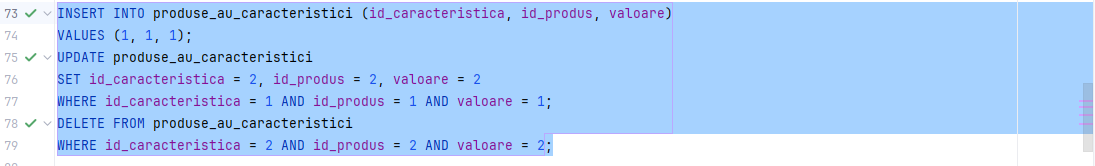
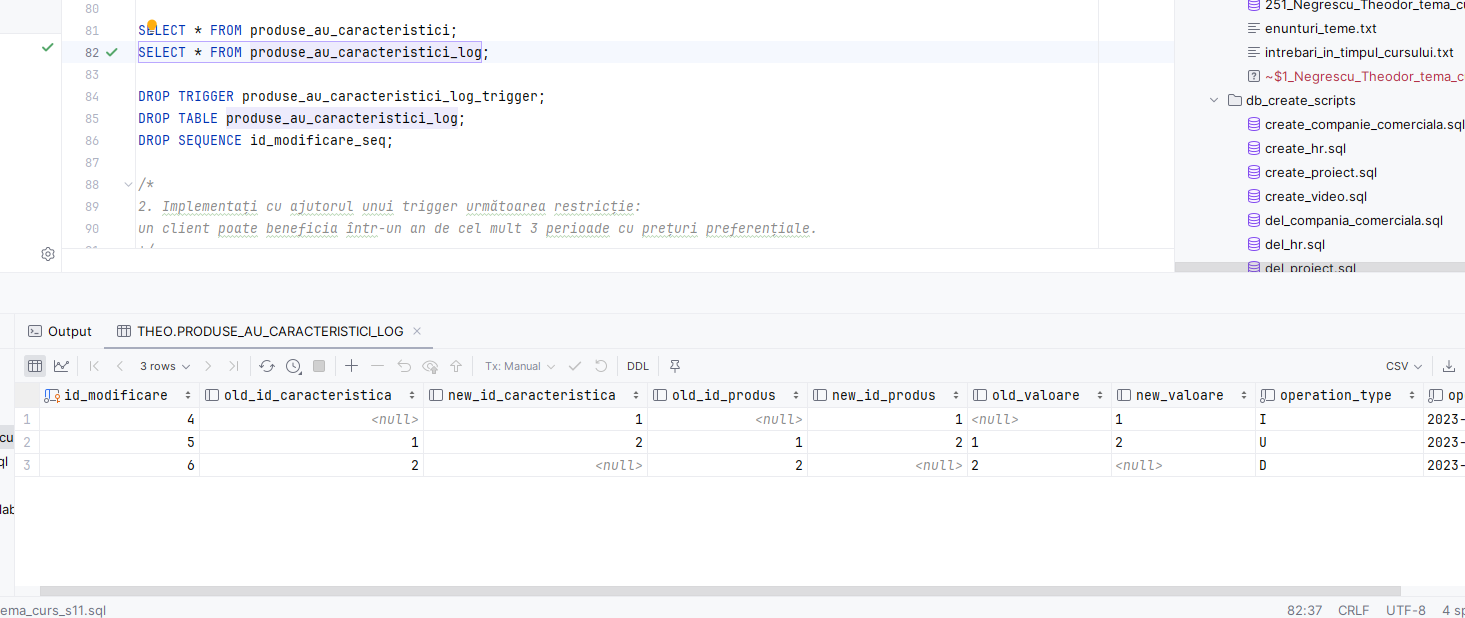
# Ex. 1

|  |
| --- |
| /\*  1. Definiti un tabel de LOG-uri in care sa puteti adăuga operațiile DML efectuate asupra unui tabel asociativ  din schema companie comercială.  În acest tabel vor fi stocate ID-ul, vechea valoare, noua valoare, tipul operației,  momentul în care s-a executat operația și de către cine (userul curent autentificat -  SELECT USER FROM dual).  Implementați un trigger care populează acest tabel cu informațiile menționate mai sus.  \*/  CREATE SEQUENCE ID\_MODIFICARE\_SEQ START WITH 1 INCREMENT BY 1;  CREATE TABLE PRODUSE\_au\_CARACTERISTICI\_log (      id\_modificare NUMBER(10) DEFAULT ID\_MODIFICARE\_SEQ.nextval PRIMARY KEY,      old\_id\_caracteristica NUMBER(38),      new\_id\_caracteristica NUMBER(38),      old\_id\_produs NUMBER(38),      new\_id\_produs NUMBER(38),      old\_valoare VARCHAR2(255),      new\_valoare VARCHAR2(255),      operation\_type VARCHAR2(1) CHECK (operation\_type IN ('I', 'U', 'D')) NOT NULL,      operation\_time DATE DEFAULT SYSDATE NOT NULL,      operation\_user VARCHAR2(30) DEFAULT USER NOT NULL  );  CREATE TRIGGER PRODUSE\_au\_CARACTERISTICI\_log\_trigger      AFTER INSERT OR UPDATE OR DELETE ON PRODUSE\_au\_CARACTERISTICI      FOR EACH ROW  DECLARE      old\_id\_caracteristica NUMBER(38) := NULL;      new\_id\_caracteristica NUMBER(38) := NULL;      old\_id\_produs NUMBER(38) := NULL;      new\_id\_produs NUMBER(38) := NULL;      old\_valoare VARCHAR2(255) := NULL;      new\_valoare VARCHAR2(255) := NULL;      operation\_type VARCHAR2(1) := NULL;  BEGIN      IF INSERTING OR UPDATING THEN          new\_id\_caracteristica := :new.id\_caracteristica;          new\_id\_produs := :new.id\_produs;          new\_valoare := :new.valoare;      END IF;      IF UPDATING OR DELETING THEN          old\_id\_caracteristica := :old.id\_caracteristica;          old\_id\_produs := :old.id\_produs;          old\_valoare := :old.valoare;      END IF;      operation\_type := CASE          WHEN INSERTING THEN 'I'          WHEN UPDATING THEN 'U'          WHEN DELETING THEN 'D'      END;      INSERT INTO PRODUSE\_au\_CARACTERISTICI\_log (          old\_id\_caracteristica,          new\_id\_caracteristica,          old\_id\_produs,          new\_id\_produs,          old\_valoare,          new\_valoare,          operation\_type      )      VALUES (          old\_id\_caracteristica,          new\_id\_caracteristica,          old\_id\_produs,          new\_id\_produs,          old\_valoare,          new\_valoare,          operation\_type      );  END;  /  INSERT INTO produse\_au\_caracteristici (id\_caracteristica, id\_produs, valoare)  VALUES (1, 1, 1);  UPDATE produse\_au\_caracteristici  SET id\_caracteristica = 2, id\_produs = 2, valoare = 2  WHERE id\_caracteristica = 1 AND id\_produs = 1 AND valoare = 1;  DELETE FROM produse\_au\_caracteristici  WHERE id\_caracteristica = 2 AND id\_produs = 2 AND valoare = 2;  SELECT \* FROM produse\_au\_caracteristici;  SELECT \* FROM produse\_au\_caracteristici\_log;  DROP TRIGGER produse\_au\_caracteristici\_log\_trigger;  DROP TABLE produse\_au\_caracteristici\_log;  DROP SEQUENCE id\_modificare\_seq; |

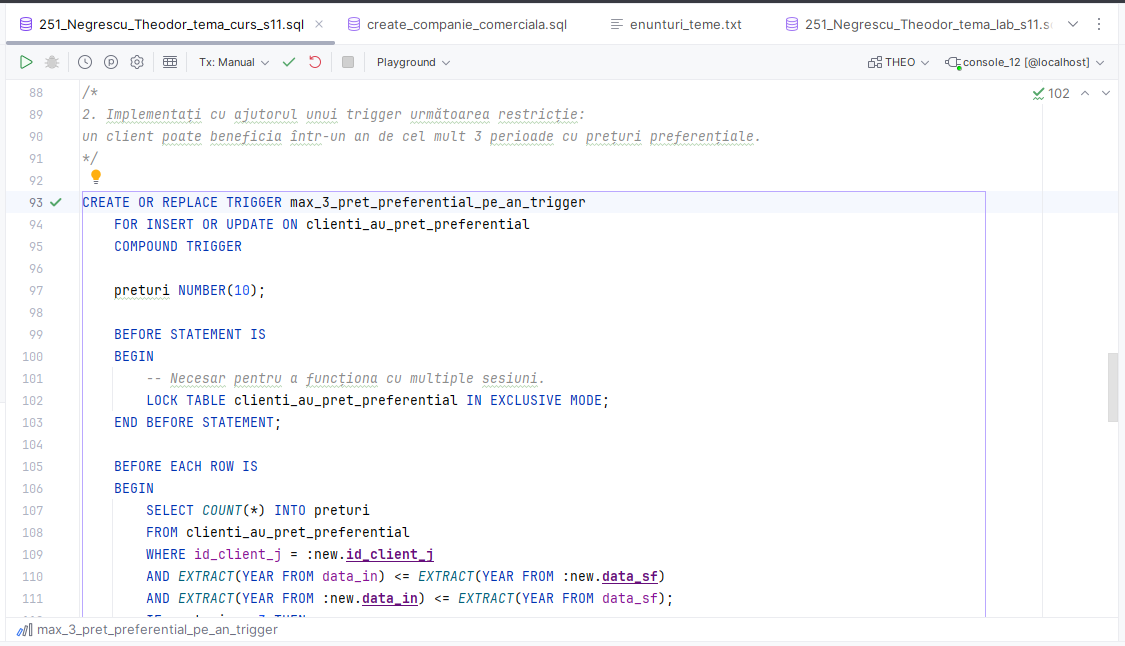


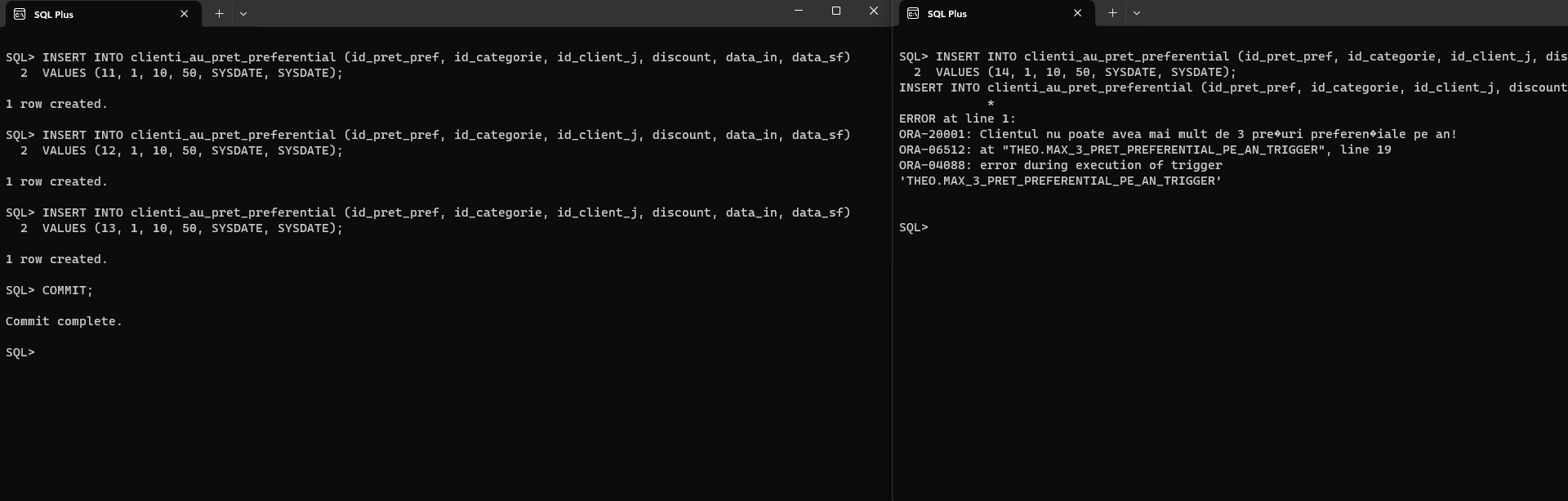




# Ex. 2

|  |
| --- |
| /\*  2. Implementați cu ajutorul unui trigger următoarea restricție:  un client poate beneficia într-un an de cel mult 3 perioade cu prețuri preferențiale.  \*/  CREATE OR REPLACE TRIGGER max\_3\_pret\_preferential\_pe\_an\_trigger      FOR INSERT OR UPDATE ON clienti\_au\_pret\_preferential      COMPOUND TRIGGER      preturi NUMBER(10);      BEFORE STATEMENT IS      BEGIN          -- Necesar pentru a funcționa cu multiple sesiuni.          LOCK TABLE clienti\_au\_pret\_preferential IN EXCLUSIVE MODE;      END BEFORE STATEMENT;      BEFORE EACH ROW IS      BEGIN          SELECT COUNT(\*) INTO preturi          FROM clienti\_au\_pret\_preferential          WHERE id\_client\_j = :new.id\_client\_j          AND EXTRACT(YEAR FROM data\_in) <= EXTRACT(YEAR FROM :new.data\_sf)          AND EXTRACT(YEAR FROM :new.data\_in) <= EXTRACT(YEAR FROM data\_sf);          IF preturi >= 3 THEN              RAISE\_APPLICATION\_ERROR(-20001, 'Clientul nu poate avea mai mult de 3 prețuri preferențiale pe an!');          END IF;      END BEFORE EACH ROW;  END;  -- din sesiunea 1  INSERT INTO clienti\_au\_pret\_preferential (id\_pret\_pref, id\_categorie, id\_client\_j, discount, data\_in, data\_sf)  VALUES (11, 1, 10, 50, SYSDATE, SYSDATE);  INSERT INTO clienti\_au\_pret\_preferential (id\_pret\_pref, id\_categorie, id\_client\_j, discount, data\_in, data\_sf)  VALUES (12, 1, 10, 50, SYSDATE, SYSDATE);  INSERT INTO clienti\_au\_pret\_preferential (id\_pret\_pref, id\_categorie, id\_client\_j, discount, data\_in, data\_sf)  VALUES (13, 1, 10, 50, SYSDATE, SYSDATE);  -- din sesiunea 2  INSERT INTO clienti\_au\_pret\_preferential (id\_pret\_pref, id\_categorie, id\_client\_j, discount, data\_in, data\_sf)  VALUES (14, 1, 10, 50, SYSDATE, SYSDATE);  DROP TRIGGER max\_3\_pret\_preferential\_pe\_an\_trigger; |

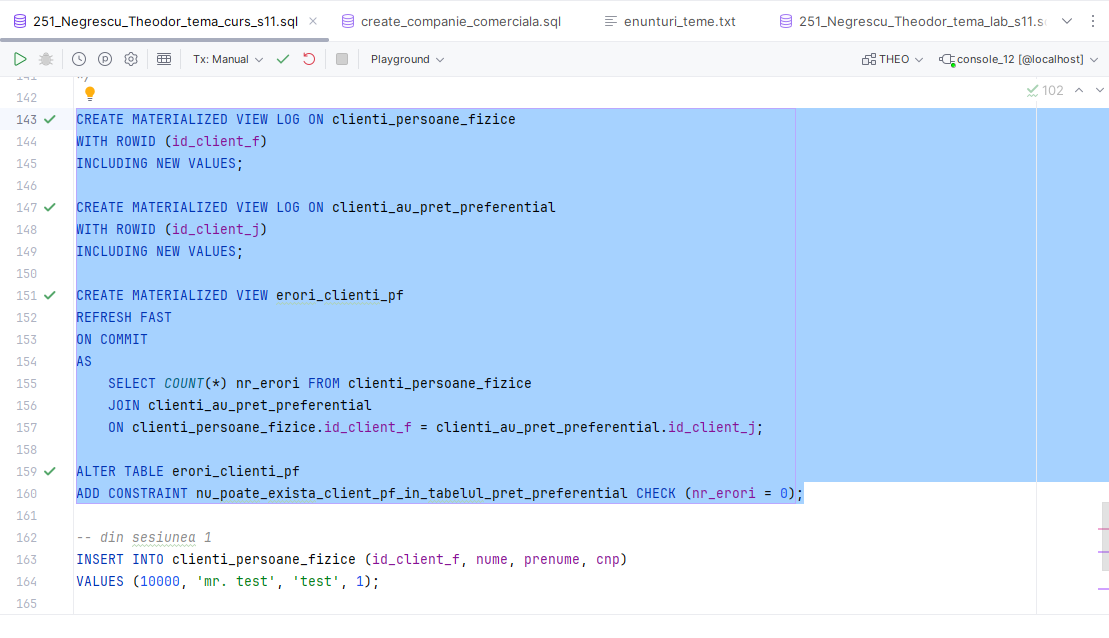


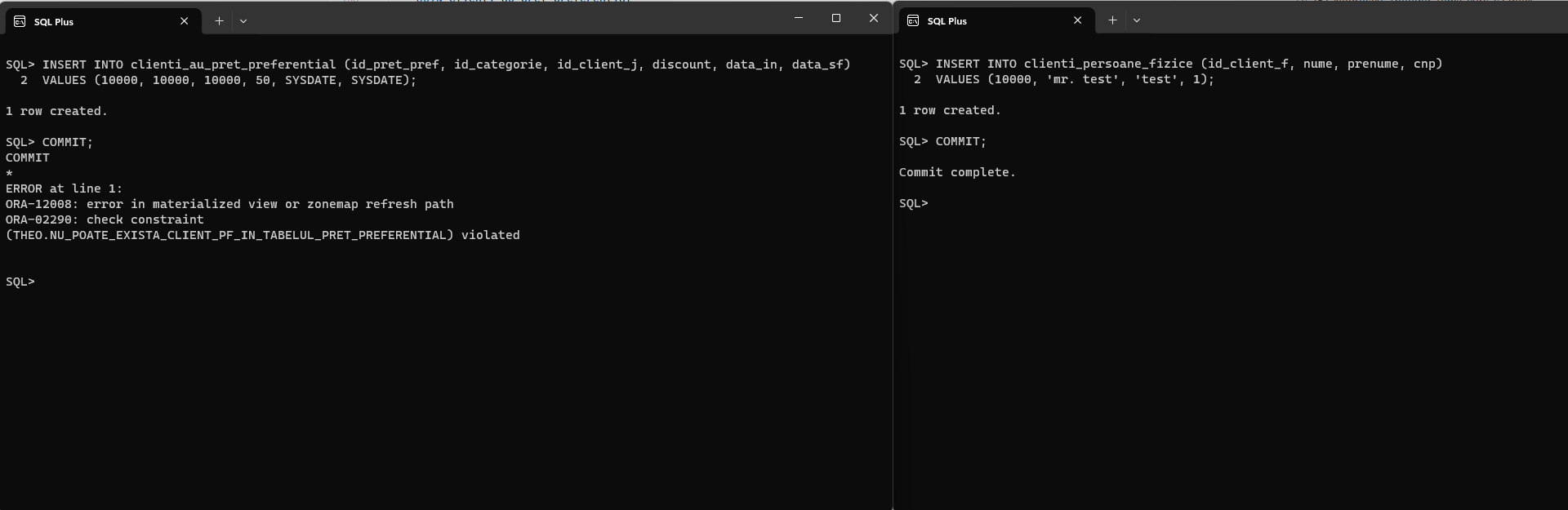


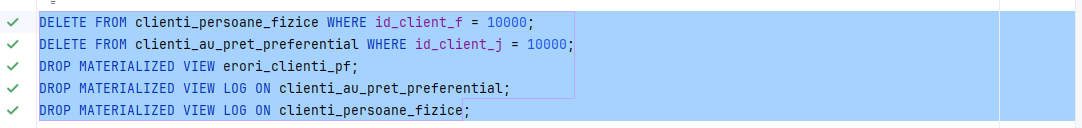


# Ex. 3

|  |
| --- |
| /\*  3. Pe un tabel dependent din schema companie comercială implementați cu ajutorul unui trigger  o constrângere de integritate la alegere.  Observație: trebuie să apară explicit pe ce tabel și care este constrangerea implementată.  Tabelul: clienti\_persoane\_fizice  Nu poate apărea în tabelul clienti\_au\_pret\_preferential.  Nu mi-am dat seama cum să fac constrângere multi-tabel cu un trigger,  așa că am implementat-o în SQL declarativ folosind un materialized view.  \*/  CREATE MATERIALIZED VIEW LOG ON clienti\_persoane\_fizice  WITH ROWID (id\_client\_f)  INCLUDING NEW VALUES;  CREATE MATERIALIZED VIEW LOG ON clienti\_au\_pret\_preferential  WITH ROWID (id\_client\_j)  INCLUDING NEW VALUES;  CREATE MATERIALIZED VIEW erori\_clienti\_pf  REFRESH FAST  ON COMMIT  AS      SELECT COUNT(\*) nr\_erori FROM clienti\_persoane\_fizice      JOIN clienti\_au\_pret\_preferential      ON clienti\_persoane\_fizice.id\_client\_f = clienti\_au\_pret\_preferential.id\_client\_j;  ALTER TABLE erori\_clienti\_pf  ADD CONSTRAINT nu\_poate\_exista\_client\_pf\_in\_tabelul\_pret\_preferential CHECK (nr\_erori = 0);  -- din sesiunea 1  INSERT INTO clienti\_persoane\_fizice (id\_client\_f, nume, prenume, cnp)  VALUES (10000, 'mr. test', 'test', 1);  -- din sesiunea 2  INSERT INTO clienti\_au\_pret\_preferential (id\_pret\_pref, id\_categorie, id\_client\_j, discount, data\_in, data\_sf)  VALUES (10000, 10000, 10000, 50, SYSDATE, SYSDATE);  DELETE FROM clienti\_persoane\_fizice WHERE id\_client\_f = 10000;  DELETE FROM clienti\_au\_pret\_preferential WHERE id\_client\_j = 10000;  DROP MATERIALIZED VIEW erori\_clienti\_pf;  DROP MATERIALIZED VIEW LOG ON clienti\_au\_pret\_preferential;  DROP MATERIALIZED VIEW LOG ON clienti\_persoane\_fizice; |







# Întergul fișier 251\_Negrescu\_Theodor\_tema\_curs\_s11.sql

|  |
| --- |
| /\*  1. Definiti un tabel de LOG-uri in care sa puteti adăuga operațiile DML efectuate asupra unui tabel asociativ  din schema companie comercială.  În acest tabel vor fi stocate ID-ul, vechea valoare, noua valoare, tipul operației,  momentul în care s-a executat operația și de către cine (userul curent autentificat -  SELECT USER FROM dual).  Implementați un trigger care populează acest tabel cu informațiile menționate mai sus.  \*/  CREATE SEQUENCE ID\_MODIFICARE\_SEQ START WITH 1 INCREMENT BY 1;  CREATE TABLE PRODUSE\_au\_CARACTERISTICI\_log (      id\_modificare NUMBER(10) DEFAULT ID\_MODIFICARE\_SEQ.nextval PRIMARY KEY,      old\_id\_caracteristica NUMBER(38),      new\_id\_caracteristica NUMBER(38),      old\_id\_produs NUMBER(38),      new\_id\_produs NUMBER(38),      old\_valoare VARCHAR2(255),      new\_valoare VARCHAR2(255),      operation\_type VARCHAR2(1) CHECK (operation\_type IN ('I', 'U', 'D')) NOT NULL,      operation\_time DATE DEFAULT SYSDATE NOT NULL,      operation\_user VARCHAR2(30) DEFAULT USER NOT NULL  );  CREATE TRIGGER PRODUSE\_au\_CARACTERISTICI\_log\_trigger      AFTER INSERT OR UPDATE OR DELETE ON PRODUSE\_au\_CARACTERISTICI      FOR EACH ROW  DECLARE      old\_id\_caracteristica NUMBER(38) := NULL;      new\_id\_caracteristica NUMBER(38) := NULL;      old\_id\_produs NUMBER(38) := NULL;      new\_id\_produs NUMBER(38) := NULL;      old\_valoare VARCHAR2(255) := NULL;      new\_valoare VARCHAR2(255) := NULL;      operation\_type VARCHAR2(1) := NULL;  BEGIN      IF INSERTING OR UPDATING THEN          new\_id\_caracteristica := :new.id\_caracteristica;          new\_id\_produs := :new.id\_produs;          new\_valoare := :new.valoare;      END IF;      IF UPDATING OR DELETING THEN          old\_id\_caracteristica := :old.id\_caracteristica;          old\_id\_produs := :old.id\_produs;          old\_valoare := :old.valoare;      END IF;      operation\_type := CASE          WHEN INSERTING THEN 'I'          WHEN UPDATING THEN 'U'          WHEN DELETING THEN 'D'      END;      INSERT INTO PRODUSE\_au\_CARACTERISTICI\_log (          old\_id\_caracteristica,          new\_id\_caracteristica,          old\_id\_produs,          new\_id\_produs,          old\_valoare,          new\_valoare,          operation\_type      )      VALUES (          old\_id\_caracteristica,          new\_id\_caracteristica,          old\_id\_produs,          new\_id\_produs,          old\_valoare,          new\_valoare,          operation\_type      );  END;  /  INSERT INTO produse\_au\_caracteristici (id\_caracteristica, id\_produs, valoare)  VALUES (1, 1, 1);  UPDATE produse\_au\_caracteristici  SET id\_caracteristica = 2, id\_produs = 2, valoare = 2  WHERE id\_caracteristica = 1 AND id\_produs = 1 AND valoare = 1;  DELETE FROM produse\_au\_caracteristici  WHERE id\_caracteristica = 2 AND id\_produs = 2 AND valoare = 2;  SELECT \* FROM produse\_au\_caracteristici;  SELECT \* FROM produse\_au\_caracteristici\_log;  DROP TRIGGER produse\_au\_caracteristici\_log\_trigger;  DROP TABLE produse\_au\_caracteristici\_log;  DROP SEQUENCE id\_modificare\_seq;  /\*  2. Implementați cu ajutorul unui trigger următoarea restricție:  un client poate beneficia într-un an de cel mult 3 perioade cu prețuri preferențiale.  \*/  CREATE OR REPLACE TRIGGER max\_3\_pret\_preferential\_pe\_an\_trigger      FOR INSERT OR UPDATE ON clienti\_au\_pret\_preferential      COMPOUND TRIGGER      preturi NUMBER(10);      BEFORE STATEMENT IS      BEGIN          -- Necesar pentru a funcționa cu multiple sesiuni.          LOCK TABLE clienti\_au\_pret\_preferential IN EXCLUSIVE MODE;      END BEFORE STATEMENT;      BEFORE EACH ROW IS      BEGIN          SELECT COUNT(\*) INTO preturi          FROM clienti\_au\_pret\_preferential          WHERE id\_client\_j = :new.id\_client\_j          AND EXTRACT(YEAR FROM data\_in) <= EXTRACT(YEAR FROM :new.data\_sf)          AND EXTRACT(YEAR FROM :new.data\_in) <= EXTRACT(YEAR FROM data\_sf);          IF preturi >= 3 THEN              RAISE\_APPLICATION\_ERROR(-20001, 'Clientul nu poate avea mai mult de 3 prețuri preferențiale pe an!');          END IF;      END BEFORE EACH ROW;  END;  -- din sesiunea 1  INSERT INTO clienti\_au\_pret\_preferential (id\_pret\_pref, id\_categorie, id\_client\_j, discount, data\_in, data\_sf)  VALUES (11, 1, 10, 50, SYSDATE, SYSDATE);  INSERT INTO clienti\_au\_pret\_preferential (id\_pret\_pref, id\_categorie, id\_client\_j, discount, data\_in, data\_sf)  VALUES (12, 1, 10, 50, SYSDATE, SYSDATE);  INSERT INTO clienti\_au\_pret\_preferential (id\_pret\_pref, id\_categorie, id\_client\_j, discount, data\_in, data\_sf)  VALUES (13, 1, 10, 50, SYSDATE, SYSDATE);  -- din sesiunea 2  INSERT INTO clienti\_au\_pret\_preferential (id\_pret\_pref, id\_categorie, id\_client\_j, discount, data\_in, data\_sf)  VALUES (14, 1, 10, 50, SYSDATE, SYSDATE);  DROP TRIGGER max\_3\_pret\_preferential\_pe\_an\_trigger;  /\*  3. Pe un tabel dependent din schema companie comercială implementați cu ajutorul unui trigger  o constrângere de integritate la alegere.  Observație: trebuie să apară explicit pe ce tabel și care este constrangerea implementată.  Tabelul: clienti\_persoane\_fizice  Nu poate apărea în tabelul clienti\_au\_pret\_preferential.  Nu mi-am dat seama cum să fac constrângere multi-tabel cu un trigger,  așa că am implementat-o în SQL declarativ folosind un materialized view.  \*/  CREATE MATERIALIZED VIEW LOG ON clienti\_persoane\_fizice  WITH ROWID (id\_client\_f)  INCLUDING NEW VALUES;  CREATE MATERIALIZED VIEW LOG ON clienti\_au\_pret\_preferential  WITH ROWID (id\_client\_j)  INCLUDING NEW VALUES;  CREATE MATERIALIZED VIEW erori\_clienti\_pf  REFRESH FAST  ON COMMIT  AS      SELECT COUNT(\*) nr\_erori FROM clienti\_persoane\_fizice      JOIN clienti\_au\_pret\_preferential      ON clienti\_persoane\_fizice.id\_client\_f = clienti\_au\_pret\_preferential.id\_client\_j;  ALTER TABLE erori\_clienti\_pf  ADD CONSTRAINT nu\_poate\_exista\_client\_pf\_in\_tabelul\_pret\_preferential CHECK (nr\_erori = 0);  -- din sesiunea 1  INSERT INTO clienti\_persoane\_fizice (id\_client\_f, nume, prenume, cnp)  VALUES (10000, 'mr. test', 'test', 1);  -- din sesiunea 2  INSERT INTO clienti\_au\_pret\_preferential (id\_pret\_pref, id\_categorie, id\_client\_j, discount, data\_in, data\_sf)  VALUES (10000, 10000, 10000, 50, SYSDATE, SYSDATE);  DELETE FROM clienti\_persoane\_fizice WHERE id\_client\_f = 10000;  DELETE FROM clienti\_au\_pret\_preferential WHERE id\_client\_j = 10000;  DROP MATERIALIZED VIEW erori\_clienti\_pf;  DROP MATERIALIZED VIEW LOG ON clienti\_au\_pret\_preferential;  DROP MATERIALIZED VIEW LOG ON clienti\_persoane\_fizice; |