
123     employees_string varchar2(500) := '';
124     nr_projects number(4) := 0;
125
126     NO_PROJECT_FOUND exception;
127     pragma exception_init (NO_PROJECT_FOUND, -20001);
128     NO_EMPLOYEES_FOUND exception;
129     pragma exception_init (NO_EMPLOYEES_FOUND, -20002);
130
131 begin
132     select count(*)
133     into nr_projects
134     from project
135     where denumire = project_name;
136
137     if nr_projects = 0 then
138         raise NO_PROJECT_FOUND;
139     end if;
140
141     select a.name || ' ' || a.prenume as nume
142     bulk collect into employees_list
143     from angajat a
144     join departament d on a.departament = d.denumire
145     join project p on d.denumire = p.departament
146     where p.denumire = project_name and nvl(p.data_final, SYSDATE) > a.data_angajare
147     order by a.id;
148
149     if employees_list.count = 0 then
150         raise NO_EMPLOYEES_FOUND;
151     end if;
152
```

The 'Demos Output' window on the right shows the results of the function call for the project 'Macheta Residence'. It displays the list of employees who worked on this project before its completion date. The output is as follows:

```
Lista Angajatilor care au lucrat la Proiectul "Macheta Residence": Dragulescu Raluca
Nu exista niciun proiect cu acest nume!
Proiectul a fost realizat de angajati care nu mai lucreaza in cadrul firmei!
```

The 'Script Output' window at the bottom shows the status of the script execution: 'Task completed in 0.081 seconds' and 'PL/SQL procedure successfully completed.'

```

1. -- 8.
2. -- Creati o Functie care sa intoarca lista Angajatilor unui Departament, care
   au lucrat la un Proiect trimis ca parametru (pentru ca proiectul a generat o
   suma mare de bani);
3. create or replace type emp_tbl as table of varchar2(30);
4. /
5.
6. create or replace function P8 (project_name proiect.denumire%type) return
   varchar2 is
7.     employees_list emp_tbl := emp_tbl();
8.     employees_string varchar2(500) := '';
9.     nr_projects number(4) := 0;
10.
11.     NO_PROJECT_FOUND exception;
12.     pragma exception_init (NO_PROJECT_FOUND, -20001);
13.     NO_EMPLOYEES_FOUND exception;
14.     pragma exception_init (NO_EMPLOYEES_FOUND, -20002);
15.
16. begin
17.     select count(*)
18.     into nr_projects
19.     from proiect
20.     where denumire = project_name;
21.
22.     if nr_projects = 0 then
23.         raise NO_PROJECT_FOUND;
24.     end if;
25.
26.     select a.numa || ' ' || a.prenume as nume
27.     bulk collect into employees_list
28.     from angajat a
29.     join departament d on a.departament = d.denumire
30.     join proiect p on d.denumire = p.departament
31.     where p.denumire = project_name and nvl(p.data_final, SYSDATE) >
       a.data_angajare
32.     order by a.id;
33.
34.     if employees_list.count = 0 then
35.         raise NO_EMPLOYEES_FOUND;
36.     end if;
37.
38.     for i in employees_list.first..employees_list.last loop
39.         employees_string := employees_string || employees_list(i) || ', ';
40.     end loop;
41.
42.     return rtrim(employees_string, ', ');
43.
44.     exception
45.         when NO_PROJECT_FOUND then
46.             dbms_output.put_line('Nu exista niciun proiect cu acest nume!');
47.             raise_application_error(-20001, 'No Project found that matched the
       string passed as parameter to P8!');
48.         when NO_EMPLOYEES_FOUND then
49.             dbms_output.put_line('Proiectul a fost realizat de angajati care
       nu mai lucreaza in cadrul firmei!');
50.             raise_application_error(-20002, 'No Employees found that were
       employed before the Project started!');
51. end P8;
52. /

```

```

53. begin    -- OK;
54.  dbms_output.put_line('Lista Angajatilor care au lucrat la Proiectul "Macheta
    Residence":  ' || P8('Macheta Residence'));
55. end;
56. /
57. begin    -- NO_PROJECT_FOUND;
58.  dbms_output.put_line('Nu exista niciun proiect cu acest nume!' ||
    P8('denumire'));
59. end;
60. /
61. begin    -- NO_EMPLOYEES_FOUND;
62.  dbms_output.put_line('Nu exista niciun angajat in prezent in firma care a
    lucrat la acest proiect!' || P8('Restaurare Castel'));
63. end;
64. /

```

9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

“Creați o Procedura care sa afiseze toate Locatiile la care a invatat un Student dat ca parametru (care si-a pierdut telefonul la o locatie).”

Retinem intr-un cursor parametrizat, pentru fiecare student, locatiile la care a invatat acesta, prin join-uri pe 5 tabele.

Procedura primeste ca parametru numele unui student.

EXCEPTIILE tratate:

- **NO_DATA_FOUND** cand nu exista studenti cu numele introdus
- **TOO_MANY_ROWS** cand exista mai multi studenti cu numele introdus
- **NO_LOCATION_FOUND** cand studentul cu numele introdus nu a invatat la nicio locatie

The screenshot displays the Oracle SQL Developer environment. The main window shows a PL/SQL procedure named P9, which is designed to find locations where a specific student has studied. The procedure takes a student name as input and uses a cursor to join five tables: STUDENT, SALA, PREDARE, GRUPE, and STUDENT_GROUP. It handles three exceptions: NO_DATA_FOUND (if the student name is not found), TOO_MANY_ROWS (if multiple students match the name), and NO_LOCATION_FOUND (if the student has not studied at any location). The procedure outputs the student's name and the locations they have studied at.

```

182 -- 9.
183 -- Creați o Procedura care sa afiseze toate Locatiile la care a invatat un Student dat ca parametru (care si-a pierdut telefonul la o locatie);
184 create or replace procedure P9(std_name varchar2) is
185   student_id number(4);
186   nr_locations number(4) := 0;
187
188   cursor l(std_id number) is
189     select distinct l.id as loc, s.id as id
190     from locatie l
191     join sala h on l.id = h.id_locatie
192     join predaire p on h.id = p.id_sala
193     join grupe g on p.id_grupa = g.id
194     join student s on g.id = s.id_grupa
195     where s.id = std_id
196     order by l.id;
197
198   NO_LOCATION_FOUND exception;
199   pragma exception_init (NO_LOCATION_FOUND, -20003);
200
201 begin
202   select id
203   into student_id
204   from student
205   where name || ' ' || prenume = std_name;
206
207   dbms_output.put_line('Studentul ' || std_name || ' (ID ' || student_id || ') a invatat in locatiile cu ID-urile:');
208
209   for location in l(student_id) loop
210     nr_locations := nr_locations + 1;
211     dbms_output.put(' <' || location.loc || '> ');
212   end loop;
213   dbms_output.new_line;
214
215   if nr_locations = 0 then
216     raise NO_LOCATION_FOUND;
217   end if;

```

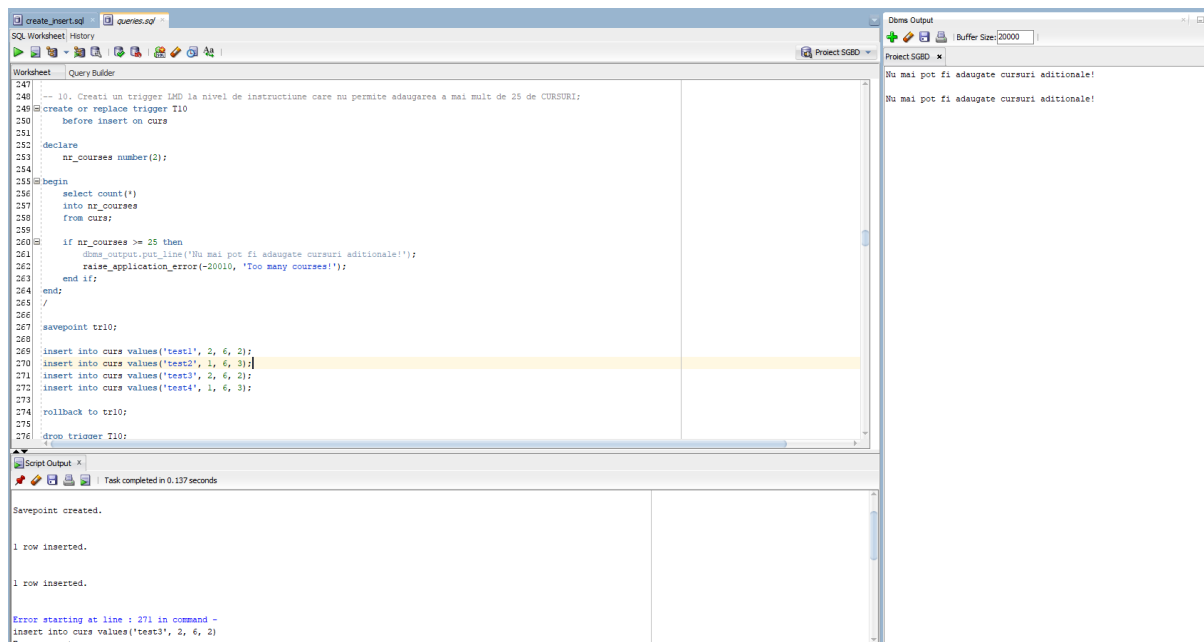
The output window shows the results of the procedure execution for two different student names. For 'Stefanescu Bob' (ID 5), it lists three locations: <1>, <2>, and <4>. For 'Pop Cosmin' (ID 12), it lists two locations: <1> and <2>. The output also includes messages indicating that no more students exist with the given name and that the specified student has not studied at any location.

```

1. - 9.
2. -- Creati o Procedura care sa afiseze toate Locatiile la care a invatat un Student dat
   ca parametru (care si-a pierdut telefonul la o locatie);
3. create or replace procedure P9(std_name varchar2) is
4.     student_id number(4);
5.     nr_locations number(4) := 0;
6.
7.     cursor l(std_id number) is
8.         select distinct l.id as loc, s.id as id
9.         from locatie l
10.        join sala h on l.id = h.id_locatie
11.        join predare p on h.id = p.id_sala
12.        join grupa g on p.id_grupa = g.id
13.        join student s on g.id = s.id_grupa
14.        where s.id = std_id
15.        order by l.id;
16.
17.     NO_LOCATION_FOUND exception;
18.     pragma exception_init (NO_LOCATION_FOUND, -20003);
19.
20. begin
21.     select id
22.     into student_id
23.     from student
24.     where nume || ' ' || prenume = std_name;
25.
26.     dbms_output.put_line('Studentul " ' || std_name || '" (ID ' || student_id || ') a
   invatat in locatiile cu ID-urile:');
27.
28.     for location in l(student_id) loop
29.         nr_locations := nr_locations + 1;
30.         dbms_output.put(' < ' || location.loc || '> ');
31.     end loop;
32.     dbms_output.new_line;
33.
34.     if nr_locations = 0 then
35.         raise NO_LOCATION_FOUND;
36.     end if;
37.
38.     exception
39.         when NO_DATA_FOUND then
40.             dbms_output.put_line('Nu exista niciun student cu acest nume!');
41.             raise_application_error(-20001, 'Query returned NO ROWS when asked about
   the student whose name was passed as parameter!');
42.         when TOO_MANY_ROWS then
43.             dbms_output.put_line('Exista mai multi studenti cu acest nume!');
44.             raise_application_error(-20002, 'Query returned MORE THAN ONE ROW when
   asked about the student whose name was passed as parameter!');
45.         when NO_LOCATION_FOUND then
46.             dbms_output.put_line('Acest student nu a invatat in nicio locatie inca!');
47.             raise_application_error(-20003, 'Query returned NO ROWS when asked about
   the location a student studied at!');
48.     end P9;
49. /
50.
51. -- OK;
52. execute P9('Stefanescu Bob');
53. -- TOO_MANY_ROWS;
54. execute P9('Barbu Matei');
55. -- NO_DATA_FOUND;
56. execute P9('nume student');
57. -- NO_LOCATION_FOUND;
58. execute P9('Pop Cosmin');
59. /

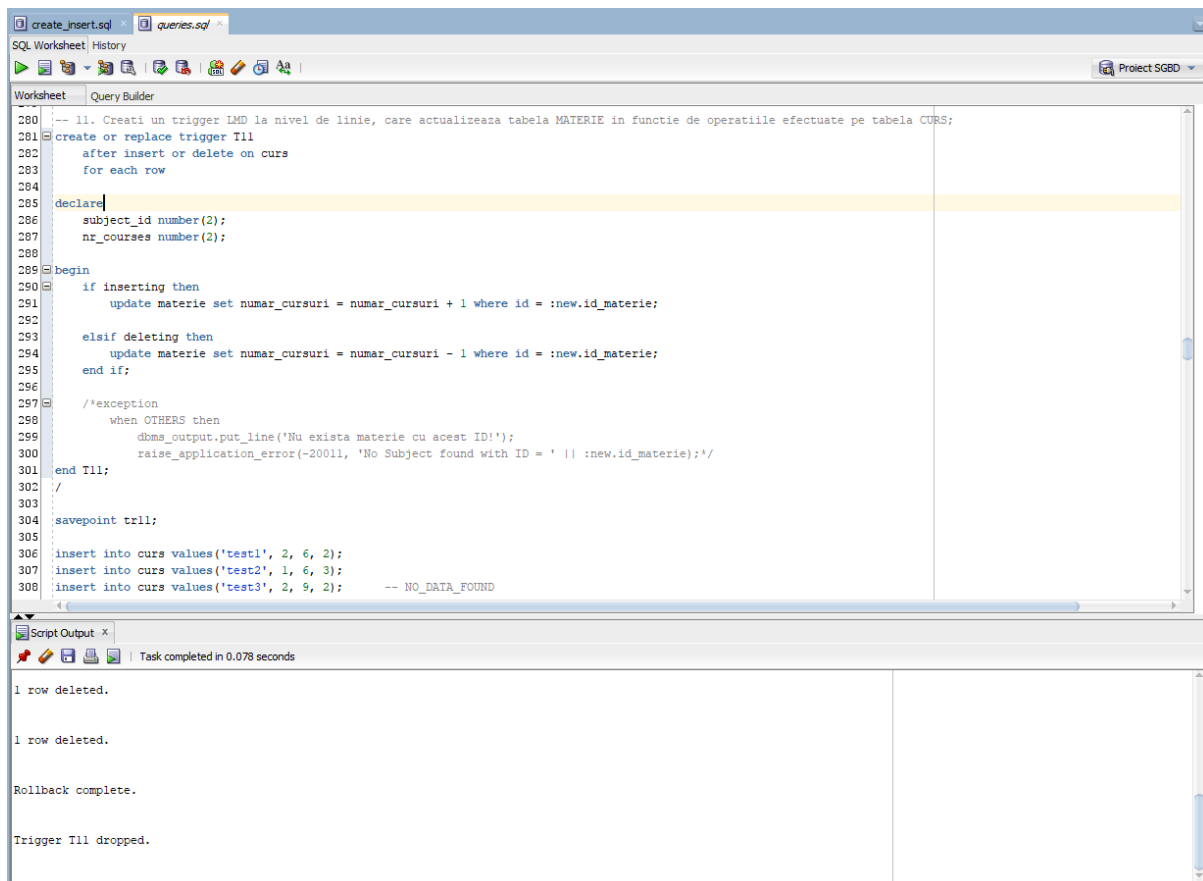
```


10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.



```
1. -- 10. Creati un trigger LMD la nivel de instructiune care nu permite adaugarea a mai  
   mult de 25 de CURSURI;  
2. create or replace trigger T10  
3.     before insert on curs  
4.  
5. declare  
6.     nr_courses number(2);  
7.  
8. begin  
9.     select count(*)  
10.    into nr_courses  
11.    from curs;  
12.  
13.    if nr_courses >= 25 then  
14.        dbms_output.put_line('Nu mai pot fi adaugate cursuri aditionale!');  
15.        raise_application_error(-20010, 'Too many courses!');  
16.    end if;  
17. end;  
18. /  
19.  
20. savepoint tr10;  
21.  
22. insert into curs values('test1', 2, 6, 2);  
23. insert into curs values('test2', 1, 6, 3);  
24. insert into curs values('test3', 2, 6, 2);  
25. insert into curs values('test4', 1, 6, 3);  
26.  
27. rollback to tr10;  
28.  
29. drop trigger T10;  
30. /
```

11. Definiți un trigger de tip LMD la nivel de linie.
Declanșați trigger-ul.



The screenshot displays an SQL IDE interface with two main panes. The top pane, titled 'Worksheet', contains a SQL script for creating and testing a trigger. The script is as follows:

```
280 -- 11. Creați un trigger LMD la nivel de linie, care actualizeaza tabela MATERIE in functie de operatiile efectuate pe tabela CURS;  
281 create or replace trigger T11  
282 after insert or delete on curs  
283 for each row  
284  
285 declare  
286     subject_id number(2);  
287     nr_courses number(2);  
288  
289 begin  
290     if inserting then  
291         update materie set numar_cursuri = numar_cursuri + 1 where id = :new.id_materie;  
292  
293     elsif deleting then  
294         update materie set numar_cursuri = numar_cursuri - 1 where id = :new.id_materie;  
295     end if;  
296  
297     /*exception  
298     when OTHERS then  
299         dbms_output.put_line('Nu exista materie cu acest ID!');  
300         raise_application_error(-20011, 'No Subject found with ID = ' || :new.id_materie);*/  
301 end T11;  
302 /  
303  
304 savepoint tr11;  
305  
306 insert into curs values('test1', 2, 6, 2);  
307 insert into curs values('test2', 1, 6, 3);  
308 insert into curs values('test3', 2, 9, 2); -- NO_DATA_FOUND
```

The bottom pane, titled 'Script Output', shows the results of the script execution:

```
Task completed in 0.078 seconds  
  
1 row deleted.  
  
1 row deleted.  
  
Rollback complete.  
  
Trigger T11 dropped.
```

```

1. -- 11. Creati un trigger LMD la nivel de linie, care actualizeaza tabela MATERIE in
   functie de operatiile efectuate pe tabela CURS;
2. create or replace trigger T11
3.   after insert or delete on curs
4.   for each row
5.
6. declare
7.   subject_id number(2);
8.   nr_courses number(2);
9.
10. begin
11.   if inserting then
12.     update materie set numar_cursuri = numar_cursuri + 1 where id =
       :new.id_materie;
13.
14.   elsif deleting then
15.     update materie set numar_cursuri = numar_cursuri - 1 where id =
       :old.id_materie;
16.   end if;
17. end T11;
18. /
19.
20. savepoint tr11;
21.
22. insert into curs values('test1', 2, 6, 2);
23. insert into curs values('test2', 1, 6, 3);
24. insert into curs values('test3', 2, 9, 2);      -- parent key not found
25.
26. delete from curs where denumire = 'test1';
27. delete from curs where denumire = 'test2';
28.
29. rollback to tr11;
30.
31. drop trigger T11;
32. /

```

The screenshot displays the SQL Developer interface with the following components:

- Script Editor:** Shows the SQL script for creating trigger T11, inserting test data, and dropping the trigger. The script is executed successfully, with a message "Task completed in 0.023 seconds".
- Script Output:** Displays the messages "Trigger T11 compiled" and "Savepoint created."
- CURS Table:** A table with columns DENUMIRE, NUMAR_CURSURI, ID_MATERIE, and CREDITE. It contains 23 rows of data, including test entries and various course titles like "Cunoasterea Spatiului" and "Diagrama".
- MATERIE Table:** A table with columns ID, DENUMIRE, and NUMAR_CURSURI. It contains 6 rows of data, including "1 Studiu Forme", "2 Istoria Arhitecturii in Europa", and "6 Limba Engleza".

SQL Worksheet: History

Project: SGBO

Worksheet: Query Builder

```

289 -- begin
290 -- if inserting then
291 --     update materie set numar_cursuri = numar_cursuri + 1 where id = :new.id_materie;
292 --
293 -- elsif deleting then
294 --     update materie set numar_cursuri = numar_cursuri - 1 where id = :old.id_materie;
295 -- end if;
296 -- end T1;
297 /
298
299 savepoint tr1;
300
301 insert into curs values('test1', 2, 6, 2);
302 insert into curs values('test2', 1, 6, 3);
303 insert into curs values('test3', 2, 9, 2); -- parent key not found
304
305 delete from curs where denumire = 'test1';
306 delete from curs where denumire = 'test2';
307
308 rollback to tr1;
309
310 drop trigger T1;
311 /
312
313 -- 12. Creati un trigger LOG, care pastreaza in tabela LOG evenimentele de la nivelul BAZEI DE DATE;
314 -- create table LOG(
315 --     baza_date varchar2(20),
316 --     utilizator varchar2(30),

```

Script Output

Task completed in 0.047 seconds

1 row inserted.

1 row inserted.

Error starting at line : 303 in command -
insert into curs values('test3', 2, 9, 2)
Error report -
ORA-02291: integrity constraint (GRUIA.SYS_C007037) violated - parent key not found

CURS

DENUMIRE	NUMAR_ORE	ID_MATERIE	CREDITE
1 Cunoasterea Spatiului	2	1	2
2 Diagrame	2	1	3
3 Eseu Grafic	4	1	2
4 Subtractie si Aditie	2	1	3
5 Elemente Teoretice	1	2	2
6 Orasul Antic	1	2	2
7 Orasul Medieval	1	2	2
8 Orasul Industrial	1	2	2
9 Umbre 2D	2	3	4
10 Axonometrie	4	3	6
11 Poliedre	2	3	4
12 Scari	6	3	5
13 Vernacular	2	4	3
14 Recuperarea Orasului	1	4	2
15 Arhitectul	1	4	2
16 Orasul Traditional	1	4	2
17 Sisteme de Proiectie	4	5	5
18 Perspectiva la Fuga	2	5	4
19 Trasarea Umbrelor	4	5	6
20 Perspectiva la Calculator	2	5	4
21 Past Tenses	1	6	2
22 Architectural Vocabulary	2	6	3
23 Useful Phrases	2	6	2
24 test2	1	6	3
25 test1	2	6	2

MATERIE

ID	DENUMIRE	NUMAR_CURSURI
1	1 Studiul Formei	4
2	2 Istoria Asezarilor in Europa	4
3	3 Geometrie Descriptiva	4
4	4 Introducere in Arh. Cont.	4
5	5 Perspectiva	4
6	6 Limba Engleza	4

SQL Worksheet: History

Project: SGBO

Worksheet: Query Builder

```

289 -- begin
290 -- if inserting then
291 --     update materie set numar_cursuri = numar_cursuri + 1 where id = :new.id_materie;
292 --
293 -- elsif deleting then
294 --     update materie set numar_cursuri = numar_cursuri - 1 where id = :old.id_materie;
295 -- end if;
296 -- end T1;
297 /
298
299 savepoint tr1;
300
301 insert into curs values('test1', 2, 6, 2);
302 insert into curs values('test2', 1, 6, 3);
303 insert into curs values('test3', 2, 9, 2); -- parent key not found
304
305 delete from curs where denumire = 'test1';
306 delete from curs where denumire = 'test2';
307
308 rollback to tr1;
309
310 drop trigger T1;
311 /
312
313 -- 12. Creati un trigger LOG, care pastreaza in tabela LOG evenimentele de la nivelul BAZEI DE DATE;
314 -- create table LOG(
315 --     baza_date varchar2(20),
316 --     utilizator varchar2(30),

```

Script Output

Task completed in 0.034 seconds

Error starting at line : 303 in command -
insert into curs values('test3', 2, 9, 2)
Error report -
ORA-02291: integrity constraint (GRUIA.SYS_C007037) violated - parent key not found

1 row deleted.

1 row deleted.

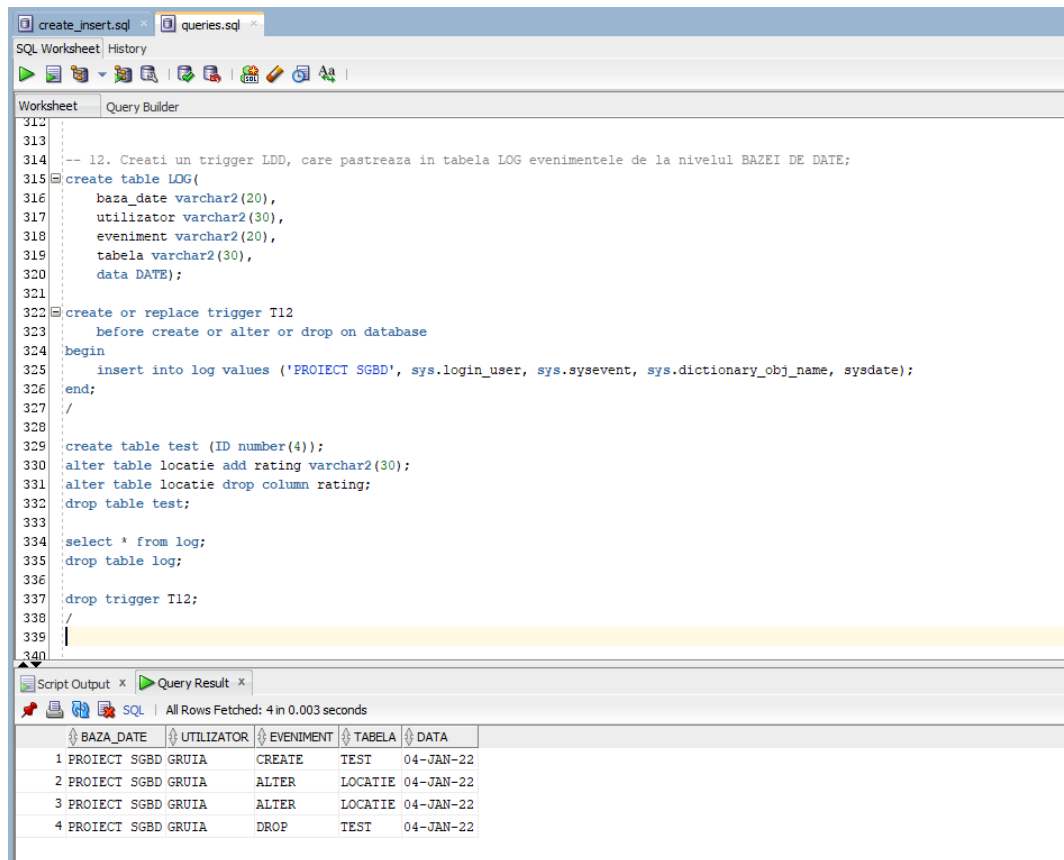
CURS

DENUMIRE	NUMAR_ORE	ID_MATERIE	CREDITE
1 Cunoasterea Spatiului	2	1	2
2 Diagrame	2	1	3
3 Eseu Grafic	4	1	2
4 Subtractie si Aditie	2	1	3
5 Elemente Teoretice	1	2	2
6 Orasul Antic	1	2	2
7 Orasul Medieval	1	2	2
8 Orasul Industrial	1	2	2
9 Umbre 2D	2	3	4
10 Axonometrie	4	3	6
11 Poliedre	2	3	4
12 Scari	6	3	5
13 Vernacular	2	4	3
14 Recuperarea Orasului	1	4	2
15 Arhitectul	1	4	2
16 Orasul Traditional	1	4	2
17 Sisteme de Proiectie	4	5	5
18 Perspectiva la Fuga	2	5	4
19 Trasarea Umbrelor	4	5	6
20 Perspectiva la Calculator	2	5	4
21 Past Tenses	1	6	2
22 Architectural Vocabulary	2	6	3
23 Useful Phrases	2	6	2

MATERIE

ID	DENUMIRE	NUMAR_CURSURI
1	1 Studiul Formei	4
2	2 Istoria Asezarilor in Europa	4
3	3 Geometrie Descriptiva	4
4	4 Introducere in Arh. Cont.	4
5	5 Perspectiva	4
6	6 Limba Engleza	4

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



The screenshot shows an SQL IDE with a script editor and a results pane. The script defines a trigger T12 that logs database events into a table named LOG. It also creates a test table, modifies it, and then drops it. The results pane shows the output of the script, which includes the creation of the LOG table, the execution of the trigger, and the creation and modification of the test table.

```
312
313
314 -- 12. Creați un trigger LDD, care pastreaza in tabela LOG evenimentele de la nivelul BAZEI DE DATE;
315 create table LOG(
316     baza_date varchar2(20),
317     utilizator varchar2(30),
318     eveniment varchar2(20),
319     tabela varchar2(30),
320     data DATE);
321
322 create or replace trigger T12
323     before create or alter or drop on database
324 begin
325     insert into log values ('PROIECT SGBD', sys.login_user, sys.sysevent, sys.dictionary_obj_name, sysdate);
326 end;
327 /
328
329 create table test (ID number(4));
330 alter table locatie add rating varchar2(30);
331 alter table locatie drop column rating;
332 drop table test;
333
334 select * from log;
335 drop table log;
336
337 drop trigger T12;
338 /
339
340
```

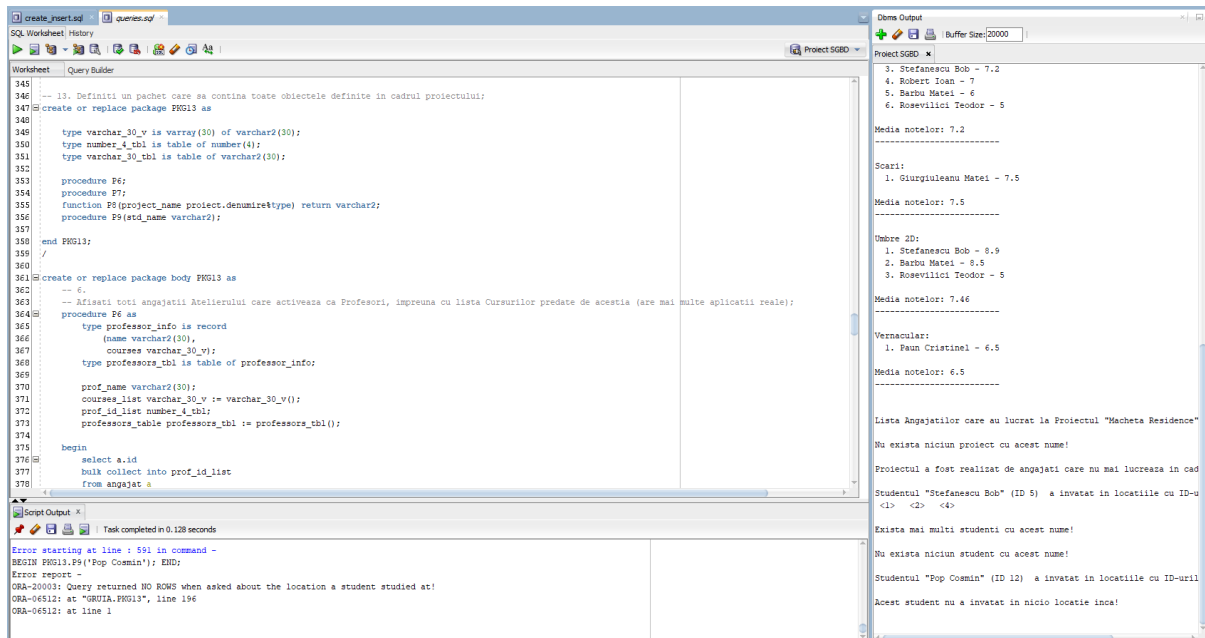
Script Output x Query Result x

SQL | All Rows Fetched: 4 in 0.003 seconds

	BAZA_DATE	UTILIZATOR	EVENTIMENT	TABELA	DATA
1	PROIECT SGBD GRUIA	CREATE	TEST	04-JAN-22	
2	PROIECT SGBD GRUIA	ALTER	LOCATIE	04-JAN-22	
3	PROIECT SGBD GRUIA	ALTER	LOCATIE	04-JAN-22	
4	PROIECT SGBD GRUIA	DROP	TEST	04-JAN-22	

1. -- 12. Creați un trigger LDD, care pastreaza in tabela LOG evenimentele de la nivelul BAZEI DE DATE;
2. create table LOG(
3. baza_date varchar2(20),
4. utilizator varchar2(30),
5. eveniment varchar2(20),
6. tabela varchar2(30),
7. data DATE);
8.
9. create or replace trigger T12
10. before create or alter or drop on database
11. begin
12. insert into log values ('PROIECT SGBD', sys.login_user, sys.sysevent, sys.dictionary_obj_name, sysdate);
13. end;
14. /
15.
16. create table test (ID number(4));
17. alter table locatie add rating varchar2(30);
18. alter table locatie drop column rating;
19. drop table test;
20.
21. select * from log;
22. drop table log;
23. drop trigger T12;
24. /

13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.



1. -- 13. Definiti un pachet care sa contina toate obiectele definite in cadrul proiectului;
2. create or replace package PKG13 as
- 3.
4. type varchar_30_v is varray(30) of varchar2(30);
5. type number_4_tbl is table of number(4);
6. type varchar_30_tbl is table of varchar2(30);
- 7.
8. procedure P6;
9. procedure P7;
10. function P8(project_name proiect.denumire%type) return varchar2;
11. procedure P9(std_name varchar2);
- 12.
13. end PKG13;
14. /
- 15.
16. create or replace package body PKG13 as
17. -- 6.
18. -- Afisati toti angajatii Atelierului care activeaza ca Profesori, impreuna cu lista Cursurilor predate de acestia (are mai multe aplicatii reale);
19. procedure P6 as
20. type professor_info is record
21. (name varchar2(30),
22. courses varchar_30_v);
23. type professors_tbl is table of professor_info;
- 24.
25. prof_name varchar2(30);
26. courses_list varchar_30_v := varchar_30_v();

```

27.         prof_id_list number_4_tbl;
28.         professors_table professors_tbl := professors_tbl();
29.
30.     begin
31.         select a.id
32.         bulk collect into prof_id_list
33.         from angajat a
34.         join job j on a.id_job = j.id
35.         where j.denumire = 'Profesor';
36.
37.         professors_table.extend(prof_id_list.count);
38.
39.         for i in prof_id_list.first..prof_id_list.last loop
40.             select nume || ' ' || prenume
41.             into prof_name
42.             from angajat
43.             where id = prof_id_list(i);
44.
45.             select distinct nume_curs
46.             bulk collect into courses_list
47.             from predare
48.             where id_profesor = prof_id_list(i);
49.
50.             professors_table(i).name := prof_name;
51.             professors_table(i).courses := courses_list;
52.         end loop;
53.
54.         dbms_output.put_line('Lista profesorilor si a cursurilor pe care le
sustin:');
55.         dbms_output.new_line;
56.         dbms_output.new_line;
57.
58.         for i in professors_table.first..professors_table.last loop
59.             dbms_output.put_line(i || ') ' || professors_table(i).name || '
(ID ' || prof_id_list(i) || ') predat: ');
60.             for j in
professors_table(i).courses.first..professors_table(i).courses.last loop
61.                 dbms_output.put_line(' - ' ||
professors_table(i).courses(j));
62.             end loop;
63.             dbms_output.new_line;
64.         end loop;
65.     end P6;
66.
67.
68.     -- 7.
69.     -- Afisati clasamentul studentilor (in functie de nota obtinuta) de la
fiecare curs predat, impreuna cu media notelor (pentru a-i trimite la
olimpiade si concursuri);
70.     procedure P7 as
71.         course_list varchar_30_tbl := varchar_30_tbl();
72.
73.         cursor c(course_name varchar2) is
74.             select n.denumire_curs as course, s.id as id, s.nume || ' ' ||
s.prenume as name, n.nota as mark
75.             from nota n
76.             join student s on n.id_student = s.id
77.             where denumire_curs = course_name
78.             order by denumire_curs asc, nota desc;

```

```

79.
80.     nr_students number(4);
81.     sum_marks_course float;
82.     previous_mark float;
83.     rank number(4);
84.
85.     begin
86.         select distinct denumire_curs
87.         bulk collect into course_list
88.         from nota
89.         order by denumire_curs;
90.
91.         for i in course_list.first..course_list.last loop
92.             nr_students := 0;
93.             sum_marks_course := 0;
94.             previous_mark := 0;
95.             rank := 0;
96.             dbms_output.put_line(course_list(i) || ':');
97.
98.             for std in c(course_list(i)) loop
99.                 if std.mark != previous_mark then
100.                     rank := rank + 1;
101.                     previous_mark := std.mark;
102.                 end if;
103.                 dbms_output.put_line(' ' || rank || '. ' || std.name || ' -
' || std.mark);
104.                 nr_students := nr_students + 1;
105.                 sum_marks_course := sum_marks_course + std.mark;
106.             end loop;
107.             dbms_output.new_line;
108.             dbms_output.put_line('Media notelor: ' || trunc(sum_marks_course
/ nr_students, 2));
109.             dbms_output.put_line('-----');
110.             dbms_output.new_line;
111.         end loop;
112.     end P7;
113.
114.
115.     -- 8.
116.     -- Creati o Functie care sa intoarca lista Angajatilor unui Departament,
    care au lucrat la un Proiect trimis ca parametru (pentru ca proiectul a
    generat o suma mare de bani);
117.     function P8 (project_name proiect.denumire%type) return varchar2 as
118.         employees_list varchar_30_tbl := varchar_30_tbl();
119.         employees_string varchar2(500) := '';
120.         nr_projects number(4) := 0;
121.
122.         NO_PROJECT_FOUND exception;
123.         pragma exception_init (NO_PROJECT_FOUND, -20001);
124.         NO_EMPLOYEES_FOUND exception;
125.         pragma exception_init (NO_EMPLOYEES_FOUND, -20002);
126.
127.     begin
128.         select count(*)
129.         into nr_projects
130.         from proiect
131.         where denumire = project_name;
132.
133.         if nr_projects = 0 then

```



```

134.         raise NO_PROJECT_FOUND;
135.     end if;
136.
137.     select a.nume || ' ' || a.prenume as nume
138.     bulk collect into employees_list
139.     from angajat a
140.     join departament d on a.departament = d.denumire
141.     join proiect p on d.denumire = p.departament
142.     where p.denumire = project_name and nvl(p.data_final, SYSDATE) >
143.         a.data_angajare
144.     order by a.id;
145.
146.     if employees_list.count = 0 then
147.         raise NO_EMPLOYEES_FOUND;
148.     end if;
149.
150.     for i in employees_list.first..employees_list.last loop
151.         employees_string := employees_string || employees_list(i) || ',
152.     ';
153.     end loop;
154.
155.     return rtrim(employees_string, ', ');
156.
157.     exception
158.         when NO_PROJECT_FOUND then
159.             dbms_output.put_line('Nu exista niciun proiect cu acest
160.         nume!');
161.             raise_application_error(-20001, 'No Project found that
162.         matched the string passed as parameter to P8!');
163.         when NO_EMPLOYEES_FOUND then
164.             dbms_output.put_line('Proiectul a fost realizat de angajati
165.         care nu mai lucreaza in cadrul firmei!');
166.             raise_application_error(-20002, 'No Employees found that
167.         were employed before the Project started!');
168.     end P8;
169.
170.
171.
172.     -- 9.
173.     -- Creati o Procedura care sa afiseze toate Locatiile la care a invatat
174.     un Student dat ca parametru (care si-a pierdut telefonul la o locatie);
175.     procedure P9(std_name varchar2) as
176.         student_id number(4);
177.         nr_locations number(4) := 0;
178.
179.         cursor l(std_id number) is
180.             select distinct l.id as loc, s.id as id
181.             from locatie l
182.             join sala h on l.id = h.id_locatie
183.             join predare p on h.id = p.id_sala
184.             join grupa g on p.id_grupa = g.id
185.             join student s on g.id = s.id_grupa
186.             where s.id = std_id
187.             order by l.id;
188.
189.         NO_LOCATION_FOUND exception;
190.         pragma exception_init (NO_LOCATION_FOUND, -20003);
191.
192.     begin
193.         select id

```

```

186.         into student_id
187.         from student
188.         where nume || ' ' || prenume = std_name;
189.
190.         dbms_output.put_line('Studentul "' || std_name || '" (ID ' ||
    student_id || ') a invatat in locatiile cu ID-urile:');
191.
192.         for location in l(student_id) loop
193.             nr_locations := nr_locations + 1;
194.             dbms_output.put(' <' || location.loc || '> ');
195.         end loop;
196.         dbms_output.new_line;
197.
198.         if nr_locations = 0 then
199.             raise NO_LOCATION_FOUND;
200.         end if;
201.
202.         exception
203.             when NO_DATA_FOUND then
204.                 dbms_output.put_line('Nu exista niciun student cu acest
    nume!');
205.                 raise_application_error(-20001, 'Query returned NO ROWS when
    asked about the student whose name was passed as parameter!');
206.             when TOO_MANY_ROWS then
207.                 dbms_output.put_line('Exista mai multi studenti cu acest
    nume!');
208.                 raise_application_error(-20002, 'Query returned MORE THAN
    ONE ROW when asked about the student whose name was passed as parameter!');
209.             when NO_LOCATION_FOUND then
210.                 dbms_output.put_line('Acest student nu a invatat in nicio
    locatie inca!');
211.                 raise_application_error(-20003, 'Query returned NO ROWS when
    asked about the location a student studied at!');
212.         end P9;
213.
214. end PKG13;
215. /
216.
217. -- TESTAM PACHETUL PKG13;
218. -- 6
219. execute PKG13.P6;
220.
221. -- 7
222. execute PKG13.P7;
223.
224. -- 8
225. begin -- OK;
226.     dbms_output.put_line('Lista Angajatilor care au lucrat la Proiectul
    "Macheta Residence": ' || PKG13.P8('Macheta Residence'));
227. end;
228. /
229. begin -- NO_PROJECT_FOUND;
230.     dbms_output.put_line('Nu exista niciun proiect cu acest nume!' ||
    PKG13.P8('denumire'));
231. end;
232. /
233. begin -- NO_EMPLOYEES_FOUND;
234.     dbms_output.put_line('Nu exista niciun angajat in prezent in firma care a
    lucrat la acest proiect!' || PKG13.P8('Restaurare Castel'));

```

```
235. end;
236. /
237.
238. -- 9
239. -- OK;
240. execute PKG13.P9('Stefanescu Bob');
241. -- TOO_MANY_ROWS;
242. execute PKG13.P9('Barbu Matei');
243. -- NO_DATA_FOUND;
244. execute PKG13.P9('nume student');
245. -- NO_LOCATION_FOUND;
246. execute PKG13.P9('Pop Cosmin');
```

Va multumesc pentru atentie acordata!

Student: Gabriel GRUIA

Grupa: 231

Data: 05.01.2021