Narodowa baza danych Covid-19

Michał Kuliński Jakub Małkowski

1 Podstawowe założenia projektu

1.1 Cel i główne założenia projektu

Celem bazy danych jest umożliwienie przechowywania kluczowych i najważniejszych informacji dotyczących ludzi dotkniętych wirusem oraz systemu z tym związanym. W bazie danych przechowywane są informacje o kwarantannach, szczepieniach, ludziach aktualnie hospitalizowanych oraz przeprowadzanych testach.
Ponadto w bazie danych znajdują się informacje o przychodniach, szpitalach i pogotowiu ratunkowym. Przechowywane są również informacje o stacjach sanitarno-epidemiologicznych, punktach szczepień i punktach testowania.

1.2 Ludzie

Tabela *People* jest centralnym punktem bazy danych, do którego można dodawać ludzi za pomocą procedury składowanej *AddPerson*, która jednocześnie przypisuje im przychodnię oraz punkt szczepień.

1.3 Przychodnie i testy

Część bazy danych zajmująca się przychodniami i testami oraz placówkami je wykonującymi. Dodanie przychodni umożliwia procedura składowana *AddClinic* a dodanie placówki wykonującej testy procedura *AddTestingFacility*.
Ponadto po uzyskaniu pozytywnego wyniku testu, automatycznie uruchamiany jest trigger poddający daną osobę kwarantannie, dodając ją do tabeli *Quarantined People*.
Możliwe jest również sprawdzenie wyniku ostatniego testu dowolnej osoby za pomocą funkcji *CheckTestResult*, która oczywiście zwróci odpowiednią wiadomość w przypadku, gdyby dana osoba nigdy nie była poddana testowi.
Sprawdzenie liczby infekcji na każdy region jest możliwe dzięki widokowi *InfectionsPerRegion*.

1.4 Szczepienia

Mała sekcja dotycząca szczepień oraz miejsc wykonujących szczepienia. Dodawanie takich placówek umożliwia procedura składowana *AddVaccinationFacility*. Ponadto procedura MakeEligibleForVaccination zmienia daną osobę na kwalifikującą się do szczepienia.

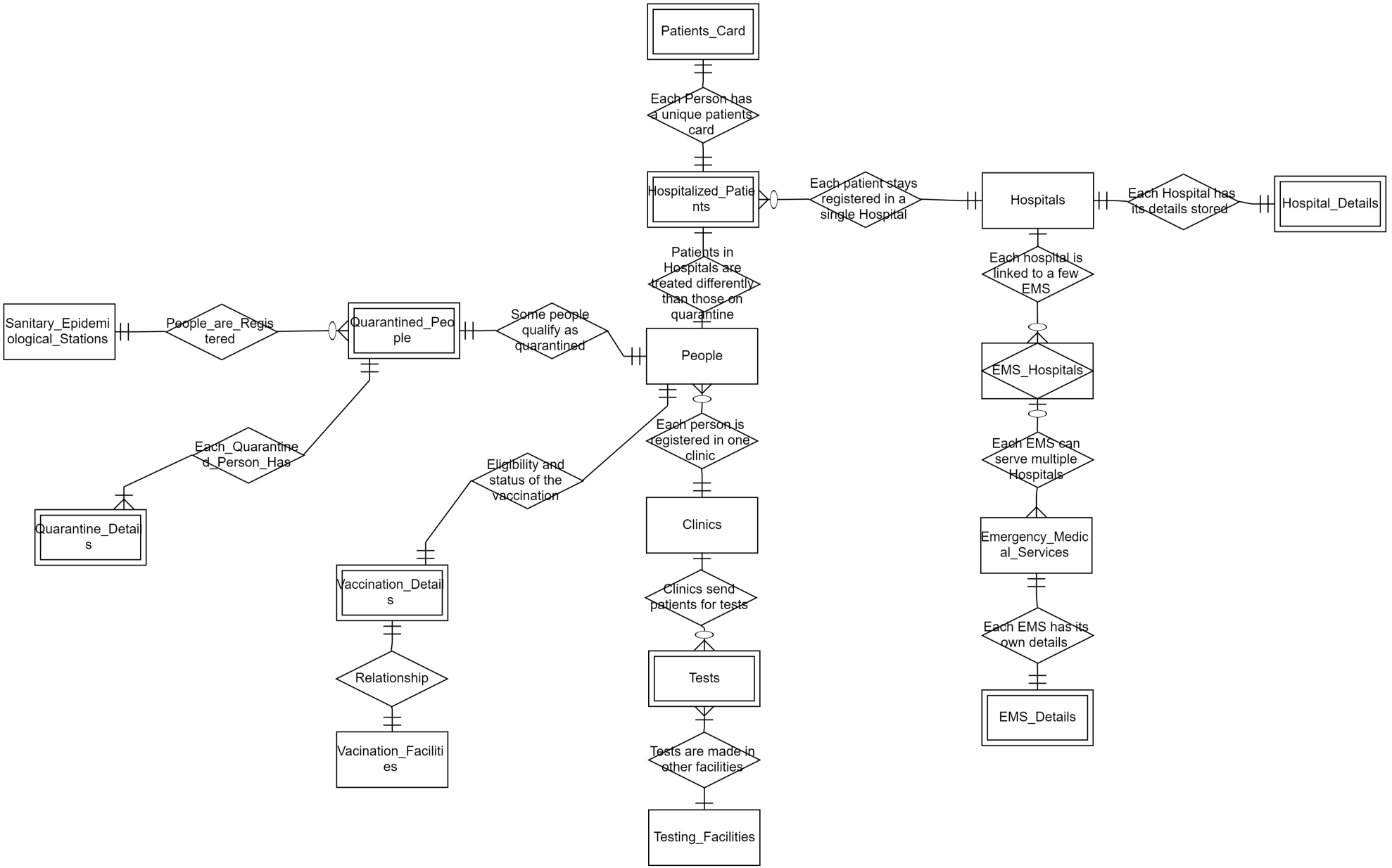
1.5 Kwarantanny i stacje sanitarno-epidemiologiczne

Część bazy zajmująca się monitorowaniem ludzi przebywających na kwarantannie, przechowywaniem o nich informacji oraz o połączonych z nimi stacjami. Wysłanie osoby na kwarantannę jest możliwe poprzez procedurę składowaną *Quarantine*, która dodaje osobę do tabeli *Quarantined People*. Ponadto dodawanie stacji jest możliwe poprzez procedurę *AddSanepid*.
Po dodaniu osoby do tabeli *Quarantined People* automatycznie uruchamiany jest trigger, który dodaje wszystkie szczegóły dotyczące kwarantanny do tabeli *Quarantine Details*.
Sprawdzenie aktualnie trwających kwarantann nałożonych przez dany sanepid jest możliwe dzięki funkcji *ActiveSanepidAssignedQuarantines*.

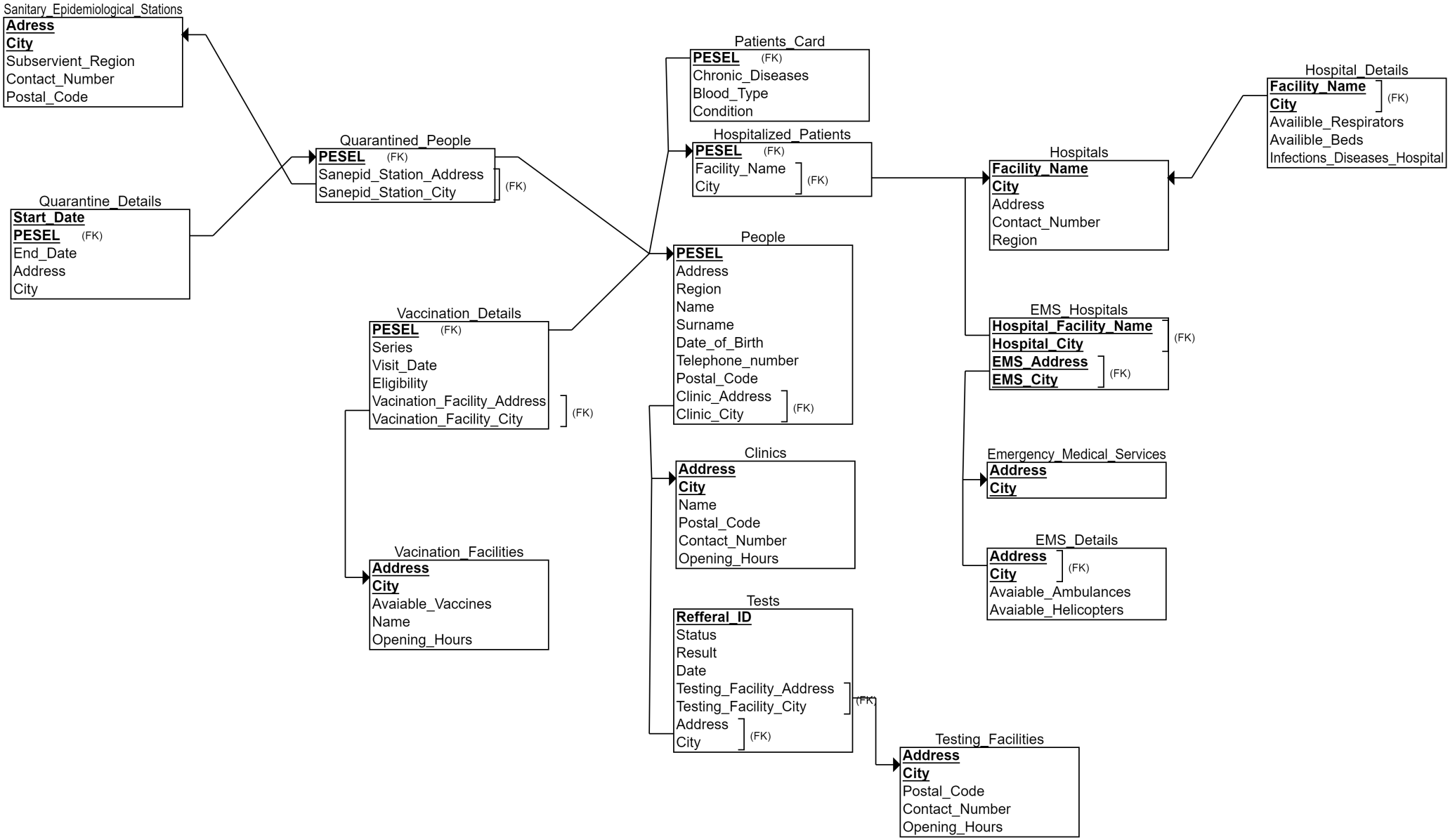
1.6 Pacjenci, szpitale i pogotwia ratunkowe

Sekcja przechowująca dane na podane powyżej tematy. Dodawanie pacjenta umożliwia procedura składowana *AddPatient*, która korzysta z funkcji *AvailableHospitals* do znalezienia szpitala z wolnym miejscem. Ponadto dodanie karty pacjenta jest możliwe poprzez procedurę *AddPatientCard* a wypis pacjenta poprzez *DischargePatient*.
Przy dodawaniu pacjenta automatycznie uruchamiany jest trigger *OccupyBed* zmniejszający liczbę wolnych łóżek o jeden w szpitalu do którego trafia pacjent. Analogiczne wypis pacjenta wiąże się ze zwiększeniem wolnych łóżek o jeden poprzez trigger *MakeBedAvailable*.
Sprawdzenie liczby wolnych łóżek jest możliwe dzięki funkcji *AvailableBeds*, a uzyskanie numeru kontaktowego poprzez funkcję *GetContactNumber*.

2 Diagram ER


Patients_Card
Each Person has a unique patients card
Hospitalized_Patients
Each patient stays registered in a single Hospital
Hospitals
Each Hospital has its details stored
Hospital_Details
Patients in Hospitals are treated differently than those on quarantine
Each hospital is linked to a few EMS
Sanitary_Epidemiological_Stations
People_are_Registered
Quarantined_People
Some people qualify as quarantined
People
EMS_Hospitals
Each_Quarantined_Person_Has
Each person is registered in one clinic
Each EMS can serve multiple Hospitals
Quarantine_Details
Eligibility and status of the vaccination
Clinics
Emergency_Medical_Services
Vaccination_Details
Clinics send patients for tests
Each EMS has its own details
Relationship
Tests
EMS_Details
Vaccination_Facilities
Tests are made in other facilities
Testing_Facilities

3 Schemat bazy danych



4 Widoki

4.1 Vaccination Eligibility

Widok przedstawiający informacje na temat kwalifikowalności do szczepienia.

```
01 |  DROP VIEW IF EXISTS dbo.Eligibility;
02 |  GO
03 |
04 |  CREATE VIEW Eligibility AS
05 |          SELECT Name, Surname, V.Eligibility From People P
06 |          INNER JOIN Vaccination_Details V ON V.PESEL = P.PESEL;
07 |  GO
```

4.2 Hospital-EMS connections

Widok przedstawiający informacje na temat współpracujących szpitali i pogotowia ratunkowego.

```
01 |  DROP VIEW IF EXISTS dbo.Hospital_EMS_Connections;
02 |  GO
03 |
04 |  CREATE VIEW Hospital_EMS_Connections
05 |  AS
06 |          SELECT H.Facility_Name, H.City, E.EMS_Address AS EMS_Address From Hospitals H
07 |          INNER JOIN EMS_Hospitals E ON E.Hospital_Facility_Name = H.Facility_Name AND E.Hospital_City = H.City
08 |          INNER JOIN EMS_Details D ON D.Address = E.EMS_Address AND D.City = E.EMS_City
09 |  GO
```

4.3 Hospitals in region

Widok przedstawiający informacje na temat szpitali w regionie oraz aktualny stan respiratorów i łóżek dla każdego z nich.

```
01 |  DROP VIEW IF EXISTS dbo.HospitalsInRegion;
02 |  GO
03 |
04 |  CREATE VIEW HospitalsInRegion AS
05 |          SELECT H.Facility_Name, H.Address, HD.Available_Beds, Hd.Available_Respirators, HD.Infections_Diseases_Hospital
06 |          FROM Hospitals H
07 |          JOIN Hospital_Details HD ON HD.Facility_Name = H.Facility_Name
08 |          ORDER BY H.Region DESC;
09 |  GO
```

4.4 Infections per region

Widok przedstawiający liczbę zakażeń na każdy region.

```
01 |  DROP VIEW IF EXISTS dbo.Infections_Per_Region;
02 |  GO
03 |
04 |  CREATE VIEW Infections_Per_Region AS
05 |          SELECT DISTINCT P.Region, (SELECT DISTINCT COUNT(T.PESEL) FROM Tests T
06 |                  GROUP BY T.Result HAVING T.Result='Positive') Infections FROM People P;
07 |  GO
```

4.5 Patients in hospital

Widok przedstawiający liczbę pacjentów na każdy szpital.

```
01 |  DROP VIEW IF EXISTS dbo.Patients_in_Hospital;
02 |  GO
03 |
04 |  CREATE VIEW Patients_in_Hospital AS
05 |          SELECT H.Facility_Name, H.Address, (SELECT Count(*) FROM Hospitalized_Patients HP
06 |          GROUP BY HP.Facility_Name
07 |          HAVING HP.Facility_Name = H.Facility_Name) AS Patients_Per_Hospital FROM Hospitals H;
08 |  GO
```

5 Procedury składowane

5.1 Add Clinic

Procedura pozwalająca dodać nową przychodnię do tabeli *Clinics*.

```
01 |  DROP PROC IF EXISTS dbo.AddClinic;
02 |  GO
03 |
04 |  CREATE PROC dbo.AddClinic
05 |
06 |  @Address VARCHAR(50),
07 |  @City VARCHAR(30),
08 |  @Name VARCHAR(50),
09 |  @Postal_Code VARCHAR(6),
10 |  @Contact_Number VARCHAR(20)
11 |
12 |  AS
13 |
14 |  INSERT INTO Clinics(Address, City, Name, Postal_Code, Contact_Number)
15 |  VALUES(@Address, @City, @Name, @Postal_Code, @Contact_Number)
16 |
17 |  GO
```

5.2 Add Person

Procedura pozwalająca dodać nową osobę do tabeli *People*. Dodatkowo zostaje jej przypisany aktualny status dotyczący szczepienia w tabeli *Vaccination Details*.

```
01 |  DROP PROC IF EXISTS dbo.AddPerson;
02 |  GO
03 |
04 |  CREATE PROC dbo.AddPerson
05 |
06 |  @Address VARCHAR(50),
07 |  @PESEL VARCHAR(11),
08 |  @Region VARCHAR(30),
09 |  @Name VARCHAR(30),
10 |  @Surname VARCHAR(30),
11 |  @Date_of_Birth DATE,
12 |  @Telephone_number VARCHAR(20),
13 |  @Postal_Code VARCHAR(6),
14 |  @Clinic_Address VARCHAR(50),
15 |  @Clinic_City VARCHAR(30),
16 |  @Vaccination_Facility_Address VARCHAR(50),
17 |  @Vaccination_Facility_City VARCHAR(30)
18 |
19 |  AS
20 |
21 |  INSERT INTO People(Address, PESEL, Region, Name, Surname, Date_of_Birth, Telephone_number, Postal_Code, Clinic_Address, Clinic_City)
22 |  VALUES(@Address, @PESEL, @Region, @Name, @Surname, @Date_of_Birth, @Telephone_number, @Postal_Code, @Clinic_Address, @Clinic_City)
23 |
24 |  INSERT INTO Vaccination_Details(PESEL, Series, Visit_Date, Eligibility, Vaccination_Facility_Address, Vaccination_Facility_City)
25 |  VALUES(@PESEL, NULL, NULL, 0, @Vaccination_Facility_Address, @Vaccination_Facility_City)
26 |
27 |  GO
```

5.3 Add Sanepid

Procedura pozwalająca dodać nowy punkt sanitarno-epidemiologiczny do tabeli *Sanitary Epidemiological Stations*.

```
01 |  DROP PROC IF EXISTS dbo.AddSanepid
02 |  GO
03 |
04 |  CREATE PROC dbo.AddSanepid
05 |
06 |  @Address VARCHAR(50),
07 |  @Subservient_Region VARCHAR(30),
08 |  @Contact_Number VARCHAR(20),
09 |  @City VARCHAR(30),
10 |  @Postal_Code VARCHAR(6)
11 |
12 |  AS
13 |
14 |  INSERT Sanitary_Epidemiological_Stations(Address, Subservient_Region, Contact_Number, City, Postal_Code)
15 |  VALUES(@Address, @Subservient_Region, @Contact_Number, @City, @Postal_Code)
16 |
17 |  GO
```

5.4 Add Testing Facility

Procedura pozwalająca dodać nowy punkt przeprowadzania testów do tabeli *Testing Facilities*.

```
01 |  DROP PROC IF EXISTS dbo.AddTestingFacility
02 |  GO
03 |
04 |  CREATE PROC dbo.AddTestingFacility
05 |
06 |  @Address VARCHAR(50),
07 |  @City VARCHAR(30),
08 |  @Postal_Code VARCHAR(6),
09 |  @Contact_Number INT
10 |
11 |  AS
12 |
13 |  INSERT INTO Testing_Facilities(Address, City, Postal_Code, Contact_Number)
14 |  VALUES(@Address, @City, @Postal_Code, @Contact_Number)
15 |
16 |  GO
```

5.5 Add Vaccination Facility

Procedura pozwalająca dodać nowy punkt przeprowadzania szczepień do tabeli *Vaccination Facilities*.

```
01 |  DROP PROC IF EXISTS dbo.AddVaccinationFacility;
02 |  GO
03 |
04 |  CREATE PROC dbo.AddVaccinationFacility
05 |
06 |  @Available_Vaccines INT,
07 |  @Address VARCHAR(50),
08 |  @City VARCHAR(30),
09 |  @Name VARCHAR(50)
10 |
11 |  AS
12 |
13 |  INSERT INTO Vaccination_Facilities(Available_Vaccines, Address, City, Name)
14 |  VALUES(@Available_Vaccines, @Address, @City, @Name)
15 |
16 |  GO
```

5.6 Make Eligible For Vaccination

Procedura pozwalająca na zmianę kwalifikowalności do szczepienia poprzez podanie numeru PESEL w tabeli *Vaccination Details*.

```
01 |  DROP PROC IF EXISTS dbo.MakeEligibleForVaccination
02 |  GO
03 |
04 |  CREATE PROC dbo.MakeEligibleForVaccination
05 |
06 |  @PESEL VARCHAR(11)
07 |
08 |  AS
09 |
10 |  UPDATE Vaccination_Details
11 |  SET Eligibility = 1
12 |  WHERE PESEL = @PESEL
13 |
14 |  GO
```

5.7 Quarantine

Procedura pozwalająca na poddanie osoby kwarantannie poprzez dodanie osoby do tabeli *Quarantined People* oraz dodanie szczegół dotyczących kwarantanny do tabeli *Quarantine Details*.

```
01 |  DROP PROC IF EXISTS dbo.Quarantine
02 |  GO
03 |
04 |  CREATE PROC dbo.Quarantine
05 |
06 |  @Start_Date DATE,
07 |  @End_Date DATE,
08 |  @Address VARCHAR(50),
09 |  @City VARCHAR(30),
10 |  @PESEL VARCHAR(11),
11 |  @Sanepid_Station_Address VARCHAR(50),
12 |  @Sanepid_Station_City VARCHAR(30)
13 |
14 |  AS
15 |
16 |  INSERT Quarantined_People(PESEL, Sanepid_Station_Address, Sanepid_Station_City)
17 |  VALUES(@PESEL, @Sanepid_Station_Address, @Sanepid_Station_City)
18 |
19 |  INSERT Quarantine_Details(Start_Date, End_Date, Address, City, PESEL)
20 |  VALUES(@Start_Date, @End_Date, @Address, @City, @PESEL)
21 |
22 |  GO
```

5.8 Add Patient Card

Procedura pozwalająca na dodanie karty pacjenta do tabeli *Patient Cards*.

```
01 |  DROP PROC IF EXISTS dbo.AddPatientCard;
02 |  GO
03 |
04 |  CREATE PROC dbo.AddPatientCard
05 |
06 |  @Chronic_Diseases VARCHAR(100),
07 |  @Blood_Type VARCHAR(10),
08 |  @Condition VARCHAR(30),
09 |  @PESEL VARCHAR(11)
10 |
11 |  AS
12 |
13 |  INSERT INTO Patients_Card(Chronic_Diseases, Blood_Type, Condition, PESEL)
14 |  VALUES(@Chronic_Diseases, @Blood_Type, @Condition, @PESEL)
15 |
16 |  GO
```

5.9 Add Patient

Procedura pozwalająca na dodanie pacjenta do tabeli *Patients*.

```
01 |  DROP PROC IF EXISTS dbo.AddPatient;
02 |  GO
03 |
04 |  CREATE PROC dbo.AddPatient
05 |
06 |  @PESEL VARCHAR(11)
07 |
08 |  AS
09 |
10 |  INSERT INTO dbo.Hospitalized_Patients(PESEL, Facility_Name, City)
11 |  SELECT TOP 1 @PESEL, H.Facility_Name, H.City FROM dbo.AVAILABLE_HOSPITALS(@PESEL) H
12 |  WHERE Available_Beds > 0
13 |  ORDER BY NEWID()
14 |  GO
```

5.10 Discharge Patient

Procedura pozwalająca na wypis pacjenta i usunięcie go z tabeli *Hospitalized Patients*.

```
01 |  DROP PROC IF EXISTS dbo.DischargePatient;
02 |  GO
03 |
04 |  CREATE PROC dbo.DischargePatient
05 |
06 |  @PESEL VARCHAR(11)
07 |
08 |  AS
09 |
10 |  BEGIN TRANSACTION
11 |          DELETE FROM Hospitalized_Patients WHERE PESEL = @PESEL
12 |  COMMIT;
```

5.11 Backup Database

Procedura tworząca kopię zapasową bazy danych.

```
01 |  DROP PROC IF EXISTS dbo.BackupDatabase
02 |  GO
03 |
04 |  CREATE PROCEDURE BackupDatabase
05 |  AS
06 |  BACKUP DATABASE TestDB
07 |  TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL11.SQLEXPRESS\MSSQL\Backup\TestDB.bak'
08 |  GO
```

6 Funkcje

6.1 Active assigned quarantines

Funkcja zwraca wszystkie bieżące kwarantanny zarządzone przez daną jednostkę sanepidu.

```
01 |  DROP FUNCTION IF EXISTS [dbo].[ACTIVE_SANEPID_ASSIGNED_QUARANTINES];
02 |  GO
03 |
04 |  CREATE FUNCTION ACTIVE_SANEPID_ASSIGNED_QUARANTINES (@SNPD_A varchar,@SNPD_C varchar)
05 |  RETURNS TABLE
06 |  AS
07 |  RETURN(
08 |          SELECT QD.PESEL,QD.Start_Date,QD.End_Date FROM Sanitary_Epidemiological_Stations T
09 |          JOIN Quarantined_People QP ON T.Address=QP.Sanepid_Station_Address AND T.City=QP.Sanepid_Station_City
10 |          JOIN Quarantine_Details QD ON QP.PESEL=QD.PESEL
11 |          WHERE T.Address = @SNPD_A AND T.City = @SNPD_C AND QD.End_Date<GETDATE()
12 |  )
13 |  GO
```

6.2 Available beds

Funkcja zwraca ilość dostępnych łóżek dla danego szpitala, wykorzystywana m.in. w funkcji *Available Hospitals*.

```
01 |  DROP FUNCTION IF EXISTS [dbo].[Available_Beds];
02 |  GO
03 |
04 |  CREATE FUNCTION Available_Beds(@HSPN varchar,@HSPC varchar)
05 |  RETURNS int
06 |  AS
07 |  BEGIN
08 |          DECLARE @BDS int;
09 |          SET @BDS = (SELECT HD.Available_Beds FROM Hospital_Details HD WHERE HD.Facility_Name=@HSPN AND HD.City=@HSPC)
10 |          RETURN @BDS;
11 |  END;
```

6.3 Available hospitals

Funkcja zwraca tabelę szpitali do których może zostać przyjęty pacjent biorąc pod uwagę region oraz liczbę dostępnych łóżek, wykorzystywana m.in. w procedurze składowanej *Add Patient*.

```
01 |  DROP FUNCTION IF EXISTS [dbo].[Available_Hospitals];
02 |  GO
03 |
04 |  CREATE FUNCTION Available_Hospitals(@PESEL varchar)
05 |          RETURNS TABLE
06 |          AS
07 |          RETURN(
08 |          SELECT H.Facility_Name,H.City,H.Address, H.Contact_Number FROM Hospitals H
09 |          INNER JOIN People P ON H.Region=P.Region AND H.Region=P.Region
10 |          WHERE P.PESEL=@PESEL AND dbo.Available_Beds(H.Address,H.City) > 0
11 |  )
12 |  GO
```

6.4 Check test result

Funkcja sprawdzająca, czy pacjent był testowany, a jeżeli tak, jaki był wynik testu.

```
01 |  DROP FUNCTION IF EXISTS [dbo].[Check_Test_Result];
02 |  GO
03 |
04 |  CREATE FUNCTION Check_Test_Result(@PESEL varchar)
05 |  RETURNS Varchar
06 |  AS
07 |  BEGIN
08 |          DECLARE @TST varchar;
09 |          IF (SELECT MAX(T.Test_Date) FROM Tests T WHERE T.Refferal_ID=@PESEL) IS NULL
10 |                  SET @TST = 'Pacjent nie byl testowany'
11 |          ELSE
12 |                  SET @TST = (SELECT TE.Result FROM Tests TE WHERE TE.Test_Date=(SELECT MAX(T.Test_Date) FROM Tests T WHERE T.Refferal_ID=@PESEL))
13 |          RETURN @TST;
14 |  END;
```

6.5 Patient's contact number

Funkcja zwracająca numer do pacjenta z podanym peselem.

```
01 |  DROP FUNCTION IF EXISTS [dbo].[GetContactNumber];
02 |  GO
03 |
04 |  CREATE FUNCTION GetContactNumber(@PESEL varchar)
05 |  RETURNS Varchar
06 |  AS
07 |  BEGIN
08 |          DECLARE @TEL varchar;
09 |          SELECT @TEL = P.Telephone_number FROM People P WHERE @PESEL = P.PESEL
10 |          RETURN @TEL;
11 |  END;
```

7 Wyzwalacze

7.1 Create quarantine details

Wyzwalacz automatycznie dodający szczegóły kwarantanny w *Quarantine Details* po poddaniu osoby kwarantannie poprzez dodanie jej do tabeli *Quarantined People*.

```
01 |  CREATE TRIGGER dbo.AutoCreateQuarantineDetails
02 |  ON Quarantined_People
03 |  AFTER INSERT
04 |  AS
05 |  BEGIN
06 |          INSERT INTO Quarantine_Details
07 |          SELECT GETDATE(), DATEADD(week,2,GETDATE()),P.Address , P.Clinic_City,QP.PESEL FROM inserted QP
08 |          JOIN People P ON QP.PESEL=P.PESEL
09 |  END
```

7.2 Generate patient card

Wyzwalacz automatycznie wywołujący procedurę dodania karty pacjenta z podanym peselem po dodaniu pacjenta do tabeli *Hospitalized Patients*.

```
01 |  DROP TRIGGER IF EXISTS dbo.AutoGeneratePatientsCard;
02 |
03 |  GO;
04 |  CREATE TRIGGER dbo.AutoGeneratePatientsCard
05 |  ON Hospitalized_Patients
06 |  AFTER INSERT
07 |  AS
08 |  BEGIN
09 |          EXEC dbo.AddPatientCard @PESEL = inserted.PESEL
10 |  END
```

7.3 Quarantine

Wyzwalacz automatycznie poddający osobę kwarantannie poprzez dodanie nowego rekordu do tabeli *Quarantined People* po uzyskaniu pozytywnego wyniku w tabeli *Tests*.

```
01 |  DROP TRIGGER IF EXISTS dbo.AutoQuarantine;
02 |  GO;
03 |
04 |  CREATE TRIGGER dbo.AutoQuarantine
05 |  ON Tests
06 |  AFTER INSERT
07 |  AS
08 |          DECLARE @TestVariable AS VARCHAR(30)
09 |          SELECT @TestVariable = i.Result FROM inserted i
10 |          IF(@TestVariable = 'Positive')
11 |                  INSERT INTO Quarantined_People
12 |                          SELECT I.PESEL, SE.Address , (SELECT TOP 1 SE.City FROM Sanitary_Epidemiological_Stations ORDER BY NEWID()) FROM People P
13 |                          JOIN inserted I ON P.PESEL=I.PESEL
14 |                          Join Sanitary_Epidemiological_Stations SE ON SE.Subservient_Region=P.Region
```

7.4 Make bed available

Wyzwalacz automatycznie zwalniający łóżko w szpitalu po usunięciu pacjenta z tabeli *Hospitalized Patients*.

```
01 |  DROP TRIGGER IF EXISTS dbo.MakeBedAvailable;
02 |  GO;
03 |
04 |  CREATE TRIGGER dbo.MakeBedAvailable
05 |  ON Hospitalized_Patients
06 |  AFTER DELETE
07 |  AS
08 |  BEGIN
09 |          BEGIN TRANSACTION
10 |          DECLARE @Fac_name AS varchar(30)
11 |          Declare @Fac_city AS varchar(30)
12 |          SELECT @Fac_name = [Facility_Name] FROM inserted
13 |          SELECT @Fac_city = [City] FROM inserted
14 |          UPDATE Hospital_Details
15 |          SET Available_Beds=Available_Beds+1
16 |          WHERE Facility_Name=@Fac_city AND City=@Fac_city
17 |          COMMIT;
18 |  END
```

7.5 Occupy bed

Wyzwalacz automatycznie zajmujący łóżko w szpitalu po dodaniu pacjenta do tabeli *Hospitalized Patients*.

```
01 |  DROP TRIGGER IF EXISTS dbo.OccupyBed;
02 |  GO;
03 |
04 |  CREATE TRIGGER dbo.OccupyBed
05 |  ON Hospitalized_Patients
06 |  AFTER INSERT
07 |  AS
08 |  BEGIN
09 |          BEGIN TRANSACTION
10 |          DECLARE @Fac_name AS varchar(30)
11 |          Declare @Fac_city AS varchar(30)
12 |          SELECT @Fac_name = [Facility_Name] FROM inserted
13 |          SELECT @Fac_city = [City] FROM inserted
14 |          UPDATE Hospital_Details
15 |          SET Available_Beds=Available_Beds-1
16 |          WHERE Facility_Name=@Fac_city AND City=@Fac_city
17 |          COMMIT;
18 |  END
```

8 Skrypt tworzący bazę danych

Kod generujący bazę danych z oprogramowaniem:

```
01 |  -- Create database
02 |  :r <path>/Database-Project/CreateDB.sql
03 |  GO
04 |
05 |  -- Create functions
06 |  :r <path>/Database-Project/functions/ActiveSanepidQuarantines.sql
07 |  :r <path>/Database-Project/functions/AvailableBeds.sql
08 |  :r <path>/Database-Project/functions/AvailableHospitals.sql
09 |  :r <path>/Database-Project/functions/CheckTestResult.sql
10 |  :r <path>/Database-Project/functions/GetContactNumber.sql
11 |  GO
12 |
13 |  -- Create stored procedures
14 |  :r <path>/Database-Project/stored procedures/AddClinic.sql
15 |  :r <path>/Database-Project/stored procedures/AddPatient.sql
16 |  :r <path>/Database-Project/stored procedures/AddPatientCard.sql
17 |  :r <path>/Database-Project/stored procedures/AddPerson.sql
```

```
:r <path>/Database-Project/stored procedures/AddSanepid.sql
:r <path>/Database-Project/stored procedures/AddTestingFacility.sql
:r <path>/Database-Project/stored procedures/AddVaccinationFacility.sql
:r <path>/Database-Project/stored procedures/BackupDatabase.sql
:r <path>/Database-Project/stored procedures/DischargePatient.sql
:r <path>/Database-Project/stored procedures/MakeEligibleForVaccination.sql
:r <path>/Database-Project/stored procedures/Quarantine.sql
GO

-- Create triggers
:r <path>/Database-Project/triggers/AutoCreateQuarantineDetails.sql
:r <path>/Database-Project/triggers/AutoGeneratePatientsCard.sql
:r <path>/Database-Project/triggers/AutoQuarantine.sql
:r <path>/Database-Project/triggers/MakeBedAvailable.sql
:r <path>/Database-Project/triggers/Occupybed.sql
GO

-- Create views
:r <path>/Database-Project/views/Eligibility.sql
:r <path>/Database-Project/views/HospitalsEMSConnections.sql
:r <path>/Database-Project/views/HospitalsInRegion.sql
:r <path>/Database-Project/views/InfectionsPerRegion.sql
:r <path>/Database-Project/views/PatientsInHospital.sql
GO
```

Kod generujący tabele:

```
IF DB_ID('TestDB') IS NOT NULL
  DROP DATABASE TestDB;
GO

CREATE DATABASE TestDB;
GO

USE TestDB;
GO

CREATE TABLE Hospitals
(
  Facility_Name VARCHAR(50) NOT NULL,
  City VARCHAR(30) NOT NULL,
  Region VARCHAR(30) NOT NULL,
  Address VARCHAR(50) NOT NULL,
  Contact_Number VARCHAR(20) NOT NULL,
  PRIMARY KEY (Facility_Name, City)
);
GO

CREATE TABLE Hospital_Details
```

```
23 |  (
24 |    Available_Respirators INT NOT NULL,
25 |    Available_Beds INT NOT NULL,
26 |    Infections_Diseases_Hospital BIT NOT NULL,
27 |    Facility_Name VARCHAR(50) NOT NULL,
28 |    City VARCHAR(30) NOT NULL,
29 |    PRIMARY KEY (Facility_Name, City),
30 |    FOREIGN KEY (Facility_Name, City) REFERENCES Hospitals(Facility_Name, City) ON DELETE CASCADE ON UPDATE CASCADE
31 |  );
32 |  GO
33 |
34 |  CREATE TABLE Sanitary_Epidemiological_Stations
35 |  (
36 |    Address VARCHAR(50) NOT NULL,
37 |    Subservient_Region VARCHAR(30) NOT NULL,
38 |    Contact_Number VARCHAR(20) NOT NULL,
39 |    City VARCHAR(30) NOT NULL,
40 |    Postal_Code VARCHAR(6) NOT NULL,
41 |    PRIMARY KEY (Address, City)
42 |  );
43 |  GO
44 |
45 |  CREATE TABLE Emergency_Medical_Services
46 |  (
47 |    Address VARCHAR(50) NOT NULL,
48 |    City VARCHAR(30) NOT NULL,
49 |    PRIMARY KEY (Address, City)
50 |  );
51 |  GO
52 |
53 |  CREATE TABLE EMS_Hospitals
54 |  (
55 |    Hospital_Facility_Name VARCHAR(50) NOT NULL,
56 |    Hospital_City VARCHAR(30) NOT NULL,
57 |    EMS_Address VARCHAR(50) NOT NULL,
58 |    EMS_City VARCHAR(30) NOT NULL,
59 |    PRIMARY KEY (Hospital_Facility_Name, Hospital_City, EMS_Address, EMS_City),
60 |    FOREIGN KEY (Hospital_Facility_Name, Hospital_City) REFERENCES Hospitals(Facility_Name, City) ON DELETE CASCADE ON UPDATE CASCADE,
61 |    FOREIGN KEY (EMS_Address, EMS_City) REFERENCES Emergency_Medical_Services(Address, City) ON DELETE CASCADE ON UPDATE CASCADE
62 |  );
63 |  GO
64 |
65 |  CREATE TABLE EMS_Details
66 |  (
67 |    Available_Ambulances INT NOT NULL,
68 |    Available_Helicopters INT NOT NULL,
69 |    Address VARCHAR(50) NOT NULL,
70 |    City VARCHAR(30) NOT NULL,
71 |    PRIMARY KEY (Address, City),
```

```
72 |     FOREIGN KEY (Address, City) REFERENCES Emergency_Medical_Services(Address, City) ON DELETE CASCADE ON UPDATE CASCADE
73 |   );
74 |   GO
75 |
76 |   CREATE TABLE Testing_Facilities
77 |   (
78 |     Address VARCHAR(50) NOT NULL,
79 |     City VARCHAR(30) NOT NULL,
80 |     Postal_Code VARCHAR(6) NOT NULL,
81 |     Contact_Number INT NOT NULL,
82 |     PRIMARY KEY (Address, City)
83 |   );
84 |   GO
85 |
86 |   CREATE TABLE Clinics
87 |   (
88 |     Address VARCHAR(50) NOT NULL,
89 |     City VARCHAR(30) NOT NULL,
90 |     Name VARCHAR(50) NOT NULL,
91 |     Postal_Code VARCHAR(6) NOT NULL,
92 |     Contact_Number VARCHAR(20) NOT NULL,
93 |     PRIMARY KEY (Address, City)
94 |   );
95 |   GO
96 |
97 |   CREATE TABLE Vaccination_Facilities
98 |   (
99 |     Available_Vaccines INT NOT NULL,
100 |     Address VARCHAR(50) NOT NULL,
101 |     City VARCHAR(30) NOT NULL,
102 |     Name VARCHAR(50) NOT NULL,
103 |     PRIMARY KEY (Address, City)
104 |   );
105 |   GO
106 |
107 |   CREATE TABLE People
108 |   (
109 |     Address VARCHAR(50) NOT NULL,
110 |     PESEL VARCHAR(11) NOT NULL,
111 |     Region VARCHAR(30) NOT NULL,
112 |     Name VARCHAR(30) NOT NULL,
113 |     Surname VARCHAR(30) NOT NULL,
114 |     Date_of_Birth DATE NOT NULL,
115 |     Telephone_number VARCHAR(20) NOT NULL,
116 |     Postal_Code VARCHAR(6) NOT NULL,
117 |     Clinic_Address VARCHAR(50) NOT NULL,
118 |     Clinic_City VARCHAR(30) NOT NULL,
119 |     PRIMARY KEY (PESEL),
120 |     FOREIGN KEY (Clinic_Address, Clinic_City) REFERENCES Clinics(Address, City) ON DELETE CASCADE ON UPDATE CASCADE
```

```
121 |  );
122 |  GO
123 |
124 |  CREATE TABLE Quarantined_People
125 |  (
126 |    PESEL VARCHAR(11) NOT NULL,
127 |    Sanepid_Station_Address VARCHAR(50) NOT NULL,
128 |    Sanepid_Station_City VARCHAR(30) NOT NULL,
129 |    PRIMARY KEY (PESEL),
130 |    FOREIGN KEY (PESEL) REFERENCES People(PESEL) ON DELETE CASCADE ON UPDATE CASCADE,
131 |    FOREIGN KEY (Sanepid_Station_Address, Sanepid_Station_City) REFERENCES Sanitary_Epidemiological_Stations(Address, City) ON DELETE CASCADE ON
        UPDATE CASCADE
132 |  );
133 |  GO
134 |
135 |  CREATE TABLE Hospitalized_Patients
136 |  (
137 |    PESEL VARCHAR(11) NOT NULL,
138 |    Facility_Name VARCHAR(50) NOT NULL,
139 |    City VARCHAR(30) NOT NULL,
140 |    PRIMARY KEY (PESEL),
141 |    FOREIGN KEY (PESEL) REFERENCES People(PESEL) ON DELETE CASCADE ON UPDATE CASCADE,
142 |    FOREIGN KEY (Facility_Name, City) REFERENCES Hospitals(Facility_Name, City) ON DELETE CASCADE ON UPDATE CASCADE
143 |  );
144 |  GO
145 |
146 |  CREATE TABLE Tests
147 |  (
148 |    Status VARCHAR(30) NOT NULL,
149 |    Result VARCHAR(30) NOT NULL,
150 |    Test_Date DATE NOT NULL,
151 |    Refferal_ID INT NOT NULL,
152 |    Testing_Facility_Address VARCHAR(50),
153 |    Testing_Facility_City VARCHAR(30),
154 |    Address VARCHAR(50),
155 |    City VARCHAR(30),
156 |    PESEL VARCHAR(11) NOT NULL,
157 |    PRIMARY KEY (Refferal_ID),
158 |    FOREIGN KEY (Testing_Facility_Address, Testing_Facility_City) REFERENCES Testing_Facilities(Address, City) ON DELETE SET NULL ON UPDATE CASCADE,
159 |    FOREIGN KEY (Address, City) REFERENCES Clinics(Address, City) ON DELETE SET NULL ON UPDATE CASCADE,
160 |    FOREIGN KEY (PESEL) REFERENCES People(PESEL) ON DELETE CASCADE ON UPDATE CASCADE
161 |  );
162 |  GO
163 |
164 |  CREATE TABLE Patient_Card
165 |  (
166 |    Chronic_Diseases VARCHAR(100) NOT NULL,
167 |    Blood_Type VARCHAR(10) NOT NULL,
168 |    Condition VARCHAR(30) NOT NULL,
```

```
169 |     PESEL VARCHAR(11) NOT NULL,
170 |     PRIMARY KEY (PESEL),
171 |     FOREIGN KEY (PESEL) REFERENCES Hospitalized_Patients(PESEL) ON DELETE CASCADE ON UPDATE CASCADE
172 |   );
173 |   GO
174 |
175 |   CREATE TABLE Quarantine_Details
176 |   (
177 |     Start_Date DATE NOT NULL,
178 |     End_Date DATE NOT NULL,
179 |     Address VARCHAR(50) NOT NULL,
180 |     City VARCHAR(30) NOT NULL,
181 |     PESEL VARCHAR(11) NOT NULL,
182 |     PRIMARY KEY (Start_Date, PESEL),
183 |     FOREIGN KEY (PESEL) REFERENCES Quarantined_People(PESEL) ON DELETE CASCADE ON UPDATE CASCADE
184 |   );
185 |   GO
186 |
187 |   CREATE TABLE Vaccination_Details
188 |   (
189 |     Series INT,
190 |     Visit_Date DATE,
191 |     Eligibility BIT NOT NULL,
192 |     PESEL VARCHAR(11) NOT NULL,
193 |     Vaccination_Facility_Address VARCHAR(50),
194 |     Vaccination_Facility_City VARCHAR(30),
195 |     PRIMARY KEY (PESEL),
196 |     FOREIGN KEY (PESEL) REFERENCES People(PESEL) ON DELETE CASCADE ON UPDATE CASCADE,
197 |     FOREIGN KEY (Vaccination_Facility_Address, Vaccination_Facility_City) REFERENCES Vaccination_Facilities(Address, City) ON DELETE SET NULL ON
        UPDATE CASCADE
198 |   );
199 |   GO
```

9 Typowe zapytania

```
INSERT INTO Clinics VALUES ('testowa 12','Bia ystok','Klinika1','42-450','512342121')
DELETE FROM People WHERE People.Address='Bia ystok'
INSERT INTO People VALUES ('Bia ystok','00303001594','Podlesie','Maciej','Trapez','1997-03-12','134125123','41-340','testowa 12','Bia ystok'),
 ('Bia ystok','12321094512','Podlesie','Marek','Towarek','1997-10-30','75425123','41-340','testowa 12','Bia ystok'),
 ('Bia ystok','01235432341','Podlesie','Micha ',' bik ','1967-12-12','210682123','41-340','testowa 12','Bia ystok'),
 ('Bia ystok','32985314123','Podlesie','Mi osz','Taczka','1915-06-30','59315632','41-340','testowa 12','Bia ystok')
INSERT INTO Hospitals VALUES ('Szpital Zakazny','Bia ystok','Losowa 15','123123123')
INSERT INTO Hospital_Details VALUES(50,50,'TRUE','Szpital Zakazny','Bia ystok')
INSERT INTO Hospitals VALUES ('Szpital Dzieciecy','Krak w','Janna 15','1412341233')
INSERT INTO Hospital_Details VALUES(40,10,'FALSE','Szpital Dzieciecy','Krak w')
INSERT INTO Testing_Facilities VALUES ('  witej   Anny 12','Bia ystok','43-312',12312312)
INSERT INTO Sanitary_Epidemiological_Stations VALUES ('Lokalna 5','Podlesie',1231421,'Bia ystok','41-350')
DELETE FROM Tests WHERE Tests.Refferal_ID = 12
INSERT INTO Tests VALUES ('Done','Positive','2020-12-14',12,'  witej   Anny 12','Bia ystok','testowa 12','Bia ystok','00303001594')
--Funkcja zwraca szpitale do ktorych moze zostac przyjety nasz pacjent o danym kodzie PESEL
SELECT * FROM dbo.AVAILABLE_HOSPITALS(12321094512)
-- Widok zwraca tabel  z ilo ci  zaka onych(pozytywnych test w) w ka dym wojew dztwie
SELECT * FROM INFECTIONS_PER_REGION
-- Widok zwraca tabel  z wszystkimi szpitalami oraz ich aktualnym stanem respirator w i wolnych   ek
SELECT * FROM HospitalsInRegion
-- Widok zwraca tabel  z poszczeg lnymi jednostkami pogotowia przypisanymi do odpowiednich szpitali
SELECT * FROM Hospital_EMS_Connections
-- Widok zwraca tabel  z informacjami o zakwalifikowaniu si  ka dego zarejerstrowanego pacjenta do szczepienia
SELECT * FROM ELIGIBILITY
-- Procedura przyjmuje pacjenta do wolnych szpitali w jego rejonie przy u yciu jego kodu PESEL
 EXEC AddPatient(12321094512)


 SELECT * FROM Hospital_Details
 SELECT * FROM Hospitals
 SELECT * FROM People
 SELECT * FROM Testing_Facilities
 SELECT * FROM Tests
 SELECT * FROM Quarantined_People
 SELECT * FROM Quarantine_Details
```