Proiect SGBD

-

Gestiunea unui magazin online

Dragancea Constantin
Grupa 234

Contents

1 Introducere

Pentru acest proiect, am ales să modelez o bază de date a unui magazin online. Aceasta cuprinde:

1. Lista utilizatorilor de pe site
2. Lista produselor, categoriilor și recenziilor produselor de pe site
3. Posibilitatea de a plasa comenzi de mai multe produse și păstrarea informațiilor despre curierul acelei comenzi
4. Lista depozitelor deținute de magazinul online
5. Păstrarea informațiilor despre disponibilitatea unui produs într-un depozit
6. Păstrarea unui tabel cu locații, ce poate fi folosit în mai multe contexte

2 Diagrama E/R

Diagrama Entitate Relație a modelului bazei de date este următoarea:

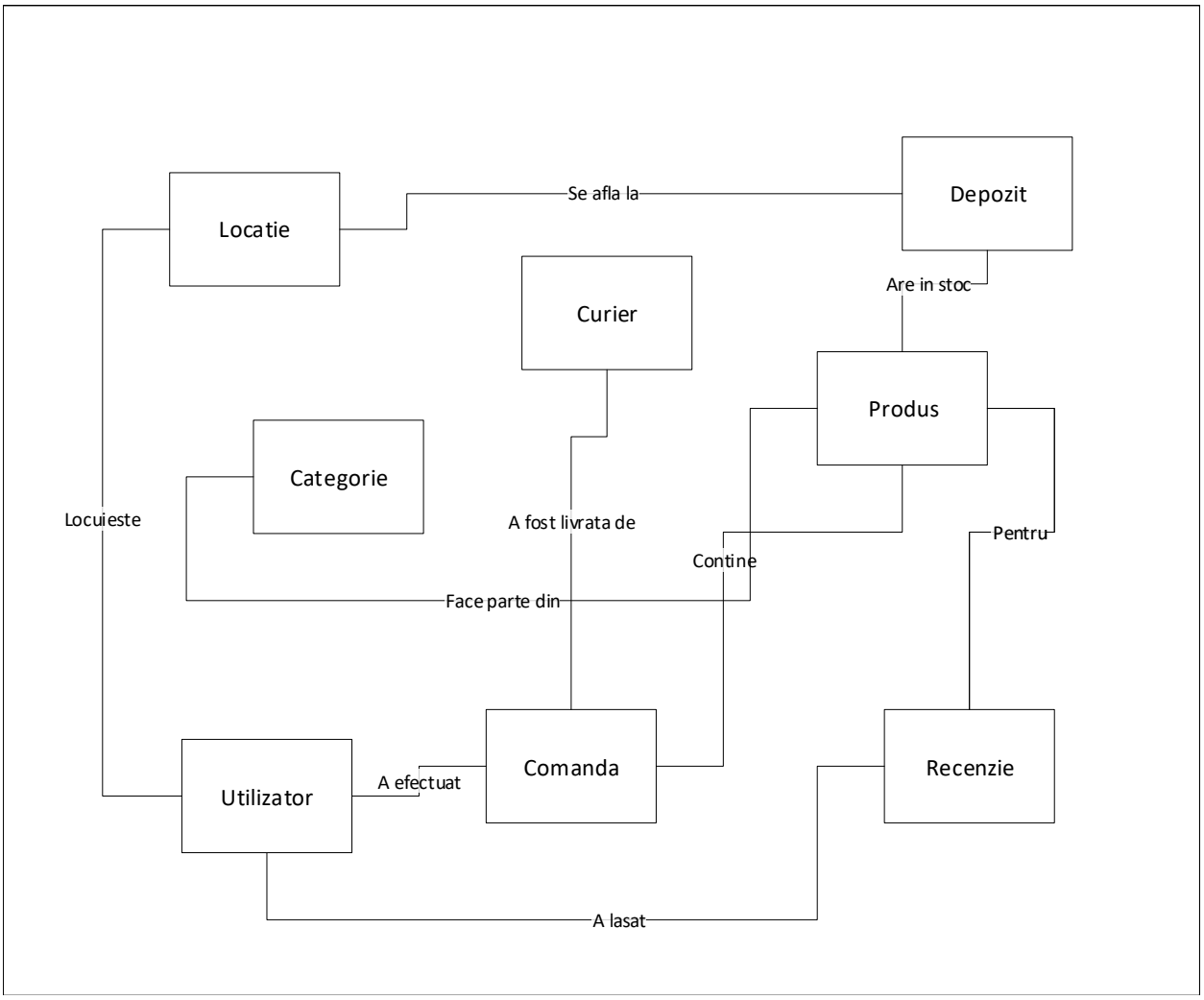

Figure 1: Diagrama Entitate/Relație

3 Diagrama Conceptuală

Mai departe, am construit diagrama conceptuală a modelului, în care se evidențiază tabelele asociative necesare rezolvării relațiilor *many-to-many*, spre deosebire de diagrama E/R.

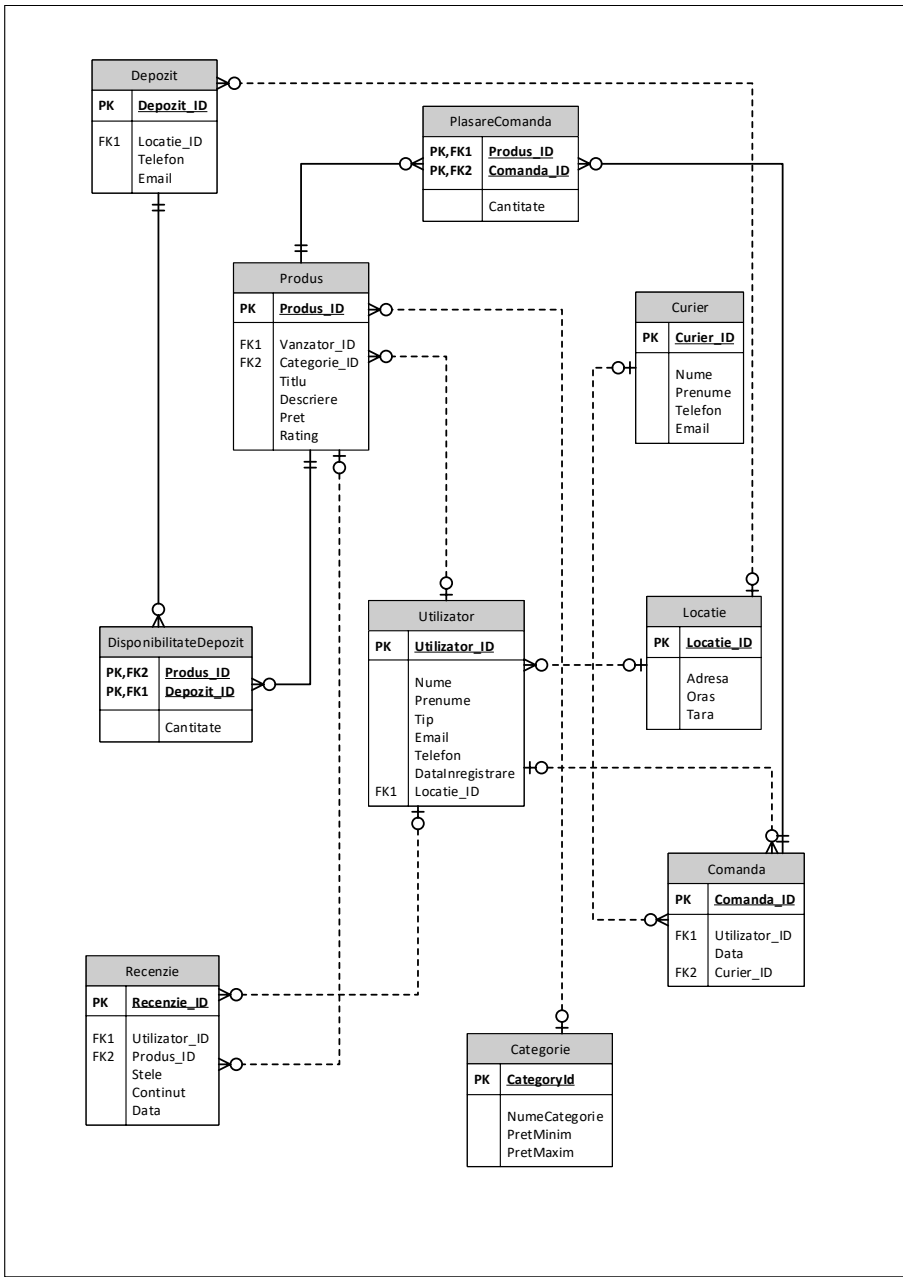

Figure 2: Diagrama Conceptuală

4 Crearea bazei de date

Crearea bazei de date se rezumă la câteva insert-uri simple. Daca ținem cont de ordinea în care creăm tabelele, putem sa definim si constrângerile de *foreign key* în momentul creării tabelelor. Evident, trebuie să creăm tabelele care nu dețin chei străine în componența lor. De exemplu:

```
create table Locatie(
  locatie_id number primary key,
  adresa     varchar2(100),
  oras       varchar2(100),
  tara     varchar2(100)
);
```

Apoi adăugam tabele ce folosesc și chei externe, de exemplu:

```
create table Utilizator(
  utilizator_id   number primary key,
  nume         varchar2(20) not null,
  prenume        varchar2(20),
  tip          varchar2(20) not null,
  email        varchar2(60),
  telefon        varchar2(10),
  DataInregistrare  date,
  locatie_id      number not null,
  foreign key (locatie_id) references Locatie(locatie_id) on
   delete set null
);
```

Iar la final adăugam și tabelele asociative, cum ar fi:

```
create table PlasareComanda(
  produs_id   number,
  comanda_id    number,
  cantitate   number,
  primary key (produs_id, comanda_id),
  foreign key (produs_id) references Produs(produs_id) on
   delete cascade,
  foreign key (comanda_id) references Comanda(comanda_id) on
   delete cascade
);
```

5 Popularea bazei de date

Pentru a demonstra funcționalitatea bazei de date, aș fi putut scrie de mână câteva insert-uri, însă, având ambiții de programator, am decis să găsesc o metodă mai "programatică" de a face acest lucru. Decât să iau date gata făcute de pe internet și să am bătăi de cap cu legile de protecție a datelor personale, voi genera aceste intrări random. Uitându-mă peste diagrama conceptuală, am nevoie să generez următoarele tipuri de informație: nume, prenume, adresă, orașe, țări, numere de telefon, email, dăți calendaristice, nume și descrieri de produse, conținut de recenzii, nume de categorii. Pentru câmpurile formate din șiruri de caractere aș putea să creez efectiv string-uri random, însă ar fi dificil de urmărit si analizat baza de date pentru o ființă umană. Cred că se poate mai bine de atât.

Voi apela la *www.kaggle.com*, un website ce conține datasets din diverse domenii. Aici am găsit liste uriașe cu nume și prenume uzuale în engleză. Astfel, pentru a alcătui numele și prenumele pentru o persoană, voi alege random din aceste 2 liste. Emailul personal îl pot construi după formatul nume.prenume@email.com . Numărul de telefon se generează ușor, atașând la prefixul 07 un număr de 8 cifre generat random.

Pentru adrese, pe același site am găsit un set, la fel de mare, cu adrese de stradă din SUA. Tot pe site, am găsit și o colecție de orașe și țara căreia aparțin. Combinându-le, pot genera o intrare în tabelul *Locatie*.

Pentru produse, am găsit o bază de date cu produse de pe un site de e-commerce indian, unde se afla și descrieri ale produselor. Pentru categorii, pot prelua un set de categorii de pe un magazin online existent, cum ar fi *www.emag.ro*. Pentru recenzii, am găsit pe *kaggle* o colecție de recenzii lăsate pe *Amazon*.

Tot gândindu-mă cum sa generez niște date calendaristice random, am găsit un dataset cu răspunsuri lăsate pe *stackoverflow*, unde era inclus și timestamp-ul la care au fost lăsate, și am decis să le iau de acolo.

A fost o experiență bună să învăț să generez intrări random, pentru că acum pot cu ușurință sa modific cantitatea care va fi generată. Dacă voi avea nevoie să fac niște *stress tests*, adică să văd cât de bine se descurcă baza mea de date din punct de vedere a eficienței timpului și spațiului, pot să o fac cu ușurință.

6 Cerința 6

Pentru cerința 6, am implementat o procedură care va aplica o reducere de 5% asupra produsului cel mai puțin vândut în ultima lună. Logica sa este că magazinul vrea să își maximizeze fluxul de bani ce trece prin el, și va accepta un profit mai mic per unitate, în schimbul unei cantități mai mari vândute. În caz de egalitate, adică dacă există mai multe produse care s-au vândut în aceeași cantitate minimă, reducerea se va aplica tuturor acestor produse. Pentru îndeplinirea acestei proceduri, am mai definit o funcție ce returneaza aceste produse vândute în cantitate minimă. În rezolvarea acestui exercițiu, am folosit un tablou indexat.

```
1  set serveroutput on;
2
3  create or replace procedure AplicaReducere
4  is
5      type tablou      is table of number index by binary_integer
       ;
6      type tablou_imbr is table of number;
7      Produse tablou_imbr;
8
9      function AflaProduse
10     return tablou_imbr
11     is
12         v       tablou;
13         rasp    tablou_imbr:=tablou_imbr();
14         mn      number;
15     begin
16         for prod in (
17             select * from produs
18         ) loop
19             v(prod.produs_id) := 0;
20         end loop;
21
22         for vanzare in (
23             select pc.* from PlasareComanda pc, comanda c
24             where pc.comanda_id = c.comanda_id and
25                 months_between(sysdate, c.data) <= 1
26         ) loop
27             v(vanzare.produs_id) := v(vanzare.produs_id) +
       vanzare.cantitate;
28         end loop;
29
30         mn := v(v.first);
31         for i in v.first .. v.last loop
32             if v(i) < mn then
```

```
                mn := v(i);
                rasp.delete(rasp.first, rasp.last);
                rasp.extend;
                rasp(rasp.last) := i;
            elsif v(i) = mn then
                rasp.extend;
                rasp(rasp.last) := i;
            end if;
        end loop;

        return rasp;
    -- nu poate exista exceptia no data found, deoarece apelam functia
    -- facand un group by inainte
    -- nu poate exista exceptia too many rows deoarece avem where rownum <= 1
    end AflaProduse;

begin
    Produse := AflaProduse;

    for i in Produse.first .. Produse.last loop
        update produs
        set pret = round(0.95 * pret, 2)
        where produs_id = Produse(i);
    end loop;
    commit;
end;
/

Execute AplicaReducere;
```

7 Cerința 7

Magazinul nostru online vrea ca atunci când într-un depozit se află puține unități ale unui produs, să organizeze o promoție de lichidare de stoc, pentru a face loc pentru alte produse. Astfel, are nevoie de o funcție, care la orice moment de timp, să afișeze pentru fiecare produs id-ul depozitului ce conține cantitatea cea mai mică de produs respectiv. Evident ne interesează doar depozitele care au măcar o unitate de produs respectiv. Dacă un produs nu se află în nici o cantitate în vreunul din depozite, nu i se poate organiza o promoție de lichidare de stoc, deci nu ne interesează. În această rezolvare folosesc un *ciclu cursor*.

```
1  create or replace procedure LichidareStoc
2  is
3      type tablou is table of number index by binary_integer;
4      Depozite    tablou;
5      prod_id      number;
6
7      function DepozitCantitateMin(
8          prod_id     number
9      )
10     return number
11     is
12         dep_id  number;
13     begin
14         select depozit_id
15         into dep_id
16         from (
17             select * from DisponibilitateDepozit
18             where produs_id = prod_id
19             order by cantitate
20         )
21         where rownum <= 1;
22
23         return dep_id;
24     -- nu poate exista exceptia no data found, deoarece apelam
       functia
25     -- facand un group by inainte
26     -- nu poate exista exceptia too many rows deoarece avem
       where
27     -- rownum <= 1
28     end DepozitCantitateMin;
29
30 begin
31     for dp in (
32         select produs_id from DisponibilitateDepozit
33         group by produs_id) loop
```

```
34          Depozite(dp.produs_id) := DepozitCantitateMin(dp.
    produs_id);
35      end loop;
36
37      for i in Depozite.first .. Depozite.last loop
38          if Depozite.exists(i) then
39              dbms_output.put_line('Produs id: '|| i||' Depozit
    id: '||Depozite(i));
40          end if;
41      end loop;
42  end;
43  /
44
45  Execute LichidareStoc;
```

8 Cerința 8

Administratorii site-ului vor să știe pentru fiecare categorie de produse, care e depozitul care deține cea mai mare cantitate de produse din acea categorie.

Avem nevoie de tabelele Categorie, Produs, si DisponibilitateDepozit

```
1  -- Functia va returna numarul de Categorii care se regasesc in
       macar un depozit
2  create or replace function IdentificareDepozite
3  return number
4  is
5      type tablou      is table of number index by binary_integer
       ;
6      type tablou_string is table of categorie.NumeCategorie%type
       index by binary_integer;
7      Categorii        tablou;
8      NumeCategorii    tablou_string;
9      dep_id           number;
10     ans              number := 0;
11
12 begin
13     for categ in (
14         select * from categorie
15     ) loop
16         NumeCategorii(categ.categorie_id) := categ.
       NumeCategorie;
17         begin
18             select depozit_id into dep_id
19             from (
20                 select dp.depozit_id, sum(dp.cantitate) as
       numar_produse
21                 from produs p, DisponibilitateDepozit dp
22                 where p.categorie_id = categ.categorie_id and
23                     dp.produs_id = p.produs_id
24                 group by dp.depozit_id
25                 order by numar_produse desc
26             )
27             where rownum <= 1;
28             Categorii(categ.categorie_id) := dep_id;
29         exception
30             -- nu putem avea exceptia too many rows deoarece am
       un
31             -- where rownum <= 1, deci putem avea maxim 1
       rezultat
32             when no_data_found then
```

```
                Categorii(categ.categorie_id) := null;
        end;
    end loop;

    for i in Categorii.first .. Categorii.last loop
        if Categorii(i) is null then
            dbms_output.put_line('Categoria '||NumeCategorii(i)
    ||' nu se regaseste in niciun depozit');
        else
            dbms_output.put_line('Categoria: '|| NumeCategorii(
    i));
            dbms_output.put_line('Depozitul cu id-ul: '||
    Categorii(i));
            ans := ans + 1;
        end if;
    end loop;
    return ans;
end;
/

begin
    dbms_output.put_line('Categorii care se regasesc in cel
    putin 1 depozit: '||IdentificareDepozite());
end;

-- Adaugam o categorie noua, astfel suntem siguri ca nu se
    regaseste
-- in niciun depozit
insert into categorie(categorie_id, NumeCategorie) values((
    select max(categorie_id) from categorie) + 1, 'CategorieNula
    ');
begin
    dbms_output.put_line('Categorii care se regasesc in cel
    putin 1 depozit: '||IdentificareDepozite());
end;
rollback;
```

9 Cerința 9

Magazinul are nevoie de o procedură care să afle valoarea comenzilor plasate de userii dintr-un anumit oraș, în primele k luni de la crearea contului

Avem nevoie de tabelele Locatie, Utilizator, Comanda, Produs si PlasareComanda

```
1  -- Functia va returna numarul de Categorii care se regasesc in
       macar un depozit
2  create or replace procedure VizualizareComenzi(
3      nume_oras    locatie.oras%type,
4      k            integer
5  )
6  is
7      type tip_raspuns is record (utilizator_id   utilizator.
       utilizator_id%type,
8                                  nume            utilizator.nume
       %type,
9                                  prenume         utilizator.
       prenume%type,
10                                 valoare         number);
11     type tablou is table of tip_raspuns;
12     raspuns tablou;
13     cnt     integer;
14
15     function AflaPretProdus(
16         prod_id     produs.produs_id%type
17     ) return produs.pret%type
18     is
19         prod_pret   number;
20     begin
21         select pret into prod_pret
22         from produs
23         where produs_id = prod_id;
24
25         return prod_pret;
26     exception
27         when no_data_found then
28             raise_application_error(-20003, 'Nu exista produs
       cu id-ul dat in baza de date');
29         -- nu putem avea exceptia too many rows doarece
       produs_id este cheie primara
30     end AflaPretProdus;
31
32     function AflaDataInregistrare(
33         u_id    utilizator.utilizator_id%type
```

```
34      ) return utilizator.DataInregistrare%type
35      is
36          dataReg     utilizator.DataInregistrare%type;
37      begin
38          select DataInregistrare into dataReg
39          from utilizator
40          where utilizator_id = u_id;
41
42          return dataReg;
43      exception
44          when no_data_found then
45              raise_application_error(-20000, 'Nu exista
    utilizator cu acest id');
46          -- nu putem avea exceptia too many rows deoarece
    utilizator_id este cheie primara
47      end AflaDataInregistrare;
48
49      function AflaValoareComenzi(
50          u_id    utilizator.utilizator_id%type
51      )
52      return number
53      is
54          suma        number:=0;
55          dataReg     date;
56      begin
57          dataReg := AflaDataInregistrare(u_id);
58          for my_comanda in (
59              select pc.* from comanda c, PlasareComanda pc
60              where utilizator_id = u_id and
61                  months_between(data, dataReg) <= k and
62                  c.comanda_id = pc.comanda_id
63          ) loop
64              suma := suma + my_comanda.cantitate *
    AflaPretProdus(my_comanda.produs_id);
65          end loop;
66          return suma;
67      end AflaValoareComenzi;
68  begin
69      select count(*) into cnt
70      from locatie
71      where oras = nume_oras;
72
73      if cnt = 0 then
74          raise_application_error(-20001, 'Nu exista oras cu
    numele dat in baza de date');
```

```
75      end if;
76
77      if k < 0 then
78          raise_application_error(-20002, 'S-a dat un numar
    negativ de luni ca parametru');
79      end if;
80
81      select u.utilizator_id, u.nume, u.prenume, 0 as valoare
82      bulk collect into raspuns
83      from utilizator u, locatie l
84      where l.oras = nume_oras and
85          l.locatie_id = u.locatie_id;
86
87      dbms_output.put_line('Valoarea comenzilor utilizatorilor
    din orasul '||nume_oras||
88          ' in ultimele '||k||' luni');
89      dbms_output.put_line('Id | Nume | Prenume | Valoarea
    comenzilor');
90      for i in raspuns.first .. raspuns.last loop
91          raspuns(i).valoare := AflaValoareComenzi(raspuns(i).
    utilizator_id);
92          dbms_output.put_line(raspuns(i).utilizator_id||' '||
    raspuns(i).nume||' '
93              ||raspuns(i).prenume||' '||raspuns(i).valoare);
94      end loop;
95  exception
96      when no_data_found then
97          raise_application_error(-20001, 'Nu exista oras cu
    numele dat in baza de date');
98  end;
99  /
100
101 -- trebuie pasat un oras care sa exista, sa consultam mai intai
     tabelul locatie
102 execute VizualizareComenzi('Tochigi', 100);
103
104 -- Dam un oras care nu exista
105 execute VizualizareComenzi('ABCD', 1);
106
107 -- Dam un numar negativ ca parametru pentru numarul de luni
108 execute VizualizareComenzi('Bucuresti', -1);
```

10 Cerința 10

Magazinul își face totalurile activităților economice la sfârșitul fiecărui trimestru. În momentul în care face aceste totaluri, vrem să ne asigurăm că datele sunt consistente, astfel nu vrem ca o cumpărătură făcută să apară într-un calcul, iar în altul nu (dacă de exemplu între completarile fișierelor/tabelelor excel 1 și 2 se mai face o comandă). Soluția este ca în momentul în care se efectueaza aceste totaluri, să fie blocată posbilitatea plasării unei comenzi. Totalurile se fac în cadrul orelor de lucru.

Astfel, în fiecare an, pe 31 martie, 30 iunie, 30 septembrie, 31 decembrie, în intervalul de ore 9:00-17:00, este blocată posibilitatea plasării unei comenzi.

```
1  create or replace trigger SfarsitTrimestru
2  before insert or delete or update on PlasareComanda
3  begin
4      if (to_char(sysdate, 'DD/MM') = '31/03' or
5          to_char(sysdate, 'DD/MM') = '30/06' or
6          to_char(sysdate, 'DD/MM') = '30/09' or
7          to_char(sysdate, 'DD/MM') = '29/12') then
8
9          raise_application_error(-20010, 'Plasarea/Modificarea/
       Stergerea comenzilor
10             este interzise in zilele in care se fac totalurile
       trimestrului!');
11     end if;
12 end;
13 /
14
15 -- Pentru declansare incercam o inserare fie in una din cele 4
       dati de mai sus,
16 -- fie adaugam in if sa verifice pentru ziua curenta
17 insert into PlasareComanda values(1, 3, 1);
```

11 Cerința 11

Vrem să menținem în tabelul de categorii niște contoare a prețului minim, respectiv maxim a unui produs din acea categorie. Astfel, trebuie să avem grijă să modificam aceste campuri când se adaugă un produs nou, sau modificâ unul existent, întrucât nu trebuie să fie grija unui utilizator simplu a bazei de date.

Îl facem trigger de tip after ca să se faca automat check-urile constrângerii de tip *foreign key*

```
create or replace trigger categ_minmax_price
after insert or update on produs
for each row
declare
    min_pr      number;
    max_pr      number;
    categ       categorie%rowtype;
begin
    min_pr := :new.pret;
    max_pr := :new.pret;

    select * into categ
    from categorie
    where categorie_id = :new.categorie_id;

    if nvl(categ.PretMinim, min_pr) < min_pr then
        min_pr := categ.PretMinim;
    end if;

    if nvl(categ.PretMaxim, max_pr) > max_pr then
        max_pr := categ.PretMaxim;
    end if;

    update categorie
    set PretMinim = min_pr,
        PretMaxim = max_pr
    where categorie_id = :new.categorie_id;
end;
/

-- Pentru declansare, inseram un produs nou
insert into Produs(produs_id, vanzator_id, categorie_id, titlu)
values ((select max(produs_id) from produs) + 1, 1, 1, 'Produs
   Test');
rollback;
```

12 Cerința 12

Voi face un sistem de log-uri pentru baza de date. Pentru aceasta voi avea nevoie de un tabel în care să păstrez log-urile, un trigger ce să provoace scrierea în aceste log-uri, un director în care să scriu fișier-ul *txt* de log-uri, privilegiile necesare de scriere, și trigger-ii ce să provoace scrierea in acel fișier text.

Mai întăi acord privilegiile necesare ca *sysdba*.

```
grant execute on utl_file to constantin;
-- permite folosirea pachetului utl_file, ce are definite
   functii de IO cu fisier

grant execute on dbms_sql to constantin;
-- permite folosirea pachetului dbms_sql, pentru a putea
   deschide cursori in modul necesar noua

create directory logs as 'E:\OracleLogs';
-- creaza folder-ul unde vor fi scrise log-urile;

grant read, write on directory logs to constantin;
-- da privilegiile necesare pentru a putea face operatii de IO
   in acel directory
```

Acum trebuie să implementez tabelele în care voi păstra intrările de log-uri și trigger-ul ce va efectua scrierea în acest tabel:

```
create table audit_user(
    nume_bd             varchar2(50),
    user_logat          varchar2(30),
    eveniment           varchar2(100),
    tip_obiect_referit  varchar2(100),
    nume_obiect_referit varchar2(100),
    data                timestamp(3),
    nr_tabele           integer,
    nr_triggere         integer
);
/
create or replace trigger audit_schema
after create or drop or alter on schema
begin
    insert into audit_user values(
        sys.database_name,
        sys.login_user,
        sys.sysevent,
        sys.dictionary_obj_type,
```

```
        sys.dictionary_obj_name,
        systimestamp(3),
        (select count(*) from user_tables),
        (select count(*) from user_triggers)
    );
end;
/
```

Combinând acest trigger cu unul care să scrie într-un fișier aceste log-uri atunci când user-ul iese, sau baza de date se închide sau întâmpină o eroare, o sa avem aceste log-uri fără nevoia de a mai intra înapoi in baza de date.

```
-- vom defini o functie ce o vom utiliza in urmatoarele triggere
-- dupa executarea acestei functii, in directorul si fisierul cu numele specificat, se va afla
-- continutul tabelului audit_schema
create or replace procedure writelogs
is
    fisier  utl_file.file_type;
    p_sql_query varchar2(300):='select
        nume_bd, user_logat, eveniment, tip_obiect_referit,
        nume_obiect_referit, data, nr_tabele, nr_triggere
    from audit_user';
    l_cursor_handle integer;
    l_dummy         number;
    l_rec_tab       dbms_sql.desc_tab;
    l_col_cnt       integer;
    l_current_line  varchar(2047);
    l_current_col   number(16);
    l_record_count  number(16):=0;
    l_column_value  varchar2(300);
    l_print_text    varchar2(300);
begin
    -- deschide fisierul pentru write
    fisier := utl_file.fopen('LOGS', 'logs.txt', 'w', 2047);

    -- deschide un cursor cu selectul din audit_user
    l_cursor_handle := dbms_sql.open_cursor;
    dbms_sql.parse(l_cursor_handle, p_sql_query, dbms_sql.native);
    l_dummy := dbms_sql.execute(l_cursor_handle);

    -- afla numele coloanelor
    dbms_sql.describe_columns(l_cursor_handle, l_col_cnt,
```

```
   l_rec_tab);
31
32      -- append to file column headers
33      l_current_col := l_rec_tab.first;
34      if (l_current_col is not null) then
35          loop
36              dbms_sql.define_column(l_cursor_handle,
   l_current_col, l_column_value, 300);
37              l_print_text := l_rec_tab(l_current_col).col_name
   || ' ';
38              utl_file.put(fisier, l_print_text);
39              l_current_col := l_rec_tab.next(l_current_col);
40              exit when (l_current_col is null);
41          end loop;
42      end if;
43      utl_file.put_line(fisier, ' ');
44
45      -- append data for each row
46      loop
47          exit when dbms_sql.fetch_rows(l_cursor_handle) = 0;
48
49          l_current_line := '';
50          for l_current_col in 1..l_col_cnt loop
51              dbms_sql.column_value(l_cursor_handle,
   l_current_col, l_column_value);
52              l_print_text := l_column_value;
53
54              l_current_line := l_current_line || l_column_value
   || ' ';
55          end loop;
56
57          l_record_count := l_record_count + 1;
58          utl_file.put_line(fisier, l_current_line);
59      end loop;
60
61      utl_file.fclose(fisier);
62      dbms_sql.close_cursor(l_cursor_handle);
63
64  exception
65      when others then
66          -- eliberam resursele de sistem
67          if dbms_sql.is_open(l_cursor_handle) then
68              dbms_sql.close_cursor(l_cursor_handle);
69          end if;
70
```

```
71          if utl_file.is_open(fisier) then
72              utl_file.fclose(fisier);
73          end if;
74
75          dbms_output.put_line(dbms_utility.format_error_stack);
76  end;
77  /
78
79  create or replace trigger logoff_write_logs
80  before logoff on schema
81  begin
82      writelogs;
83  end;
84  /
85
86  create or replace trigger log_erori
87  after servererror on schema
88  begin
89      writelogs;
90  end;
91  /
92
93  create or replace trigger shutdown_write_logs
94  before shutdown on schema
95  begin
96      writelogs;
97  end;
98  /
```

13 Cerința 13

```
create or replace package proiect
is
    type tablou          is table of number index by
   binary_integer;
    type tablou_imbr     is table of number;
    type tip_rasp_com    is record(
        utilizator_id    utilizator.utilizator_id%type,
        nume             utilizator.nume%type,
        prenume          utilizator.prenume%type,
        valoare          number);
    type tablou_raspuns is table of tip_rasp_com;

    procedure AplicaReducere;

    function AflaProduse return tablou_imbr;

    procedure LichidareStoc;

    function DepozitCantitateMin(
        prod_id     number
    ) return number;

    procedure VizualizareComenzi(
        nume_oras   locatie.oras%type,
        k           integer);

    function AflaPretProdus(
        prod_id     produs.produs_id%type
    ) return produs.pret%type;

    function AflaValoareComenzi(
        u_id        utilizator.utilizator_id%type,
        k           integer
    ) return number;
end proiect;
/

create or replace package body proiect
is
    function AflaProduse
    return tablou_imbr
    is
```

```
42          v       tablou;
43          rasp    tablou_imbr:=tablou_imbr();
44          mn      number;
45      begin
46          for prod in (
47              select * from produs
48          ) loop
49              v(prod.produs_id) := 0;
50          end loop;
51
52          for vanzare in (
53              select pc.* from PlasareComanda pc, Comanda c
54              where pc.comanda_id = c.comanda_id and
55                  months_between(sysdate, c.data) <= 1
56          ) loop
57              v(vanzare.produs_id) := v(vanzare.produs_id) +
    vanzare.cantitate;
58          end loop;
59
60          mn := v(v.first);
61          for i in v.first .. v.last loop
62              if v(i) < mn then
63                  rasp.delete(rasp.first, rasp.last);
64                  rasp.extend;
65                  rasp(rasp.last) := i;
66              elsif v(i) = mn then
67                  rasp.extend;
68                  rasp(rasp.last) := i;
69              end if;
70          end loop;
71
72          return rasp;
73      end Aflaproduse;
74
75      procedure AplicaReducere
76      is
77          Produse     tablou_imbr;
78      begin
79          Produse := AflaProduse;
80
81          for i in Produse.first .. Produse.last loop
82              update produs
83              set pret = round(0.95 * pret, 2)
84              where produs_id = Produse(i);
85          end loop;
```

```
86          commit;
87      end AplicaReducere;
88
89      function DepozitCantitateMin(
90          prod_id     number
91      ) return number
92      is
93          dep_id  number;
94      begin
95          select depozit_id into dep_id
96          from (
97              select * from DisponibilitateDepozit
98              where produs_id = prod_id
99              order by cantitate
100         )
101         where rownum <= 1;
102
103         return dep_id;
104     exception
105         when no_data_found then
106             raise_application_error(-20020, 'Produsul dat nu se
    gaseste in
107                 niciun depozit');
108     end DepozitcantitateMin;
109
110     procedure LichidareStoc
111     is
112         Depozite    tablou;
113         prod_id     number;
114     begin
115         for dp in (
116             select produs_id from DisponibilitateDepozit
117             group by produs_id) loop
118
119             Depozite(dp.produs_id) := DepozitCantitateMin(dp.
    produs_id);
120         end loop;
121
122         for i in Depozite.first .. Depozite.last loop
123             if Depozite.exists(i) then
124                 dbms_output.put_line('Produs id: '||i||'
    Depozit id: '||Depozite(i));
125             end if;
126         end loop;
127     end LichidareStoc;
```

```
128
129     function AflaPretProdus(
130         prod_id     produs.produs_id%type
131     ) return produs.pret%type
132     is
133         prod_pret   number;
134     begin
135         select pret into prod_pret
136         from produs
137         where produs_id = prod_id;
138
139         return prod_pret;
140     exception
141         when no_data_found then
142             raise_application_error(-20021, 'Nu exista produs
    cu id-ul dat');
143     end AflaPretProdus;
144
145     function AflaDataInregistrare(
146         u_id    utilizator.utilizator_id%type
147     ) return utilizator.DataInregistrare%type
148     is
149         dataReg     utilizator.DataInregistrare%type;
150     begin
151         select DataInregistrare into dataReg
152         from utilizator
153         where utilizator_id = u_id;
154
155         return dataReg;
156     exception
157         when no_data_found then
158             raise_application_error(-20022, 'Nu exista
    utilizator cu acest id');
159     end AflaDataInregistrare;
160
161     function AflaValoareComenzi(
162         u_id    utilizator.utilizator_id%type,
163         k       integer
164     ) return number
165     is
166         suma    number:=0;
167         dataReg date;
168     begin
169         dataReg := AflaDataInregistrare(u_id);
170         for my_comanda in (
```

```
171             select pc.* from comanda c, PlasareComanda pc
172             where utilizator_id = u_id and
173                 months_between(data, dataReg) <= k and
174                 c.comanda_id = pc.comanda_id
175         ) loop
176             suma := suma + my_comanda.cantitate *
    AflaPretProdus(my_comanda.produs_id);
177         end loop;
178
179         return suma;
180     end AflaValoareComenzi;
181
182     procedure VizualizareComenzi(
183         nume_oras       locatie.oras%type,
184         k               integer)
185     is
186         raspuns     tablou_raspuns;
187         cnt         integer;
188     begin
189         select count(*) into cnt
190         from locatie
191         where oras = nume_oras;
192
193         if cnt = 0 then
194             raise_application_error(-20023, 'Nu exista oras cu
    numele dat in baza de date');
195         end if;
196
197         if k < 0 then
198             raise_application_error(-20024, 'S-a dat un numar
    negativ de luni ca parametru');
199         end if;
200
201         select u.utilizator_id, u.nume, u.prenume, 0 as valoare
202         bulk collect into raspuns
203         from utilizator u, locatie l
204         where l.oras = nume_oras and
205             l.locatie_id = u.locatie_id;
206
207         dbms_output.put_line('Valoarea comenzilor
    utilizatorilor din orasul '||nume_oras||
208             ' in ultimele '||k||' luni');
209         dbms_output.put_line('Id | Nume | Prenume | Valoarea
    comenzilor');
210         for i in raspuns.first .. raspuns.last loop
```

```
211             raspuns(i).valoare := AflaValoareComenzi(raspuns(i)
    .utilizator_id, k);
212             dbms_output.put_line(raspuns(i).utilizator_id||' '
    || raspuns(i).nume||' '
213                 ||raspuns(i).prenume||' '||raspuns(i).valoare);
214         end loop;
215     exception
216     when no_data_found then
217         raise_application_error(-20001, 'Nu exista oras cu
    numele dat in baza de date');
218     end VizualizareComenzi;
219 end proiect;
220 /
```

14 Cerința 14

Pentru cerința 14 am scris un pachet capabil să scrie conținutul unui tabel specificat într-un fișier specificat.

```
-- Cerinta 14

-- Un pachet ce contine functiile necesare pentru a scrie
   continutul unui tabel intr-un fisier

create or replace package wfile
is
    procedure write_to_file(
        nume_tabel      user_tables.table_name%type,
        nume_fisier     varchar2
    );

    procedure check_table_exists(
        nume_tabel  user_tables.table_name%type
    );
end wfile;
/

create or replace package body wfile
is
    procedure check_table_exists(
        nume_tabel   user_tables.table_name%type
    )
    is
        cnt integer;
    begin
        select count(*) into cnt
        from user_tables
        where lower(table_name) = lower(nume_tabel);

        if (cnt = 0) then
            raise_application_error(-20050, 'Nu exista acest
   tabel in user space');
        end if;
        return;
    end check_table_exists;

    procedure write_to_file(
        nume_tabel      user_tables.table_name%type,
        nume_fisier     varchar2)
```

```
39      is
40          fisier           utl_file.file_type;
41          p_sql_query      varchar2(300):='select * from ';
42          l_cursor_handle  integer;
43          l_dummy          number;
44          l_rec_tab        dbms_sql.desc_tab;
45          l_col_cnt        integer;
46          l_current_line   varchar(2047);
47          l_current_col    number(16);
48          l_record_count   number(16):=0;
49          l_column_value   varchar2(300);
50          l_print_text     varchar2(300);
51      begin
52          check_table_exists(nume_tabel);
53
54          p_sql_query := p_sql_query || nume_tabel;
55          -- deschide fisierul pentru write
56          -- LOGS reprezinta un directory creat din sql, ce are
    incorporata si adresa pe disk unde se va scrie fisierul
57          -- pentru a scrie in alta locatie, trebuie sa declaram
    acel directory, si sa acordam drepturi de read-write
58          -- utilizatorului pentru acel fisier
59          fisier := utl_file.fopen('LOGS', nume_fisier, 'w',
    2047);
60
61          -- deschide un cursor cu selectul din audit_user
62          l_cursor_handle := dbms_sql.open_cursor;
63          dbms_sql.parse(l_cursor_handle, p_sql_query, dbms_sql.
    native);
64          l_dummy := dbms_sql.execute(l_cursor_handle);
65
66          -- afla numele coloanelor
67          dbms_sql.describe_columns(l_cursor_handle, l_col_cnt,
    l_rec_tab);
68
69          -- append to file column headers
70          l_current_col := l_rec_tab.first;
71          if (l_current_col is not null) then
72              loop
73                  dbms_sql.define_column(l_cursor_handle,
    l_current_col, l_column_value, 300);
74                  l_print_text := l_rec_tab(l_current_col).
    col_name || ' ';
75                  utl_file.put(fisier, l_print_text);
76                  l_current_col := l_rec_tab.next(l_current_col);
```

```
77                  exit when (l_current_col is null);
78              end loop;
79          end if;
80          utl_file.put_line(fisier, ' ');
81
82          -- append data for each row
83          loop
84              exit when dbms_sql.fetch_rows(l_cursor_handle) = 0;
85
86              l_current_line := '';
87              for l_current_col in 1..l_col_cnt loop
88                  dbms_sql.column_value(l_cursor_handle,
    l_current_col, l_column_value);
89                  l_print_text := l_column_value;
90
91                  l_current_line := l_current_line ||
    l_column_value || ' ';
92              end loop;
93
94              l_record_count := l_record_count + 1;
95              utl_file.put_line(fisier, l_current_line);
96          end loop;
97
98          utl_file.fclose(fisier);
99          dbms_sql.close_cursor(l_cursor_handle);
100
101      exception
102          when others then
103              -- eliberam resursele de sistem
104              if dbms_sql.is_open(l_cursor_handle) then
105                  dbms_sql.close_cursor(l_cursor_handle);
106              end if;
107
108              if utl_file.is_open(fisier) then
109                  utl_file.fclose(fisier);
110              end if;
111
112              dbms_output.put_line(dbms_utility.
    format_error_stack);
113      end write_to_file;
114
115 end wfile;
116 /
117
118
```

```
119 Execute wfile.write_to_file('audit_user', 'test.txt');
120 /
```

15 Concluzii

În acest proiect am reușit sa modelez activitatea unui magazin online și să valorific cunoștințele mele în *SQL* și *PL/SQL*. Am implementat proceduri și funcții, am folosit tipuri de date și obiecte complexe precum record, tabel index, tabel imbricat și cursor.

Alături de acest pdf, se vor mai găsi și fișierele aferente bazei de date:

- *createDatabase.sql* - crearea tabelelor, constrângerilor și trigger-ilor necesare modelului
- *populateDatabase.sql* - script-ul de populare a bazei de date cu niște date exemplu, pentru a putea testa subprogramele create
- *Cerintai.sql* - fișierul ce conține implementarea cerinței i
- *PopulateDatabase.py* - un script *Python* folosit pentru adăugarea inserărilor random în baza de date.
- *PopulateDatabase.sql* - script-ul *SQL* produs de cel de mai sus ce conține insert-urile propriu-zise

16 Screenshots

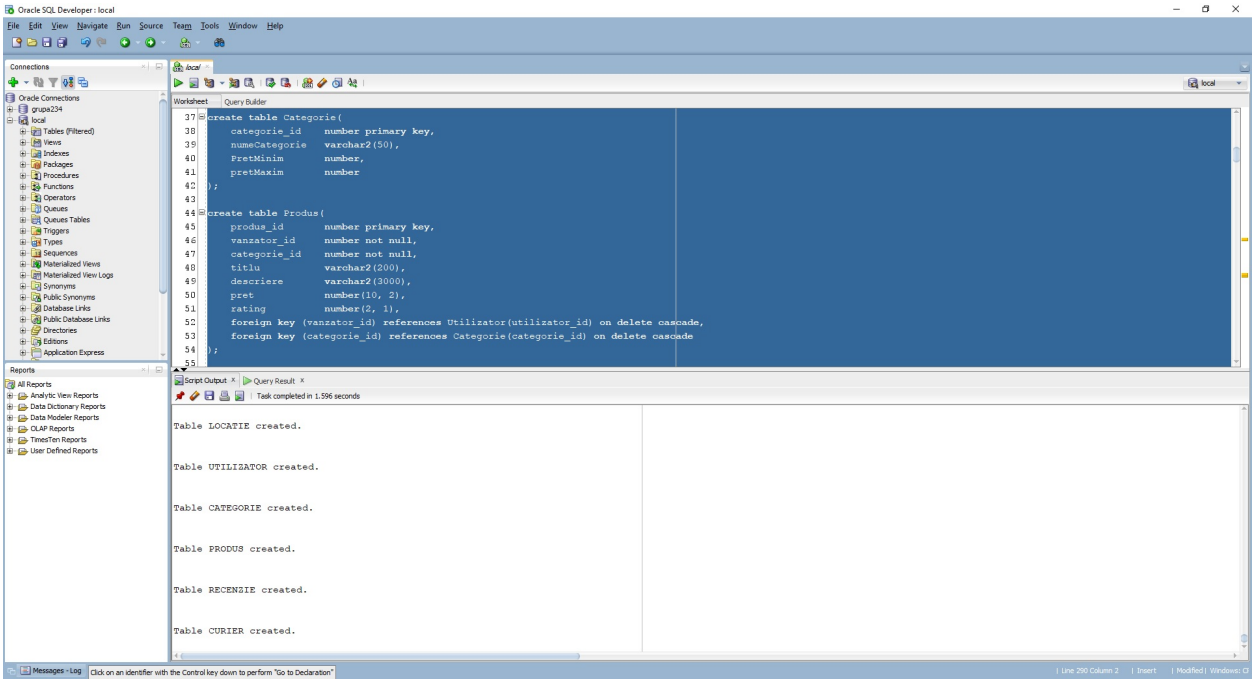

Figure 3: Crearea bazei de date

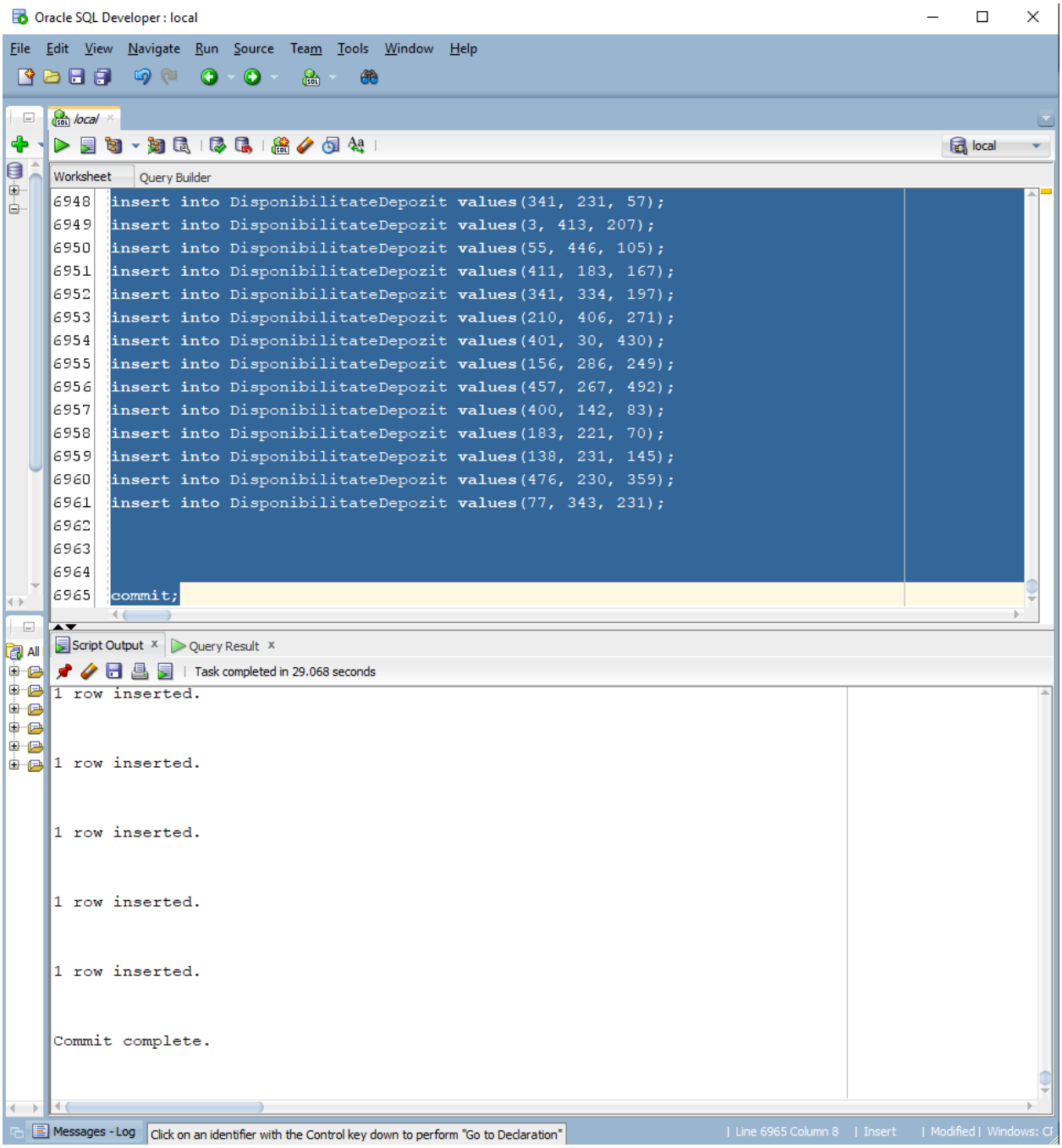

Figure 4: Popularea bazei de date

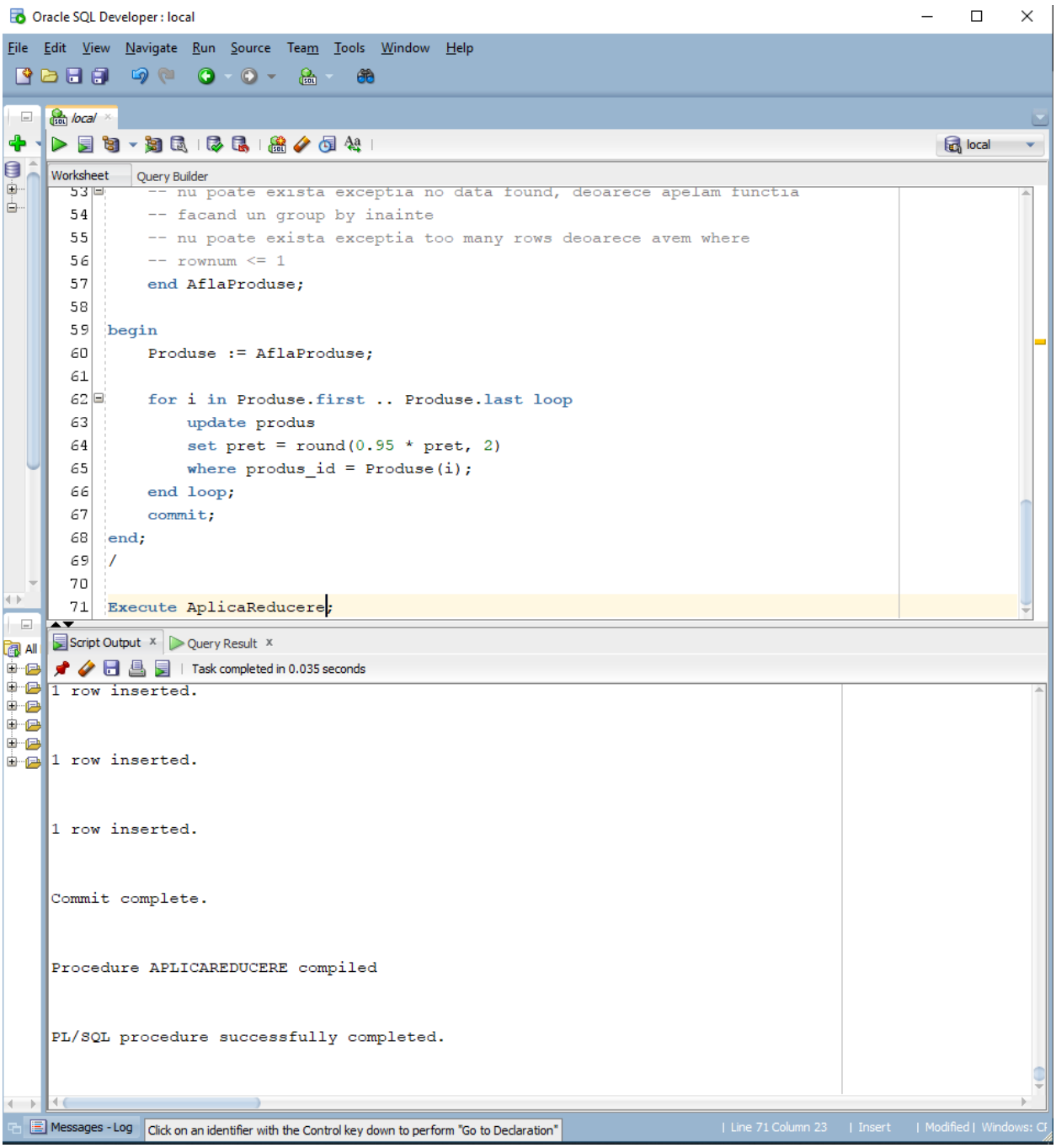

Figure 5: Execuția cerinței 6

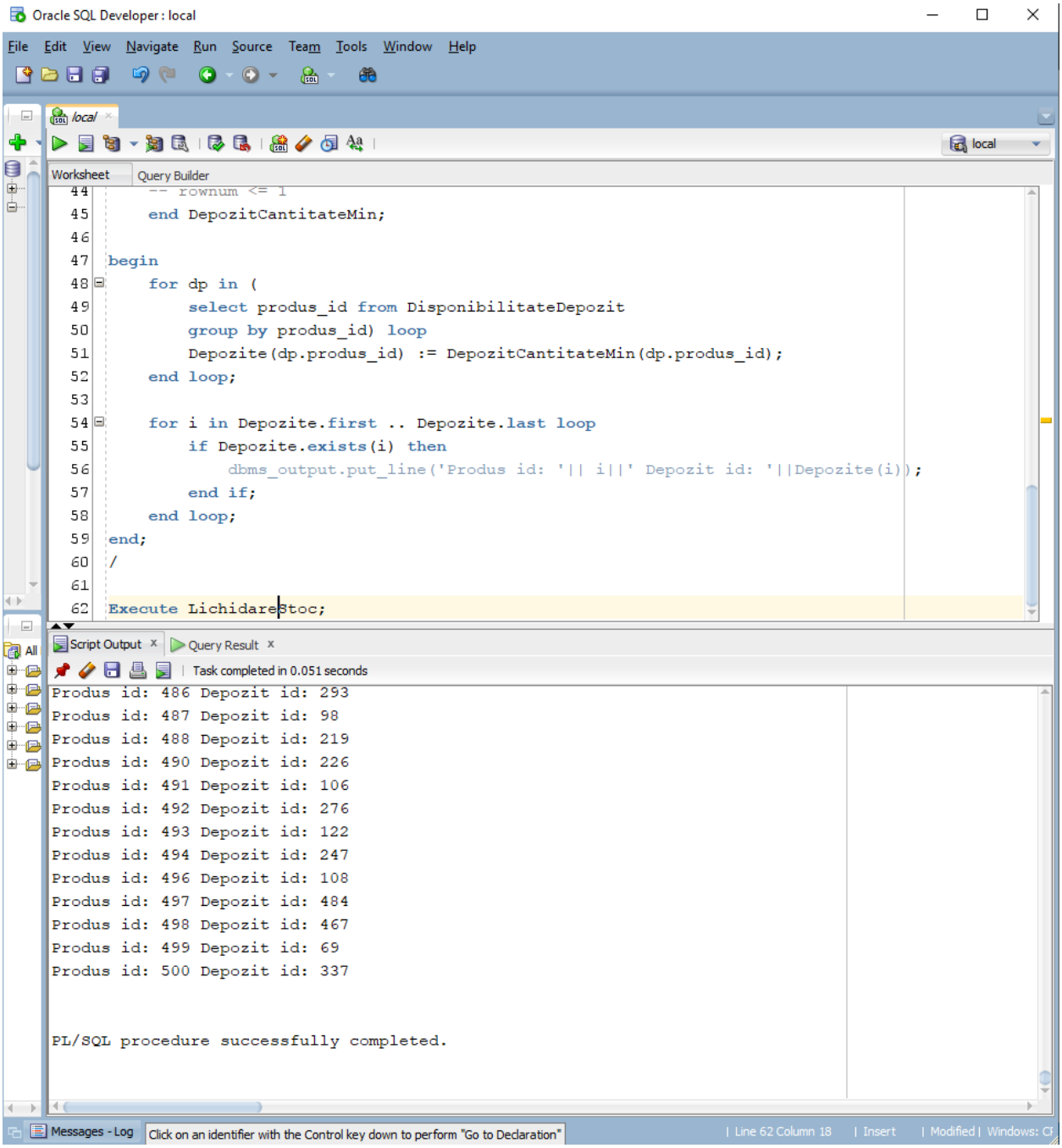

Figure 6: Execuția cerinței 7

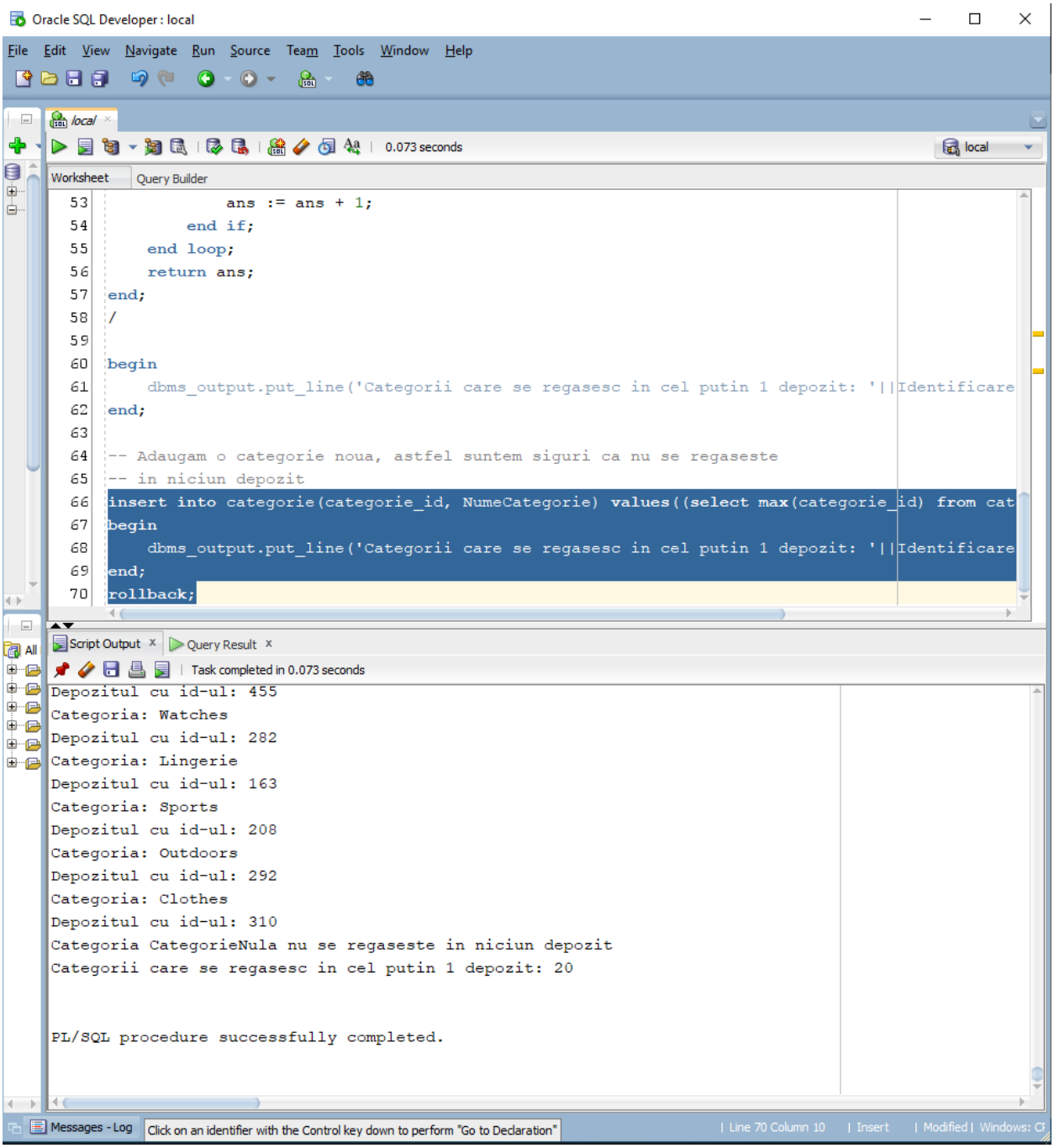

Figure 7: Execuția cerinței 8

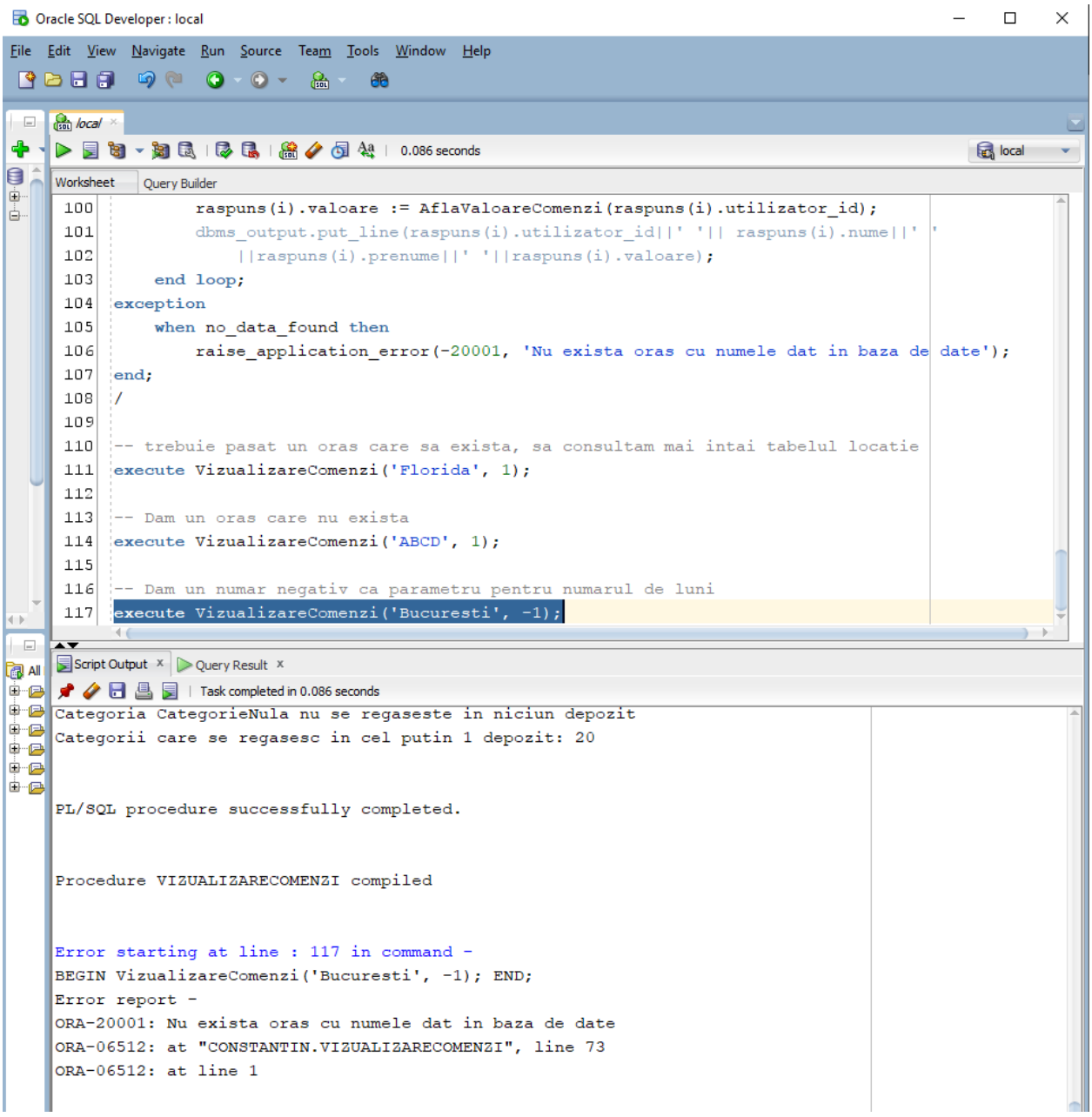

Figure 8: Execuția cerinței 9

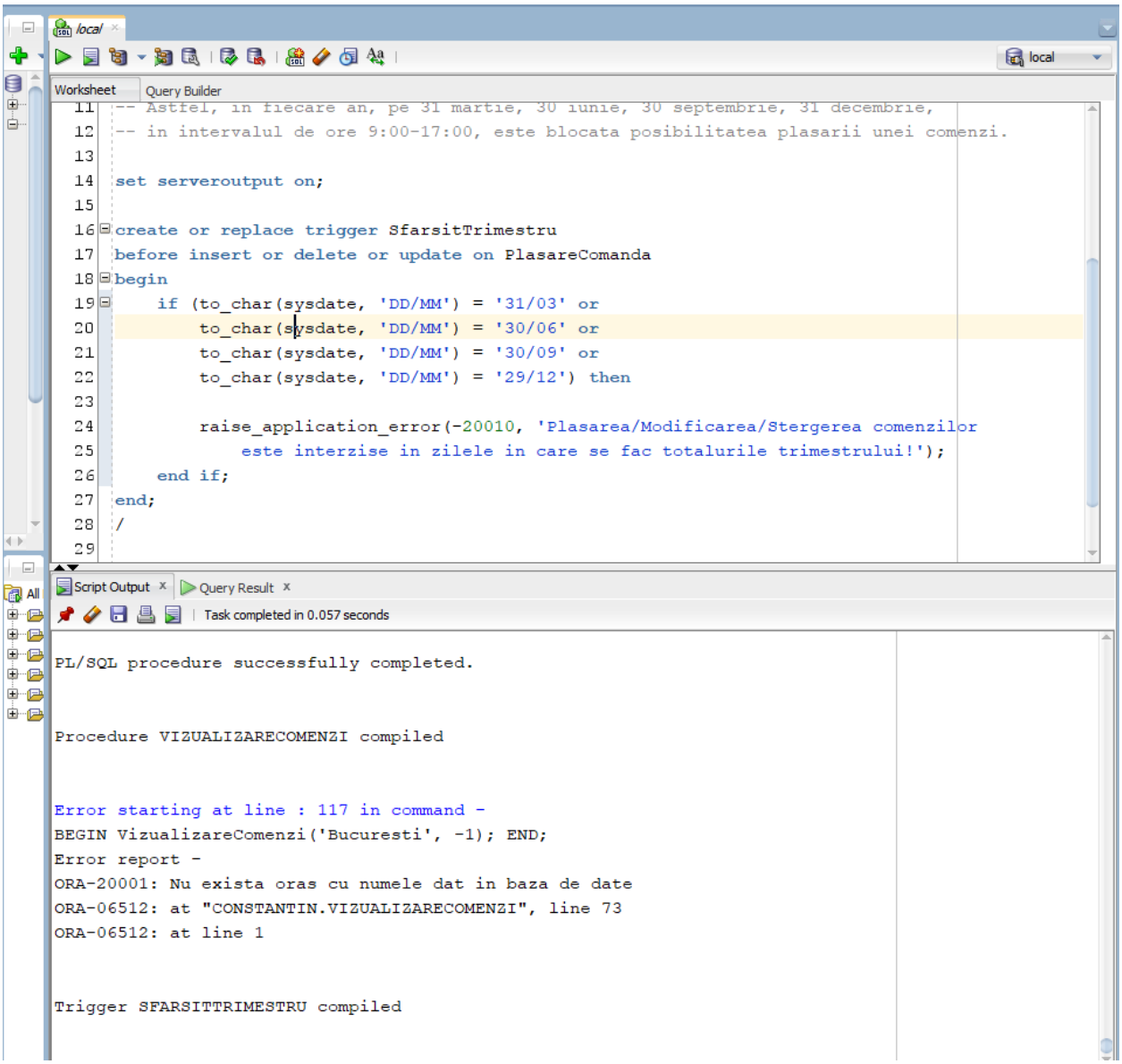

Figure 9: Execuția cerinței 10

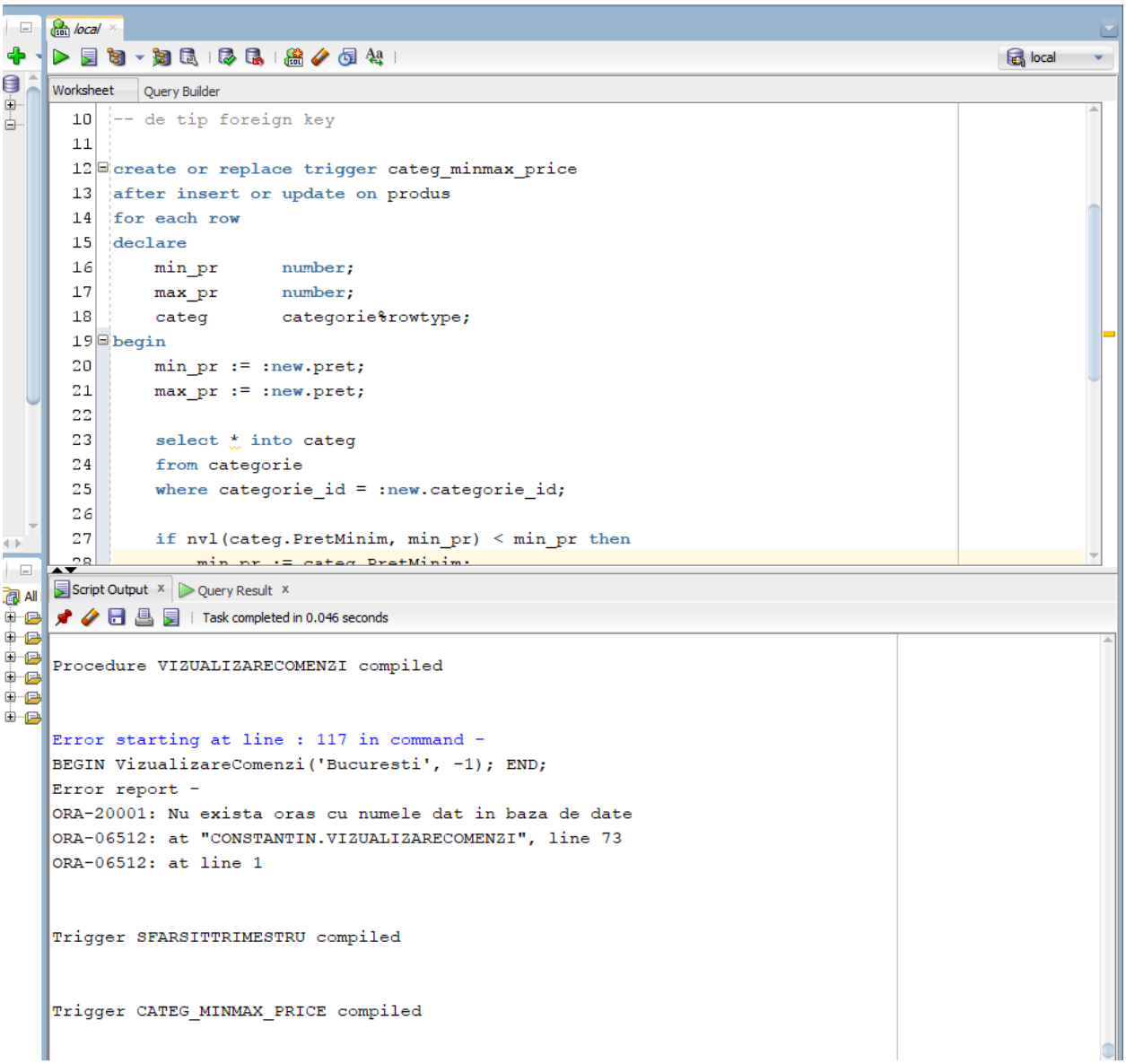

Figure 10: Execuția cerinței 11

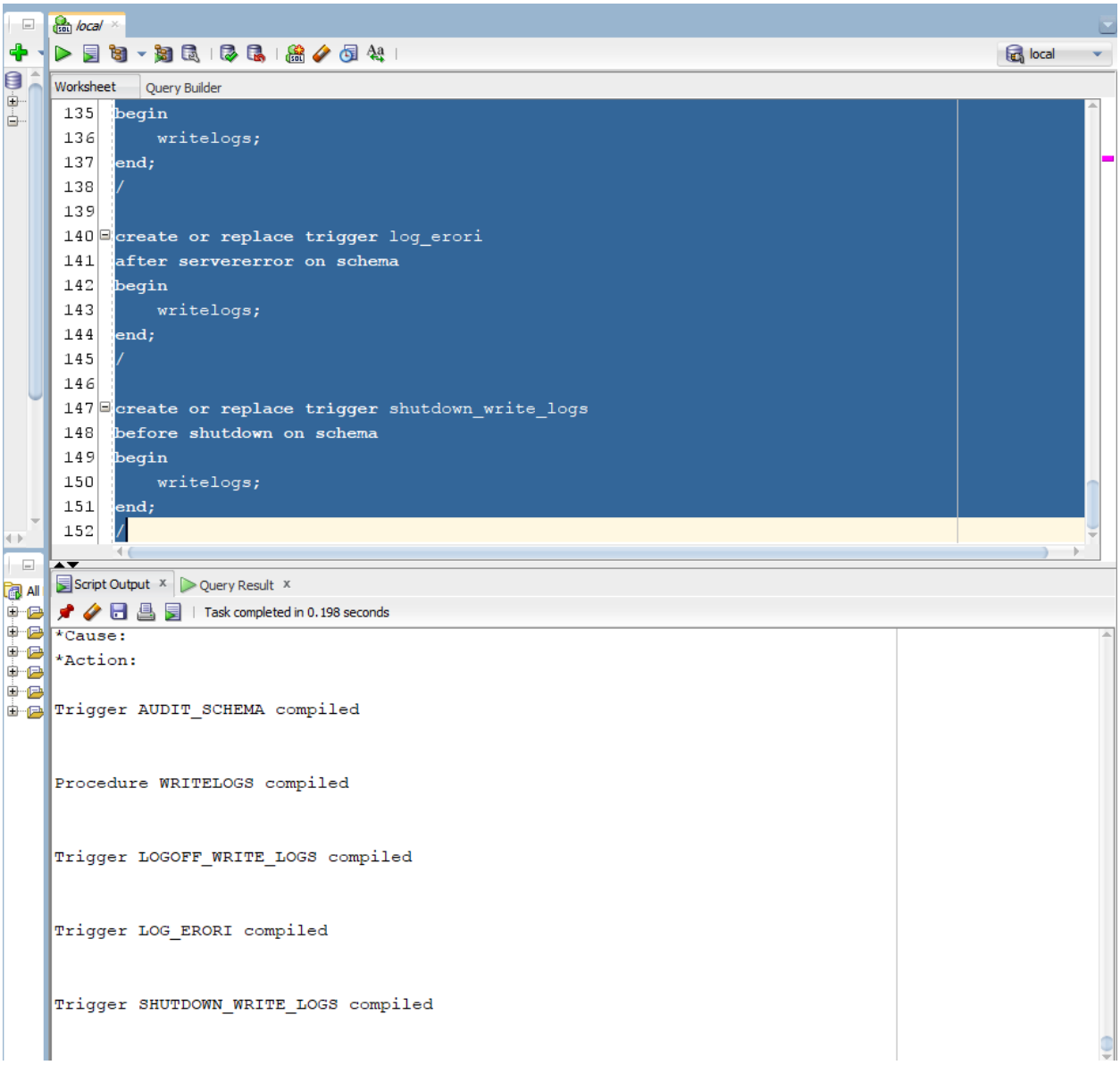

Figure 11: Execuția cerinței 12

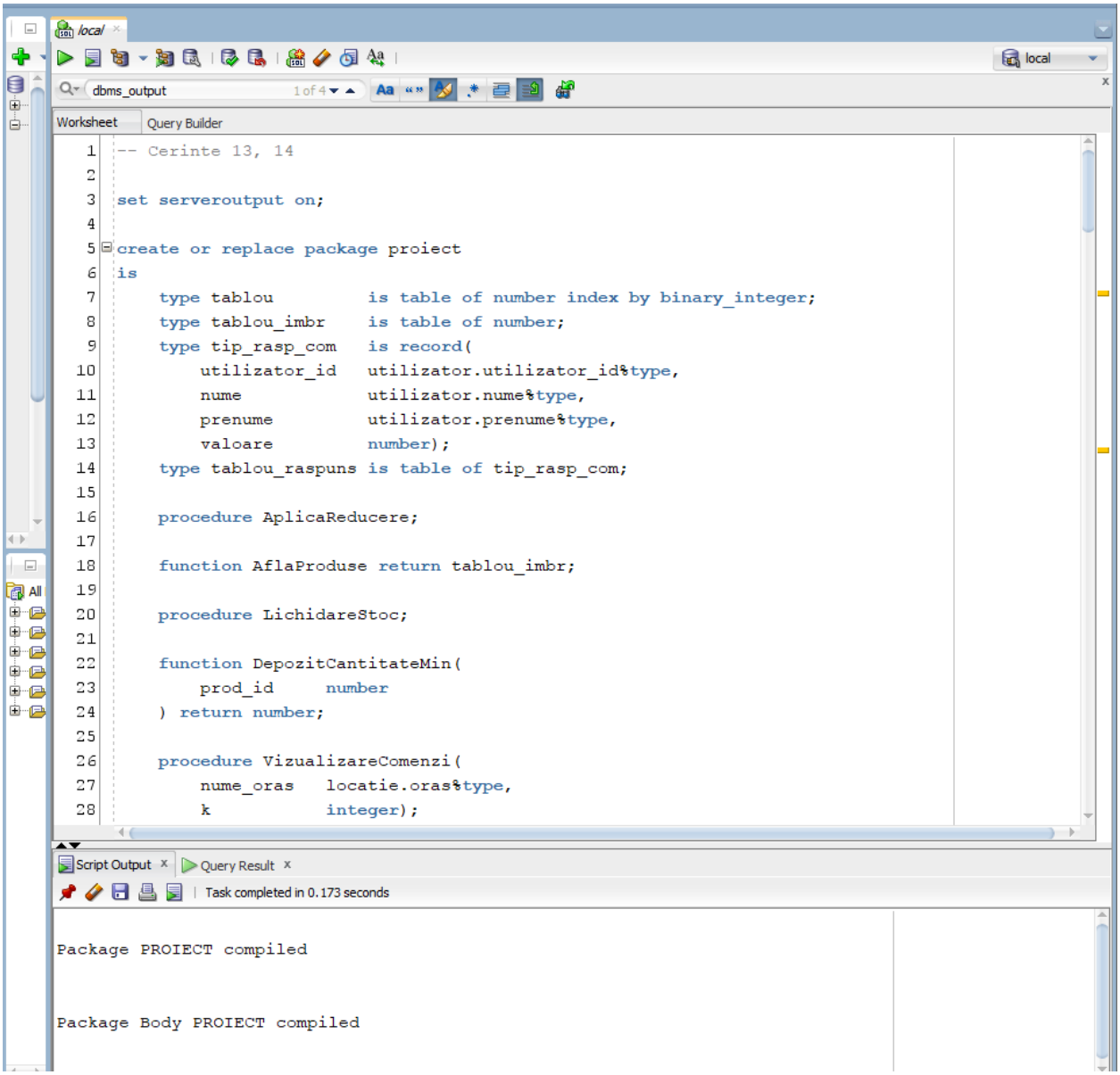

Figure 12: Execuția cerinței 13

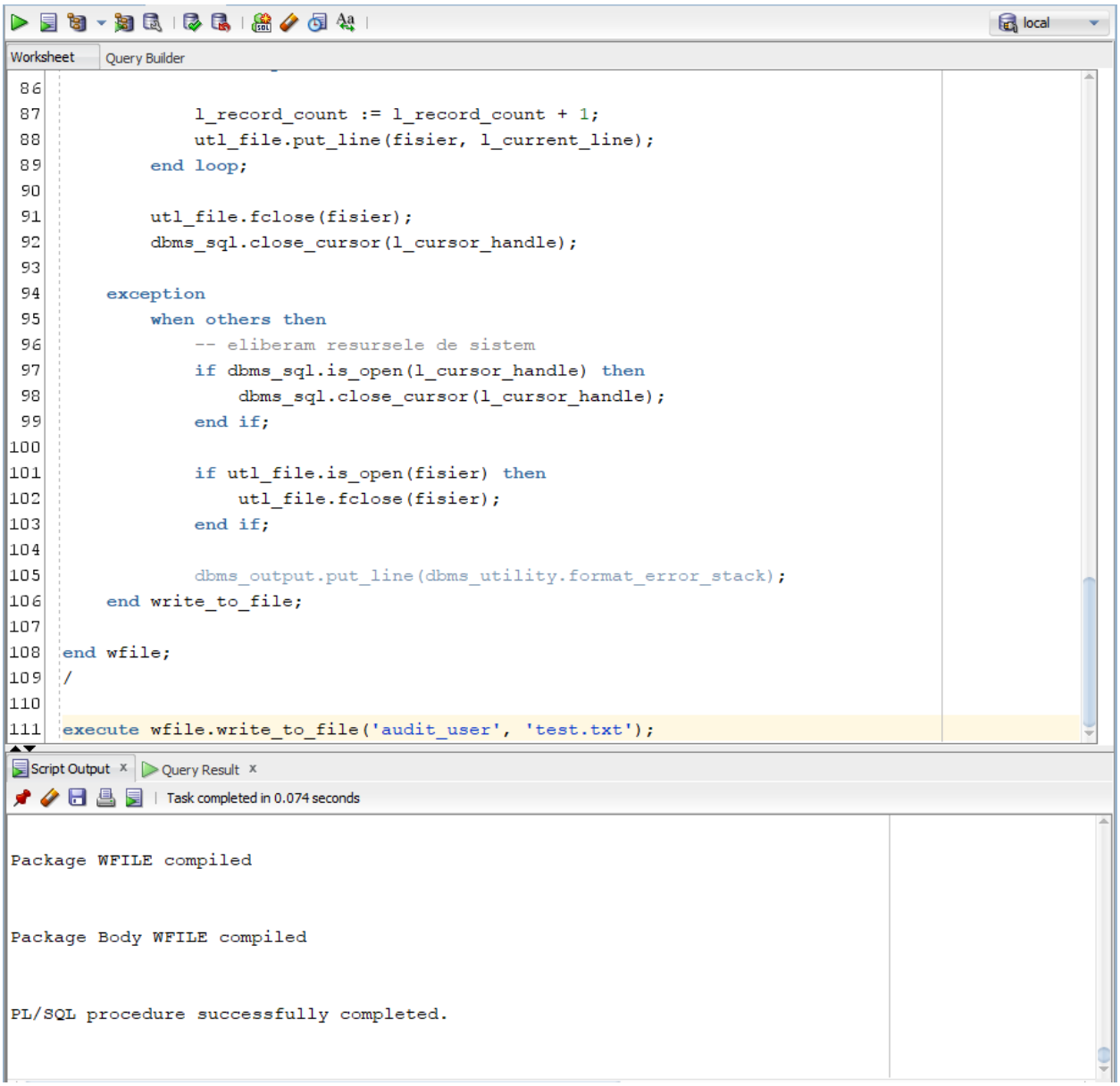

Figure 13: Execuția cerinței 14