Enescu Irina Stefania
Grupa 233

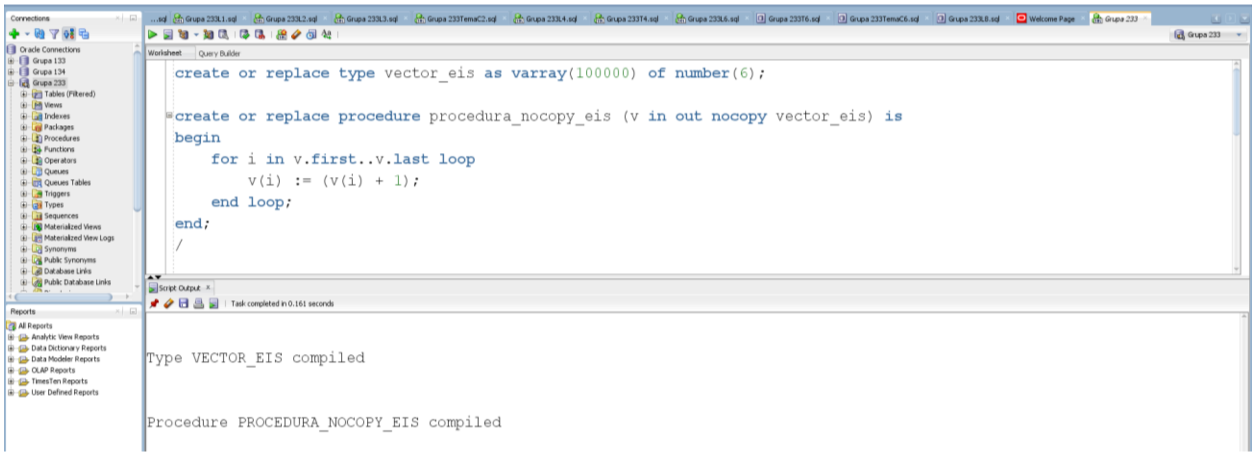
Tema Curs SGDB

(24 noiembrie 2022)

1. Sa se compare timpul de executie al unei proceduri in momentul in care parametrul se transmite cu IN OUT NOCOPY, respectiv cu IN OUT.

Cream o procedura cu NOCOPY, transmitand parametrul prin referinta:

```
create or replace type vector_eis as varray(100000) of number(6);

create or replace procedure procedura_nocopy_eis (v in out nocopy vector_eis) is
begin
   for i in v.first..v.last loop
      v(i) := (v(i) + 1);
   end loop;
end;
/
```

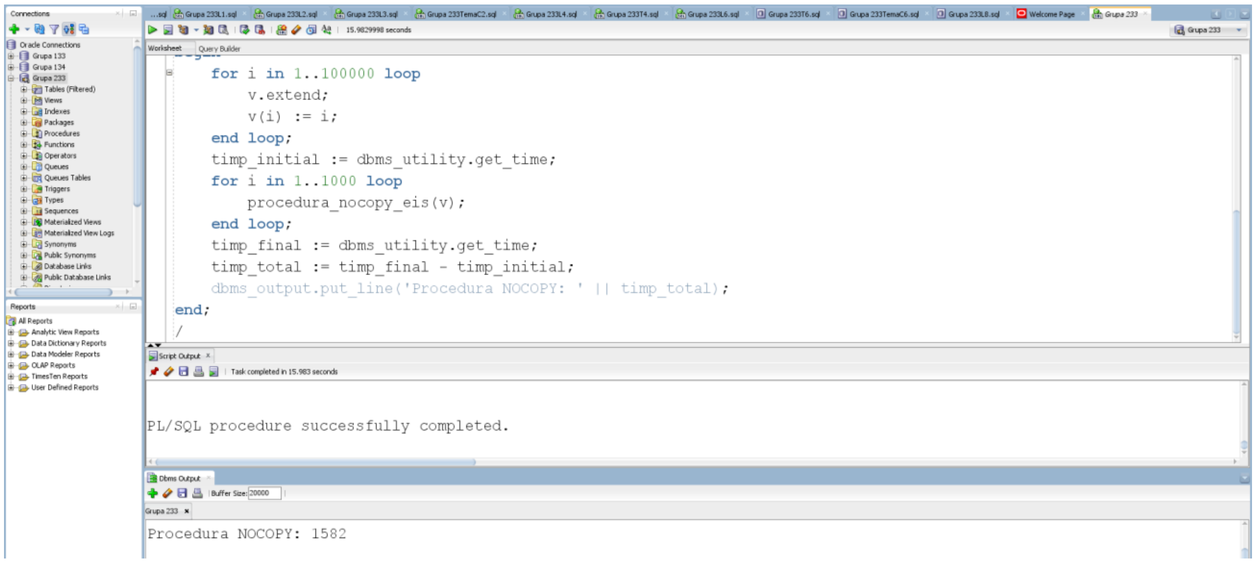


```
create or replace type vector_eis as varray(100000) of number(6);

create or replace procedure procedura_nocopy_eis (v in out nocopy vector_eis) is
begin
   for i in v.first..v.last loop
      v(i) := (v(i) + 1);
   end loop;
end;
/
```

```
Task completed in 0.161 seconds

Type VECTOR_EIS compiled


Procedure PROCEDURA_NOCOPY_EIS compiled
```

Apelam procedura de 1000 pentru a obtine diferente de timp considerabile:

```
declare
   v vector_eis := vector_eis();
   timp_initial number;
   timp_final number;
   timp_total number;
begin
   for i in 1..100000 loop
      v.extend;
      v(i) := i;
   end loop;
   timp_initial := dbms_utility.get_time;
   for i in 1..1000 loop
      procedura_nocopy_eis(v);
   end loop;
   timp_final := dbms_utility.get_time;
   timp_total := timp_final - timp_initial;
   dbms_output.put_line('Procedura NOCOPY: ' || timp_total);
end;
/
```



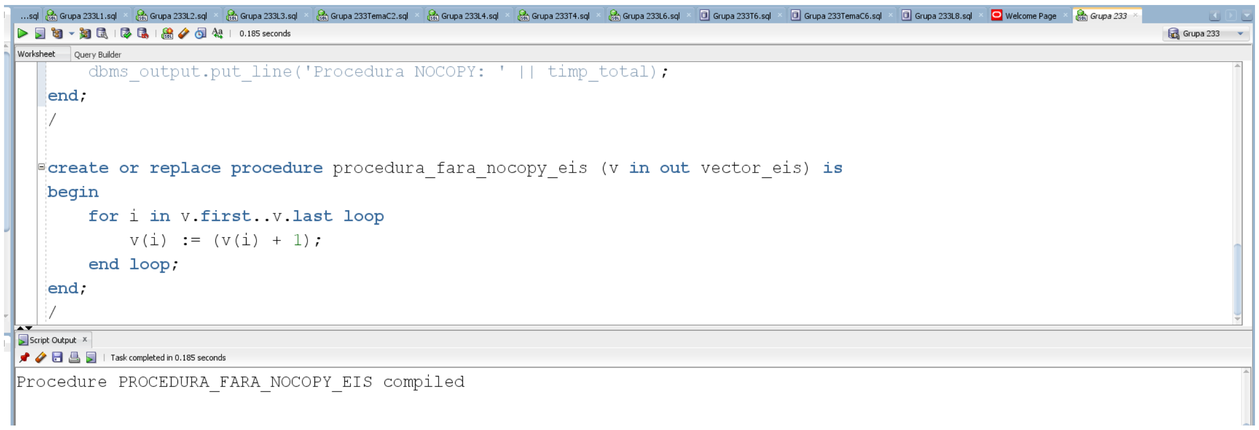
```
    for i in 1..100000 loop
        v.extend;
        v(i) := i;
    end loop;
    timp_initial := dbms_utility.get_time;
    for i in 1..1000 loop
        procedura_nocopy_eis(v);
    end loop;
    timp_final := dbms_utility.get_time;
    timp_total := timp_final - timp_initial;
    dbms_output.put_line('Procedura NOCOPY: ' || timp_total);
end;
/
```

```
Task completed in 15.983 seconds

PL/SQL procedure successfully completed.
```

```
Procedura NOCOPY: 1582
```

Cream o procedura fara NOCOPY, transmitand parametrul prin valoare:

```
create or replace procedure procedura_fara_nocopy_eis (v in out vector_eis) is
begin
   for i in v.first..v.last loop
      v(i) := (v(i) + 1);
   end loop;
end;
/
```

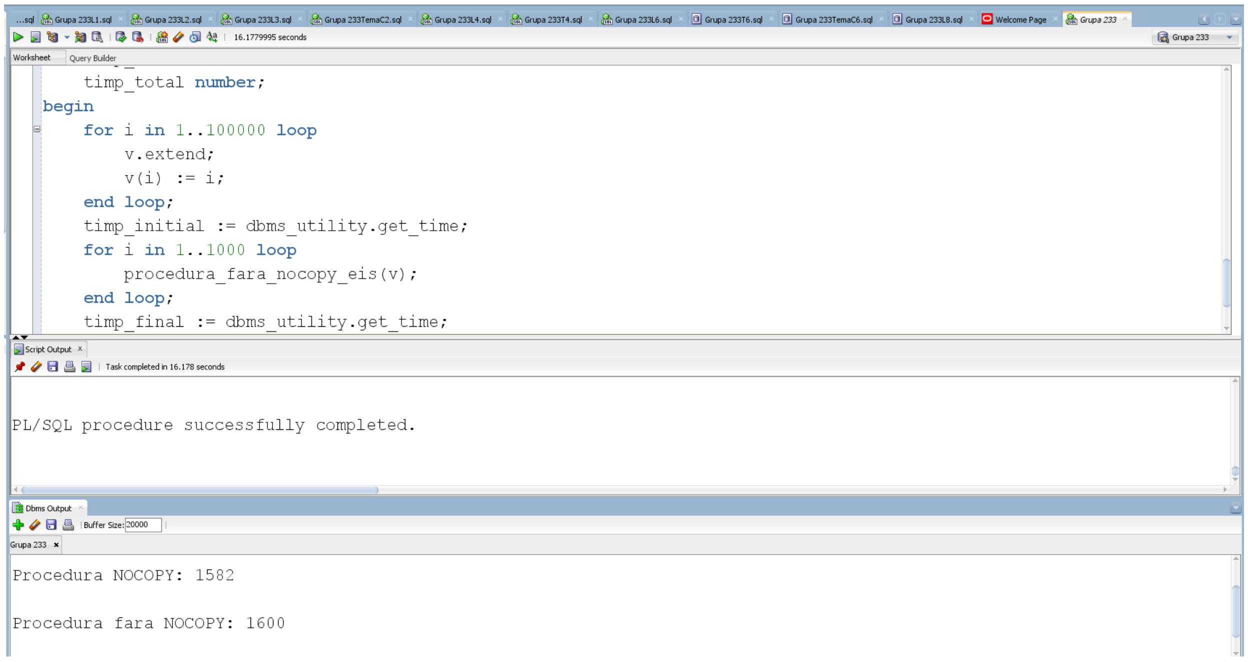


```
    dbms_output.put_line('Procedura NOCOPY: ' || timp_total);
end;
/

create or replace procedure procedura_fara_nocopy_eis (v in out vector_eis) is
begin
    for i in v.first..v.last loop
        v(i) := (v(i) + 1);
    end loop;
end;
/
```

```
Task completed in 0.185 seconds
Procedure PROCEDURA_FARA_NOCOPY_EIS compiled
```

Apelam procedura de 1000 pentru a obtine diferente de timp considerabile:

```
declare
   v vector_eis := vector_eis();
   timp_initial number;
   timp_final number;
   timp_total number;
begin
   for i in 1..100000 loop
      v.extend;
      v(i) := i;
   end loop;
   timp_initial := dbms_utility.get_time;
   for i in 1..1000 loop
      procedura_fara_nocopy_eis(v);
   end loop;
   timp_final := dbms_utility.get_time;
   timp_total := timp_final - timp_initial;
   dbms_output.put_line('Procedura fara NOCOPY: ' || timp_total);
end;
/
```



```
    timp_total number;
begin
    for i in 1..100000 loop
        v.extend;
        v(i) := i;
    end loop;
    timp_initial := dbms_utility.get_time;
    for i in 1..1000 loop
        procedura_fara_nocopy_eis(v);
    end loop;
    timp_final := dbms_utility.get_time;
```

```
PL/SQL procedure successfully completed.
```

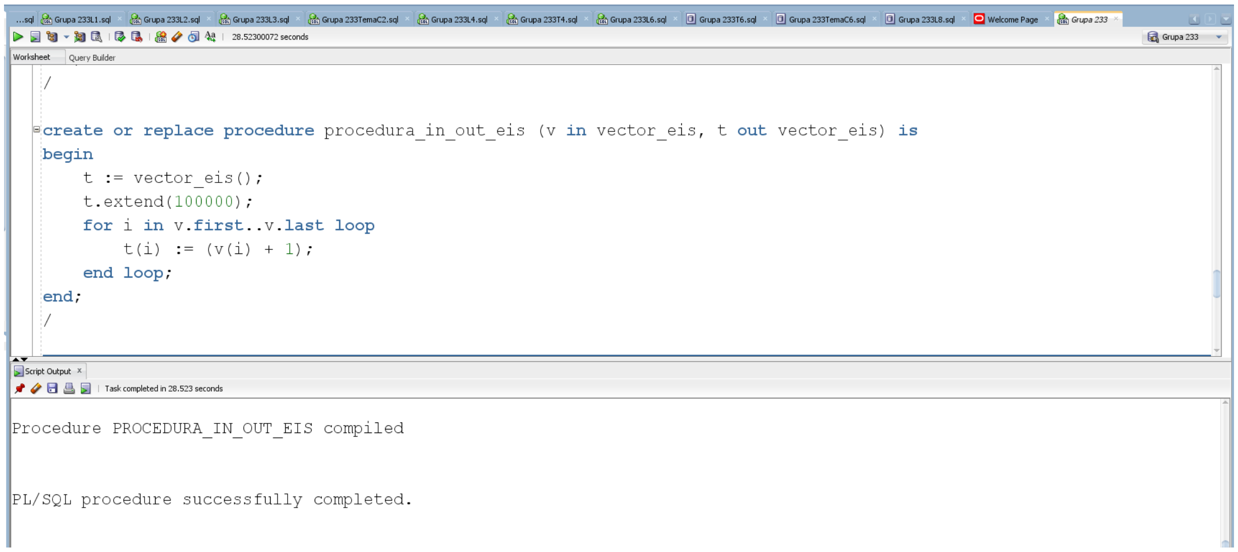
```
Procedura NOCOPY: 1582

Procedura fara NOCOPY: 1600
```

Concluzie: Procedura cu NOCOPY este mai rapida decat procedura fara NOCOPY.

2. La exercitiul anterior am observat cum se comporta proceduri cu parametrii IN OUT NOCOPY si IN OUT. Care sunt timpii de rulare in cazul in care transmitem parametrii cu IN - OUT si IN - IN OUT?

Cream procedura cu IN - OUT:

```
create or replace procedure procedura_in_out_eis (v in vector_eis, t out vector_eis) is
begin
   t := vector_eis();
   t.extend(100000);
   for i in v.first..v.last loop
      t(i) := (v(i) + 1);
   end loop;
end;
/
```



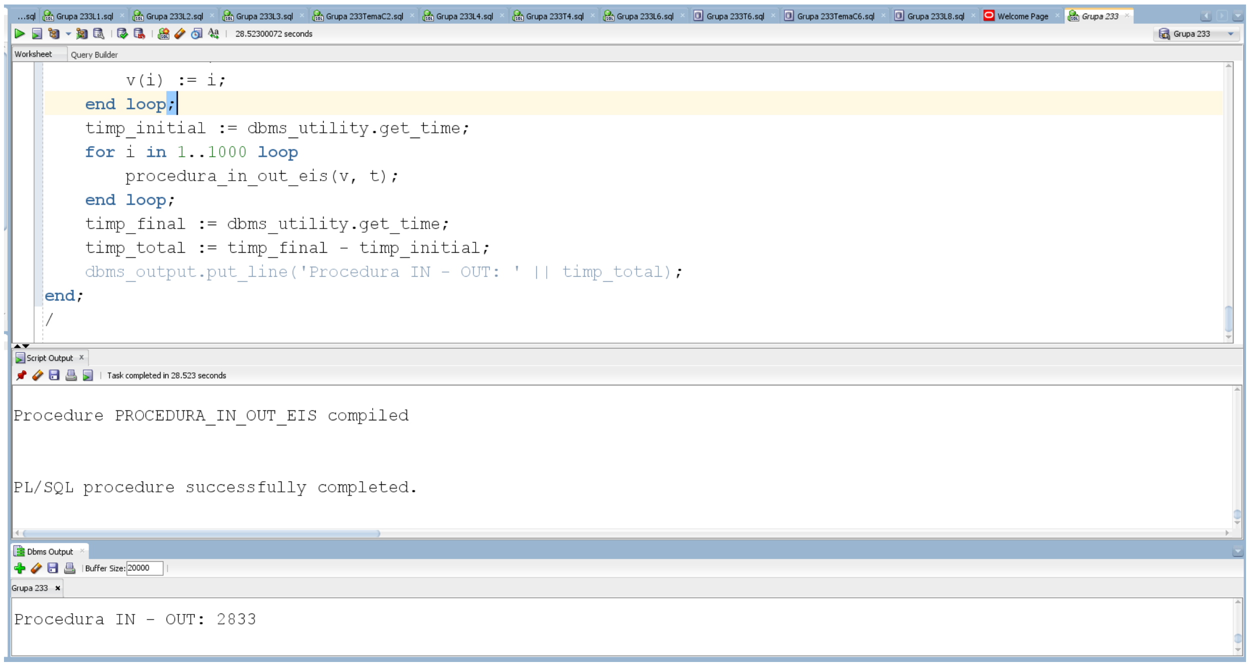
```
/

create or replace procedure procedura_in_out_eis (v in vector_eis, t out vector_eis) is
begin
    t := vector_eis();
    t.extend(100000);
    for i in v.first..v.last loop
        t(i) := (v(i) + 1);
    end loop;
end;
/
```

```
Procedure PROCEDURA_IN_OUT_EIS compiled


PL/SQL procedure successfully completed.
```

Apelam procedura de 1000 pentru a obtine diferente de timp considerabile:

```
declare
   v vector_eis := vector_eis();
   t vector_eis := vector_eis();
   timp_initial number;
   timp_final number;
   timp_total number;
begin
   for i in 1..100000 loop
      v.extend;
      v(i) := i;
   end loop;
   timp_initial := dbms_utility.get_time;
   for i in 1..1000 loop
      procedura_in_out_eis(v, t);
   end loop;
   timp_final := dbms_utility.get_time;
   timp_total := timp_final - timp_initial;
   dbms_output.put_line('Procedura IN - OUT: ' || timp_total);
```

end;
/



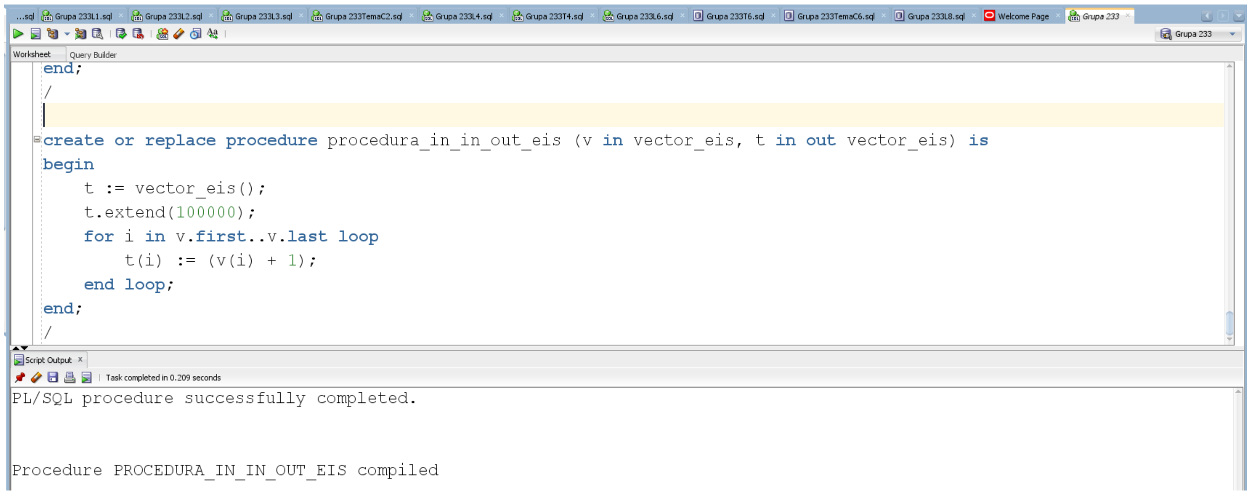
```
        v(i) := i;
    end loop;
    timp_initial := dbms_utility.get_time;
    for i in 1..1000 loop
        procedura_in_out_eis(v, t);
    end loop;
    timp_final := dbms_utility.get_time;
    timp_total := timp_final - timp_initial;
    dbms_output.put_line('Procedura IN - OUT: ' || timp_total);
end;
/
```

```
Procedure PROCEDURA_IN_OUT_EIS compiled


PL/SQL procedure successfully completed.
```

```
Procedura IN - OUT: 2833
```

Cream procedura cu IN - IN OUT:

```
create or replace procedure procedura_in_in_out_eis (v in vector_eis, t in out vector_eis) is
begin
    t := vector_eis();
    t.extend(100000);
    for i in v.first..v.last loop
        t(i) := (v(i) + 1);
    end loop;
end;
/
```

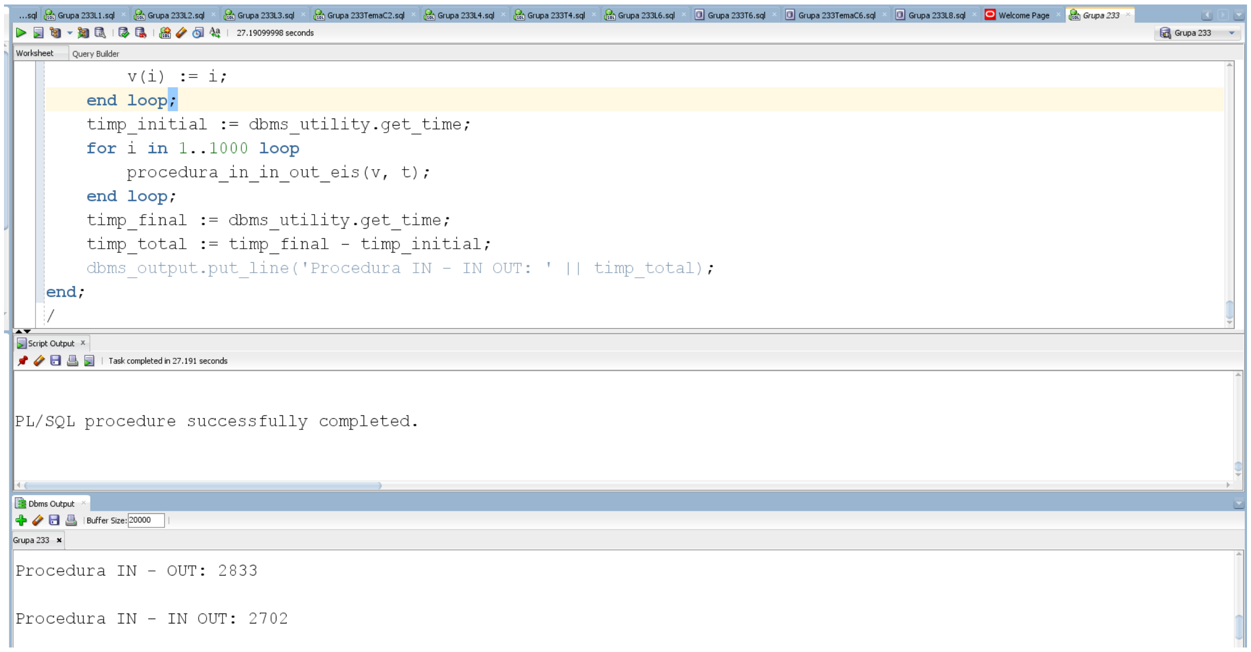


```
end;
/

create or replace procedure procedura_in_in_out_eis (v in vector_eis, t in out vector_eis) is
begin
    t := vector_eis();
    t.extend(100000);
    for i in v.first..v.last loop
        t(i) := (v(i) + 1);
    end loop;
end;
/
```

```
PL/SQL procedure successfully completed.


Procedure PROCEDURA_IN_IN_OUT_EIS compiled
```

Apelam procedura de 1000 pentru a obtine diferente de timp considerabile:

```
declare
   v vector_eis := vector_eis();
   t vector_eis := vector_eis();
   timp_initial number;
   timp_final number;
   timp_total number;
begin
   for i in 1..100000 loop
      v.extend;
      v(i) := i;
   end loop;
   timp_initial := dbms_utility.get_time;
   for i in 1..1000 loop
      procedura_in_in_out_eis(v, t);
   end loop;
   timp_final := dbms_utility.get_time;
   timp_total := timp_final - timp_initial;
   dbms_output.put_line('Procedura IN - IN OUT: ' || timp_total);
end;
/
```



```
        v(i) := i;
    end loop;
    timp_initial := dbms_utility.get_time;
    for i in 1..1000 loop
        procedura_in_in_out_eis(v, t);
    end loop;
    timp_final := dbms_utility.get_time;
    timp_total := timp_final - timp_initial;
    dbms_output.put_line('Procedura IN - IN OUT: ' || timp_total);
end;
/
```

Task completed in 27.191 seconds

```
PL/SQL procedure successfully completed.
```

```
Procedura IN - OUT: 2833

Procedura IN - IN OUT: 2702
```

Concluzie: Procedura cu IN - IN OUT este mai rapida decat procedura cu IN - OUT.

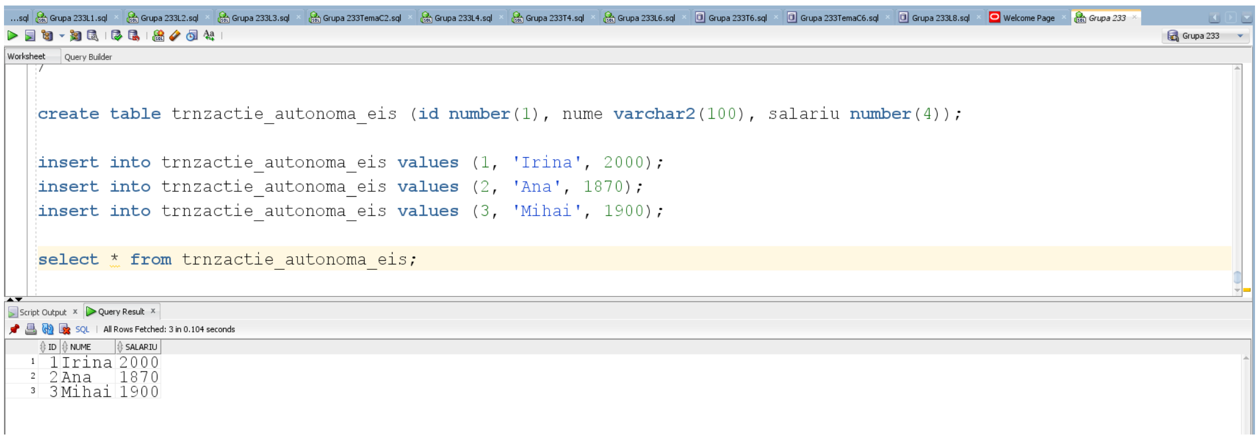
3. Tranzactii autonome. Ce se intampla in momentul in care dam commit intr-un bloc cu tranzactie autonoma si apoi rollback in exterior?

Cream un tabel auxiliar pe care sa lucram. Introducem cateva date:

create table trnzactie_autonoma_eis (id number(1), nume varchar2(100), salariu number(4));

insert into trnzactie_autonoma_eis values (1, 'Irina', 2000);
insert into trnzactie_autonoma_eis values (2, 'Ana', 1870);
insert into trnzactie_autonoma_eis values (3, 'Mihai', 1900);

select * from trnzactie_autonoma_eis;



```
create table trnzactie_autonoma_eis (id number(1), nume varchar2(100), salariu number(4));

insert into trnzactie_autonoma_eis values (1, 'Irina', 2000);
insert into trnzactie_autonoma_eis values (2, 'Ana', 1870);
insert into trnzactie_autonoma_eis values (3, 'Mihai', 1900);

select * from trnzactie_autonoma_eis;
```

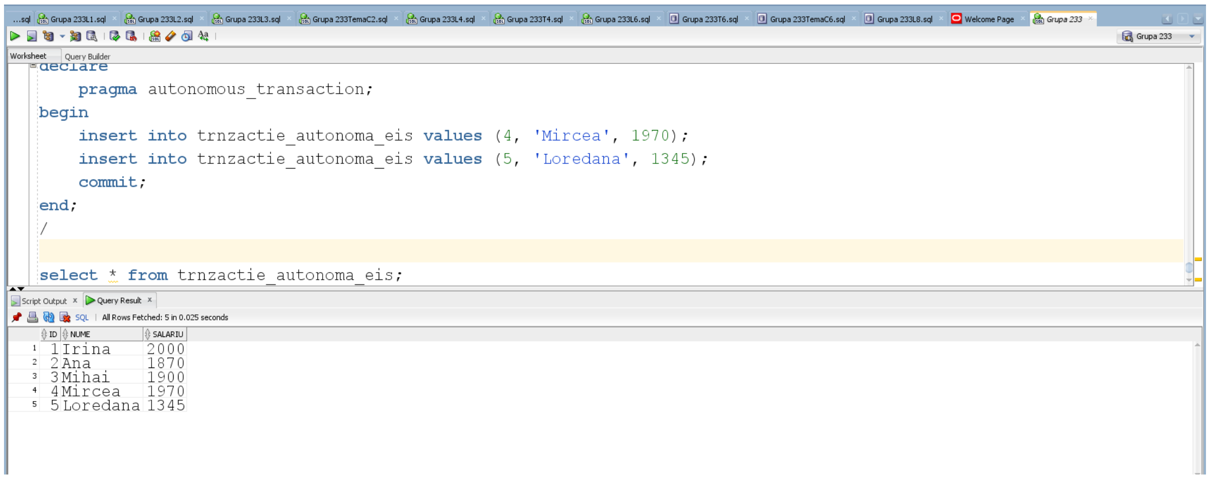
All Rows Fetched: 3 in 0.104 seconds

```
   ID NUME  SALARIU
1   1 Irina 2000
2   2 Ana   1870
3   3 Mihai 1900
```

Facem un bloc cu tranzactie autonoma in care mai inseram doua valori si dam commit:

```
declare
   pragma autonomous_transaction;
begin
   insert into trnzactie_autonoma_eis values (4, 'Mircea', 1970);
   insert into trnzactie_autonoma_eis values (5, 'Loredana', 1345);
   commit;
end;
/
```

select * from trnzactie_autonoma_eis;



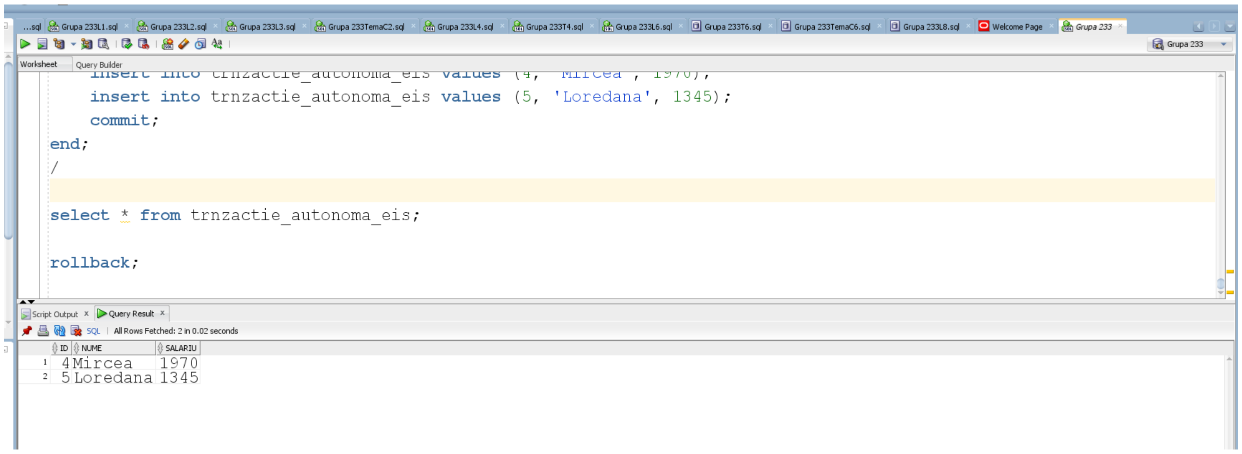
```
declare
    pragma autonomous_transaction;
begin
    insert into trnzactie_autonoma_eis values (4, 'Mircea', 1970);
    insert into trnzactie_autonoma_eis values (5, 'Loredana', 1345);
    commit;
end;
/

select * from trnzactie_autonoma_eis;
```

All Rows Fetched: 5 in 0.025 seconds

```
   ID NUME     SALARIU
1   1 Irina    2000
2   2 Ana      1870
3   3 Mihai    1900
4   4 Mircea   1970
5   5 Loredana 1345
```

Dam rollback:

rollback;
select * from trnzactie_autonoma_eis;



```
    insert into trnzactie_autonoma_eis values (4, 'Mircea', 1970);
    insert into trnzactie_autonoma_eis values (5, 'Loredana', 1345);
    commit;
end;
/

select * from trnzactie_autonoma_eis;

rollback;
```

All Rows Fetched: 2 in 0.02 seconds

```
  ID NUME     SALARIU
1  4 Mircea   1970
2  5 Loredana 1345
```

Concluzie: In momentul in care am facut commit in blocul cu tranzactie autonoma s-au salvat doar inregistrarile introduse in tabel in blocul respectiv. Toate celelalte inregistrari introduse anterior au disparut in momentul in care am dat rollback.