Projekt: Lekarz Kardiolog

1.

Krótki opis projektu

Zleceniodawca prowadzi poradnię kardiologiczną, chce usprawnić procesy odnoszące się do zapisu różnych informacji o pacjentach, ich wizytach, zaleceniach nadanych od lekarza prowadzącego.

Chce mieć również możliwość zarządzania poradnią poprzez zapis lekarzy w niej pracujących, opinię pacjentów na ich temat i odnośnie konkretnych wizyt, oraz system rezerwacji wizyt.

Wymagania funkcjonalne

Przechowywanie danych

1. Pacjenci
    a. Przechowywanie ***obowiązkowo*** danych osobowych, daty urodzenia, adresu korespondencji, numer telefonu komórkowego, adres e-mail, alergii.
2. Lekarze
    a. Przechowywanie **obowiązkowo** danych osobowych, adresu korespondencji, numeru telefonu komórkowego, adresu e-mail
3. Rezerwacje
    a. Przechowywanie **obowiązkowo** daty, powodu, lekarza odpowiedzialnego za wizytę oraz pacjenta oraz statusu rezerwacji
4. Wizyty
    a. Przechowywanie **obowiązkowo** pacjenta, lekarza oraz powód wizyty.
    b. Jeśli wizyta się odbyła, to dodatkowo lekarz ma możliwość przechowania zaleceń do dalszego leczenia.
5. Opinie
    a. Przechowujemy **obowiązkowo** ocenę od **1** do **5** oraz komentarz pacjenta
6. Dokumenty
    a. Przechowujemy **obowiązkowo** dokumenty w imieniu pacjenta takie jak, np. Ankiety.

Operacje

1. Chcemy aby kierownik miał podgląd na wszystkie dane w bazie danych,
2. Chcemy aby specjaliści ds. mediów społecznościowych posiadali możliwość:
    a. Wyszukiwania opinii na temat lekarzy poprzez konkretne frazy zawarte w komentarzach od klientów placówki
    b. Mieli podgląd do ocen wystawionych przez klientów dla określonych lekarzy
3. Chcemy aby osoby pracujące na recepcji, miały możliwość:
    a. Rejestracji pacjenta w placówce,
    b. Dostępu, edycji, anulowania rezerwacji,
    c. Dostępu do danych lekarzy,
    d. Dostępu do danych pacjentów ale bez danych związanych z dokumentacją medyczną,
    e. Rezerwowania wizyt w imieniu pacjentów,
    f. Dodawania różnorakich dokumentów w imieniu pacjenta.
4. Chcemy aby lekarze mieli możliwość:
    a. Dostępu oraz edycji danych pacjentów,
    b. Dostępu do danych lekarzy,
    c. Dostępu do rezerwacji wizyt do których lekarz jest przypisany,
    d. Dostępu do wszystkich wizyt pacjentów do których lekarz jest przypisany.
    e. Dodawania różnorakich dokumentów w imieniu pacjenta

Dodatkowe wymogi

1. Rezerwowanie wizyt
    a. Zarówno lekarz jak i pacjent nie może mieć dwóch wizyt w tej samej godzinie.
    b. Wizyty można rezerwować całodobowo.
    c. Recepcjonista może zmienić datę wizyty pacjentowi, ale data nie może być mniejsza niż aktualna data
    d. Recepcjonista może zmienić lekarza pacjentowi, jeśli lekarz jest w określonej dacie dostępny
2. Dodawanie dokumentów
    a. Recepcjonista oraz Lekarz mogą dodawać różne dokumenty w imieniu pacjenta
3. Osoby pracujące na recepcji, nie mogą posiadać dostępu do dokumentacji medycznej pacjenta, jaką są np. **Alergie**
4. Każdy z podmiotów będzie mógł edytować dane tylko i wyłącznie poprzez zdefiniowane procedury / widoki.

5. Każdy z użytkowników, ma posiadać bardzo ograniczone uprawnienia. Lista Użytkowników:
    a. Recepcja
    b. Kierownik
    c. Specjalista ds. Mediów Społecznościowych
    d. Lekarze – Brown, Higgins, Jones, Addams

Wymagania niefunkcjonalne

1. Wydajność zapytań
    a. Wprowadzenie odpowiednich indeksów w tabelach w celu przyspieszenia operacji przeszukiwania.
2. Atomowość
    a. Przeprowadzanie atomowych operacji na bazie danych, co pozwoli na zachowanie spójności danych.
3. Bezpieczeństwo danych**:**
    a. Ograniczenie dostępu do tabel poprzez określonych użytkowników, nadanie uprawnień czy przypisanie użytkowników do określonych schematów, z których będą mieli ograniczone uprawnienia.
4. Kopie zapasowe i odzyskiwanie**:**
    a. Regularne planowanie kopii zapasowych bazy danych oraz strategii odzyskiwania danych w przypadku awarii.
5. Logowanie operacji**:**
    a. Mechanizmy logowania operacji, audyty w (np. rejestrowanie zmian w danych pacjentów i wizyt).
6. Spójność danych**:**
    a. Zapewnienie integralności danych poprzez klucze główne / obce

2.

Implementacja

Schemat bazy danych

Architektura bazy danych

1. Model danych:

Baza danych została zaprojektowana w modelu hybrydowym (relacyjny + dokumentowy).
Większość danych jest przechowywana w modelu relacyjnym (pacjenci, lekarze, rezerwacje, wizyty, opinie, dokumenty).
Alergie w tabeli pacjenci, dodatkowe informacje w tabeli wizyt, zalecenia lekarza w tabeli wizyty oraz komentarze w tabeli opinie są przechowywane jako dane w formacie JSON w kolumnie alergie w tabeli pacjenci.
Dokumenty są przechowywanie w formie binarnej.

2. Struktura tabel:

```
CREATE TABLE placowka.pacjenci (
    id INT PRIMARY KEY IDENTITY(1,1),
    pesel NVARCHAR(12) UNIQUE NOT NULL,
    imie NVARCHAR(255) NOT NULL,
    nazwisko NVARCHAR(255) NOT NULL,
    data_urodzenia DATE NOT NULL,
    adres NVARCHAR(255) NOT NULL,
    numer_telefonu NVARCHAR(32) NOT NULL,
    email NVARCHAR(255) NOT NULL,
    alergie NVARCHAR(MAX),
);
```

```sql
CREATE TABLE placowka.lekarze (
    id INT PRIMARY KEY IDENTITY(1,1),
    imie NVARCHAR(255) NOT NULL,
    nazwisko NVARCHAR(255) NOT NULL,
    adres NVARCHAR(255) NOT NULL,
    numer_telefonu NVARCHAR(32),
    email NVARCHAR(255) NOT NULL
);

CREATE TABLE placowka.dokumenty (
    id INT PRIMARY KEY IDENTITY(1,1),
    id_pacjenta INT REFERENCES placowka.pacjenci(id),
    tresc VARBINARY(MAX), -- binarne dane
);

CREATE TABLE placowka.wizyty (
    id INT PRIMARY KEY IDENTITY(1,1),
    id_pacjenta INT REFERENCES placowka.pacjenci(id),
    id_rezerwacji INT REFERENCES placowka.rezerwacje(id),
    zalecenia_lekarza NVARCHAR(MAX) DEFAULT '{"zalecenia":[]}', -- JSON
    dodatkowe_informacje NVARCHAR(MAX) DEFAULT '{"info":[]}'-- JSON
);
```

```sql
CREATE TABLE placowka.wizyty (
    id INT PRIMARY KEY IDENTITY(1,1),
    id_pacjenta INT REFERENCES placowka.pacjenci(id),
    id_rezerwacji INT REFERENCES placowka.rezerwacje(id),
    zalecenia_lekarza NVARCHAR(MAX) DEFAULT '{"zalecenia":[]}', -- JSON
    dodatkowe_informacje NVARCHAR(MAX) DEFAULT '{"info":[]}'-- JSON
);

CREATE TABLE placowka.opinie (
    id INT IDENTITY(1,1),
    id_pacjenta INT REFERENCES placowka.pacjenci(id),
    id_lekarza INT REFERENCES placowka.lekarze(id),
    ocena INT CHECK (ocena BETWEEN 1 AND 5),
    komentarz NVARCHAR(MAX), -- JSON
    CONSTRAINT PK_opinie PRIMARY KEY (id),
);
```

3. Opis schematów w bazie danych:

Baza danych składa się z kilku kluczowych schematów i tabel:

- **Schemat placowka:** Przechowuje główne tabele związane z pacjentami, lekarzami, dokumentami, wizytami, opiniami oraz rezerwacjami.
    - Tabela pacjenci zawiera szczegóły o pacjentach, w tym np. ich dane kontaktowe, PESEL, oraz alergie (kolumna JSON).
    - Tabela lekarze przechowuje dane kontaktowe lekarzy.
    - Tabela dokumenty przechowuje dokumenty pacjentów w postaci binarnej.
    - Tabela rezerwacje zawiera informacje o zaplanowanych wizytach pacjentów u lekarzy.
    - Tabela wizyty przechowuje szczegóły wizyt, w tym zalecenia lekarzy.
    - Tabela opinie zawiera oceny i komentarze pacjentów na temat lekarzy.
    - Schemat ten, oraz wszelkie procedury w nim zawarte są wykorzystywane przez więcej niż 1 podmiot.
- **Schemat recepcja:** Przechowuje procedury i widoki związane z operacjami recepcji, takie jak rejestracja pacjentów czy zarządzanie wizytami.
    - Schemat ten wykorzystywany jest przez podmioty pracujące na recepcji
- **Schemat lekarz:** Zawiera procedury do zarządzania danymi medycznymi, w tym dodawanie/aktualizowanie alergii pacjentów oraz wprowadzanie dodatkowych informacji medycznych.

4. Triggery:

Na operacje dodawanie nowego lekarza został zaimplementowany trigger, na podstawie którego dodajemy id_lekarza oraz nazwisko które został dodane do bazy danych do tabeli pomocniczej, dzięki której widoki w schemacie lekarza pobierają dane tylko i wyłącznie lekarza (który jest zalogowany) który wykonuje zapytania

5. Indeksowanie:

Aby zoptymalizować często wykonywane zapytania, zaprojektowano następujące indeksy:

- **Indeks wierszowy:**
    - **IX_rezerwacje_data_wizyty** na kolumnie data_wizyty w tabeli placowka.rezerwacje. Indeks ten przyspiesza zapytania dotyczące dostępności lekarzy w danym terminie.
- **Indeks pełnotekstowy:**
    - Indeks pełnotekstowy na kolumnie komentarz w tabeli placowka.opinie, co umożliwia szybkie przeszukiwanie opinii pacjentów za pomocą funkcji CONTAINS w procedurze **WyszukajOpinie**.

6. Procedury / widoki:

Do określonych schematów opisanych wyżej zostały zaimplementowane określone widoki oraz procedury, które enkapsulują logikę biznesową oraz pola do których określone podmioty nie powinny mieć dostępu.

Lista procedur:

1. W schemacie **placowka**
    a. *WyszukajOpinie – odpowiada za wyszukanie opinii na podstawie frazy w komentarzu przy użyciu indeksu pełno-tekstowego.*
    b. *DodajDokument – odpowiada za dodawanie różnorakich dokumentów w formie binarnej dla określonego pacjenta*
        i. *Walidacja:*
            1. *Jeśli pacjent nie istnieje, błąd*
2. *W schemacie* recepcja
    a. ZarejestrujPacjenta – odpowiada za rejestrację pacjenta
        i. Walidacja:
            1. Jeśli pacjent już istnieje, błąd
    b. ZaktualizujDanePacjenta
        i. Walidacja:
            1. Jeśli pacjent nie istnieje, błąd
    c. RezerwujWizyte
        i. Walidacja:
            1. Jeśli lekarz dla którego rezerwujemy wizytę nie jest dostępny (ma wizytę w tej dacie ze statusem *0 (**ZAPLANOWANE**), błąd*
            2. *Jeśli pacjent który rezerwuję wizytę nie jest dostępny (*ma wizytę w tej dacie ze statusem 0 (**ZAPLANOWANE**), błąd
    d. ZaktualizujDateRezerwacji
        i. Walidacja:
            1. Jeśli pacjent dla którego chcemy zaktualizować rezerwację nie istnieje, błąd

2. Jeśli rezerwacja nie istnieje dla tego pacjenta (ma inny status niż zaplanowana), błąd
3. Jeśli pacjent rezerwujący wizytę ma już zaplanowaną wizytę w dacie na którą chce przełożyć, błąd
4. Jeśli istnieje już inna rezerwacja do tego lekarza w dacie w której pacjent chce zmienić rezerwację, błąd

e. PrzypiszInnegoLekarzaDoWizyty
   i. Walidacja:
      1. Jeśli pacjent dla którego chcemy zaktualizować rezerwację nie istnieje błąd
      2. Jeśli rezerwacja nie istnieje dla tego pacjenta (ma inny status niż zaplanowana), błąd
      3. Jeśli lekarz do którego pacjent chce się zapisać nie jest wtedy dostępny, błąd

f. AnulujRezerwacje
   i. Walidacja:
      1. Jeśli pacjent o podanym peselu nie istnieje, błąd
      2. Jeśli pacjent nie ma rezerwacji w podanej dacie, błąd

3. W schemacie *lekarz*

a. ZarejestrujNowaAlergiePacjentowi
   1. Jeśli pacjent nie ma takiej alergii, dodajemy ją
   2. Jeśli pacjent ma alergię, nic nie rób

b. UsunWyleczonaAlergiePacjentowi
   1. Usuwa alergie pacjentowi

c. DodajDodatkoweInformacjeDoWizyty
   1. Jeśli pacjent ma już dodatkowe informacje, dodajemy kolejne
   2. Jeśli nie dodajemy bezpośrednio

d. DodajZaleceniaDoWizyty
   1. Kazde dodanie zalecenia nadpisuje poprzednie

e. ZakonczWizyte
   i. Walidacja:
      1. Jeśli nie ma rezerwacji o podanym ID błąd

Lista widoków:

1. *W schemacie recepcja*
   a. *Pacjenci – enkapsuluje pole Alergie*
2. *W schemacie lekarz*
   a. *MojeWizyty -- wyświetla tylko wizyty lekarza pobierającego dane,*
   b. *MojeRezerwacje -- wyświetla tyko rezerwacje lekarza pobierającego dane*

3.

Skrypty SQL

1. Skrypty zawierające DDL są przechowywane w pliku ddl.sql

4.

Uprawnienia w bazie danych

1. Każdy z użytkowników bazy ma przypisany swój login na SQL serwer, oraz ma bardzo restrykcyjne uprawnienia odczytywania / zapisywania danych w bazie danych. Operacje zapisywania / aktualizowania danych są głównie uzewnętrzniane do schematów poprzez procedury. Uprawnienia zostały przypisane na podstawie "kontekstu". Przy niektórych kontekstach, zostały stworzone konkretne schematy w bazie do którego przypisane są widoki / procedury, w celu enkapsulacji logiki biznesowej czy też w celu ukrycia określonych pól, jak np. pola Alergie w tabeli pacjenci gdy to recepcjonista pobiera jego dane.
2. Wyróżniamy 4 konteksty:
    a. Kierownik,
    b. Lekarz,
    c. Recepcjonista,
    d. Specjalista DS. Mediów społecznościowych.

Kierownik:

1. Zyskuje uprawnienia **SELECT** do schematu *placowka*, czyli do wszystkich tabel zawartej w naszej bazie danych oraz **EXEC** do procedury *WyszukajOpinie*, w której to wykorzystujemy funkcje która korzysta z indeksu pełno-tekstowego do wyszukiwania opinii

Specjalista DS. Mediów społecznościowych:

1. Zyskuje uprawnienia **SELECT** tylko do tabeli *opinie*, oraz **EXEC** do procedury *WyszukajOpinie,* w której to wykorzystujemy funkcje która korzysta z indeksu pełno-tekstowego do wyszukiwania opinii

Recepcja:

1. Osoby pracujące na recepcji zyskują uprawnienia **EXEC** do schematu *recepcja.*

2. W schemacie *recepcja,* zainicjalizowany jest widok do tabeli *pacjenci* dzięki któremu recepcjonista ma dostęp do pacjentów placówki, ale nie ma dostępu do pola *alergie*, przy pobieraniu danych.
3. W schemacie *recepcja,* zainicjalizowane są procedury które odpowiadają rejestrowaniu pacjentów, aktualizacji ich danych oraz rejestrowaniu wizyt, i ich aktualizacji.
4. Oprócz widoków / procedur które enkapsulują logikę biznesową, czy też ukrywają pola z wrażliwymi danymi, recepcjonista otrzymuje uprawnienia:
    a. **SELECT** do placowka.rezerwacje
    b. **EXEC** do placowka.DodajDokument
    c. **SELECT** do placowka.lekarze

Lekarz:

1. Osoby pracujące na stanowisku lekarza, zyskują uprawnienia **EXEC** do schematu lekarz
2. W schemacie *lekarz* zainicjalizowane są procedury odpowiadające za edycje danych w tabeli *rezerwacje*, *pacjenci*, *wizyty*
3. W schemacie lekarz zainicjalizowane się widoki, dzięki którym lekarz pobiera dane tylko i wyłącznie dla swoich rezerwacji / wizyt (Jest to możliwe dzięki tabeli pomocniczej, która mapuje ID lekarza do nazwy aktualnie zalogowanego użytkownika).

5.

1. Skrypty z przykładowymi poleceniami: testy.sql

2. Skrypty z dodawaniem danych: seed.sql

6.

Procedury utrzymania bazy danych

1. **Kontrola integralności bazy danych**
    a. Harmonogram: Codziennie o 4 rano
    b. Skrypt: integralnosc_bazy_job.sql
2. **Tworzenie kopii zapasowej bazy danych**
    a. Harmonogram: Codziennie o 2 rano
    b. Skrypt: backup_job.sql
3. **Aktualizacja statystyk**
    a. Harmonogram: Codziennie o 5 rano
    b. Skrypt: aktualizacja_statystyk_job.sql