

Dokumentacja — projekt bazy danych

Czerwiec 2025

1. Spis użytych technologii:

1. Jupyter Notebook – użyty do połączenia skryptów generujących dane i wypełniających tabele oraz opisów poszczególnych tabel.
2. Python (wersja 3.12.1) – użyty do napisania skryptów. Wykorzystano następujące biblioteki:
 - a) NumPy
 - b) Pandas
 - c) Collections
 - d) Typing
 - e) Datetime
3. R (wersja 4.4.1) – użyty do przeprowadzenia analizy danych. Wykorzystano następujące pakiety:
 - a) RMariaDB
 - b) Ggplot2
 - c) Knitr
4. Quarto – użyty do połączenia skryptów w języku R oraz komentarzy zawierających treść raportu, a następnie wygenerowania gotowego raportu.
5. LaTeX – użyty do napisania dokumentacji.

2. Lista plików i opis ich zawartości:

1. StarWars.ipynb – plik Jupyter Notebook z podstawowymi informacjami o firmie, opisem warunków, z których korzystaliśmy przy wypełnianiu tabel danymi oraz skryptami napisanymi w języku Python, do generowania oraz wypełniania danymi bazy.
2. customer_names.csv – plik CSV, zawierający listę imion i nazwisk potencjalnych klientów, pobrany ze strony [Kaggle](#).
3. StarWars.json.vuerd – schemat bazy danych ERD w formacie Vuerd.
4. Raport.qmd – plik Quarto z tekstem raportu oraz skryptami w języku R, służącymi do połączenia się z bazą danych oraz uzyskania wyników zapytań, a także do analizy otrzymanych wyników.
5. Raport.html – plik wynikowy, otrzymany po zrenderowaniu pliku Quarto, zawierający analizę danych z bazy.
6. styles.css – plik CSS, służący do określenia stylu dokumentu HTML (wyśrodkowanie podpisów do każdego wykresu).
7. Dokumentacja_bazy_danych.pdf - dokumentacja projektu w formacie PDF.

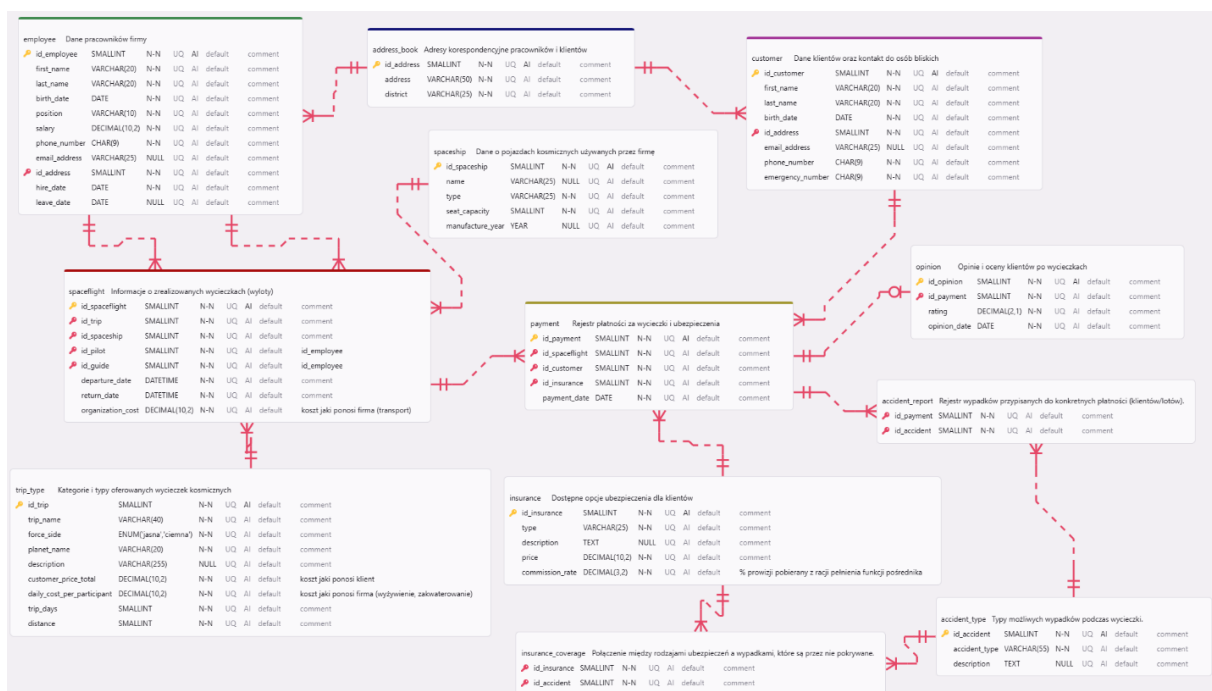
3. Kolejność i sposób uruchamiania kolejnych plików, aby uzyskać gotowy projekt:

W celu uruchomienia wszystkich plików, niezbędne jest posiadanie Pythona w wersji przynajmniej 3.11 oraz R w wersji 4.4.1. wraz z wymienionymi bibliotekami i pakietami, a także odpowiednie środowiska programistyczne. Niezbędna będzie także możliwość obsługi plików Jupyter Notebook oraz Quarto.

Kolejność uruchamiania plików:

1. W pierwszej kolejności należy otworzyć plik StarWars.ipynb. Uruchomienie wszystkich komórek opcją Run All pozwoli na połączenie się z bazą danych, następnie stworzenie odpowiednich tabel oraz wypełnienie ich danymi według określonych warunków.
2. Następnie należy otworzyć plik Raport.qmd, zawierający tekst raportu oraz komórki ze skryptem napisanym w języku R, służącym do połączenia z bazą danych oraz wykonania zdefiniowanych zapytań. Wykonanie opcji „Render” pozwoli na wywołanie każdej komórki kodu oraz uzyskanie wyników, a także stworzenie gotowego raportu w formacie HTML.
3. Po utworzeniu pliku Raport.html dostępny jest tekst raportu, wykonanego na podstawie analizy danych z bazy.

4. Schemat projektu bazy danych:



Rysunek 1. Schemat projektu bazy danych

5. Lista zależności funkcyjnych z wyjaśnieniem dla każdej relacji:

5.1. Lista zależności funkcyjnych w tabelach

1. **employee**: $id_employee \rightarrow first_name, last_name, birth_date, position, salary, phone_number, email_address, id_address, hire_date, leave_date$
 $id_employee$ – klucz główny, $id_address$ – klucz obcy.
 Id pracownika wskazuje jednoznacznie na konkretną osobę, którą opisują imię, nazwisko, data urodzenia i inne dane personalne, a także data zatrudnienia i (ewentualnie) odejścia z firmy, zajmowane stanowisko i otrzymywana wypłata.
2. **address_book**: $id_address \rightarrow address, district$
 $id_address$ – klucz główny.
 Każdy adres ma przypisane w tabeli unikalne id.
3. **customer**: $id_customer \rightarrow first_name, last_name, birth_date, id_address, email_address, phone_number, emergency_number$
 $id_customer$ – klucz główny, $id_address$ – klucz obcy.
 Podobnie jak w przypadku tabeli **employee**, id klienta wskazuje jednoznacznie na konkretną osobę o określonych danych personalnych.

4. **spaceship:** $id_spaceship \rightarrow name, type, seat_capacity, manufacture_year$
 $id_spaceship$ – klucz główny.
 Każdy posiadany przez firmę pojazd kosmiczny charakteryzuje się pewną nazwą, typem, pojemnością i rokiem produkcji.
5. **spaceflight:** $id_spaceflight \rightarrow id_trip, id_spaceship, id_pilot, id_guide, departure_date, return_date, organisation_cost$
 $id_spaceflight$ – klucz główny, $id_trip, id_spaceship, id_pilot, id_guide$ – klucze obce.
 Każdy wylot związany jest z pewną wycieczką, pojazdem, pilotem, przewodnikiem, datą wyjazdu oraz powrotu i kosztem organizacji.
6. **payment:** $id_payment \rightarrow id_spaceflight, id_customer, id_insurance, payment_date$
 $id_payment$ – klucz główny, $id_spaceflight, id_customer, id_insurance$ – klucze obce.
 Każda płatność związana jest z określonym lotem, jest uiszczana przez pewną osobę, która wykupiła ubezpieczenie lub nie, oraz jest dokumentowana określonego dnia.
7. **trip_type:** $id_trip \rightarrow trip_name, force_side, planet_name, description, customer_price_total, daily_cost_per_participant, trip_days, distance$
 id_trip – klucz główny.
 Każda wycieczka ma inny typ, przypisaną stronę mocy, nazwę planety i opis, wiąże się z określonymi kosztami, trwa pewną ilość dni, a w jej trakcie przebywana jest określona odległość.
8. **opinion:** $id_opinion \rightarrow id_payment, rating, opinion_date$
 $id_opinion$ – klucz główny, $id_payment$ – klucz obcy.
 Określona opinia została wystawiona po danej wycieczce, wyraża pewną ocenę i została wystawiona konkretnego dnia.
9. **insurance:** $id_insurance \rightarrow type, description, price, commission_rate$
 $id_insurance$ – klucz główny.
 Każde ubezpieczenie ma przypisany typ, opis, cenę i procent prowizji pobierany z racji pełnienia funkcji pośrednika.
10. **insurance.coverage:** $id_accident, id_insurance$
 $id_accident, id_insurance$ – klucze obce.
 Różne wypadki są przyporządkowane do różnych typów ubezpieczeń.
11. **accident.report:** $id_payment, id_accident$
 $id_payment, id_accident$ – klucze obce.
 Każda osoba, która wybrała się w podróż, mogła ulec wielu wypadkom.
12. **accident_type:** $id_accident \rightarrow accident_type, description$
 $id_accident$ – klucz główny.
 Id wypadku wskazuje jednoznacznie na typ i opis wypadku.

5.2. Lista zależności pomiędzy tabelami

- **employee** i **spaceflight**: $employee.id_employee \rightarrow spaceflight.id_pilot$
 Relacja 1:N (jeden do wielu, pracownik może latać jako pilot na wiele wycieczek).
- **employee** i **spaceflight**: $employee.id_employee \rightarrow spaceflight.id_guide$
 Relacja 1:N (jeden do wielu, pracownik może latać jako przewodnik na wiele wycieczek).
- **address_book** i **employee**: $address_book.id_address \rightarrow employee.id_address$
 Relacja 1:N (jeden do wielu, jeden adres może być przypisany do wielu pracowników).
- **address_book** i **customer**: $address_book.id_address \rightarrow customer.id_address$
 Relacja 1:N (jeden do wielu, jeden adres może być przypisany do wielu klientów).
- **spaceship** i **spaceflight**: $spaceship.id_spaceship \rightarrow spaceflight.id_spaceship$
 Relacja 1:N (jeden do wielu, jeden statek odbył wiele lotów).
- **trip_type** i **spaceflight**: $trip_type.id_trip \rightarrow spaceflight.id_trip$
 Relacja 1:N (jeden do wielu, każdy typ wycieczki mógł odbyć się wielokrotnie).
- **spaceflight** i **payment**: $spaceflight.id_spaceflight \rightarrow payment.id_spaceflight$
 Relacja 1:N (jeden do wielu, na tą samą wycieczkę poleciało wiele osób).
- **customer** i **payment**: $customer.id_customer \rightarrow payment.id_customer$
 Relacja 1:N (jeden do wielu, jeden klient mógł wielokrotnie pojechać na wycieczkę).
- **payment** i **opinion**: $payment.id_payment \rightarrow opinion.id_payment$
 Relacja 0:1 (zero lub jeden, po każdej wycieczce można wystawić co najwyżej jedną opinię).
- **insurance** i **payment**: $insurance.id_insurance \rightarrow payment.id_insurance$
 Relacja 1:N (jeden do wielu, jeden typ ubezpieczenia mógł zostać wykupiony w wielu rezerwacjach).

- **insurance i insurance_coverage:**
 $insurance.id_insurance \rightarrow insurance_coverage.id_insurance$
 Relacja 1:N (jeden do wielu, jeden typ ubezpieczenia pokrywa kilka wypadków).
- **accident_type i insurance_coverage:**
 $(insurance_coverage.id_insurance, insurance_coverage.id_accident)$
 Relacja N:M (wiele do wielu, jeden typ wypadku jest pokrywany przez wiele ubezpieczeń, jedno ubezpieczenie może obejmować wiele typów wypadków).
- **accident_report i payment:**
 $(accident_report.id_payment, accident_report.id_accident)$
 Relacja N:M (wiele do wielu, każdy typ wypadku został zgłoszony w przypadku wielu rezerwacji, każda osoba, która dokonała rezerwacji, może być związana z więcej niż jednym zgłoszonym wypadkiem).
- **accident_type i accident_report:**
 $accident_type.id_accident \rightarrow accident_report.id_accident$
 Relacja 1:N (jeden do wielu, każdy typ wypadku został zgłoszony wiele razy).

6. Uzasadnienie, że baza jest w EKNF:

Baza jest w EKNF, ponieważ każda z komórek zawiera pojedynczą wartość i wszystkie rekordy są unikalne (baza spełnia wymogi 1NF), w tabelach nie ma częściowych zależności (baza spełnia wymogi 2NF) i nie zawierają one zależności przechodnich (spełniony wymóg 3 NF). Ponadto wartości w poszczególnych polach tabel zależą tylko od klucza głównego.

Występowanie relacji N:M nie narusza EKNF, ponieważ w przypadku tabeli insurance_coverage oraz accident_report klucz główny traktujemy jako parę kluczy obcych – w tabelach nie występują inne atrybuty zależne od lewej lub prawej kolumny klucza (tabele te zawierają wyłącznie klucze główne, nie występują tam inne kolumny). Wszystkie zależności funkcyjne odnoszą się do całego klucza, jeżeli istnieją. Z tego powodu nie są naruszone zasady EKNF.

7. Opis, co było najtrudniejsze podczas realizacji projektu:

Najtrudniejsza w realizacji projektu była część związana z tworzeniem tabel, zwłaszcza w kwestii stworzenia spójnego systemu opłat za wycieczki (m.in. za zakwaterowanie czy paliwo) oraz stworzenie tabel z informacjami na temat ubezpieczeń tak, aby baza danych spełniała również wymogi EKNF. Część ta wymagała dobrego przemyślenia schematu bazy danych oraz warunków, branych pod uwagę przy wstawianiu wartości.