DOKUMENTACJA

1. Spis użytych technologii

Użyte technologie w naszym projekcie to:

- MariaDB
- Python
- Matplotlib
- Faker - moduł w python do generowania losowych danych
- ReportLab
- Visual Studio Code
- Biblioteka mariadb

2. Lista plików i opis ich zawartości

- *schemat_bazy.sql* - Plik SQL zawierający wszystkie instrukcje **CREATE TABLE** definiujące strukturę bazy danych (tabele, klucze główne i obce).
- *klienci(1).py* - Plik Python wypełniający tabele klienci losując płeć i dane osobowe przy użyciu modułu Faker. Jest ustawiony na 183 klientów.
- *pracownicy(2).py* - Plik Python wypełniający tabele pracownicy losując dane osobowe, zarobki i datę zatrudnienia z zakresu działania firmy i pasującą do później odbytych wycieczek
- *oferta(3).sql* - Plik SQL wypełniający tabele oferta, każda posiada cenę z jedną osobę, koszt organizacji, miejsce pobytu, zapewnione atrakcje oraz krótki opis
- *promocje(4).sql* - Plik SQL wypełniający tabele promocje 5 różnymi promocjami z kodami zniżkowymi i wartością procentową zniżki
- *zrealizowane_wycieczki(5).sql* - Plik SQL wypełniający tabele zrealizowane_wycieczki poprzez wpisanie 17 zrealizowanych wycieczek wraz z datą i osobą która opiekowała się grupą
- *transakcje(6).sql* - Plik SQL wypełniający tabele transakcje, każdy klient płaci indywidualnie za siebie a płatności następują na miejscu zbiórki w dniu wyjazdu. Do każdej płatności przypisany jest też sprzedawca i może być też naliczona promocja.
- *opinie(7).sql* - Plik SQL wypełniający tabele opinie, opinii przypisana jest wycieczka a przez to także opiekun i każdy z tych aspektów jest oceniany w skali 1-5.
- raport.py - Plik Python, wykonuje analizę bazy danych i drukuje raport.

3. Kolejność i sposób uruchamiania plików, aby uzyskać gotowy projekt

3.1. Stworzenie schematu bazy danych

- Uruchom plik *schemat_bazy.sql* (katalog: “Część 1 - projekt i utworzenie schematu”) w środowisku MariaDB.

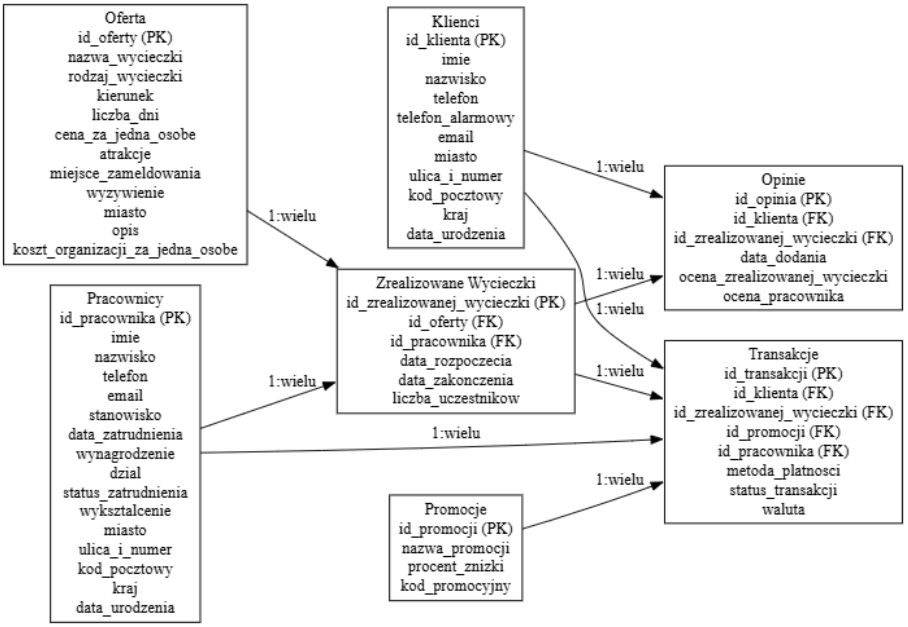
3.2. Wstawienie danych do bazy

- Uruchom pliki z folderu “Część 2 - skryptowe wypełnianie bazy danych” w środowisku Python gdy jest to plik z rozszerzeniem .py lub przez MariaDB SQL gdy to plik z rozszerzeniem .sql. Każdy plik na końcu tytułu ma liczbę i według ich kolejności należy te pliki uruchomić.

3.3. Wygenerowanie raportu

- Uruchom skrypt raport.py.

4. Schemat projektu bazy danych POPRAWIONE



5. Dla każdej relacji listę zależności funkcyjnych z wyjaśnieniem

5.1. Tabela: pracownicy

- **Atrybuty:**
  - id_pracownika, imie, nazwisko, telefon, email, stanowisko, data_zatrudnienia, wynagrodzenie, dzial, status_zatrudnienia, wyksztalcenie, miasto, ulica_i_numer, kod_pocztowy, kraj, data_urodzenia.
- **Zależności funkcyjne:**
  - id_pracownika → wszystkie inne atrybuty
  - email → id_pracownika

5.2. Tabela: klienci

- **Atrybuty:**
  - id_klienta, imie, nazwisko, telefon, telefon_alarmowy, email, miasto, ulica_i_numer, kod_pocztowy, kraj, data_urodzenia.
- **Zależności funkcyjne:**
  - id_klienta → wszystkie inne atrybuty
  - email → id_klienta

5.3. Tabela: oferta

- **Atrybuty:**
  - id_oferty, nazwa_wycieczki, rodzaj_wycieczki, kierunek, liczba_dni, cena_za_jedna_osobe, atrakcje, miejsce_zameldowania, wyzywienie, miasto, opis, koszt_organizacji_za_jedna_osobe.
- **Zależności funkcyjne:**
  - id_oferty → wszystkie inne atrybuty

5.4. Tabela: zrealizowane_wycieczki

- **Atrybuty:**
  - id_zrealizowanej_wycieczki, id_oferty, id_pracownika, data_rozpoczecia, data_zakonczenia, liczba_uczestnikow.
- **Zależności funkcyjne:**
  - id_zrealizowanej_wycieczki → wszystkie inne atrybuty

5.5. Tabela: opinie

- **Atrybuty:**
  - id_opinia, id_klienta, id_zrealizowanej_wycieczki, data_dodania, ocena_zrealizowanej_wycieczki, ocena_pracownika.
- **Zależności funkcyjne:**
  - id_opinia → wszystkie inne atrybuty

5.6. Tabela: promocje

- **Atrybuty:**
  - id_promocji, nazwa_promocji, procent_znizki, kod_promocyjny.
- **Zależności funkcyjne:**
  - id_promocji → wszystkie inne atrybuty
  - kod_promocyjny → id_promocji

5.7. Tabela: transakcje

- **Atrybuty:**
  - id_transakcji, id_klienta, id_zrealizowanej_wycieczki, metoda_platnosci, status_transakcji, id_promocji, waluta, id_pracownika.
- **Zależności funkcyjne:**
  - id_transakcji → wszystkie inne atrybuty

6. **Uzasadnienie, że baza jest w EKNF** (Efektywnej Kluczowej Normalnej Formie) **POPRAWIONE**

EKNF zakłada, że baza danych spełnia następujące kryteria:

- **1NF:** Relacja jest w pierwszej postaci normalnej, jeśli każda krotka *r* należy do iloczynu kartezjańskiego $T_1 \times T_2 \times \cdots \times T_k$.
  - Oznacza to, że każda kolumna zawiera pojedyncze wartości (atomowość), a dane w kolumnach mają jednolity typ.
- **2NF:** Relacja jest w 2NF, jeśli spełnia wymagania 1NF oraz każda nietrywialna zależność funkcyjna:
  - Nie zaczyna się od podzbioru właściwego nadklucza,
  - Kończy się na atrybucie głównym.
- **3NF:** Relacja jest w 3NF, jeśli spełnia wymagania 2NF oraz każda nietrywialna zależność funkcyjna:
  - Zaczyna się od nadklucza albo kończy się na atrybucie głównym.
- **Postać Normalna Boyce'a-Codda (BCNF)**: Relacja jest w BCNF, jeśli każda nietrywialna zależność funkcyjna zaczyna się od nadklucza.
  - Oznacza to, że jedynym źródłem zależności funkcyjnych jest klucz główny.

Analiza tabel w bazie danych:

6.1. **Tabela: pracownicy - POPRAWIONE**

- **Pierwsza Postać Normalna (1NF):**
  - Każda kolumna (np. imie, nazwisko, telefon) zawiera pojedyncze wartości, np. imię jednego pracownika czy jeden numer telefonu.
  - Typy danych w kolumnach są spójne: np. imie i nazwisko są typu VARCHAR, a data_zatrudnienia typu DATE.
  - Klucz główny id_pracownika jednoznacznie identyfikuje każdą krotkę w tabeli.

Wniosek: Tabela spełnia wymagania 1NF.

- **Druga Postać Normalna (2NF):**
  - Klucz główny id_pracownika jest prosty (niezłożony), więc nie ma fragmentów klucza głównego.
  - Wszystkie kolumny (np. imie, nazwisko, telefon, stanowisko) są w pełni zależne od id_pracownika.

- Nie istnieją częściowe zależności.

Wniosek: Tabela spełnia wymagania 2NF.

- **Trzecia Postać Normalna (3NF):**
  - Każda kolumna jest bezpośrednio zależna od klucza głównego id_pracownika. Na przykład: telefon, email, ulica_i_numer, kod_pocztowy są bezpośrednio związane z pracownikiem.
  - Nie ma sytuacji, w której jeden atrybut niekluczowy zależy od innego.

Wniosek: Tabela spełnia wymagania 3NF.

- **Postać Normalna Boyce'a-Codda (BCNF):**
  - Wszystkie zależności funkcyjne zaczynają się od klucza głównego id_pracownika.
  - Przykłady zależności: id_pracownika → imie, nazwisko, telefon, email, kod_pocztowy.
  - Nie ma innych zależności funkcyjnych, które pochodziłyby od innych kolumn.

Wniosek: Tabela spełnia wymagania BCNF.

- **Eksportowana Kluczowa Normalna Forma (EKNF):**
  - W tabeli nie występują atrybuty nadmiarowe ani żadne nietrywialne zależności funkcyjne poza tymi, które zaczynają się od klucza głównego.
  - Wszystkie atrybuty w tabeli (np. miasto, stanowisko, telefon) są logicznie przypisane do konkretnego pracownika i wynikają z id_pracownika.

Wniosek: Tabela spełnia wymagania EKNF.

<u>Podsumowanie:</u>

Tabela **pracownicy** spełnia wymagania każdej z przedstawionych postaci normalnych:

1. **1NF:** Dane są atomowe i mają spójne typy.
2. **2NF:** Wszystkie kolumny są w pełni zależne od klucza głównego id_pracownika.
3. **3NF:** Nie ma tranzytywnych zależności między atrybutami niekluczowymi.
4. **BCNF:** Wszystkie zależności funkcyjne zaczynają się od nadklucza.
5. **EKNF:** Dane są zorganizowane bez nadmiarowych zależności, a każdy atrybut zależy wyłącznie od klucza głównego.

Tabela pracownicy jest więc w Eksportowanej Kluczowej Normalnej Formie (EKNF).

6.2. Tabela: klienci

- Klucz główny: id_klienta.
- Atrybuty opisują szczegóły klienta, brak redundancji danych.
- Unikalne klucze (email) nie naruszają założeń postaci normalnych.
- Nie występują zależności przechodnie ani częściowe.
- Wszystkie determinanty są kluczami kandydującymi.

Tabela spełnia wymagania **EKNF**.

6.3. Tabela: oferta

- Klucz główny: id_oferty.
- Atrybuty opisują szczegóły oferty, nie ma redundancji.
- Wszystkie inne atrybuty są zależne tylko od klucza głównego (id_oferty).
- Nie ma zależności przechodnich ani częściowych.
- Wszystkie determinanty są kluczami kandydującymi.

Tabela spełnia wymagania **EKNF**.

6.4. Tabela: zrealizowane_wycieczki

- Klucz główny: id_zrealizowanej_wycieczki.
- Obce klucze id_oferty i id_pracownika są zależne od klucza głównego.
- Brak redundancji danych.
- Wszystkie determinanty są kluczami kandydującymi.

Tabela spełnia wymagania **EKNF**.

6.5. Tabela: opinie

- Klucz główny: id_opinia.
- Obce klucze (id_klienta, id_zrealizowanej_wycieczki) są zależne tylko od klucza głównego.
- Brak zależności przechodnich.
- Wszystkie determinanty są kluczami kandydującymi.

Tabela spełnia wymagania **EKNF**.

6.6. Tabela: promocje

- Klucz główny: id_promocji.
- Atrybuty nazwa_promocji, procent_znizki i kod_promocyjny są zależne wyłącznie od klucza głównego.
- kod_promocyjny jest unikalny, ale nie powoduje redundancji ani zależności przechodnich.
- Wszystkie determinanty są kluczami kandydującymi.

Tabela spełnia wymagania **EKNF**.

6.7. Tabela: transakcje

- Klucz główny: id_transakcji.
- Obce klucze (id_klienta, id_zrealizowanej_wycieczki, id_promocji, id_pracownika) są w pełni zależne od klucza głównego.
- Brak zależności przechodnich.
- Wszystkie determinanty są kluczami kandydującymi.

Tabela spełnia wymagania **EKNF**.

Wniosek:

Cała baza danych spełnia wymagania **EKNF**.

7. Opis, co było najtrudniejsze podczas realizacji projektu

- Obsługa polskich znaków w pliku PDF - Konieczność rejestracji czcionki obsługującej znaki diakrytyczne (np. ą, ę) w ReportLab.
- Drukowanie raportu tak aby wszystko było spójne i się “nie rozjeżdżało”
- Rysowanie wykresów aby zwizualizować analizę bazy
- Generowanie losowych danych do bazy przez GUS okazało się zbyt czasochłonne i prostszym rozwiązaniem okazał się moduł Faker
- Skonstruowanie schematu bazy, który umożliwia zautomatyzowanie wypełniania jej
- Wypełnianie tabeli transakcje indeksami klientów, którzy widnieją w bazie, a liczba transakcji sumuje się do liczby klientów na wszystkich wycieczkach