# Database Documentation

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

7 lipca 2024

## 1 List of Used Technologies

The following technologies were used to prepare this project:

- Python integrated with SQL for analysis and scripting database population,
- SQL with MariaDB dialect,
- ERD editor,
- Libraries: numpy, datetime, pandas, matplotlib, and fpdf for report generation

## 2 File List and Content Description

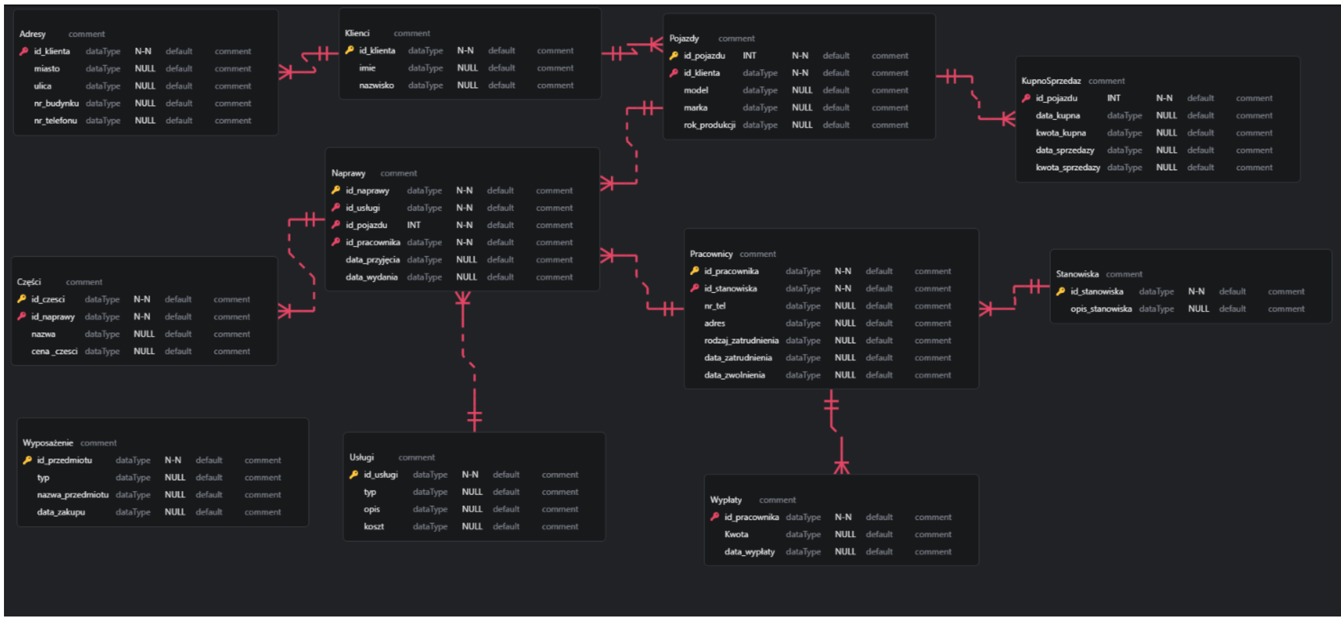
1. MI.csv - male first names database,
2. Zi.csv - female first names database,
3. ZN.csv - female last names database,
4. MN.csv - male last names database,
5. UliceZg.csv - street names database for Zgorzelec,
6. UliceWro.csv - street names database for Wrocław,
7. UliceWał.csv - street names database for Wałbrzych,
8. UliceOpole.csv - street names database for Opole,
9. UliceLeszno.csv - street names database for Leszno,
10. UliceLeg.csv - street names database for Legnica,
11. UliceJel.csv - street names database for Jelenia Góra,
12. RS.csv - vehicle brands and models database,
13. baza.erd - database schema file

14. Projekt-Uzupelnienie-Danych.ipynb - main script for creating and populating the database,
15. raport.py - file for generating charts, tables, and the final report.
16. DejaVuSansCondensed.ttf - file containing the font used in the report

## 3 File Execution Order

1. Projekt-Uzupełnienie-Danych.ipynb - this file will create all tables in the database and populate them,
2. raport.py - this file will generate the report.pdf containing the entire database analysis.

## 4 Database Schema


Adresy comment
id_klienta dataType N-N default comment
miasto dataType NULL default comment
ulica dataType NULL default comment
nr_budynku dataType NULL default comment
nr_telefonu dataType NULL default comment
Klienci comment
id_klienta dataType N-N default comment
imie dataType NULL default comment
nazwisko dataType NULL default comment
Pojazdy comment
id_pojazdu INT N-N default comment
id_klienta dataType N-N default comment
model dataType NULL default comment
marka dataType NULL default comment
rok_produkcji dataType NULL default comment
KupnoSprzedaz comment
id_pojazdu INT N-N default comment
data_kupna dataType NULL default comment
kwota_kupna dataType NULL default comment
data_sprzedazy dataType NULL default comment
kwota_sprzedazy dataType NULL default comment
Naprawy comment
id_naprawy dataType N-N default comment
id_usługi dataType N-N default comment
id_pojazdu INT N-N default comment
id_pracownika dataType N-N default comment
data_przyjęcia dataType NULL default comment
data_wydania dataType NULL default comment
Części comment
id_czesci dataType N-N default comment
id_naprawy dataType N-N default comment
nazwa dataType NULL default comment
cena _czesci dataType NULL default comment
Pracownicy comment
id_pracownika dataType N-N default comment
id_stanowiska dataType N-N default comment
nr_tel dataType NULL default comment
adres dataType NULL default comment
rodzaj_zatrudnienia dataType NULL default comment
data_zatrudnienia dataType NULL default comment
data_zwolnienia dataType NULL default comment
Stanowiska comment
id_stanowiska dataType N-N default comment
opis_stanowiska dataType NULL default comment
Wyposażenie comment
id_przedmiotu dataType N-N default comment
typ dataType NULL default comment
nazwa_przedmiotu dataType NULL default comment
data_zakupu dataType NULL default comment
Usługi comment
id_usługi dataType N-N default comment
typ dataType NULL default comment
opis dataType NULL default comment
koszt dataType NULL default comment
Wypłaty comment
id_pracownika dataType N-N default comment
Kwota dataType NULL default comment
data_wypłaty dataType NULL default comment

Rysunek 1: Database schema created in the ERD editor

## 5 List of Functional Dependencies

For convenience, we have established abbreviations for the individual tables

- A - Addresses
- B - Clients
- C - Vehicles
- D - PurchaseSale
- E - Repairs
- F - Payments
- G - Parts
- H - Equipment
- I - Services
- J - Employees
- K - Positions

Set of functional dependencies with explanations

1. AC $\rightarrow$ B - tables are connected by the client id key to link vehicles and addresses to the client,
2. DE $\rightarrow$ C - tables are connected by the vehicle id key to link repairs and whether the car was bought and sold to the given vehicle,
3. E $\rightarrow$ I - tables are connected by the service id key to link the repair with the service, i.e., what was done during it,
4. G $\rightarrow$ E - tables are connected by the repair id key to link the part with the repair in which it was used,
5. EF $\rightarrow$ J - tables are connected by the employee id key to link the repair and payments with the appropriate employee,
6. J $\rightarrow$ K - tables are connected by the position id key to link the employee and the position they are employed in.

## 6 Justification that the Database is in BCNF

Bernstein's algorithm is used to transform database relations into BCNF (Boyce-Codd Normal Form), which is a form of 3NF. This process consists of several steps described below.

### 6.1 Functional Dependencies Again

- $AC \to B$ (key: *id_client*)
- $DE \to C$ (key: *id_vehicle*)
- $E \to I$ (key: *id_service*)
- $G \to E$ (key: *id_repair*)
- $EF \to J$ (key: *id_employee*)
- $J \to K$ (key: *id_position*)

### 6.2 Identification of Candidate Keys

Identified candidate keys for each dependency:

- $AC \to B$: Candidate key is $AC$
- $DE \to C$: Candidate key is $DE$
- $E \to I$: Candidate key is $E$
- $G \to E$: Candidate key is $G$
- $EF \to J$: Candidate key is $EF$
- $J \to K$: Candidate key is $J$

## 6.3 Building Relation Schemas for Each Functional Dependency

For each dependency, we create a separate relation schema:

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

## 6.4 Checking and Merging Relation Schemas

We check if we can merge any relation schemas without introducing redundancy:

- Relation $R_3(E, I)$ and $R_4(G, E)$:
  - They can be merged because $G \to E$ and $E \to I$ give $G \to I$, which does not create redundancy.
  - Resulting relation: $R_{34}(G, E, I)$
- Relation $R_3(E, I)$, $R_4(G, E)$, and $R_5(EF, J)$:
  - They can be merged because $EF \to J$, where $E \to I$, so we create a relation.
  - Resulting relation: $R_{345}(E, I, J)$

## 6.5 Removing Redundant Dependencies

We check if any dependencies are redundant:

- Connections $R_{345}$ and $R_5$ and $R_6$ are redundant.

### 6.6 Verification of 3NF

We check each relation for 3NF compliance:

- $R_1(AC, B)$: $AC \to B$ is correct because $AC$ is a candidate key.
- $R_2(DE, C)$: $DE \to C$ is correct because $DE$ is a candidate key.
- $R_{345}(G, E, I, J)$: $G \to E$ and $E \to I$ and $EF \to J$ are correct.
- $R_6(J, K)$: $J \to K$ is correct because $J$ is a candidate key.

### 6.7 Summary

The relation schemas have been transformed to 3NF using Bernstein's algorithm. Each relation now complies with 3NF, which means the database is optimized for minimizing redundancy and maintaining data consistency, hence it is in BCNF.

## 7 The Most Challenging Elements During the Project

The most challenging part of the entire project was the scripted population of the database and appropriately planning the connections to ensure the database is in BCNF and makes sense. During scripted population, tables had to be invoked in the correct order and removed appropriately to avoid generating errors.