Dokumentacja Bazy Danych

Weronika Jaszkiewicz, Dominik Hołoś, Katarzyna Rudzińska, Weronika Pyrtak

Czerwiec 2025

Spis treści

1 Wstęp

Celem niniejszej dokumentacji jest przedstawienie projektu bazy danych opracowanego w ramach przedmiotu Bazy Danych. Projekt dotyczy firmy Space-U, specjalizującej się w organizacji załogowych, konsumenckich lotów kosmicznych.

Zaprojektowana baza danych umożliwia kompleksowe zarządzanie działalnością firmy. Uwzględnia m.in. informacje o klientach, pracownikach, rakietach, typach podróży, a także szczegóły dotyczące transakcji, rezerwacji oraz miejsc docelowych.

Dokumentacja zawiera opis przyjętych założeń, technologii, strukturę plików, graficzny schemat bazy danych oraz szczegółowy opis tabel i relacji między nimi. Całość kończy się podsumowaniem pracy nad projektem.

2 Spis użytych technologii

Do realizacji projektu użyto następujących technologii:

1. Python 3.9.12 – użyty do napisania skryptów generujących dane i wypełniających bazę.
2. Biblioteki Pythona:
   - `sqlalchemy` – narzędzie do obsługi baz danych.
   - `urllib.parse` – moduł do parsowania i manipulowania URL-ami.
   - `pandas` 2.2.3 – biblioteka do manipulacji i analizy danych.
   - `numpy` 2.1.3 – narzędzie do obliczeń numerycznych.
   - `random` – moduł do generowania liczb pseudolosowych.
   - `collections` – rozszerzone typy kontenerów danych, np. deque, Counter.
   - `faker` – biblioteka do generowania fałszywych danych testowych.
   - `datetime` – moduł do pracy z datami i czasem.
   - `itertools` – narzędzia do efektywnego iterowania.
   - `csv` – moduł do czytania i zapisywania plików CSV.
3. ERD Editor 2.0.4 – wykorzystany do stworzenia schematu bazy danych w formie diagramu ERD.
4. R 4.4.1 – użyty do przygotowania raportu oraz analizy danych.
5. Pakiety R:
   - `RMariaDB` – komunikacja z bazą MariaDB/MySQL.
   - `DBI` – interfejs do baz danych.
   - `RMySQL` – komunikacja z MySQL.
   - `dplyr` – pakiet do manipulacji danych.
   - `ggplot2` – narzędzie do wizualizacji danych.
   - `tidyr` – pakiet do porządkowania danych.
   - `lubridate` – praca z datami i czasem.
   - `RColorBrewer` – palety kolorów w wizualizacjach.
   - `scales` – skalowanie danych w wizualizacjach.
6. LaTeX – użyty do przygotowania dokumentacji projektu.

3 Spis plików

Opis plików zawartych w projekcie:

1. Katalog .vscode - zawierający plik settings.json niezbędny do połącznia z bazą
2. Pliki csv służące jako baza do generowania danych zawartych w bazie:
    - planets_example_names.csv - lista przykładowych nazw planet zaczerpnięte z mitologii i łaciny
    - system_example_names.csv - lista przykłądowych nazw dla układów planetarnych zaczerpnięte od imion bogów, mitologii i łaciny
    - galaxies_example_names.csv - lista przykładowcyh nazw dla galaktyk
3. Pliki csv zawierające wygenerowane dane gotowe do dodania do bazy:
    - booking_final.csv
    - client_contact_final.csv
    - client_final.csv
    - cost_final.csv
    - emergency_contact_final.csv
    - employee_final.csv
    - galaxy_final.csv
    - planet_final.csv
    - planet_system_final.csv
    - rocket_final.csv
    - transaction_final.csv
    - trip_employee_final.csv
    - trip_final.csv
    - trip_type_final.csv
4. Skrypty Jupyter Notebook:
    - data_creation.ipynb - skrypt odpowiedzialny za generowanie danych do plików csv
    - data_insertion.ipynb - skrypt odpowiedzialny za wstawianie danych z pliów csv do bazy danych.
    - distance_calculation_func.ipynb - skrypt zawierający funkcję liczenia odległości
5. project.session.sql - plik SQL odpowiedzialny za stworzenie pustych tabel w bazie danych
6. diagram.erd - plik zawiera schemat bazy danych w formacie ERD
7. Katalog Raport_Bazy_Danych
    - Raport_Bazy_Danych.qmd - plik zawierający kod źródłowy R generujący raport
    - Raport_Bazy_Danych.html - plik zawierający raport
8. Katalog Dokumentacja, któy zawiera:
    - dokumentacja.pdf - dokumentacja projektu w formacie PDF
    - diagram.png - graficzny schemat bazy danych

4 Kolejność i sposób uruchamiania

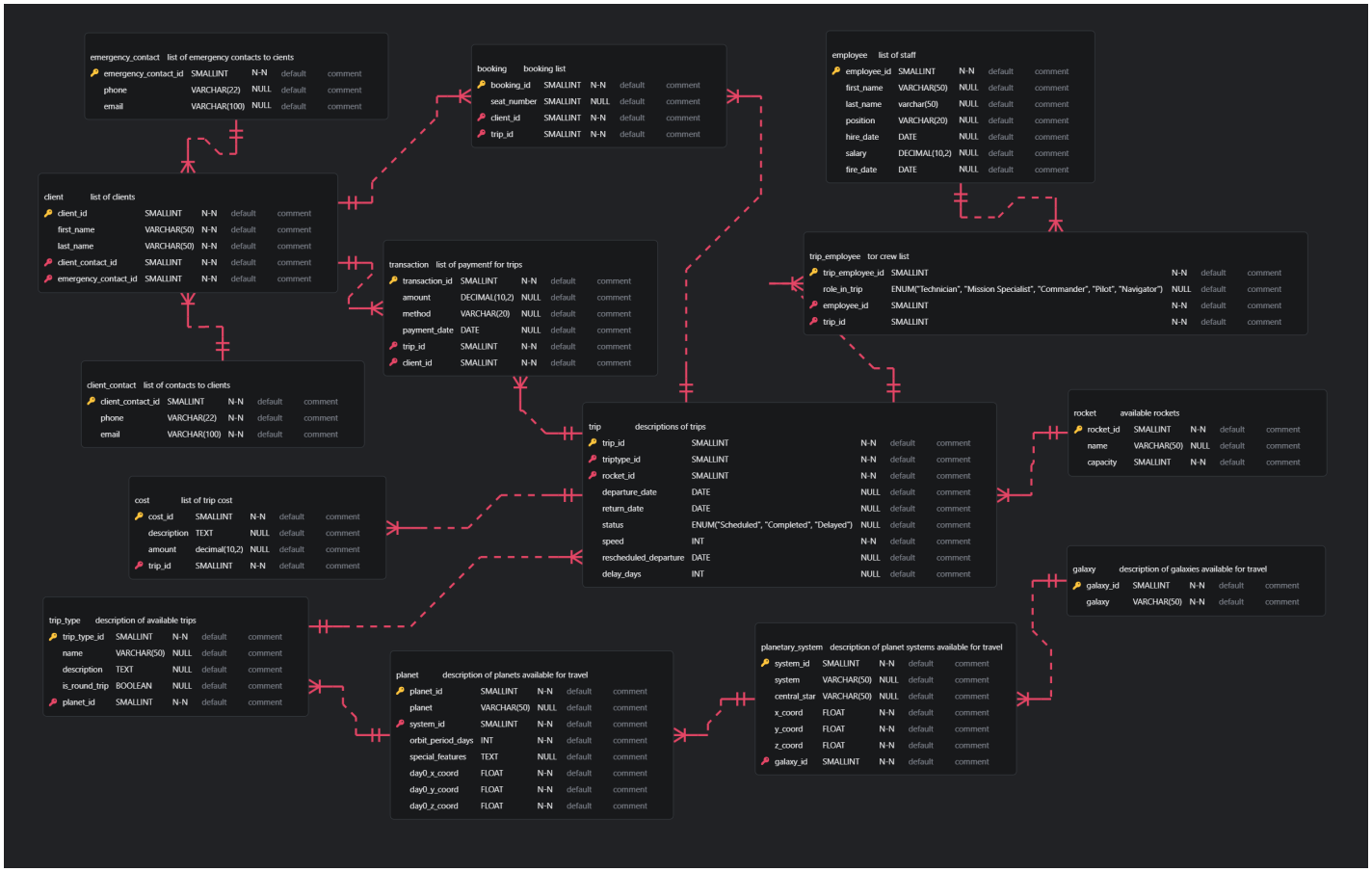
Przed próbą uruchomienia bazy danych należy upewnić się, że połączenie z interentem jest aktywne. Kojenym krokiem jest sprawdzenie czy posiada się Pythona w wersji co najmniej 3.9.12 oraz R 4.4.1. Trzeba się upewnić, że w posiada się wszystkie wymagane do projektu pliki wymienione w Spisie plików (patrz pkt 3). W celu uruchomienia projektu należy uruchomić następujące pliki w podanej kolejności.

1. data_creation.ipynb - generowanie danych
2. projekt.session.sql - tworzenie tabel w bazie
3. data_insertion - wypełnianie bazy odpowiednimi danymi
4. Raport_Bazy_Danych.qmd - plik do generowania raportu, należy z niego wyrenderować plik HTML

5 Schemat bazy danych

5.1 Graficzny schemat bazy danych

Na poniższym rysunku znajduje się schemat bazy danych pzedstawiony za pomoca diagramu ERD



Rysunek 1: Schemat bazy danych

Na schemacie 1) przy niektórych nazwach kolumn w tabelach znajdują się znaczniki kluczy. Żółty klucz oznacza klucz główny danej tabeli, a klucz czerwony - klucz obcy.

5.2 Opis tabel

W skład bazy danych wchodzą następujące tabele:

1. booking
   - booking_id - unikalny identyfikator rezerwacji;
   - seat_number - numer przypisanego miejsca;
   - client_id - identyfikator klienta dokonującego rezerwację;
   - trip_id - identyfikator wycieczki objętej rezerwacją;

   Zależności funkcyjne:

   $$\texttt{booking\_id} \rightarrow \texttt{seat\_number, client\_id, trip\_id}$$

2. transaction
   - transaction_id - unikalny identyfikator transakcji;
   - amount - kwota zapłaty;
   - method - metoda płatności;
   - payment_date - dokładna data płatności;
   - trip_id - identyfikator wycieczki;
   - client_id - identyfikator klienta dokonującego transakcji;

   Zależności funkcyjne:

   $$\texttt{transaction\_id} \rightarrow \texttt{amount, method, payment\_date, trip\_id, client\_id}$$

3. client
   - client_id - unikalny identyfikator klienta;
   - first_name - imię klienta;
   - last_name - nazwisko klienta;
   - client_contact_id - identyfiaktor danych kontaktowych klienta;
   - emergency_contact_id - identyfikator kontaktu awaryjnego klienta;

   Zależności funkcyjne:

   $$\texttt{client\_id} \rightarrow \texttt{first\_name, last\_name, client\_contact\_id, emergency\_contact\_id}$$

4. emergency_contact
   - emergency_contact_id - unikalny identyfikator kontaktu awaryjnego;
   - phone - numer telefonu do kontaktu awaryjnego;
   - email - adres e-mail do kontaktu awaryjnego;

   Zależności funkcyjne:

   $$\texttt{emergency\_contact\_id} \rightarrow \texttt{phone, email}$$

5. client_contact
   - client_contact_id - unikalny indentyfikator kontaktu do klienta;
   - phone - numer telefonu klienta;
   - email - adres e-mail klienta;

Zależności funkcyjne:

$$\texttt{client\_contact\_id} \rightarrow \texttt{phone, email}$$

6. cost

   - cost_id - unikalny indentyfikator kosztu;
   - description - opis czego dotyczy koszt;
   - amount - wartość kosztu;
   - trip_id - identyfikator wycieczki, której dotyczy koszt;

   Zależności funkcyjne:

   $$\texttt{cost\_id} \rightarrow \texttt{description, amount, trip\_id}$$

7. employee

   - employee_id - unikalny indentyfikator pracownika;
   - first_name - imię pracownika;
   - last_name - nazwisko pracownika;
   - position - stanowsiko w firmie;
   - hire_date - data zatrudnienia;
   - salary - wynagrodzenie pracownika;
   - fire_date - data zwolnienia;

   Zależności funkcyjne:

   $$\texttt{employee\_id} \rightarrow \texttt{first\_name, last\_name, position, hire\_date, salary, fire\_date}$$

8. trip

   - trip_id - unikalny identyfikator wycieczki;
   - triptype_id - identyfikator typu wycieczki;
   - rocket_id - identyfikator rakiety użytej do podróży;
   - departure_date - data startu;
   - return_date - data powrotu;
   - status - status wycieczki;
   - speed - prędkość lotu;
   - rescheduled_departure – przełożona data wylotu;
   - delay_days – liczba dni opóźnienia;

   $$\texttt{trip\_id} \rightarrow \texttt{triptype\_id}, \texttt{rocket\_id}, \texttt{departure\_date}, \texttt{return\_date}, \texttt{status}, \texttt{speed}, \texttt{rescheduled\_departure}, \texttt{delay\_days}$$

9. galaxy

   - galaxy_id - unikalny identyfikator galaktyki;
   - galaxy - nazwa galaktyki;

   Zależności funkcyjne:

   $$\texttt{galaxy\_id} \rightarrow \texttt{galaxy}$$

10. trip_emloyee

    - tripemployee_id - uniklany identyfikator przypisania pracownika do konkretnej wycieczki;
    - role_in_trip - rola pracownika w wycieczce
    - employee_id - identyfikator pracownika przypisanego do wycieczki;
    - trip_id - identyfikator wycieczki, do której przypisano pracownika

    Zależności funkcyjne:

    $$\texttt{tripemployee\_id} \rightarrow \texttt{role\_in\_trip, employee, trip\_id}$$

11. rocket

    - rocket_id - uniklany identyfikator rakiety;
    - name - nazwa rakiety;
    - capacity - liczba miejsc w rakiecie;
    - manufacturer - producet rakiety;
    - max_range - maksymalny zasięg rakiety;

    Zależności funkcyjne:

    $$\texttt{rocket\_id} \rightarrow \texttt{name, capacity, manufacturer, max\_range}$$

12. trip_type

    - tripe_type_id - uniklany identyfikator typu wycieczki;
    - name - typ wycieczki;
    - description - opis typu wycieczki;
    - is_round_trip - czy wycieczka jest w obie strony;
    - planet_id - identyfikator planety docelowej wycieczki;

    Zależności funkcyjne:

    $$\texttt{trip\_type\_id} \rightarrow \texttt{name, description, is\_round\_trip, planet\_id}$$

13. planet

    - planet_id - uniklany identyfikator planety;
    - planet - nazwa planety;
    - system_id - identyfikator układu planetarnego, w którym znajduje się dana planeta;
    - orbit_period_days - czas obiegu wokół gwiazdy;
    - special_features - cechy szczególne planety;
    - day0_x_coord - współrzędna X pierwszego dnia obiegu;
    - day0_y_coord - współrzędna Y pierwszego dnia obiegu;
    - day0_z_coord - współrzędna Z pierwszego dnia obiegu;

    Zależności funkcyjne:

    $$\texttt{planet\_id} \rightarrow \texttt{planet, system\_id, orbit\_period\_days, special\_features, day0\_x\_coord, day0\_y\_coord, day0\_z\_coord}$$

14. planetary_system

- system_id - unikalny identyfikator układu planetarnego;
- system - nazwa systemu;
- central_star - nazwa gwiazdy centralnej;
- x_coord - współrzędna X centrum układu;
- y_coord - współrzędna Y centrum układu;
- z_coord - współrzędna Z centrum układu;
- galaxy_id - identyfikator galaktyki, w której znajduje się układ.

Zależności funkcyjne:

`system_id` → `system, central_star, x_coord, y_coord, z_coord, galaxy_id`

5.3 Opis relacji między tabelami

Pomiędzy tabelami występują następujące relacje:

- **trip i trip_employee**

  Relacja 1:n - jedna wycieczka może mieć przypisanych wielu pracowników.

  Klucz główny tabeli `trip` (`trip_id`) występuje jako klucz obcy w tabeli `trip_employee`.

- **employee i trip_employee**

  Relacja 1:n - jeden pracownik może uczestniczyć w wielu wycieczkach.

  Klucz główny tabeli `employee` (`employee_id`) występuje jako klucz obcy w tabeli `trip_employee`.

- **trip i booking**

  Relacja 1:n - jedna wycieczka może mieć wiele rezerwacji.

  Klucz główny tabeli `trip` (`trip_id`) występuje jako klucz obcy w tabeli `booking`.

- **client i booking**

  Relacja 1:n - jeden klient może dokonać wielu rezerwacji.

  Klucz główny tabeli `client` (`client_id`) występuje jako klucz obcy w tabeli `booking`.

- **client i transaction**

  Relacja 1:n - jeden klient może dokonać wielu transakcji.

  Klucz główny tabeli `client` (`client_id`) występuje jako klucz obcy w tabeli `transaction`.

- **trip i transaction**

  Relacja 1:n - jedna wycieczka może być opłacona w wielu transakcjach.

  Klucz główny tabeli `trip` (`trip_id`) występuje jako klucz obcy w tabeli `transaction`.

- **client i client_contact**

  Relacja 1:1 - jeden klient ma jeden kontakt.

  Klucz główny tabeli `client_contact` (`client_contact_id`) występuje jako klucz obcy w tabeli `client`.

- **client i emergency_contact**

  Relacja 1:1 - jeden klient ma przypisaną jedną osobę kontaktową w nagłych wypadkach.

  Klucz główny tabeli `emergency_contact` (`emergency_contact_id`) występuje jako klucz obcy w tabeli `client`.

- **trip i cost**

  Relacja 1:n - jedna wycieczka może mieć wiele kosztów.

  Klucz główny tabeli `trip` (`trip_id`) występuje jako klucz obcy w tabeli `cost`.

- **trip i rocket**

  Relacja n:1 - wiele wycieczek może być przypisanych do jednej rakiety.

  Klucz główny tabeli `rocket` (`rocket_id`) występuje jako klucz obcy w tabeli `trip`.

- **trip i triptype**

  Relacja n:1 - wiele wycieczek może mieć ten sam typ.

  Klucz główny tabeli `triptype` (`triptype_id`) występuje jako klucz obcy w tabeli `trip`.

- **triptype i planet**

  Relacja n:1 - wiele typów wycieczek może dotyczyć jednej planety.

  Klucz główny tabeli `planet` (`planet_id`) występuje jako klucz obcy w tabeli `triptype`.

- **planet i planetary_system**

  Relacja n:1 - wiele planet należy do jednego układu planetarnego.

  Klucz główny tabeli `planetary_system` (`system_id`) występuje jako klucz obcy w tabeli `planet`.

- **planetary_system i galaxy**

  Relacja n:1 - wiele układów planetarnych znajduje się w jednej galaktyce.

  Klucz główny tabeli `galaxy` (`galaxy_id`) występuje jako klucz obcy w tabeli `planetary_system`.

5.4 Uzasadnienie, że baza jest EKNF

Aby baza danych była uznawana za bazę znormalizwoaną musi spełniać następujące warunki:

1. Forma normlana 1NF - *dane atomowe, brak duplikatóww tabeli*
   Uzasadnienie: Każdy atrybut posiad pojedyncze, nierozkładalne wartości oraz każda tebla posiada jednoznaczny klucz główny.

2. Forma normlana 2NF - *brak częściowych zależności od kluczy złożonych*
   Uzasadnienie: W bazie nie występują klucze złożone, a wszystkie atrybuty niekluczowe są zależne od całego klucza głównego.

3. Forma normalna 3NF - *brak zależności od innych atrybutó niekluczowych*
   Uzadanienie: Nie ma przypadków, w których atrybut niekluczowy zależy od innego niekluczowego atrybutu.

4. EKNF (rozszerzona 3NF) - *Każda nietrywialna zależność funkcyjna ma postać, w której lewa strona jest nadkluczem lub prawa strona zawiera atrybut elementarny*
   Uzasadnienie: Wszystkie zależności funkcyjne w bazie spełniają ten warunek.

Zatem wszystkie warunki są spełnione, czyli baza jest EKNF

6 Podsumowanie projketu

Największym wyzwaniem podczas realizacji projektu było zaplanowanie struktury bazy danych oraz logiczne powiązanie między tabelami. Ten etap był niezwykle ważny, ponieważ dobrze przemyślana struktura bazy stanowiła fundament dla dalszych działań i zapewniała poprawność działania całego systemu.

Trudnością było również wprowadzenie ruchu orbitalnego oraz zrozumienie i zastosowanie praw logicznych, które są z nim powiązane. Wymagało to nie tylko dogłębnej analizy teoretycznej, ale również przemyślenia, jak te zasady odzwierciedlić w strukturze bazy i relacjach między danymi.

Mimo trudności, projekt pozwolił nam znacząco poszerzyć wiedzę z zakresu baz danych oraz rozwinąć kompetencje związane z pracą zespołową, takie jak planowanie zadań i efektywna komunikacja. Zdobyte doświadczenia na pewno będą przydatne w dalszej nauce oraz przyszłej karierze zawodowej.