Sisteme de gestiune a bazelor de date

Butelca Radu - 243

August 2022

目录

1 Prezentarea proiectului

Baza de date ce va urma sa fie prezentata in acest proiect a fost creata cu scopul de a stoca informatiile unui spital privat, astfel incat deciziile sa fie facute intr-un mod eficient.

2 Cerintele obligatorii

2.1 Prezentați pe scurt baza de date (utilitatea ei)

Avem medicii in jurul carora se desfasoara cele mai multe actiuni(rol, locul de parcare stabilit, sala de operatii in care lucreaza, pacientii de care au grija), pacientii ce dispun de o fisa medicala ce rezulta un contract pentru platirea serviciilor, precum si resursele(echipamentele necesare).

Entitatea "Medic" contine id_medic (de tip intreg, cheie primara nenula), id_rol(cheie straina pentru tabelul Rol), id_spital(cheie straina pentru spital), id_loc_parcare(cheie straina pentru locul de parcare) id_sala_operatii(cheie straina pentru sala de operatii), id_sef(cheie straina pentru seful de departament), numele medicului (de tip varchar2 nenul), prenume(de tip varchar2 nenul), data nasterii(de tip varchar2, stocheaza data nasterii medicului), sex (sexul doctorului de tip varchar2).

Entitatea "Vizite" contine id_vizita(cheie primara de tip intreg), id_medic_responsabil(cheie straina de tip intreg), data vizitei - data(de tip date), cat si informatii despre vizitator- nume_vizitator(varchar2), prenume_vizitator(varchar2).

Entitatea "Pacient" contine id_pacient(cheie primara de tip intreg), id_medic(cheie straina), nume(varchar2), prenume(varchar2).

Entitatea "Fisa medicala" contine id_fisa(cheie primara de tip intreg), id_pacient(cheie straina de tip intreg catre entitatea pecient), prescriptii(varchar2), medicamente(varchar2).

Entitatea "Tranzactie" contine id_tranzactie(cheie primara de tip intreg), id_fisa_medicala(cheie externa catre entitatea fisa medicala de tip intreg), id_responsabil(cheie exterena catre entitatea medic de tip intreg), detalii(varchar2), data (date).

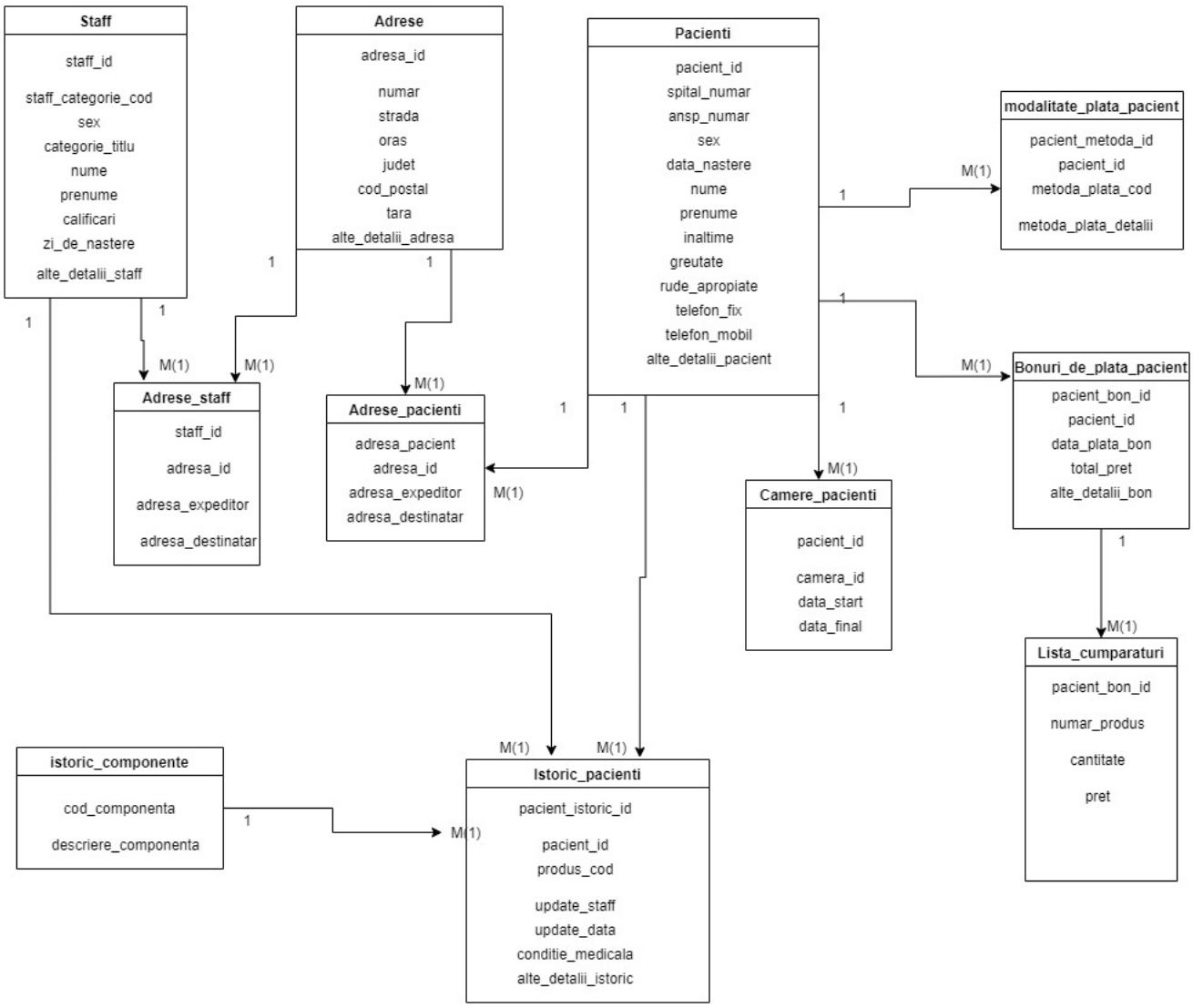
Entitatea "Loc de parcare" contine id_parcare(cheie primara de tip intreg), Zona(varchar).

Entitatea "Spital" contine id_spital(cheie primara de tip intreg), director(cheie externa catre entitatea Medic).

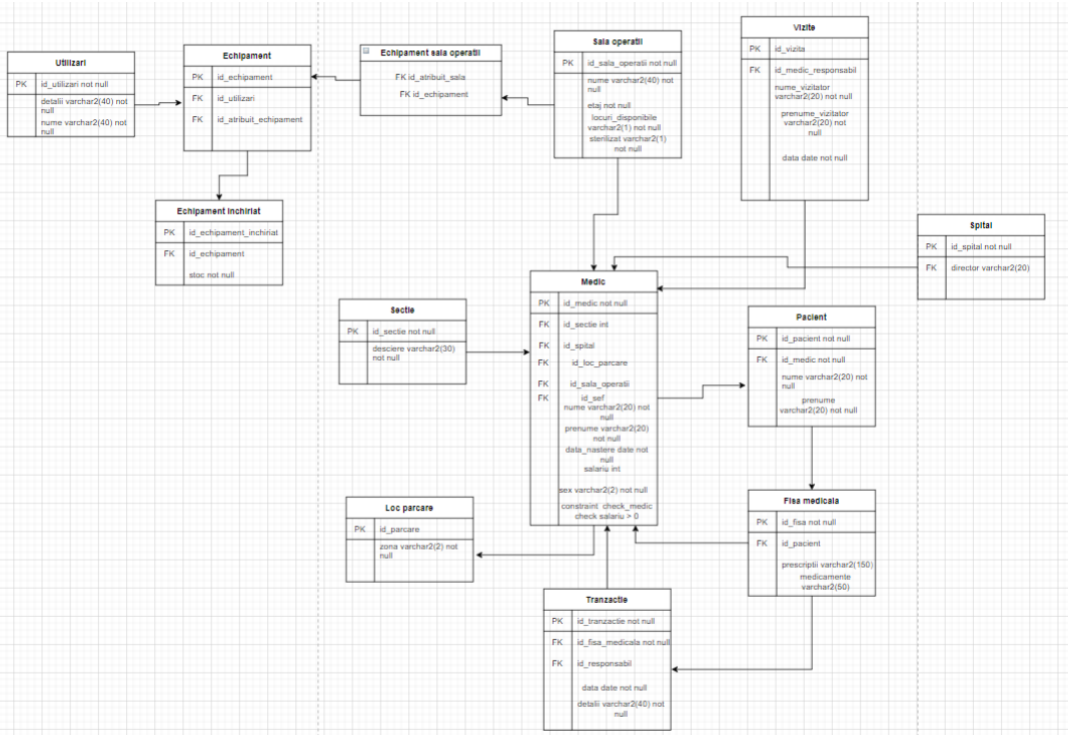
Entitatea "Sectie" contine id_sectie(cheie primara de tip intreg), descriere(varchar2).

Entitatea "Sala operatii" contine id_sala_operatii(cheie prima de tip intreg), Nume(varchar2), etaj(varchar2), locuri_disponibile(varchar), sterilizat(varchar2). Entitatea "Utilizari" contine id_utilizari(cheie primara de tip intreg), detalii(varchar2), nume(varchar2).

2.2 Realizați diagrama entitate-relație (ERD).



2.3 Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.



2.4 Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).

```
    create table parcari(
 id_parcare int not null primary key,
 nume varchar2(30) not null,
 etaj int not null
 )
```

```
create table sectii(
id_sectie int not null primary key,
nume varchar2(40) not null unique
)
```

```
 create table sali_operatie(
 id_sala_operatii int not null primary key,
```

```
 nume varchar2(20) not null unique,
 etaj int not null,
 locuri_disponibile int not null,
 sterilizat varchar2(5) not null,
 constraint check_sali_operatie check ( sterilizat = 'DA' or sterilizat = 'NU')
 )



create table medici(
id_medic int not null primary key,
id_sectie int not null,
id_sala_operatii int,
id_parcare int,
id_sef int,foreign key(id_sef) references medici(id_medic),
id_spital int,
nume varchar2(20) not null,
prenume varchar2(20) not null,
data_nastere date not null,
salariu int,
sex varchar2(3) not null,
constraint check_medici check (salariu > 0)
)

create table spitale(
id_departament int not null primary key,
id_director int,
foreign key(id_director) references medici(id_medic),
nume varchar2(20) not null
)



create table pacienti(
id_pacient int not null primary key,
id_medic_responsabil int not null,
foreign key(id_medic_responsabil) references medici(id_medic),
nume varchar2(20) not null
)

 create table utilizari(
 id_utilizare int not null primary key,
 descriere varchar2(80),
```

```
nume varchar2(30)
)
--drop table utilizari;

create table echipament(
id_echipament int not null primary key,
nume varchar2(60),
id_utilizare int not null  references utilizari(id_utilizare)
)

create table echipament_sala_operatii
(
id_sala_operatii int constraint fk_eso references sali_operatie(id_sala_operatii),
id_echipament int constraint fk_eso2 references echipament(id_echipament),
constraint pk_eso primary key (id_sala_operatii, id_echipament)
)




create table vizite(
id_vizita int not null primary key,
id_medic_responsabil int,
foreign key(id_medic_responsabil) references medici(id_medic),
data date,
nume_vizitator varchar2(20) not null,
prenume_vizitator varchar2(20) not null
)

create table fise_medicale(
id_fisa_medicala int not null primary key,
prescriptii varchar2(80) not null unique,
medicamente varchar2(100),
id_pacient int not null,
foreign key(id_pacient) references pacienti(id_pacient)
)

create table tranzactii(
id_tranzactie int not null primary key,
data date,
descriere varchar2(50),
```

```
 suma int not null,
 id_medic_responsabil int not null,
 foreign key(id_medic_responsabil) references medici(id_medic),
 id_fisa_medicala int not null,
 foreign key(id_fisa_medicala) references fise_medicale(id_fisa_medicala),
 constraint check_tranzactii check(suma > 0)
 )


--drop table echipament;
 alter table medici
 add
 foreign key(id_sectie) references sectii(id_sectie);

 alter table medici
 add
 foreign key(id_sala_operatii) references sali_operatie(id_sala_operatii);

  alter table medici
 add
 foreign key(id_parcare) references parcari(id_parcare);


 alter table medici
 add
 foreign key(id_departament) references spitale(id_departament);



----------------------
```

2.5 Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

```
    create sequence parcari_sequence
increment by 1
start with 9
nomaxvalue
minvalue 1
nocache
```

```
nocycle;

insert into parcari
values(parcari_sequence.nextval, 'A', 3);
insert into parcari
values(parcari_sequence.nextval, 'C', 2);
insert into parcari
values(parcari_sequence.nextval, 'B', 3);
insert into parcari
values(parcari_sequence.nextval, 'B', 2);
insert into parcari
values(parcari_sequence.nextval, 'C', 4);
insert into parcari
values(parcari_sequence.nextval, 'B', 2);
insert into parcari
values(parcari_sequence.nextval, 'C', 3);


select*
from parcari


--drop sequence spital_sequence
create sequence spital_sequence
increment by 1
start with 9
nomaxvalue
minvalue 1
nocache
nocycle;

insert into spitale
values(spital_sequence.nextval, null,'Floreasca');
insert into spitale
values(spital_sequence.nextval, null,'Cismigiu');
insert into spitale
values(spital_sequence.nextval, null,'Izvor');
insert into spitale
values(spital_sequence.nextval, null,'Grozavesti');
insert into spitale
values(spital_sequence.nextval, null,'Petrache Poenaru');
```

```
insert into spitale
values(spital_sequence.nextval, null,'Gara de Nord');
insert into spitale
values(spital_sequence.nextval, null,'Victoriei');

select*
from spitale;

---drop sequence sala_operatii_sequence
create sequence sala_operatii_sequence
increment by 1
start with 9
nomaxvalue
minvalue 1
nocache
nocycle;

insert into sali_operatie
values(sala_operatii_sequence.nextval, 'Dragomir', 2, 32, 'DA');
insert into sali_operatie
values(sala_operatii_sequence.nextval, 'Martar', 3, 2, 'NU');
insert into sali_operatie
values(sala_operatii_sequence.nextval, 'Vicu', 1, 13, 'NU');
insert into sali_operatie
values(sala_operatii_sequence.nextval, 'Vasea', 3, 14, 'DA');
insert into sali_operatie
values(sala_operatii_sequence.nextval, 'Trandafir', 1, 16, 'DA');
insert into sali_operatie
values(sala_operatii_sequence.nextval, 'Clopot', 3, 222, 'DA');

select*
from sali_operatie

---drop sequence sectii_sequence
create sequence sectii_sequence
increment by 1
start with 9
nomaxvalue
minvalue 1
nocache
nocycle;
```

```
insert into sectii
values(sectii_sequence.nextval, 'ATI');
insert into sectii
values(sectii_sequence.nextval, 'Hemodializa');
insert into sectii
values(sectii_sequence.nextval, 'Chirurgie');
insert into sectii
values(sectii_sequence.nextval, 'Neonatologie');
insert into sectii
values(sectii_sequence.nextval, 'Neurochirurgie');
insert into sectii
values(sectii_sequence.nextval, 'Gastrologie');
select*
from sectii;



---drop sequence medici_sequence
create sequence medici_sequence
increment by 1
start with 9
nomaxvalue
minvalue 1
nocache
nocycle;
drop sequence medici_sequence;

select sequence medici_sequence.nextval from dual;
select sequence medici_sequence.currval from dual;


insert into medici
values(medici_sequence.nextval, 10, 10, 10, null, 10, 'Mihai', 'Mironica', to_date('1960/6/22', 'yyyy/mm/d

insert into medici
values(medici_sequence.nextval, 11, 11, 11, null, 11, 'Madalina', 'Manole', to_date('1980/8/12', 'yyyy/mm/
insert into medici
values(medici_sequence.nextval, 12, 12, 12, null, 12, 'Lupu', 'Catrinel', to_date('1990/9/22', 'yyyy/mm/dd
insert into medici
values(medici_sequence.nextval, 13, 13, 13, null, 13, 'Mihalceanu', 'Liviu', to_date('1999/3/22', 'yyyy/mm
```

```
insert into medici
values(medici_sequence.nextval, 14, 14, 14, null, 14, 'Dan', 'Brown', to_date('2001/2/22', 'yyyy/mm/dd'), 
insert into medici
values(medici_sequence.nextval, 15, 15, 15, null, 15, 'Carlita', 'Carmen', to_date('1982/6/22', 'yyyy/mm/d
insert into medici
values(medici_sequence.nextval, 13, 12, 12, null, 15, 'Crasnuta', 'Gabriel', to_date('1982/6/22', 'yyyy/mm
select*
from medici;

---drop sequence pacienti_sequence
create sequence pacienti_sequence
increment by 1
start with 10
nomaxvalue
minvalue 1
nocache
nocycle;



insert into pacienti
values(pacienti_sequence.nextval, 13, 'Tanasa');
insert into pacienti
values(pacienti_sequence.nextval, 10, 'Gabor');
insert into pacienti
values(pacienti_sequence.nextval, 11, 'Lungu');
insert into pacienti
values(pacienti_sequence.nextval, 12, 'Scurtu');
insert into pacienti
values(pacienti_sequence.nextval, 15, 'Tabacaru');
insert into pacienti
values(pacienti_sequence.nextval, 15, 'Garcea');
insert into pacienti
values(pacienti_sequence.nextval, 14, 'Borcea');
insert into pacienti
values(pacienti_sequence.nextval, 12, 'Becali');



select*
from pacienti;
```

```
---drop sequence fise_medicale_sequence
create sequence fise_medicale_sequence
increment by 1
start with 10
nomaxvalue
minvalue 1
nocache
nocycle;


insert into fise_medicale
values(fise_medicale_sequence.nextval, 'Dimineata', 'Paracetamol', 11);
insert into fise_medicale
values(fise_medicale_sequence.nextval, 'Seara', 'Nurofen', 12);
insert into fise_medicale
values(fise_medicale_sequence.nextval, 'Amiaza', 'Diazepam', 13);
insert into fise_medicale
values(fise_medicale_sequence.nextval, 'Amiaza si seara', 'Xanax', 14);
insert into fise_medicale
values(fise_medicale_sequence.nextval, 'In curand', 'Strepsils', 15);
insert into fise_medicale
values(fise_medicale_sequence.nextval, '4 ori pe zi', 'Teraflu extra', 16);

select*
from fise_medicale;


---drop sequence tranzactii_sequence
create sequence tranzactii_sequence
increment by 1
start with 10
nomaxvalue
minvalue 1
nocache
nocycle;
-- 13 19 20 21 23
--11 13 14 15 16 17

insert into tranzactii
```

```
values(tranzactii_sequence.nextval, to_date('2021/6/12', 'yyyy/mm/dd'), 'Valid', 120, 13, 11  );
insert into tranzactii
values(tranzactii_sequence.nextval, to_date('2021/3/10', 'yyyy/mm/dd'), 'Invalid', 120, 11, 16  );
insert into tranzactii
values(tranzactii_sequence.nextval, to_date('2021/1/15', 'yyyy/mm/dd'), 'Invalid', 200, 14, 15  );
insert into tranzactii
values(tranzactii_sequence.nextval, to_date('2021/4/14', 'yyyy/mm/dd'), 'Valid', 400, 15, 13  );
insert into tranzactii
values(tranzactii_sequence.nextval, to_date('2020/6/10', 'yyyy/mm/dd'), 'Valid', 520, 11, 14  );

select*
from tranzactii;

---drop sequence vizite_sequence
create sequence vizite_sequence
increment by 1
start with 10
nomaxvalue
minvalue 1
nocache
nocycle;
-- 13 19 20 21 23
insert into vizite
values(vizite_sequence.nextval, 13,to_date('2021/5/12', 'yyyy/mm/dd'), 'Chelmu', 'Ingrid' );
insert into vizite
values(vizite_sequence.nextval, 16,to_date('2021/7/10', 'yyyy/mm/dd'), 'Vatavu', 'Mariana' );
insert into vizite
values(vizite_sequence.nextval, 15 ,to_date('2021/2/13', 'yyyy/mm/dd'), 'Finutu', 'Ionut' );
insert into vizite
values(vizite_sequence.nextval, 12,to_date('2021/4/16', 'yyyy/mm/dd'), 'Comisu', 'Dragos' );
insert into vizite
values(vizite_sequence.nextval, 10,to_date('2021/3/15', 'yyyy/mm/dd'), 'Gurguta', 'Daniel' );
insert into vizite
values(vizite_sequence.nextval, 14,to_date('2021/5/12', 'yyyy/mm/dd'), 'Macovei', 'Bogdan' );

select*
from vizite;

--drop sequence utilizari_sequence
create sequence utilizari_sequence
increment by 1
```

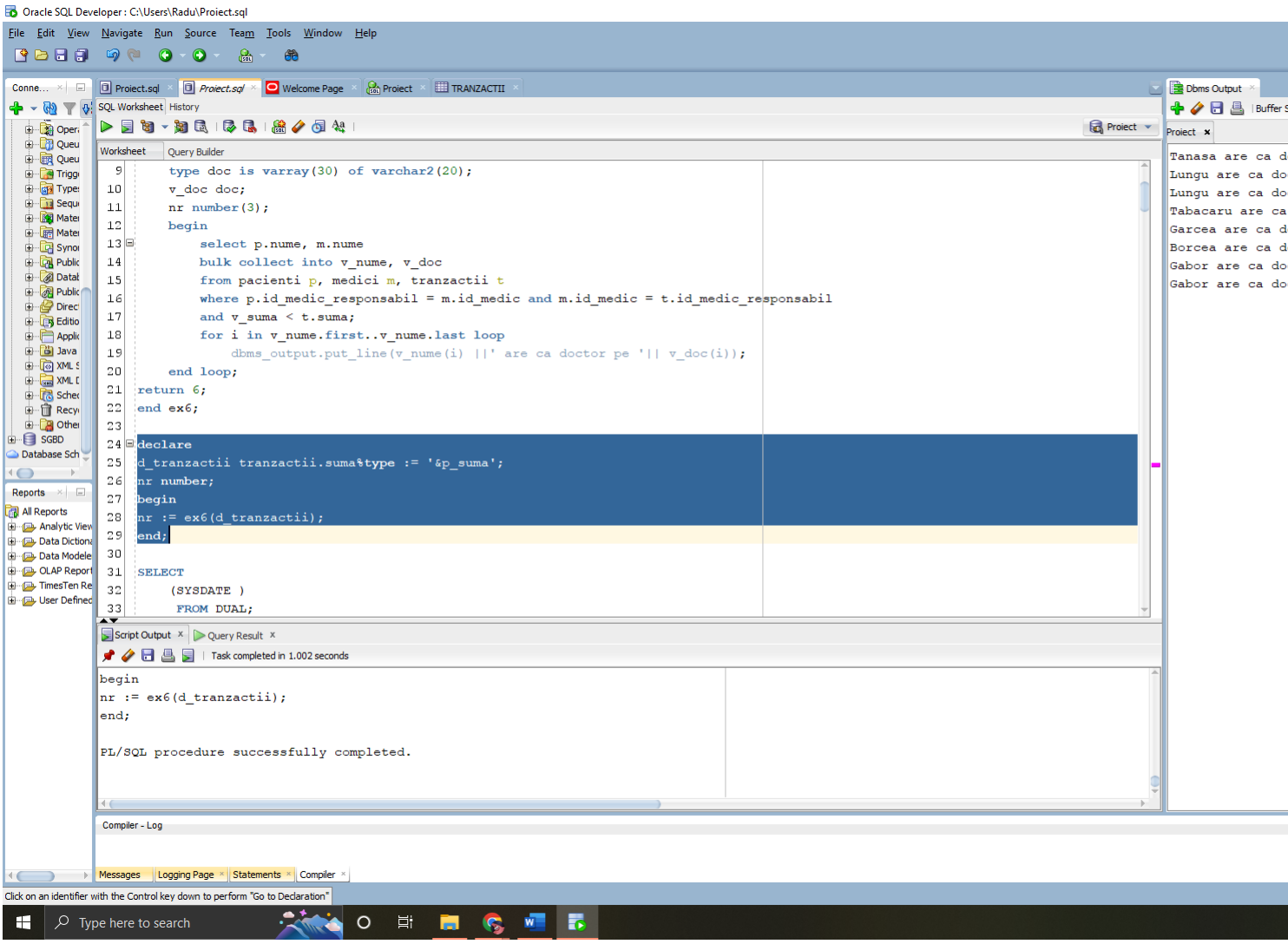
```
start with 10
nomaxvalue
minvalue 1
nocache
nocycle;

insert into utilizari
values(utilizari_sequence.nextval, 'locomotor', 'scaun cu roti');
insert into utilizari
values(utilizari_sequence.nextval, 'asculta batai inima', 'stetoscop');
insert into utilizari
values(utilizari_sequence.nextval, 'vizionare', 'raze x');
insert into utilizari
values(utilizari_sequence.nextval, 'la creier', 'encefalograma');
insert into utilizari
values(utilizari_sequence.nextval, 'la ureche', 'irg');

select*
from utilizari;



--drop sequence echipament_sequence
create sequence echipament_sequence
increment by 1
start with 10
nomaxvalue
minvalue 1
nocache
nocycle;
-- 11 13 14 15 16
--scaun stetoscop raze x encefalograma irg
insert into echipament
values(echipament_sequence.nextval, 'scaun', 11  );
insert into echipament
values(echipament_sequence.nextval, 'stetoscop', 12  );
insert into echipament
values(echipament_sequence.nextval, 'aparat raze x', 13 );
insert into echipament
values(echipament_sequence.nextval, 'aparat encefalograma', 14  );
insert into echipament
values(echipament_sequence.nextval, 'aparat irg',  15 );
```

```
select*
from echipament;


--drop sequence echipament_sala_operatii




--echip 11 12 13 14 15
--sala 10 12 13 14 15 16
insert into echipament_sala_operatii
values( 10, 11 );
insert into echipament_sala_operatii
values( 10, 12 );
insert into echipament_sala_operatii
values( 10, 13 );
insert into echipament_sala_operatii
values( 10, 14 );
insert into echipament_sala_operatii
values( 10, 15 );
insert into echipament_sala_operatii
values( 12, 11 );
insert into echipament_sala_operatii
values( 13, 11 );
insert into echipament_sala_operatii
values( 14, 11 );
insert into echipament_sala_operatii
values( 15, 11 );
insert into echipament_sala_operatii
values( 15, 12 );
```

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

Sa se afiseze numele complet al pacientilor si numele medicului responsabil care are o tranzactie mai mare de 200 de roni.

```
    create or replace function ex6
    (v_suma tranzactii.suma%type)
    return number is
    type nume is table of pacienti.nume%type index by binary_integer;
    v_nume nume;
    type doc is varray(30) of varchar2(20);
    v_doc doc;
    nr number(3);
    begin
        select p.nume, m.nume
        bulk collect into v_nume, v_doc
        from pacienti p, medici m, tranzactii t
        where p.id_medic_responsabil = m.id_medic and m.id_medic = t.id_medic_responsabil
        and v_suma < t.suma;
        for i in v_nume.first..v_nume.last loop
            dbms_output.put_line(v_nume(i) ||' are ca doctor pe '|| v_doc(i));
    end loop;
return 6;
end ex6;

declare
d_tranzactii tranzactii.suma%type := '&p_suma';
nr number;
begin
nr := ex6(d_tranzactii);
end;
```

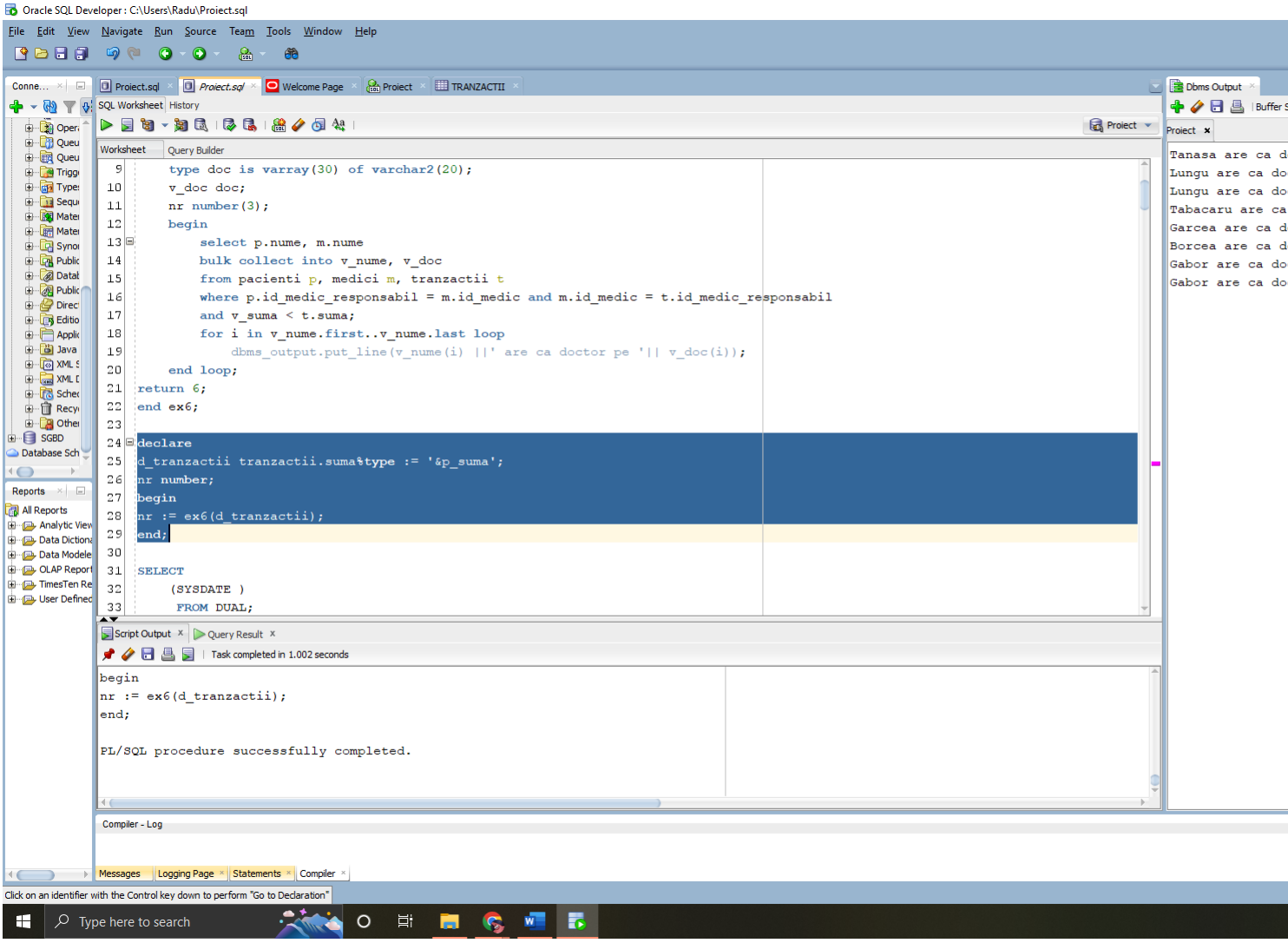


2.7 Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

Pentru fiecare medic printati pe ecran pacientii pe care ii trateaza.

```
create or replace procedure f7
as
nume_medic medici.nume%type;
nr_pac number;
cursor c is
    select m.nume, count(id_pacient)
    from medici m, pacienti p
    where m.id_medic = p.id_medic_responsabil
    group by m.nume;
begin
```

```
    open c;
    loop
        fetch c into nume_medic, nr_pac;
        exit when c%notfound;

        if nr_pac = 0 then
            dbms_output.put_line( 'Medicul nu are niciun pacient!');
        else
            dbms_output.put_line( 'Medicul ' || nume_medic || ' are ' || nr_pac);
        end if;
    end loop;
    close c;
end f7;

execute f7();
```



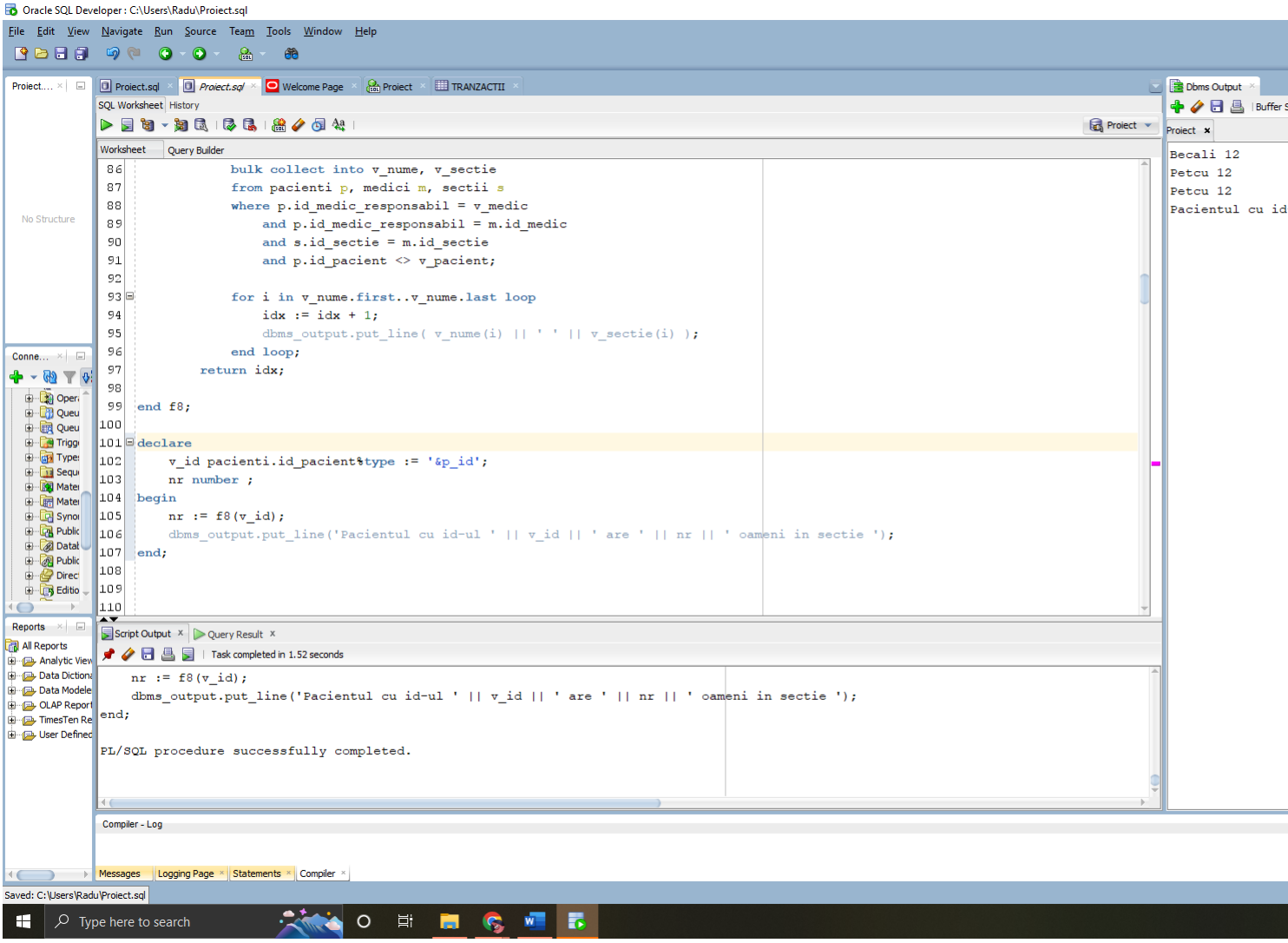
2.8 Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

–Pentru un pacient citit de la tastatura sa se afiseze cati colegi de sectie are acesta.

```
create or replace function f8
    (v_pacient pacienti.id_pacient%type)
    return number is
        type nume is table of pacienti.nume%type index by binary_integer;
        type sectie is table of sectii.id_sectie%type index by binary_integer;
        v_nume nume;
        v_sectie sectie;
```

```
        idx number:=0;
        v_medic pacienti.id_medic_responsabil%type;
        begin
            select id_medic_responsabil
            into v_medic
            from pacienti
            where id_pacient = v_pacient;

            select p.nume, s.id_sectie
            bulk collect into v_nume, v_sectie
            from pacienti p, medici m, sectii s
            where p.id_medic_responsabil = v_medic
                and p.id_medic_responsabil = m.id_medic
                and s.id_sectie = m.id_sectie
                and p.id_pacient <> v_pacient;

            for i in v_nume.first..v_nume.last loop
                idx := idx + 1;
                dbms_output.put_line( v_nume(i) || ' ' || v_sectie(i) );
            end loop;
        return idx;

end f8;


declare
    v_id pacienti.id_pacient%type := '&p_id';
    nr number ;
begin
    nr := f8(v_id);
    dbms_output.put_line('Pacientul cu id-ul ' || v_id || ' are ' || nr || ' oameni in sectie ');
end;
```

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

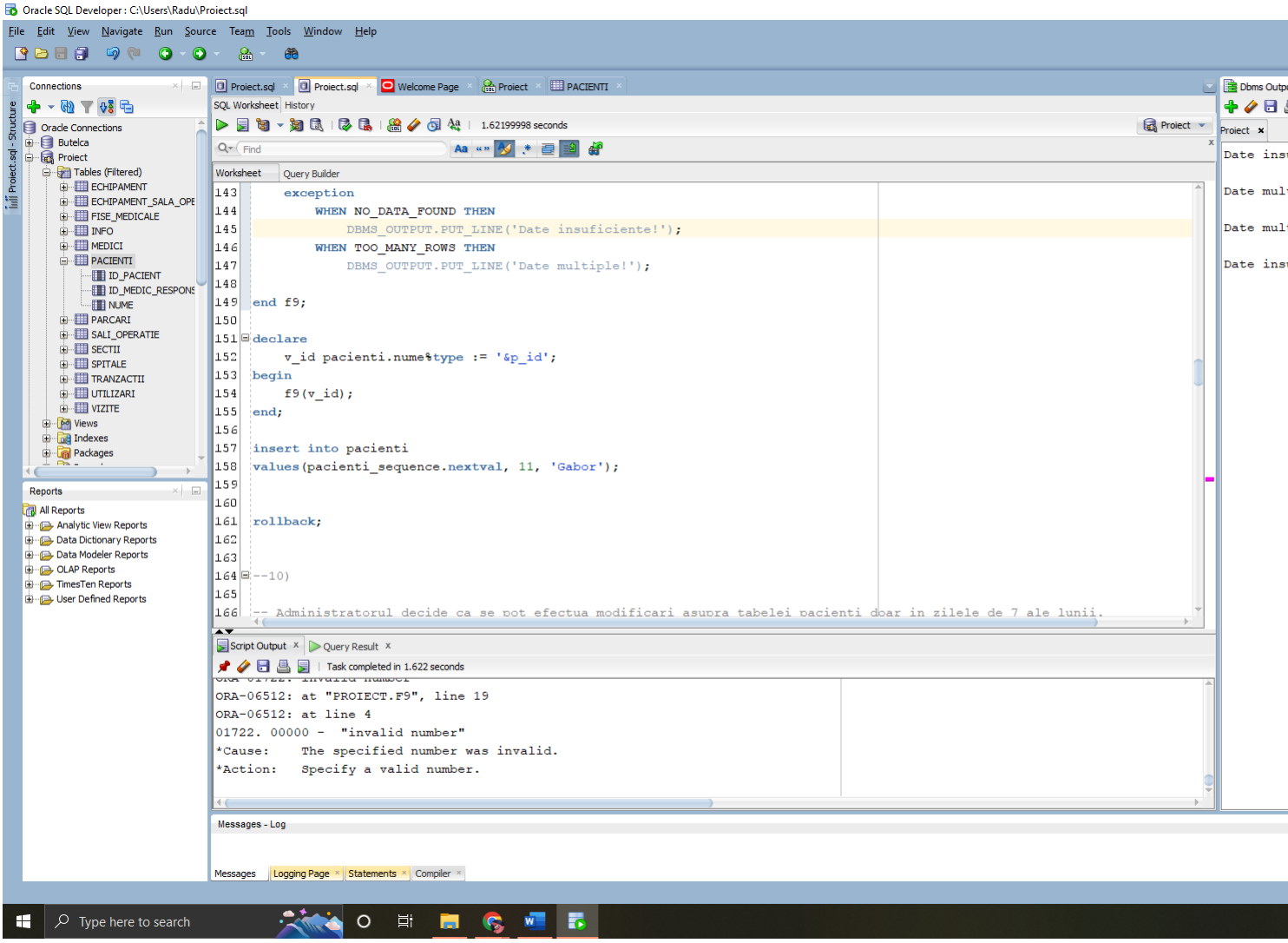


2.9 Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO_DATA_FOUND și TOO_MANY_ROWS. Apelați subprogramul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

Pentru un pacient citit de la tastatura afisati numele, prenumele si locul de parcare al medicului sau.

```
create or replace procedure f9
(v_pacient pacienti.nume%type)

is
```

```
    v_nume medici.nume%type;
    v_prenume medici.prenume%type;
    v_parcare parcari.nume%type;
    v_id pacienti.id_pacient%type;



    begin
        select id_pacient
        into v_id
        from pacienti
        where v_pacient = nume;



        select m.nume, m.prenume, p.nume
            into v_nume, v_prenume, v_parcare
            from medici m, pacienti pac, parcari p, tranzactii t, fise_medicale fm
            where  pac.id_pacient = v_pacient
            and t.id_medic_responsabil = m.id_medic and t.id_fisa_medicala = fm.id_fisa_medicala
            and fm.id_pacient = pac.id_pacient and m.id_parcare = p.id_parcare;

    dbms_output.put_line(v_nume||' '||v_prenume||' cu parcarea '|| v_parcare  ||'!');



    exception
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Date insuficiente!');
        WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Date multiple!');

end f9;

declare
    v_id pacienti.nume%type := '&p_id';
begin
    f9(v_id);
end;
```

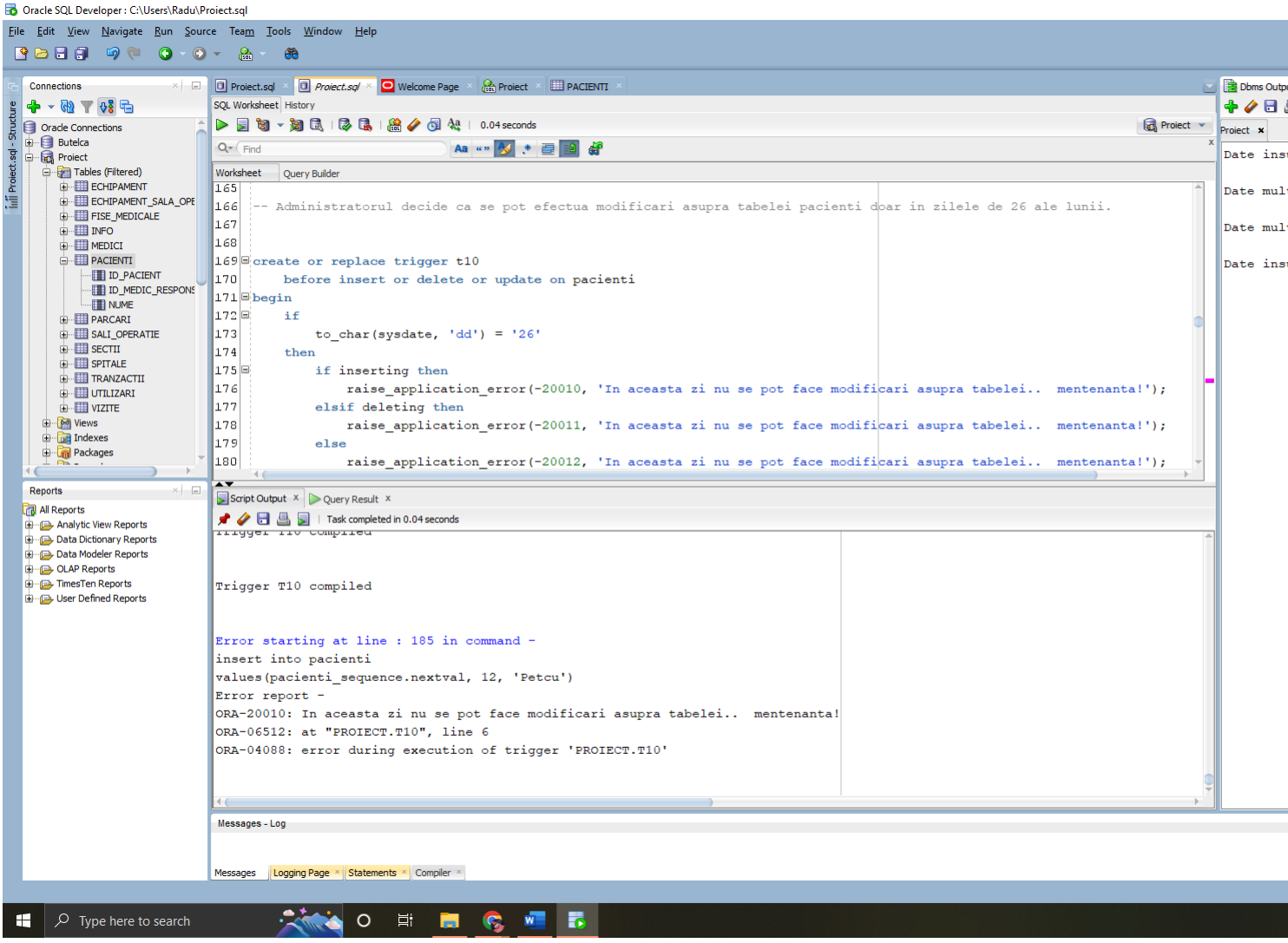


2.10 Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

Presupunem ca se pot efectua modificari asupra tabelelor doar in zilele de 7 ale lunilor.

```
    create or replace trigger t10
    before insert or delete or update on pacienti
begin
    if
        to_char(sysdate, 'dd') = '24'
    then
        if inserting then
            raise_application_error(-20010, 'In aceasta zi nu se pot face modificari asupra tabelei..  men
        elsif deleting then
            raise_application_error(-20011, 'In aceasta zi nu se pot face modificari asupra tabelei..  men
        else
```

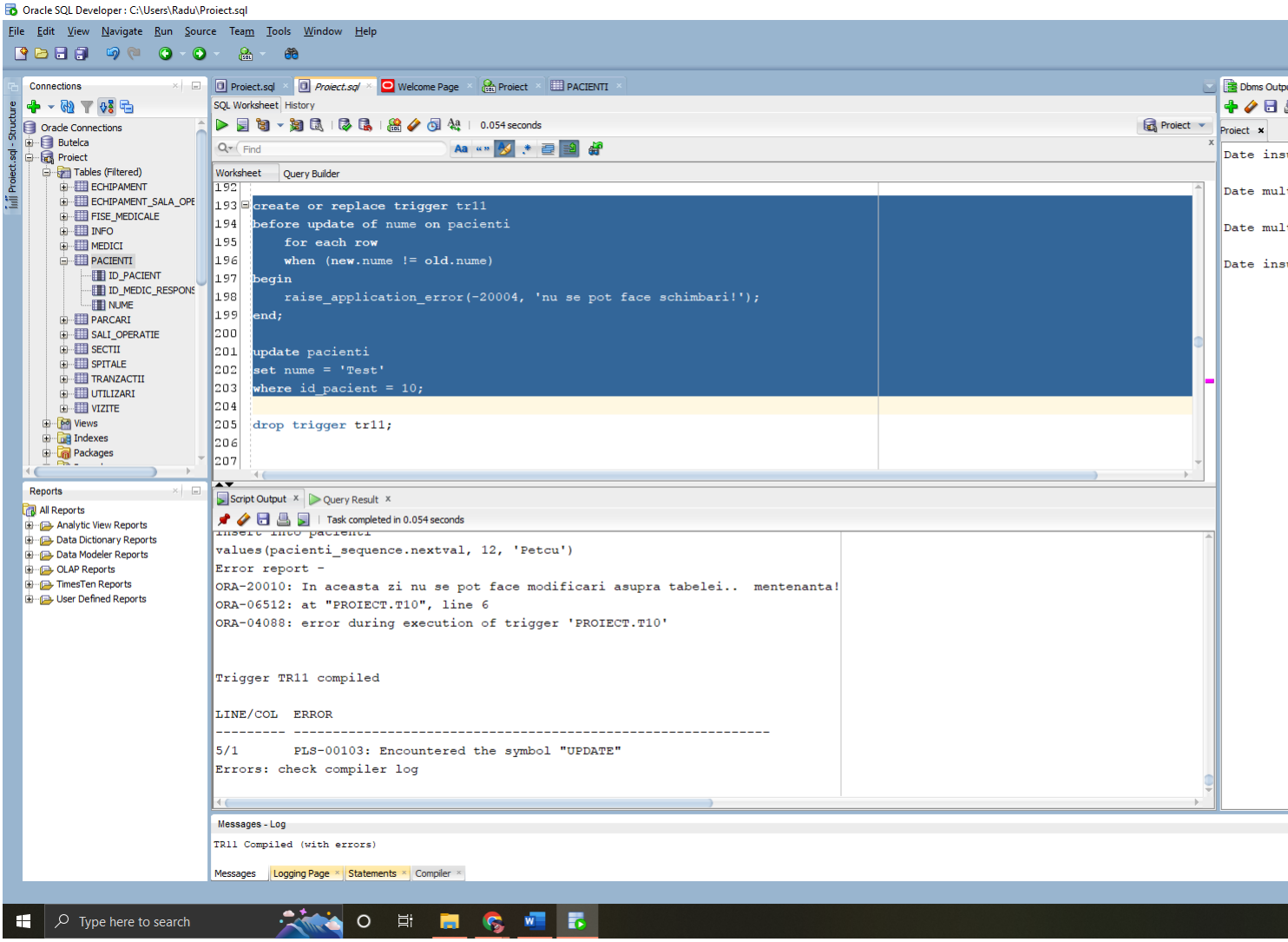
```
            raise_application_error(-20012, 'In aceasta zi nu se pot face modificari asupra tabelei..  men
        end if;
    end if;
end;

insert into pacienti
values(pacienti_sequence.nextval, 12, 'Petcu');
```



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

Creati un trigger care blocheaza permisiunea utilizatorului de a modifica data unui anunt.

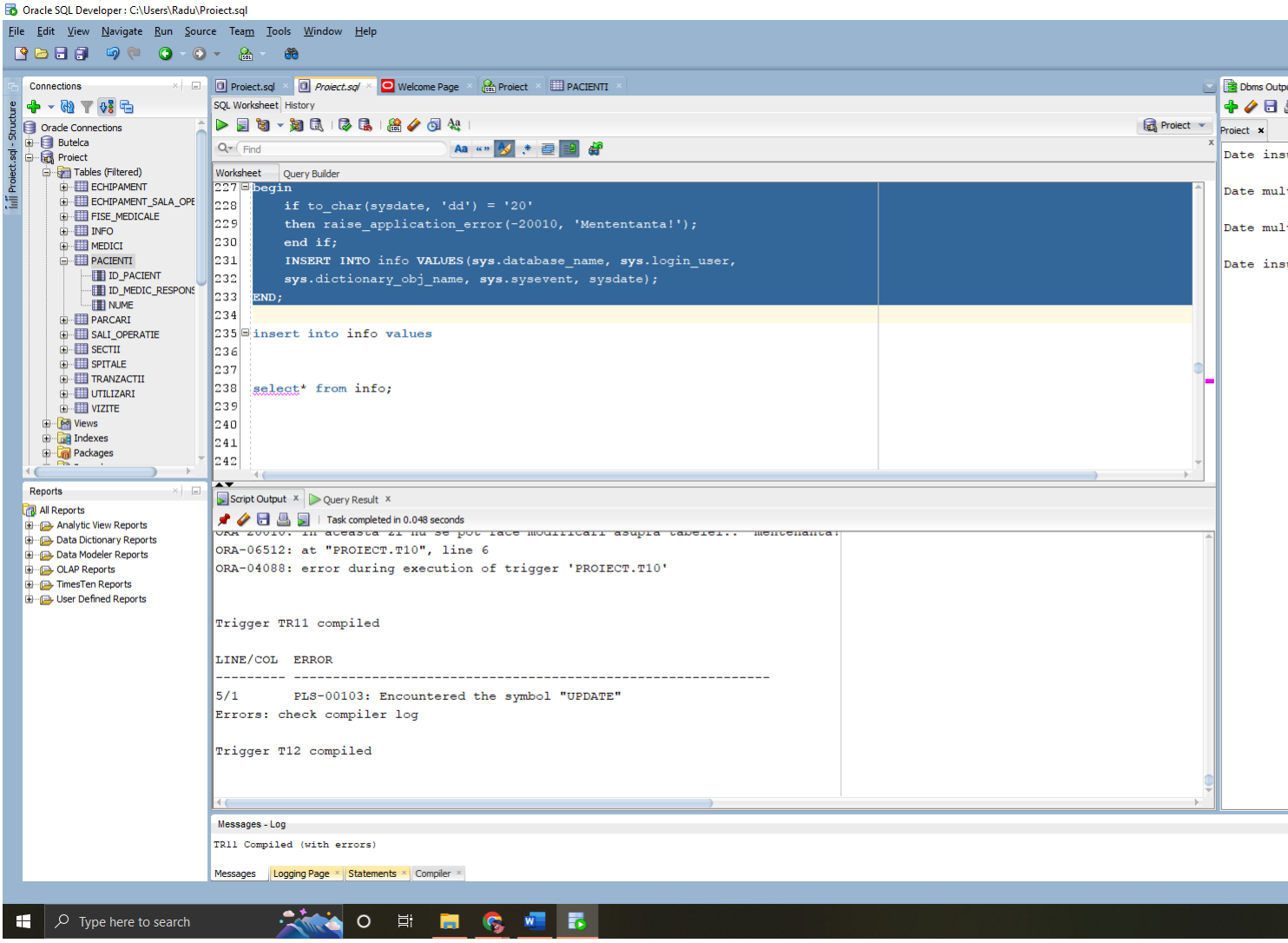
```
    create or replace trigger tr11
before update of nume on pacienti
    for each row
    when (new.nume != old.nume)
begin
    raise_application_error(-20004, 'nu se pot face schimbari!');
end;

update pacienti
set nume = 'Test'
where id_pacient = 10;
```



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

Creati un trigger care sa salveze modificari ale bazei de date si blocati adaugarea/modificarea/stergerea in zilele de mentenanta.

```
    create table info
(
    bd_name varchar2(30),
    active_user varchar2(40),
    obiect varchar2(30),
    commnad varchar2(25),
    Data date
);


drop table info;


create or replace trigger t12
    after create or drop or alter on schema
begin
    if to_char(sysdate, 'dd') = '20'
    then raise_application_error(-20010, 'Mententanta!');
    end if;
    INSERT INTO info VALUES(sys.database_name, sys.login_user,
    sys.dictionary_obj_name, sys.sysevent, sysdate);
END;
```



3 Cerintele suplimentare

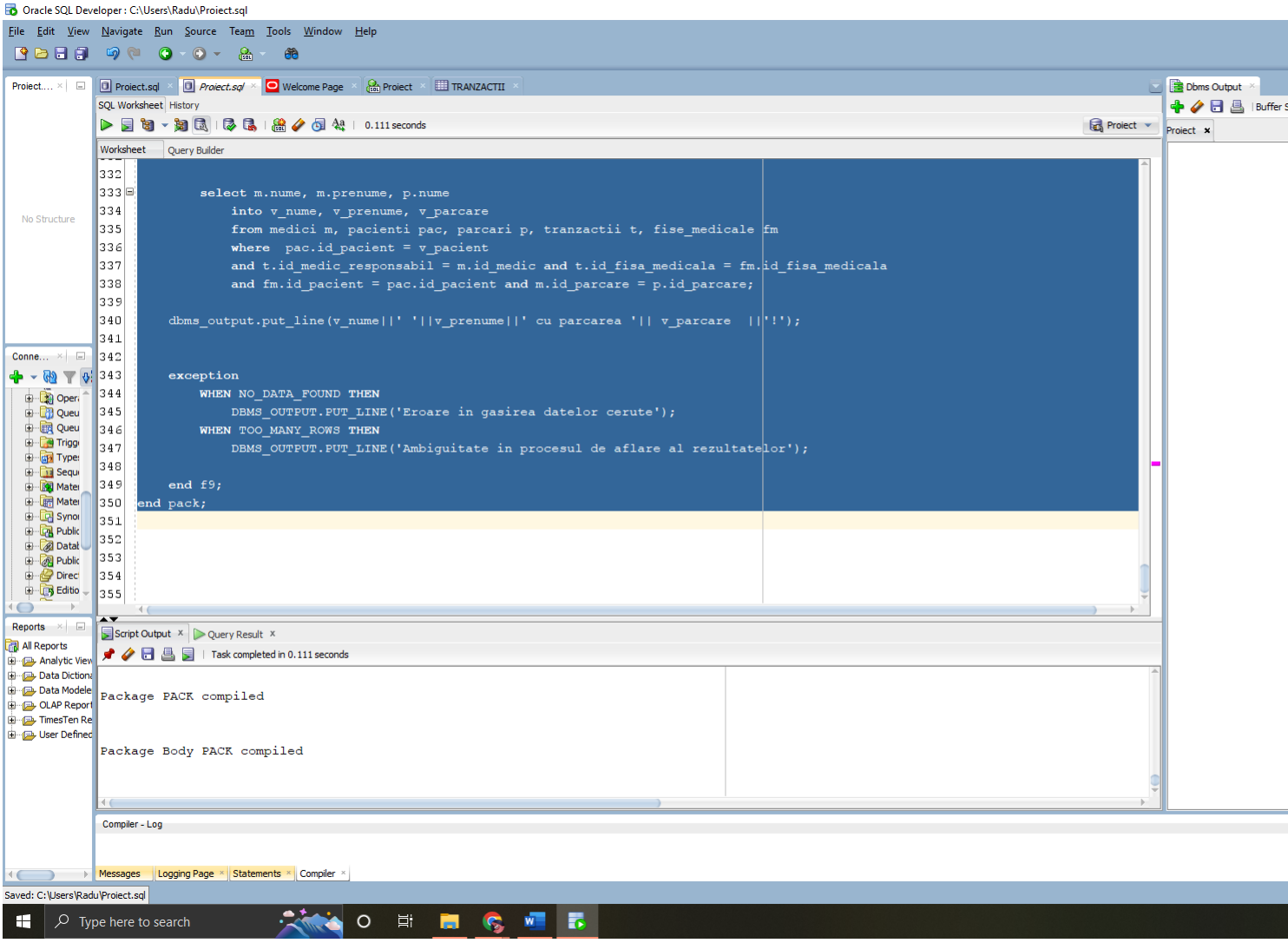
3.1 Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.

```
create or replace package pack as
function ex6(v_suma tranzactii.suma%type)
return number;



function f8(v_pacient pacienti.id_pacient%type)
return number;

procedure f9(v_pacient pacienti.nume%type);
```

```
end pack;
/


create or replace package body pack as
        function ex6
        (v_suma tranzactii.suma%type)
        return number is
        type nume is table of pacienti.nume%type index by binary_integer;
        v_nume nume;
        type doc is varray(30) of varchar2(20);
        v_doc doc;
        nr number(3);
        begin
            select p.nume, m.nume
            bulk collect into v_nume, v_doc
            from pacienti p, medici m, tranzactii t
            where p.id_medic_responsabil = m.id_medic and m.id_medic = t.id_medic_responsabil
            and v_suma < t.suma;
            for i in v_nume.first..v_nume.last loop
                dbms_output.put_line(v_nume(i) ||' are ca doctor pe '|| v_doc(i));
        end loop;
        return 6;
        end ex6;



        function f8
    (v_pacient pacienti.id_pacient%type)
    return number is
        type nume is table of pacienti.nume%type index by binary_integer;
        type sectie is table of sectii.id_sectie%type index by binary_integer;
        v_nume nume;
        v_sectie sectie;
        idx number:=0;
        v_medic pacienti.id_medic_responsabil%type;
        begin
            select id_medic_responsabil
            into v_medic
            from pacienti
```

```
        where id_pacient = v_pacient;

        select p.nume, s.id_sectie
        bulk collect into v_nume, v_sectie
        from pacienti p, medici m, sectii s
        where p.id_medic_responsabil = v_medic
            and p.id_medic_responsabil = m.id_medic
            and s.id_sectie = m.id_sectie
            and p.id_pacient <> v_pacient;

        for i in v_nume.first..v_nume.last loop
            idx := idx + 1;
            dbms_output.put_line( v_nume(i) || ' ' || v_sectie(i) );
        end loop;
    return idx;

end f8;


procedure f9
(v_pacient pacienti.nume%type)

is

v_nume medici.nume%type;
v_prenume medici.prenume%type;
v_parcare parcari.nume%type;
v_id pacienti.id_pacient%type;


begin
    select id_pacient
    into v_id
    from pacienti
    where v_pacient = nume;


    select m.nume, m.prenume, p.nume
        into v_nume, v_prenume, v_parcare
        from medici m, pacienti pac, parcari p, tranzactii t, fise_medicale fm
        where  pac.id_pacient = v_pacient
```

```
            and t.id_medic_responsabil = m.id_medic and t.id_fisa_medicala = fm.id_fisa_medicala
            and fm.id_pacient = pac.id_pacient and m.id_parcare = p.id_parcare;

    dbms_output.put_line(v_nume||' '||v_prenume||' cu parcarea '|| v_parcare  ||'!');


    exception
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Eroare in gasirea datelor cerute');
        WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Ambiguitate in procesul de aflare al rezultatelor');

    end f9;
end pack;
```



3.2 Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).