**Dokumentacja projektowa**

• Zdefiniowanie problemu – podstawowe informacje na temat utworzonej bazy danych.

Księgarnia potrzebuje systemu, za pomocą którego może wydajniej pracować. System powinien pozwalać na archiwizację sprzedaży książek oraz ich zwrotów, a także zamówienia do hurtownii.

• Przedstawić zakładane warunki pracy aplikacji.

Aplikacja kliencka ma być dostępna w dwóch miejscach - przy kasie oraz w magazynie. Dla kasjera są udostępnione opcje dokonania zakupu przez klienta oraz jego zwrotu. Magazynier . Aplikacja serwera ma być dostępna dla klientów przez internet, i ma zarządzać bazą danych, to jest wprowadzać nowe rekordy, aktualizować je i udostępniać widoki.

• Zamodelowanie rozwiązania problemu przy użyciu diagramów związków encji. Do tego celu należy wykorzystać narzędzia graficzne do zobrazowania wyników pracy analitycznej.



Ksiazki - zawiera informacje o numerze ISBN książkek, ich tytułach i autorach, dostępnych w księgarni

Klienci - zawiera dane klientów księgarni

Zakupy - zawiera klucze zakupionych książek, klientów którzy je kupili, oraz za jaką cenę

Zwroty - zawiera klucze zakupów, które zostały zwrócone, daty dokonanych zwrotów oraz ich powody

Zamowienia - zawiera datę złożenia zamówienia oraz datę jego odbioru, która jest uzupełniana później

ZamowieniaKsiazek - zawiera informacje które i ile książek jest w którym zamówieniu

• Przedstawienie fragmentu kodu odpowiedzialnego za wprowadzenie danych inicjalnych.

Za tworzenie bazy danych odpowiedzialny jest skrypt CreateQuery.sql:

CREATE DATABASE Ksiegarnia

GO

USE Ksiegarnia

GO

CREATE TABLE Ksiazki

(

ISBN char(13) PRIMARY KEY,

Tytul varchar(50) NOT NULL,

Autor varchar(50) NOT NULL,

)

GO

CREATE TABLE Zamowienia

(

ID\_Zamowienia INTEGER IDENTITY(1,1) PRIMARY KEY,

Data\_Zlozenia date NOT NULL,

Data\_Odbioru date,

)

GO

CREATE TABLE ZamowieniaKsiazek

(

ID\_Zamowienia INTEGER FOREIGN KEY REFERENCES Zamowienia NOT NULL,

ISBN char(13) FOREIGN KEY REFERENCES Ksiazki NOT NULL,

Ilosc INTEGER NOT NULL,

CONSTRAINT ID\_ZK PRIMARY KEY (ID\_Zamowienia, ISBN)

)

GO

CREATE TABLE Klienci

(

ID\_Klienta INTEGER IDENTITY(1,1) PRIMARY KEY,

Imie varchar(50) NOT NULL,

Nazwisko varchar(50) NOT NULL,

)

GO

CREATE TABLE Zakupy

(

ID\_Zakupu INTEGER IDENTITY(1,1) PRIMARY KEY,

ISBN char(13) FOREIGN KEY REFERENCES Ksiazki NOT NULL,

ID\_Klienta INTEGER FOREIGN KEY REFERENCES Klienci NOT NULL,

Zaplata INTEGER NOT NULL,

)

GO

CREATE TABLE Zwroty

(

ID\_Zwroty INTEGER IDENTITY(1,1) PRIMARY KEY,

ID\_Zakupu INTEGER FOREIGN KEY REFERENCES Zakupy NOT NULL,

Data\_Zwrotu datetime NOT NULL,

Powod varchar(100) NOT NULL,

)

GO

Wstawienie przykładowych danych do bazy (plik SampleDataQuery.sql):

INSERT INTO Ksiazki VALUES('1234567890123', 'Tytuł', 'Autor');

GO

INSERT INTO Ksiazki VALUES('5678901234567', 'Tekst', 'Ktoś');

GO

INSERT INTO Ksiazki VALUES('9012345678901', 'Popularne', 'Sławny');

GO

INSERT INTO Klienci VALUES('Stefan','Kot');

GO

INSERT INTO Klienci VALUES('Andrzej', 'Roman');

GO

• Wskazanie bazy użytkowników i ich uprawnień.

Użytkownicy przechowywani są w pliku users.txt, gdzie przechowywane są również ich role. Role są w pliku roles.txt, w którym do każdej roli wymienione są uprawnienia. Oba pliki są przechowywane przy aplikacji serwera.

• Przedstawienie wyników testów (z punktu widzenia posiadanych przez danego użytkownika uprawnień) na podstawie skonstruowanych zapytań demonstrujących wybrany model kontroli dostępu.

Test 1 - użytkownik z rolami clerk i supplier, zalogowany w roli clerk, próbuje złożyć zamówienie (wymagana rola supplier)

Wynik: serwer zwraca BAD\_ROLE.

Test 2 - użytkownik z rolami clerk i supplier, zalogowany w roli clerk, próbuje obsłużyć zwrot zakupu (wymagana rola clerk)

Wynik: otwiera się menu obsługi zwrotu, które można wypełnić.

Test 3 - użytkownik z rolą supplier próbuje się zalogować w roli clerk

Wynik: serwer zwraca INCORRECT\_ROLE

Test 4 - zalogowany użytkownik próbuje się ponownie zalogować z innego miejsca

Wynik: serwer zwraca ALREADY\_LOGGED