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

1 Wstęp

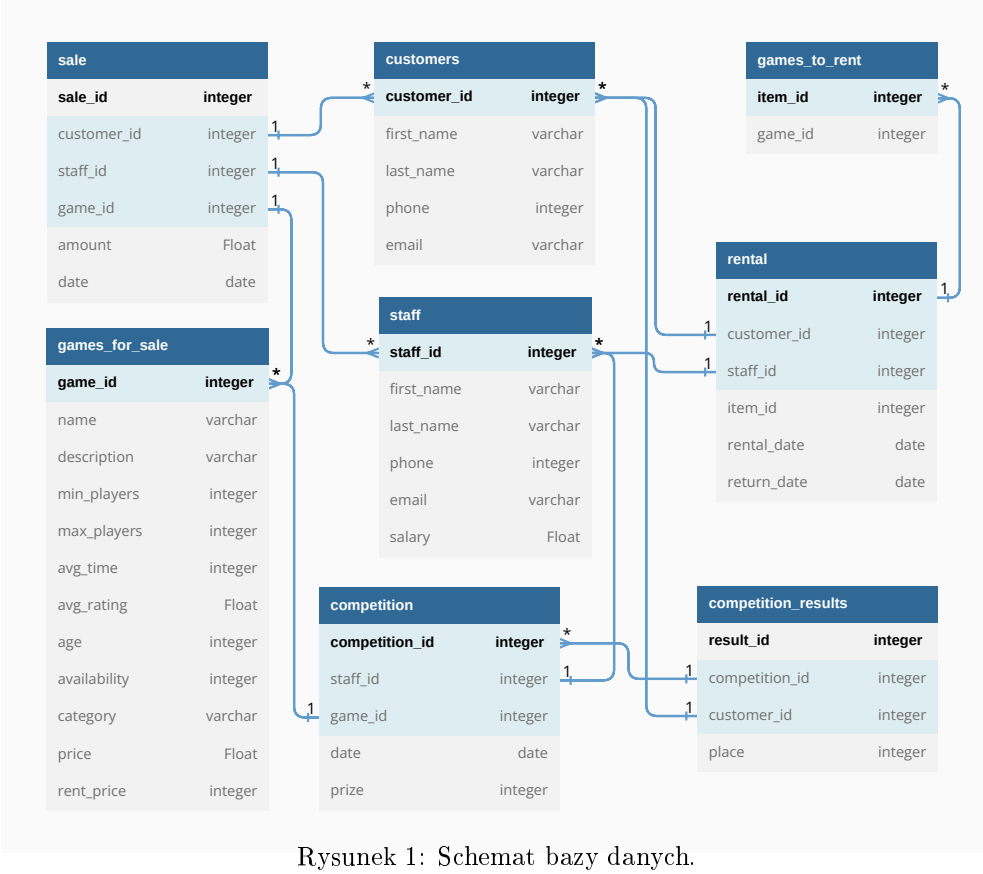
Niniejsza dokumentacja przedstawia proces tworzenia bazy danych sklepu Geeks & Dragons w ramach projektu na zaliczenie z kursu Bazy Danych z 2023 roku. Zadanie polegało na wysymulowaniu danych, jakie mogłyby się pojawić w bazie przykładowego sklepu, mającego w ofercie gry nieelektroniczne, który dodatkowo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

- **cusomer** – 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 organizownych turniejów,
- **competition_results** – dane dotyczące osiągnietych 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 rownież dane rzeczywiste, co zostanie opisane w dalszej części tego rozdziału. Na sam koniec, po połączniu 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 wybieranego 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 jednakowy 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 Kaggle. Główną zmianą w oryginalnym zbiorze danych było usunięcie kolumn z danymi, których nie planowaliśmy brać pod uwagę. Ponadto rozszerzyliśmy ją o trzy dodatowe 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 wypożycznia powstaje na podstawie tabeli games_for_sale wybieramy z niej jedynie kolumnę games_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** – id klienta,
- **staff_id** – id pracownika,
- **game_id** – 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. Nastepnie 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

Przechowuje ona ogólne informacje dotyczące turniejów. Zawiera 5 kolumn:

- **competition_id** – unikalne id dla każdego turnieju,
- **staff_id** – id prowadzącego turniej,
- **game_id** – id gry turniejowej,
- **date** – data rozegrania turnieju,
- **prize** – pula nagród.

Turnieje rozgrywane są co piątek z wyłączeniem dni ustawowo wolnych od pracy oraz Wielkiego Piątku. Aby wziąć udział w turnieju, gracz musi posiadać konto. Spośród stałych klientów losujemy trzydziestu graczy. Jako prowadzący przypisany jest losowo wybrany pracownik wypożyczalni. Losowa jest także nagroda za dany turniej. Nie w każdym wydarzeniu biorą udział wszyscy zadeklarowani. Graczy nie może być mniej niż minimalna oraz nie może przekraczać trzykrotności maksymalnej ilości graczy dla danej gry. Gry turniejowe wybierane są losowo spośród dwudziestu najlepiej ocenianych gier, a daty przypisywane są na cały rok kalendarzowy.

3.8 Tabela competition_results

Tabela ta zawiera szczegółowe informacje dotyczące wyników w turnieju każdego uczestnika . Zawiera 4 kolumny:

- **competition_id** – id turnieju,
- **customer_id** – id ,
- **place** – miejsce w turnieju,
- **result_id** – unikalne id wyniku,

Miejsca zajmowane przez uczestników są losowe.

3.9 Połączenie z bazą

Za połączenie z bazą oraz wypełnienie jej wygenerownymi 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 klasą 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 wypelniane 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 znajduję się spis technologii wykorzystywanych podczas pracy nad projektem, których użycie było wspominane w niniejszej dokumenatcji. Dla każdej pozycji jest dołączony link do dokumnetacji.

- MariaDB (dokumntacja)
- Python (dokumntacja)
- SQLAlchemy (dokumntacja)
- Selenium (dokumntacja)
- BeautifulSoup (dokumntacja)
- RMarkdown (dokumntacja)
- Reticulate (dokumntacja)

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

Dodatkowo w folderze data znajduje się description_and_price.py, który posłużył do rozszerzenia danych z grami o ich opis oraz cenę. Skrypt pythonowy zawarty w pliku łączy się ze stroną internetową boardgamegeek.com, skąd ściąga opisy gier, a nastepnie ich ceny. Tak uzyskane dane program dołącza do zbioru gier. W folderze data znajduje się już zbiór zawierający te kolumny, wiec kompilowanie tego pliku nie jest wymagane.

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 podzielowny na poszczególne etapy, co ułatwiło rodzielenie zadań pośród członków grupy zajmującej sie 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 nadzieje na wysoką, w domyśle możliwie najwyższą, punktacje za wykonane zadanie.