Zaawansowane Systemy Baz Danych – Etap 2 „Przychodnia”

Michał Ankiersztajn 311171

## UDF

Główne różnice między funkcją, a procedurą:

|  |  |
| --- | --- |
| **Funkcja** | **Procedura** |
| Może tylko czytać dane | Może czytać i modyfikować dane |
| Autonomiczne funkcje mogą być częścią zapytania | Nie może być częścią zapytania |
| Kompilowana za każdym razem | Kompilowana raz i używana wielokrotnie |
| Brak transakcji | Wsparcie dla transakcji |
| Nie może wywoływać procedur | Może wywoływać funkcje |
| Zawsze zwraca wynik na końcu | Procedura może zwracać wynik za pomocą OUT i IN OUT |

Podsumowując, funkcje lepiej wykorzystywać dla mniejszych i reużywalnych fragmentów kodu. Kod ten można potem użyć zarówno w innych funkcjach, jak i procedurach. Dodatkowo są one bardziej ograniczone, a co za tym idzie prostsze do zrozumienia! Największym ograniczeniem jest readonly dla danych.

Procedury są dużo bardziej skomplikowane i na wiele więcej pozwalają, jeśli funkcja nie jest wstanie wykonać tego co trzeba – najlepiej napisać procedurę.

Różnica pomiędzy wyzwalaczami, a procedurami jest taka, że wyzwalacze odpalane są automatycznie przy zmianach w tabeli, a procedurę musi ręcznie wywołać użytkownik. Dodatkowo wyzwalacze z powyższego powodu nie mogą zwracać żadnych wartości i są pewnego rodzaju automatyzacją naszej bazy danych.

Dodatkowo bardzo przydatna tabela z prezentacji „**KH\_Wykład\_ZSBD\_2024\_wyklad\_4.pdf”:**

**Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Opis wygenerowany automatycznie**

**Funkcje i procedury:**

1. **Manager:**

**Procedura pozwalają przypisać wiele specjalizacji do doktora po nazwach specjalizacji**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie**

Przykład użycia:

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

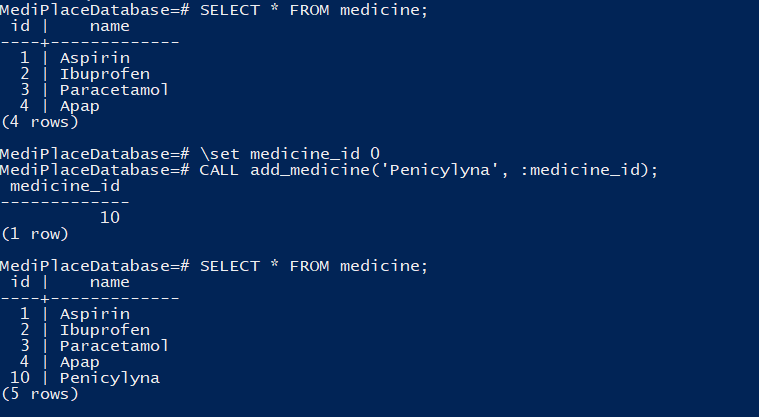
1. **Doktor:**

**Procedura pozwalająca na dodanie leku**

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Przykład użycia

****

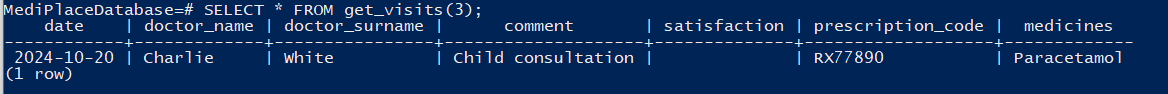
1. **Pacjent:**

**Funkcja wyświetlająca wizyty pacjenta wraz z imieniem i nazwiskiem doktora, kodem recepty, listą leków i oceną pacjenta:**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Przykładowe użycie:



**Funkcja wyświetlająca doktorów w zależności od podanych specjalizacji, daty oraz minimalnej ilości specjalizacji.**

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

Przykładowe użycie:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Procedura usuwająca wizytę o ile jest ona w przyszłości:**

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Przykład użycia:

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

**Wnioski:**

Pisząc funkcje i procedury zauważyłem, że najlepiej używać snake\_case’a dla zmiennych (przynajmniej przy tak przyjętym nazewnictwie dla table i kolumn jak w tym projekcie).

Funkcje są świetnym zabezpieczeniem przed modyfikacją danych.

Zarówno procedury i funkcje pozwalają na zwiększenie bezpieczeństwa systemu ograniczając to co użytkownik może zrobić.

## Złożona procedura

Procedura lekarza, która pozwala na stworzenie recepty z losowym kodem i dodanie listy leków. Jeśli lek nie istnieje zostaje on dodany do bazy danych. Wykorzystuje ona jedną z poprzednich procedur, która dodawała leki:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie



Początkowo tak wyglądał mój kod, okazuje się jednak, że PostgreSQL tworzy savepointy przy klauzuli EXCEPTION, więc korzystanie z explicit COMMIT powoduje subtransaction error. Oznacza to, że należy usunąć COMMIT ze skryptu, ponieważ wykorzystuje on już mechanizm transakcji pod spodem dzięki blokowi z EXCEPTION. Aktualizuję skrypt usuwając linijkę COMMIT.

Przykład użycia:

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Jak widać dane poprawnie zostały dodane do wszystkich 3 tabel.

## Triggery

W bazie PostgreSQL, aby użyć triggera trzeba zdefiniować funkcję, która najpierw zwraca TRIGGER.

**Przy usunięciu wizyty, powinien zostać usunięty komentarz, recepta i połączenie między receptą, a listą leków.**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Korzystam z BEFORE, ponieważ chciałbym uniknąć sytuacji w której w bazie choćby przez chwilę będą tabele posiadające referencje do nieistniejącej tabeli.

Przykład użycia:

PRZED:

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

PO:

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

**Wizyty powinny być tworzone tylko w przyszłości**

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Przykład użycia:



Jedną z największych wad jest to, że przy wywaleniu się TRIGGERa możemy nawet nie wiedzieć, że to się wydażyło. Początkowo ustawiłem RETURN NEW; przy delete co zwracało NULL, jednak nie wiedziałem wtedy, że NULL oznacza abort i cofnięcie transakcji. Przez takie ukryte mechanizmy można coś przypadkiem pominąć.

Dodatkowo, edycje triggerów są skomplikowane i mogą spowodować breaking changes.

## Agregacje

W wcześniejszych podpunktach zostały opisane potrzebne agregacje, wyzwalacze, a perspektywy w poprzednim sprawozdaniu.

Nie udało mi się zaimplementować poniższych agregacji:

* Dodanie oceny przez pacjenta.
* Edycja komentarza wizyty przez doktora
* Autouzupełnianie nazw leków przez lekarza

Nie pisałem dodatkowych procedur i funkcji dla ADMINa, ponieważ w PostgreSQL podstawowo istnieje wiele takich przydatnych procedur, część z nich wykorzystałem w skryptach jak np. CURRENT\_DATE

## Automatyzacja zadania z wykorzystaniem jobów

Przy aktualnym ustawieniu tabel ciężko jest dodać sensownego joba operującego na tabelach, dlatego dodam nową tabelę na miesięczne raporty, jest to bardzo przydatne z punktu biznesowego, aby przychodnia mogła się rozwijać wiedząc, co miało wpływ na satysfakcję, liczbę wizyt, pacjentów, doktorów i ich wykształcenie:

Nowa Tabela:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Procedura generująca raport (nie mieści się na screenie, jest na dole w pliku **scripts.sql**)

Do automatyzacji skorzystam z CRONa:

Najpierw instalacja przez terminal docker desktop:

Kolejno komendy:

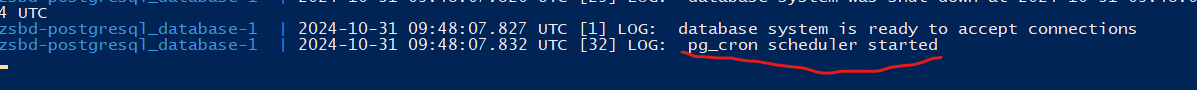
apt-get install postgresql-17-cron

Potem w /var/lib/postgresql/data dodaję shared\_preload\_libraries:

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

I restartuję serwer



Schedule:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Ostatniego dnia miesiąca o godz. 23:59 będzie generowany miesięczny raport.

Działanie:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Dla testu działania scheduling ustawiłem 15 sekund, realnie będzie ta komenda co wyżj w sekcji ‘Schedule’.

## Kopia zapasowa

Tworzenie backupu:



Weryfikacja:

Jak to zrobię?

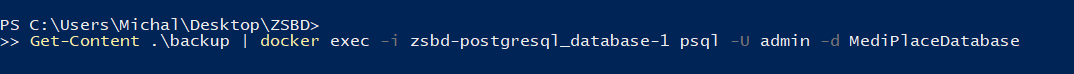
1. Usunę starą bazę danych



1. Stworzę ją od nowa bez skryptu init.sql



1. Wczytam backup



1. Eksportuję nową bazę danych



1. Porównuję, czy pliki są takie same, wklejając zawartość backup oraz backup\_new

<https://www.diffchecker.com/RExtzkyV/>

Skorzystałem z tej strony ignorując nowe linie, ponieważ przy eksporcie nowej bazy dodawane były znaki nowej linii, które nie wpływały na odtworzenie bazy danych.

## Konfiguracja MS SQL Server

Skorzystam z dockera o bardzo podobnej konfiguracji:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Tak jak wcześniej będę to uruchamiał za pomocą:

* docker-compose pull
* docker-compose up –build

Ciekawostka – MS SQL wymusza trudniejsze hasło podczas, gdy PostgreSQL pozwala na dowolne. Początkowo chciałem ustawić je jako takie samo, jednak failowało to builda.

## Analiza różnic PostgreSQL vs MS SQL Server

* MS SQL korzysta z T-SQL, a PostgreSQL korzysta z PL/pgSQL
* MS SQL korzysta z TRY CATCH zamiast EXCEPTION
* MS SQL nie wspiera pętli FOREACH
* MS SQL nie wymaga użycia procedur lub funkcji na triggerach
* MS SQL pozwala na pisanie triggerów w miejscu ich tworzenia, a nie przez funkcje/procedury
* MS SQL nie posiada typu ARRAY
* MS SQL posiada unikalność fk w ramach bazy danych, a nie pojedynczej tabeli.
* MS SQL Parametry funkcji i procedur zaczynają się od @
* MS SQL ma ograniczone i mniej zaawansowane indeksy względem PostgreSQL
* MS SQL ma ograniczone i mniej zaawansowane triggery względem PostgreSQL
* MS SQL nie posiada typu SERIAL
* MS SQL funkcje i procedury przyjmują parametry kolejno podawane, a postgreSQL przyjmuje je w nawiasie.
* MS SQL nie tworzy automatycznie transakcji na blokach TRY CATCH
* Różnice w składni przy dodawaniu permissionów dla użytkowników

Więcej różnic, których nie wypisałem: <https://www.enterprisedb.com/blog/microsoft-sql-server-mssql-vs-postgresql-comparison-details-what-differences>

Osobiście preferuję składnię w PostgreSQL, jest ona dla mnie prostsza i bardziej intuicyjna, składnia MS SQL Server jest według mnie po prostu lekko przekombinowana. Miałem też tu spore problemy z widokiem z terminala, zmusiło mnie to do instalacji DBeavera, aby sensownie operować danymi.

## Migracja PostgreSQL > MS SQL Server

Aby migracja przebiegła jak najprościej się da, wezmę wcześniej stworzony backup i skorzystam z narzędzia: <https://www.sqlines.com/online>

Po skorzystaniu z narzędzia z init.sql musiałem poprawić fk, ponieważ wcześniej były one unikalne w ramach tabeli, a nie bazy.

Użytkowników, dane, procedury, funkcje, indeksy oraz triggery przemigrowałem za pomocą ChatGPT:

<https://chatgpt.com/share/672388f0-1b74-8011-b355-20cf021c8871>

Popełniłem błąd próbując najpierw stworzyć triggery i procedury, a potem zaimportować dane, co spowodowało, że musiałem się cofnąć. Główną przyczyną był trigger nie pozwalający na dodawanie Appointment w przeszłości (co przy okazji zweryfikowało, że ta funkcjonalność działa).

Dodatkowo trzeba od nowa przypisać dostęp do poszczególnych procedur i funkcji dla użytkowników. Część z rzeczy musiałem sam przepisać (jak np. permissiony do procedur i funkcji), jednak Chat w tym wypadku wygenerował dużo sensownego i działającego kodu.

Cały skrypt z migracją znajduje się w katalogu **MS SQL Server/SQL/init.sql**

Okazuje się, że ChatGPT jest świetnym narzędziem do migracji danych z jednego systemu na drugi. Poprzednie narzędzie sqlines nie potrafiło przekonwertować procedur i funkcji nie powodując wielu błędów. Popełnia on czasem błędy, jednak są one raczej niewielkie.

SQLines okazało się tragicznym narzędziem, jedyne w czym podołało to tworzenie tabel i insert danych. 90% procedur, funkcji i triggerów nie została poprawnie przeprowadzona.

## 10. Działanie bazy na MS SQL Server

MS SQL posiada dużo gorsze narzędzie z poziomu terminala. Skorzystam z programu DBeaver:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie

Tabele:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Opis wygenerowany automatycznie

Indeksy i procedury:

Obraz zawierający tekst, zrzut ekranu, Czcionka, dokument

Opis wygenerowany automatycznie

Próba usunięcia appointment(działający trigger):

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

Brak autoincrementu id (trzeba było dodać identity(1,1), co spowodowało potrzebę usunięcia ze skryptu id insertów:

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

Błąd przy próbie dodania leku:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Wygenerowana procedura miała błędy. Po poprawie:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Wizyty danego pacjenta:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, linia

Opis wygenerowany automatycznie

Dostępni doktorzy:

Obraz zawierający tekst, linia, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

Dodawanie specjalizacji doktorowi:

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

Dodanie leku:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Opis wygenerowany automatycznie

Tworzenie recepty:

Obraz zawierający tekst, Czcionka, linia, numer

Opis wygenerowany automatycznie

## Bibliografia:

<https://www.postgresql.org/docs/current/xfunc.html>

<https://www.geeksforgeeks.org/difference-between-trigger-and-procedure-in-dbms/>

<https://www.geeksforgeeks.org/difference-between-function-and-procedure/>

<https://neon.tech/postgresql/postgresql-triggers>

<https://www.postgresql.org/docs/current/backup-dump.html#BACKUP-DUMP-RESTORE>

<https://dev.to/mdarifulhaque/how-to-backup-a-postgresql-database-in-docker-step-by-step-guide-cp2>

<https://forums.docker.com/t/postgresql-backup-and-restore-with-powershell/32125>

<https://github.com/citusdata/pg_cron>

<https://medium.com/@seventechnologiescloud/local-sqlserver-database-via-docker-compose-the-ultimate-guide-f1d9f0ac1354>

<https://khalidabuhakmeh.com/running-sql-server-queries-in-docker>

<https://stackoverflow.com/questions/71688125/odbc-driver-18-for-sql-serverssl-provider-error1416f086>