**Temă**

*Cerința*: Să se ilustreze printr-un exemplu pe schema aleasă pentru proiect apariția erorii mutating table și să se trateze această eroare. Se va implementa trigger-ul care generează eroarea cu comenzile de declanșare și soluția. Studiati exemplul 8 din laborator6\_PLSQL.pdf si exemplul 8.17 din fisierul Materiale/cursuri/SGBD8 - PLSQL Triggeri - mutating - trigger-compus.sql.

*Rezolvare*: Implementați cu ajutorul unui trigger următoarea restricție: o secție nu poate avea mai mult de 3 camere pe un etaj.

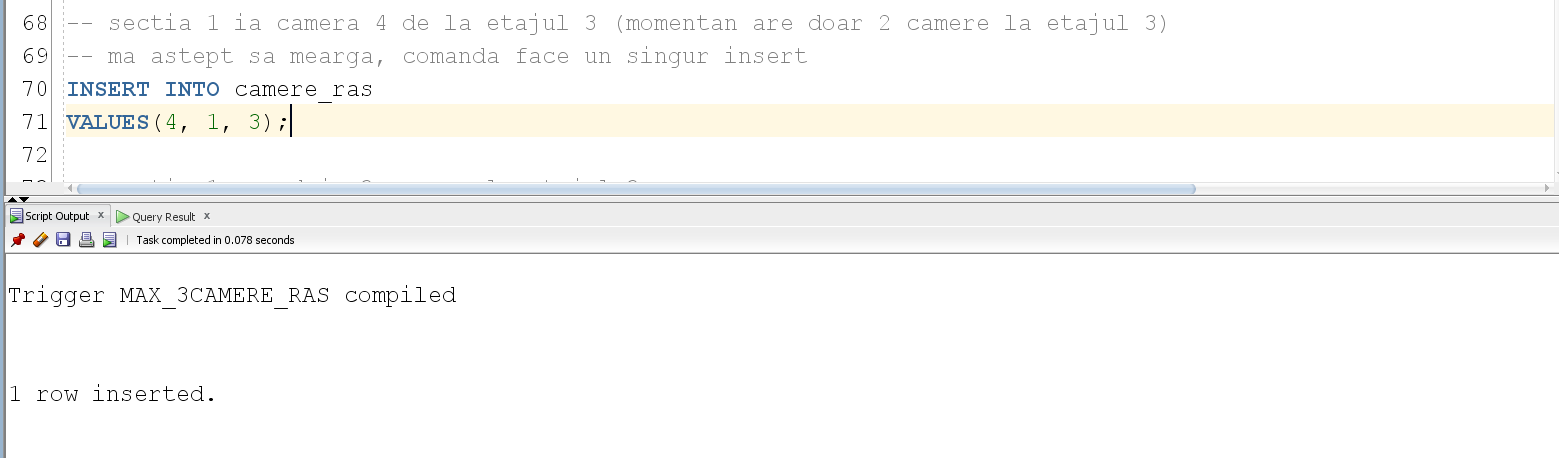
Pentru rezolvarea acestei cerințe se va folosi schema CLINICĂ MEDICALĂ folosită în proiectul personal. Aceasta este deja implementată pe *grupa\_251*, dar atașez mai jos crearea și popularea tabelelor cu care se va lucra:

| DROP TABLE etaj\_ras CASCADE CONSTRAINTS;  DROP TABLE sectie\_ras CASCADE CONSTRAINTS;  DROP TABLE camere\_ras CASCADE CONSTRAINTS;  CREATE TABLE sectie\_ras  (id NUMBER PRIMARY KEY,  denumire VARCHAR2(30) NOT NULL);    CREATE TABLE etaj\_ras  (id NUMBER PRIMARY KEY,  nume VARCHAR2(30) NOT NULL);  CREATE TABLE camere\_ras -- entitate dependenta de etaj  (id NUMBER,  id\_sectie CONSTRAINT fk\_camere\_sectie REFERENCES sectie\_ras(id) ON DELETE SET NULL,  id\_etaj CONSTRAINT fk\_camere\_etaj REFERENCES etaj\_ras(id) ON DELETE CASCADE,  CONSTRAINT pk\_camere PRIMARY KEY (id, id\_etaj));  INSERT INTO etaj\_ras  VALUES (1, 'Subsol');  INSERT INTO etaj\_ras  VALUES (2, 'Parter');  INSERT INTO etaj\_ras  VALUES (3, 'Etajul 1');  INSERT INTO etaj\_ras  VALUES (4, 'Etajul 2');  INSERT INTO etaj\_ras  VALUES (5, 'Etajul 3');  INSERT INTO etaj\_ras  VALUES (6, 'Mansarda');  --------------------------------------------  INSERT INTO sectie\_ras  VALUES (1, 'Cardiologie');  INSERT INTO sectie\_ras  VALUES (2, 'Dermatologie');  INSERT INTO sectie\_ras  VALUES (3, 'Urologie');  INSERT INTO sectie\_ras  VALUES (4, 'Medicina interna');  INSERT INTO sectie\_ras  VALUES (5, 'Pediatrie');  INSERT INTO sectie\_ras  VALUES (6, 'Endocrinologie');  -----------------------------------------------------  INSERT INTO camere\_ras  VALUES (1, 1, 1);  INSERT INTO camere\_ras  VALUES (2, 6, 1);  INSERT INTO camere\_ras  VALUES (3, 6, 1);  INSERT INTO camere\_ras  VALUES (1, 1, 2);  INSERT INTO camere\_ras  VALUES (2, 3, 2);  INSERT INTO camere\_ras  VALUES (3, 4, 2);  INSERT INTO camere\_ras  VALUES (1, 1, 3);  INSERT INTO camere\_ras  VALUES (2, 1, 3);  INSERT INTO camere\_ras  VALUES (3, 2, 3);  INSERT INTO camere\_ras  VALUES (1, 2, 4);  INSERT INTO camere\_ras  VALUES (2, 5, 4);  INSERT INTO camere\_ras  VALUES (3, 5, 4);  INSERT INTO camere\_ras  VALUES (1, 3, 5);  INSERT INTO camere\_ras  VALUES (2, 3, 5);  INSERT INTO camere\_ras  VALUES (3, 4, 5);  INSERT INTO camere\_ras  VALUES (1, 3, 6);  INSERT INTO camere\_ras  VALUES (2, 1, 6);  INSERT INTO camere\_ras  VALUES (3, 2, 6);  ----------------------------------------  COMMIT; |
| --- |

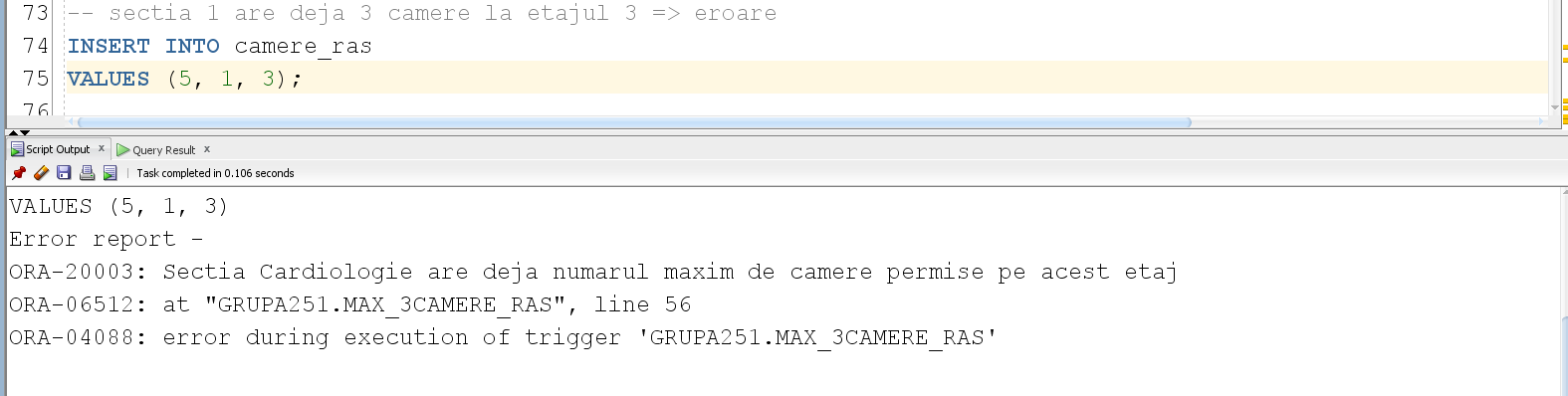
Inițial, încercăm implementarea unui trigger care să transpună restricția:

| CREATE OR REPLACE TRIGGER max\_3camere\_ras  BEFORE INSERT OR UPDATE OF id\_etaj, id\_sectie ON camere\_ras  FOR EACH ROW  DECLARE  nr\_camere NUMBER;  nume\_sectie sectie\_ras.denumire%TYPE;  ct\_sectie NUMBER;  sectie\_inexistenta EXCEPTION;  nr\_maxim EXCEPTION;  etaj\_inexistent EXCEPTION;  ct\_etaj NUMBER;  mesaj\_trigger VARCHAR2(100);  BEGIN  -- vedem daca exista etajul dat  SELECT COUNT(\*)  INTO ct\_etaj  FROM etaj\_ras  WHERE id = :NEW.id\_etaj;    IF ct\_etaj = 0 THEN  RAISE etaj\_inexistent;  END IF;  SELECT COUNT(\*)  INTO nr\_camere  FROM camere\_ras  WHERE id\_etaj = :NEW.id\_etaj AND id\_sectie = :NEW.id\_sectie;  IF :NEW.id\_sectie IS NOT NULL THEN  -- sectie\_ras nu e mutating, pot sa il interoghez  SELECT COUNT(\*)  INTO ct\_sectie  FROM sectie\_ras  WHERE id = :NEW.id\_sectie;    IF ct\_sectie = 0 THEN  RAISE sectie\_inexistenta;  END IF;  -- selectam numele sectiei  SELECT denumire  INTO nume\_sectie  FROM sectie\_ras  WHERE id = :NEW.id\_sectie;    IF nr\_camere = 3 THEN  mesaj\_trigger := 'Sectia ' || nume\_sectie || ' are deja numarul maxim de camere permise pe acest etaj';  RAISE nr\_maxim;  END IF;    END IF;  EXCEPTION  WHEN sectie\_inexistenta THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Sectia data nu exista');  WHEN etaj\_inexistent THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Etajul dat nu exista');  WHEN nr\_maxim THEN  RAISE\_APPLICATION\_ERROR(-20003, mesaj\_trigger);  END;  /  -- sectia 1 ia camera 4 de la etajul 3 (momentan are doar 2 camere la etajul 3)  -- ma astept sa mearga, comanda face un singur insert  INSERT INTO camere\_ras  VALUES(4, 1, 3);  -- sectia 1 are deja 3 camere la etajul 3 => eroare  INSERT INTO camere\_ras  VALUES (5, 1, 3);  ROLLBACK;  -- nu exista etajul cu id-ul dat => eroare  INSERT INTO camere\_ras  VALUES (1, 2, 7);  -- nu exista sectia data => eroare  INSERT INTO camere\_ras  VALUES (5, 8, 1);  -- insert into ... select ... => mutating table  INSERT INTO camere\_ras  SELECT 4, 1, 3  FROM DUAL;  -- mutating table  -- sectia 6 are 2 camere la etajul 1  -- sectia 5 nu are nicio camera la etajul 1  -- => operatia ar trebui sa poata fi executata, dar obtinem mutating  UPDATE camere\_ras  SET id\_sectie = 5  WHERE id\_sectie = 6;  ROLLBACK;  ALTER TRIGGER max\_3camere\_ras DISABLE; |
| --- |

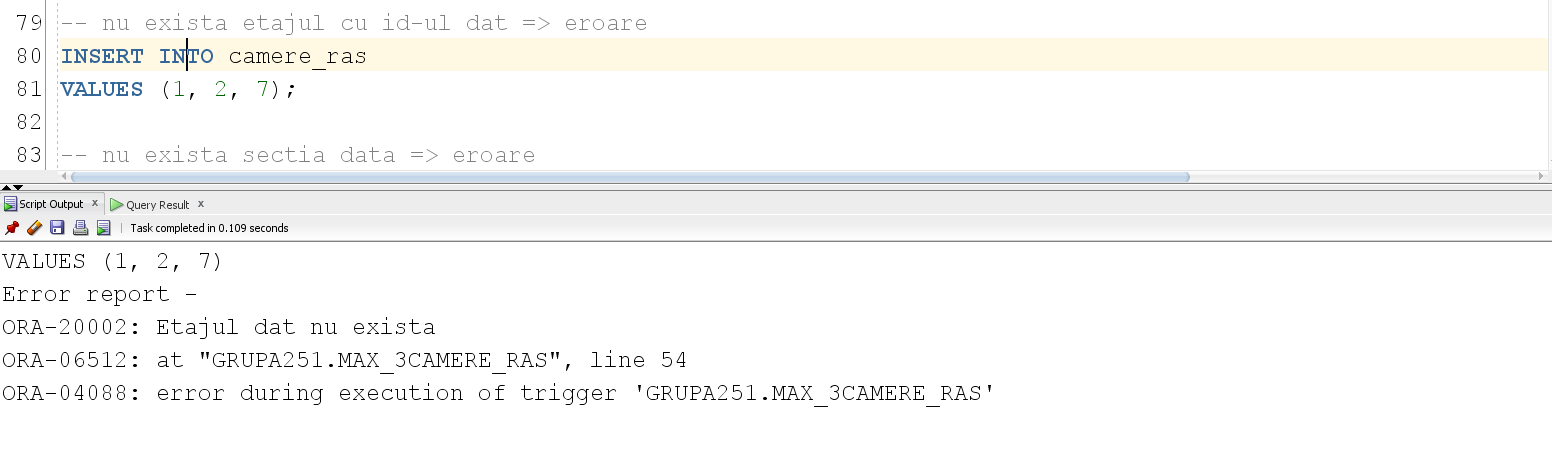
*Exemplul 1*



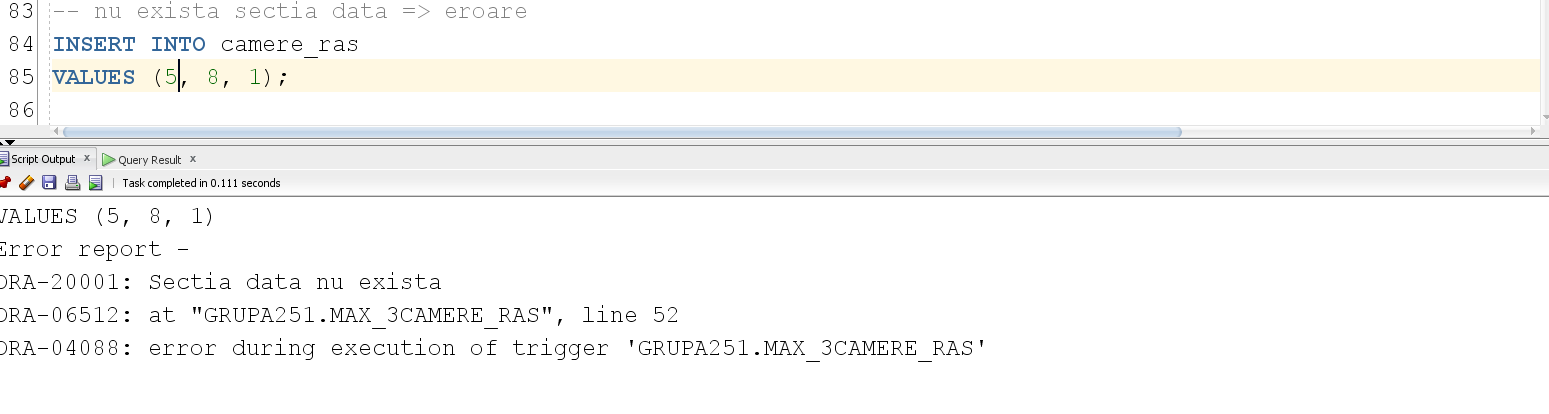
*Exemplul 2*

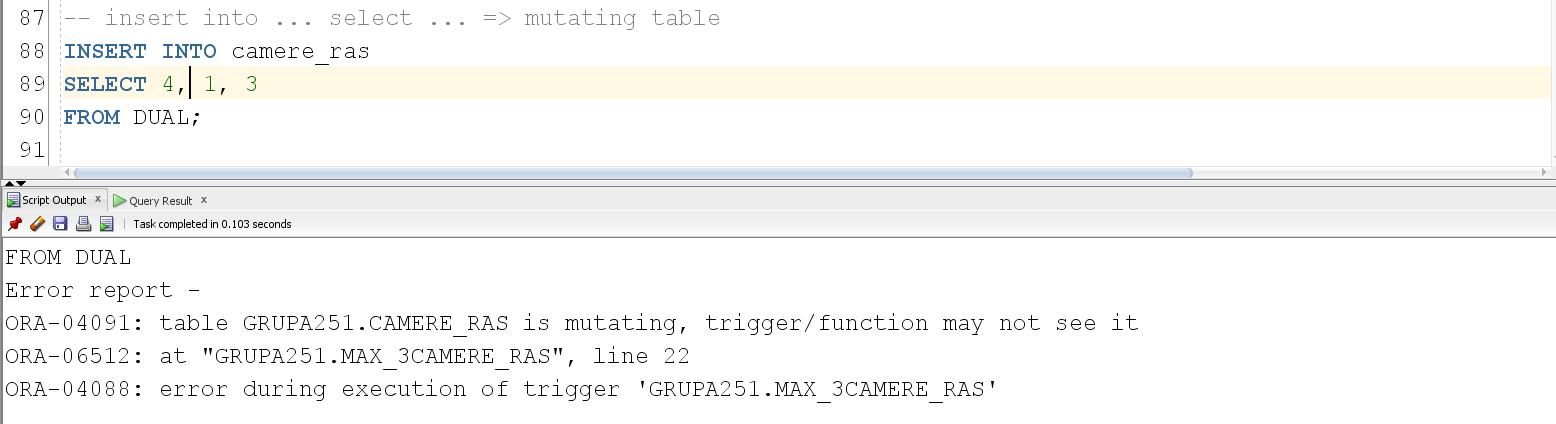


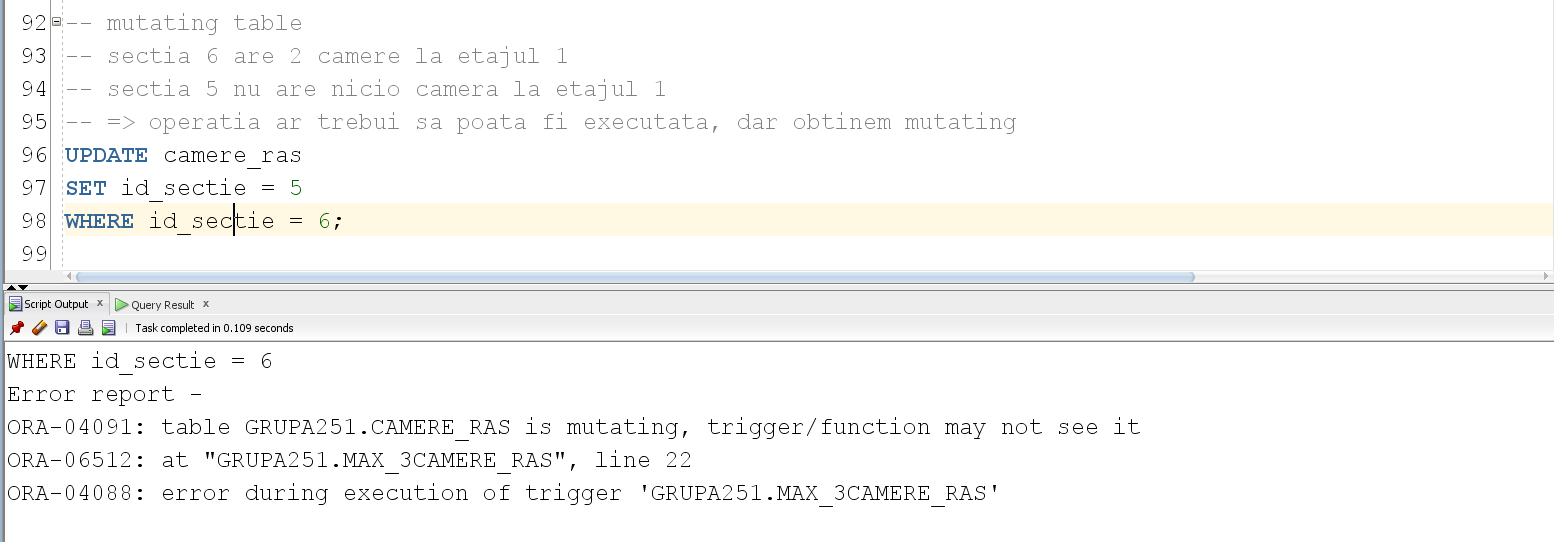
*Exemplul 3*



*Exemplul 4*

*Exemplul 5*

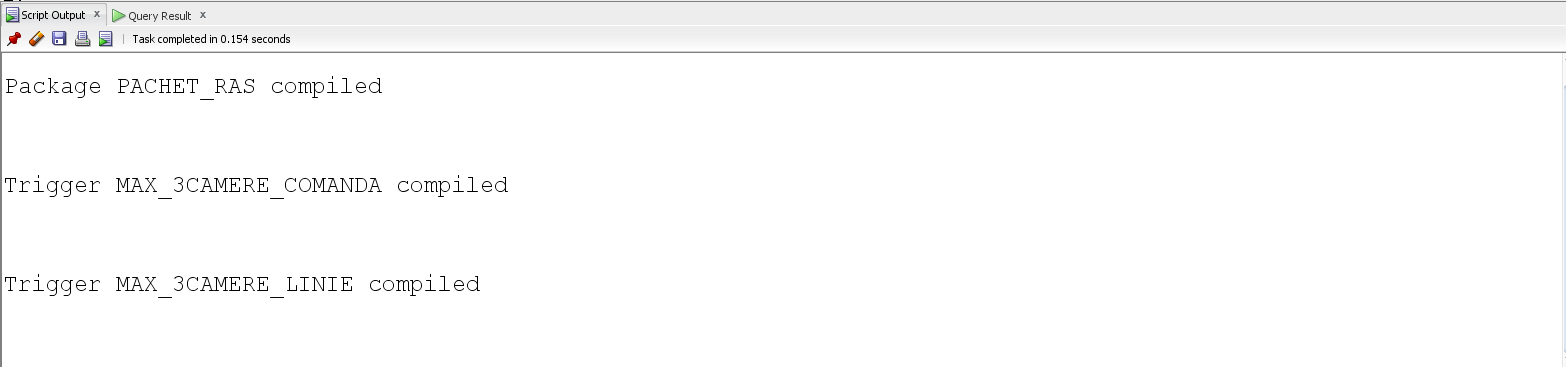


*Exemplul 6*

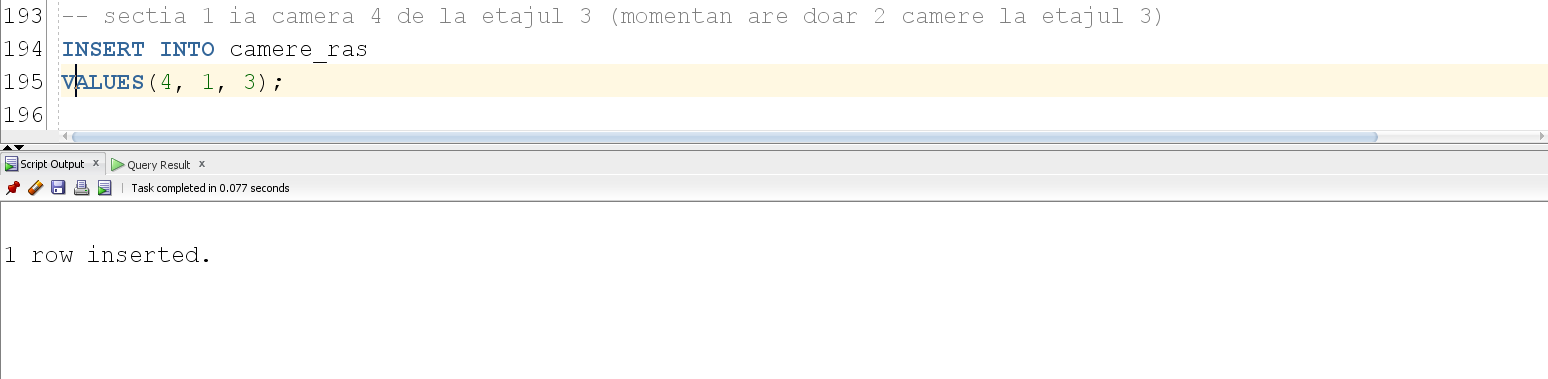
Observăm că se obține eroarea *mutating table*, deoarece în timp ce modificăm tabelul camere\_ras, facem interogări asupra lui pentru a vedea câte camere am asociat pentru a anumită secție pe etajul dat. Pentru a evita această eroare, avem 2 posibilități: facem un ***pachet cu 2 trigger***i sau facem un ***trigger compus***.

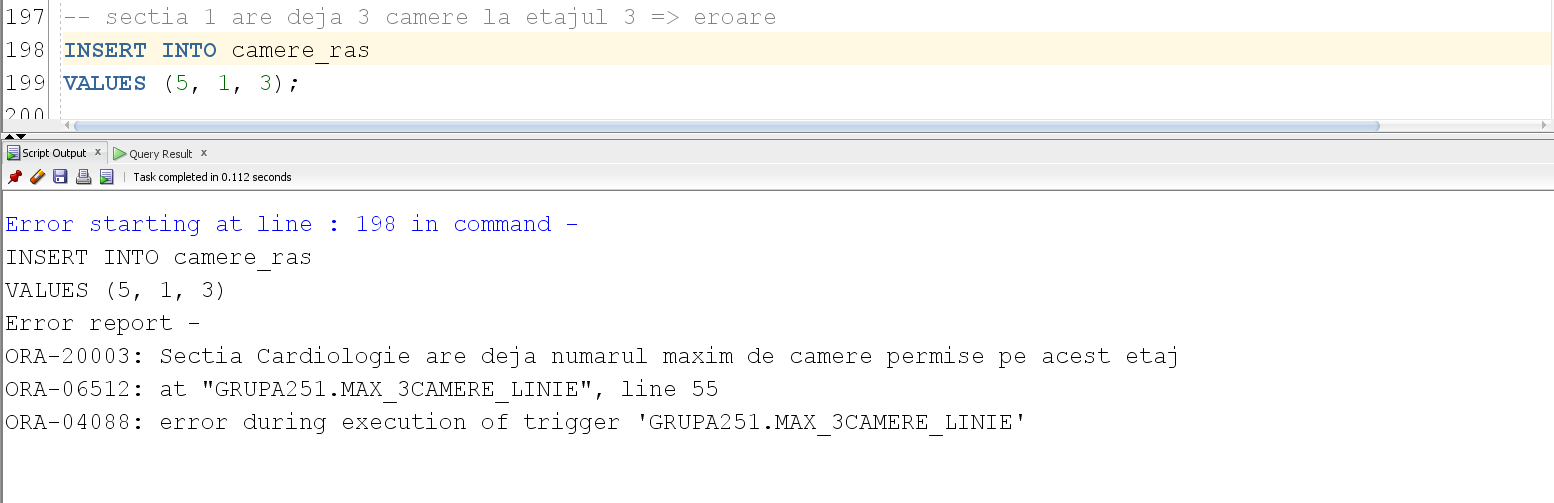
Mai jos vom trata prima variantă (pachet cu 2 triggeri):

| CREATE OR REPLACE PACKAGE pachet\_ras  AS  TYPE linie IS RECORD  (id\_sectie camere\_ras.id\_sectie%TYPE,  id\_etaj camere\_ras.id\_etaj%TYPE,  ct\_camere NUMBER);  TYPE tabl\_ind IS TABLE OF linie  INDEX BY PLS\_INTEGER;  info tabl\_ind;  contor NUMBER := 0;  END;  /  CREATE OR REPLACE TRIGGER max\_3camere\_comanda  BEFORE INSERT OR UPDATE OF id\_etaj, id\_sectie ON camere\_ras  BEGIN  pachet\_ras.contor := 0;  -- cate camere are sectia aia pe etajul dat  SELECT id\_sectie, id\_etaj, COUNT(\*)  BULK COLLECT INTO pachet\_ras.info  FROM camere\_ras  GROUP BY id\_sectie, id\_etaj;    END;  /  CREATE OR REPLACE TRIGGER max\_3camere\_linie  BEFORE INSERT OR UPDATE OF id\_etaj, id\_sectie ON camere\_ras  FOR EACH ROW  DECLARE  -- verificari input  ct\_etaj NUMBER;  etaj\_inexistent EXCEPTION;  ct\_sectie NUMBER;  sectie\_inexistenta EXCEPTION;  -- variabile trigger  nume\_sectie sectie\_ras.denumire%TYPE;  mesaj\_trigger VARCHAR2(100);  nr\_maxim EXCEPTION;  BEGIN    SELECT COUNT(\*)  INTO ct\_etaj  FROM etaj\_ras  WHERE id = :NEW.id\_etaj;    IF ct\_etaj = 0 THEN  RAISE etaj\_inexistent;  END IF;    IF :NEW.id\_sectie IS NOT NULL THEN  SELECT COUNT(\*)  INTO ct\_sectie  FROM sectie\_ras  WHERE id = :NEW.id\_sectie;    IF ct\_sectie = 0 THEN  RAISE sectie\_inexistenta;  END IF;    FOR i IN 1..pachet\_ras.info.LAST LOOP  IF pachet\_ras.info(i).id\_sectie = :NEW.id\_sectie AND pachet\_ras.info(i).id\_etaj = :NEW.id\_etaj  AND pachet\_ras.info(i).ct\_camere + pachet\_ras.contor = 3 THEN  -- luam numele sectiei  SELECT denumire  INTO nume\_sectie  FROM sectie\_ras  WHERE id = :NEW.id\_sectie;    mesaj\_trigger := 'Sectia ' || nume\_sectie || ' are deja numarul maxim de camere permise pe acest etaj';  RAISE nr\_maxim;  END IF;  END LOOP;  pachet\_ras.contor := pachet\_ras.contor + 1;    END IF;  EXCEPTION  WHEN etaj\_inexistent THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Etajul dat nu exista');  WHEN sectie\_inexistenta THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Sectia data nu exista');  WHEN nr\_maxim THEN  RAISE\_APPLICATION\_ERROR(-20003, mesaj\_trigger);  END;  /  -- sectia 1 ia camera 4 de la etajul 3 (momentan are doar 2 camere la etajul 3)  INSERT INTO camere\_ras  VALUES(4, 1, 3);  -- sectia 1 are deja 3 camere la etajul 3 => eroare  INSERT INTO camere\_ras  VALUES (5, 1, 3);  ROLLBACK;  -- nu exista etajul cu id-ul dat => eroare  INSERT INTO camere\_ras  VALUES (1, 2, 7);  -- nu exista sectia data => eroare  INSERT INTO camere\_ras  VALUES (5, 8, 1);  -- insert into ... select ... => merge  INSERT INTO camere\_ras  SELECT 4, 1, 3  FROM DUAL;  ROLLBACK;  -- sectia 6 are 2 camere la etajul 1  -- sectia 5 nu are nicio camera la etajul 1  -- => operatia ar trebui sa poata fi executata  UPDATE camere\_ras  SET id\_sectie = 5  WHERE id\_sectie = 6;  ROLLBACK;  -- exemplul 7  INSERT INTO camere\_ras  VALUES(4, NULL, 1);  INSERT INTO camere\_ras  VALUES(5, NULL, 1);  INSERT INTO camere\_ras  VALUES(6, NULL, 1);  UPDATE camere\_ras  SET id\_sectie = 6  WHERE id\_sectie IS NULL AND id\_etaj = 1;  ROLLBACK;  ALTER TRIGGER max\_3camere\_comanda DISABLE;  ALTER TRIGGER max\_3camere\_linie DISABLE; |
| --- |

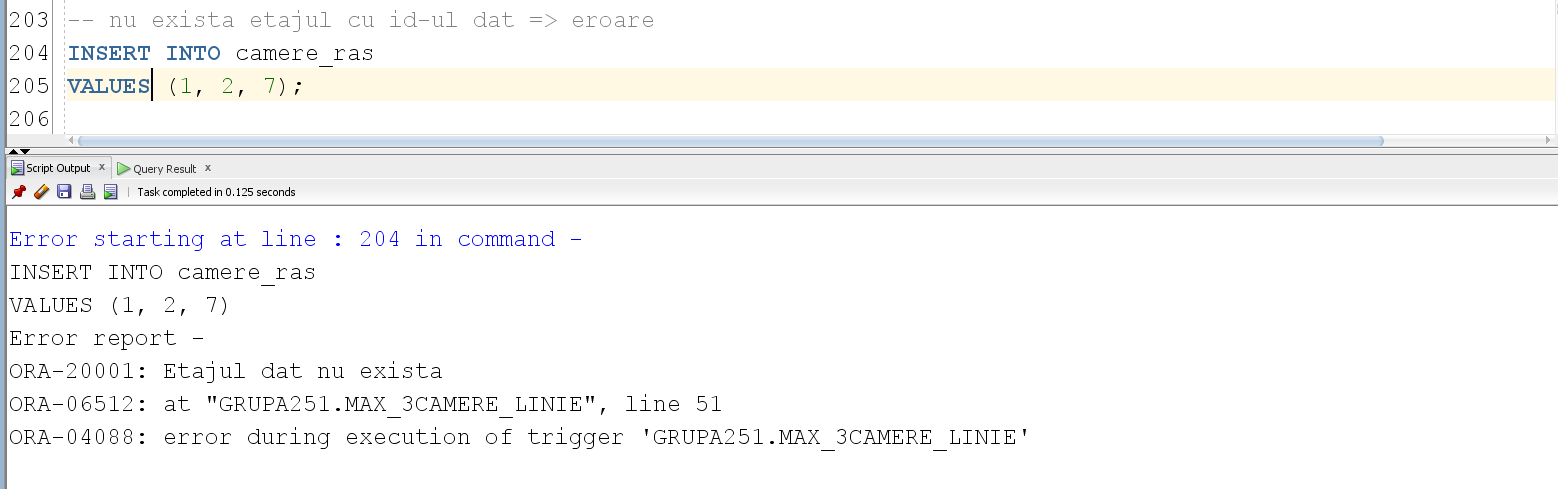


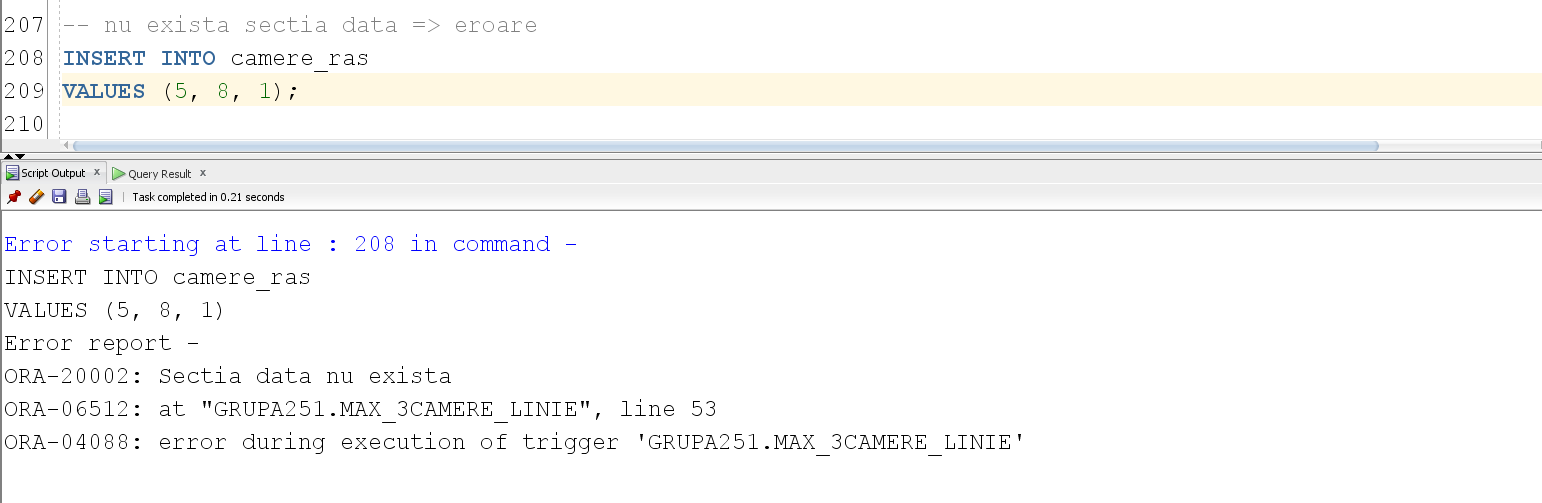
*Exemplul 1*

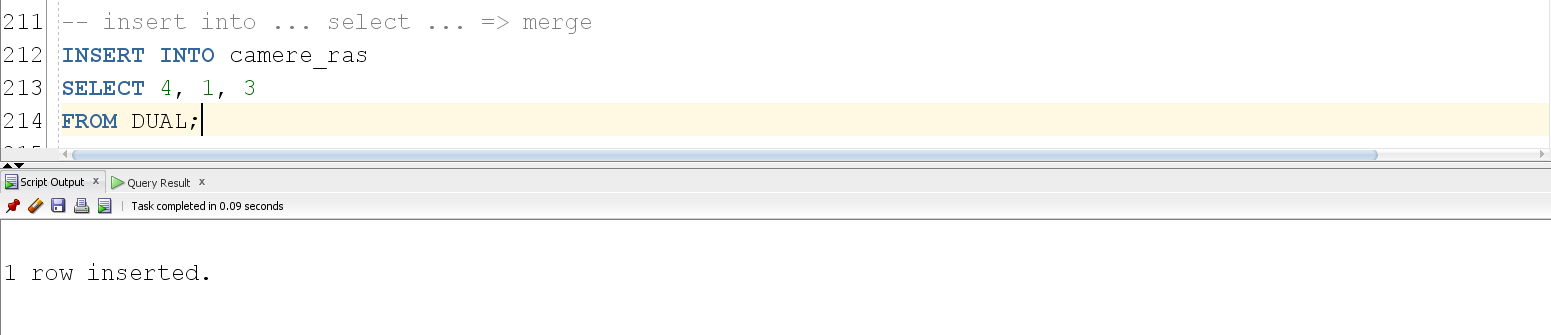
*Exemplul 2*



*Exemplul 3*

*Exemplul 4*

*Exemplul 5*



*Exemplul 6*

*Exemplul 7*