Marcin Żołądkiewicz 218479

Bartosz Lenartowicz 218518

Termin zajęć: Śr. TN 1610 – 1750

Rozproszona baza danych pacjentów firmy medycznej

ROZPROSZONE I OBIEKTOWE BAZY DANYCH

# CEL I ZAKRES ZADANIA

Celem zadania był projekt oraz realizacja rozproszonej bazy danych. Do czynności wchodzących w skład projektu zaliczało się stworzenie rozproszonej bazy danych umieszczoną na co najmniej dwóch komputerach i konfiguracja wzajemnego połączenia tych baz. Bazy miała być przeznaczona dla firmy medycznej, która obsługuje wiele przychodni. Praca z rozproszoną bazą danych powinna być nieodczuwalna dla użytkownika. Użytkownik nie powinien zdawać sobie sprawy, że pracuje z rozproszoną bazą danych. Dodatkową czynnością do obsługiwania bazy było stworzenie aplikacji dostępowej dla użytkownika.

# FUNKCJONALNY OPIS DZIAŁANIA SYSTEMU

Baza danych składa się z 10 tabel rozlokowanych po 5 tabel na każdym serwerze. Tabele o tej samej nazwie połączone są ze sobą. Odczyt danych z tabel umożliwiają użytkownikowi widoki. Edycja danych możliwa jest poprzez procedury. Dostarcza to użytkownikowi możliwość tworzenia i usuwania i wyświetlania rekordów z tabel przy następujących założeniach. Przychodnię można dodać gdy adres i nazwa nie pokrywa się z jakąkolwiek inną przychodnią. Usunąć przychodnię można gdy żaden z lekarzy nie jest zapisany do przychodni. Dodawanie lekarzy jest ograniczone unikatowym imieniem i nazwiskiem lekarza. Dodatkowo każdy lekarz powinien być zapisanych do jednej kliniki. Usunięcie lekarza powoduje usunięcie badań lekarskich w bazie. Pacjent może być dodany gdy posiada unikatowe imię i nazwisko. Gdy pacjent zostanie usunięty automatycznie usuwają się wszystkie wizyty i badania pacjenta. Dodawanie wizyt lekarskich jest możliwe tylko gdy zostanie poprawnie wybrany lekarz i pacjent do badania. Nie ma ograniczeń usuwania wizyt. Dodawanie badania jest możliwe gdy zostanie przypisane do istniejącego w bazie pacjenta. Również jak w wizytach tak i badanie można usuwać bez konsekwencji.

# PRZYJĘTE ZAŁOŻENIA PODCZAS REALIZACJI

Projekt bazy danych został zrealizowany na dwóch maszynach wirtualnych. Na każdej z maszyn został zainstalowany system operacyjny w postaci serveru. Pierwsza system operacyjny to Microsoft Server 2016, a drugi to Ubuntu Server 16.04. Wybór systemów nie został przypadkowy. Oba serwery zostały wybrane ze względu na wsparcie do instalacji i konfiguracji programu MsSQL Service . Każdy z serwerów został skonfigurowany w taki sposób by posiadał statyczny adres IP. Pozwala to łączyć się zdalnie z maszyną za każdym razem w ten sam sposób. Na każdym z serwerów został zainstalowanych program do obsługi bazy danych Microsoft SQL Server. Dodatkowo na komputerze fizycznym został zainstalowany program Microsoft SQL Server Management Studio 17 umożliwiający zdalne zarządzanie bazą danych. Taka konfiguracja powodowała względnie porosty dostęp do bazy danych nie wymagający dwóch komputerów fizycznych. Po skończeniu konfiguracji baz danych należało ją jedynie przenieś na maszynę wirtualną na innym komputerze fizycznym co powodowało rozproszenie geograficzne. Następnie zostanie przedstawiony wzór projektu.

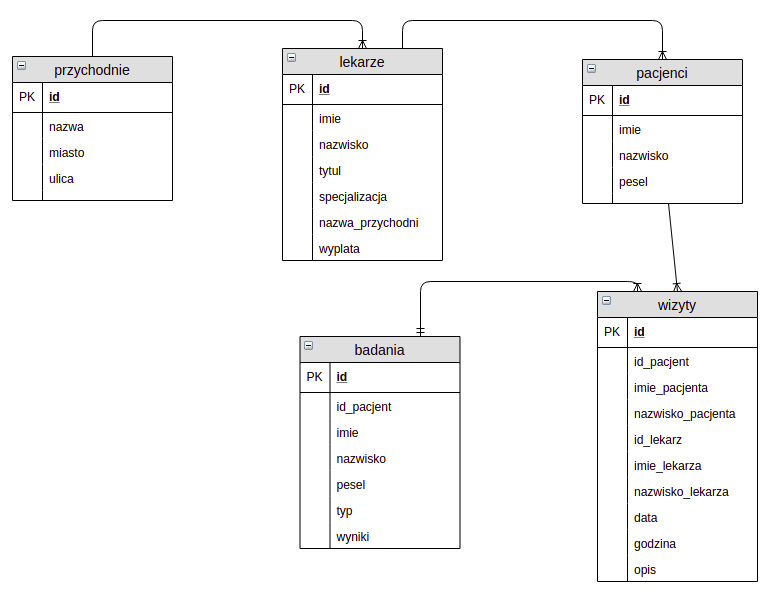
Baza obsługuje firmę medyczną. Firma składa się z kilka przychodni służby zdrowia. Tabela przychodni jest podzielona poziomo. Zawiera trzy kolumny: nazwa przychodni, miasto i ulica. W zależności czy placówka znajduje się w Nysie czy poza nią przychodnia jest umieszczona na odpowiedniej bazie. Schemat bazy jest ukazany na Rysunku 2.

Każdy lekarz przynależy do przychodni. Tabela lekarz jest podzielona pionowo. Kolumny imie i nazwisko umieszczone są w każdej bazie lecz kolumny tytul i specjalizacja umieszczona jest na jednej, a wyplata i przychodnia w której pracuje lekarz w drugiej. Schemat podziału lekarza pokazany jest na Rysunku 3.  
W każdej z baz istnieje tabela pacjentów. Oznacza to że tabela ta podzielona jest poziomo. W zależności czy id pacjenta jest parzyste umieszczone jest na odpowiedniej bazie. Pozwala to zrównoważyć obciążenie bazy.  
Każdy z pacjentów ma możliwość zapisania się do lekarza. Za taką czynność odpowiedzialna jest tabela wizyty. Tabela ta jest podzielona poziomo. W jednej bazie znajduje się data, godzina i opis wizyty, a na bazie drugiej id pacjenta.   
Ostatnią tabelą również jest tabela badania. Tabela również jak i tabele lekarz i wizyty jest dzielona pionowo. Na bazie pierwszej umieszczony jest typ i wynik badania, a w bazie drugiej id pacjenta do której przynależy dane badania.   
Rekordy tabel dzielonych pionowo są połączone za pomocą klucza głównego. Tabele połączone poziomo posiadają nie zachodzące na siebie klucze główne.

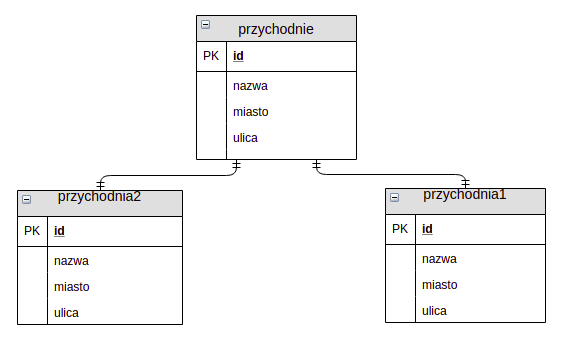
# OPIS IMPLEMENTACJI BAZ DANYCH

## Struktura systemu

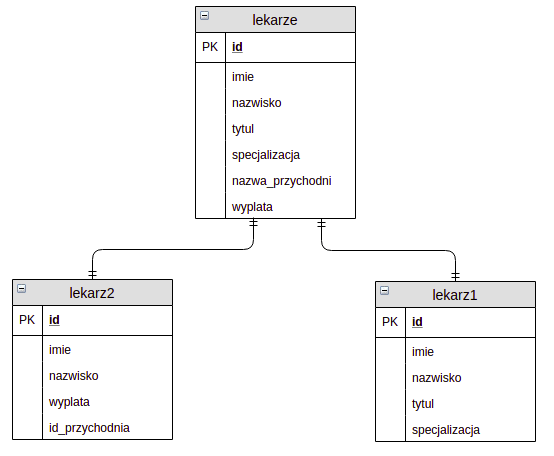
Poniżej zostały zaprezentowane diagramy UML systemu.



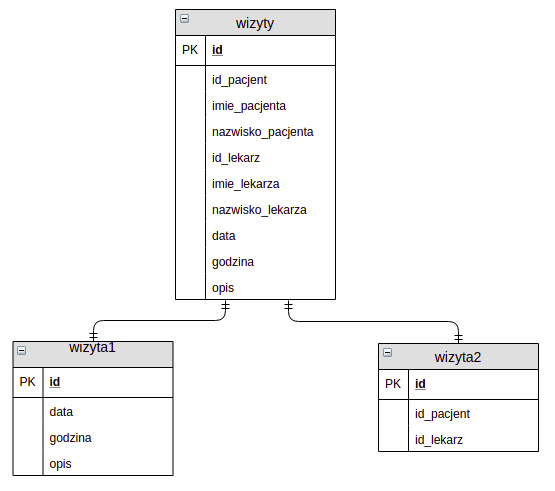
Rysunek 1 Diagram widoków bazy danych



Rysunek 2 Podział przychodni w bazach



Rysunek 3 Podział lekarzy w bazach



Rysunek 4 Podzial wizyt w bazie

Rysunek 1 przedstawia diagram UML przedstawiający dostępne widoki. Każdy widok złożony jest z dwóch lub więcej tabel umieszczonych na dwóch serwerach. Schemat podziału informacji miedzy bazami zaprezentowany jest na Rysunku 2 i 3. Na rysunku 2 widać że widok przychodnie pobiera informacje z przychodni 1 i 2 gdzie każda jest rozlokowana na osobnej maszynie.

Przykłady implementacji wybranych elementów bazy danych

Każda z opisanych procedur była wykonywana na każdym z serwerów.   
Pierwszą procedurą było usunięcie bazy danych pod warunkiem że już istnieje. Pozwalało to późniejsze uniknięcie problemów.

IF EXISTS ( SELECT \* FROM master..sysdatabases WHERE name = N'firma\_medyczna' )

BEGIN

DROP DATABASE firma\_medyczna

END

GO

Następnie została aktywowana procedura dodawania bazy danych.

CREATE DATABASE firma\_medyczna ON PRIMARY(

name = firma\_medyczna,

filename = N'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\firma\_medyczna.mdf',

size = 100MB,

maxsize = 200MB,

filegrowth = 10%)

LOG ON( name = firma\_medyczna\_log,

filename = N'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\firma\_medyczna\_log.ldf',

size = 30MB,

maxsize = 50MB,

filegrowth = 10%)

GO

Po stworzeniu bazy danych firma\_medyczna na każdym serwerze zostają stworzone tabele w przychodnie, lekarze, pacjenci, badania i wizyty na każdej z baz. Przy dodawaniu tabel uwzględnia się podział poziomy lub pionowy tabel. Dodanie tabeli prezentuje procedura poniżej:

CREATE TABLE przychodnia( int NOT NULL CONSTRAINT PK\_przychodnia PRIMARY KEY,

nazwa varchar(30) NOT NULL,

miasto varchar(30) NOT NULL,

ulica varchar(30) NOT NULL)

GO

Następnie każda z baz danych została uzupełniona o przykładowe rekordy.

Kolejnym krokiem było połączenie dwóch baz danych ze sobą. W skład połączenia wchodziło 11 procedur ustawień połączenia. Jedna z nich została przedstawiona poniżej:  
  
EXEC sp\_addlinkedserver

@server = 'Linked\_firma\_medyczna\_1',

@srvproduct = '',

@provider = 'SQLNCLI',

@datasrc = '192.168.43.5',

@catalog = 'firma\_medyczna'

GO

Następnie gdy bazy posiadały możliwość komunikacji ze sobą zostały dodane procedury synonimów ułatwiające późniejsze dodawanie widoków i procedur dodawania oraz usuwania. Implementacja synonimu przychodni przedstawiona jest poniżej:

CREATE SYNONYM przychodnia1 FOR Linked\_firma\_medyczna\_1.firma\_medyczna.dbo.przychodnia;

Następnie zostały dodane widoki tabel. Tabela przychodnie została podzielona pionowa dlatego dodawanie z tabel polega na ich połączeniu metodą UNION. Poniższa procedura prezentuje takie połączenie.

CREATE VIEW przychodnie

AS

SELECT p.id AS N'id',

p.nazwa AS N'nazwa',

p.miasto AS N'miasto',

p.ulica AS N'ulica'

FROM przychodnia1 p

UNION

SELECT p.id AS N'id',

p.nazwa AS N'nazwa',

p.miasto AS N'miasto',

p.ulica AS N'ulica'

FROM przychodnia2 p

GO

Widok tabeli dzielonej poziomo wizyty, która dodatkowo pobiera informację o pacjentach i lekarzach przedstawiony jest poniżej.   
  
CREATE VIEW wizyty

AS SELECT w1.id AS N'id',

w2.id\_pacjent AS N'id\_pacjent',

p.imie AS N'imie\_pacjenta',

p.nazwisko AS N'nazwisko\_pacjeta',

w2.id\_lekarz AS N'id\_lekarz',

l.imie AS N'imie\_lekarza',

l.nazwisko AS N'nazwisko\_lekarza',

w1.data AS N'data',

w1.godzina AS N'godzina',

w1.opis AS N'opis'

FROM wizyta1 w1

JOIN wizyta2 w2 ON (w2.id = w1.id)

JOIN pacjenci p ON (p.id = w2.id\_pacjent)

JOIN lekarze l ON (l.id = w2.id\_lekarz)

GO

W analogiczny sposób zostały dodane kolejne widoki.   
Następnie zostały dodane procedury dodawania i usuwania. Procedura dodawania lekarza została przedstawiona poniżej:

CREATE PROCEDURE addDoctor (

@imie varchar(30),

@nazwisko varchar(30),

@tytul varchar(30),

@specjalizacja varchar(30),

@id\_przychodnia int,

@wyplata int)

AS

BEGIN

DECLARE @id int

IF NOT EXISTS(SELECT \* FROM lekarze l WHERE l.imie = @imie AND

l.nazwisko = @nazwisko )

BEGIN

SELECT @id = max(l.id)+1 FROM lekarze l

IF( @id IS NULL ) SET @id = 1;

IF EXISTS(SELECT \* FROM przychodnie p WHERE p.id = @id\_przychodnia)

BEGIN

INSERT INTO lekarz1 (id, imie, nazwisko, tytul, specjalizacja)

VALUES(@id, @imie, @nazwisko, @tytul, @specjalizacja)

INSERT INTO lekarz2 (id, imie, nazwisko, wyplata, id\_przychodnia)

VALUES(@id, @imie, @nazwisko, @wyplata, @id\_przychodnia)

END

ELSE PRINT N'Clinic with id=' + @id\_przychodnia + N'not exist.'

END

ELSE PRINT N'Already exist doctor ' + @imie + N' ' + @nazwisko + N'.'

END

GO

## Sposób uruchamiania i testowania aplikacji

Aplikację testuje się za pomocą komend wykonywanych na programie zarządzającym bazą danych Microsoft Menagment SQL Service lub poprzez aplikację dostępową. Procedury w pierwszej aplikacji uruchamiane są według poniższego schematu:

EXEC dodaj\_badanie

@imie = N'Wojciech',

@nazwisko = N'Górski',

@wyniki = N'Złamanie drugiej koci śródstopia',

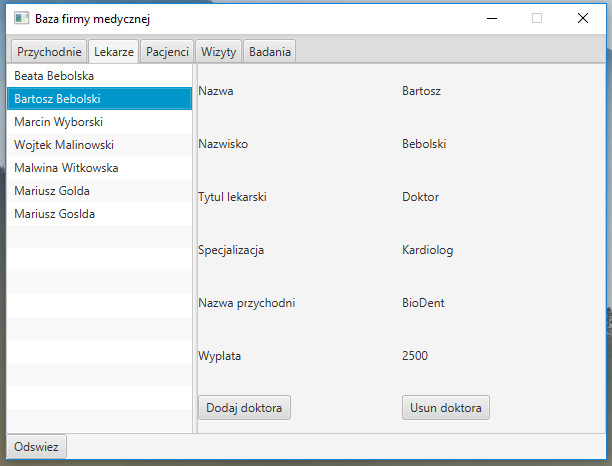
@typ = N'Rentgen'

GO

Opis korzystania z aplikacji dostępowej przedstawiony jest w następnym rozdziale. Testowanie aplikacji odbyło się za pomocą sprawdzania czy procedura generuje błędy i naprawianiu ich jeżeli było to możliwe.

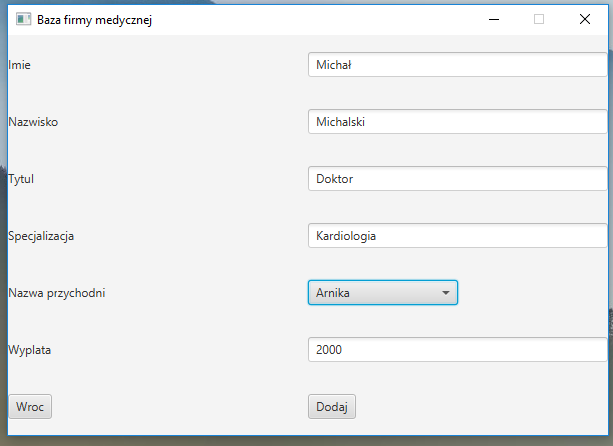
Opis implementacji aplikacji dostępowej

Aplikacja dostępowa przedstawiona jest na Rysunku 5. Aplikacja po uruchomieniu automatycznie loguje się do bazy danych. Gui posiada 5 zakładek z czego każda odpowiada jednej z tabel. Po wybraniu elementu z listy z lewej stronie detale przedstawiane są po prawej. W taki sposób pokazane są informację o przychodniach, lekarzach i pacjentach.

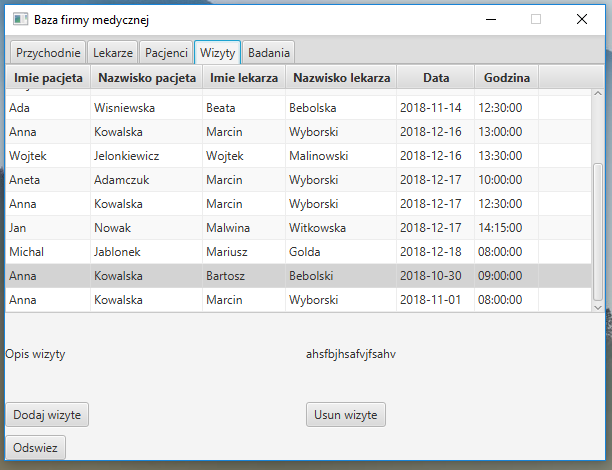


Rysunek 5 Widok zakładki lekarze

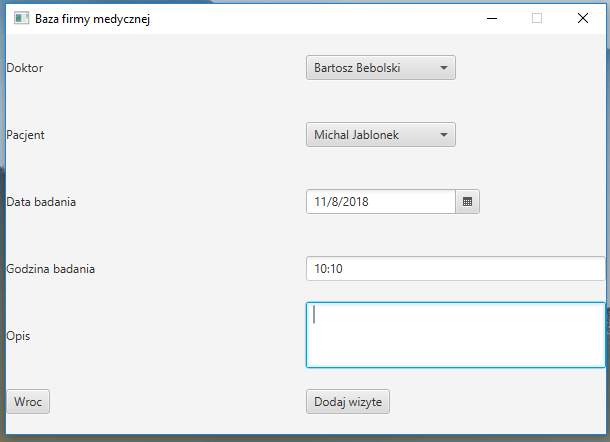
Dodatkowo po przyciśnięciu przycisku Dodaj doktora otwiera się widok dodawania lekarzy. Z listy przychodni w bazie należy wybrać tą do której chcemy dodać lekarza. Ważne jest też by uzupełnić imie i nazwisko lekarza. Gdy przejdziemy na zakładkę wizyty pokaże się tabela wizyt przedstawiona na Rysunku 7. Gdy chcemy dodać wizytę aplikacja przeniesie nas na widok zaprezentowany na Rysunku 8



Rysunek 6 DODAJ DOKTORA



Rysunek 7 WIDOK WIZYT



Rysunek 8 DODAJ WIZTE

Gdy potrzebujemy usunąć jakikolwiek element należy wybrać go z listy i przycisnąć przycisk Usun. Całość GUI jest stworzona w taki sposób by umożliwić jak najbardziej intuicyjne obsługiwanie przez użytkownika.

# Podsumowanie

W trakcie tworzenia aplikacji udało się utworzyć w pełni działający rozproszony system baz danych wraz z aplikacją dostępową. Podzielona w ten sposób baza danych pozwala na dokładniejsze ustrukturyzowanie danych. W przypadku gdy użytkownik aplikacji przeszukuje bazę w poszukiwaniu przychodni z miasta Nysa system ma ułatwione działanie (na jednej bazie znajdują się przychodnie pochodzące z danego miasta, a na następnej z innych), co wpływa pozytywnie na szybkość wykonanych zapytań. Do wad tego rozwiązania niewątpliwie można zaliczyć skomplikowaną procedurę tworzenia bazy danych, pomimo tego uważamy że korzyści przewyższają wady.

# Wykorzystane pozycje literaturowe

<https://docs.microsoft.com/en-us/sql/>

<https://docs.oracle.com/javase>

<https://www.oracle.com/technetwork/java/javafx/documentation/index.html>