Dokumentacja bazy danych

Bartłomiej Dymalski,Bartosz Cholewa,
Radosław Tertel, Bartłomiej Brzozowski,
Ignacy Nowakowski

7 lipca 2024

1 Spis użytych technologii

Do przygotowania niniejszego projektu wykorzystano:

- Python zintegrowany z SQL do analizy oraz skryptowego wypełniania bazy danych,
- SQL z dialektem MariaDB,
- edytor ERD,
- biblioteki numpy,datetime,pandas,matplotlib oraz fpdf do generowania raportu

2 Lista plików i opis ich zawartości

1. MI.csv - baza imion męskich,
2. Zi.csv - baza imion żeńskich,
3. ZN.csv - baza nazwisk żeńskich,
4. MN.csv - baza nazwisk męskich,
5. UliceZg.csv - baza nazw ulic w Zgorzelcu,
6. UliceWro.csv - baza nazw ulic we Wrocławiu,
7. UliceWał.csv - baza nazw ulic w Wałbrzychu,
8. UliceOpole.csv - baza nazw ulic w Opolu,
9. UliceLeszno.csv - baza nazw ulic w Lesznie,
10. UliceLeg.csv - baza nazw ulic w Legnicy,
11. UliceJel.csv - baza nazw ulic w Jeleniej Górze,
12. RS.csv - baza z markami i modelami pojazdów,
13. baza.erd - plik ze schematem bazy

14. Projekt-Uzupelnienie-Danych.ipynb - główny skrypt tworzący bazy oraz uzupełniający je,
15. raport.py - plik generujący wykresy, tabele oraz raport końcowy.
16. DejaVuSansCondensed.ttf-plik zawierający czcionkę używaną w raporcie

3 Kolejność uruchamiania plików

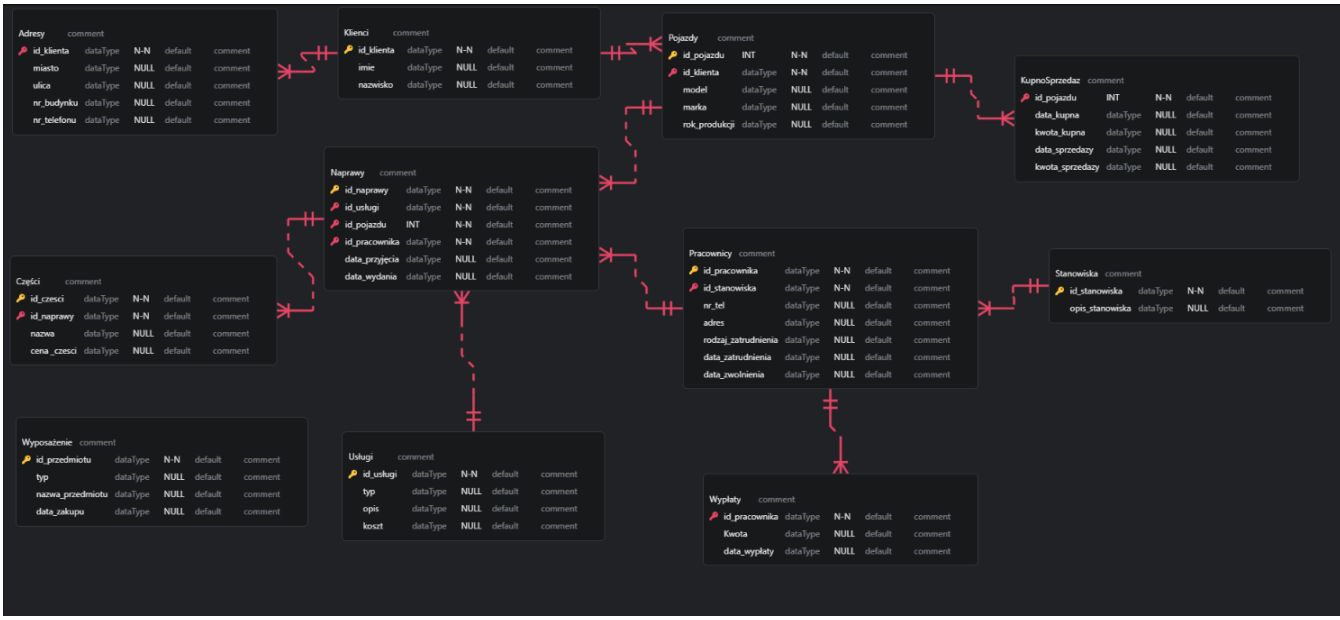
1. Projekt-Uzupełnienie-Danych.ipynb - plik stworzy wszystkie tabele w bazie danych oraz je wypełni,
2. raport.py - plik wygeneruje raport.pdf zawierający całą analizę bazy danych.

4 Schemat Bazy danych

5 Lista zależności funkcyjnych

Dla ułatwienia ustaliliśmy oznaczenia dla poszczególnych tabel

- A - Adresy
- B - Klienci
- C - Pojazdy
- D - KupnoSprzedaz
- E - Naprawy
- F - Wypłaty
- G - Części
- H - Wyposażenie
- I - Usługi



Rysunek 1: Schemat bazy danych wykonany w edytorze ERD

- J - Pracownicy
- K - Stanowiska

Zbiór zależności funkcyjnych wraz z wyjaśnieniami

1. AC $\rightarrow$ B - tabele łączy klucz id_klienta, aby dało się połączyć pojazdy oraz adresy z klientem,
2. DE $\rightarrow$ C - tabele łączy klucz id_pojazdu, żeby dało się powiązać naprawy oraz czy samochód został kupiony i sprzedany z danym pojazdem,
3. E $\rightarrow$ I - tabele łączy klucz id_usługi, żeby dało się powiązać naprawę z usługą, czyli co podczas niej zostało zrobione,
4. G $\rightarrow$ E - tabele łączy klucz id_naprawy, żeby dało się powiązać część z naprawą w jakiej ją użyto,
5. EF $\rightarrow$ J - tabele łączy klucz id_pracownika, żeby dało się powiązać naprawę oraz wypłaty z odpowiednim pracownikiem,

6. J → K - tabele łączy klucz id_stanowiska, żeby dało się powiązać pracownika oraz stanowisko na jakim jest zatrudniony.

6 Uzasadnienie, że baza jest EKNF

Algorytm Bernsteina jest używany do przekształcania relacji baz danych do postaci EKNF (Eksportowej Kodowalnej Normalnej Formy), która jest postacią 3NF. Proces ten składa się z kilku etapów, które opisano poniżej.

6.1 Jeszcze raz relacje funkcyjne

- $AC \rightarrow B$ (klucz: $id\_klienta$)
- $DE \rightarrow C$ (klucz: $id\_pojazdu$)
- $E \rightarrow I$ (klucz: id_usugi)
- $G \rightarrow E$ (klucz: $id\_naprawy$)
- $EF \rightarrow J$ (klucz: $id\_pracownika$)
- $J \rightarrow K$ (klucz: $id\_stanowiska$)

6.2 Identyfikacja kluczy kandydujących

Zidentyfikowane klucze kandydujące dla każdej zależności:

- $AC \rightarrow B$: Klucz kandydujący to AC
- $DE \rightarrow C$: Klucz kandydujący to DE
- $E \rightarrow I$: Klucz kandydujący to E
- $G \rightarrow E$: Klucz kandydujący to G
- $EF \rightarrow J$: Klucz kandydujący to EF
- $J \rightarrow K$: Klucz kandydujący to J

6.3 Budowa schematów relacji dla każdej zależności funkcjonalnej

Dla każdej zależności tworzymy osobny schemat relacji:

- $R_1(AC, B)$
- $R_2(DE, C)$
- $R_3(E, I)$
- $R_4(G, E)$
- $R_5(EF, J)$
- $R_6(J, K)$

6.4 Sprawdzanie i łączenie schematów relacji

Sprawdzamy, czy możemy połączyć jakieś schematy relacji bez wprowadzenia redundancji:

- Relacja $R_3(E, I)$ oraz $R_4(G, E)$:
  - Można je połączyć, ponieważ $G \to E$ i $E \to I$ dają $G \to I$, co nie tworzy redundancji.
  - Powstała relacja: $R_{34}(G, E, I)$
- Relacja $R_3(E, I)$, $R_4(G, E)$, i $R_5(EF, J)$:
  - Można je połączyć, ponieważ $EF \to J$, gdzie $E \to I$, więc tworzymy relację.
  - Powstała relacja: $R_{345}(E, I, J)$

6.5 Usunięcie nadmiarowych zależności

Sprawdzamy, czy jakieś zależności są nadmiarowe:

- Połączenia R_{345} oraz R_5 i R_6 są nadmiarowe.

6.6 Weryfikacja 3NF

Sprawdzamy każdą relację pod kątem 3NF:

- $R_1(AC, B)$: $AC \to B$ jest prawidłowe, ponieważ AC jest kluczem kandydującym.
- $R_2(DE, C)$: $DE \to C$ jest prawidłowe, ponieważ DE jest kluczem kandydującym.
- $R_{345}(G, E, I, J)$: $G \to E$ i $E \to I$ oraz $EF \to J$ są prawidłowe.
- $R_6(J, K)$: $J \to K$ jest prawidłowe, ponieważ J jest kluczem kandydującym.

6.7 Podsumowanie

Schematy relacji zostały przekształcone do postaci 3NF przy użyciu algorytmu Bernsteina. Każda relacja jest teraz zgodna z 3NF, co oznacza, że baza danych jest zoptymalizowana pod kątem minimalizacji redundancji i zachowania spójności danych a więc jest EKNF.

7 Najtrudniejsze elementy podczas realizacji projektu

Najtrudniejszy element całego projektu stanowiło skryptowe wypełnianie bazy oraz odpowiednie zaplanowanie połączeń, tak aby baza była EKNF i aby wszystko miało sens. Przy skryptowym wypełnianiu należało wywoływać tabele w odpowiedniej kolejności oraz usuwać je odpowiednio aby nie generować błędów.