**--partea de SQL cu initializarile bazei de date (ex 4,5):**

CREATE TABLE locatie

(cod\_locatie NUMBER(10) NOT NULL,

judet VARCHAR2(20),

oras VARCHAR2(20),

strada VARCHAR2(20) ,

numar\_cladire NUMBER(10),

apartament NUMBER(10),

CONSTRAINT pk\_cod\_locatie PRIMARY KEY (cod\_locatie)

);

CREATE TABLE lutier

(cod\_lutier NUMBER(10) NOT NULL,

nume VARCHAR2(20),

prenume VARCHAR2(20),

nr\_tel NUMBER(10) ,

email VARCHAR2(20),

cod\_locatie NUMBER(10) NOT NULL,

CONSTRAINT pk\_cod\_lutier PRIMARY KEY (cod\_lutier),

CONSTRAINT fk\_cod\_locatie FOREIGN KEY (cod\_locatie)

REFERENCES locatie(cod\_locatie)

);

CREATE TABLE magazin

(cod\_magazin NUMBER(10) NOT NULL,

nume VARCHAR2(20),

ora\_deschidere NUMBER(4),

ora\_inchidere NUMBER(4),

cod\_locatie NUMBER(10) NOT NULL,

CONSTRAINT pk\_cod\_magazin PRIMARY KEY (cod\_magazin),

CONSTRAINT fk1\_cod\_locatie FOREIGN KEY (cod\_locatie)

REFERENCES locatie(cod\_locatie),

CONSTRAINT chk\_ora\_deschidere CHECK (ora\_deschidere >= 0 AND ora\_deschidere < 2400),

CONSTRAINT chk\_ora\_inchidere CHECK (ora\_inchidere >= 0 AND ora\_inchidere < 2400)

);

CREATE TABLE depozit

(cod\_depozit NUMBER(10) NOT NULL,

nume VARCHAR2(20),

cod\_locatie NUMBER(10) NOT NULL,

CONSTRAINT pk\_cod\_depozit PRIMARY KEY (cod\_depozit),

CONSTRAINT fk2\_cod\_locatie FOREIGN KEY (cod\_locatie)

REFERENCES locatie(cod\_locatie)

);

CREATE TABLE functie

(cod\_functie NUMBER(10) NOT NULL,

nume VARCHAR2(20),

salariu NUMBER(10),

CONSTRAINT pk\_cod\_functie PRIMARY KEY (cod\_functie),

CONSTRAINT chk\_salariu CHECK (salariu > 0)

);

CREATE TABLE angajat

(cod\_angajat NUMBER(10) NOT NULL,

nume VARCHAR2(20),

prenume VARCHAR2(20),

data\_angajare DATE,

cod\_magazin NUMBER(10) NOT NULL,

cod\_functie NUMBER(10) NOT NULL,

CONSTRAINT pk\_cod\_angajat PRIMARY KEY (cod\_angajat),

CONSTRAINT fk1\_cod\_magazin FOREIGN KEY (cod\_magazin)

REFERENCES magazin(cod\_magazin),

CONSTRAINT fk\_cod\_functie FOREIGN KEY (cod\_functie)

REFERENCES functie (cod\_functie)

);

CREATE TABLE vehicul

(nr\_inmat VARCHAR2(20) NOT NULL,

categorie VARCHAR2(20) NOT NULL,

ani\_vechime NUMBER(3),

cod\_depozit NUMBER(10) NOT NULL,

CONSTRAINT pk\_nr\_inmat PRIMARY KEY (nr\_inmat),

CONSTRAINT fk1\_cod\_depozit FOREIGN KEY (cod\_depozit)

REFERENCES depozit(cod\_depozit)

);

CREATE TABLE sofer

(cod\_sofer NUMBER(10) NOT NULL,

nume VARCHAR2(20),

prenume VARCHAR2(20),

telefon NUMBER(20),

categorie\_carnet VARCHAR2(20) NOT NULL,

cod\_depozit NUMBER(10) NOT NULL,

CONSTRAINT pk\_cod\_sofer PRIMARY KEY (cod\_sofer),

CONSTRAINT fk2\_cod\_depozit FOREIGN KEY (cod\_depozit)

REFERENCES depozit(cod\_depozit)

);

CREATE TABLE oprire

(localitate VARCHAR2(20) NOT NULL,

CONSTRAINT pk\_localitate PRIMARY KEY (localitate)

);

CREATE TABLE transport

(cod\_transport NUMBER(10) NOT NULL,

nr\_inmat VARCHAR2(20) NOT NULL,

cod\_sofer NUMBER(10) NOT NULL,

data\_plecare DATE,

ora\_plecare NUMBER(4),

CONSTRAINT pk\_cod\_transport PRIMARY KEY (cod\_transport),

CONSTRAINT fk1\_cod\_sofer FOREIGN KEY (cod\_sofer)

REFERENCES sofer(cod\_sofer),

CONSTRAINT fk1\_nr\_inmat FOREIGN KEY (nr\_inmat)

REFERENCES vehicul(nr\_inmat),

CONSTRAINT chk1\_ora\_plecare CHECK (ora\_plecare >= 0 AND ora\_plecare < 2400)

);

CREATE TABLE oprire\_transport

(cod\_transport NUMBER(10) NOT NULL,

localitate VARCHAR2(20) NOT NULL,

include\_vama VARCHAR2(2),

durata VARCHAR2(20),

CONSTRAINT pk\_cod\_transport\_localitate PRIMARY KEY (cod\_transport, localitate),

CONSTRAINT fk\_cod\_transport FOREIGN KEY (cod\_transport)

REFERENCES transport(cod\_transport),

CONSTRAINT fk\_localitate FOREIGN KEY (localitate)

REFERENCES oprire(localitate),

CONSTRAINT chk\_include\_vama CHECK (LOWER(include\_vama) LIKE 'da' OR LOWER(include\_vama) LIKE 'nu')

);

CREATE TABLE livrare

(cod\_livrare NUMBER(10) NOT NULL,

cod\_transport NUMBER(10) NOT NULL,

cod\_magazin NUMBER(10) NOT NULL,

status\_livrare VARCHAR2(20) NOT NULL,

CONSTRAINT pk\_cod\_livrare PRIMARY KEY (cod\_livrare),

CONSTRAINT fk1\_cod\_transport FOREIGN KEY (cod\_transport)

REFERENCES transport(cod\_transport),

CONSTRAINT fk2\_cod\_magazin FOREIGN KEY (cod\_magazin)

REFERENCES magazin(cod\_magazin)

);

CREATE TABLE produs

(cod\_produs NUMBER(10) NOT NULL,

cod\_depozit NUMBER(10),

cod\_magazin NUMBER(10),

cod\_lutier NUMBER(10),

cod\_livrare NUMBER(10),

tip VARCHAR2(20),

nume VARCHAR2(20),

pret NUMBER(10),

greutate NUMBER(10),

material\_predominant VARCHAR2(20),

producator VARCHAR2(20),

CONSTRAINT pk\_cod\_produs PRIMARY KEY (cod\_produs),

CONSTRAINT fk4\_cod\_depozit FOREIGN KEY (cod\_depozit)

REFERENCES depozit(cod\_depozit),

CONSTRAINT fk4\_cod\_magazin FOREIGN KEY (cod\_magazin)

REFERENCES magazin(cod\_magazin),

CONSTRAINT fk4\_cod\_lutier FOREIGN KEY (cod\_lutier)

REFERENCES lutier(cod\_lutier),

CONSTRAINT fk4\_cod\_livrare FOREIGN KEY (cod\_livrare)

REFERENCES livrare(cod\_livrare),

CONSTRAINT chk4\_pret CHECK (pret >= 0),

CONSTRAINT chk4\_greutate CHECK (greutate >= 0),

CONSTRAINT chk\_singur\_loc CHECK (DECODE(cod\_depozit, NULL, 0, 1) + DECODE(cod\_magazin, NULL, 0, 1) + DECODE(cod\_lutier, NULL, 0, 1) + DECODE(cod\_livrare, NULL, 0, 1) <= 1)

);

ALTER TABLE produs

MODIFY cod\_depozit DEFAULT NULL;

ALTER TABLE produs

MODIFY cod\_magazin DEFAULT NULL;

ALTER TABLE produs

MODIFY cod\_lutier DEFAULT NULL;

ALTER TABLE produs

MODIFY cod\_livrare DEFAULT NULL;

ALTER TABLE livrare

MODIFY status\_livrare DEFAULT 'pe drum';

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(1, 'Ilfov', 'Bucuresti', 'Strada Soarelui', 2, 6);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(2, 'Ilfov', 'Bucuresti', 'Calea Victoriei', 44, 15);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(3, 'Ilfov', 'Bucuresti', 'Strada Aurel Vlaicu', 63, 22);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(4, 'Ilfov', 'Bucuresti', 'Strada Dealului', 1, 19);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(5, 'Sibiu', 'Sibiu', 'Strada Ion Creanga', 23, 55);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(6, 'Sibiu', 'Sibiu', 'Strada Morii', 50, 14);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(7, 'Sibiu', 'Sibiu', 'Strada Rasaritului', 33, 33);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(8, 'Constanta', 'Constanta', 'Strada Brezei', 53, 84);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(9, 'Arges', 'Pitesti', 'Calea Mihai Viteazul', 52, 45);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(10, 'Dolj', 'Craiova', 'Aleea Dumbravei', 12, 14);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(11, 'Constanta', 'Constanta', 'Aleea Scoicilor', 1, 38);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(12, 'Arges', 'Topoloveni', 'S-duta Topolovenilor', 2, 4);

INSERT INTO locatie(cod\_locatie, judet, oras, strada, numar\_cladire, apartament) VALUES(13, 'Dolj', 'Craiova', 'Via Sarii', 1, 8);

INSERT INTO lutier(cod\_lutier, nume, prenume, nr\_tel, email, cod\_locatie) VALUES(1, 'Ionescu', 'Gigel', '0732103441', 'igigleandr@yahoo.com', 1);

INSERT INTO lutier(cod\_lutier, nume, prenume, nr\_tel, email, cod\_locatie) VALUES(6, 'Ionescu', 'Andrei', '0732214562', 'iandreii@yahoo.com', 1);

INSERT INTO lutier(cod\_lutier, nume, prenume, nr\_tel, email, cod\_locatie) VALUES(2, 'Popescu', 'Marcel', '0730286884', 'tituspoma@yahoo.com', 2);

INSERT INTO lutier(cod\_lutier, nume, prenume, nr\_tel, email, cod\_locatie) VALUES(3, 'Stroe', 'Andrei', '0734940265', 'stroe23@yahoo.com', 3);

INSERT INTO lutier(cod\_lutier, nume, prenume, nr\_tel, email, cod\_locatie) VALUES(4, 'Stoici', 'Alexandru', '0735273197', 'stoalex2@yahoo.com', 4);

INSERT INTO lutier(cod\_lutier, nume, prenume, nr\_tel, email, cod\_locatie) VALUES(5, 'Aioanei', 'Marius', '0730784294', 'aioaneity@google.com', 5);

INSERT INTO magazin(cod\_magazin, nume, ora\_deschidere, ora\_inchidere, cod\_locatie) VALUES(1, 'Geo+Alina Soarelui', 800, 1900, 1);

INSERT INTO magazin(cod\_magazin, nume, ora\_deschidere, ora\_inchidere, cod\_locatie) VALUES(2, 'Geo+Alina Morii', 800, 2000, 6);

INSERT INTO magazin(cod\_magazin, nume, ora\_deschidere, ora\_inchidere, cod\_locatie) VALUES(3, 'Geo+Alina Tanta', 1200, 1600, 8);

INSERT INTO magazin(cod\_magazin, nume, ora\_deschidere, ora\_inchidere, cod\_locatie) VALUES(4, 'Geo+Alina Viteazul', 800, 1900, 9);

INSERT INTO magazin(cod\_magazin, nume, ora\_deschidere, ora\_inchidere, cod\_locatie) VALUES(5, 'Geo+Alina Dumbravii', 700, 1800, 10);

INSERT INTO depozit(cod\_depozit, nume, cod\_locatie) VALUES (1, 'Ilfov Stocks', 1);

INSERT INTO depozit(cod\_depozit, nume, cod\_locatie) VALUES (2, 'Depozit Dealului', 4);

INSERT INTO depozit(cod\_depozit, nume, cod\_locatie) VALUES (3, 'ShineDepo', 7);

INSERT INTO depozit(cod\_depozit, nume, cod\_locatie) VALUES (4, 'Magazia lui Marin', 11);

INSERT INTO depozit(cod\_depozit, nume, cod\_locatie) VALUES (5, 'Depozit Blocului', 12);

INSERT INTO depozit(cod\_depozit, nume, cod\_locatie) VALUES (6, 'Depozit Blocului', 12);

INSERT INTO depozit(cod\_depozit, nume, cod\_locatie) VALUES (7, 'Depozit599', 13);

INSERT INTO functie(cod\_functie, nume, salariu) VALUES (1, 'Director Stocuri', 4400);

INSERT INTO functie(cod\_functie, nume, salariu) VALUES (2, 'Agent Vanzari', 2800);

INSERT INTO functie(cod\_functie, nume, salariu) VALUES (3, 'Manager Vanzari', 3700);

INSERT INTO functie(cod\_functie, nume, salariu) VALUES (4, 'Specialist Electr.', 3200);

INSERT INTO functie(cod\_functie, nume, salariu) VALUES (5, 'Resp. Intretinere', 2300);

INSERT INTO angajat(cod\_angajat, cod\_magazin, cod\_functie, nume, prenume, data\_angajare) VALUES (1, 1, 1, 'Pavelescu', 'Tiberiu', '22-04-2015');

INSERT INTO angajat(cod\_angajat, cod\_magazin, cod\_functie, nume, prenume, data\_angajare) VALUES (2, 2, 1, 'Raif', 'Andreea', '27-10-2020');

INSERT INTO angajat(cod\_angajat, cod\_magazin, cod\_functie, nume, prenume, data\_angajare) VALUES (3, 3, 1, 'Marcelo', 'Juan', '10-11-2018');

INSERT INTO angajat(cod\_angajat, cod\_magazin, cod\_functie, nume, prenume, data\_angajare) VALUES (4, 4, 1, 'Chitoi', 'Dragos', '16-04-2017');

INSERT INTO angajat(cod\_angajat, cod\_magazin, cod\_functie, nume, prenume, data\_angajare) VALUES (5, 5, 1, 'Pravale', 'Diana', '25-08-2012');

INSERT INTO vehicul(nr\_inmat, categorie, ani\_vechime, cod\_depozit) VALUES ('DJ 131 ANA', 'BE',7,1);

INSERT INTO vehicul(nr\_inmat, categorie, ani\_vechime, cod\_depozit) VALUES ('IF 50 PTR', 'C',4,3);

INSERT INTO vehicul(nr\_inmat, categorie, ani\_vechime, cod\_depozit) VALUES ('IF 237 MSU', 'C', 1, 3);

INSERT INTO vehicul(nr\_inmat, categorie, ani\_vechime, cod\_depozit) VALUES ('IF 09 TMY', 'C', 3, 4);

INSERT INTO vehicul(nr\_inmat, categorie, ani\_vechime, cod\_depozit) VALUES ('SB 672 CMN', 'C', 5, 5);

INSERT INTO vehicul(nr\_inmat, categorie, ani\_vechime, cod\_depozit) VALUES ('CT 116 MLG', 'BE', 4,6);

INSERT INTO vehicul(nr\_inmat, categorie, ani\_vechime, cod\_depozit) VALUES ('AG 198 IBY', 'C', 3, 7);

INSERT INTO sofer(cod\_sofer, nume, prenume, telefon, categorie\_carnet, cod\_depozit) VALUES (1, 'Pirvu', 'George', 0773721895, 'BE', 1);

INSERT INTO sofer(cod\_sofer, nume, prenume, telefon, categorie\_carnet, cod\_depozit) VALUES (2, 'Lupu', 'Toma', 0743265830, 'C', 3);

INSERT INTO sofer(cod\_sofer, nume, prenume, telefon, categorie\_carnet, cod\_depozit) VALUES (3, 'Lupu', 'Vasile', 0243195285, 'C', 3);

INSERT INTO sofer(cod\_sofer, nume, prenume, telefon, categorie\_carnet, cod\_depozit) VALUES (4, 'Niculescu', 'Petra', 0218933829, 'C', 4);

INSERT INTO sofer(cod\_sofer, nume, prenume, telefon, categorie\_carnet, cod\_depozit) VALUES (5, 'Amzu', 'Nicu', 0246942666, 'C', 5);

INSERT INTO sofer(cod\_sofer, nume, prenume, telefon, categorie\_carnet, cod\_depozit) VALUES (6, 'Cismiu', 'Andra', 0770883829, 'BE', 6);

INSERT INTO sofer(cod\_sofer, nume, prenume, telefon, categorie\_carnet, cod\_depozit) VALUES (7, 'Lica', 'Tanase', 0040012421691, 'C', 7);

INSERT INTO oprire(localitate) VALUES ('Bais');

INSERT INTO oprire(localitate) VALUES ('Slatina');

INSERT INTO oprire(localitate) VALUES ('Lehliu Gara');

INSERT INTO oprire(localitate) VALUES ('Fetesti');

INSERT INTO oprire(localitate) VALUES ('Medgidia');

INSERT INTO oprire(localitate) VALUES ('Ramnicu Valcea');

INSERT INTO oprire(localitate) VALUES ('Alexandria');

INSERT INTO transport(cod\_transport, nr\_inmat, cod\_sofer, data\_plecare, ora\_plecare) VALUES(1, 'AG 198 IBY', 7, '25-05-2021', 700);

INSERT INTO transport(cod\_transport, nr\_inmat, cod\_sofer, data\_plecare, ora\_plecare) VALUES(2, 'AG 198 IBY', 7, '06-06-2021', 800);

INSERT INTO transport(cod\_transport, nr\_inmat, cod\_sofer, data\_plecare, ora\_plecare) VALUES(3, 'DJ 131 ANA', 1, '30-05-2021', 930);

INSERT INTO transport(cod\_transport, nr\_inmat, cod\_sofer, data\_plecare, ora\_plecare) VALUES(4, 'DJ 131 ANA', 1, '17-06-2021', NULL);

INSERT INTO transport(cod\_transport, nr\_inmat, cod\_sofer, data\_plecare, ora\_plecare) VALUES(5, 'AG 198 IBY', 7, '28-06-2021', NULL);

INSERT INTO transport(cod\_transport, nr\_inmat, cod\_sofer, data\_plecare, ora\_plecare) VALUES(6, 'DJ 131 ANA', 7, '19-05-2021', NULL);

INSERT INTO transport(cod\_transport, nr\_inmat, cod\_sofer, data\_plecare, ora\_plecare) VALUES(7, 'IF 09 TMY', 4, '28-06-2021', NULL);

INSERT INTO transport(cod\_transport, nr\_inmat, cod\_sofer, data\_plecare, ora\_plecare) VALUES(8, 'SB 672 CMN', 5, '01-07-2021', 1200);

INSERT INTO transport(cod\_transport, nr\_inmat, cod\_sofer, data\_plecare, ora\_plecare) VALUES(9, 'SB 672 CMN', 5, '01-07-2021', 1930);

INSERT INTO transport(cod\_transport, nr\_inmat, cod\_sofer, data\_plecare, ora\_plecare) VALUES(10, 'CT 116 MLG', 6, '01-07-2021', 2230);

INSERT INTO oprire\_transport(cod\_transport, localitate, include\_vama, durata) VALUES (1, 'Bais', 'nu', NULL);

INSERT INTO oprire\_transport(cod\_transport, localitate, include\_vama, durata) VALUES (1, 'Slatina', 'nu', 'aprox. 30 min.');

INSERT INTO oprire\_transport(cod\_transport, localitate, include\_vama, durata) VALUES (2, 'Slatina', 'nu', 'aprox. 1h');

INSERT INTO oprire\_transport(cod\_transport, localitate, include\_vama, durata) VALUES (3, 'Lehliu Gara', 'nu', NULL);

INSERT INTO oprire\_transport(cod\_transport, localitate, include\_vama, durata) VALUES (3, 'Fetesti', 'nu', 'aprox. 30 min.');

INSERT INTO oprire\_transport(cod\_transport, localitate, include\_vama, durata) VALUES (4, 'Lehliu Gara', 'nu', 'in jur de doua ore');

INSERT INTO oprire\_transport(cod\_transport, localitate, include\_vama, durata) VALUES (4, 'Fetesti', 'nu', 'o ora');

INSERT INTO oprire\_transport(cod\_transport, localitate, include\_vama, durata) VALUES (4, 'Medgidia', 'nu', '2 h - 2h 30');

INSERT INTO oprire\_transport(cod\_transport, localitate, include\_vama, durata) VALUES (5, 'Ramnicu Valcea', 'nu', '2 ore cel mult 3');

INSERT INTO oprire\_transport(cod\_transport, localitate, include\_vama, durata) VALUES (6, 'Alexandria', 'nu', 'in jur de 1h, 1h30');

INSERT INTO livrare(cod\_livrare, cod\_transport, cod\_magazin) VALUES(1, 1, 4);

INSERT INTO livrare(cod\_livrare, cod\_transport, cod\_magazin) VALUES(2, 2, 4);

INSERT INTO livrare(cod\_livrare, cod\_transport, cod\_magazin) VALUES(3, 3, 3);

INSERT INTO livrare(cod\_livrare, cod\_transport, cod\_magazin) VALUES(4, 4, 3);

INSERT INTO livrare(cod\_livrare, cod\_transport, cod\_magazin) VALUES(5, 5, 5);

INSERT INTO livrare(cod\_livrare, cod\_transport, cod\_magazin) VALUES(6, 6, 1);

INSERT INTO livrare(cod\_livrare, cod\_transport, cod\_magazin) VALUES(7, 7, 1);

INSERT INTO livrare(cod\_livrare, cod\_transport, cod\_magazin) VALUES(8, 8, 2);

INSERT INTO livrare(cod\_livrare, cod\_transport, cod\_magazin) VALUES(9, 9, 4);

INSERT INTO livrare(cod\_livrare, cod\_transport, cod\_magazin) VALUES(10, 10, 1);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(1, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', NULL, 4, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(2, 'instrument', 'chitara el. fretless', 4400, NULL, 'artar', 'eBanmaltez', NULL, 4, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(3, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', NULL, 4, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(4, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', NULL, 4, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(5, 'instrument', 'chitara el. DMustang', 1600, NULL, 'stejar', 'DnM Raider', NULL, 4, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(6, 'instrument', 'chitara el. :Haver', 1600, 3.6, 'stejar', 'Hoover', NULL, 4, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(7, 'instrument', 'drum set full iron', 1600, 34, 'otel', 'Jerryhmann', NULL, 4, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(8, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', NULL, 3, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(9, 'instrument', 'drum set full iron', 1600, 34, 'otel', 'Jerryhmann', NULL, 2, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(10, 'instrument', 'chitara el. DMustang', 1600, NULL, 'stejar', 'DnM Raider', NULL, 2, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(11, 'instrument', 'chitara el. JumpingW', 2590, 3, 'frasin', 'Harvey Benington', NULL, 1, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(12, 'instrument', 'chitara el. JumpingW', 2590, 3, 'frasin', 'Harvey Benington', NULL, 1, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(13, 'audio', 'combo chitara 400V', 5000, 22, 'fier', 'GraySun', NULL, 3, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(14, 'inregistrare/audio', 'interfata RedPumpKin', 2800, 1, 'otel', 'Mathers', NULL, 2, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(15, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 4, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(16, 'instrument', 'chitara el. fretless', 4400, NULL, 'artar', 'eBanmaltez', 4, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(17, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 4, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(18, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 4, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(19, 'instrument', 'chitara el. DMustang', 1600, NULL, 'stejar', 'DnM Raider', 4, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(20, 'instrument', 'chitara el. :Haver', 1600, 3.6, 'stejar', 'Hoover', 4, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(30, 'instrument', 'drum set full iron', 1600, 34, 'otel', 'Jerryhmann', 4, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(31, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 3, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(32, 'instrument', 'drum set full iron', 1600, 34, 'otel', 'Jerryhmann', 2, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(33, 'instrument', 'chitara el. DMustang', 1600, NULL, 'stejar', 'DnM Raider', 2, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(34, 'instrument', 'chitara el. JumpingW', 2590, 3, 'frasin', 'Harvey Benington', 1, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(35, 'instrument', 'chitara el. JumpingW', 2590, 3, 'frasin', 'Harvey Benington', 1, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(36, 'audio', 'combo chitara 400V', 5000, 22, 'fier', 'GraySun', 3, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_depozit, cod\_magazin, cod\_lutier, cod\_livrare) VALUES(37, 'inregistrare/audio', 'interfata RedPumpKin', 2800, 1, 'otel', 'Mathers', 2, NULL, NULL, NULL);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(21, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 3);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(22, 'instrument', 'chitara el. fretless', 4400, NULL, 'artar', 'eBanmaltez', 2);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(23, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 1);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(24, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 5);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(25, 'instrument', 'chitara el. DMustang', 1600, NULL, 'stejar', 'DnM Raider', 1);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(26, 'instrument', 'chitara el. :Haver', 1600, 3.6, 'stejar', 'Hoover', 6);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(27, 'instrument', 'drum set full iron', 1600, 34, 'otel', 'Jerryhmann', 2);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(28, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 3);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(29, 'instrument', 'drum set full iron', 1600, 34, 'otel', 'Jerryhmann', 2);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(38, 'instrument', 'chitara el. DMustang', 1600, NULL, 'stejar', 'DnM Raider', 1);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(39, 'instrument', 'chitara el. JumpingW', 2590, 3, 'frasin', 'Harvey Benington', 2);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(40, 'instrument', 'chitara el. JumpingW', 2590, 3, 'frasin', 'Harvey Benington', 3);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(41, 'audio', 'combo chitara 400V', 5000, 22, 'fier', 'GraySun', 6);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_lutier) VALUES(42, 'inregistrare/audio', 'interfata RedPumpKin', 2800, 1, 'otel', 'Mathers', 6);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(43, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 3);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(44, 'instrument', 'chitara el. fretless', 4400, NULL, 'artar', 'eBanmaltez', 2);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(45, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 1);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(46, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 5);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(47, 'instrument', 'chitara el. DMustang', 1600, NULL, 'stejar', 'DnM Raider', 1);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(48, 'instrument', 'chitara el. :Haver', 1600, 3.6, 'stejar', 'Hoover', 6);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(49, 'instrument', 'drum set full iron', 1600, 34, 'otel', 'Jerryhmann', 2);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(50, 'instrument', 'bas acustic', 1600, NULL, 'stejar', 'eBanmaltez', 3);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(51, 'instrument', 'drum set full iron', 1600, 34, 'otel', 'Jerryhmann', 2);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(52, 'instrument', 'chitara el. DMustang', 1600, NULL, 'stejar', 'DnM Raider', 1);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(53, 'instrument', 'chitara el. JumpingW', 2590, 3, 'frasin', 'Harvey Benington', 2);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(54, 'instrument', 'chitara el. JumpingW', 2590, 3, 'frasin', 'Harvey Benington', 3);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(55, 'audio', 'combo chitara 400V', 5000, 22, 'fier', 'GraySun', 6);

INSERT INTO produs(cod\_produs, tip, nume, pret, greutate, material\_predominant, producator, cod\_livrare) VALUES(56, 'inregistrare/audio', 'interfata RedPumpKin', 2800, 1, 'otel', 'Mathers', 6);

COMMIT;

**--partea de pl sql (ex 6-13):**

-- ex 7: Creati o procedură care primeste ca parametru de intrare codul unei livrări si îi setează status\_livrare ca 'sosita',

-- iar pentru fiecare produs care are cod\_livrare egal cu parametrul dat, setează cod\_livrare cu NULL si cod\_magazin

-- cu codul magazinului ce corespunde livrarii in cauza. Se va folosi un cursor pentru a retine produsele ce trebuiesc actualizate.

CREATE OR REPLACE PROCEDURE sosireLivrare(codLivrare IN NUMBER) AS

TYPE tabelCoduriProduse IS TABLE OF NUMBER;

coduriProduse tabelCoduriProduse :=tabelCoduriProduse();

codMagazin NUMBER;

codProdusCrt NUMBER;

CURSOR c IS

SELECT cod\_produs

BULK COLLECT INTO coduriProduse

FROM produs

WHERE cod\_livrare = codLivrare;

BEGIN

IF (codLivrare IS NULL)

THEN

DBMS\_OUTPUT.PUT\_LINE('sosireLivrare: Parametrul dat nu poate fi NULL!');

RETURN;

END IF;

SELECT cod\_magazin

INTO codMagazin

FROM livrare

WHERE cod\_livrare = codLivrare;

UPDATE livrare

SET status\_livrare = 'sosita'

WHERE cod\_livrare = codLivrare;

OPEN c;

LOOP

FETCH c INTO codProdusCrt;

EXIT WHEN c%NOTFOUND;

UPDATE produs

SET cod\_livrare = NULL, cod\_magazin = codMagazin

WHERE cod\_produs = codProdusCrt;

DBMS\_OUTPUT.PUT\_LINE('Produsul cu codul ' || codProdusCrt || ' a fost livrat la magazinul cu codul ' || codMagazin );

END LOOP;

CLOSE c;

DBMS\_OUTPUT.PUT\_LINE('Livrarea cu codul ' || codLivrare || ' s-a incheiat');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('sosireLivrare: Nu exista o livrare cu codul '|| codLivrare || '!');

END;

/

SELECT cod\_produs produs, NVL(cod\_magazin, 0) magazin, NVL(cod\_livrare, 0) livrare, nume FROM produs;

SELECT \* FROM livrare;

EXECUTE sosireLivrare(1);

/

SELECT cod\_produs produs, NVL(cod\_magazin, 0) magazin, NVL(cod\_livrare, 0) livrare, nume FROM produs;

SELECT \* FROM LIVRARE;

ROLLBACK;

/

-- ex 9: Creati o procedura care primeste un pararametru de intrare pentru codul unui produs

-- si are un parametru de iesire pentru numele depozitului in care se afla acesta

-- si doi parametri de iesire pentru numarul de inmatriculare al unui vehicul, respectiv codul unui sofer,

-- ale caror depozite coincid cu cel al depozitului in care se afla produsul.

-- Vehiculul si soferul indicati nu vor avea un transport care sa aiba livrari neincheiate.

-- Daca produsul dat nu exista sau nu are un cod\_depozit nenul sau daca nu se gaseste o pereche disponibila de vehicul si sofer,

-- parametrii de iesire vor fi setati toti cu -1.

CREATE OR REPLACE PROCEDURE soferiVehiculeDisponibile (codProdus IN produs.cod\_produs%TYPE, numeDepozit OUT depozit.nume%TYPE, codVehiculDisponibil OUT vehicul.nr\_inmat%TYPE, codSoferDisponibil OUT sofer.cod\_sofer%TYPE) AS

TYPE perecheVehiculSofer IS RECORD

(nrInmat vehicul.nr\_inmat%TYPE,

codSofer sofer.cod\_sofer%TYPE

);

TYPE tabelPerecheVehiculSofer IS TABLE OF perecheVehiculSofer;

perechiDisponibile tabelPerecheVehiculSofer := tabelPerecheVehiculSofer();

BEGIN

WITH transporturiNeincheiate AS

(SELECT t.cod\_transport codT

FROM transport t

WHERE

(SELECT COUNT(l.cod\_livrare)

FROM livrare l

WHERE l.cod\_transport = t.cod\_transport -- join intre livrare si transport

AND LOWER(l.status\_livrare) NOT LIKE 'sosita' -- livrare neincheiata

) > 0

)

SELECT DISTINCT d.nume, v.nr\_inmat, s.cod\_sofer

INTO numeDepozit, codVehiculDisponibil, codSoferDisponibil

FROM produs p, depozit d, vehicul v, sofer s

WHERE p.cod\_produs = codProdus -- produsul cautat

AND d.cod\_depozit = p.cod\_depozit -- join intre depozit si produs

AND v.cod\_depozit = d.cod\_depozit -- join intre vehicul si depozit

AND s.cod\_depozit = d.cod\_depozit -- join intre sofer si depozit

AND v.nr\_inmat NOT IN

(SELECT v1.nr\_inmat -- vehiculele cu transporturi neincheiate

FROM vehicul v1, transport t

WHERE t.nr\_inmat = v.nr\_inmat -- join intre transport si vehicul

AND t.cod\_transport IN

(SELECT codT -- multimea transporturilior cu livrari neincheiate

FROM transporturiNeincheiate

)

)

AND s.cod\_sofer NOT IN

(SELECT s1.cod\_sofer -- soferii cu transporturi neincheiate

FROM sofer s1, transport t

WHERE t.cod\_sofer = s1.cod\_sofer -- join intre transport si sofer

AND t.cod\_transport IN

(SELECT codT -- multimea transporturilior cu livrari neincheiate

FROM transporturiNeincheiate

)

);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('soferiVehiculeDisponibile: Nu am gasit perechi disponibile de vehicule si soferi pentru produsul cu codul ' || codProdus || ' sau cod\_depozit este NULL sau produsul dat nu exista.' );

numeDepozit := -1;

codVehiculDisponibil := -1;

codSoferDisponibil:= -1;

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('soferiVehiculeDisponibile: S-au gasit mai multe perechi disponibile de vehicule si soferi pentru produsul cu codul ' || codProdus);

WITH transporturiNeincheiate AS

(SELECT t.cod\_transport codT

FROM transport t

WHERE

(SELECT COUNT(l.cod\_livrare)

FROM livrare l

WHERE l.cod\_transport = t.cod\_transport -- join intre livrare si transport

AND LOWER(l.status\_livrare) NOT LIKE 'sosita' -- livrare neincheiata

) > 0

)

SELECT DISTINCT v.nr\_inmat, s.cod\_sofer

BULK COLLECT INTO perechiDisponibile

FROM produs p, depozit d, vehicul v, sofer s

WHERE p.cod\_produs = codProdus -- produsul cautat

AND d.cod\_depozit = p.cod\_depozit -- join intre depozit si produs

AND v.cod\_depozit = d.cod\_depozit -- join intre vehicul si depozit

AND s.cod\_depozit = d.cod\_depozit -- join intre sofer si depozit

AND v.nr\_inmat NOT IN

(SELECT v1.nr\_inmat -- vehiculele cu transporturi neincheiate

FROM vehicul v1, transport t

WHERE t.nr\_inmat = v.nr\_inmat -- join intre transport si vehicul

AND t.cod\_transport IN

(SELECT codT -- multimea transporturilior cu livrari neincheiate

FROM transporturiNeincheiate

)

)

AND s.cod\_sofer NOT IN

(SELECT s1.cod\_sofer -- soferii cu transporturi neincheiate

FROM sofer s1, transport t

WHERE t.cod\_sofer = s1.cod\_sofer -- join intre transport si sofer

AND t.cod\_transport IN

(SELECT codT -- multimea transporturilior cu livrari neincheiate

FROM transporturiNeincheiate

)

);

codVehiculDisponibil := perechiDisponibile(1).nrInmat;

codSoferDisponibil:= perechiDisponibile(1).codSofer;

SELECT d.nume

INTO numeDepozit

FROM depozit d, sofer s

WHERE s.cod\_sofer = codSoferDisponibil -- soferul gasit

AND d.cod\_depozit = s.cod\_depozit; -- join intre depozit si sofer

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('soferiVehiculeDisponibile: Alta exceptie pentru produsul cu codul ' || codProdus);

END;

/

SELECT cod\_produs produs, cod\_depozit depozit FROM produs;

DECLARE

TYPE coduriProduse IS VARRAY(5) OF produs.cod\_produs%TYPE;

varrayCoduriProduse coduriProduse:=coduriProduse();

codV vehicul.nr\_inmat%TYPE;

codS sofer.cod\_sofer%TYPE;

numeD depozit.nume%TYPE;

BEGIN

FOR i IN 1..5 LOOP

varrayCoduriProduse.extend();

END LOOP;

varrayCoduriProduse(1) := 14;

varrayCoduriProduse(2) := 15;

varrayCoduriProduse(3) := 35;

varrayCoduriProduse(4) := 36;

varrayCoduriProduse(5) := 37;

FOR i IN varrayCoduriProduse.FIRST..varrayCoduriProduse.LAST LOOP

soferiVehiculeDisponibile(varrayCoduriProduse(i), numeD, codV, codS);

IF codS >= 0

THEN

DBMS\_OUTPUT.PUT\_LINE('Pentru produsul cu codul ' || varrayCoduriProduse(i) || ': ');

DBMS\_OUTPUT.PUT\_LINE(' nume depozit: ' || numeD);

DBMS\_OUTPUT.PUT\_LINE(' nr inmat vehicul disponibil: ' || codV);

DBMS\_OUTPUT.PUT\_LINE(' cod sofer disponibil: ' || codS);

DBMS\_OUTPUT.NEW\_LINE();

ELSE

DBMS\_OUTPUT.NEW\_LINE();

END IF;

END LOOP;

END;

/

-- ex 6: Creati o procedura care primeste ca parametru de intrare un tabel indexat cu livrari si are parametrii de iesire doua tabele imbricate

-- in care vor fi introduse codurile vehiculelor, respectiv soferilor care se ocupa de transporturile incheiate (ale caror ultime livrari au sosit acum).

-- Modificati si livrarile de actualizat folosind procedura sosireLivrare (de la ex 7).

CREATE OR REPLACE TYPE varrayCoduriLivrari IS VARRAY(100) OF NUMBER(10);

/

CREATE OR REPLACE TYPE tblCoduriVehicule IS TABLE OF VARCHAR2(20);

/

CREATE OR REPLACE TYPE tblCoduriSoferi IS TABLE OF VARCHAR2(10);

/

CREATE OR REPLACE PROCEDURE elibereazaVehiculeSoferi(nrLivrari IN NUMBER, coduriLivrari IN varrayCoduriLivrari, coduriVehicule OUT tblCoduriVehicule, coduriSoferi OUT tblCoduriSoferi) AS

codTransport NUMBER;

BEGIN

FOR i IN coduriLivrari.FIRST..coduriLivrari.LAST LOOP

sosireLivrare(coduriLivrari(i)); -- se incheie livrarea crt

SELECT cod\_transport

INTO codTransport

FROM livrare

WHERE cod\_livrare = coduriLivrari(i);

SELECT DISTINCT v.nr\_inmat

BULK COLLECT INTO coduriVehicule

FROM vehicul v, transport t

WHERE t.cod\_transport = codTransport -- transportul ce corespunde livrarii crt

AND t.nr\_inmat = v.nr\_inmat -- join intre transport si vehicul

AND t.cod\_transport NOT IN -- transport care sa nu fie in mulimea transporturilior cu livrari neincheiate

(SELECT t1.cod\_transport

FROM transport t1

WHERE

(SELECT COUNT(l.cod\_livrare)

FROM livrare l

WHERE l.cod\_transport = t.cod\_transport -- join intre livrare si transport

AND LOWER(l.status\_livrare) NOT LIKE 'sosita' -- livrare neincheiata

) > 0

);

SELECT DISTINCT s.cod\_sofer

BULK COLLECT INTO coduriSoferi

FROM sofer s, transport t

WHERE t.cod\_transport = codTransport -- transportul ce corespunde livrarii crt

AND t.cod\_sofer = s.cod\_sofer -- join intre transport si sofer

AND t.cod\_transport NOT IN -- transport care sa nu fie in mulimea transporturilior cu livrari neincheiate

(SELECT t1.cod\_transport

FROM transport t1

WHERE

(SELECT COUNT(l.cod\_livrare)

FROM livrare l

WHERE l.cod\_transport = t.cod\_transport -- join intre livrare si transport

AND LOWER(l.status\_livrare) NOT LIKE 'sosita' -- livrare neincheiata

) > 0

);

END LOOP;

END;

/

SELECT \* FROM livrare;

SELECT \* FROM transport;

SELECT cod\_produs produs, cod\_livrare livrare, cod\_magazin magazin FROM produs;

DECLARE

coduriLivrari varrayCoduriLivrari := varrayCoduriLivrari();

coduriVehicule tblCoduriVehicule := tblCoduriVehicule();

coduriSoferi tblCoduriSoferi := tblCoduriSoferi();

BEGIN

SELECT cod\_livrare

BULK COLLECT INTO coduriLivrari

FROM livrare

WHERE cod\_livrare IN (1,2,3, 5,6, 7);

elibereazaVehiculeSoferi(4, coduriLivrari, coduriVehicule, coduriSoferi);

DBMS\_OUTPUT.PUT('Nr inmatriculare vehicule eliberate: ');

FOR i IN coduriVehicule.FIRST..coduriVehicule.LAST LOOP

DBMS\_OUTPUT.PUT(coduriVehicule(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT('Coduri soferi eliberati: ');

FOR i IN coduriSoferi.FIRST..coduriSoferi.LAST LOOP

DBMS\_OUTPUT.PUT(coduriSoferi(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

END;

/

SELECT \* FROM livrare;

SELECT cod\_produs produs, cod\_livrare livrare, cod\_magazin magazin FROM produs;

ROLLBACK;

-- ex 8: Creati o functie ce primeste ca parametru de intrare codul unui produs si returneaza codul unui lutier

-- aflat in acelasi judet in care se gaseste magazinul in care sta produsul dat. Daca produsul nu se afla intr-un magazin

-- sau daca in judetul respectiv nu este niciun lutier, functia va returna -1.

CREATE OR REPLACE FUNCTION lutierLocal(codProdus IN NUMBER)

RETURN lutier.cod\_lutier%TYPE AS

TYPE coduriLutieri IS TABLE OF lutier.cod\_lutier%TYPE;

judetLocatie locatie.judet%TYPE;

codLutier lutier.cod\_lutier%TYPE;

tblCoduriLutieri coduriLutieri := coduriLutieri();

BEGIN

SELECT l.judet

INTO judetLocatie

FROM produs p, magazin m, locatie l

WHERE p.cod\_produs = codProdus -- alegem produsul cu codul dat

AND m.cod\_magazin = p.cod\_magazin -- join intre magazin si produs

AND l.cod\_locatie = m.cod\_locatie; -- join intre locatie si magazin

SELECT lut.cod\_lutier

INTO codLutier

FROM lutier lut, locatie loc

WHERE loc.judet = judetLocatie -- judetul gasit

AND lut.cod\_locatie = loc.cod\_locatie; -- join intre lutier si locatie

RETURN codLutier;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('lutierLocal: Atentie! In produs, cod\_magazin nu trebuie sa fie null. Posibil sa nu fie un lutier in judetul magazinului.');

RETURN -1;

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('lutierLocal: Mai multi lutieri sunt la dispozitie. A fost returnat primul gasit.');

SELECT lut.cod\_lutier

BULK COLLECT INTO tblCoduriLutieri

FROM lutier lut, locatie loc

WHERE loc.judet = judetLocatie -- judetul gasit

AND lut.cod\_locatie = loc.cod\_locatie; -- join intre lutier si locatie

codLutier := tblCoduriLutieri(1);

RETURN codLutier;

END lutierLocal;

/

SELECT p.cod\_produs produs, m.cod\_magazin magazin, l.judet judet

FROM produs p, magazin m, locatie l

WHERE p.cod\_magazin IS NOT NULL

AND m.cod\_magazin = p.cod\_magazin

AND l.cod\_locatie = m.cod\_locatie;

SELECT lut.cod\_lutier lutier, loc.judet judet

FROM lutier lut, locatie loc

WHERE loc.cod\_locatie = lut.cod\_locatie;

DECLARE

TYPE coduriLutieri IS VARRAY(4) OF lutier.cod\_lutier%TYPE;

varrayCoduriLutieri coduriLutieri:=coduriLutieri();

codLutierLocal lutier.cod\_lutier%TYPE;

BEGIN

varrayCoduriLutieri.extend();

varrayCoduriLutieri.extend();

varrayCoduriLutieri.extend();

varrayCoduriLutieri.extend();

varrayCoduriLutieri(1):= 10; -- un lutier in judet

varrayCoduriLutieri(2):= 11; -- mai multi lutieri in judet (too many rows)

varrayCoduriLutieri(3):= 13; -- niciun lutier in judet (no data found)

varrayCoduriLutieri(4):= 15; -- cod magazin null (no data found)

FOR i IN varrayCoduriLutieri.FIRST..varrayCoduriLutieri.LAST LOOP

codLutierLocal := lutierLocal(varrayCoduriLutieri(i)); -- vezi pt codurile de produse: 10, 11, 13

IF codLutierLocal > -1

THEN

DBMS\_OUTPUT.PUT\_LINE('Pentru produsul cu codul ' || varrayCoduriLutieri(i) || ', codul unui lutier din judet: ' || codLutierLocal);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Pentru produsul cu codul ' || varrayCoduriLutieri(i) || ', eroare intampinata! Vedeti detalii mai sus');

END IF;

DBMS\_OUTPUT.NEW\_LINE();

END LOOP;

END;

/

-- ex 10: Creati un trigger de tip LMD la nivel de comanda cu executare dupa, pentru update-uri pe coloana status\_livrare a tabelului livrare.

-- Triggerul va calcula pretul maxim al unui produs aflat intr-un magazin si va actualiza

-- preturile tuturor produselor aflate in magazine si avand acel pret maxim, prin reducere cu 5%.

CREATE OR REPLACE TRIGGER ofertaProdus

AFTER UPDATE OF status\_livrare ON livrare

DECLARE

TYPE coduriProduse IS TABLE OF produs.cod\_produs%TYPE;

tblCoduriProduse coduriProduse := coduriProduse();

pretMaxim produs.pret%TYPE;

pretCrt produs.pret%TYPE;

codMagazinCrt produs.cod\_magazin%TYPE;

BEGIN

SELECT MAX(pret)

INTO pretMaxim

FROM produs

WHERE cod\_magazin IS NOT NULL;

SELECT cod\_produs

BULK COLLECT INTO tblCoduriProduse

FROM produs

WHERE pret = pretMaxim

AND cod\_magazin IS NOT NULL;

FOR i IN tblCoduriProduse.FIRST..tblCoduriProduse.LAST LOOP

SELECT pret, cod\_magazin

INTO pretCrt, codMagazinCrt

FROM produs

WHERE cod\_produs = tblCoduriProduse(i);

DBMS\_OUTPUT.PUT\_LINE('Produsul cu codul ' || tblCoduriProduse(i) || ', aflat in magazinul cu codul ' || codMagazinCrt || ', este redus prin oferta de la ' || pretCrt || ' lei la ' || 95/100\*pretCrt || ' lei!');

END LOOP;

FORALL i IN tblCoduriProduse.FIRST..tblCoduriProduse.LAST

UPDATE produs

SET pret = 95/100 \* pret

WHERE cod\_produs = tblCoduriProduse(i);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('ofertaProdus: Eroare no\_data\_found; este posibil sa nu existe niciun produs in magazine.');

END;

/

SELECT cod\_produs produs, NVL(cod\_magazin, 0) magazin, pret, NVL(cod\_livrare, 0) livrare, nume FROM produs;

SELECT \* FROM livrare;

EXECUTE sosireLivrare(2);

/

SELECT cod\_produs produs, NVL(cod\_magazin, 0) magazin, pret, NVL(cod\_livrare, 0) livrare, nume FROM produs;

SELECT \* FROM LIVRARE;

ROLLBACK;

DROP TRIGGER ofertaProdus;

-- ex 11: Creati un trigger de tip LMD la nivel de linie cu executare inainte, pentru update-uri pe coloana status\_livrare a tabelului livrare.

-- Triggerul va adauga intr-un depozit anume (adica va insera in tabelul produs) cate un produs nou pentru fiecare produs din livrarea ce se va actualiza.

-- Produsele noi vor prelua de la cele livrate: tip, nume, pret, greutate, material\_predominant, producator

-- Depozitul se va alege in felul urmator: prioritatea cea mai mare o au depozitele din acelasi judet cu magazinul la care ajunge livrarea,

-- apoi mai au prioritate depozitele cu un numar cat mai mic de produse (calculat inainte de actualizarile triggerului).

SELECT m.cod\_magazin, loc.judet

FROM magazin m, locatie loc

WHERE m.cod\_locatie = loc.cod\_locatie;

SELECT d.cod\_depozit, loc.judet

FROM depozit d, locatie loc

WHERE d.cod\_locatie = loc.cod\_locatie;

CREATE SEQUENCE seqCoduriProduse

START WITH 57

INCREMENT BY 1

MINVALUE 1

MAXVALUE 999999999

CYCLE;

CREATE OR REPLACE TRIGGER completareStocDepozit

BEFORE UPDATE OF status\_livrare ON livrare

FOR EACH ROW

DECLARE

TYPE produseNoi IS TABLE OF produs%ROWTYPE;

TYPE coduriDepozite IS TABLE OF depozit.cod\_depozit%TYPE;

tblProduseNoi produseNoi := produseNoi();

tblCoduriDepozite coduriDepozite := coduriDepozite();

codLivrare livrare.cod\_livrare%TYPE;

codDepozit depozit.cod\_depozit%TYPE;

codMagazin livrare.cod\_magazin%TYPE;

BEGIN

codLivrare := :OLD.cod\_livrare;

codMagazin := :OLD.cod\_magazin;

-- retinem depozitele din acelasi judet cu magazinul la care ajunge livrarea curenta:

SELECT d.cod\_depozit

BULK COLLECT INTO tblCoduriDepozite

FROM locatie loc, depozit d

WHERE loc.judet = -- locatiile aflate in judetul curent

(SELECT loc.judet judetulCrt

FROM magazin m, locatie loc

WHERE m.cod\_magazin = codMagazin -- magazinul la care ajunge livrarea curenta

AND loc.cod\_locatie = m.cod\_locatie) -- join intre locatie si magazin (adica locatia magazinului)

AND d.cod\_locatie = loc.cod\_locatie -- join intre depozit si locatie (adica depozitul de la locatie)

ORDER BY -- depozitele cu mai putine produse au prioritate

(SELECT COUNT(p.cod\_produs)

FROM produs p

WHERE p.cod\_depozit = d.cod\_depozit -- join intre produs si depozit

);

-- daca nu am gasit un depozit in acelasi judet, consideram toate depozitele

IF tblCoduriDepozite.COUNT = 0

THEN

SELECT d.cod\_depozit

BULK COLLECT INTO tblCoduriDepozite

FROM depozit d

ORDER BY -- depozitele cu mai putine produse au prioritate

(SELECT COUNT(p.cod\_produs)

FROM produs p

WHERE p.cod\_depozit = d.cod\_depozit -- join intre produs si depozit

);

END IF;

codDepozit := tblCoduriDepozite(1); -- retinem doar primul depozit (cu prioritatea cea mai mare) din cate avem la dispozitie

SELECT seqCoduriProduse.NEXTVAL, codDepozit, NULL, NULL, NULL, p.tip, p.nume, p.pret, p.greutate, p.material\_predominant, p.producator

BULK COLLECT INTO tblProduseNoi

FROM produs p

WHERE p.cod\_livrare = codLivrare;

FOR i IN tblProduseNoi.FIRST..tblProduseNoi.LAST LOOP

-- inseram in produs o linie cu toate datele din tblProduseNoi(i):

INSERT INTO produs VALUES(tblProduseNoi(i).cod\_produs, tblProduseNoi(i).cod\_depozit, tblProduseNoi(i).cod\_magazin, tblProduseNoi(i).cod\_lutier, tblProduseNoi(i).cod\_livrare,

tblProduseNoi(i).tip, tblProduseNoi(i).nume, tblProduseNoi(i).pret, tblProduseNoi(i).greutate, tblProduseNoi(i).material\_predominant, tblProduseNoi(i).producator);

DBMS\_OUTPUT.PUT\_LINE('In depozitul cu codul ' || tblProduseNoi(i).cod\_depozit || ' a fost adaugat produsul cu codul ' || tblProduseNoi(i).cod\_produs);

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('completareStocDepozit: Eroare no\_data\_found; este posibil sa nu fie produse in livrarea respectiva');

END;

/

SELECT \*

FROM produs;

EXECUTE sosireLivrare(1);

/

SELECT \*

FROM produs;

ROLLBACK;

DROP TRIGGER completareStocDepozit;

DROP SEQUENCE seqCoduriProduse;

-- ex 12: Creati un trigger la nivel de schema pentru drop, cu executare inainte.

-- Triggerul verifica daca obiectul de sters este tabelul produs si apoi, in functie de valoarea unui boolean

-- definit intr-un pachet separat pentru operarea permisiunilor, triggerul fie va permite stergerea tabelului,

-- fie va ridica o eroare difinita de utilizator.

CREATE OR REPLACE PACKAGE pkgPermisiuni AS

permisiuneDropProdus BOOLEAN := TRUE;

PROCEDURE setPermisiuneDropProdus (val IN BOOLEAN);

END pkgPermisiuni;

/

CREATE OR REPLACE PACKAGE BODY pkgPermisiuni AS

PROCEDURE setPermisiuneDropProdus (val IN BOOLEAN) IS

BEGIN

permisiuneDropProdus := val;

IF permisiuneDropProdus

THEN

DBMS\_OUTPUT.PUT\_LINE('permisiuneDropProdus a fost setata la TRUE');

ELSE

DBMS\_OUTPUT.PUT\_LINE('permisiuneDropProdus a fost setata la FALSE');

END IF;

END;

END pkgPermisiuni;

/

CREATE OR REPLACE TRIGGER verificareStergere

BEFORE DROP ON SCHEMA

DECLARE

permisiunea BOOLEAN;

BEGIN

IF LOWER(SYS.DICTIONARY\_OBJ\_NAME) LIKE 'produs'

THEN

permisiunea := pkgPermisiuni.permisiuneDropProdus;

IF permisiunea = FALSE

THEN

DBMS\_OUTPUT.PUT\_LINE('verificareStergere: (' || SYSDATE || ') Nu a fost permisa stergerea tabelului produs.');

RAISE\_APPLICATION\_ERROR(-20998, 'verificareStergere: Nu a fost permisa stergerea tabelului produs! Setati permisiuneDropProdus la TRUE si incercati iar!');

ELSE

DBMS\_OUTPUT.PUT\_LINE('verificareStergere: (' || SYSDATE || ') Tabelulul produs a fost sters cu succes.');

END IF;

END IF;

END;

/

EXECUTE pkgPermisiuni.setPermisiuneDropProdus(FALSE);

/

DROP TABLE produs;

EXECUTE pkgPermisiuni.setPermisiuneDropProdus(TRUE);

/

DROP TABLE produs;

ROLLBACK;

DROP TRIGGER verificareStergere;

-- ex 13: Creati un pachet musicShopExtensii ce va cuprinde toate tipurile, functiile si procedurile create pana acum pentru operarea bazei de date MusicShp.

CREATE OR REPLACE PACKAGE musicShopExtensii AS

-- de la ex 7:

PROCEDURE sosireLivrare(codLivrare IN NUMBER);

-- de la ex 9:

PROCEDURE soferiVehiculeDisponibile (codProdus IN produs.cod\_produs%TYPE, numeDepozit OUT depozit.nume%TYPE, codVehiculDisponibil OUT vehicul.nr\_inmat%TYPE, codSoferDisponibil OUT sofer.cod\_sofer%TYPE);

-- de la ex 6:

TYPE varrayCoduriLivrari IS VARRAY(100) OF NUMBER(10);

TYPE tblCoduriVehicule IS TABLE OF VARCHAR2(20);

TYPE tblCoduriSoferi IS TABLE OF VARCHAR2(10);

PROCEDURE elibereazaVehiculeSoferi(nrLivrari IN NUMBER, coduriLivrari IN musicShopExtensii.varrayCoduriLivrari, coduriVehicule OUT musicShopExtensii.tblCoduriVehicule,

coduriSoferi OUT musicShopExtensii.tblCoduriSoferi);

-- de la ex 8:

FUNCTION lutierLocal(codProdus IN NUMBER)

RETURN lutier.cod\_lutier%TYPE;

END musicShopExtensii;

/

CREATE OR REPLACE PACKAGE BODY musicShopExtensii AS

-- de la ex 7:

PROCEDURE sosireLivrare(codLivrare IN NUMBER) IS

TYPE tabelCoduriProduse IS TABLE OF NUMBER;

coduriProduse tabelCoduriProduse :=tabelCoduriProduse();

codMagazin NUMBER;

codProdusCrt NUMBER;

CURSOR c IS

SELECT cod\_produs

BULK COLLECT INTO coduriProduse

FROM produs

WHERE cod\_livrare = codLivrare;

BEGIN

IF (codLivrare IS NULL)

THEN

DBMS\_OUTPUT.PUT\_LINE('sosireLivrare: Parametrul dat nu poate fi NULL!');

RETURN;

END IF;

SELECT cod\_magazin

INTO codMagazin

FROM livrare

WHERE cod\_livrare = codLivrare;

UPDATE livrare

SET status\_livrare = 'sosita'

WHERE cod\_livrare = codLivrare;

OPEN c;

LOOP

FETCH c INTO codProdusCrt;

EXIT WHEN c%NOTFOUND;

UPDATE produs

SET cod\_livrare = NULL, cod\_magazin = codMagazin

WHERE cod\_produs = codProdusCrt;

DBMS\_OUTPUT.PUT\_LINE('Produsul cu codul ' || codProdusCrt || ' a fost livrat la magazinul cu codul ' || codMagazin );

END LOOP;

CLOSE c;

DBMS\_OUTPUT.PUT\_LINE('Livrarea cu codul ' || codLivrare || ' s-a incheiat');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('sosireLivrare: Nu exista o livrare cu codul '|| codLivrare || '!');

END;

-- de la ex 9:

PROCEDURE soferiVehiculeDisponibile (codProdus IN produs.cod\_produs%TYPE, numeDepozit OUT depozit.nume%TYPE, codVehiculDisponibil OUT vehicul.nr\_inmat%TYPE, codSoferDisponibil OUT sofer.cod\_sofer%TYPE) AS

TYPE perecheVehiculSofer IS RECORD

(nrInmat vehicul.nr\_inmat%TYPE,

codSofer sofer.cod\_sofer%TYPE

);

TYPE tabelPerecheVehiculSofer IS TABLE OF perecheVehiculSofer;

perechiDisponibile tabelPerecheVehiculSofer := tabelPerecheVehiculSofer();

BEGIN

WITH transporturiNeincheiate AS

(SELECT t.cod\_transport codT

FROM transport t

WHERE

(SELECT COUNT(l.cod\_livrare)

FROM livrare l

WHERE l.cod\_transport = t.cod\_transport -- join intre livrare si transport

AND LOWER(l.status\_livrare) NOT LIKE 'sosita' -- livrare neincheiata

) > 0

)

SELECT DISTINCT d.nume, v.nr\_inmat, s.cod\_sofer

INTO numeDepozit, codVehiculDisponibil, codSoferDisponibil

FROM produs p, depozit d, vehicul v, sofer s

WHERE p.cod\_produs = codProdus -- produsul cautat

AND d.cod\_depozit = p.cod\_depozit -- join intre depozit si produs

AND v.cod\_depozit = d.cod\_depozit -- join intre vehicul si depozit

AND s.cod\_depozit = d.cod\_depozit -- join intre sofer si depozit

AND v.nr\_inmat NOT IN

(SELECT v1.nr\_inmat -- vehiculele cu transporturi neincheiate

FROM vehicul v1, transport t

WHERE t.nr\_inmat = v.nr\_inmat -- join intre transport si vehicul

AND t.cod\_transport IN

(SELECT codT -- multimea transporturilior cu livrari neincheiate

FROM transporturiNeincheiate

)

)

AND s.cod\_sofer NOT IN

(SELECT s1.cod\_sofer -- soferii cu transporturi neincheiate

FROM sofer s1, transport t

WHERE t.cod\_sofer = s1.cod\_sofer -- join intre transport si sofer

AND t.cod\_transport IN

(SELECT codT -- multimea transporturilior cu livrari neincheiate

FROM transporturiNeincheiate

)

);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('soferiVehiculeDisponibile: Nu am gasit perechi disponibile de vehicule si soferi pentru produsul cu codul ' || codProdus || ' sau cod\_depozit este NULL sau produsul dat nu exista.' );

numeDepozit := -1;

codVehiculDisponibil := -1;

codSoferDisponibil:= -1;

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('soferiVehiculeDisponibile: S-au gasit mai multe perechi disponibile de vehicule si soferi pentru produsul cu codul ' || codProdus);

WITH transporturiNeincheiate AS

(SELECT t.cod\_transport codT

FROM transport t

WHERE

(SELECT COUNT(l.cod\_livrare)

FROM livrare l

WHERE l.cod\_transport = t.cod\_transport -- join intre livrare si transport

AND LOWER(l.status\_livrare) NOT LIKE 'sosita' -- livrare neincheiata

) > 0

)

SELECT DISTINCT v.nr\_inmat, s.cod\_sofer

BULK COLLECT INTO perechiDisponibile

FROM produs p, depozit d, vehicul v, sofer s

WHERE p.cod\_produs = codProdus -- produsul cautat

AND d.cod\_depozit = p.cod\_depozit -- join intre depozit si produs

AND v.cod\_depozit = d.cod\_depozit -- join intre vehicul si depozit

AND s.cod\_depozit = d.cod\_depozit -- join intre sofer si depozit

AND v.nr\_inmat NOT IN

(SELECT v1.nr\_inmat -- vehiculele cu transporturi neincheiate

FROM vehicul v1, transport t

WHERE t.nr\_inmat = v.nr\_inmat -- join intre transport si vehicul

AND t.cod\_transport IN

(SELECT codT -- multimea transporturilior cu livrari neincheiate

FROM transporturiNeincheiate

)

)

AND s.cod\_sofer NOT IN

(SELECT s1.cod\_sofer -- soferii cu transporturi neincheiate

FROM sofer s1, transport t

WHERE t.cod\_sofer = s1.cod\_sofer -- join intre transport si sofer

AND t.cod\_transport IN

(SELECT codT -- multimea transporturilior cu livrari neincheiate

FROM transporturiNeincheiate

)

);

codVehiculDisponibil := perechiDisponibile(1).nrInmat;

codSoferDisponibil:= perechiDisponibile(1).codSofer;

SELECT d.nume

INTO numeDepozit

FROM depozit d, sofer s

WHERE s.cod\_sofer = codSoferDisponibil -- soferul gasit

AND d.cod\_depozit = s.cod\_depozit; -- join intre depozit si sofer

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('soferiVehiculeDisponibile: Alta exceptie pentru produsul cu codul ' || codProdus);

END;

-- de la ex 6:

PROCEDURE elibereazaVehiculeSoferi(nrLivrari IN NUMBER, coduriLivrari IN musicShopExtensii.varrayCoduriLivrari, coduriVehicule OUT musicShopExtensii.tblCoduriVehicule,

coduriSoferi OUT musicShopExtensii.tblCoduriSoferi) IS

codTransport NUMBER;

BEGIN

FOR i IN coduriLivrari.FIRST..coduriLivrari.LAST LOOP

musicShopExtensii.sosireLivrare(coduriLivrari(i)); -- se incheie livrarea crt

SELECT cod\_transport

INTO codTransport

FROM livrare

WHERE cod\_livrare = coduriLivrari(i);

SELECT DISTINCT v.nr\_inmat

BULK COLLECT INTO coduriVehicule

FROM vehicul v, transport t

WHERE t.cod\_transport = codTransport -- transportul ce corespunde livrarii crt

AND t.nr\_inmat = v.nr\_inmat -- join intre transport si vehicul

AND t.cod\_transport NOT IN -- transport care sa nu fie in mulimea transporturilior cu livrari neincheiate

(SELECT t1.cod\_transport

FROM transport t1

WHERE

(SELECT COUNT(l.cod\_livrare)

FROM livrare l

WHERE l.cod\_transport = t.cod\_transport -- join intre livrare si transport

AND LOWER(l.status\_livrare) NOT LIKE 'sosita' -- livrare neincheiata

) > 0

);

SELECT DISTINCT s.cod\_sofer

BULK COLLECT INTO coduriSoferi

FROM sofer s, transport t

WHERE t.cod\_transport = codTransport -- transportul ce corespunde livrarii crt

AND t.cod\_sofer = s.cod\_sofer -- join intre transport si sofer

AND t.cod\_transport NOT IN -- transport care sa nu fie in mulimea transporturilior cu livrari neincheiate

(SELECT t1.cod\_transport

FROM transport t1

WHERE

(SELECT COUNT(l.cod\_livrare)

FROM livrare l

WHERE l.cod\_transport = t.cod\_transport -- join intre livrare si transport

AND LOWER(l.status\_livrare) NOT LIKE 'sosita' -- livrare neincheiata

) > 0

);

END LOOP;

END;

-- de la ex 8:

FUNCTION lutierLocal(codProdus IN NUMBER)

RETURN lutier.cod\_lutier%TYPE IS lutierul lutier.cod\_lutier%TYPE;

TYPE coduriLutieri IS TABLE OF lutier.cod\_lutier%TYPE;

judetLocatie locatie.judet%TYPE;

codLutier lutier.cod\_lutier%TYPE;

tblCoduriLutieri coduriLutieri := coduriLutieri();

BEGIN

SELECT l.judet

INTO judetLocatie

FROM produs p, magazin m, locatie l

WHERE p.cod\_produs = codProdus -- alegem produsul cu codul dat

AND m.cod\_magazin = p.cod\_magazin -- join intre magazin si produs

AND l.cod\_locatie = m.cod\_locatie; -- join intre locatie si magazin

SELECT lut.cod\_lutier

INTO codLutier

FROM lutier lut, locatie loc

WHERE loc.judet = judetLocatie -- judetul gasit

AND lut.cod\_locatie = loc.cod\_locatie; -- join intre lutier si locatie

RETURN codLutier;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('lutierLocal: Atentie! In produs, cod\_magazin nu trebuie sa fie null. Posibil sa nu fie un lutier in judetul magazinului.');

RETURN -1;

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('lutierLocal: Mai multi lutieri sunt la dispozitie. A fost returnat primul gasit.');

SELECT lut.cod\_lutier

BULK COLLECT INTO tblCoduriLutieri

FROM lutier lut, locatie loc

WHERE loc.judet = judetLocatie -- judetul gasit

AND lut.cod\_locatie = loc.cod\_locatie; -- join intre lutier si locatie

codLutier := tblCoduriLutieri(1);

RETURN codLutier;

END lutierLocal;

END musicShopExtensii;

/

SELECT \* FROM livrare;

SELECT \* FROM transport;

SELECT cod\_produs produs, cod\_livrare livrare, cod\_magazin magazin FROM produs;

DECLARE

coduriLivrari musicShopExtensii.varrayCoduriLivrari := musicShopExtensii.varrayCoduriLivrari();

coduriVehicule musicShopExtensii.tblCoduriVehicule := musicShopExtensii.tblCoduriVehicule();

coduriSoferi musicShopExtensii.tblCoduriSoferi := musicShopExtensii.tblCoduriSoferi();

BEGIN

SELECT cod\_livrare

BULK COLLECT INTO coduriLivrari

FROM livrare

WHERE cod\_livrare IN (1,2,3, 5,6, 7);

musicShopExtensii.elibereazaVehiculeSoferi(4, coduriLivrari, coduriVehicule, coduriSoferi);

DBMS\_OUTPUT.PUT('Nr inmatriculare vehicule eliberate: ');

FOR i IN coduriVehicule.FIRST..coduriVehicule.LAST LOOP

DBMS\_OUTPUT.PUT(coduriVehicule(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT('Coduri soferi eliberati: ');

FOR i IN coduriSoferi.FIRST..coduriSoferi.LAST LOOP

DBMS\_OUTPUT.PUT(coduriSoferi(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

END;

/

SELECT \* FROM livrare;

SELECT cod\_produs produs, cod\_livrare livrare, cod\_magazin magazin FROM produs;

ROLLBACK;

DROP PACKAGE musicShopExtensii;