# **Dokumentacja Bazy Danych**

## Autorzy

Patryk Midera, Paweł Prochot

Spis treści

[Dokumentacja Bazy Danych 1](#_Toc64065533)

[Autorzy 1](#_Toc64065534)

[Cel i założenia 4](#_Toc64065535)

[Diagram 6](#_Toc64065536)

[Tabele 7](#_Toc64065537)

[Rabaty 7](#_Toc64065538)

[Osoby 8](#_Toc64065539)

[Placówki 9](#_Toc64065540)

[Pracownicy 10](#_Toc64065541)

[Klienci 12](#_Toc64065542)

[Urlopy 14](#_Toc64065543)

[Hurtownie 15](#_Toc64065544)

[Kategorie 16](#_Toc64065545)

[Produkty do naprawy 17](#_Toc64065546)

[Czesci do naprawy 18](#_Toc64065547)

[Stan magazynowy części 19](#_Toc64065548)

[Zamowienia 20](#_Toc64065549)

[Szczegoly zamowien 21](#_Toc64065550)

[Typ gwarancji 22](#_Toc64065551)

[Gwarancje 23](#_Toc64065552)

[Zlecenia 25](#_Toc64065553)

[Przebieg Zlecen 27](#_Toc64065554)

[Czesci uzyte do zlecenia 28](#_Toc64065555)

[Reklamacje 29](#_Toc64065556)

[Funkcje i Widoki 31](#_Toc64065557)

[Informacje\_o\_zamowieniu 31](#_Toc64065558)

[Pracownicy\_na\_urlopie 32](#_Toc64065559)

[Aktualnie\_zatrudnieni\_pracownicy 33](#_Toc64065560)

[Gwarancje\_klienta 34](#_Toc64065561)

[Zlecenia\_dla\_kategorii 35](#_Toc64065562)

[Czesci\_ze\_wszystkich\_placowek 36](#_Toc64065563)

[Aktualne\_rabaty 37](#_Toc64065564)

[Przychody\_miesieczne 38](#_Toc64065565)

[Calkowity\_koszt\_zamówienia 39](#_Toc64065566)

[Wydatki\_miesieczne 40](#_Toc64065567)

[Niezrealizowane\_zlecenia 41](#_Toc64065568)

[Przebieg\_zlecenia 42](#_Toc64065569)

[Zlecenia\_realizowane\_przez\_pracownika 43](#_Toc64065570)

[Pracownicy\_zarabiajacy\_mniej\_niz 44](#_Toc64065571)

[Znajdz\_produkt 45](#_Toc64065572)

[Procedury 46](#_Toc64065573)

[Zysk\_z\_dnia 46](#_Toc64065574)

[Nowy\_produkt 46](#_Toc64065575)

[Nowe\_zlecenie 47](#_Toc64065576)

[Nowe\_zlecenie\_gwarancyjne 50](#_Toc64065577)

[Nowa\_reklamacja 52](#_Toc64065578)

[Transfer\_miedzy\_placowkami 54](#_Toc64065579)

[Zamawianie 56](#_Toc64065580)

[Nowa\_gwarancja 58](#_Toc64065581)

[Bledy 60](#_Toc64065582)

[Insert\_or\_update 60](#_Toc64065583)

[WstawAlboEdytujOsoby 61](#_Toc64065584)

[UsunOsoby 62](#_Toc64065585)

[Wyzwalacze 63](#_Toc64065586)

[Przenies\_czesci 63](#_Toc64065587)

[Blokada\_aktualizacji\_zlecen 65](#_Toc64065588)

[Stworz\_klienta 66](#_Toc64065589)

[Przeslij\_czesci\_po\_zakonczeniu\_zamowienia 67](#_Toc64065590)

[Blokada\_zwolnienia\_pracownika\_z\_niezrealizowanymi\_zleceniami 69](#_Toc64065591)

[Typy własne 70](#_Toc64065592)

[Czesci 70](#_Toc64065593)

[Kody błędów 71](#_Toc64065594)

[Strategia pielęgnacji bazy danych 72](#_Toc64065595)

[Aplikacja 73](#_Toc64065596)

## Cel i założenia

* Celem tego projektu jest stworzenie Bazy Danych gromadzącej i przetwarzającej informacji na temat funkcjonowania Serwisu naprawczego sprzętów RTV.
* **Jakiego typu informacje przechowuje**?
  + Podstawowe informacje o pracownikach i klientach korzystających z serwisu naprawczego, oraz o placówkach posiadanych przez serwis.
  + Informacje o częściach których używa się do naprawy zleconych towarów, ich zaopatrzeniowcach (hurtownie) oraz historię ich zamówień.
  + Historię zleceń i ich przebieg.
  + Informacje o naprawianych towarach.
  + Informacje o gwarancjach zakupionych na dany towar.
  + Różne rabaty dla klientów, urlopy pracowników itd.
* **Możliwości**:
  + Pozwala monitorować przebieg zleceń.
  + Zezwala na składanie reklamacji na źle wykonaną naprawę urządzenia.
  + Pozwala tworzyć różnego typu gwarancje.
  + Gromadzi informacje o urlopach pracowników.
  + Samoczynnie dokonuje „zamagazynowania” części do napraw, po dostarczeniu zamówienia lub w przypadku zamknięcia placówki, dostarcza zamagazynowane w niej części do innej posesji jeśli taka istnieje.
  + Ułatwia użytkownikowi wprowadzanie danych dotyczących zleceń, gwarancji, zamówień itd.
  + Pozwala w łatwy sposób podać informację o przeniesieniu części do naprawy z jednej placówki do drugiej.
  + Wyświetla informacje (w ujęciu miesięcznym) na temat pieniędzy zarobionych z zleceń i gwarancji, oraz kosztów poniesionych w ramach zamówień części z hurtowni.
  + Potrafi filtrować pracowników zatrudnionych od zwolnionych, oraz od tych będących na urlopie.
  + Umożliwia wyświetlanie jakie zlecenia są realizowane w danej chwili przez pracownika.
  + Pozostałe funkcjonalności są omawiane przy opisie implementacji.
* **Ograniczenia przyjęte przy projektowaniu**:
  + Baza danych przechowuje tylko faktyczną cenę jaką ponosi klient w ramach zlecenia/gwarancji.
  + Analogicznie z zamówieniami z hurtowni.
  + Przechowuje tylko aktualną (w sensie bieżącą) cenę pojedynczej części z Hurtowni.
  + Jeśli cena się zmieniła w czasie, to informację o niej można uzyskać jedynie z historii zamówień, gdzie przy danej części będzie widnieć inna cena niż przy aktualnej.
  + Każda część ma określonego dostawcę, tzn. nie można zakupić tej części z innej hurtowni jeśli nie jest ona zapisana jako dostawca tej części.
  + Zamówienia z jednej hurtowni odbywają się w ramach dnia i placówki, tzn. jeżeli zostanie złożone zamówienie na daną część, to jeżeli istniało już zamówienie z tej samej placówki z tego samego dnia, to składane zamówienie będzie dołączonego do już istniejącego.

## Diagram

Poniższy diagram przedstawia wszystkie tabele w bazie danych, oraz ich powiązania między sobą.

Aby uprościć zrozumienie zależności między tabelami, przy opisie każdej z nich pojawi się mniejszy diagram pokazujący jej powiązania.



## Tabele

### Rabaty

#### Wizualizacja tabeli

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID\_rabatu | Nazwa | Procent zniżki | Data wprowadzenia | Data wygaśnięcia |
| INT  PK  IDENTITY(1,1) | **NVARCHAR(50)**  **NOT NULL**  **UNIQUE** | **REAL**  **NOT NULL** | **DATE**  **NOT NULL** | **DATE** |

#### Kod

CREATE TABLE rabaty

(

id\_rabatu INT PRIMARY KEY IDENTITY(1, 1),

nazwa NVARCHAR(50) NOT NULL UNIQUE,

[procent znizki] REAL NOT NULL,

[data wprowadzenia] DATE NOT NULL,

[data wygasniecia] DATE

)

#### Diagram powiązań

#### Opis

* Rabaty służą do zmniejszenia klientom cen zleceń i mają określoną datę ważności.
* Rabat można przypisać do bezpośrednio do klienta, oraz do zlecenia.

### Osoby

#### Wizualizacja tabeli

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID\_osoby | Imie | Nazwisko | Plec | Adres | Numer kontaktowy | E-mail |
| INT  PK  IDENTITY(1,1) | **NVARCHAR(50)**  **NOT NULL** | **NVARCHAR(50)**  **NOT NULL** | **CHAR(1)**  **NOT NULL** | **VARCHAR(60)** | **VARCHAR(24)** | **NVRACHAR(60)** |

#### Kod

CREATE TABLE osoby

(

id\_osoby INT PRIMARY KEY IDENTITY(1, 1),

imie NVARCHAR(50) NOT NULL,

nazwisko NVARCHAR(50) NOT NULL,

plec CHAR(1) NOT NULL,

adres NVARCHAR(60),

[numer kontaktowy] VARCHAR(24),

[e-mail] NVARCHAR(60),

--Ograniczenie znaku płci

CONSTRAINT [Znak plci] CHECK(plec IN ('M', 'K'))

)

#### Diagram powiązań

#### Opis

* Tabela przedstawia podstawowe informacje o klientach oraz pracownikach.
* Na kolumnę **Plec** nałożony jest ogranicznik **CHECK** sprawdzający czy znak jest literą **K** lub **M**.

### Placówki

#### Wizualizacja tabeli

|  |  |  |
| --- | --- | --- |
| ID\_placowki | Adres | Numer kontaktowy |
| INT  PK  IDENTITY(1,1) | **NVARCHAR(60)**  **NOT NULL**  **UNIQUE** | **VARCHAR(24)** |

#### Kod

CREATE TABLE placowki

(

id\_placowki INT PRIMARY KEY IDENTITY(1, 1),

adres NVARCHAR(60) NOT NULL UNIQUE,

[numer kontaktowy] VARCHAR(24)

)

#### Diagram powiązań

#### Opis

* Serwis naprawczy może posiadać kilka placówek.
* Każda placówka zatrudnia swoich pracowników którzy realizują w niej zlecenia.
* Każda placówka jest dodatkowo magazynem na części, co symbolizuje dowiązanie

### Pracownicy

#### Wizualizacja tabeli

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID\_pracownika | ID\_placowki | PESEL | Zarobki | Data zatrudnienia | Przewidywana data zwolnienia | Data zwolnienia |
| INT  PK | **INT**  **NOT NULL** | **CHAR(11)**  **NOT NULL**  **UNIQUE** | **INT**  **NOT NULL** | **DATE**  **NOT NULL** | **DATE** | **DATE** |

#### Kod

CREATE TABLE pracownicy

(

id\_pracownika INT PRIMARY KEY,

id\_placowki INT NOT NULL,

pesel CHAR(11) NOT NULL UNIQUE,

zarobki INT NOT NULL CHECK(zarobki >= 0),

[data zatrudnienia] DATE NOT NULL,

[przewidywana data zwolnienia] DATE,

[data zwolnienia] DATE,

FOREIGN KEY (id\_pracownika) REFERENCES osoby(id\_osoby),

FOREIGN KEY (id\_placowki) REFERENCES placowki(id\_placowki) ON DELETE

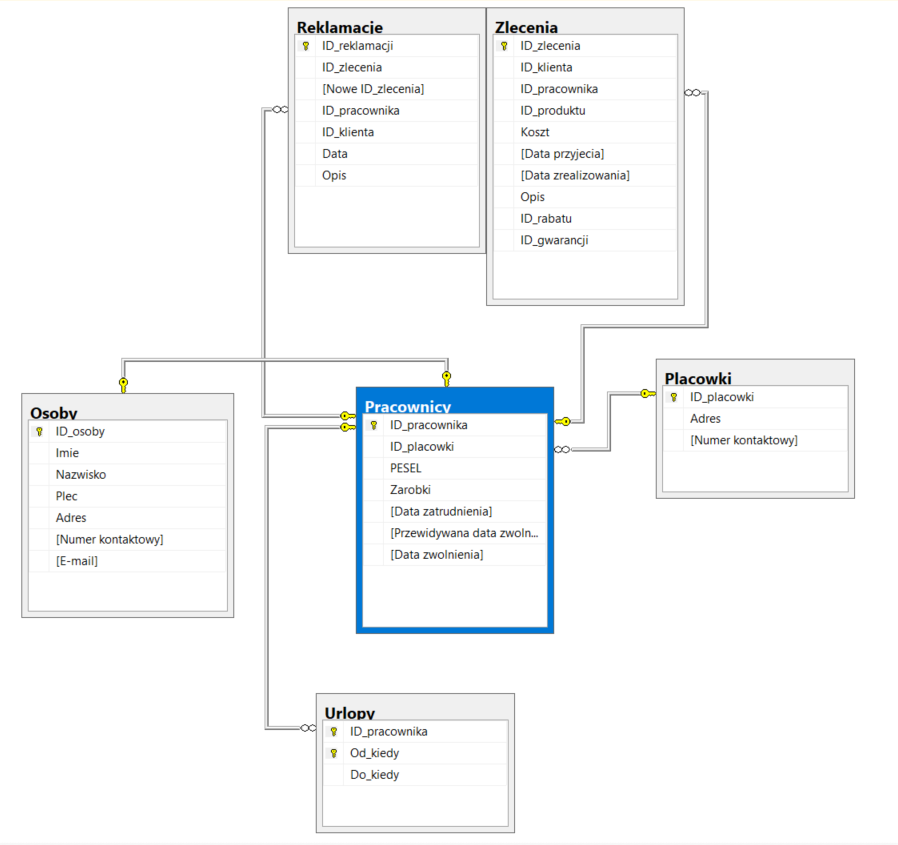
CASCADE

)

#### Opis

* Tak jak mówi nazwa tabeli, dostarcza ona informacji o pracownikach.
* Każdy pracownik jest przypisany do danej placówki.
* Jeżeli pracownik ma umowę na czas nieokreślony do pole **Przewidywana data zwolnienia** jest **Nullem**.
* Jeżeli **Data zwolnienia** jest **Nullem**  to znaczy że pracownik nie został zwolniony i dalej pracuje w firmie.
* Zwolnieni pracownicy nie mogą wykonywać zleceń.
* Ta tabela dziedziczy po tabeli **Osoby**.

#### Diagram powiązań



### Klienci

#### Wizualizacja tabeli

|  |  |  |
| --- | --- | --- |
| ID\_klienta | Rabat\_Indywidualny | Data\_Rejestracji |
| INT  PK | **INT** | **DATE**  **NOT NULL** |

#### Kod

CREATE TABLE klienci

(

id\_klienta INT PRIMARY KEY,

rabat\_indywidualny INT,

data\_rejestracji DATE NOT NULL,

FOREIGN KEY(id\_klienta) REFERENCES osoby(id\_osoby) ON DELETE CASCADE,

FOREIGN KEY(rabat\_indywidualny) REFERENCES rabaty(id\_rabatu) ON DELETE SET

NULL

)

#### Opis

* Tabel dziedziczy po tabeli **Osoby**.
* Każdemu klientowi można przypisać konkretny rabat.
* Dodatkowo przechowywana jest informacja kiedy klient został zapisany do bazy danych.
* Klient jest zleceniodawcą, oraz może zakupywać gwarancje.

#### Diagram powiązań

### Urlopy

#### Wizualizacja tabeli

|  |  |  |
| --- | --- | --- |
| ID\_pracownika | Od\_kiedy | Do\_kiedy |
| INT  PK | **DATE**  **PK**  **NOT NULL** | **DATE**  **NOT NULL** |

#### Kod

CREATE TABLE urlopy

(

id\_pracownika INT,

od\_kiedy DATE NOT NULL,

do\_kiedy DATE NOT NULL,

PRIMARY KEY(id\_pracownika, od\_kiedy),

FOREIGN KEY (id\_pracownika) REFERENCES pracownicy(id\_pracownika) ON DELETE

CASCADE

)

#### Diagram powiązań

#### Opis

* Każdy pracownik może zażądać urlopu.
* Klucz główny jest złożony z atrybutów **ID\_pracownika**, oraz **Od\_kiedy**.
* W przypadku usunięcia pracownika, wszystkie jego urlopy zostaną automatycznie usunięte.

### Hurtownie

#### Wizualizacja tabeli

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID\_hurtowni | Nazwa | Adres | Numer kontaktowy | Opis |
| INT  PK | **NVARCHAR(30)**  **NOT NULL**  **UNIQUE** | **NVARCHAR(60)**  **NOT NULL**  **UNIQUE** | **NVARCHAR(24)** | **NVARCHAR(255)** |

#### Kod

CREATE TABLE hurtownie

(

id\_hurtowni INT PRIMARY KEY IDENTITY(1, 1),

nazwa NVARCHAR(30) NOT NULL UNIQUE,

adres NVARCHAR(60) NOT NULL UNIQUE,

[numer kontaktowy] NVARCHAR(24),

[opis] NVARCHAR(255)

)

#### Diagram powiązań

#### Opis

* Hurtownia jest dostawcą części do placówek.
* Każda hurtownia ma swoją nazwę, oraz adres gdzie się znajduje.
* Opcjonalnie numer kontaktowy do niej, oraz opis co dokładnie dostarcza hurtownia.

### Kategorie

#### Wizualizacja tabeli

|  |  |  |
| --- | --- | --- |
| ID\_kategorii | Nazwa | Opis |
| INT  PK  IDENTITY(1,1) | **NVARCHAR(15)**  **NOT NULL**  **UNIQUE** | **NVARCHAR(255)** |

#### Kod

CREATE TABLE kategorie

(

id\_kategorii INT PRIMARY KEY IDENTITY(1, 1),

nazwa NVARCHAR(15) NOT NULL UNIQUE,

opis NVARCHAR(255)

)

#### Diagram powiązań

#### Opis

* Każdy zlecony produkt do naprawy musi zostać podpięty do jakieś kategorii produktów z tej tabeli.
* Dodatkowo może być dostarczony opis.

### Produkty do naprawy

#### Wizualizacja tabeli

|  |  |  |  |
| --- | --- | --- | --- |
| ID\_produktu | ID\_kategorii | Nazwa | Producent |
| INT  PK  IDENTITY(1,1) | **INT**  **NOT NULL** | **NVARCHAR(15)**  **NOT NULL**  **UNIQUE** | **NVARCHAR(24)** |

#### Kod

CREATE TABLE [produkty do naprawy]

(

id\_produktu INT PRIMARY KEY IDENTITY(1, 1),

id\_kategorii INT NOT NULL,

nazwa NVARCHAR(15) NOT NULL UNIQUE,

producent NVARCHAR(24)

FOREIGN KEY (id\_kategorii) REFERENCES kategorie(id\_kategorii)

)

#### Diagram powiązań

#### Opis

* Tabela dostarcza informacji o produktach które klienci złożyli do naprawy.
* Każdy produkt należy do jakiejś kategorii i ma swoją nazwę, oraz opcjonalnie producenta.
* Jeżeli klient zdecyduje się na zakup gwarancji, to kupuje ją na dokładnie jeden produkt.

### Czesci do naprawy

#### Wizualizacja tabeli

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID\_czesci | Nazwa | Opis | Cena | ID\_hurtowni |
| INT  PK  IDENTITY(1,1) | **NVARCHAR(15)**  **NOT NULL**  **UNIQUE** | **NVARCHAR(255)** | **MONEY**  **NOT NULL** | **INT**  **NOT NULL** |

#### Kod

CREATE TABLE [czesci do naprawy]

(

id\_czesci INT PRIMARY KEY IDENTITY(1, 1),

nazwa NVARCHAR(15) NOT NULL UNIQUE,

opis NVARCHAR(255),

cena MONEY NOT NULL,

id\_hurtowni INT NOT NULL,

FOREIGN KEY (id\_hurtowni) REFERENCES hurtownie(id\_hurtowni)

)

#### Diagram powiązań

#### Opis

* Każdy wiersz tabeli dostarcza informację o konkretnej części która może być wymagana do naprawienia produktu przez pracownika.
* Ma on swoją nazwę i opcjonalnie opis.
* Atrybut **Cena** dostarcza informacji ile kosztuje pojedyncza część, a **ID\_hurtowni** kto jest dostawcą tej części.

### Stan magazynowy części

#### Wizualizacja tabeli

|  |  |  |
| --- | --- | --- |
| ID\_placowki | ID\_czesci | Ilosc |
| INT  PK | **INT**  **PK** | **INT**  **NOT NULL** |

#### Kod

CREATE TABLE [stan magazynowy czesci]

(

id\_placowki INT,

id\_czesci INT,

ilosc INT NOT NULL CHECK(ilosc >= 0),

PRIMARY KEY(id\_placowki, id\_czesci),

FOREIGN KEY (id\_placowki) REFERENCES placowki(id\_placowki),

FOREIGN KEY (id\_czesci) REFERENCES [czesci do naprawy](id\_czesci),

)

#### Diagram powiązań

#### Opis

* Klucz główny w tabeli jest złożony z **ID\_placowki** i **ID\_czesci**.
* Ilość części w placówce jest zmienna w czasie i kupuje się je do placówki za pomocą zamówień do hurtowni.
* Na Ilość towarów w magazynie nałożony jest warunek **CHECK** **>= 0**.

### Zamowienia

#### Wizualizacja tabeli

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID\_zamowienia | ID\_hurtowni | ID\_placowki | Data\_zamowienia | Data\_dostarczenia |
| INT  PK  IDENTITY(1,1) | **INT**  **NOT NULL** | **INT**  **NOT NULL** | **DATE**  **NOT NULL** | **DATE** |

#### Kod

CREATE TABLE zamowienia

(

id\_zamowienia INT PRIMARY KEY IDENTITY(1, 1),

id\_hurtowni INT NOT NULL,

id\_placowki INT NOT NULL,

data\_zamowienia DATE NOT NULL,

data\_dostarczenia DATE,

FOREIGN KEY (id\_hurtowni) REFERENCES hurtownie(id\_hurtowni),

FOREIGN KEY (id\_placowki) REFERENCES placowki(id\_placowki)

)

#### Diagram powiązań

#### Opis

* Tabela ta przedstawia zamówienia części z hurtowni który jest dostawcom zdefiniowanym w tabeli **Czesci do naprawy**.
* Każdego dnia w ramach jednej placówki i jednej hurtowni, istnieje tylko jeden rekord z zamówieniem.
* Jeżeli **Data\_dostarczenia** jest **Nullem** to znaczy że zamówienie nie dotarło jeszcze do placówki.
* Jeżeli w pole **Data\_dostarczenia** zostanie wpisana data, to zamawiane części (zdefiniowane w tabeli **Szczegoly zamowien**) będą przesłane do placówki.

### Szczegoly zamowien

#### Wizualizacja tabeli

|  |  |  |  |
| --- | --- | --- | --- |
| ID\_zamowienia | ID\_czesci | Cena | Ilosc |
| INT  PK | **INT**  **PK** | **MONEY**  **NOT NULL** | **INT**  **NOT NULL** |

#### Kod

CREATE TABLE [szczegoly zamowien]

(

id\_zamowienia INT,

id\_czesci INT,

cena MONEY NOT NULL,

ilosc INT NOT NULL CHECK(ilosc >= 1),

PRIMARY KEY (id\_zamowienia, id\_czesci),

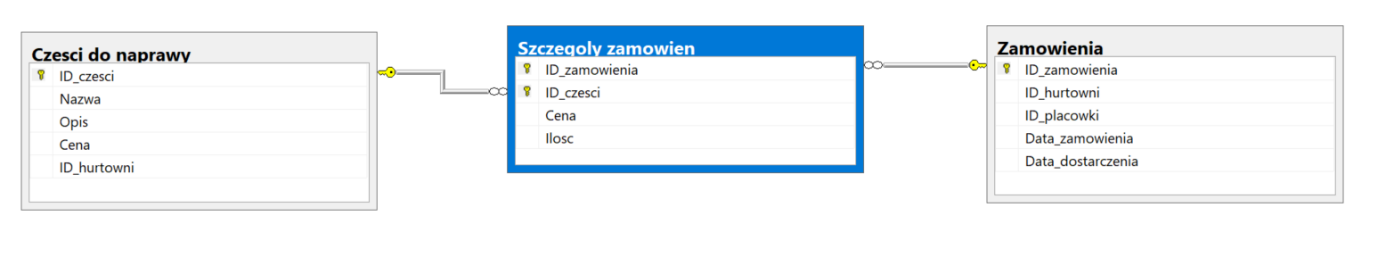
FOREIGN KEY (id\_zamowienia) REFERENCES [zamowienia](id\_zamowienia) ON

DELETE CASCADE,

FOREIGN KEY (id\_czesci) REFERENCES [czesci do naprawy](id\_czesci)

)

#### Diagram powiązań



#### Opis

* Kluczem główny jest złożony. Składa się z pół **ID\_zamowienia** i **ID\_czesci**.
* Pole **Cena** dostarcza informacji o tym ile kosztowały wszystkie części w momencie realizacji tego zamówienia.
* Na pole **Ilosc** nałożony jest warunek **CHECK >= 1**.

### Typ gwarancji

#### Wizualizacja tabeli

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID\_typu\_gwarancji | ID\_kategorii | Cena | Czas trwania | Nazwa |
| INT  PK  IDENTITY(1,1) | **INT**  **NOT NULL** | **MONEY**  **NOT NULL** | **INT**  **NOT NULL** | **NVARCHAR(50)**  **NOT NULL**  **UNIQUE** |

#### Kod

CREATE TABLE [typ gwarancji]

(

id\_typu\_gwarancji INT PRIMARY KEY IDENTITY(1, 1),

id\_kategorii INT NOT NULL,

cena MONEY NOT NULL,

[czas trwania] INT NOT NULL,

nazwa NVARCHAR(50) NOT NULL UNIQUE,

FOREIGN KEY (id\_kategorii) REFERENCES kategorie(id\_kategorii) ON DELETE

CASCADE

)

#### Diagram powiązań

#### Opis

* Definiuje rodzaj gwarancji jaką może zakupić klient.
* Każda gwarancja jest przypisana do jakiejś kategorii i ma swoją cenę oraz nazwę.

### Gwarancje

#### Wizualizacja tabeli

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID\_gwarancji | ID\_produktu | ID\_klienta | ID\_pracownika | Typ gwarancji | Data rozpoczecia | Data zakonczenia |
| INT  PK  IDENTITY(1,1) | **INT**  **NOT NULL** | **INT**  **NOT NULL** | **INT** | **DATE**  **NOT NULL** | **DATE**  **NOT NULL** | **DATE**  **NOT NULL** |

#### Kod

CREATE TABLE gwarancje

(

id\_gwarancji INT PRIMARY KEY IDENTITY(1, 1),

id\_produktu INT NOT NULL,

id\_klienta INT NOT NULL,

id\_pracownika INT,

[typ gwarancji] INT NOT NULL,

[data rozpoczecia] DATE NOT NULL,

[data zakonczenia] DATE NOT NULL,

FOREIGN KEY (id\_klienta) REFERENCES klienci(id\_klienta),

FOREIGN KEY (id\_produktu) REFERENCES [produkty do naprawy](id\_produktu),

FOREIGN KEY (id\_pracownika) REFERENCES pracownicy(id\_pracownika) ON DELETE

SET NULL,

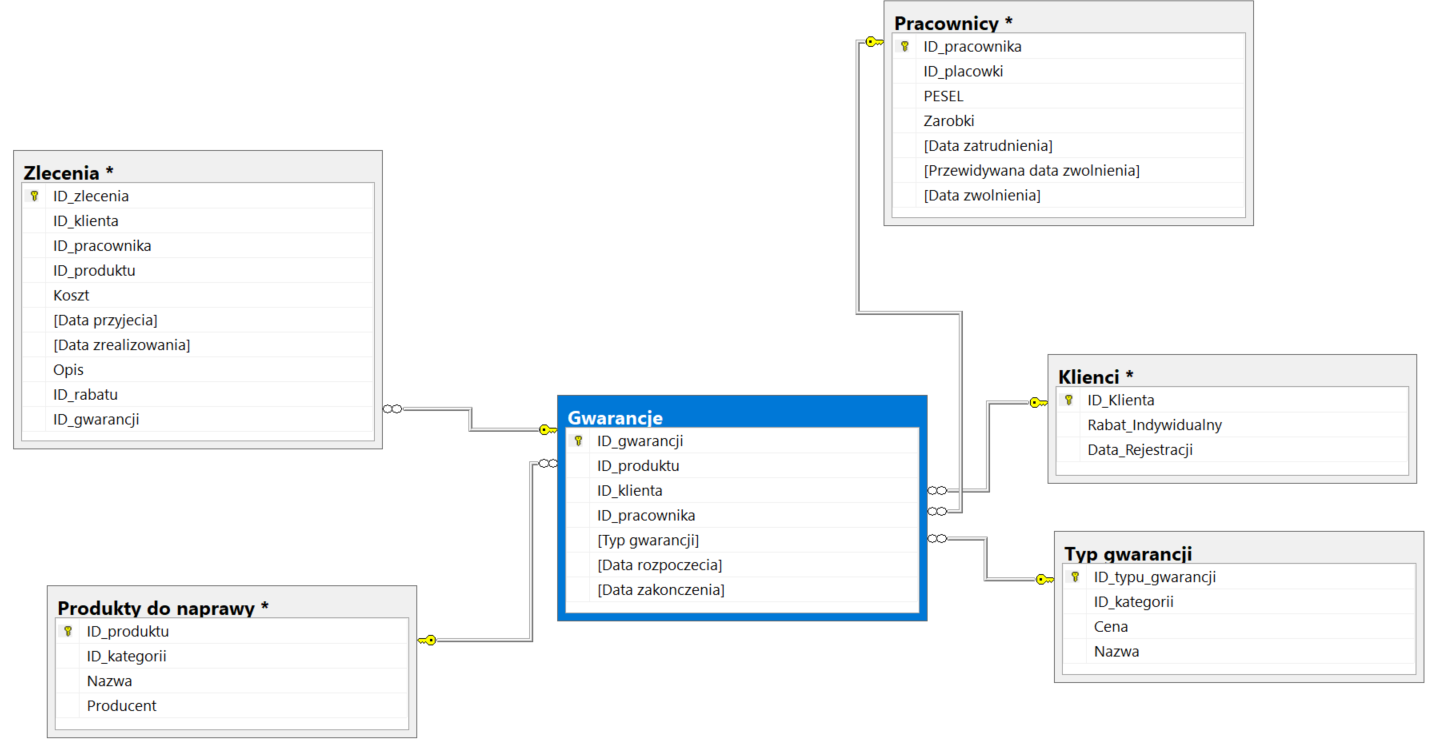
FOREIGN KEY ([typ gwarancji]) REFERENCES [typ gwarancji](id\_typu\_gwarancji)

)

#### Opis

* Gwarancję może zakupić klient reprezentowany przez **ID\_klienta**, na konkretny produkt (**ID\_produktu**).
* **ID\_pracownika** oznacza pracownika który dokonał sprzedaży usługi gwarancyjnej klientowi, jeżeli pracownik zostanie usunięty z Bazy Danych to zostanie usunięty z każdego rekordu i zamieniony na **NULL**.
* **Typ gwarancji** oznacza typ gwarancji jakim został objęty produkt, oraz jej koszt.
* Kategoria **ID\_produktu** musi zgadzać się z kategorią na jakiej zdefiniowany jest **Typ gwarancji**.
* **Data rozpoczecia** i **Data zakonczenia** wyznacza okres na jaki działa usługa.
* Zakupienie przez klienta gwarancji, pozwala mu zgłaszać produkt do naprawy ze zerową cenę.

#### Diagram powiązań



### Zlecenia

#### Wizualizacja tabeli

Poniższa wizualizacja została podzielona na dwie części, ponieważ była za duża.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID\_zlecenia | ID\_klienta | ID\_pracownika | ID\_produktu | Koszt |
| INT  PK  IDENTITY(1,1) | **INT**  **NOT NULL** | **INT**  **NOT NULL** | **INT**  **NOT NULL** | **MONEY**  **NOT NULL** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data przyjęcia | Data  zrealizowania | Opis | ID\_rabatu | ID\_gwarancji |
| DATE  NOT NULL | **DATE** | **NVARCHAR(255)** | **INT** | **INT** |

#### Kod

CREATE TABLE zlecenia

(

id\_zlecenia INT PRIMARY KEY IDENTITY(1, 1),

id\_klienta INT NOT NULL,

id\_pracownika INT NOT NULL,

id\_produktu INT NOT NULL,

koszt MONEY NOT NULL,

[data przyjecia] DATE NOT NULL,

[data zrealizowania] DATE,

opis NVARCHAR(255),

id\_rabatu INT,

id\_gwarancji INT,

FOREIGN KEY (id\_klienta) REFERENCES klienci(id\_klienta),

FOREIGN KEY (id\_pracownika) REFERENCES pracownicy(id\_pracownika),

FOREIGN KEY (id\_produktu) REFERENCES [produkty do naprawy](id\_produktu),

FOREIGN KEY (id\_gwarancji) REFERENCES gwarancje(id\_gwarancji),

FOREIGN KEY (id\_rabatu) REFERENCES rabaty(id\_rabatu)

)

#### Opis

* Każde zlecenie jest składane przez klienta (**ID\_klienta**), a realizowane przez pracownika (**ID\_pracownika**).
* ID\_produktu wskazuje jaki produkt został zlecony do naprawy.
* Koszt to cena brutto zlecenia.
  + Obliczana jest za pomocą pierwotnej ceny – procent rabatu na tym zleceniu – procent rabatu indywidualnego klienta.
  + Jeżeli klient posiada gwarancję to koszt zlecenia jest zerowy.
* Przy każdym zleceniu można podać rabat który obniża jego koszt.
* Pole **ID\_gwarancji** wskazuje na to czy klient ma gwarancję na ten produkt. Jeżeli nie miał to pole jest **Nullem**.

#### Diagram powiązań

### Przebieg Zlecen

#### Wizualizacja tabeli

|  |  |  |  |
| --- | --- | --- | --- |
| ID\_przebiegu\_zlecenia | ID\_zlecenia | Data | Opis |
| INT  PK | **INT**  **NOT NULL** | **DATE**  **NOT NULL** | **NVARCHAR(255)**  **NOT NULL** |

#### Kod

CREATE TABLE [przebieg zlecen]

(

id\_przebiegu\_zlecenia INT PRIMARY KEY IDENTITY(1, 1),

id\_zlecenia INT NOT NULL,

data DATE NOT NULL,

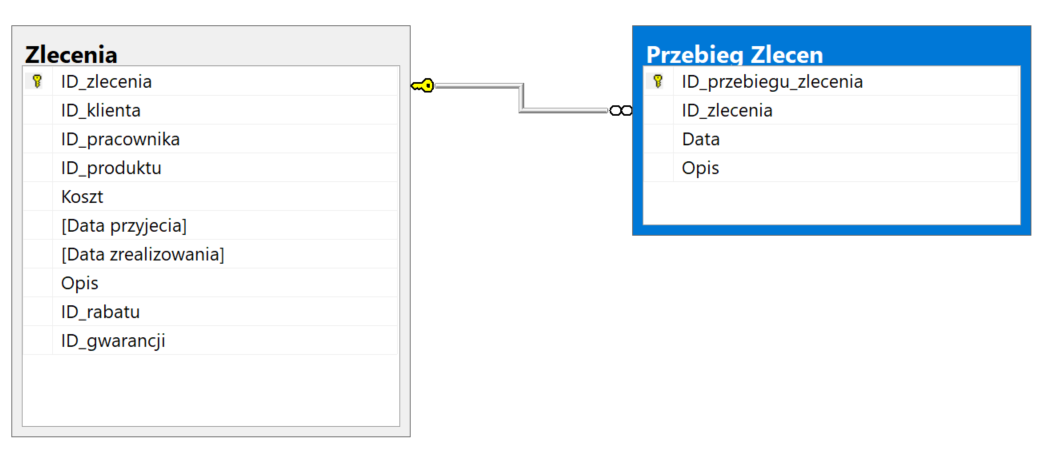
opis NVARCHAR(255) NOT NULL,

FOREIGN KEY (id\_zlecenia) REFERENCES zlecenia(id\_zlecenia) ON DELETE

CASCADE

)

#### Diagram powiązań



#### Opis

* Tabela zawierająca informację o przebiegu zleceń.
* Zlecenie definiowane jest przez **ID\_zlecenia**, a kolejność akcji wykonanych w ramach zlecenia przez **Data**.
* Wymagany jest **Opis** co zostało wykonane w ramach zlecenia.

### Czesci uzyte do zlecenia

#### Wizualizacja tabeli

|  |  |  |  |
| --- | --- | --- | --- |
| ID\_zlecenia | ID\_czesci | Ilosc | Data |
| INT  PK | **INT**  **PK** | **INT**  **NOT NULL** | **DATE**  **NOT NULL** |

#### Kod

CREATE TABLE [czesci uzyte do zlecenia]

(

id\_zlecenia INT,

id\_czesci INT,

ilosc INT NOT NULL CHECK(ilosc >= 1),

data DATE NOT NULL,

PRIMARY KEY(id\_zlecenia, id\_czesci),

FOREIGN KEY (id\_zlecenia) REFERENCES zlecenia(id\_zlecenia) ON DELETE

CASCADE,

FOREIGN KEY (id\_czesci) REFERENCES [czesci do naprawy](id\_czesci),

)

#### Diagram powiązań

#### Opis

* W ramach zlecenia potrzebne są części które potrzebuje naprawiający.
* Części które pobierane są z magazynu w ramach zlecenia, są zapisywane w tej tabeli.
* Na pole **Ilosc** nałożony jest warunek **CHECK >= 1**.
* W przypadku usunięcia zlecenia z Bazy Danych, cała historia użytych części zostanie usunięta.

### Reklamacje

#### Wizualizacja tabeli

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID\_reklamacji | ID\_zlecenia | Nowe ID\_zlecenia | ID\_pracownika | ID\_klienta | DATA | Opis |
| INT  PK  IDENTITY(1,1) | **INT**  **NOT NULL** | **INT**  **NOT NULL** | **INT** | **INT**  **NOT NULL** | **DATE**  **NOT NULL** | **NVARCHAR(255)** |

#### Kod

CREATE TABLE reklamacje

(

id\_reklamacji INT PRIMARY KEY IDENTITY(1, 1),

id\_zlecenia INT NOT NULL,

[nowe id\_zlecenia] INT NOT NULL,

id\_pracownika INT,

id\_klienta INT NOT NULL,

data DATE NOT NULL,

opis NVARCHAR(255),

FOREIGN KEY (id\_klienta) REFERENCES klienci(id\_klienta),

FOREIGN KEY (id\_zlecenia) REFERENCES zlecenia(id\_zlecenia),

FOREIGN KEY ([nowe id\_zlecenia]) REFERENCES zlecenia(id\_zlecenia),

FOREIGN KEY (id\_pracownika) REFERENCES pracownicy(id\_pracownika) ON DELETE

SET NULL

)

#### Opis

* Jeżeli klient uzna że zlecenie zostało źle wykonane i produkt nie został naprawiony, to ma prawo do reklamacji.
* **ID\_zlecenia** wskazuje na zlecenie które zostało niepoprawnie wykonane.
* **Nowe ID\_zlecenia** wskazuje na nowe zlecenie w ramach którego rozpatrywana jest ta reklamacja.
* **ID\_pracownika** wskazuje tutaj na pracownika który przyjął reklamacje (niekoniecznie się nią zajmuje), jeżeli zostanie on usunięty z Bazy Danych to w to miejsce pojawi się **NULL.**

#### Diagram powiązań

## Funkcje i Widoki

### Informacje\_o\_zamowieniu

#### Typ:

*Funkcja*

#### Argumenty:

**@ID\_zamowienia INT**

#### Typ zwracany:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID\_zamowienia | ID\_hurtowni | ID\_placowki | Cena | Data\_zamowienia | Data\_dostarczenia |
| INT | **INT** | **INT** | **MONEY** | **DATE** | **DATE** |

#### Opis:

* Funkcja dostarcza informacji o konkretnym zamówieniu, reprezentowanym przez argument **@ID\_zamowienia**.
* **Cena** jest sumaryczną ceną wszystkich zamawianych części w obrębie danego zamówienia.

#### Kod:

CREATE FUNCTION Informacje\_o\_zamowieniu (@ID\_Zamowienia INT)

RETURNS TABLE

AS

RETURN (

SELECT Z.ID\_zamowienia, Z.ID\_hurtowni, Z.ID\_placowki, SUM(CAST(S.Cena AS MONEY)) AS Koszt, Z.Data\_zamowienia, Z.Data\_dostarczenia

FROM Zamowienia AS Z

JOIN [Szczegoly zamowien] AS S ON (Z.ID\_zamowienia = S.ID\_zamowienia)

WHERE Z.ID\_zamowienia = @ID\_Zamowienia

GROUP BY Z.ID\_zamowienia, Z.ID\_hurtowni, Z.ID\_placowki, Z.Data\_zamowienia, Z.Data\_dostarczenia

)

### Pracownicy\_na\_urlopie

#### Typ:

*Widok*

#### Typ zwracany:

|  |  |  |  |
| --- | --- | --- | --- |
| ID\_pracownika | ID\_placowki | Od\_kiedy | Do\_kiedy |
| INT | **INT** | **DATE** | **DATE** |

#### Opis:

* Widok reprezentuje wszystkich zatrudnionych pracowników, którzy są obecnie na urlopie.

#### Kod:

CREATE VIEW Pracownicy\_na\_urlopie

AS

SELECT U.ID\_pracownika, P.ID\_placowki, U.Od\_kiedy, U.Do\_kiedy

FROM Urlopy AS U

JOIN Pracownicy AS P ON (P.ID\_pracownika = U.ID\_pracownika)

WHERE Do\_kiedy > GETDATE() --Przed końcem urlopu

AND P.[Data zwolnienia] IS NULL --Pracownik który nie został zwolniony

### Aktualnie\_zatrudnieni\_pracownicy

#### Typ:

*Widok*

#### Typ zwracany:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID\_pracownika | ID\_placowki | PESEL | Data zatrudnienia | Przewidywana data zwolnienia |
| INT | **INT** | **CHAR(11)** | **DATE** | **DATE** |

#### Opis:

* Widok wyświetla wszystkich zatrudnionych pracowników we wszystkich placówkach.

#### Kod:

CREATE VIEW Aktualnie\_zatrudnieni\_pracownicy

AS

SELECT P.ID\_pracownika, P.ID\_placowki, P.PESEL, P.[Data zatrudnienia], P.[Przewidywana data zwolnienia]

FROM Pracownicy AS P

JOIN Osoby AS O ON (O.ID\_osoby = P.ID\_pracownika)

WHERE [Data zwolnienia] IS NULL

### Gwarancje\_klienta

#### Typ:

*Funkcja*

#### Argumenty:

**@ID\_klienta INT**.

#### Typ zwracany:

Tabela z kolumnami identycznymi jak w tabeli **Gwarancje**.

#### Opis:

* Funkcja zwraca informację o wszystkich aktywnych gwarancjach posiadanych przez pracownika.

#### Kod:

RETURNS TABLE

AS

RETURN (

SELECT G.\*

FROM Gwarancje AS G

WHERE ID\_klienta = @ID\_Klienta

AND G.[Data zakonczenia] > GETDATE()

)

### Zlecenia\_dla\_kategorii

#### Typ:

*Funkcja*

#### Argumenty:

**@ID\_kategorii INT**

#### Typ zwracany:

Tabela z kolumnami identycznymi jak w tabeli **Zlecenia**.

#### Opis:

* Funkcja zwraca wszystkie wykonywane zlecenia, dla produktów kategorii danej argumentem **@ID\_kategorii**.

#### Kod:

CREATE FUNCTION Zlecenia\_dla\_kategorii(@ID\_kategorii INT)

RETURNS TABLE

AS

RETURN (

SELECT Z.\*

FROM Zlecenia AS Z

JOIN [Produkty do naprawy] AS P ON (P.ID\_produktu = Z.ID\_produktu)

WHERE P.ID\_kategorii = @ID\_kategorii

AND Z.[Data zrealizowania] IS NULL

)

### Czesci\_ze\_wszystkich\_placowek

#### Typ:

*Widok*

#### Typ zwracany:

|  |  |
| --- | --- |
| ID\_zamowienia | Ilosc |
| INT | **INT** |

#### Opis:

* Widok zwraca ilość wszystkich części z placówek, z podziałem na konkretną część.

#### Kod:

CREATE VIEW Czesci\_ze\_wszystkich\_placowek

AS

SELECT S.ID\_czesci, COUNT(S.Ilosc) [Ilosc]

FROM [Stan magazynowy czesci] AS S

GROUP BY S.ID\_czesci

### Aktualne\_rabaty

#### Typ:

*Widok*

#### Typ zwracany:

Tabela takiej samej postaci jak tabela **Rabaty**.

#### Opis:

* Widok zwraca wszystkie rabaty, których data wygaśnięcia jeszcze nie minęła.
* Wszystkie rabaty wyświetlane przez ten widok, można przypisać klientowi lub do zlecenia.

#### Kod:

CREATE VIEW Aktualne\_rabaty

AS

SELECT \*

FROM Rabaty

WHERE [Data wygasniecia] IS NULL

OR [Data wygasniecia] < GETDATE()

### Przychody\_miesieczne

#### Typ:

*Widok*

#### Typ zwracany:

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Przychod ze zlecen | Przychod z gwarancji | Suma przychodow |
| VARCHAR(7) | **MONEY** | **MONEY** | **MONEY** |

#### Opis:

* Widok zwraca zsumowane przychody ze zleceń, oraz z gwarancji w poszczególnym miesiącu, na podstawie daty przyjęcia zlecenia i zakupu gwarancji.
* Data ma postać **MM-YYYY** lub **M-YYYY** w zależności od długości zapisu miesiąca.

#### Kod:

CREATE VIEW Przychody\_miesieczne

AS

SELECT

R.Data AS Data,

SUM(CAST(R.[Przychod ze zlecen] AS MONEY)) AS [Przychod ze zlecen],

SUM(CAST(R.[Przychod z gwarancji] AS MONEY)) AS [Przychod z gwarancji],

SUM(CAST(R.[Przychod ze zlecen] + R.[Przychod z gwarancji] AS MONEY)) AS [Suma przychodow]

FROM

(SELECT

CAST(MONTH(Z.[Data przyjecia]) AS VARCHAR(2)) + '-' + CAST(YEAR(Z.[Data przyjecia]) AS VARCHAR(4)) AS Data,

Z.Koszt AS [Przychod ze zlecen],

0.00 AS [Przychod z gwarancji]

FROM Zlecenia AS Z

UNION ALL

SELECT

CAST(MONTH(G.[Data rozpoczecia]) AS VARCHAR(2)) + '-' + CAST(YEAR(G.[Data rozpoczecia]) AS VARCHAR(4)) AS Data,

0.00 AS [Przychod ze zlecen],

T.Cena AS [Przychod z gwarancji]

FROM Gwarancje AS G

JOIN [Typ gwarancji] T ON (G.[Typ gwarancji] = T.ID\_typu\_gwarancji)

) AS R

GROUP BY Data

### Calkowity\_koszt\_zamówienia

#### Typ:

*Funkcja*

#### Argumenty:

**@ID\_zamowienia INT**

#### Typ zwracany:

**INT**

#### Opis:

* Funkcja zwraca sumaryczny koszt zamówienia podanego argumentem **@ID\_zamowienia**.

#### Kod:

CREATE FUNCTION Calkowity\_koszt\_zamowienia(@ID\_zamowienia INT)

RETURNS INT

AS

BEGIN

RETURN (

SELECT SUM(S.Cena)

FROM [Szczegoly zamowien] AS S

WHERE S.ID\_zamowienia = @ID\_zamowienia

)

END

### Wydatki\_miesieczne

#### Typ:

*Widok*

#### Typ zwracany

|  |  |
| --- | --- |
| Data | Suma wydatkow |
| VARCHAR(7) | **MONEY** |

#### Opis:

* Widok wyświetla zsumowane wydatki na zamówieniach z danego miesiące (na podstawie daty zrealizowania).
* Data ma postać **MM-YYYY** lub **M-YYYY** w zależności od długości zapisu miesiąca.

#### Kod:

CREATE VIEW Wydatki\_miesieczne

AS

SELECT

CAST(MONTH(Z.Data\_dostarczenia) AS VARCHAR(2)) + '-' + CAST(YEAR(Z.Data\_dostarczenia) AS VARCHAR(4)) AS Data,

SUM(CAST(dbo.Calkowity\_koszt\_zamowienia(Z.ID\_zamowienia) AS MONEY)) AS [Suma wydatkow]

FROM Zamowienia AS Z

GROUP BY CAST(MONTH(Z.Data\_dostarczenia) AS VARCHAR(2)) + '-' + CAST(YEAR(Z.Data\_dostarczenia) AS VARCHAR(4))

### Niezrealizowane\_zlecenia

#### Typ:

*Widok*

#### Typ zwracany:

Tabela takiej samej postaci jak tabela **Zlecenia**.

#### Opis:

* Widok zwraca listę wszystkich zleceń które nie zostały jeszcze ukończone. Ich data zrealizowania jest **Nullem**.

#### Kod:

CREATE VIEW Niezrealizowane\_zlecenia

AS

SELECT \*

FROM Zlecenia AS Z

WHERE Z.[Data zrealizowania] IS NULL

### Przebieg\_zlecenia

#### Typ:

*Funkcja*

#### Argumenty:

**@ID\_zlecenia INT**

#### Typ zwracany:

|  |  |  |
| --- | --- | --- |
| ID\_przebiegu\_zlecenia | Data | Opis |
| INT | **DATE** | **VARCHAR(255)** |

#### Opis:

* Funkcja zwraca przebieg zlecenia podanego argumentem **@ID\_zlecenia**.
* Historia zmian w obrębie zlecenia wyświetlana jest od najnowszej, do najstarszej.
* Wyświetlanych jest tylko 100 najnowszych rekordów.

#### Kod:

CREATE FUNCTION Przebieg\_zlecenia(@ID\_zlecenia INT)

RETURNS TABLE

AS

RETURN (

SELECT TOP 100 WITH TIES --Musi być TOP ponieważ nie da się dodać bez tego ORDER BY w funkcji :(

P.ID\_przebiegu\_zlecenia,

P.Data,

P.Opis

FROM [Przebieg Zlecen] AS P

ORDER BY P.Data DESC

)

### Zlecenia\_realizowane\_przez\_pracownika

#### Typ:

*Funkcja*

#### Argumenty:

**@ID\_pracownika INT**

#### Typ zwracany:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID\_zlecenia | ID\_klienta | Data przyjęcia | ID\_produktu | Koszt | ID\_gwarancji | ID\_rabatu |
| INT | **INT** | **DATE** | **INT** | **MONEY** | **INT** | **INT** |

#### Opis:

* Funkcja zwraca wszystkie zlecenia które obecnie realizuje pracownik.

#### Kod:

CREATE FUNCTION Zlecenia\_realizowane\_przez\_pracownika(@ID\_Pracownika INT)

RETURNS TABLE

AS

RETURN (

SELECT Z.ID\_zlecenia, Z.ID\_klienta, Z.[Data przyjecia], Z.ID\_produktu, Z.Koszt, Z.ID\_gwarancji, Z.ID\_rabatu

FROM Zlecenia AS Z

WHERE Z.ID\_pracownika = @ID\_Pracownika

AND Z.[Data zrealizowania] IS NULL

)

### Pracownicy\_zarabiajacy\_mniej\_niz

#### Typ:

*Funkcja*

#### Argumenty:

**@Kwota INT**

#### Typ zwracany:

Tabela taka sama jak tabela **Pracownicy**.

#### Opis:

* Zwraca wszystkich pracowników którzy zarabiają mniej niż kwota podana argumentem

#### Kod:

CREATE FUNCTION Pracownicy\_zarabiajacy\_mniej\_niz(@Kwota INT)

RETURNS TABLE

AS

RETURN (

SELECT \*

FROM Pracownicy

WHERE Zarobki < @Kwota

)

### Znajdz\_produkt

#### Typ:

*Funkcja*

#### Argumenty

* **@nazwa\_produktu NVARCHAR(15)**
* **@producent NVARCHAR(24)**
* **@kategoria NVARCHAR(15)**

#### Typ zwracany:

**INT**

#### Opis:

* Funkcja zwraca **ID\_produktu** podanego definiowanego argumentami.
* Jeśli produktu nie ma w bazie danych, to zwracany jest **NULL**.

#### Kod:

CREATE FUNCTION Znajdz\_produkt(@nazwa\_produktu nvarchar(15), @producent nvarchar(24), @kategoria nvarchar(15))

RETURNS INT

AS

BEGIN

DECLARE @ID INT

SELECT TOP 1 @ID=P.ID\_produktu

FROM [Produkty do naprawy] AS P RIGHT JOIN Kategorie AS K

ON K.ID\_kategorii = P.ID\_kategorii

WHERE @kategoria = K.Nazwa AND @nazwa\_produktu = P.Nazwa AND @producent = P.Producent

RETURN @ID

END

## Procedury

### Zysk\_z\_dnia

* Argumenty:
  + **@Dzien DATE**
* Opis:
  + Procedura wyświetla zsumowany zysk ze zleceń, z konkretnego dnia podanego argumentem.

### Nowy\_produkt

* Argumenty:
  + **@nazwa\_produktu NVARCHAR(15)**
  + **@producent NVARCHAR(24)**
  + **@kategoria NVARCHAR(15)**
  + **@ID INT OUTPUT**
* Opis**:**
  + Procedura dodaje nowy produkt do tabeli **Produkty do naprawy** pod warunkiem, że w tabeli **Kategorie** istnieje podana przez argument kategoria.
  + Jeżeli kategoria nie istnieje to zwraca **@ID = NULL**.
  + Jeżeli kategoria istnieje, to wstawia nowy produkt to tabeli **Produkty do naprawy** i zwraca przez **@ID** nowe **ID** tego produktu.

### Nowe\_zlecenie

* **Argumenty:**
  + **@klient\_id INT**
  + **@pracownik\_id INT**
  + **@nazwa\_produktu NVARCHAR(15)**
  + **@producent NVARCHAR(24)**
  + **@kategoria NVARCHAR(15)**
  + **@koszt MONEY**
  + **@rabat NVARCHAR(50)**
  + **@opis NVARCHAR(255)**
* **Opis:**
  + Procedura tworzy nowe zlecenie w bazie danych na podstawie informacji podanych przez argumenty.
  + Jeżeli procedura nie jest już objęta transakcją, to jest ona uruchamiana w poziomie izolacji **READ COMMITED**.
  + Produkt do naprawy definiowany jest przez **@nazwa\_produktu**, **@producent**, **@kategoria** i procedura wyszukuje lub dodaje ten produkt do bazy danych.
  + Jeżeli nazwa rabatu jest poprawna, to rabat jest przypisywany do zlecenia.
  + Procedura wylicza cenę na podstawie **@koszt**, **@rabat** i indywidualnego rabatu klienta.
  + Mając już wszystkie potrzebne informacje, wstawia rekord do tabeli **Zlecenia**, oraz 1 rekord do tabeli **Przebieg zlecen** informujący o rozpoczęciu zlecenia.
* **Może wyrzucić błędy:**
  + **50002**
  + **50003**
  + **50004**
  + **50005**

#### Kod:

CREATE PROCEDURE Nowe\_zlecenie(

@klient\_id INT,

@pracownik\_id INT,

@nazwa\_produktu nvarchar(15),

@producent nvarchar(24),

@kategoria nvarchar(15),

@koszt MONEY,

@rabat nvarchar(50),

@opis nvarchar(255)

)

AS

BEGIN TRY

DECLARE @tranCount INT = @@TRANCOUNT

IF @tranCount =0

BEGIN TRAN noweZlecenie

IF @klient\_id NOT IN ( SELECT ID\_klienta FROM Klienci)

RAISERROR(50002, -1, -1)--klient nie istnieje

IF @pracownik\_id NOT IN ( SELECT ID\_pracownika FROM Pracownicy)

RAISERROR(50003, -1, -1)--pracownik nie istnieje

DECLARE @ID INT--produktu

SELECT @ID = dbo.Znajdz\_produkt(@nazwa\_produktu,@producent,@kategoria)

IF @ID IS NULL

BEGIN

EXEC dbo.Nowy\_produkt @nazwa\_produktu,@producent,@kategoria,@ID OUTPUT

IF @ID IS NULL

RAISERROR(50004, -1, -1)--produktu nie bylo w bazie i nie da sie - --go dodac bo kategoria jest niepoprawna

END

DECLARE @rabat\_id INT

DECLARE @rabat\_id\_2 INT

DECLARE @rab REAL

IF @rabat IS NOT NULL

BEGIN

SELECT @rabat\_id=R.ID\_rabatu, @rab = R.[Procent znizki]

FROM RABATY AS R

WHERE R.Nazwa = @rabat

IF @rabat\_id IS NULL

RAISERROR(50005, -1, -1)--podany rabat jest bledny

SET @koszt = @koszt \* (1- @rab)

END

SELECT @rabat\_id\_2= K.Rabat\_Indywidualny

FROM Klienci AS K

WHERE K.ID\_klienta = @klient\_id

IF @rabat\_id\_2 IS NOT NULL

BEGIN

SELECT @rab=R.[Procent znizki]

FROM Rabaty AS R

WHERE R.ID\_rabatu = @rabat\_id\_2

SET @koszt = @koszt \* (1- @rab)

END

INSERT INTO Zlecenia(ID\_klienta,ID\_pracownika,ID\_produktu,Koszt,

[Data przyjecia],Opis,ID\_rabatu)

VALUES(@klient\_id,@pracownik\_id,@ID,@koszt,GETDATE(),@opis,@rabat\_id)

INSERT INTO [Przebieg zlecen](ID\_zlecenia,Data,Opis)

VALUES(@@IDENTITY,GETDATE(),N'Przyjecie zlecenia')

IF @tranCount = 0

COMMIT TRAN noweZlecenie

END TRY

BEGIN CATCH

ROLLBACK TRAN noweZlecenie

EXEC Bledy

END CATCH

### Nowe\_zlecenie\_gwarancyjne

* **Argumenty:**
  + **@gwarancja\_id** **INT**
  + **@klient\_id INT**
  + **@pracownik\_id INT**
  + **@opis NVARCHAR(255)**
* **Opis:**
  + Procedura tworzy nowe zlecenie z zerowym kosztem, ponieważ jest to zlecenie gwarancyjne.
  + Jeżeli procedura nie jest już objęta transakcją, to jest ona uruchamiana w poziomie izolacji **READ COMMITED**.
  + Każda gwarancja opisuje jakiś produkt, więc produkt nie jest podawany argumentem.
  + Mając już wszystkie potrzebne informacje, wstawia rekord do tabeli **Zlecenia**, oraz 1 rekord do tabeli **Przebieg zlecen** informujący o rozpoczęciu zlecenia.
* **Może wyrzucić błędy:**
  + **50002**
  + **50003**
  + **50010**

#### Kod:

CREATE PROCEDURE Nowe\_zlecenie\_gwarancyjne(

@gwarancja\_id INT,

@klient\_id INT,

@pracownik\_id INT,

@opis nvarchar(255)

)

AS

BEGIN TRY

DECLARE @tranCount INT = @@TRANCOUNT

IF @tranCount =0

BEGIN TRAN noweZlecenieGwarancyjne

IF @klient\_id NOT IN ( SELECT ID\_klienta FROM Klienci)

RAISERROR(50002, -1, -1)--klient nie istnieje

IF @pracownik\_id NOT IN ( SELECT ID\_pracownika FROM Pracownicy)

RAISERROR(50003, -1, -1)--pracownik nie istnieje

IF @gwarancja\_id NOT IN ( SELECT ID\_gwarancji FROM Gwarancje )

RAISERROR(50010, -1, -1)--gwarancja nie istnieje

DECLARE @produkt\_id INT

SELECT @produkt\_id = G.ID\_produktu

FROM Gwarancje AS G

WHERE G.ID\_gwarancji = @gwarancja\_id

INSERT INTO Zlecenia(ID\_klienta,ID\_pracownika,ID\_produktu,Koszt,

[Data przyjecia],Opis,ID\_rabatu,ID\_gwarancji)

VALUES(@klient\_id,@pracownik\_id,@produkt\_id,0,GETDATE(),@opis,NULL,@gwarancja\_id)

INSERT INTO [Przebieg zlecen](ID\_zlecenia,Data,Opis)

VALUES(@@IDENTITY,GETDATE(),N'Przyjecie zlecenia z gwarancji')

IF @tranCount = 0

COMMIT TRAN noweZlecenieGwarancyjne

END TRY

BEGIN CATCH

ROLLBACK TRAN noweZlecenieGwarancyjne

EXEC Bledy

END CATCH

### Nowa\_reklamacja

* Argumenty:
  + **@zlecenie\_id** **INT**
  + **@klient\_id INT**
  + **@pracownik\_id INT**
  + **@opis NVARCHAR(255)**
* Opis:
  + Procedura tworzy nowe zlecenie z zerowym kosztem, ponieważ jest to reklamacja źle zrealizowanego zlecenia.
  + Jeżeli procedura nie jest już objęta transakcją, to jest ona uruchamiana w poziomie izolacji **READ COMMITED**.
  + Wyszukuje one zlecenie wskazywane przez **@zlecenie\_id** i tworzy nowe zlecenie na ten sam produkt, oraz umieszcza informację o reklamacji w tabeli **Reklamacje**.
* Może wyrzucić błędy:
  + **50002**
  + **50003**
  + **50006**

#### Kod:

CREATE PROCEDURE Nowa\_reklamacja(

@zlecenie\_id INT,

@klient\_id INT,

@pracownik\_id INT,

@opis nvarchar(255)

)

AS

BEGIN TRY

DECLARE @tranCount INT = @@TRANCOUNT

IF @tranCount =0

BEGIN TRAN nowaReklamacja

IF @klient\_id NOT IN ( SELECT ID\_klienta FROM Klienci)

RAISERROR(50002, -1, -1)--klient nie istnieje

IF @pracownik\_id NOT IN ( SELECT ID\_pracownika FROM Pracownicy)

RAISERROR(50003, -1, -1)--pracownik nie istnieje

DECLARE @ID INT--produktu

SELECT @ID = Z.ID\_produktu

FROM Zlecenia AS Z

WHERE Z.ID\_zlecenia=@zlecenie\_id

IF @ID IS NULL

RAISERROR(50006, -1, -1)--zlecenia nie ma w bazie

INSERT INTO Zlecenia(ID\_klienta,ID\_pracownika,ID\_produktu,Koszt,

[Data przyjecia],Opis,ID\_rabatu)

VALUES(@klient\_id,@pracownik\_id,@ID,0,GETDATE(),@opis,NULL)

SELECT @ID = @@IDENTITY

INSERT INTO [Przebieg zlecen](ID\_zlecenia,Data,Opis)

VALUES(@ID,GETDATE(),N'Przyjecie reklamacji')

INSERT INTO Reklamacje(ID\_zlecenia,

[Nowe ID\_zlecenia],ID\_pracownika,ID\_klienta,Data,Opis)

VALUES(@zlecenie\_id,@ID,@pracownik\_id,@klient\_id,GETDATE(),@opis)

IF @tranCount = 0

COMMIT TRAN nowaReklamacja

END TRY

BEGIN CATCH

ROLLBACK TRAN nowaReklamacja

EXEC Bledy

END CATCH

### Transfer\_miedzy\_placowkami

* **Argumenty:**
  + **@placowka** **INT**
  + **@placowka\_docelowa INT**
  + **@id\_czesci INT**
  + **@ile INT**
* **Opis:**
  + Procedura przesyła części z jednej placówki do drugiej o ile w placówce znajduje się wystarczająco dużo części (**@ile**).
  + Jeżeli procedura nie jest już objęta transakcją, to jest ona uruchamiana w poziomie izolacji **READ COMMITED**.
* **Może wyrzucić błędy:**
  + **50007**
  + **50008**
  + **50009**
  + **50011**
  + **50012**

#### Kod:

CREATE PROCEDURE Transfer\_miedzy\_placowkami(

@placowka INT,

@placowka\_docelowa INT,

@id\_czesci INT,@ile INT

)

AS

BEGIN TRY

DECLARE @tranCount INT = @@TRANCOUNT

IF @tranCount =0

BEGIN TRAN transferMiedzyPlacowkami

IF @placowka NOT IN ( SELECT ID\_placowki FROM Placowki)

RAISERROR(50007, -1, -1)--placowka nie istnieje

IF @placowka\_docelowa NOT IN ( SELECT ID\_placowki FROM Placowki)

RAISERROR(50008, -1, -1)--placowka docelowa nie istnieje

IF @id\_czesci NOT IN (SELECT ID\_czesci FROM [Czesci do naprawy])

RAISERROR(50009, -1, -1)--czesc nie istnieje

DECLARE @stan INT

SELECT @stan= SMC.Ilosc

FROM [Stan magazynowy czesci] AS SMC

WHERE SMC.ID\_placowki=@placowka AND SMC.ID\_czesci = @id\_czesci

IF @stan IS NULL

RAISERROR(50011, -1, -1)--Brak czesci w placowce

IF @stan < @ile

RAISERROR(50012, -1, -1)--za malo czesci w placowce

UPDATE [Stan magazynowy czesci]

SET Ilosc= Ilosc - @ile

WHERE ID\_placowki=@placowka AND ID\_czesci = @id\_czesci

DELETE FROM [Stan magazynowy czesci] WHERE Ilosc = 0

SET @stan = NULL

SELECT @stan= SMC.Ilosc

FROM [Stan magazynowy czesci] AS SMC

WHERE SMC.ID\_placowki=@placowka\_docelowa AND SMC.ID\_czesci = @id\_czesci

IF @stan IS NULL--sprawdzam czy placowka docelowa ma dane o tej czesci

BEGIN

INSERT INTO [Stan magazynowy czesci]

VALUES (@placowka\_docelowa,@id\_czesci,@ile)

END

ELSE--jak ma to dodaje do ilosci

BEGIN

UPDATE [Stan magazynowy czesci]

SET Ilosc= Ilosc + @ile

WHERE ID\_placowki=@placowka\_docelowa AND ID\_czesci = @id\_czesci

END

IF @tranCount = 0

COMMIT TRAN transferMiedzyPlacowkami

END TRY

BEGIN CATCH

ROLLBACK TRAN transferMiedzyPlacowkami

EXEC Bledy

END CATCH

### Zamawianie

* **Argumenty:**
  + **@czesc INT**
  + **@placowka** **INT**
  + **@ile INT**
* **Opis:**
  + Procedura tworzy zamówienie na części podane argumentem.
  + Jeżeli istnieje już zamówienie z tego samego dnia to procedura podpina zamawiane części pod to zamówienie.
  + W przeciwnym wypadku tworzy nowe zamówienie składające się z części podanych argumentem.
  + Cenę i dostawcę części pobiera z tabeli **Czesci**.
  + Jeżeli procedura nie jest już objęta transakcją, to jest ona uruchamiana w poziomie izolacji **READ COMMITED**.
* **Może wyrzucić błędy:**
  + **50007**
  + **50009**

#### Kod:

CREATE PROCEDURE Zamawianie(

@czesc INT,

@placowka INT,

@ile INT

)

AS

BEGIN TRY

DECLARE @tranCount INT = @@TRANCOUNT

IF @tranCount =0

BEGIN TRAN zamawianie

IF @czesc NOT IN (SELECT ID\_czesci FROM [Czesci do naprawy])

RAISERROR(50009, -1, -1)--czesc nie istnieje

IF @placowka NOT IN ( SELECT ID\_placowki FROM Placowki)

RAISERROR(50007, -1, -1)--placowka nie istnieje

DECLARE @hurtownia INT

DECLARE @cena MONEY

SELECT @hurtownia = CDN.ID\_hurtowni, @cena = CDN.Cena

FROM [Czesci do naprawy] AS CDN

WHERE CDN.ID\_czesci = @czesc

DECLARE @zamowienie INT

SELECT @zamowienie= Z.ID\_zamowienia

FROM Zamowienia AS Z

WHERE Z.ID\_hurtowni = @hurtownia AND Z.Data\_zamowienia = CAST(GETDATE() AS DATE) AND Z.ID\_placowki = @placowka

IF @zamowienie IS NULL --nie bylo dzisiaj takiego zamowienia

BEGIN

INSERT INTO Zamowienia (ID\_hurtowni,ID\_placowki,Data\_zamowienia,Data\_dostarczenia)

VALUES(@hurtownia,@placowka,GETDATE(),NULL)

SET @zamowienie = @@IDENTITY

INSERT INTO [Szczegoly zamowien]

VALUES(@zamowienie,@czesc,@cena,@ile)

END

ELSE--bylo dzisiaj zamowienie

BEGIN

DECLARE @ile\_juz\_bylo INT -- dla sprawdzenia czy moze juz byla ta czesc

--dzisiaj zamawiana i tylko zwiekszymy ilosc

SELECT @ile\_juz\_bylo = SZ.Ilosc

FROM [Szczegoly zamowien] AS SZ

WHERE @zamowienie= SZ.ID\_zamowienia AND @czesc=SZ.ID\_czesci

IF @ile\_juz\_bylo IS NULL --dzisiaj nie bylo tej czesci w zamowieniu

BEGIN

INSERT INTO [Szczegoly zamowien]

VALUES(@zamowienie,@czesc,@cena,@ile)

END

ELSE --dopisujemy ilosc

BEGIN

UPDATE [Szczegoly zamowien]

SET Ilosc = Ilosc + @ile

WHERE @zamowienie= ID\_zamowienia AND @czesc=ID\_czesci

END

END

IF @tranCount = 0

COMMIT TRAN zamawianie

END TRY

BEGIN CATCH

ROLLBACK TRAN zamawianie

EXEC Bledy

END CATCH

### Nowa\_gwarancja

* **Argumenty:**
  + **@nazwa\_produktu** **NVARCHAR(15)**
  + **@producent NVARCHAR(24)**
  + **@kategoria NVARCHAR(15)**
  + **@klient\_id INT**
  + **@pracownik\_id INT**
  + **@gwarancja NVARCHAR(50)**
* **Opis:**
  + Procedura tworzy gwarancję dla klienta **@klient\_id**, **@pracownik\_id** sprzedaje gwarancję.
  + Jeżeli procedura nie jest już objęta transakcją, to jest ona uruchamiana w poziomie izolacji **READ COMMITED**.
  + **@gwarancja** opisuje typ gwarancji.
  + Jeżeli produkt opisany przez **@nazwa\_produktu**, **@producent** i **@kategoria** nie istnieje, to zostanie utworzony.
* **Może wyrzucić błędy:**
  + **50002**
  + **50003**
  + **50004**
  + **50010**

#### Kod:

CREATE PROCEDURE Nowa\_gwarancja(

@nazwa\_produktu nvarchar(15),

@producent nvarchar(24),

@kategoria nvarchar(15),

@klient\_id INT,

@pracownik\_id INT,

@gwarancja NVARCHAR(50)

)

AS

BEGIN TRY

DECLARE @tranCount INT = @@TRANCOUNT

IF @tranCount =0

BEGIN TRAN nowaGwarancja

IF @klient\_id NOT IN ( SELECT ID\_klienta FROM Klienci)

RAISERROR(50002, -1, -1)--klient nie istnieje

IF @pracownik\_id NOT IN ( SELECT ID\_pracownika FROM Pracownicy)

RAISERROR(50003, -1, -1)--pracownik nie istnieje

DECLARE @kategoria\_id INT

SELECT @kategoria\_id = K.ID\_kategorii

FROM Kategorie AS K

WHERE K.Nazwa= @kategoria

IF @kategoria\_id IS NULL

RAISERROR(50004, -1, -1)--to nie obslugujemy takiej kategorii

DECLARE @gwarancja\_id INT

DECLARE @dlugosc\_gwarancji INT

SELECT @gwarancja\_id = TG.ID\_typu\_gwarancji,

@dlugosc\_gwarancji=TG.[Czas trwania]

FROM [Typ gwarancji] AS TG

WHERE TG.Nazwa = @gwarancja AND TG.ID\_kategorii = @kategoria\_id

IF @gwarancja\_id IS NULL

RAISERROR(50010, -1, -1)--to gwarancja nie istnieje

DECLARE @produkt\_id INT

SELECT @produkt\_id = dbo.Znajdz\_produkt(@nazwa\_produktu,@producent,@kategoria)

IF @produkt\_id IS NULL

EXEC dbo.Nowy\_produkt @nazwa\_produktu, @producent, @kategoria, @produkt\_id OUTPUT

DECLARE @data DATE

SET @data = GETDATE()

INSERT INTO Gwarancje(ID\_produktu, ID\_klienta, ID\_pracownika, [Typ gwarancji], [Data rozpoczecia], [Data zakonczenia])

VALUES (@produkt\_id, @klient\_id, @pracownik\_id, @gwarancja\_id, @data, DATEADD(dd, @dlugosc\_gwarancji,@data))

IF @tranCount = 0

COMMIT TRAN nowaGwarancja

END TRY

BEGIN CATCH

ROLLBACK TRAN nowaGwarancja

EXEC Bledy

END CATCH

### Bledy

* **Opis:**
  + Procedura wyrzuca (**RAISERROR**) ostatni złapany błąd.

#### Kod:

CREATE PROCEDURE Bledy

AS

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE()

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

### Insert\_or\_update

* Argumenty:
  + **@Do\_dodania Czesci**(typ własny)
* Opis:
  + Procedura dodaje lub aktualizuje (jeśli nie istniały) części w magazynie zdefiniowanym przez Argument.
  + Argument przechowuje **ID\_placowki**, **ID\_czesci** oraz **Ilosc**.

#### Kod:

CREATE PROCEDURE Insert\_or\_update(@Do\_dodania Czesci READONLY)

AS

BEGIN

DECLARE @Ilosc\_do\_dodania INT = (SELECT top 1 Ilosc FROM @Do\_dodania)

DECLARE @Placowka INT = (SELECT TOP 1 ID\_placowki FROM @Do\_dodania)

DECLARE @ID\_czesci INT = (SELECT TOP 1 ID\_czesci FROM @Do\_dodania)

IF EXISTS (SELECT \* FROM [Stan magazynowy czesci] WHERE ID\_placowki = @Placowka AND ID\_czesci = @ID\_czesci)

UPDATE [Stan magazynowy czesci] SET

Ilosc = Ilosc + @Ilosc\_do\_dodania

WHERE ID\_placowki = @Placowka

AND ID\_czesci = @ID\_czesci;

ELSE

INSERT INTO [Stan magazynowy czesci] (ID\_placowki, ID\_czesci, Ilosc)

VALUES (@Placowka, @ID\_czesci, @Ilosc\_do\_dodania);

END

### WstawAlboEdytujOsoby

* **Argumenty:**
  + **@ID\_osoby INT**
  + **@Imie NVARCHAR(50)**
  + **@Nazwisko NVARCHAR(50)**
  + **@Plec CHAR(1)**
  + **@Adres NVARCHAR(60)**
  + **@NumerKontaktowy VARCHAR(24)**
  + **@Email NVARCHAR(60)**
* **Opis:**
  + Dodaje do bazy danych, lub edytuje Osobę określaną przez argumenty.

#### Kod:

CREATE PROCEDURE WstawAlboEdytujOsoby(

@ID\_osoby INT,

@Imie NVARCHAR(50),

@Nazwisko NVARCHAR(50),

@Plec CHAR(1),

@Adres NVARCHAR(60),

@NumerKontaktowy VARCHAR(24),

@Email NVARCHAR(60))

AS

IF @ID\_osoby = 0

INSERT INTO Osoby (Imie,Nazwisko,Plec,Adres,[Numer kontaktowy],[E-mail])

VALUES (@Imie,@Nazwisko,@Plec,@Adres,@NumerKontaktowy,@Email)

ELSE

UPDATE Osoby

SET

Imie = @Imie,

Nazwisko = @Nazwisko,

Plec = @Plec,

Adres = @Adres,

[Numer kontaktowy] = @NumerKontaktowy,

[E-mail] = @Email

WHERE ID\_osoby = @ID\_osoby

### UsunOsoby

* **Argumenty:**
  + **@ID\_osoby INT**
* **Opis:**
  + Usuwa z bazy danych osobę o podanym przez argument ID.

#### Kod:

CREATE PROCEDURE UsunOsoby(@ID\_osoby INT)

AS

DELETE FROM Osoby

WHERE ID\_Osoby = @ID\_osoby

## Wyzwalacze

### Przenies\_czesci

* **Cel i działanie:**
  + Celem tego **triggera** jest przeniesienie części z jednej placówki (usuwanej) do drugiej (nieusuwanej).
  + Przenosi części, tylko jeżeli istnieje placówka która nie jest usuwana, oraz jeżeli nie narusza warunków integralnościowych w innych tabelach.
* **Typ i tabela:**
  + Założony jest na tabelę **Placowki**.
  + Wyzwalacz jest typu **INSTEAD OF DELETE**.
* **Dodatkowe informacje:**
  + Trigger korzysta z typu tabelowego **Czesci**, oraz z procedury **Insert\_or\_delete**.

#### Kod:

CREATE TRIGGER Przenies\_czesci

ON Placowki

INSTEAD OF DELETE

AS

BEGIN

--Wybieramy placówkę do której wyślemy części

DECLARE @Placowka\_odbiorca INT;

SET @Placowka\_odbiorca = (SELECT TOP 1 ID\_placowki

FROM Placowki

WHERE ID\_placowki NOT IN (SELECT ID\_placowki FROM deleted));

--Jeżeli nie ma placówki-odbiorcy, to znaczy że usuwamy wszystkie placówki.

IF @Placowka\_odbiorca IS NULL

BEGIN

DELETE Placowki WHERE ID\_placowki IN (SELECT ID\_placowki FROM deleted);

RETURN;

END

--Zadeklarujmy potrzebne rzeczy

DECLARE @Przenoszone\_czesci Czesci;

DECLARE @VAL1 INT;

DECLARE @VAL2 INT;

DECLARE Iterator CURSOR

FOR (--Wybieramy części do przerucenia bez podziału na placówki.

SELECT S.ID\_czesci, SUM(Ilosc)

FROM deleted AS D

JOIN [Stan magazynowy czesci] AS S

ON (S.ID\_placowki = D.ID\_placowki)

GROUP BY ID\_czesci

)

FOR READ ONLY;

--Iterujmy

OPEN Iterator;

FETCH Iterator INTO @VAL1, @VAL2;

INSERT INTO @Przenoszone\_czesci VALUES (@Placowka\_odbiorca, @VAL1,@VAL2);

WHILE @@FETCH\_STATUS = 0

BEGIN

BEGIN TRY

--Przenieśmy części

EXEC Insert\_or\_update @Do\_dodania = @Przenoszone\_czesci;

--Kolejna iteracja

DELETE FROM @Przenoszone\_czesci;

FETCH Iterator INTO @VAL1, @VAL2;

INSERT INTO @Przenoszone\_czesci VALUES (@Placowka\_odbiorca, @VAL1, @VAL2);

END TRY

BEGIN CATCH

ROLLBACK

EXEC Bledy

BREAK

END CATCH

END

--Zamykamy iterator

CLOSE Iterator;

DEALLOCATE Iterator;

--Usuwamy części z placówek

DELETE FROM [Stan magazynowy czesci]

WHERE ID\_placowki IN (SELECT ID\_placowki FROM deleted)

--Usuwamy placówki

DELETE FROM Placowki

WHERE ID\_placowki IN (SELECT ID\_placowki FROM deleted)

END

### Blokada\_aktualizacji\_zlecen

* **Cel:**
  + Celem tego **triggera** jest zablokowanie aktualizowania **przebiegu zleceń**, jeżeli zlecenie zostało już zakończone (**Data zrealizowania** jest **Nullem**).
* **Typ:**
  + Założony jest na tabelę **Przebieg zlecen**.
  + Wyzwalacz jest typu **AFTER INSERT**.
* **Dodatkowe informacje:**
  + Może wyrzucić błąd **50001**.

#### Kod:

CREATE TRIGGER Blokada\_aktualizacji\_zlecen

ON [Przebieg zlecen]

AFTER INSERT

AS

BEGIN

DECLARE @data DATE;

--Sprawdzenie czy zlecenie zostało już zakończone

SET @data = (SELECT TOP 1 Z.[Data zrealizowania]

FROM Zlecenia AS Z

JOIN inserted AS I ON (Z.ID\_zlecenia = I.ID\_zlecenia)

WHERE Z.[Data zrealizowania] IS NOT NULL) --Jeśli wszystkie będą NULLem to zapytanie nic nie zwróci (czyli NULL)

IF @data IS NOT NULL

BEGIN

RAISERROR(50001, -1, -1);

END

END

### Stworz\_klienta

* **Cel:**
  + Celem tego **triggera** jest automatyczne stworzenie klienta, po dodaniu nowego rekordu do tabeli **osoby**.
* **Typ:**
  + Założony jest na tabelę **Osoby**.
  + Trigger jest typu **AFTER INSERT**.
* **Dodatkowe informacje:**
  + Każdy dodany klient nie posiada rabatu.

#### Kod:

CREATE TRIGGER Stworz\_klienta

ON Osoby

AFTER INSERT

AS

BEGIN

INSERT INTO Klienci(ID\_Klienta, Rabat\_Indywidualny, Data\_Rejestracji)

SELECT I.ID\_osoby, NULL, GETDATE()

FROM inserted AS I

END

### Przeslij\_czesci\_po\_zakonczeniu\_zamowienia

* **Cel:**
  + Celem tego **triggera** jest przesłanie wszystkich części z właśnie zrealizowanych zamówień (zamiana **Data\_dostarczenia** z **NULL’a** na jakąś datę) do placówek na które zostały one zlecone.
  + Jeżeli data była już wpisana i użytkownik będzie chciał ją zamienić na **NULL’a**, to operacja ta zostanie przerwana ponieważ takie działanie mogło by doprowadzić do wielokrotnego przesyłania do magazynu tych samych części.
* **Typ:**
  + Założony jest na tabelę **Zamowienia**.
  + Wyzwalacz jest typu **AFTER UPDATE**.
* **Dodatkowe informacje:**
  + Może wyrzucić błąd **50013**.
  + Trigger korzysta z typu tabelowego **Czesci**, oraz z procedury **Insert\_or\_delete**.

#### Kod:

CREATE TRIGGER Przeslij\_czesci\_po\_zakonczeniu\_zamowienia

ON Zamowienia

AFTER UPDATE

AS

BEGIN

--Jeżeli użytkownik próbuje zmienić datę na NULL to operacja powinna zostać przerwana.

--Gdyby użytkownik mógł zmienić na NULL'a to wtedy mógłby znowu ustawić datę zakończenia co ponownie przesłałoby towary.

IF EXISTS (SELECT \*

--FROM deleted AS D

--JOIN inserted AS I ON (D.ID\_zamowienia = I.ID\_zamowienia)

FROM Zamowienia AS D

JOIN Zamowienia aS I ON (D.ID\_zamowienia = I.ID\_zamowienia)

WHERE D.Data\_dostarczenia IS NOT NULL AND I.Data\_dostarczenia IS NULL

)

BEGIN

RAISERROR(50013, -1, -1);

RETURN;

END

--Zadeklarujmy pomocnicze zmienne

DECLARE @Przenoszone\_czesci Czesci;

DECLARE @VAL1 INT;

DECLARE @VAL2 INT;

DECLARE @VAL3 INT;

DECLARE Iterator CURSOR

FOR (

SELECT I.ID\_placowki, SZ.ID\_czesci, SUM(SZ.Ilosc)

FROM inserted AS I

JOIN deleted AS D ON (D.ID\_zamowienia = I.ID\_zamowienia)

JOIN [Szczegoly zamowien] AS SZ

ON (SZ.ID\_zamowienia = I.ID\_zamowienia)

WHERE D.Data\_dostarczenia IS NULL

AND I.Data\_dostarczenia IS NOT NULL --Sprawdzamy czy zamówienie

--zostało zrealizowane

GROUP BY I.ID\_placowki, SZ.ID\_czesci

)

FOR READ ONLY;

--Iterujemy po wszystkich częściach

OPEN Iterator;

FETCH Iterator INTO @VAL1, @VAL2, @VAL3;

INSERT INTO @Przenoszone\_czesci VALUES (@VAL1, @VAL2, @VAL3);

WHILE @@FETCH\_STATUS = 0

BEGIN

BEGIN TRY

--Przenosimy części

EXEC Insert\_or\_update @Do\_dodania = @Przenoszone\_czesci;

--Wybieramy kolejne czesci

DELETE FROM @Przenoszone\_czesci;

FETCH Iterator INTO @VAL1, @VAL2, @VAL3;

INSERT INTO @Przenoszone\_czesci VALUES (@VAL1, @VAL2, @VAL3);

END TRY

BEGIN CATCH

ROLLBACK

EXEC Bledy

BREAK

END CATCH

END

--Zamykamy iterator

CLOSE Iterator;

DEALLOCATE Iterator;

END

### Blokada\_zwolnienia\_pracownika\_z\_niezrealizowanymi\_zleceniami

* **Cel:**
  + Celem tego **triggera** jest zablokowanie oznaczenia pracownika jako zwolnionego, jeżeli realizuje on jakieś nieskończone jeszcze zlecenie.
* **Typ:**
  + Założony jest na tabelę **Pracownicy**.
  + Wyzwalacz jest typu **AFTER UPDATE**.
* **Dodatkowe informacje:**
  + Może wyrzucić błąd **50014**.

#### Kod:

CREATE TRIGGER Blokada\_zwolnienia\_pracownika\_z\_niezrealizowanymi\_zleceniami

ON Pracownicy

AFTER UPDATE

AS

BEGIN

IF EXISTS (

SELECT I.ID\_pracownika

FROM inserted AS I

JOIN deleted AS D ON (I.ID\_pracownika = D.ID\_pracownika)

JOIN Zlecenia AS Z ON (Z.ID\_pracownika = I.ID\_pracownika)

--Sprawdzamy czy pracownik został zwolniony

WHERE (D.[Data zwolnienia] IS NULL AND I.[Data zwolnienia] IS NOT NULL)

AND (Z.[Data zrealizowania] IS NULL)

)

BEGIN

ROLLBACK;

RAISERROR(50014, -1, -1);

END

END

## Typy własne

### Czesci

* **Typ tabelowy postaci:**

|  |  |  |
| --- | --- | --- |
| ID\_placowki | ID\_czesci | Ilosc |
| INT  NOT NULL | **INT**  **NOT NULL** | **INT**  **NOT NULL**  **CHECK >= 0** |

* **Zastosowanie:**
  + Argument w procedurze **Insert\_or\_update**.
  + Stosowany w wyzwalaczach:
    - **Przenies\_czesci**
    - **Przeslij\_czesci\_po\_zakonczeniu\_zamowienia**

#### Kod:

CREATE TYPE Czesci AS TABLE

(

ID\_placowki INT NOT NULL,

ID\_czesci INT NOT NULL,

Ilosc INT NOT NULL CHECK(Ilosc >= 0)

);

## Kody błędów

|  |  |
| --- | --- |
| Kod błędu | Treść |
| 50001 | „Zlecenie zostało już zakończone, nie możesz zaktualizować przebiegu” |
| 50002 | „Nieprawidlowy klient” |
| 50003 | „Nieprawidlowy pracownik” |
| 50004 | „Nieprawidlowa kategoria” |
| 50005 | „Nieprawidlowy rabat” |
| 50006 | „Nieprawidlowe zlecenie” |
| 50007 | „Nieprawidlowa placowka” |
| 50008 | „Nieprawidlowa placowka docelowa” |
| 50009 | „Nieprawidlowa czesc” |
| 50010 | „Nieprawidlowa gwarancja” |
| 50011 | „Brak czesci w placówce” |
| 50012 | „Za malo części w placowce” |
| 50013 | „Nie możesz zmienić daty zakończenia zamówienia z powrotem na NULL!” |
| 50014 | „Pracownik nie ukonczyl wszystkich zlecen!” |

## Strategia pielęgnacji bazy danych

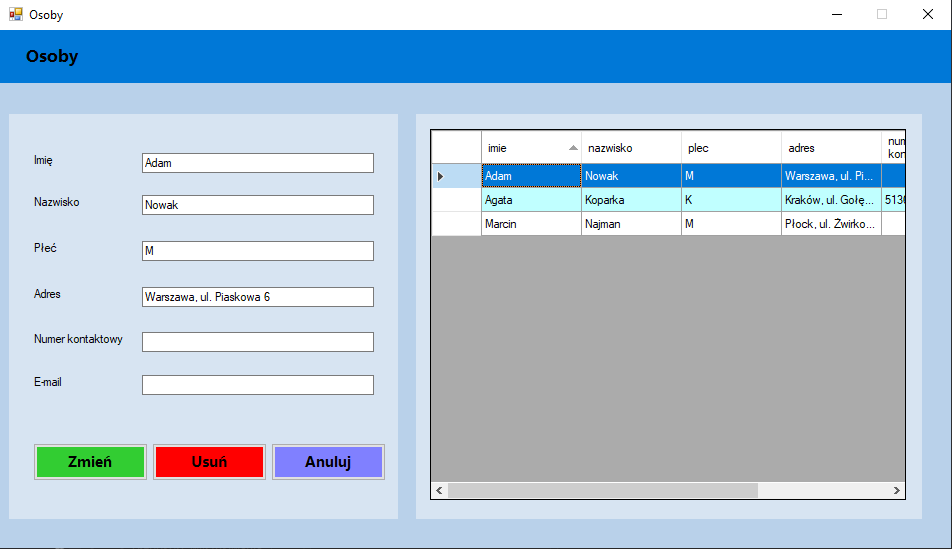
* Pełna kopia zapasowa Bazy Danych będzie tworzona co 4 dni.
* Kopia różnicowa będzie tworzona co 1 dzień.
* Co 30 minut kopia zapasowa dziennika transakcji.

Baza danych, ze względu na swoją specyfikację szybko może urosnąć do dużych rozmiarów, więc pełna kopia zapasowa jest tworzona tylko co 4 dni, a różnicowa co 1.

Ze względu na kluczowość dziennika transakcji, jego kopia jest robiona co 30 minut.

## Aplikacja

Dodatkowym elementem projektu jest aplikacja obsługująca podstawowe działania (wyświetlanie, dodawanie, edytowanie, usuwanie) na tabeli **Osoby.**



Aplikacja po prawej stronie wyświetla aktualną zawartość tabeli **Osoby** z pominięciem wartości ID\_osoby. Korzystając z textboxów i zielonego przycisku “Zapisz” możemy dodawać kolejne rekordy do tabeli. Edytowanie rekordów odbywa się poprzez podwójne kliknięcie na wybrany rekord, co sprawia, że textboxy wypełniają się wartościami rekordu, a zielony przycisk zamiast “Zapisz” wyświetla “Zmień”. Analogicznie do edytowania możemy usunąć rekord poprzez podwójne jego kliknięcie oraz wybranie opcji “Usuń”. Przycisk “Anuluj” czyści zawartość textboxów. Aplikacja wyświetla również komunikaty dotyczące przebiegu podejmowanych przez użytkownika działań. Komunikat o błędzie pojawi się w przypadku wpisania niepoprawnych danych podczas zapisu lub edycji oraz w przypadku chęci usunięcia osoby, która jest pracownikiem lub jako klient podejmowała jakieś działania jak np. utworzenie zlecenia naprawy.