Bazy danych
Projekt zaliczeniowy - dokumentacja

Adrian Radziewicz 268816
Damian Bojarun 268814
Jakub Borcoń 268834
Marzena Rybak 268951

30 czerwca 2023

Spis treści

1 Wstęp

Niniejszy projekt polegał na stworzeniu testowej bazy danych automatycznie uzupełnianej losowymi danymi o charakterze podobnym do realnej bazy danych jaką mógłby prowadzić stacjonarny sklep z grami planszowymi zajmujący się również wypożyczaniem gier oraz organizacją turniejów. Dokumentacja stworzona do generowanej bazy danych zawiera informacje o stworzeniu bazy jak i jej charakterystykach.

2 Założenia

W projekcie zostały przyjęte liczne założenia dotyczące sklepu z grami planszowymi, dla którego tworzona jest baza. Poniżej wypunktowane zostały najważniejsze z tych założeń.

- Sklep działa od 01.06.2018.
- W trakcie trwania obostrzeń związanych z COVID-19 sklep ograniczył swoją działalność poprzez przerwanie prowadzenia sprzedaży, wypożyczeń i/lub turniejów. Dotyczy to dat 12.03-01.06.2020 oraz 01.11.2020-01.03.2021.
- Sklep prowadzi wyłącznie sprzedaż stacjonarną w jednej lokalizacji.

3 Spis użytych technologii

Jako bazę danych wybrano *PostgresSQL*, która została postawiona na *Dockerowym* kontenerze. Dane zostały wygenerowany w języku programowania *Python3*, przy pomocy bibliotek: *numpy* (losowość), *requests* (pobieranie danych z BGG XML API2), *simpy* (symulacje wypożyczeń) oraz *peewee* (sterownik ORM bazy). Dodatkowo zostały wykorzystane biblioteki: *json*, *tqdm*, *datetime*, *time*, *pandas*. W trakcie pracy wykorzystano również program DBeaver do przeglądania zawartości bazy danych.

Do otworzenia raportu potrzebna jest aplikacja obsługująca pliki PDF.

4 Lista plików

- **scrap.ipynb** - pobieranie danych z BGG XML API2
- **insert_data.ipynb** - generowanie danych
- **Dockerfile** - plik definicji Dockera bazy danych
- **docker-compose.yml** - plik definicji środowiska dockerowego
- **src/orm.py** - definicje klas ORM
- **db/docker.env** - definicje nazwy bazy, konta użytkownika i hasła

- **db/setup_clear/00_init_structure.sql** - plik zawierający pustą strukturę bazy danych
- **db/setup_backup/00_backup.sql** - plik zawierający zrzut bazy danych wygenerowany poleceniem *'pg dump'*
- **raport.ipynb** - analiza wygenerowanych danych
- **raport.html** - analiza wygenerowanych danych w wersji HTML
- **dokumentacja.pdf** - dokumentacja bazy danych

5 Uruchamianie plików

Przed uruchomieniem plików należy zainstalować Dockera, a także Pythona w wersji co najmniej 3.8. Kolejnym krokiem przygotowawczym jest pobranie plików projektu i rozpakowanie ich, bądź alternatywnie sklonowanie z repozytorium na githubie:
'git clone https://github.com/marzena1478520/Board-games-database.git'. Następnie należy zainstalować paczki poleceniem *'pip install -r requirements.txt'*.

5.1 Uruchomienie bazy

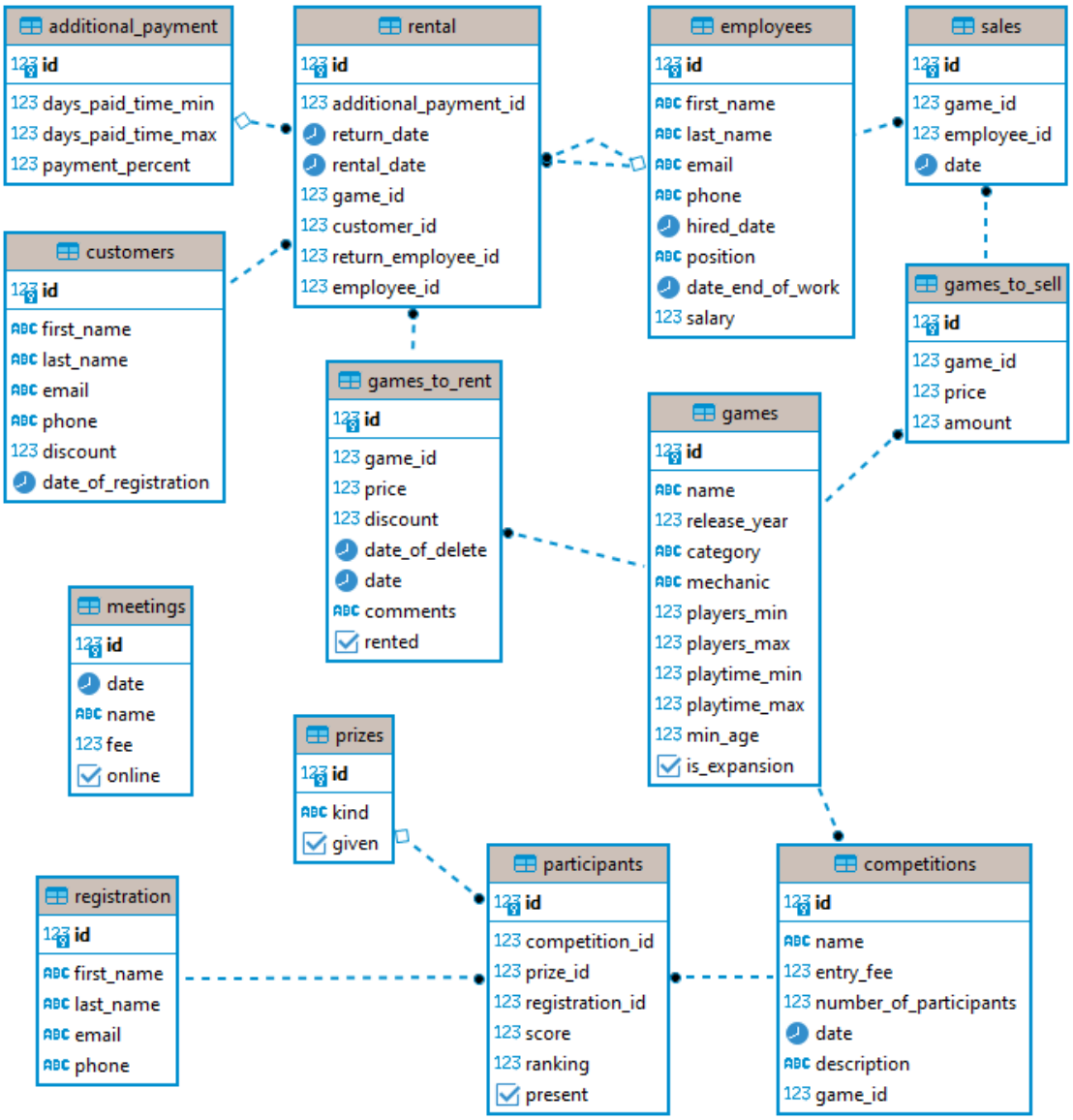
Aby uruchomić bazę danych należy w głównym folderze projektu uruchomić polecenie: *'docker compose up -d'*. Domyślnie baza będzie dostępna pod adresem: *localhost:5432*.

5.2 Generowanie nowej bazy

Aby wygenerować bazę od zera należy zmienić zawartość linii 18 pliku *docker-compose.yml* z: *'- ./db/setup_backup:/docker-entrypoint-initdb.d'* na *'- ./db/setup_clear:/docker-entrypoint-initdb.d'*. Następnie należy uruchomić wszystkie komórki w notebookach w kolejności: *scrap.ipynb*, *insert_data.ipynb*.

6 Schemat bazy danych

Schemat bazy danych zgodny z rysunkiem 1.



Rysunek 1: Schemat bazy danych

7 Relacje

7.1 Tabela *additional_payment*

$id \rightarrow \{days\_paid\_time\_min,\ days\_paid\_time\_max,\ payment\_percent\}$

Na podstawie *id* jesteśmy wstanie odczytać sugerowany czas, dla którego powinna zostać naliczona określona dodatkowa opłata procentowa.

7.2 Tabela *customers*

$id \rightarrow \{first\_name, last\_name, email, phone, discount, date\_of\_registration\}$

Każdy klient, który wypożycza gry ma swoje unikalne *id*. Dzięki niemu możemy odczytać podstawowe informacje o naszym kliencie takie jak jego imię, nazwisko, adres e-mail, numer telefonu, jaki ma rabat oraz kiedy się zarejestrował. Data rejestracji zazwyczaj jest datą pierwszego wypożyczenia gry.

7.3 Tabela *rental*

$id \rightarrow \{additional\_payment\_id,\ return\_date,\ rental\_date,\ game\_id,\ customer\_id,\ return\_employee\_id,\ employee\_id\}$

Na podstawie *id* wiemy jaki pracownik, wypożyczył grę naszemu klientowi, który pracownik obsługiwał zwrot oraz ile musiał dodatkowo zapłacić za wypożyczenie. Wiemy kiedy nastąpiło wypożyczenie oraz kiedy nastąpił zwrot gry.

- *additional_payment_id* $\leftarrow$ *id* z tabeli *additional_payment*

  W tabeli *rental* wykorzystane jest *id* z tabeli *additional_payment* w celu identyfikacji wielkości dodatkowej opłaty za wypożyczenie gry. Wielkość wpisywana w *additional_payment_id* bazuje na datach wypożyczenia i oddania jednak nie jest zdeterminowana przez te wielkości, ponieważ jako założenie zostało przyjęte, że *additional_payment_id* wpisuje pracownik ręcznie na podstawie tych danych, ale może uwzględnić również uzasadnienie podane przez klienta zmniejszając lub anulując dodatkową opłatę. Wartym zauważenia jest fakt, że *additional_payment_id* może zawierać wartość *NULL* niepojawiającą się w *additional_payment*.

- *customer_id* $\leftarrow$ *id* z tabeli *customers*

  Wypożyczenie gry powiązane jest z konkretnym klientem, o którym informacje znajdują się w tabeli *customers*.

- *return_employee_id* $\leftarrow$ *id* z tabeli *employees*

  Wypożyczenie gry powiązane jest z konkretnym pracownikiem odsługującym zwrot, o którym informacje znajdują się w tabeli *employees*.

- *employee_id* ← *id* z tabeli *employees*

  Wypożyczenie gry powiązane jest z konkretnym pracownikiem obsługującym wypożyczenie, o którym informacje znajdują się w tabeli *employees*.

7.4 Tabela *employess*

id → {*first_name*, *last_name*, *email*, *phone*, *hired_date*, *position*, *date_end_of_work*, *salary*}

Na podstawie *id* jesteśmy wstanie zidentyfikować konkretnego pracownika. Możemy odczytać jego imię, nazwisko, adres e-mail oraz numer telefonu. Wiemy także ile dany pracownik zarabia, na jakim jest stanowisku, kiedy został zatrudniony oraz zwolniony.

7.5 Tabela *sales*

id → {*sales*, *game_id*, *date*}

Na podstawie *id* sprzedaży wiemy kiedy została sprzedana konkretna gra oraz przez jakiego pracownika.

- *game_id* ← *id* z tabeli *games_to_sell*

  Sprzedane gry powiązane są z grami dostępnymi do sprzedaży, o których informacje znajdują się w tabeli *games_to_sell*.

- *employee_id* ← *id* z tabeli *employees*

  Sprzedane gry powiązane są z pracownikiem, który zajmował się tą sprzedażą. Informacje o tym pracowniku znajdują się w tabeli *employees*.

7.6 Tabela *games_to_rent*

id → {*game_id*, *price*, *discount*, *date_of_delate*, *date*, *comments*, *rented*}

Znając *id* mamy informacje o grze do wypożyczenia. Wiemy z jakim tytułem gry jest powiązana, ile kosztuje jej wypożyczenie, czy jest na nią rabat, kiedy została umieszczona w magazynie, kiedy została usunięta oraz czy jest aktualnie na wypożyczeniu klienta. Do gry przypisany jest również komentarz mówiący na przykład o brakujących elementach.

- *game_id* ← *id* z tabeli *games*

  Dostępne egzemplarze do wypożyczeń są opisane poprzez *id* gry, o której informacje znajdują się w tabeli *games*.

7.7 Tabela *games*

$id \rightarrow \{name, relase\_year, category, mechanic, players\_min, players\_max, playtime\_min, min\_age, is\_expansion\}$

Każda gra ma swoje *id*. Dzięki niemu znamy nazwę, kategorię oraz mechaniki gry, a także minimalny i maksymalny czas gry oraz liczbę graczy, rok wydania, minimalny wiek gracza oraz czy gra jest rozszerzeniem.

7.8 Tabela *games_to_sell*

$id \rightarrow \{game\_id, price, amount\}$

Na podstawie *id* rozróżniamy gry do sprzedaży. Wiemy ile gier danego typu jest dostępnych do sprzedaży oraz ile kosztuje jedna gra.

- *game_id* $\leftarrow$ *id* z tabeli *games*

  Dostępne egzemplarze sprzedażowe są opisane poprzez *id* gry, o której informacje znajdują się w tabeli *games*.

7.9 Tabela *prizes*

$id \rightarrow \{kind, given\}$

Na podstawie *id* rozróżniamy nagrody przekazywane uczestnikom zawodów. Wiemy jaki rodzaj nagrody został przyznany lub jest dostępny oraz czy nagroda została odebrana.

7.10 Tabela *registration*

$id \rightarrow \{first\_name, last\_name, email, phone\}$

Na podstawie *id* rozróżniamy osoby, które zapisały się na turniej. Znamy ich imię, nazwisko, email oraz numer telefonu.

7.11 Tabela *participants*

$id \rightarrow \{competition\_id, prize\_id, registration\_id, score, ranking, present\}$

Na podstawie *id* rozróżniamy uczestników biorących udział w różnych zawodach. Wiemy, który zawodnik otrzymał nagrodę (na podstawie rejestracji), w jakich zawodach, ile punktów uzyskał, który był w rankingu oraz czy pojawił się na konkursie.

- *prize_id* $\leftarrow$ *id* z tabeli *prizes*

  Uczestnicy biorący udział w zawodach otrzymują nagrodę opisaną w tabeli *prizes*. Możliwa jest wartość *NULL* dla zawodnika, któremu nie została przyznana nagroda.

- *registration_id* ← *id* z tabeli *registration*

  Informacje o uczestnikach biorących udział w zawodach przechowywane są w tabeli *registration*.

- *competition_id* ← *id* z tabeli *competition*

  W tabeli jest zawarta informacja o zawodach, w których dany uczestnik brał udział poprzez *id* z tabeli *competitions*.

7.12 Tabela *competitions*

id → {*name*, *entry_fee*, *numbers_of_participants*, *date*, *description*, *game_id*}

Dzięki *id* możemy rozróżnić każdy konkurs. Wiemy jak się on nazywał, ile wynosi wpisowe, ilu uczestników wzięło w nim udział, kiedy się on odbył/odbędzie, jakiej gry będzie dotyczyć oraz podstawowe informacje o zawodach (opis).

- *game_id* ← *id* z tabeli *games*

  Każdy turniej jest powiązany z konkretną grą planszową z tabeli *games* poprzez jej *id*.

7.13 Tabela *meetings*

id → {*date*, *name*, *fee*, *online*}

Dzięki *id* możemy zidentyfikować spotkanie otrzymując jego datę, nazwę, opłatę za wejście oraz typ spotkania (czy odbywało się zdalnie czy stacjonarnie).

8 EKNF

Każda relacja w stworzonej bazie danych jest w postaci normalnej klucza elementarnego (EKNF), ponieważ każda zależność funkcyjna zaczyna się od nadklucza. Tabele będące w EKNF również spełniają trzecią postać normalną (3NF).

9 Realizacja projektu

Podczas realizacji projektu najtrudniejszą częścią było stworzenie schematu bazy oraz generowanie danych w taki sposób, aby dane przypominały w możliwie największym stopniu dane rzeczywistego sklepu. Dane, które zawiera baza są danymi pozbawionymi nietypowych i nieszablonowych rekordów (potencjalnych błędów), które najprawdopodobniej miałyby miejsce dla rzeczywistego sklepu. Założone zostało, że sklep od początku swojej działalności nie zmieniał nic w schemacie bazy danych (nie zostawały dodane tabele ani kolumny), co również jest mało prawdopodobną sytuacją.

Ciekawą częścią projektu było szukanie informacji o rzeczywistych danych, takich jak na przykład imiona, nazwiska, informacje o grach planszowych. W rzeczywistej sytuacji takie

informacje byłyby zbierane ręcznie, bez potrzeby szukania źródeł tych danych. W naszym projekcie zdecydowaliśmy się na pobranie informacji o grach przy pomocy API BGG, co znacznie ułatwiło stworzenie tabeli z grami.

Kolejną ciekawą rzeczą w projekcie było wykorzystanie biblioteki *peewee*, która pozwala na podejście do bazy danych jako do obiektów w języku programowania Python. Wykorzystanie jej pozwoliło na rozwinięcie nowych umiejętności w pracy z bazą danych i zrozumienie nowego podejścia. Początkowo sprawiała ona problemy w trakcie realizacji zapytań, lecz wraz z kolejnymi zapytaniami, tworzenie ich stawało się coraz bardziej intuicyjne.