Dokumentacja - projekt

Olga Foriasz, Tomasz Warzecha, Zuzanna Klaman, Maria Borowiak, Mateusz Martynów

Spis treści

1 Spis użytych technologii

Do wykonania projektu użyte zostały następujące techonolgie wraz z bibliotekami:

- Python
  - sqlalchemy
  - sqlalchemy.ext.declarative
  - sqlalchemy.orm
  - faker
  - datetime
  - random
- SQL
- Pakiet R (R Markdown)
  - RMariaDB
  - DBI
  - kableExtra
  - ggplot2

Oraz narzędzia do zarządzania bazą danych:

- ERD Editor
- HeidiSQL

2 Spis plików i ich zawartość

1. **treść_pytań.txt** - plik tekstowy zawierający pytania, na podstawie których przeprowadzona została analiza danych w bazie
2. **folder zapytania sql** - folder zawierający pliki z zapytaniami w języku SQL, dającymi odpowiedź na zadane pytania w pliku treść_pytań.txt:
   1. pytanie_1.sql
   2. pytanie_2.sql
   3. pytanie_3.sql
   4. pytanie_4.sql
   5. pytanie_5.sql
   6. pytanie_6.sql
   7. pytanie_7.sql
   8. pytanie_8.sql
3. **klienci.py** - plik w języku Python, generujący losowe dane do tabeli klienci
4. **pracownicy_zrealizowane.py** - plik w języku Python, generujący losowe dane do tabeli pracownicy oraz zrealizowane
5. **transakcje.py** - plik w języku Python, generujący losowe dane do tabeli transakcje_klienci, transakcje_firma
6. **rodzaje_kosztyOrganizacji** - plik w języku Python, generujący losowe dane do tabeli rodzaje oraz koszty_organizacji
7. **tabele.json.vuerd** - plik erd, generujący schemat bazy oraz pomagający w skryptowym wypełnieniu bazy

8. **raport.Rmd** - plik rmd zawierający raport w języku R
9. **Raport—projekt.html** - plik html zawierający wygenerowany raport z pliku raport.Rmd
10. **dane_do_bazy.txt** - plik zawierający dane do połączenia się z bazą
11. **znizki.py** - plik w języku Python, generujący dane do tabeli zniżki
12. **sql_do_stworzenia_bazy.txt** - plik tekstowy zawierający zapytania SQL potrzebne do stworzenia bazy

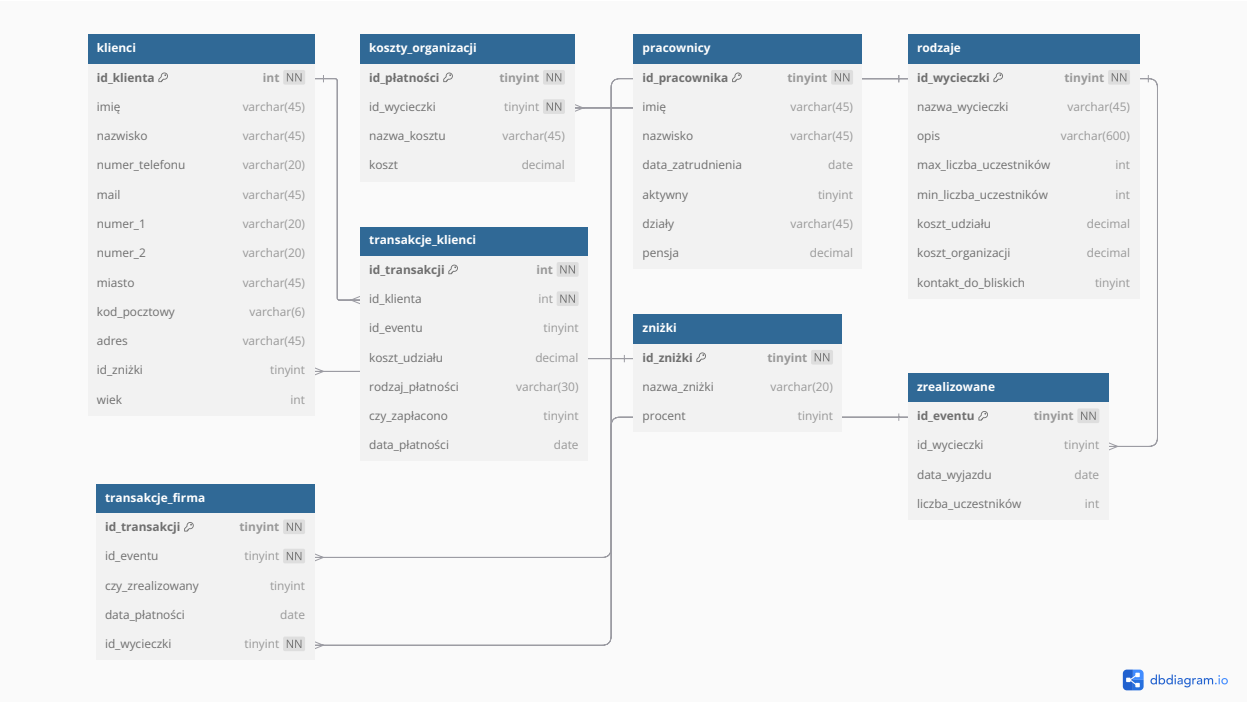
3 Kolejność oraz sposób uruchamiania plików

Aby otrzymać gotowy projekt należy uruchamiać kolejne pliki zgodnie z instrukcją:

1. **dane_do_bazy.txt** - należy otworzyć plik tekstowy oraz połączyć się z bazą przy użyciu danych umieszczonych w pliku
2. **treść_pytań.txt** - następnie zapoznać się z treścią zagadnień, do których zostanie przeprowadzona analiza danych
3. **raport.Rmd** - należy uruchomić plik, w którym napisany został raport oraz który zawiera analizę danych(kody napisane w SQL pochodzą z folderu zapytania sql, który został załączony do projektu)
4. **Raport—projekt.html** - ostatnim w kolejności do uruchomienia jest raport w pliku html, który jest wygenerowany za pomocą pliku raport.Rmd. Zawiera przeprowadzoną analizę danych wraz z wnioskami i wykresami. Stanowi podsumowanie pracy nad pojektem

- Jeśli chcielibyśmy wygenerować bazę od zera, w celu sprawdzenia poprawności tworzenia bazy, należy uruchomić polecenie SQL z pliku sql_do_stworzenia_bazy.txt. Następnie w podanej kolejności uruchomić pliki generujące dane: rodzaje_kosztyOrganizacji.py, pracownicy_zrealizowane.py, znizki.py, klienci.py, transakcje.py

4 Schemat bazy



5 Lista zależności funkcyjnych

Poniżej znajduje się lista zależności funkcyjnych wraz z uzasadnieniem:

- Relacje w poszczególnych tabelach:

  1. **Tabela klienci**:

id_klienta → {imię, nazwisko, numer_telefonu, mail, numer_1, numer_2, miasto, kod_pocztowy, adres, id_zniżki, wiek} id_klienta wciąż jednoznacznie identyfikuje wszystkie dane klienta

  2. **Tabela koszty_organizacji**:

id_płatności → {id_wycieczki, nazwa_kosztu, koszt} id_płatności jednoznacznie identyfikuje koszt organizacyjny

  3. **Tabela pracownicy**:

id_pracownika → {imię, nazwisko, data_zatrudnienia, aktywny, działy, pensja} id_pracownika jednoznacznie identyfikuje pracownika

  4. **Tabela rodzaje**:

id_wycieczki → {nazwa_wycieczki, opis, max_liczba_uczestników, min_liczba_uczestników, koszt_udziału, koszt_organizacji, kontakt_do_bliskich} id_wycieczki jednoznacznie identyfikuje rodzaj wycieczki

  5. **Tabela transakcje_firma**:

id_transakcji → {id_eventu, czy_zrealizowany, data_płatności, id_wycieczki} id_transakcji jednoznacznie identyfikuje transakcje firmy

6. **Tabela transakcje_klienci**:

id_transakcji → {id_klienta, id_eventu, koszt_udziału, rodzaj_płatności, czy_zapłacono, data_płatności} id_transakcji jednoznacznie identyfikuje transakcję firmy

7. **Tabela zniżki**:

id_zniżki → {nazwa_zniżki, procent} id_zniżki jednoznacznie identyfikuje zniżkę

8. **Tabela zrealizowane**:

id_eventu → {id_wycieczki, data_wyjazdu, liczba_uczestników} id_eventu jednoznacznie identyfikuje dane dotyczące zrealizowanych wyjazdów

- Relacje między tabelami:

1. **klienci.id_klienta → transakcje_klienci.id_klienta**:

typ relacji: **jeden do wielu** : Ponieważ jeden klient może mieć wiele transakcji

2. **klienci.id_zniżki → zniżki.id_zniżki**:

typ relacji: **wiele do jednego** : Ponieważ wielu klientów może mieć tę samą zniżkę

3. **transakcje_firma.id_eventu → zreazlizowane.id_eventu**:

typ relacji: **wiele do jednego**: Ponieważ wiele transakcji może dotyczyć jednego eventu

4. **transakcje_firma.id_wycieczki → rodzaje.id_wycieczki**:

typ relacji: **wiele do jednego**: Ponieważ może być wiele transakcji do jednej wycieczki

5. **zrealizowane.id_wycieczki → rodzaje.id_wycieczki**: typ relacji: **wiele do jednego**: Ponieważ wiele razy mógł zostać zrealizowany jeden rodzaj wycieczki
6. **koszty_organizacji.id_wycieczki → rodzaje.id_wycieczki**: typ relacji: **wiele do jednego**: Ponieważ wiele kosztów organizacji może dotyczyć jednoego rodzaju wycieczki

6 Uzasadnienie, że baza spełnia wymogi EKNF

Baza jest:

1. W **pierwszej postaci normalnej (1NF)**, ponieważ wartości atrybutów są atomowe(niepodzielne), żadna z tabel nie zawiera powtarzających się informacji oraz kolejność wierszy, bądź kolumn nie ma znaczenia dla danych. Ponadto, w każdej kolumnie występuje jeden typ danych. Tabele mają zdefiniowane klucze, a także elemnty niebędące kluczami głównymi są od nich w zależności funkcyjnej.
2. W **drugiej postaci normalnej (2NF)**, co można uargumentować, ponieważ spełnia kryteria 1NF, a ponadto tabele nie posiadają zależności częściowych.
3. W **trzeciej postaci normalnej (3NF)** - spełnia założenia 2Nf, a co więcej atrybuty niekluczowe zależą bezpośrednio od klucza głównego
4. **EKNF**, ponieważ spełnia 3NF, a także każda kolumna(będąca kluczem), od której zależą inne dane, jednocześnie determinuje cały wiersz.

7 Najtrudniejsze etapy realizacji projektu

1. Jednym z najtrudniejszych etapów realizacji projektu okazało się skryptowe wypełnienie bazy. Ze względu na wynikające problemy i potrzeby zmiany bazy, na bieżąco trzeba było aktualizować dane. Dodatkowo usuwanie poszczególnych kolumn, wymuszało ponowne wypełnienie bazy. Etap sam w sobie był czasochłonny i wymagał dużego nakładu pracy, szczególnie przy wielkokrotnym powtarzaniu czynności.

2. Drugim trudnym etapem realizacji była normalizacja bazy danych, ze względu na płynność oraz złożoność zagadnienia, zajęło to więcej czasu niż było planowane. W tym etapie mieliśmy także trudność w stwierdzeniu czy baza w 100 procentach spełnia wymogi EKNF.