Proiect SGBD

- Atelier Arhitectura -

***CUPRINS***

[1. Prezentarea pe scurt a bazei de date (utilitatea ei) 2](#_Toc92204544)

[2. Diagrama entitate-relatie (ERD) – in FN3 3](#_Toc92204545)

[3. Diagrama conceptuala 3](#_Toc92204546)

[4. Implementarea diagramei conceptuale in Oracle 4](#_Toc92204547)

[5. Popularea tabelelor cu inregistrari 17](#_Toc92204548)

[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](#_Toc92204549)

[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](#_Toc92204550)

[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](#_Toc92204551)

[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](#_Toc92204552)

[10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul. 33](#_Toc92204553)

[11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul. 34](#_Toc92204554)

[12. Definiți un trigger de tip LDD. Declanșați trigger-ul. 37](#_Toc92204555)

[13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului. 38](#_Toc92204556)

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

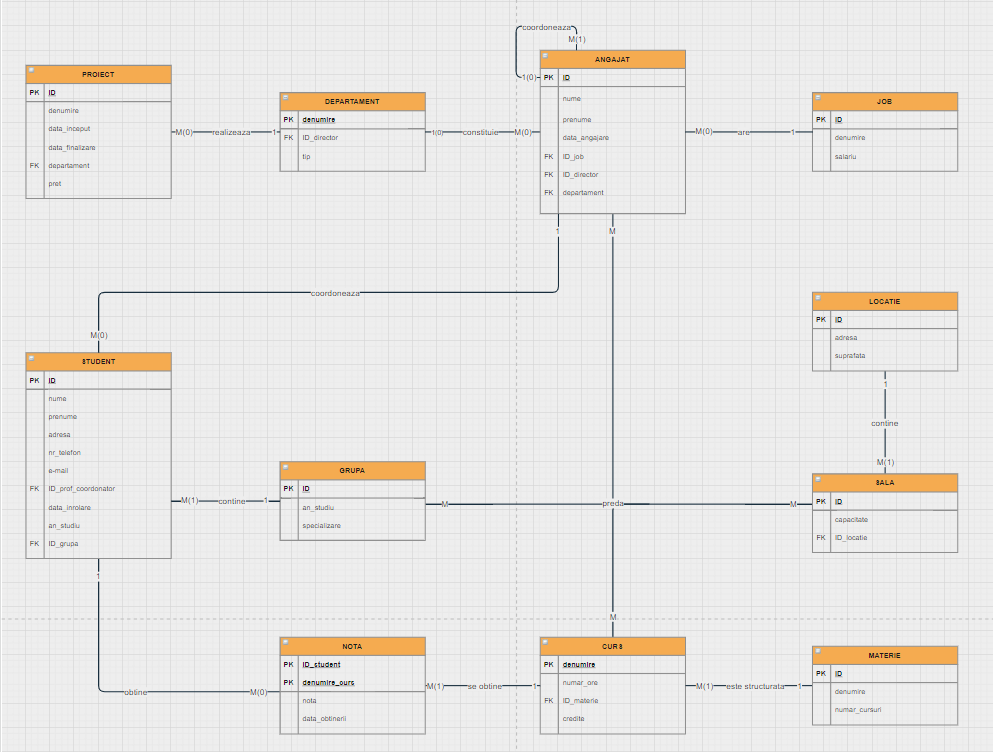
In prezentarea aceasta va voi introduce in cadrul **“*Atelierului de Arhitectura*”**, unde putem gasi specialisti in domeniul Arhitecturii, al Design-ului Interior si al Urbanismului, angajati in diverse Proiecte inovative, impreuna cu Profesorii care se ocupa de formarea viitorilor Arhitecti, prin predarea secretelor meseriei si prin indrumarea acestora catre succes in aceasta cariera esentiala in societate, bazata pe creativitate, stapanirea conceptelor teoretice si multa practica.

Scopul gestiunii acestei bazei de date este de a facilita operarea cu informatii si date legate de toate aspectele ce implica un Atelier de Arhitectura: realizarea proiectelor de catre angajati, activitatile de predare ale profesorilor si cele de invatare ale studentilor etc.

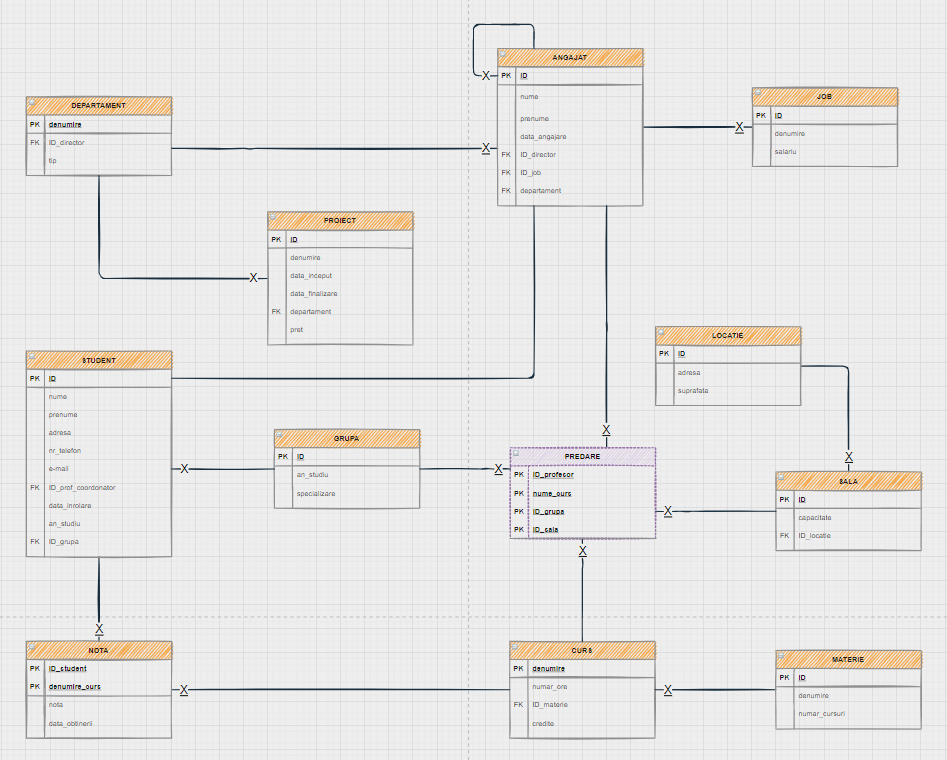
Prezentarea, pe scurt, a entitatilor si constrangerilor ce definesc baza de date:

* 12 tabele, dintre care 2 asociative (NOTA si PREDARE)
* fiecare angajat are un manager (sau null, in cazul in care acesta este managerul atelierului)
* fiecare angajat face parte dintr-un singur departament
* fiecare angajat primeste salariul in functie de job-ul in care activeaza
* un proiect poate fi realizat de un singur departament

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

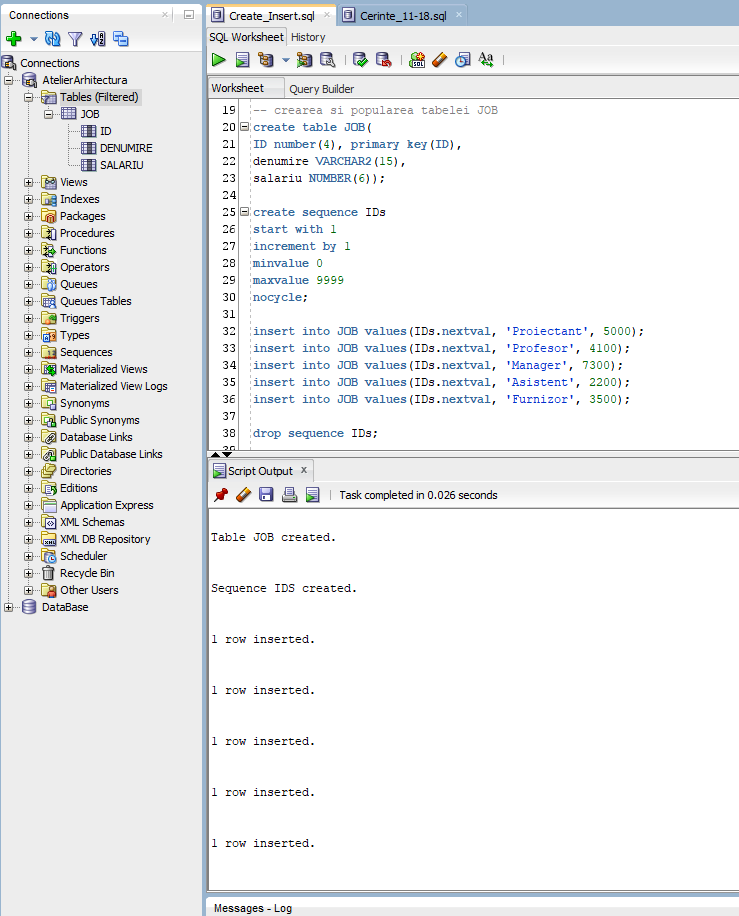


# 3. Diagrama conceptuala

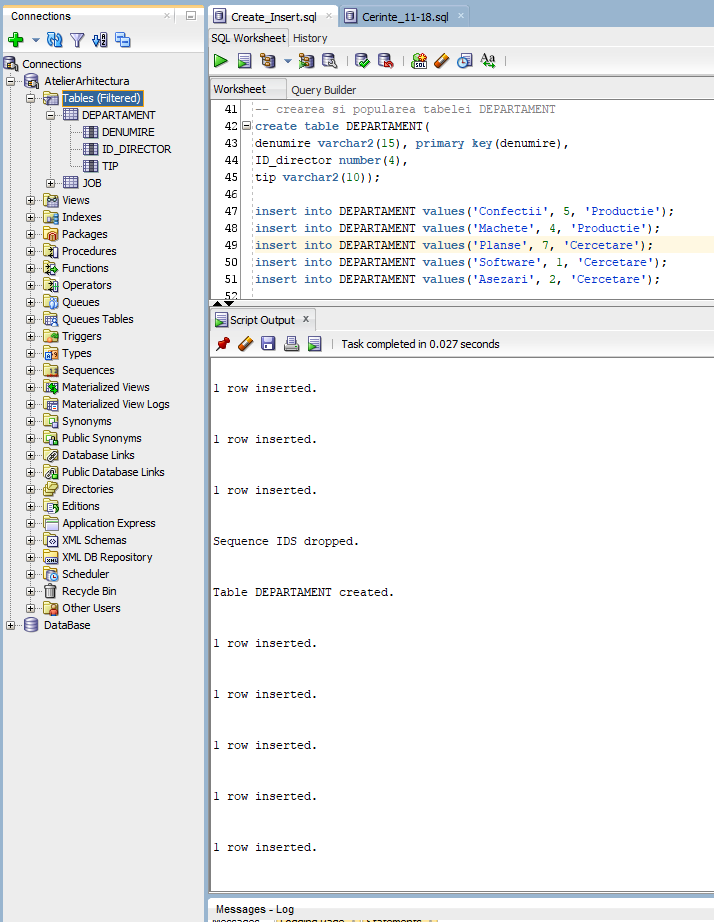


## 4. Implementarea diagramei conceptuale in Oracle

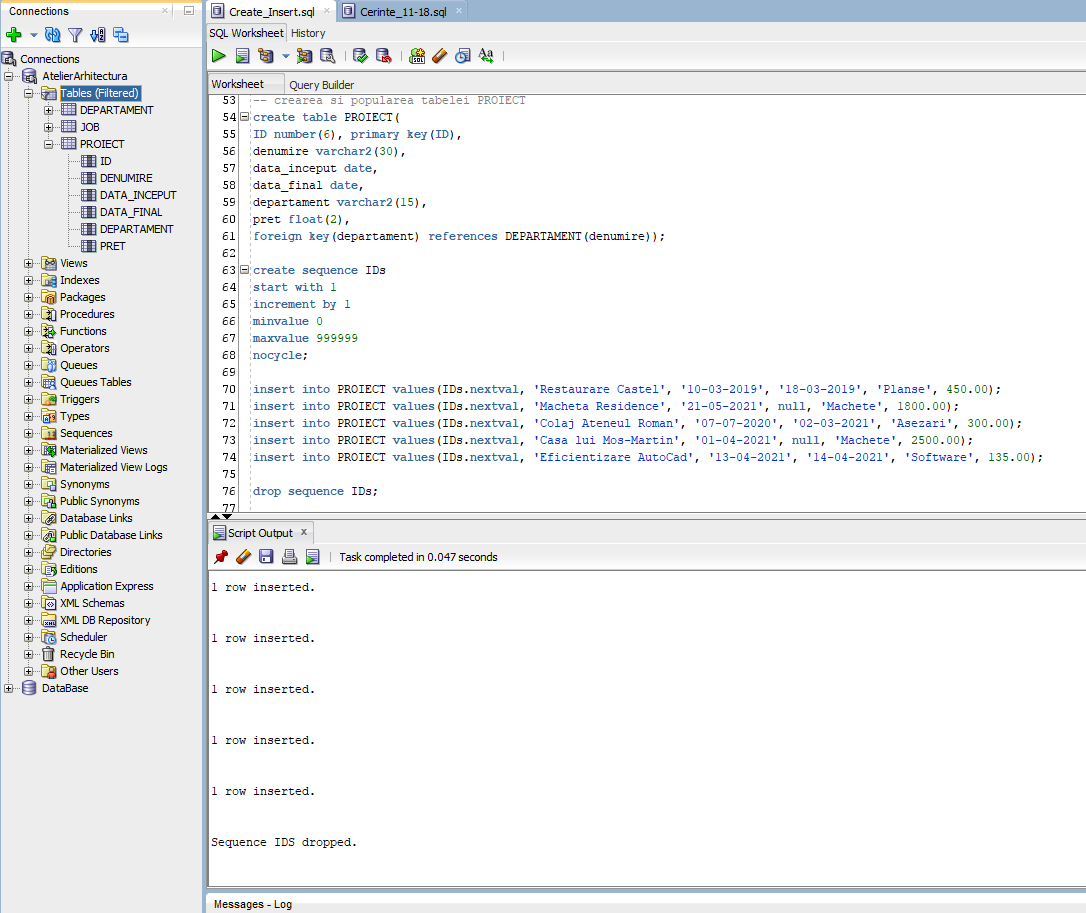
1. JOB



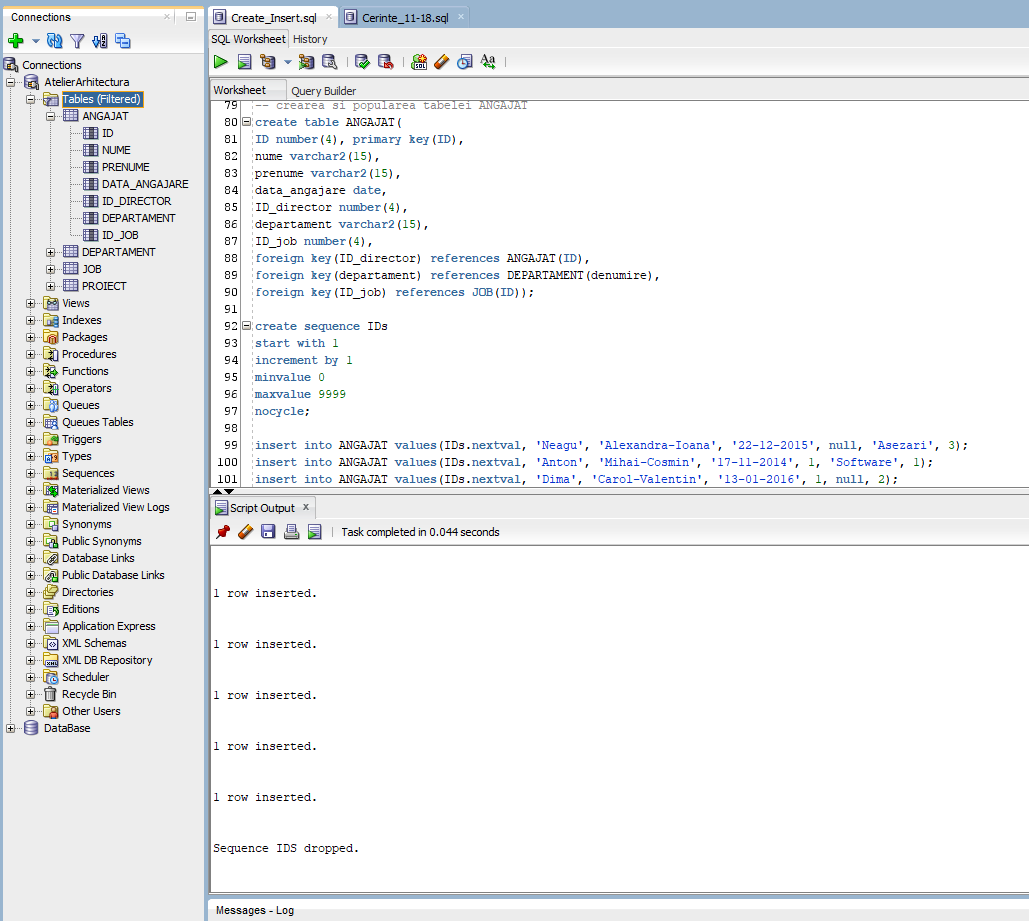
1. DEPARTAMENT



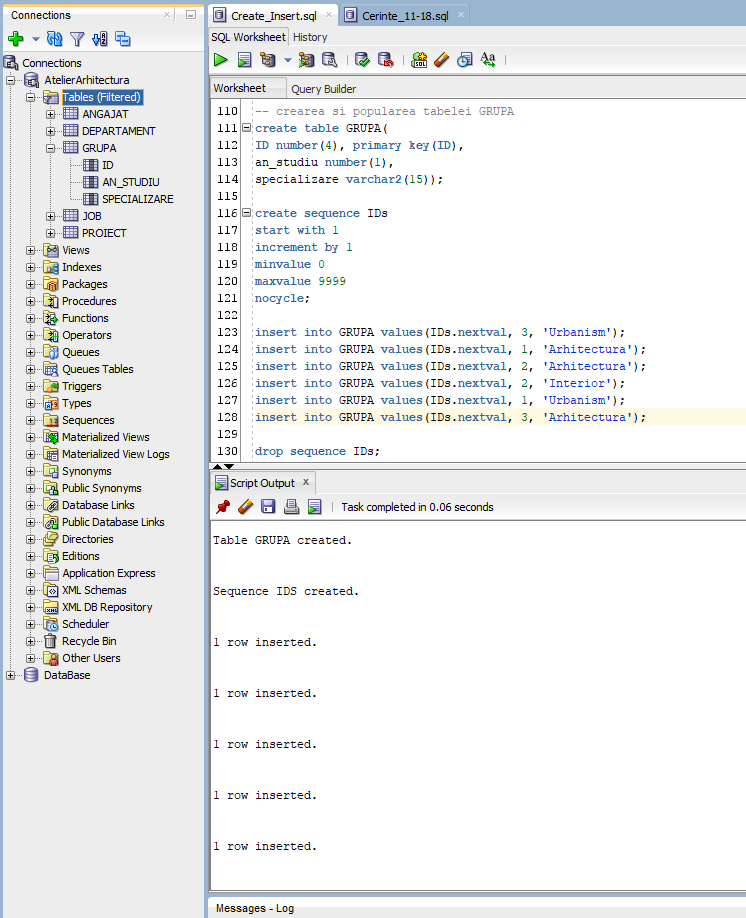
1. PROIECT



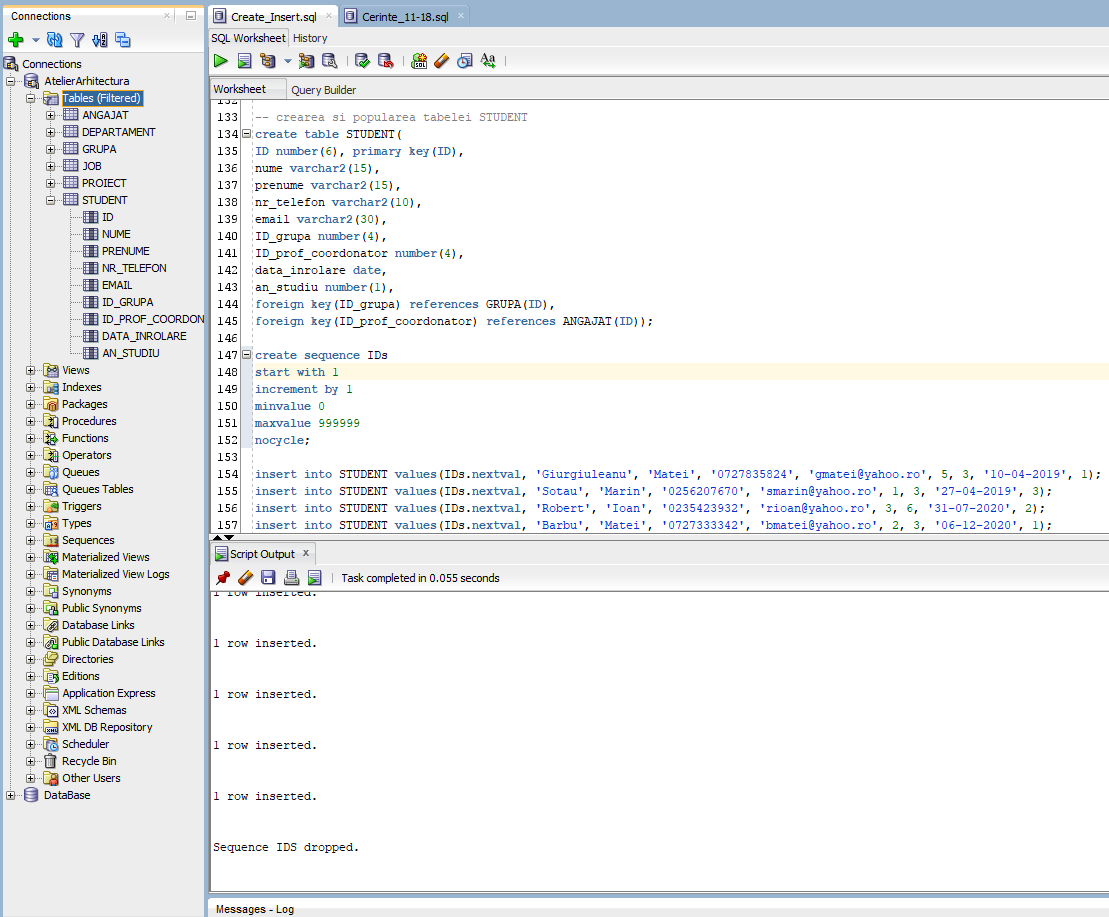
1. ANGAJAT



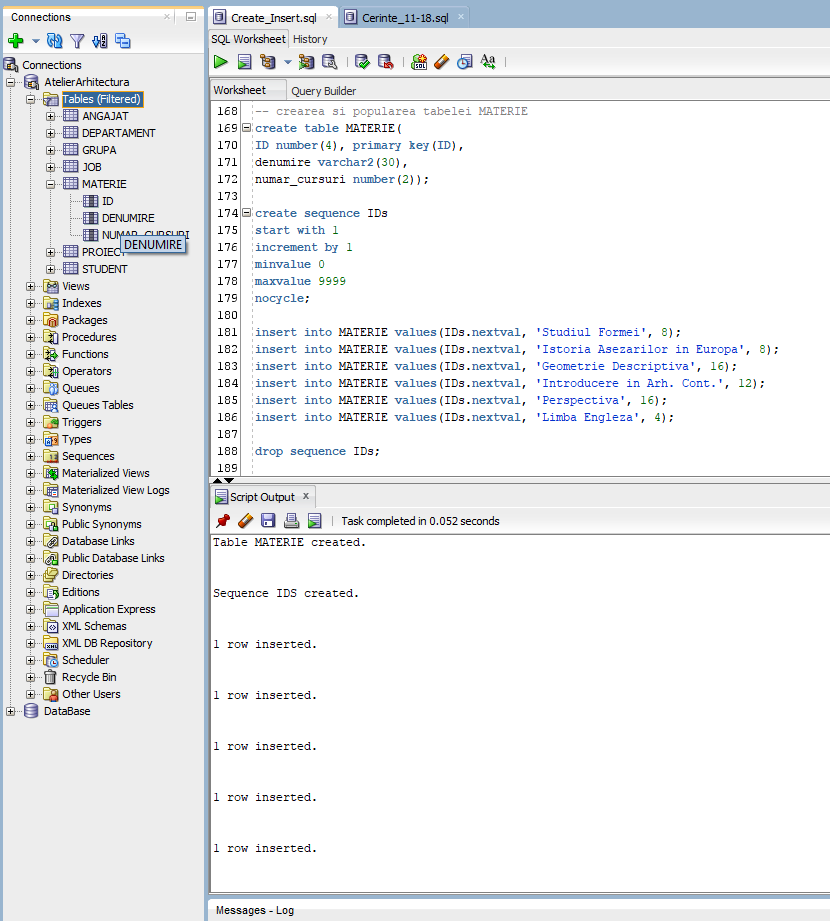
1. GRUPA



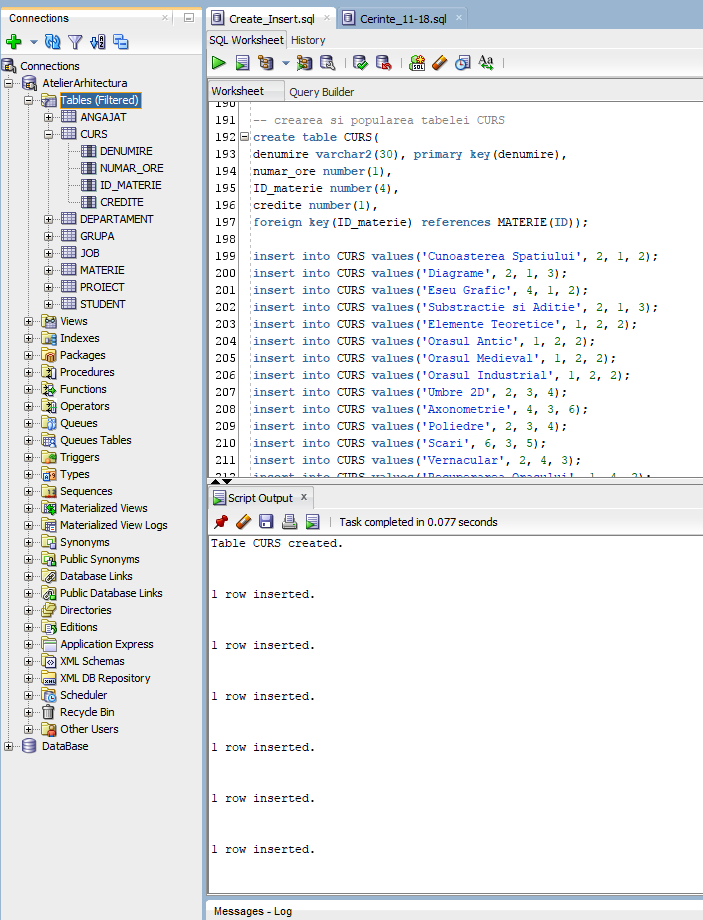
1. STUDENT



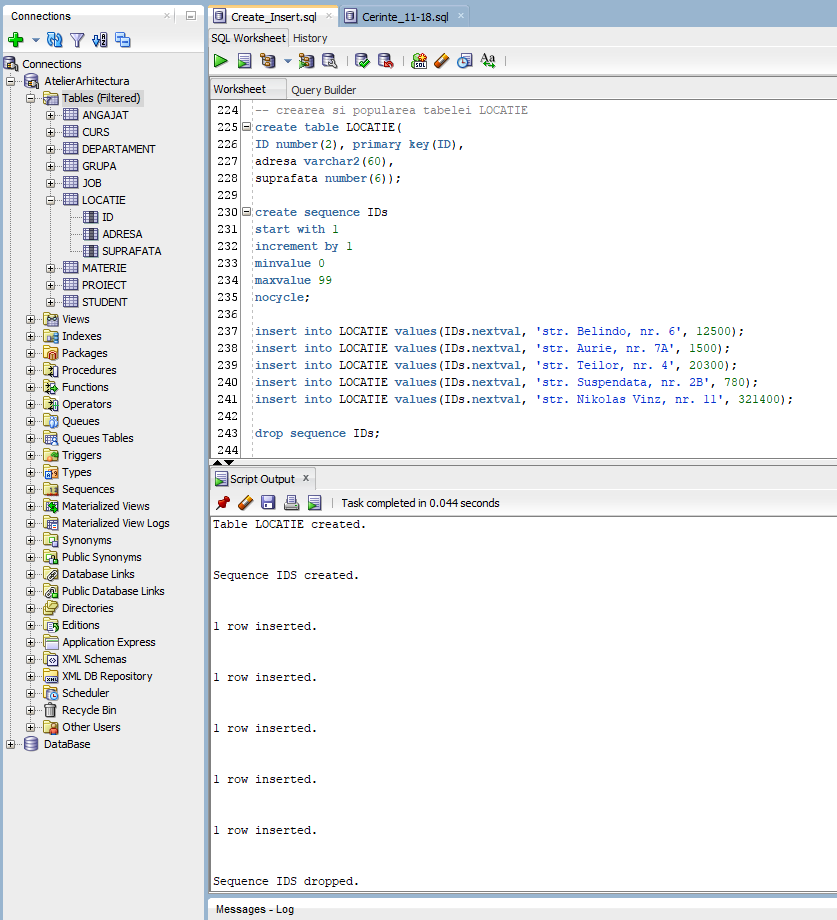
1. MATERIE



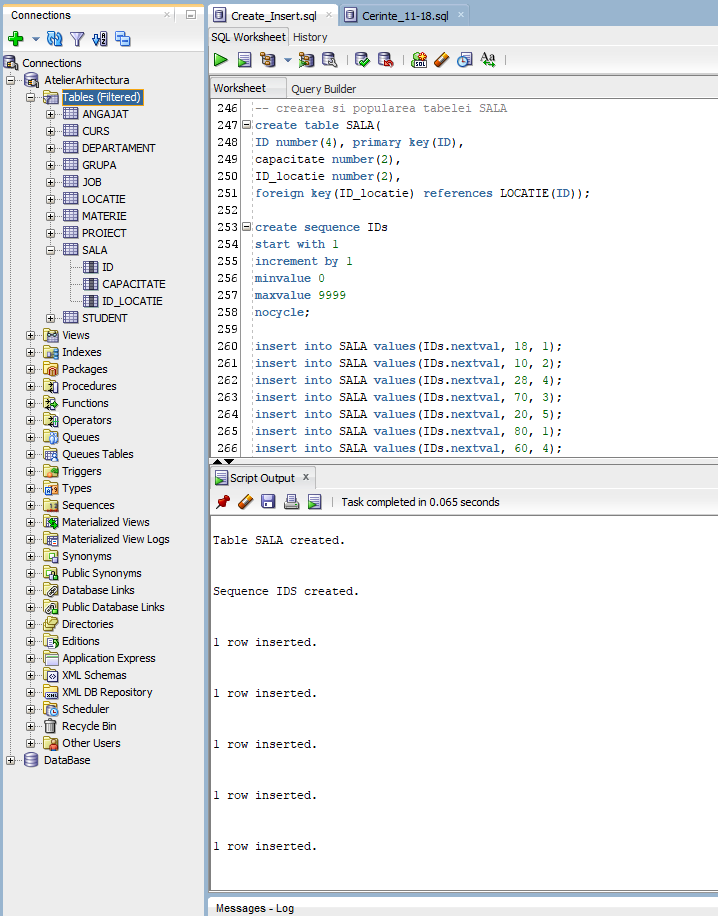
1. CURS



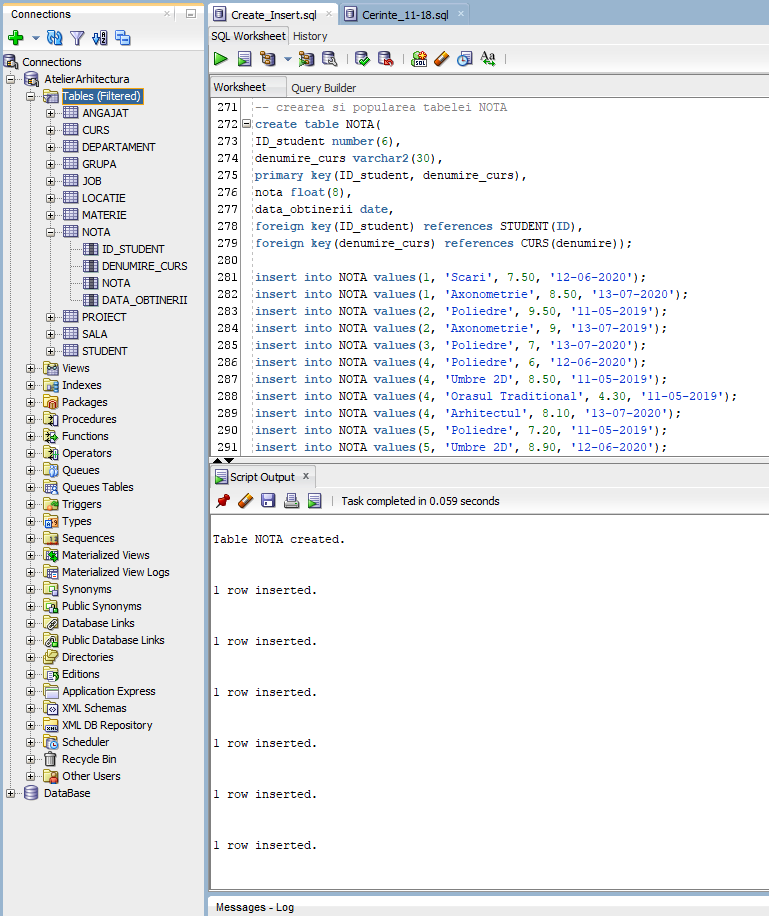
1. LOCATIE



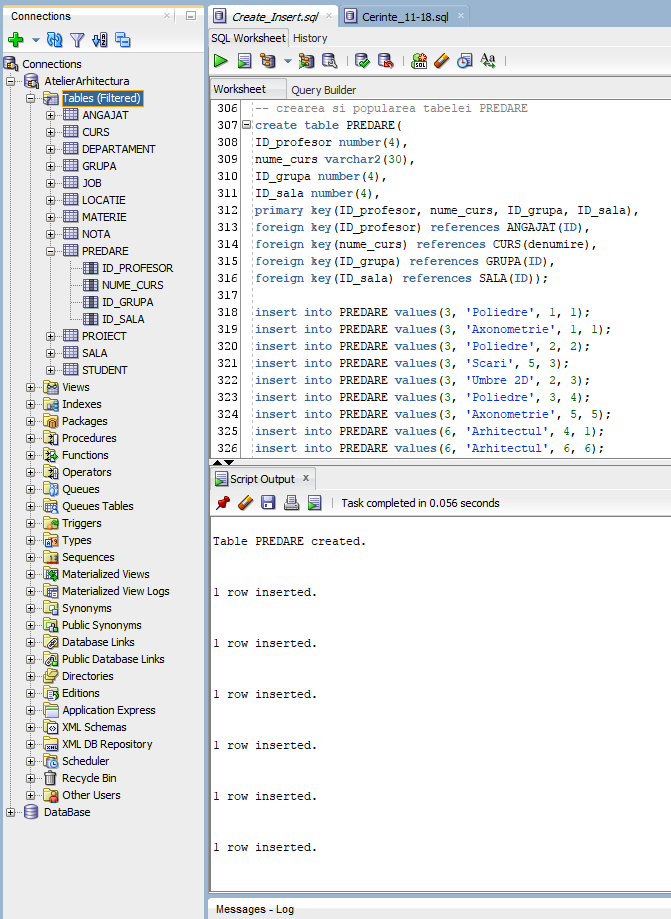
1. SALA



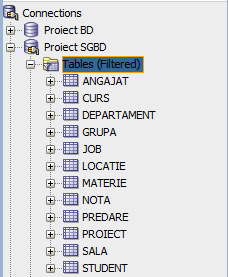
1. NOTA



1. PREDARE



1. 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;
14. **ALTER** SESSION **SET** NLS\_DATE\_FORMAT = 'DD-MM-YYYY';

17. -- crearea si popularea tabelei JOB
18. **create** **table** JOB(
19. ID number(4), **primary** **key**(ID),
20. denumire varchar2(15),
21. salariu number(6));
23. **create** **sequence** IDs
24. start **with** 1
25. increment **by** 1
26. minvalue 0
27. maxvalue 9999
28. nocycle;
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);
36. **drop** **sequence** IDs;
37. **commit**;

40. -- crearea si popularea tabelei DEPARTAMENT
41. **create** **table** DEPARTAMENT(
42. denumire varchar2(15), **primary** **key**(denumire),
43. ID\_director number(4),
44. tip varchar2(10));
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');
52. -- crearea si popularea tabelei 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));
62. **create** **sequence** IDs
63. start **with** 1
64. increment **by** 1
65. minvalue 0
66. maxvalue 999999
67. nocycle;
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);
75. **drop** **sequence** IDs;
76. **commit**;

79. -- crearea si popularea tabelei 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));
92. **create** **sequence** IDs
93. start **with** 1
94. increment **by** 1
95. minvalue 0
96. maxvalue 9999
97. nocycle;
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);
107. **drop** **sequence** IDs;
108. **commit**;
110. -- crearea si popularea tabelei GRUPA
111. **create** **table** GRUPA(
112. ID number(4), **primary** **key**(ID),
113. an\_studiu number(1),
114. specializare varchar2(15));
116. **create** **sequence** IDs
117. start **with** 1
118. increment **by** 1
119. minvalue 0
120. maxvalue 9999
121. nocycle;
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. **drop** **sequence** IDs;
130. **commit**;
132. -- crearea si popularea tabelei STUDENT
133. **create** **table** STUDENT(
134. ID number(6), **primary** **key**(ID),
135. nume varchar2(15),
136. prenume varchar2(15),
137. nr\_telefon varchar2(10),
138. email varchar2(30),
139. ID\_grupa number(4),
140. ID\_prof\_coordonator number(4),
141. data\_inrolare **date**,
142. an\_studiu number(1),
143. **foreign** **key**(ID\_grupa) **references** GRUPA(ID),
144. **foreign** **key**(ID\_prof\_coordonator) **references** ANGAJAT(ID));
146. **create** **sequence** IDs
147. start **with** 1
148. increment **by** 1
149. minvalue 0
150. maxvalue 999999
151. nocycle;
153. **insert** **into** STUDENT **values**(IDs.nextval, 'Giurgiuleanu', 'Matei', '0727835824', 'gmatei@yahoo.ro', 5, 3, '10-04-2019', 1);
154. **insert** **into** STUDENT **values**(IDs.nextval, 'Sotau', 'Marin', '0256207670', 'smarin@yahoo.ro', 1, 3, '27-04-2019', 3);
155. **insert** **into** STUDENT **values**(IDs.nextval, 'Robert', 'Ioan', '0235423932', 'rioan@yahoo.ro', 3, 6, '31-07-2020', 2);
156. **insert** **into** STUDENT **values**(IDs.nextval, 'Barbu', 'Matei', '0727333342', 'bmatei@yahoo.ro', 2, 3, '06-12-2020', 1);
157. **insert** **into** STUDENT **values**(IDs.nextval, 'Stefanescu', 'Bob', '0269559228', 'sbob@yahoo.ro', 2, 6, '28-09-2021', 1);
158. **insert** **into** STUDENT **values**(IDs.nextval, 'Paun', 'Cristinel', '0212520355', 'pcristinel@yahoo.ro', 6, 6, '06-11-2021', 3);
159. **insert** **into** STUDENT **values**(IDs.nextval, 'Bitulescu', 'Carol', '0214082820', 'bcarol@yahoo.ro', 1, 3, '20-08-2021', 3);
160. **insert** **into** STUDENT **values**(IDs.nextval, 'Popos', 'Florentin', '0740142399', 'pflorentin@yahoo.ro', 4, 3, '13-02-2021', 1);
161. **insert** **into** STUDENT **values**(IDs.nextval, 'Dobre', 'Marcel', '0721328241', 'dmarcel@yahoo.ro', 4, 3, '24-03-2020', 2);
162. **insert** **into** STUDENT **values**(IDs.nextval, 'Rosevilici', 'Teodor', '0744555788', 'rteodor@yahoo.ro', 2, 6, '19-07-2019', 1);
164. **drop** **sequence** IDs;
165. **commit**;

168. -- crearea si popularea tabelei MATERIE
169. **create** **table** MATERIE(
170. ID number(4), **primary** **key**(ID),
171. denumire varchar2(30),
172. numar\_cursuri number(2));
174. **create** **sequence** IDs
175. start **with** 1
176. increment **by** 1
177. minvalue 0
178. maxvalue 9999
179. nocycle;
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);
188. **drop** **sequence** IDs;
189. **commit**;

192. -- crearea si popularea tabelei CURS
193. **create** **table** CURS(
194. denumire varchar2(30), **primary** **key**(denumire),
195. numar\_ore number(1),
196. ID\_materie number(4),
197. credite number(1),
198. **foreign** **key**(ID\_materie) **references** MATERIE(ID));
200. **insert** **into** CURS **values**('Cunoasterea Spatiului', 2, 1, 2);
201. **insert** **into** CURS **values**('Diagrame', 2, 1, 3);
202. **insert** **into** CURS **values**('Eseu Grafic', 4, 1, 2);
203. **insert** **into** CURS **values**('Substractie si Aditie', 2, 1, 3);
204. **insert** **into** CURS **values**('Elemente Teoretice', 1, 2, 2);
205. **insert** **into** CURS **values**('Orasul Antic', 1, 2, 2);
206. **insert** **into** CURS **values**('Orasul Medieval', 1, 2, 2);
207. **insert** **into** CURS **values**('Orasul Industrial', 1, 2, 2);
208. **insert** **into** CURS **values**('Umbre 2D', 2, 3, 4);
209. **insert** **into** CURS **values**('Axonometrie', 4, 3, 6);
210. **insert** **into** CURS **values**('Poliedre', 2, 3, 4);
211. **insert** **into** CURS **values**('Scari', 6, 3, 5);
212. **insert** **into** CURS **values**('Vernacular', 2, 4, 3);
213. **insert** **into** CURS **values**('Recuperarea Orasului', 1, 4, 2);
214. **insert** **into** CURS **values**('Arhitectul', 1, 4, 2);
215. **insert** **into** CURS **values**('Orasul Traditional', 1, 4, 2);
216. **insert** **into** CURS **values**('Sisteme de Proiectie', 4, 5, 5);
217. **insert** **into** CURS **values**('Perspectiva la Fuga', 2, 5, 4);
218. **insert** **into** CURS **values**('Trasarea Umbrelor', 4, 5, 6);
219. **insert** **into** CURS **values**('Perspectiva la Calculator', 2, 5, 4);
220. **insert** **into** CURS **values**('Past Tenses', 1, 6, 2);
221. **insert** **into** CURS **values**('Architectural Vocabulary', 2, 6, 3);
222. **insert** **into** CURS **values**('Useful Phrases', 2, 6, 2);
224. **commit**;
226. -- crearea si popularea tabelei LOCATIE
227. **create** **table** LOCATIE(
228. ID number(2), **primary** **key**(ID),
229. adresa varchar2(60),
230. suprafata number(6));
232. **create** **sequence** IDs
233. start **with** 1
234. increment **by** 1
235. minvalue 0
236. maxvalue 99
237. nocycle;
239. **insert** **into** LOCATIE **values**(IDs.nextval, 'str. Belindo, nr. 6', 12500);
240. **insert** **into** LOCATIE **values**(IDs.nextval, 'str. Aurie, nr. 7A', 1500);
241. **insert** **into** LOCATIE **values**(IDs.nextval, 'str. Teilor, nr. 4', 20300);
242. **insert** **into** LOCATIE **values**(IDs.nextval, 'str. Suspendata, nr. 2B', 780);
243. **insert** **into** LOCATIE **values**(IDs.nextval, 'str. Nikolas Vinz, nr. 11', 321400);
245. **drop** **sequence** IDs;
246. **commit**;

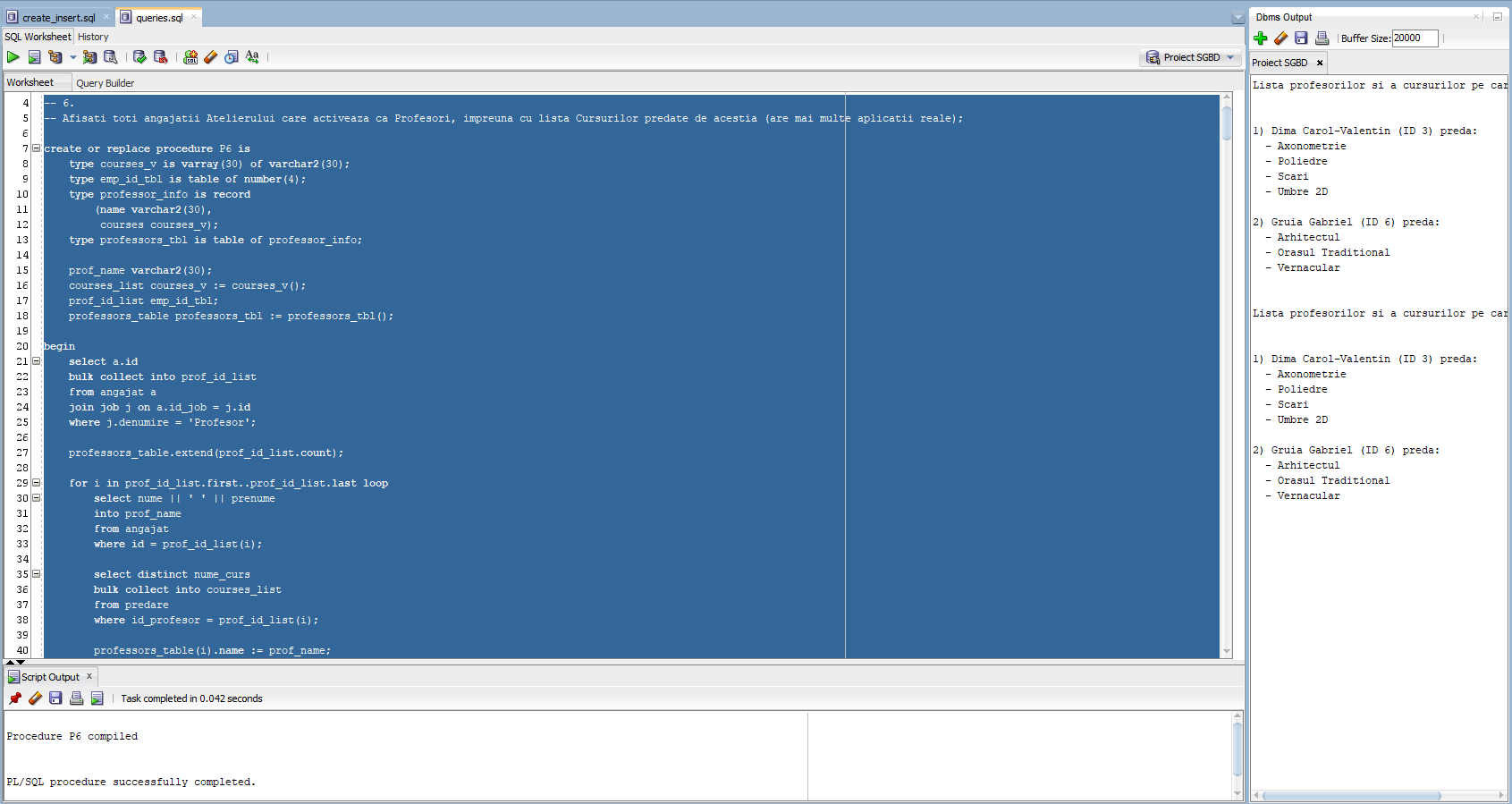
249. -- crearea si popularea tabelei SALA
250. **create** **table** SALA(
251. ID number(4), **primary** **key**(ID),
252. capacitate number(2),
253. ID\_locatie number(2),
254. **foreign** **key**(ID\_locatie) **references** LOCATIE(ID));
256. **create** **sequence** IDs
257. start **with** 1
258. increment **by** 1
259. minvalue 0
260. maxvalue 9999
261. nocycle;
263. **insert** **into** SALA **values**(IDs.nextval, 18, 1);
264. **insert** **into** SALA **values**(IDs.nextval, 10, 2);
265. **insert** **into** SALA **values**(IDs.nextval, 28, 4);
266. **insert** **into** SALA **values**(IDs.nextval, 70, 3);
267. **insert** **into** SALA **values**(IDs.nextval, 20, 5);
268. **insert** **into** SALA **values**(IDs.nextval, 80, 1);
269. **insert** **into** SALA **values**(IDs.nextval, 60, 4);
271. **drop** **sequence** IDs;
272. **commit**;

275. -- crearea si popularea tabelei NOTA
276. **create** **table** NOTA(
277. ID\_student number(6),
278. denumire\_curs varchar2(30),
279. **primary** **key**(ID\_student, denumire\_curs),
280. nota **float**(8),
281. data\_obtinerii **date**,
282. **foreign** **key**(ID\_student) **references** STUDENT(ID),
283. **foreign** **key**(denumire\_curs) **references** CURS(denumire));
285. **insert** **into** NOTA **values**(1, 'Scari', 7.50, '12-06-2020');
286. **insert** **into** NOTA **values**(1, 'Axonometrie', 8.50, '13-07-2020');
287. **insert** **into** NOTA **values**(2, 'Poliedre', 9.50, '11-05-2019');
288. **insert** **into** NOTA **values**(2, 'Axonometrie', 9, '13-07-2019');
289. **insert** **into** NOTA **values**(3, 'Poliedre', 7, '13-07-2020');
290. **insert** **into** NOTA **values**(4, 'Poliedre', 6, '12-06-2020');
291. **insert** **into** NOTA **values**(4, 'Umbre 2D', 8.50, '11-05-2019');
292. **insert** **into** NOTA **values**(4, 'Orasul Traditional', 4.30, '11-05-2019');
293. **insert** **into** NOTA **values**(4, 'Arhitectul', 8.10, '13-07-2020');
294. **insert** **into** NOTA **values**(5, 'Poliedre', 7.20, '11-05-2019');
295. **insert** **into** NOTA **values**(5, 'Umbre 2D', 8.90, '12-06-2020');
296. **insert** **into** NOTA **values**(5, 'Orasul Traditional', 8.50, '11-05-2019');
297. **insert** **into** NOTA **values**(5, 'Arhitectul', 10, '13-07-2020');
298. **insert** **into** NOTA **values**(6, 'Arhitectul', 9, '11-05-2019');
299. **insert** **into** NOTA **values**(6, 'Vernacular', 6.50, '13-07-2020');
300. **insert** **into** NOTA **values**(7, 'Poliedre', 8.50, '11-05-2019');
301. **insert** **into** NOTA **values**(7, 'Axonometrie', 7.10, '12-06-2020');
302. **insert** **into** NOTA **values**(8, 'Arhitectul', 3.20, '12-06-2020');
303. **insert** **into** NOTA **values**(9, 'Arhitectul', 9.10, '11-05-2019');
304. **insert** **into** NOTA **values**(10, 'Poliedre', 5, '13-07-2020');
305. **insert** **into** NOTA **values**(10, 'Umbre 2D', 5, '11-05-2019');
306. **insert** **into** NOTA **values**(10, 'Orasul Traditional', 4, '12-06-2020');
307. **insert** **into** NOTA **values**(10, 'Arhitectul', 6, '13-07-2020');
309. **commit**;
311. -- crearea si popularea tabelei PREDARE
312. **create** **table** PREDARE(
313. ID\_profesor number(4),
314. nume\_curs varchar2(30),
315. ID\_grupa number(4),
316. ID\_sala number(4),
317. **primary** **key**(ID\_profesor, nume\_curs, ID\_grupa, ID\_sala),
318. **foreign** **key**(ID\_profesor) **references** ANGAJAT(ID),
319. **foreign** **key**(nume\_curs) **references** CURS(denumire),
320. **foreign** **key**(ID\_grupa) **references** GRUPA(ID),
321. **foreign** **key**(ID\_sala) **references** SALA(ID));
323. **insert** **into** PREDARE **values**(3, 'Poliedre', 1, 1);
324. **insert** **into** PREDARE **values**(3, 'Axonometrie', 1, 1);
325. **insert** **into** PREDARE **values**(3, 'Poliedre', 2, 2);
326. **insert** **into** PREDARE **values**(3, 'Scari', 5, 3);
327. **insert** **into** PREDARE **values**(3, 'Umbre 2D', 2, 3);
328. **insert** **into** PREDARE **values**(3, 'Poliedre', 3, 4);
329. **insert** **into** PREDARE **values**(3, 'Axonometrie', 5, 5);
330. **insert** **into** PREDARE **values**(6, 'Arhitectul', 4, 1);
331. **insert** **into** PREDARE **values**(6, 'Arhitectul', 6, 6);
332. **insert** **into** PREDARE **values**(6, 'Vernacular', 6, 3);
333. **insert** **into** PREDARE **values**(6, 'Orasul Traditional', 2, 6);
334. **insert** **into** PREDARE **values**(6, 'Arhitectul', 2, 2);
336. **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.

“Afisati toti angajatii Atelierului care activeaza ca Profesori, impreuna cu lista Cursurilor predate de acestia (are mai multe aplicatii reale)”.

Retinem intr-un nested table, ID-urile angajatilor care activaeza ca profesori in cadrul atelierului apoi, pentru fiecare dintre acestia, obtinem si afisam lista cursurilor (din tabela “PREDARE”) pe care acestia le-au sustinut.



1. -- 6.
2. -- Afisati toti angajatii Atelierului care activeaza ca Profesori, impreuna cu lista Cursurilor predate de acestia (are mai multe aplicatii reale);
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;
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();
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';
24. professors\_table.extend(prof\_id\_list.count);
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);
32. **select** **distinct** nume\_curs
33. bulk collect **into** courses\_list
34. **from** predare
35. **where** id\_profesor = prof\_id\_list(i);
37. professors\_table(i).**name** := prof\_name;
38. professors\_table(i).courses := courses\_list;
39. **end** loop;
41. dbms\_output.put\_line('Lista profesorilor si a cursurilor pe care le sustin:');
42. dbms\_output.new\_line;
43. dbms\_output.new\_line;
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. /

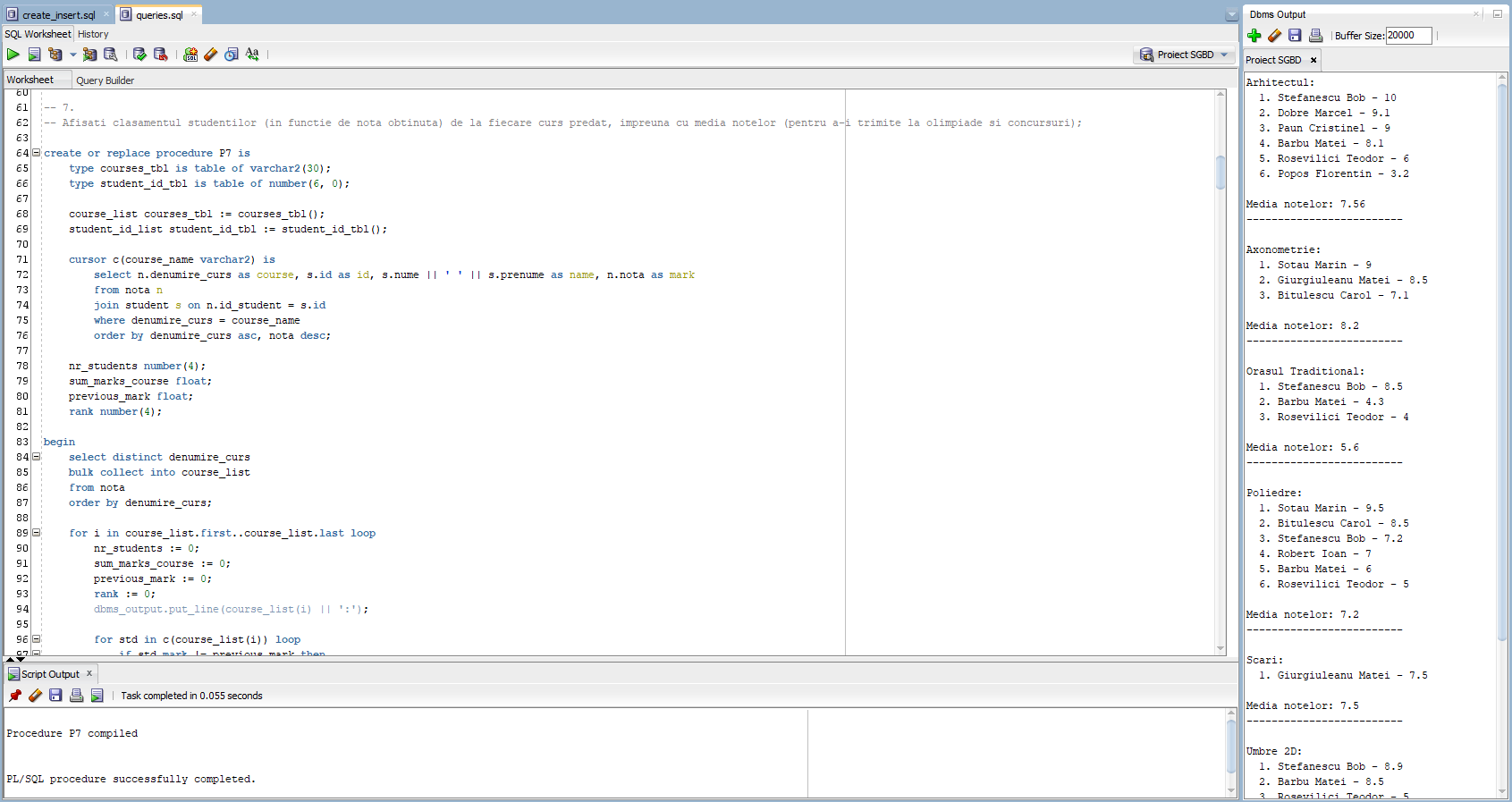
## 7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

“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.



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);
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);
8. course\_list courses\_tbl := courses\_tbl();
9. student\_id\_list student\_id\_tbl := student\_id\_tbl();
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**;
18. nr\_students number(4);
19. sum\_marks\_course **float**;
20. previous\_mark **float**;
21. rank number(4);
23. **begin**
24. **select** **distinct** denumire\_curs
25. bulk collect **into** course\_list
26. **from** nota
27. **order** **by** denumire\_curs;
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) || ':');
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.

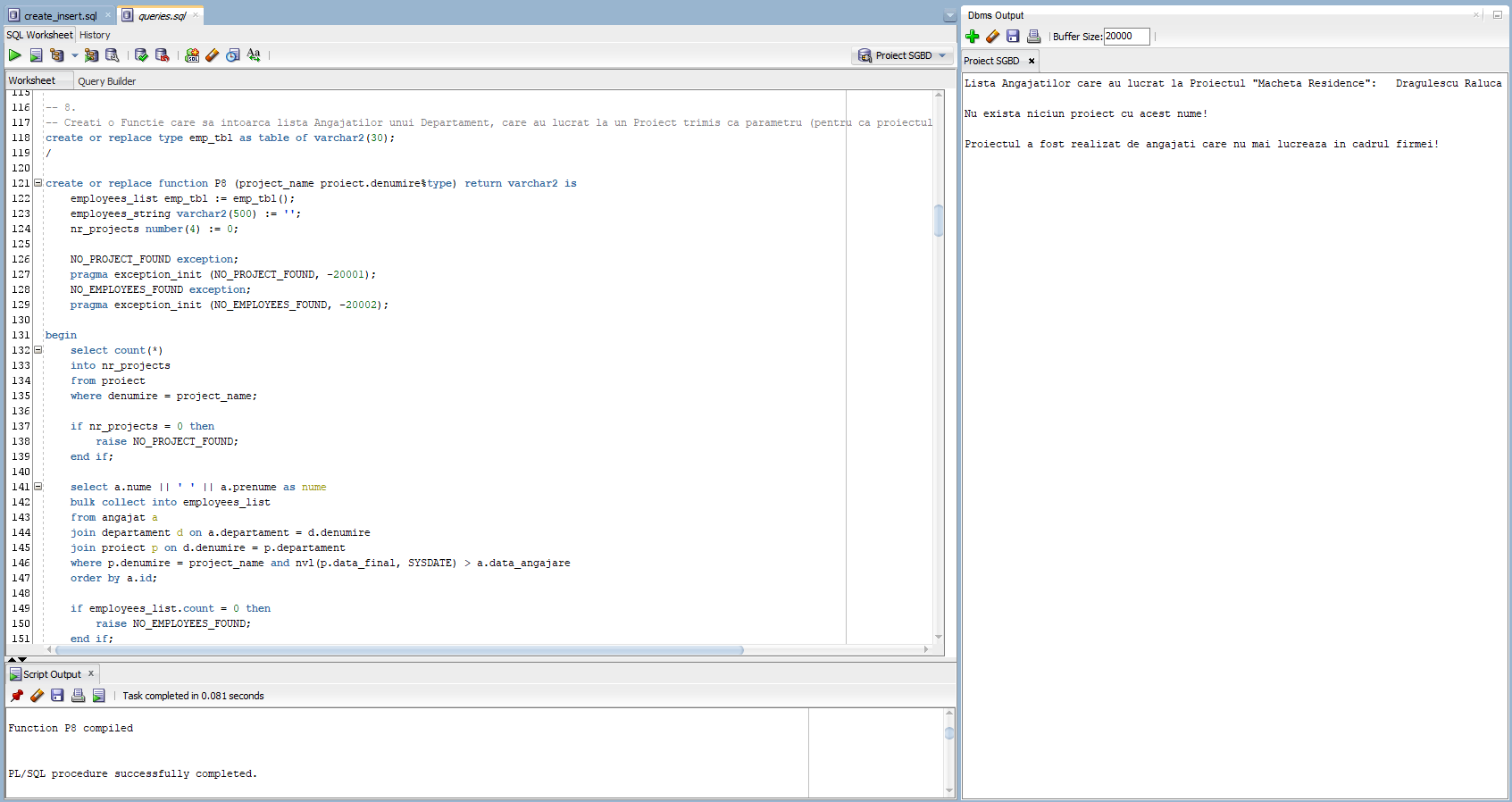
“Creati o Functie care sa intoarca lista Angajatilor unui Departament, care au lucrat la un Proiect trimis ca parametru (pentru ca proiectul a generat o suma mare de bani)”.

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.

Functia primeste 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.



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. /
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;
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);
16. **begin**
17. **select** count(\*)
18. **into** nr\_projects
19. **from** proiect
20. **where** denumire = project\_name;
22. if nr\_projects = 0 **then**
23. raise NO\_PROJECT\_FOUND;
24. **end** if;
26. **select** a.nume || ' ' || 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;
34. if employees\_list.count = 0 **then**
35. raise NO\_EMPLOYEES\_FOUND;
36. **end** if;
38. **for** i in employees\_list.**first**..employees\_list.**last** loop
39. employees\_string := employees\_string || employees\_list(i) || ', ';
40. **end** loop;
42. **return** rtrim(employees\_string, ', ');
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.

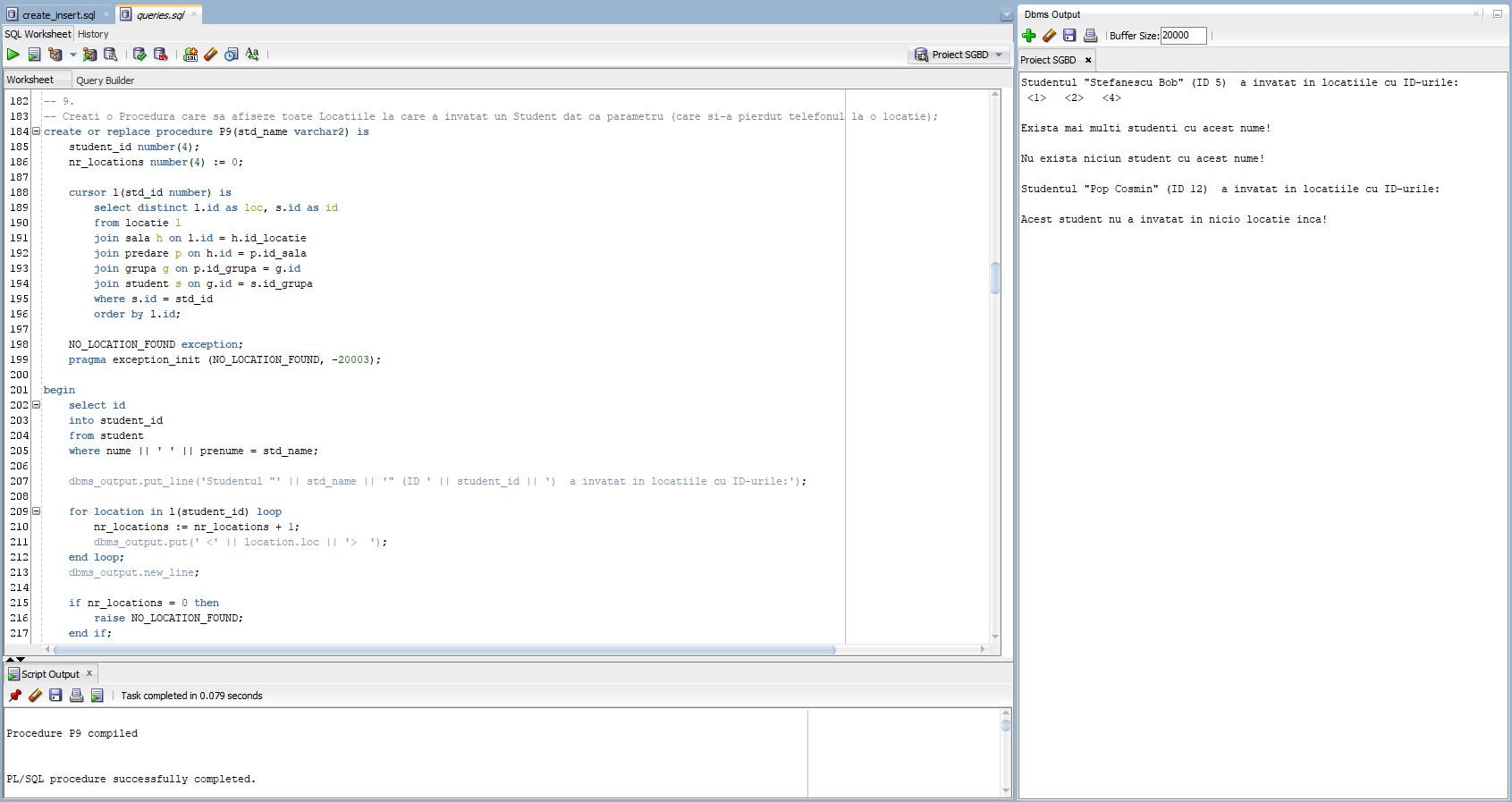
“Creati o Procedura care sa afiseze toate Locatiile la care a invatat un Student dat ca parametru (care si-a pierdut telefonul la o locatie).”

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.

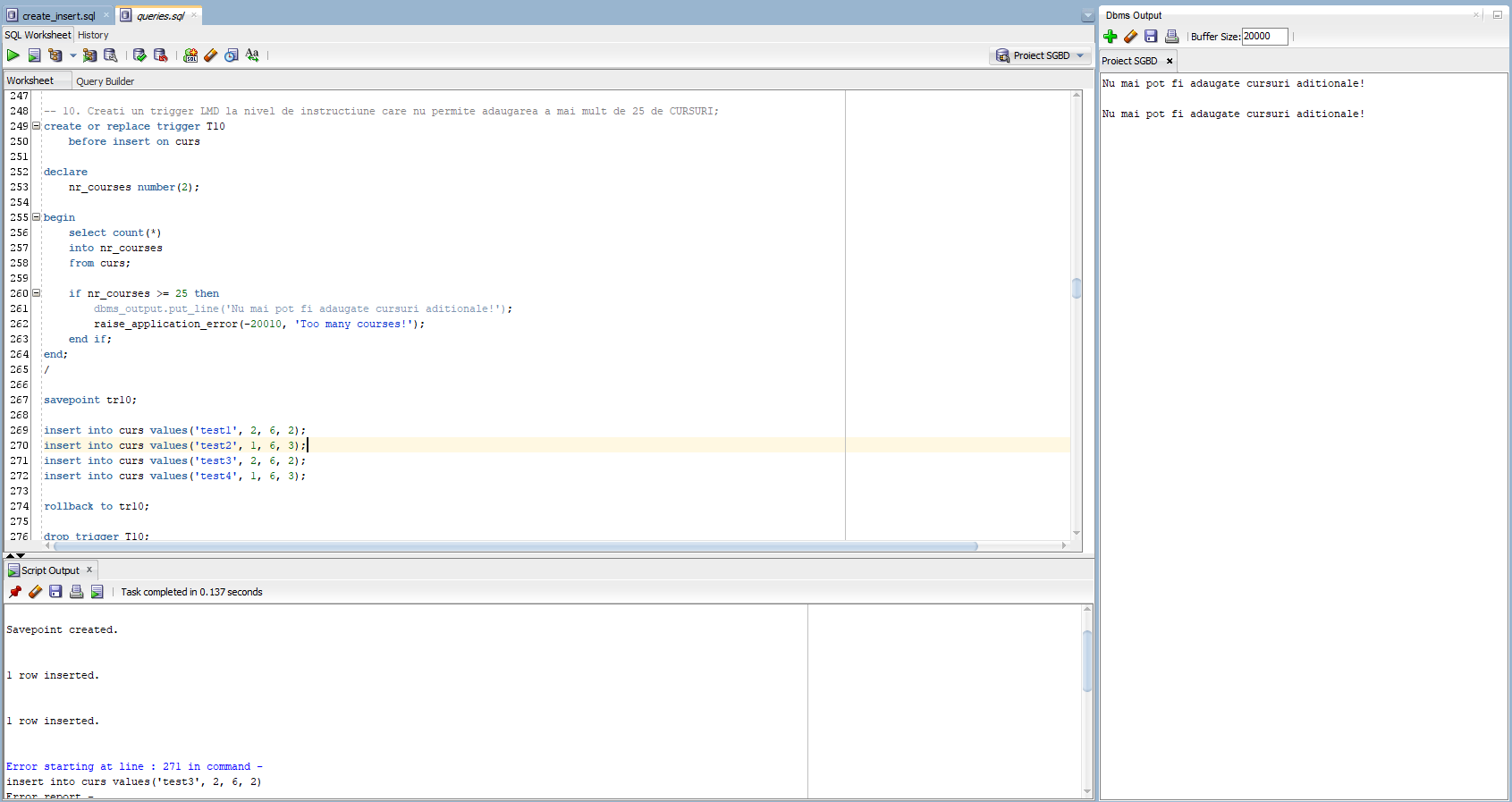
EXCEPTIILE tratate:

* 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



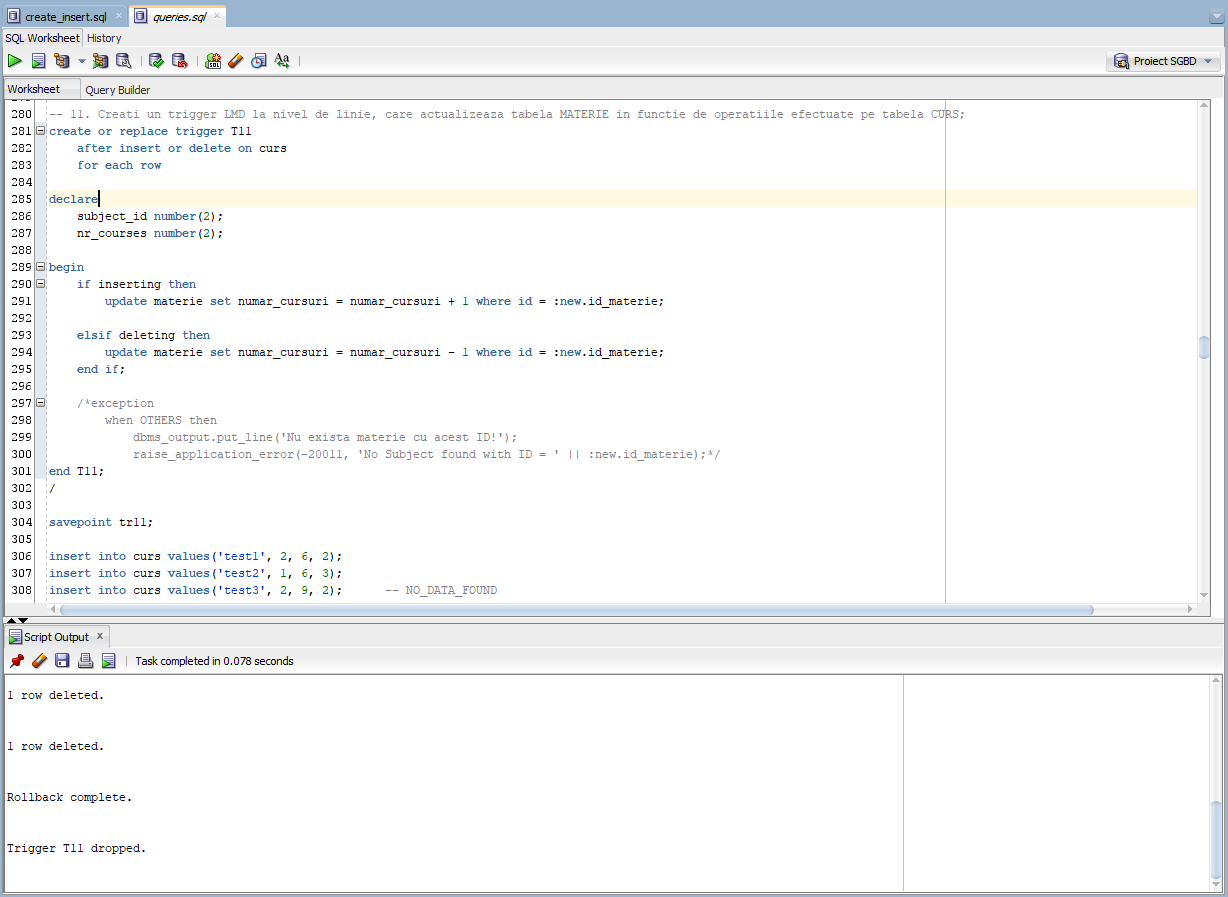
1. – 9.
2. -- Creati o Procedura care sa afiseze toate Locatiile la care a invatat un Student dat ca parametru (care si-a pierdut telefonul la o locatie);
3. **create** or replace **procedure** P9(std\_name varchar2) **is**
4. student\_id number(4);
5. nr\_locations number(4) := 0;
7. **cursor** l(std\_id 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;
17. NO\_LOCATION\_FOUND exception;
18. pragma exception\_init (NO\_LOCATION\_FOUND, -20003);
20. **begin**
21. **select** id
22. **into** student\_id
23. **from** student
24. **where** nume || ' ' || prenume = std\_name;
26. dbms\_output.put\_line('Studentul "' || std\_name || '" (ID ' || student\_id || ')  a invatat in locatiile cu ID-urile:');
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;
34. if nr\_locations = 0 **then**
35. raise NO\_LOCATION\_FOUND;
36. **end** if;
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. /
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.

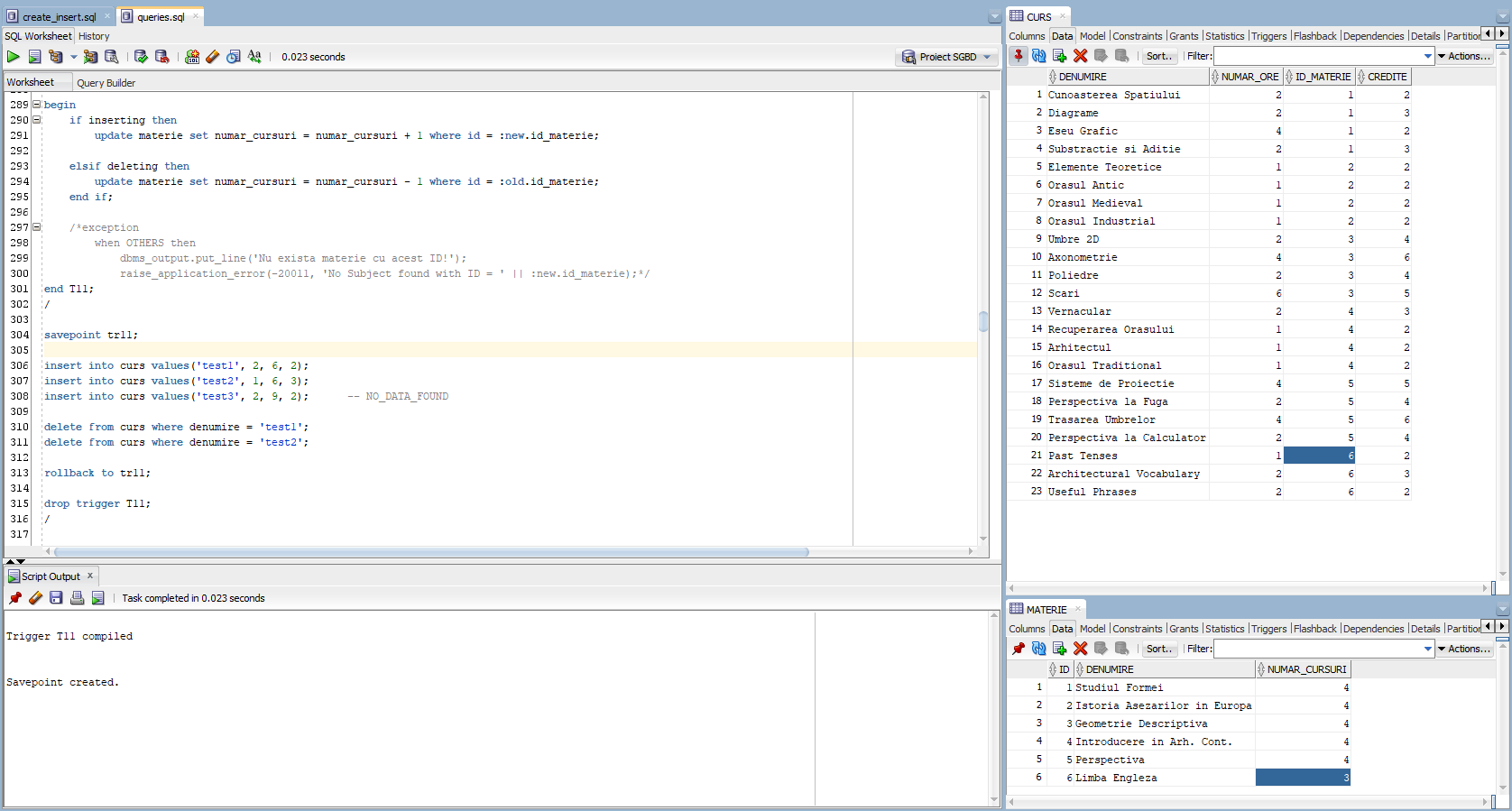


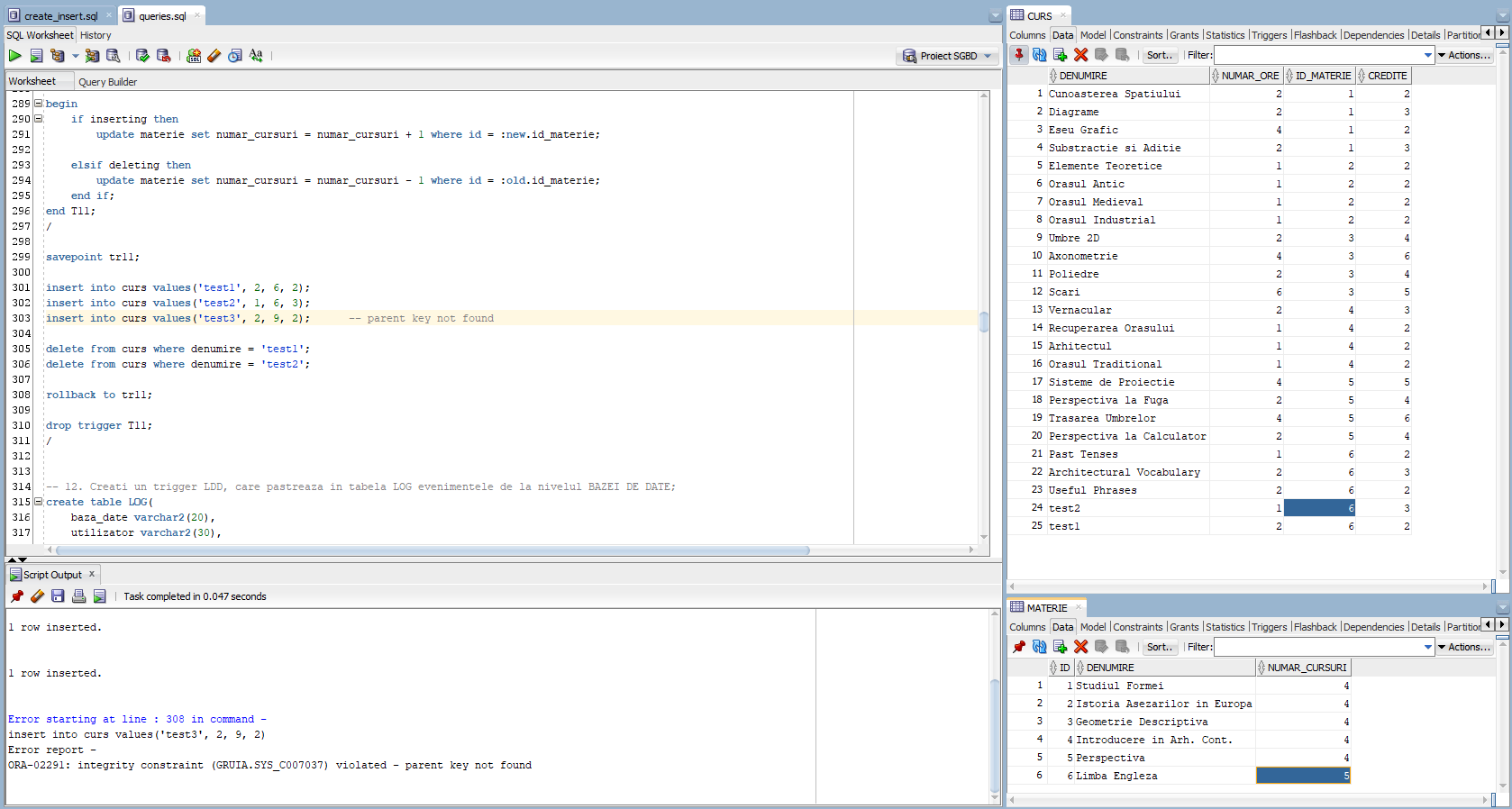
1. -- 10. Creati un trigger LMD la nivel de instructiune care nu permite adaugarea a mai mult de 25 de CURSURI;
2. **create** or replace **trigger** T10
3. before **insert** **on** curs
5. **declare**
6. nr\_courses number(2);
8. **begin**
9. **select** count(\*)
10. **into** nr\_courses
11. **from** curs;
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. /
20. savepoint tr10;
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);
27. **rollback** **to** tr10;
29. **drop** **trigger** T10;
30. /

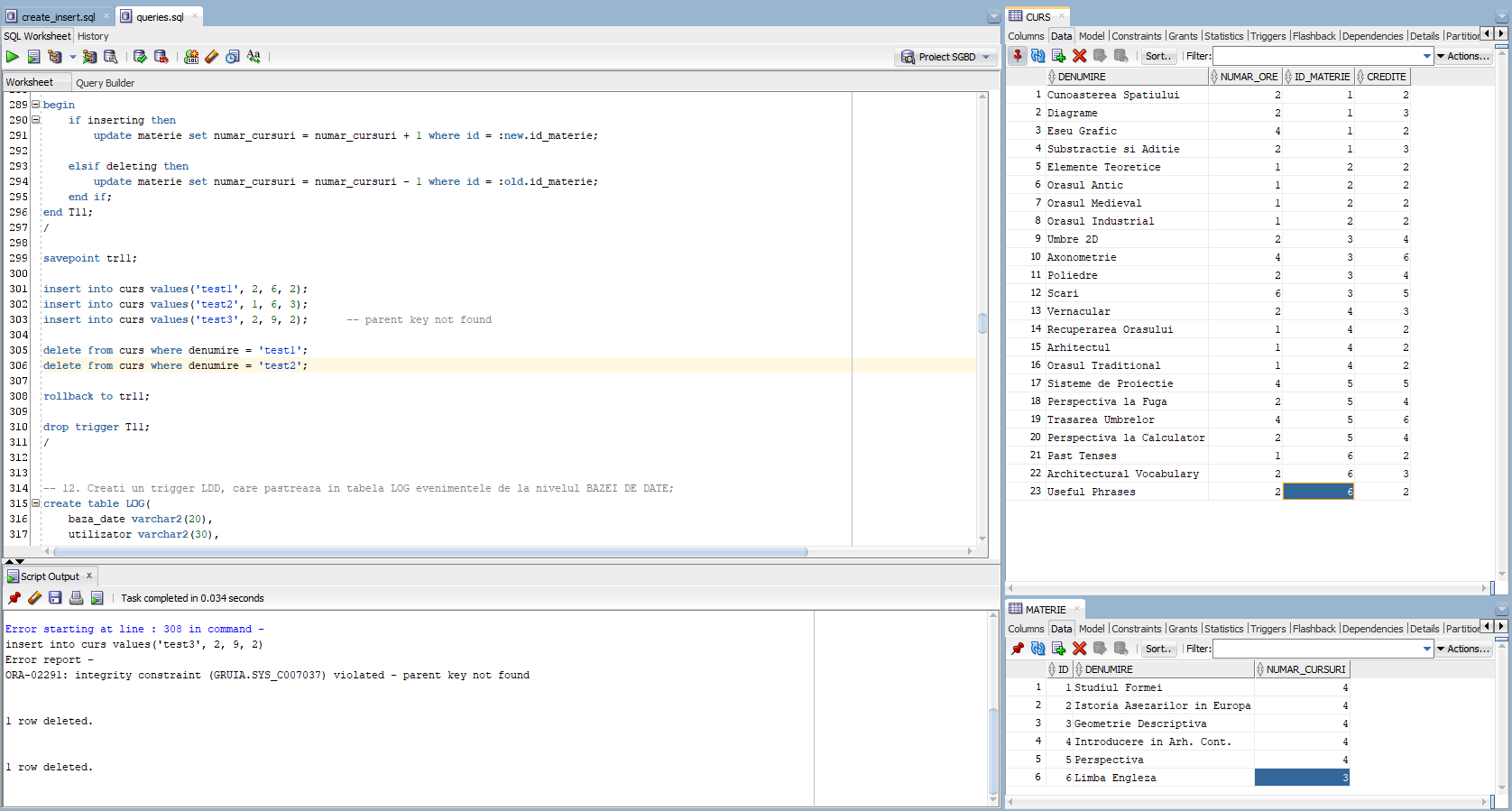
## 11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.



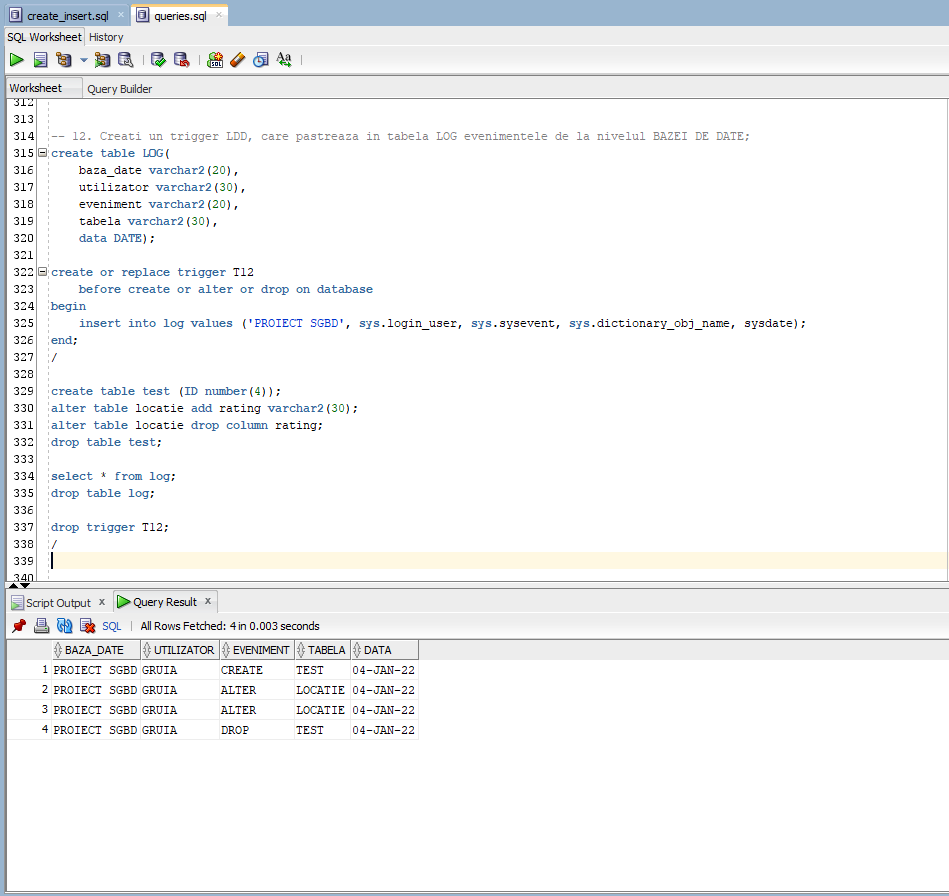
1. -- 11. Creati un trigger LMD la nivel de linie, care actualizeaza tabela MATERIE in functie de operatiile efectuate pe tabela CURS;
2. **create** or replace **trigger** T11
3. **after** **insert** or **delete** **on** curs
4. **for** each row
6. **declare**
7. subject\_id number(2);
8. nr\_courses number(2);
10. **begin**
11. if inserting **then**
12. **update** materie **set** numar\_cursuri = numar\_cursuri + 1 **where** id = :new.id\_materie;
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. /
20. savepoint tr11;
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
26. **delete** **from** curs **where** denumire = 'test1';
27. **delete** **from** curs **where** denumire = 'test2';
29. **rollback** **to** tr11;
31. **drop** **trigger** T11;
32. /





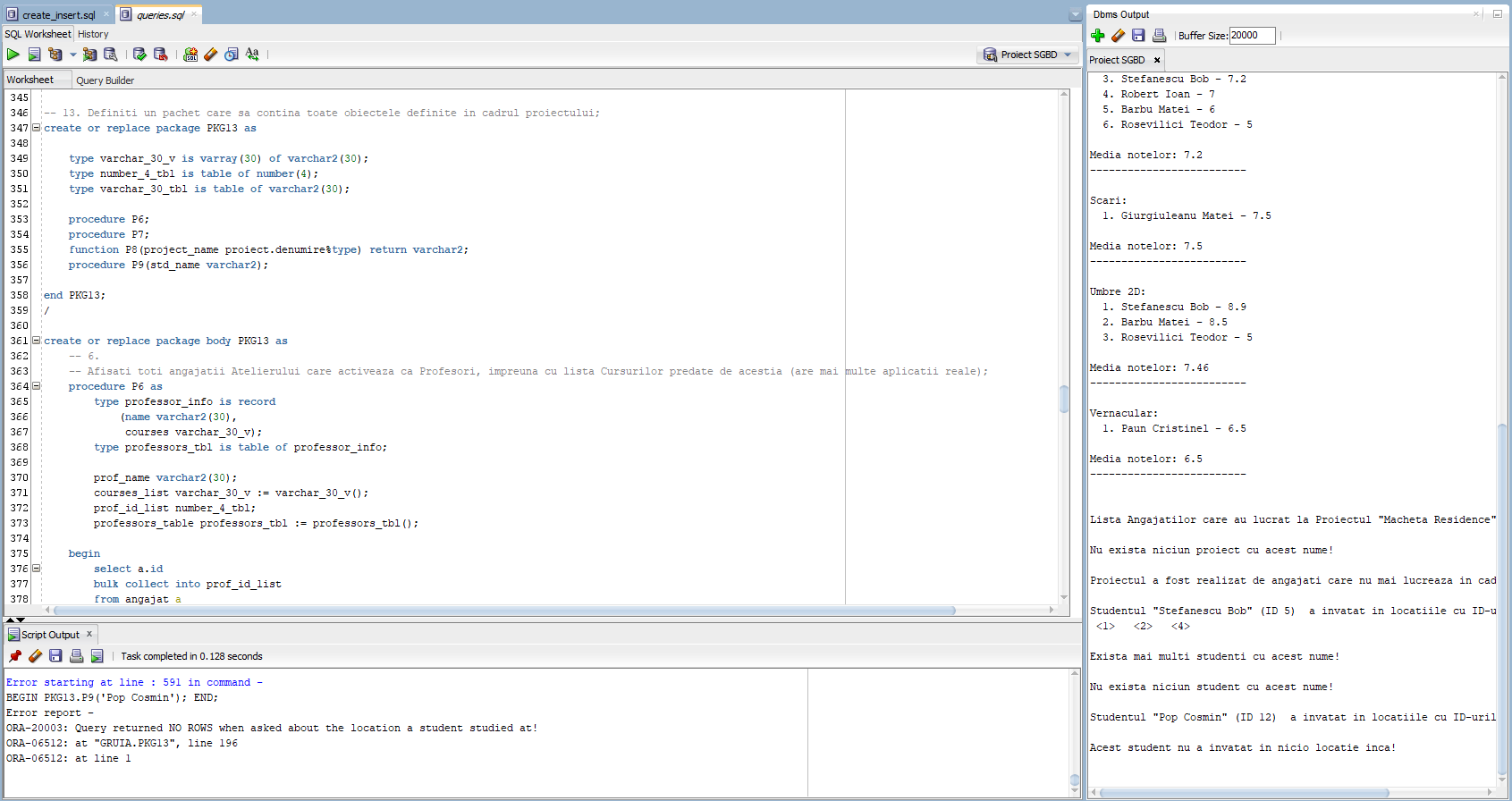


## 12. Definiți un trigger de tip LDD. Declanșați trigger-ul.



1. -- 12. Creati 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**);
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. /
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;
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**
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);
8. **procedure** P6;
9. **procedure** P7;
10. **function** P8(project\_name proiect.denumire%type) **return** varchar2;
11. **procedure** P9(std\_name varchar2);
13. **end** PKG13;
14. /
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;
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();
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';
37. professors\_table.extend(prof\_id\_list.count);
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);
45. **select** **distinct** nume\_curs
46. bulk collect **into** courses\_list
47. **from** predare
48. **where** id\_profesor = prof\_id\_list(i);
50. professors\_table(i).**name** := prof\_name;
51. professors\_table(i).courses := courses\_list;
52. **end** loop;
54. dbms\_output.put\_line('Lista profesorilor si a cursurilor pe care le sustin:');
55. dbms\_output.new\_line;
56. dbms\_output.new\_line;
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) || ') preda: ');
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;

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();
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**;
80. nr\_students number(4);
81. sum\_marks\_course **float**;
82. previous\_mark **float**;
83. rank number(4);
85. **begin**
86. **select** **distinct** denumire\_curs
87. bulk collect **into** course\_list
88. **from** nota
89. **order** **by** denumire\_curs;
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) || ':');
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;

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;
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);
127. **begin**
128. **select** count(\*)
129. **into** nr\_projects
130. **from** proiect
131. **where** denumire = project\_name;
133. if nr\_projects = 0 **then**
134. raise NO\_PROJECT\_FOUND;
135. **end** if;
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) > a.data\_angajare
143. **order** **by** a.id;
145. if employees\_list.count = 0 **then**
146. raise NO\_EMPLOYEES\_FOUND;
147. **end** if;
149. **for** i in employees\_list.**first**..employees\_list.**last** loop
150. employees\_string := employees\_string || employees\_list(i) || ', ';
151. **end** loop;
153. **return** rtrim(employees\_string, ', ');
155. exception
156. **when** NO\_PROJECT\_FOUND **then**
157. dbms\_output.put\_line('Nu exista niciun proiect cu acest nume!');
158. raise\_application\_error(-20001, 'No Project found that matched the string passed as parameter to P8!');
159. **when** NO\_EMPLOYEES\_FOUND **then**
160. dbms\_output.put\_line('Proiectul a fost realizat de angajati care nu mai lucreaza in cadrul firmei!');
161. raise\_application\_error(-20002, 'No Employees found that were employed before the Project started!');
162. **end** P8;

165. -- 9.
166. -- 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);
167. **procedure** P9(std\_name varchar2) **as**
168. student\_id number(4);
169. nr\_locations number(4) := 0;
171. **cursor** l(std\_id number) **is**
172. **select** **distinct** l.id **as** loc, s.id **as** id
173. **from** locatie l
174. join sala h **on** l.id = h.id\_locatie
175. join predare p **on** h.id = p.id\_sala
176. join grupa g **on** p.id\_grupa = g.id
177. join student s **on** g.id = s.id\_grupa
178. **where** s.id = std\_id
179. **order** **by** l.id;
181. NO\_LOCATION\_FOUND exception;
182. pragma exception\_init (NO\_LOCATION\_FOUND, -20003);
184. **begin**
185. **select** id
186. **into** student\_id
187. **from** student
188. **where** nume || ' ' || prenume = std\_name;
190. dbms\_output.put\_line('Studentul "' || std\_name || '" (ID ' || student\_id || ')  a invatat in locatiile cu ID-urile:');
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;
198. if nr\_locations = 0 **then**
199. raise NO\_LOCATION\_FOUND;
200. **end** if;
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;
214. **end** PKG13;
215. /
217. -- TESTAM PACHETUL PKG13;
218. -- 6
219. **execute** PKG13.P6;
221. -- 7
222. **execute** PKG13.P7;
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. /
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 atentia acordata!**

Student: Gabriel GRUIA

Grupa: 231

Data: 05.01.2021