**--usuwanie wcześniejszych danych**

DROP RULE archiwium\_1 ON produkty;

DROP TABLE przed\_podwyzka;

DROP TRIGGER archiwum\_2 ON produkty;

DROP FUNCTION producent\_produkt(varchar(20));

DROP VIEW wartosc;

DROP TABLE elementy;

DROP TABLE zamowienia;

DROP TABLE produkty;

DROP TABLE klienci;

DROP TABLE przed\_obnizka;

DROP FUNCTION archiwum\_2();

--SET client\_encoding='UTF-8';

SET datestyle TO 'ISO,European';

**--tworzenie tabel bazy**

CREATE TABLE klienci

(

id\_klienta integer PRIMARY KEY,

imie varchar(16) NOT NULL,

nazwisko varchar(32) NOT NULL,

adres varchar(32),

kod char(6),

miasto varchar(32),

telefon varchar(16),

email varchar(32)

);

CREATE TABLE produkty

(

id\_produktu integer PRIMARY KEY,

rodzaj varchar(32) NOT NULL,

komponenty varchar(32) NOT NULL,

producent varchar(32) NOT NULL,

model varchar(32) NOT NULL,

gwarancja varchar(16),

cena numeric(6,2)

);

CREATE TABLE zamowienia

(

id\_zamowienia integer PRIMARY KEY,

termin\_zamowienia date NOT NULL,

termin\_dostawy date NOT NULL,

forma\_dostawy varchar(16) NOT NULL,

id\_klienta integer REFERENCES klienci(id\_klienta)

);

CREATE TABLE elementy

(

id\_elementu integer PRIMARY KEY,

id\_zamowienia integer REFERENCES zamowienia(id\_zamowienia),

id\_produktu integer REFERENCES produkty(id\_produktu),

ilosc integer NOT NULL

);

**--wprowadzanie danych do tabel**

INSERT INTO klienci(id\_klienta, imie, nazwisko, adres, kod, miasto, telefon, email)

VALUES (1,'Jan','Kowalski','Kadetów 6/3','80-298','Gdańsk',NULL,NULL),

(2,'Władysław','Dobrowolski','Grunwaldzka 111','80-123','Gdańsk','58260860','wdobrowolski@wp.pl'),

(3,'Anna','Szepietowska','Podchorążych 10/1','82-386','Sopot','663123432',NULL),

(4,'Artur','Leszczyński','Grunwaldzka 24','81-245','Gdynia','581235478','a.leszczynski@onet.pl'),

(5,'Jan','Dobrowolski','Miszewskiego 2/4','09-400','Płock',NULL,'jan\_d@gmail.pl'),

(6,'Janusz','Miszewski','Grunwaldzka 12/4','89-411','Gdańsk',NULL,'janusz@gmail.pl'),

(7,'Maria','Kaczyńska','Bażyńskiego 22/14','80-421','Gdańsk',583442134,'maria\_kaczynska@o2.pl'),

(8,'Krzysztof','Malinowski','Jana z Kolna 37','85-100','Gdynia',NULL,NULL),

(9,'Jan','Kowalski','Miszewskiego 10/14','82-111','Gdynia','666123654','jan-kowalski@upc.pl'),

(10,'Joanna','Napieralska','Kadetów 2/14','80-298','Gdańsk',NULL,'jnapieralska@gmail.pl');

INSERT INTO produkty(id\_produktu, rodzaj, komponenty, producent, model, gwarancja, cena)

VALUES (1,'użytkowe','CD','Verbatim','DVD-R 10szt.',NULL,10),

(2,'sprzęt','drukarka','Canon','MP550','2 lata',316),

(3,'oprogramowanie','systemy operacyjne','Microsoft','Windows 7','5 lat',320),

(4,'sprzęt','drukarka','HP','1018','2 lata',520),

(5,'oprogramowanie','systemy operacyjne','Microsoft','Windows XP','2 lata',200),

(6,'sprzęt','notbook','Toshiba','A300-121','2 lata',1700),

(7,'oprogramowanie','gry','Gameover','Street Fighter 4',NULL,60),

(8,'oprogramowanie','gry','Steam','Counter Strike',NULL,50),

(9,'oprogramowanie','gry','Gameover','Street Fighter 4 full',NULL,120),

(10,'użytkowe','CD','TDK','DVD-RW 10szt.',NULL,45);

INSERT INTO zamowienia(id\_zamowienia, termin\_zamowienia, termin\_dostawy, forma\_dostawy, id\_klienta)

VALUES (1,'15-03-2011','17-03-2011','kurier',1),

(2,'22-04-2011','23-04-2011','poczta',6),

(3,'22-04-2011','29-04-2011','osobisty',9),

(4,'22-04-2011','23-04-2011','poczta',3),

(5,'25-04-2011','30-04-2011','kurier',6),

(6,'25-04-2011','26-04-2011','osobisty',8),

(7,'27-04-2011','29-04-2011','osobisty',2),

(8,'29-04-2011','01-05-2011','poczta',8),

(9,'30-04-2011','01-05-2011','kurier',7),

(10,'30-04-2011','02-05-2011','poczta',1),

(11,'01-05-2011','03-05-2011','poczta',4),

(12,'01-05-2011','01-05-2011','osobisty',5),

(13,'02-05-2011','02-05-2011','osobisty',3),

(14,'04-05-2011','06-05-2011','poczta',4),

(15,'05-05-2011','06-05-2011','osobisty',2);

INSERT INTO elementy(id\_elementu, id\_zamowienia, id\_produktu, ilosc)

VALUES (1,1,1,1),

(2,2,2,5),

(3,3,3,4),

(4,3,1,1),

(5,4,1,4),

(6,4,2,5),

(7,4,3,4),

(8,5,2,1),

(9,5,3,4),

(10,6,1,2),

(11,7,2,1),

(12,8,10,1),

(13,9,9,1),

(14,10,7,2),

(15,11,1,2),

(16,12,9,1),

(17,13,5,1),

(18,14,7,1),

(19,14,3,1),

(20,15,10,1);

**--przegląd wszystkich danych utworzonych tabel**

SELECT \* FROM klienci;

SELECT \* FROM produkty;

SELECT \* FROM zamowienia;

SELECT \* FROM elementy;

**--przegląd wybranych kolumn z pojedynczych tabel**

SELECT imie,nazwisko,adres FROM klienci;

SELECT producent,model,cena FROM produkty;

**--dodanie dodatkowej kolumny do jednej z tabel**

ALTER TABLE klienci ADD COLUMN status varchar(16);

SELECT \* FROM klienci;

**--wybór 2 kolumn tabeli spełniający określony warunek**

SELECT model,producent FROM produkty WHERE producent='Gameover';

**--zastosowanie funkcji agregującej liczącej ilość określonej kolumny tabeli i nadanie jej nowej nazwy**

SELECT count(id\_zamowienia) AS ilosc\_zamowien FROM zamowienia;

**--tabela łącząca dane z dwóch innych tabel spełniające określony warunek**

SELECT id\_klienta,nazwisko,id\_zamowienia FROM zamowienia NATURAL JOIN klienci WHERE id\_zamowienia=3;

**--wybór określonych kolumn z dwóch połączonych tabel uporządkowanych względem identyfikatora klienta**

SELECT id\_klienta,imie,nazwisko,termin\_dostawy FROM klienci k NATURAL JOIN zamowienia z

GROUP BY k.id\_klienta,k.imie,k.nazwisko,z.termin\_dostawy ORDER BY id\_klienta;

**--wyszukiwanie danych i pokazanie w dodatkowej kolumnie przez instrukcę wyboru**

SELECT imie,nazwisko,miasto,

CASE WHEN nazwisko='Kowalski' OR nazwisko='Dobrowolski' THEN 'kolega'

ELSE 'inny'

END AS znajomi

FROM klienci;

**--usuwanie danych z określonych kolumn spełniających określony warunek**

SELECT id\_elementu,ilosc FROM elementy;

--DELETE FROM elementy WHERE ilosc<2;

SELECT id\_elementu,ilosc FROM elementy;

SELECT producent,model FROM produkty;

--DELETE FROM produkty WHERE producent='Toshiba';

SELECT producent,model FROM produkty;

**--aktualizacja danych jednej z tabel (zmiana nazwy)**

SELECT rodzaj,producent,model FROM produkty;

UPDATE produkty SET rodzaj='programy' WHERE rodzaj='oprogramowanie';

SELECT rodzaj,producent,model FROM produkty;

**--przegląd określonych danych jednej z tabel**

SELECT producent,model,cena FROM produkty WHERE producent='Canon';

**--dodanie kolumny do jednej z tabel**

ALTER TABLE produkty ADD COLUMN rabat numeric(6,2);

SELECT \* FROM produkty;

**--wyznaczenie sposobu obliczania danych i ich aktualizacja w dodatkowej kolumnie na podstawie danych z innej kolumny**

UPDATE produkty SET rabat=0.10\*cena;

**--dodanie następnej kolumny**

ALTER TABLE produkty ADD COLUMN nowa\_cena numeric(6,2);

**--aktualizacja danych na podstawie wartości z dwóch innych kolumn**

UPDATE produkty SET nowa\_cena=cena-rabat;

SELECT \* FROM produkty;

**--pokazanie danych z jednej z tabel spełniających określony warunek**

SELECT producent,model,cena,nowa\_cena FROM produkty WHERE producent='Microsoft';

**--utworzenie perspektywy zapamiętującej określone zapytanie**

CREATE VIEW wartosc

AS SELECT sum(cena) AS wartosc\_towarow From produkty;

SELECT \* FROM wartosc;

**--wybór danych z dwóch połączonych tabel uporządkowanych względem identyfikatora jednej z tabel**

SELECT id\_klienta,imie,nazwisko,termin\_dostawy FROM klienci k NATURAL JOIN zamowienia z

GROUP BY k.id\_klienta,k.imie,k.nazwisko,z.termin\_dostawy ORDER BY id\_klienta;

**--wybór liczby danych pod określoną nazwą na podstawie danych z drugiej tabeli**

SELECT count(k.id\_klienta) AS liczba\_zamowien FROM zamowienia z

INNER JOIN klienci k ON z.id\_klienta=k.id\_klienta;

**--wybór liczby danych pod określoną nazwą na podstawie danych z drugiej tabeli spełniajacych okreslony warunek**

SELECT count(k.id\_klienta) AS zamowienia\_po\_17\_kwietnia FROM zamowienia z

INNER JOIN klienci k ON z.id\_klienta=k.id\_klienta WHERE termin\_zamowienia>'27-04-2011';

**--wybór w określonym porządku danych z dwóch tabel**

SELECT termin\_dostawy,imie,nazwisko FROM klienci k

INNER JOIN zamowienia z ON z.id\_klienta=k.id\_klienta

GROUP BY k.id\_klienta,imie,nazwisko,termin\_dostawy ORDER BY termin\_dostawy;

**--wybór danych z funkcją agregującą na podstawie trzech połączonych tabel w postaci zagnieżdzonej**

SELECT imie, nazwisko, sum(ilosc \* cena) AS wydatki

FROM ( ( ( klienci k

INNER JOIN zamowienia z

ON z.id\_klienta = k.id\_klienta

)

INNER JOIN elementy e

ON e.id\_zamowienia = z.id\_zamowienia

)

INNER JOIN produkty p

ON e.id\_produktu = p.id\_produktu

)

GROUP BY k.id\_klienta, imie, nazwisko ORDER BY nazwisko;

**--wybór danych z kilku tabel połączonych za pomocą instrukcji wyboru**

SELECT z.id\_zamowienia,e.id\_elementu,k.imie,k.nazwisko,p.model

FROM zamowienia z, klienci k, produkty p, elementy e

WHERE e.id\_zamowienia=z.id\_zamowienia AND e.id\_produktu=p.id\_produktu AND z.id\_klienta=k.id\_klienta;

**--funkcja wybierająca nazwę produktu po podaniu producenta**

CREATE FUNCTION producent\_produkt(varchar(20))

RETURNS varchar(20) AS

'

SELECT model FROM produkty WHERE producent=$1;

'

LANGUAGE sql;

SELECT producent\_produkt('Microsoft');

**--reguła zapisująca stare ceny przed ich zmianą**

CREATE TABLE przed\_obnizka

(

id\_produktu integer PRIMARY KEY,

producent varchar(32) NOT NULL,

model varchar(32) NOT NULL,

cena numeric(6,2),

zmiana timestamp

);

CREATE RULE archiwium\_1 AS

ON UPDATE TO produkty

WHERE old.cena<>new.cena

DO ALSO INSERT INTO przed\_obnizka

VALUES (old.id\_produktu, old.producent, old.model, old.cena, current\_timestamp);

**-- przykład użycia tej reguły**

DELETE FROM przed\_obnizka;

SELECT producent,model,cena FROM produkty WHERE model LIKE '%MP550%';

SELECT producent,model,cena FROM przed\_obnizka;

UPDATE produkty SET cena=cena\*0.9

WHERE model LIKE '%MP550%';

SELECT producent,model,cena FROM produkty

WHERE model LIKE '%MP550%';

SELECT producent,model,cena FROM przed\_obnizka;

**--przykład wyzwalacza o działaniu takim jak poprzednia reguła z innym parametrem**

CREATE TABLE przed\_podwyzka

(

id\_produktu integer PRIMARY KEY,

producent varchar(32) NOT NULL,

model varchar(32) NOT NULL,

cena numeric(6,2),

zmiana timestamp

);

CREATE FUNCTION archiwum\_2()

RETURNS TRIGGER AS $$

BEGIN

IF old.cena<>new.cena

THEN INSERT INTO przed\_podwyzka VALUES (old.id\_produktu, old.producent, old.model, old.cena, current\_timestamp);

RAISE NOTICE 'podwyżka ceny towaru nr: %',old.id\_produktu;

END IF;

RETURN NULL;

END

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER archiwum\_2

AFTER UPDATE ON produkty

FOR EACH ROW EXECUTE PROCEDURE archiwum\_2();

**-- przykład użycia tego wyzwalacza**

DELETE FROM przed\_podwyzka;

SELECT producent,model,cena FROM produkty WHERE model LIKE '%1018%';

SELECT producent,model,cena FROM przed\_podwyzka;

UPDATE produkty SET cena=cena\*1.1

WHERE model LIKE '%1018%';

SELECT producent,model,cena FROM produkty

WHERE model LIKE '%1018%';

SELECT producent,model,cena FROM przed\_podwyzka;