Piotr Górkowski, 184515 23.12.2021

Temat bazy danych

1. Krótki opis bazy danych:

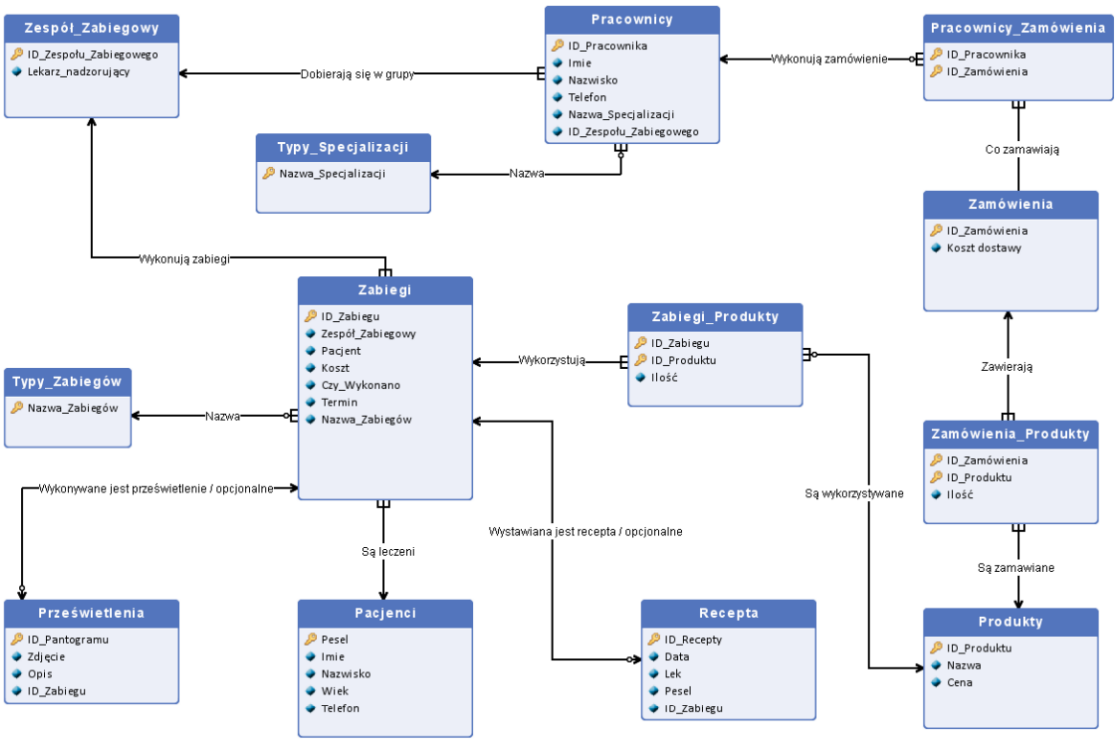
   Baza danych służąca do administrowania gabinetem stomatologicznym.

   Klientem jest właściciel, który zamierza nadzorować pracą gabinetu stomatologicznego.

   Celem bazy jest zarządzanie i nadzorowanie gabinetem stomatologicznym, w celu usprawnienia jego działań oraz archiwizacji danych na temat wykonanych zabiegów. Użytkownikiem jest właściciel, albo osoba która sprawuje pieczę nad działalnością gabinetu.

   Klient wymaga by baza przechowywała szczegółowe informację o przeprowadzonych zabiegach, grupach zabiegowych które je wykonały, o wykorzystanych produktach, o koszcie usługi, o ewentualnych receptach i prześwietleniach. Chciałby filtrować zabiegi wykonywane przez dane grupy zabiegowe, albo przeprowadzone na danym pacjencie. Jakie produkty zamówił dany pracownik, w celu uzupełnienia braków czy jaki lekarz nadzoruję dany zespół zabiegowy. Dodatkowo klient chce mieć wgląd w stan magazynu gabinetu, aby nadzorować o stanie produktów potrzebnych do zabiegów.

2. Schemat graficzny bazy danych (diagram ERD):



3. Opis zbioru encji (typy, klucze, ...)

Pracownicy

Zbiór wszystkich pracowników pracujących w gabinecie stomatologicznym, kontakt do nich oraz ich specjalizacja. Każdy pracownik należy do jednego zespołu zabiegowego. Dodawane podczas zatrudnienia nowego pracownika. Nie są usuwane po zwolnieniu się pracownika (poprzez archiwizację). Liczebność – kilkanaście. Roczny przyrost około 1-2 (gdy będzie potrzeba zatrudnienia nowego pracownika).

Name	Primary key	Type/Domain	Description
ID_Pracownika	**Yes**	Tekst – 5 znaków bez spacji w formacie, PXXXX	Identyfikator pracownika

```
ID_Pracownika int identity PRIMARY KEY NOT NULL,
```

Zamiana ID z PXXXX na automatycznie inkrementowanego inta

Imie	No	Tekst, bez spacji, maksymalnie 40 znaków	Imię pracownika

```
Imie varchar(40),
```

Nazwisko	No	Tekst, bez spacji, maksymalnie 80 znaków	Nazwisko pracownika

```
Nazwisko varchar(80),
```

Telefon	No	Tekst, cyfry w formacie XXX-XXX-XXX	Numer telefonu do pracownika

```
Telefon char(11) CHECK (Telefon like
'[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9]'),
```

Nazwa_Specjalizacji	No	Tekst, identyfikator pobrany ze zbioru encji Typy_Specjalizacji	Specjalizacja pracownika

```
Nazwa_Specjalizacji varchar(100),
FOREIGN KEY (Nazwa_Specjalizacji) references Typy_Specjalizacji
(Nazwa_Specjalizacji) ON UPDATE CASCADE ON DELETE SET NULL,
```

ID_Zespołu_Zabiegowego	No	Tekst – 5 znaków w formacie ZZXXX	Zespół wykonujący zabieg

```
ID_Zespolu_Zabiegowego char(5) CHECK
(ID_Zespolu_Zabiegowego like 'ZZ[0-9][0-9][0-9]'),
```

```
FOREIGN KEY (ID_Zespolu_Zabiegowego) references Zespol_Zabiegowy
(ID_Zespolu_Zabiegowego) ON UPDATE CASCADE
```

Zamówienia			
Zbiór wszystkich zamówień produktów wykonanych w gabinecie stomatologicznym. Dodawane po złożeniu zamówienia. Nie są usuwane (poprzez archiwizację). Liczebność – kilka tysięcy. Przyrost około kilkadziesiąt rocznie.			
Name	**Primary key**	**Type/Domain**	**Description**
ID_Zamówienia	**Yes**	Tekst – 7 znaków bez spacji w formacie, ZAXXXXX	Numer identyfikacyjny zamówienia

```
ID_Zamowienia int identity PRIMARY KEY NOT NULL,
```

Zamiana ID z PXXXX na automatycznie inkrementowanego inta

Koszt dostawy	No	Dodatnia liczba zmiennoprzecinkowa, z przedziału (0, 9999999>	Koszt zamówienia

```
Koszt_dostawy float CHECK (Koszt_dostawy > 0 AND Koszt_dostawy <= 9999999) NOT NULL
```

Zabiegi			
Zbiór wszystkich wykonanych lub zaplanowanych zabiegów w gabinecie. Wyróżniamy pacjenta, zespół zabiegowy, datę wykonania zabiegu, koszt. O tym czy zabieg jest zaplanowany czy wykonany mówi atrybut Czy_Wykonano. Dodawane podczas rejestracji nowego zabiegu. Nie są usuwane (poprzez archiwizację). Liczebność - kilkadziesiąt tysięcy. Przyrost około kilka tysięcy rocznie.			
Name	**Primary key**	**Type/Domain**	**Description**
ID_Zabiegu	**Yes**	Tekst – 7 znaków w formacie ZBXXXXX	Numer identyfikacyjny zabiegu

```
ID_Zabiegu int identity PRIMARY KEY NOT NULL,
```

Zamiana ID z ZBXXXXX na automatycznie inkrementowanego inta

Zespół_Zabiegowy	No	Tekst – 5 znaków w formacie ZZXXX	Zespół wykonujący zabieg

```
Zespol_Zabiegowy char(5) CHECK (Zespol_Zabiegowy like 'ZZ[0-9][0-9][0-9]') NOT NULL,
FOREIGN KEY (Zespol_Zabiegowy) references Zespol_Zabiegowy (ID_Zespolu_Zabiegowego)
ON UPDATE CASCADE,
```

Pacjent	No	Tekst, 11 znaków	Leczony pacjent, jego pesel

```
Pacjent varchar(11) NOT NULL,
FOREIGN KEY (Pacjent) references Pacjenci (Pesel) ON UPDATE CASCADE,
```

Koszt	No	Liczba naturalna z przedziału (0, 99999>	Koszt zabiegu

```
Koszt float CHECK (Koszt > 0 AND Koszt <= 99999) NOT NULL,
```

Czy_Wykonano	No	TAK lub NIE	Mówi czy zabieg został już wykonany czy jest dopiero zaplanowany

```
Czy_Wykonano varchar(3) CHECK (Czy_Wykonano like 'TAK' OR Czy_Wykonano like 'NIE') NOT NULL,
```

Termin	No	Data w formacie: DD-MM-RRRR	Termin zabiegu

```
Termin date NOT NULL,
```

Zamiana fromatu z DD-MM-RRRR na MM-DD-RRRR

Nazwa_Zabiegów	No	Tekst, identyfikator pobrany ze zbioru encji Typy_Zabiegów	Nazwa zabiegu

```
Nazwa_Zabiegow varchar(100) NOT NULL,
FOREIGN KEY (Nazwa_Zabiegow) references Typy_Zabiegow (Nazwa_Zabiegow) ON UPDATE CASCADE
```

Pacjenci

Zbiór wszystkich pacjentów leczonych w gabinecie stomatologicznym. Zawiera ich dane osobowe oraz kontaktowe. Dodawane podczas rejestracji nowego pacjenta. Nie są usuwane (poprzez archiwizację). Liczebność – kilka tysięcy. Przyrost około kilkaset rocznie.

Name	Primary key	Type/Domain	Description
Pesel	**Yes**	Tekst, 11 znaków	Pesel pacjenta

```
Pesel varchar(11) PRIMARY KEY NOT NULL,
```

Imie	No	Tekst, bez spacji, maksymalnie 40 znaków	Imię pacjenta

```
Imie varchar(40),
```

Nazwisko	No	Tekst, bez spacji, maksymalnie 80 znaków	Nazwisko pacjenta

```
Nazwisko varchar(80),
```

Wiek	No	Liczba naturalna z przedziału (0, 150>	Wiek pacjenta

```
Wiek int CHECK (Wiek > 0 AND Wiek <= 150),
```

Telefon	No	Tekst, cyfry w formacie XXX-XXX-XXX	Numer kontaktowy pacjenta

```
Telefon char(11) CHECK (Telefon like '[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9]')
```

Recepta			
Zbiór wszystkich recept wystawionych w gabinecie stomatologicznym. Opcjonalne podczas wykonywania zabiegu. Dodawane podczas wystawienia recepty. Nie są usuwane (poprzez archiwizację). Liczebność – kilkadziesiąt tysięcy. Przyrost około kilka tysięcy rocznie.			
Name	**Primary key**	**Type/Domain**	**Description**
ID_Recepty	**Yes**	Tekst – 6 znakow w formacie RXXXXX	Numer identyfikacyjny recepty
`ID_Recepty int identity PRIMARY KEY NOT NULL,`			
Zamiana ID z RXXXXX na automatycznie inkrementowanego inta			
Data	No	Data w formacie: DD-MM-RRRR	Data wystawienia recepty
`Data date NOT NULL,`			
Zamiana fromatu z DD-MM-RRRR na MM-DD-RRRR			
Lek	No	Tekst, maksymalnie 80 znaków	Lek zapisany na receptę
`Lek varchar(80) NOT NULL,`			
Pesel	No	Tekst, 11 znaków	Pesel pacjenta, który otrzymał receptę
`Pesel varchar(11) NOT NULL,`			
`FOREIGN KEY (Pesel) references Pacjenci (Pesel) ON UPDATE CASCADE,`			
ID_Zabiegu	No	Tekst – 7 znaków w formacie ZBXXXXX	Numer identyfikacyjny zabiegu
`ID_Zabiegu int,`			
`FOREIGN KEY (ID_Zabiegu) references Zabiegi (ID_Zabiegu) ON DELETE CASCADE`			
Jak w ww. tabeli Zabiegi zmiana formatu ID			

Produkty			
Zbiór prduktów używanych w gabinecie stomatologicznym. Chodzi o typ produktów, a nie o dokładny egzemplarz (np. wszystkie nici dentystyczne firmy ABC, a nie poszczególne sztuki). Dodawane podczas wprowadzania nowego produktu do bazy. Nie są usuwane (poprzez archiwizację). Liczebność – kilkaset. Przyrost około kilkadziesiąt rocznie.			
Name	**Primary key**	**Type/Domain**	**Description**
ID_Produktu	**Yes**	Tekst – 7 znaków w formacie MXXXXXX	Numer identyfikacyjny produktu
`ID_Produktu int identity PRIMARY KEY NOT NULL,`			
Zamiana ID z MXXXXXX na automatycznie inkrementowanego inta			
Nazwa	No	Tekst, maksymalnie 80 znaków	Nazwa produktu

```
Nazwa varchar(80) NOT NULL,
```

Cena	No	Liczba zmiennoprzecinkowa, z przedziału (0, 99999>	Cena produktu

```
Cena float CHECK (Cena > 0 AND Cena <= 99999) NOT NULL
```

Zabiegi_Produkty			
Zbiór encji łączący tabelę Zabiegi oraz Produkty. Klucz główny jest złożony z ID_Zabiegu i ID_Produktu. Tabela przyporządkująca produkty używane do poszczególnych zabiegów. Nowa encja powstaje, gdy pojawia się nowa encja w tabeli Zabiegi. Encje nie są usuwane. Liczebność – kilkadziesiąt tysięcy. Przyrost około kilka tysięcy rocznie.			
Name	**Primary key**	**Type/Domain**	**Description**
Klucz złożony `PRIMARY KEY (ID_Zabiegu, ID_Produktu),`			
ID_Zabiegu	**Yes**	Tekst – 7 znaków w formacie ZBXXXXX	Numer identyfikacyjny zabiegu

`ID_Zabiegu int NOT NULL,`
`FOREIGN KEY (ID_Zabiegu) references Zabiegi (ID_Zabiegu) ON DELETE CASCADE,`
Jak w ww. tabeli Zabiegi zmiana formatu ID			

ID_Produktu	**Yes**	Tekst – 7 znaków w formacie MXXXXXX	Numer identyfikacyjny produktu

`ID_Produktu int NOT NULL,`
`FOREIGN KEY (ID_Produktu) references Produkty (ID_Produktu) ON DELETE CASCADE`
Jak w ww. tabeli Produkty zmiana formatu ID			

Ilość	No	Liczba natualna z przedziału (0, 99999>	Ile sztuk produktu zużyto podczas zabiegu
`Ilosc int CHECK (Ilosc > 0 AND Ilosc <= 99999) NOT NULL,`			

Zespół_Zabiegowy			
Zbiór zespołów zabiegowych, w które dobierają się pracownicy. Każdym zespołem nadzoruje jeden lekarz. Dodawany podczas utworzenia nowego zespołu zabiegowego. Nie są usuwane (poprzez archiwizację). Liczebność – kilkadziesiąt. Przyrost około 1-2 (kiedy poprzedni zespół się rozwiąże).			
Name	**Primary key**	**Type/Domain**	**Description**
ID_Zespołu_Zabiegowego	**Yes**	Tekst – 5 znaków w formacie ZZXXX	Numer identyfikujący zespół zabiegowy

```
ID_Zespolu_Zabiegowego char(5) CHECK (ID_Zespolu_Zabiegowego like 'ZZ[0-9][0-9][0-9]') PRIMARY KEY,
```

Lekarz_nadzorujący	No	Tekst – 5 znaków bez spacji w formacie, PXXXX	Jeden pracownik, który nadzoruje zespołem

```
Lekarz_nadzorujacy int          --opiekun
```

Jest to ID pracownika dlatego format jest zmieniony z PXXXX na int

Prześwietlenia			
Zbiór wszystkich prześwietleń zrobionych w gabinecie stomatologicznym. Dodawane podczas wykonywania prześwietlenia. Nie są usuwane (poprzez archiwizację). Liczebność – kilkadziesiąt tysięcy. Przyrost około kilkaset rocznie.			
Name	**Primary key**	**Type/Domain**	**Description**
ID_Pantogramu	**Yes**	Tekst – 7 znaków w formacie PAXXXXX	Numer identyfikacyjny prześwietlenia

```
ID_Pantogramu int identity PRIMARY KEY NOT NULL,
```

Zamiana ID z PAXXXXX na automatycznie inkrementowanego inta

Zdjęcie	No	Tekst, maksymalnie 120 znaków	Ścieżka do zdjęcia

```
Zdjecie varchar(120) NOT NULL,
```

Opis	No	Tekst, maksymalnie 500 znaków	Opis prześwietlenia

```
Opis varchar(500),
```

ID_Zabiegu	No	Tekst – 7 znaków w formacie ZBXXXXX	Numer identyfikacyjny zabiegu

```
ID_Zabiegu int NOT NULL,
FOREIGN KEY (ID_Zabiegu) references Zabiegi (ID_Zabiegu) ON DELETE CASCADE
```

Jak w ww. tabeli Zabiegi zmiana formatu ID

Typy_Specjalizacji			
Słownik nazw specjalizacji pracowników. Liczebność – kilkanaście.			
Name	**Primary key**	**Type/Domain**	**Description**
Nazwa_Specjalizacji	**Yes**	Tekst, maksymalnie 100 znaków	Nazwa specjalizacji

```
Nazwa_Specjalizacji varchar(100) PRIMARY KEY
```

Typy_Zabiegów			
Słownik typów zabiegów. Liczebność – kilkanaście.			
Name	**Primary key**	**Type/Domain**	**Description**

Nazwa_Zabiegów	**Yes**	Tekst, maksymalnie 100 znaków	Nazwa zabiegu

```
Nazwa_Zabiegow varchar(100) PRIMARY KEY NOT NULL
```

Pracownicy_Zamówienia			
Zbiór encji łączący tabelę Pracownicy oraz Zamówienia. Klucz główny jest złożony z ID_Pracownika i ID_Zamówienia. Zamówienie wykonane przez danego pracownika. Nowa encja powstaje, gdy pracownik wykonuje nowe zamówienie. Encje nie są usuwane. Liczebność około kilka tysięcy. Przyrost około kilkaset rocznie.			
Name	**Primary key**	**Type/Domain**	**Description**
Klucz złożony			

```
PRIMARY KEY (ID_Pracownika, ID_Zamowienia),
```

ID_Pracownika	**Yes**	Tekst – 5 znaków bez spacji w formacie, PXXXX	Numer identyfikacyjny pracownika

```
ID_Pracownika int NOT NULL,
FOREIGN KEY (ID_Pracownika) references Pracownicy (ID_Pracownika) ON DELETE CASCADE,
```

Jak w ww. tabeli Pracownicy zmiana formatu ID

ID_Zamówienia	**Yes**	Tekst – 7 znaków bez spacji w formacie, ZAXXXXX	Numer identyfikacyjny zamówienia

```
ID_Zamowienia int NOT NULL,
FOREIGN KEY (ID_Zamowienia) references Zamowienia (ID_Zamowienia) ON DELETE CASCADE
```

Jak w ww. tabeli Zamówienia zmiana formatu ID

Zamówienia_Produkty			
Zbiór encji łączący tabelę Produkty oraz Zamówienia. Klucz główny jest złożony z ID_Produktu i ID_Zamówienia. Zamówione produkty podczas zamówienia. Nowa encja powstaje, gdy realizowane jest nowe zamówienie. Encje nie są usuwane. Liczebność około kilka tysięcy. Przyrost około kilkaset rocznie.			
Name	**Primary key**	**Type/Domain**	**Description**
Klucz złożony			

```
PRIMARY KEY (ID_Zamowienia, ID_Produktu),
```

ID_Zamówienia	**Yes**	Tekst – 7 znaków bez spacji w formacie, ZAXXXXX	Numer identyfikacyjny zamówienia

```
ID_Zamowienia int NOT NULL,
FOREIGN KEY (ID_Zamowienia) references Zamowienia (ID_Zamowienia) ON DELETE CASCADE,
```

Jak w ww. tabeli Zamówienia zmiana formatu ID

ID_Produktu	**Yes**	Tekst – 7 znaków w formacie MXXXXXX	Numer identyfikacyjny produktu

```
ID_Produktu int NOT NULL,
FOREIGN KEY (ID_Produktu) references Produkty (ID_Produktu) ON DELETE CASCADE
```

Jak w ww. tabeli Produkty zmiana formatu ID

Ilość	No	Liczba natualna z przedziału (0, 99999>	Ile sztuk produktu, który zamówiono.

```
Ilosc int CHECK (Ilosc > 0 AND Ilosc <= 99999) NOT NULL,
```

4. Schemat relacyjnej bazy danych

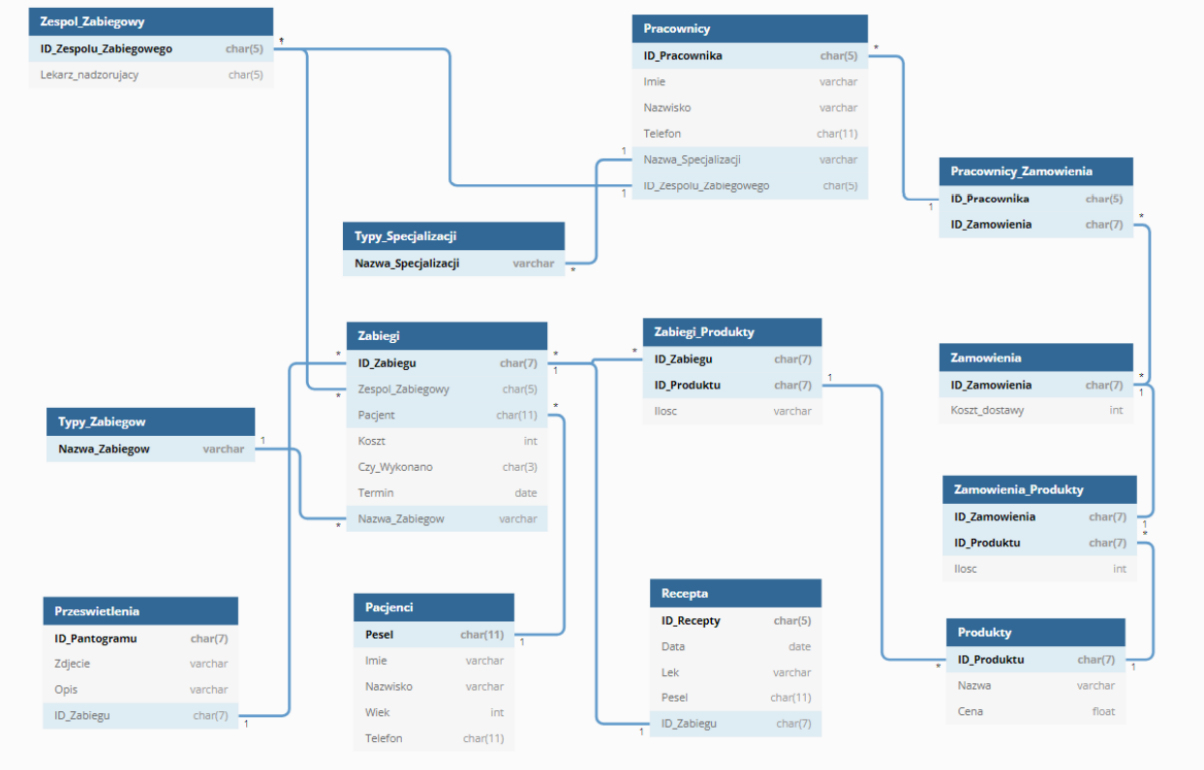


Table Zabiegi{
ID_Zabiegu char(7) pk
Zespol_Zabiegowy char(5)
Pacjent char(11)
Koszt int
Czy_Wykonano char(3)
Termin date
Nazwa_Zabiegow varchar
}
Table Typy_Zabiegow{
Nazwa_Zabiegow varchar pk
}

```
Ref: "Typy_Zabiegow"."Nazwa_Zabiegow" < "Zabiegi"."Nazwa_Zabiegow"
Table Przeswietlenia{
 ID_Pantogramu char(7) pk
 Zdjecie varchar
 Opis varchar
 ID_Zabiegu char(7)
}
Ref: "Przeswietlenia"."ID_Zabiegu" < "Zabiegi"."ID_Zabiegu"
Table Pacjenci{
 Pesel char(11) pk
 Imie varchar
 Nazwisko varchar
 Wiek int
 Telefon char(11)
}
Ref: "Pacjenci"."Pesel" < "Zabiegi"."Pacjent"
Table Recepta{
 ID_Recepty char(5) pk
 Data date
 Lek varchar
 Pesel char(11)
 ID_Zabiegu char(7)
}
Ref: "Recepta"."ID_Zabiegu" < "Zabiegi"."ID_Zabiegu"
Table Zabiegi_Produkty{
 ID_Zabiegu char(7) pk
 ID_Produktu char(7) pk
 Ilosc varchar
}
Ref: "Zabiegi"."ID_Zabiegu" < "Zabiegi_Produkty"."ID_Zabiegu"
Table Produkty{
 ID_Produktu char(7) pk
 Nazwa varchar
 Cena float
}
Ref: "Zabiegi_Produkty"."ID_Produktu" < "Produkty"."ID_Produktu"
Table Zamowienia_Produkty{
```

```
 ID_Zamowienia char(7) pk
 ID_Produktu char(7) pk
 Ilosc int
}
Ref: "Produkty"."ID_Produktu" < "Zamowienia_Produkty"."ID_Produktu"
Table Zamowienia{
 ID_Zamowienia char(7) pk
 Koszt_dostawy int
}
Ref: "Zamowienia_Produkty"."ID_Zamowienia" < "Zamowienia"."ID_Zamowienia"
Table Pracownicy_Zamowienia{
 ID_Pracownika char(5) pk
 ID_Zamowienia char(7) pk
}
Ref: "Zamowienia"."ID_Zamowienia" < "Pracownicy_Zamowienia"."ID_Zamowienia"
Table Pracownicy{
 ID_Pracownika char(5) pk
 Imie varchar
 Nazwisko varchar
 Telefon char(11)
 Nazwa_Specjalizacji varchar
 ID_Zespolu_Zabiegowego char(5)
}
Ref: "Pracownicy_Zamowienia"."ID_Pracownika" < "Pracownicy"."ID_Pracownika"
Table Typy_Specjalizacji{
 Nazwa_Specjalizacji varchar pk
}
Ref: "Pracownicy"."Nazwa_Specjalizacji" < "Typy_Specjalizacji"."Nazwa_Specjalizacji"
Table Zespol_Zabiegowy{
 ID_Zespolu_Zabiegowego char(5) pk
 Lekarz_nadzorujacy char(5)
}
Ref: "Zespol_Zabiegowy"."ID_Zespolu_Zabiegowego" < "Zabiegi"."Zespol_Zabiegowy"
Ref: "Pracownicy"."ID_Zespolu_Zabiegowego" <
"Zespol_Zabiegowy"."ID_Zespolu_Zabiegowego”
```

5. Szczegółowy opis utworzonych tabel pod kątem zastosowanych ograniczeń np. NOT NULL, UNIQUE, CHECK, DEFAULT, klucze …

```
CREATE TABLE Typy_Zabiegow (
    Nazwa_Zabiegow varchar(100) PRIMARY KEY NOT NULL
)
```

Nazwa_Zabiegow – posiada NOT NULL, ponieważ potrzebujemy w Tabeli z nazwami zabiegów nazwy, a nie pustego rekordu.

```
CREATE TABLE Pacjenci (
    Pesel varchar(11) PRIMARY KEY NOT NULL,
    Imie varchar(40),
    Nazwisko varchar(80),
    Wiek int CHECK (Wiek > 0 AND Wiek <= 150),
    Telefon char(11) CHECK (Telefon like '[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9]')
)
```

Pesel – NOT NULL, każdy pacjent musi mieć wprowadzony Pesel, **wiek** pacjenta jest sprawdzany w przedziale (0, 150> w celu minimalizacji błędów podczas wprowadzania rekordów do bazy, oraz nałożono format zapisu **numeru telefonu** w postaci XXX-XXX-XXX, poprzez co łatwiej czyta się bazę pacjentów.

```
CREATE TABLE Produkty (
    ID_Produktu int identity PRIMARY KEY NOT NULL,
    Nazwa varchar(80) NOT NULL,
    Cena float CHECK (Cena > 0 AND Cena <= 99999) NOT NULL
)
```

ID_Produktu – pomimo identity wprowadzono zabezpieczenie NOT NULL, potrzebujemy także **nazwy produktu,** ponieważ samo ID nic nam nie da podczas czytania rekordów. Każdy produkt musi mieć także wprowadzoną **cenę**, która jest sprawdzana z przedziału (0, 99999> w celu minimalizacji błędów. Cena może być używana między innymi do obliczania całkowitych kosztów dostaw.

```
CREATE TABLE Zamowienia (
    ID_Zamowienia int identity PRIMARY KEY NOT NULL,
    Koszt_dostawy float CHECK (Koszt_dostawy > 0 AND Koszt_dostawy <= 9999999) NOT NULL
)
```

ID_Zamowienia – pomimo identity jako zabezpieczenie przed rekordem bez ID jest NOT NULL, w rekordzie **Koszt_dostawy** potrzebujemy wartosci liczbowej z przedzialu (0, 9999999> w celu przyszłego obliczenia

kosztu całkowitego zamówienia, z tego powodu jest NOT NULL oraz CHECK czy wartość jest liczbowa.

```
CREATE TABLE Typy_Specjalizacji (
    Nazwa_Specjalizacji varchar(100) PRIMARY KEY
)
```

Typy_Specjalizacji– posiada NOT NULL, ponieważ potrzebujemy w tabeli z typami specjalizacji nazwy, a nie pustego rekordu.

```
CREATE TABLE Zespol_Zabiegowy (
    ID_Zespolu_Zabiegowego char(5) CHECK (ID_Zespolu_Zabiegowego like 'ZZ[0-9][0-9][0-9]') PRIMARY KEY,
    Lekarz_nadzorujacy int      --opiekun
)
```

ID_Zespolu_Zabiegowego – musi byc w konwencji ZZXXX, stąd CHECK, jakiś lekarz może być osobą odpowiedzialną za dany zespół, ale nie musi być.

```
CREATE TABLE Pracownicy (
    ID_Pracownika int identity PRIMARY KEY NOT NULL,
    Imie varchar(40),
    Nazwisko varchar(80),
    Telefon char(11) CHECK (Telefon like '[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9]'),
    Nazwa_Specjalizacji varchar(100),
    ID_Zespolu_Zabiegowego char(5) CHECK (ID_Zespolu_Zabiegowego like 'ZZ[0-9][0-9][0-9]'),
    FOREIGN KEY (Nazwa_Specjalizacji) references Typy_Specjalizacji (Nazwa_Specjalizacji) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (ID_Zespolu_Zabiegowego) references Zespol_Zabiegowy (ID_Zespolu_Zabiegowego) ON UPDATE CASCADE
)
```

ID_Pracownika – pomimo zastosowanego identity zabezpieczenie w postaci NOT NULL, **Telefon** musi być w konwencji XXX-XXX-XXX stąd narzucony CHECK, aby wszystko w bazie było w takim samym formacie. **ID_Zespolu_Zabiegowego** musi być w konwencji ZZXXX, ponieważ tylko takie rekordy figurują w tabeli Zespol_Zabiegowy. Kiedy zmienimy nazwę danej specjalizacji pracownika, to ta nazwa powinna się zaktualizować u każdego pracownika, który wcześniej posiadał daną specjalizację, z kolei kiedy usuniemy daną specjalizację u każdego pracownika powinien pojawić się NULL. Kiedy zaktualizujemy ID_Zespolu_Zabiegowego to analogicznie powinno się ono zaktualizować u każdego pracownika należącego do danego zespołu zabiegowego. Brak kaskadowego usuwania, ponieważ nie chcemy stracić rekordów pracowników podczas usunięcia np. Zespołu zabiegowego, albo specjalizacji.

```
CREATE TABLE Zabiegi (
    ID_Zabiegu int identity PRIMARY KEY NOT NULL,
    Zespol_Zabiegowy char(5) CHECK (Zespol_Zabiegowy like 'ZZ[0-9][0-9][0-9]') NOT NULL,
    Pacjent varchar(11) NOT NULL,
    Koszt float CHECK (Koszt > 0 AND Koszt <= 99999) NOT NULL,
    Czy_Wykonano varchar(3) CHECK (Czy_Wykonano like 'TAK' OR Czy_Wykonano like 'NIE') NOT NULL,
    Termin date NOT NULL,
    Nazwa_Zabiegow varchar(100) NOT NULL,
    FOREIGN KEY (Zespol_Zabiegowy) references Zespol_Zabiegowy (ID_Zespolu_Zabiegowego) ON UPDATE CASCADE,
    FOREIGN KEY (Pacjent) references Pacjenci (Pesel) ON UPDATE CASCADE,
    FOREIGN KEY (Nazwa_Zabiegow) references Typy_Zabiegow (Nazwa_Zabiegow) ON UPDATE CASCADE
)
```

ID_Zabiegu – pomimo identity zabezpieczenie w postaci NOT NULL, aby nie dostać rekordu bez ID. Każdy zabieg jest wykonywany przez dany **Zespol_Zabiegowy,** który zapisany jest w konwencji ZZXXX, z tego powodu CHECK, który waliduje nazwę. Zabieg wykonywany jest na **Pacjencie,** z tego powodu NOT NULL, ponieważ nie może być zabiegu bez pacjenta. Analogicznie za każdy zabieg się płaci z tego powodu NOT NULL przy atrybucie **Koszt.** W tabeli Zabiegi znajdują się już wykonane jak i zaplanowane zabiegi, stąd atrybut **Czy_wykonano,** który wskazuje TAK lub NIE, w celach wyszukiwania w bazie potrzebujemy tego atrybuty, aby łatwiej się wyszukiwało. Tak samo potrzebujemy **Daty_Zabiegu** i **Nazwy_Zabiegu**, w celach przejrzystszego czytania rekordów i administrowania gabinetem. Podczas akutalizacji nazw **Zespolu_zabiegoowego, Peselu pacjenta** czy **Nazwy_zabiegu**, zaktualizuje to się w zabiegach. Brak usuwania kaskadowego, ponieważ nie chcemy utracić zabiegów podczas rozwiązania się np. Zespołu zabiegowego w celach archiwizacyjnych.

```
CREATE TABLE Recepta (
    ID_Recepty int identity PRIMARY KEY NOT NULL,
    Data date NOT NULL,
    Lek varchar(80) NOT NULL,
    Pesel varchar(11) NOT NULL,
    ID_Zabiegu int,
    FOREIGN KEY (Pesel) references Pacjenci (Pesel) ON UPDATE CASCADE,
    FOREIGN KEY (ID_Zabiegu) references Zabiegi (ID_Zabiegu) ON DELETE CASCADE
)
```

Podczas zapisywania recept w bazie potrzeba: **ID_Recepty, Daty** wystawienia, **Leku** oraz **Peselu** pacjenta, stąd NOT NULL przy powyższych atrybutach. Kiedy ktoś popełnił błąd i wprowadził zły **Pesel** do bazy pacjentów to przy poprawie chcemy, aby zaktualizował nam się kaskadowo. Z kolei kiedy wyrzucamy jakiś zabieg z bazy zabiegów, to przynależąca do niego recepta też powinna zostać usunięta.

```
CREATE TABLE Zabiegi_Produkty (
    ID_Zabiegu int NOT NULL,
    ID_Produktu int NOT NULL,
    Ilosc int CHECK (Ilosc > 0 AND Ilosc <= 99999) NOT NULL,
    PRIMARY KEY (ID_Zabiegu, ID_Produktu),
    FOREIGN KEY (ID_Zabiegu) references Zabiegi (ID_Zabiegu) ON DELETE CASCADE,
    FOREIGN KEY (ID_Produktu) references Produkty (ID_Produktu) ON DELETE CASCADE
)
```

W tabeli asocjacyjnej Zabiegi_Produkty potrzebujemy **ID_Zabiegu** oraz **ID_Produktu,** dodatkowo potrzebna jest ilość użytego danego produktu w czasie tego zabiegu stąd NOT NULL przy powyższych atrybutach. Z reguły **zabiegi** oraz **produkty** nie są usuwane w celach archiwizacyjnych jednak kiedy zostały one dodane przez pomyłkę to usuniemy dany zabieg, bądź dany produkt i usuwamy też zużycie danego produktu podczas danego zabiegu, ponieważ nie jest to już potrzebne.

```
CREATE TABLE Przeswietlenia (
    ID_Pantogramu int identity PRIMARY KEY NOT NULL,
    Zdjecie varchar(120) NOT NULL,
    Opis varchar(500),
    ID_Zabiegu int NOT NULL,
    FOREIGN KEY (ID_Zabiegu) references Zabiegi (ID_Zabiegu) ON DELETE CASCADE
)
```

Podczas zapisywania pantogramów w bazie potrzeba: **ID_Pantogramu, Zdjęcia** (ścieżki do zdjęcia) oraz **Zabiegu** w którym prześwietlenie zostało wykonane, stąd NOT NULL przy powyższych atrybutach. Kiedy wyrzucamy jakiś zabieg z bazy zabiegów, to przynależące do niego prześwietlenie też powinno zostać usunięte.

```
CREATE TABLE Pracownicy_Zamowienia (
    ID_Pracownika int NOT NULL,
    ID_Zamowienia int NOT NULL,
    PRIMARY KEY (ID_Pracownika, ID_Zamowienia),
    FOREIGN KEY (ID_Pracownika) references Pracownicy (ID_Pracownika) ON DELETE CASCADE,
    FOREIGN KEY (ID_Zamowienia) references Zamowienia (ID_Zamowienia) ON DELETE CASCADE
)
```

W tabeli asocjacyjnej Pracownicy_Zamowienia potrzebujemy **ID_Pracownika** oraz **ID_Zamowienia,** stąd NOT NULL przy powyższych atrybutach. Z reguły pracownicy i zamówienia nie są usuwane w celach archiwizacyjnych, jednak jeżeli dany pracownik bądź zamówienia zostało dodane przez pomyłkę to należące do nich zamówienia też powinny zostać usunięte.

```
CREATE TABLE Zamowienia_Produkty (
    ID_Zamowienia int NOT NULL,
    ID_Produktu int NOT NULL,
    Ilosc int CHECK (Ilosc > 0 AND Ilosc <= 99999) NOT NULL,
    PRIMARY KEY (ID_Zamowienia, ID_Produktu),
    FOREIGN KEY (ID_Zamowienia) references Zamowienia (ID_Zamowienia) ON DELETE CASCADE,
    FOREIGN KEY (ID_Produktu) references Produkty (ID_Produktu) ON DELETE CASCADE
)
```

W tabeli asocjacyjnej Zamowienia_Produkty potrzebujemy **ID_Produktu** oraz **ID_Zamowienia** oraz **ilości** zamówionych produktów podczas zamówienia**,** stąd NOT NULL przy powyższych atrybutach. Z reguły pracownicy i zamówienia nie są usuwane w celach archiwizacyjnych, jednak jeżeli dany produkt bądź zamówienia zostało dodane przez pomyłkę to należące do nich zamówienia też powinny zostać usunięte.