Proiect final la Sisteme de Gestiune a Bazelor de Date
Crearea si gestionarea unei companii de închirieri de locuințe

Nicolae Ducal
Grupa 234

Decembrie 31, 2020

Cuprins

Introducere. Diagramele bazei de date

1.1 Prezentarea bazei de date (Cerința nr. 1)

Pentru acest proiect am pregătit baza de date a unei companii de închirieri de locuințe. Compania are mai multe sedii, atât în orașe mari ale României, cât și în alte orașe europene. Angajații care lucrează în aceste sedii pot lucra pe joburi, cum ar fi: director, secretar, consultant, agent imobiliar. Fiecare sediu are câte un director, care poate conduce doar acel sediu. Agenții imobiliari sunt angajații care se ocupa de rezervări. Clienții companiei pot rezerva două tipuri de imobil: locuințe și camere de hotel. Fiecare locuință are un proprietar, care a fost dispus să ofere locuința pentru închirieri. Hotelurile dispun de mai multe camere, care pot fi date în chirie. Atât locuințele, cât și hotelurile se află în locații unice. Ambele tipuri de imobil dispun de diferite servicii, care au un anumit rating. Procesul de rezervare are loc loc în felul următor: clientul alege una din locuințe sau camere din hotel care sunt disponibile pentru perioada necesara lui pentru cazare (care poate fi orice perioada disponibila, după data curentă). Pentru aprobare, un agent imobiliar confirmă rezervarea. Pentru buna funcționare a bazei de date, rezervările care au expirat (în care clienții deja s-au decazat) se transferă în istoricul rezervărilor locuinței, respectiv a hotelului, pentru a scăpa de date în plus. Acest lucru se efectuează din motiv că tabelele Rezervărilor sunt accesate foarte des, fiind necesar ca acestea să funcționeze rapid, iar transferul datelor într-un istoric ar îmbunătăți constant acest lucru. Pentru o analogie cu marile companii contemporane de acest tip, pot menționa: Airbnb, Booking.com.

1.2 Diagrama Entitate-Relație a bazei de date (Cerința nr. 2)

Următorul lucru în efectuarea proiectului este definirea diagramei Entitate-Relație (ER) a bazei de date, în care identificăm entitățile principale și stabilim relații de diferite tipuri între ele (de tip Many-to-Many, One-To-Many etc.). Aceasta poate fi vizualizată pe următoarea pagină:

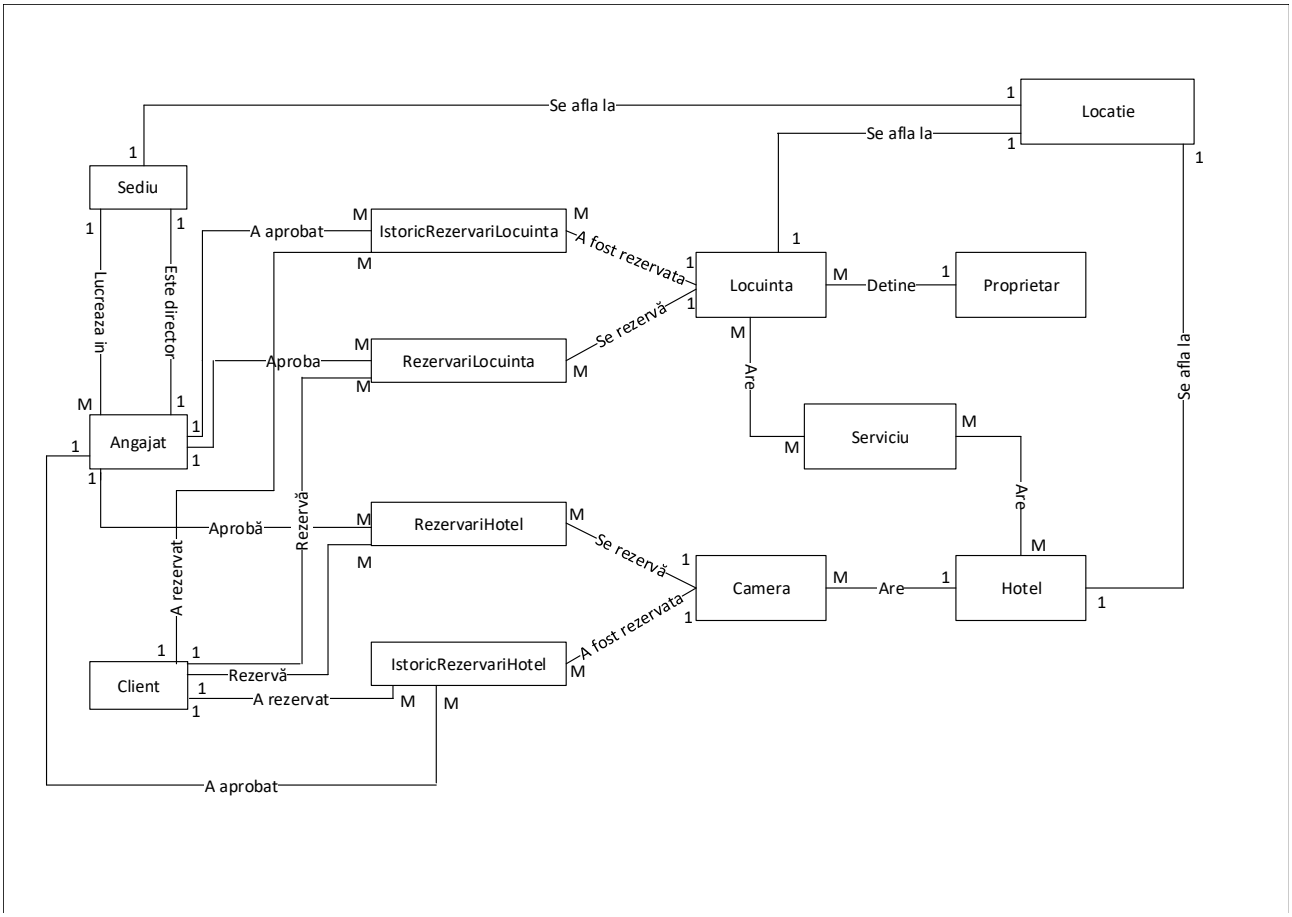

Figure 1.1: Diagrama Entitate-Relație a companiei

1.3 Diagrama conceptuala a bazei de date (Cerința nr. 3)

După efectuarea diagramei Entitate-Relație, putem trece la efectuarea diagramei conceptuale a bazei de date, în care adăugăm atributele necesare și includem și tabelele asociative. Această diagramă poate fi vizualizată pe următoarea pagină:

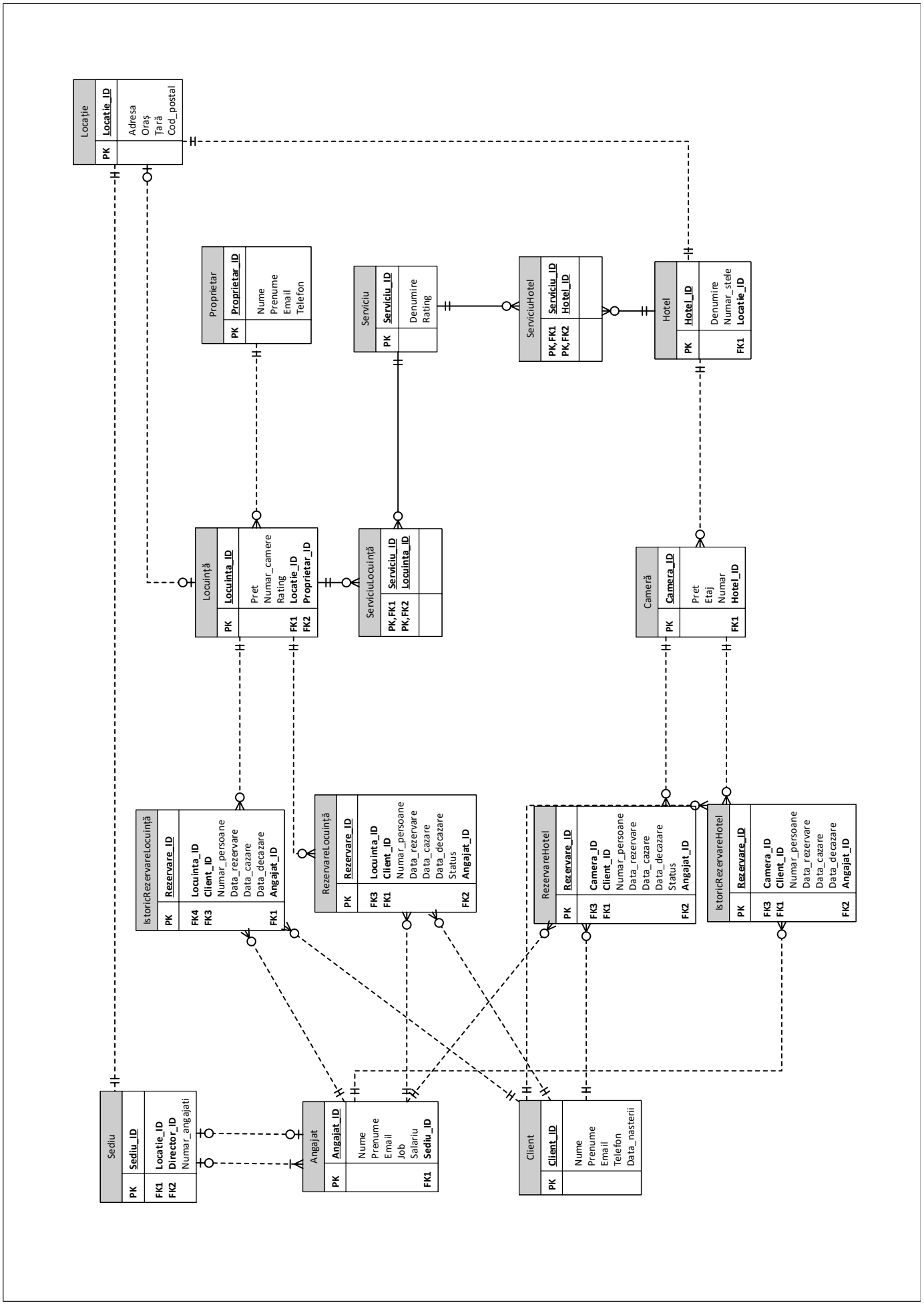

Figure 1.2: Diagrama conceptuală a companiei

Crearea tabelelor și inserarea datelor

2.1 Crearea tabelelor în Oracle (Cerința nr. 4)

Pentru următoarea cerință trebuie să creăm toate tabelele definite în diagrama conceptuală de mai sus. De asemenea, trebuie să definim toate constrângerile de integritate (cheile primare, cheile externe, verificări pe coloane, constrangerea de unicitate). Inițial, am creat o conexiune locala, cu ajutorul Oracle Express 11g. Voi prezenta mai departe comenzile utilizate pentru a crea baza de date. Pentru a avea acces mai bun la comenzi, puteti accesa acest link.

```
--Tabelul Proprietar
create table proprietar (
  proprietar_id number(8) primary key,
  nume varchar2(30) not null,
  prenume varchar2(30) not null,
  email varchar2(30) not null,
  telefon varchar2(15));

--Tabelul Client
--Va avea o constrangere unica pentru tuplul (nume, prenume, email, data_nasterii)
create table client (
  client_id number primary key,
  nume varchar2(30) not null,
  prenume varchar2(30) not null,
  email varchar2(30) not null,
  telefon varchar2(15),
  data_nasterii date not null,
  constraint unq_client unique (nume, prenume, email, data_nasterii));

--Tabelul Serviciu
--Constrangeri: Denumire unica, ratingul sa fie intre 1 si 10
create table serviciu (
  serviciu_id number primary key,
  denumire varchar2(50) not null,
  rating number(2),
  constraint unq_serviciu unique (denumire),
  constraint chk_serviciu check (1 <= rating and rating <= 10));
```

```
--Tabelul Locatie
--Va avea o constrangere unica pentru tuplul (adresa, oras, tara)
create table locatie (
  locatie_id number(5) primary key,
  adresa varchar2(80) not null,
  cod_postal number(10),
  oras varchar2(50) not null,
  tara varchar2(50) not null,
  constraint unq_locatie unique(adresa, oras, tara));

--Tabelul Hotel
--Constrangeri: Cheie externa - locatie_id, check ca numarul de stele sa fie
--intre 1 si 5
create table hotel (
  hotel_id number primary key,
  denumire varchar2(30) not null,
  numar_stele number(2) not null,
  locatie_id number(5),
  constraint fk1_hotel foreign key(locatie_id) references locatie(locatie_id)
      on delete set null,
  constraint chk_hotel check(1 <= numar_stele and numar_stele <= 5));

--Tabelul Locuinta
--Constrangeri: Cheie externa - locatie_id si ratingul e un numar zecimal
--intre 1 si 10
create table locuinta (
  locuinta_id number primary key,
  pret number(10) not null,
  numar_camere number(3) not null,
  rating number(3, 3) not null,
  locatie_id number(5),
  proprietar_id number not null,
  constraint fk1_locuinta foreign key(locatie_id) references
      locatie(locatie_id) on delete set null,
  constraint fk2_locuinta foreign key(proprietar_id) references
      proprietar(proprietar_id) on delete cascade);

--Tabelul Camera
--Constrangeri: Cheie externa - hotel_id,
--combinatie unica de (hotel_id, etaj, numar)
create table camera (
  camera_id number primary key,
  pret number(10) not null,
  etaj number(3) not null,
  numar number(5) not null,
  hotel_id number not null,
  constraint unq_camera unique(hotel_id, etaj, numar),
  constraint fk1_camera foreign key(hotel_id) references
      hotel(hotel_id) on delete cascade);
```

```
--Tabel asociativ Serviciu-Locuinta
create table serviciu_locuinta (
  serviciu_id number not null,
  locuinta_id number not null,
  constraint pk_serviciu_locuinta primary key(serviciu_id, locuinta_id),
  constraint fk1_serviciu_locuinta foreign key(serviciu_id)
      references serviciu(serviciu_id) on delete cascade,
  constraint fk2_serviciu_locuinta foreign key(locuinta_id)
      references locuinta(locuinta_id) on delete cascade);

--Tabel asociativ Serviciu-Hotel
create table serviciu_hotel (
  serviciu_id number not null,
  hotel_id number not null,
  constraint pk_serviciu_hotel primary key(serviciu_id, hotel_id),
  constraint fk1_serviciu_hotel foreign key(serviciu_id)
      references serviciu(serviciu_id) on delete cascade,
  constraint fk2_serviciu_hotel foreign key(hotel_id)
      references hotel(hotel_id) on delete cascade);

--Tabelul Sediu
--Constrangeri: Cheie externa - director_id - unica
--Constrangeri: Cheie externa - locatie_id
create table sediu (
  sediu_id number primary key,
  locatie_id number(5),
  director_id number,
  numar_angajati number,
  constraint fk1_sediu foreign key(locatie_id)
      references locatie(locatie_id) on delete set null,
  constraint unq_sediu unique(director_id));

--Tabelul Angajat
create table angajat (
  angajat_id number primary key,
  nume varchar2(30) not null,
  prenume varchar2(30) not null,
  email varchar2(30) not null,
  job varchar2(30) not null,
  salariu number,
  sediu_id number not null,
  constraint unq_angajat unique(nume, prenume, email),
  constraint fk1_angajat foreign key(sediu_id)
      references sediu(sediu_id) on delete cascade);

--Adaug constrangerea pentru director_id din sediu
alter table sediu
add constraint fk2_sediu foreign key(director_id)
    references angajat(angajat_id) on delete set null;
```

```
--Tabelul Rezervare_locuinta
 create table rezervare_locuinta (
   rezervare_id number primary key,
   locuinta_id number not null,
   client_id number not null,
   numar_persoane number not null,
   data_rezervare date not null,
   data_cazare date not null,
   data_decazare date not null,
   status varchar2(30),
   angajat_id number not null,
   constraint fk1_rezervare_locuinta foreign key(locuinta_id)
       references locuinta(locuinta_id) on delete cascade,
   constraint fk2_rezervare_locuinta foreign key(client_id)
       references client(client_id) on delete cascade,
   constraint fk3_rezervare_locuinta foreign key(angajat_id)
       references angajat(angajat_id) on delete cascade,
   constraint ck1_rezervare_locuinta
       check(data_rezervare <= data_cazare and data_cazare <= data_decazare));

--Tabelul Rezervare_Hotel
 create table rezervare_hotel (
   rezervare_id number primary key,
   camera_id number not null,
   client_id number not null,
   numar_persoane number not null,
   data_rezervare date not null,
   data_cazare date not null,
   data_decazare date not null,
   status varchar2(30),
   angajat_id number not null,
   constraint fk1_rezervare_hotel foreign key(camera_id)
       references camera(camera_id) on delete cascade,
   constraint fk2_rezervare_hotel foreign key(client_id)
       references client(client_id) on delete cascade,
   constraint fk3_rezervare_hotel foreign key(angajat_id)
       references angajat(angajat_id) on delete cascade,
   constraint ck1_rezervare_hotel
       check(data_rezervare <= data_cazare and data_cazare <= data_decazare));
```

```
--Tabelul Istoric_Rezervare_locuinta
create table istoric_rezervare_locuinta (
  rezervare_id number primary key,
  locuinta_id number not null,
  client_id number not null,
  numar_persoane number not null,
  data_rezervare date not null,
  data_cazare date not null,
  data_decazare date not null,
  angajat_id number not null,
  constraint fk1_istoric_rezervare_locuinta foreign key(locuinta_id)
      references locuinta(locuinta_id) on delete cascade,
  constraint fk2_istoric_rezervare_locuinta foreign key(client_id)
      references client(client_id) on delete cascade,
  constraint fk3_istoric_rezervare_locuinta foreign key(angajat_id)
      references angajat(angajat_id) on delete cascade,
  constraint ck1_istoric_rezervare_locuinta
      check(data_rezervare <= data_cazare and data_cazare <= data_decazare));

--Tabelul Istoric_Rezervare_Hotel
create table istoric_rezervare_hotel (
  rezervare_id number primary key,
  camera_id number not null,
  client_id number not null,
  numar_persoane number not null,
  data_rezervare date not null,
  data_cazare date not null,
  data_decazare date not null,
  angajat_id number not null,
  constraint fk1_istoric_rezervare_hotel foreign key(camera_id)
      references camera(camera_id) on delete cascade,
  constraint fk2_istoric_rezervare_hotel foreign key(client_id)
      references client(client_id) on delete cascade,
  constraint fk3_istoric_rezervare_hotel foreign key(angajat_id)
      references angajat(angajat_id) on delete cascade,
  constraint ck1_istoric_rezervare_hotel
      check(data_rezervare <= data_cazare and data_cazare <= data_decazare));
```

De asemenea, voi introduce câteva imagini care demonstrează corectitudinea definirii tabelelor anume prin rularea acestora:

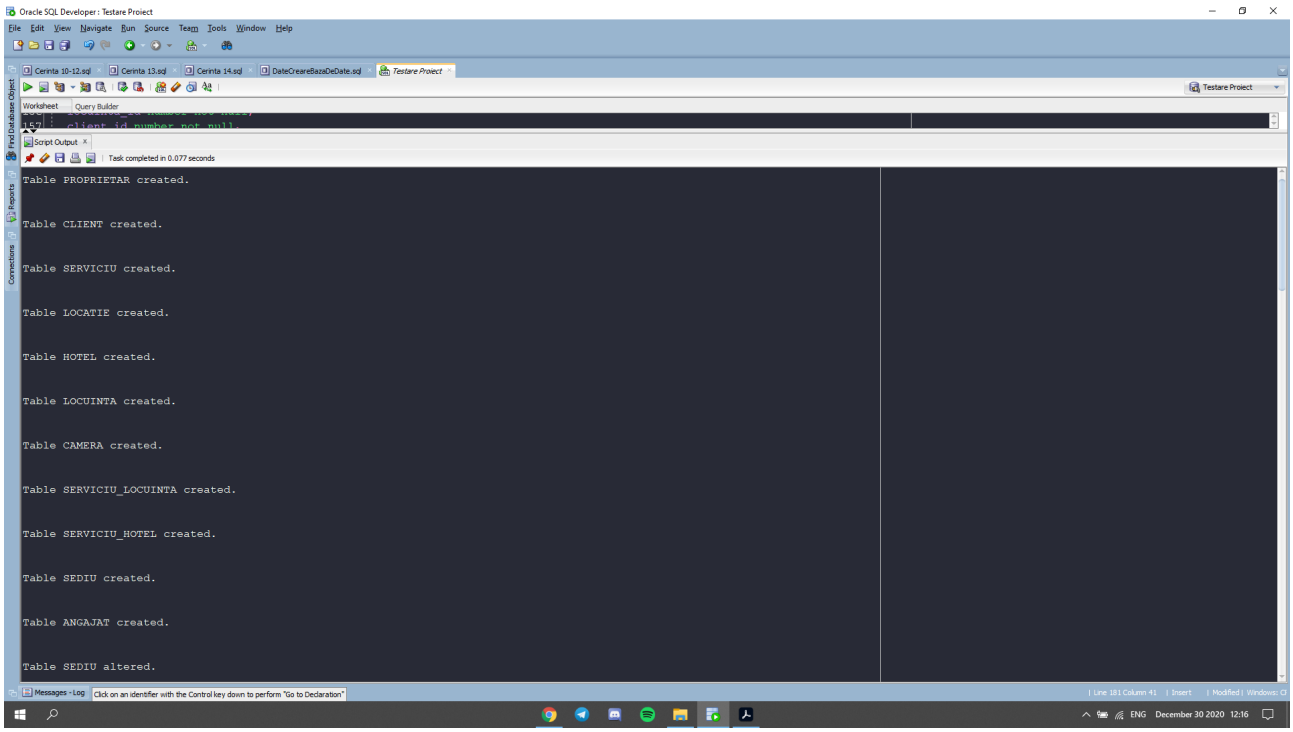

Figure 2.1: Crearea tabelelor (1)

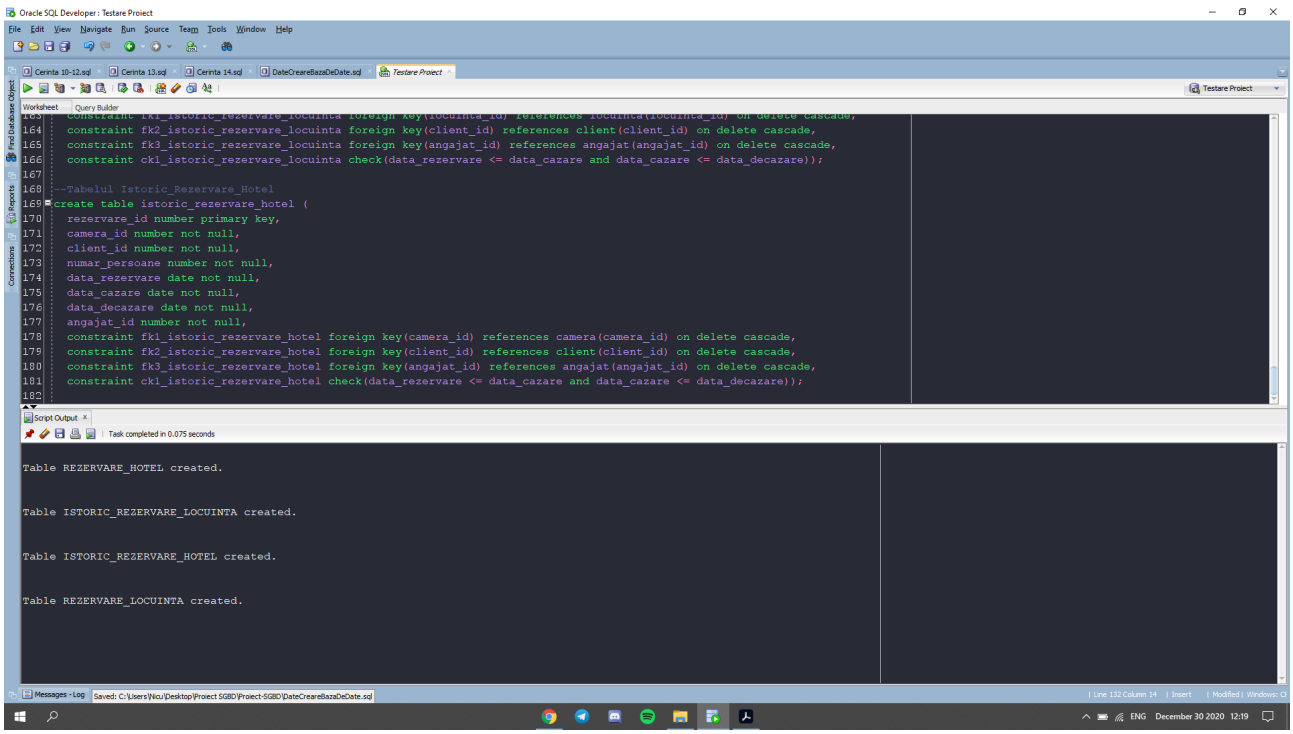

Figure 2.2: Crearea tabelelor (2)

2.2 Inserarea în tabel a datelor (Cerința nr. 5)

Următorul pas în crearea bazei noastre date este introducerea datelor coerente pentru tabelele noastre. Voi prezenta aici cele mai utilizate linii pe care le-am folosit în exemplele din cerințele ce urmează. Toate liniile utilizte în proiect pot fi consultate si utilizate la acest link. Remarcă: Datele mari au fost generate cu utilizarea unui script de Python.

```
--Proiect SGBD: Cerinta nr. 5
--Insertia datelor coerente in tabele

--Date pentru tabelul Locatie
insert into locatie values
    (1, 'Main Street 49', 680953, 'Tirana', 'Albania');
insert into locatie values
    (2, 'Field Alley 26', 920458, 'Andorra la Vella', 'Andorra');
insert into locatie values
    (3, 'Station Street 407', 236405, 'Vienna', 'Austria');
insert into locatie values
    (4, 'Church Street 197', 421670, 'Minsk', 'Belarus');
insert into locatie values
    (5, 'Mill Street 60', 486664, 'Brussels', 'Belgium');
insert into locatie values
    (6, 'School Street 452', 773878, 'Sarajevo', 'Bosnia and Herzegovina');
insert into locatie values
    (7, 'Garden Street 136', 315450, 'Sofia', 'Bulgaria');
insert into locatie values
    (8, 'Short Street 153', 670665, 'Zagreb', 'Croatia');
insert into locatie values
    (9, 'Station Street 424', 151291, 'Prague', 'Czechia');
insert into locatie values
    (10, 'Lark Street 122', 550665, 'Copenhagen', 'Denmark');
insert into locatie values
    (11, 'Birch Street 487', 477967, 'Tallinn', 'Estonia');
insert into locatie values
    (12, 'Angle Street 182', 901047, 'Helsinki', 'Finland');
insert into locatie values
    (13, 'Beach Road 329', 301537, 'Paris', 'France');
insert into locatie values
    (14, 'Church Street 499', 759825, 'Berlin', 'Germany');
insert into locatie values
    (15, 'School Street 354', 641379, 'Gibraltar', 'Gibraltar');
insert into locatie values
    (16, 'Rome Street 60', 126060, 'Athens', 'Greece');
insert into locatie values
    (17, 'Garibaldi Street 369', 856714, 'Saint Peter Port', 'Guernsey');
insert into locatie values
    (18, 'Marconi Street 356', 641600, 'Budapest', 'Hungary');
insert into locatie values
    (19, 'Linden Street 374', 886157, 'Reykjavik', 'Iceland');
insert into locatie values
    (20, 'Forest Street 291', 467551, 'Dublin', 'Ireland');
insert into locatie values
    (21, 'Birch Street 145', 822400, 'Douglas', 'Isle of Man');
insert into locatie values
    (22, 'Field Street 83', 793403, 'Rome', 'Italy');
insert into locatie values
    (23, 'Forest Street 171', 767090, 'Saint Helier', 'Jersey');
```

```
insert into locatie values
    (24, 'Sunny Street 48', 618860, 'Pristina', 'Kosovo');
insert into locatie values
    (25, 'Central Street 277', 733341, 'Riga', 'Latvia');
insert into locatie values
    (26, 'Youth Street 425', 122415, 'Vaduz', 'Liechtenstein');
insert into locatie values
    (27, 'School Street 380', 865229, 'Vilnius', 'Lithuania');
insert into locatie values
    (28, 'Ring Road 152', 531194, 'Luxembourg', 'Luxembourg');
insert into locatie values
    (29, 'Grand Street 378', 343951, 'Valletta', 'Malta');
insert into locatie values
    (30, 'Stefan cel Mare Street 63', 459969, 'Chisinau', 'Moldova');
insert into locatie values
    (31, 'Station Street 192', 989326, 'Monaco', 'Monaco');
insert into locatie values
    (32, 'Main Street 463', 652441, 'Podgorica', 'Montenegro');
insert into locatie values
    (33, 'Village Street 285', 283507, 'Amsterdam', 'Netherlands');
insert into locatie values
    (34, 'Church Street 139', 432308, 'Skopje', 'North Macedonia');
insert into locatie values
    (35, 'Field Street 152', 171528, 'Oslo', 'Norway');
insert into locatie values
    (36, 'Meadow Street 139', 797676, 'Warsaw', 'Poland');
insert into locatie values
    (37, 'School Street 265', 825955, 'Lisbon', 'Portugal');
insert into locatie values
    (38, 'Lujerului Street 315', 488255, 'Bucuresti', 'Romania');
insert into locatie values
    (39, 'Tverskaia Street 165', 935472, 'Moscow', 'Russia');
insert into locatie values
    (40, 'School Street 167', 574556, 'San Marino', 'San Marino');
insert into locatie values
    (41, 'Preseren Street 68', 512202, 'Belgrade', 'Serbia');
insert into locatie values
    (42, 'Garden Street 319', 351035, 'Bratislava', 'Slovakia');
insert into locatie values
    (43, 'Ring Road 303', 728262, 'Ljubljana', 'Slovenia');
insert into locatie values
    (44, 'Grand Street 339', 948719, 'Madrid', 'Spain');
insert into locatie values
    (45, 'School Street 287', 657165, 'Longyearbyen', 'Svalbard');
insert into locatie values
    (46, 'Lark Street 443', 875896, 'Stockholm', 'Sweden');
insert into locatie values
    (47, 'Birch Street 495', 615401, 'Bern', 'Switzerland');
insert into locatie values
    (48, 'Angle Street 236', 525700, 'Kiev', 'Ukraine');
```

```
insert into locatie values
    (49, 'Tea Street 136', 607859, 'London', 'United Kingdom');
insert into locatie values
    (50, 'Academiei Street 392', 358891, 'Bucuresti', 'Romania');
insert into locatie values
    (51, 'Aviatorilor Street 405', 372338, 'Bucuresti', 'Romania');
insert into locatie values
    (52, 'Aviatorilor Street 433', 103239, 'Bucuresti', 'Romania');
insert into locatie values
    (53, 'Dristor Street 179', 162182, 'Bucuresti', 'Romania');
insert into locatie values
    (54, 'Lujerului Street 358', 302618, 'Bucuresti', 'Romania');
insert into locatie values
    (55, 'Gorjului Street 218', 654009, 'Bucuresti', 'Romania');
insert into locatie values
    (56, 'Erorilor Street 95', 596877, 'Bucuresti', 'Romania');
insert into locatie values
    (57, 'Gorjului Street 479', 538786, 'Bucuresti', 'Romania');
insert into locatie values
    (58, 'Erorilor Street 23', 157817, 'Bucuresti', 'Romania');
insert into locatie values
    (59, 'Tineretului Street 143', 855434, 'Cluj', 'Romania');
insert into locatie values
    (60, 'Universitatii Street 113', 604271, 'Cluj', 'Romania');
insert into locatie values
    (61, 'Unirii Street 190', 940977, 'Cluj', 'Romania');
insert into locatie values
    (62, 'Unirii Street 398', 506606, 'Iasi', 'Romania');
insert into locatie values
    (63, 'Unirii Street 436', 765577, 'Timisoara', 'Romania');

--Date pentru tabelul Client
insert into client values
    (1, 'Smith', 'Emma', 'smith.emma@icloud.com',
    99155828, to_date('22-03-2003', 'DD-MM-YYYY'));
insert into client values
    (2, 'Johnson', 'Olivia', 'johnsonolivia@yahoo.com',
    91375434, to_date('29-04-1993', 'DD-MM-YYYY'));
insert into client values
    (3, 'Williams', 'Noah', 'williams.noah@gmail.com',
    43260301, to_date('25-04-1954', 'DD-MM-YYYY'));
insert into client values
    (4, 'Brown', 'Liam', 'brown.liam@icloud.com',
    72581670, to_date('11-05-2003', 'DD-MM-YYYY'));
insert into client values
    (5, 'Jones', 'Sophia', 'jones.sophia@outlook.com',
    65576421, to_date('20-08-1952', 'DD-MM-YYYY'));
insert into client values
    (6, 'Miller', 'Mason', 'miller_mason@outlook.com',
    25636148, to_date('11-08-1953', 'DD-MM-YYYY'));
```

```
insert into client values
    (7, 'Davis', 'Ava', 'davis_ava@icloud.com',
    33240263, to_date('19-12-1984', 'DD-MM-YYYY'));
insert into client values
    (8, 'Garcia', 'Jacob', 'garcia_jacob@apple.com',
    20279417, to_date('05-04-1960', 'DD-MM-YYYY'));
insert into client values
    (9, 'Rodriguez', 'William', 'rodriguez.william@gmail.com',
    52637808, to_date('27-11-2002', 'DD-MM-YYYY'));
insert into client values
    (10, 'Wilson', 'Isabella', 'wilsonisabella@yahoo.com',
    93911010, to_date('01-02-1976', 'DD-MM-YYYY'));
insert into client values
    (11, 'Martinez', 'Ethan', 'martinez_ethan@apple.com',
    99246903, to_date('16-11-2001', 'DD-MM-YYYY'));
insert into client values
    (12, 'Anderson', 'Mia', 'anderson_mia@apple.com',
    78950147, to_date('18-06-1993', 'DD-MM-YYYY'));
insert into client values
    (13, 'Taylor', 'James', 'taylor.james@icloud.com',
    44575251, to_date('14-03-1960', 'DD-MM-YYYY'));
insert into client values
    (14, 'Thomas', 'Alexander', 'thomasalexander@icloud.com',
    37609610, to_date('09-08-1994', 'DD-MM-YYYY'));
insert into client values
    (15, 'Hernandez', 'Michael', 'hernandezmichael@apple.com',
    65063852, to_date('25-03-1982', 'DD-MM-YYYY'));

--Date pentru Hotel
insert into hotel values (1, 'Grand Romania', 5, 59);
insert into hotel values (2, 'Hotel Plaza Bucuresti', 4, 58);
insert into hotel values (3, 'Iasi Resort and Spa', 4, 62);
insert into hotel values (4, 'Iulius Hotel', 2, 61);
insert into hotel values (5, 'ARIA RESORT', 3, 1);
insert into hotel values (6, 'Hotel California', 3, 2);
insert into hotel values (7, 'Hotel Cismigiu', 1, 3);
insert into hotel values (8, 'Liberton', 5, 4);
insert into hotel values (9, 'Grand Plaza', 5, 5);
insert into hotel values (10, 'New York Plaza', 3, 6);

--Date pentru serviciu
--Putem folosi o secventa
create sequence secv_serviciu
increment by 10
start with 10;

insert into serviciu values (secv_serviciu.nextval, 'Parcare', 8);
insert into serviciu values (secv_serviciu.nextval, 'Pat pentru copii', 10);
insert into serviciu values (secv_serviciu.nextval, 'Masina de spalat rufe', 9);
insert into serviciu values (secv_serviciu.nextval, 'Masina de spalat vase', 10);
```

```
insert into serviciu values (secv_serviciu.nextval, 'Paturi separate', 6);
insert into serviciu values (secv_serviciu.nextval, 'Bucatarie moderna', 10);
insert into serviciu values (secv_serviciu.nextval, 'Cabina de dus', 9);
insert into serviciu values
    (secv_serviciu.nextval, 'Teren de joaca pentru copii', 10);
insert into serviciu values (secv_serviciu.nextval, 'Deserivire in camera', 10);
insert into serviciu values
    (secv_serviciu.nextval, 'Bucatarie si sufragerie comuna', 8);
insert into serviciu values (secv_serviciu.nextval, 'Camere spatioase', 9);
```

Observație: Comenzile de insert de mai sus au fost introduse în acest proiect pentru a prezenta câteva exemple coerente dintre ele. Din cauza abundenței mare de date utilizate în proiect (circa 500 de linii de insert), nu le voi introduce în continuare pe celelalte, pentru a nu crea o redundanță mare în conținutul acestui proiect (pentru că ar utiliza mai mult de 15 pagini). Pentru a utiliza toate inserturile utilizate în proiect, sugerez călduros să fie accesat fișierul SQL pe link-ul de Github al proiectului meu, care poate fi accesat aici.

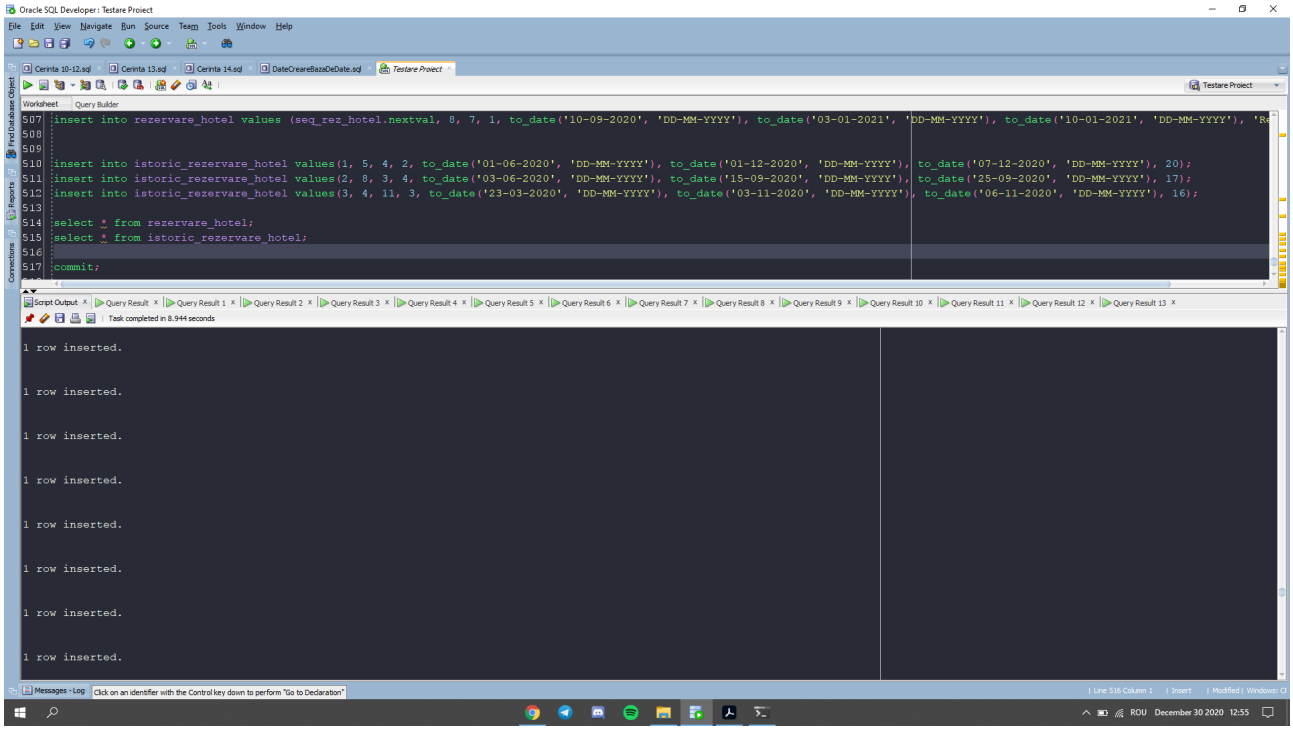

Figure 2.3: Inserarea datelor in tabele

Subprograme. Functii. Proceduri (Cerințele nr. 6-9)

În acestă parte a referatului voi prezenta rezolvările cerințelor nr. 6 - 9, prezentând niște subprograme ce ar corespunde cu unele necesități ale companiei în viața reală.

3.1 Cerința nr. 6

Cerința: Definiți un subprogram stocat care să utilizeze un tip de colecție studiat. Apelați subprogramul.

Condiția problemei: Compania noastră trebuie să facă un raport anual la fiecare sfarșit de an. Raportul constă din următoarele date: un tablou cu angajatii companiei care sunt agenți imobiliari (pentru că doar ei pot aproba rezervările de locuință și camere din hoteluri) și câte rezervări de camere și locuințe au efectuat aceștia in anul solicitat (în cazul nostru, va fi anul 2020). De asemenea, să se menționeze angajatul cu cea mai bună statistică și să i se ofere o mărire de salariu egala cu 10% din media salariilor tuturor salariilor agenților imobiliari. (Dacă sunt mai mulți, să se afișeze toți, respectiv să li se mareasca salariul tuturor). Remarcă: angajatul trebuie să aibă cel puțin o rezervare facută ca să poată intra in acest top. Observație: Trebuie să fie luate in considerare nu doar rezervările care încă sunt valabile, ci și cele care au fost finalizate (adică rezervarile în urma carora s-au decazat clienții).

Rezolvare: Pentru a rezolva această problema, putem folosi două colecții studiate la curs și la laborator: tipul de date record și tabel (indexat, imbricat, sau vectori). Pentru problema noastră vom folosi un tabel imbricat. Vom face un tip record care sa conțină numele si prenumele angajatului, împreuna cu numărul de rezervari efectuate pe parcursul anului. Apoi, vom face tabelul nostru imbricat de acest tip record și vom mentine în el toate datele necesare. La final, îi vom afișa pe toți și vom efectua rezolva restul problemei (partea cu salariile).

În continuare voi prezenta în referat codul cu rezolvarea în PL\SQL a problemei. Codul în format .SQL poate fi vizualizat și utilizat pe acest link.

```
--Ducal Nicolae, grupa 234
--Proiect SGBD: Cerinta nr. 6

set serveroutput on;

--Procedura noastra primeste ca parametru de intrare anul,
--caruia vrem sa ii facem raportul
create or replace procedure raport_anual
  (v_an in varchar2)
is
  type statistica_angajat is record
    (id angajat.angajat_id%type,
     nume angajat.nume%type,
     prenume angajat.prenume%type,
     efectuate number);

  type raport is table of statistica_angajat;

  type indecsi_best is table of number;

  efect_loc number;
  efect_hot number;
  efect_ist_loc number;
  efect_ist_hot number;
  cur_stat statistica_angajat;
  rap raport := raport();
  idx integer;
  avg number := 0;
  best_stat number := 0;
  best_angajati indecsi_best := indecsi_best();
begin
  --Vom folosi un cursor ca sa parcurgem prin toti angajatii de tip agent imobiliar
  idx := 1;
  for ang in (select angajat_id, nume, prenume
              from angajat
              where job = 'Agent imobiliar') loop

    --Numarul de rezervari curente de locuinte
    select count(*) into efect_loc
    from rezervare_locuinta
    where angajat_id = ang.angajat_id
    and to_char(extract(year from data_rezervare)) = v_an;

    --Numarul de rezervari de locuinte din istoric
    select count(*) into efect_ist_loc
    from istoric_rezervare_locuinta
    where angajat_id = ang.angajat_id
    and extract(year from data_rezervare) = v_an;

    --Numarul de rezervari curente de hoteluri
```

```
    select count(*) into efect_hot
    from rezervare_hotel
    where angajat_id = ang.angajat_id
    and to_char(extract(year from data_rezervare)) = v_an;

    --Numarul de rezervari de hoteluri din istoric
    select count(*) into efect_ist_hot
    from istoric_rezervare_hotel
    where angajat_id = ang.angajat_id
    and extract(year from data_rezervare) = v_an;

    cur_stat.id := ang.angajat_id;
    cur_stat.nume := ang.nume;
    cur_stat.prenume := ang.prenume;
    cur_stat.efectuate := efect_loc + efect_hot + efect_ist_loc + efect_ist_hot;

    rap.extend;
    rap(idx) := cur_stat;
    idx := idx + 1;
  end loop;

  --Afisam raportul:
  for i in rap.first..rap.last loop
    dbms_output.put_line(i || ') ' || rap(i).nume || ' ' || rap(i).prenume
        || ' a efectuat ' || rap(i).efectuate || ' rezervari.');

    --Vom pastra in best_stat numarul maxim de rezervari
    --efectuate de catre un agent imobiliar
    if best_stat < rap(i).efectuate then
      best_stat := rap(i).efectuate;
    end if;
  end loop;

  --Afisam cei mai buni angajati
  idx := 0;
  for i in rap.first..rap.last loop
    if rap(i).efectuate = best_stat then
      --Pastram indexul din tabelul raport
      idx := idx + 1;
      best_angajati.extend;
      best_angajati(idx) := i;
    end if;
  end loop;

  dbms_output.put_line('');
  if idx = 0 then
    dbms_output.put_line('Nici un angajat nu poate fi considerat
        cel mai bun in acest an');
  else
    if idx = 1 then
```

```
      dbms_output.put_line('Cel mai bun agent imobiliar din acest an este '
        || rap(best_angajati(idx)).nume || ' ' || rap(best_angajati(idx)).prenume);
    else
      dbms_output.put_line('Cei mai buni agenti imobiliari din acest an sunt: ');
      for i in best_angajati.first..best_angajati.last loop
        dbms_output.put_line(i || ') ' || rap(best_angajati(i)).nume
            || ' ' || rap(best_angajati(i)).prenume);
      end loop;
    end if;

    --Marim salariile:
    select avg(salariu) into avg
    from angajat
    where job = 'Agent imobiliar';

    for i in best_angajati.first..best_angajati.last loop
      update angajat
      set salariu = floor(salariu + 0.1 * avg)
      where angajat_id = rap(best_angajati(i)).id;
    end loop;
  end if;
end raport_anual;
/

--Apelam functia pentru anul 2020;
begin
  raport_anual('2020');
end;
/

select * from angajat where job = 'Agent imobiliar';
rollback;
```

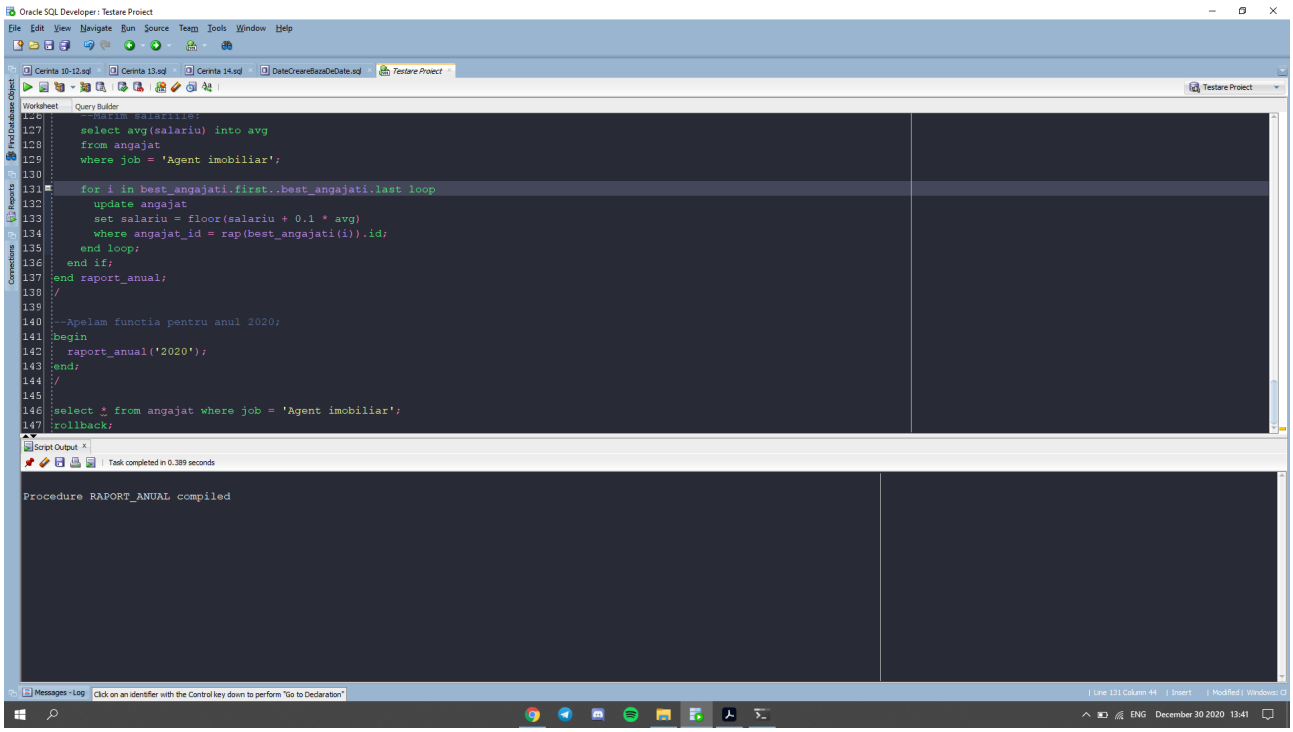

Figure 3.1: Compilarea procedurii raport_anual

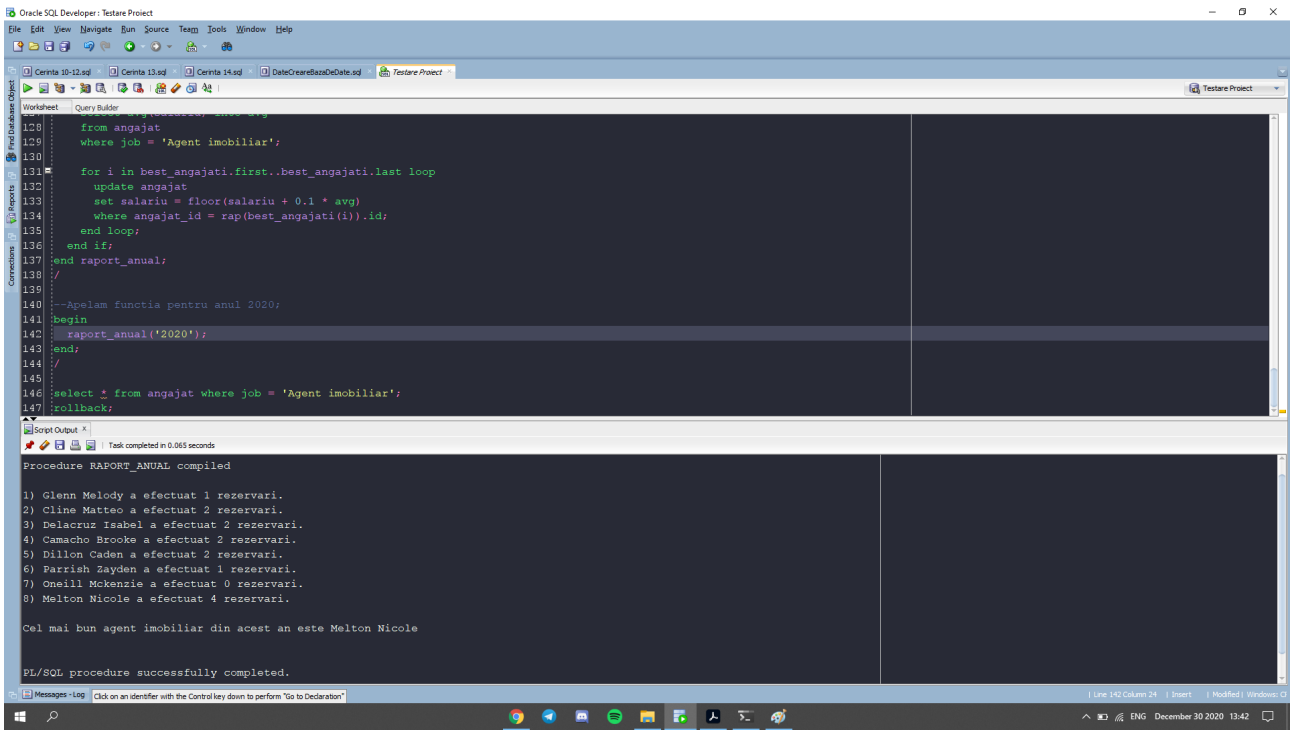

Figure 3.2: Rularea procedurii cu parametrul 2020

3.2 Cerința nr. 7

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

Condiția problemei: Să se afișeze cele mai bune locuințe și hoteluri la momentul de față. Țara și orașul pot fi introduse de către utilizator. Dacă nu, se afișează locuințele/hotelurile din toate locațiile disponibile. O locuință sau un hotel este considerat bună, dacă ratingul mediu al

serviciilor de care beneficiază locuința/hotelul este cel puțin 9.5 (ratingul maxim posibil este 10). Să se afișeze mai întâi toate locuințele cele mai bune, specificandu-se: adresa, orașul, țara și proprietarul. De asemenea, să se afișeze ratingul și serviciile de care dispune. Pentru Hotel, să se specifice denumirea, locația (adresa, orașul, țara), numărul de stele și serviciile.

Rezolvare: Vom menține datele necesare despre locuințe și hoteluri în două cursoare explicite. Apoi vom parcurge prin serviciile de care beneficiază fiecare din ele. Pentru hoteluri vom folosi un ciclu cursor. Exercițiul este rezolvat cu utilizarea a 8 tabele. Codul în format .SQL poate fi vizualizat și utilizat pe acest link.

```
--Ducal Nicolae, grupa 234
--Proiect SGBD: Cerinta nr. 7

set serveroutput on;

create or replace procedure determina_top
 (tara_sol varchar2 default '%', oras_sol varchar2 default '%')
is
  cursor c_loc is
    select lo.locuinta_id loc_id, lc.adresa adresa,
      lc.oras oras, lc.tara tara, pr.nume nume, pr.prenume prenume
    from locuinta lo, locatie lc, proprietar pr
    where lo.locatie_id = lc.locatie_id and pr.proprietar_id = lo.proprietar_id
    and lc.tara like(tara_sol) and lc.oras like(oras_sol);

  cursor c_hot is
    select h.hotel_id hotel_id, h.denumire denumire, h.numar_stele stele,
      lc.adresa adresa, lc.oras oras, lc.tara tara
    from hotel h, locatie lc
    where h.locatie_id = lc.locatie_id
    and lc.tara like(tara_sol) and lc.oras like(oras_sol);

  v_loc_id locuinta.locuinta_id%type;
  v_adresa locatie.adresa%type;
  v_oras locatie.oras%type;
  v_tara locatie.tara%type;
  v_nume proprietar.nume%type;
  v_prenume proprietar.prenume%type;
  good integer;
  check_oras_tara integer;
  rating_mediu number;
  iter number;

begin
  --Sa vedem daca orasul si tara sunt valide:
  select count(*) into check_oras_tara
  from locatie lc
  where lc.tara like(tara_sol) and lc.oras like(oras_sol);

  if check_oras_tara = 0 then
    raise_application_error('-20001', 'Nu exista date despre locatia indicata');
```

```
end if;

--Iteram initial prin locuinte
--Good va fi indicator ca avem sau nu locuinte bune
--Daca e 0 - nu, daca e 1 - da, daca e 2 - a fost 1 deja
good := 0;
open c_loc;
loop
  fetch c_loc into v_loc_id, v_adresa, v_oras, v_tara, v_nume, v_prenume;
  exit when c_loc%notfound;

  select avg(s.rating) into rating_mediu
  from serviciu_locuinta sl, serviciu s
  where sl.serviciu_id = s.serviciu_id
  and sl.locuinta_id = v_loc_id;

  if nvl(rating_mediu, 0) >= 9.5 then
    if good = 0 and oras_sol = '%' and tara_sol = '%' then
      dbms_output.put_line('Cele mai bune locuinte sunt: ');
    elsif good = 0 and oras_sol = '%' then
      dbms_output.put_line('Cele mai bune locuinte din ' || tara_sol
          || ' sunt: ');
    elsif good = 0 then
      dbms_output.put_line('Cele mai bune locuinte din ' || tara_sol
          || ', ' || oras_sol || ' sunt: ');
    end if;

    good := good + 1;

    if oras_sol = '%' and tara_sol = '%' then
      dbms_output.put_line(good || ') ' || v_adresa || ', ' || v_oras
          || ', ' || v_tara);
    elsif oras_sol = '%' then
      dbms_output.put_line(good || ') ' || v_adresa || ', ' || v_oras);
    else
      dbms_output.put_line(good || ') ' || v_adresa);
    end if;

    dbms_output.put_line('Rating: ' || rating_mediu);
    dbms_output.put_line('Proprietar: ' || v_nume || ' ' || v_prenume);
    dbms_output.put('Servicile propuse: ');
    --Folosim un ciclu cursor pentru a prezenta serviciile:
    for i in (select denumire
              from serviciu_locuinta sl, serviciu s
              where sl.serviciu_id = s.serviciu_id
              and sl.locuinta_id = v_loc_id) loop
      dbms_output.put(i.denumire || '; ');
    end loop;
    dbms_output.put_line('');
    dbms_output.put_line('');
```

```
  end if;
end loop;

if good = 0 then
  dbms_output.put_line('Nu au fost identificate locuinte care ar
      indeplini criteriului');
end if;

--Repetam pentru hotel
good := 0;
for hot in c_hot loop
  select avg(s.rating) into rating_mediu
  from serviciu_hotel sh, serviciu s
  where sh.serviciu_id = s.serviciu_id
  and sh.hotel_id = hot.hotel_id;

  if nvl(rating_mediu, 0) >= 9.5 then
    if good = 0 and oras_sol = '%' and tara_sol = '%' then
      dbms_output.put_line('Cele mai bune hoteluri sunt: ');
    elsif good = 0 and oras_sol = '%' then
      dbms_output.put_line('Cele mai bune hoteluri din ' || tara_sol
          || ' sunt: ');
    elsif good = 0 then
      dbms_output.put_line('Cele mai bune hoteluri din ' || tara_sol
          || ', ' || oras_sol || ' sunt: ');
    end if;

    good := good + 1;

    if oras_sol = '%' and tara_sol = '%' then
      dbms_output.put_line(good || ') ' || hot.denumire || ', ' || hot.stele
          || ' stele, ' || hot.adresa || ', ' || hot.oras || ', ' || hot.tara);
    elsif oras_sol = '%' then
      dbms_output.put_line(good || ') ' || hot.denumire || ', ' || hot.stele
          || ' stele, ' || hot.adresa || ', ' || hot.oras);
    else
      dbms_output.put_line(good || ') ' || hot.denumire || ', ' || hot.stele
          || ' stele, ' || hot.adresa);
    end if;

    dbms_output.put('Servicile propuse: ');
    --Folosim un ciclu cursor pentru a prezenta serviciile:
    for i in (select denumire
              from serviciu_hotel sh, serviciu s
              where sh.serviciu_id = s.serviciu_id
              and sh.hotel_id = hot.hotel_id) loop
      dbms_output.put(i.denumire || '; ');
    end loop;
    dbms_output.put_line('');
    dbms_output.put_line('');
```

```
    end if;
  end loop;

  if good = 0 then
    dbms_output.put_line('Nu au fost identificate hoteluri care ar
        indeplini criteriului');
  end if;
end determina_top;
/

--Cazul general
begin
  determina_top();
end;
/

--Cazul doar cu țara specificată
begin
  determina_top('Romania');
end;
/

--Cazul cu țara și orașul indicat
begin
  determina_top('Romania', 'Cluj');
end;
/
```

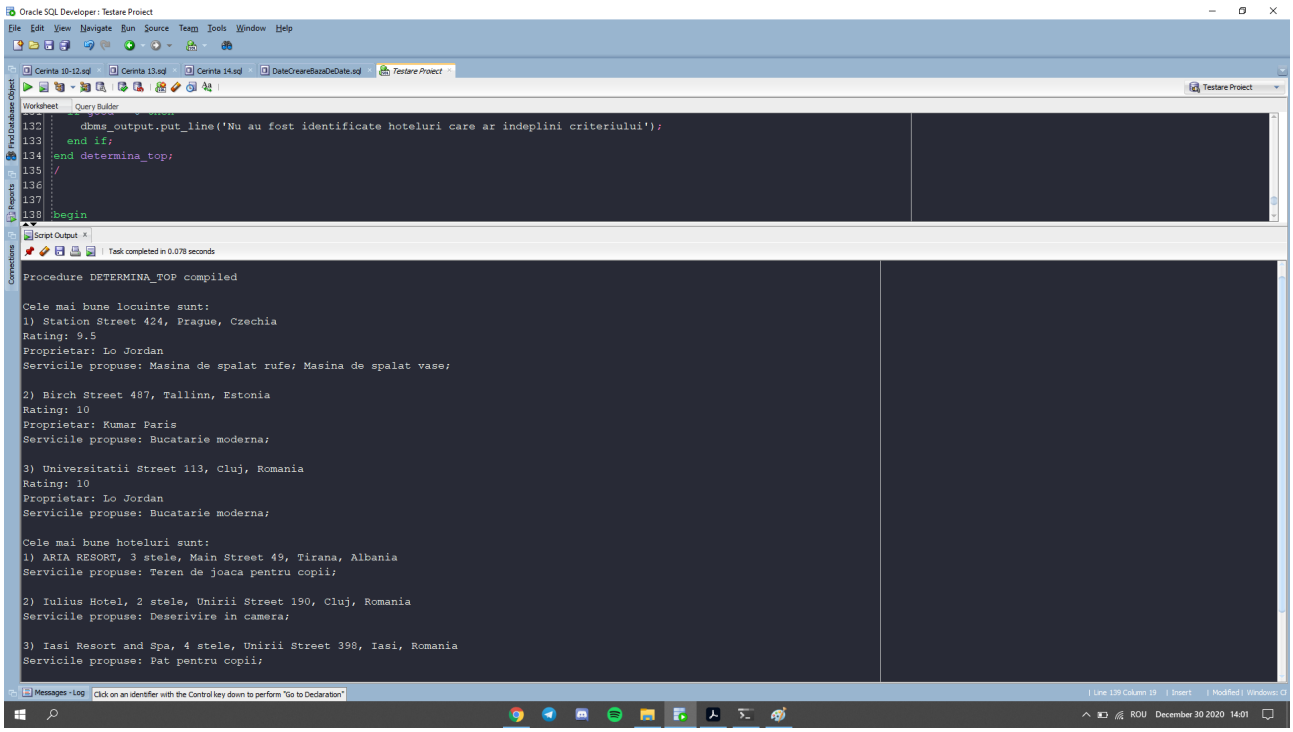

Figure 3.3: Compilarea si rularea pe cazul fără țara și orașul indicat

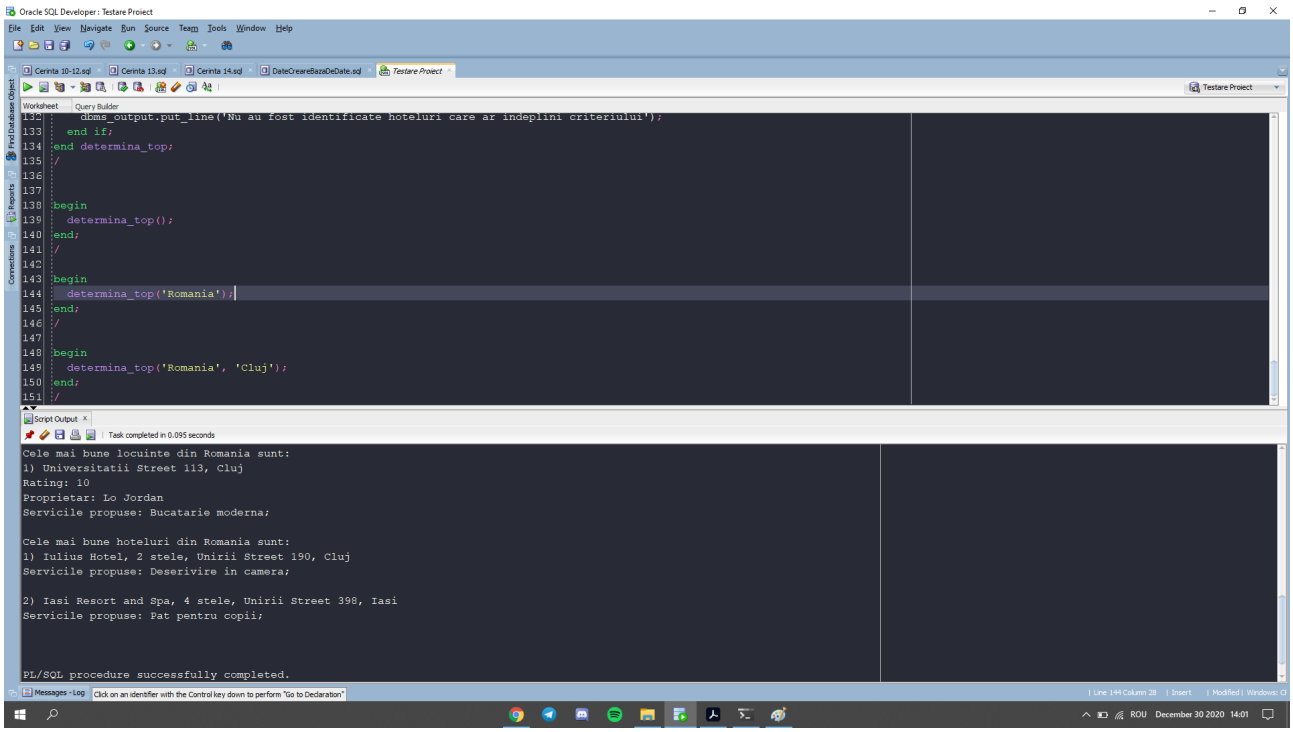

Figure 3.4: Rularea pe cazul cu țara specificată

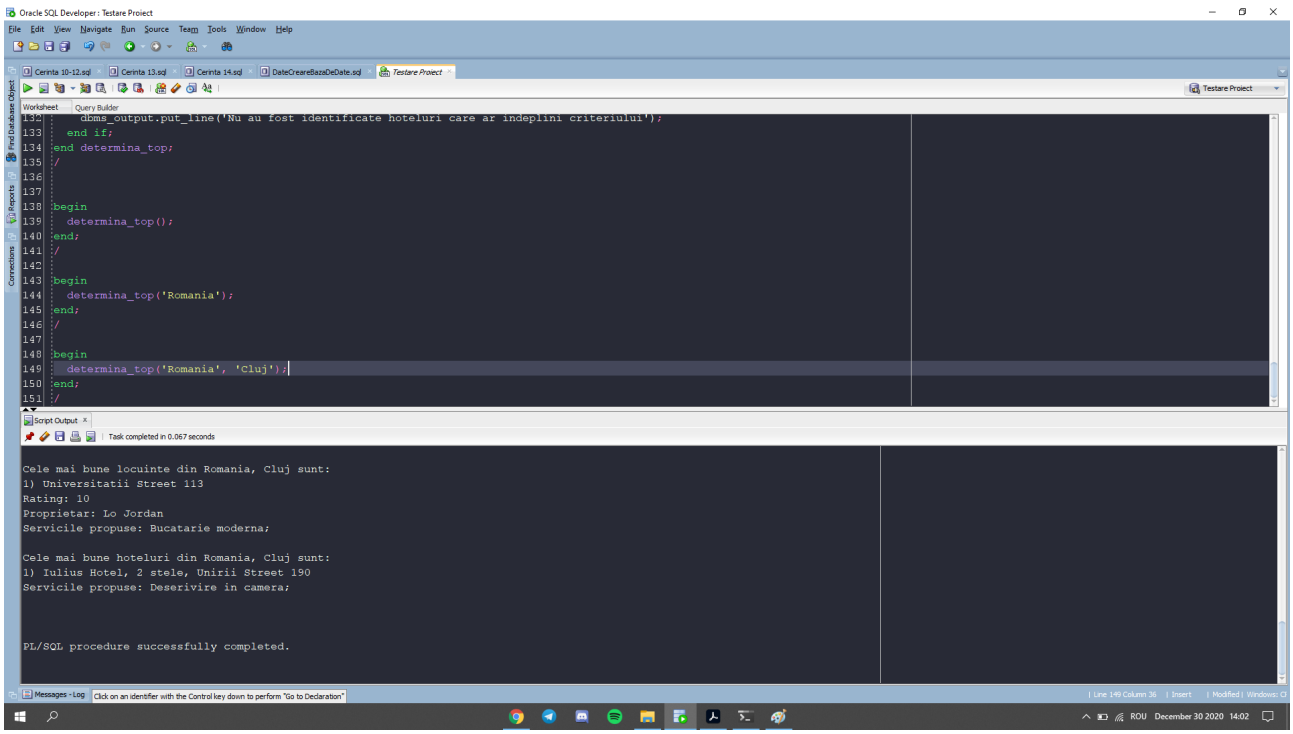

Figure 3.5: Rularea pe cazul cu țara și orașul specificat

3.3 Cerința nr. 8

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

Condiție problemă: Să se implementeze o funcție care returneaza costul total achitat de către clientul care a chelutit cea mai mare sumă de bani pentru cazări într-un anumit an. Anul va fi

transmis ca parametru funcției și vor fi luate in calcul anul rezervarii, nu cazarii. Observație: Costul pentru o rezervare se va calcula in felul urmator:

1. Dacă persoana s-a cazat, trebuie să plătească în ziua cazării toată perioada indicată în rezervare, adică: preț_locuință/cameră * numărul de zile de cazare (de la data cazării până la data decazarii)
2. Dacă persoana încă nu s-a cazat și a efectuat doar rezervarea, se va plăti un avans in valoare de jumătate din costul care ar trebui sa fie platit la cazare. La cazare va fi achitata cea de-a doua parte a sumei. În caz de anulare a rezervarii, suma va fi restituita doar în anumite condiții. Restituirea nu ne intereseaza pentru această cerință.

Codul în format .SQL poate fi vizualizat și utilizat pe acest link.

```
--Ducal Nicolae, grupa 234
--Proiect SGBD: Cerinta nr. 8

set serveroutput on;

create or replace function costul_maxim
  (v_an varchar2)
return locuinta.pret%type is
  --Vom folosi un cursor ca sa iteram doar prin clientii care facut careva rezervari
  --in anul solicitat
  cursor c_cl is select c.client_id id
         from client c, rezervare_locuinta rl
         where rl.client_id = c.client_id and
         to_char(extract(year from rl.data_rezervare)) = v_an
         union
         select c.client_id id
         from client c, istoric_rezervare_locuinta irl
         where irl.client_id = c.client_id and
         to_char(extract(year from irl.data_rezervare)) = v_an
         union
         select c.client_id id
         from client c, rezervare_hotel rh
         where rh.client_id = c.client_id and
         to_char(extract(year from rh.data_rezervare)) = v_an
         union
         select c.client_id id
         from client c, istoric_rezervare_hotel irh
         where irh.client_id = c.client_id and
         to_char(extract(year from irh.data_rezervare)) = v_an;
  max_cost locuinta.pret%type := 0;
  cur_cost locuinta.pret%type := 0;
  fara_inregistrari exception;
  good integer;
  verifica_an number;
begin
  --Verificam daca e bun anul introdus
  verifica_an := to_number(v_an);
  --Verificam daca avem inregistrari
```

```
good := 0;
for i in c_cl loop
  good := good + 1;
end loop;
if good = 0 then
  raise fara_inregistrari;
end if;


for cl in c_cl loop
  --Pentru toate rezervarile din istoric, e evident ca se va lua suma pentru
  --toate zilele de cazare;
  cur_cost := 0;
  for rez in (select irl.data_cazare d_caz, irl.data_decazare d_decaz,
                  lo.pret pret
              from istoric_rezervare_locuinta irl, locuinta lo
              where client_id = cl.id and lo.locuinta_id = irl.locuinta_id) loop
    cur_cost := cur_cost + rez.pret * round(rez.d_decaz - rez.d_caz);
  end loop;

  for rez in (select irh.data_cazare d_caz, irh.data_decazare d_decaz,
                  cam.pret pret
              from istoric_rezervare_hotel irh, camera cam
              where client_id = cl.id and cam.camera_id = irh.camera_id) loop
    cur_cost := cur_cost + rez.pret * round(rez.d_decaz - rez.d_caz);
  end loop;

  --Pentru rezervarile care inca sunt valabile si pentru cele cazate,
  --aplicam formulele
  for rez in (select rl.data_cazare d_caz, rl.data_decazare d_decaz,
                rl.status status, lo.pret pret
              from rezervare_locuinta rl, locuinta lo
              where client_id = cl.id and lo.locuinta_id = rl.locuinta_id) loop
    if rez.status = 'Rezervat' then
      cur_cost := cur_cost + 0.5 * rez.pret * round(rez.d_decaz - rez.d_caz);
    elsif rez.status = 'Cazat' then
      cur_cost := cur_cost + rez.pret * round(rez.d_decaz - rez.d_caz);
    end if;
  end loop;

  for rez in (select rh.data_cazare d_caz, rh.data_decazare d_decaz,
                rh.status status, cam.pret pret
              from rezervare_hotel rh, camera cam
              where client_id = cl.id and cam.camera_id = rh.camera_id) loop
    if rez.status = 'Rezervat' then
      cur_cost := cur_cost + 0.5 * rez.pret * round(rez.d_decaz - rez.d_caz);
    elsif rez.status = 'Cazat' then
      cur_cost := cur_cost + rez.pret * round(rez.d_decaz - rez.d_caz);
    end if;
  end loop;
```

```
    --Verificam daca e maxim cheltuielile persoanei in cauza
    if cur_cost > max_cost then max_cost := cur_cost; end if;
    --dbms_output.put_line(cl.id || ' ' || cur_cost);
  end loop;
return max_cost;
exception
  when value_error then
    raise_application_error('-20010', 'Anul introdus este gresit (nu este numeric)');
  when fara_inregistrari then
    raise_application_error('-20011', 'Nu a fost identificata nici o inregistrare
      despre cheltuieli');
  when others then
    raise_application_error('-20012', 'Alta eroare');
end costul_maxim;
/

--Caz bun:
begin
  dbms_output.put_line('Costul maxim in anul 2020 este ' || costul_maxim('2020'));
end;
/

--Eroarea de an:
begin
  dbms_output.put_line('Costul maxim in anul 2020 este ' || costul_maxim('202a'));
end;
/

--Nu avem inregistrari
begin
  dbms_output.put_line('Costul maxim in anul 2018 este ' || costul_maxim('2018'));
end;
/
```

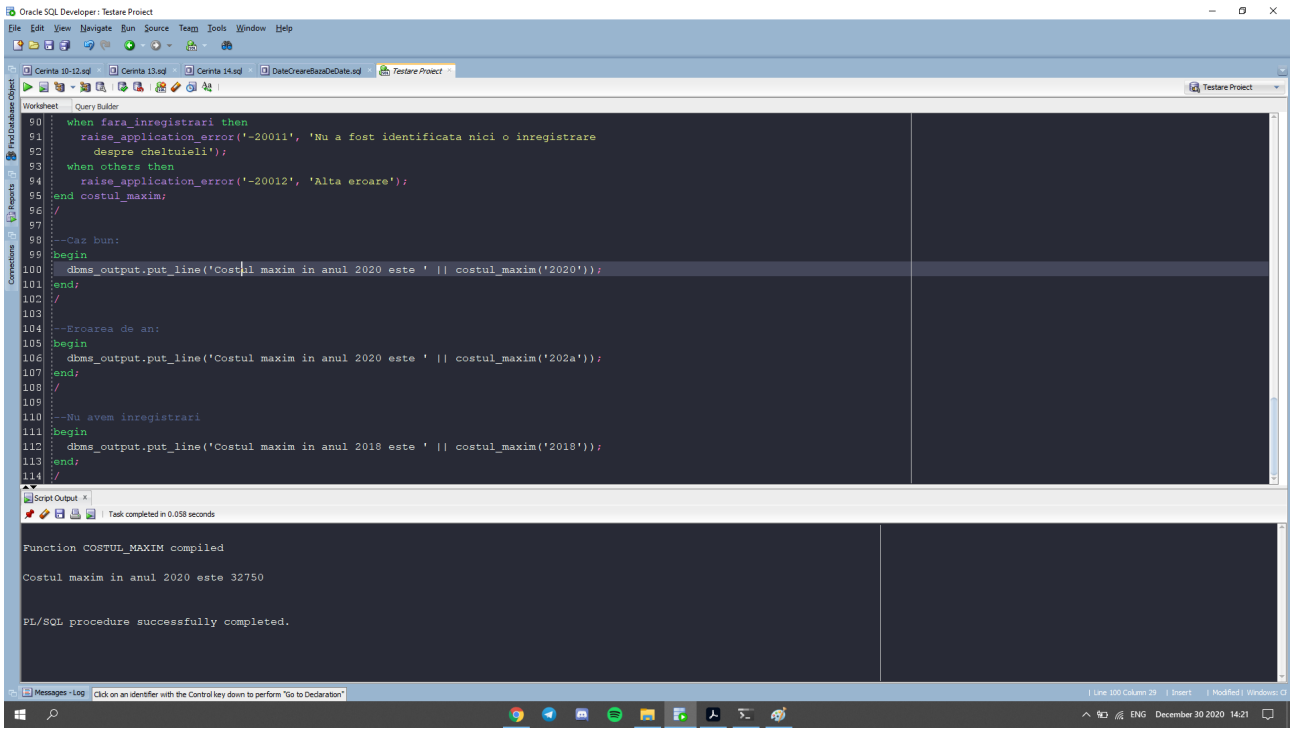

Figure 3.6: Compilarea si rularea pe cazul cu anul 2020

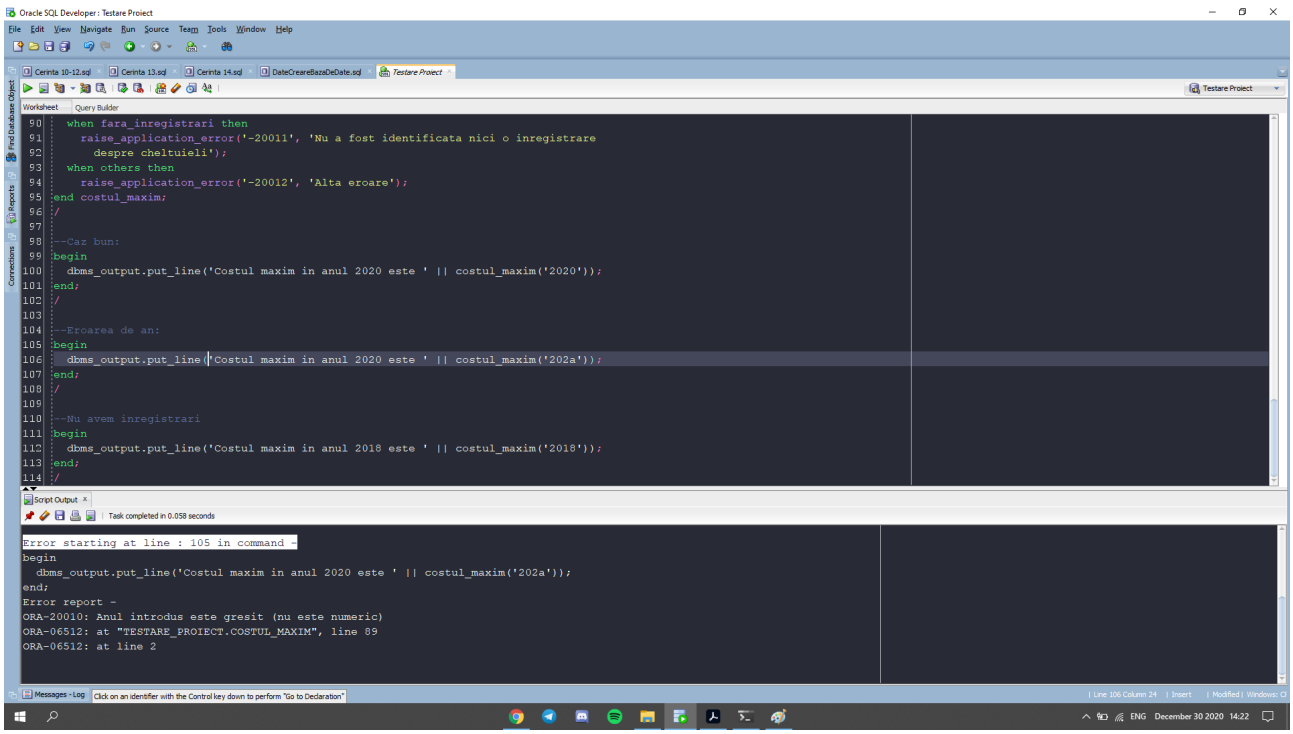

Figure 3.7: Rularea pe cazul cu anul introdus greșit

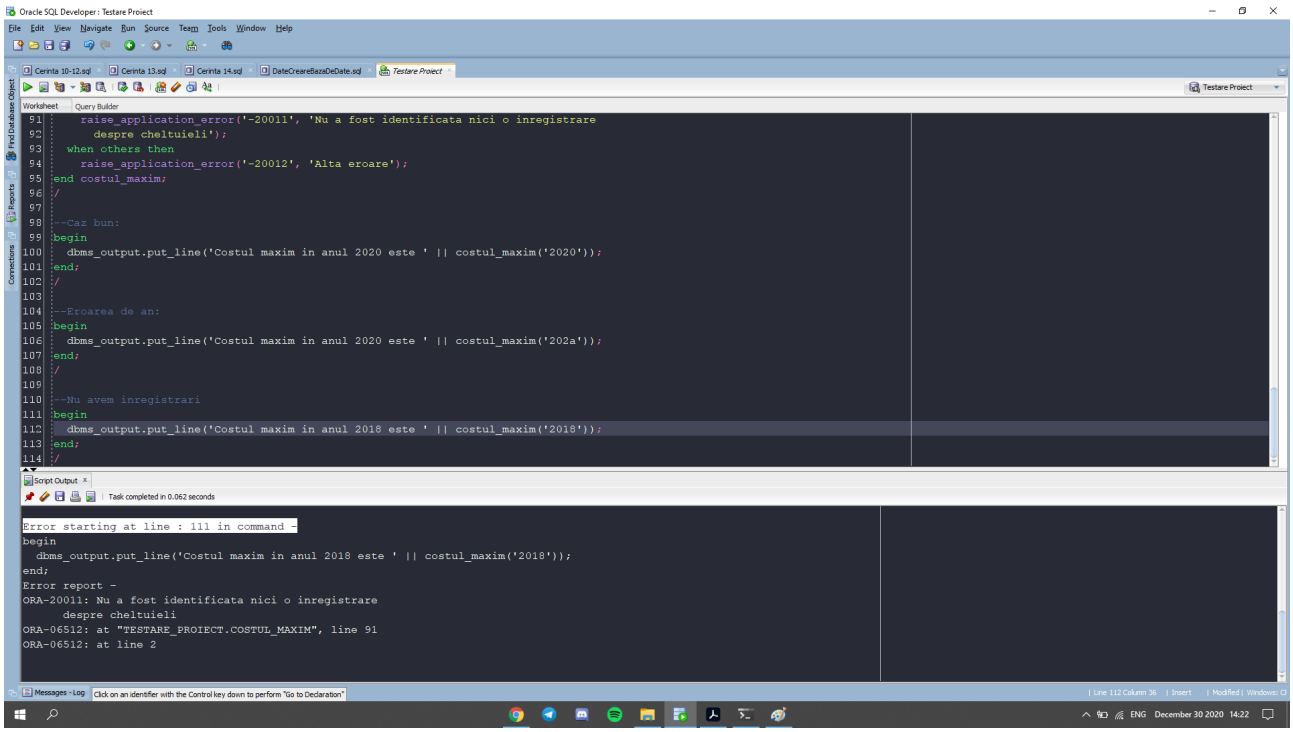

Figure 3.8: Rularea pe cazul cand nu există înregistrări despre an

3.4 Cerința nr. 9

Cerința: Definiți un subprogram stocat de tip procedură care să utilizeze 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Condiția problemei: Să se determine rezervările de locuințe care încă sunt valabile, (clienții încă sunt cazați sau au rezervat locuința) din orașul și țara (care vor fi transmise ca parametru funcției), care au fost efectuate de către angajații ce lucrează în sediile companiei din aceeasi țară și oraș. (Remarcă: ca si in cazul cerințelor anterioare, se vor lua in considerare doar angajatii care sunt agenți imobiliari). Să se afișeze în formatul: "Locuința de la adresa _ este rezervata de pe _ pana pe _". Codul în format .SQL poate fi vizualizat și utilizat pe acest link.

```
--Ducal Nicolae, grupa 234
--Proiect SGBD: Cerinta nr. 9

set serveroutput on;

create or replace procedure informatii_rezervari_locale
  (tara_sol locatie.tara%type, oras_sol locatie.oras%type)
is
  type t_angajati is table of angajat.angajat_id%type;
  angajati_locali t_angajati := t_angajati();

  row_id number := 0;
  good integer;
  lipsa_sediu exception;
  lipsa_agenti exception;
begin
```

```
--Verficam daca exista sedii in acest oras + tara
select count(*) into good
from sediu s, locatie l
where s.locatie_id = l.locatie_id
and l.oras = oras_sol and l.tara = tara_sol;

if good = 0 then
  raise lipsa_sediu;
end if;

--Verificam daca exista agenti imobiliari in sediile din acest oras
select count(*) into good
from sediu s, locatie l, angajat a
where s.locatie_id = l.locatie_id
and s.sediu_id = a.sediu_id
and l.oras = oras_sol and l.tara = tara_sol
and a.job = 'Agent imobiliar';

if good = 0 then
  raise lipsa_agenti;
end if;

--Veriicam rezervarile efectuate de catre ei
select angajat_id bulk collect into angajati_locali
from sediu s, locatie l, angajat a
where s.locatie_id = l.locatie_id
and s.sediu_id = a.sediu_id
and l.oras = oras_sol and l.tara = tara_sol
and a.job = 'Agent imobiliar';

for i in angajati_locali.first..angajati_locali.last loop
  for rez in (select rl.data_cazare d_caz, rl.data_decazare d_decaz,
                  lc.adresa adresa
              from rezervare_locuinta rl, locuinta lo, locatie lc
              where rl.locuinta_id = lo.locuinta_id
              and lc.locatie_id = lo.locatie_id
              and lc.oras = oras_sol and lc.tara = tara_sol
              and rl.angajat_id = angajati_locali(i)) loop
    if row_id = 0 then
      dbms_output.put_line('Locuintele rezervate din ' || oras_sol || ', '
          || tara_sol || ' de catre agentii locali sunt: ');
    end if;
    row_id := row_id + 1;
    dbms_output.put_line(row_id || ') ' || 'Adresa: ' || rez.adresa
        ||  '; Rezervata de pe ' || to_char(rez.d_caz) || ' pana pe '
        || to_char(rez.d_decaz));
  end loop;
end loop;

if row_id = 0 then
```

```
    dbms_output.put_line('Nu exista rezervari in orasul indicat');
  end if;
exception
  when lipsa_sediu then
    raise_application_error('-20021', 'Nu exista nici un sediu la aceasta locatie');
  when lipsa_agenti then
    raise_application_error('-20020', 'Nu lucreaza nici un agent imobiliar la
        aceasta locatie');
  when others then
    raise_application_error('-20020', 'Alta eroare');
end informatii_rezervari_locale;
/

--Varianta valida
begin
  informatii_rezervari_locale('Romania', 'Bucuresti');
end;
/

--Nu exista sediu
begin
  informatii_rezervari_locale('Romania', 'Arad');
end;
/
--Nu exista angajati
begin
  informatii_rezervari_locale('Romania', 'Timisoara');
end;
/
```

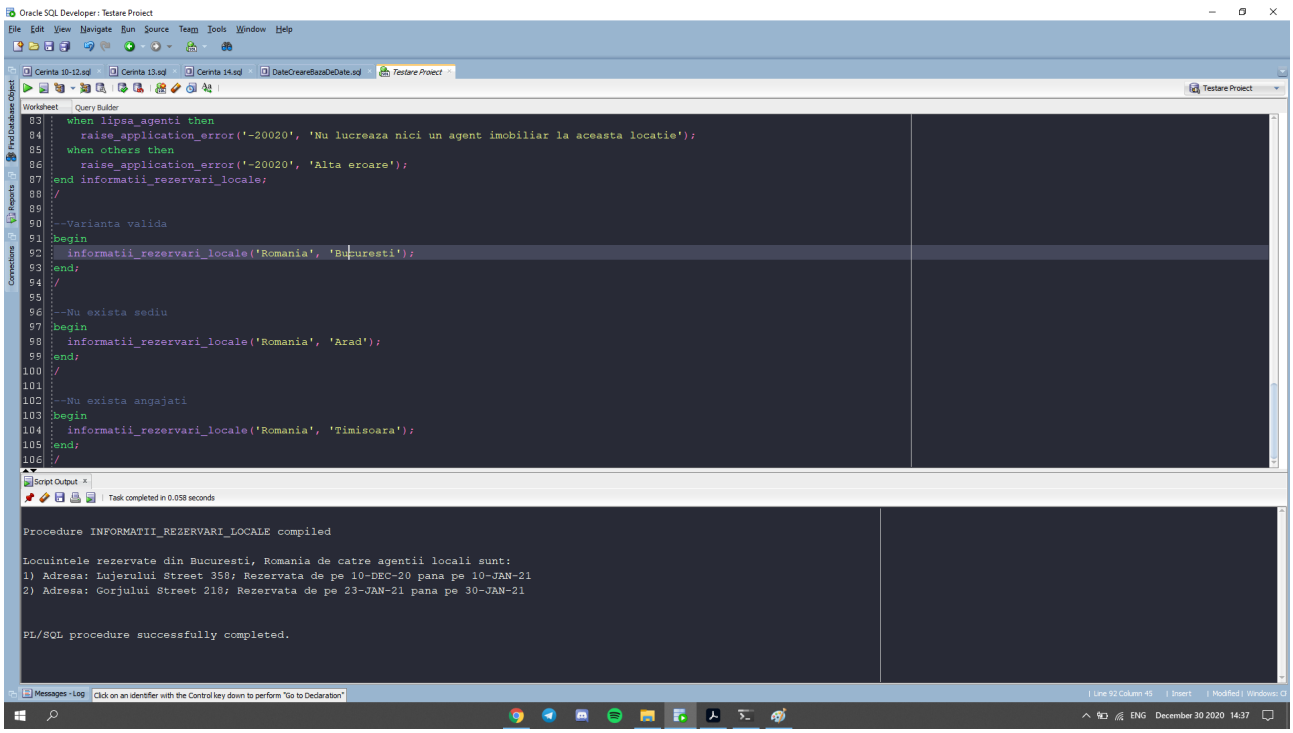

Figure 3.9: Compilarea si rularea pe cazul Romania, Bucuresti

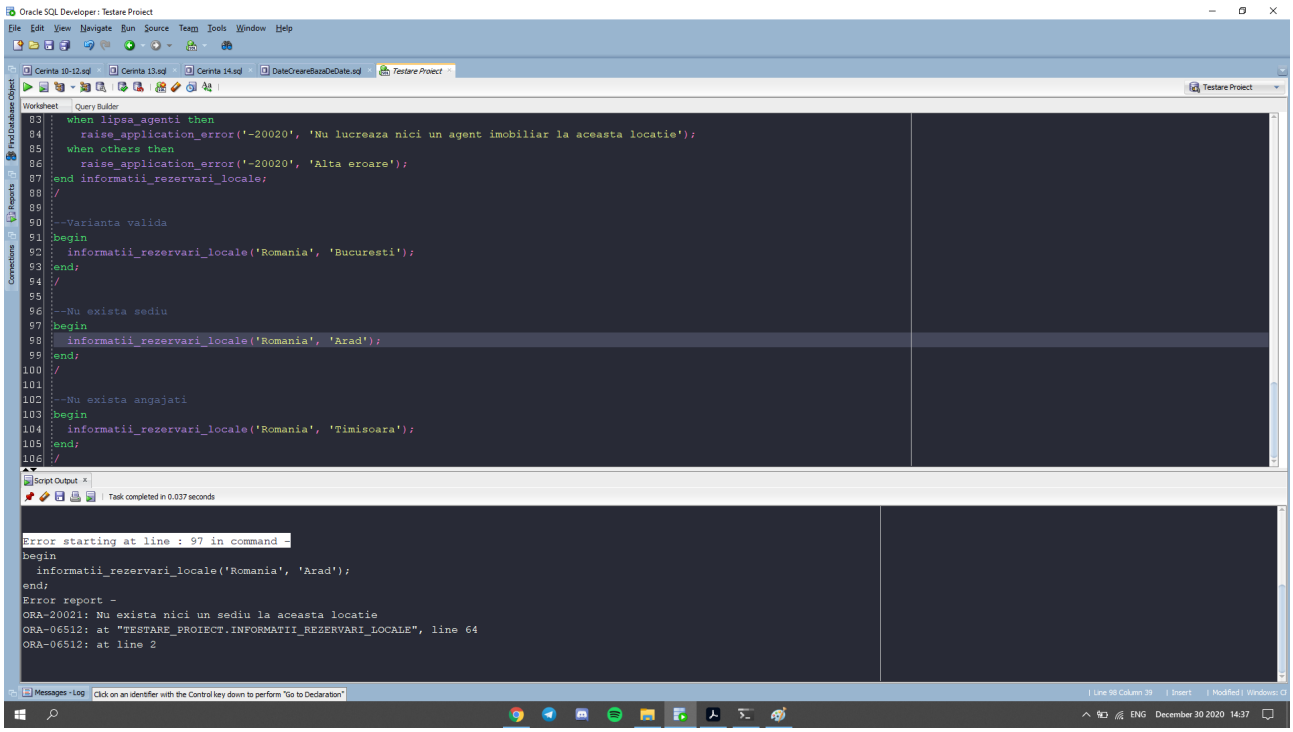

Figure 3.10: Rularea pe cazul cu un oras in care nu exista sediu

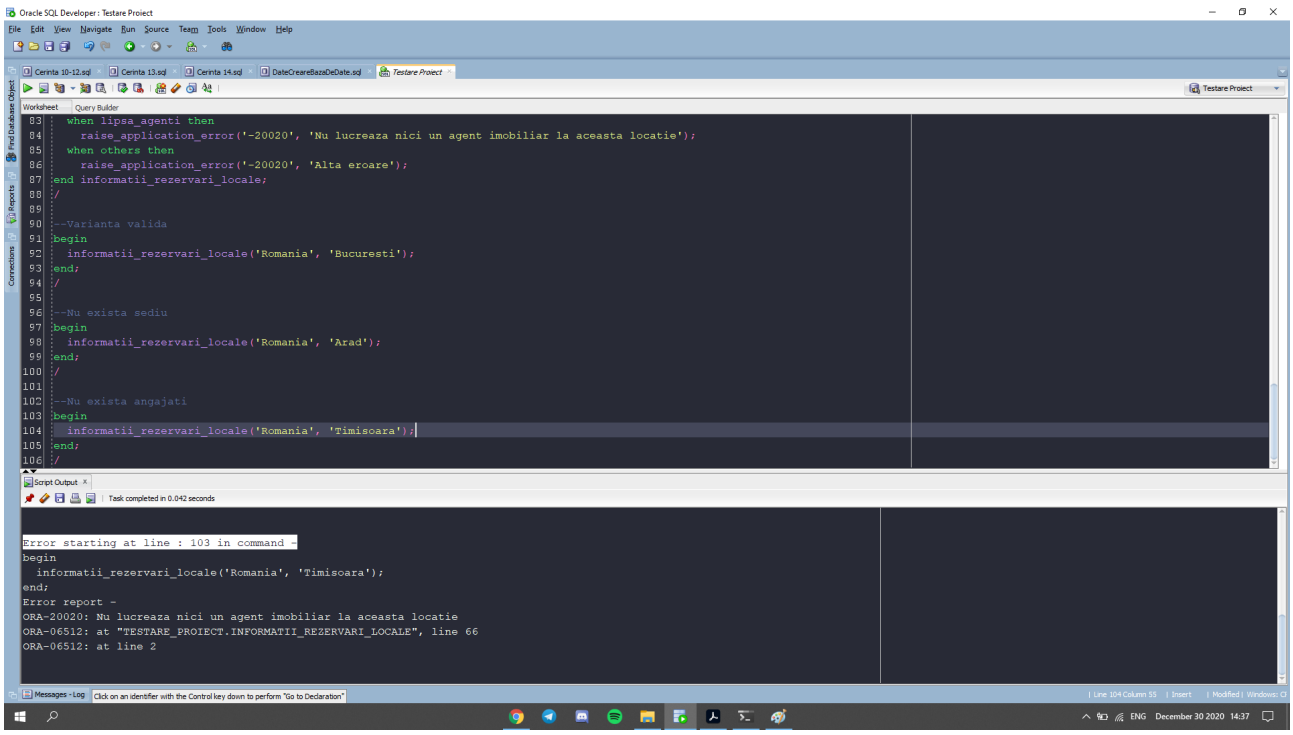

Figure 3.11: Rularea pe cazul cand nu există agenti in sediu

Triggere (Cerințele nr. 10-12)

În această parte a proiectului voi defini câțiva triggeri pentru baza noastră de date. Codul în format .SQL poate fi vizualizat și utilizat pe acest link.

4.1 Trigger LMD la nivel de comandă (Cerința nr. 10)

Cerință: Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul. *Rezolvare:*

4.2 Trigger LMD la nivel de linie (Cerința nr. 11)

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

Rezolvare: Unul din trigger-ele care am putea face este triggerul care ține cont de numărul angajaților dintr-un sediu. Când inserăm un angajat într-un sediu, adăugăm 1, când transferăm un angajat în alt sediu, facem -1 la sediul vechi și +1 la sediul nou, când ștergem, scădem 1. Codul poate fi

```
--Ducal Nicolae, grupa 234
--Proiect SGBD: Cerintele nr. 11

set serveroutput on;

create or replace trigger trig11_sediu
  after insert or update or delete on angajat
  for each row
begin
  if inserting then
    --Adaugam +1 la numarul de angajati din acel sediu
    update sediu
    set numar_angajati = numar_angajati + 1
    where sediu_id = :NEW.sediu_id;
  elsif updating('sediu_id') then
    --Adaugam +1 la noul sediu si -1 la vechiul sediu
    update sediu
    set numar_angajati = numar_angajati - 1
    where sediu_id = :OLD.sediu_id;

    update sediu
    set numar_angajati = numar_angajati + 1
```

```
    where sediu_id = :NEW.sediu_id;
  elsif deleting then
    --Scadem 1 din sediul din care pleaca
    update sediu
    set numar_angajati = numar_angajati - 1
    where sediu_id = :OLD.sediu_id;
  end if;
end;
/

--Verificam
select * from sediu;
select * from angajat;

update angajat
set sediu_id = 10
where angajat_id = 2;

rollback;
```

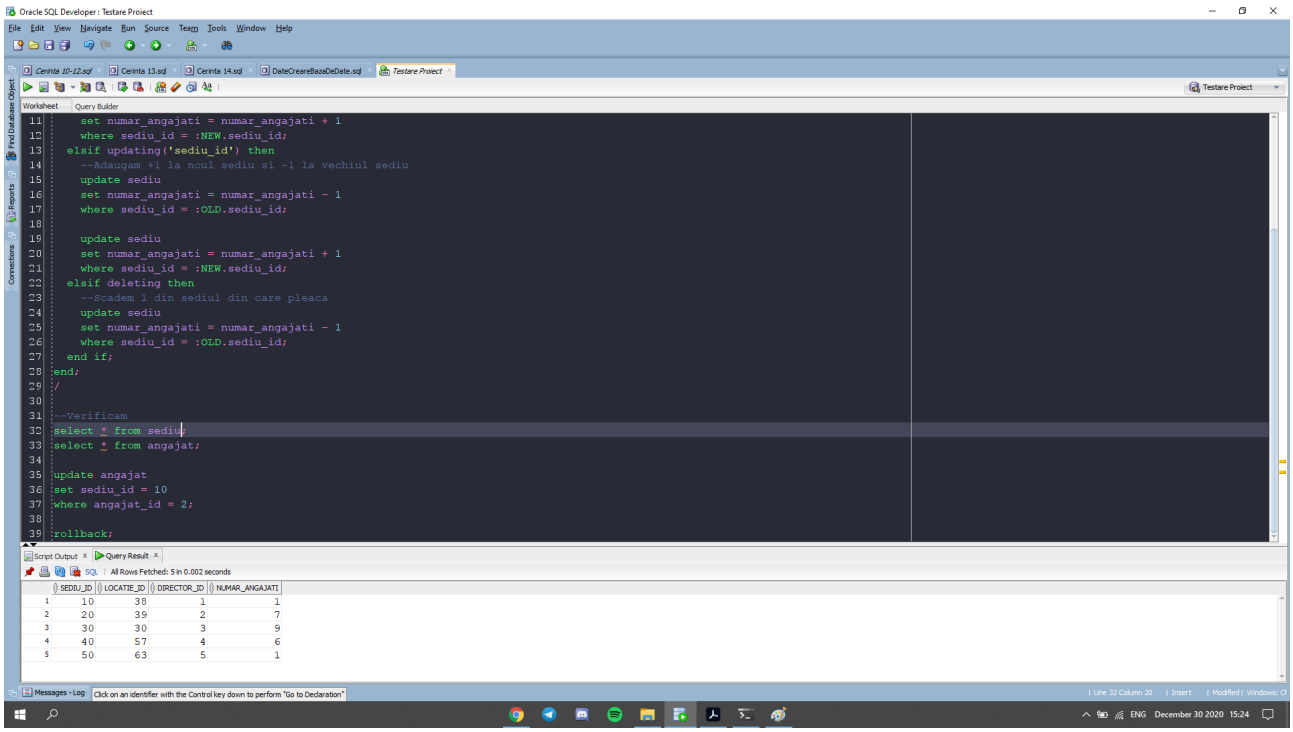

Figure 4.1: Datele despre sedii înainte de update

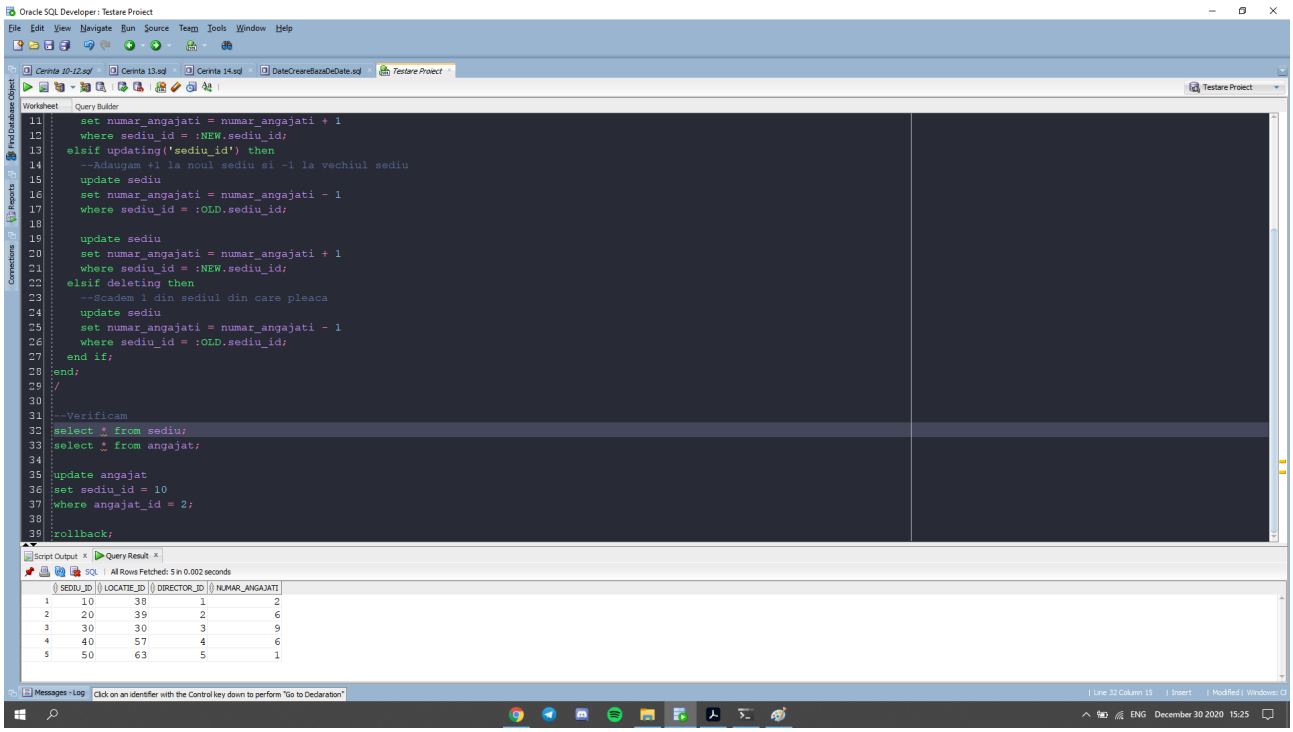

Figure 4.2: Datele despre sedii după update

4.3 Trigger LDD (Cerința nr. 12)

Cerință: Definiți un trigger de tip LDD. Declanșați trigger-ul. *Rezolvare:*

Definirea unui pachet cu subprogramele anterioare (Cerința nr. 13)

5.1 Pachetul Companie_Închirieri

În această parte a proiectului am inclus toate tipurile de date, cursoarele, procedurile și funcțiile în pachetului **Companie_Închirieri**. Codul pentru aceast pachet în format .SQL poate fi accesat la acest link.

```
--Ducal Nicolae, grupa 234
--Proiect SGBD: Cerinta 13

set serveroutput on;

--Partea declarativa
create or replace package companie_inchirieri as
  --Cursoare, tipuri de date
  --Cerinta 6
  type statistica_angajat is record
    (id angajat.angajat_id%type,
     nume angajat.nume%type,
     prenume angajat.prenume%type,
     efectuate number);

  type raport is table of statistica_angajat;

  type indecsi_best is table of number;

  --Cerinta 7
  type linie_cursor_loc is record
    (locuinta_id locuinta.locuinta_id%type,
     adresa locatie.adresa%type,
     oras locatie.oras%type,
     tara locatie.tara%type,
     nume proprietar.nume%type,
     prenume proprietar.prenume%type);
  cursor c_loc(oras_sol locatie.oras%type, tara_sol locatie.tara%type)
    return linie_cursor_loc;
```

```
  type linie_cursor_hot is record
    (hotel_id hotel.hotel_id%type,
     denumire hotel.denumire%type,
     stele hotel.numar_stele%type,
     adresa locatie.adresa%type,
     oras locatie.oras%type,
     tara locatie.tara%type);
  cursor c_hot(oras_sol locatie.oras%type, tara_sol locatie.tara%type)
    return linie_cursor_hot;

  --Cerinta 8
  type linie_id_client is record
    (id client.client_id%type);
  cursor c_cl(v_an varchar2) return linie_id_client;

  --Cerinta 9
  type t_angajati is table of angajat.angajat_id%type;

  --Proceduri, functii
  --Cerinta 6
  procedure raport_anual
    (v_an in varchar2);

  --Cerinta 7
  procedure determina_top
    (tara_sol varchar2 default '%', oras_sol varchar2 default '%');

  --Cerinta 8
  function costul_maxim
    (v_an varchar2)
  return locuinta.pret%type;

  --Cerinta 9
  procedure informatii_rezervari_locale
    (tara_sol locatie.tara%type, oras_sol locatie.oras%type);

end companie_inchirieri;
/
```

```
--Body-ul pachetului
create or replace package body companie_inchirieri as
  --Cursoare, tipuri de date
  --Cerinta 7
  cursor c_loc(oras_sol locatie.oras%type, tara_sol locatie.tara%type)
    return linie_cursor_loc is
    select lo.locuinta_id loc_id, lc.adresa adresa,
      lc.oras oras, lc.tara tara, pr.nume nume, pr.prenume prenume
    from locuinta lo, locatie lc, proprietar pr
    where lo.locatie_id = lc.locatie_id and pr.proprietar_id = lo.proprietar_id
    and lc.tara like(tara_sol) and lc.oras like(oras_sol);

  cursor c_hot(oras_sol locatie.oras%type, tara_sol locatie.tara%type)
    return linie_cursor_hot is
    select h.hotel_id hotel_id, h.denumire denumire, h.numar_stele stele,
      lc.adresa adresa, lc.oras oras, lc.tara tara
    from hotel h, locatie lc
    where h.locatie_id = lc.locatie_id
    and lc.tara like(tara_sol) and lc.oras like(oras_sol);

  --Cerinta 8
  cursor c_cl(v_an varchar2) return linie_id_client is
         select c.client_id id
         from client c, rezervare_locuinta rl
         where rl.client_id = c.client_id and
         to_char(extract(year from rl.data_rezervare)) = v_an
         union
         select c.client_id id
         from client c, istoric_rezervare_locuinta irl
         where irl.client_id = c.client_id and
         to_char(extract(year from irl.data_rezervare)) = v_an
         union
         select c.client_id id
         from client c, rezervare_hotel rh
         where rh.client_id = c.client_id and
         to_char(extract(year from rh.data_rezervare)) = v_an
         union
         select c.client_id id
         from client c, istoric_rezervare_hotel irh
         where irh.client_id = c.client_id and
         to_char(extract(year from irh.data_rezervare)) = v_an;

  --Functiile si procedurile
  --Cerinta 6
  procedure raport_anual
    (v_an in varchar2)
  is
    efect_loc number;
    efect_hot number;
    efect_ist_loc number;
```

```
  efect_ist_hot number;
  cur_stat statistica_angajat;
  rap raport := raport();
  idx integer;
  avg number := 0;
  best_stat number := 0;
  best_angajati indecsi_best := indecsi_best();
begin
  --Vom folosi un cursor ca sa parcurgem prin toti angajatii
  --de tip agent imobiliar
  idx := 1;
  for ang in (select angajat_id, nume, prenume
              from angajat
              where job = 'Agent imobiliar') loop

    --Numarul de rezervari curente de locuinte
    select count(*) into efect_loc
    from rezervare_locuinta
    where angajat_id = ang.angajat_id
    and to_char(extract(year from data_rezervare)) = v_an;

    --Numarul de rezervari de locuinte din istoric
    select count(*) into efect_ist_loc
    from istoric_rezervare_locuinta
    where angajat_id = ang.angajat_id
    and extract(year from data_rezervare) = v_an;

    --Numarul de rezervari curente de hoteluri
    select count(*) into efect_hot
    from rezervare_hotel
    where angajat_id = ang.angajat_id
    and to_char(extract(year from data_rezervare)) = v_an;

    --Numarul de rezervari de hoteluri din istoric
    select count(*) into efect_ist_hot
    from istoric_rezervare_hotel
    where angajat_id = ang.angajat_id
    and extract(year from data_rezervare) = v_an;

    cur_stat.id := ang.angajat_id;
    cur_stat.nume := ang.nume;
    cur_stat.prenume := ang.prenume;
    cur_stat.efectuate := efect_loc + efect_hot + efect_ist_loc + efect_ist_hot;

    rap.extend;
    rap(idx) := cur_stat;
    idx := idx + 1;
  end loop;

  --Afisam raportul:
```

```
  for i in rap.first..rap.last loop
    dbms_output.put_line(i || ') ' || rap(i).nume || ' ' || rap(i).prenume
        || ' a efectuat ' || rap(i).efectuate || ' rezervari.');
    --Vom pastra in best_stat numarul maxim de rezervari efectuate de
    --catre un agent imobiliar
    if best_stat < rap(i).efectuate then
      best_stat := rap(i).efectuate;
    end if;
  end loop;

  --Afisam cei mai buni angajati
  idx := 0;
  for i in rap.first..rap.last loop
    if rap(i).efectuate = best_stat then
      --Pastram indexul din tabelul raport
      idx := idx + 1;
      best_angajati.extend;
      best_angajati(idx) := i;
    end if;
  end loop;

  dbms_output.put_line('');
  if idx = 0 then
    dbms_output.put_line('Nici un angajat nu poate fi considerat
        cel mai bun in acest an');
  else
    if idx = 1 then
      dbms_output.put_line('Cel mai bun agent imobiliar din acest an este '
          || rap(best_angajati(idx)).nume || ' '
          || rap(best_angajati(idx)).prenume);
    else
      dbms_output.put_line('Cei mai buni agenti imobiliari din acest an sunt: ');
      for i in best_angajati.first..best_angajati.last loop
        dbms_output.put_line(i || ') ' || rap(best_angajati(i)).nume || ' '
            || rap(best_angajati(i)).prenume);
      end loop;
    end if;

    --Marim salariile:
    select avg(salariu) into avg
    from angajat
    where job = 'Agent imobiliar';

    for i in best_angajati.first..best_angajati.last loop
      update angajat
      set salariu = floor(salariu + 0.1 * avg)
      where angajat_id = rap(best_angajati(i)).id;
    end loop;
  end if;
end raport_anual;
```

```
--Cerinta 7
procedure determina_top
  (tara_sol varchar2 default '%', oras_sol varchar2 default '%')
is
  v_loc_id locuinta.locuinta_id%type;
  v_adresa locatie.adresa%type;
  v_oras locatie.oras%type;
  v_tara locatie.tara%type;
  v_nume proprietar.nume%type;
  v_prenume proprietar.prenume%type;
  good integer;
  check_oras_tara integer;
  rating_mediu number;
  iter number;

begin
  --Sa vedem daca orasul si tara sunt valide:
  select count(*) into check_oras_tara
  from locatie lc
  where lc.tara like(tara_sol) and lc.oras like(oras_sol);

  if check_oras_tara = 0 then
    raise_application_error('-20001', 'Nu exista date despre locatia indicata');
  end if;

  --Iteram initial prin locuinte
  --Good va fi indicator ca avem sau nu locuinte bune
  --Daca e 0 - nu, daca e 1 - da, daca e 2 - a fost 1 deja
  good := 0;
  open c_loc(oras_sol, tara_sol);
  loop
    fetch c_loc into v_loc_id, v_adresa, v_oras, v_tara, v_nume, v_prenume;
    exit when c_loc%notfound;

    select avg(s.rating) into rating_mediu
    from serviciu_locuinta sl, serviciu s
    where sl.serviciu_id = s.serviciu_id
    and sl.locuinta_id = v_loc_id;

    if nvl(rating_mediu, 0) >= 9.5 then
      if good = 0 and oras_sol = '%' and tara_sol = '%' then
        dbms_output.put_line('Cele mai bune locuinte sunt: ');
      elsif good = 0 and oras_sol = '%' then
        dbms_output.put_line('Cele mai bune locuinte din '
            || tara_sol || ' sunt: ');
      elsif good = 0 then
        dbms_output.put_line('Cele mai bune locuinte din '
            || tara_sol || ', ' || oras_sol || ' sunt: ');
      end if;
```

```
    good := good + 1;

    if oras_sol = '%' and tara_sol = '%' then
      dbms_output.put_line(good || ') ' || v_adresa || ', ' || v_oras
          || ', ' || v_tara);
    elsif oras_sol = '%' then
      dbms_output.put_line(good || ') ' || v_adresa || ', ' || v_oras);
    else
      dbms_output.put_line(good || ') ' || v_adresa);
    end if;

    dbms_output.put_line('Rating: ' || rating_mediu);
    dbms_output.put_line('Proprietar: ' || v_nume || ' ' || v_prenume);
    dbms_output.put('Servicile propuse: ');
    --Folosim un ciclu cursor pentru a prezenta serviciile:
    for i in (select denumire
              from serviciu_locuinta sl, serviciu s
              where sl.serviciu_id = s.serviciu_id
              and sl.locuinta_id = v_loc_id) loop
      dbms_output.put(i.denumire || '; ');
    end loop;
    dbms_output.put_line('');
    dbms_output.put_line('');
  end if;
end loop;
close c_loc;

if good = 0 then
  dbms_output.put_line('Nu au fost identificate locuinte care ar
      indeplini criteriului');
end if;

--Repetam pentru hotel
good := 0;
for hot in c_hot(oras_sol, tara_sol) loop
  select avg(s.rating) into rating_mediu
  from serviciu_hotel sh, serviciu s
  where sh.serviciu_id = s.serviciu_id
  and sh.hotel_id = hot.hotel_id;

  if nvl(rating_mediu, 0) >= 9.5 then
    if good = 0 and oras_sol = '%' and tara_sol = '%' then
      dbms_output.put_line('Cele mai bune hoteluri sunt: ');
    elsif good = 0 and oras_sol = '%' then
      dbms_output.put_line('Cele mai bune hoteluri din '
          || tara_sol || ' sunt: ');
    elsif good = 0 then
      dbms_output.put_line('Cele mai bune hoteluri din '
          || tara_sol || ', ' || oras_sol || ' sunt: ');
```

```
      end if;

      good := good + 1;

      if oras_sol = '%' and tara_sol = '%' then
        dbms_output.put_line(good || ') ' || hot.denumire || ', '
            || hot.stele || ' stele, ' || hot.adresa || ', ' || hot.oras
            || ', ' || hot.tara);
      elsif oras_sol = '%' then
        dbms_output.put_line(good || ') ' || hot.denumire || ', '
            || hot.stele || ' stele, ' || hot.adresa || ', ' || hot.oras);
      else
        dbms_output.put_line(good || ') ' || hot.denumire || ', '
            || hot.stele || ' stele, ' || hot.adresa);
      end if;

      dbms_output.put('Servicile propuse: ');
      --Folosim un ciclu cursor pentru a prezenta serviciile:
      for i in (select denumire
                from serviciu_hotel sh, serviciu s
                where sh.serviciu_id = s.serviciu_id
                and sh.hotel_id = hot.hotel_id) loop
        dbms_output.put(i.denumire || '; ');
      end loop;
      dbms_output.put_line('');
      dbms_output.put_line('');
    end if;
  end loop;

  if good = 0 then
    dbms_output.put_line('Nu au fost identificate hoteluri care ar
        indeplini criteriului');
  end if;
end determina_top;

--Cerinta 8
function costul_maxim
  (v_an varchar2)
return locuinta.pret%type is
  max_cost locuinta.pret%type := 0;
  cur_cost locuinta.pret%type := 0;
  fara_inregistrari exception;
  good integer;
  verifica_an number;
begin
  --Verificam daca e bun anul introdus
  verifica_an := to_number(v_an);
  --Verificam daca avem inregistrari
  good := 0;
  for i in c_cl(v_an) loop
```

```
  good := good + 1;
end loop;
if good = 0 then
  raise fara_inregistrari;
end if;


for cl in c_cl(v_an) loop
  --Pentru toate rezervarile din istoric, e evident ca se va lua suma pentru
  --toate zilele de cazare;
  cur_cost := 0;
  for rez in (select irl.data_cazare d_caz, irl.data_decazare d_decaz,
                lo.pret pret
              from istoric_rezervare_locuinta irl, locuinta lo
              where client_id = cl.id and lo.locuinta_id = irl.locuinta_id) loop
    cur_cost := cur_cost + rez.pret * round(rez.d_decaz - rez.d_caz);
  end loop;

  for rez in (select irh.data_cazare d_caz, irh.data_decazare d_decaz,
                cam.pret pret
              from istoric_rezervare_hotel irh, camera cam
              where client_id = cl.id and cam.camera_id = irh.camera_id) loop
    cur_cost := cur_cost + rez.pret * round(rez.d_decaz - rez.d_caz);
  end loop;

  --Pentru rezervarile care inca sunt valabile si pentru cele cazate,
  --aplicam formulele
  for rez in (select rl.data_cazare d_caz, rl.data_decazare d_decaz,
                rl.status status, lo.pret pret
              from rezervare_locuinta rl, locuinta lo
              where client_id = cl.id and lo.locuinta_id = rl.locuinta_id) loop
    if rez.status = 'Rezervat' then
      cur_cost := cur_cost + 0.5 * rez.pret * round(rez.d_decaz - rez.d_caz);
    elsif rez.status = 'Cazat' then
      cur_cost := cur_cost + rez.pret * round(rez.d_decaz - rez.d_caz);
    end if;
  end loop;

  for rez in (select rh.data_cazare d_caz, rh.data_decazare d_decaz,
                rh.status status, cam.pret pret
              from rezervare_hotel rh, camera cam
              where client_id = cl.id and cam.camera_id = rh.camera_id) loop
    if rez.status = 'Rezervat' then
      cur_cost := cur_cost + 0.5 * rez.pret * round(rez.d_decaz - rez.d_caz);
    elsif rez.status = 'Cazat' then
      cur_cost := cur_cost + rez.pret * round(rez.d_decaz - rez.d_caz);
    end if;
  end loop;

  --Verificam daca e maxime cheltuilelile persoanei in cauza
```

```
    if cur_cost > max_cost then max_cost := cur_cost; end if;
    --dbms_output.put_line(cl.id || ' ' || cur_cost);
  end loop;
return max_cost;
exception
  when value_error then
    raise_application_error('-20010', 'Anul introdus este gresit (nu este numeric)');
  when fara_inregistrari then
    raise_application_error('-20011', 'Nu a fost identificata nici o inregistrare
      despre cheltuieli');
  when others then
    raise_application_error('-20012', 'Alta eroare');
end costul_maxim;

--Cerinta 9
procedure informatii_rezervari_locale
  (tara_sol locatie.tara%type, oras_sol locatie.oras%type)
is
  angajati_locali t_angajati := t_angajati();

  row_id number := 0;
  good integer;
  lipsa_sediu exception;
  lipsa_agenti exception;
begin
  --Verficam daca exista sedii in acest oras + tara
  select count(*) into good
  from sediu s, locatie l
  where s.locatie_id = l.locatie_id
  and l.oras = oras_sol and l.tara = tara_sol;

  if good = 0 then
    raise lipsa_sediu;
  end if;

  --Verificam daca exista agenti imobiliari in sediile din acest oras
  select count(*) into good
  from sediu s, locatie l, angajat a
  where s.locatie_id = l.locatie_id
  and s.sediu_id = a.sediu_id
  and l.oras = oras_sol and l.tara = tara_sol
  and a.job = 'Agent imobiliar';

  if good = 0 then
    raise lipsa_agenti;
  end if;

  --Veriicam rezervarile efectuate de catre ei
  select angajat_id bulk collect into angajati_locali
  from sediu s, locatie l, angajat a
```

```
    where s.locatie_id = l.locatie_id
    and s.sediu_id = a.sediu_id
    and l.oras = oras_sol and l.tara = tara_sol
    and a.job = 'Agent imobiliar';

    for i in angajati_locali.first..angajati_locali.last loop
      for rez in (select rl.data_cazare d_caz, rl.data_decazare d_decaz,
                    lc.adresa adresa
                  from rezervare_locuinta rl, locuinta lo, locatie lc
                  where rl.locuinta_id = lo.locuinta_id
                  and lc.locatie_id = lo.locatie_id
                  and lc.oras = oras_sol and lc.tara = tara_sol
                  and rl.angajat_id = angajati_locali(i)) loop
        if row_id = 0 then
          dbms_output.put_line('Locuintele rezervate din ' || oras_sol
              || ', ' || tara_sol || ' de catre agentii locali sunt: ');
        end if;
        row_id := row_id + 1;
        dbms_output.put_line(row_id || ') ' || 'Adresa: ' || rez.adresa ||
          '; Rezervata de pe ' || to_char(rez.d_caz) || ' pana pe '
          || to_char(rez.d_decaz));
      end loop;
    end loop;

    if row_id = 0 then
      dbms_output.put_line('Nu exista rezervari in orasul indicat');
    end if;
  exception
    when lipsa_sediu then
      raise_application_error('-20021', 'Nu exista nici un sediu la aceasta locatie');
    when lipsa_agenti then
      raise_application_error('-20020', 'Nu lucreaza nici un agent imobiliar
          la aceasta locatie');
    when others then
      raise_application_error('-20020', 'Alta eroare');
  end informatii_rezervari_locale;

end companie_inchirieri;
/
```

Țin să menționez că nu voi prezenta imagini cu rularea fiecărei funcții și proceduri din pachet, pentru că rularea acestora a fost studiată în cerințele precedente. Pentru această cerință voi prezenta doar compilarea pachetului. Pentru a verifica corectitudinea pachetului, puteți vizita link-ul cu fișierul .SQL de pe Github-ul meu.

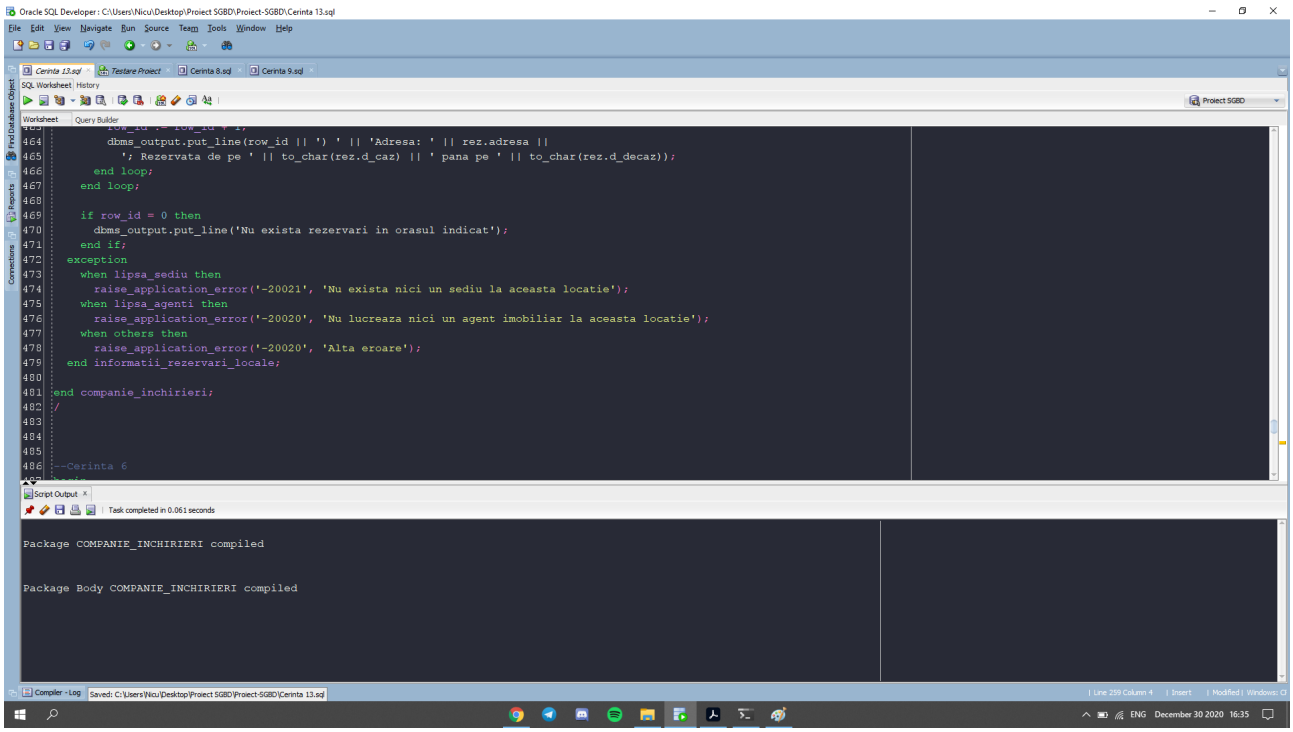

Figure 5.1: Compilarea pachetului

Definirea unui pachet cu elemente complexe (Cerința nr. 14)

În această parte a proiectului am definit pachetul **Companie_Complex** cu unele tipurile de date, cursoare, proceduri și funcți care nu au fost folosite anterior în acest proiect în. Codul pentru aceast pachet în format .SQL poate fi accesat la acest link.

Concluzii

În cadrul acestui proiect am realizat o bază de date a unei companii de închirieri, aplicând și mai mult cunoștințele acumulate la cursul de Baze de Date din primul an și cursul de Sisteme de Gestiune a Bazelor de Date din acest an.