Dokumentacja projektu SPACE-U

Maja Wiącek, Jakub Lis, Paweł Skrzypczyński, Tymon Świtalski

bazy danych 2025

Spis treści

1 Użyte technologie

1.1 Projekt i utworzenie schematu

Przy konstruowaniu projektu bazy danych korzystaliśmy z kartki papieru i długopisu do rozplanowania potrzebnych tabel i zależności pomiędzy nimi. Do utworzenia schematu bazy danych użyliśmy rozszerzenia *ERD editor* w programie *Visual Studio Code*.

1.2 Skryptowe wypełnienie bazy

W celu skryptowego wypełniania bazy utworzono plik `Generator.py`. Skrypt wykorzystuje **Python** jako język programowania. Wykorystane biblioteki to:

- `mysql.connector` - biblioteka do łączenia się z bazą danych oraz wykonywania na niej operacji.
- `datetime` - Biblioteka do pracy z datami i czasami.
- `faker` - Biblioteka do generowania realistycznie wyglądających danych (np. imion, adresów, numerów telefonu).
- `random` - do generowania losowych wartości.
- `decimal` - do precyzyjnych obliczeń dziesiętnych.
- `math` - do operacji matematycznych.

1.3 Analiza danych

Podczas analizy danych skorzystaliśmy z języka R. Użyliśmy następujących bibliotek:

- `RMariaDB` - do połączenia oraz obsługi bazy danych (m. in. funkcje `dbConnect` oraz `bGetQuery`);
- `ggplot2` - do wizualizacji uzyskanych wyników.

1.4 Raport

Przy pisaniu raportu skorzystaliśmy z oprogramowania `Sweave`, dzięki któremu mogliśmy zintegrować analizę danych wykonaną w języku R z `LaTeX`.

---

2 Pliki i ich zawartość

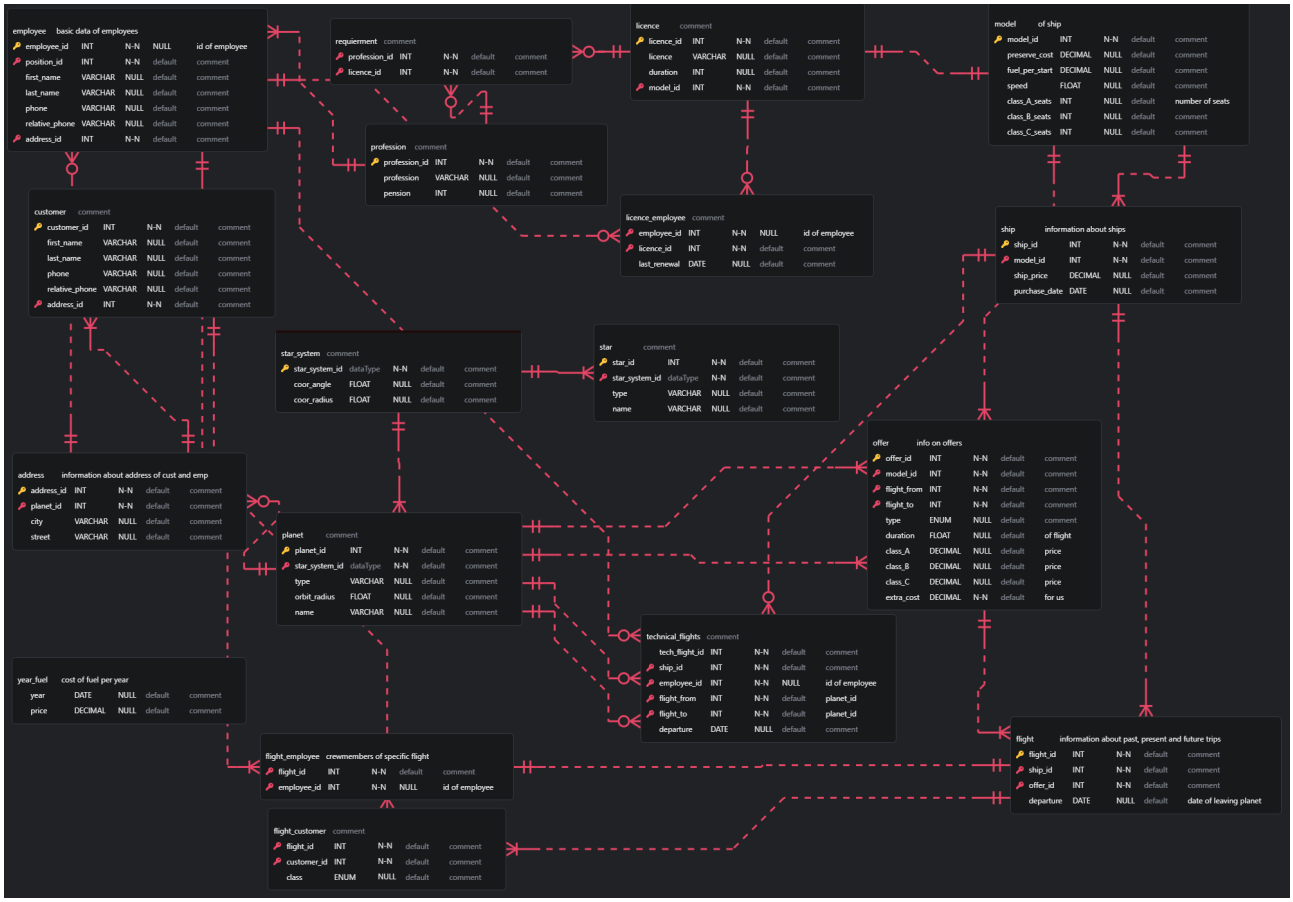
- `struktura.erd.json` - schemat bazy danych w formacie używanym przez ERD editor.
- `create_database.sql` - kod SQL generujący pustą bazę danych, wygenerowny na podstawie struktury w `struktura.erd.json`.
- `generator.py` - skrypt odpowiadający za generowanie danych w bazie.
- `raport.Rnw` - analiza danych i generator raportu do pliku pdf.

---

3 Kolejność i sposób uruchamiania plików

1. Należy zainstalować MariaDB Server przy pomocy instalatora.
2. Instalator pozwala na konfigurację - ustawienie hasła, wybór portu.
3. W słowniku `DB_CONFIG` należy zmienić dane na odpowiednie dla naszej konfiguracji.
4. Uruchamiamy plik `create_database.sql` w `SQL`.
5. Następnie uzupełniamy bazę danych za pomocą pliku `generator.py` (przed uruchomieniem pliku należy zmienić dane do połączenia z bazą danych na takie same, jak podczas konfiguracji przy instalacji MariaDB Server).
6. Ostatnim krokiem jest uruchomienie pliku `raport.Rnw` (ponownie, jak w przypadku generatora, należy zmienić dane do połączenia).

4 Schemat projektu bazy danych



Rysunek 1: Schemat projektu bazy danych

5 Zależności funkcyjne relacji

Globalne oznaczenia dla tej sekcji są następujące: duże litery stoją za kolejnymi atrybutami danej relacji według porządku w jakim zostają w tej relacji tworzone. Zbiór zależności funkcyjnych relacji X jest oznaczony przez Σ_X.

W każdym zbiorze opisane zostaną wszystkie zależności poza pierwszymi. Pierwsze zależności są wynikiem posiadania unikatowego numeru ID lub w inny sposób wynika wprost, że równość tych atrybutów implikuje równość na wszystkich atrybutach.

Kolejne zbiory zależności funkcyjnych z opisami wyglądają, jak następuje:

$\Sigma_{address} = \{A \to ABCD\}$

$\Sigma_{customer} = \{A \to ABCDEF, E \to A\}$ - równość *address_id* pociąga równość *customer_id* (każde ID adresu jest unikatowe dla danego klienta

$\Sigma_{employee} = \{A \to ABCDEFG, E \to A\}$ - równość *address_id* implikuje równość *employee_id* (każde ID adresu jest unikatowe dla danego pracownika

$\Sigma_{flight} = \{A \to ABCD\}$

$\Sigma_{flight\_customer} = \{AB \to ABC\}$

$\Sigma_{flight\_employee} = \{AB \to AB\}$

$\Sigma_{licence} = \{A \to ABCD, B \to A\}$ - nazwa licencji *licence* pociąga ID licencji (*licence_id*)
$\Sigma_{licence\_employee} = \{AB \to ABC\}$
$\Sigma_{model} = \{A \to ABCDEFG\}$
$\Sigma_{offer} = \{A \to ABCDEFGHJ\}$
$\Sigma_{planet} = \{A \to ABCDE\}$
$\Sigma_{profession} = \{A \to ABC, B \to A\}$ - nazwa profesji *profession* pociąga jej ID (*profession_id*)
$\Sigma_{requirement} = \{AB \to AB\}$
$\Sigma_{ship} = \{A \to ABCD\}$
$\Sigma_{star} = \{A \to ABCD\}$
$\Sigma_{star\_system} = \{A \to ABC\}$
$\Sigma_{technical\_flights} = \{A \to ABCDEF, CF \to A\}$ - techniczny lot (*tech_flight_id*) jest jednoznacznie wyznaczony przez ID pracownika (*employee_id*) oraz datę wylotu (*departure*)
$\Sigma_{year\_fuel} = \{A \to AB\}$

---

6 Uzasadnienie, że baza jest w EKNF

W każdej relacji wszystkie (nietrywialne) zależności funkcyjne albo zaczynają się od klucza kandydującego, albo kończą na atrybucie elementarnym. Przypatrzmy się bliżej każdej z relacji.

Relacje *customer*, *employee*, *licence*, *profession* oraz *technical_flight* są jedynymi, w których występują dwie zależności funkcyjne. Dostrzeżmy, że we wszystkich klucz kandydujący (a nawet elementarny) jest złożony z jednego atrybutu - A (mniejszego nadklucza być nie może, a elementarność wynika wprost z zależności $A \to X$, gdzie X jest zbiorem atrybutów każdej z tych relacji). Stąd wynika spełnienie warunku bycia w postaci EKNF przez pierwsze zależności funkcyjne we wszystkich relacjach (zaczynają się bowiem od klucza kandydującego). Przypatrując się drugim zależnościom funkcyjnym, widzimy, że dla wszystkich relacji kończą się na atrybucie A, który, jako (jedyny) element klucza elementarnego, jest atrybutem elementarnym. Zatem zależności te kończą się na atrybucie elementarnym.

Zauważmy, że w relacjach *flight_employee*, *requirement* występują jedynie zależności trywialne, zatem warunek bycia EKNF jest oczywiście spełniony.

Pozostałe zbiory zależności funkcyjnych składają się z jednej zależności. W każdym przypadku zaczyna się od klucza kandydującego (a nawet elementarnego) danej relacji.

---

7 Jakie napotkaliśmy trudności?

Najtrudniejsze podczas realizacji projektu było sensowne wygenerowanie bazy danych. Trzeba było zadbać szczegółowo o logikę i chronologie, tak aby obiekty nie były w dwóch miejscach na raz. Po wygenerowaniu bazy okazało się że dane nie są wystarczająco zróżnicowane i realistyczne, by przeprowadzić na nich analizę. Konieczne było modyfikowanie generatora, tak aby dawał lepsze wyniki.