Music Shop

Документация на курсова работа – УП3-Зимен семестър 2023/24г.

Момчил Милков - 20621535

Борис Луков - 20621544

Стоян Орцев - 20621550

Илия Чакъров - 18621779

# Задание на проекта

Цел:

Моделиране и изграждане на База от Данни, обслужваща система за продажба на музикални продукти.

Описание и изисквания:  
Има три основни обекта: Музикален продукт(Продукт), Продажба и Наличност. Всеки продукт се характеризира с вид, година на издаване, наименование, изпълнител, жанр, музикална компания и единична цена.

Всеки клиент трябва да бъде описан с достатъчно идентифициращи данни, както и данни за контакт. При наличие на чувствителни данни, по желание могат да бъдат криптирани по свободно избран метод. Всеки клиент може да прави неограничен брой покупки. Но една покупка може да е само от един клиент.

Служителите трябва да се характеризират с име, телефон и позиция. Една поръчка(продажба) може да е зачислена само към един служител. Но един служител може да има много продажби.

Продажбите трябва да съдържат информация за клиента, служителя, датата на продажба, броя и вида стоки в нея и крайна цена. При наложимост да се ползват допълнителни таблици.

Да се предвидят роли в базата данни, съответстващи на – служител и администратор(управител).

Нека служителите да имат правата за :

- селектиране на данни от всички таблици

- въвеждане на данни във всички таблици освен в таблици Служители и Наличности.

Но да нямат право да:

- анулират продажба ( при отказ от клиента)

- изтриват данни от всички таблици

Администраторите да имат права да извършват всички горепосочени действия.

Етапи в реализацията:

Изграждане на Entity Relationship Diagram

Нормализиране на модела на БД

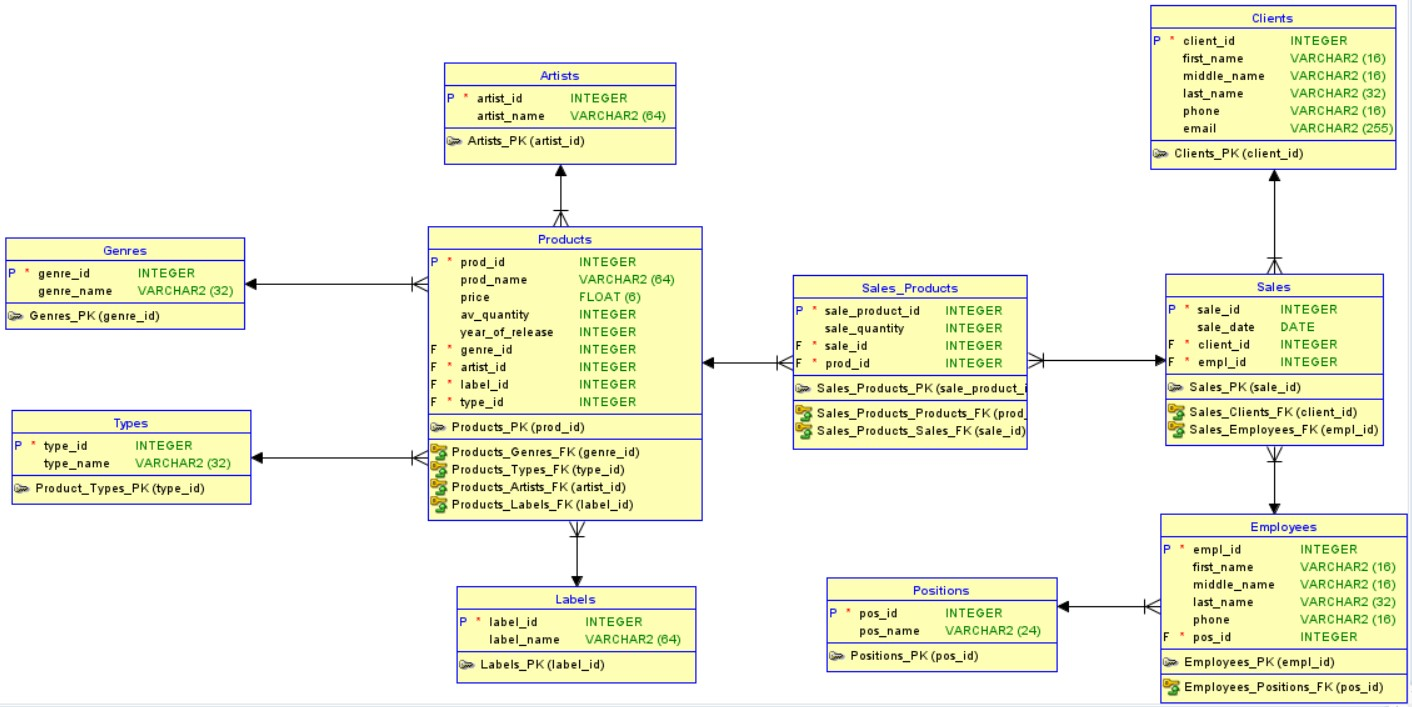
Създаване и манипулиране на таблиците в Базата Данни

Описание на заданието и решението

# Проектиране на БД

Таблици

БД съдържа 10 таблици: Products, Sales\_Products, Sales, Genres, Types, Artists, Labels, Positions, Employees и Clients. За 7 от тях: Artists, Clients, Employees, Labels, Products, Sales\_Products, Sales се съдържат Insert, Update Procedures и Sequences, които се грижат за автоматичната инкрементация на ID(PK) стойностите на всеки ред от таблицата.. Релационният модел на БД изглежда така:



Тригери и Sequence-и

За всяка една от таблиците съществува “Before Insert” тригер и sequence (Нека за пример да вземем таблица Artists: Тригер- AR\_TR\_AI, Sequence- ARTIST\_SEQ; Синтаксисът за останалите таблици е аналогичен.), които се грижат за автоматичната инкрементация на ID(PK) стойностите на всеки ред от таблицата.

Процедури

За всяка една от таблиците съществуват процедури за Записване и Обновяване на данни, които приемат от потребителя данните за реда, който той иска да въведе, или обнови (Пример: ART\_INS, ART \_UPD). Също така, за всяка една от справките, предвидени в проекта, присъстват процедури, приемащи като входни данни критерият, по който ще се извършва избраната справка: FIND\_CLIENT\_PRODUCTS, FIND\_EMPLOYEE\_SALES, FIND\_LATEST\_SALES, FIND\_SALE\_PRODUTCS, RETURN\_SALE.

# Реализация

SQL команди – DDL, DML

Create Tables:

--------------------------------------------------------

-- DDL for Table Creation: artists

--------------------------------------------------------

CREATE TABLE artists (

artist\_id INTEGER NOT NULL,

artist\_name VARCHAR2(64)

);

ALTER TABLE artists ADD CONSTRAINT artists\_pk PRIMARY KEY ( artist\_id );

--------------------------------------------------------

-- DDL for Table Creation: clients

--------------------------------------------------------

CREATE TABLE clients (

client\_id INTEGER NOT NULL,

first\_name VARCHAR2(16),

middle\_name VARCHAR2(16),

last\_name VARCHAR2(32),

phone VARCHAR2(16),

email VARCHAR2(255)

);

ALTER TABLE clients ADD CONSTRAINT clients\_pk PRIMARY KEY ( client\_id );

--------------------------------------------------------

-- DDL for Table Creation: employees

--------------------------------------------------------

CREATE TABLE employees (

empl\_id INTEGER NOT NULL,

first\_name VARCHAR2(16),

middle\_name VARCHAR2(16),

last\_name VARCHAR2(32),

phone VARCHAR2(16),

pos\_id INTEGER NOT NULL

);

ALTER TABLE employees ADD CONSTRAINT employees\_pk PRIMARY KEY ( empl\_id );

--------------------------------------------------------

-- DDL for Table Creation: genres

--------------------------------------------------------

CREATE TABLE genres (

genre\_id INTEGER NOT NULL,

genre\_name VARCHAR2(32)

);

ALTER TABLE genres ADD CONSTRAINT genres\_pk PRIMARY KEY ( genre\_id );

--------------------------------------------------------

-- DDL for Table Creation: labels

--------------------------------------------------------

CREATE TABLE labels (

label\_id INTEGER NOT NULL,

label\_name VARCHAR2(64)

);

ALTER TABLE labels ADD CONSTRAINT labels\_pk PRIMARY KEY ( label\_id );

--------------------------------------------------------

-- DDL for Table Creation: positions

--------------------------------------------------------

CREATE TABLE positions (

pos\_id INTEGER NOT NULL,

pos\_name VARCHAR2(24)

);

ALTER TABLE positions ADD CONSTRAINT positions\_pk PRIMARY KEY ( pos\_id );

--------------------------------------------------------

-- DDL for Table Creation: products

--------------------------------------------------------

CREATE TABLE products (

prod\_id INTEGER NOT NULL,

prod\_name VARCHAR2(64),

price FLOAT(12),

av\_quantity INTEGER,

year\_of\_release INTEGER,

genre\_id INTEGER NOT NULL,

artist\_id INTEGER NOT NULL,

label\_id INTEGER NOT NULL,

type\_id INTEGER NOT NULL

);

ALTER TABLE products ADD CONSTRAINT products\_pk PRIMARY KEY ( prod\_id );

--------------------------------------------------------

-- DDL for Table Creation: sales

--------------------------------------------------------

CREATE TABLE sales (

sale\_id INTEGER NOT NULL,

sale\_date DATE,

client\_id INTEGER NOT NULL,

empl\_id INTEGER NOT NULL

);

ALTER TABLE sales ADD CONSTRAINT sales\_pk PRIMARY KEY ( sale\_id );

--------------------------------------------------------

-- DDL for Table Creation: sales\_products

--------------------------------------------------------

CREATE TABLE sales\_products (

sale\_prod\_id INTEGER NOT NULL,

sale\_quantity INTEGER,

sale\_id INTEGER NOT NULL,

prod\_id INTEGER NOT NULL

);

ALTER TABLE sales\_products ADD CONSTRAINT sales\_products\_pk PRIMARY KEY ( sale\_prod\_id );

--------------------------------------------------------

-- DDL for Table Creation: types

--------------------------------------------------------

CREATE TABLE types (

type\_id INTEGER NOT NULL,

type\_name VARCHAR2(32)

);

ALTER TABLE types ADD CONSTRAINT product\_types\_pk PRIMARY KEY ( type\_id );

--------------------------------------------------------

-- DDL for Foreign Key/s Definition in Table: employees

--------------------------------------------------------

ALTER TABLE employees

ADD CONSTRAINT employees\_positions\_fk FOREIGN KEY ( pos\_id )

REFERENCES positions ( pos\_id );

--------------------------------------------------------

-- DDL for Foreign Key/s Definition in Table: products

--------------------------------------------------------

ALTER TABLE products

ADD CONSTRAINT products\_artists\_fk FOREIGN KEY ( artist\_id )

REFERENCES artists ( artist\_id );

ALTER TABLE products

ADD CONSTRAINT products\_genres\_fk FOREIGN KEY ( genre\_id )

REFERENCES genres ( genre\_id );

ALTER TABLE products

ADD CONSTRAINT products\_labels\_fk FOREIGN KEY ( label\_id )

REFERENCES labels ( label\_id );

ALTER TABLE products

ADD CONSTRAINT products\_types\_fk FOREIGN KEY ( type\_id )

REFERENCES types ( type\_id );

--------------------------------------------------------

-- DDL for Foreign Key/s Definition in Table: sales

--------------------------------------------------------

ALTER TABLE sales

ADD CONSTRAINT sales\_clients\_fk FOREIGN KEY ( client\_id )

REFERENCES clients ( client\_id );

ALTER TABLE sales

ADD CONSTRAINT sales\_employees\_fk FOREIGN KEY ( empl\_id )

REFERENCES employees ( empl\_id );

--------------------------------------------------------

-- DDL for Foreign Key/s Definition in Table: sales\_products

--------------------------------------------------------

ALTER TABLE sales\_products

ADD CONSTRAINT sales\_products\_products\_fk FOREIGN KEY ( prod\_id )

REFERENCES products ( prod\_id );

ALTER TABLE sales\_products

ADD CONSTRAINT sales\_products\_sales\_fk FOREIGN KEY ( sale\_id )

REFERENCES sales ( sale\_id );

Insert

Важно е преди да се правят тези Insert-и да има имплементирани всички procedures, triggers & sequences, както и Insert-ите да бъдат изпълнени в дадената поредност.

-- positions

INSERT INTO positions values (1,'default');

INSERT INTO positions values (2,'administrator');

-- types

INSERT INTO types values (1,'CD');

INSERT INTO types values (2,'digital');

INSERT INTO types values (3,'vinyl');

-- genres

INSERT INTO genres values (1,'rock');

INSERT INTO genres values (2,'pop');

INSERT INTO genres values (3,'hip-hop');

INSERT INTO genres values (4,'jazz');

INSERT INTO genres values (5,'country');

INSERT INTO genres values (6,'classical');

INSERT INTO genres values (7,'techno');

INSERT INTO genres values (8,'blues');

INSERT INTO genres values (9,'metal');

-- clients

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Bozhidar','Dimitrov','Vitanov','+359-988-5555-45','Bozhidar.Dimitrov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Nedelcho','Petrov','Velikov','+359-485-5533-12','Nedelcho.Petrov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Tzako','Ivanov','Zhivkov','+359-885-5511-52','Tzako.Ivanov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Atanas','Georgiev','Ivanov','+359-885-5574-99','Atanas.Georgiev@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Naso','Vasilev','Alexandrov','+359-988-5551-39','Naso.Vasilev@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Kupen','Nikolov','Todorov','+359-988-5556-96','Kupen.Nikolov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Radomir','Kirov','Popov','+359-485-5510-35','Radomir.Kirov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Radul','Stoyanov','Bunev','+359-485-5593-20','Radul.Stoyanov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Kaloyan','Marinov','Dachev','+359-895-5523-74','Kaloyan.Marinov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Kiril','Todorov','Iliev','+359-988-5558-10','Kiril.Todorov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Hristo','Hristov','Yankov','+359-204-3170-91','Hristo.Hristov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Grudi','Yordanov','Savov','+359-259-5903-56','Grudi.Yordanov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Boyko','Simeonov','Kirilov','+359-248-7280-94','Boyko.Simeonov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Manush','Yankov','Pironev','+359-222-9895-48','Manush.Yankov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Lyobomir','Stanimirov','Rysinov','+359-209-5098-32','Lyobomir.Stanimirov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Anghel','Zahariev','Tsankov','+359-272-2321-99','Anghel.Zahariev@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Zitko','Borisov','Vanko','+359-208-9905-97','Zitko.Borisov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Ivan','Atanasov','Dinev','+359-278-0877-13','Ivan.Atanasov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Angel','Velikov','Petkov','+359-245-6916-34','Angel.Velikov@gmail.com');

Insert into CLIENTS (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,EMAIL) values ('Nedyalko','Bogdanov','Venev','+359-289-2100-25','Nedyalko.Bogdanov@gmail.com');

-- employees

Insert into EMPLOYEES (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,POS\_ID) values ('Lilyan','Dimitrov','Dobrev','+359-485-5564-00',2);

Insert into EMPLOYEES (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,POS\_ID) values ('Gregor','Vasilev','Kostov','+359-875-5596-65',2);

Insert into EMPLOYEES (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,POS\_ID) values ('Lubomir','Petrov','Nakov','+359-485-5580-06',2);

Insert into EMPLOYEES (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,POS\_ID) values ('Grudi','Yordanov','Romanov','+359-988-5552-38',2);

Insert into EMPLOYEES (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,POS\_ID) values ('Grudi','Stoyanov','Atanasov','+359-875-5518-01',2);

Insert into EMPLOYEES (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,POS\_ID) values ('Lilyan','Marinov','Dimitrov','+359-988-5552-26',2);

Insert into EMPLOYEES (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,POS\_ID) values ('Yakov','Ivanov','Romanov','+359-895-5594-68',2);

Insert into EMPLOYEES (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,POS\_ID) values ('Radko','Todorov','Lukanov','+359-988-5557-20',2);

Insert into EMPLOYEES (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,POS\_ID) values ('Nikolai','Nikolov','Manev','+359-895-5597-44',1);

Insert into EMPLOYEES (FIRST\_NAME,MIDDLE\_NAME,LAST\_NAME,PHONE,POS\_ID) values ('Ludmil','Georgiev','Bachev','+359-988-5559-85',1);

-- labels

Insert into LABELS (label\_name) values ('Virginia Records');

Insert into LABELS (label\_name) values ('Amazon Studios');

Insert into LABELS (label\_name) values ('Studentska 14 Productions');

Insert into LABELS (label\_name) values ('MVP');

Insert into LABELS (label\_name) values ('jojo`s');

-- artists

Insert into ARTISTS (artist\_name) values ('Orlin Goranov');

Insert into ARTISTS (artist\_name) values ('Drake');

Insert into ARTISTS (artist\_name) values ('Katy Perry');

Insert into ARTISTS (artist\_name) values ('Santa Claus');

Insert into ARTISTS (artist\_name) values ('Boga42');

Insert into ARTISTS (artist\_name) values ('MC Momi');

Insert into ARTISTS (artist\_name) values ('Dirigenta');

Insert into ARTISTS (artist\_name) values ('Rado Rumenov');

Insert into ARTISTS (artist\_name) values ('Toshko Kukata');

Insert into ARTISTS (artist\_name) values ('Ilian40');

-- products

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Hakuna Matata',20.00,100,1998,1,1,1,1);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Love is in the Air',22.00,100,2001,2,2,2,1);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('I Love',23.00,100,2013,3,3,3,1);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Matraka',25.00,100,2022,4,4,4,2);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Carolina',31.50,100,2023,5,5,5,2);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Fly',15.50,100,1999,6,6,1,2);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Do It',11.50,100,1990,7,7,2,3);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Make Me',15.50,100,2019,8,8,3,3);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('When Did This Happen?',10.99,100,2018,9,9,4,3);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('X Files',21.99,100,2022,1,10,5,1);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Zuzi',22.99,100,2022,2,1,1,1);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Palm Beach',6.99,100,1998,3,2,2,2);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Make it Rain',7.40,100,2003,4,3,3,2);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Mr. Producer',2.40,100,2010,5,4,4,3);

Insert into PRODUCTS (PROD\_NAME,PRICE,AV\_QUANTITY,YEAR\_OF\_RELEASE,GENRE\_ID,ARTIST\_ID,LABEL\_ID,TYPE\_ID) values ('Malka Doza Levski',50.40,100,2011,6,5,5,3);

-- sales

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('13-APR-21','DD-MON-RR'),1,1);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('10-JAN-22','DD-MON-RR'),2,2);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('12-AUG-22','DD-MON-RR'),3,3);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('20-AUG-16','DD-MON-RR'),4,5);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('31-JUL-17','DD-MON-RR'),5,6);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('17-JUL-20','DD-MON-RR'),6,8);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('22-JAN-21','DD-MON-RR'),7,3);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('13-JUN-13','DD-MON-RR'),8,1);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('26-SEP-13','DD-MON-RR'),9,9);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('13-MAR-14','DD-MON-RR'),10,10);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('21-MAY-14','DD-MON-RR'),11,10);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('30-AUG-14','DD-MON-RR'),12,7);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('24-MAR-15','DD-MON-RR'),13,1);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('06-MAR-16','DD-MON-RR'),14,10);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('28-SEP-16','DD-MON-RR'),15,5);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('11-MAY-17','DD-MON-RR'),16,6);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('28-OCT-17','DD-MON-RR'),17,8);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('09-AUG-18','DD-MON-RR'),18,10);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('23-APR-19','DD-MON-RR'),19,9);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('12-JUN-21','DD-MON-RR'),20,5);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('26-SEP-13','DD-MON-RR'),1,3);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('13-MAR-14','DD-MON-RR'),2,3);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('21-MAY-19','DD-MON-RR'),3,4);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('30-AUG-14','DD-MON-RR'),4,1);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('24-MAR-15','DD-MON-RR'),5,7);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('06-MAR-16','DD-MON-RR'),6,7);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('28-SEP-16','DD-MON-RR'),7,4);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('11-MAY-17','DD-MON-RR'),8,8);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('28-OCT-17','DD-MON-RR'),9,8);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('09-AUG-18','DD-MON-RR'),10,9);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('13-APR-22','DD-MON-RR'),1,1);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('10-JAN-12','DD-MON-RR'),2,2);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('12-AUG-18','DD-MON-RR'),3,3);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('20-AUG-20','DD-MON-RR'),4,5);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('31-JUL-03','DD-MON-RR'),5,6);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('17-JUL-09','DD-MON-RR'),6,8);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('22-JAN-10','DD-MON-RR'),7,3);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('13-JUN-18','DD-MON-RR'),8,1);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('26-SEP-19','DD-MON-RR'),9,9);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('13-MAR-17','DD-MON-RR'),10,10);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('21-MAY-08','DD-MON-RR'),11,10);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('30-AUG-08','DD-MON-RR'),12,7);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('24-MAR-11','DD-MON-RR'),13,1);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('06-MAR-12','DD-MON-RR'),14,10);

Insert into SALES (SALE\_DATE,CLIENT\_ID,EMPL\_ID) values (to\_date('28-SEP-15','DD-MON-RR'),15,5);

-- sales\_products

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,1,1);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,2,2);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,3,3);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (2,4,4);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,5,5);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,6,6);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,7,7);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (3,8,8);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,9,9);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,10,10);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,11,11);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,12,12);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,13,13);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,14,14);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,15,15);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,16,1);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,17,2);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,18,3);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,19,4);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (2,20,5);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,21,6);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,22,7);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,23,8);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,24,9);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,25,10);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,26,11);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,27,12);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,28,13);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,29,14);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,30,15);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,31,1);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,32,2);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,33,3);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,34,4);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,35,5);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,36,6);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,36,7);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (5,38,8);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,39,9);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,40,10);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,41,11);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,42,12);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,43,13);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,44,14);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,45,15);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,15,1);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,14,2);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (4,13,3);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,12,4);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,11,5);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (3,10,6);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,9,7);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,8,8);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,7,9);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,6,10);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,5,11);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,4,12);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (5,3,13);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,2,14);

Insert into SALES\_PRODUCTS (SALE\_QUANTITY,SALE\_ID,PROD\_ID) values (1,1,15);

PL/SQL процедури/тригери/курсори

Тригери: Auto Increment

Да се изпълняват по отделно:

--------------------------------------------------------

-- DDL for Trigger AR\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE EDITIONABLE TRIGGER "ADMIN"."AR\_TR\_AI" -- ARtist\_TRigger\_AutoIncrement

before insert on artists

for each row

WHEN (NEW.artist\_id is null) begin

:NEW.artist\_id := ARTIST\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "ADMIN"."AR\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger CL\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE EDITIONABLE TRIGGER "ADMIN"."CL\_TR\_AI"

before insert on clients

for each row

WHEN (NEW.client\_id is null) begin

:NEW.client\_id := CLIENT\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "ADMIN"."CL\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger EM\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE EDITIONABLE TRIGGER "ADMIN"."EM\_TR\_AI"

before insert on employees

for each row

WHEN (NEW.empl\_id is null) begin

:NEW.empl\_id := EMPLOYEE\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "ADMIN"."EM\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger LA\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE EDITIONABLE TRIGGER "ADMIN"."LA\_TR\_AI"

before insert on labels

for each row

WHEN (NEW.label\_id is null) begin

:NEW.label\_id := LABEL\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "ADMIN"."LA\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger PR\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE EDITIONABLE TRIGGER "ADMIN"."PR\_TR\_AI"

before insert on products

for each row

WHEN (NEW.prod\_id is null) begin

:NEW.prod\_id := PRODUCT\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "ADMIN"."PR\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger SA\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE EDITIONABLE TRIGGER "ADMIN"."SA\_TR\_AI"

before insert on sales

for each row

WHEN (NEW.sale\_id is null) begin

:NEW.sale\_id := SALE\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "ADMIN"."SA\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger SAPR\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE EDITIONABLE TRIGGER "ADMIN"."SAPR\_TR\_AI"

before insert on sales\_products

for each row

WHEN (NEW.sale\_prod\_id is null) begin

:NEW.sale\_prod\_id := SALE\_PRODUCT\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "ADMIN"."SAPR\_TR\_AI" ENABLE;

Sequences

CREATE SEQUENCE "ARTIST\_SEQ" MINVALUE 1 MAXVALUE

9999999999999999999999999999 INCREMENT BY 1 START WITH 1

NOCACHE ORDER NOCYCLE ;

CREATE SEQUENCE "CLIENT\_SEQ" MINVALUE 1 MAXVALUE

9999999999999999999999999999 INCREMENT BY 1 START WITH 1

NOCACHE ORDER NOCYCLE ;

CREATE SEQUENCE "EMPLOYEE\_SEQ" MINVALUE 1 MAXVALUE

9999999999999999999999999999 INCREMENT BY 1 START WITH 1

NOCACHE ORDER NOCYCLE ;

CREATE SEQUENCE "LABEL\_SEQ" MINVALUE 1 MAXVALUE

9999999999999999999999999999 INCREMENT BY 1 START WITH 1

NOCACHE ORDER NOCYCLE ;

CREATE SEQUENCE "PRODUCT\_SEQ" MINVALUE 1 MAXVALUE

9999999999999999999999999999 INCREMENT BY 1 START WITH 1

NOCACHE ORDER NOCYCLE ;

CREATE SEQUENCE "SALE\_SEQ" MINVALUE 1 MAXVALUE

9999999999999999999999999999 INCREMENT BY 1 START WITH 1

NOCACHE ORDER NOCYCLE ;

CREATE SEQUENCE "SALE\_PRODUCT\_SEQ" MINVALUE 1 MAXVALUE

9999999999999999999999999999 INCREMENT BY 1 START WITH 1

NOCACHE ORDER NOCYCLE ;

Insert процедури (с примерни използвания)

--------------------------------------------------------

-- DDL for Procedure ART\_INS

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "ART\_INS" -- ARTist\_INSert

(artist\_name artists.artist\_name%type)

as

begin

insert into

artists(artist\_name)

values(artist\_name);

end ART\_INS;

/

--------------------------------------------------------

-- DDL for Procedure CLI\_INS

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "CLI\_INS"

(first\_name clients.first\_name%type,

middle\_name clients.middle\_name%type,

last\_name clients.last\_name%type,

phone clients.phone%type,

email clients.email%type)

as

begin

insert into

clients(first\_name,middle\_name,last\_name,phone,email)

values(first\_name,middle\_name,last\_name,phone,email);

end CLI\_INS;

/

--------------------------------------------------------

-- DDL for Procedure EMP\_INS

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "EMP\_INS"

(first\_name employees.first\_name%type,

middle\_name employees.middle\_name%type,

last\_name employees.last\_name%type,

phone employees.phone%type,

pos\_id employees.pos\_id%type)

as

begin

insert into

employees(first\_name,middle\_name,last\_name,phone,pos\_id)

values(first\_name,middle\_name,last\_name,phone,pos\_id);

end EMP\_INS;

/

-- exec emp\_ins('fname\_test','mname\_test','lname\_test','phone\_test',1); примерно използване

--------------------------------------------------------

-- DDL for Procedure LAB\_INS

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "LAB\_INS"

(label\_name labels.label\_name%type)

as

begin

insert into

labels(label\_name)

values(label\_name);

end LAB\_INS;

/

--------------------------------------------------------

-- DDL for Procedure PRO\_INS

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "PRO\_INS"

(prod\_name products.prod\_name%type,

price products.price%type,

av\_quantity products.av\_quantity%type,

year\_of\_release products.year\_of\_release%type,

genre\_id products.genre\_id%type,

artist\_id products.artist\_id%type,

label\_id products.label\_id%type,

type\_id products.type\_id%type)

as

begin

insert into

products(prod\_name,price,av\_quantity,year\_of\_release,genre\_id,artist\_id,label\_id,type\_id)

values(prod\_name,price,av\_quantity,year\_of\_release,genre\_id,artist\_id,label\_id,type\_id);

end PRO\_INS;

/

--------------------------------------------------------

-- DDL for Procedure SAL\_INS

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "SAL\_INS"

(sale\_date sales.sale\_date%type,

client\_id sales.client\_id%type,

empl\_id sales.empl\_id%type)

as

begin

insert into

sales(sale\_date,client\_id,empl\_id)

values(sale\_date,client\_id,empl\_id);

end SAL\_INS;

/

-- exec sal\_ins(TO\_DATE('17/12/2015', 'DD/MM/YYYY'),1,1); примерно използване

--------------------------------------------------------

-- DDL for Procedure SAL\_PRO\_INS

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "SAL\_PRO\_INS"

(sale\_quantity sales\_products.sale\_quantity%type,

sale\_id sales\_products.sale\_id%type,

product\_id sales\_products.prod\_id%type)

as

available\_quantity products.av\_quantity%type;

begin

SELECT av\_quantity

INTO available\_quantity

FROM products

WHERE prod\_id = product\_id;

-- Проверяваме, дали има достатъчно продукт.

if available\_quantity >= sale\_quantity then

UPDATE products

SET av\_quantity = av\_quantity - sale\_quantity

WHERE prod\_id = product\_id;

INSERT INTO sales\_products(sale\_quantity, sale\_id, prod\_id)

VALUES (sale\_quantity, sale\_id, product\_id);

else

DBMS\_OUTPUT.PUT\_LINE('Insufficient available quantity. Sale not processed.');

end if;

end SAL\_PRO\_INS;

/

Update процедури

--------------------------------------------------------

-- DDL for Procedure ART\_UPD

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "ART\_UPD" -- ARTist\_UPDate

(ID integer,

n\_artist\_name artists.artist\_name%type)

as

begin

update artists

set artist\_name = n\_artist\_name

where artist\_id = ID;

end ART\_UPD;

/

--------------------------------------------------------

-- DDL for Procedure CLI\_UPD

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "CLI\_UPD"

(ID integer,

n\_first\_name clients.first\_name%type,

n\_middle\_name clients.middle\_name%type,

n\_last\_name clients.last\_name%type,

n\_phone clients.phone%type,

n\_email clients.email%type)

as

begin

update clients

set first\_name = n\_first\_name,

middle\_name = n\_middle\_name,

last\_name = n\_last\_name,

phone = n\_phone,

email = n\_email

where client\_id = ID;

end CLI\_UPD;

/

--------------------------------------------------------

-- DDL for Procedure EMP\_UPD

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "EMP\_UPD"

(ID integer,

n\_first\_name employees.first\_name%type,

n\_middle\_name employees.middle\_name%type,

n\_last\_name employees.last\_name%type,

n\_phone employees.phone%type,

n\_pos\_id employees.pos\_id%type)

as

begin

update employees

set first\_name = n\_first\_name,

middle\_name = n\_middle\_name,

last\_name = n\_last\_name,

phone = n\_phone,

pos\_id = n\_pos\_id

where empl\_id = ID;

end EMP\_UPD;

/

--------------------------------------------------------

-- DDL for Procedure LAB\_UPD

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "LAB\_UPD"

(ID integer,

n\_label\_name labels.label\_name%type)

as

begin

update labels

set label\_name = n\_label\_name

where label\_id = ID;

end LAB\_UPD;

/

--------------------------------------------------------

-- DDL for Procedure PRO\_UPD

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "PRO\_UPD"

(ID integer,

n\_prod\_name products.prod\_name%type,

n\_price products.price%type,

n\_av\_quantity products.av\_quantity%type,

n\_year\_of\_release products.year\_of\_release%type,

n\_genre\_id products.genre\_id%type,

n\_artist\_id products.artist\_id%type,

n\_label\_id products.label\_id%type,

n\_type\_id products.type\_id%type)

as

begin

update products

set prod\_name = n\_prod\_name,

price = n\_price,

av\_quantity = n\_av\_quantity,

year\_of\_release = n\_year\_of\_release,

genre\_id = n\_genre\_id,

artist\_id = n\_artist\_id,

label\_id = n\_label\_id,

type\_id = n\_type\_id

where prod\_id = ID;

end PRO\_UPD;

/

--------------------------------------------------------

-- DDL for Procedure SAL\_PRO\_UPD

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "SAL\_PRO\_UPD"

(ID integer,

n\_sale\_quantity sales\_products.sale\_quantity%type,

n\_sale\_id sales\_products.sale\_id%type,

n\_prod\_id sales\_products.prod\_id%type)

as

begin

update sales\_products

set sale\_quantity = n\_sale\_quantity,

sale\_id = n\_sale\_id,

prod\_id = n\_prod\_id

where sale\_prod\_id = ID;

end SAL\_PRO\_UPD;

/

--------------------------------------------------------

-- DDL for Procedure SAL\_UPD

--------------------------------------------------------

set define off;

CREATE OR REPLACE PROCEDURE "SAL\_UPD"

(ID integer,

n\_sale\_date sales.sale\_date%type,

n\_client\_id sales.client\_id%type,

n\_empl\_id sales.empl\_id%type)

as

begin

update sales

set sale\_date = n\_sale\_date,

client\_id = n\_client\_id,

empl\_id = n\_empl\_id

where sale\_id = ID;

end SAL\_UPD;

/

Select(Query) Процедури:

Резултатите от справките се извеждат в dbms конзолата

--------------------------------------------------------

-- Да се направи справка за брой продажби на служител, подредени по дата.

--------------------------------------------------------

create or replace PROCEDURE FIND\_EMPLOYEE\_SALES (

firstName EMPLOYEES.FIRST\_NAME%TYPE,

lastName EMPLOYEES.LAST\_NAME%TYPE

) IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Sales of employee ' || firstName || ' ' || lastName || ', ordered by date:'||CHR(10));

DBMS\_OUTPUT.PUT\_LINE(

RPAD('First Name', 16) || ' |' ||

RPAD('Last Name', 16) || ' |' ||

RPAD('SaleID', 6) || ' |' ||

RPAD('Date', 4)

);

FOR cust\_record IN (

SELECT e.FIRST\_NAME, e.LAST\_NAME, s.SALE\_ID, s.SALE\_DATE

FROM SALES s

JOIN EMPLOYEES e ON s.EMPL\_ID = e.EMPL\_ID

WHERE e.FIRST\_NAME LIKE firstName AND e.LAST\_NAME LIKE lastName

ORDER BY s.SALE\_DATE DESC

)

LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(cust\_record.FIRST\_NAME, 16) || ' |' ||

RPAD(cust\_record.LAST\_NAME, 16) || ' |' ||

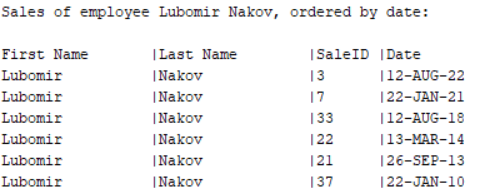
RPAD(cust\_record.SALE\_ID, 6) || ' |' ||

TO\_CHAR(cust\_record.SALE\_DATE)

);

END LOOP;

END FIND\_EMPLOYEE\_SALES;



--------------------------------------------------------

--Да се направи справка за последните 5 продажби на стоки, подредени по служител.

--------------------------------------------------------

create or replace PROCEDURE FIND\_LATEST\_SALES

IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Latest 5 Sales, ordered by employee:'||CHR(10));

DBMS\_OUTPUT.PUT\_LINE(

RPAD('SaleID', 6) || ' |' ||

RPAD('First Name', 16) || ' |' ||

RPAD('Last Name', 16) || ' |' ||

RPAD('Date', 4)

);

FOR cust\_record IN (

SELECT s.SALE\_ID, e.FIRST\_NAME, e.LAST\_NAME, s.SALE\_DATE, e.EMPL\_ID

FROM SALES s

JOIN EMPLOYEES e ON s.EMPL\_ID = e.EMPL\_ID

JOIN SALES\_PRODUCTS sp ON s.SALE\_ID = sp.SALE\_ID

ORDER BY s.SALE\_DATE DESC, e.EMPL\_ID

FETCH FIRST 5 ROWS ONLY

)

LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(cust\_record.SALE\_ID, 6) || ' |' ||

RPAD(cust\_record.FIRST\_NAME, 16) || ' |' ||

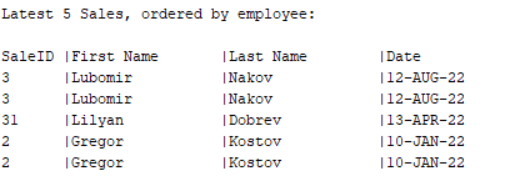
RPAD(cust\_record.LAST\_NAME, 16) || ' |' ||

TO\_CHAR(cust\_record.SALE\_DATE)

);

END LOOP;

END FIND\_LATEST\_SALES;



--------------------------------------------------------

--Да се направи справка за закупени стоки от клиент, подредени по вид и дата.

--------------------------------------------------------

create or replace PROCEDURE FIND\_CLIENT\_PRODUCTS (

firstName CLIENTS.FIRST\_NAME%TYPE,

lastName CLIENTS.LAST\_NAME%TYPE

) IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Client sales report for: ' || firstName || ' ' || lastName || ', ordered by type and date:' || CHR(10));

DBMS\_OUTPUT.PUT\_LINE(

RPAD('First Name', 16) || ' |' ||

RPAD('Last Name', 16) || ' |' ||

RPAD('Product', 32) || ' |' ||

RPAD('Quantity', 8) || ' |' ||

RPAD('Price', 8) || ' |' ||

RPAD('Type', 7) || ' |' ||

RPAD('SaleID', 6) || ' |' ||

RPAD('Date', 4)

);

FOR cust\_record IN (

SELECT c.FIRST\_NAME, c.LAST\_NAME, p.PROD\_NAME, sp.SALE\_QUANTITY, p.PRICE, t.TYPE\_NAME, s.SALE\_DATE, s.SALE\_ID

FROM SALES s

JOIN CLIENTS c ON s.CLIENT\_ID = c.CLIENT\_ID

JOIN SALES\_PRODUCTS sp ON s.SALE\_ID = sp.SALE\_ID

JOIN PRODUCTS p ON sp.PROD\_ID = p.PROD\_ID

JOIN TYPES t ON p.TYPE\_ID = t.TYPE\_ID

WHERE c.FIRST\_NAME LIKE firstName AND c.LAST\_NAME LIKE lastName

ORDER BY t.TYPE\_NAME, s.SALE\_DATE DESC

)

LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(cust\_record.FIRST\_NAME, 16) || ' |' ||

RPAD(cust\_record.LAST\_NAME, 16) || ' |' ||

RPAD(cust\_record.PROD\_NAME, 32) || ' |' ||

RPAD(cust\_record.SALE\_QUANTITY, 8) || ' |' ||

RPAD(cust\_record.PRICE, 8) || ' |' ||

RPAD(cust\_record.TYPE\_NAME, 7) || ' |' ||

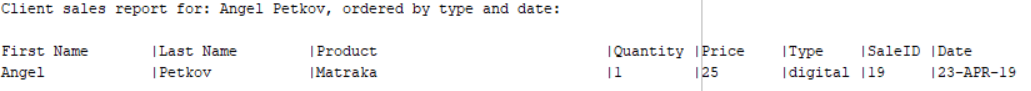
RPAD(cust\_record.SALE\_ID, 6) || ' |' ||

TO\_CHAR(cust\_record.SALE\_DATE)

);

END LOOP;

END FIND\_CLIENT\_PRODUCTS;



--------------------------------------------------------

-- Справка, извeждаща всички закупени продукти в дадена продажба.

--------------------------------------------------------

create or replace PROCEDURE FIND\_SALE\_PRODUCTS (

saleID SALES.SALE\_ID%TYPE

) IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Products in sale ID: ' || saleID || CHR(10));

DBMS\_OUTPUT.PUT\_LINE(

RPAD('SaleID', 6) || ' |' ||

RPAD('Product', 32) || ' |' ||

RPAD('Quantity', 8) || ' |' ||

RPAD('Price', 8) || ' |' ||

RPAD('Type', 7) || ' |' ||

RPAD('Date', 4)

);

FOR cust\_record IN (

SELECT s.SALE\_ID, p.PROD\_NAME, sp.SALE\_QUANTITY, p.PRICE, t.TYPE\_NAME, s.SALE\_DATE

FROM SALES s

JOIN SALES\_PRODUCTS sp ON s.SALE\_ID = sp.SALE\_ID

JOIN PRODUCTS p ON sp.PROD\_ID = p.PROD\_ID

JOIN TYPES t ON p.TYPE\_ID = t.TYPE\_ID

WHERE s.SALE\_ID LIKE saleID

)

LOOP

DBMS\_OUTPUT.PUT\_LINE(

RPAD(cust\_record.SALE\_ID, 6) || ' |' ||

RPAD(cust\_record.PROD\_NAME, 32) || ' |' ||

RPAD(cust\_record.SALE\_QUANTITY, 8) || ' |' ||

RPAD(cust\_record.PRICE, 8) || ' |' ||

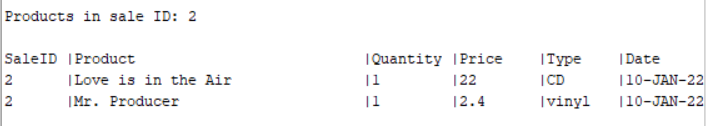
RPAD(cust\_record.TYPE\_NAME, 7) || ' |' ||

TO\_CHAR(cust\_record.SALE\_DATE)

);

END LOOP;

END FIND\_SALE\_PRODUCTS;



Return Sale Procedure

create or replace PROCEDURE return\_sale(return\_sale\_id sales.sale\_id%TYPE)

AS

product\_id sales\_products.prod\_id%TYPE;

quantity sales\_products.sale\_quantity%TYPE;

BEGIN

FOR rec IN (SELECT prod\_id, sale\_quantity

FROM sales\_products

WHERE sale\_id = return\_sale\_id)

LOOP

product\_id := rec.prod\_id;

quantity := rec.sale\_quantity;

-- Възстановяваме количествата на продуктите

UPDATE products

SET av\_quantity = av\_quantity + quantity

WHERE prod\_id = product\_id;

END LOOP;

-- Премахваме редовете от таблица Sales\_Products

DELETE FROM sales\_products

WHERE sale\_id = return\_sale\_id;

-- Премахваме реда от таблица Sales

DELETE FROM sales

WHERE sale\_id = return\_sale\_id;

END return\_sale;

Triggers

Да се изпълняват по отделно

--------------------------------------------------------

-- DDL for Trigger AR\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE TRIGGER "AR\_TR\_AI" -- ARtist\_TRigger\_AutoIncrement

before insert on artists

for each row

WHEN (NEW.artist\_id is null) begin

:NEW.artist\_id := ARTIST\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "AR\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger CL\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE TRIGGER "CL\_TR\_AI"

before insert on clients

for each row

WHEN (NEW.client\_id is null) begin

:NEW.client\_id := CLIENT\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "CL\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger EM\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE TRIGGER "EM\_TR\_AI"

before insert on employees

for each row

WHEN (NEW.empl\_id is null) begin

:NEW.empl\_id := EMPLOYEE\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "EM\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger LA\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE TRIGGER "LA\_TR\_AI"

before insert on labels

for each row

WHEN (NEW.label\_id is null) begin

:NEW.label\_id := LABEL\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "LA\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger PR\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE TRIGGER "PR\_TR\_AI"

before insert on products

for each row

WHEN (NEW.prod\_id is null) begin

:NEW.prod\_id := PRODUCT\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "PR\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger SA\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE TRIGGER "SA\_TR\_AI"

before insert on sales

for each row

WHEN (NEW.sale\_id is null) begin

:NEW.sale\_id := SALE\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "SA\_TR\_AI" ENABLE;

--------------------------------------------------------

-- DDL for Trigger SAPR\_TR\_AI

--------------------------------------------------------

CREATE OR REPLACE TRIGGER "SAPR\_TR\_AI"

before insert on sales\_products

for each row

WHEN (NEW.sale\_prod\_id is null) begin

:NEW.sale\_prod\_id := SALE\_PRODUCT\_SEQ.NEXTVAL;

end;

/

ALTER TRIGGER "SAPR\_TR\_AI" ENABLE;

Потребителски роли и привилегии (DDL, DML)

---user creation---

create user CLIENT identified by client;

create user EMPLOYEE identified by employee;

create user ADMIN identified by admin;

---role creation---

create ROLE clients;

create ROLE employees;

create ROLE admins;

---granting privileges to roles---

---admin---

GRANT SELECT, INSERT, DELETE, UPDATE ON ARTISTS TO admins;

GRANT SELECT, INSERT, DELETE, UPDATE ON CLIENTS TO admins;

GRANT SELECT, INSERT, DELETE, UPDATE ON EMPLOYEES TO admins;

GRANT SELECT, INSERT, DELETE, UPDATE ON GENRES TO admins;

GRANT SELECT, INSERT, DELETE, UPDATE ON LABELS TO admins;

GRANT SELECT, INSERT, DELETE, UPDATE ON POSITIONS TO admins;

GRANT SELECT, INSERT, DELETE, UPDATE ON PRODUCTS TO admins;

GRANT SELECT, INSERT, DELETE, UPDATE ON SALES TO admins;

GRANT SELECT, INSERT, DELETE, UPDATE ON SALES\_PRODUCTS TO admins;

GRANT SELECT, INSERT, DELETE, UPDATE ON TYPES TO admins;

GRANT EXECUTE ON ART\_INS TO admins;

GRANT EXECUTE ON ART\_UPD TO admins;

GRANT EXECUTE ON CLI\_INS TO admins;

GRANT EXECUTE ON CLI\_UPD TO admins;

GRANT EXECUTE ON EMP\_INS TO admins;

GRANT EXECUTE ON EMP\_UPD TO admins;

GRANT EXECUTE ON LAB\_INS TO admins;

GRANT EXECUTE ON LAB\_UPD TO admins;

GRANT EXECUTE ON PRO\_INS TO admins;

GRANT EXECUTE ON PRO\_UPD TO admins;

GRANT EXECUTE ON SAL\_INS TO admins;

GRANT EXECUTE ON SAL\_UPD TO admins;

GRANT EXECUTE ON SAL\_PRO\_INS TO admins;

GRANT EXECUTE ON SAL\_PRO\_UPD TO admins;

GRANT EXECUTE ON FIND\_CLIENT\_PRODUCTS TO admins;

GRANT EXECUTE ON FIND\_EMPLOYEE\_SALES TO admins;

GRANT EXECUTE ON FIND\_LATEST\_SALES TO admins;

GRANT EXECUTE ON FIND\_SALE\_PRODUCTS TO admins;

GRANT EXECUTE ON RETURN\_SALE TO admins;

---clients---

GRANT SELECT ON ARTISTS TO clients;

GRANT SELECT ON GENRES TO clients;

GRANT SELECT ON LABELS TO clients;

GRANT SELECT ON PRODUCTS TO clients;

GRANT SELECT ON TYPES TO clients;

GRANT EXECUTE ON return\_sale TO clients;

---employees---

GRANT SELECT ON ARTISTS TO employees;

GRANT SELECT ON CLIENTS TO employees;

GRANT SELECT ON EMPLOYEES TO employees;

GRANT SELECT ON GENRES TO employees;

GRANT SELECT ON LABELS TO employees;

GRANT SELECT ON POSITIONS TO employees;

GRANT SELECT ON PRODUCTS TO employees;

GRANT SELECT ON SALES TO employees;

GRANT SELECT ON SALES\_PRODUCTS TO employees;

GRANT SELECT ON TYPES TO employees;

GRANT INSERT, UPDATE ON ARTISTS TO employees;

GRANT INSERT, UPDATE ON CLIENTS TO employees;

GRANT INSERT, UPDATE ON GENRES TO employees;

GRANT INSERT, UPDATE ON LABELS TO employees;

GRANT INSERT, UPDATE ON POSITIONS TO employees;

GRANT INSERT, UPDATE ON SALES TO employees;

GRANT INSERT, UPDATE ON SALES\_PRODUCTS TO employees;

GRANT INSERT, UPDATE ON TYPES TO employees;

GRANT EXECUTE ON ART\_INS TO employees;

GRANT EXECUTE ON CLI\_INS TO employees;

GRANT EXECUTE ON EMP\_INS TO employees;

GRANT EXECUTE ON LAB\_INS TO employees;

GRANT EXECUTE ON PRO\_INS TO employees;

GRANT EXECUTE ON SAL\_INS TO employees;

GRANT EXECUTE ON SAL\_PRO\_INS TO employees;

GRANT EXECUTE ON FIND\_CLIENT\_PRODUCTS TO employees;

GRANT EXECUTE ON FIND\_EMPLOYEE\_SALES TO employees;

GRANT EXECUTE ON FIND\_LATEST\_SALES TO employees;

GRANT EXECUTE ON FIND\_SALE\_PRODUCTS TO employees;

---revoking privileges from roles---

---clients---

REVOKE INSERT, DELETE, UPDATE ON ARTISTS FROM clients;

REVOKE INSERT, DELETE, UPDATE ON CLIENTS FROM clients;

REVOKE INSERT, DELETE, UPDATE ON EMPLOYEES FROM clients;

REVOKE INSERT, DELETE, UPDATE ON GENRES FROM clients;

REVOKE INSERT, DELETE, UPDATE ON LABELS FROM clients;

REVOKE INSERT, DELETE, UPDATE ON POSITIONS FROM clients;

REVOKE INSERT, DELETE, UPDATE ON PRODUCTS FROM clients;

REVOKE INSERT, DELETE, UPDATE ON SALES FROM clients;

REVOKE INSERT, DELETE, UPDATE ON SALES\_PRODUCTS FROM clients;

REVOKE INSERT, DELETE, UPDATE ON TYPES FROM clients;

---employees---

REVOKE DELETE ON ARTISTS FROM employees;

REVOKE DELETE ON CLIENTS FROM employees;

REVOKE DELETE ON EMPLOYEES FROM employees;

REVOKE DELETE ON GENRES FROM employees;

REVOKE DELETE ON LABELS FROM employees;

REVOKE DELETE ON POSITIONS FROM employees;

REVOKE DELETE ON PRODUCTS FROM employees;

REVOKE DELETE ON SALES FROM employees;

REVOKE DELETE ON SALES\_PRODUCTS FROM employees;

REVOKE DELETE ON TYPES FROM employees;

REVOKE EXECUTE ON RETURN\_SALE TO employees;

---granting roles to users---

grant admins to ADMIN;

grant employees to EMPLOYEE;

grant clients to CLIENT;