Tema Lab 6 PL/SQL – Ungureanu Matei

E5. Pe baza informațiilor din schemă creați și populați cu date următoarele două tabele:

- emp\_test\_\*\*\* (employee\_id – cheie primară, last\_name, first\_name, department\_id);

- dept\_test\_\*\*\* (department\_id – cheie primară, department\_name).

Definiți un declanșator care va determina ștergeri și modificări în cascadă:

- ștergerea angajaților din tabelul emp\_test\_\*\*\* dacă este eliminat departamentul acestora din tabelul dept\_test\_\*\*\*;

- modificarea codului de departament al angajaților din tabelul emp\_test\_\*\*\* dacă departamentul respectiv este modificat în tabelul dept\_test\_\*\*\*.

Testați și rezolvați problema în următoarele situații:

- nu este definită constrângere de cheie externă între cele două tabele;

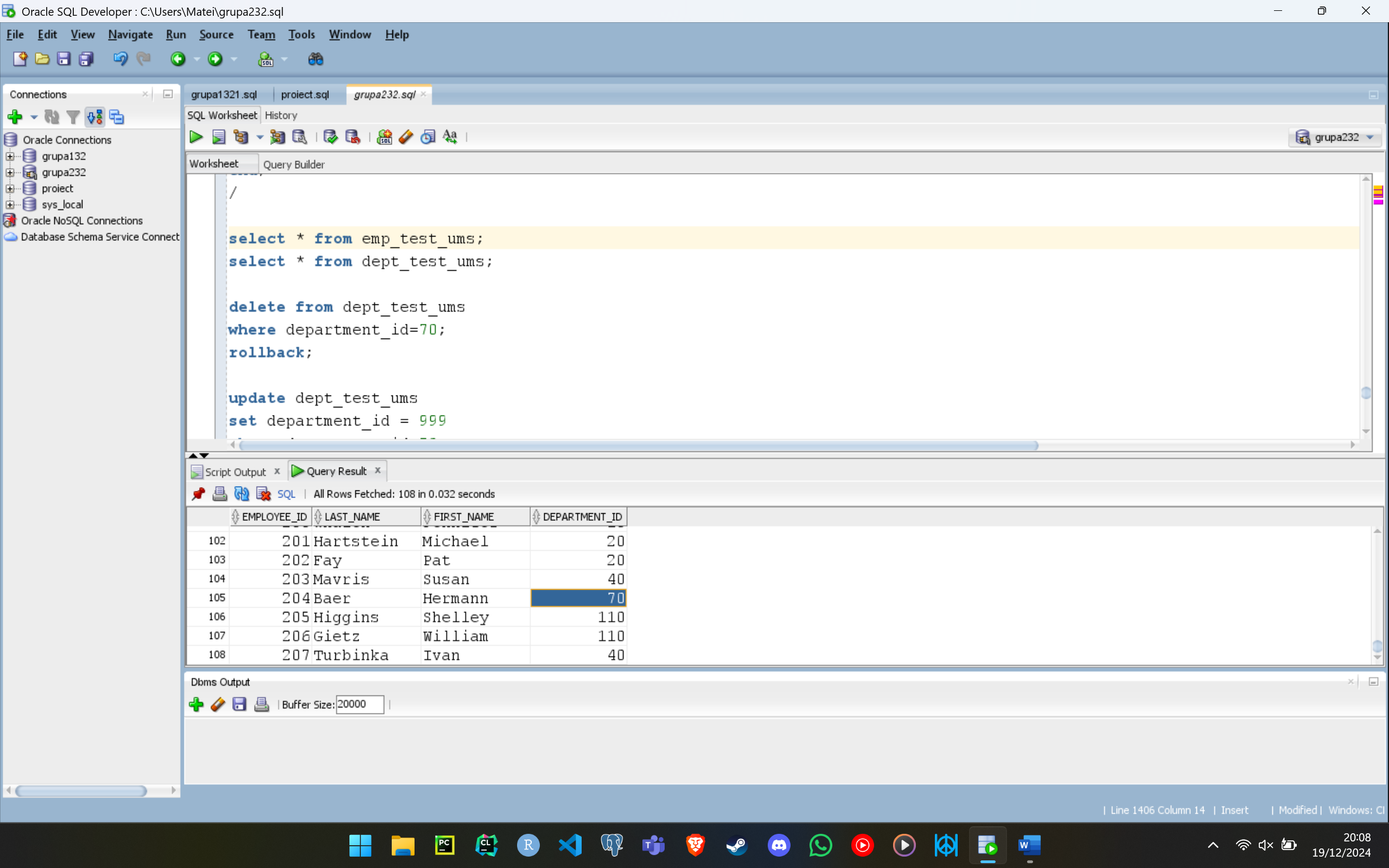
- este definită constrângerea de cheie externă între cele două tabele;

- este definită constrângerea de cheie externă între cele două tabele cu opțiunea ON DELETE CASCADE;

- este definită constrângerea de cheie externă între cele două tabele cu opțiunea ON DELETE SET NULL.

Comentați fiecare caz în parte.

Departamentul ales: 70 (are doar un angajat)



--fara FK

create table dept\_test\_ums (

department\_id number(3) primary key,

department\_name varchar2(50)

);

create table emp\_test\_ums (

employee\_id number(3) primary key,

last\_name varchar2(50),

first\_name varchar2(50),

department\_id number(3)

);

begin

for i in (

select department\_id, department\_name

from departments

) loop

insert into dept\_test\_ums (department\_id, department\_name)

values (i.department\_id, i.department\_name);

end loop;

for i in (

select employee\_id, last\_name, first\_name, department\_id

from employees

) loop

insert into emp\_test\_ums (employee\_id, last\_name, first\_name, department\_id)

values (i.employee\_id, i.last\_name, i.first\_name, i.department\_id);

end loop;

end;

/

commit;

create or replace trigger tema6ex5\_ums

after delete or update of department\_id on dept\_test\_ums

for each row

begin

if deleting then

delete from emp\_test\_ums

where department\_id = :old.department\_id;

elsif updating then

update emp\_test\_ums

set department\_id = :new.department\_id

where department\_id = :old.department\_id;

end if;

end;

/

select \* from emp\_test\_ums;

select \* from dept\_test\_ums;

delete from dept\_test\_ums

where department\_id=70;

rollback;

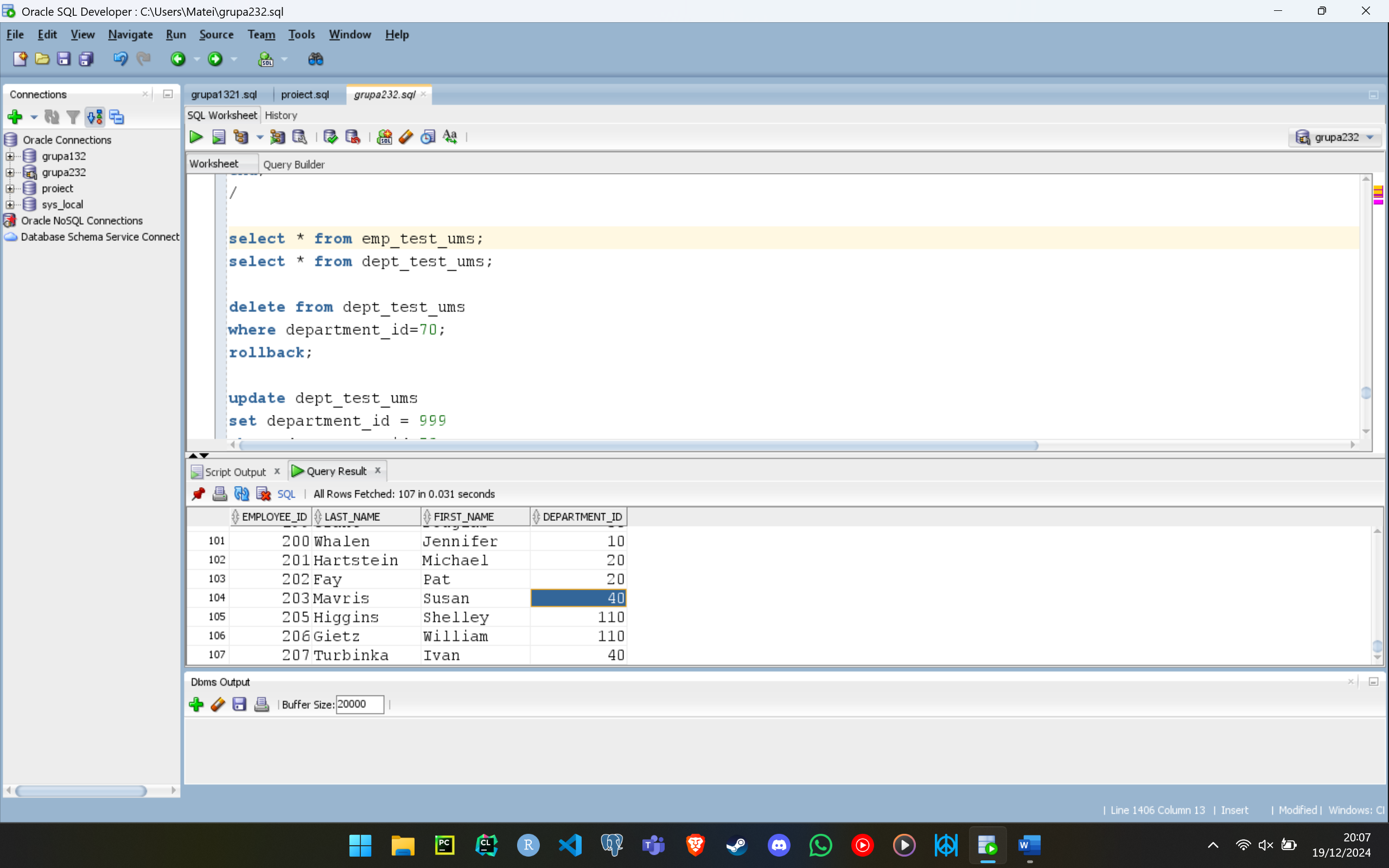
update dept\_test\_ums

set department\_id = 999

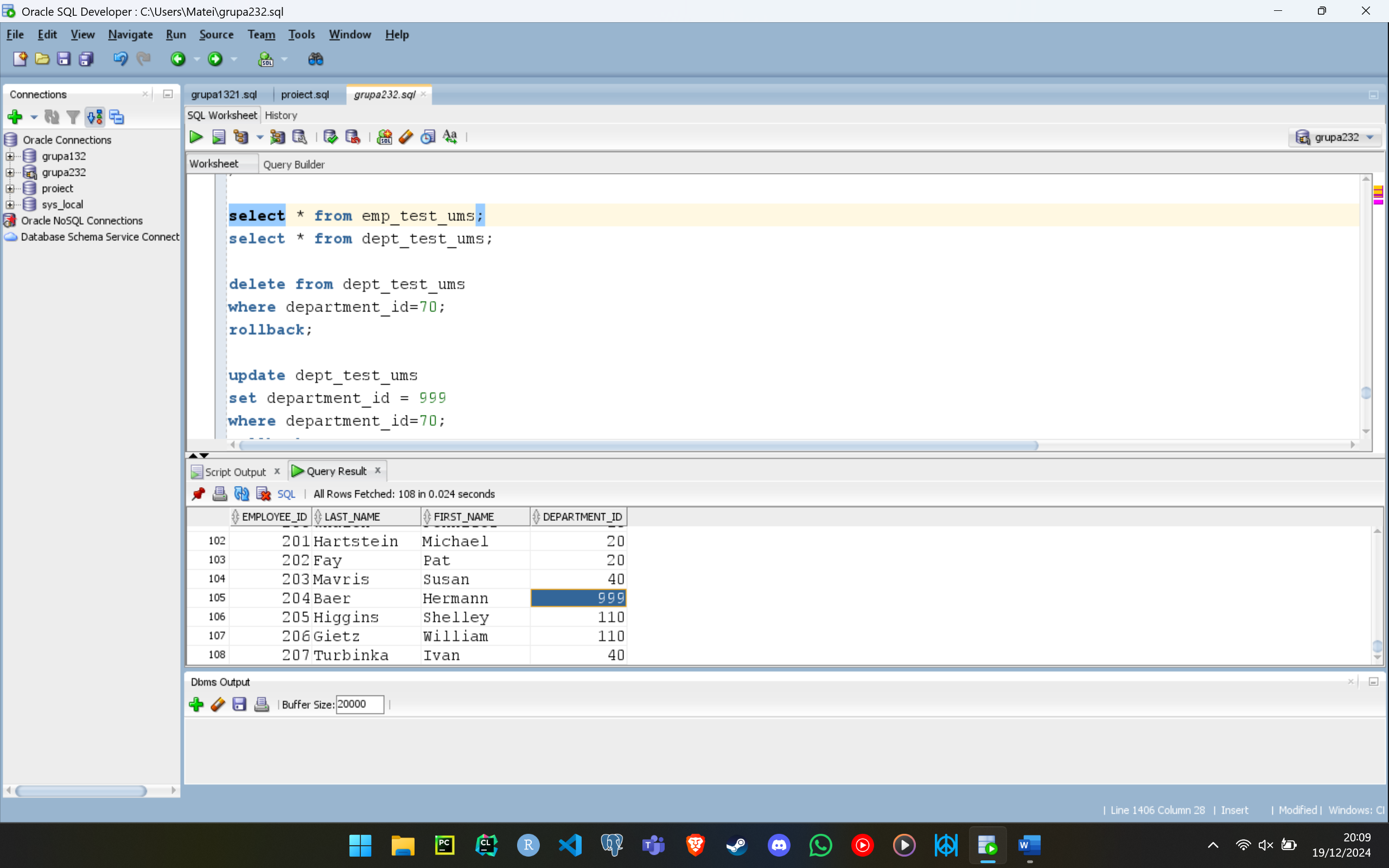
where department\_id=70;

rollback;

--dupa delete:



--dupa update:



--cu FK

create table dept\_test\_ums (

department\_id number(3) primary key,

department\_name varchar2(50)

);

create table emp\_test\_ums (

employee\_id number(3) primary key,

last\_name varchar2(50),

first\_name varchar2(50),

department\_id number(3) references dept\_test\_ums(department\_id)

);

begin

for i in (

select department\_id, department\_name

from departments

) loop

insert into dept\_test\_ums (department\_id, department\_name)

values (i.department\_id, i.department\_name);

end loop;

for i in (

select employee\_id, last\_name, first\_name, department\_id

from employees

) loop

insert into emp\_test\_ums (employee\_id, last\_name, first\_name, department\_id)

values (i.employee\_id, i.last\_name, i.first\_name, i.department\_id);

end loop;

end;

/

commit;

create or replace trigger tema6ex5\_ums

before delete or update of department\_id on dept\_test\_ums

for each row

begin

if deleting then

delete from emp\_test\_ums

where department\_id = :old.department\_id;

elsif updating then

update emp\_test\_ums

set department\_id = :new.department\_id

where department\_id = :old.department\_id;

end if;

end;

/

commit;

select \* from emp\_test\_ums;

select \* from dept\_test\_ums;

delete from dept\_test\_ums

where department\_id=70;

rollback;

update dept\_test\_ums

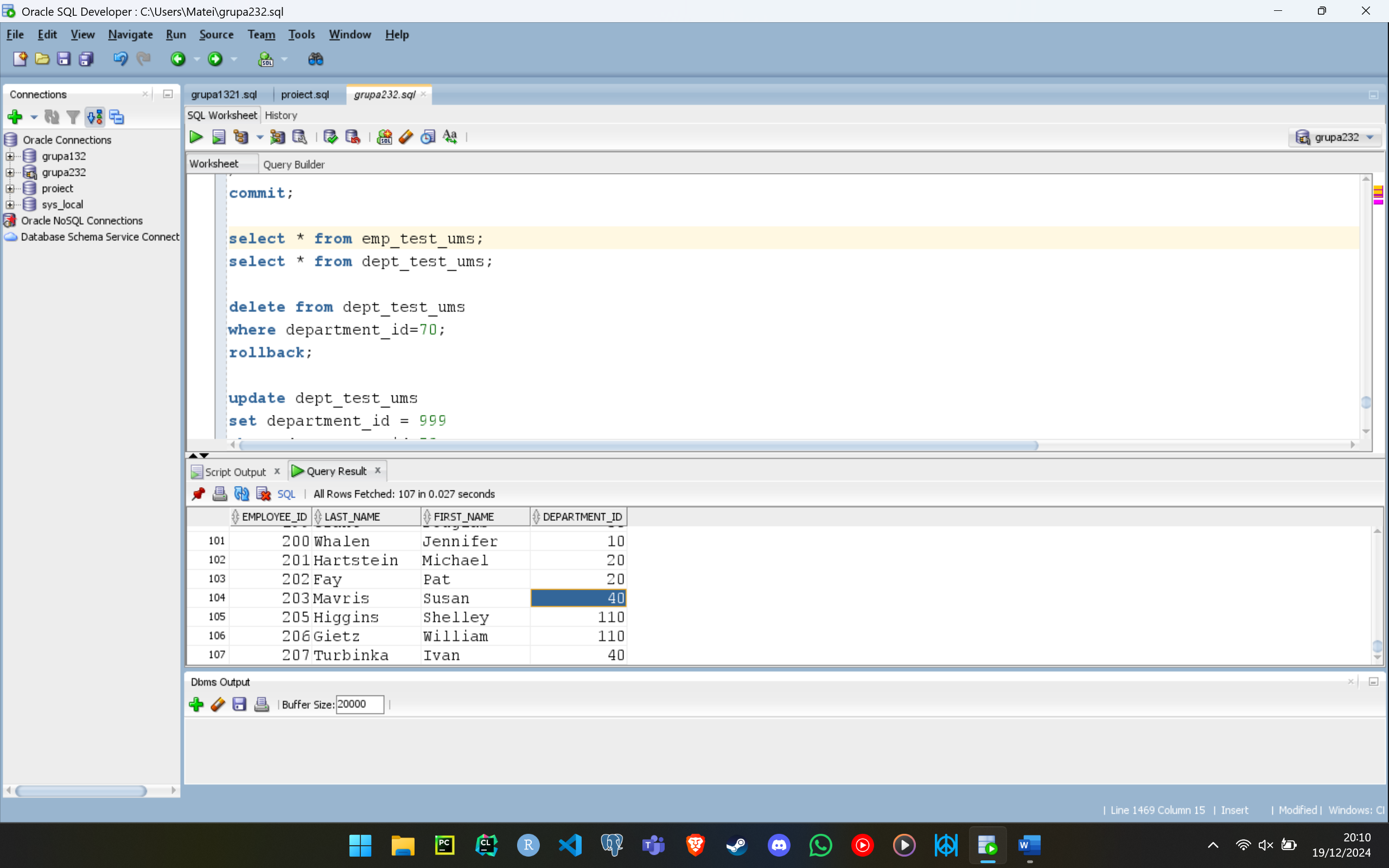
set department\_id = 999

where department\_id=70;

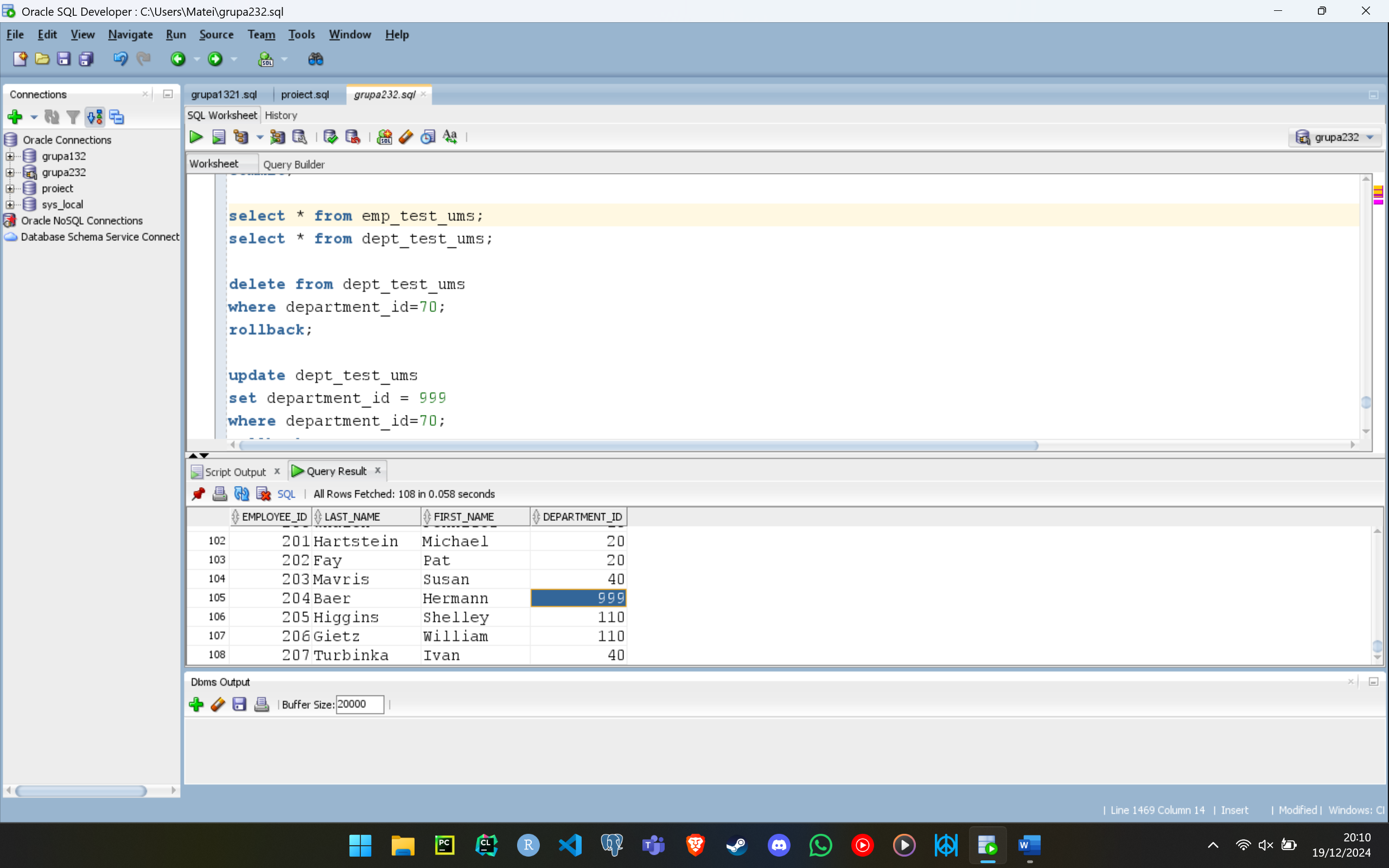
rollback;

In mare, nu am modificat nimic in cazul cand am FK fata de cazul cand nu am FK. Am modificat totusi la trigger momentul cand se executa: am pus before ca sa evit erori de departament inexistent, dar aparent merge si cu after, pentru ca verifica abia la sfarsitul tranzactiei constrangerile.

--dupa delete:



--dupa update:



--cu FK si ON DELETE CASCADE

create table dept\_test\_ums (

department\_id number(3) primary key,

department\_name varchar2(50)

);

create table emp\_test\_ums (

employee\_id number(3) primary key,

last\_name varchar2(50),

first\_name varchar2(50),

department\_id number(3) references dept\_test\_ums(department\_id) on delete cascade

);

begin

for i in (

select department\_id, department\_name

from departments

) loop

insert into dept\_test\_ums (department\_id, department\_name)

values (i.department\_id, i.department\_name);

end loop;

for i in (

select employee\_id, last\_name, first\_name, department\_id

from employees

) loop

insert into emp\_test\_ums (employee\_id, last\_name, first\_name, department\_id)

values (i.employee\_id, i.last\_name, i.first\_name, i.department\_id);

end loop;

end;

/

commit;

create or replace trigger tema6ex5\_ums

after delete or update of department\_id on dept\_test\_ums

for each row

begin

if updating then

update emp\_test\_ums

set department\_id = :new.department\_id

where department\_id = :old.department\_id;

end if;

end;

/

commit;

select \* from emp\_test\_ums;

select \* from dept\_test\_ums;

delete from dept\_test\_ums

where department\_id=70;

rollback;

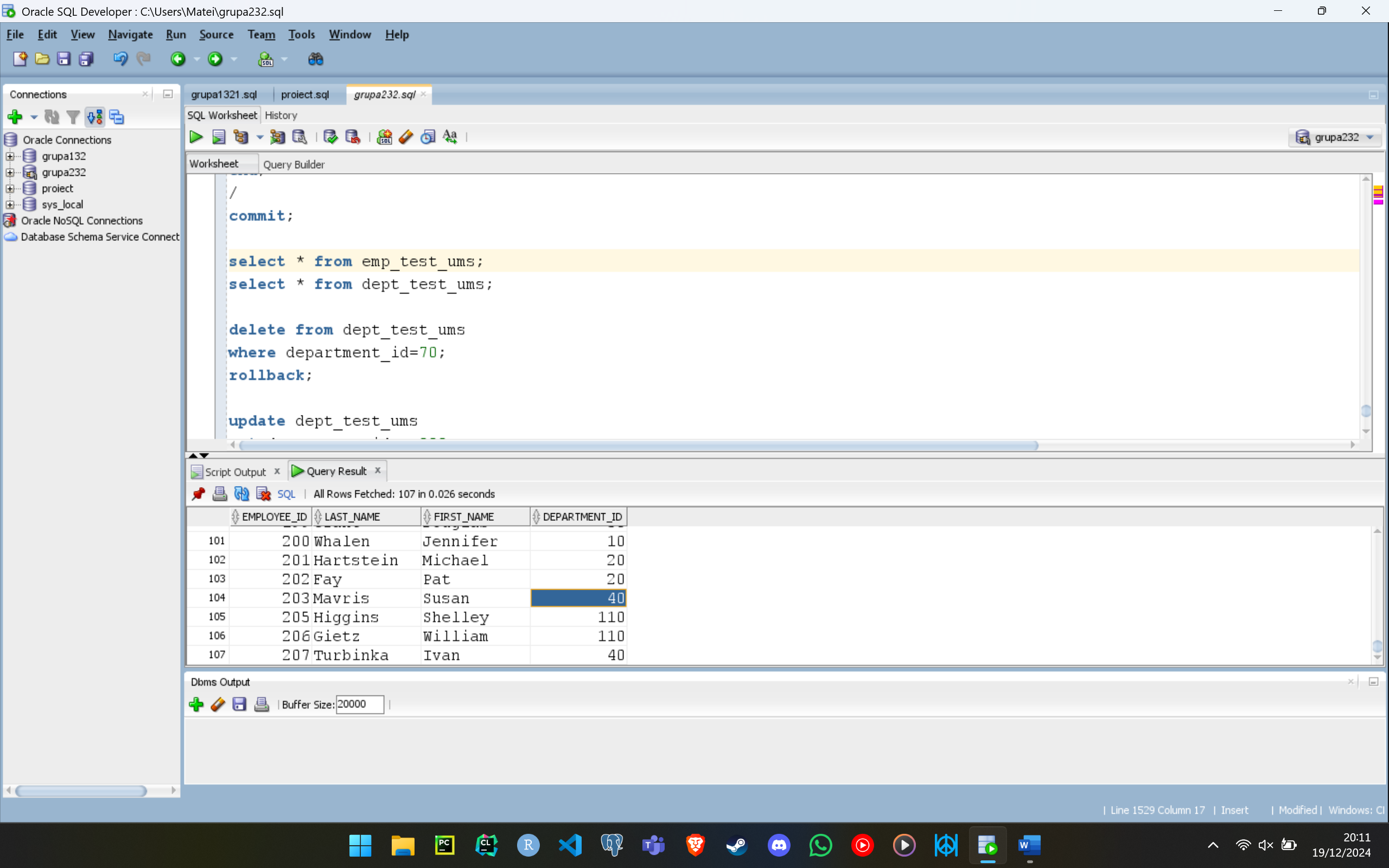
update dept\_test\_ums

set department\_id = 999

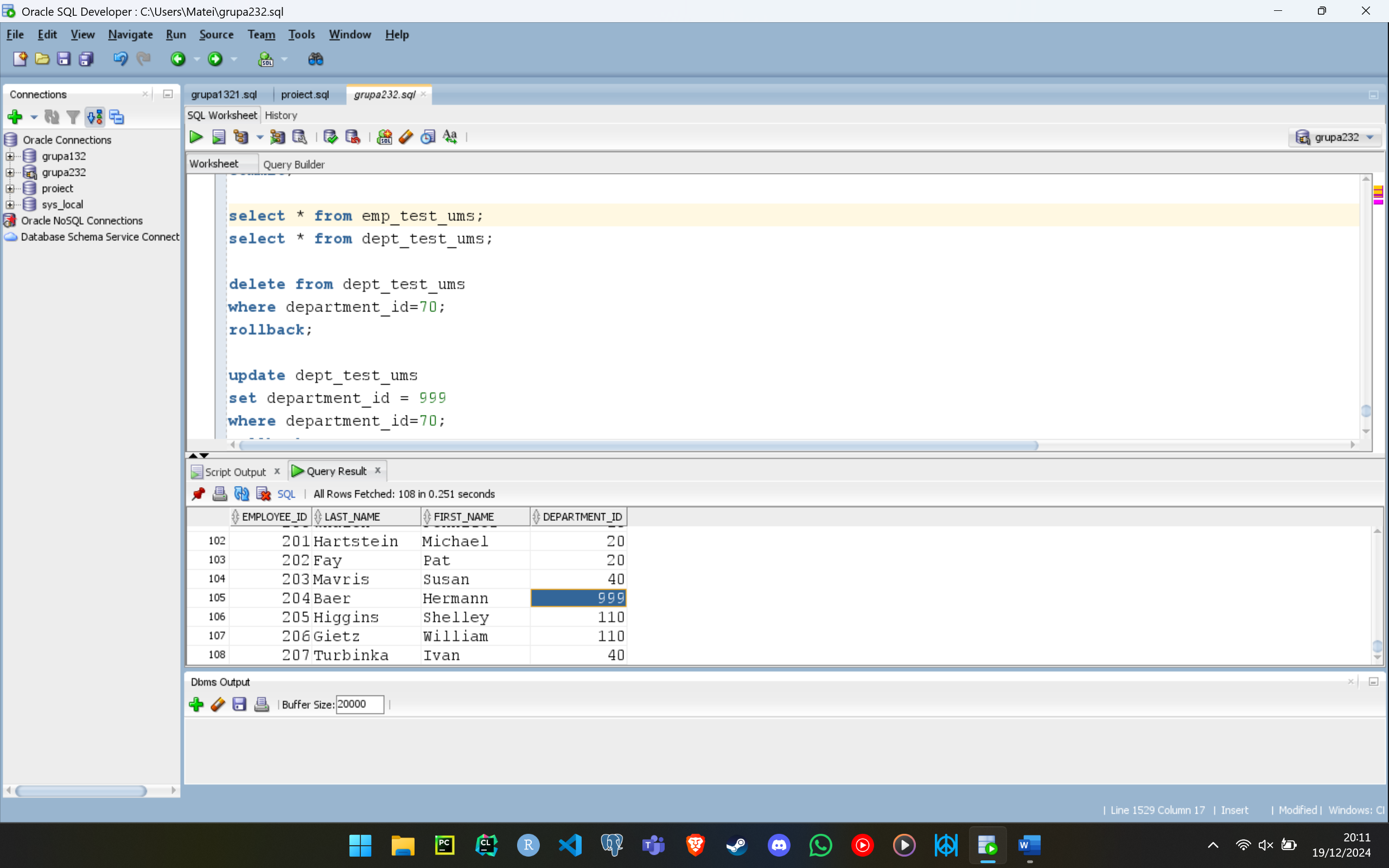
where department\_id=70;

rollback;

--dupa delete:



--dupa update:



--cu FK si ON DELETE SET NULL

create table dept\_test\_ums (

department\_id number(3) primary key,

department\_name varchar2(50)

);

create table emp\_test\_ums (

employee\_id number(3) primary key,

last\_name varchar2(50),

first\_name varchar2(50),

department\_id number(3) references dept\_test\_ums(department\_id) on delete set null

);

begin

for i in (

select department\_id, department\_name

from departments

) loop

insert into dept\_test\_ums (department\_id, department\_name)

values (i.department\_id, i.department\_name);

end loop;

for i in (

select employee\_id, last\_name, first\_name, department\_id

from employees

) loop

insert into emp\_test\_ums (employee\_id, last\_name, first\_name, department\_id)

values (i.employee\_id, i.last\_name, i.first\_name, i.department\_id);

end loop;

end;

/

commit;

create or replace trigger tema6ex5\_ums

after delete or update of department\_id on dept\_test\_ums

for each row

begin

if updating then

update emp\_test\_ums

set department\_id = :new.department\_id

where department\_id = :old.department\_id;

end if;

end;

/

commit;

select \* from emp\_test\_ums;

select \* from dept\_test\_ums;

delete from dept\_test\_ums

where department\_id=70;

rollback;

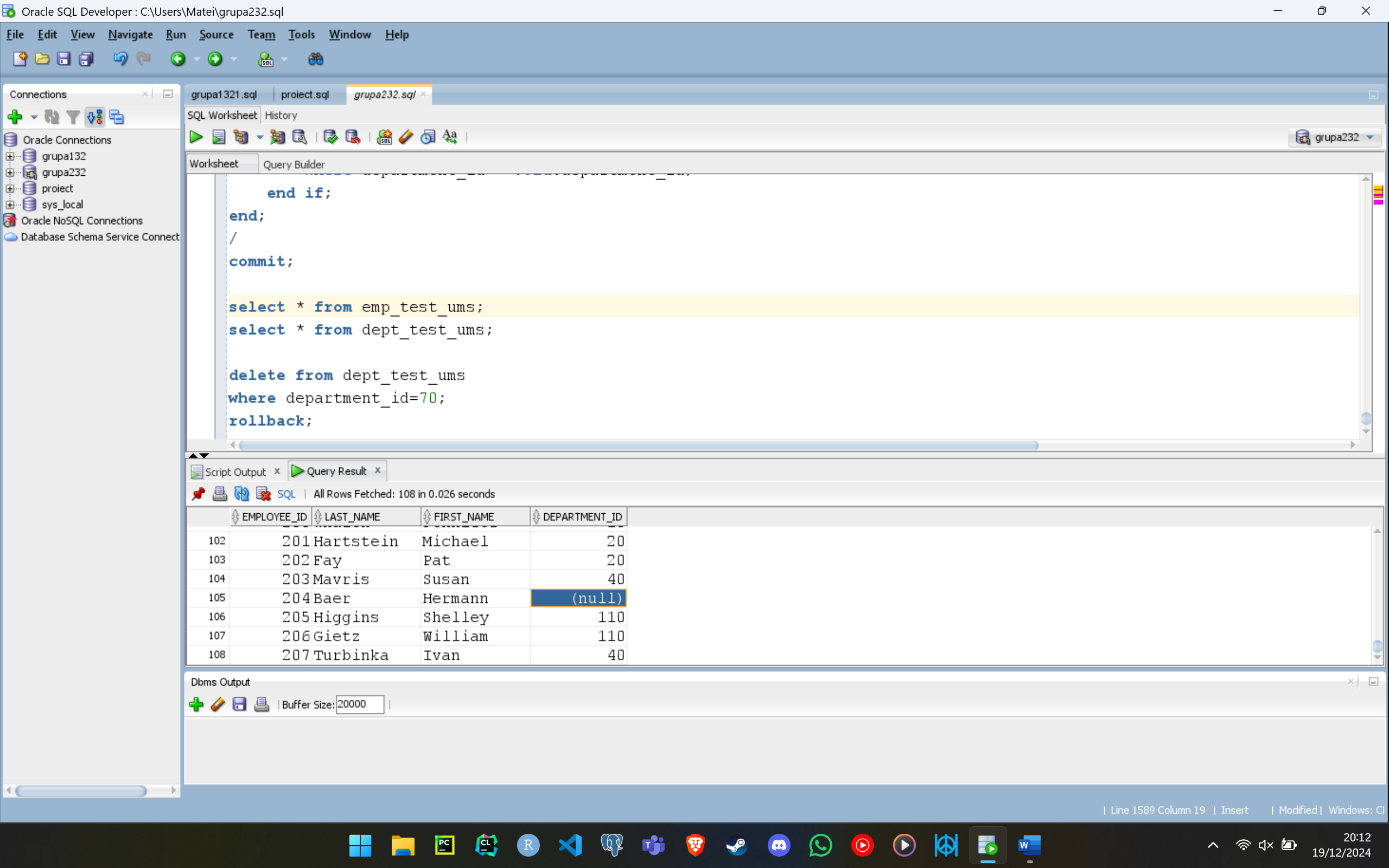
update dept\_test\_ums

set department\_id = 999

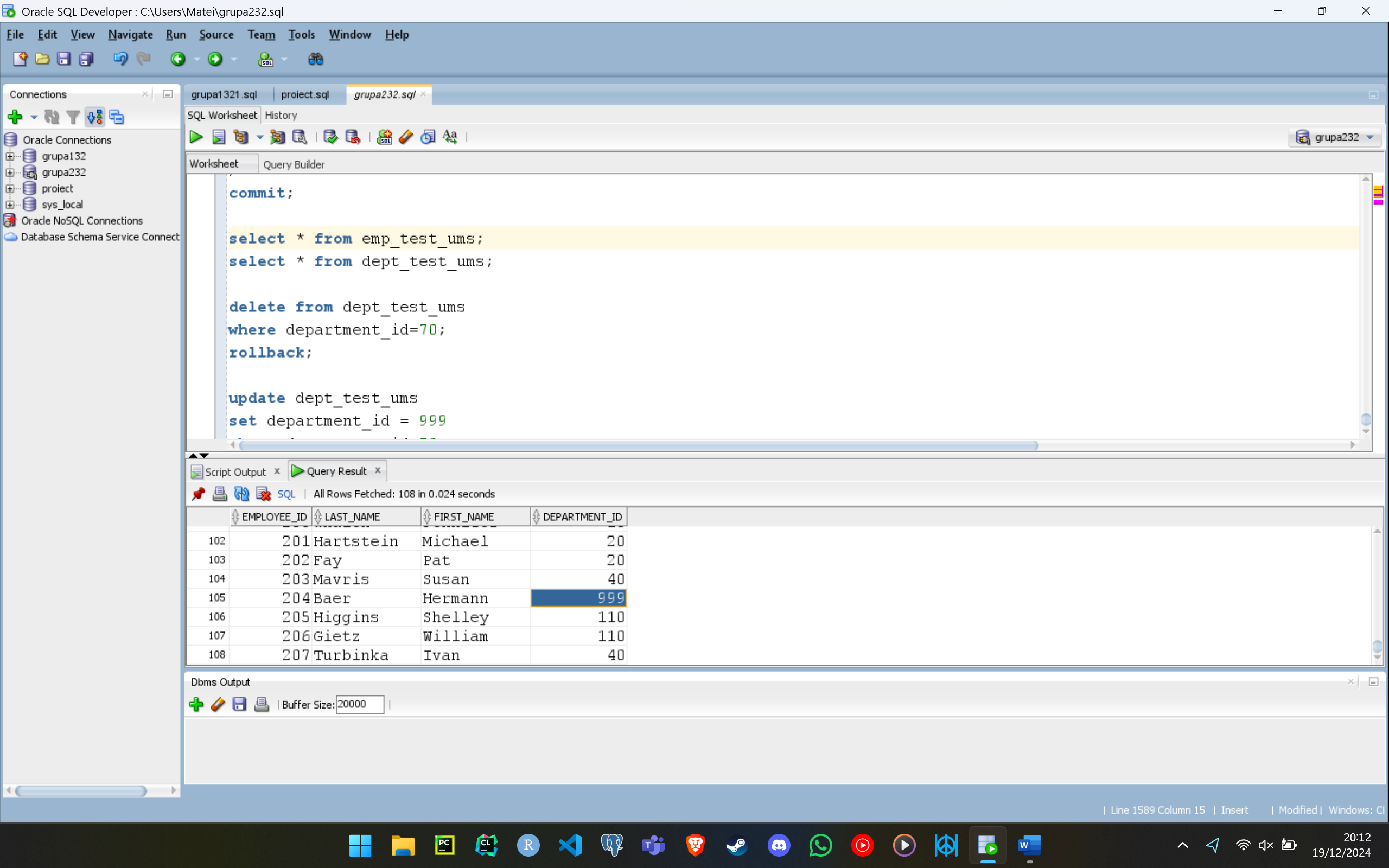
where department\_id=70;

rollback;

--dupa delete:



--dupa update:



E6. Creați un tabel cu următoarele coloane:

- user\_id (SYS.LOGIN\_USER);

- nume\_bd (SYS.DATABASE\_NAME);

- erori (DBMS\_UTILITY.FORMAT\_ERROR\_STACK);

- data.

Definiți un declanșator sistem (la nivel de bază de date) care să introducă date în acest tabel referitoare la erorile apărute.

create table erori (

user\_id varchar2(128),

nume\_bd varchar2(128),

erori clob,

data date

);

create or replace trigger erori\_trigger

after servererror on database

begin

insert into erori (user\_id, nume\_bd, erori, data)

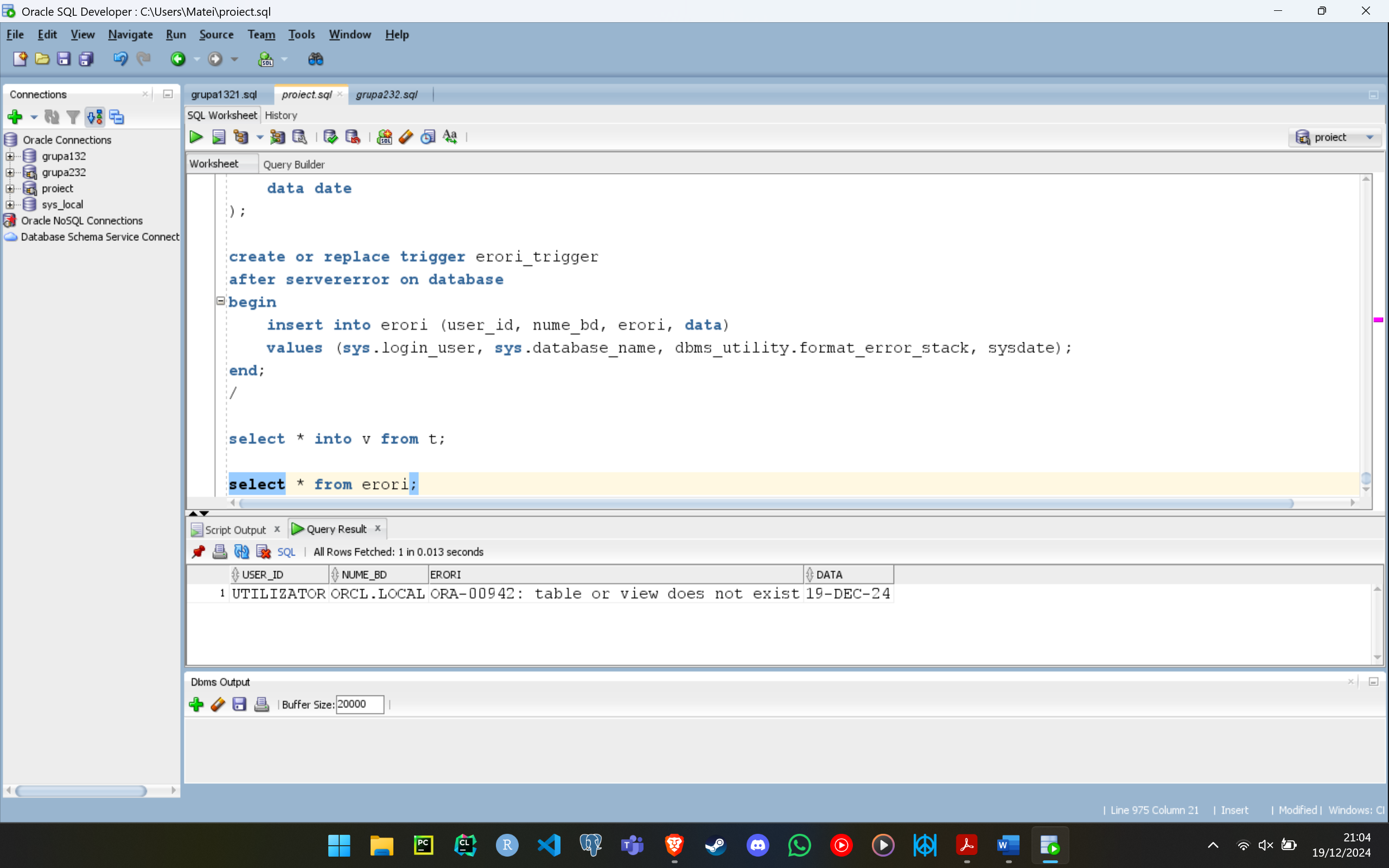
values (sys.login\_user, sys.database\_name, dbms\_utility.format\_error\_stack, sysdate);

end;

/

select \* into v from t;

select \* from erori;



E7. Adaptați cerința exercițiului 4 pentru diagrama proiectului prezentată la materia Baze de Date din anul I. Rezolvați acest exercițiu în PL/SQL, folosind baza de date proprie.

(ENUNT ORIGINAL) Creați tabelul info\_dept\_\*\*\* cu următoarele coloane:

- id (codul departamentului) – cheie primară;

- nume\_dept (numele departamentului);

- plati (suma alocată pentru plata salariilor angajaților care lucrează în departamentul respectiv).

Introduceți date în tabelul creat anterior corespunzătoare informațiilor existente în schemă.

Definiți un declanșator care va actualiza automat câmpul plati atunci când se introduce un nou salariat, respectiv se șterge un salariat sau se modifică salariul unui angajat.

(ENUNT ADAPTAT) Creați tabelul info\_cinema cu următoarele coloane:

- id (codul cinemaului) – cheie primară;

- nume\_cinema (numele cinemaului);

- plati (suma alocată pentru plata salariilor angajaților care lucrează în cinemaul respectiv).

Introduceți date în tabelul creat anterior corespunzătoare informațiilor existente în schemă.

Definiți un declanșator care va actualiza automat câmpul plati atunci când se introduce un nou salariat, respectiv se șterge un salariat sau se modifică salariul unui angajat.

create table info\_cinema (

id number(3) primary key,

nume\_cinema varchar2(50),

plati number(6)

);

declare

v\_total number;

v\_nume cinemauri.nume\_cinema%type;

begin

for i in (

select id\_cinema, nume\_cinema

from cinemauri

) loop

v\_total := 0;

for j in (

select id\_angajat, salariu

from angajati

where id\_cinema = i.id\_cinema

) loop

v\_total := v\_total + j.salariu;

end loop;

insert into info\_cinema (id, nume\_cinema, plati)

values (i.id\_cinema, i.nume\_cinema, v\_total);

end loop;

end;

/

create or replace procedure modific\_plati

(v\_codd info\_cinema.id%type, v\_plati info\_cinema.plati%type)

as

begin

update info\_cinema

set plati = NVL(plati, 0) + v\_plati

where id = v\_codd;

end;

/

create or replace trigger tema6ex7

after delete or update or insert of salariu on angajati

for each row

begin

if DELETING then

modific\_plati(:old.id\_cinema, -1\*:old.salariu);

elsif UPDATING then

modific\_plati(:old.id\_cinema,:new.salariu-:old.salariu);

else

modific\_plati(:new.id\_cinema, :new.salariu);

end if;

end;

/

begin

update angajati

set salariu = salariu - 1000

where id\_angajat = 103;

end;

/

select \* from info\_cinema;

