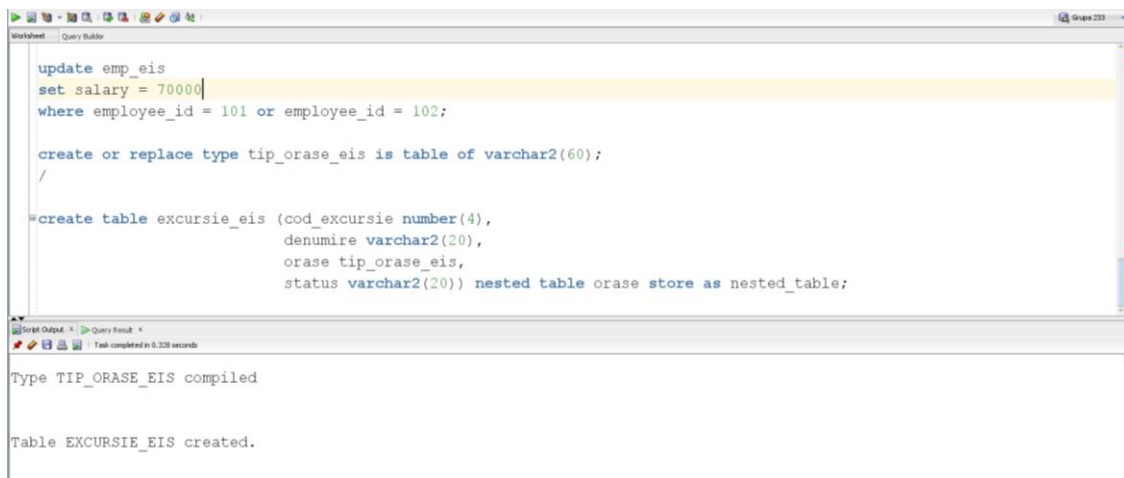


1. Dati exemplu de un trigger LMD care sa lucreze pe un tablou imbricat.

Pentru acest exercitiu m-am folosit de o tema de laborator mai veche si am creat un tabel cu excursii ce contine o coloana de orase sub forma de tablou imbricat.

```
create or replace type tip_orase_eis is table of varchar2(60);
```

```
create table excursie_eis (cod_excursie number(4),  
                           denumire varchar2(20),  
                           orase tip_orase_eis,  
                           status varchar2(20)) nested table orase store as nested_table;
```



Am facut un trigger care sa nu permita introducerea oraselor daca acestea nu sunt scrise cu majuscula.

```
create or replace trigger trigger_excursii
```

```
before insert or update on excursie_eis
```

```
for each row
```

```
begin
```

```
if :new.orase is not null then
```

```
for i in :new.orase.first .. :new.orase.last loop
```

```
if initcap(:new.orase(i)) != :new.orase(i) then
```

```

        raise_application_error(-20003, 'Orasele trebuie scrise cu majuscula!');
    end if;
end loop;
end if;
end;
/

```

```

create or replace trigger trigger_excursii
before insert or update on excursie_eis
for each row
begin
    if :new.orase is not null then
        for i in :new.orase.first .. :new.orase.last loop
            if initcap(:new.orase(i)) != :new.orase(i) then
                raise_application_error(-20003, 'Orasele trebuie scrise cu majuscula!');
            end if;
        end loop;
    end if;
end;

```

Trigger TRIGGER_EXCURSII compiled

Am inserat doua valori ce respecta cerintele:

```
insert into excursie_eis
```

```
values (10, 'Prima mea excursie', tip_orase_eis('Brasov', 'Bucuresti'), 'disponibila');
```

```
insert into excursie_eis
```

```
values (20, 'Hai la mare', tip_orase_eis('Constanta', 'Mangalia', 'Fetesti'), 'disponibila');
```

Pentru aceste doua inserari totul a mers ok:

```

/

insert into excursie_eis
values (10, 'Prima mea excursie', tip_orase_eis('Brasov', 'Bucuresti'), 'disponibila');

insert into excursie_eis
values (20, 'Hai la mare', tip_orase_eis('Constanta', 'Mangalia', 'Fetesti'), 'disponibila');

select * from excursie_eis;

```

	COD_EXCURSIE	DENUMIRE	ORASE	STATUS
1	10	Prima mea excursie	GRUPA233.TIP ORASE EIS('Brasov', 'Bucuresti')	disponibila
2	20	Hai la mare	GRUPA233.TIP ORASE EIS('Constanta', 'Mangalia', 'Fetesti')	disponibila

Dupa, am inserat o valoare ce nu respecta cerintele:

insert into excursie_eis

values (30, 'Hai la munte', tip_orase_eis('Brasov', 'sibiu', 'Busteni', 'Sinaia'), 'disponibila');

Pentru aceasta a fost declansat triggerul:



```
insert into excursie_eis
values (10, 'Prima mea excursie', tip_orase_eis('Brasov', 'Bucuresti'), 'disponibila');

insert into excursie_eis
values (20, 'Hai la mare', tip_orase_eis('Constanta', 'Mangalia', 'Fetesti'), 'disponibila');

select * from excursie_eis;

insert into excursie_eis
values (30, 'Hai la munte', tip_orase_eis('Brasov', 'sibiu', 'Busteni', 'Sinaia'), 'disponibila');
```

Error starting at line : 88 in command -
insert into excursie_eis
values (30, 'Hai la munte', tip_orase_eis('Brasov', 'sibiu', 'Busteni', 'Sinaia'), 'disponibila')
Error report -
ORA-20003: Orasele trebuie scrise cu majuscula!
ORA-06512: at "GRUPA233.TRIGGER_EXCURSII", line 5
ORA-04088: error during execution of trigger 'GRUPA233.TRIGGER_EXCURSII'

2. Ce este un compound trigger? Dati un exemplu care sa ilustreze ordinea in care sunt executate instructiunile.

Un compound trigger¹ este un trigger care poate actiona in mai multe momente ale unei comenzi de delete, insert sau update. Practic, este un trigger care combina alte patru tipuri de trigger, avand urmatoarele sectiuni:

- Before section
- Before each row section
- After each row section
- After section

Avantajul lui este ca imita comportamentul unei proceduri si actioneaza automat. Cu ajutorul compound trigger se poate evita si eroarea mutating-table.

Sintaxa generala:

```
create or replace trigger <trigger name>
for <insert|update|delete> <of column_name> on <tablename>
COMPOUND TRIGGER
<declare section>

BEFORE
<before section>
```

¹ <https://www.databasejournal.com/oracle/compound-triggers-in-oracle-11g/>

BEFORE EACH ROW

<before each row section>

AFTER EACH ROW

<after each row section>

AFTER

<after section>

END;

Cateva dintre restrictiile² pentru compound trigger:

- Corpul unui compound trigger trebuie sa fie un bloc compound trigger
- Un compound trigger trebuie sa fie un trigger LMD
- Un compound trigger trebuie definit pe un tabel sau pe o vizualizare
- Partea declarativa a sa nu poate include pragma autonomous_transaction
- O exceptie aparuta intr-o sectiune trebuie gestionata in aceeasi sectiune
- Daca o sectiune include GOTO, acel GOTO trebuie sa ne duca in aceeasi sectiune
- Nu putem avea OLD, NEW, PARENT in partea declarativa, in BEFORE sau in AFTER

Un exemplu:

Am creat o copie a tabelului employees si urmatorul trigger. Scopul sau este de a pune in evidenta ordinea in care se executa instructiunile.

```
drop table emp_eis;
```

```
create table emp_eis as select * from employees;
```

```
create or replace trigger emp_eis_tema_curs
```

```
for update of salary on emp_eis
```

```
compound trigger
```

```
suma number := 0;
```

```
before statement is
```

```
begin
```

² <https://www.viralpatel.net/compound-triggers-in-oracle-11g-tutorial-example/>

```

-- Afiseaza suma inainte de update pe salariu.

dbms_output.put_line('1. Suntem in before statement');

select sum(salary) into suma from emp_eis;

dbms_output.put_line('1. Suma: ' || suma);

end before statement;


before each row is

begin

-- Daca noul salariu va fi 7000, nu il modifica.

dbms_output.put_line('2. Suntem in before each row statement');

if (:new.salary = 70000)

    then :new.salary := :old.salary;

end if;

end before each row;


after each row is

begin

dbms_output.put_line('3. Suntem in after each row statement');

end after each row;


after statement is

begin

dbms_output.put_line('4. Suntem in after statement');

select sum(salary) into suma from emp_eis;

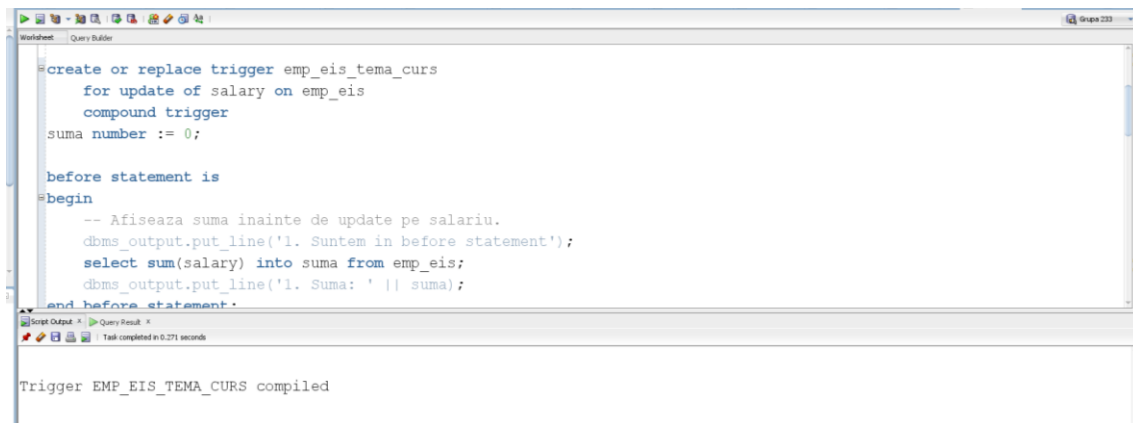
dbms_output.put_line('4. Suma: ' || suma);

end after statement;


end;

/

```



The screenshot shows the SQL Developer interface with a query window titled 'Query Builder'. The SQL code defines a trigger named 'emp_eis_tema_curs' that fires on an update to the 'salary' column of the 'emp_eis' table. The trigger is a 'compound trigger' and includes a 'before statement' section. Inside this section, it declares a variable 'suma' of type 'number' and initializes it to 0. The 'begin' block contains a comment in Romanian, followed by a call to 'dbms_output.put_line' to print '1. Suntem in before statement'. Then, it performs a 'select sum(salary) into suma from emp_eis;' and another call to 'dbms_output.put_line' to print the sum. The trigger ends with 'end before statement;'. Below the query window, the 'Script Output' pane shows the message 'Trigger EMP_EIS_TEMA_CURS compiled'.

```
create or replace trigger emp_eis_tema_curs
for update of salary on emp_eis
compound trigger
suma number := 0;

before statement is
begin
-- Afiseaza suma inainte de update pe salariu.
dbms_output.put_line('1. Suntem in before statement');
select sum(salary) into suma from emp_eis;
dbms_output.put_line('1. Suma: ' || suma);
end before statement;
```

Trigger EMP_EIS_TEMA_CURS compiled

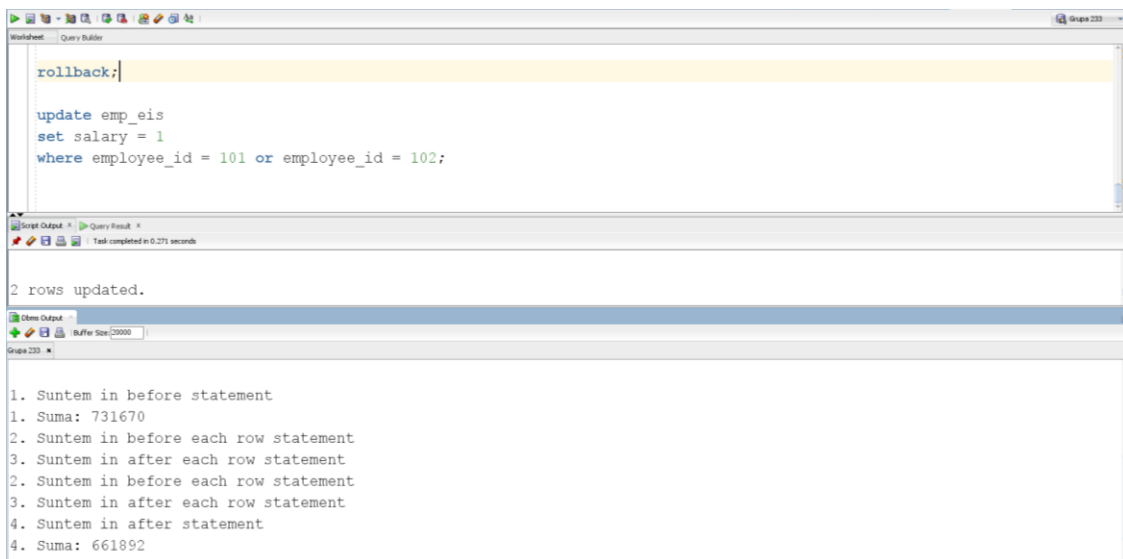
Am rulat urmatorul update:

```
update emp_eis
```

```
set salary = 1
```

```
where employee_id = 101 or employee_id = 102;
```

Se observa clar ordinea executiei si ca salariile au fost modificate.



The screenshot shows the SQL Developer interface with a query window titled 'Query Builder'. The SQL code consists of a 'rollback;' statement followed by an 'update emp_eis' statement that sets 'salary = 1' for 'employee_id = 101 or employee_id = 102'. Below the query window, the 'Script Output' pane shows the message '2 rows updated.'. The 'DBMS Output' pane is expanded, showing a list of messages from the trigger's execution: '1. Suntem in before statement', '1. Suma: 731670', '2. Suntem in before each row statement', '3. Suntem in after each row statement', '2. Suntem in before each row statement', '3. Suntem in after each row statement', '4. Suntem in after statement', and '4. Suma: 661892'.

```
rollback;
```

```
update emp_eis
set salary = 1
where employee_id = 101 or employee_id = 102;
```

2 rows updated.

1. Suntem in before statement
1. Suma: 731670
2. Suntem in before each row statement
3. Suntem in after each row statement
2. Suntem in before each row statement
3. Suntem in after each row statement
4. Suntem in after statement
4. Suma: 661892

Acum rulez un nou update:

```
update emp_eis
```

```
set salary = 70000
```

```
where employee_id = 101 or employee_id = 102;
```

Se observa clar ca salariile nu au fost modificate, lucru stabilit in sectiunea before each row.

Worksheet Query Builder

```
rollback;

update emp_eis
set salary = 70000
where employee_id = 101 or employee_id = 102;
```

Script Output: X Query Result: X
Task completed in 0.123 seconds

2 rows updated.

Items Output: X
Buffer Size: 20000

Grupa 233 X

1. Suntem in before statement
1. Suma: 661892
2. Suntem in before each row statement
3. Suntem in after each row statement
2. Suntem in before each row statement
3. Suntem in after each row statement
4. Suntem in after statement
4. Suma: 661892