Baza danych dla warsztatu „Pimp My Wheels” - dokumentacja

Grupa G: Kamila Grząka, Jakub Kempa, Szymon Stano, Julia Wołk-Łaniewska, Agata Żabska

28 czerwca 2024

1 Spis użytych technologii

W wykonanym przez nas projekcie wykorzystaliśmy następujące technologie:

- Python,
- MariaDB,
- RStudio (Posit Cloud),
- KnitR,
- LaTeX.

2 Lista plików i opis ich zawartości

W celu wygenerowania bazy danych oraz odnoszącego sie do niej raportu niezbędne są pliki:

1. `czesci.xlsx` - plik w formacie Excel, który zawiera spis wszystkich części potrzebnych do wykonywania oferowanych przez nas usług.
2. `uslugi.xlsx` - plik w formacie Excel, który zawiera spis wszystkich usług oferowanych przez nasz warsztat wraz z dodatkowymi informacjami dotyczącymi poszczególnych usług.
3. `wyposazenie.xlsx` - plik w formacie Excel, który zawiera spis wszystkich elementów wyposażenia niezbędnych do wykonania oferowanych przez nas usług.
4. `uslugi_czesci.xlsx` - plik Excel, który zawiera informacje o połączeniach między usługami a częściami potrzebnymi do ich wykonania.
5. `stanowiska.xlsx` - plik w formacie Excel, który zawiera informacje o stanowiskach funkcjonujących w naszym warsztacie wraz z informacjami odnośnie płac.
6. `cars.csv` - plik CSV z samochodami wraz z ich parametrami technicznymi.
7. `bikes.csv` - plik CSV z motorami wraz z ich parametrami technicznymi.
8. `clients.csv` - plik CSV zawierający szczegółowe informacje odnośnie osób pojawiających się w naszym warsztacie, zarówno klientów jak i pracowników.
9. `imiona.csv` - plik CSV zawierający imiona osób pojawiających się w naszej bazie danych, zarówno klientów, jak i pracowników.
10. `nazwiska.csv` - plik CSV zawierający nazwiska osób pojawiających się w naszej bazie danych, zarówno klientów, jak i pracowników.
11. `ulice.csv` - plik CSV zawierający ulice wykorzystywane przez nas do tworzenia adresów klientów i pracowników.
12. `uszips.csv` - plik CSV zawierający kody pocztowe wykorzystywane przez nas do tworzenia adresów klientów i pracowników.

13. `kraje.csv` - plik CSV zawierający kraje wykorzystywane przez nas do tworzenia adresów klientów i pracowników.

14. `_init_.py` - plik zawierający kod w języku *Python*, który umożliwia importowanie modułów z podfolderów.

15. `Adresy.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Adresy, jej relacje oraz wypełnia ją danymi.

16. `base.py` - plik zawierający kod w języku *Python*, który inicjalizuje silnik i importuje go.

17. `Czesci.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Czesci, jej relacje oraz wypełnia ją danymi.

18. `Klienci.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Klienci, jej relacje oraz wypełnia ją danymi.

19. `Kraje.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Kraje, jej relacje oraz wypełnia ją danymi.

20. `Kupno_Sprzedaz.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Kupno_Sprzedaz, jej relacje oraz wypełnia ją danymi.

21. `Miejscowosci.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Miejscowosci, jej relacje oraz wypełnia ją danymi.

22. `Naprawy.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Naprawy, jej relacje oraz wypełnia ją danymi.

23. `Naprawy_Pojazdy.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Naprawy_Pojazdy, jej relacje oraz wypełnia ją danymi.

24. `Pracownicy.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Pracownicy, jej relacje oraz wypełnia ją danymi.

25. `Renowacje.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Renowacje, jej relacje oraz wypełnia ją danymi.

26. `Renowacje_Pojazdy.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Renowacje_Pojazdy, jej relacje oraz wypełnia ją danymi.

27. `Stanowiska.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Stanowiska, jej relacje oraz wypełnia ją danymi.

28. `Uslugi.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Uslugi, jej relacje oraz wypełnia ją danymi.

29. `Uslugi_Czesci.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Uslugi_Czesci, jej relacje oraz wypełnia ją danymi.

30. `Uslugi_Napraw.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Uslugi_Napraw, jej relacje oraz wypełnia ją danymi.

31. `Uslugi_Renowacji.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Uslugi_Renowacji, jej relacje oraz wypełnia ją danymi.

32. `Wyposazenie.py` - plik zawierający kod w języku *Python*, który tworzy tabelę Wyposazenie, jej relacje oraz wypełnia ją danymi.

33. `fill_database.py` - plik zawierający kod w języku *Python*, który usuwa wszystkie tabele bazy danych, generuje je na nowo, a następnie wypełnia danymi wykorzystując wszystkie powyższe pliki.

34. `requirements.txt` - plik zawierający informacje o wersjach wykorzystywanych w projekcie bibliotek języka *Python*

35. `Raport.Rmd` - plik, który zawiera zapytania oraz kod generujący raport,

36. `Raport.pdf` - plik, zawierający przykładowy raport.

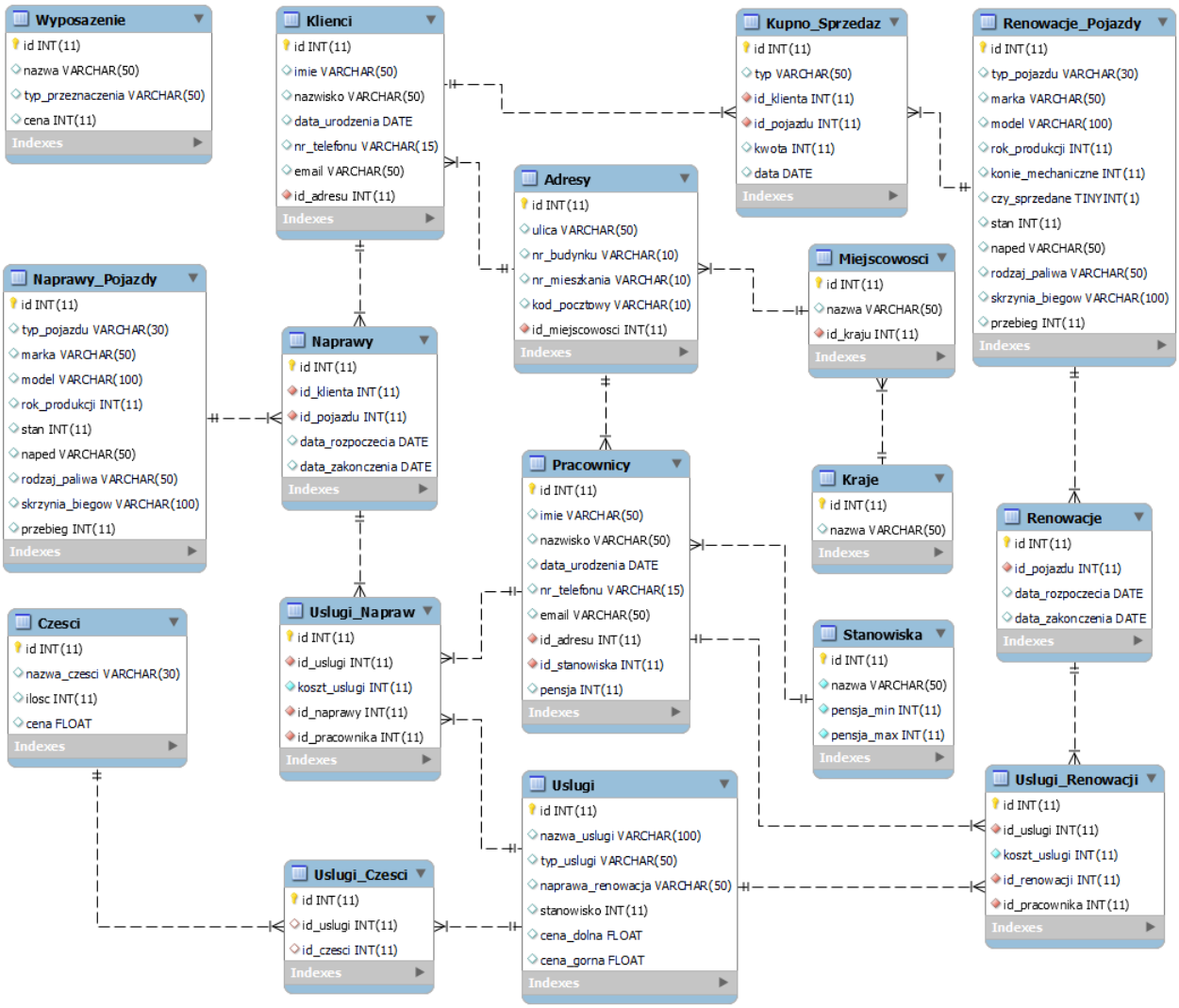
3 Kolejność i sposób uruchamiania plików

Aby wygenerować bazę danych oraz raport musimy wykonać następujące kroki:

1. pobrać załączony plik `GRUPA 08.zip` oraz rozpakować go,
2. zainstalować wymagane biblioteki języka *Python* za pomocą komendy `pip install -r requirements.txt`
3. uruchomić plik `fill_database.py`,
4. uruchomić plik `raport.Rmd` przez program *RStudio*, lub przez jego wersję online Posit Cloud (preferowane)
   (a) założyć darmowe konto na wyżej wymienionej stronie lub zalogować się na swoje konto
   (b) utworzyć nowy projekt przyciskiem ***New Project*** → ***New RStudio Project***
   (c) odnaleźć przycisk ***Upload***, dodać plik `output.Rmd` oraz go otworzyć
   (d) zainstalować wymagane pakiety, za pomocą przycisku **install** umieszczonym w pojawiającym się żółtym pasku ostrzegawczym
   (e) po zainstalowaniu kliknąć przycisk ***Knit*** umieszczony nad treścią naszego pliku

4 Schemat projektu bazy danych

Poniższy schemat przedstawia projekt naszej bazy danych - tabele oraz relacje zachodzące pomiędzy nimi.



Rysunek 1: Schemat bazy danych

Zależności funkcyjne między tabelami

W przedstawionym kodzie można zidentyfikować kilka zależności funkcyjnych między różnymi tabelami baz danych. Zależności te wynikają z zastosowania kluczy obcych i relacji między tabelami. Poniżej znajduje się omówienie tych zależności.

Zależności między tabelami

- **Adresy**:
  - `Adresy.id_miejscowosci` → `Miejscowosci.id` (Adres należy do jednej miejscowości)
  - `Adresy.id` → `Klienci.id_adresu` (Klient może mieć jeden adres)
  - `Adresy.id` → `Pracownicy.id_adresu` (Pracownik może mieć jeden adres)
- **Czesci**:
  - `Czesci.id` → `Uslugi_Czesci.id_czesci` (Część może być używana w wielu usługach)
- **Kraje**:
  - `Kraje.id` → `Miejscowosci.id_kraju` (Miejscowość znajduje się w jednym kraju)
- **Kupno_Sprzedaz**:
  - `Kupno_Sprzedaz.id_klienta` → `Klienci.id` (Każda transakcja jest powiązana z jednym klientem)
  - `Kupno_Sprzedaz.id_pojazdu` → `Renowacje_Pojazdy.id` (Każda transakcja dotyczy jednego pojazdu)
- **Miejscowosci**:
  - `Miejscowosci.id_kraju` → `Kraje.id` (Miejscowość znajduje się w jednym kraju)
  - `Miejscowosci.id` → `Adresy.id_miejscowosci` (Miejscowość może mieć wiele adresów)
- **Naprawy_Pojazdy**:
  - `Naprawy_Pojazdy.id` → `Naprawy.id_pojazdu` (Naprawa dotyczy jednego pojazdu)
- **Naprawy**:
  - `Naprawy.id_klienta` → `Klienci.id` (Naprawa jest zlecona przez jednego klienta)
  - `Naprawy.id_pojazdu` → `Naprawy_Pojazdy.id` (Naprawa dotyczy jednego pojazdu)
  - `Naprawy.id` → `Uslugi_Napraw.id_naprawy` (Naprawa może mieć wiele usług naprawczych)
- **Pracownicy**:
  - `Pracownicy.id_adresu` → `Adresy.id` (Pracownik ma jeden adres)
  - `Pracownicy.id_stanowiska` → `Stanowiska.id` (Pracownik ma jedno stanowisko)
  - `Pracownicy.id` → `Uslugi_Napraw.id_pracownika` (Pracownik może być przypisany do wielu usług naprawczych)
  - `Pracownicy.id` → `Uslugi_Renowacji.id_pracownika` (Pracownik może być przypisany do wielu usług renowacyjnych)
- **Renowacje_Pojazdy**:
  - `Renowacje_Pojazdy.id` → `Renowacje.id_pojazdu` (Renowacja dotyczy jednego pojazdu)
  - `Renowacje_Pojazdy.id` → `Kupno_Sprzedaz.id_pojazdu` (Pojazd może być przedmiotem wielu transakcji)
- **Renowacje**:
  - `Renowacje.id_pojazdu` → `Renowacje_Pojazdy.id` (Renowacja dotyczy jednego pojazdu)
  - `Renowacje.id` → `Uslugi_Renowacji.id_renowacji` (Renowacja może mieć wiele usług renowacyjnych)

- **Stanowiska**:
  - `Stanowiska.id` → `Pracownicy.id_stanowiska` (Stanowisko może być przypisane do wielu pracowników)
- **Uslugi_Czesci**:
  - `Uslugi_Czesci.id_uslugi` → `Uslugi.id` (Usługa może mieć wiele części)
  - `Uslugi_Czesci.id_czesci` → `Czesci.id` (Część może być używana w wielu usługach)
- **Uslugi_Napraw**:
  - `Uslugi_Napraw.id_uslugi` → `Uslugi.id` (Usługa naprawy jest powiązana z jedną usługą)
  - `Uslugi_Napraw.id_naprawy` → `Naprawy.id` (Usługa naprawy jest częścią jednej naprawy)
  - `Uslugi_Napraw.id_pracownika` → `Pracownicy.id` (Usługa naprawy jest realizowana przez jednego pracownika)
- **Uslugi_Renowacji**:
  - `Uslugi_Renowacji.id_uslugi` → `Uslugi.id` (Usługa renowacji jest powiązana z jedną usługą)
  - `Uslugi_Renowacji.id_renowacji` → `Renowacje.id` (Usługa renowacji jest częścią jednej renowacji)
  - `Uslugi_Renowacji.id_pracownika` → `Pracownicy.id` (Usługa renowacji jest realizowana przez jednego pracownika)
- **Uslugi**:
  - `Uslugi.id` → `Uslugi_Czesci.id_uslugi` (Usługa może mieć wiele części)
  - `Uslugi.id` → `Uslugi_Napraw.id_uslugi` (Usługa może być częścią wielu napraw)
  - `Uslugi.id` → `Uslugi_Renowacji.id_uslugi` (Usługa może być częścią wielu renowacji)

5 Normalizacja

Tabele przedstawione w naszej bazie danych cechują się tym, że wszystkie zależności funkcyjne w obrębie tabeli wychodzą od identyfikatora (id). Zapewnia to jednoznaczność i spójność naszych danych. Na tej podstawie możemy stwierdzić, że wykonana przez nas baza danych spełnia założenia BCNF (Boyce-Codd Normal Form), zatem jest również w EKNF (Elementary Key Normal Form). Poniżej przedstawiliśmy szczegółowo rozpisane zależności dla jednej z tabel. W pozostałych przypadkach wygląda to analogicznie.

Tabela: Pracownicy
Zmienne: id, imie, nazwisko, data_urodzenia, nr_telefonu, email, id_adresu, id_stanowiska, pensja
Klucz główny: id
Zależności funkcyjne: $\sum = \{\text{id} \rightarrow \text{imie},\ \text{id} \rightarrow \text{nazwisko},\ \text{id} \rightarrow \text{data\_urodzenia},\ \text{id} \rightarrow \text{nr\_telefonu},\ \text{id} \rightarrow \text{email},\ \text{id} \rightarrow \text{id\_adresu},\ \text{id} \rightarrow \text{id\_stanowiska},\ \text{id} \rightarrow \text{pensja}\}$

6 Co było najtrudniejsze?

1. Praca z bazą danych, dla której połączenie notorycznie jest przerywane z uwagi na zbyt dużą ilość zalogowanych użytkowników,
2. Ustalenie zależności funkcyjnych dla relacji w bazie danych,
3. Odpowiednia komunikacja, związana z wysyłaniem zapytań do bazy i wypełnianiem bazy nowymi danymi
4. Tworzenie i poszukiwanie odpowiednich plików z danymi, potrzebnych do wypełnienia bazy