Projekt zaliczeniowy PIABD

***Biblioteka***

Daniel Niczyporuk

178739

Krystian Biegajło

178676

Spis Treści

[1. Opis 2](#_Toc9502238)

[1.1 Cel projektu 2](#_Toc9502239)

[1.2 Opis projektu 2](#_Toc9502240)

[1.2.1 Schemat bazy 3](#_Toc9502241)

[1.2.2 Relacje 3](#_Toc9502242)

[1.2.3 Dodatkowe funkcjonalności 3](#_Toc9502243)

[1.2.4 Funkcje 4](#_Toc9502244)

[1.2.5 Widoki 5](#_Toc9502245)

[1.2.6 Procedury 8](#_Toc9502246)

[1.2.7 Wyzwalacze (Triggery) 9](#_Toc9502247)

[1.2.8 Indeksy 9](#_Toc9502248)

[1.2.9 Tablica audytu 10](#_Toc9502249)

1. Opis

Cel projektu

Projekt jest wykonany w ramach zajęć Programowania i Administracji Baz Danych. Projekt ma na celu stworzenie bazy danych dla biblioteki, która może być wykorzystana przez bibliotekę internetową. Zastosowane widoki, procedury, funkcje są stworzone tak aby miały praktyczne zastosowanie w działaniu biblioteki. Założeniem projektu jest pokazanie funkcjonalności przykładowo wygenerowanej bazy danych. Przedstawienie działania podstawowych operacji np. wyszukiwania książek po kategoriach, wyświetlania opinii o książkach czy liczeniu ile czasu osoba miała wypożyczoną książkę. Projekt jest kierowany użytkownikom i administratorom internetowych bibliotek.

Opis projektu

Projekt zawiera implementacje przykładowej bazy danych dla biblioteki. Baza danych zawiera tabele :

\*Adresy \*Osoby \*Wypożyczenia \*Pracownicy \*Książki

\*Autorzy \*Wydawnictwa\*Kategoria \*Opinie

\*tabela relacji autorów z książkami

\*tabela relacji książek z kategoriami

Skrypt generujący bazę danych jest napisany w języku Java. Wykorzystywany jest package javafaker, który pozwala generować losowe dane na wybrany temat. (strona projektu: <https://github.com/DiUS/java-faker> ) Tworzone dane są w plikach .json które przekonwertowaliśmy do plików .sql gdzie znajdują się INSERT’y do bazy danych. Korzystaliśmy z strony (<http://convertjson.com/json-to-sql.htm>) . Połączone wszystkie pliki .sql z INSERT’ami i tworzeniem bazy danych znajdują się w zbiorczym pliku Databas\_And\_Insert.sql. Został on wygenerowany przes Sql Server Management Studio po stworzeniu tabel i dodaniu do nich danych.

Użytkownik może zarejestrować się w serwisie. Może wyświetlić indeks książek dostępnych w bibliotece. Zobaczyć szczegółowy opis książki i złożyć wniosek o wypożyczenie. Przypisany zostaje zatrudniony pracownik, który zarządza wypożyczeniem. Użytkownik może wyświetlić ile jest książek z danej kategorii, sprawdzić opinie i je dodawać. Pracowników można zatrudniać i wyrzucać. Można poznać informacje o autorze książek i wydawnictwie. Pracownik może sprawdzić jakie książki, kiedy i kto wypożyczył. Można sprawdzić wszystkie opinie jakie dodał użytkownik. Użytkownik może oczywiście oddać książkę. Można sprawdzić czas wypożyczenia danej książki przez konkretną osobę. Użytkownik może również poznać liczbę wszystkich książek oraz liczbę egzemplarzy książek o danym tytule lub liczbę książek danego wydawnictwa.

* + 1. Schemat bazy

A screenshot of a computer

Description automatically generated

* + 1. Relacje

W bazie występować będą następujące relacje:

1. wiele do wielu pomiędzy Autorem i Książką,

2. wiele do wielu pomiędzy Kategorią i Książką,

3. jeden do wielu pomiędzy Książka i Opinią,

4. jeden do wielu pomiędzy Osobą i Opinią,

5. jeden do wielu pomiędzy Osobą i Adresem,

6. jeden do zera pomiędzy Osobą i Pracownikiem,

7. wiele do jednego pomiędzy Wypożyczeniem i Książką,

8. wiele do jednego pomiędzy Wypożyczeniem i Osobą,

9. wiele do jednego pomiędzy Wypożyczeniem i Pracownikiem

* + 1. Dodatkowe funkcjonalności

Z dodatkowych funkcjonalności użytkownik może:

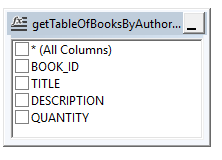
* Dodać opinię do wybranej książki
* Sortować książki po autorze lub kategoriach, gdzie książka może należeć do wielu kategorii i autorów.
* Zobaczyć historię wypożyczeń oraz których książek nie zwrócił
  + 1. Funkcje

**Funkcja** definiowana przez użytkownika to obiekt bazy danych, który zawiera w sobie jedną lub więcej instrukcji języka SQL, przygotowany do ponownego użycia. Funkcje zawsze muszą zwracać jakąś wartość lub tabelę. Funkcje mogą mieć parametry wejściowe. W funkcjach można używać tylko zapytania SELECT.

W projekcie stworzone są następujące funkcje:

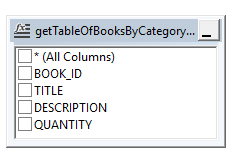
\***getTableOfBooksByAuthor()**

Funkcja zwraca tablice książek z numerami ID, tytułami, opisami i ilościami, które należą do danego autora o zadanym imieniu i nazwisku.



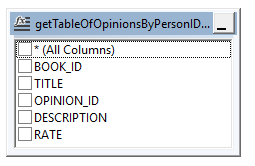
\***getTableOfBooksByCategory()**

Funkcja zwraca tablice książek z numerami ID, tytułami, opisami i ilościami, które pochodzą z zadanej w parametrze nazwie kategorii.



\***getTableOfOpinionsByPersonId()**

Funkcja zwraca tablice książek z numerami ID, tytułami, opiniami , oceną i numerami ID opinii, po podaniu w parametrze numeru id osoby.



\* **getBorrowTimeInMonthsByBorrowID()**

Funkcja zwraca czas wypożyczenia w miesiącach, po podaniu parametru numer ID wypożyczenia.

\* **getSumOfBooks()**

Funkcja zwraca sumę książek znajdujących się w bazie danych. Funkcja nie posiada parametrów wejściowych.

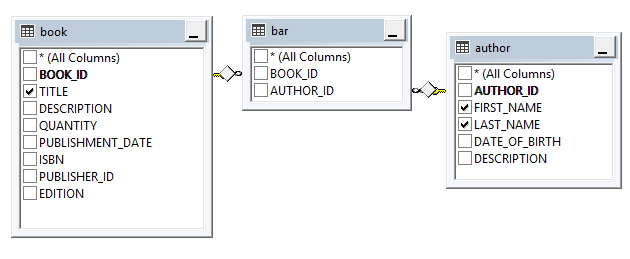
* + 1. Widoki

**Widok** (perspektywa) – jest to zapisane w bazie danych nazwane zapytanie pobierające dane z jednej lub wielu encji (tabel) . Do widoku odwołujemy się tak jak do nazwy tabeli.

Widoki wykorzystywane w projekcie :

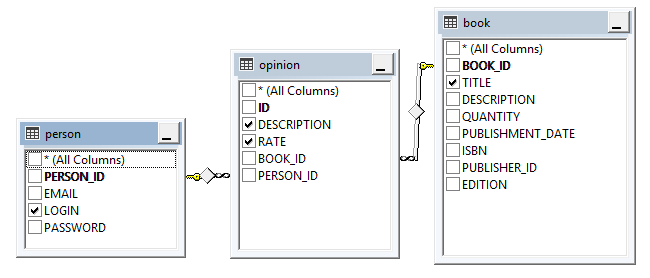
\***dbo.BOOK\_AND\_AUTHOR**

Widok pokazuje tabelę z tytułami książek, imiona i nazwiska autorów. Pola: book.title, author.first\_name, author.last\_name.



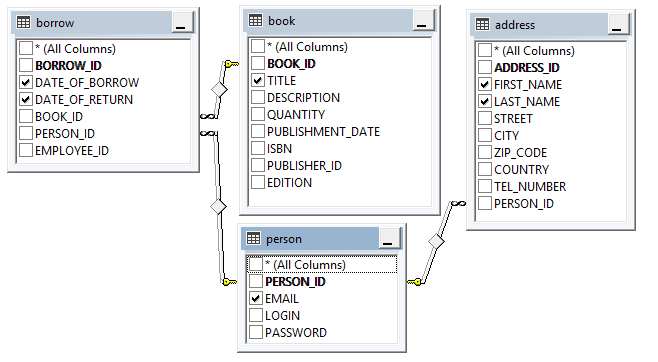
\***dbo.BOOK\_WITH\_OPINION**

Widok pokazuje tabelę z tytułami książek, komentarzemi, ocenami i loginami osób. Pola: book.title, opinion.rate, opinion.description, person.login



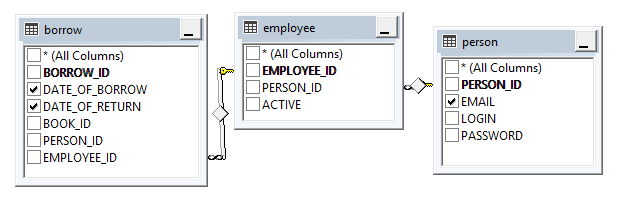
\***dbo.BORROW\_WITH\_OPINIONS\_AND\_PERSON**

Widok pokazuje tabelę z tytułami książek, datami wypożyczenia, datami oddania, imiona nazwiska i emaile osób. Pola: book.title, borrow.DATE\_OF\_BORROW, borrow.DATE\_OF\_RETURN, person.EMAIL, address.FIRST\_NAME, address.LAST\_NAME

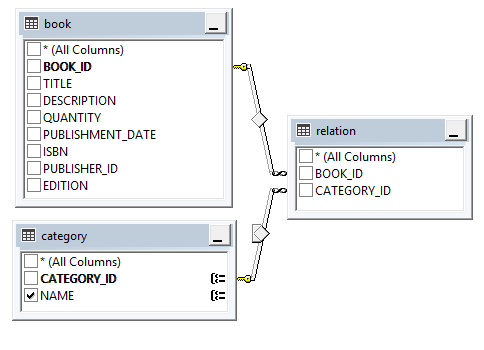


\***dbo.BORROW\_WITH\_EMPLOYEE**

Widok pokazuje tabelę z mailami osób, datami wypożyczenia i datami zwrotu. Pola: person.email, borrow.date\_of\_borrow, borrow.date\_of\_return

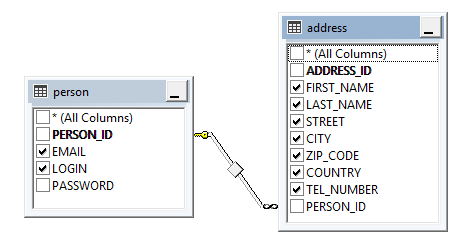
\***dbo.CATEGORIES\_WITH\_COUNT\_OF\_BOOKS**

Widok pokazuje tabelę wszystkich kategorii książek i liczbę egzemplarzy w bazie danych. Pola: category.[NAME], count(\*) AS "NUMBERS\_OF\_BOOKS".



\***dbo.PERSON\_WITH\_ADDRESS**

Widok pokazuje tabelę z loginami, imionami i nazwiskami osób, a także ich adresy czyli nazwa ulicy miasta kod pocztowy kraj oraz numer telefonu. Pola: person.EMAIL, person.LOGIN, address.FIRST\_NAME, address.LAST\_NAME, address.STREET, address.CITY, address.ZIP\_CODE, address.COUNTRY, address.TEL\_NUMBER



* + 1. Procedury

**Procedury** składowane stanowią część schematu bazy danych. Stosuje się je do wykonywania powtarzających się, logicznie takich samych operacji na (bazie)danych, nie wymagających ingerencji ze strony użytkownika.

Procedury stworzone w projekcie:

\* **getCountOfBookByTitle** @Title varchar(255) = NULL, @Quantity int OUT

Procedura liczy ile jest danych egzemplarzy książek w bibliotece o zadanym w parametrze tytule książki.

\* **setBorrowAsReturned** @Borrow\_Id int = NULL

Procedura ustawia datę oddania książki na aktualną. W parametrze podajemy numer wypożyczenia.

\* **getOpinionsForBookWithId** @Book\_Id int = NULL

Procedura wypisuje tabelę opinii (numer id opnii, treść opinii, ocenę) i tytuł książki dla zadanego w parametrze numeru ID książki.

\* **getAllBooksFromPublisherWithName**  @Publisher\_Name VARCHAR(255) = NULL

Procedura wypisuje wszystkie książki (tytuły, numery ISBN, ilości, wydania) i nazwę wydawnictwa, dla wskazanego w parametrze numeru ID wydawnictwa.

\* **dismissEmployeWithID** @Employee\_Id INT = NULL

Procedura zmienia stan pracownika na nieaktywny, po podaniu w parametrze numeru ID pracownika.

* + 1. Wyzwalacze (Triggery)

Triggery to instrukcje, które pozwalają na wykonywanie operacji w trakcie wykonywania instrukcji. Przykładowo możemy zmienić wartość danej w tabeli lub widoku korzystając z instrukcji DML, czyli wykonać INSERT UPDATE DELETE.

W projekcie używamy trigerów:

\***triggerInsertAddress**

**\*triggerUpdateAddress**

**\*triggerDeleteAddress**

**\*triggerInsertCategory**

**\*triggerUpdateCategory**

**\*triggerDeleteCategory**

**\*triggerInsertBook**

**\*triggerUpdateBook**

**\*triggerDeleteBook**

**\*triggerInsertBorrow**

**\*triggerUpdateBorrow**

**\*triggerDeleteBorrow**

Gdy aktualizujemy tablice czyli wprowadzamy nowe dane lub je aktualizujemy czy tez usuwamy to trigger monitorując operacje na bazie danych dodaje to zdarzenie do tablicy audytowej.

* + 1. Indeksy

To obiekty bazodanowe niezależne logicznie i fizycznie od tabeli. Pozwalają uzyskać szybszy dostęp do danych. Indeksy zakłada się na kolumnę w tabeli lub kilka kolumn naraz. Indeksy przechowują wartości kolumn na które są nakładane oraz ROWID wiersza.

W projekcie używamy indeksów:

\* **person\_Name**

Wyszukuje konkretny rząd w tabeli autorów po imieniu i nazwisku.

\* **categoryName**

Wyszukuje konkretny rząd w tabeli kategorii po kategorii.

\* **addressTelNumber**

Wyszukuje konkretny rząd w tabeli adresów po numerze telefonu.

* + 1. Tablica audytu

DROP TABLE IF EXISTS Audits

GO

CREATE TABLE Audits (

AUDIT\_ID INTEGER IDENTITY(1,1) PRIMARY KEY,

RECORD\_ID INTEGER,

TABLE\_NAME VARCHAR(255),

TYPE\_OF\_CHANGE VARCHAR(255),

UPDATED\_BY\_USERNAME VARCHAR(255),

UPDATE\_TIME DATETIME

)

GO

Tablica audytu zawiera pola: numer ID audytu, numer ID rekordu, nazwa tabeli, typ zmiany, kto aktualizował, czas aktualizacji. Można z niej odczytać informacje o wprowadzanych zmianach na danych w tabelach w bazie danych. Wiemy który rząd się zmienił w jakiej tabeli jaka była operacja przeprowadzona kiedy i jaki użytkownik ją wykonał.

* + 1. Generowanie danych do postaci XML

Dane z bazy danych zrzutować można w łatwy sposób dopisując do zapytania select FOR XML PATH(‘nazwa\_rekordu’). Przykłady select zwracający tabele książek wygląda następująco.

SELECT [BOOK\_ID]

,[TITLE]

,[DESCRIPTION]

,[QUANTITY]

,[PUBLISHMENT\_DATE]

,[ISBN]

,[PUBLISHER\_ID]

,[EDITION]

FROM [Grupa\_4].[dbo].[Book]

FOR XML PATH ('Book')

Struktura danych xml wygląda następująco:

