

BAZY DANYCH

# BAZA DANYCH GEEKS & DRAGONS

DOKUMENTACJA

Szymon Malec, 262276  
Adam Wrzesiński, 262317  
Michał Wiktorowski, 262330  
Weronika Zmyślona, 262284

Politechnika Wrocławska  
Wydział Matematyki - Matematyka Stosowana

# Contents

<b>1</b>	<b>Wstęp</b>	<b>2</b>
<b>2</b>	<b>Schemat bazy danych</b>	<b>2</b>
<b>3</b>	<b>Skryptowe wypełnienie bazy</b>	<b>3</b>
3.1	Tabela customers . . . . .	3
3.2	Tabela staff . . . . .	3
3.3	Tabela games_for_sale . . . . .	4
3.4	Tabela games_to_rent . . . . .	4
3.5	Tabela sale . . . . .	5
3.6	Tabela rental . . . . .	5
3.7	Tabela competition . . . . .	5
3.8	Tabela competition_results . . . . .	5
3.9	Połączenie z bazą . . . . .	5
<b>4</b>	<b>Analiza danych i generowanie raportu</b>	<b>5</b>
<b>5</b>	<b>Technologie</b>	<b>6</b>
<b>6</b>	<b>Zarządzanie plikami</b>	<b>6</b>
<b>7</b>	<b>Baza w postaci EKNF</b>	<b>6</b>
<b>8</b>	<b>Podsumowanie</b>	<b>8</b>

# 1 Wstęp

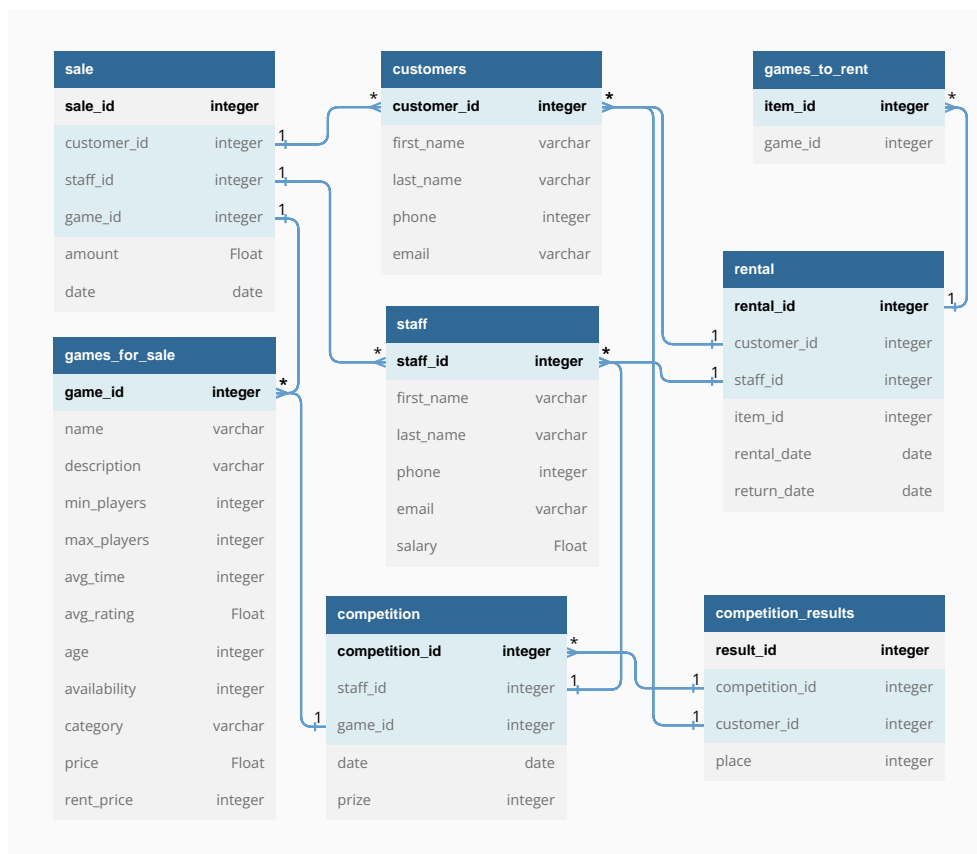
Niniejsza dokumentacja przedstawia proces tworzenia bazy danych sklepu Geeks & Dragons w ramach projektu na zaliczenie z kursu Bazy Danych z 2023 roku. Zadanie polegało na wysymulowaniu danych, jakie mogłyby się pojawić w bazie przykładowego sklepu, mającego w ofercie gry nieelektroniczne, który dodatkowo prowadzi wypożyczalnię gier oraz organizuje turnieje. Projekt został wykonany z wykorzystaniem języka programowania Python.

Pierwszą częścią projektu było wykonanie schematu bazy danych, a następnie wygenerowanie losowych danych. Uzyskane dane należało eksportować do bazy i umieścić na serwerze za pomocą MySQL. Kolejna część polegała na przeprowadzeniu analizy danych z wysymulowanego zbioru i przedstawienie jej w formie raportu. Ponadto został opisany przebieg normalizacji bazy danych do postaci EKNF.

## 2 Schemat bazy danych

Schemat bazy danych (Rysunek 1.) uwzględnia 8 tabel przechowujących następujące informacje:

- **customer** – dane dotyczące klientów sklepu,
- **staff** – dane dotyczące pracowników sklepu,
- **games\_for\_sale** – dane dotyczące gier dostępnych na sprzedaż,
- **games\_to\_rent** – dane dotyczące gier dostępnych do wypożyczenia,
- **sale** – dane na temat sprzedaży gier,
- **rental** – dane na temat wypożyczeń gier,
- **competition** – dane dotyczące organizowanych turniejów,
- **competition\_results** – dane dotyczące osiągniętych wyników przez uczestników turniejów.



Rysunek 1: Schemat bazy danych.

## 3 Skryptowe wypełnienie bazy

W celu przygotowania do wypełnienia bazy losowo wygenerowanymi danymi, zostały stworzone pliki csv odpowiadające poszczególnym tabelom z bazy danych. Oprócz danych generowanych losowo, pojawiają się również dane rzeczywiste, co zostanie opisane w dalszej części tego rozdziału. Na sam koniec, po połączeniu z serwerem, dane zostały zaimportowane do bazy.

### 3.1 Tabela customers

Tabela customers zawiera informacje na temat 1329 klientów, które zorganizowane są w 5 następujących kolumnach:

- **cusomer\_id** – unikalne id dla każdego klienta,
- **first\_name** – imię klienta,
- **last\_name** – nazwisko klienta,
- **phone** – numer telefonu klienta,
- **email** – adres email klienta.

Dane w tabeli zostały wygenerowane w sposób losowy. Wartości w tabeli cusomer\_id wygenerowane zostały na pomocą odpowiedniego przedziału dodatnich liczb całkowitych. Imiona i nazwiska klientów zostały wylosowane z odpowiednim prawdopodobieństwem korzystając z danych zamieszczonych na stronie Głównego Urzędu Statystycznego. W pierwszej kolejności został wygenerowany wektor określający płeć klienta, z odpowiednim prawdopodobieństwem występowania danej płci, na podstawie liczby ludności we Wrocławiu z podziałem na mężczyzn i kobiety. Następnie w zależności od płci zostały przypisane imiona i nazwiska z odpowiednich tabeli danych pobranych ze wspomnianej strony internetowej.

Następna kolumna zawiera losowo generowane numery kontaktowe klientów. Numer w każdym przypadku składa się z połączenia losowo wybranego wyróżnika sieci telefonicznej (2 cyfry) oraz liczby z przedziału (1000000, 9000000). Lista dostępnych wyróżników została stworzona na podstawie informacji dostępnych na stronie Wikipedii. W ten sposób uzyskaliśmy indywidualny 9-cyfrowy numer dla każdego klienta.

Ostatnia kolumna zawiera adresy email, dla których identyfikatory użytkownika są generowane losowo na podstawie imion i nazwisk klientów. Identyfikatory adresów email składają się przykładowo:

- z połączenia imienia i nazwiska lub części nazwiska klienta,
- z połączenia imienia lub nazwiska i losowej liczby z przedziału (100, 10000).

Każdy adres email składa się również z domeny, która była losowana z listy domen stworzonej na podstawie rankingu najpopularniejszych serwisów pocztowych w Polsce dostępnego na stronie interaktywnie.com. Oczywiście, domeny były losowane z odpowiednim prawdopodobieństwem określonym według liczby użytkowników korzystających z wybranych serwisów pocztowych.

### 3.2 Tabela staff

Tabela staff zawiera 6 kolumn z następującymi informacjami na temat 17 pracowników sklepu:

- **staff\_id** – zawiera unikalne id dla każdego pracownika,
- **first\_name** – imię pracownika,
- **last\_name** – nazwisko pracownika,
- **phone** – numer telefonu pracownika,
- **email** – adres email pracownika,
- **salary** – miesięczne wynagrodzenie pracownika (cena brutto).

Dane w pierwszych 4 kolumnach zostały wygenerowane w sposób analogiczny jak w przypadku tabeli customers. Nastąpiła zmiana w generowaniu adresów email dla pracowników. Są one tworzone w jednaki sposób dla każdego pracownika – składają się z połączenia imienia i nazwiska za pomocą kropki oraz odrębnej domeny firmy @dragons.com.

Ostatnia kolumna określa miesięczne wynagrodzenie pracownika, które były losowane z listy konkretnych pensji [4310, 5140, 6530, 7280, 8320], które zostały wybrane na podstawie możliwych zarobków na stanowiskach dotyczących obsługi klienta. Losowanie odbyło się z ustalonym prawdopodobieństwem, zakładającym, że doświadczonych pracowników o większych zarobkach jest mniej.

### 3.3 Tabela games\_for\_sale

Tabela **games\_for\_rent** zawiera podstawowe informacje dotyczące 3723 unikatowych gier nieelektronicznych. Są to kolejno:

- **game\_id** - unikatowy identyfikator gry
- **name** - tytuł gry
- **description** - krótki opis gry
- **min\_players** - minimalna liczba graczy
- **max\_players** - maksymalna liczba graczy
- **avg\_time** - średni czas trwania rozgrywki
- **avg\_rating** - średnia ocena gry
- **age** - minimalne ograniczenie wiekowe
- **availability** - ilość egzemplarzy dostępnych do sprzedaży
- **category** - kategorie, do których zalicza się dana gra (może być więcej niż jedna kategoria)
- **price** - aktualna cena gry
- **rent\_price** - cena za wypożyczenie (dzienna)

Dane pochodzą z forum i sklepu internetowego boardgamegeek.com, natomiast zostały one pobrane ze strony <https://www.kaggle.com/datasets/mrpantherson/board-game-data>. Główną zmianą w oryginalnym zbiorze danych było usunięcie kolumn z danymi, których nie planowaliśmy brać pod uwagę. Ponadto rozszerzyliśmy ją o trzy dodatkowe informacje: opis gry (description), cenę (price), oraz koszt wypożyczenia (rent\_price).

W celu wyciągnięcia informacji o cenie i opisie gry, zdecydowaliśmy się zbadać kod źródłowy każdej strony (kolumna bgg\_url z oryginalnej tabeli). Do tego zadania wykorzystaliśmy trzy bardzo istotne biblioteki: requests, BeautifulSoup, oraz Selenium, które dokładniej opisujemy w sekcji Technologie. Natomiast koszt wypożyczenia został wygenerowany następująco: grą których cena znajduje się w określonym zakresie, przypisujemy odpowiednią cenę wypożyczenia:

Cena produktu	Cena wypożyczenia
[0, 100)	2
[100, 500)	5
[500, 1000)	10
[1000, 1500)	15
> 1500	20

### 3.4 Tabela games\_to\_rent

Tabela gier do wypożyczenia powstaje na podstawie tabeli games\_for\_sale wybieramy z niej jedynie kolumnę game\_id, a następnie dla każdego identyfikatora (czyli dla każdego tytułu gry) losujemy liczbę z przedziału [0, 10] (z pewną ustaloną przewagą zer). Wylosowana liczba będzie liczbą egzemplarzy przeznaczonych do wypożyczenia. Duplikujemy każdy wiersz o wylosowany numer i każdemu egzemplarzowi przypisujemy unikatowy identyfikator. Umieszczamy je w kolumnie item\_id.

### 3.5 Tabela sale

W tabeli znajduje się 6 kolumn:

- **sale\_id** – unikalne id dla każdej sprzedaży,
- **customer\_id** – unikalne id klienta,
- **staff\_id** – unikalne id pracownika,
- **game\_id** – unikalne id gry,
- **amount** – kwota transakcji,
- **date** – data transakcji.

Dane w tabeli obejmują 117810 transakcji. Każda sprzedaż (wiersz w tabeli) była generowana w następujący sposób:

- **customer\_id** jest losowane z równym prawdopodobieństwem ze wszystkich dostępnych id klientów, a następnie z prawdopodobieństwem 0.83 zamieniane na NULL, co ma symulować klientów, którzy nie mają założonego konta w sklepie,
- **staff\_id** jest losowane z równym prawdopodobieństwem ze wszystkich dostępnych id pracowników,
- **game\_id** jest losowane z tabeli **games\_for\_sale** z prawdopodobieństwem proporcjonalnym do oceny gry (**avg\_rating**),
- **amount** to wartość **price** z tabeli **games\_for\_sale** powiększona o losową wartość proporcjonalną do tego, jak dawno transakcja miała miejsce (gra mogła być kiedyś droższa),
- **date** jest losowane spośród wszystkich dni pracujących od otwarcia sklepu do dziś z prawdopodobieństwem proporcjonalnym do czasu, jaki upłynął od otwarcia sklepu (sprzedaż rośnie wraz z czasem).

### 3.6 Tabela rental

Tabela ta generowana jest podobnie do tabeli **sale**, z kilkoma różnicami. Jedną z nich jest to, że tutaj **customer\_id** już nie zawiera wartości NULL, ponieważ klient, by wypożyczyć grę, musi mieć założone konto. Następnie kolumna **item\_id** jest tworzona po wylosowaniu **game\_id** (ponownie opieramy się na **avg\_rating**) i dopasowaniu odpowiadającego mu **item\_id**. W przypadku kolumny **return\_date** losujemy czas wypożyczenia gry z rozkładu Poissona i dodajemy do **rental\_date**.

### 3.7 Tabela competition

### 3.8 Tabela competition\_results

### 3.9 Połączenie z bazą

Za połączenie z bazą oraz wypełnienie jej wygenerowanymi danymi odpowiada plik **Connection.py**. Przesłanie danych do bazy możliwe jest dzięki bibliotece **SQLAlchemy**, która napisana jest w języku programowania Python i służy do pracy z bazami danych oraz mapowania obiektowo-relacyjnego.

W pierwszej kolejności tworzymy obiekt **Base**, za pomocą funkcji **declarative\_base()**, który jest klasą bazową. Następnie, dla każdej z tabel bazy, tworzymy klasę dziedziczącą z klasy podstawowej **Base**. W następnym kroku, za pomocą funkcji **create\_engine()**, zapisujemy do zmiennej **engine** połączenie z konkretną bazą. Dalej z wywołaniem funkcji **Base.metadata.create\_all(engine)** generowane są puste tabele bazy danych. Na sam koniec, z racji zapisywania generowanych danych losowych do tabel z wykorzystaniem biblioteki **pandas**, ponownie jest ona wykorzystywana i za pomocą funkcji **to\_sql()** tabele wypełniane są odpowiednimi danymi.

## 4 Analiza danych i generowanie raportu

Raport został napisany w języku R Markdown z wykorzystaniem Pythona. W pliku **Raport.rmd** znajduje się kod którego przekompilowanie zwraca plik pdf. Raport jest napisany tak by był uniwersalny

i dostosowywał się do zmian w bazie danych. Plik `rmd` zawiera w sobie kod pythonowy, który w trakcie kompilacji pobiera z bazy dane i na ich podstawie tworzy wykresy i wylicza poszczególne statystyki.

Raport odpowiada na następujące pytania i zagadnienia:

- Ranking na pracownika miesiąca.
- Ranking zawodników turniejowych.
- Które gry przynoszą największy dochód ze sprzedaży, a które z wypożyczeń?
- Które kategorie są najpopularniejsze?
- Jak zmieniała się sprzedaż na przestrzeni lat?
- Czy ocena gry i cena wpływają na sprzedaż?
- Ile gier dostępne jest w sklepie?

## 5 Technologie

Poniżej znajdują się spis technologii wykorzystywanych podczas pracy nad projektem, których użycie było wspomniane w niniejszej dokumentacji.

- MariaDB
- Python
- SQLAlchemy
- Selenium
- BeautifulSoup
- RMarkdown
- Reticulate

## 6 Zarządzanie plikami

Cały proces generowania danych i łączenia z bazą oraz analizy i generowania raportu został tak zorganizowany w plikach, aby mógł być odtworzony w bardzo łatwy sposób. Poniżej przedstawiona jest lista plików wraz z opisem ich zawartości:

- **Data\_generation.py** – Plik generuje dane to tabel w pandas, które następnie zapisuje do plików csv w folderze database.
- **Connection.py** – Plik łączy się z bazą oraz wczytuje dane z plików csv znajdujących się w folderze database do tabel w pandas. Następnie tabele te przesyła do bazy z wykorzystaniem biblioteki SQLAlchemy.
- **Raport.rmd** – Plik generuje raport z analizą danych.

Aby uzyskać gotowy projekt, należy najpierw utworzyć w folderze głównym plik tekstowy o nazwie `python_path.txt` i wpisać w nim ścieżkę do pythona na swoim komputerze. Następnie należy w następującej kolejności uruchomić wymienione powyżej pliki:

1. `Data_generation.py`
2. `Connection.py`
3. `Raport.rmd`

## 7 Baza w postaci EKNF

W tej części dokumentacji zawarto listę zależności funkcyjnych dla każdej relacji oraz uzasadnienie, że baza jest w EKNF.

### **Tabela games\_for\_sale**

Zależność funkcyjna: {game\_id, name, description} → pozostałe kolumny

Uzasadnieniem jest fakt, że game\_id jest kluczem głównym tabeli, a name i description są wartościami unikalnymi dla każdego game\_id, więc każda pozostała kolumna zależy funkcyjnie od zbioru tych trzech atrybutów. Jest to więc nietrywialna zależność funkcyjna, która zaczyna się od nadklucza, a więc nie zaburza to postaci EKNF bazy.

### **Tabela games\_to\_rent**

Zależność funkcyjna: item\_id → game\_id

Uzasadnieniem jest fakt, że item\_id jest kluczem głównym tabeli, więc game\_id zależy funkcyjnie od niego. Relacja ta nie zaburza postaci EKNF bazy, ponieważ tabela games\_to\_rent nie posiada żadnych niekluczowych zależności funkcyjnych.

### **Tabela sale**

Zależność funkcyjna: sale\_id → pozostałe kolumny

Uzasadnieniem jest fakt, że sale\_id jest kluczem głównym tabeli, więc każda pozostała kolumna zależy funkcyjnie od niego. Relacja ta nie zaburza postaci EKNF bazy, ponieważ tabela games\_to\_rent nie posiada żadnych niekluczowych zależności funkcyjnych.

### **Tabela rental**

Zależność funkcyjna: rental\_id → pozostałe kolumny

Uzasadnieniem jest fakt, że sale\_id jest kluczem głównym tabeli, więc każda pozostała kolumna zależy funkcyjnie od niego. Relacja ta nie zaburza postaci EKNF bazy, ponieważ tabela games\_to\_rent nie posiada żadnych niekluczowych zależności funkcyjnych.

### **Tabela customers**

Zależność funkcyjna: {customer\_id phone, email} → pozostałe kolumny

Uzasadnieniem jest fakt, że customer\_id jest kluczem głównym tabeli, a phone i email są wartościami unikalnymi dla każdego customer\_id, więc każda pozostała kolumna zależy funkcyjnie od zbioru tych trzech atrybutów. Jest to więc nietrywialna zależność funkcyjna, która zaczyna się od nadklucza, a więc nie zaburza to postaci EKNF bazy.

### **Tabela staff**

Zależność funkcyjna: {staff\_id phone, email} → pozostałe kolumny

Uzasadnieniem jest fakt, że staff\_id jest kluczem głównym tabeli, a phone i email są wartościami unikalnymi dla każdego staff\_id, więc każda pozostała kolumna zależy funkcyjnie od zbioru tych trzech atrybutów. Jest to więc nietrywialna zależność funkcyjna, która zaczyna się od nadklucza, a więc nie zaburza to postaci EKNF bazy.

### **Tabela competition**

Zależność funkcyjna: competition\_id → pozostałe kolumny

Uzasadnieniem jest fakt, że competition\_id jest kluczem głównym tabeli, więc każda pozostała kolumna zależy funkcyjnie od niego. Relacja ta nie zaburza postaci EKNF bazy, ponieważ tabela games\_to\_rent nie posiada żadnych niekluczowych zależności funkcyjnych.

### **Tabela competition\_results**

Zależność funkcyjna: result\_id → pozostałe kolumny



Uzasadnieniem jest fakt, że `result_id` jest kluczem głównym tabeli, więc każda pozostała kolumna zależy funkcyjnie od niego. Relacja ta nie zaburza postaci EKNF bazy, ponieważ tabela `games_to_rent` nie posiada żadnych niekluczowych zależności funkcyjnych.

Podsumowując, baza jest w EKNF, ponieważ każda z występujących w niej relacji nie zaprzecza tej postaci.

## 8 Podsumowanie

W dokumentacji został przedstawiony proces tworzenia bazy danych sklepu Geeks & Dragons od momentu tworzenia schemtu bazy, aż po analizę wykonywaną na stworzonej bazie. Przebieg powstawania bazy został podzielony na poszczególne etapy, co ułatwiło rozdzielenie zadań pośród członków grupy zajmującej się projektem. Najtrudniejszym zadaniem podczas realizacji projektu było przede wszystkim zrozumienie i uzasadnienie, że stworzona baza jest w postaci EKNF. Ostatecznie grupa projektowa jest bardzo zadowolona z efektów końcowych i ma nadzieję na wysoką, w domyśle możliwie najwyższą, punktację za wykonane zadanie.