Projekt zaliczeniowy
Programowanie serwera baz danych

Jakub Karalus
380950

2019/2020

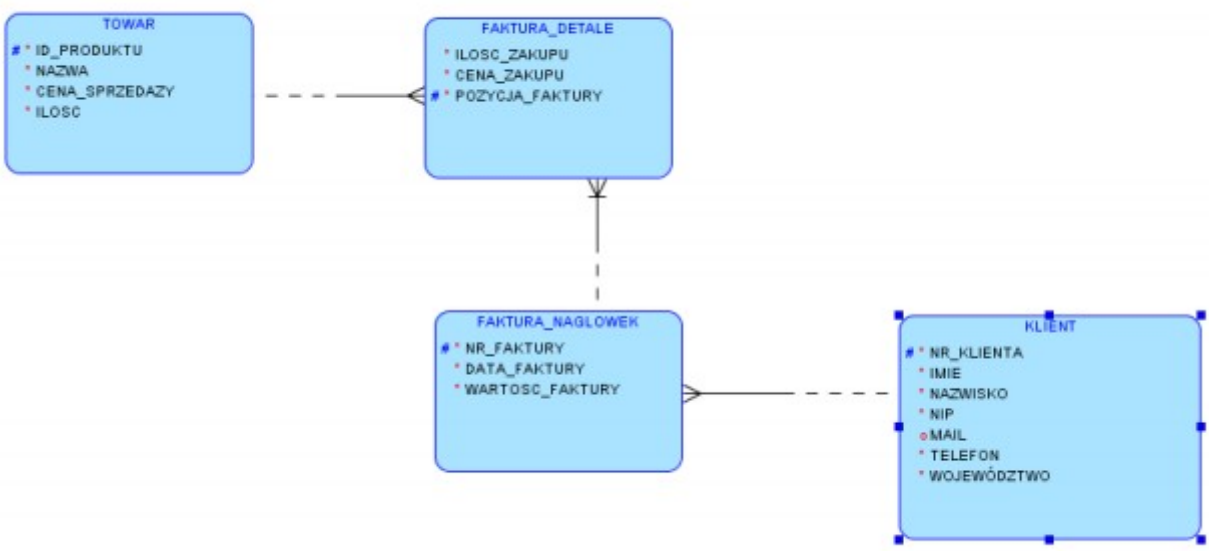
Analiza biznesowa

Projekt obrazuje hurtownię sportową, produkty sprzedawane są stacjonarnie w bardzo dużych ilościach. W sklepie możemy znaleźć wszelkiego rodzaju produkty sportowe, zaczynając od butów, aż po ubrania.

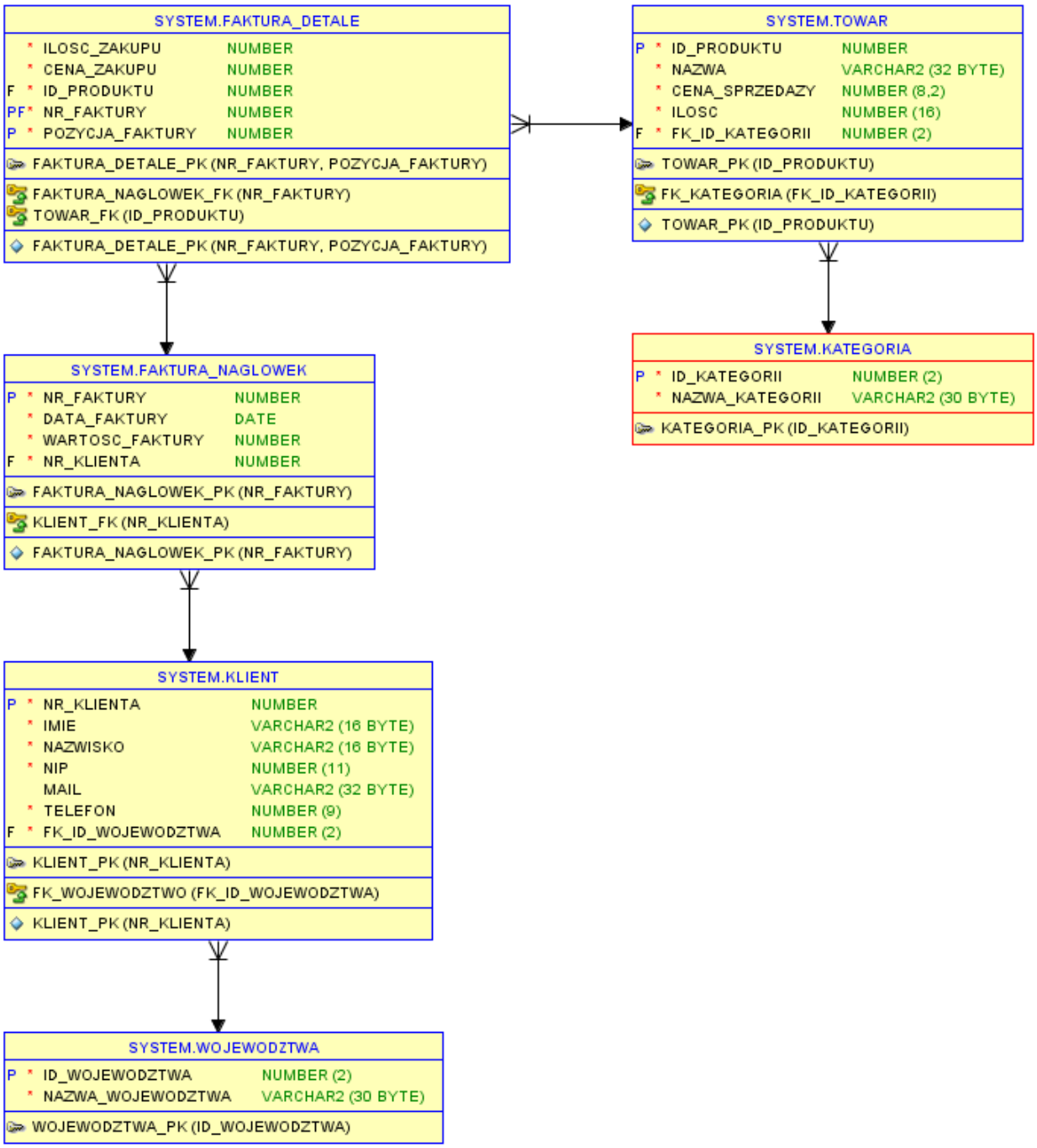
Informacje wstępne:

- Zamówienia w sklepie składają tylko zarejestrowani klienci, którzy podali wszystkie informacje o sobie, takie jak imię, nazwisko, województwo, telefon itd.
- Obsługa sklepu dodaje na bieżąco produkty
- Sklep wystawia faktury
- Hurtownia nie uwzględnia cen zakupu produktów, operuje tylko na ceny sprzedaży
- Klienci mogą na jedną fakturę wziąć dowolną ilość produktów
- Sklep przyjmuje klientów zarówno prywatnych, jak i firmy, NIP można zastąpić numerem pesel
- Złożone zamówienie można edytować oraz anulować, jak również zmienić lub usunąć poszczególne pozycje
- Hurtownia posiada ograniczony stan produktów
- Każdy produkt posiada kategorię, sklep ma z góry narzucone kategorie, aby nie było kilku kategorii podobnych do siebie, jak np. “Bluza” oraz “bluza”
- Stan produktów jest zmieniany przy każdej transakcji
- Sklep nie uwzględnia podatku VAT

Model logiczny projektu



Model relacyjny projektu



Oprogramowanie ORACLE

Oprogramowanie tworzy bazę danych, wprowadza dane do tabel, tworzy generator oraz widoki potrzebne do zobrazowania zysków firmy.

1.Tworzymy tabele potrzebne do projektu oraz tworzymy klucze główne oraz obce

```
-- HURTOWNIA SPORTOWA
-- JAKUB KARALUS 380950
CREATE TABLE faktura_detale (
 ilosc_zakupu NUMBER NOT NULL,
 cena_zakupu NUMBER NOT NULL,
 id_produktu NUMBER NOT NULL,
 nr_faktury NUMBER NOT NULL,
 pozycja_faktury NUMBER NOT NULL
);
ALTER TABLE faktura_detale ADD CONSTRAINT faktura_detale_pk PRIMARY KEY (
nr_faktury,

pozycja_faktury );
CREATE TABLE faktura_naglowek (
 nr_faktury NUMBER NOT NULL,
 data_faktury DATE NOT NULL,
 wartosc_faktury NUMBER NOT NULL,
 nr_klienta NUMBER NOT NULL
);
ALTER TABLE faktura_naglowek ADD CONSTRAINT faktura_naglowek_pk PRIMARY
KEY ( nr_faktury );
CREATE TABLE klient (
 nr_klienta NUMBER NOT NULL,
 imie VARCHAR2(16) NOT NULL,
 nazwisko VARCHAR2(16) NOT NULL,
 nip NUMBER(11) NOT NULL,
 mail VARCHAR2(32),
 telefon NUMBER(9) NOT NULL,
 województwo VARCHAR2(30) NOT NULL
);
ALTER TABLE klient ADD CONSTRAINT klient_pk PRIMARY KEY ( nr_klienta );
CREATE TABLE towar (
 id_produktu NUMBER NOT NULL,
 nazwa VARCHAR2(32) NOT NULL,
 cena_sprzedazy NUMBER(8, 2) NOT NULL,
 ilosc NUMBER(16) NOT NULL
);
ALTER TABLE towar ADD CONSTRAINT towar_pk PRIMARY KEY ( id_produktu );
ALTER TABLE faktura_detale
 ADD CONSTRAINT faktura_naglowek_fk FOREIGN KEY ( nr_faktury )
 REFERENCES faktura_naglowek ( nr_faktury );
ALTER TABLE faktura_naglowek
 ADD CONSTRAINT klient_fk FOREIGN KEY ( nr_klienta )
 REFERENCES klient ( nr_klienta );
ALTER TABLE faktura_detale
 ADD CONSTRAINT towar_fk FOREIGN KEY ( id_produktu )
 REFERENCES towar ( id produktu );
```

2. Tworzymy tabele słownikowe

```
-- tabela slownikowa wojewodztwa
create table WOJEWODZTWA(
id_wojewodztwa number(2) primary key,
nazwa_wojewodztwa varchar(30) not null
);
alter table klient
drop column województwo;
alter table klient
add fk_id_wojewodztwa number(2) not null;
alter table klient
add constraint fk_wojewodztwo foreign key (fk_id_wojewodztwa) references
WOJEWODZTWA(id_wojewodztwa);
-- tabela slownikowa kategoria
create table KATEGORIA(
id_kategorii number(2) primary key,
nazwa_kategorii varchar(30) not null
);
alter table towar
add fk_id_kategorii number(2) not null;
alter table towar
add constraint fk_kategoria foreign key (fk_id_kategorii) references
KATEGORIA(id_kategorii);
```

3. Wprowadzamy przykładowe dane

```
-- inserty do kategorii
insert into KATEGORIA VALUES(
1,
'Bluzy'
);
insert into KATEGORIA VALUES(
2,
'Spodnie'
);
insert into KATEGORIA VALUES(
3,
'Spodenki'
);
insert into KATEGORIA VALUES(
4,
'Podkoszulki'
);
insert into KATEGORIA VALUES(
5,
'Odzież termoaktywna'
);
insert into KATEGORIA VALUES(
6,
'Czapki'
);
insert into KATEGORIA VALUES(
7,
'Kominy'
);
insert into KATEGORIA VALUES(
8,
'Skarpetki'
);
insert into KATEGORIA VALUES(
9,
'Bielizna'
);
insert into KATEGORIA VALUES(
10,
'Rękawiczki'
);
```

```sql
-- inserty do wojewodztw
insert into WOJEWODZTWA values(
1,
'dolnośląskie'
);
insert into WOJEWODZTWA values(
2,
'kujawsko-pomorskie'
);
insert into WOJEWODZTWA values(
3,
'lubelskie'
);
insert into WOJEWODZTWA values(
4,
'lubuskie'
);
insert into WOJEWODZTWA values(
5,
'łódzkie'
);
insert into WOJEWODZTWA values(
6,
'małopolskie'
);
insert into WOJEWODZTWA values(
7,
'mazowieckie'
);
insert into WOJEWODZTWA values(
8,
'opolskie'
);
insert into WOJEWODZTWA values(
9,
'podkarpackie'
);
insert into WOJEWODZTWA values(
10,
'podlaskie'
);
insert into WOJEWODZTWA values(
11,
'pomorskie'
```

4. Tworzymy trigger, który będzie działał na tabeli FAKTURA_DETALE, aby przy wprowadzaniu nowej transakcji odejmować ilość towaru z hurtowni, działa również dla zmieniania istniejącej już transakcji oraz usuwania transakcji.

```
-- trigger gdy ktos dodaje nowy produkt do swojej transakcji
create or replace TRIGGER TR_FAKTURA_DETALE_ZMIANA
BEFORE INSERT OR UPDATE OR DELETE ON FAKTURA_DETALE
FOR EACH ROW
BEGIN
 if INSERTING then
 update TOWAR
 set ilosc = ilosc - :NEW.ilosc_zakupu
 where id_produktu = :NEW.id_produktu;
 end if;
 if UPDATING then
 update TOWAR
 set ilosc = ilosc + :OLD.ilosc_zakupu - :NEW.ilosc_zakupu
 where id_produktu = :NEW.id_produktu;
 end if;
 if DELETING then
 update TOWAR
 set ilosc = ilosc + :OLD.ilosc_zakupu
 where id_produktu = :OLD.id_produktu;
 end if;
END;
```

5. Tworzymy sekwencję, która zaczyna się od 1, służy do numerowania faktur

```
-- generowanie transakcji
create sequence sq_nowa_transakcja;
```

6. Tworzymy funkcję, która zwraca ile ktoś kupił pozycji na jednej fakturze, ograniczamy pozycje do max 10(można zmienić w dowolnym momencie)

```
-- losuj ile ktos kupil pozycji
create or replace function fn_daj_liczbe_pozycji return number
as
v_ile number;
begin
select round(dbms_random.value(1, 10)) into v_ile from dual;
return v_ile;
end;
```

7. Tworzymy funkcję, która zwraca ile ktoś kupił sztuk danego produktu

```
-- losuj ile ktos kupil sztuk
create or replace function fn_daj_ilosc return number
as
v_ile number;
begin
select round(dbms_random.value(1, 100)) into v_ile from dual;
return v_ile;
end;
```

8. Tworzymy funkcję, która zwróci nam kto dokonuje transakcji

```
-- losuj kto kupil
create or replace function fn_daj_nr_klienta return number
as
v_nr number;
begin
select nr_klienta into v_nr from (select nr_klienta from klient order by
dbms_random.value) where rownum = 1;
return v_nr;
end;
```

9. Tworzymy funkcję, która zwróci nam jaki konkretnie produkt ktoś kupił

```
-- losuj co kupil(produkt)
create or replace function fn_daj_numer_produktu return number
as
v_id number;
begin
select id_produktu into v_id from
 (select id_produktu from towar
 order by dbms_random.value)
 where rownum = 1;
return v_id;
end;
```

10. Tworzymy funkcję, która dla wylosowanego wcześniej produktu zwróci nam jego cenę

```
-- daj cene produktu wylosowanego
create or replace function fn_daj_cene_produktu(v_id_produktu number)
return number
as
v_cena number;
begin
select cena_sprzedazy into v_cena from towar where id_produktu =
v_id_produktu;
return v_cena;
end;
```

11. Tworzymy procedurę, która wygeneruje nam pojedynczą pozycję na fakturze

```
-- procedury:
-- generuj pozycje, czyli kupno jednego towaru
create or replace procedure pr_generuj_pozycje(v_nr_faktury number,
v_pozycja_faktury number) as
v_nr_produktu number;
v_cena_produktu number;
v_ilosc_produktu number;
begin
v_nr_produktu := fn_daj_numer_produktu;
v_cena_produktu := fn_daj_cene_produktu(v_nr_produktu);
v_ilosc_produktu := fn_daj_ilosc;
insert into faktura_detale(nr_faktury, pozycja_faktury, id_produktu,
ilosc_zakupu, cena_zakupu)
values(v_nr_faktury, v_pozycja_faktury, v_nr_produktu, v_ilosc_produktu,
v_cena_produktu);
update faktura_naglowek
set wartosc_faktury = nvl(wartosc_faktury, 0) +
v_cena_produktu*v_ilosc_produktu
where nr_faktury = v_nr_faktury;
null;
end;
```

12. Tworzymy procedurę, która wygeneruje nam dane na temat kto kupił oraz ustawiamy początkową wartość faktury na 0

```
-- generuj nagłówek, czyli podstawowe dane, takie jak kto kupił, kiedy
create or replace procedure pr_generuj_naglowek(v_data_faktury date
default sysdate, v_nr_faktury number) as
v_nr_klienta number;
begin
v_nr_klienta := FN_DAJ_NR_KLIENTA;
insert into faktura_naglowek(nr_faktury, nr_klienta, data_faktury,
wartosc_faktury) values(v_nr_faktury, v_nr_klienta, v_data_faktury, 0);
null;
end;
```

13. Tworzymy procedurę, która wygeneruje nam wszystkie pozycje, które ktoś kupił

```
-- generuj detale, czyli co kupił, ile sztuk itp
create or replace procedure pr_generuj_detale(v_nr_faktury number) as
v_ile_pozycji number;
begin
v_ile_pozycji := fn_daj_liczbe_pozycji;
for v_pozycja_faktury in 1 .. v_ile_pozycji
loop
pr_generuj_pozycje(v_nr_faktury, v_pozycja_faktury);
end loop;
end;
```

14. Tworzymy procedurę, która wygeneruje nam wszystkie poprzednie rzeczy (zamykamy poprzednie procedury w jednej)

```
-- generuj całą transakcję, czyli nagłówek, detale
create or replace procedure pr_generuj_transakcje(v_data date default
sysdate) as
v_nr_faktury number;
begin
v_nr_faktury := sq_nowa_transakcja.nextval;
pr_generuj_naglowek(v_data, v_nr_faktury);
pr_generuj_detale(v_nr_faktury);
end;
```

15. Wywołujemy procedurę, np. 20 razy

```
-- dodawanie faktur
begin
for i in 1..20 loop
pr_generuj_transakcje();
end loop;
end;
```

16. Tworzymy widok, który pokaże nam dokładnie kto dokonał w hurtowni największych zakupów pod względem wydanych pieniędzy

```
-- kto najwięcej kupił
create or replace view vw_klienci("Imię", "Nazwisko", "Wartość sprzedaży")
as
select klient.imie, klient.nazwisko ,
sum(faktura_naglowek.wartosc_faktury) from faktura_naglowek
join klient on klient.nr_klienta = faktura_naglowek.nr_klienta
group by klient.imie, klient.nazwisko
order by sum(faktura_naglowek.wartosc_faktury) desc;
```

17. Tworzymy widok, który pokaże nam, które województwa przynoszą największe pieniądze (przydatne, gdy np. Będziemy otwierać nową hurtownię)

```
-- klienci z których województw kupują najwięcej
create or replace view vw_wojewodztwa("Województwo", "Wartość sprzedaży")
as
select wojewodztwa.nazwa_wojewodztwa,
sum(faktura_naglowek.wartosc_faktury) from faktura_naglowek
join klient on klient.nr_klienta = faktura_naglowek.nr_klienta
join wojewodztwa on wojewodztwa.id_wojewodztwa = klient.fk_id_wojewodztwa
group by wojewodztwa.nazwa_wojewodztwa
order by sum(faktura_naglowek.wartosc_faktury) desc;
```

18. Tworzymy widok, który pokaże nam, które kategorie przynoszą największe pieniądze

```
-- które kategorie są najchętniej kupowane
create or replace view vw_kategorie("Kategoria", "Wartość sprzedaży") as
select kategoria.nazwa_kategorii, sum(faktura_naglowek.wartosc_faktury)
from faktura_naglowek
join faktura_detale on faktura_detale.nr_faktury =
faktura_naglowek.nr_faktury
join towar on towar.id_produktu = faktura_detale.id_produktu
join kategoria on kategoria.id_kategorii = towar.fk_id_kategorii
group by kategoria.nazwa_kategorii
order by sum(faktura_naglowek.wartosc_faktury) desc;
```

19. Tworzymy widok, który pokaże nam, które przedmioty przynoszą największe pieniądze

```
-- co najchętniej kupują klienci, przynoszą największe pieniądze
create or replace view vw_produkt("Produkt", "Wartość sprzedaży") as
select towar.nazwa, sum(faktura_naglowek.wartosc_faktury) from
faktura_naglowek
join faktura_detale on faktura_detale.nr_faktury =
faktura_naglowek.nr_faktury
join towar on towar.id_produktu = faktura_detale.id_produktu
group by towar.nazwa
order by sum(faktura_naglowek.wartosc_faktury) desc;
```

20. Tworzymy widok, który pokaże nam, które przedmioty są najczęsciej kupowane (pod względem ilościowym)

```
-- który towar został sprzedany w największej ilości
create or replace view vw_ilosc_towar("Nazwa produktu", "Ilość sprzedanych
sztuk") as
select towar.nazwa, sum(faktura_detale.ilosc_zakupu) from faktura_detale
join towar on towar.id_produktu = faktura_detale.id_produktu
group by towar.nazwa
order by sum(faktura_detale.ilosc_zakupu) desc;
```

Instrukcja instalacji projektu i sprawdzenia jego poprawności

Aby wykonać projekt należy wkleić poniższy kod SQL do narzędzia Oracle SQL Developer, a następnie odpytać poszczególne tabele, które wyświetlą nam zawartość tabel, polecenami:

```
SELECT * FROM KLIENT;
SELECT * FROM TOWAR;
SELECT * FROM FAKTURA_NAGLOWEK;
SELECT * FROM FAKTURA_DETALE;
SELECT * FROM KATEGORIA;
SELECT * FROM WOJEWODZTWA;
```

WAŻNE!

Jeśli kod SQL nie wykonuje się poprawnie w programie Oracle SQL Developer, proponuje wszystkie triggery, funkcje oraz procedury wykonać “ręcznie”, tj. Zaznaczyć trigger, funkcję oraz procedury pojedynczo oraz kliknąć Run Statement (Ctrl + Enter)

```
-- HURTOWNIA SPORTOWA
-- JAKUB KARALUS 380950
CREATE TABLE faktura_detale (
 ilosc_zakupu NUMBER NOT NULL,
 cena_zakupu NUMBER NOT NULL,
 id_produktu NUMBER NOT NULL,
 nr_faktury NUMBER NOT NULL,
 pozycja_faktury NUMBER NOT NULL
);
ALTER TABLE faktura_detale ADD CONSTRAINT faktura_detale_pk PRIMARY KEY (
nr_faktury,

pozycja_faktury );
CREATE TABLE faktura_naglowek (
 nr_faktury NUMBER NOT NULL,
 data_faktury DATE NOT NULL,
 wartosc_faktury NUMBER NOT NULL,
 nr_klienta NUMBER NOT NULL
);
ALTER TABLE faktura_naglowek ADD CONSTRAINT faktura_naglowek_pk PRIMARY
KEY ( nr_faktury );
CREATE TABLE klient (
 nr_klienta NUMBER NOT NULL,
 imie VARCHAR2(16) NOT NULL,
 nazwisko VARCHAR2(16) NOT NULL,
 nip NUMBER(11) NOT NULL,
 mail VARCHAR2(32),
 telefon NUMBER(9) NOT NULL,
```

```
 województwo VARCHAR2(30) NOT NULL
);
ALTER TABLE klient ADD CONSTRAINT klient_pk PRIMARY KEY ( nr_klienta );
CREATE TABLE towar (
 id_produktu NUMBER NOT NULL,
 nazwa VARCHAR2(32) NOT NULL,
 cena_sprzedazy NUMBER(8, 2) NOT NULL,
 ilosc NUMBER(16) NOT NULL
);
ALTER TABLE towar ADD CONSTRAINT towar_pk PRIMARY KEY ( id_produktu );
ALTER TABLE faktura_detale
 ADD CONSTRAINT faktura_naglowek_fk FOREIGN KEY ( nr_faktury )
 REFERENCES faktura_naglowek ( nr_faktury );
ALTER TABLE faktura_naglowek
 ADD CONSTRAINT klient_fk FOREIGN KEY ( nr_klienta )
 REFERENCES klient ( nr_klienta );
ALTER TABLE faktura_detale
 ADD CONSTRAINT towar_fk FOREIGN KEY ( id_produktu )
 REFERENCES towar ( id_produktu );
-- tabela slownikowa wojewodztwa
create table WOJEWODZTWA(
id_wojewodztwa number(2) primary key,
nazwa_wojewodztwa varchar(30) not null
);
alter table klient
drop column województwo;
alter table klient
add fk_id_wojewodztwa number(2) not null;
alter table klient
add constraint fk_wojewodztwo foreign key (fk_id_wojewodztwa) references
WOJEWODZTWA(id_wojewodztwa);
-- tabela slownikowa kategoria
create table KATEGORIA(
id_kategorii number(2) primary key,
nazwa_kategorii varchar(30) not null
);
alter table towar
add fk_id_kategorii number(2) not null;
alter table towar
add constraint fk_kategoria foreign key (fk_id_kategorii) references
KATEGORIA(id_kategorii);
-- inserty do kategorii
insert into KATEGORIA VALUES(
1,
'Bluzy'
);
insert into KATEGORIA VALUES(
2,
'Spodnie'
);
insert into KATEGORIA VALUES(
3,
```

```
'Spodenki'
);
insert into KATEGORIA VALUES(
4,
'Podkoszulki'
);
insert into KATEGORIA VALUES(
5,
'Odzież termoaktywna'
);
insert into KATEGORIA VALUES(
6,
'Czapki'
);
insert into KATEGORIA VALUES(
7,
'Kominy'
);
insert into KATEGORIA VALUES(
8,
'Skarpetki'
);
insert into KATEGORIA VALUES(
9,
'Bielizna'
);
insert into KATEGORIA VALUES(
10,
'Rękawiczki'
);
insert into KATEGORIA VALUES(
11,
'Buty do biegania'
);
insert into KATEGORIA VALUES(
12,
'Buty do koszykówki'
);
insert into KATEGORIA VALUES(
13,
'Rakiety'
);
-- inserty do wojewodztw
insert into WOJEWODZTWA values(
1,
'dolnośląskie'
);
insert into WOJEWODZTWA values(
2,
'kujawsko-pomorskie'
);
insert into WOJEWODZTWA values(
```

```
3,
'lubelskie'
);
insert into WOJEWODZTWA values(
4,
'lubuskie'
);
insert into WOJEWODZTWA values(
5,
'łódzkie'
);
insert into WOJEWODZTWA values(
6,
'małopolskie'
);
insert into WOJEWODZTWA values(
7,
'mazowieckie'
);
insert into WOJEWODZTWA values(
8,
'opolskie'
);
insert into WOJEWODZTWA values(
9,
'podkarpackie'
);
insert into WOJEWODZTWA values(
10,
'podlaskie'
);
insert into WOJEWODZTWA values(
11,
'pomorskie'
);
insert into WOJEWODZTWA values(
12,
'śląskie'
);
insert into WOJEWODZTWA values(
13,
'świętokrzyskie'
);
insert into WOJEWODZTWA values(
14,
'warmińsko-mazurskie'
);
insert into WOJEWODZTWA values(
15,
'wielkopolskie'
);
insert into WOJEWODZTWA values(
```

```
16,
'zachodniopomorskie'
);
-- inserty do klientow
create sequence sq_nowy_klient;
insert into KLIENT values(
sq_nowy_klient.nextval,
'Jan',
'Abacki',
11111111111,
'jan@abacki.com',
123456789,
1
);
insert into KLIENT values(
sq_nowy_klient.nextval,
'Adam',
'Bbacki',
11111111112,
'adam@babacki.com',
123456780,
2
);
insert into KLIENT values(
sq_nowy_klient.nextval,
'Kamil',
'Obacki',
11111111113,
'kamil@obacki.pl',
123456781,
3
);
insert into KLIENT values(
sq_nowy_klient.nextval,
'Piotrek',
'Klebacki',
11111111114,
'piotrek@klebacki.com',
123456782,
4
);
insert into KLIENT values(
sq_nowy_klient.nextval,
'Józef',
'Opolański',
11111111115,
'jozef@opolanski.edu.pl',
123456783,
5
);
-- inserty do produktow
create sequence sq_nowy_towar;
```

```
insert into TOWAR values(
sq_nowy_towar.nextval,
'Bluza NIKE 001',
149.99,
2000,
1
);
insert into TOWAR values(
sq_nowy_towar.nextval,
'Bluza NIKE 002',
199.99,
500,
1
);
insert into TOWAR values(
sq_nowy_towar.nextval,
'Bluza ADIDAS 001',
99.99,
2500,
1
);
insert into TOWAR values(
sq_nowy_towar.nextval,
'Spodnie dresowe NIKE 001',
249.99,
8000,
2
);
insert into TOWAR values(
sq_nowy_towar.nextval,
'Spodnie dresowe ADIDAS 001',
149.99,
6000,
2
);
insert into TOWAR values(
sq_nowy_towar.nextval,
'Spodnie krótkie ADIDAS 001',
79.99,
2000,
3
);
insert into TOWAR values(
sq_nowy_towar.nextval,
'Podkoszulka Reebok 001',
79.99,
2000,
4
);
insert into TOWAR values(
sq_nowy_towar.nextval,
'Czapka PUMA 001',
```

```
39.99,
12000,
6
);
-- trigger gdy ktos dodaje nowy produkt do swojej transakcji
create or replace TRIGGER TR_FAKTURA_DETALE_ZMIANA
BEFORE INSERT OR UPDATE OR DELETE ON FAKTURA_DETALE
FOR EACH ROW
BEGIN
 if INSERTING then
 update TOWAR
 set ilosc = ilosc - :NEW.ilosc_zakupu
 where id_produktu = :NEW.id_produktu;
 end if;
 if UPDATING then
 update TOWAR
 set ilosc = ilosc + :OLD.ilosc_zakupu - :NEW.ilosc_zakupu
 where id_produktu = :NEW.id_produktu;
 end if;
 if DELETING then
 update TOWAR
 set ilosc = ilosc + :OLD.ilosc_zakupu
 where id_produktu = :OLD.id_produktu;
 end if;
END;
-- generowanie transakcji
create sequence sq_nowa_transakcja;
-- losuj ile ktos kupil pozycji
create or replace function fn_daj_liczbe_pozycji return number
as
v_ile number;
begin
select round(dbms_random.value(1, 10)) into v_ile from dual;
return v_ile;
end;
-- losuj ile ktos kupil sztuk
create or replace function fn_daj_ilosc return number
as
v_ile number;
begin
select round(dbms_random.value(1, 100)) into v_ile from dual;
return v_ile;
end;
-- losuj kto kupil
create or replace function fn_daj_nr_klienta return number
as
v_nr number;
begin
select nr_klienta into v_nr from (select nr_klienta from klient order by
dbms_random.value) where rownum = 1;
return v_nr;
end;
```

```
-- losuj co kupil(produkt)
create or replace function fn_daj_numer_produktu return number
as
v_id number;
begin
select id_produktu into v_id from
 (select id_produktu from towar
 order by dbms_random.value)
 where rownum = 1;
return v_id;
end;
-- daj cene produktu wylosowanego
create or replace function fn_daj_cene_produktu(v_id_produktu number)
return number
as
v_cena number;
begin
select cena_sprzedazy into v_cena from towar where id_produktu =
v_id_produktu;
return v_cena;
end;
-- procedury:
-- generuj pozycje, czyli kupno jednego towaru
create or replace procedure pr_generuj_pozycje(v_nr_faktury number,
v_pozycja_faktury number) as
v_nr_produktu number;
v_cena_produktu number;
v_ilosc_produktu number;
begin
v_nr_produktu := fn_daj_numer_produktu;
v_cena_produktu := fn_daj_cene_produktu(v_nr_produktu);
v_ilosc_produktu := fn_daj_ilosc;
insert into faktura_detale(nr_faktury, pozycja_faktury, id_produktu,
ilosc_zakupu, cena_zakupu)
values(v_nr_faktury, v_pozycja_faktury, v_nr_produktu, v_ilosc_produktu,
v_cena_produktu);
update faktura_naglowek
set wartosc_faktury = nvl(wartosc_faktury, 0) +
v_cena_produktu*v_ilosc_produktu
where nr_faktury = v_nr_faktury;
null;
end;
-- generuj nagłówek, czyli podstawowe dane, takie jak kto kupił, kiedy
create or replace procedure pr_generuj_naglowek(v_data_faktury date
default sysdate, v_nr_faktury number) as
v_nr_klienta number;
begin
v_nr_klienta := FN_DAJ_NR_KLIENTA;
insert into faktura_naglowek(nr_faktury, nr_klienta, data_faktury,
wartosc_faktury) values(v_nr_faktury, v_nr_klienta, v_data_faktury, 0);
null;
end;
```

```
-- generuj detale, czyli co kupił, ile sztuk itp
create or replace procedure pr_generuj_detale(v_nr_faktury number) as
v_ile_pozycji number;
begin
v_ile_pozycji := fn_daj_liczbe_pozycji;
for v_pozycja_faktury in 1 .. v_ile_pozycji
loop
pr_generuj_pozycje(v_nr_faktury, v_pozycja_faktury);
end loop;
end;
-- generuj całą transakcję, czyli nagłówek, detale
create or replace procedure pr_generuj_transakcje(v_data date default
sysdate) as
v_nr_faktury number;
begin
v_nr_faktury := sq_nowa_transakcja.nextval;
pr_generuj_naglowek(v_data, v_nr_faktury);
pr_generuj_detale(v_nr_faktury);
end;
-- dodawanie faktur
begin
for i in 1..20 loop
pr_generuj_transakcje();
end loop;
end;
-- kto najwięcej kupił
create or replace view vw_klienci("Imię", "Nazwisko", "Wartość sprzedaży")
as
select klient.imie, klient.nazwisko ,
sum(faktura_naglowek.wartosc_faktury) from faktura_naglowek
join klient on klient.nr_klienta = faktura_naglowek.nr_klienta
group by klient.imie, klient.nazwisko
order by sum(faktura_naglowek.wartosc_faktury) desc;
-- klienci z których województw kupują najwięcej
create or replace view vw_wojewodztwa("Województwo", "Wartość sprzedaży")
as
select wojewodztwa.nazwa_wojewodztwa,
sum(faktura_naglowek.wartosc_faktury) from faktura_naglowek
join klient on klient.nr_klienta = faktura_naglowek.nr_klienta
join wojewodztwa on wojewodztwa.id_wojewodztwa = klient.fk_id_wojewodztwa
group by wojewodztwa.nazwa_wojewodztwa
order by sum(faktura_naglowek.wartosc_faktury) desc;
-- które kategorie są najchętniej kupowane
create or replace view vw_kategorie("Kategoria", "Wartość sprzedaży") as
select kategoria.nazwa_kategorii, sum(faktura_naglowek.wartosc_faktury)
from faktura_naglowek
join faktura_detale on faktura_detale.nr_faktury =
faktura_naglowek.nr_faktury
join towar on towar.id_produktu = faktura_detale.id_produktu
join kategoria on kategoria.id_kategorii = towar.fk_id_kategorii
group by kategoria.nazwa_kategorii
order by sum(faktura_naglowek.wartosc_faktury) desc;
```

```
-- co najchętniej kupują klienci, przynoszą największe pieniądze
create or replace view vw_produkt("Produkt", "Wartość sprzedaży") as
select towar.nazwa, sum(faktura_naglowek.wartosc_faktury) from
faktura_naglowek
join faktura_detale on faktura_detale.nr_faktury =
faktura_naglowek.nr_faktury
join towar on towar.id_produktu = faktura_detale.id_produktu
group by towar.nazwa
order by sum(faktura_naglowek.wartosc_faktury) desc;
-- który towar został sprzedany w największej ilości
create or replace view vw_ilosc_towar("Nazwa produktu", "Ilość sprzedanych
sztuk") as
select towar.nazwa, sum(faktura_detale.ilosc_zakupu) from faktura_detale
join towar on towar.id_produktu = faktura_detale.id_produktu
group by towar.nazwa
order by sum(faktura_detale.ilosc_zakupu) desc;
```

Aby zobaczyć, czy widoki działają należy wykonać poniższe polecenia SQL:

```
select * from vw_klienci;
select * from vw_wojewodztwa;
select * from vw_kategorie;
select * from vw_produkt;
select * from vw_ilosc_towar;
```

Skrypt usuwający cały projekt

```
drop table KLIENT cascade constraints; drop table TOWAR cascade constraints; drop table
WOJEWODZTWA cascade constraints; drop table KATEGORIA cascade constraints; drop table
FAKTURA_NAGLOWEK cascade constraints; drop table FAKTURA_DETALE cascade
constraints;drop
function fn_daj_nr_klienta; drop function fn_daj_numer_produktu; drop function
fn_daj_cene_produktu; drop function fn_daj_ilosc_produktu; drop function fn_daj_liczbe_pozycji;
drop procedure pr_generuj_pozycje; drop procedure pr_generuj_naglowek; drop procedure
pr_generuj_detale; drop procedure pr_generuj_transakcje;
drop view vw_ilosc_towar; drop view vw_kategorie; drop view vw_klienci; drop view vw_produkt;
drop
view vw_wojewodztwa;
drop sequence sq_nowy_klient; drop sequence sq_nowy_towar; drop sequence
sq_nowa_transakcja;
drop trigger TR_FAKTURA_DETALE_ZMIANA;
```