**Tema SGBD curs 8**

Ex1: Comparati din punct de vedere al optimizarii cele 3 tipuri de colectii. Acelasi exemplu rezolvat folosind fiecare dintre cele 3 colectii. Care varianta este optima?  
    ObservatiI: Utilizati dbms\_utility.get\_time. Veti lucra pe schema HR. Formulati in limbaj natural problema rezovata in cele 3 moduri.

**Rezolvare:**

Pentru compararea din punct de vedere al optimizarii pentru cele 3 tipuri de colectii (tabel imbricat, tabel indexat, vector) vom rezolva aceeasi problema pentru compararea si folosirea corecta a functiei “dbms\_utility.get\_time”

Enunt: Dorim sa stocam informatii despre angajati din cadrul unei companii. Informatiile ce trebuiesc stocate sunt numele, prenumele si salariul.

**→folosind tabele indexate**

**Cod SQL:**

--E1 CU tabel indexat

declare

type emp\_tab\_ind is table of varchar2(100) index by pls\_integer;

info\_emp emp\_tab\_ind;

start\_time number;

begin

start\_time := dbms\_utility.get\_time;

for emp\_rec in (select employee\_id, first\_name, last\_name, salary

from employees e

where (select count(\*) from job\_history j where e.employee\_id = j.employee\_id) >1)

loop

info\_emp(emp\_rec.employee\_id) := emp\_rec.first\_name || ' ' || emp\_rec.last\_name || ': ' || emp\_rec.salary;

end loop;

--folosesc acest for doar pt comparatie nu pt logica programului

for i in 1 .. 1000000 --fara for imi afisa timp 0

loop --am cautat solutii si am gasit ca un for de acest gen ar putea

--produce un delay de aprox o secunda, astfel timpul total - 1 ar fi timpul folosit de program

null; --doar pt delay deci nu facem nimic

end loop;

--afisare info

for i in info\_emp.first .. info\_emp.last

loop

if info\_emp.exists(i) then

dbms\_output.put\_line('Angajatul cu id-ul ' || i || ' a fost angajat de cel putin doua ori');

dbms\_output.put\_line(info\_emp(i));

dbms\_output.new\_line;

end if;

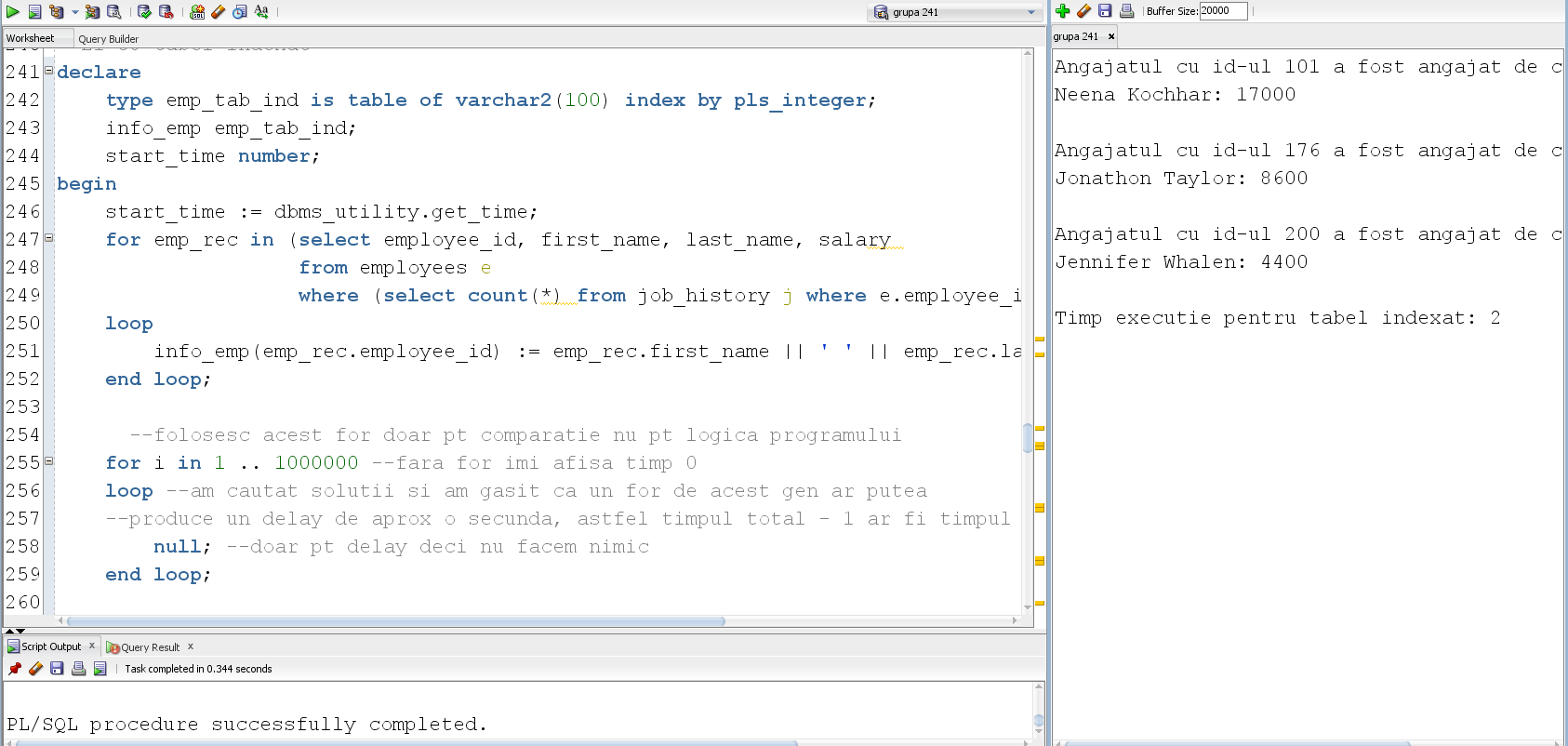
end loop;

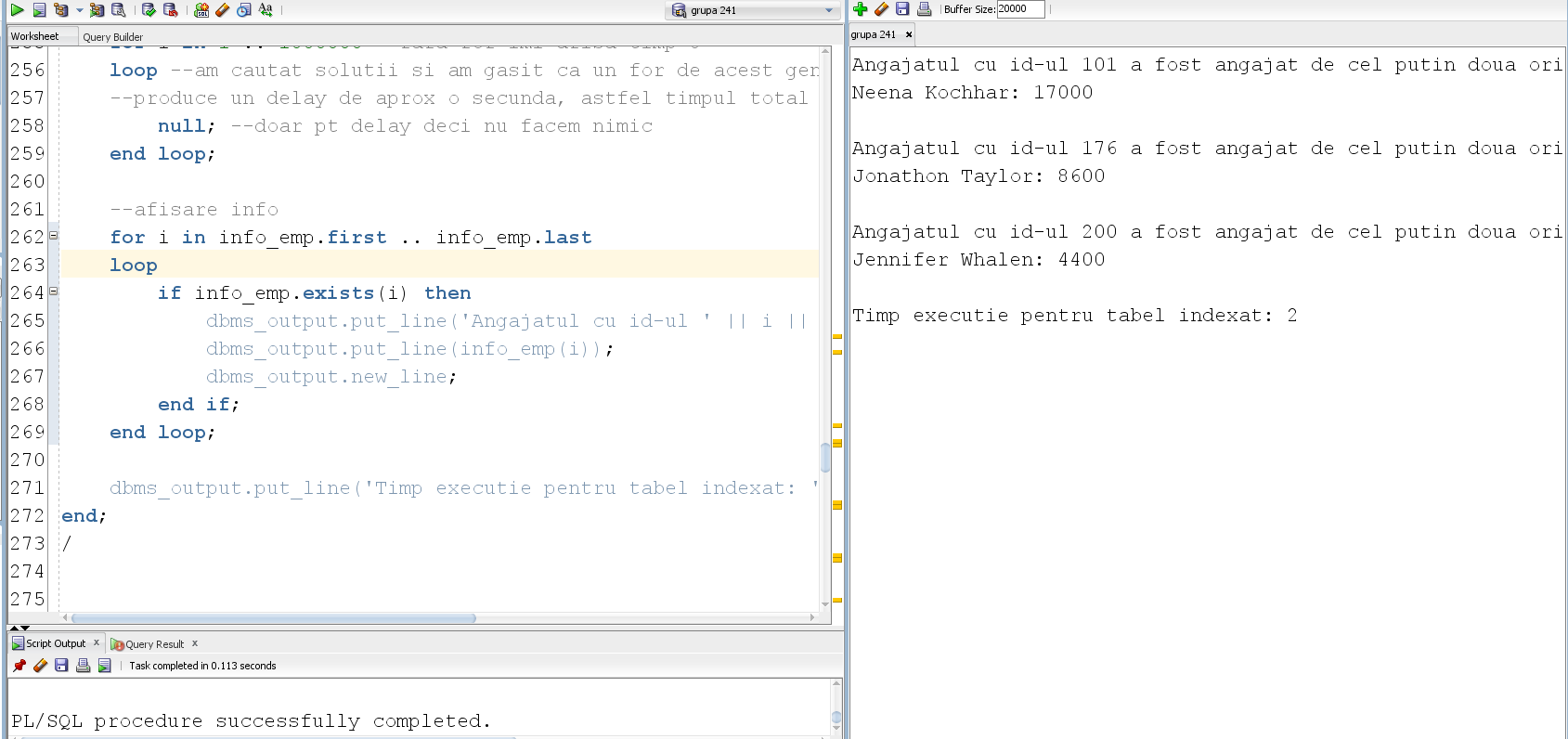
dbms\_output.put\_line('Timp executie pentru tabel indexat: ' || (dbms\_utility.get\_time - start\_time));

end;

/

**Print screen-uri:**

****

****

**→folosind tabele imbricate**

**Cod SQL:**

--cu tabele imbricate

declare

type emp\_tab\_imbr is table of varchar2(100);

info\_emp emp\_tab\_imbr;

start\_time number;

begin

start\_time := dbms\_utility.get\_time;

--salvare date in tabel imbricat

select first\_name || ' ' || last\_name || ': ' ||salary

bulk collect into info\_emp

from employees e

where (select count(\*) from job\_history j where e.employee\_id = j.employee\_id) >1 ;

--folosesc acest for doar pt comparatie nu pt logica programului

for i in 1 .. 1000000 --fara for imi afisa timp 0

loop --am cautat solutii si am gasit ca un for de acest gen ar putea

--produce un delay de aprox o secunda, astfel timpul total - 1 ar fi timpul folosit de program

null; --doar pt delay deci nu facem nimic

end loop;

for i in info\_emp.first .. info\_emp.last

loop

dbms\_output.put\_line('Angajatul ' || i || ' a fost angajat de cel putin doua ori');

dbms\_output.put\_line(info\_emp(i));

dbms\_output.new\_line;

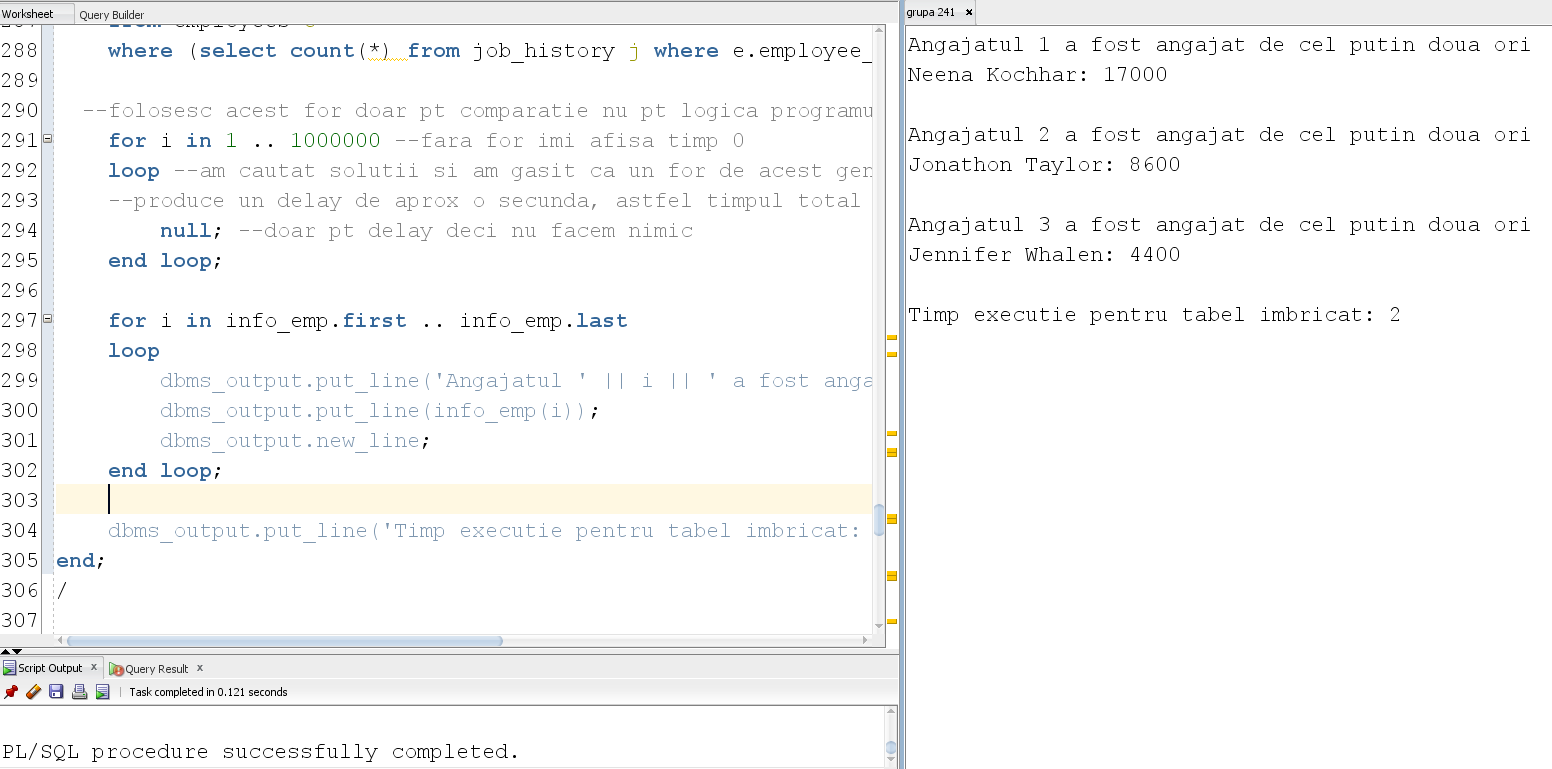
end loop;

dbms\_output.put\_line('Timp executie pentru tabel imbricat: ' || (dbms\_utility.get\_time - start\_time));

end;

/

**Print screen-uri:**

****

**→folosind vectori**

**Cod SQL:**

--vector

declare

type emp\_v is varray(100) of varchar2(100);

info\_emp emp\_v;

start\_time number;

begin

start\_time := dbms\_utility.get\_time;

--salvare informatii in vector

select first\_name || ' ' || last\_name || ': ' || salary

bulk collect into info\_emp

from employees e

where (select count(\*) from job\_history j where e.employee\_id = j.employee\_id) > 1;

--folosesc acest for doar pt comparatie nu pt logica programului

for i in 1 .. 1000000 --fara for imi afisa timp 0

loop --am cautat solutii si am gasit ca un for de acest gen ar putea

--produce un delay de aprox o secunda, astfel timpul total - 1 ar fi timpul folosit de program

null; --doar pt delay deci nu facem nimic

end loop;

--afisare informatii din vector

for i in 1 .. info\_emp.count

loop

dbms\_output.put\_line('Angajatul ' || i || 'a fost angajat de cel putin doua ori');

dbms\_output.put\_line(info\_emp(i));

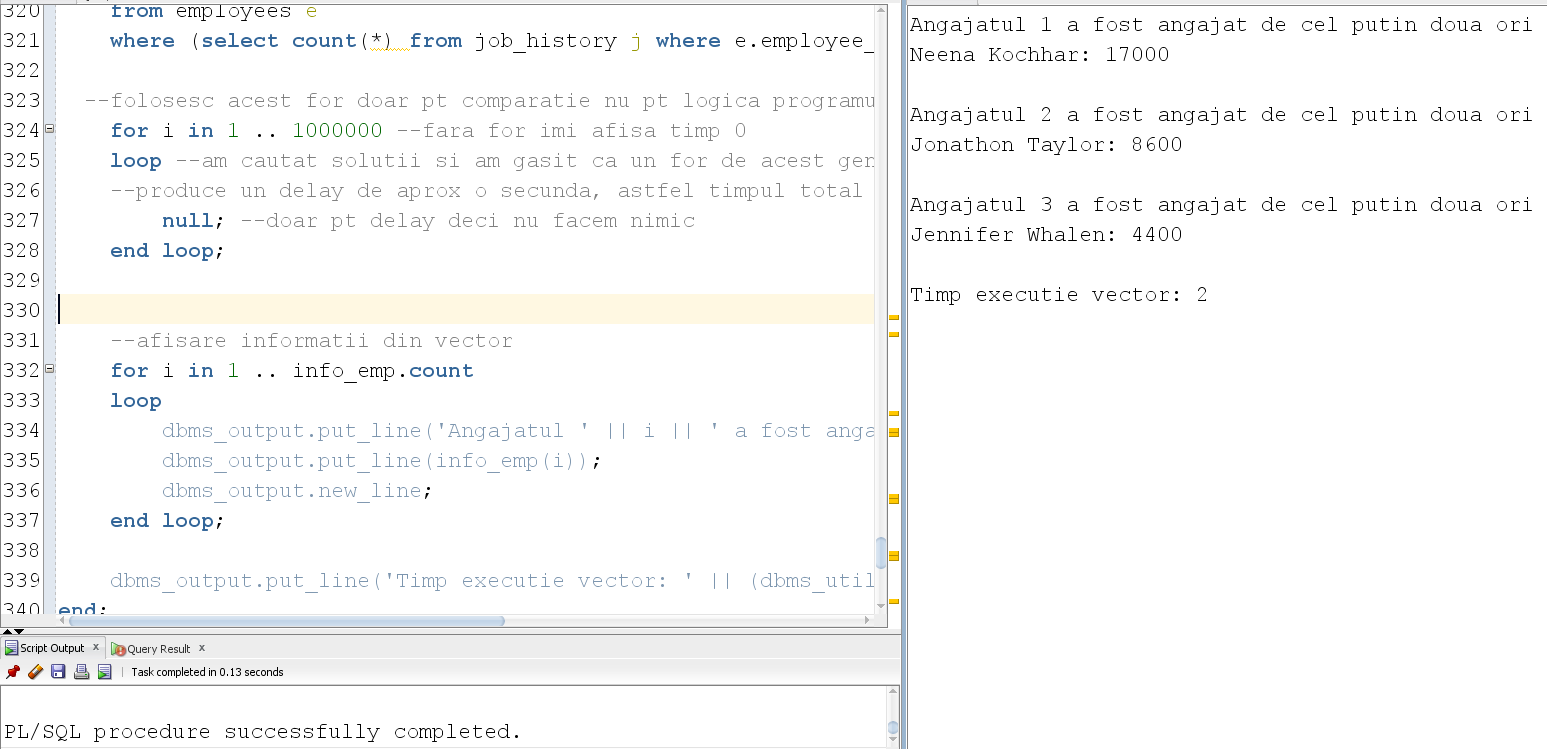
dbms\_output.new\_line;

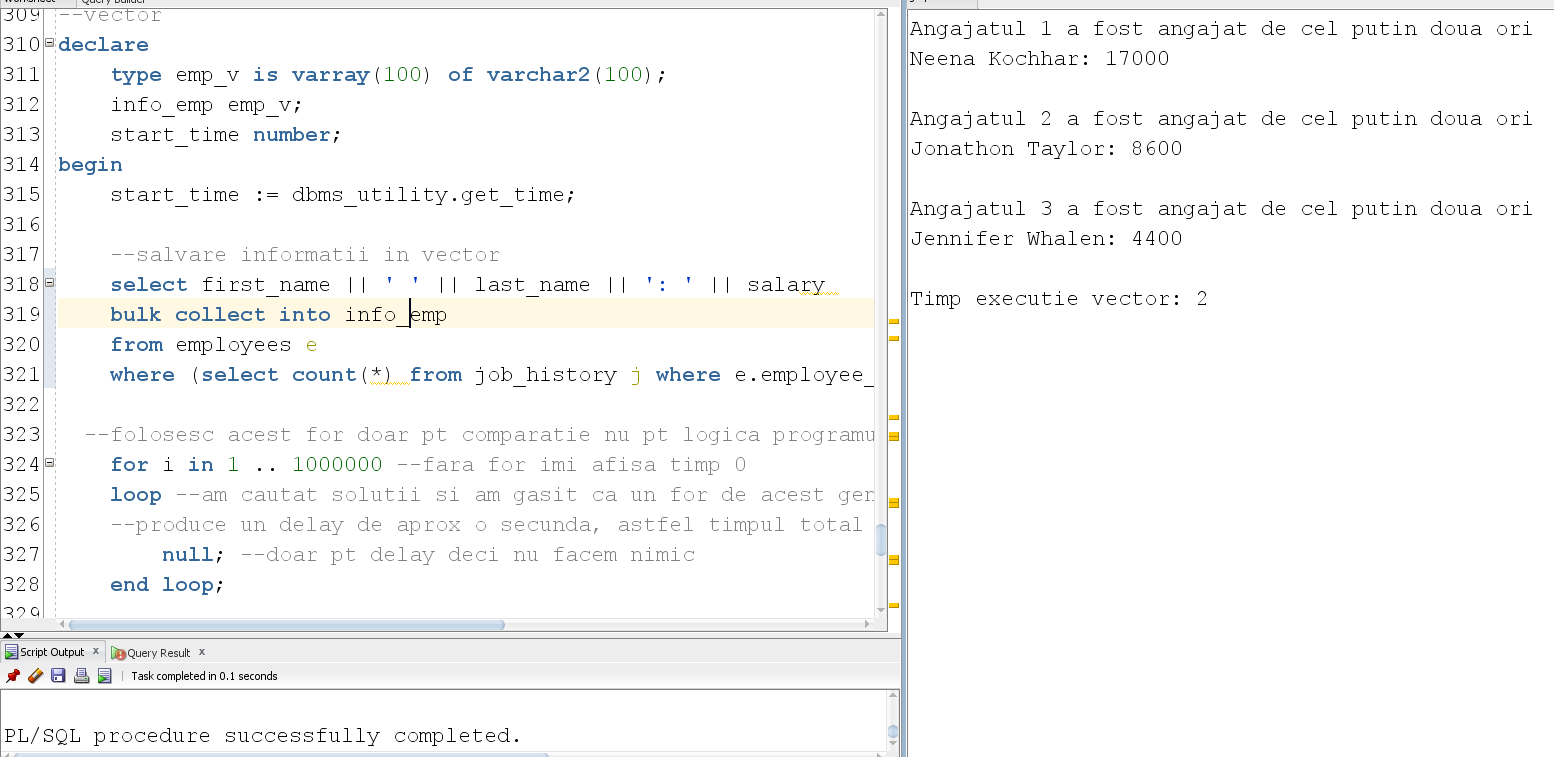
end loop;

dbms\_output.put\_line('Timp executie vector: ' || (dbms\_utility.get\_time - start\_time));

end;

/

**Print screen-uri: **

****

**Obs:** Am folosit un for care simuleaza operatii timp de o secunda deoarece fara acest artificiu, timpul pentru fiecare din cele 3 exemple ar fi fost 0. Cu acest for am reusit sa aproximez rezultatele in asa fel incat sa imi pot da seama ca utilizarea vectorului este cea mai ineficienta dintre cele 3, deoarece este singurul exemplu care, cu ajutorul artificiului cu for de o secunda, a crescut timpul executiei la 2; in celelalte cazuri acesta este 1.

Ex2: Pe schema video, compuneti in limbaj natural o problema a carei rezolvare implica folosirea a doua cursoare dintre care unul sa fie cursor parametrizat. Rezolvati aceeasi problema folosind doua cursoare clasice, doua ciclu cursoare si expresii cursor.

Enunt: Afisati informatii despre membrii(id, nume, oras) care au returnat titluri mai devreme intr-o perioada specificata si numarul de zile in avans in care au inapoiat titlurile.

**Cod SQL:**

--E2

declare

--declarare cursor cu parametri pentru start si end date (perioada in care cautam)

cursor cursor\_returnari (p\_start\_date date, p\_end\_date date) is

select r.member\_id, r.act\_ret\_date, r.exp\_ret\_date,

(r.act\_ret\_date - r.exp\_ret\_date) as zile\_avans

from rental r

where r.act\_ret\_date < r.exp\_ret\_date and r.act\_ret\_date between p\_start\_date and p\_end\_date;

v\_member\_id member.member\_id%type;

v\_act\_ret\_date rental.act\_ret\_date%type;

v\_exp\_ret\_date rental.exp\_ret\_date%type;

v\_zile\_avans number;

cursor c\_info\_memb is

select member\_id, last\_name, first\_name, city

from member

where member\_id = v\_member\_id;

begin

dbms\_output.put\_line('Lista membri care au returnat titlurile mai devreme decat termenul limita');

open cursor\_returnari(SYSDATE - 300, SYSDATE);

loop

fetch cursor\_returnari into v\_member\_id, v\_act\_ret\_date, v\_exp\_ret\_date, v\_zile\_avans;

exit when cursor\_returnari%NOTFOUND;

dbms\_output.put\_line('Id: ' || v\_member\_id);

dbms\_output.put\_line('Data returnarii: ' || v\_act\_ret\_date);

dbms\_output.put\_line('Termenul returnarii: ' || v\_exp\_ret\_date);

dbms\_output.put\_line('Zile in avans: ' || abs(v\_zile\_avans));

--deschidere cursor neparametrizat

open c\_info\_memb;

declare

v\_member\_info c\_info\_memb%ROWTYPE;

begin

loop

fetch c\_info\_memb into v\_member\_info;

exit when c\_info\_memb%NOTFOUND;

dbms\_output.put\_line('Nume: ' || v\_member\_info.last\_name || ' ' || v\_member\_info.first\_name);

dbms\_output.put\_line('Oras: ' || v\_member\_info.city);

dbms\_output.new\_line;

end loop;

end;

--inchidere cursor neparametrizat

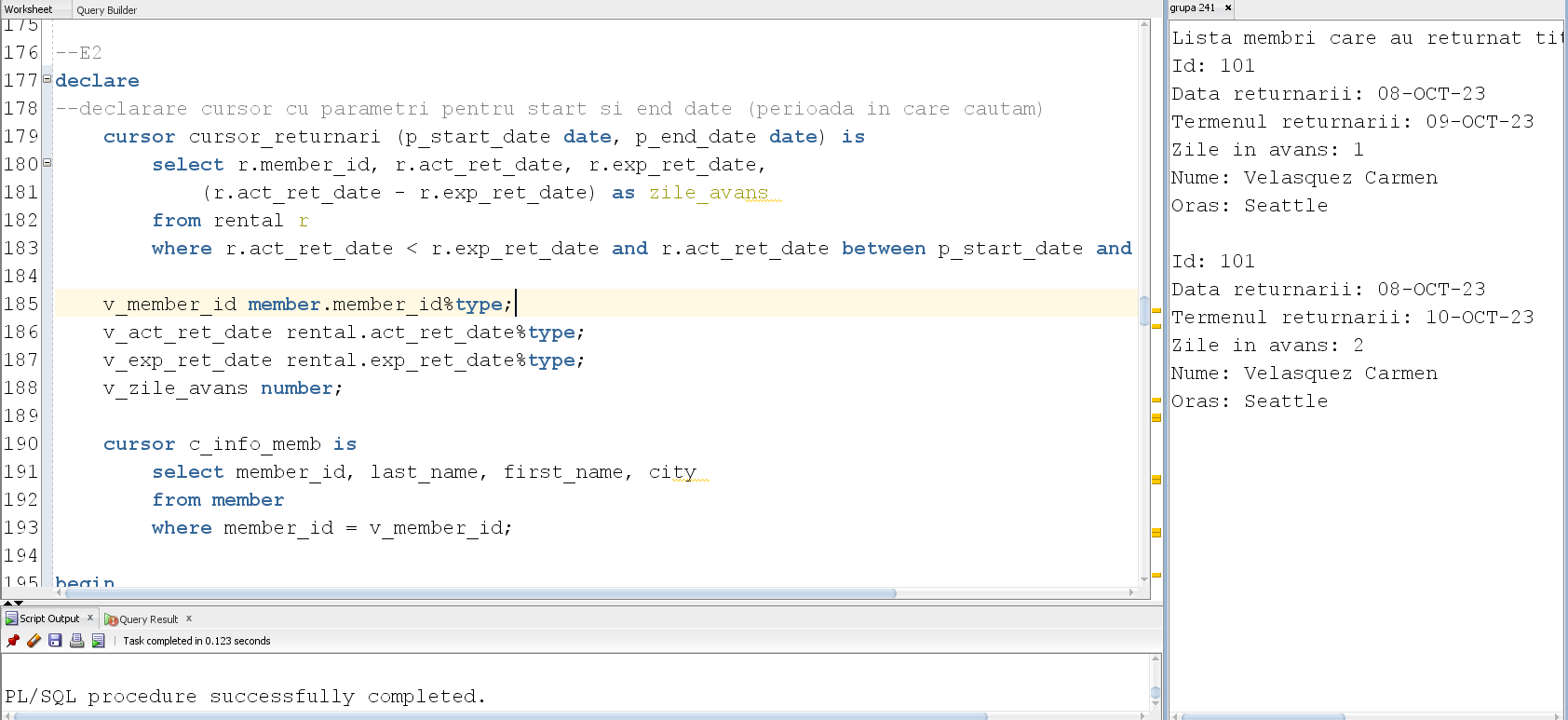
close c\_info\_memb;

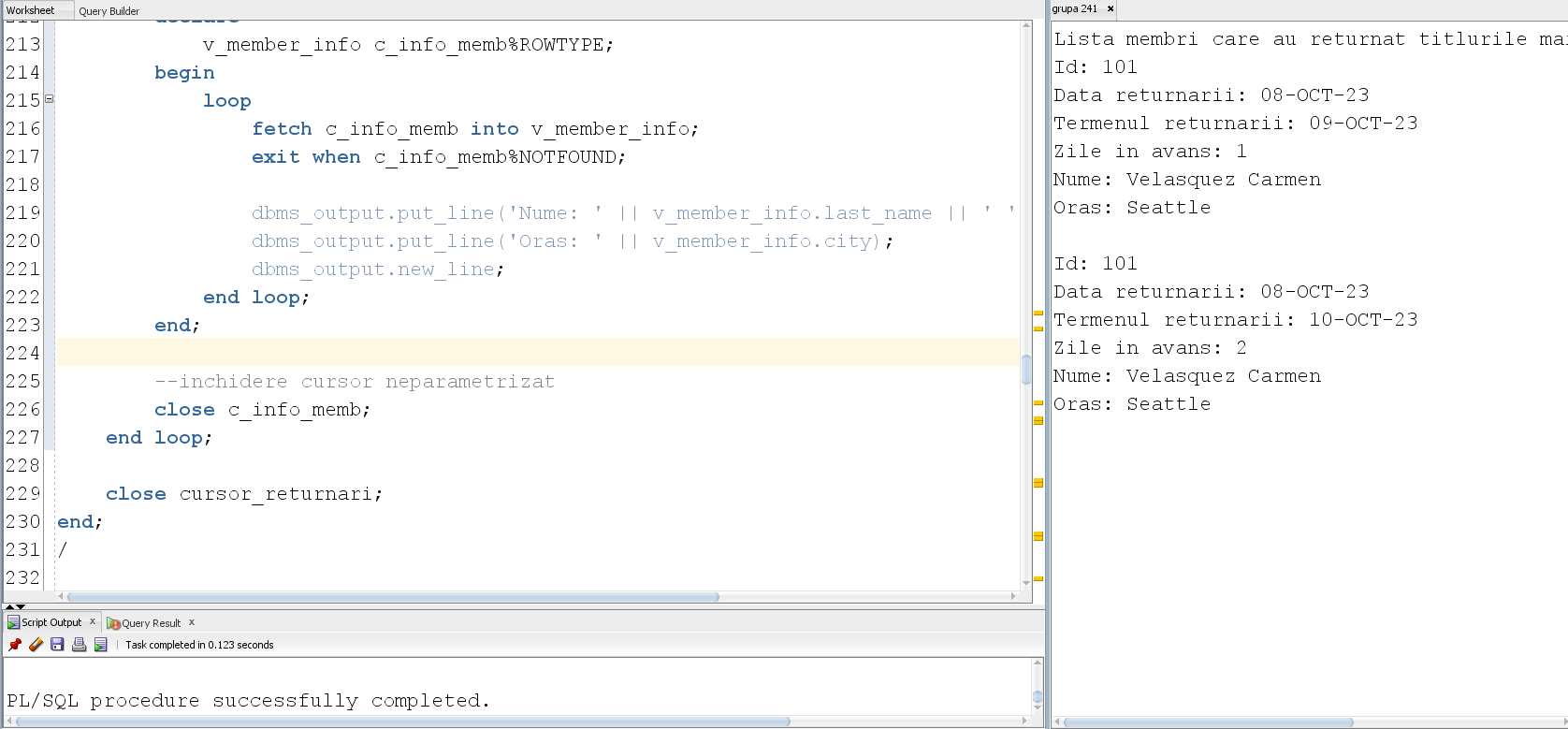
end loop;

close cursor\_returnari;

end;

/

**Print screen-uri:**

****

Ex3: Pe schema video, exemplificati, ca la curs, ce se intampla cand avem doua sesiuni si folosim  select...for update in prima sesiune, iar in a doua avem select...for update nowait/ wait n / skip locked.

**Rezolvare:** La folosirea ‘select … for update’ intr-o sesiune, aceasta instructiune marcheaza inregistrarile din select pentru actualizare si blocheaza liniile pana la incheierea tranzactiei curente. Daca in alta sesiune se face o alta operatie ‘select … for update nowait’ si o inregistrare este deja blocata pentru actualizare (din prima sesiune) atunci a doua sesiune va primi o eroare.

**Cod SQL:**

--E3

--sesiunea 1

--blocarea unei inregistrari specifice

begin

for r in (select \* from rental where title\_id = 93 for update) loop

dbms\_output.put\_line('Sesiunea 1: ' || r.book\_date);

end loop;

--commit; -- la finalul tranz

end;

/

--sesiunea 2

--incercare blocare aceeasi inregistrare fara asteptare (nowait)

begin

for r in (select \* from rental where rental\_id = 10 for update nowait) loop

--daca nu se asteapta -> eroare imediat ca nu se poate bloca instructiunea

end loop;

exception

when others then

dbms\_output.put\_line('Eroare (NOWAIT)');

end;

/

--incercare blocare alta instructiune cu wait

begin

for r in (select \* from rental where rental\_id = 1000 for update wait 10) loop

--se asteapta pana cand se poate bloca instructiunea sau pana se termina timeout specificat

dbms\_output.put\_line('Sesiunea 2: ' || r.book\_date);

end loop;

exception

when others then

dbms\_output.put\_line('Eroare (WAIT)');

end;

/