Bazy Danych II

Prowadzący:
prof. Dr hab. Inż. Andrzej Nowak

Projekt:
Obsługa zamówień hurtowych i detalicznych

Etap: 11
Ocena projektów baz danych.

Data:
07.06.2021

Adam Juraszek
Informatyka
Semestr 4
grupa 1a

I - Określenie celu i wymagań

Pierwszym i najważniejszym etapem w projektowaniu bazy danych jest zawsze określenie celu dla jakiego tworzymy naszą bazę danych, oraz wymogów jakie powinna spełniać. Również ważnym elementem jest zwrócenie uwagi na różne sposoby osiągnięcia naszego celu jak i oszacowanie ryzyka związanego z każdym z nich. Złe przygotowanie do projektu bardzo często prowadzi do popełniania dużej ilości błędów które powodują poważne problemy a w efekcie zabierają cenny czas który musimy poświęcić na ich rozwiązywanie.

1.Cel projektu:

Stworzenie bazy danych która odzwierciedli kawałek rzeczywistości jakim jest obsługa zamówień hurtowych oraz detalicznych, w celu możliwości określenia jakie artykuły powinny zostać przygotowane dla poszczególnego klienta, gdzie powinny zostać wysłane oraz jaka należność jest od niego wymagana w celu finalizacji transakcji.

2.Wymogi:

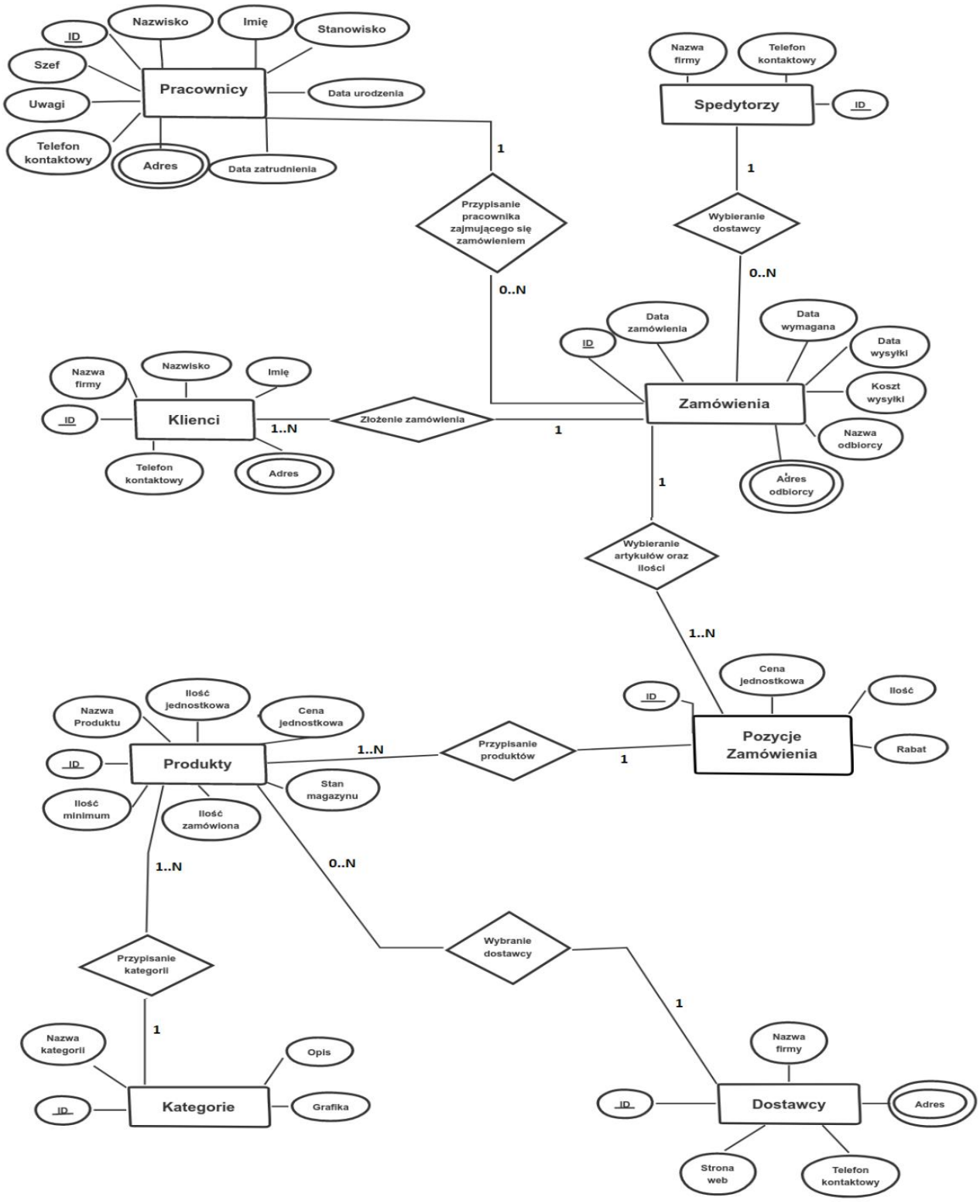
1) Dodanie Klientów
2) Dodanie Produktów
3) Dodanie Pracowników
4) Umożliwienie wyświetlenia informacji o: Klientach, Produktach, Pracownikach
5) Dodanie:
   - informacji o klientach (dane: imię, nazwisko, adres itd.)
   - informacji na temat każdego artykułu (ilość, cena)
   - informacji kto zajmował się danym zamówieniem

Ważne:
- każde zamówienie jest przypisywane do danego pracownika i odbiorcy i mają oni wgląd w swoje zamówienia wraz z przełożonym danego pracownika
- bazę należy uzupełniać na bieżąco, przy obsłudze każdego zamówienia zgodnie z kolejnością wpłynięcia każdego zamówienia
- Każdy pracownik ma obowiązek wprowadzać pełne zamówienie do systemu
- Zamówienie nie może zostać zrealizowane jeśli którykolwiek z zamówionych towarów nie jest dostępy w magazynie
- Prawidłowo uzupełnione zamówienie powinno pozwolić otrzymać informację:

- kto obsługuje zamówienie?
- dla kogo zamówienie jest realizowane?
- jakie produkty zawiera zamówienie wraz z ilością i ceną?
- Jaka jest należność za zamówienie?
- gdzie należy dostarczyć zamówienie?
- jaki jest termin rozpoczęcia i zakończenia realizacji zamówienia?

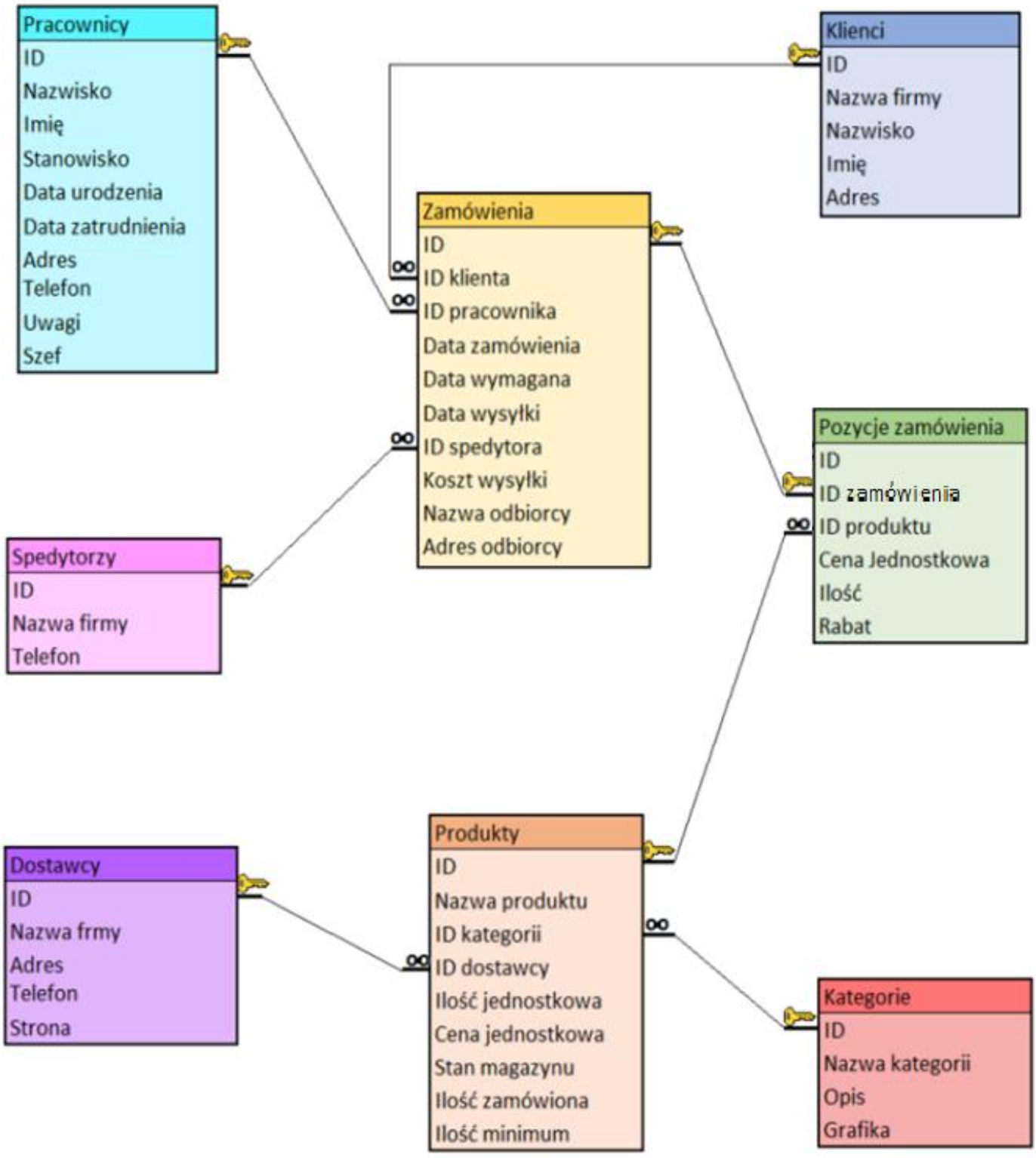
II - Definicja DZE

Dzięki dokładnemu określeniu naszego celu projektowania bazy danych, oraz potrzebnej funkcjonalności jesteśmy gotowi przygotować uproszczony model naszej bazy danych, na tym etapie określamy jakie dane będziemy chcieli przechowywać w naszej bazie danych oraz jakie zależności będą łączyć nasze tabele z danymi



III - Transformacja DZE do modelu relacyjnego

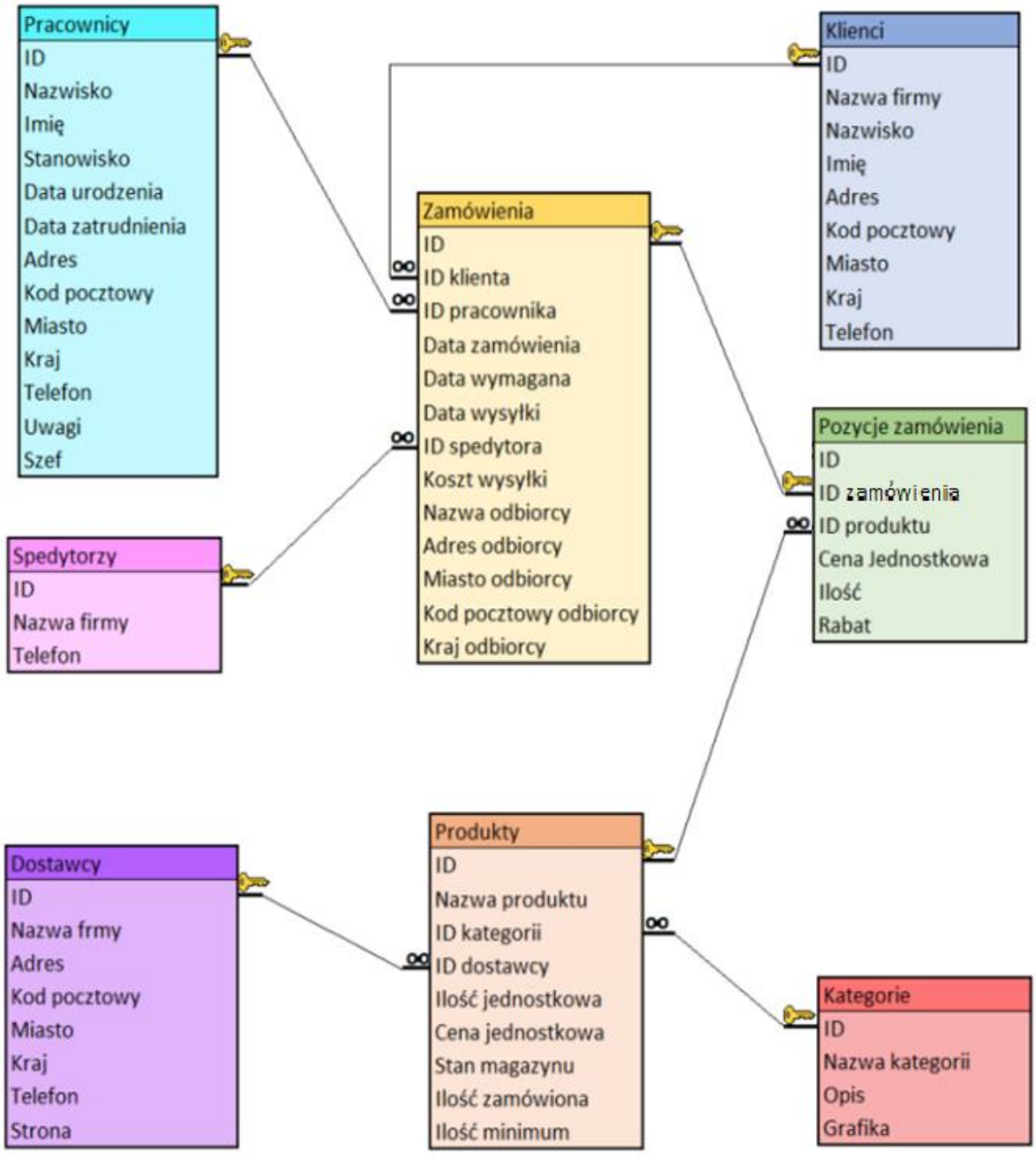
Dzięki dobrze zaprojektowanemu DZE jesteśmy w stanie Stworzyć model relacyjny, który stanowi bardzo dokładne odwzorowanie struktury naszej bazy danych. Na tym etapie porządkujemy nasze dane i określamy zależności miedzy relacjami



IV - Normalizacja schematów relacyjnych

Po utworzeniu modelu relacyjnego należy przeprowadzić jego Normalizację. Na tym etapie prac projektowych skupiamy się głownie na zastąpieniu atrybutów wielowartościowych na poszczególne mniejsze atrybuty (Dane Adresowe: Adres, miasto, kod pocztowy, kraj). Szczególną uwagę należy zwrócić na dublujące się rekordy.

Po przeprowadzeniu pierwszego etapu normalizacji otrzymany schemat spełnia warunki wymagane do kolejno drugiej oraz trzeciej postaci normalnej.



V - Definicja zasad poprawności danych

Kolejnym etapem na drodze projektowania naszej bazy danych jest zdefiniowanie jakie dane zostaną wpisane do naszej bazy danych. Musimy się zastanowić jakie są możliwości uzupełnienia naszych atrybutów informacjami, jaki format mogą otrzymać poszczególne atrybuty, co nie może się znaleźć w poszczególnych atrybutach, oraz gdzie brak informacji nie będzie wpływać negatywnie na żądaną funkcjonalność. Warto również uzasadnić stosowność wyboru konkretnego formatu.

Pracownicy:

- **Nazwisko**
  Tekst nie zawierający cyfr oraz znaków specjalnych, nie może być null z uwagi na konieczność wpisania pełnych danych pracownika
- **Imię**
  Tekst nie zawierający cyfr oraz znaków specjalnych, nie może być null z uwagi na konieczność wpisania pełnych danych pracownika
- **Stanowisko**
  Tekst nie zawierający cyfr oraz znaków specjalnych, nie może być null, ponieważ każdy pracownik jest zatrudnony na jakimś stanowisku
- **Data urodzenia**
  Musi być w formacie daty, nie może być null z uwagi na konieczność wpisania pełnych danych pracownika
- **Data zatrudnienia**
  Musi być w formacie daty, nie może być null z uwagi na konieczność wpisania pełnych danych pracownika
- **Adres**
  Tekst zawierający cyfr oraz znaki specjalne, nie może być null, ponieważ każdy pracownik posiada adres zamieszkania
- **Kod pocztowy**
  Cyfry oraz znak specjalny ” - „ nie może być null, ponieważ każdy pracownik posiada adres zamieszkania
- **Miasto**
  Tekst mogący zawierać znaki specjalne, nie może być null, ponieważ każdy pracownik posiada adres zamieszkania
- **Kraj**
  Tekst, nie może być null, ponieważ każdy pracownik posiada adres zamieszkania
- **Telefon**
  Cyfry, może zawierać znaki specjalne, nie może być null, ponieważ każdy pracownik posiada telefon
- **Uwagi**
  Tekst zawierający cyfr oraz znaki specjalne, może być null, ponieważ nie każdy pracownik musi dostać przypisane uwagi
- **Szef**
  Odpowiada ID pracownika, może być null, ponieważ nie każdy pracownik musi posiadać przełożonego

Klienci:

- **Nazwa Firmy**
  Tekst mogący zawierać znaki specjalne, może być null, ponieważ nie każdy klient będzie klientem firmowym
- **Nazwisko**
  Tekst nie zawierający cyfr oraz znaków specjalnych, nie może być null z uwagi na konieczność wpisania pełnych danych Klienta (w przypadku firmy – przedstawiciel)
- **Imię**
  Tekst nie zawierający cyfr oraz znaków specjalnych, nie może być null z uwagi na konieczność wpisania pełnych danych Klienta (w przypadku firmy – przedstawiciel)
- **Adres**
  Tekst zawierający cyfr oraz znaki specjalne, nie może być null, ponieważ każdy Klient posiada adres zamieszkania (firma – siedzibę)
- **Kod pocztowy**
  Cyfry oraz znak specjalny ” - „ nie może być null, ponieważ każdy Klientk posiada adres zamieszkania (firma – siedzibę)
- **Miasto**
  Tekst mogący zawierać znaki specjalne, nie może być null, ponieważ każdy Klient posiada adres zamieszkania (firma – siedzibę)
- **Kraj**
  Tekst, nie może być null, ponieważ każdy Klient posiada adres zamieszkania (firma – siedzibę)
- **Telefon**
  Cyfry, może zawierać znaki specjalne, nie może być null, ponieważ każdy Klient posiada telefon kontaktowy

Zamówienia:

- **Data zamówienia**
  Musi być w formacie daty, nie może być null z uwagi na konieczność wpisania pełnych danych zamówienia
- **Data wymagana**
  Musi być w formacie daty, nie może być null z uwagi na konieczność wpisania pełnych danych zamówienia, nie może być mniejsza niż Data zamówienia
- **Data wysyłki**
  Musi być w formacie daty, nie może być null z uwagi na konieczność wpisania pełnych danych zamówienia, nie może być mniejsza niż Data zamówienia
- **Koszt wysyłki**
  Cyfry, może być null, ponieważ zamówienie nie musi zostać wysłane, może być odebrane w siedzibie
- **Nazwa odbiorcy**
  Tekst mogący zawierać znaki specjalne, nie może być null z uwagi na konieczność wpisania pełnych danych zamówienia
- **Adres odbiorcy**
  Tekst zawierający cyfr oraz znaki specjalne, nie może być null z uwagi na konieczność wpisania pełnych danych zamówienia
- **Miasto odbiorcy**
  Tekst mogący zawierać znaki specjalne, nie może być null z uwagi na konieczność wpisania pełnych danych zamówienia
- **Kod pocztowy odbiorcy**
  Cyfry oraz znak specjalny ” - „ nie może być null z uwagi na konieczność wpisania pełnych danych zamówienia
- **Kraj odbiorcy**
  Tekst, nie może być null z uwagi na konieczność wpisania pełnych danych zamówienia

Pozycje zamówienia:

- **Cena jednostkowa**
  Cyfry, nie może być null, ponieważ każda pozycja w zamówieniu ma własny koszt
- **Ilość**
  Cyfry, nie może być null, ponieważ każda pozycja w zamówieniu ma własną ilość
- **Rabat**
  Cyfry, może być null, ponieważ Nie każdy produkt dostaje rabat

Spedytorzy:

- **Nazwa firmy**
  Tekst mogący zawierać znaki specjalne, nie może być null nie może być null z uwagi na konieczność wpisania pełnych danych spedytora
- **Telefon**
  Cyfry, może zawierać znaki specjalne, nie może być null, ponieważ każdy spedytor posiada telefon kontaktowy

Dostawcy:

- **Nazwa firmy**
  Tekst mogący zawierać znaki specjalne, nie może być null z uwagi na konieczność wpisania pełnych danych dostwcy
- **Adres**
  Tekst zawierający cyfr oraz znaki specjalne, nie może być null, ponieważ każdy dostawca posiada siedzibę
- **Kod pocztowy**
  Cyfry oraz znak specjalny ” - „ nie może być null, ponieważ każdy dostawca posiada siedzibę
- **Miasto**
  Tekst mogący zawierać znaki specjalne, nie może być, ponieważ każdy dostawca posiada siedzibę
- **Kraj**
  Tekst, nie może być null, ponieważ każdy dostawca posiada siedzibę
- **Telefon**
  Cyfry, może zawierać znaki specjalne, nie może być null, ponieważ każdy dostawca posiada telefon kontaktowy
- **Strona**
  Tekst mogący zawierać znaki specjalne, może być null, ponieważ nie każdy dostawca musi posiadać stronę intetnetową

Produkty

- **Nazwa produktu**
  Tekst nie zawierający cyfr oraz znaków specjalnych, nie może być null, z uwagi na konieczność wpisania pełnych danych o produkcie
- **Ilość jednostkowa**
  Cyfry bez znaków specjalnych, nie może być null, z uwagi na konieczność wpisania pełnych danych o produkcie
- **Cena jednostkowa**
  Cyfry bez znaków specjalnych, nie może być null, z uwagi na konieczność wpisania pełnych danych o produkcie
- **Stan magazynu**
  Cyfry bez znaków specjalnych, nie może być null, ponieważ każdy produkt posiada jakąś ilość w magazynie
- **Ilość zamówiona**
  Cyfry bez znaków specjalnych, może być null, ponieważ nie każdy produkt musi być zamówiony w celu uzupełnienia zapasów
- **Ilość minimum**
  Cyfry bez znaków specjalnych, nie może być null, ponieważ każdy produkt posiada minimalną ilość jaka powinna być w magazynie

Kategorie:

- **Nazwa kategorii**
  Tekst nie zawierający cyfr oraz znaków specjalnych, nie może być null, z uwagi na konieczność wpisania nazwy kategorii
- **Opis**
  Tekst mogący zawierać znaki specjalne, może być null, ponieważ nie każda kategoria musi posiadać opis
- **Grafika**
  Tekst, zawierający znaki specjalne, może być null, ponieważ nie każda kategoria musi posiadać grafikę (tekst – nazwa pliku oraz rozszerzenie)

VI - Definicja schematu bazy danych (SQL), utworzenie bazy danych.

Etap polega na utworzeniu naszej bazy danych z uwzględnieniem wszystkich informacji oraz założeń jakie uzyskaliśmy na podstawie poprzednich etapów. Generujemy pustą strukturę bazy danych pozbawioną informacji.

```
USE master
GO
IF DB_ID(N'BD2_Projekt_OZHID') IS NOT NULL
    DROP DATABASE BD2_Projekt_OZHID;
GO

CREATE DATABASE BD2_Projekt_OZHID
GO

USE [BD2_Projekt_OZHID]
GO

DROP TABLE IF EXISTS dbo.Pracownicy;
GO
CREATE TABLE Pracownicy
(
[ID] INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
[Nazwisko] NVARCHAR(20) NOT NULL,
[Imię] NVARCHAR(10) NOT NULL,
[Stanowisko] NVARCHAR(40) NOT NULL,
[DataUrodzenia] DATE NOT NULL,
[DataZatrudnienia] DATE NOT NULL,
[Adres] NVARCHAR(60) NOT NULL,
[KodPocztowy] NVARCHAR(10) NOT NULL,
[Miasto] NVARCHAR(40) NOT NULL,
[Kraj] NVARCHAR(20) NOT NULL,
[Telefon] NVARCHAR(25) NOT NULL,
[Uwagi] NVARCHAR(MAX) NULL,
[SZef] INT NULL FOREIGN KEY REFERENCES Pracownicy(ID)
);

DROP TABLE IF EXISTS dbo.Klienci;
GO
CREATE TABLE Klienci
(
[Id] INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
[NazwaFirmy] NVARCHAR(40) NULL,
[Nazwisko] NVARCHAR(20) NOT NULL,
[Imię] NVARCHAR(10) NOT NULL,
[Adres] NVARCHAR(60) NOT NULL,
[KodPocztowy] NVARCHAR(10) NOT NULL,
[Miasto] NVARCHAR(40) NOT NULL,
[Kraj] NVARCHAR(20) NOT NULL,
[Telefon] NVARCHAR(25) NOT NULL,
);

DROP TABLE IF EXISTS dbo.Spedytorzy;
GO
CREATE TABLE Spedytorzy
(
[ID] INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
[NazwaFirmy] NVARCHAR(40) NOT NULL,
[Telefon]  NVARCHAR(25) NOT NULL,
);
```

```
DROP TABLE IF EXISTS dbo.Zamówienia;
GO
CREATE TABLE Zamówienia
(
[ID] INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
[IdKlienta] INT NOT NULL FOREIGN KEY REFERENCES Klienci(Id),
[IdPracownika] INT NOT NULL FOREIGN KEY REFERENCES Pracownicy(Id),
[DataZamówienia] DATETIME NOT NULL,
[DataWymagana] DATETIME NOT NULL,
[DataWysyłki] DATETIME NOT NULL,
[IdSpedytora] INT NULL FOREIGN KEY REFERENCES Spedytorzy(Id),
[KosztWysyłki] MONEY NULL,
[NazwaOdbiorcy] NVARCHAR(40) NOT NULL,
[AdresOdbiorcy] NVARCHAR(60) NOT NULL,
[MiastoOdbiorcy] NVARCHAR(40) NOT NULL,
[KodPocztowyOdbiorcy] NVARCHAR(10) NOT NULL,
[KrajOdbiorcy] NVARCHAR(20) NOT NULL,
);

DROP TABLE IF EXISTS dbo.Dostawcy;
GO
CREATE TABLE Dostawcy
(
[ID] INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
[NazwaFirmy] NVARCHAR(40) NOT NULL,
[Adres] NVARCHAR(60) NOT NULL,
[KodPocztowy] NVARCHAR(10) NOT NULL,
[Miasto] NVARCHAR(40) NOT NULL,
[Kraj] NVARCHAR(20) NOT NULL,
[Telefon] NVARCHAR(25) NOT NULL,
[Strona] NVARCHAR(MAX) NOT NULL,
);

DROP TABLE IF EXISTS dbo.Kategorie;
GO
CREATE TABLE Kategorie
(
[ID] INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
[NazwaKategorii] NVARCHAR(20) NOT NULL,
[Opis]  NVARCHAR(MAX) NULL,
[Grafika] IMAGE NULL
);

DROP TABLE IF EXISTS dbo.Produkty;
GO
CREATE TABLE Produkty
(
[ID] INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
[NazwaProduktu] NVARCHAR(40) NOT NULL,
[IdKategorii] INT NOT NULL FOREIGN KEY REFERENCES Kategorie(Id),
[IdDostawcy] INT NOT NULL FOREIGN KEY REFERENCES Dostawcy(Id),
[IlośćJednostkowa] SMALLINT NOT NULL,
[CenaJednostkowa] MONEY NOT NULL,
[StanMagazynu] SMALLINT NOT NULL,
[IlośćZamówiona] SMALLINT NULL,
[IlośćMinimum] SMALLINT NOT NULL
);

DROP TABLE IF EXISTS dbo.PozycjeZamówienia;
GO
CREATE TABLE PozycjeZamówienia
(
[ID] INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
[IdZamówienia] INT NOT NULL FOREIGN KEY  REFERENCES Zamówienia(Id),
[IdProduktu] INT NOT NULL FOREIGN KEY REFERENCES Produkty(Id),
[CenaJednostkowa] MONEY NOT NULL,
[Ilość] SMALLINT NOT NULL,
[Rabat] REAL NULL,
);
```

VII - Definicja niedeklaratywnych mechanizmów sprawdzania poprawności danych.

Jak sama nazwa wskazuje ten etap projektu poświęcony jest uwzględnieniu wszystkich form zabezpieczeń które pomogą wprowadzić poprawne dane do naszej basy oraz zagwarantują uzyskanie założonej funkcjonalności. Na podstawie określonych ograniczeń w etapie piątym przystępujemy do określenia jak zabezpieczymy oraz czego użyjemy do ochrony naszej bazy danych przed niewłaściwymi danymi.

Pracownicy:

- **Nazwisko** Wyzwalacz do sprawdzenia wprowadzonego tekstu
- **Imię** Wyzwalacz do sprawdzenia wprowadzonego tekstu
- **Stanowisko** Wyzwalacz do sprawdzenia wprowadzonego tekstu
- **Data urodzenia** Ograniczenia ustawione przy tworzeniu tabeli
- **Data zatrudnienia** Ograniczenia ustawione przy tworzeniu tabeli
- **Adres** Ograniczenia ustawione przy tworzeniu tabeli
- **Kod pocztowy** Wyzwalacz do sprawdzenia wprowadzonego tekstu
- **Miasto** Ograniczenia ustawione przy tworzeniu tabeli
- **Kraj** Ograniczenia ustawione przy tworzeniu tabeli
- **Telefon** Ograniczenia ustawione przy tworzeniu tabeli
- **Uwagi** Ograniczenia ustawione przy tworzeniu tabeli
- **Szef** Wyzwalacz do sprawdzenia wprowadzonych danych

Klienci:

- **Nazwa Firmy** Ograniczenia ustawione przy tworzeniu tabeli
- **Nazwisko** Wyzwalacz do sprawdzenia wprowadzonego tekstu
- **Imię** Wyzwalacz do sprawdzenia wprowadzonego tekstu
- **Adres** Ograniczenia ustawione przy tworzeniu tabeli
- **Kod pocztowy** Wyzwalacz do sprawdzenia wprowadzonego tekstu
- **Miasto** Ograniczenia ustawione przy tworzeniu tabeli
- **Kraj** Ograniczenia ustawione przy tworzeniu tabeli
- **Telefon** Ograniczenia ustawione przy tworzeniu tabeli

Zamówienia:

- **Data zamówienia** Ograniczenia ustawione przy tworzeniu tabeli
- **Data wymagana** Wyzwalacz do sprawdzenia wprowadzonych danych (nie może być mniejsza niż Data zamówienia)
- **Data wysyłki** Wyzwalacz do sprawdzenia wprowadzonych danych (nie może być mniejsza niż Data zamówienia)
- **Koszt wysyłki** Ograniczenia ustawione przy tworzeniu tabeli
- **Nazwa odbiorcy** Ograniczenia ustawione przy tworzeniu tabeli
- **Adres odbiorcy** Ograniczenia ustawione przy tworzeniu tabeli
- **Miasto odbiorcy** Ograniczenia ustawione przy tworzeniu tabeli
- **Kod pocztowy odbiorcy** Wyzwalacz do sprawdzenia wprowadzonego tekstu
- **Kraj odbiorcy** Ograniczenia ustawione przy tworzeniu tabeli

Pozycje zamówienia:

- **Cena jednostkowa** Ograniczenia ustawione przy tworzeniu tabeli
- **Ilość** Ograniczenia ustawione przy tworzeniu tabeli
- **Rabat** Ograniczenia ustawione przy tworzeniu tabeli

Spedytorzy:

- **Nazwa firmy** Ograniczenia ustawione przy tworzeniu tabeli
- **Telefon** Ograniczenia ustawione przy tworzeniu tabeli

Dostawcy:

- **Nazwa firmy** Ograniczenia ustawione przy tworzeniu tabeli
- **Adres** Ograniczenia ustawione przy tworzeniu tabeli
- **Kod pocztowy** Wyzwalacz do sprawdzenia wprowadzonego tekstu
- **Miasto** Ograniczenia ustawione przy tworzeniu tabeli
- **Kraj** Ograniczenia ustawione przy tworzeniu tabeli
- **Telefon** Ograniczenia ustawione przy tworzeniu tabeli
- **Strona** Ograniczenia ustawione przy tworzeniu tabeli

Produkty

- **Nazwa produktu** Wyzwalacz do sprawdzenia wprowadzonego tekstu,
- **Ilość jednostkowa** Ograniczenia ustawione przy tworzeniu tabeli
- **Cena jednostkowa** Ograniczenia ustawione przy tworzeniu tabeli
- **Stan magazynu** Ograniczenia ustawione przy tworzeniu tabeli
- **Ilość zamówiona** Ograniczenia ustawione przy tworzeniu tabeli
- **Ilość minimum** Ograniczenia ustawione przy tworzeniu tabeli

Kategorie:

- **Nazwa kategorii** Wyzwalacz do sprawdzenia wprowadzonego tekstu
- **Opis** Ograniczenia ustawione przy tworzeniu tabeli
- **Grafika** Ograniczenia ustawione przy tworzeniu tabeli

VIII - Implementacja niedeklaratywnych mechanizmów sprawdzania poprawności danych.

Po określeniu wymaganych ograniczeń dla implementacji poszczególnych fragmentów informacji należy wprowadzić mechanizmy służące do sprawdzania poprawności danych do naszej bazy.

```
USE BD2_Projekt_OZHID
GO


-- Pracownicy - 3 ograniczenia i 2 triggery
ALTER TABLE dbo.Pracownicy DROP CONSTRAINT IF EXISTS chNazwiskoPracownicy
GO
ALTER TABLE dbo.Pracownicy
ADD CONSTRAINT chNazwiskoPracownicy CHECK (Nazwisko NOT LIKE '%[^A-Z, ^a-z]%' );
GO
ALTER TABLE dbo.Pracownicy DROP CONSTRAINT IF EXISTS chImiePracownicy
GO
ALTER TABLE dbo.Pracownicy
ADD CONSTRAINT chImiePracownicy CHECK (Imię NOT LIKE '%[^A-Z, ^a-z]%' );
GO
ALTER TABLE dbo.Pracownicy DROP CONSTRAINT IF EXISTS chStanowiskoPracownicy
GO
ALTER TABLE dbo.Pracownicy
ADD CONSTRAINT chStanowiskoPracownicy CHECK (Stanowisko NOT LIKE '%[^A-Z, ^a-z]%' );
GO
DROP TRIGGER IF EXISTS trKodPocztowyPracownicy
GO
CREATE TRIGGER trKodPocztowyPracownicy
ON Pracownicy
FOR INSERT
AS
IF (
(SELECT COUNT(KodPocztowy)
FROM inserted
WHERE KodPocztowy NOT LIKE '[0-9][0-9]-[0-9][0-9][0-9]')>0
)
BEGIN
PRINT 'BŁĄD - Wprowadzony KodPocztowy w tabeli Pracownicy jest niepoprawny';
ROLLBACK TRANSACTION;
END
GO
```

```sql
-- Klienci - 2 ograniczenia i 1 trigger
ALTER TABLE dbo.Klienci DROP CONSTRAINT IF EXISTS chNazwiskoKlienci
GO
ALTER TABLE dbo.Klienci
ADD CONSTRAINT chNazwiskoKlienci CHECK (Nazwisko NOT LIKE '%[^A-Z, ^a-z]%' );
GO
ALTER TABLE dbo.Klienci DROP CONSTRAINT IF EXISTS chImieKlienci
GO
ALTER TABLE dbo.Klienci
ADD CONSTRAINT chImieKlienci CHECK (Imię NOT LIKE '%[^A-Z, ^a-z]%' );
GO
DROP TRIGGER IF EXISTS trKodPocztowyKlienci
GO
CREATE TRIGGER trKodPocztowyKlienci
ON Klienci
FOR INSERT
AS
IF (
(SELECT COUNT(KodPocztowy)
FROM inserted
WHERE KodPocztowy NOT LIKE '[0-9][0-9]-[0-9][0-9][0-9]')>0
)
BEGIN
PRINT 'BŁĄD - Wprowadzony KodPocztowy w tabeli Klienci jest niepoprawny';
ROLLBACK TRANSACTION;
END
GO
```

```sql
-- Zamówienia - 3 triggery
DROP TRIGGER IF EXISTS trDataWymaganaZamówienia
GO
CREATE TRIGGER trDataWymaganaZamówienia
ON Zamówienia
FOR INSERT
AS
IF(
(SELECT DataWymagana
FROM inserted)
<
(SELECT DataZamówienia
FROM inserted)
)
BEGIN
PRINT 'BŁĄD - Wprowadzona DataWymagana w tabeli Zamówienia jest niepoprawna';
ROLLBACK TRANSACTION;
END
GO
DROP TRIGGER IF EXISTS trDataWysyłkiZamówienia
GO
CREATE TRIGGER trDataWysyłkiZamówienia
ON Zamówienia
FOR INSERT
AS
IF(
(SELECT DataWysyłki
FROM inserted)
<
(SELECT DataZamówienia
FROM inserted)
)
BEGIN
PRINT 'BŁĄD - Wprowadzona DataWysyłki w tabeli Zamówienia jest niepoprawna';
ROLLBACK TRANSACTION;
END
GO
DROP TRIGGER IF EXISTS trKodPocztowyZamówienia
GO
CREATE TRIGGER trKodPocztowyZamówienia
ON Zamówienia
FOR INSERT
AS
IF (
(SELECT COUNT(KodPocztowy)
FROM inserted
WHERE KodPocztowy NOT LIKE '[0-9][0-9]-[0-9][0-9][0-9]')>0
)
BEGIN
PRINT 'BŁĄD - Wprowadzony KodPocztowy w tabeli Zamówienia jest niepoprawny';
ROLLBACK TRANSACTION;
END
GO
```

```
-- Pozycje zamówienia - brak konieczności dodania dodatkowych ograniczeń
-- Spedytorzy - brak konieczności dodania dodatkowych ograniczeń
-- Dostawcy - 1 trigger
DROP TRIGGER IF EXISTS trKodPocztowyDostawcy
GO
CREATE TRIGGER trKodPocztowyDostawcy
ON Dostawcy
FOR INSERT
AS
IF (
(SELECT COUNT(KodPocztowy)
FROM inserted
WHERE KodPocztowy NOT LIKE '[0-9][0-9]-[0-9][0-9][0-9]')>0
)
BEGIN
PRINT 'BŁĄD - Wprowadzony KodPocztowy w tabeli Dostawcy jest niepoprawny';
ROLLBACK TRANSACTION;
END
GO
-- Produtky - 1 trigger
DROP TRIGGER IF EXISTS trNazwaProduktu
GO
CREATE TRIGGER trNazwaProduktu
ON Produkty
FOR INSERT
AS
IF (
(SELECT COUNT(NazwaKategorii)
FROM inserted
WHERE NazwaKategorii LIKE '%![a-z,A-Z,ą,ę,ó,ś,ł,ł,ż,ź,ć, ]%')>0
)
BEGIN
PRINT 'BŁĄD - Wprowadzona NazwaProduktu w tabeli Produkty jest niepoprawna';
ROLLBACK TRANSACTION;
END
GO
-- Kategorie - 1 trigger
DROP TRIGGER IF EXISTS trNazwaKategorii
GO
CREATE TRIGGER trNazwaKategorii
ON Kategorie
FOR INSERT
AS
IF (
(SELECT COUNT(NazwaKategorii)
FROM inserted
WHERE NazwaKategorii LIKE '%![a-z,A-Z,ą,ę,ó,ś,ł,ł,ż,ź,ć, ]%')>0
)
BEGIN
PRINT 'BŁĄD - Wprowadzona NazwaKategorii w tabeli Kategorie jest niepoprawna';
ROLLBACK TRANSACTION;
END
GO
```

IX - Implementacja kodu wspomagającego aplikację użytkową.

Ostatnim z elementów tworzenia bazy danych jest wprowadzenie kodu który będzie wspomagał naszą bazę danych. Elementami wspomagającymi naszą bazę są: widoki, procedury oraz funkcje. Wszystkie z wymienionych pełnią rolę ułatwienia dostępu do konkretnych informacji. Zamiast Tworzyć długie i skomplikowane zapytanie możemy korzystać z gotowych rozwiązań.

```
USE BD2_Projekt_OZHID
GO

-- Widok wyświtlający dane
-- potrzebne do wystawienia faktury dla klienta do każdego zamówienia

--(ID zamówienia, Datę zamówienia, Pracowik który obsługuje zamówienie,
--Odbiorca zamówienia, Adres kupującego, Całkowita wartość zamówienia + uwzględnić rabat)
DROP VIEW IF EXISTS dbo.v_Faktury
GO

CREATE VIEW v_Faktury (
[ID],
[Data],
[Pracownik],
[Klient],
[Adres],
[Wartość zamówienia])
AS (
SELECT z.ID, z.DataZamówienia, p.Imię + ' ' + p.Nazwisko,
   CONCAT(k.NazwaFirmy +', ', k.Imię, ' ',  k.Nazwisko),
   z.AdresOdbiorcy + ', ' + z.KodPocztowyOdbiorcy + ', ' + z.MiastoOdbiorcy + ', ' +
z.KrajOdbiorcy,
   ( SELECT t1.Wartość
     FROM ( SELECT ID, SUM(CenaJednostkowa * Ilość * (1 - Rabat)) AS Wartość
               FROM PozycjeZamówienia
                  GROUP BY ID) t1) [Wartość Zamówienia]
FROM Zamówienia z, Pracownicy p, Klienci k
WHERE z.IdPracownika = p.Id
AND z.IdKlienta = k.ID
);
GO

/*
Przykład wywołania widoku faktury

SELECT *
FROM v_Faktury
*/
```

```sql
--Funkcja zwracająca produkty które należy uzupełnić
--w celu szbkiego zapełnienia magazynu

--(wyświetla produkty których stan minimum na magazynie jest
--przekorczony )
DROP FUNCTION IF EXISTS dbo.f_ProduktyDoUzupełnienia
GO

CREATE FUNCTION dbo.f_ProduktyDoUzupełnienia ()
RETURNS TABLE
AS
RETURN(   SELECT p.NazwaProduktu, p.StanMagazynu, p.IlośćMinimum
          FROM Produkty p, PozycjeZamówienia pz
          WHERE p.ID = pz.IdProduktu
          AND pz.Ilość < p.IlośćMinimum
     );
GO

/*
Przykład wywołania funkcji:

SELECT *
FROM f_ProduktyDoUzupełnienia ()
*/

--Procedura wpisująca zestawienie transakcji danego klienta:
--w celu rozliczenia miesięcznego z klientem

--(IDKlienta, NazwaKlienta, IDzamówienia i datę zamówienia.
--(Parametrem wejściowym jest identyfikator klienta)


DROP PROCEDURE IF EXISTS dbo.spu_Zamowienia_Klienta;
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE dbo.spu_Zamowienia_Klienta
@id AS NVARCHAR(5)
AS
BEGIN
SELECT k.ID 'ID Klienta', k.NazwaFirmy 'Nazwa Firmy', z.ID 'ID
Zamówienia', z.DataZamówienia 'Data Zamówienia'
FROM Klienci AS k JOIN Zamówienia AS z
ON k.ID = z.IDklienta
WHERE k.ID = @id
END;
GO

/*
Przykłąd uruchowmienia procedury

EXEC dbo.spu_Zamowienia_Klienta
@id = '13'
*/
```

```
--Procedura zmieniające dane na podstawie transakcji
-- aby zoptymalizować działanie bazy danych
-- zmniejszenie czasu potrzebnego na obsługę bazy

--(zmniejszenie stanu magazynu o zamówione produkty)
DROP PROCEDURE IF EXISTS dbo.spu_ZmieńStanProdukty
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE dbo.spu_ZmieńStanProdukty
@IDProduktu AS INT,
@IlośćZamawiana AS SMALLINT

AS
BEGIN
   SET NOCOUNT ON;
   UPDATE Produkty
   SET StanMagazynu = StanMagazynu - @IlośćZamawiana
   WHERE ID = @IDProduktu
   SET NOCOUNT OFF;
END;
GO

/*
Przykład uruchowmienia procedury:

za pomocą dodatkowego triggera który sprawdzi możliwość realizacji zamówienia,
oraz zmodyfikuje ilość produktów dostępnych w magazynie
*/
DROP TRIGGER IF EXISTS dbo.trPotrzebnyTowar
GO

CREATE TRIGGER dbo.trPotrzebnyTowar
ON PozycjeZamówienia
FOR INSERT, UPDATE
AS
IF ((SELECT p.StanMagazynu
   FROM Produkty p, inserted i
   WHERE p.Id = i.IdProduktu) - (SELECT Ilość
                                                    FROM inserted)<0)
   BEGIN
   PRINT 'UWAGA - Wprowadzona Ilość jest niedostępna, zamówienia nie można zrealizować';
   ROLLBACK TRANSACTION;
   END
ELSE
   BEGIN
   DECLARE @ID int
   DECLARE @Ilość int
   SELECT @ID =(SELECT IdProduktu
                FROM inserted)
   SELECT @Ilość=( SELECT Ilość
                        FROM inserted)
          EXEC dbo.spu_ZmieńStanProdukty
          @ID, @Ilość
   END
GO
```

X - Testowanie bazy danych.

Ostatnim etapem po stworzeniu bazy danych jest wprowadzenie do niej danych i sprawdzenie jej funkcjonalności wraz z zabezpieczeniami.

1.IMPLEMENTACJA POPRAWNYCH DANYCH

```
USE BD2_Projekt_OZHID
GO
/*
--KODY UŻYWANE PODCZAS WPROWADZANIA DANYCH I TESTOWANIA:
--resetowanie IDENTYFI COUNT
--aby wszystkie nowowprowadzone dane były indeksowane od 1 z krokiem 1
DBCC CHECKIDENT ('Pracownicy', RESEED, 0);
GO
DBCC CHECKIDENT ('Spedytorzy', RESEED, 0);
GO
DBCC CHECKIDENT ('Produkty', RESEED, 0);
GO
DBCC CHECKIDENT ('Dostawcy', RESEED, 0);
GO
DBCC CHECKIDENT ('Kategorie', RESEED, 0);
GO
DBCC CHECKIDENT ('Klienci', RESEED, 0);
GO
DBCC CHECKIDENT ('PozycjeZamówienia', RESEED, 0);
GO
DBCC CHECKIDENT ('Zamówienia', RESEED, 0);
GO

--Kasowanie zawartości tabel
--aby ponownie spróbować wprowadzić wszystkie wartości po nieudanej próbie
DELETE Spedytorzy
GO
DELETE Produkty
GO
DELETE Kategorie
GO
DELETE Dostawcy
GO
DELETE Klienci
GO
DELETE Pracownicy
GO
DELETE Zamówienia
GO
DELETE PozycjeZamówienia
GO
*/


--Poprawne dane zaimplementowane do tabeli:

--wprowadzam dane Spedytorów (Po wykonaniu zapytania brak błędów)
INSERT INTO Spedytorzy
VALUES
('Kurier 9',                '567567567'),
('Poczta krajowa',    '875676455'),
('Automat paczkowy',  '698736547');
```

```sql
--wprowadzam dane Kategorii (po wykonaniu zapytania brak błędów)

INSERT INTO Kategorie
VALUES
('Nabiał',     'Produkty mleczne',                    NULL),
('Fastfood','Żywność wysoko przetworzona',NULL),
('Warzywa',    'Świeże warzywa',                      NULL),
('Owoce',      'Świeże owoce',                              NULL),
('Napoje',     'Produkty do wypicia',                 NULL),
('Odzierz',     NULL,                                       NULL),
('Słodycze','Słodkie przekąski',               NULL),
('Kwiaty',      NULL,                                       NULL),
('Książki','Książki w formie papierowej', NULL);

--wprowadzam dane Dostawcy (po wykonaniu zapytania brak błędów)
INSERT INTO Dostawcy
VALUES
('Warzywniak Rolnictwo','ul.Piastowska 22',      '43-300',     'Bielsko',
 'Polska',     '147295321',  'www.warzywniakrol.pl'),
('Szmatka Original',  'ul.Przędzalnicza 3',      '00-778',     'Warszawa',
 'Poslka',     '438745239',  'www.szmatka.pl'),
('Ten Smak',                'ul.Konesera 102',         '93-480',     'Łódz',
 'Polska',     '987645623',  'www.smakten.pl'),
('Bukiecik',                'ul.Polna 19',                     '30-063',
 'Kraków',     'Polska',     '698774356',  'www.kwiatybukiecik.pl'),
('Popisanka',               'ul.Artystów 1',           '54-007',     'Wrocław',
 'Polska',     '253485769',  'www.popisanka.pl');

--wprowadzam dane Pracownicy (po wykonaniu zapytania brak błędów)
INSERT INTO Pracownicy
VALUES
('Kowalski',   'Tadeusz',    'Szef',                    '1961-01-16', '1990-07-04',
 'ul.Prosta 3',             '43-300',     'Bielsko',    'Polska',     '764557816',
  NULL,  NULL),
('Malarz',           'Anna',             'Sekretarka', '1984-03-05', '2004-08-21',
 'ul.Kolista 12',     '00-778',     'Warszawa',   'Poslka',     '164856487',  NULL,
  1),
('Adamski',          'Jakub',      'Kierownik',  '1959-09-11', '1999-02-26',
 'ul.Złoty potok 7', '00-778',     'Warszawa',   'Polska',     '768547247',
'Urlop',   1),
('Szpak',            'Mirosław',   'Magazynier', '1977-02-20', '1996-08-08',
 'ul.Kolini 1',             '93-480',     'Łódz',             'Polska',
 '458674258',  NULL,  4),
('Gadowski',   'Michał',     'Magazynier', '1970-08-08', '1996-01-24', 'ul.Młyńska
65','30-063',     'Kraków',     'Poslka',     '456785674',  NULL,  2),
('Kowalska',   'Jadwiga',    'Magazynier', '1989-12-29', '2017-06-16', 'al.Grzybiarzy
40','54-007',     'Wrocław',    'Polska',     '546645678',  NULL,  2),
('Tyczka',           'Joanna',     'Kierownik',  '1999-06-01', '2019-11-16',
 'al.Zamkowa 33',     '43-300',     'Bielsko',    'Polska',     '453455645',  NULL,
  1),
('Wykręt',           'Daniel',     'Informatyk', '1998-07-21', '2007-01-06',
 'ul.Wysoka 187',     '93-480',     'Łódz',             'Polska',     '546489878',
  NULL,  7),
('Pięta',            'Błarzej',    'Logistyk',         '2000-12-24', '2019-06-11',
 'ul.Leśna 34',             '30-063',     'Kraków',     'Polska',     '457658657',
'Podwyżka',7),
('Czul',       'Małgorzata','Sprzedawca', '2002-05-25', '2020-11-09', 'ul.Malarzy
61','54-007',     'Wrocław',    'Polska',     '666857886',  NULL,  7);
```

```
   --wprowadzam dane Klienci (po wykonaniu zapytania brak błędów)
   INSERT INTO Klienci
   VALUES
   ('Gugle',            'Kowalski',   'Karol',      'ul.Pegaza 60',            '43-
300',      'Bielsko',    'Polska',     '465987549'),
   ('CornWin',    'Janik',      'Artur',      'ul.Fiołkowej 23',   '00-778',
    'Warszawa',   'Poslka',     '435675897'),
   (NULL,         'Gruszka',    'Jan',        'al.Sadownicza 54',  '93-480',     'Łódz',
          'Polska',     '435890725'),
   (NULL,         'Brzoza',     'Julia',      'ul.Leśna 12',             '30-063',
    'Kraków',     'Poslka',     '354834456'),
   ('Oowca',            'Broda',      'Kuba',             'al.Równości 2',     '54-
007',      'Wrocław',    'Polska',     '986245475'),
   (NULL,         'Adamska',    'Natalia',    'al.Kolista 80',     '43-300',
    'Bielsko',    'Polska',     '457696763'),
   ('Piramida finanse','Dolar','Piotr',      'ul.Bankowa 99',     '00-778',
    'Warszawa',   'Polska',     '453242544'),
   ('Baranek s.a','Cinek',     'Grzegorz',   'ul.Morksa 7',             '93-480',
    'Łódz',             'Polska',     '234598685'),
   (NULL,         'Węgiel',     'Krystyna',   'ul.Złotników 74',   '30-063',
    'Kraków',     'Polska',     '758694456'),
   (NULL,         'Bułka',      'Stefan',     'ul.Piekarska 66',   '54-007',
    'Wrocław',    'Polska',     '356368696');

   --Powyższe dane nie łączyły się zależnościami ze soba,
   --dlatego wymagały wprowadzenia ich w pierwszej kolejności
   --aby móc zaimplementować pozostałe dane do tabel

   --wprowadzam dane Produkty (po wykonaniu zapytania brak błędów)
   INSERT INTO Produkty
   VALUES
   ('Ser biały',         1,3,   '1',   3.99,  100,   0,           60),
   ('Ser żółty',         1,3,   '1',   4.49,  60,          0,           60),
   ('Pizza',                    2,3,   '1',   8.98,  30,          0,           20),
   ('Hot dog',                  2,3,   '1',   6.59,  15,          25,          40),
   ('Zapiekanka',        2,3,   '1',   5,           30,          20,          40),
   ('Pomidor',                  3,1,   '1',   1.99,  60,          0,           50),
   ('Brzoskwinia',              4,1,   '1',   1.85,  60,          20,          50),
   ('Woda mineralna',    5,3,   '1',   2.9,   36,          36,          36),
   ('Chipsy',                   2,3,   '1',   5.12,  10,          20,          20),
   ('Klapki',                   6,2,   '1',   45.99, 200,         50,          10),
   ('Spodnie',                  6,2,   '1',   129.89,      100,         0,
    10),
   ('KitKat',                   7,3,   '1',   3.5,   30,          0,           25),
   ('Tulipan',                  8,4,   '1',   2,           70,          0,
    7),
   ('Róża',              8,4,   '1',   4.5,   55,          22,          11),
   ('Bazy danych dla opornych',9,5,'1',96.99,300,         100,         10);

   --wprowadzam dane Zamówienia (po wykonaniu zapytania brak błędów)
   --z uwagi na Triggery sprawdzające poproawność dat każdy wiersz wprowadzam osobno
   INSERT Zamówienia VALUES
   (2,     1,     '2021-04-04', '2021-04-18', '2021-04-15', 1,     7.99,  'Gugle Kowalski
Karol',    'ul.Prosta 3',             '43-300',     'Bielsko',    'Polska')
   INSERT Zamówienia VALUES
   (6,     8,     '2021-04-13', '2021-04-27', '2021-04-17', 3,     9.99,  'Adamska
Natalia',         'al.Zamkowa 33',     '43-300',     'Bielsko',    'Polska')
   INSERT Zamówienia VALUES
   (7,     2,     '2021-04-24', '2021-05-08', '2021-04-25', NULL, NULL,   'Piramida
finanse Dolar Piotr','ul.Złoty potok 7','00-778','Warszawa','Polska')
```

```
--wprowadzam dane PozycjiZamówienia (po wykonaniu zapytania brak błędów)
--z uwagi na Trigger sprawdzający dostepność towaru każdy wiersz wprowadzam osobno
INSERT PozycjeZamówienia VALUES
(1,     1,     3.99,  4,     0)
INSERT PozycjeZamówienia VALUES
(1,     7,     1.85,  10,    0.10)
INSERT PozycjeZamówienia VALUES
(1,     12,    3.5,   1,     0)
INSERT PozycjeZamówienia VALUES
(2,     3,     8.98,  3,     0)
INSERT PozycjeZamówienia VALUES
(2,     12,    3.5,   2,     0)
INSERT PozycjeZamówienia VALUES
(2,     2,     4.49,  1,     0)
INSERT PozycjeZamówienia VALUES
(2,     8,     2.9,   4,     0)
INSERT PozycjeZamówienia VALUES
(2,     9,     5.12,  3,     0)
INSERT PozycjeZamówienia VALUES
(3,     15,    69.99, 200,   0.25)
INSERT PozycjeZamówienia VALUES
(3,     10,    45.99, 50,    0.10)
INSERT PozycjeZamówienia VALUES
(3,     11,    129.89,60,    0.10)
```

2.TESTOWANIE DZIAŁANIA ZABEZPIECZEŃ BRZED BŁĘDNYM WPROWADZENIEM DANYCH

```
--TESTOWANIE BAZY POD KĄTEM DOTRZYMANIA ZAŁOŻONYCH FUNKCJONALONŚCI

--1. DZIAŁANIE ZABEZPIECZEŃ:

--TABELA ZAMÓWIENIA:
--Wprowadzenie złej daty wysyłki
--BŁĄD - Wprowadzona DataWysyłki w tabeli Zamówienia jest niepoprawna
INSERT Zamówienia VALUES
(2,     1,     '2021-04-15', '2021-04-18', '2021-04-04', 1,     7.99,  'Gugle Kowalski
Karol',    'ul.Prosta 3',             '43-300',     'Bielsko',    'Polska')

--Wprowadzenie złej daty wysyłki
--BŁĄD - Wprowadzona DataWymagana w tabeli Zamówienia jest niepoprawna
INSERT Zamówienia VALUES
(2,     1,     '2021-04-16', '2021-04-04', '2021-04-15', 1,     7.99,  'Gugle Kowalski
Karol',    'ul.Prosta 3',             '43-300',     'Bielsko',    'Polska')

--TABELA PRACOWNICY:
--Wprowadzenie złego kodu pocztowego
--BŁĄD - Wprowadzony KodPocztowy w tabeli Pracownicy jest niepoprawny
INSERT Pracownicy VALUES
('Gadowski',   'Michał',     'Magazynier', '1970-08-08', '1996-01-24', 'ul.Młyńska
65','30--063',    'Kraków',     'Poslka',     '456785674',  NULL,  2)

--Wprowadzenie błędnego nazwiska
--The INSERT statement conflicted with the CHECK constraint "chNazwiskoPracownicy".
--The conflict occurred in database "BD2_Projekt_OZHID", table "dbo.Pracownicy",
column 'Nazwisko'.
INSERT Pracownicy VALUES
('Gadow2ski',  'Michał',     'Magazynie5r',       '1970-08-08', '1996-01-24',
 'ul.Młyńska 65',     '30-063',     'Kraków',     'Poslka',     '456785674',  NULL,
 2)
```

```
   --Wprowadzenie błędnego imienia
   --The INSERT statement conflicted with the CHECK constraint "chImiePracownicy".
   --The conflict occurred in database "BD2_Projekt_OZHID", table "dbo.Pracownicy",
column 'Imię'.
   INSERT Pracownicy VALUES
   ('Gadowski',   'M-ichał',    'Magazynier', '1970-08-08', '1996-01-24', 'ul.Młyńska
65','30-063',     'Kraków',     'Poslka',     '456785674',  NULL,  2)

   --YABELA POZYCJE ZAMÓWIENIA
   --wprowadzenie zamówienia z ilością toworu większą niż dostępna w magazynie
   --UWAGA - Wprowadzona Ilość jest niedostępna, zamówienia nie można zrealizować
   INSERT PozycjeZamówienia VALUES
   (3,     11,    129.89,      60,          0.10)

   --TABELA KLIENCI:
   --Wprowadzenie złego kodu pocztowego
   --BŁĄD - Wprowadzony KodPocztowy w tabeli Klienci jest niepoprawny
   INSERT Klienci VALUES
   (NULL,         'Brzoza',     'Julia',      'ul.Leśna 12',             '30-0-3',
    'Kraków',     'Poslka',     '354834456')

   --Wprowadzenie błędnego nazwiska
   --The INSERT statement conflicted with the CHECK constraint "chNazwiskoKlienci".
   --The conflict occurred in database "BD2_Projekt_OZHID", table "dbo.Klienci", column
'Nazwisko'.
   INSERT Klienci VALUES
   (NULL,         'Brzo9za',    'Julia',      'ul.Leśna 12',             '30-063',
    'Kraków',     'Poslka',     '354834456')

   --Wprowadzenie błędnego imienia
   --The INSERT statement conflicted with the CHECK constraint "chImieKlienci".
   --The conflict occurred in database "BD2_Projekt_OZHID", table "dbo.Klienci", column
'Imię'.
   INSERT Klienci VALUES
   (NULL,         'Brzoza',     'Ju-9lia',    'ul.Leśna 12',             '30-063',
    'Kraków',     'Poslka',     '354834456')

   --TABELA KATEGORIE
   --Wprowadzenie błędnej nazyw kategorii
   --BŁĄD - Wprowadzona NazwaKategorii w tabeli Kategorie jest niepoprawna
   INSERT Kategorie VALUES
   ('0braz',      NULL,         NULL)

   --TABEla dostawcy
   --Wprowadzenie błędnej nazyw
   --BŁĄD - Wprowadzony KodPocztowy w tabeli Dostawcy jest niepoprawny
   INSERT Dostawcy VALUES
   ('Ten Smak',   'ul.Konesera 102',   '93-4800',    'Łódz',       'Polska',
    '987645623',  'www.smakten.pl')
```

```sql
--2. - DZIAŁANIE WIDOKU
-- Widok wyświtlający dane
-- potrzebne do wystawienia faktury dla klienta do każdego zamówienia

--(ID zamówienia, Datę zamówienia, Pracowik który obsługuje zamówienie,
--Odbiorca zamówienia, Adres kupującego, Całkowita wartość zamówienia + uwzględnić
rabat)
DROP VIEW IF EXISTS dbo.v_Faktury
GO

CREATE VIEW v_Faktury (
[ID],
[Data],
[Pracownik],
[Klient],
[Adres],
[Wartość zamówienia])
AS (
SELECT z.ID, z.DataZamówienia, p.Imię + ' ' + p.Nazwisko,
 CONCAT(k.NazwaFirmy +', ', k.Imię, ' ',  k.Nazwisko),
 z.AdresOdbiorcy + ', ' + z.KodPocztowyOdbiorcy + ', ' + z.MiastoOdbiorcy + ', ' +
z.KrajOdbiorcy,
 ( SELECT t1.Wartość
  FROM ( SELECT z.ID, sum(pz.CenaJednostkowa * pz.Ilość * (1 - pz.Rabat)) AS
Wartość
            FROM PozycjeZamówienia pz, Zamówienia z
              WHERE pz.IdZamówienia = z.ID
              GROUP BY z.ID) t1
 WHERE t1.ID = z.ID) [Wartość Zamówienia]
FROM Zamówienia z, Pracownicy p, Klienci k
WHERE z.IdPracownika = p.Id
AND z.IdKlienta = k.ID
);
GO

--Przykład wywołania widoku faktury
SELECT *
FROM v_Faktury

--3. - DZIAŁANIE FUNKCJI:
--Funkcja zwracająca produkty które należy uzupełnić
--w celu szbkiego zapełnienia magazynu

--(wyświetla produkty których stan minimum na magazynie jest
--przekorczony )
DROP FUNCTION IF EXISTS dbo.f_ProduktyDoUzupełnienia
GO

CREATE FUNCTION dbo.f_ProduktyDoUzupełnienia ()
RETURNS TABLE
AS
RETURN( SELECT p.NazwaProduktu, p.StanMagazynu, p.IlośćMinimum
        FROM Produkty p, PozycjeZamówienia pz
        WHERE p.ID = pz.IdProduktu
        AND p.StanMagazynu < p.IlośćMinimum
   );
GO

--Przykład wywołania funkcji:

SELECT *
FROM f_ProduktyDoUzupełnienia ()
```

```sql
--4. - DZIAŁANIE PROCEDURY:
--Procedura wpisująca zestawienie transakcji danego klienta:
--w celu rozliczenia miesięcznego z klientem

--(IDKlienta, NazwaKlienta, IDzamówienia i datę zamówienia.
--(Parametrem wejściowym jest identyfikator klienta)
DROP PROCEDURE IF EXISTS dbo.spu_Zamowienia_Klienta;
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE dbo.spu_Zamowienia_Klienta
@id AS NVARCHAR(5)
AS
BEGIN
SELECT k.ID 'ID Klienta',CONCAT(k.NazwaFirmy +', ', k.Imię, ' ',
k.Nazwisko)'Klient',
z.ID 'ID Zamówienia', z.DataZamówienia 'Data Zamówienia'
FROM Klienci AS k JOIN Zamówienia AS z
ON k.ID = z.IDklienta
WHERE k.ID = @id
END;
GO

--Przykład uruchowmienia procedury
EXEC dbo.spu_Zamowienia_Klienta
@id = '6'


--Procedura zmieniające dane na podstawie transakcji
-- aby zoptymalizować działanie bazy danych
-- zmniejszenie czasu potrzebnego na obsługę bazy

--(zmniejszenie stanu magazynu o zamówione produkty)
DROP PROCEDURE IF EXISTS dbo.spu_ZmieńStanProdukty
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE dbo.spu_ZmieńStanProdukty
@IDProduktu AS INT,
@IlośćZamawiana AS SMALLINT

AS
BEGIN
  SET NOCOUNT ON;
  UPDATE Produkty
  SET StanMagazynu = StanMagazynu - @IlośćZamawiana
  WHERE ID = @IDProduktu
  SET NOCOUNT OFF;
END;
GO
```

```
/*
Przykład uruchomienia procedury:

za pomocą dodatkowego triggera który sprawdzi możliwość realizacji zamówienia,
oraz zmodyfikuje ilość produktów dostępnych w magazynie
*/
DROP TRIGGER IF EXISTS dbo.trPotrzebnyTowar
GO

CREATE TRIGGER dbo.trPotrzebnyTowar
ON PozycjeZamówienia
FOR INSERT, UPDATE
AS
IF ((SELECT p.StanMagazynu
  FROM Produkty p, inserted i
  WHERE p.Id = i.IdProduktu) - (SELECT Ilość
                                                    FROM inserted)<0)
  BEGIN
  PRINT 'UWAGA - Wprowadzona Ilość jest niedostępna, zamówienia nie można
zrealizować';
  ROLLBACK TRANSACTION;
  END
ELSE
  BEGIN
  DECLARE @ID int
  DECLARE @Ilość int
  SELECT @ID =(SELECT IdProduktu
               FROM inserted)
  SELECT @Ilość=( SELECT Ilość
                  FROM inserted)
         EXEC dbo.spu_ZmieńStanProdukty
         @ID, @Ilość
  END
GO

--Najlepszym przykładem jest sprawdzenie ilości dostepnych książek o bazach danych
-- ich ilość została zmiejszona gdy w 3 zamówieniu klient zamówił 200 sztuk
-- trigger po sprawdzeniu czy operacja jest możliwa do wykonania wywołał
-- procedurę która zmieniła ilość książki zapisaną w stan magazynu o 200 stuk
```