

Jan Lewandowski, Szymon Jończak-Lis,
Cezary Czetyrba, Michał Mickiewicz

Dokumentacja projektu z baz danych

20.01.2025

1. Spis użytych technologii

- Python
 - wersja ≥ 3.9
 - moduły:
 - pandas
 - mysql-connector-python
 - python-dotenv
- R
 - wersja $\geq 4.4.2$
 - pakiety:
 - RMariaDB
 - ggplot2
 - ggrepel
 - lattice
 - dplyr
 - tidyverse
 - ggpubr
- RStudio
 - wersja \geq
2024.12.0+467
- Visual Studio Code
 - Dodatki
 - ERD editor
 - Inline SQL
- Git, GitHub
- Git Bash w systemie Windows

2. Struktura projektu

```
/
├── .github/
├── dane_statystyczne/
│   ├── Imiona_nadane_dzieciom_w_Polsce_w_*.csv
│   ├── nazwiska_meskie.csv
│   └── nazwiska_zenskie.csv
├── dokumentacja/
│   ├── dokumentacja.pdf
│   ├── dokumentacja.tex
│   └── schemat_bazy_danych.vuerd
├── obrazy/
│   ├── baza_danych.fullsize.png
│   └── baza_danych.png
├── raport/
│   ├── raport.html
│   ├── raport.pdf
│   └── raport.Rmd
├── wypelnienie/
│   ├── custom_util.py
│   ├── do_wypelnienia.py
│   ├── sql.py
│   └── wypelnienie.py
├── .env.example
├── .gitignore
├── LICENSE
├── README.md
└── requirements.txt
```

Projekt podzielony jest na główne foldery i pliki:

- **.github**
 - Folder zawierający pliki konfiguracyjne dla GitHub Actions. W naszym przypadku jest to analiza stylistyczna kodu.
- **dane_statystyczne**
 - Folder zawierający pliki z danymi statystycznymi. Są to pliki CSV z imionami i nazwiskami.
 - Źródła:
 - [Nazwiska w rejestrze PESEL](#)
 - [Imiona nadane dzieciom w Polsce](#)
- **dokumentacja**
 - Folder zawierający pliki z dokumentacją projektu. Znajduje się tam plik `dokumentacja.pdf`, który jest wygenerowany z pliku `dokumentacja.tex` oraz `schemat_bazy_danych.vuerd` zawierający schemat bazy danych w formacie ERD Editor.
- **obrazy**

- Folder zawierający obrazy wykorzystane w raporcie oraz dokumentacji.
- **raport**
 - Folder zawierający pliki z raportem projektu. Znajdują się tam pliki `raport.pdf` oraz `raport.html`, które są generowane z pliku `raport.Rmd`.
- **wypelnienie**
 - Folder zawierający skrypty do wypełnienia bazy danych. Znajduje się tam główny plik `wypelnienie.py`, który wypełnia bazę danych.
 - `custom_util.py` - plik z funkcjami pomocniczymi.
 - `do_wypelnienia.py` - plik z danymi na temat wycieczek i pracowników.
 - `sql.py` - plik z zapytaniami SQL.
- **.env.example**
 - Przykładowy plik środowiskowy.
- **.gitignore**
 - Lista plików ignorowanych przez Git.
- **LICENSE**
 - Licencja projektu.
- **README.md**
 - Ogólny opis projektu.

3. Instrukcja uruchomienia

Aby uruchomić projekt potrzebny jest Python w wersji 3.9 lub nowszej. Po sklonowaniu repozytorium należy zainstalować wymagane moduły Pythona:

```
1 cd projekt-bazy-danych
2 pip install -r requirements.txt
```

Następnie należy skopiować plik `.env.example` i zmienić jego nazwę na `.env`. W pliku tym należy podać hasło do połączenia z bazą danych.

```
1 cp .env.example .env
2 vim .env
```

Po zainstalowaniu modułów i uzupełnieniu pliku `.env` można uruchomić skrypt wypełniający bazę danych.

```
1 python wypelnienie/wypelnienie.py
```

Kiedy zakończy się on poprawnie wypisując Wypelniono baze danych, możemy wygenerować raport. Można go otworzyć w RStudio i uruchomić przyciskiem Knit lub mając Rscript i pandoc poprzez umieszczenie

```
export PATH=$PATH:"C:\Program Files\R\R-4.4.2\bin\x64"
export PATH=$PATH:"C:\Program Files\RStudio\resources\app
\bin\quarto\bin\tools"
```

```
Rscript -e 'rmarkdown::render(input="raport/raport.Rmd", output_format="all")'
```

Teraz możemy obejrzeć gotowy raport, znajduje się on w pliku `raport/raport.pdf` w formacie PDF oraz `raport/raport.html` w formacie HTML.

4. Schemat bazy danych

Rysunek 1. Schemat bazy danych

5. Analiza tabel

5.1. Tabela `rodzaje_transportu`

Tabela jest postaci:

`rodzaje_transportu(id_rodzaju_transportu, nazwa, opis)`

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_rodzaju_transportu \rightarrow nazwa$$

$$id_rodzaju_transportu \rightarrow opis$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.2. Tabela `koszty_miasta`

Tabela jest postaci:

`koszty_miasta(id_kosztu_miasta, koszt, cena_dla_klienta, id_miasta)`

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_kosztu_miasta \rightarrow koszt$$

$$id_kosztu_miasta \rightarrow cena_dla_klienta$$

$$id_kosztu_miasta \rightarrow id_miasta$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.3. Tabela `miasta`

Tabela jest postaci:

`miasta(id_miasta, miasto, id_kraju)`

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_miasta \rightarrow miasto$$

$$id_miasta \rightarrow id_kraju$$

Zakładamy, że miasto nie definiuje jednoznacznie kraju. Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.4. Tabela `kraje`

Tabela jest postaci:

`kraje(id_kraju, kraj)`

Jedyną nietrywialną zależnością funkcyjną jest:

$$id_kraju \rightarrow kraj$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.5. Tabela transport_propozycja_wycieczki

Tabela jest postaci:

transport_propozycja_wycieczki(id_rodzaju_transportu, id_propozycji, id_kosztu_miasta)

W tej tabeli nie występują nietrywialne zależności funkcyjne, wszystkie trzy kolumny tworzą klucz elementarny, a pomiędzy nimi nie ma zależności. Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.6. Tabela adresy

Tabela jest postaci:

adresy(id_adresu, adres, adres2, id_miasta, kod_pocztowy)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_adresu \rightarrow adres$$

$$id_adresu \rightarrow adres2$$

$$id_adresu \rightarrow id_miasta$$

$$id_adresu \rightarrow kod_pocztowy$$

Zakładamy, że kod pocztowy nie definiuje jednoznacznie miasta i adres nie definiuje kodu pocztowego.

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.7. Tabela miejsca_wycieczki

Tabela jest postaci:

miejsca_wycieczki(id_miejsca, nazwa, id_adresu, koszt, cena_dla_klienta, id_kontrahenta)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_miejsca \rightarrow nazwa$$

$$id_miejsca \rightarrow id_adresu$$

$$id_miejsca \rightarrow koszt$$

$$id_miejsca \rightarrow cena_dla_klienta$$

$$id_miejsca \rightarrow id_kontrahenta$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.8. Tabela propozycja_koszt_u_kontrahenta

Tabela jest postaci:

propozycja_koszt_u_kontrahenta(id_propozycji, id_kosztu_u_kontrahenta)

W tej tabeli nie występują nietrywialne zależności funkcyjne, wszystkie trzy kolumny tworzą klucz elementarny, a pomiędzy nimi nie ma zależności. Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.9. Tabela koszty_u_kontrahenta

Tabela jest postaci:

koszty_u_kontrahenta(id_kosztu_u_kontrahenta, nazwa, koszt, cena_dla_klienta, id_kontrahenta)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_kosztu_u_kontrahenta \rightarrow nazwa$$

$$id_kosztu_u_kontrahenta \rightarrow koszt$$

$$id_kosztu_u_kontrahenta \rightarrow cena_dla_klienta$$

$$id_kosztu_u_kontrahenta \rightarrow id_kontrahenta$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.10. Tabela kontrahenci

Tabela jest postaci:

kontrahenci(id_kontrahenta, nazwa, opis, email, id_adresu)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_kontrahenta \rightarrow nazwa$$

$$id_kontrahenta \rightarrow opis$$

$$id_kontrahenta \rightarrow email$$

$$id_kontrahenta \rightarrow id_adresu$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.11. Tabela propozycje_wycieczki

Tabela jest postaci:

propozycje_wycieczki(id_propozycji, nazwa, opis, ograniczenia, min_liczba_osob, max_liczba_osob, nasze_koszty_razem, cena_dla_klienta, id_miejsca)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_propozycji \rightarrow nazwa$$

$$id_propozycji \rightarrow opis$$

$$id_propozycji \rightarrow ograniczenia$$

$$id_propozycji \rightarrow min_liczba_osob$$

$$id_propozycji \rightarrow max_liczba_osob$$

$$id_propozycji \rightarrow nasze_koszty_razem$$

$$id_propozycji \rightarrow cena_dla_klienta$$

$$id_propozycji \rightarrow id_miejsca$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.12. Tabela wycieczki

Tabela jest postaci:

wycieczki(id_wycieczki, czas_wyjazdu, czas_powrotu, liczba_osob, id_propozycji)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_wycieczki \rightarrow czas_wyjazdu$$

$$id_wycieczki \rightarrow czas_powrotu$$

$$id_wycieczki \rightarrow liczba_osob$$

$$id_wycieczki \rightarrow id_propozycji$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.13. Tabela transakcje_kontrahenci

Tabela jest postaci:

transakcje_kontrahenci(id_transakcji, kwota, data_transakcji, id_kontrahenta, id_wycieczki)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_transakcji \rightarrow kwota$$

$$id_transakcji \rightarrow czas_wyjazdu$$

$$id_transakcji \rightarrow data_transakcji$$

$$id_transakcji \rightarrow id_kontrahenta$$

$$id_transakcji \rightarrow id_wycieczki$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.14. Tabela rodzaje_uslug_dodatkowych

Tabela jest postaci:

rodzaje_uslug_dodatkowych(id_uslugi_dodatkowej, nazwa, opis_uslugi, koszt, cena_dla_klienta, id_kontrahenta)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_uslugi_dodatkowej \rightarrow nazwa$$

$$id_uslugi_dodatkowej \rightarrow opis_uslugi$$

$$id_uslugi_dodatkowej \rightarrow koszt$$

$$id_uslugi_dodatkowej \rightarrow cena_dla_klienta$$

$$id_uslugi_dodatkowej \rightarrow id_kontrahenta$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.15. Tabela uslugi_dodatkowe

Tabela jest postaci:

uslugi_dodatkowe(id_wycieczki, id_uslugi)

W tej tabeli nie występują nietrywialne zależności funkcyjne, dwie kolumny tworzą klucz elementarny, a pomiędzy nimi nie ma zależności. Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.16. Tabela pracownik_wycieczka

Tabela jest postaci:

pracownik_wycieczka(id_wycieczki, id_pracownika)

W tej tabeli nie występują nietrywialne zależności funkcyjne, dwie kolumny tworzą klucz elementarny, a pomiędzy nimi nie ma zależności. Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.17. Tabela pracownicy

Tabela jest postaci:

pracownicy(id_pracownika, imie, nazwisko, id_stanowiska, id_telefonu, uwagi)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_pracownika \rightarrow imie$$

$$id_pracownika \rightarrow nazwisko$$

$$id_pracownika \rightarrow id_stanowiska$$

$$id_pracownika \rightarrow id_telefonu$$

$$id_pracownika \rightarrow uwagi$$

Zakładamy, że imię i nazwisko nie identyfikują jednoznacznie pracownika. Skoro każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.18. Tabela telefony

Tabela jest postaci:

telefony(id_telefonu, telefon, numer_bliskiego)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_telefonu \rightarrow telefon$$

$$id_telefonu \rightarrow numer_bliskiego$$

Zakładamy, że numery telefonów nie są unikalne, zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.19. Tabela stanowiska

Tabela jest postaci:

stanowiska(id_stanowiska, nazwa_stanowiska, wysokosc_pensji)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_stanowiska \rightarrow nazwa_stanowiska$$

$$id_stanowiska \rightarrow wysokosc_pensji$$

Zakładamy, że nie ma zależności, które zaczynają się od stanowiska, zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.20. Tabela klient_wycieczka

Tabela jest postaci:

klient_wycieczka(id_wycieczki, id_klienta)

W tej tabeli nie występują nietrywialne zależności funkcyjne, dwie kolumny tworzą klucz elementarny, a pomiędzy nimi nie ma zależności. Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.21. Tabela transakcje_pracownicy

Tabela jest postaci:

transakcje_pracownicy(id_transakcji, kwota, data_transakcji, id_pracownik)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_transakcji \rightarrow kwota$$

$$id_transakcji \rightarrow czas_wyjazdu$$

$$id_transakcji \rightarrow data_transakcji$$

$$id_transakcji \rightarrow id_pracownika$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.22. Tabela transakcje_klienci

Tabela jest postaci:

transakcje_klienci(id_transakcji, kwota, data_transakcji, id_klienta, id_wycieczki)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_transakcji \rightarrow kwota$$

$$id_transakcji \rightarrow czas_wyjazdu$$

$$id_transakcji \rightarrow data_transakcji$$

$$id_transakcji \rightarrow id_klienta$$

$$id_transakcji \rightarrow id_wycieczki$$

Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

5.23. Tabela klienci

Tabela jest postaci:

klienci(id_klienta, imie, nazwisko, plec, email, id_telefonu)

Jedynymi nietrywialnymi zależnościami funkcyjnymi są:

$$id_klienta \rightarrow imie$$

$$id_klienta \rightarrow nazwisko$$

$$id_klienta \rightarrow plec$$

$$id_klienta \rightarrow email$$

$$id_klienta \rightarrow id_telefonu$$

Zakładamy, że imię i nazwisko nie identyfikują jednoznacznie klienta. Zatem każda nietrywialna zależność funkcyjna zaczyna się od nadklucza i tabela spełnia wymagania bycia w EKNF.

W żadnym połączeniu tabel nie ma zależności tranzytywnych, każda tabela pojedynczo jest w EKNF, zatem to pokazuje, że cała baza danych jest w EKNF.

6. Co było najtrudniejsze w realizacji projektu

- Dopasowanie realistycznych kosztów usług i podróży.
- Aktualizacja wypełniania bazy danych po zmianach w schemacie.
- Projektowanie schematu i struktury bazy.
- Normalizacja bazy, trudność spowodowana liczbą tabel.
- Stworzenie skryptu znacząco automatyzującego wypełnianie bazy danych.
- Wspólna praca przy pomocy systemu kontroli wersji w przypadku konfliktów.