Projekt bazy danych dla sklepu internetowego

Przedmiot: **Bazy danych**

Prowadzący: **dr inż. Arkadiusz Mirakowski**

Autor:

Mateusz Motyl

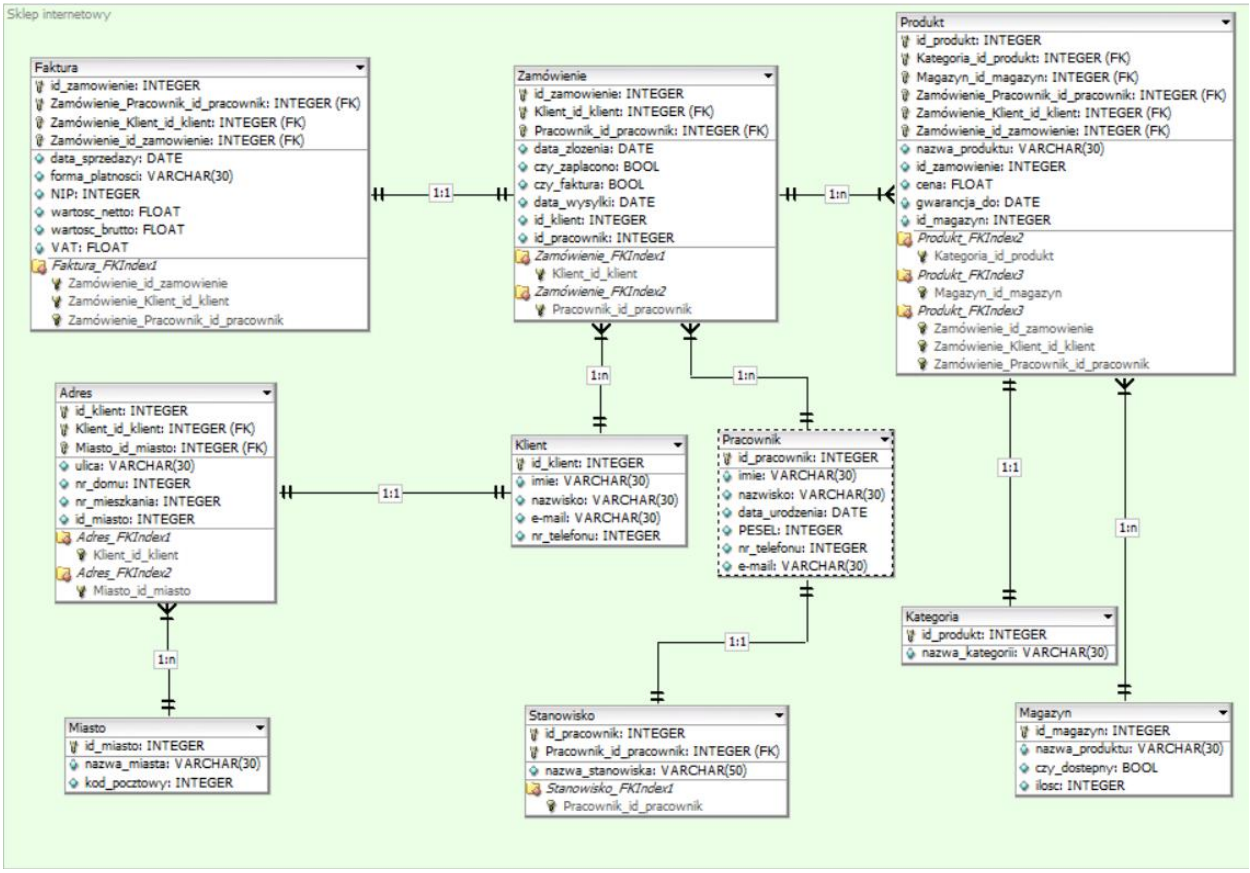
Grupa 3, Informatyka (O), I rok

Gdańsk, 2020

1. Opis bazy danych

Baza danych związana jest z pracą sklepu internetowego. Umożliwia zakup wielu produktów przez jednego kupującego. Dostarcza informacji na temat klienta, jego adresu zamieszkania, województwa, a także dokonanych przez niego zakupów. Baza danych przechowuje również informacje na temat pracownika obsługującego dane zamówienie, jak i stanowisko na którym pracuje. Po dokonaniu zakupów, kupujący może otrzymać (jedną) fakturę. Sklep zawiera informacje na temat dostępności produktów w magazynie i umożliwia podzielenie ich na kategorie.

2. Diagram związków encji – DBDesigner



Rys. 1. Diagram związków encji – DBDesigner4

3. Omówienie tabel

- **Klient** – tabela zawiera dane osobowe kupującego (imie, nazwisko, e-mail, nr telefonu), połączona z tabelą Adres relacją typu 1:1, do jednego klienta przypisany jest jeden adres zamieszkania oraz jest w relacji 1:n z tabelą Zamówienie (kupujący może dokonać wielu zamówień),
- **Adres** – tabela zawiera informacje na temat adresu zamieszkania klienta, połączona jest z tabelą Klient za pomocą kluczy głównych oraz zawiera klucz obcy – id_miasto, wiele adresów przyporządkowanych jest jednemu miastu,
- **Miasto** – tabela zawiera dane na temat nazw miast oraz kodów pocztowych miast zamieszkiwanych przez klientów,

- **Zamówienie** – tabela zawiera dane na temat zamówień złożonych przez klienta, pracownika, który zajmował się danym zamówieniem (klucz obcy – id_pracownik), date złożenia zamówienia czy możliwość wzięcia faktury, jest w relacji 1:1 z tabelą Faktura między kluczami głównymi, przez co do jednego zamówienia można pozyskać jedną fakturę, zamówienie może składać się z kilku produktów (relacja 1:n z Produkt),
- **Produkt** – tabela zawiera dane na temat zamawianych produktów (nazwy, ceny, okresu gwarancji), a także zawiera klucz obcy id_magazyn, w którym przechowywane są produkty, produkt może należeć do jednej kategorii,
- **Kategoria** – tabela zawiera dane odnośnie kategorii, do której należą produkty,
- **Magazyn** – tabela zawiera dane dotyczące dostępności oraz liczbie produktów, magazyn przechowuje wiele produktów,
- **Pracownik** – tabela zawiera dane o pracowniku sklepu internetowego (imie, nazwisko, data urodzenia, PESEL, nr telefonu oraz e-mail),
- **Stanowisko** – tabela zawiera dane na temat stanowiska zajmowanego przez pracownika,
- **Faktura** – tabela zawiera dane potrzebne do wystawienia faktury (NIP, wartość zamówienia czy stawka VAT), a także klucz obcy z numerem zamówienia.

4. Definicje tabel

Klient

nazwa pola	typ pola	inne cechy pola
id_klient	int	PK
imie	varchar(30)	-
nazwisko	varchar(30)	-
e-mail	varchar(30)	-
nr_telefonu	int	-

Adres

nazwa pola	typ pola	inne cechy pola
id_klient	int	PK
ulica	varchar(30)	-
nr_domu	int	-
nr_mieszkania	int	-
id_miasto	int	FK

Miasto

nazwa pola	typ pola	inne cechy pola
id_miasto	int	PK
nazwa_miasta	varchar(30)	-
kod_pocztowy	int	-

Zamówienie

nazwa pola	typ pola	inne cechy pola
id_zamowienie	int	PK
data_zlozenia	date	-
czy_zaplacono	bool	-
czy_faktura	bool	-
data_wysylki	date	-
id_klient	int	FK
id_pracownik	int	FK

Produkt

nazwa pola	typ pola	inne cechy pola
id_produkt	int	PK
nazwa_produktu	varchar(30)	-
id_zamowienie	int	FK
cena	float	-
gwarancja_do	date	-
id_magazyn	int	FK

Kategoria

nazwa pola	typ pola	inne cechy pola
id_produkt	int	PK
nazwa_kategorii	varchar(30)	-

Magazyn

nazwa pola	typ pola	inne cechy pola
id_magazyn	int	PK
nazwa_produktu	varchar(30)	-
czy_dostepny	bool	-
ilosc	int	-

Pracownik

nazwa pola	typ pola	inne cechy pola
id_pracownik	int	PK
imie	varchar(30)	-
nazwisko	varchar(30)	-
data_urodzenia	date	-
PESEL	int	-
nr_telefonu	int	-
e-mail	varchar(30)	-

Stanowisko

nazwa pola	typ pola	inne cechy pola
id_pracownik	int	PK
nazwa_stanowiska	varchar(50)	-

Faktura

nazwa pola	typ pola	inne cechy pola
id_zamowienie	int	PK
data_sprzedazy	date	-
forma_platnosci	varchar(30)	-
NIP	int	-
wartosc_netto	float	-
wartosc_brutto	float	-
VAT	float	-

5. Zawartość tabel

Klient

id_klient	imie	nazwisko	e-mail	nr_telefonu
136	Jan	Kowalski	jan.kowal@wp.pl	543 354 354
141	Maria	Nowak	marino@o2.pl	433 332 222
142	Jacek	Malinowski	maliniak@wp.pl	432 111 222
150	Mariusz	Kaminski	kamyk123@wp.pl	999 888 777

Adres

id_klient	ulica	nr_domu	nr_mieszkania	id_miasto
136	Skarszewska	2	15	15
141	Kościuszki	7	3	9
142	Moniuszki	10	29	31
150	Batorego	6	20	9

Miasto

id_miasto	nazwa_miasta	kod_pocztowy
15	Koszalin	75-025
9	Słupsk	76-200
31	Ustka	76-270

Zamówienie

id_zamowienie	data_zlozenia	czy_zaplacono	czy_faktura	data_wysylki	id_klient	id_pracownik
441	2020/4/11	1	0	2020/4/15	136	75
515	2020/5/10	1	1	2020/5/11	141	75
523	2020/5/11	1	1	2020/5/12	142	81
530	2020/6/10	1	0	2020/6/11	150	75

Produkt

id_produkt	nazwa_produktu	id_zamowienia	cena	gwarancja_do	id_magazyn
53425	rower miejski	441	1476.00	2022/5/10	1
53454	rower górski	441	2153.00	2022/5/10	1
432	komplet sztućców	515	184.50	2021/5/11	7
1234	telewizor LED	523	2399.00	2023/6/10	3
513	radio cyfrowe	530	99.0	2023/6/11	3

Kategoria

id_produkt	nazwa_kategorii
53425	rowery
53454	rowery
432	artykuły domowe
1234	telewizory
513	radia

Magazyn

id_magazyn	nazwa_produktu	czy_dostepny	ilosc
1	rower miejski	1	7
1	rower górski	1	12
7	komplet sztućców	1	23
3	telewizor LED	1	100
3	radio cyfrowe	1	50

Pracownik

id_pracownik	imie	nazwisko	data_urodzenia	PESEL	nr_telefonu	e-mail
75	Mariusz	Bąk	1999/3/1	994343	423 432 432	baku@wp.pl
81	Monika	Zielonka	1996/5/12	964252	111 222 333	zielona@wp.pl

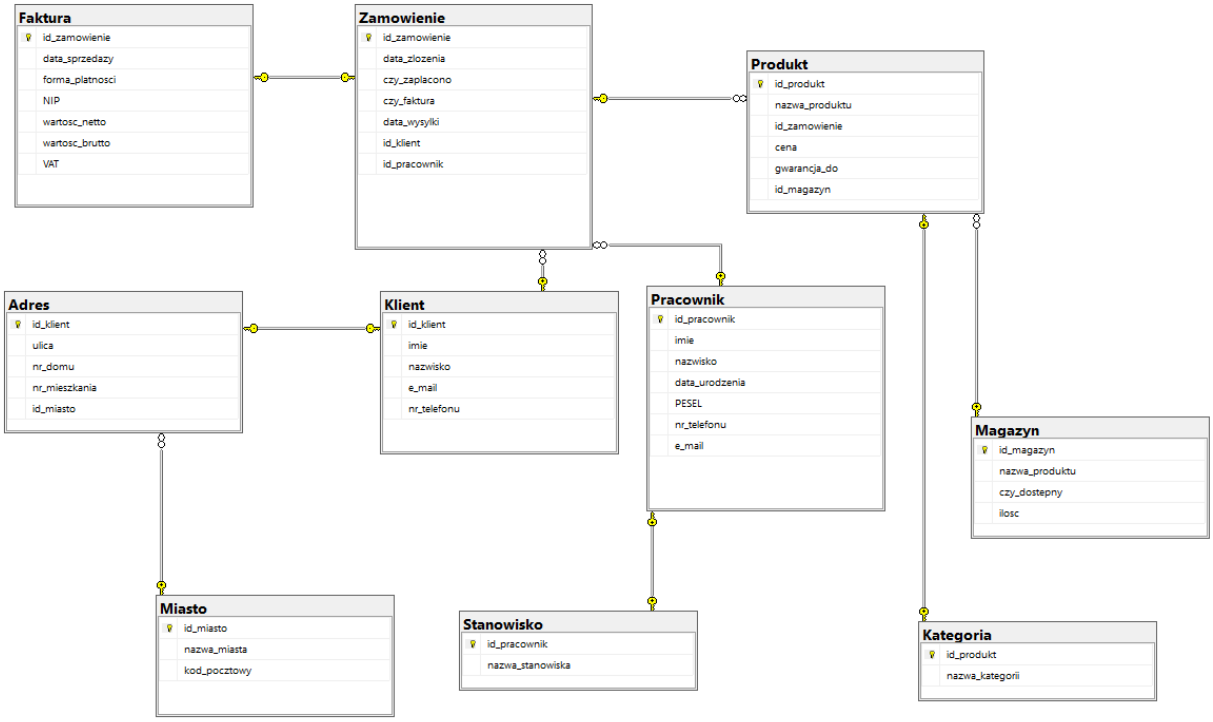
Stanowisko

id_pracownik	nazwa_stanowiska
75	manager sklepu
81	specjalista ds. obsługi klienta

Faktura

id_zamowienie	data_sprzedazy	forma_platnosci	NIP	wartosc_netto	wartosc_brutto	VAT
515	2020/5/10	gotówka	1234	150.00	184.50	23%
523	2020/5/11	gotówka	6542	1950.41	2399.00	23%

6. Diagram związków encji – MS SQL Server



Rys. 2. Diagram związków encji – MS SQL Server

```
--DROP TABLE IF EXISTS Klient

CREATE TABLE Klient

(

id_klient int primary key identity,

imie varchar(30),

nazwisko varchar(30),

e_mail varchar(30),

nr_telefonu int,

)


--DROP TABLE IF EXISTS Miasto

CREATE TABLE Miasto

(

id_miasto int primary key identity,

nazwa_miasta varchar(30),

kod_pocztowy int

)
```

```
--DROP TABLE IF EXISTS Adres
CREATE TABLE Adres
(
id_klient int primary key identity,
ulica varchar(30),
nr_domu int,
nr_mieszkania int,
id_miasto int,
foreign key (id_miasto) references Miasto(id_miasto)
)

ALTER TABLE Klient
ADD CONSTRAINT FK_Klient foreign key (id_klient)
REFERENCES Adres(id_klient)


--DROP TABLE IF EXISTS Stanowisko
CREATE TABLE Stanowisko
(
id_pracownik int primary key identity,
nazwa_stanowiska varchar(50)
)

--DROP TABLE IF EXISTS Pracownik
CREATE TABLE Pracownik
(
id_pracownik int primary key identity,
imie varchar(30),
nazwisko varchar(30),
data_urodzenia date,
PESEL int,
nr_telefonu int,
e_mail varchar(30)
)
```

```
ALTER TABLE Stanowisko
ADD CONSTRAINT FK_Stanowisko foreign key (id_pracownik)
REFERENCES Pracownik(id_pracownik)

--DROP TABLE IF EXISTS Magazyn
CREATE TABLE Magazyn
(
id_magazyn int primary key identity,
nazwa_produktu varchar(30),
czy_dostepny bit, -- bool
ilosc int
)

--DROP TABLE IF EXISTS Kategoria
CREATE TABLE Kategoria
(
id_produkt int primary key identity,
nazwa_kategorii varchar(30)
)

--DROP TABLE IF EXISTS Zamowienie
CREATE TABLE Zamowienie
(
id_zamowienie int primary key identity,
data_zlozenia date,
czy_zaplacono bit,
czy_faktura bit,
data_wysylki date,
id_klient int,
id_pracownik int,
foreign key (id_klient) references Klient(id_klient),
foreign key (id_pracownik) references Pracownik(id_pracownik)
)
```

```
--DROP TABLE IF EXISTS Produkt
CREATE TABLE Produkt
(
id_produkt int primary key identity,
nazwa_produktu varchar(30),
id_zamowienie int,
cena float,
gwarancja_do date,
id_magazyn int,
foreign key (id_zamowienie) references Zamowienie(id_zamowienie),
foreign key (id_magazyn) references Magazyn(id_magazyn)
)

ALTER TABLE Produkt
ADD CONSTRAINT FK_Produkt foreign key (id_produkt)
REFERENCES Kategoria(id_produkt)

--      DROP TABLE IF EXISTS Faktura
CREATE TABLE Faktura
(
id_zamowienie int primary key identity,
data_sprzedazy date,
forma_platnosci varchar(30),
NIP int,
wartosc_netto float,
wartosc_brutto float,
VAT float
)

ALTER TABLE Faktura
ADD CONSTRAINT FK_Faktura foreign key (id_zamowienie)
REFERENCES Zamowienie(id_zamowienie)
```

7. Omówienie i prezentacja widoków

dane_klienta – jest to widok, który zawiera dane na temat klienta połączone z trzech tabel (Klient, Adres, Miasto),

```
CREATE VIEW dane_klienta AS
SELECT imie, nazwisko, nr_telefonu, ulica, nr_domu, nr_mieszkania, nazwa_miasta,
kod_pocztowy
FROM Klient INNER JOIN Adres ON Klient.id_klient = Adres.id_klient
INNER JOIN Miasto ON Adres.id_miasto = Miasto.id_miasto
```

imie	nazwisko	nr_telefonu	ulica	nr_domu	nr_mieszkania	nazwa_miasta	kod_pocztowy
Jan	Kowalski	543354354	Skarszewska	2	15	Koszalin	75025
Maria	Nowak	433332222	Kosciuszki	7	3	Slupsk	76200
Jacek	Malinowski	432111222	Moniuszki	10	29	Ustka	76270
Mariusz	Kaminski	999888777	Batorego	6	20	Slupsk	76200

Rys. 3. Wynik działania widoku **dane_klienta**

statystyki_miasto – jest to widok przedstawiający ilość klientów w poszczególnych miastach,

```
CREATE VIEW statystyki_miasto AS
SELECT COUNT(*) AS 'ilosc klientow', nazwa_miasta AS 'miasto'
FROM Klient INNER JOIN Adres ON Klient.id_klient = Adres.id_klient
INNER JOIN Miasto ON Adres.id_miasto = Miasto.id_miasto
GROUP BY nazwa_miasta
```

ilosc klientow	miasto
1	Koszalin
2	Slupsk
1	Ustka

Rys. 4. Wynik działania widoku **statystyki_miasto**

zamowienie_pracownik – jest to widok, zawierający informacje na temat pracownika i zamówień przez niego obsługiwanych.

```
CREATE VIEW zamowienie_pracownik AS
SELECT imie, nazwisko, nazwa_stanowiska, id_zamowienie,data_zlozenia, data_wysylki,
czy_zaplacono, czy_faktura
FROM Pracownik INNER JOIN Stanowisko ON Pracownik.id_pracownik = Stanowisko.id_pracownik
INNER JOIN Zamowienie ON Zamowienie.id_pracownik = Pracownik.id_pracownik
```

imie	nazwisko	nazwa_stanowiska	id_zamowienie	data_zlozenia	data_wysylki	czy_zaplacono	czy_faktura
Mariusz	Bak	manager sklepu	441	2020-04-11	2020-04-15	1	0
Mariusz	Bak	manager sklepu	515	2020-05-10	2020-05-11	1	1
Monika	Zielonka	specjalista ds. obslugi klienta	523	2020-05-11	2020-05-12	1	1
Mariusz	Bak	manager sklepu	530	2020-06-10	2020-06-11	1	0

Rys. 5. Wynik działania widoku **zamowienie_pracownik**

8. Omówienie i prezentacja funkcji

dbo.f1 – jest to funkcja, która zwraca wynik tabelaryczny, w zależności od zadanego argumentu wyświetla wszystkie zamówienia z danego miesiąca,

```
CREATE FUNCTION f1(@miesiac int)
RETURNS TABLE AS
RETURN (SELECT*FROM Zamowienie WHERE MONTH(data_zlozenia) = @miesiac)
```

id_zamowienie	data_zlozenia	czy_zaplacono	czy_faktura	data_wysylki	id_klient	id_pracownik
515	2020-05-10	1	1	2020-05-11	141	75
523	2020-05-11	1	1	2020-05-12	142	81

Rys. 6. Wynik działania funkcji **dbo.f1(5)**

dbo.f2 – jest to funkcja, która przyjmuje dwa argumenty (imie, nazwisko) oraz zwraca ilość produktów zamówionych przez danego klienta,

```
CREATE FUNCTION f2(@imie varchar(30), @nazwisko varchar(30))
RETURNS int
BEGIN
DECLARE @ilosc int
SET @ilosc = (SELECT COUNT(id_produkt) FROM Zamowienie
INNER JOIN Klient ON Zamowienie.id_klient = Klient.id_klient
INNER JOIN Produkt ON Zamowienie.id_zamowienie = Produkt.id_zamowienie
WHERE imie = @imie AND nazwisko = @nazwisko)
RETURN @ilosc
END
```

Ilosc zamowionych produktow
2

Rys. 7. Wynik działania funkcji **dbo.f2('Jan', 'Kowalski')**

dbo.f3 – jest to funkcja, która zwraca sumę zamówienia złożonego przez klienta, a jako argument przyjmuje jego numer (id_klient).

```
CREATE FUNCTION f3(@id_klient int)
RETURNS float
BEGIN
DECLARE @suma float
SET @suma = (SELECT SUM(cena) FROM Produkt INNER JOIN Zamowienie
ON Produkt.id_zamowienie = Zamowienie.id_zamowienie
WHERE Zamowienie.id_klient = @id_klient)
RETURN @suma
END
```

Łączna suma zakupow
3629

Rys. 8. Wynik działania funkcji **dbo.f3(136)**

9. Omówienie i prezentacja wyzwalaczy

wyz01 – jest to wyzwalacz podłączony do tabeli Produkt, który w wyniku dodania do zamówienia nowych produktów nalicza zniżkę 10%, jeżeli te osobno kosztują co najmniej 150 złotych (jedno zamówienie – jedna łączna zniżka)

```
CREATE TRIGGER wyz01
ON Produkt
FOR INSERT, DELETE
AS
DECLARE @znizka float = 0.1
DECLARE @id_ostatni int = (SELECT TOP 1 id_zamowienie FROM Produkt ORDER BY id_zamowienie
DESC)
DECLARE @cena_ostatni float = (SELECT SUM(cena) FROM Produkt WHERE id_zamowienie =
@id_ostatni)
DECLARE @znizka_ostatni float = @cena_ostatni * @znizka
IF @cena_ostatni >= 150 UPDATE Produkt SET znizka = @znizka_ostatni WHERE id_zamowienie =
@id_ostatni
```

id_produkt	nazwa_produktu	id_zamowienie	cena	gwarancja_do	id_magazyn	znizka
53425	rower miejski	441	1476	2022-05-10	1	NULL
53454	rower gorski	441	2153	2022-05-10	1	NULL
432	komplet sztuccow	515	184,5	2021-05-11	7	NULL
1234	telewizor LED	523	2399	2023-06-10	3	NULL
513	radio cyfrowe	530	99	2023-06-11	3	NULL

Rys. 9. Dodanie nowej kolumny **zniżnka**

id_produkt	nazwa_produktu	id_zamowienie	cena	gwarancja_do	id_magazyn	znizka
53425	rower miejski	441	1476	2022-05-10	1	NULL
53454	rower gorski	441	2153	2022-05-10	1	NULL
432	komplet sztuccow	515	184,5	2021-05-11	7	NULL
1234	telewizor LED	523	2399	2023-06-10	3	NULL
513	radio cyfrowe	530	99	2023-06-11	3	NULL
495	szczoteczka soniczna	535	170	2021-07-12	4	37
113	sluchawki douszne	535	200	2021-03-17	8	37

Rys. 10. Wynik działania wyzwalacza **wyz01**

wyz02 – jest to wyzwalacz podłączony do tabeli Faktura. Przyznaje klientowi, który zdecydował się na zakupy po 20 dniu danego miesiąca oraz wzięcie faktury, bon na następne zakupy (wartość zakupów przemnożona przez 0.10) ,

```
CREATE TRIGGER wyz02
ON Faktura
FOR INSERT, DELETE
AS
DECLARE @przelicznik float = 0.10
DECLARE @id_ostatni int = (SELECT TOP 1 id_zamowienie FROM FAKTURA ORDER BY id_zamowienie
DESC)
DECLARE @data_ostatni int = (SELECT DAY(data_sprzedazy) FROM Faktura WHERE id_zamowienie
= @id_ostatni)
DECLARE @wartosc_ostatni float = (SELECT wartosc_netto FROM Faktura WHERE id_zamowienie =
@id_ostatni)
IF @data_ostatni > 20 UPDATE Faktura SET bon = @wartosc_ostatni*@przelicznik WHERE
id_zamowienie = @id_ostatni
```

id_zamowienie	data_sprzedazy	forma_platnosci	NIP	wartosc_netto	wartosc_brutto	VAT	bon
515	2020-05-10	gotowka	1234	150	184,5	23	NULL
523	2020-05-11	gotowka	6542	1950,41	2399	23	NULL

Rys. 10. Dodanie nowej kolumny **bon**

id_zamowienie	data_sprzedazy	forma_platnosci	NIP	wartosc_netto	wartosc_brutto	VAT	bon
515	2020-05-10	gotowka	1234	150	184,5	23	NULL
523	2020-05-11	gotowka	6542	1950,41	2399	23	NULL
550	2020-05-21	gotowka	9182	320	393,6	23	32

Rys. 11. Wynik działania wyzwalacza **wyz02**

wyz03 – jest to wyzwalacz podłączony do tabeli Stanowisko, definiuje stanowisko każdego nowego pracownika jako okres próbny

```
CREATE TRIGGER wyz03
ON Stanowisko
FOR INSERT, DELETE
AS
DECLARE @nazwa_stanowiska varchar(30) = 'okres próbny'
DECLARE @id_ostatni int = (SELECT TOP 1 id_pracownik FROM Stanowisko ORDER BY
id_pracownik DESC)
UPDATE Stanowisko SET nazwa_stanowiska = @nazwa_stanowiska WHERE id_pracownik =
@id_ostatni

--test
INSERT INTO Stanowisko VALUES
(90, NULL)
```

id_pracownik	nazwa_stanowiska
75	manager sklepu
81	specjalista ds. obslugi klienta
90	okres próbny

Rys.12. Wynik działania wyzwalacza **wyz03**