

# Dokumentacja „Geeks & Dragons”

Urszula Spik, Natalia Jelito, Maciej Karczewski, Magdalena Szymkowiak

2023-06-30

## 1 Spis użytych technologii

Do stworzenia projektu zostały użyte następujące technologie:

1. Tworzenie bazy danych:
  - Python z bibliotekami:
    - pandas,
    - numpy,
    - random,
    - datetime,
    - urllib.parse,
    - sqlalchemy,
    - itertools,
    - faker,
  - MariaDB,
2. Analiza danych:
  - R z bibliotekami:
    - RMariaDB,
    - ggplot2,
    - dplyr,
    - dbplyr,
    - tidyr,
    - kableExtra,
    - emojiFont,
    - stringr,
    - colorspace,
    - wesanderson.
3. Raport i dokumentacja:
  - Rmarkdown
4. Tworzenie schematu baz danych:
  - Miro

## 2 Lista plików i opis ich zawartości

W folderze **bazy\_danych** znajdują się pliki tworzące cały projekt. Struktura zawartości folderu wygląda następująco

```
bazy_danych
├── data
│   ├── address.csv
│   ├── english_first_name.csv
│   ├── english_last_name.csv
│   └── games.csv
```

```

├── polish_female_last_name.csv
├── polish_female_name.csv
├── polish_male_last_name.csv
├── polish_female_name.csv
├── dokumentacja.pdf
├── Raport
│   ├── raport.Rmd
│   └── raport.pdf
├── src
│   ├── __init__.py
│   ├── create.sql
│   └── generate.py
├── main.py
└── requirements.txt

```

W folderze *data* znajdują się pliki *.csv*, których użyliśmy do generowania adresów, imion, nazwisk zarówno klientów sklepu jak i też samych pracowników. Użyliśmy również ich do generowania gier, które można wypożyczyć albo kupić. Źródła danych:

- *address.csv* - <https://bip.um.wroc.pl/artukul/389/50007/aktualne-wykazy-nazw-ulic-i-adresow>
- *english\_first\_name.csv* - <https://catalog.data.gov/dataset/popular-baby-names/resource/02e8f55e-2157-4cb2-961a-2aabb75cbc8b>
- *english\_last\_name.csv* - [https://www.census.gov/topics/population/genealogy/data/2010\\_surnames.html](https://www.census.gov/topics/population/genealogy/data/2010_surnames.html)
- *games.csv* - <https://www.kaggle.com/datasets/andrewmvd/board-games>
- *polish\_female\_last\_name.csv* oraz *polish\_male\_last\_name.csv* - <https://dane.gov.pl/pl/dataset/1681,nazwiska-osob-zyjacych-wystepujace-w-rejestrze-pesel>
- *polish\_female\_name.csv* oraz *polish\_male\_name.csv* - <https://dane.gov.pl/pl/dataset/219,imi ona-nadawane-dzieciom-w-polsce>

Plik *dokumentacja.pdf* jest plikiem zawierającym dokumentację.

W folderze *Raport* znajdują się plik do generowania raportu oraz plik z gotowym już raportem.

Kolejnym folderem jest *src*. W środku znajdują się pliki, które służą do generowania i tworzenia bazy danych. Plik *\_\_init\_\_.py*, potrzebny jest do możliwości importowania pythonowych funkcji z innych plików znajdujących się w tym samym folderze. Plik *create.sql* tworzy nam tabele do bazy danych. Plik *generate.py* zawiera funkcje, przy pomocy których generujemy dane do wypełnienia stworzonej wcześniej bazy danych.

Plik *main.py* służy do wygenerowania tabel z danymi oraz wgrania ich na serwer.

Plik *requirements.txt* zawiera potrzebne pakiety dla Pythona.

### 3 Kolejność i sposób uruchamiania plików, aby uzyskać gotowy projekt

1. Posiadaj Pythona 3.7.9.
2. Zainstaluj potrzebne pakiety komendą `pip install -r requirements.txt`.
3. Uruchom program `main.py python main.py`.
4. Uruchom i skompiluj za pomocą Knit plik `report.rmd`.

### 4 Schemat bazy danych i zależności funkcyjne w tabelach

Klucze główne zaznaczamy przez pogrubienie.

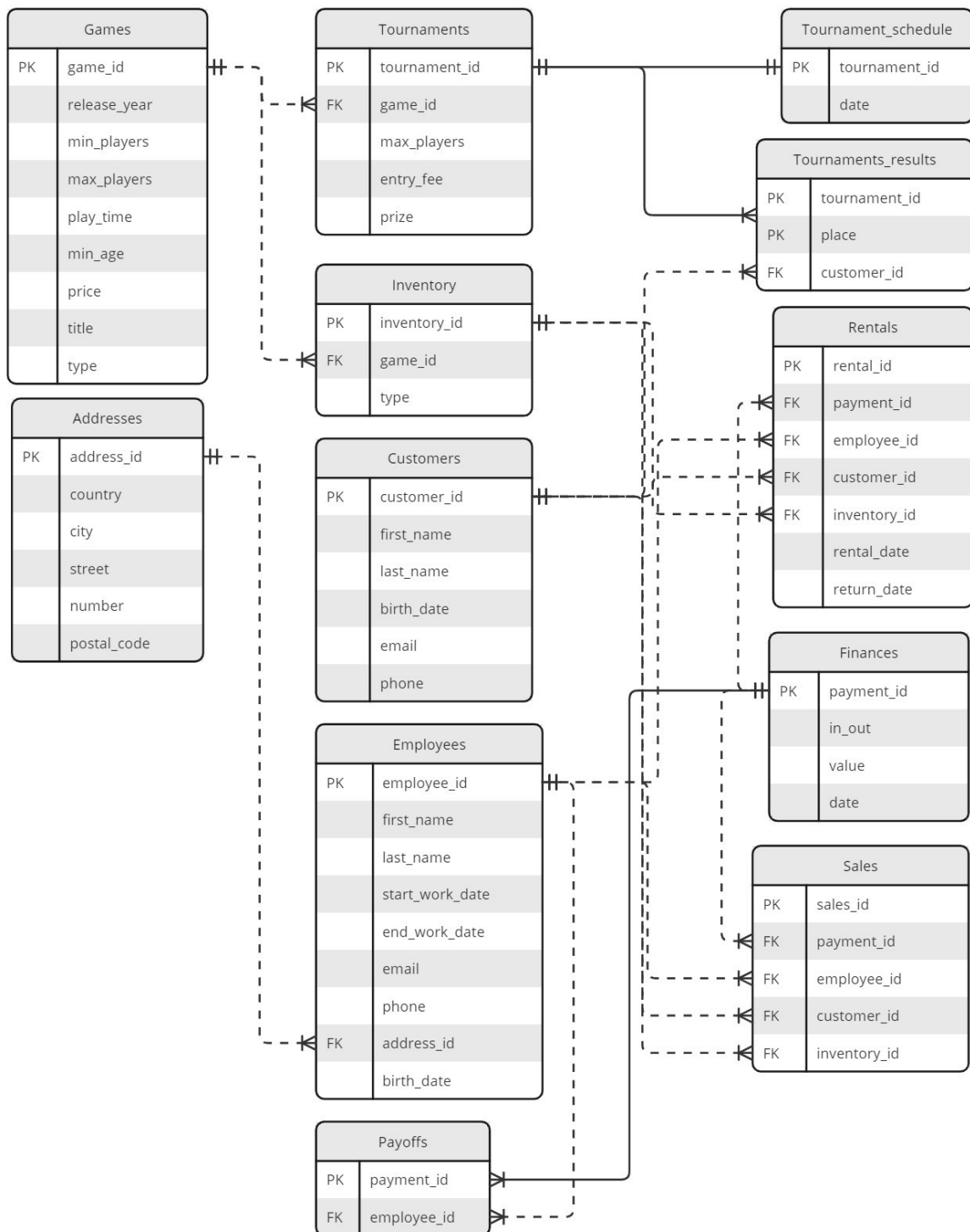


Figure 1: Schemat bazy danych

## 4.1 Tabela Customers

R(customer\_id, first\_name, last\_name, birth\_date, email, phone)

- customer\_id [SMALLINT UNSIGNED] : numer identyfikacyjny klienta
- first\_name [VARCHAR(45)]: imię klienta
- last\_name [VARCHAR(50)]: nazwisko klienta
- birth\_date [DATE]: data urodzenia klienta
- email [VARCHAR(50)]: email klienta
- phone [VARCHAR(12)]: numer telefonu klienta

$$\Sigma = \{ \text{customer\_id} \rightarrow \text{first\_name}, \text{customer\_id} \rightarrow \text{last\_name}, \text{customer\_id} \rightarrow \text{birth\_date}, \\ \text{customer\_id} \rightarrow \text{email}, \text{email} \rightarrow \text{customer\_id}, \text{customer\_id} \rightarrow \text{phone}, \text{phone} \rightarrow \text{customer\_id} \}$$

Informacje o dokładnym kliencie możemy pozyskać przez customer\_id. Dokładną identyfikację klienta zapewnia także jego email lub jego numer telefonu, ponieważ zakładamy, że każdy ma jeden unikalny numer telefonu oraz adres email.

## 4.2 Tabela Employees

R(employee\_id, first\_name, last\_name, start\_work\_date, end\_work\_date, email, phone, address\_id)

- employee\_id [SMALLINT UNSIGNED]: numer identyfikacyjny pracownika
- first\_name [VARCHAR(45)]: imię pracownika
- last\_name [VARCHAR(50)]: nazwisko pracownika
- start\_work\_date [DATE]: data zatrudnienia (rozpoczęcie pracy)
- end\_work\_date [DATE]: data zakończenia pracy
- email [VARCHAR(50)]: adres email pracownika
- phone [VARCHAR(12)]: numer telefonu pracownika
- address\_id [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli Addresses : numer identyfikacyjny adresu.

$$\Sigma = \{ \text{employee\_id} \rightarrow \text{first\_name}, \text{employee\_id} \rightarrow \text{last\_name}, \text{employee\_id} \rightarrow \text{start\_work\_date}, \\ \text{employee\_id} \rightarrow \text{start\_work\_date}, \text{employee\_id} \rightarrow \text{email}, \text{email} \rightarrow \text{employee\_id}, \text{employee\_id} \rightarrow \text{phone}, \text{phone} \rightarrow \text{employee\_id} \}$$

Pracownik jest jednoznacznie zdefiniowany przez employee\_id. Tak jak w przypadku klientów, pracownika też możemy rozpoznać po jego numerze telefonu lub email.

## 4.3 Tabela Addresses

R(address\_id, country, city, street, number, postal\_code)

- address\_id [SMALLINT UNSIGNED]: numer identyfikacyjny adresu.
- country [VARCHAR(50)]: państwo
- city [VARCHAR(50)]: miasto
- street [VARCHAR(50)]: ulica
- number [VARCHAR(10)]: numer budynku
- postal\_code [VARCHAR(10)]: kod pocztowy.

$$\Sigma = \{ \text{address\_id} \rightarrow \text{country}, \text{address\_id} \rightarrow \text{city}, \text{address\_id} \rightarrow \text{street}, \text{address\_id} \rightarrow \text{number}, \\ \text{address\_id} \rightarrow \text{postal\_code}, \text{country} + \text{city} + \text{street} + \text{number} + \text{postal\_code} \rightarrow \text{address\_id} \}$$

Adres jest dokładnie, jednoznacznie wyznaczony dla danego `adress_id`. Pozostałe pola tylko razem wskażą nam dokładny adres.

#### 4.4 Tabela Games:

R(**game\_id**,title, release\_year,min\_players,max\_players,play\_time,min\_age,type,price)

- `game_id` [SMALLINT UNSIGNED]: numer identyfikacyjny gry
- `title` [VARCHAR(300)]: oryginalny tytuł gry
- `release_year` [DATE]: rok wydania gry
- `min_players` [TINYINT UNSIGNED]: minimalna liczba graczy
- `max_players` [TINYINT UNSIGNED]: maksymalna liczba graczy
- `play_time` [SMALLINT UNSIGNED]: szacowany czas gry w minutach
- `min_age` [TINYINT UNSIGNED]: wiek minimalny do zagrania w grę
- `type` [VARCHAR(18)]: typ gry tzn czy jest to gra czy rozszerzenie do gry
- `price` [DECIMAL]: cena za kupno gry

$$\Sigma = \{\text{game\_id} \rightarrow \text{title}, \text{game\_id} \rightarrow \text{release\_year}, \text{game\_id} \rightarrow \text{min\_players}, \text{game\_id} \rightarrow \text{max\_players}, \\ \text{game\_id} \rightarrow \text{play\_time}, \text{game\_id} \rightarrow \text{min\_age}, \text{game\_id} \rightarrow \text{type}, \text{game\_id} \rightarrow \text{price}\}$$

Gra jest wyznaczona jednoznacznie przez `game_id`. Połączenie wszystkich pozostałych informacji nie musi nam jednoznacznie określić gry ( Tytuły nie muszą być unikatowe)

#### 4.5 Tabela Inventory

R(**inventory\_id**, game\_id,type)

- `inventory_id` [SMALLINT UNSIGNED]: numer identyfikacyjny inwentarza
- `game_id` [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli Games): numer identyfikacyjny grę
- `type` [CHAR(1)]: przeznaczenie gry na magazynie ( czy na sprzedaż, czy na wypożyczenie, czy na turniej)

$$\Sigma = \{\text{inventory\_id} \rightarrow \text{game\_id}, \text{inventory\_id} \rightarrow \text{type}\}$$

Gra w magazynie jest wyznaczona jednoznacznie przez `inventory_id`. Reszta informacji nie pozwala nam tego określić ponieważ możemy mieć kilka takich samych tytułów o takim samym przeznaczeniu.

#### 4.6 Tabela Tournaments

R(**tournament\_id**, game\_id,max\_players,entry\_fee,prize)

- `tournament_id` [SMALLINT UNSIGNED]: numer identyfikacyjny turnieju
- `game_id` [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli Games): numer identyfikacyjny grę w jaką się gra na turnieju
- `max_players` [TINYINT UNSIGNED]: maksymalna liczba graczy na turnieju
- `entry_fee` [DECIMAL]: wpisowe
- `prize` [DECIMAL]: nagroda za turniej zamiast

$$\Sigma = \{\text{tournament\_id} \rightarrow \text{game\_id}, \text{tournament\_id} \rightarrow \text{max\_players}, \\ \text{tournament\_id} \rightarrow \text{entry\_fee}, \text{tournament\_id} \rightarrow \text{prize}\}$$

Turniej jest jednoznacznie wyznaczony jedynie przez `tournament_id`. Zakładamy, że nagroda nie jest bezpośrednio zależna od wpisowego i liczby graczy.

#### 4.7 Tabela `Tournament_schedule`

$R(\text{tournament\_id}, \text{date})$

- `tournament_id` [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli `Tournaments`): numer identyfikacyjny turnieju
- `date` [DATE]: data turnieju

$$\Sigma = \{\text{tournament\_id} \rightarrow \text{date}\}$$

Tabela łączy turniej z jego datą. W jeden dzień może odbywać się więcej niż 1 turniej

#### 4.8 Tabela `Tournaments_results`

$R(\text{tournament\_id}, \text{place}, \text{customer\_id})$

- `tournament_id` [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli `Tournaments`): numer identyfikacyjny turnieju
- `place` [SMALLINT UNSIGNED]: miejsce zajęte w turnieju przez gracza (klienta)
- `customer_id` [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli `Customers`): numer identyfikacyjny klienta

$$\Sigma = \{\text{tournament\_id} + \text{place} \rightarrow \text{client\_id}\}$$

Identyfikator turnieju w połączeniu z miejscem wyznacza nam jednoznacznie osobę, która te miejsce zajęła w turnieju. Zakładamy, że w turniejach nie ma miejsc *ex aequo*.

#### 4.9 Tabela `Finances`

$R(\text{payment\_id}, \text{in\_out}, \text{value}, \text{date})$

- `payment_id` [MEDIUMINT UNSIGNED]: numer identyfikacyjny płatności
- `in_out` [TINYINT]: czy jest to płatność czy przychód
- `value` [DECIMAL]: wartość płatności
- `date` [DATE]: data płatności

$$\Sigma = \{\text{payment\_id} \rightarrow \text{in\_out}, \text{payment\_id} \rightarrow \text{value}, \text{payment\_id} \rightarrow \text{date}\}$$

Tylko `payment_id` wyznacza nam jednoznacznie przepływ pieniężny.

#### 4.10 Tabela `Payoffs`: $R(\text{payment\_id}, \text{employee\_id})$

- `payment_id` [MEDIUMINT UNSIGNED] (klucz obcy odwołujący się do tabeli `Finances`): numer identyfikacyjny płatności
- `employee_id` [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli `Employees`): numer identyfikacyjny pracownika

$$\Sigma = \{\text{payment\_id} \rightarrow \text{employee\_id}\}$$

Każda wypłata jest jednoznacznie zdefiniowana przez `payment_id` i ma przypisanego pracownika. Relacją w drugą stronę nie zachodzi ponieważ pracownik może mieć kilka wypłat.

## 4.11 Tabela Rentals

R(**rental\_id**, payment\_id, employee\_id, customer\_id, inventory\_id, rental\_date, return\_date)

- rental\_id [SMALLINT UNSIGNED]: numer identyfikacyjny wypożyczenia
- payment\_id [MEDIUMINT UNSIGNED] (klucz obcy odwołujący się do tabeli Finances) : numer identyfikacyjny płatności
- employee\_id [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli Employees): numer identyfikacyjny pracownika
- customer\_id [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli Customers): numer identyfikacyjny klienta
- inventory\_id [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli Inventory): numer identyfikacyjny inwentarza
- rental\_date [DATE]: data wypożyczenia gry
- return\_date [DATE]: data oddania gry

$$\Sigma = \{\text{rental\_id} \rightarrow \text{payment\_id}, \text{rental\_id} \rightarrow \text{employee\_id}, \text{rental\_id} \rightarrow \text{customer\_id}, \text{rental\_id} \rightarrow \text{inventory\_id}, \\ \text{rental\_id} \rightarrow \text{rental\_date}, \text{rental\_id} \rightarrow \text{return\_date}, \text{payment\_id} \rightarrow \text{rental\_id}\}$$

Wypożyczenie jest jednoznacznie wyznaczone przez rental\_id, tak jak payment id. Pozostałe pola nie muszą jednoznacznie wyznaczyć wypożyczenia.

## 4.12 Tabela Sales

R(**sales\_id**, payment\_id, employee\_id, customer\_id, inventory\_id)

- sales\_id [SMALLINT UNSIGNED]: numer identyfikacyjny sprzedaży gry
- payment\_id [MEDIUMINT UNSIGNED] (klucz obcy odwołujący się do tabeli Finances) : numer identyfikacyjny płatności
- employee\_id [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli Employees): numer identyfikacyjny pracownika
- customer\_id [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli Customers): numer identyfikacyjny klienta
- inventory\_id [SMALLINT UNSIGNED] (klucz obcy odwołujący się do tabeli Inventory): numer identyfikacyjny inwentarza

$$\Sigma = \{\text{sales\_id} \rightarrow \text{payment\_id}, \text{sales\_id} \rightarrow \text{employee\_id}, \text{sales\_id} \rightarrow \text{customer\_id}, \\ \text{sales\_id} \rightarrow \text{game\_id}, \text{payment\_id} \rightarrow \text{sales\_id}\}$$

Każda sprzedaż ma unikatowy sales\_id. Transakcja sprzedaży może być też jednoznacznie wyznaczona przez payment\_id.

## 5 Postać normalna bazy danych

Jak możemy zauważyć w naszej bazie danych w każdej tabeli wszystkie nietrywialne relacje zaczynają się od nadklucza więc spełnia postać BCNF z czego wynika także że spełnia postać normalną EKNF.

## 6 Co było najtrudniejsze w projekcie?

Największym wyzwaniem w projekcie było poprawne wygenerowanie danych i wrzucenie ich na serwer. Wyzwaniem było także zrozumienie i zaprojektowanie bazy danych w ten sposób by spełniała postać normalną EKNF.