Politechnika Wrocławska
Wydział Elektroniki

BAZY DANYCH – PROJEKT

INTERNETOWY SKLEP Z ODZIEŻĄ

Autorzy:

Alicja Myśliwiec 248867
Daria Milczarska 248894

Prowadzący:

Dr inż. Piotr Lechowicz

1. Wstęp

1.1. Opis słowny systemu

W ramach projektu została wykonana aplikacja bazodanowa firmy posiadającej w swoim kapitale odzież. Firma będzie zarządzać zarówno posiadanymi produktami, jak i pracownikami oraz będzie prowadzić sklep internetowy.

Wśród pracowników znajdują się osoby odpowiedzialne za codzienne sprawne funkcjonowanie firmy, począwszy od kierownika, aż po osoby odpowiedzialne za dostarczanie produktów oraz sortowanie i pakowanie. Jest możliwość, aby kilka osób zajmowało to samo stanowisko.

Klientami firmy mogą być osoby zarówno fizyczne jak i inne firmy. Każdy klient będzie miał swoje konto na stronie internetowej, do którego będzie mógł logować się za pomocą loginu i hasła. Klienci mogą kupować dowolną ilość produktów. Istnieje możliwość wystawienia faktury za dokonany zakup.

Firma zajmuje się sprzedażą produktów, które mają określony wygląd i rozmiar oraz są dostępne w pewnej liczbie sztuk. Wygląd każdego produktu zawiera informacje o jego kolorze, przykładowe zdjęcie oraz mówi jakiego modelu jest on odmianą. Każdy model ubrania będzie miał swój opis, jego aktualną cenę oraz informację, do jakiej kategorii ubrań należy. Kategoria będzie nam mówić z jakiego działu (zimowy, letni, damski, męski) jest dana sztuka towaru i jakiego jest rodzaju (spodnie, koszula, sukienka). Prawie każdy rodzaj ubrania (np. kurtka, spodnie) może być z różnego działu, ale niektóre rodzaje będą przypisane tylko do jednego działu (np. sukienka może być tylko damska). Firma prowadzi również spis produktów, które zostały już sprzedane. W takich spisach znajdzie się cena sprzedaży, data, numer klienta, który kupił produkt i wszystkie inne niezbędne informacje.

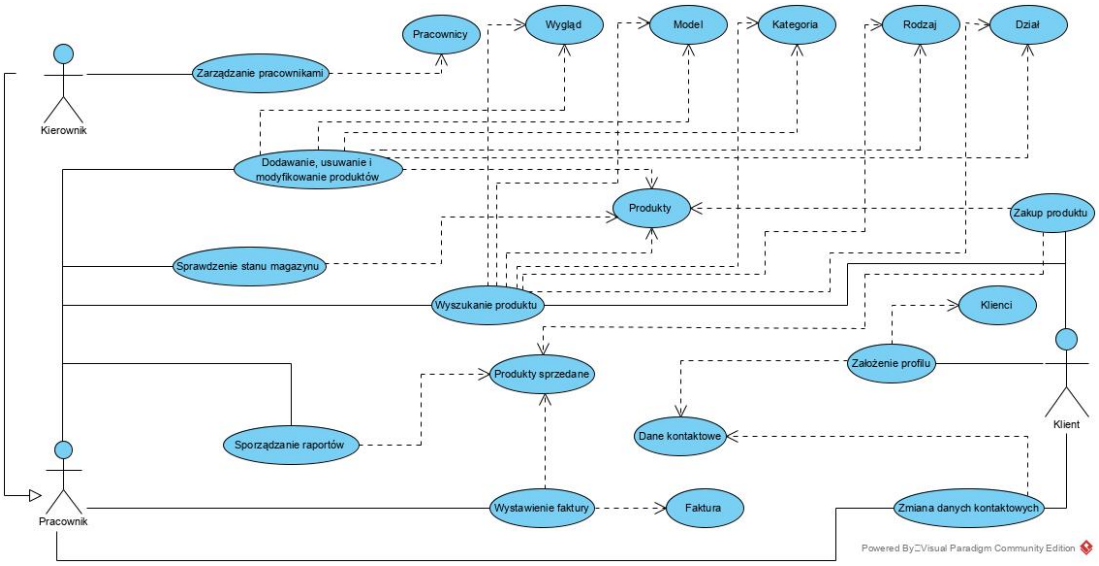
Aby nabyć produkty należy złożyć zamówienie, zatem w odpowiedniej tabeli zebrane będą informacje na temat zamówień poszczególnych klientów. Aby powiązać zamówienia z produktami powstanie tabela szczegóły zamówienia, która przechowywać będzie numery zamówień i numery zamówionych produktów.

Oczywiście w celu podjęcia pracy w firmie lub też założenia konta na stronie internetowej należy podać swoje dane osobowe i kontaktowe, zatem firma powinna w jednym miejscu przechowywać dane kontaktowe zarówno klientów jak i pracowników, takie jak numer telefonu, adres e-mail czy adres zamieszkania.

1.2. Opis wymagań funkcjonalnych

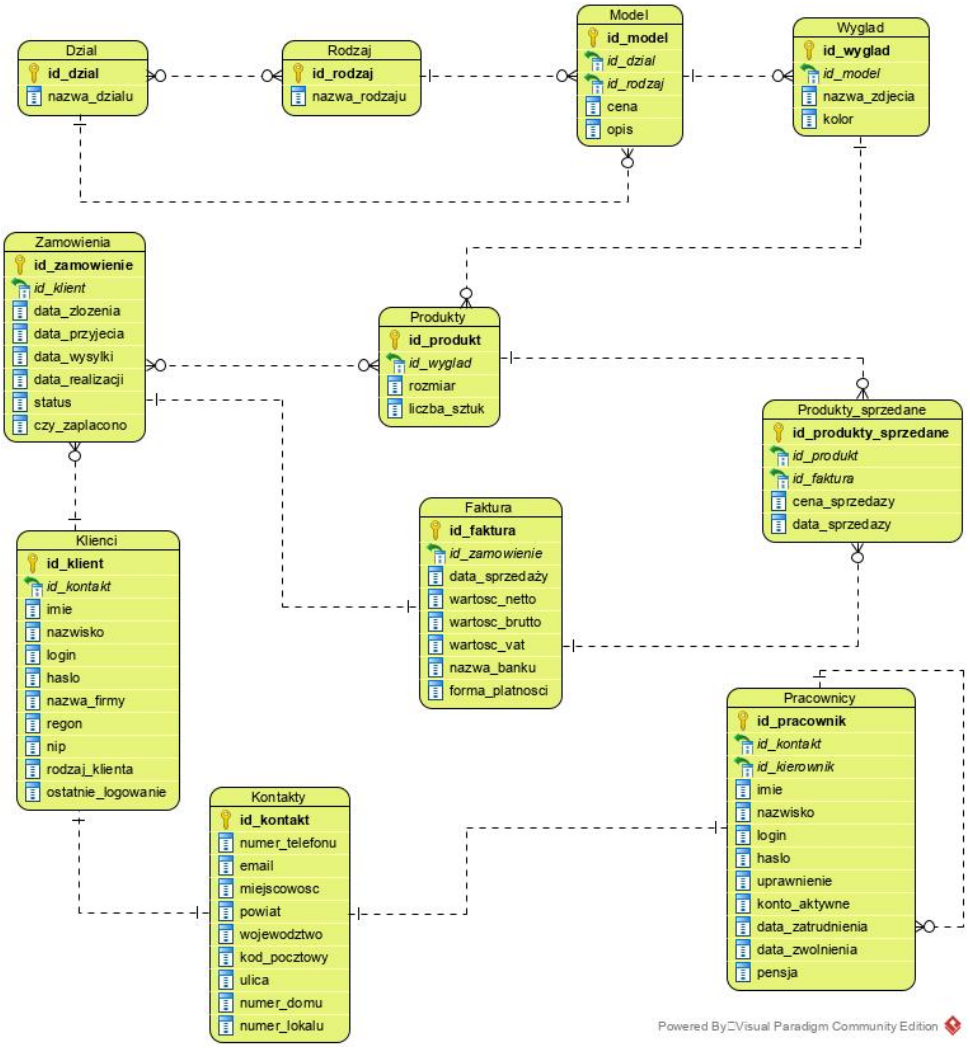
- Aplikacja pozwala na wygodniejszy wgląd do podstawowych informacji o poszczególnych produktach lub osobach
- Szybkie dodawanie i usuwanie sztuk oraz pracowników
- Łatwa zmiana danych kontaktowych
- Wizualizacja ilości sztuk danego rozmiaru lub ilości sztuk sprzedanych w danym miesiącu
- Generowanie zestawień produktów (np. wszystkie damskie koszulki w rozmiarze S o cenie mniejszej niż 50 zł)
- Porównanie sprzedaży (np. z działu damskiego i męskiego)
- Zestawienie łącznego oraz średniego dochodu z danego miesiąca/roku
- Sprawdzenie stanu magazynu
- Założenie konta
- Wystawienie faktury sprzedaży
- Zakup produktu przez klienta

1.3. Przypadki użycia

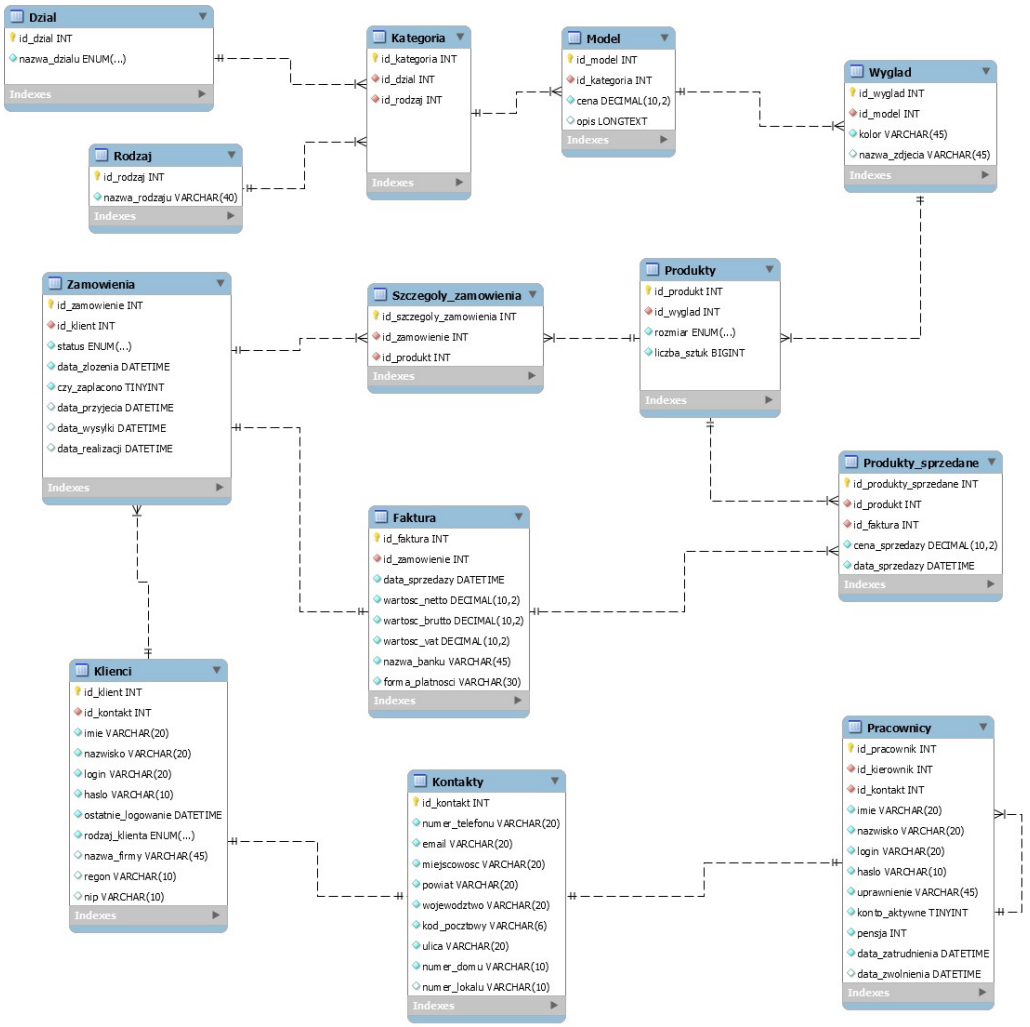


2. Model bazy danych

2.1. Model logiczny



2.2. Model fizyczny



2.3. Opis transakcji

- Tabela *Produkty* – podczas dodawaniu nowego rekordu do tabeli, wcześniej musi zostać utworzony rekord w tabeli *Wyglad*, który będzie powiązany z rekordem w tabeli *Produkty*.
- Tabela *Wyglad* – podczas dodawaniu nowego rekordu do tabeli, wcześniej musi zostać utworzony rekord w tabeli *Model*, który będzie powiązany z rekordem w tabeli *Wyglad*.
- Tabela *Model* – podczas dodawaniu nowego rekordu do tabeli, wcześniej musi zostać utworzony rekord w tabeli *Model*, który będzie powiązany z rekordem w tabeli *Model*.
- Tabela *Zamowienia* – podczas dodawaniu nowego rekordu do tabeli, wcześniej musi zostać utworzony rekord w tabeli *Klienci*, który będzie powiązany z rekordem w tabeli *Zamowienia*.
- Tabela *Klienci* – podczas dodawaniu nowego rekordu do tabeli, wcześniej musi zostać utworzony rekord w tabeli *Kontakty* , który będzie powiązany z rekordem w tabeli *Klienci*.
- Tabela *Faktura* – podczas dodawaniu nowego rekordu do tabeli, wcześniej musi zostać utworzony rekord w tabeli *Zamowienia* , który będzie powiązany z rekordem w tabeli *Faktura*.
- Tabela *Pracownicy* – podczas dodawaniu nowego rekordu do tabeli, wcześniej musi zostać utworzony rekord w tabeli *Kontakty*, który będzie powiązany z rekordem w tabeli *Pracownicy*.
- Tabela *Kategoria* – podczas dodawaniu nowego rekordu do tabeli, wcześniej muszą zostać utworzone rekordy w tabeli *Rodzaj* i w tabeli *Dzial*, które będą powiązane z rekordem w tabeli *Kategoria*.
- Tabela *Szczegoly_zamowienia* – podczas dodawaniu nowego rekordu do tabeli, wcześniej muszą zostać utworzone rekordy w tabeli *Zamowienia* i w tabeli *Produkty*, które będą powiązane z rekordem w tabeli *Szczegoly_zamowienia*.

- Tabela *Produkty_sprzedane* – podczas dodawaniu nowego rekordu do tabeli, wcześniej muszą zostać utworzone rekordy w tabeli *Produkty* i w tabeli *Faktura*, które będą powiązane z rekordem w tabeli *Produkty_sprzedane*.
- Tabela *Klienci* – podczas usuwania rekordu z tabeli, należy również usunąć rekord z tabeli *Kontakty* powiązany z rekordem w tabeli *Klienci.*
- Tabela *Pracownicy* – podczas usuwania rekordu z tabeli, należy również usunąć rekord z tabeli *Kontakty* powiązany z rekordem w tabeli *Pracownicy.*

2.4. Opis widoków

- Widok *Produkt* – id_produkt, id_wyglad, id_model, id_kategoria, id_dzial, id_rodzaj, rozmiar, kolor, cena, opis, nazwa_dzialu, nazwa_rodzaju, liczba_sztuk.
- Widok *Zamowienie* – id_zamowienie, id_klient, imie(klienta), nazwisko(klienta), rodzaj_klienta, status, data_zlozenia, czy_zaplacono.
- Widok *Klient* – id_klient, id_kontakt, imie, nazwisko, rodzaj_klienta, ostatnie_logowanie, numer_telefonu, email.
- Widok *Pracownik* – id_pracownik, id_kierownik, id_kontakt, imie, nazwisko, uprawnienie, konto_aktywne, pensja, data_zatrudnienia.
- Widok *Produkt*_sprzedany– id_produkt, id_faktura, id_model, id_kategoria, id_dzial, id_rodzaj, rozmiar, kolor, cena, opis, nazwa_dzialu, nazwa_rodzaju, cena_sprzedazy, data_sprzedazy.

2.5. Opis wyzwalaczy

- Tabela *Produkty_sprzedane* – podczas dodawaniu nowego rekordu do tabeli, pozycja *liczba_sztuk* (tabela *Produkty*) będzie zmniejszania**.**
- Tabela *Klienci* – podczas usuwania rekordu z tabeli, automatycznie zostaje usunięty rekord z tabeli *Kontakty* powiązany z rekordem w tabeli *Klienci.*
- Tabela *Pracownicy* – podczas usuwania rekordu z tabeli, automatycznie zostaje usunięty rekord z tabeli *Kontakty* powiązany z rekordem w tabeli *Pracownicy.*

2.6. Opis funkcji

- Obliczanie dochodu ze sprzedaży (tabela *Produkty_sprzedane,* sumowanie pozycji *cena_sprzedazy*)
- Sprawdzenie stanu magazynu (tabela *Produkty*, sumowanie pozycji *liczba_sztuk*)
- Obliczanie wartości netto i brutto oraz vat potrzebnych do faktury (tabela *Faktura*, pozycje *wartość_netto, wartosc_brutto, wartość_vat*)
- Generacja średniej płacy pracowników (tabela *Pracownicy*, pozycja *pensja*)

3. Implementacja bazy danych

3.1. Środowisko programistyczne

Do zaimplementowania bazy danych zostało wykorzystane środowisko MySQL wraz z programem do zarządzania bazą danych MySQL Workbench.

3.2. Przykładowe skrypty tworzące bazę danych

- Tworzenie tabeli *Faktura*

```
-- -----------------------------------------------------
-- Table `mydb`.`Faktura`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Faktura` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Faktura` (
  `id_faktura` INT NOT NULL AUTO_INCREMENT,
  `id_zamowienie` INT NOT NULL,
  `data_sprzedazy` DATETIME NOT NULL,
  `wartosc_netto` DECIMAL(10,2) NOT NULL,
  `wartosc_brutto` DECIMAL(10,2) NOT NULL,
  `wartosc_vat` DECIMAL(10,2) NOT NULL,
  `nazwa_banku` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
  `forma_platnosci` VARCHAR(30) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
  PRIMARY KEY (`id_faktura`),
  CONSTRAINT `fk_Faktura_Zamowienia1`
    FOREIGN KEY (`id_zamowienie`)
    REFERENCES `mydb`.`zamowienia` (`id_zamowienie`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_polish_ci;

CREATE INDEX `fk_Faktura_Klienci1_idx` ON `mydb`.`Faktura` (`id_zamowienie` ASC) VISIBLE;
```

- Tworzenie tabeli *Pracownicy*

```
CREATE TABLE IF NOT EXISTS `mydb`.`Pracownicy` (
  `id_pracownik` INT NOT NULL AUTO_INCREMENT,
  `id_kierownik` INT NULL,
  `id_kontakt` INT NOT NULL,
  `imie` VARCHAR(20) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
  `nazwisko` VARCHAR(20) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
  `login` VARCHAR(20) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
  `haslo` VARCHAR(10) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
  `uprawnienie` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
  `konto_aktywne` CHAR(3) NOT NULL,
  `pensja` INT NOT NULL,
  `data_zatrudnienia` DATETIME NOT NULL,
  `data_zwolnienia` DATETIME NULL,
  PRIMARY KEY (`id_pracownik`),
  CONSTRAINT `fk_Pracownicy_Kontakty1`
    FOREIGN KEY (`id_kontakt`)
    REFERENCES `mydb`.`Kontakty` (`id_kontakt`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pracownicy_Pracownicy1`
    FOREIGN KEY (`id_kierownik`)
    REFERENCES `mydb`.`Pracownicy` (`id_pracownik`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_polish_ci;

CREATE INDEX `id_kontakt` ON `mydb`.`Pracownicy` (`id_kontakt` ASC) VISIBLE;

CREATE INDEX `fk_Pracownicy_Pracownicy1_idx` ON `mydb`.`Pracownicy` (`id_kierownik` ASC) VISIBLE;
```

- Tworzenie tabeli *Kontakty*

```
116 -- -----------------------------------------------------
117 -- Table `mydb`.`Kontakty`
118 -- -----------------------------------------------------
119 DROP TABLE IF EXISTS `mydb`.`Kontakty` ;
120
121 CREATE TABLE IF NOT EXISTS `mydb`.`Kontakty` (
122   `id_kontakt` INT NOT NULL AUTO_INCREMENT,
123   `numer_telefonu` VARCHAR(20) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
124   `email` VARCHAR(40) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
125   `miejscowosc` VARCHAR(20) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
126   `powiat` VARCHAR(20) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
127   `wojewodztwo` VARCHAR(20) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
128   `kod_pocztowy` VARCHAR(6) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
129   `ulica` VARCHAR(20) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
130   `numer_domu` VARCHAR(10) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NOT NULL,
131   `numer_lokalu` VARCHAR(10) CHARACTER SET 'utf8' COLLATE 'utf8_polish_ci' NULL,
132   PRIMARY KEY (`id_kontakt`))
133 ENGINE = InnoDB
134 DEFAULT CHARACTER SET = utf8
135 COLLATE = utf8_polish_ci;
136
```

- Uzupełnianie tabel *Dzial i Rodzaj*

```
351 -- -----------------------------------------------------
352 -- Data for table Dzial
353 -- -----------------------------------------------------
354 START TRANSACTION;
355
356 INSERT INTO Dzial (`id_dzial`, `nazwa_dzialu`) VALUES ('LD', 'letni damski');
357 INSERT INTO Dzial (`id_dzial`, `nazwa_dzialu`) VALUES ('ZD', 'zimowy damski');
358 INSERT INTO Dzial (`id_dzial`, `nazwa_dzialu`) VALUES ('LM', 'letni meski');
359 INSERT INTO Dzial (`id_dzial`, `nazwa_dzialu`) VALUES ('ZM', 'zimowy meski');
360
361 COMMIT;
362 -- -----------------------------------------------------
363 -- Data for table Rodzaj
364 -- -----------------------------------------------------
365 START TRANSACTION;
366
367 INSERT INTO Rodzaj (`id_rodzaj`, `nazwa_rodzaju`) VALUES (1, 'koszulka');
368 INSERT INTO Rodzaj (`id_rodzaj`, `nazwa_rodzaju`) VALUES (2, 'spodnie');
369 INSERT INTO Rodzaj (`id_rodzaj`, `nazwa_rodzaju`) VALUES (3, 'sukienka');
370 INSERT INTO Rodzaj (`id_rodzaj`, `nazwa_rodzaju`) VALUES (4, 'kurtka');
371 INSERT INTO Rodzaj (`id_rodzaj`, `nazwa_rodzaju`) VALUES (5, 'bluza');
372 INSERT INTO Rodzaj (`id_rodzaj`, `nazwa_rodzaju`) VALUES (6, 'sweter');
373 INSERT INTO Rodzaj (`id_rodzaj`, `nazwa_rodzaju`) VALUES (7, 'koszula');
374 INSERT INTO Rodzaj (`id_rodzaj`, `nazwa_rodzaju`) VALUES (8, 'spódnica');
375
376 COMMIT;
377
```

- Tworzenie widoku *Produkty*

```
-- -----------------------------------------------------
-- `Widok Produkty`
-- -----------------------------------------------------
DROP VIEW IF EXISTS `View_Produkt`;
CREATE VIEW `View_Produkt` AS
    SELECT `id_produkt`, Wyglad.id_wyglad, `rozmiar`, `liczba_sztuk`, `kolor`, `cena`,
           `opis`, `nazwa_dzialu`, `nazwa_rodzaju`
    FROM produkty
    JOIN wyglad ON produkty.id_wyglad = wyglad.id_wyglad
    JOIN model ON model.id_model = wyglad.id_model
    JOIN kategoria ON kategoria.id_kategoria = model.id_kategoria
    JOIN rodzaj ON kategoria.id_rodzaj = rodzaj.id_rodzaj
    JOIN dzial ON dzial.id_dzial = kategoria.id_dzial;
```

- Tworzenie widoku *Produkt_sprzedany*

```
-- -----------------------------------------------------
-- `Widok Produkt_sprzedany`
-- -----------------------------------------------------
DROP VIEW IF EXISTS `View_Produkt_sprzedany`;
CREATE VIEW `View_Produkt_sprzedany` AS
    SELECT Produkty.id_produkt, `rozmiar`, `kolor`, `cena`, `opis`, `nazwa_dzialu`, `nazwa_rodzaju`, `cena_sprzedazy`,
           Produkty_sprzedane.data_sprzedazy
    FROM `produkty_sprzedane`
    JOIN `Faktura` USING (id_faktura)
    JOIN `Produkty` USING (id_produkt)
    JOIN `Wyglad` ON wyglad.id_wyglad = produkty.id_wyglad
    JOIN `Model` ON wyglad.id_model = model.id_model
    JOIN `Kategoria` ON model.id_kategoria = kategoria.id_kategoria
    JOIN `Rodzaj` ON kategoria.id_rodzaj = rodzaj.id_rodzaj
    JOIN `Dzial` ON dzial.id_dzial = kategoria.id_dzial;
```

- Tworzenie wyzwalaczy *usuniecie_konta_klienta, usuniecie_konta_pracownika, sprzedaz_produktu*

```
-- -----------------------------------------------------
-- `Wyzwalacz usuniecie_konta_klienta`
-- -----------------------------------------------------
DROP TRIGGER IF EXISTS `usuniecie_konta_klienta`;
delimiter //
CREATE TRIGGER usuniecie_konta_klienta
AFTER DELETE ON `klienci`
FOR EACH ROW
    BEGIN
    DELETE FROM `Kontakty` where id_kontakt = old.id_kontakt;
    UPDATE `Zamowienia` SET id_klient = NULL;
    END
    //
-- -----------------------------------------------------
-- `Wyzwalacz usuniecie_konta_pracownika`
-- -----------------------------------------------------
DROP TRIGGER IF EXISTS `usuniecie_konta_pracownika`;
delimiter //
CREATE TRIGGER `usuniecie_konta_pracownika`
AFTER DELETE ON `pracownicy`
FOR EACH ROW
    BEGIN
    DELETE FROM `Kontakty` where id_kontakt = old.id_kontakt;
    END
    //

-- -----------------------------------------------------
-- `Wyzwalacz sprzedaz_produktu`
-- -----------------------------------------------------
DROP TRIGGER IF EXISTS `sprzedaz_produktu`;
delimiter //
CREATE TRIGGER `sprzedaz_produktu`
AFTER INSERT ON `produkty_sprzedane`
FOR EACH ROW
    BEGIN
    UPDATE `Produkty` SET liczba_sztuk = liczba_sztuk - 1 where id_produkt = new.id_produkt;
    END
    //

SET GLOBAL log_bin_trust_function_creators = 1;
```

- Tworzenie funkcji *dochod*

```
446 -- ------------------------------------------------------
447 -- `Funkcja dochod`
448 -- ------------------------------------------------------
449 DROP FUNCTION IF EXISTS `dochod`;
450 delimiter //
451 CREATE FUNCTION dochod() RETURNS DECIMAL(10,2)
452     BEGIN
453     RETURN (SELECT SUM(cena_sprzedazy) FROM produkty_sprzedane);
454     END
455     //
456
```

- Tworzenie funkcji *vat*

```
467 -- ------------------------------------------------------
468 -- `Funkcja vat`
469 -- ------------------------------------------------------
470 DROP FUNCTION IF EXISTS `vat`;
471 delimiter //
472 CREATE FUNCTION vat(cena DECIMAL(10,2)) RETURNS DECIMAL(10,2)
473     BEGIN
474     RETURN 0.23 * cena;
475     END
476     //
```

4. Przypadki użycia

4.1. Sprawdzenie maili oraz wypłat pracowników, którzy są przypisani do kierownika i zarabiają powyżej 3000.

SELECT nazwisko, pensja, id_kierownik, email
FROM Pracownicy JOIN Konakty USING (id_kontakt)
WHERE pensja>3000 AND id_kierownik IS NOT NULL

	nazwisko	pensja	id_kierownik	email
▶	Kowalski	4000	1	hubert.kowalski@gmail.com
	Pelc	4200	1	stefan.pelc@gmail.com
	Paluch	3200	1	ewelina.paluch@gmail.com

4.2. Sprawdzenie ilości pracowników oraz ich sumaryczną pensję.

SELEXT COUNT(*) AS ‘Liczba pracownikow’, SUM(pensja)
FROM Pracownicy;

	Liczba pracowników	SUM(pensja)
▶	8	36100

4.3. Sprawdzenie rocznego dochodu firmy, obecnego stanu magazynu oraz średnich zarobków pracowników

SELECT dochod(), stan_magazynu(), srednie_zarobki();

	dochod()	stan_magazynu()	srednie_zarobki()
▶	383.94	21594	4513

4.1. Sprawdzenie informacji o produktach sprzedanych.

```
SELECT *
FROM View_Produkt_sprzedany;
```

id_produkt	rozmiar	kolor	cena	opis	nazwa_dzialu	nazwa_rodzaju	cena_sprzedazy	data_sprzedazy
17	XS	czarny	69.99	NULL	letni damski	koszula	69.99	2020-03-25 00:00:00
431	L	czarny	99.99	NULL	letni meski	bluza	99.99	2020-02-12 00:00:00
256	M	czarny	45.99	NULL	letni damski	koszulka	45.99	2020-04-20 00:00:00
53	XS	zielony	75.99	NULL	letni meski	koszula	75.99	2019-10-23 00:00:00
129	S	czarny	45.99	NULL	letni damski	koszulka	45.99	2019-06-01 00:00:00
2	XS	czarny	45.99	NULL	letni damski	koszulka	45.99	2019-07-12 00:00:00

5. Wnioski

Baza danych pozwala zarządzać pracą większych firm, urzędów, jednostek gospodarczych i zakładów. Bazę danych można budować na różne sposoby, w zależności od potrzeby. Istotne jest, aby wcześniej dobrze zaplanować jej strukturę. Dlatego też tak ważne jest dokładne przeanalizowanie, jakie informacje baza danych powinna przechowywać oraz kto będzie miał do niej dostęp, a następnie stworzenie odpowiednich diagramów, które będą pomocne przy implementacji bazy. W naszej bazie pojawiły się wszystkie rodzaje relacji, natomiast najczęściej występującą jest relacje jeden do wielu. Co ważne, gdy w bazie pojawi się relacja wiele do wielu, ze względu na strukturę bazy trzeba stworzyć osobną tabelę łączącą. Należy mieć na uwadze, że pojedyncza tabela może być reprezentacją pewnej encji, relacji między nimi, albo może stanowić zawartość całej bazy danych. Oprócz samego umieszczania danych w bazie, jesteśmy w stanie również stworzyć funkcje i wyzwalacze ułatwiające jej obsługę. Uwzględniając relacje między obiektami, możemy stworzyć wyzwalacze, które będą te relacje mieć na uwadze, dzięki czemu nie będziemy musieli za każdym razem pamiętać o wykonaniu czynności na drugiej tabeli, ponieważ będzie ona wykonywana automatycznie. Warto zwrócić uwagę na pola, które powinny zmieniać swoją wartość po dodaniu lub usuwaniu rekordów. Oczywiście im więcej informacji chcemy zawrzeć w naszej bazie, tym będzie ona bardziej rozbudowana, będzie zawierać więcej związków i relacji, będziemy mieli więcej możliwości pracy na niej i więcej przypadków użycia. Wiąże się to również z wieloma komplikacjami i zmianami w trakcie implementacji.