Baza Danych dla projektu Biblioteka

1. Opis systemu

Opracowanie Systemu Baz Danych wspomagającego funkcjonowanie Biblioteki. System powinien spełniać wszystkie funkcje prawidłowego działania biblioteki. System zostanie utworzony od podstaw.

1.1 Lista wymagań:

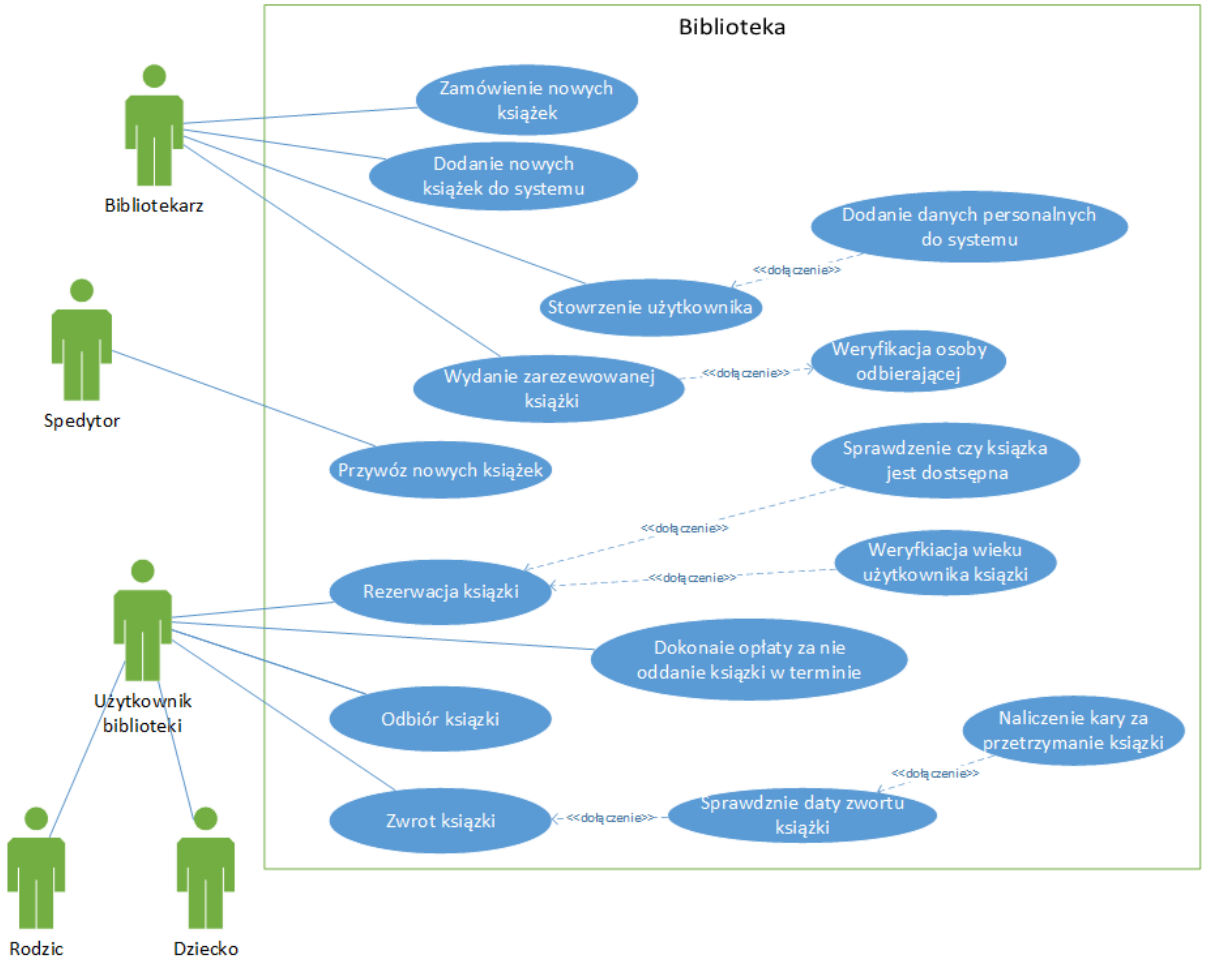
- System powinien zawierać modele relacyjnych baz danych w celu uniknięcia redundancji
- System powinien zawierać osobne tabele dla klientów, dostawców oraz pracowników
- System powinien rozdzielić klientów na płeć męską oraz żeńską
- System powinien rozdzielić klientów na osoby pełnoletnie oraz nieletnie
- System powinien powiązać osobę nieletnią z osobą pełniącą opiekę nad nią
- System powinien automatycznie przypisywać unikalny numer klienta dla nowo zarejestrowanych
- System powinien sprawdzić dostępność żądanego produktu
- System powinien sprawdzać terminowość zwrotu książki
- System powinien wysyłać powiadomienia o zbliżającym się terminie zwrotu książki
- W przypadku braku zwrotu, system powinien naliczyć karę pieniężną naliczaną za każdy dzień zwłoki
- System powinien pozwolić na zarezerwowanie książki dla klienta z czasem oczekiwania max. 2 dni
- System powinien pozwolić klientowi na zalogowanie do swojego konta
- System powinien pozwalać użytkownikowi na podgląd do jego konta
- System powinien pozwolić na modyfikację oraz aktualizację danych przez pracownika bądź użytkownika po weryfikacji konta
- System powinien wyświetlać informację o dostępności żądanych produktów
- System powinien weryfikować pełnoletność użytkownika przy rejestracji oraz logowaniu do systemu

1.2 Opis uwarunkowań:

- Stworzenie modelu bazy danych
- Stworzenie tabel relacyjnych
- Stworzenie tabel klient, dorosły, dziecko, pracownik, rezerwacja, książka, kara
- Wprowadzenie pól danych zaprezentowanych w modelu Bazy Danych
- Powiązanie odpowiednich tabel ze sobą kluczami głównymi oraz obcymi
- Wprowadzenie constrainów nakładających ograniczenia wiekowe dla klientów nieletnich
- Wprowadzenie constrainu zabraniającego rezerwację książki gdy jej stan magazynowy wynosi 0

2. Przypadki użycia

Diagram przypadków użycia



Opis przypadków użycia

Nazwa: Stworzenie użytkownika **Aktorzy:** Bibliotekarz
Scenariusz: 1. Bibliotekarz tworzy nowy profil osoby w systemie biblioteki 2. Weryfikuje tożsamość osoby rejestrującej się 3. Zapisuje dane w systemie
Nazwa: Wydanie zarezerwowanej książki **Aktorzy:** Bibliotekarz

Scenariusz:
1. Bibliotekarz wyszukuje zarezerwowaną książkę i przygotowuję do wypożyczenia
2. Bibliotekarz weryfikuje tożsamość osoby wypożyczającej książkę
3. Bibliotekarz wydaje książkę użytkownikowi biblioteki

Nazwa: Zamówienie nowych książek
Aktorzy: Bibliotekarz

Scenariusz:
1. Bibliotekarz sporządza listę potrzebnych książek
2. Bibliotekarz wysyła zamówienie na potrzebne książki

Nazwa: Dodanie książek do systemu
Aktorzy: Bibliotekarz

Scenariusz:
1. Bibliotekarz dodaje książki do systemu

Nazwa: Przywóz nowych książek
Aktorzy: Spedytor

Scenariusz:
1. Odbiór zamówienia
2. Przywóz do biblioteki zamówionych książek

Nazwa: Weryfikacja wieku użytkownika
Aktorzy: Użytkownik biblioteki

Scenariusz:
1. System sprawdza czy książka posiada wymagany minimalny wiek
2. System sprawdza czy użytkownik osiągnął minimalny wiek dla wybranej książki

Nazwa: Sprawdzanie czy książką jest dostępna
Aktorzy: Użytkownik biblioteki (system)

Scenariusz:
1. System sprawdza czy książka jest dostępna w bibliotece czy została już wydana

Nazwa: Odbiór książki
Aktorzy: Użytkownik biblioteki

1. Użytkownik odbiera osobiście książkę w bibliotece

Nazwa: Zwrot książki
Aktorzy: Użytkownik biblioteki

1. Użytkownik zwraca osobiście książkę w bibliotece

Nazwa: Sprawdzenie daty zwrotu książki
Aktorzy: Użytkownik biblioteki (system)

1. System sprawdza czy książka została zwrócona w terminie

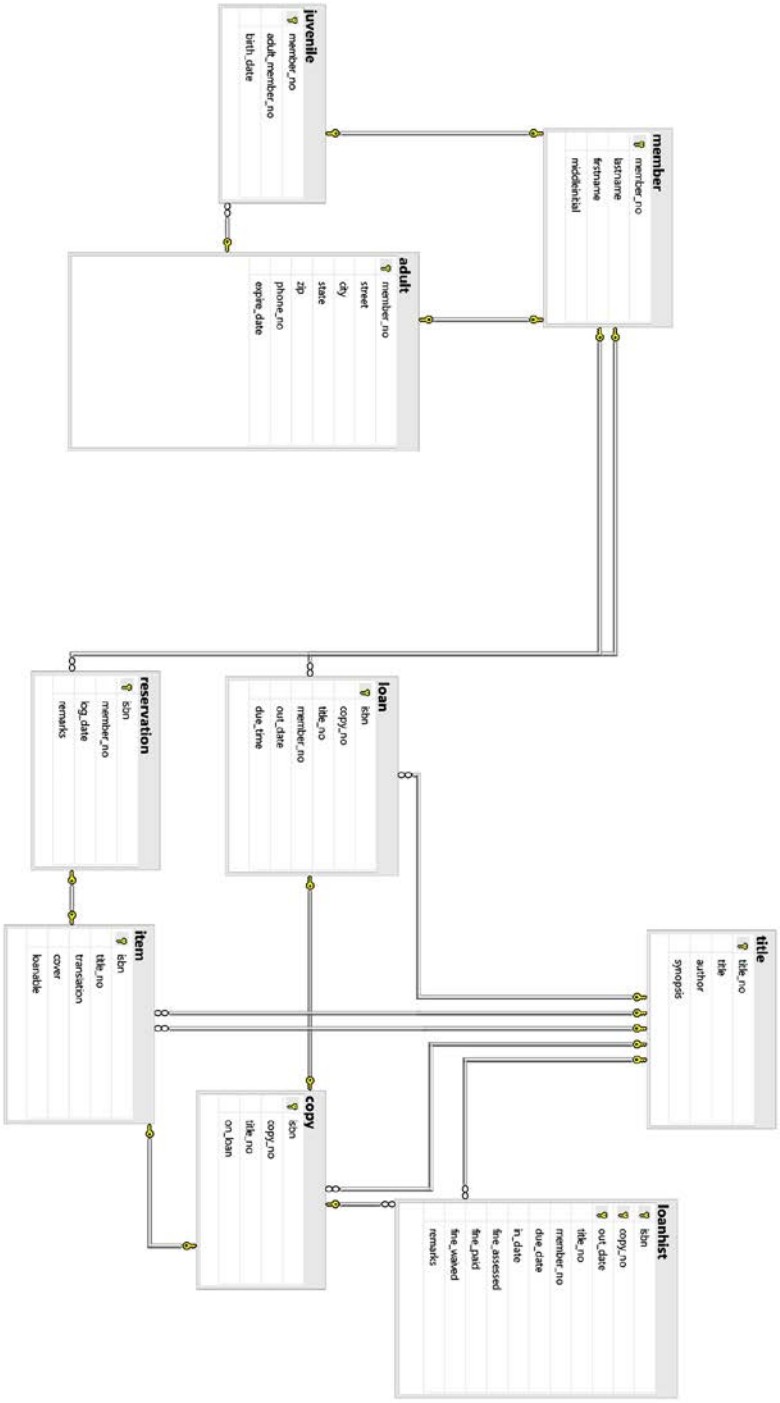
Nazwa: Naliczanie kary za przetrzymanie książki
Aktorzy: Użytkownik biblioteki (system)

1. System nalicza opłatę za przetrzymanie książki

Nazwa: Dokonanie opłaty za nie oddanie książki w terminie **Aktorzy:** Użytkownik biblioteki
1. Użytkownik dokonuje opłaty za przetrzymanie książki 2. System odnotowuje, że opłata została dokonana

3. Projekt bazy danych

3.1 Schemat bazy danych



3.2 Opis poszczególnych tabel

Nazwa tabeli: Member Tabela zawiera spis użytkowników biblioteki, zawiera numer identyfikujący użytkownika, który jest kluczem głównym, imię, inicjał drugiego imienia oraz nazwisko nazwisko

Nazwa atrybutu	Typ	Opis/Uwagi
mamber_no	Integer	numer identyfikujący, klucz główny, pole nie może być puste
lastname	Varchar	Nazwisko użytkownika, pole nie może być puste
firstname	Varchar	Imię użytkownika, pole nie może być puste
middleinitial	Varchar	Iniciał drugiego imienia użytkownika, pole może być puste

Nazwa tabeli: Juvenile

Tabela zawiera spis użytkowników niepełnoletnich, którzy podlegają opiece dorosłychi, zawiera numer identyfikujący użytkownika, który jest kluczem głównym, imię, inicjał drugiego imienia oraz nazwisko nazwisko

Nazwa atrybutu	Typ	Opis/Uwagi
mamber_no	Integer	numer identyfikujący, klucz główny, pole nie może być puste, klucz obcy odwołujący się do klucza głównego member_no tabeli member
adult_member_no	Varchar	numer identyfikujący opiekuna, pole nie może być puste, klucz obcy odwołujący się do klucza głównego member_no tabeli adult
birth_date	datetime	data urodzin w celu weryfikacji pełnoletności użytkownika, pole nie może być puste

Nazwa tabeli: Adult

Tabela zawiera spis użytkowników pełnoletnich i opiekunów użytkowników nieletnich, zawiera numer identyfikujący użytkownika, który jest kluczem głównym, adres zamieszkania, numer kontaktowy oraz datę oddania wypożyczonej książki

Nazwa atrybutu	Typ	Opis/Uwagi
mamber_no	Integer	numer identyfikujący, klucz główny, klucz obcy odwołujący się do klucza głównego member_no tabeli member, pole nie może być puste
street	Varchar	nazwa ulicy zamieszkania, pole nie może być puste
city	Varchar	nazwa miasta zamieszkania, pole nie może być puste
state	Varchar	Stan/Województwo miejsca zamieszkania, pole nie może być puste
zip	char	kod pocztowy, pole nie może być puste
phone_no	char	numer telefonu, pole nie może być puste
expire_date	datetime	data zwrotu/oddania książki

Nazwa tabeli: Title

Tabela zawiera spis tytułów ksiąg jakie zawiera biblioteka, w skład tabeli wchodzą, numer identyfikacyjny książkę, który jest kluczem głównym, tytuł jednostki, autora, oraz synopsis czyli krótki opis woluminu,

Nazwa atrybutu	Typ	Opis/Uwagi
title_no	Integer	numer identyfikujący, klucz główny, pole nie może być puste
title	Varchar	tytuł materiału, pole nie może być puste

author	Varchar	Imię i nazwisko autora, pole nie może być puste
synopsis	Text	krótki opis produktu, pole może być puste

Nazwa tabeli: Item Tabela zawiera stan dostępności produktów do dyspozycji, zawiera kod ISBN który jest kluczem głównym, tytuł do którego się odwołujemy oraz w jakim tłumaczeniu został przełożony		
Nazwa atrybutu	Typ	Opis/Uwagi
isbn	Integer	Uniwersalny kod książek, klucz główny, pole nie może być puste
title_no	Integer	numer identyfikujący klucz obcy odwołujący się do klucza głównego title_no tabeli title, pole nie może być puste
translation	Varchar	opis w jakim języku jest dany produkt, pole może być puste
cover	Varchar	Iniciał drugiego imienia użytkownika, pole może być puste
loanable	bit	dostępnośc produktu, pole może być puste

Nazwa tabeli: copy Tabela zawierające kopie stanu wypożyczeń, w skład tabeli wchodzi kod isbn, numer kopii książki, numer tytułu oraz stan fizyczny na magazynie		
Nazwa atrybutu	Typ	Opis/Uwagi
ISBN	Integer	Uniwersalny kod książek, klucz główny nieklastrowany, pole nie może być puste

copy_no	Integer	Numer identyfikujący kopię danego tytułu, klucz główny nieklastrowany, pole nie może być puste
title_no	Integer	Numer identyfikujący tytuł produktu, pole nie może być puste
on_loan	bit	wartość pokazująca czy produkt został wypożyczony czy nie, pole nie może być puste

Nazwa tabeli: loan Tabela zawierające dane użytkowników którym nałożono karę grzywny za przetrzymanie książek		
Nazwa atrybutu	Typ	Opis/Uwagi
ISBN	Integer	Uniwersalny kod książek, klucz główny, klucz obcy odwołujący sie do klucza głównego isbn tabeli copy, pole nie może być puste
copy_no	Integer	Numer identyfikujący kopię danego tytułu, pole nie może być puste
title_no	Integer	Numer identyfikujący tytuł produktu, klucz obcy odwołujący się do klucza głównego title_no w tabeli title, pole nie może być puste
member_no	Integer	Numer identyfikujący użytkownika, klucz obcy odwołujący sie do klucz głównego member_no tabeli member, pole nie może być puste
out_date	date	data wydania książki, pole nie może być puste
due_time	datetime	data planowanego zwrotu książki, pole nie może być puste

Nazwa tabeli: loanhist

Tabela wykazująca historie zadłużeń oraz czy grzywna została uiszczona

Nazwa atrybutu	Typ	Opis/Uwagi
ISBN	Integer	Uniwersalny kod książek,klucz główny klastrowany, klucz obcy odwołujący się do klucza głównego isbn tabeli copy, pole nie może być puste
copy_no	Integer	Numer identyfikujący kopię danego tytułu, klucz główny klastrowany, pole nie może być puste
member_no	Integer	Numer identyfikujący użytkownika, pole nie może być puste
out_date	datetime	data wydania produktu, klucz główny klastrowany, pole nie może być puste
due_time	datetime	data planowanego zwrotu książki, pole może być puste
in_date	datetime	data faktycznego zwrotu produktu, pole może być puste
fine_assessed	money	kara pieniężna nałożona na podstawie nieterminowego zwrotu książki pole może być puste
fine_paid	money	wartość wpłaconej grzywny, pole może być puste
fine_waived	money	wartość kary umorzonej, pole może być puste
remarks	varchar	uwagi, pole może być puste

Nazwa tabeli: reservation Tabela zawiera informacje odnośnie rezerwacji danych tytułów		
Nazwa atrybutu	Typ	Opis/Uwagi
ISBN	Integer	Uniwersalny kod książek, klucz główny nieklastrowany, klucz obcy odwołujący się do klucza głównego isbn tabeli item, pole nie może być puste
member_no	Integer	Numer identyfikujący użytkownika, klucz główny nieklastrowany, klucz obcy odwołujący się do klucza głównego member_no tabeli member, pole nie może być puste
log_date	Datetime	Data rezerwacji, pole może być puste
remarks	varchar	uwagi, pole może być puste

4. Implementacja

4.1 Kod poleceń DDL

```
CREATE DATABASE [biblioteka]

USE [biblioteka]

GO

CREATE TABLE member(
      member_no INT identity(1,1) NOT NULL PRIMARY KEY,
      lastname VARCHAR (50) NOT NULL,
      firstname VARCHAR (25) NOT NULL,
      middleinitial VARCHAR (1) NULL
)



CREATE TABLE juvenile(
      member_no INT IDENTITY(1,1) NOT NULL PRIMARY KEY FOREIGN KEY
REFERENCES member (member_no),
```

```
      adult_member_no INT  NOT NULL FOREIGN KEY REFERENCES adult
(member_no),
      birth_date datetime NOT NULL
)

create table adult(
      member_no INT IDENTITY(1,1) NOT NULL PRIMARY KEY FOREIGN KEY
REFERENCES member (member_no),
      street varchar(25) not null,
      city varchar (25) not null,
      [state] varchar (25) not null,
      zip char (10) not null,
      phone_no char (14) not null,
      expire_date datetime not null
      )

CREATE TABLE title(
      title_no int identity(1,1) Not null PRIMARY KEY,
      title varchar(60) not null,
      author varchar (31) not null,
      synopsis text null
      )



Create table item(
      isbn int not null PRIMARY KEY,
      title_no int not null FOREIGN KEY REFERENCES title(title_no),
      translation varchar(8) null,
      cover varchar(8) null,
      loanable bit Default 1 null
)

create table [copy](
      isbn int not null PRIMARY KEY FOREIGN KEY REFERENCES item(isbn),
      copy_no int not null,
      title_no int not null FOREIGN KEY REFERENCES title(title_no),
      on_loan bit Default 1 not null
)

            ALTER TABLE [copy]
            ADD CONSTRAINT PK_copy PRIMARY KEY NONCLUSTERED
            (
                  [isbn] ASC,
                  [copy_no] ASC
            )

create table loan(

      isbn int not null PRIMARY KEY FOREIGN KEY REFERENCES [copy](isbn),
      copy_no int not null,
      title_no int not null FOREIGN KEY REFERENCES [title](title_no),
      member_no int not null FOREIGN KEY REFERENCES [member](member_no),
```

```
	out_date datetime not null,
	due_time datetime not null
)

create table loanhist(

	isbn int not null FOREIGN KEY REFERENCES [copy](isbn),
	copy_no int not null,
	out_date datetime not null,
	title_no int not null FOREIGN KEY REFERENCES title(title_no),
	member_no int not null,
	due_date datetime null,
	in_date datetime null,
	fine_assessed money null,
	fine_paid money null,
	fine_waived money null,
	remarks varchar(255) null

	)

		ALTER TABLE loanhist
		ADD CONSTRAINT PK_Loan PRIMARY KEY CLUSTERED
		(
			[isbn] ASC,
			[copy_no] ASC,
			[out_date] ASC
		)


create table reservation(

	isbn int not null FOREIGN KEY REFERENCES item(isbn),
	member_no int not null FOREIGN KEY REFERENCES member(member_no),
	log_date datetime null,
	remarks varchar(255) null

)

		ALTER TABLE reservation
		ADD  CONSTRAINT PK_reservation PRIMARY KEY NONCLUSTERED
			(
				[member_no] ASC,
				[isbn] ASC
			)
```

4.2 Widoki

Tworzenie widoku dla wszystkich ludzi zapisanych w bibliotece wliczając dzieci oraz dorosłych.

```
CREATE VIEW All_Members as
SELECT m.lastname, m.firstname, j.member_no from member as m
JOIN juvenile as j
ON m.member_no = j.adult_member_no
ORDER BY m.lastname, m,firstname
```

Tworzenie widoku wszystkich dostępnych książek wraz z ISBN każdej książki

```
CREATE VIEW Books as
SELECT t.title, t.author, l.isbn from title as t
JOIN loanhist as l
ON t.title_no = l.title_no
Order by t.author
```

Tworzenie widoku dla grupy ludzi którzy zapłacili bądź opłatę za nie oddanie danej książki w odpowiednim czasie

```
CREATE VIEW Paid as
SELECT l.fine_paid, l.fine_assessed, t.title_no, lo.member_no from laonhist
as l
JOIN title as t
ON l.title_no = t.title_no
JOIN loan as lo
ON t.title_no = lo.title_no
ORDER BY l.fine_paid
```

Tworzenie widoku dla członków biblioteki ktorzy musza oddac książkę wraz z tytulem ksiazki i data oddania

```
CREATE VIEW Members_Due as
SELECT m.lastname, m.firstname, l.due_date, t.title from members as m
JOIN loan as l
ON m.member_no = l.member_no
JOIN title as t
ON l.title_no = t.title_no
```

Tworzenie widoku dla zarezerwowanych książek wraz z osoba ktora chce tę książkę wypożyczyć

```
CREATE VIEW Resrvation as
```

```
SELECT r.log_date, l.due_date, t.title, m.lastname, m.firstname from
reservation as r
JOIN loan as l
ON r.isbn = l.isbn
JOIN title as t
ON l.title_no = t.title_no
JOIN member as m
ON t.member_no = m.member_no
ORDER BY l.due_date
```

4.3 Procedury składowane

Procedura dodająca członka biblioteki do tabeli member

```
CREATE PROCEDURE dbo.add_to_members
@mem_no INT = NULL,
@first VARCHAR(50) = NULL,
@last VARCHAR(25) = NULL,
@middle VARCHAR(50) = NULL,

AS
DECLARE @error = AS NVARCHAR(500);
IF @mem_no IS NULL OR @first IS NULL OR @last IS NULL
      BEGIN SET @error = ‘Wrong data’;
      RAISEERROR(@error, 16,1);
      RETURN;
END

INSERT INTO members(member_no, firstname, lastname, middlename)
VALUES(@mem_no, @first, @last, @middle);

GO
```

Procedura dodająca książkę biblioteki do tabeli title, item

```
CREATE PROCEDURE dbo.add_book_to_liblary
@tit_no INT = NULL,
@titleProc VARCHAR(60) = NULL,
@authorProc VARCHAR(31) = NULL,
@synopsisProc TEXT = NULL,
@isbnProc INT = NULL,
@translationProc VARCHAR(8) = NULL,
@coverProc VARCHAR(8) = NULL,
@loanableProc BIT DEFAULT 1 = NULL,


AS
DECLARE @error = AS NVARCHAR(500);
IF @tit_no IS NULL OR @titleProc IS NULL OR @isbnProc IS NULL
      BEGIN SET @error = ‘Wrong data’;
      RAISEERROR(@error, 16,1);
```

```
      RETURN;
END

INSERT INTO title(title_no, title, author, synopsis)
VALUES(@tit_no, @title_Proc, @authorProc, @synopsisProc)
INSERT INTO item(isbn, title_no, translation, cover, loanable)
VALUES(@isbnProc, @tit_no, @translationProc, @coverProc, @loanableProc);

GO
```

Procedura która pokazuje członków biblioteki którzy muszą oddać książkę w podanym terminie.

```
CREATE PROCEDURE dbo.BookDue @due datetime = NULL
AS
SELECT t.title from title as t
JOIN loan as l
ON t.title_no = l.title_no
WHERE due_date = @due
GO
```

Procedura która pokazuję kto wypożyczł daną książkę

```
CREATE PROCEDURE dbo.ShotBooks @titleName = NULL
AS
SELECT t.title, m.firstname, m.lastname from title
JOIN loan as l
ON t.title_no = l.title_no
JOIN member as m
ON l.member_no = m.member_no
WHERE @titleName = t.title
GO
```

4.4 Triggery

Trigger ograniczający sesję użytkownika ‘guest’ do jednego w danym momencie

```
CREATE LOGIN guest WITH PASSWORD = 'guest'
go
      GRANT VIEW SERVER STATE TO guest;
go

CREATE TRIGGER connection_limit_trigger
ON ALL SERVER WITH EXECUTE AS 'guest'
FOR LOGON
AS
BEGIN
IF ORIGINAL_LOGIN()= 'guest' AND
(SELECT COUNT(*)FROM sys.dm_exec_sessions
      WHERE is_user_process = 1 AND original_login_name = 'guest') > 1
      PRINT 'tylko jeden uzytkownik na raz moze byc dostepny poczekaj do
zakonczenia sesji'
      ROLLBACK;
END;
```

Trigger aktywujący sie dla usuwania i modyfikowania tabel

```
CREATE TRIGGER da_safety
ON DATABASE
FOR DROP_TABLE, ALTER_TABLE
AS
   PRINT 'You must disable Trigger "safety" to drop or alter tables!'
   ROLLBACK;
```

Trigger wysyłający maila przypominającego

```
CREATE TRIGGER upomnienie
ON [biblioteka].[member]
AFTER LOGON
AS

   EXEC msdb.dbo.sp_send_dbmail
        @profile_name = 'Biblioteka Administrator',
        @recipients = 'admin@biblioteka.com',
        @body = 'Proszę pamietać o terminowym zwrocie książęk oraz spłacie
kar',
        @subject = 'Przypomnienie';
```

Trigger upominający dla administratorów i programistów:

```
CREATE TRIGGER przypomnienie
```

```
ON [biblioteka].[member]
AFTER INSERT, UPDATE, DELETE
AS

print 'powiadom użytkowników o zmianach'
```

Trigger zapobiegający działaniom na bazie przez kogokolwiek prócz administratora:

```
CREATE TRIGGER uprawnienia
ON ALL SERVER
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
      IF ORIGINAL_LOGIN() != 'SysAdm'
            PRINT'nie masz uprawnień do wykonywanie tych zadań'
            ROLLBACK;
END;
```