

# Katedra Biosensorów i Przetwarzania Sygnałów Biomedycznych

Wydział Inżynierii Biomedycznej

POLITECHNIKA ŚLĄSKA



Cyberbezpieczeństwo w medycznych bazach  
i systemach danych

## PROJEKT

Dominika Gabor

Agnieszka Radziun

Kamil Suchanek

**Kierunek studiów:** *Inżynieria Biomedyczna*

**Specjalność:** *Przetwarzanie i Analiza Informacji Biomedycznej*

Prowadzący projekt: Dr inż. Rafał Doniec

Zabrze 2020

# Spis treści:

1.	Medyczne bazy danych.....	2
2.	Zabezpieczenia baz danych.....	4
2.1.	Ograniczanie dostępu.....	4
2.2.	Rozwiązania kryptografii.....	4
2.3.	Zabezpieczenia z poziomu wiersza.....	5
3.	Projekt bazy danych.....	6
3.1.	Schemat projektu.....	6
4.	Implementacja projektu.....	7
4.1.	Wdrożenie rozwiązania.....	9
5.	Zabezpieczenia bazy danych.....	10
5.1.	Szyfrowanie danych.....	10
5.2.	Szyfrogramy numeru pesel.....	11
5.3.	Szyfrowanie dwukierunkowe.....	13
5.4.	Szyfrowanie symetryczne.....	18
6.	Podsumowanie.....	20

# 1. Medyczne bazy danych

W marcu 2009 roku Ministerstwo Zdrowia zaprezentowało "Strategię E-Zdrowie Polska na lata 2009-2015", która mówi o wykorzystaniu wielu możliwości mediów cyfrowych umożliwiających wprowadzanie, przechowywanie oraz zarządzanie danymi medycznymi.<sup>[2,3]</sup> Głównym elementem elektronicznego zapisu jest baza danych, która zapewnia:

- ❖ usprawnienie procesu wymiany informacji w ramach jednostki oraz zespołu jednostek;
- ❖ wzrost bezpieczeństwa dokumentacji medycznej;
- ❖ polepszenie identyfikacji i czytelności osób, które dokonały w niej wpisu;
- ❖ podniesienie mobilności danych (również na terenie Unii Europejskiej);
- ❖ zwiększenie ilości informacji pozwalających na postawienie diagnozy;
- ❖ natychmiastowy wgląd do historii pacjenta: poddane procedury, leki, historia choroby;
- ❖ obniżenie kosztów archiwizacji danych.<sup>[3]</sup>

Bazy danych to środowisko oprogramowania, fizyczne urządzenia oraz bezpieczeństwo i administracja umożliwiająca: wprowadzanie, przechowywanie, modyfikowanie, usuwanie oraz pobieranie w sposób uporządkowany dużych ilości informacji. Bazy danych są stworzone z tabel, które zawierają kolumny (pola) i wiersze (rekordy). Kolumny zawierają pewien szczególny rodzaj informacji (określony poprzez właściwy typ danych np. int dla liczb całkowitych), który opisany jest poprzez wiersze.<sup>[2]</sup>

Występują trzy modele bazy danych: jednorodny, obiektowy i relacyjny. W przypadku modelu jednorodnego wszystkie dane są zawarte w jednej tabeli i najczęściej wykorzystywany jest w arkuszach kalkulacyjnych np. Excel. Natomiast w modelu obiektowym informacje są przechowywane w postaci całych obiektów, z którymi związany jest ich stan i zachowanie. Model ten jest dobrze dopasowany do obiektowych języków programowania takich jak C# czy Java. Najczęściej jednak jest spotykany

relacyjny model bazy danych.<sup>[1]</sup> Umożliwia on stosowanie powiązań (relacji) pomiędzy różnymi tabelami, wśród których wyróżniamy relacje:

- ❖ jeden do jednego - typ relacji, w której występuje tylko jedna tabela, gdzie jednej wartości w kolumnie, odpowiada druga wartość w drugiej kolumnie;
- ❖ jeden do wielu - typ relacji, w którym jednej wartości w jednej kolumnie odpowiada kilka wartości w drugiej kolumnie;
- ❖ wiele do jednego - jest to odwrotność relacji jeden do wielu;
- ❖ wiele do wielu - wielu wartościom w jednej kolumnie, odpowiada wiele wartości w drugiej kolumnie.<sup>[2]</sup>

Aby relacje między tabelami były jednoznaczne, połączone tabele muszą posiadać klucz podstawowy, który jest identyfikatorem poszczególnych wierszy. W przypadku gdy w tabeli występują wartości z drugiej tabeli (identyfikatory rekordu) wtedy taką kolumnę nazywamy kluczem obcym.<sup>[1]</sup>

Najbardziej znanymi serwerami relacyjnych baz danych są: *MariaDB*, *MySQL*, *SQL Server*, *Oracle DataBase*. Każdy z nich ma swoje unikatowe cechy lecz łączy je fakt, że wszystkie wykorzystują język SQL (Structured Query Language) czyli strukturalny język zapytań. Jego działanie polega na wysyłaniu zapytań (poleceń, kwerend) do bazy i dostarczaniu użytkownikowi wymaganych danych jak i tworzeniu oraz przekształcaniu elementów bazodanowych takich jak tabele czy procedury.<sup>[1]</sup>

Procedury są wykorzystywane aby uniknąć pisania niezmiennie tego samego kodu poprzez zapis pewnego zapytania, które zostanie wykorzystane wielokrotnie. Możliwe jest również przekazywanie parametrów do procedury, dzięki czemu jej wynik będzie zależny od danych wejściowych, a kod stanie się bardziej uniwersalny.<sup>[2]</sup>

## 2. Zabezpieczenia baz danych

Bazy i systemy danych ze względu na prywatność i bezpieczeństwo informacji wymagają pewnych zabezpieczeń. Formy i realizacja owych zależą od stopnia i rodzaju szacowanego zagrożenia. Dane nie mające wtórnej wartości oraz te publiczne będą wymagały wiarygodnego i stabilnego systemu bez nacisku na zabezpieczenia, natomiast systemy operujące na danych wrażliwych mogą być celem kradzieży i manipulacji.

### 2.1. Ograniczanie dostępu

Silnik bazodanowy zazwyczaj spełnia funkcje typowe dla serwera i nie tylko, należy go odpowiednio skonfigurować aby nie zezwalał na każdy rodzaj połączeń. W przypadku bazy danych PostgreSQL odpowiada za to plik konfiguracyjny *pg\_hba.conf*.

HBA oznacza uwierzytelnienie oparte na hoście. Zawiera on rekordy ułożone kolejno wiersz po wierszu. Każdy rekord określa typ połączenia, zakres adresów IP klienta, nazwę bazy danych, nazwę użytkownika i metodę uwierzytelnienia, która ma być używana dla połączeń spełniających te parametry [4].

Dostęp do serwera baz danych ograniczany jest również dla przypadkowych i nieproszonych nasłuchujących oraz osób, które zdołały uzyskać dostęp bezpośredni mimo postawionych blokad poprzez szyfrowanie treści osadzonej w bazie danych oraz wymienianej z klientem w danej chwili.

### 2.2. Rozwiązania kryptografii

Niektóre systemy bazodanowe wspierają natywnie obsługę połączeń SSL do szyfrowania komunikacji klient-serwer. SSL jest standardową technologią zabezpieczeń służącą do ustanawiania szyfrowanego łącza między serwerem internetowym a przeglądarką. To łącze zapewnia, że wszystkie dane przekazywane między serwerem internetowym a przeglądarkami pozostają prywatne i zintegrowane.

Inną opcją jest szyfrowanie danych w spoczynku. Operacja ta polega na szyfrowaniu danych w całej bazie danych, w poszczególnych tabelach albo kolumnach. Wiele systemów bazodanowych posiada pewne rozszerzenia i wbudowane funkcje kryptografii.

Ze względu na sposób realizacji i zastosowanie szyfrowanie takie można podzielić na:

❖ jednokierunkowe:

W tym przypadku nie zamierzamy odszyfrowywać danych do wglądu. Jedynym celem jest upewnienie się, że osoba, która chce uzyskać dostęp do pewnych obszarów, zna treść zaszyfrowaną, np.: hasło.

❖ dwukierunkowe:

Symetryczne i asymetryczne, pierwsze polega na szyfrowaniu i odszyfrowywaniu danych przy pomocy jednego klucza, natomiast drugie polega na szyfrowaniu z pomocą klucza nazywanego publicznym oraz odszyfrowywaniu z pomocą klucza zwanego prywatnym.

## 2.3. Zabezpieczenia z poziomu wiersza

Oprócz bardziej oczywistych metod ochrony danych, systemy bazodanowe oferują również rozwiązania na poziomie mechanizmów działania silnika. Zabezpieczenia tego rodzaju komplikują zarządzanie bazą danych.

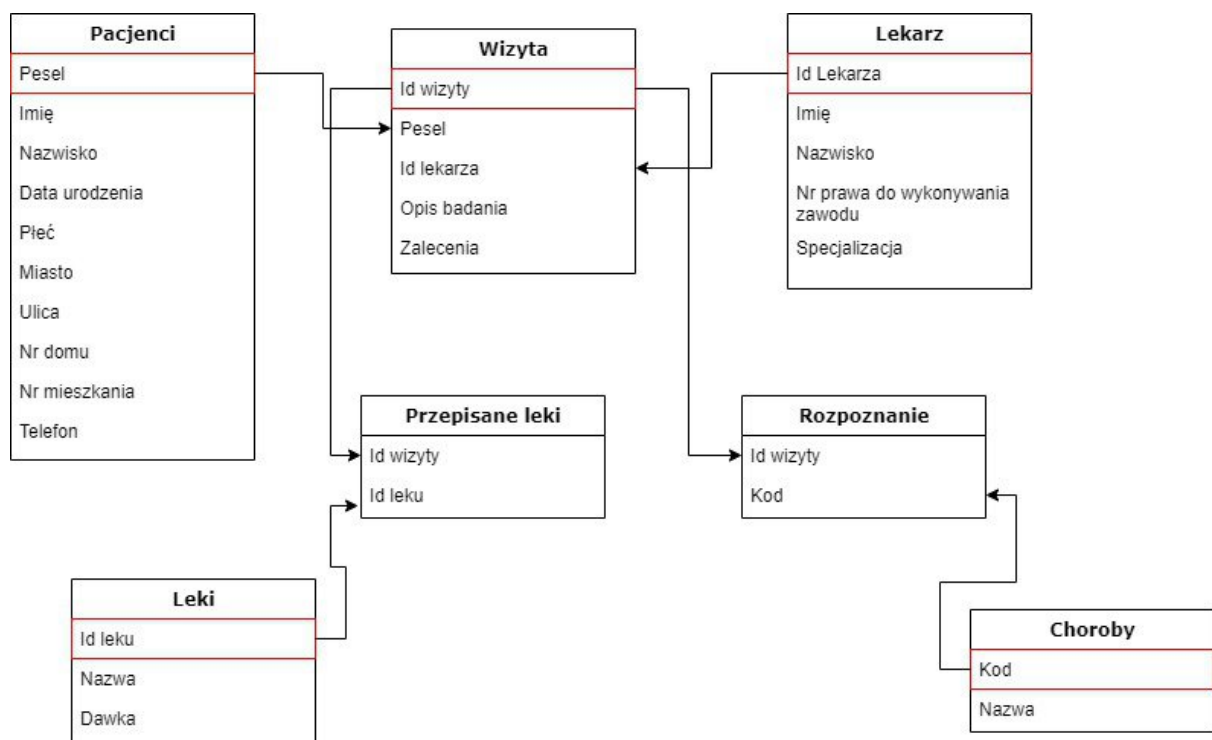
Ogólnie polega to na przyznawaniu uprawnień (GRANT) dla poszczególnych kolumn i tabel do wyświetlania rekordów, ich modyfikacji. Przykładowo standardowy klient poprzez aplikację zewnętrzną może być przyporządkowany do pewnego spektrum ról w bazie danych, mających bardzo ograniczone możliwości, natomiast inny rodzaj użytkownika z poziomu bazy danych może mieć większe uprawnienia, jednak nie stosuje komunikacji przez wspomnianą aplikację, dzięki czemu można ograniczyć możliwy, szkodliwy wpływ nieprzyjaznych zapytań do bazy danych z poziomu aplikacji klienckiej.

### 3. Projekt bazy danych

Zaprojektowano ideową bazę danych dla przychodni, a konkretnie części zarządzającej danymi pacjentów, związanych z wizytą. Wykorzystano tu relacyjny model bazy danych, który umożliwia przedstawienie powiązań (relacji) pomiędzy różnymi tabelami,. Relacje występujące w projekcie to jeden do wielu oraz wiele do wielu.

#### 3.1. Schemat projektu

Projekt bazy składa się z siedmiu tablic: *Pacjenci*, *Wizyta*, *Lekarz*, *Leki*, *Choroby*, *Przepisane leki* i *Rozpoznanie*. Na schemacie zaznaczono strzałkami relacje pomiędzy poszczególnymi tabelami oraz przy pomocy czerwonych ramek oznaczono klucze główne tablic.

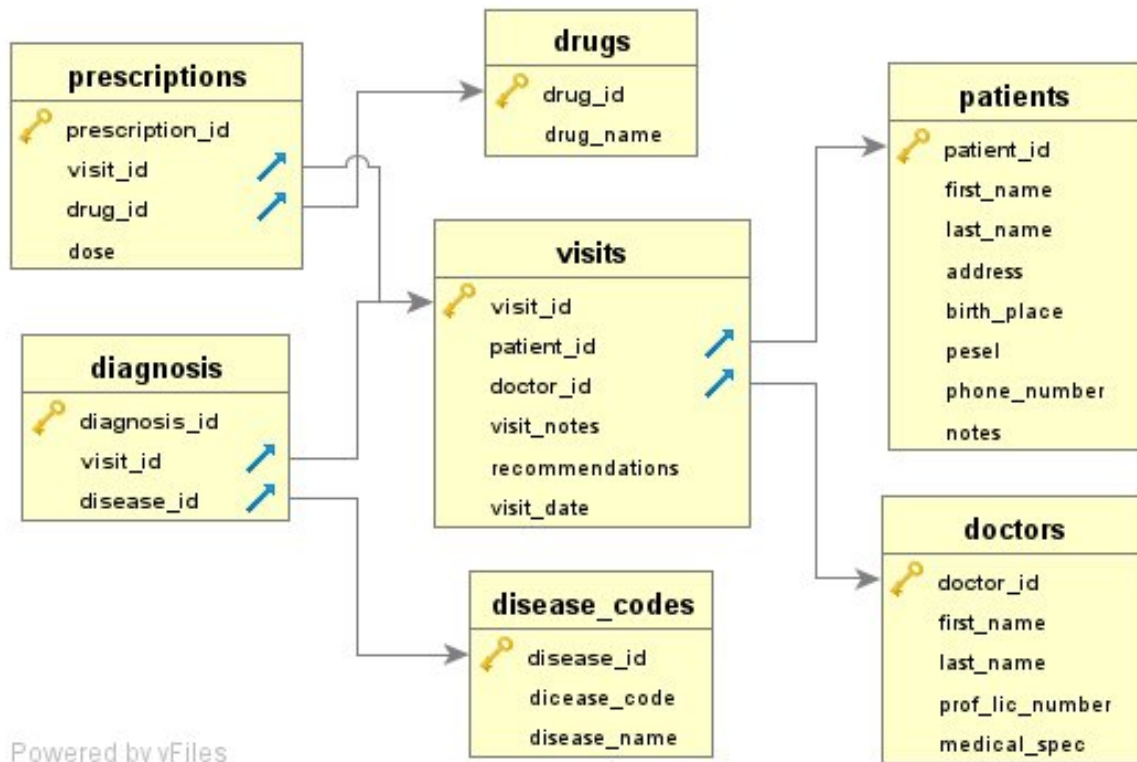


Rys. 1 - Schemat projektu bazy danych

## 4. Implementacja projektu

Baza danych została zaimplementowana przy pomocy silnika PostgreSQL. Postgres jest potężnym systemem obiektowo-relacyjnych baz danych typu open source, aktywnie rozwijanym od ponad 30 lat [5].

Baza przedstawia się następująco:



Rys. 2 - Schemat zaimplementowanej bazy danych.  
[Obraz uzyskany przy pomocy narzędzia DbVisualizer (Free)]

Względem projektu bazy pewne kolumny zostały zmienione/zredukowane w celu optymalizacji wynikowej bazy danych.

Zaimplementowana baza danych jest przykładowym zbiorem relacji danych w przychodni. Uwzględnieni zostali pacjenci oraz lekarze, jako uproszczenie grupy pracowników danej placówki. Pacjent może mieć wiele wizyt a owa wiele towarzyszących jej aspektów jak np.: recepta i diagnoza.



❖ Kod SQL:

```
CREATE DATABASE cwmbd_project  
WITH OWNER = postgres  
ENCODING = 'UTF8'  
LC_COLLATE = 'Polish_Poland.1250'  
LC_CTYPE = 'Polish_Poland.1250'  
TABLESPACE = pg_default  
CONNECTION LIMIT = -1;
```

```
CREATE TABLE patients(  
    patient_id SERIAL PRIMARY KEY,  
    first_name VARCHAR(20),  
    last_name VARCHAR(20),  
    address VARCHAR(100),  
    birth_place VARCHAR(20),  
    pesel VARCHAR(11) UNIQUE,  
    phone_number VARCHAR(20),  
    notes VARCHAR(100));
```

```
CREATE TABLE drugs(  
    drug_id SERIAL PRIMARY KEY,  
    drug_name VARCHAR(20));
```

```
CREATE TABLE disease_codes(  
    disease_id SERIAL PRIMARY KEY,  
    disease_code VARCHAR(5) UNIQUE,  
    disease_name VARCHAR(100));
```

```
CREATE TABLE doctors(  
    doctor_id SERIAL PRIMARY KEY,  
    first_name VARCHAR(20),  
    last_name VARCHAR(20),  
    prof_lic_number VARCHAR(7) UNIQUE,  
    medical_spec VARCHAR(20));
```

```
CREATE TABLE visits(  
    visit_id SERIAL PRIMARY KEY,  
    patient_id SERIAL REFERENCES patients,  
    doctor_id SERIAL REFERENCES doctors,  
    visit_notes VARCHAR(200),  
    recommendations VARCHAR(200),  
    visit_date DATE);
```

```
CREATE TABLE prescriptions(  
    prescription_id SERIAL PRIMARY KEY,  
    visit_id SERIAL REFERENCES visits,  
    drug_id SERIAL REFERENCES drugs,  
    dose VARCHAR(20));
```

```
CREATE TABLE diagnosis(  
    diagnosis_id SERIAL PRIMARY KEY,  
    visit_id SERIAL REFERENCES visits,  
    disease_id SERIAL REFERENCES  
        disease_codes);
```

## 4.1. Wdrożenie rozwiązania

Baza została wdrożona na bezpłatny serwer przy pomocy serwisu Heroku. Przed utworzeniem obiektu bazy w serwisie Heroku należało stworzyć umownie obiekt aplikacji. Następnie zainstalować dodatek zapewniający silnik PostgreSQL dla owej aplikacji. Panel właściciela aplikacji Heroku umożliwia podgląd bazy dla zapytań tylko do odczytu oraz poznanie parametrów dostępu do bazy danych. Parametry te ulegają częstym zmianom. Oprócz standardowego połączenia z bazą danych przy pomocy adresu hosta, nazwy bazy danych, użytkownika i hasła, Heroku umożliwia dostęp do aplikacji/bazy danych za pośrednictwem narzędzia linii poleceń Heroku CLI.

Bazy danych hostowane za pośrednictwem serwisu Heroku przechowywane są w formie zaszyfrowanego woluminu, w ten sam sposób, przez cały okres jego życia przez usługę Amazon. Baza danych nie jest szyfrowana z poziomu silnika bazy danych. Taką funkcję należy zaimplementować samemu poprzez np.: skorzystanie z gotowych bibliotek.

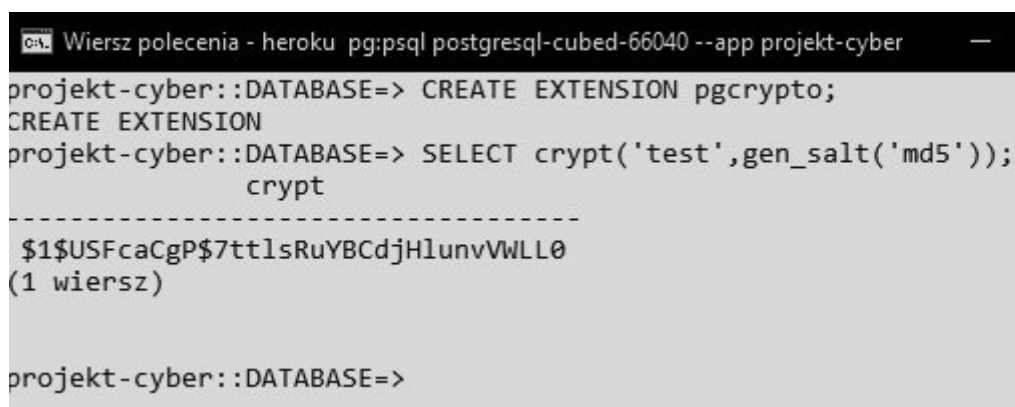
Szyfrowanie z poziomu SZBD wymaga zaufania administratorowi bazy danych ze względu na to, że zakodowane wyjścia i klucze przechowywane są na tym samym serwerze. Inną opcją zabezpieczenia danych jest szyfrowanie z poziomu aplikacji klienckiej, w ten sposób zabezpiecza się hasła do kont poszczególnych użytkowników danego serwisu. W przypadku zapomnienia hasła, nie można uzyskać przypomnienia, hasło można jedynie zresetować.

## 5. Zabezpieczenia bazy danych

Baza danych PostgreSQL hostowana za pośrednictwem serwisu Heroku posiada kilka gotowych zabezpieczeń zarówno typowych dla tego silnika baz danych jak i platformy wdrażającej,

### 5.1. Szyfrowanie danych

Jak już wspomniano, cała baza danych wdrożona na serwerze amazona jest zaszyfrowana. Dodatkowym zabezpieczeniem może być szyfrowanie z poziomu SZBD przy pomocy modułu pgcrypto, zapewniającego funkcje kryptografii [6].



```
Wiersz polecenia - heroku pg:psql postgresql-cubed-66040 --app projekt-cyber
projekt-cyber::DATABASE=> CREATE EXTENSION pgcrypto;
CREATE EXTENSION
projekt-cyber::DATABASE=> SELECT crypt('test',gen_salt('md5'));
      crypt
-----
$1$USFcaCgP$7ttlsRuYBCdjHlunvVWLL0
(1 wiersz)

projekt-cyber::DATABASE=>
```

Rys. 3 - Terminal po zalogowaniu się za pośrednictwem narzędzia linii poleceń Heroku CLI z komenda `psql` - utworzenie rozszerzenia silnika oraz sprawdzenie działania.

Dane, które możemy chcieć zabezpieczyć podlegają pewnym praktykom, adekwatnym do istniejącego/zakładanego zagrożenia. W przypadku danych medycznych, należy obawiać się o dane osobowe i powiązane, uznane za wrażliwe w myśl istniejących przepisów - np.: wszelkie dane z wizyt lekarskich, przebieg chorób.

W zależności od przyjętej strategii można wybrać różne podejścia do szyfrowania danych, w tym:

- ❖ Zabezpieczenie bazy danych szyfrując cały obszar pamięci jaki zajmuje z poziomu systemu operacyjnego - poszczególne dane z tabel są na bieżąco odszyfrowywane podczas stosowania zapytań do bazy,

- ❖ Szyfrowanie poszczególnych tabel i kolumn - ukrywanie treści przez administratora lub osobę uprawnioną, która posiada odpowiednie klucze, potrzebne do odczytania prawdziwej treści.
- ❖ Szyfrowanie haseł, wartości które nie podlegają odczytowi, a weryfikacji, tylko osoba, która tą treść zna i dostarczy ją do systemu, uzyska dostęp. Treść ta znajduje się w postaci zaszyfrowanej w bazie danych, nigdy nie inaczej. Po podaniu w aplikacji klienckiej/panelu administratora, jest szyfrowana i system sprawdza, czy w bazie znajduje się odpowiadający temu rekord.

Szyfrowanie danych całego woluminu pamięci zostało już zrealizowane przez usługę serwerowni firmy Amazon. Firma ta posiada zabezpieczenia trudne do zweryfikowania dla klienta, więc w ramach projektu wykonane zostaną dwie pozostałe metody.

## 5.2. Szyfrogramy numeru pesel

W tym podejściu, numery pesel zostaną przeniesione do oddzielnej tabeli. W przychodni będzie potrzebny numer pesel aby zidentyfikować pacjenta, numer ID zaszyfrowanej wersji peselu będzie prowadził jako klucz obcy do rekordów tego pacjenta w innych tabelach.

- ❖ Usunięcie peseli:

```
projekt-cyber::DATABASE=> select*from patients;
```

patient_id	first_name	last_name	address	birth_place	phone_number	notes	pesel
1	Kamil	Suchenek	B awatkowa 9/13	Gliwice	467598762	---	467598762
2	Dominika	Gab`r	Nak o il skie Morcinka 9	Tarnowskie G`ry	765398267	al.tetracykliny	765398267
3	Agnieszka	Rudziun	Tarnogorska 8 Stare Tarnowice	Otwi`cim	654329873	-	654329873
4	Natalia	Matuszek	S oneczna 5 Krupski M yn	Lubliniec	876539846	agorafobia	876539846
5	Marcin	Kasprzyk	Celiny 8 T`pkowice	Ruda il ska	987654323	po odwyku	987654323

(5 wierszy)

```
projekt-cyber::DATABASE=> ALTER TABLE patients
projekt-cyber::DATABASE-> DROP COLUMN pesel;
ALTER TABLE
projekt-cyber::DATABASE=> select*from patients;
```

patient_id	first_name	last_name	address	birth_place	phone_number	notes
1	Kamil	Suchenek	B awatkowa 9/13	Gliwice	467598762	---
2	Dominika	Gab`r	Nak o il skie Morcinka 9	Tarnowskie G`ry	765398267	al.tetracykliny
3	Agnieszka	Rudziun	Tarnogorska 8 Stare Tarnowice	Otwi`cim	654329873	-
4	Natalia	Matuszek	S oneczna 5 Krupski M yn	Lubliniec	876539846	agorafobia
5	Marcin	Kasprzyk	Celiny 8 T`pkowice	Ruda il ska	987654323	po odwyku

(5 wierszy)

Rys. 4 - Fragment terminala - usunięcie kolumny z numerami pesel.

- ❖ Utworzenie tabeli peseli i umieszczenie w niej nowych zaszyfrowanych numerów:

```
projekt-cyber::DATABASE=> CREATE TABLE pesels(
projekt-cyber::DATABASE(> pesel_id SERIAL PRIMARY KEY,
projekt-cyber::DATABASE(> pesel VARCHAR UNIQUE);
CREATE TABLE
projekt-cyber::DATABASE=> INSERT INTO pesels (pesel) VALUES
projekt-cyber::DATABASE-> (crypt('1111111111',GEN_SALT('md5'))),
projekt-cyber::DATABASE-> (crypt('2222222222',GEN_SALT('md5'))),
projekt-cyber::DATABASE-> (crypt('3333333333',GEN_SALT('md5'))),
projekt-cyber::DATABASE-> (crypt('4444444444',GEN_SALT('md5'))),
projekt-cyber::DATABASE-> (crypt('5555555555',GEN_SALT('md5')));
INSERT 0 5
projekt-cyber::DATABASE=> SELECT * FROM pesels;
 pesel_id |
-----+-----
          1 | $1$1gJBcscr$ucCciRe7mnSotjVbB7mCk1
          2 | $1$4hPsCpKf$VdV1B2fytWP2LOpKG1/aC1
          3 | $1$uN01sugz$VZaw7P8slX4ZXGDz6AK8L1
          4 | $1$C5KTRWU7$F0TtMXLOaVfN9uRzEFVrp0
          5 | $1$i9Qqix15$NUPg6hK7BIL.z8w5viDrx0
(5 wierszy)
```

Rys. 5 - Fragment terminala - utworzenie tabeli peseli oraz wstawienie nowych wartości.

Wartości wstawione nie posiadają znanej wady szyfrowania md5, zaszyfrowanie nawet 2 takich samych wartości zwróci 2 inne, dzięki dodaniu "soli" do algorytmu, utrudnia to również odnalezienie oryginalnego tekstu i komplikuje wynik. Aktualnie algorytm md5 uchodzi za niezbyt bezpieczne rozwiązanie, poleca się zastosowanie w zamian algorytmów SHA-N albo dodanie soli.

- ❖ Zapytanie z poziomu klienta:

Po dodaniu kolumny:

```
ALTER TABLE patients ADD COLUMN pesel_id SERIAL UNIQUE REFERENCES pesels;
```

oraz wstawieniu wartości pesel\_id do tabeli patients:

```
UPDATE patients SET pesel_id = 1 WHERE patient_id = 1;
UPDATE patients SET pesel_id = 2 WHERE patient_id = 2;
UPDATE patients SET pesel_id = 3 WHERE patient_id = 3;
UPDATE patients SET pesel_id = 4 WHERE patient_id = 4;
UPDATE patients SET pesel_id = 5 WHERE patient_id = 5;
```

```

projekt-cyber::DATABASE=> SELECT * FROM patients WHERE
projekt-cyber::DATABASE-> pesel_id = (SELECT pesel_id FROM pesels WHERE
projekt-cyber::DATABASE(> pesel = crypt('1111111111', pesel));
patient_id | first_name | last_name | address | birth_place | phone_number | notes | pesel_id
-----+-----+-----+-----+-----+-----+-----+-----
1 | Kamil | Suchenek | B|awatkowa 9/13 | Gliwice | 467598762 | --- | 1
(1 wiersz)

```

Rys. 6 - Fragment terminala - odszukanie rekordu pacjenta.

Tego rodzaju zapytanie odszukuje rekord pacjenta o ile poda się jego pesel.

#### ❖ Porównanie czasu zapytania z szyfrowaniem i bez:

Zestawiono czas realizacji zapytania na serwerze lokalnym, przy pierwszym uruchomieniu zapytania.

- bez szyfrowania: 193ms aż do 112-125ms przy powtarzaniu operacji:

```

SELECT * FROM patients WHERE
pesel_id = (SELECT pesel_id FROM pesels WHERE
              pesel = '1111111111');

```

- z szyfrowaniem: 173ms aż do 130-150ms przy powtarzaniu operacji:

```

SELECT * FROM patients WHERE
pesel_id = (SELECT pesel_id FROM pesels WHERE
              pesel = crypt('1111111111', pesel));

```

Zapytanie bez szyfrowania częściej zwraca szybciej wynik niż zapytanie z szyfrowaniem, różnica jest niewielka, gdyż dotyczy szyfrowania tylko jednej wartości a nie ich zbioru.

### 5.3. Szyfrowanie dwukierunkowe

Metoda ta jest odpowiednia, gdy mamy zamiar zataić zawartość przed niepożądanymi podmiotami, a udostępnić ją zweryfikowanym użytkownikom.

Rozszerzenie pgcrypto oferuje funkcje szyfrowania PGP. Obrona metoda wykorzystuje dwa klucze: publiczny, który służy do zaszyfrowania treści oraz prywatny, dzięki któremu można odczytać treść oryginalną. W celu wygenerowania kluczy wykorzystano narzędzie GnuPG (<https://www.gnupg.org/download/>).

Procedura wygląda następująco:

- ❖ modyfikacja bazy danych - zmiana typów danych na bytea, w celu przechowywania postaci zaszyfrowanej:

```
CREATE TABLE patients(  
    patient_id SERIAL PRIMARY KEY,  
    first_name bytea,  
    last_name bytea,  
    address bytea,  
    birth_place bytea,  
    pesel bytea UNIQUE,  
    phone_number bytea,  
    notes bytea);
```

- ❖ utworzenie pary kluczy przy pomocy oprogramowania Gnupgp,

Klucze utworzono i wyeksportowano przy pomocy poleceń poleceń zaznaczonych na żółto z rysunku nr 7. Polecenie edit-key pozwala na odczytanie identyfikatorów kluczy, które są potrzebne w celu ich eksportu do plików.

#### F.26.3.9. Generating PGP Keys With GnuPG

To generate a new key:

```
gpg --gen-key
```

The preferred key type is "DSA and Elgamal".

For RSA encryption you must create either DSA or RSA sign-only key as master and then add an RSA encryption subkey with `gpg --edit-key`.

To list keys:

```
gpg --list-secret-keys
```

To export a public key in ASCII-armor format:

```
gpg -a --export KEYID > public.key
```

To export a secret key in ASCII-armor format:

```
gpg -a --export-secret-keys KEYID > secret.key
```

You need to use `dearmor()` on these keys before giving them to the PGP functions. Or if you can handle binary data, you can drop `-a` from the command.

*Rys. 7 - Fragment dokumentacji pgcrypto [odnośnik bez numeru 2]*

- ❖ zaimplementowanie szyfrowania w poleceniu INSERT:



```

INSERT INTO patients(first_name,last_name,address,birth_place,pesel,phone_number,notes)
SELECT  pgp_pub_encrypt(pat.fn, dearmor('key')) AS first_name,
        pgp_pub_encrypt(pat.ln, dearmor('key')) AS last_name,
        pgp_pub_encrypt(pat.ad, dearmor('key')) AS address,
        pgp_pub_encrypt(pat.bi, dearmor('key')) AS birth_place,
        pgp_pub_encrypt(pat.pe, dearmor('key')) AS pesel,
        pgp_pub_encrypt(pat.ph, dearmor('key')) AS phone_number,
        pgp_pub_encrypt(pat.no, dearmor('key')) AS notes
FROM (VALUES
('Kamil','Suchenek','Bławatkowa 9/13','Gliwice','43898765672','467598762','---'),
('Dominika','Gabór','Nakło Śląskie Morcinka 9','Tarnowskie Góry','87654678926','765398267','al.tetracykliny'),
('Agnieszka','Rudziun','Tarnogorska 8 Stare Tarnowice','Oświęcim','67489356728','654329873','-'),
('Natalia','Matuszek','Słoneczna 5 Krupski Młyn','Lubliniec','78754378961','876539846','agorafobia'),
('Marcin','Kasprzyk','Celiny 8 Tąpkowice','Ruda Śląska','87546783218','987654323','po odwyku') )
AS pat(fn,ln,ad,bi,pe,ph,no); /*CROSS JOIN

```

Rys. 8 - Instrukcja wsadowa dla szyfrowania danych kluczem publicznym.

Na rysunku nr 8, przedstawiono wstawianie danych do tablicy patients, w miejscach 'key' znajduje się interpretacja klucza w ASCII:

```

-----BEGIN PGP PUBLIC KEY BLOCK-----
mQENBF56bLUBCADNoxSLBT/ou9RMtInCxbjVgmjvclDVf8NG8Suew/XZi2Bs5PjO
d7aR/kvk/DkBRD96/Q6yW0vt9OYf32tQOZUgylOqnpBC1o0TmWPYTFcb50KJNDwa
D85d70PG9EHvIAsGpAKDWIzhGiQd01+TU2tb2OjBfxf78w2tK8G4704IqRGHMkhl
ijGxoCapaVFQyuT6RR+/iDOmZ0K7rsFNgVm4q/iFSrQIsXzKkph0oXD++CgCYwK
eq4YScVclT07OLwI+ksOADBp2j0rsj9We7EFcWMFoHFZnaEV+K5EnfzCWPGFuR1
KmvAcDycoopM5f9ygeHa7+bHHvhcXcKj24qvABEBAAQKkthbWlsIFN1Y2hhbmVr
IDxrYW1pbHN1Y2hhbmVrOTVAZ21haWwWuY29tPokBVAQTAQgAPhYhBj/cTkD3Sw6j
3O+1ONZ6nKHmVU8LBQJeemy1AhsDBQKdWmcABQsJCACCBhUKCQgLAQAgMBAh4B
AheAAAJENZ6nKHmVU8LBQJeemy1AhsDBQKdWmcABQsJCACCBhUKCQgLAQAgMBAh4B
SjDWer2Y+g4C6pqKB1M6X1FsQ8rmj/BX+cBWbYuPDjIlgPb3UiS8Fk5uASLxMel7a
/EPyWjSpGPiPVrjaTedGqbKjPlZSzn79vwGBYBqgdsN2qD07ZOq3hFjMXEPTXp8
Fm4VnsnMkHYt/6jliH2DVM8L+Em+C0a0VyhCsp7qbyi6yL5WCX7wn+LHIq7HVW/
vITHLxh4I2WgSeOqWtfXFKZkl7yXjRjDKCjG3BJUGSj9/zsq/9xzYTF4QMjX4IR
7y513Wsox5Qp5lirtX4A+ib0u5wr9+NWmduhn3tFV5volPO5AQ0EXnpstQEIAMH3
u9ja07LivZlxPlyrBktlQE0TBZ755f1YWdtHUHlj9F50kd2g7lxKHxjWD+gG5Qb
Adm9PWS/kxt+IjyvapMF/6Whe7j/qG555Pav+S1YE4A73PorH6L0KD/WchKPS7yxw
ZUW6iadwkaHqLePYEq2eKlkdYhWmDx4fluHTGW0yfiTAzEOWMjki8UH16kECzmy
aeYrPumylxXQdWwa+YScGgZP8nj/DXTLTXTHEww4KvkcpZNY8w5ly7ocRmx5vQ
sdWX7fZHzhVjc7BvKDLs314id8ccb6rd3amh5+nhqH91A5IAuWmryGsuQhQueWuW
TSY7RsvbhbcADCvYlwUAEQEAAYkBAQYAQgAjhYhBj/cTkD3Sw6j3O+1ONZ6nKHm
VU8LBQJeemy1AhsMBQKdWmcAAAJENZ6nKHmVU8LA1QH/2Xf7iyxkR6SAduREMre
xbe17CdbAFrbjMHxQA9XEif2mtUV+qAVBRGneKL+oAKbnyUgljvB2KokvD+icfP
AYWcmD9mMeAwxbqVGz3dFCTb2aEbeV2jTgTz2G+Nd4rPk52pbCZgUjuA9001Ac
MkXspCQ3+QmwmfEdzY4wX5AmCq6fq1U4TKQIKWVl+zDR8EXYAGPKp0cVJAetU/o
E040Rr3UORdye/610VKwBqpu83bZr4U56xngM1cRtdL6oTLjCzudnITFZuHyFGU
VTTpLsn/YVwKWjNfPg3XXmM/QrFTYsk/f4/C60dW6eimjNTjysY1wiCCAwXvMH
GzM=
=pLjO
-----END PGP PUBLIC KEY BLOCK-----

```

❖ sprawdzenie rozwiązania:

3

**SELECT \* FROM patients;**

Data Output

	patient_id [PK] integer	first_name bytea	last_name bytea	address bytea	birth_place bytea	pesel bytea
1	1	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]
2	2	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]
3	3	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]
4	4	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]
5	5	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]

Rys. 9 - fragment narzędzia PgAdmin4 - dane po zaszyfrowaniu.



```

38 SELECT patient_id,
39 pgp_key_id(first_name) AS first_name,
40 pgp_key_id(last_name) AS last_name
41 FROM patients;

```

	patient_id [PK] integer	first_name text	last_name text
1	1	084C98FD32C...	084C98FD32C...
2	2	084C98FD32C...	084C98FD32C...
3	3	084C98FD32C...	084C98FD32C...
4	4	084C98FD32C...	084C98FD32C...
5	5	084C98FD32C...	084C98FD32C...

```

C:\Users\kamis>gpg --edit-key Kamil
gpg (GnuPG) 2.2.20; Copyright (C) 2020 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Dostępny jest klucz tajny.

sec  rsa2048/D67A9CA1E6554F0B
    utworzono: 2020-03-24  wygasa: 2022-03-24  użycie: SC
    zaufanie: absolutne  poprawność: absolutne
ssb  rsa2048/084C98FD32CA83EB
    utworzono: 2020-03-24  wygasa: 2022-03-24  użycie: E
[  absolutne  ] (1). Kamil Suchanek <kamilsuchanek95@gmail.com>

gpg> exit

Niepoprawne polecenie (spróbuj „help”)

gpg> quit

```

Rys. 10 - Fragment narzędzia PgAdmin4 oraz terminala (po prawej) - kolorem żółtym zaznaczono pokrywające się numery ID zastosowanego klucza.

```

43 SELECT patient_id,
44 pgp_pub_decrypt(first_name, keys.privkey, 'password') AS first_name,
45 pgp_pub_decrypt(last_name, keys.privkey, 'password') AS last_name,
46 pgp_pub_decrypt(pesel, keys.privkey, 'password') AS pesel
47 FROM patients
48 CROSS JOIN
49 (SELECT dearmor('-----BEGIN PGP PRIVATE KEY BLOCK-----
50
51 lQPGBF56bLUBCADNoxSLBT/ou9RMtlnCxbJVgmjvcLDVf8NG8Suew/XZI2Bs5PjO
52 d7aR/kvk/DkBRD96/Q6yW0vt90Yf32tQOZUgylOqnpBC1o0TmWPYTFcb50KJNDwa
53 D85d70BC05UyTAcGcAKDwI7bCfOdQ1LTH2+b3078Fxf78w3+K8C4704TcBCUMkbl

```

	patient_id [PK] integer	first_name text	last_name text	pesel text
1	1	Kamil	Suchanek	43898765672
2	2	Dominika	Gabór	87654678926
3	3	Agnieszka	Rudziun	67489356728
4	4	Natalia	Matuszek	78754378961
5	5	Marcin	Kasprzyk	87546783218

```

-----BEGIN PGP PRIVATE KEY BLOCK-----

lQPGBF56bLUBCADNoxSLBT/ou9RMtlnCxbJVgmjvcLDVf8NG8Suew/XZI2Bs5PjO
d7aR/kvk/DkBRD96/Q6yW0vt90Yf32tQOZUgylOqnpBC1o0TmWPYTFcb50KJNDwa
D85d70BC05UyTAcGcAKDwI7bCfOdQ1LTH2+b3078Fxf78w3+K8C4704TcBCUMkbl
-----END PGP PRIVATE KEY BLOCK-----

```

Rys. 11 - Fragment narzędzia PgAdmin4 - Proces odszyfrowywania danych.

Uzyskiwanie danych wymaga klucza prywatnego oraz hasła, które można dodać przy generowaniu klucza.

❖ Zestawienie czasu realizacji zapytania z szyfrowaniem i bez:

➤ bez szyfrowania: 133ms aż do 119-130ms powtarzając czynność:

```
SELECT patient_id, first_name, last_name, pesel
FROM patients;
select*from patients;
```

➤ z szyfrowaniem: 1 sekunda i 459ms aż do sekundy i 440ms:

```
SELECT patient_id,
pgp_pub_decrypt(first_name, keys.privkey, 'password ') AS first_name,
pgp_pub_decrypt(last_name, keys.privkey, 'password ') AS last_name,
pgp_pub_decrypt(pesel, keys.privkey, 'password ') AS pesel
FROM patients
CROSS JOIN
(SELECT dearmor('-----BEGIN PGP PRIVATE KEY BLOCK-----
```

```
IQPGBF56bLUBCADNoxSLBT/ou9RMtInCxbJvGmjvclDVf8NG8Suew/XZI2Bs5PjO
d7aR/kvk/DkBRD96/Q6yW0vt9OYf32tQOZUgylOqnpBC1o0TmWPYTFcb50KJNDwa
D85d70PG9EHvIAsGpAKDWIzhGiQd01+TU2tb2OjBFxf78w2tK8G47O4lqRGHMkhl
iJGxoCapaVFQyuT6RR+/jD0mZ0K7rsFNqVm4q/iFSrQlsXzKkph0oXD++CgCYwK
eg4YScVclT07OLwl+jksOADBp2J0rsj9W67EfcWMFoHFZnaEV+K5EnfzCWPFGFuR1
KmvAcDycoopM5f9ygeHa7+bHHvhcXcKj24qvABEBAAH+BwMCyzsf2dsuHmPFD/dq
pK1SM5Buy5bCoBKUvVvsQ8+d06pC2xUvNXf17Kvs2lI3MPTIWP4ptyE9sHe/FPRA
Wf0GRI/Y+6+90uKeo53ojb8fMmrvbPldtarS4qlQ57wcWn90JbflwQ94ZQNQKcXz
7suonW18khhHjlbWmcqvAdBmLmuRgozypnQ/hjxyrueKTSUvAv6epZUKbouWY3bp
7k5vnrJt4pRH71NUcYzSzkgleA+BxfS8v4Cjw3lYQV2hl2CZSlv4ZvT4a2KEs/
/HU5sweXuhnWIE2d1y0Sfmc7xMmHdXC7+IWQx1fmAGlgvvhxjNc4Mup6sSkpQldj
CV1XHsYkdjHlUUV244SDevrU2Y4k6ibvs89j4Rx00/W9o2L1DUAczYrt+0steqPT
R6CBMYPX8X66rXjY58lyayFjzy9+Pl6FXm7qawuExBxconUdM/iSyQFyZe8lVNr
OC08E6ZoADiAlnoD5Dq5x3lvaF2Kzb7wra4Ob1N1ictvClg9KGKoPzqLRrRtJDU
A6to9sq7SzlAKS2ymheNgc0APeYtrrMrWieUgONyZX4w4QTEXUNXu2xzA6sraBqg
agwJNaxl5hpPAsUARKh96Pw+WOWCcysz7vAdt875UNvusQpbNdeI7UjRe01O3tnc6
+HKL6a8lal1mdMqp4MncemjgkTD/BXYKuWfZxY3bkySYs+FKOv69off+Sfhu/o
g/OXkxiQUvffeM9ypPeHPuSiGor417JlzhxdQ2Q7kjoIyJkdCtQie3l9el7Hts
5VBZblGHa18shKfJA6Mkls2vXqT7DsxCFkuAMyZyUg7XnM6j50hRYVsdCmFmb
j84fRDiPprrGjW0W0hWZ45m9HG6jKzdaJkedkPhyzwefrnETE6+iiQsXClUwcqYeD
NlMCZKX07kKa2tCpLWY1pbcBTdWNoYW5layA8a2f2aWxzdwNoYW5lazz1QgdTYWls
LmNvbT6jAVQEEwEIAAD4WIO5f3E5A90sOidzvtTjWepyh5IVPCwUCXnpstQjAwUj
A8jNAAULCQqHAgYVgCkIcWIEFgIDAQleAQIXgAAKCRDWepyh5IVPCxkKCADJRya/
9dhlhx0tP5aFy1CFDDsBeh9jEhpoT2VUow1nq9mP0AuaigdT0I9RbELfjvyQ
V/nAm2LjwySID2911kvB2ObqEi8THiO2vxD8lv0qYD5T1a42k3nRqm5CaSGUs5+
/b8BgWAAqnbDdgg9O2Tqt4RaiTFxD7V6fBZuF7ZjzB2Lf+o5Yodg1TPC/hjvgtG
tFcoXArD+6m8ousi+Vgl+8j/ixyKux1cP7yExy8YeJZdl0EnjqlRVXsSmZJe8sY0
YwynCRtI1Bkpf87Kvjcc2ExeEDCV+JUeBudd1rKMeUKeZSK7V+AppW9LUCk/fj
Vpnb0Z97X1Ur6FDznQPGBF56bLUBCADB97vSWjuy5b2ZCt5cqwSrXZUBNEwWe+eX
9WFnbR1B5Y/RUtJA9oOyMSh141g/oBuUGwH2vT1kv5F7fp8r2qTBF+loXu4/6hu
eT2r/ktWBOAO9z6Kx+i9AJ/1nByj0u8ScGVfuomncjGoai3j2BKtnii5HWlCFpg8
eH5k7RltMn4kwMxDljCZCPFB9epBA55smnv2Kz7pspcV0F3VsGvmEgoKmtJ/yfw
107U10xxMOCr5HKWTWPMOZcu6HEZseb0LHVl+8xWR70I3Owb5Ay7N9eJXfHHG+q
3d2p0Uvp4X0B/dQOSALlpg2Brk86Lnlrlk0m00bL24W3AAw2jcfABEBAAH+BwMC
ej9orXG3SznFsiEHnkUy4D1M6C7Pp/BPG8Rz5MqgVWL3jHY76ilI0uPOonG+RuJO
RtEmCd0tXCCyT3PdC7LvfifeN80dxqUtkXOCkJOdcBKrK9naS+Jd5f6OFFUvVpY
JScLdi34Aa6jrxPOHyfyqZnHTCy1STl/oa0bWR9Rjq3s2/qUIBPXpe5Lw6Y8GGZb
KeHl5jeUhx0KUfLoNqK5atbqKZdZrdeu+MevDHm52/DZUyqX8yAOKsyht/Wtbyw
MnjlbVDAUjWwZgRj/bnrjVMbn0FfoKJQDbnQhJOvXOTxs1T61j7fENW6gBjT1B6
F0yDj48Kj0jdyVpp69s5mqHHLPrue1RkMJo9vrrHC8CH3T0Qf+KyrXYEm4iRH46GV
564yUEgn8lwqlqYZOI0tXN01j9pxc1ThqztzDMoE3OZzIKfN4S2753Ba3/1aOB
ircZuCu6TRIG0yoz6WPqymwXZwCEggP+H1NhfdpTyYGV2chLY5KaPtR5jAN0ThR
YWykWVBoHah0gerc5qjHRIAGVg3H1pFKHMT4e50Nsn2lhFi+34Ch2xDonUrvIhdv
pzGUkDNxXegpolg6zP355NHqBUj/mBXWGFRTtooxAHFBUFHwPYLUUmEryR0z11rUG
CK5bp5FcUb5nl7mt+tvgmDa9hc+MsHytrfjivmQODCs8CkdZWqermZkny091hUj
P5/K1ENP3yggWYEGZTRBgXLDnRkVpyzQswtPvK4Yp/q8zqFK/AqfbjzRVaxB3oP
je1bxuTDXH55cHuWxV6b5pE6qZZO4Zp+ax1dkZw65Dv5EKXnj2KF7C0LoiLkTtbk
M4vHjgvc5GNHIN9p0VdJgCpHjK/6LL3vadnuwCpAJLx/qwBvP25pZqdD2buxojs
Lzsq0Z35EtznD2rgyEtt6vdZt68riQE8BBgBCAAmFIEEn9xOQpDlDonc77U41nqc
oeZVTwsFAI56bLUCGwwFCQPCZwAACgkQInqcoeZVTwsDVAf/Zd/uLLGRHpiB25EQ
yt7f7XsJ1sAWtskwwfAD1cR6m/aa1RX60bUfEad4g0v6gCRUfj5om8HYagg8P6N
x88Bh2wx32Y4DDFupUbPd0UjVZorR5Xa0lOBPODYb413is+SzaltxmBQm4D07T
UBWYReyk7f5CbCZ8R3NjBfKcYrp+DVThORClpZwX7M0vWrdqAY8bnRX8BB60l
T+gTjJRgvdQ6tEPj7rU5UrAGom7zdvOvhrLGoAZvxFN0vqhMskLO52chMvm4fl
UZRvNOKuyf9hXApZWm0WkbddeYz9CsVniyT9j8LR1bp6KaM1PKOxjWLCUIIDBe
8wcbMw==
=79ID
-----END PGP PRIVATE KEY BLOCK-----
```

') As privkey) As keys;

Tym razem różnica w czasie realizacji zapytania jest znaczna.

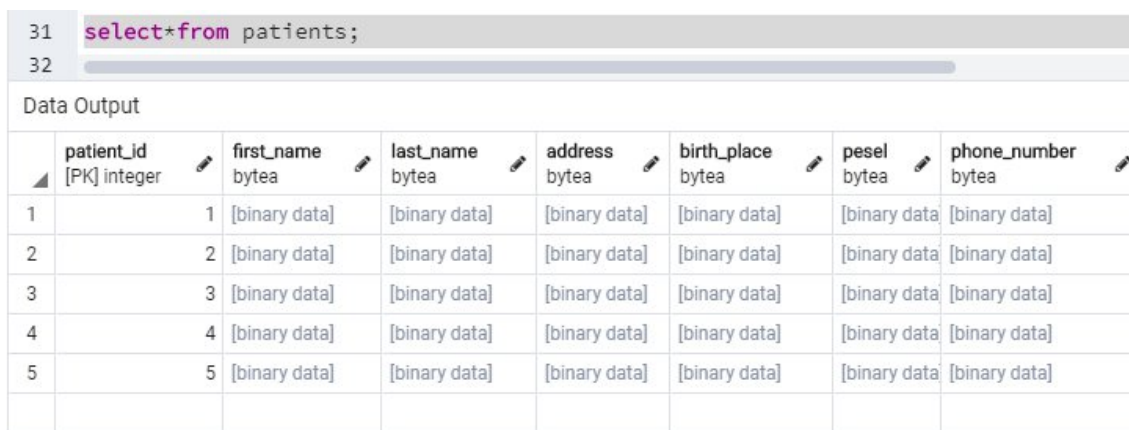
## 5.4. Szyfrowanie symetryczne

Wykonano szyfrowanie jednym kluczem, służącym zarówno do ukrywania jak i wyjawiania zaszyfrowanej treści. Klucz taki często zawarty jest po stronie klienta, w ten sposób dane nie są zagrożone od strony administratora bazy danych.

Klucz może stanowić prawie dowolny ciąg znaków. zastosowano ponownie funkcje oferowane przez rozszerzenie pgcrypto. Kod:

```
INSERT INTO
patients(first_name,last_name,address,birth_place,pesel,phone_number,notes)
SELECT pgp_sym_encrypt(pat.fn, 'password') AS first_name,
       pgp_sym_encrypt(pat.ln, 'password') AS last_name,
       pgp_sym_encrypt(pat.ad, 'password') AS address,
       pgp_sym_encrypt(pat.bi, 'password') AS birth_place,
       pgp_sym_encrypt(pat.pe, 'password') AS pesel,
       pgp_sym_encrypt(pat.ph, 'password') AS phone_number,
       pgp_sym_encrypt(pat.no, 'password') AS notes
FROM (VALUES
('Kamil','Suchenek','Bławatkowa 9/13','Gliwice','43898765672','467598762','---'),
('Dominika','Gabór','Nakło Śląskie Morcinka 9','Tarnowskie Góry','87654678926','765398267','al.tetracykliny'),
('Agnieszka','Rudziun','Tarnogorska 8 Stare Tarnowice','Oświęcim','67489356728','654329873','-'),
('Natalia','Matuszek','Słoneczna 5 Krupski Młyn','Lubliniec','78754378961','876539846','agorafobia'),
('Marcin','Kasprzyk','Celiny 8 Tąpkowice','Ruda Śląska','87546783218','987654323','po odwyku') )
AS pat(fn,ln,ad,bi,pe,ph,no);
```

### ❖ Zaszyfrowane dane:



The screenshot shows a SQL query in PgAdmin4: `select*from patients;`. Below the query, the 'Data Output' table is displayed. The table has 8 columns: `patient_id` (integer), `first_name` (bytea), `last_name` (bytea), `address` (bytea), `birth_place` (bytea), `pesel` (bytea), `phone_number` (bytea), and an empty column. The first 5 rows show encrypted data for the first five records in the `patients` table. The data is represented as binary values in square brackets.

	patient_id [PK] integer	first_name bytea	last_name bytea	address bytea	birth_place bytea	pesel bytea	phone_number bytea	
1	1	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]	
2	2	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]	
3	3	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]	
4	4	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]	
5	5	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]	[binary data]	

Rys. 12 - Fragment narzędzia PgAdmin4 - Zaszyfrowane dane.

### ❖ Zapytanie klienta do odszyfrowania danych:

```
SELECT patient_id,
pgp_sym_decrypt(first_name, 'password') AS first_name,
pgp_sym_decrypt(last_name, 'password') AS last_name,
pgp_sym_decrypt(pesel, 'password') AS pesel
FROM patients;
```

❖ Zestawienie czasu zapytań z szyfrowaniem i bez:

➤ z szyfrowaniem: 150ms aż do 140ms

```
SELECT patient_id,  
pgp_sym_decrypt(first_name, 'password') AS first_name,  
pgp_sym_decrypt(last_name, 'password') AS last_name,  
pgp_sym_decrypt(pesel, 'password') AS pesel  
FROM patients;
```

➤ bez szyfrowania: 133ms aż do 119-130ms powtarzając czynność:

```
SELECT patient_id, first_name, last_name, pesel  
FROM patients;  
select*from patients;
```

Podobnie jak w przypadku szyfrowania peselu, różnica w czasie realizacji zapytań z szyfrowaniem i bez istnieje, lecz jest niewielka.

## 6.Podsumowanie

Zaprojektowano, zaimplementowano i wdrożono przykładową, wzorcową bazę danych medycznych, dla danego aspektu pracy przychodni. Przedstawiono i zastosowano dwa sposoby zabezpieczenia bazy przy pomocy szyfrowania danych za pomocą algorytmu md5 z pomocą "soli" oraz pary kluczy szyfrowania dwukierunkowego.

Opracowanie stanowi podstawowe, zręczne wprowadzenie do praktycznego aspektu szyfrowania danych stosując silnik bazodanowy PostgreSQL.

## Literatura:

- [1] "SQL Wydanie III" Danuta Mendrala, Marcin Szeliga
- [2] "Php i MySQL Od nowicjusza do wojownika ninja" Kevin Yank
- [3] Wykłady Systemy informatyczne w medycynie "Zarządzanie indywidualną dokumentacją medyczną" Piotr Zarychta
- [4] [postgresql.org/docs/9.1/auth-pg-hb-a-conf.html](https://www.postgresql.org/docs/9.1/auth-pg-hba-conf.html) [dostęp 27.03.2020]
- [5] [postgresql.org/](https://www.postgresql.org/) [dostęp 14.03.2020]
- [6] [postgresql.org/docs/current/pgcrypto.html](https://www.postgresql.org/docs/current/pgcrypto.html) [dostęp 24.03.2020 - "current" oznacza wersję PostgreSQL 12].

### Dodatkowe źródła merytoryczne:

- ❖ [stackoverflow.com/questions/24496536/best-practices-for-using-pgcrypto-pgp-encryption-with-heroku-and-rails](https://stackoverflow.com/questions/24496536/best-practices-for-using-pgcrypto-pgp-encryption-with-heroku-and-rails)
- ❖ [stackoverflow.com/questions/10752456/how-to-enable-contrib-modules-on-heroku-postgres-database](https://stackoverflow.com/questions/10752456/how-to-enable-contrib-modules-on-heroku-postgres-database)
- ❖ [postgresonline.com/journal/archives/165-Encrypting-data-with-pgcrypto.html](https://postgresonline.com/journal/archives/165-Encrypting-data-with-pgcrypto.html)
- ❖ [severalnines.com/database-blog/how-secure-your-postgresql-database-10-tips](https://severalnines.com/database-blog/how-secure-your-postgresql-database-10-tips)
- ❖ [dbrnd.com/2016/03/postgresql-best-way-for-password-encryption-using-pgcrypto-cryptographic-functions/](https://dbrnd.com/2016/03/postgresql-best-way-for-password-encryption-using-pgcrypto-cryptographic-functions/)

[dostęp do powyższych źródeł 26.03.2020]

*Zbiór skryptów jest dostępny na repozytorium:*

*[github.com/KamilSuchanek95/projekt-cyber](https://github.com/KamilSuchanek95/projekt-cyber)*

*[Gałąź master zawiera podstawową implementację bazy bez szyfrowania, gałąź pesel zawiera implementację dla szyfrowania tylko numeru pesel, a gałąź gpg zawiera skrypty dla szyfrowania całej tabeli pacjentów, kod dla szyfrowania symetrycznego nie został uwzględniony, jest analogiczny dla gałęzi gpg z drobną zmianą ujętą w przedstawionym sprawozdaniu.]*