Zaawansowane Systemy Baz Danych – Etap 1 „Przychodnia”

Michał Ankiersztajn 311171

## Części 1-4

W plikach znajduje się Case Study, realnie PESEL powinien znaleźć się na recepcie, jednak, aby nie złamać 3NF nie jest tam umieszczony. Będzie można się dostać do PESELu pacjenta poprzez wizytę.

Załączam tu poglądowo schemat z zaktualizowanymi wielkościami varchar:

Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Opis wygenerowany automatycznie

## 5. Decyzje

Uwaga załączony razem z plikiem PDF jest projekt, poniższe decyzje dotyczą pliku **SQl/init.sql**, który jest odpalany przy tworzeniu bazy danych.

Zdecydowałem się zmienić wielkości varchar, dla imienia na 55, choć istnieją historycznie nazwiska, która mają 963 znaków szansa, że będzie potrzeba na ich skorzystanie jest nikła. Dodatkowo jest to skrajny procent społeczeństwa, na tyle skrajny, że nawet jeśli obetniemy ich nazwisko do 55 znaków i zostawimy imię to nadal jesteśmy ich w stanie zidentyfikować.

Email został zmieniony na 254 ponieważ jest to maksymalna długość maila.

Komentarz został wyspecyfikowany na 520, ponieważ jest to zazwyczaj wystarczająca liczba znaków, aby dodać krótki i znaczący opis.

Korzystam ze składni CREATE TABLE IF NOT EXISTS, aby uniknąć błędów przy powtórnym tworzeniu bazy danych. Jeśli tabele już istnieją to nie chcemy ich nadpisywać, a zostawić tak jak były.

Korzystam ze składni SERIAL zamiast INTEGER dla PRIMARY KEY, abym nie musiał ‘ręcznie’ zarządzać ID w tabelach, co znacznie uprości proces populacji tabel.

Korzystam ze składni CONSTRAINT name FOREIGN KEY, aby w wypadku błędu wiedzieć, który CONSTRAINT go spowodował.

Wszystkie dane przechowywane są jako NOT NULL, ponieważ wszystkie dane w wypisanych tabelach są wymagane od samego początku. Jeśli dane są opcjonalne (jak np. recepta, czy ocena) to są one w relacji 0..1 - 1

Zdecydowałem się użyć UNIQUE dla:

1. Specialization(Name) – nazwa musi być unikalna, aby uniknąć duplikacji danych. Nie chcemy mieć wielu tabel dla jednej i tej samej specjalizacji.
2. Prescription(Code) – kod recepty musi być unikalny, jeśli z jakiegoś powodu wystąpił duplikat oznacza to po prostu błąd.

Tworząc bazę danych jednocześnie tworzę scheme **public** do której podstawowo dodawane są wszystkie tabele, dzięki temu jestem w stanie ją wyeksportować komendą:

**docker exec -it zsbd-postgresql\_database-1 pg\_dump -U admin --schema-only MediPlaceDatabase > schema.sql**

Plik **schema.sql** został załączony do wglądu

## 6. Wypełnienie danymi

Do wypełnienia danymi skorzystałem z ChatGPT. Podałem mu schemę bazy danych i poprosiłem o wypełnienie ‘mockowymi’ danymi. Lekko zmodyfikowałem wrzucone przez niego dane, aby baza była bardziej skomplikowana: link do rozmowy: <https://chatgpt.com/share/67179ff9-8688-8011-8b19-db8e45283eee>

## 7. Stworzenie użytkowników

Baza danych będzie posiadała 4 użytkowników.

Dostęp będzie weryfikowany za pomocą komendy:

**SELECT \* FROM pg\_tables WHERE tableowner = 'USER';**

Zrzuty z tej komendy dla każdego użytkownika są załączone razem ze sprawozdaniem.

1. **Admin**

Użytkownik, który jest podstawowo tworzony razem z bazą danych i ma dostęp do wszystkiego. Jego uprawnienia wyciągam w następujący sposób:

Obraz zawierający tekst, zrzut ekranu, Jaskrawoniebieski, Czcionka

Opis wygenerowany automatycznie

Uprawnienia załączone w pliku **admin\_permissions.txt**

1. **Manager**

* Pełna kontrola nad doktorami, specjalizacjami i ich połączeniem
* Wgląd w wyniki doktorów poprzez wizyty + oceny

Stworzenie użytkownika + nadanie uprawnień:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Zrzut uprawnień:

Obraz zawierający tekst, Czcionka, Jaskrawoniebieski, zrzut ekranu

Opis wygenerowany automatycznie

Uprawnienia załączone w pliku **manager\_permissions.txt**

1. **Pacjent**

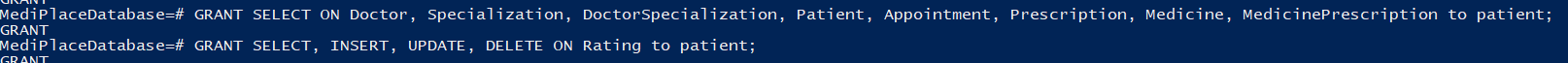
* Kontroluje ocenę wizyty
* Ma wgląd w większość tabel – nie widzi innych pacjentów i managerów

Stworzenie użytkownika:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Nadanie uprawnień:



Zrzut uprawnień analogicznie jak wcześniej, załączony w pliku **patient\_permissions.txt**

1. **Doktor**

* Pełna kontrola nad wizytami
* Może dodawać leki
* Może czytać, dodawać i usuwać recepty, ale nie je modyfikować, aby zapobiec sytuacją, gdzie doktor próbuje zmienić kod recepty po tym jak pacjent już jedną odebrał. Chcemy, aby pacjent najpierw skonsultował się z doktorem za pomocą wizyty zanim otrzyma kolejną receptę.
* Może dodawać leki do recepty.
* Wgląd w większość tabel – jedynie nie widzi managerów

Stworzenie użytkownika:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Nadanie uprawnień:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Zrzut uprawnień analogicznie jak wcześniej, załączony w pliku **doctor\_permissions.txt**

## 8. Dwa zapytania

1. Zdecydowałem się stworzyć zapytanie, które zwraca 5 doktorów z najwyższą średnią satysfakcją na swoich wizytach. Jest to potrzebne ze strony biznesu, aby manager był w stanie zidentyfikować dobrze prosperujących doktorów i dać im np. podwyżkę, analogicznie można sprawdzić najniższą ocenę i takich doktorów zwolnić.

Liczą się tylko doktorzy, którzy mają co najmniej 30 wizyt i 5 ocen:

Obraz zawierający tekst, zrzut ekranu, Czcionka, Jaskrawoniebieski

Opis wygenerowany automatycznie

Moja baza ma zbyt mało danych, jeśli usunąć linijkę HAVING COUNT… to uzyskamy:

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Jeden z doktorów nie otrzymał jeszcze żadnej oceny, dlatego się nie wyświetla – jest to pożądane działanie.

1. Sprawdzenie przez pacjenta doktorów którzy są dostępni w zakresie dat i zależnie od specjalizacji:

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

Wyświetla się tylko Alice Brown jest ona jedynym doktorem z obydwoma specjalizacjami, który jest dostępny danego dnia.

## 9. Perspektywy

W systemie przychodni ze względu na prywatność większość z widoków nie powinien być używany wprost a przez funkcję lub query, która filtruje po ID danego doktora, managera, czy pacjenta.

1. Dla Managera:

* **doctorsPerformanceView** – wszyscy doktorów wraz z danymi ilościowymi, ile odbyli wizyt, ile wizyt dostało oceny, jaka jest średnia ocen ich wizyt

1. Dla Doktora i Pacjenta:

* **appointmentDetailsView** – szczegóły na temat wizyty wraz z oceną, kodem recepty i listą leków

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Jeśli dana wizyta nie ma oceny, kodu lub leków to nie zostaną one wyświetlone.

1. Dla wszystkich:

* **doctorDetailsView** – szczegóły na temat doktora wraz z listą jego specjalizacji, liczbą odbytych wizyt i średnią oceną

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

## 10. Indeksy

Sprawdzam, które kolumny domyślnie mają ustawione indeksy:

Obraz zawierający tekst, zrzut ekranu, Czcionka, menu

Opis wygenerowany automatycznie

Przy okazji sprawdziłem też definicję indexów, wszystkie z nich korzystają z algorytmu btree. Oznacza to, że PostgreSQL domyślnie tworzy indeksy dla kluczy głównych oraz kolumn typu UNIQUE. Dodatkowo są to specjalne indeksy typu UNIQUE, które nie pozwalają na powtórki danych.

Indeksy, które warto założyć to również te na klucze obce, ponieważ są one często wykorzystywane w przeszukiwaniu(przykład dla paru kolumn):

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Wszystkie te indeksy korzystają z algorytmu btree.

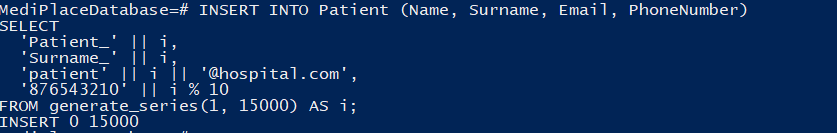
Kolejną wartościową kolumną z indeksami są daty, w tym przypadku występuje tylko data wizyty.



Teoretycznie moglibyśmy założyć indeksy na imię i nazwisko pacjenta oraz doktora, bo są to typowo często przeszukiwane kolumny dla systemów. **Jednak z założenia funkcjonalności tego systemu nie ma to większego sensu** (nie ma planowanego dostępu do wyszukiwarki doktorów/pacjentów, a jedynie dostęp do nich poprzez **id wizyty lub specjalizacje**).

W aplikacji nie ma też miejsca na indeksy kompozytowe, żadna z tabel nie ma być docelowo przeszukiwana po 2 lub więcej kolumnach na raz.

Test efektywności indeksów, przetestuję 3 rodzaje indeksów. Najpierw wrzucę po 15-30 tyś. Mockowych rekordów to tabel z których będę korzystał, przykład z pacjentem:



Do testów będę urachamiał SELECT 1\_000\_000 razy z odpalonym timingiem, aby uzyskać dokładniejsze dane statystyczne:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Jest wiele różnych [typów indeksów](https://www.postgresql.org/docs/current/indexes-types.html), jednak ja zdecydowałem się przetestować to na tych 3:

**Brak indeksów:**

Tu jestem w stanie usunąć indeks na klucze obce oraz datę ze względu na ustawione constrainty.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Btree:**

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Btree w tym wypadku jest niewiele szybsze od braku indeksów.

**HASH:**

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

HASH jest bardzo podobne do btree. Dlaczego? Wynika to z tego na jakich operatorach logicznych działają dane algorytmy. Btree działa na (<, <=, =, >=, >), a Hash na (=). Dla danego zapytania korzystamy jedynie z (=), co oznacza, że algorytmy Hash i Btree powinny uzyskać bliźniaczy wynik.

Dla danego zapytania najlepiej działa

Zrzut wszystkich indeksów w bazie danych wraz ze szczegółami znajduje się w pliku **indexes.txt**

# Bibliografia:

<https://www.postgresqltutorial.com/>

<https://stackoverflow.com/questions/14486241/postgresql-export-the-schema-of-a-database>

<https://stackoverflow.com/questions/5331320/how-to-save-psql-output-to-a-file>

<https://stackoverflow.com/questions/22049212/copying-files-from-docker-container-to-host>

<https://stackoverflow.com/questions/760210/how-do-you-create-a-read-only-user-in-postgresql>

<https://chatgpt.com/share/67179ff9-8688-8011-8b19-db8e45283eee>

<https://en.wikipedia.org/wiki/Hubert_Blaine_Wolfeschlegelsteinhausenbergerdorff_Sr>.

<https://stackoverflow.com/questions/1199190/what-is-the-optimal-length-for-an-email-address-in-a-database>

<https://simplebackups.com/blog/postgresql-how-to-list-indexes/>

<https://www.postgresql.org/docs/current/sql-createindex.html>

<https://www.postgresql.org/docs/current/indexes-types.html>