Dokumentacja

Zuzanna Bilińska, Joanna Kołaczek, Łukasz Simbiga, Filip Lasko

21 czerwca 2023

Spis treści

1 Wstęp

W dokumencie tym przedstawiono najważniejsze informację dotyczące projektu: Bazy danych. Zespół, będący autorem projektu, pracował w składzie: Zuzanna Bilińska, Joanna Kołaczek, Łukasz Simbiga, Filip Lasko. Tematy, które opracowaliśmy to:

- utworzenie schematu,
- wypełnianie bazy,
- analiza danych,
- raport,
- dokumentacja.

2 Spis użytych technologii

Technologie użyte podczas projektu Bazy danych:

- MariaDB,
- Python - użyte biblioteki: mysql.connector, pandas, random, math, numpy, datetime, matplotlib.pyplot, plotly, io,
- Python - Jupyter Notebook,
- TeX - TeXstudio.

3 Lista Plików

Lista plików, które utworzyliśmy podczas projektowania bazy danych:

- Generowanie_danych.ipynb,
- Sposób_generowania_danych.pdf,
- Raport_BD.ipynb,
- Dokumentacja.pdf.

4 Kolejność i sposób uruchamiania plików

Utworzenie bazy danych Zanim otworzysz plik sql_in_python.ipynb pamiętaj, że baza danych została już utworzona na serwerze giniewicz.it, więc zbędne jest uruchamianie tych plików i ponowne tworzenie bazy danych, jednak jeżeli naszym celem jest jej kreacja, powinniśmy uruchomić plik sql_in_python.ipynb. Otworzyć możemy go tylko i wyłącznie za pomocą Jupyter

Notebook. W notatniku za pomocą funkcji create_db_connection(...) łączymy się z bazą danych, a następnie za pomocą odpowiednich funkcji tworzymy tabele oraz je wypełniamy. Kontekst ten został szerzej opisany w dokumencie Sposób_generowania_danych.pdf.

Analiza oraz raport Jeżeli naszym celem jest utworzenie raportu zawierającego analizę danych, należy uruchomić plik Raport_BD.ipynb, aby następnie połączyć się z bazą danych za pomocą funkcji create_db_connection(...). Następnie, po wywołaniu reszty funkcji, w szczególności:

$$with\ open('html\_report.html'\ ,'w')\ as\ f:$$
$$f.write(html),$$

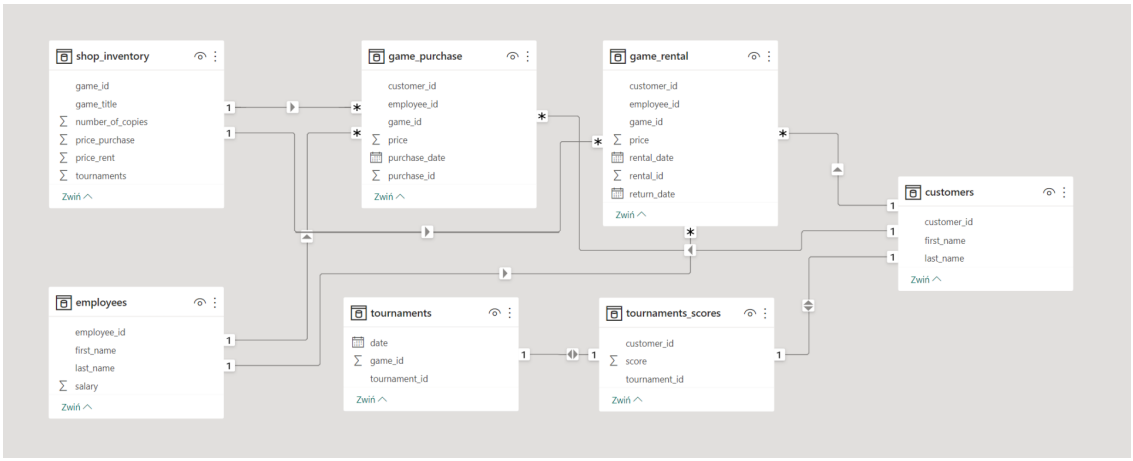
powstanie nam dokument, zapisany w html, przedstawiający raport z zawartą w nim analizą.

Dokumentacja Jeżeli chcemy poznać dodatkowe informację o projekcie Bazy danych, powinniśmy uruchomić plik Dokumentacja.pdf.

5 Schemat projektu bazy danych

Na rysunku 1 przedstawiony został schemat bazy danych, łącznie wyprodukowaliśmy siedem tabel, z czego każda tabela zawiera, co najmniej 3 kolumny z danymi. Baza danych przechowuje informację o:

- klientach,
- pracownikach,
- sprzedaży gier,
- wypożyczeniu gier,
- odbytych turniejach,
- asortymencie,
- wynikach turniejów.



Rysunek 1: Schemat bazy danych

6 Lista zależności funkcyjnych

Zależność funkcyjna to pojęcie używane w kontekście baz danych relacyjnych, które opisuje zależność między atrybutami w tabeli. Mówiąc prościej, zależność funkcyjna oznacza, że wartość jednego lub więcej atrybutów w tabeli jednoznacznie determinuje wartość innego atrybutu. Reprezentowana jest w postaci $A \rightarrow B$, gdzie A i B są zbiorami atrybutów. Oznacza to, że dla każdej kombinacji wartości atrybutów w zbiorze A, istnieje jednoznacznie określona wartość atrybutu lub atrybutów w zbiorze B. Może to oznaczać, że jeden atrybut determinuje inny, lub że grupa atrybutów determinuje inną grupę atrybutów.

W rozdziale tym zostaną zaprezentowane zależności funkcyjne, wystepujące w bazie danych, której schemat został zaprezentowany na rysunku 1. Dla każdej tabeli opiszemy występujące atrybuty oraz znajdziemy zależności funkcyjne, aby określić możliwe nadklucze, z których poźniej wybierzemy klucz główny.

customer_id	first_name	last_name
1	Łukasz	Simbiga
2	Wiktoria	Simbiga
⋮	⋮	⋮
1850	Piotr	Krawczyk

Tabela 1: Customers.

Customers W tabeli 1 przedstawiono część danych o klientach wraz z przypisanymi im atrybutami:

- $customer\_id_{INT}$ - numer klienta, przypisywany inny dla każdego klienta,

- $first\_name_{STR}$ - imię klienta,
- $last\_name_{STR}$ - nazwisko klienta.

Relacja jaką tworzy ta tabela zapisana może zostać następująco:

$$Customers = \{(1, Lukasz, Simbiga), \ldots, (1850, Piotr, Krawczyk)\}.$$

Z powyższych atrybutów jedynie {customer_id} jest unikatowe, ponieważ tylko w kolumnie, opisanej tym atrybutem, nie występują powtórzenia. Dodatkowo, połączenie atrybutów {first_name, last_name} nie gwarantuje nam unikatowości, ponieważ w bazie klientów występują osoby o takim samym imieniu i nazwisku. Stąd posiadać będziemy 4 nadklucze: {customer_id}, {customer_id, first_name}, {customer_id, last_name}, {customer_id, first_name, last_name}. Z czego tylko {customer_id} spełnia definicje klucza kandydującego, więc jest kluczem głównym.

Zależności funkcyjne, występujące w tabeli Customers, opisać możemy następująco:

$$\Sigma = \{A \to ABC\},$$

gdzie:

- A to customer_id,
- B to first_name,
- C to last_name.

Tak jak wspominaliśmy jedynie atrybut/zbiór atrybutów: {customer_id} jest unikatowy. Tylko znajomość numeru klienta gwarantuje nam możliwość poznania wszystkich innych danych tego samego klienta z tabeli 1, dlatego też zastosowana powyżej zależność funkcyjna jest dobrze opisana.

employee_id	first_name	last_name	salary
1	Adam	Plotnik	4200
2	Błażej	Słowik	4500
⋮	⋮	⋮	⋮
8	Damian	Skóra	4400

Tabela 2: Employees.

Employees W tabeli 2 zoabczyć możemy część danych o pracownikach oraz przypisane im atrybuty:

- $employee\_id_{INT}$ - numer pracownika, przypisywany inny dla każdego pracownika,
- $first\_name_{STR}$ - imię pracownika,
- $last\_name_{STR}$ - nazwisko pracownika,
- $salary_{INT}$ - zarobki pracownika.

Relacja jaką tworzy nam ta tabela zostać może zapisana następująco:

$$Employees = \{(1, Adam, Plotnik, 4200), \ldots, (8, Damian, Skora, 4400)\}.$$

Z powyższej relacji moglibyśmy wypisać wiele występujących nadkluczy. Jednak wśród nich występują 3 klucze kandydujące, które zarazem są atrybutami głównymi: {employee_id}, {first_name}, {last_name}. Jest tak, ponieważ każdy z tych atrybutów jest unikatowy dla tabeli 2. Następnie możemy wybrać klucz główny, którym będzie {employee_id}. Zależności funkcyjne, występujące w tabeli Employees, opisać możemy następująco:

$$\Sigma = \{A \to ABCD, B \to ABCD, C \to ABCD\},$$

gdzie:

- A to employee_id,
- B to first_name,
- C to last_name,
- D to salary.

Powodem wyboru takich zależności funkcyjnych był fakt, że każdy z atrybutów A,B,C jest unikatowy, jednak liczyć się trzeba z uwagą, iż przybycie do grona pracowników osoby o identycznym imieniu bądź nazwisku, skutkować będzie usunięciem odpowiedniej pozycji z grona zależności funkcyjnych. Przykładem może być pojawienie się pracownika o takim samym imieniu jak jeden z aktualnych pracowników,lecz różnych zarobkach, wtedy zależności funkcyjne opiszemy:

$$\Sigma = \{A \to ABCD, C \to ABCD, BD \to ABCD\}.$$

purchase_id	customer_id	game_id	employee_id	purchase_date	price
1	1	4	3	2020-04-22	100.00
2	2	65	1	2020-04-22	60.00
⋮	⋮	⋮	⋮	⋮	⋮
5807	1850	44	7	2022-02-14	125.00

Tabela 3: Game_purchase.

Game_purchase W tabeli 3 przedstawiono część danych, które odpowiadają za informację o sprzedanych grach. Dodatkowo, w tabeli znajdziemy atrybuty reprezentujące odpowiednie kolumny:

- $purchase\_id_{INT}$ - numer sprzedaży,
- $customer\_id_{INT}$ - numer klienta, który zakupił grę,
- $game\_id_{INT}$ - numer inforumujący, jaka gra została sprzedana,
- $employee\_id_{INT}$ - numer pracownika, który sprzedał grę,

- $purchase\_date_{DATE}$ - data sprzedaży,
- $price_{DECIMAL}$ - cena, po której sprzedano grę.

Relacje zawartą w tabeli 3 opisać możemy następująco:

$Game\_purchase = \{(1,1,4,3,2020-04-22,100.00),\ldots,(5807,1850,44,7,2022-02-14,125.00)\}$.

Dla powyższej relacji, wybranie {purchase_id} jako klucza głównego jest jedynym sensownym pomysłem. Zapewnia nam on występowanie, co najmnej jednej zależności funkcyjnej:

$$\Sigma = \{A \to ABCDEF, BCE \to ABCDEF\},$$

gdzie:

- A to purchase_id,
- B to customer_id,
- C to game_id,
- D to employee_id,
- E to purchase_date,
- F to price.

Z przyczyn oczywistych pojawia się w gronie zależności funkcyjnych: $A \to ABCDEF$. Co ciekawe, do tego grona dołączyć możemy dołączyć $BCE \to ABCDEF$, ponieważ klucz kandydujący {customer_id, game_id, purchase_date} jest unikatowy. Wpisując dowolne dane pod te atrybuty, jesteśmy w stanie poznać resztę informacji o odnotowanej sprzedaży.

rental_id	customer_id	game_id	employee_id	rental_date	return_date	price
1	1	43	2	2020-04-21	2020-05-18	30.00
2	2	65	1	2020-04-22	2020-05-18	15.00
⋮	⋮	⋮	⋮	⋮	⋮	⋮
19397	1850	58	3	2022-02-14	NULL	30.00

Tabela 4: Game_rental.

Game_rental W tabeli 4 przedstawiono część danych, które odpowiadają za informację o wypożyczonych grach. Dodatkowo, w tabeli znajdziemy atrybuty reprezentujące odpowiednie kolumny:

- $rental\_id_{INT}$ - numer wypożyczenia,
- $customer\_id_{INT}$ - numer klienta, który wypożyczył grę,
- $game\_id_{INT}$ - numer inforumujący, jaka gra została wypożyczona,
- $employee\_id_{INT}$ - numer pracownika, który wypożyczył grę,

- $rental\_date_{DATE}$ - data wypożyczenia gry,
- $return\_date_{DATE}$ - data zwrotu gry,
- $price_{DECIMAL}$ - cena, po której sprzedano grę.

Relacje zawartą w tabeli 4 opisać możemy następująco:

$$Game\_rental = \{(1, 1, 43, 2{,}2020-04-21{,}2020-05-18{,}30.00), \ldots, \\ (19397, 1850, 58, 3, 2022-02-14{,}NULL, 30.00)\}.$$

Jako klucz główny najlepiej w tym wypadku przyjąć {rental_id}. Atrybut ten jest unikatowy dla każdej pozycji w danych. Tak, jak w przypadku Game_purchase zapewnieni jesteśmy o tym, że atrybut główny utworzy nam zależność funkcyjna z każdym innym atrybutem, jednak sprawdźmy, czy istnieją jakieś inne zależności:

$$\Sigma = \{A \rightarrow ABCDEFG\},$$

gdzie:

- A to rental_id,
- B to customer_id,
- C to game_id,
- D to employee_id,
- E to rental_date,
- F to return_date,
- G to price.

Niestety nie doszukaliśmy się innych zależności funkcyjnych w tabeli 4. Powodem tej sytuacji jest najprawdopodobniej fakt, iż nasza tabela posiada, aż 19397 wierszy. Gdybyśmy nie uwzględnili atrybutu {rental_id}, to w tabeli występowały duplikaty. Chociażby wywołanie danych BCDEFG, nie gwarantuje nam, że poznamy informację kryjącą sie pod A, ponieważ w naszej wypożyczalni wystąpiła sytuacja, że ten sam klient wypożyczył 2 razy tę samą grę, tego same dnia i tę 2 gry zwrócił w tym samym dniu. Obsługiwał go ten sam pracownik i zapłacił taką samą cenę. Dodatkowo, nie znaleźliśmy powiązań między poszczególnymi atrybutami, więc zakończyliśmy poszukiwania zależności funkcyjnych.

tournament_id	tournament_date	game_id
1	2020-06-06 17:00	1
2	2020-07-04 16:00	2
⋮	⋮	⋮
25	2022-06-03 10:00	1

Tabela 5: Game_tournaments.

Game_tournaments W tabeli 5 zaprezentowano część danych inforumjących o odbytych turniejach oraz przypisane im atrybuty:

- $tournament\_id_{INT}$ - numer turnieju, przypisywany inny dla każdego turnieju,
- $tournament\_date_{DATETIME}$ - data odbycia turnieju,
- $game\_id_{INT}$ - numer gry.

Relacja jaka tworzy nam tabela 5, możemy zapisać następująco:

$$Game\_tournaments = \{(1{,}2020-06-06\ 17:00,1),\ldots,(25{,}2020-06-03\ 10:00,1)\}.$$

Z powyższej relacji wyłonić możemy dwa atrybuty główne(klucze kandydujące): {tournament_id}, {tournament_date}. Obydwa argumenty pasują do definicji klucza głównego, ponieważ są one unikatowe. Poznając informację ukrytą pod wybranym atrybutem, jesteśmy w stanie odszyfrować resztę danych z tabeli. Kluczem głównym w tej sytuacji zostanie {tournament_id}, ponieważ mamy pewność, że ten atrybut pozostanie unikatowy na zawsze, jeżeli nie zmienimy reguły numeracji turniejów. Z zaistniałych faktów, wyprowadzimy zależności funkcjonalne:

$$\Sigma = \{A \to ABC, B \to ABC\},$$

gdzie:

- A to tournament_id,
- B to tournament_date,
- C to game_id.

game_id	game_title	tournaments	number_of_copies	price_purchase	price_rent
1	Szachy	1	10	80	20
2	Pandemic Legacy	1	20	80	20
⋮	⋮	⋮	⋮	⋮	⋮
81	Dungeon Lords	0	20	60	15

Tabela 6: Shop_inventory.

Shop_inventory W tabeli 6 przedstawiono część danych, które odpowiadają za informację o grach zanjdujących się w asortymencie. Dodatkowo, w tabeli znajdziemy atrybuty reprezętujące odpowiednie kolumny:

- $game\_id_{INT}$ - numer gry,
- $game\_title_{STR}$ - tytuł gry,
- $tournaments_{INT}$ - liczba informująca, czy gra jest turniejowa(jeżeli 1 to turniejowa, jeżeli 0 to nie),
- $number\_of\_copies_{INT}$ - liczba kopii gier,
- $price\_purchase_{INT}$ - cena zakupu gry,

- $price\_rent_{INT}$ - cena wypożyczenia gry.

Relacja, którą tworzy nam tabela 6 możemy zapisać:

$$Shop\_inventory = \{(1,Szachy,1,10,80,20), \ldots, (81,DungeonLordes, 0,20,60,15)\}.$$

Dla powyższej relacji atrybutami głównymi(także kluczami kandydującymi) są: {game_id} oraz {game_title}. Podane atrybuty są unikatowe, dlatego przypisać możemy je do klucza głównego. Tak jak w poprzednich przypadkach, jako klucz główny wybierzemy tylko jeden atrybut, którym będzie {game_id}. Zależności funkcjonalne występujące w tabeli możemy opisać następująco:

$$\Sigma = \{A \to ABCDEF, B \to ABCDEF\},$$

gdzie:

- A to game_id,
- B to game_title,
- C to tournaments,
- D to number_of_copies,
- E to price_purchase,
- F to price_rent.

Niestety, żaden z pozostałych argumentów {C,D,E,F} oraz kombinacji argumentów, np.: CDEF nie spełnia własności zależności funkcjonalnych. Pomijając atrybuty: {game_id} oraz {game_title} sprawimy, iż w tabeli pojawią się duplikaty.

tournament_id	customer_id	score
1	9	1
1	10	1
⋮	⋮	⋮
21	508	1

Tabela 7: Tournaments_scores.

Tournaments_scores W tabeli 7 przedstawiono część danych, które odpowiadają za informację o punktach rozdanych konkretnym zawodnikom za konkretne trunieje. Dodatkowo, w tabeli znajdziemy atrybuty reprezentujące odpowiednie kolumny:

- $tournament\_id_{INT}$ - numer turnieju, za który przyznawane są punkty,
- $customer\_id_{INT}$ - numer klienta,
- $score_{INT}$ - otrzymane punkty.

Relacja utworzona przez tabele 7 opisana może zostać następująco:

$$Tournaments\_score = \{(1,1,1,10), \ldots, (21,201,508,1)\}$$

Z powyższej relacji wyłonić możemy klucz kandydujący: {tournament_id, customer_id}, który będzie również kluczem głównym. W tabeli 7 nie pojawia się atrybut, który odpowiadałby za numeracja każdego wiersza z osobna, lecz jego role zastępuje kombinacja atrybutów {tournament_id, customer_id}. Nie jest możliwe, aby ten sam klient wystartował 2 razy w tym samym turnieju, dlatego ta kombinacja jest unikatowa. Stąd zależność funkcyjna dla relacji Tournaments_score wygląda następująco:

$$\Sigma = \{A \rightarrow ABC\},$$

gdzie:

- A to tournament_id,
- B to customer_id,
- C to score.

7 Uzasadnienie, że baza jest w EKNF

Relacja jest w postaci normalnej klucza elementarnego (EKNF), jeśli każda nietrywialna zależność funkcyjna albo zaczyna się od nadklucza albo kończy się na atrybucie elementarnym. Definicja ta pozwoli odpowiedzieć nam na pytanie, czy baza danych skonstruowana przez nas jest w EKNF. Zauważyć należy, iż wszystkie zależności funkcyjne opisane przez nas w poprzednim rozdziale, posiadają jedną wspólną cechę: każda zależność funkcyjna zaczyna się od nadklucza. Niemożliwe jest znalezienie atrybutu/kombinacji atrybutów, które nie są nadkluczem, a z których jednoznacznie wynikałoby, że poznanie każdej danej pod tym atrybutem/kombinacją atrybutów pozwoli znaleźć każdą daną ukryta pod innym atrybutem. W skrócie, jeżeli jakiś atrybut/kombinacja atrybutów objaśniały nam inny atrybut, to były to tylko sytuacje, gdy ten atrybut objaśniający był nadkluczem. I taka zależność definiuje naszą bazę danych. Podsumowujac baza danych, skonstruowana przez nas zaespół, jest w postaci normalnej klucza elementarnego (EKNF).

8 Napotkane trudności

Podczas projektu napotkaliśmy wiele trudności, które opisaliśmy w poniższych punktach:

- synchornizacja danych - podczas wypełniania tabel dodatkową uwagę musieliśmy poświęcić na zgodność danych pomiędzy tabelami,
- konflikt oznaczeń - różnice w określaniu wartości niezidentyfikowanej(Python - None, MariaDB - NULL) dodały troche trudności podczas wypełniania bazy,
- różnica w indeksowaniu - Python z zasady indeksuje dane w listach/tabelach od 0, my nasze wiersze w tabelach indeksowaliśmy od 1, różnica ta sprawiła nam lekkie problemy podczas analizy danych.