Projekt Bazy Danych – Biblioteka
(zarządzanie wypożyczeniami e-booków, audiobooków, książek)

1. Cel i wymagania:

Klasa: Biblioteka

Atrybuty:
• ID Biblioteki: Unikalny identyfikator biblioteki (klucz główny).
• Nazwa: Nazwa biblioteki.
• Adres: Adres siedziby biblioteki.
• Kierownik: Informacja o kierowniku biblioteki.

Metody:
• DodajKierownika(kierownik): Dodaje nowego kierownika biblioteki.
• AktualizujKierownika(nowy_kierownik): Aktualizuje dane kierownika.
• UsunKierownika(): Usuwa dane kierownika.

Klasa: Czytelnik

Atrybuty:
• ID Czytelnika: Unikalny identyfikator czytelnika (klucz główny).
• Imię: Imię czytelnika.
• Nazwisko: Nazwisko czytelnika.
• PESEL: Numer PESEL czytelnika.
• Adres: Adres zamieszkania.
• Data Rejestracji: Data zapisu do biblioteki.
• Status: Aktywny lub nieaktywny czytelnik.

Metody:
• DodajCzytelnika(dane_czytelnika): Dodaje nowego czytelnika do bazy.
• AktualizujDaneCzytelnika(nowe_dane): Aktualizuje dane czytelnika.
• DezaktywujCzytelnika(): Zmienia status czytelnika na nieaktywny.
• UsunCzytelnika(): Usuwa dane czytelnika z bazy.

Klasa: Książka

Atrybuty:
• ID Książki: Unikalny identyfikator książki (klucz główny).
• Tytuł: Tytuł książki.
• Autor: Autor książki.
• ISBN: Numer ISBN.
• Rok Wydania: Rok publikacji.
• Status: Dostępna / Wypożyczona / Zarezerwowana / Usunięta.

Metody:
• DodajKsiążkę(dane_książki): Dodaje nową książkę do bazy.
• AktualizujDaneKsiążki(nowe_dane): Aktualizuje dane książki.
• ZmieńStatus(nowy_status): Zmienia status dostępności książki.
• UsunKsiążkę(): Usuwa książkę z bazy.

Klasa: Wypożyczenie

Atrybuty:
• ID Wypożyczenia: Unikalny identyfikator wypożyczenia (klucz główny).
• ID Czytelnika: Referencja do wypożyczającego czytelnika.
• ID Książki: Referencja do wypożyczonej książki.
• Data Wypożyczenia: Data rozpoczęcia wypożyczenia.
• Termin Zwrotu: Termin zwrotu książki.
• Status: W trakcie / Zwrócona / Przeterminowana.

Metody:
• ZarejestrujWypożyczenie(dane_wypożyczenia): Tworzy nowe wypożyczenie.
• ZaktualizujStatus(nowy_status): Aktualizuje status wypożyczenia.
• ZarejestrujZwrot(data_zwrotu): Rejestruje zwrot książki.
• UsunWypożyczenie(): Usuwa informacje o wypożyczeniu.

Klasa: Rezerwacja

Atrybuty:
• ID Rezerwacji: Unikalny identyfikator rezerwacji (klucz główny).
• ID Czytelnika: Referencja do czytelnika rezerwującego.
• ID Książki: Referencja do rezerwowanej książki.
• Data Rezerwacji: Data złożenia rezerwacji.
• Status Rezerwacji: Aktywna / Anulowana / Zrealizowana.

Metody:
• DodajRezerwację(dane_rezerwacji): Tworzy nową rezerwację.
• AnulujRezerwację(): Anuluje rezerwację.
• AktualizujStatusRezerwacji(nowy_status): Zmienia status rezerwacji.

2. Definicja DZE

DZE (Diagram Związków Encji) - przedstawia logiczny model danych, w którym uwzględnia się encje (np. Biblioteka, Czytelnik, Książka), ich atrybuty oraz relacje między nimi.

Encje i ich atrybuty:

Biblioteka

- ID_Biblioteki (PK)
- Nazwa
- Adres
- Kierownik

Czytelnik

- ID_Czytelnika (PK)
- Imię

- Nazwisko
- PESEL
- Adres
- Data_Rejestracji
- Status

Książka

- ID_Ksiazki (PK)
- Tytuł
- Autor
- ISBN
- Rok_Wydania
- Status

Wypożyczenie

- ID_Wypożyczenia (PK)
- ID_Czytelnika (FK)
- ID_Ksiazki (FK)
- Data_Wypożyczenia
- Termin_Zwrotu
- Data_Zwrotu
- Status

Rezerwacja

- ID_Rezerwacji (PK)
- ID_Czytelnika (FK)
- ID_Ksiazki (FK)
- Data_Rezerwacji
- Status_Rezerwacji

Relacje między encjami:

Czytelnik wypożycza książki (1:N)

Czytelnik rezerwuje książki (1:N)

Wypożyczenie i Rezerwacja odnoszą się do Książek (N:1)

Każda Biblioteka ma jednego Kierownika (1:1, atrybut)

3. Transformacja DZE do modelu relacyjnego – Normalizacja

I postać normalna (1NF)

Wszystkie dane są atomowe (np. pojedyncze pola jak Imię, PESEL, Data_Rejestracji, itp.)

II postać normalna (2NF)

Wszystkie atrybuty niekluczowe są zależne funkcjonalnie od całego klucza głównego.

Nie ma złożonych kluczy głównych, ta postać jest zachowana automatycznie.

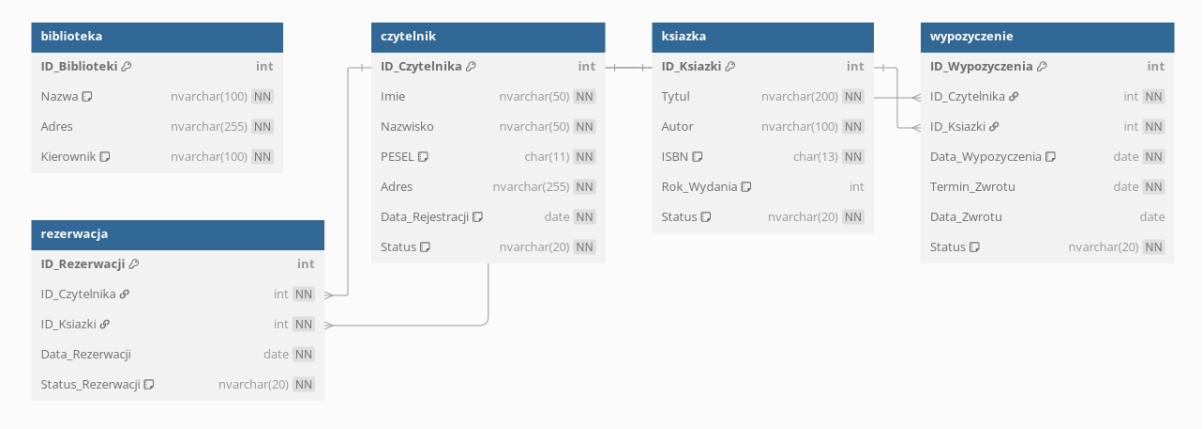
III postać normalna (3NF)

Brak zależności przechodnich — np. dane o książce nie zależą od ID wypożyczenia, tylko są przechowywane w osobnej tabeli Książka.

Model relacyjny wynikowy (tabele):

- Biblioteka(ID_Biblioteki, Nazwa, Adres, Kierownik)
- Czytelnik(ID_Czytelnika, Imie, Nazwisko, PESEL, Adres, Data_Rejestracji, Status)
- Ksiazka(ID_Ksiazki, Tytul, Autor, ISBN, Rok_Wydania, Status)
- Wypozyczenie(ID_Wypozyczenia, ID_Czytelnika, ID_Ksiazki, Data_Wypozyczenia, Termin_Zwrotu, Data_Zwrotu, Status)
- Rezerwacja(ID_Rezerwacji, ID_Czytelnika, ID_Ksiazki, Data_Rezerwacji, Status_Rezerwacji)

Każda tabela posiada klucz główny oraz klucze obce i odpowiednie ograniczenia (CHECK, NOT NULL, FOREIGN KEY, TRIGGER).



4. Definicja zasad poprawności danych

Tabela: Biblioteka

Pole	Reguła poprawności	Opis
ID_Biblioteki	PRIMARY KEY	Wartość unikalna, nie może się powtarzać.
Nazwa	NOT NULL, CHECK (Nazwa NOT LIKE '%[!@#$%^&*(),.?":{}|<>]%' AND Nazwa <> '')	Wartość jest wymagana (nie może być pusta) i nie może zawierać znaków specjalnych.
Adres	NOT NULL	Adres musi być podany.
Kierownik	NOT NULL (przez CHECK)	Kierownik musi być przypisany (nie może być NULL).

Tabela: Czytelnik

Pole	Reguła poprawności	Opis
ID_Czytelnika	PRIMARY KEY	Unikalny identyfikator czytelnika.
Imie	NOT NULL	Imię nie może być puste.
Nazwisko	NOT NULL	Nazwisko nie może być puste.
PESEL	NOT NULL, CHECK (ISNUMERIC = 1 AND LEN = 11)	Musi zawierać dokładnie 11 cyfr.
Adres	NOT NULL	Pole wymagane.
Data_Rejestracji	NOT NULL, CHECK (<= GETDATE())	Data rejestracji nie może być z przyszłości.
Status	NOT NULL, CHECK (IN ('Aktywny', 'Nieaktywny'))	Dozwolone tylko dwie wartości.

Tabela: Książka

Pole	Reguła poprawności	Opis
ID_Ksiazki	PRIMARY KEY	Unikalny identyfikator książki.
Tytul	NOT NULL	Tytuł książki jest wymagany.
Autor	NOT NULL	Autor musi być podany.
ISBN	NOT NULL, CHECK (ISNUMERIC = 1 AND LEN = 13)	ISBN musi mieć 13 cyfr.
Rok_Wydania	CHECK (BETWEEN 1000 AND YEAR(GETDATE()))	Rok musi być realistyczny i nie z przyszłości.
Status	NOT NULL, CHECK (IN ('Dostępna', 'Wypożyczona', 'Zarezerwowana', 'Usunięta'))	Ograniczony zbiór wartości.

Tabela: Wypożyczenie

Pole	Reguła poprawności	Opis
ID_Wypozyczenia	PRIMARY KEY	Unikalny identyfikator wypożyczenia.
ID_Czytelnika	FOREIGN KEY	Musi istnieć w tabeli Czytelnik.
ID_Ksiazki	FOREIGN KEY	Musi istnieć w tabeli Ksiazka.
Data_Wypozyczenia	NOT NULL, CHECK (<= GETDATE())	Nie może być z przyszłości.
Termin_Zwrotu	Trigger do sprawdzania, czy Termin_Zwrotu > Data_Wypozyczenia	Musi być późniejszy niż data wypożyczenia.
Status	NOT NULL, CHECK (IN ('W trakcie', 'Zwrócona', 'Przeterminowana'))	Określa stan wypożyczenia.

Tabela: Rezerwacja

Pole	Reguła poprawności	Opis
ID_Rezerwacji	PRIMARY KEY	Unikalny identyfikator rezerwacji.
ID_Czytelnika	FOREIGN KEY	Musi istnieć w tabeli Czytelnik.
ID_Ksiazki	FOREIGN KEY	Musi istnieć w tabeli Ksiazka.
Data_Rezerwacji	Trggier do sprawdzania czy Data_Rezerwacji < CAST(GETDATE() AS DATE)	Nie może być z przeszłości.

Status_Rezerwacji	NOT NULL, CHECK (IN ('Aktywna', 'Anulowana', 'Zrealizowana'))	Tylko określone statusy są dozwolone.

5. Definicja schematu bazy danych z implementacją deklaratywnych metod sprawdzania poprawności danych

```
USE Biblioteka;
GO

-- Tabela: Biblioteka
CREATE TABLE Biblioteka (
    ID_Biblioteki INT PRIMARY KEY,
    Nazwa NVARCHAR(100) NOT NULL CHECK (Nazwa NOT LIKE '%[!@#$%^&*(),.?":{}|<>]%'
AND Nazwa <> ''),
    Adres NVARCHAR(255) NOT NULL,
    Kierownik NVARCHAR(100),
    CONSTRAINT CK_Biblioteka_Kierownik_NotEmpty CHECK (Kierownik IS NOT NULL)
);

-- Tabela: Czytelnik
CREATE TABLE Czytelnik (
    ID_Czytelnika INT PRIMARY KEY,
    Imie NVARCHAR(50) NOT NULL,
    Nazwisko NVARCHAR(50) NOT NULL,
    PESEL CHAR(11) NOT NULL,
    Adres NVARCHAR(255) NOT NULL,
    Data_Rejestracji DATE NOT NULL CHECK (Data_Rejestracji <= GETDATE()),
    Status NVARCHAR(20) NOT NULL CHECK (Status IN ('Aktywny', 'Nieaktywny')),
    CONSTRAINT CK_PESEL_Format CHECK (ISNUMERIC(PESEL) = 1 AND LEN(PESEL) = 11)
);

-- Tabela: Ksiazka
CREATE TABLE Ksiazka (
    ID_Ksiazki INT PRIMARY KEY,
    Tytul NVARCHAR(200) NOT NULL,
    Autor NVARCHAR(100) NOT NULL,
    ISBN CHAR(13) NOT NULL,
    Rok_Wydania INT CHECK (Rok_Wydania BETWEEN 1000 AND YEAR(GETDATE())),
    Status NVARCHAR(20) NOT NULL CHECK (Status IN ('Dostępna', 'Wypożyczona',
'Zarezerwowana', 'Usunięta')),
    CONSTRAINT CK_ISBN_Format CHECK (ISNUMERIC(ISBN) = 1 AND LEN(ISBN) = 13)
);

-- Tabela: Wypozyczenie
CREATE TABLE Wypozyczenie (
    ID_Wypozyczenia INT PRIMARY KEY,
    ID_Czytelnika INT NOT NULL,
    ID_Ksiazki INT NOT NULL,
    Data_Wypozyczenia DATE NOT NULL CHECK (Data_Wypozyczenia <= GETDATE()),
    Termin_Zwrotu DATE NOT NULL,
    Data_Zwrotu DATE,
    Status NVARCHAR(20) NOT NULL CHECK (Status IN ('W trakcie', 'Zwrócona',
'Przeterminowana')),
    FOREIGN KEY (ID_Czytelnika) REFERENCES Czytelnik(ID_Czytelnika),
    FOREIGN KEY (ID_Ksiazki) REFERENCES Ksiazka(ID_Ksiazki)
);

-- Trigger: Walidacja dat wypożyczenia
```

```
GO
CREATE TRIGGER TRG_Wypozyczenie_Termin
ON Wypozyczenie
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE Termin_Zwrotu <= Data_Wypozyczenia
    )
    BEGIN
        RAISERROR ('Termin zwrotu musi być późniejszy niż data wypożyczenia.', 16,
1);
        ROLLBACK TRANSACTION;
    END
END;
GO

-- Tabela: Rezerwacja (bez błędnego CHECK, używamy triggera)
CREATE TABLE Rezerwacja (
    ID_Rezerwacji INT PRIMARY KEY,
    ID_Czytelnika INT NOT NULL,
    ID_Ksiazki INT NOT NULL,
    Data_Rezerwacji DATE NOT NULL,
    Status_Rezerwacji NVARCHAR(20) NOT NULL CHECK (Status_Rezerwacji IN
('Aktywna', 'Anulowana', 'Zrealizowana')),
    FOREIGN KEY (ID_Czytelnika) REFERENCES Czytelnik(ID_Czytelnika),
    FOREIGN KEY (ID_Ksiazki) REFERENCES Ksiazka(ID_Ksiazki)
);
GO

-- Trigger: Walidacja daty rezerwacji
CREATE TRIGGER TRG_Rezerwacja_Data
ON Rezerwacja
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE Data_Rezerwacji < CAST(GETDATE() AS DATE)
    )
    BEGIN
        RAISERROR('Data rezerwacji nie może być wcześniejsza niż dzisiejsza.', 16,
1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

6. Implementacja niedeklaratywnych mechanizmów sprawdzania poprawności danych

Procedury dla klasy: Biblioteka

```
-- Dodaj kierownika
CREATE PROCEDURE DodajKierownika
```

```
    @ID_Biblioteki INT,
    @Kierownik NVARCHAR(100)
AS
BEGIN
    UPDATE Biblioteka
    SET Kierownik = @Kierownik
    WHERE ID_Biblioteki = @ID_Biblioteki;
END
GO

-- Aktualizuj kierownika
CREATE PROCEDURE AktualizujKierownika
    @ID_Biblioteki INT,
    @NowyKierownik NVARCHAR(100)
AS
BEGIN
    UPDATE Biblioteka
    SET Kierownik = @NowyKierownik
    WHERE ID_Biblioteki = @ID_Biblioteki;
END
GO

-- Usuń kierownika
CREATE PROCEDURE UsunKierownika
    @ID_Biblioteki INT
AS
BEGIN
    UPDATE Biblioteka
    SET Kierownik = NULL
    WHERE ID_Biblioteki = @ID_Biblioteki;
END
GO
```

Procedury dla klasy: Czytelnik

```
-- Dodaj czytelnika
CREATE PROCEDURE DodajCzytelnika
    @ID_Czytelnika INT,
    @Imie NVARCHAR(50),
    @Nazwisko NVARCHAR(50),
    @PESEL CHAR(11),
    @Adres NVARCHAR(255),
    @Data_Rejestracji DATE,
    @Status NVARCHAR(20)
AS
BEGIN
    INSERT INTO Czytelnik (ID_Czytelnika, Imie, Nazwisko, PESEL, Adres,
Data_Rejestracji, Status)
    VALUES (@ID_Czytelnika, @Imie, @Nazwisko, @PESEL, @Adres, @Data_Rejestracji,
@Status);
END
GO

-- Aktualizuj dane czytelnika
CREATE PROCEDURE AktualizujDaneCzytelnika
    @ID_Czytelnika INT,
    @Imie NVARCHAR(50),
    @Nazwisko NVARCHAR(50),
    @PESEL CHAR(11),
    @Adres NVARCHAR(255)
```

```
AS
BEGIN
    UPDATE Czytelnik
    SET Imie = @Imie, Nazwisko = @Nazwisko, PESEL = @PESEL, Adres = @Adres
    WHERE ID_Czytelnika = @ID_Czytelnika;
END
GO

-- Dezaktywuj czytelnika
CREATE PROCEDURE DezaktywujCzytelnika
    @ID_Czytelnika INT
AS
BEGIN
    UPDATE Czytelnik
    SET Status = 'Nieaktywny'
    WHERE ID_Czytelnika = @ID_Czytelnika;
END
GO

-- Usuń czytelnika
CREATE PROCEDURE UsunCzytelnika
    @ID_Czytelnika INT
AS
BEGIN
    IF EXISTS (SELECT 1 FROM Wypozyczenie WHERE ID_Czytelnika = @ID_Czytelnika)
    BEGIN
        RAISERROR('Czytelnik ma aktywne wypożyczenia.', 16, 1);
        RETURN;
    END
    DELETE FROM Czytelnik WHERE ID_Czytelnika = @ID_Czytelnika;
END
GO
```

Procedury dla klasy: Ksiazka

```
-- Dodaj książkę
CREATE PROCEDURE DodajKsiazke
    @ID_Ksiazki INT,
    @Tytul NVARCHAR(200),
    @Autor NVARCHAR(100),
    @ISBN CHAR(13),
    @Rok_Wydania INT,
    @Status NVARCHAR(20)
AS
BEGIN
    INSERT INTO Ksiazka (ID_Ksiazki, Tytul, Autor, ISBN, Rok_Wydania, Status)
    VALUES (@ID_Ksiazki, @Tytul, @Autor, @ISBN, @Rok_Wydania, @Status);
END
GO

-- Aktualizuj dane książki
CREATE PROCEDURE AktualizujDaneKsiazki
    @ID_Ksiazki INT,
    @Tytul NVARCHAR(200),
    @Autor NVARCHAR(100),
    @ISBN CHAR(13),
    @Rok_Wydania INT
AS
BEGIN
```

```
    UPDATE Ksiazka
    SET Tytul = @Tytul, Autor = @Autor, ISBN = @ISBN, Rok_Wydania = @Rok_Wydania
    WHERE ID_Ksiazki = @ID_Ksiazki;
END
GO

-- Zmień status książki
CREATE PROCEDURE ZmienStatusKsiazki
    @ID_Ksiazki INT,
    @NowyStatus NVARCHAR(20)
AS
BEGIN
    UPDATE Ksiazka
    SET Status = @NowyStatus
    WHERE ID_Ksiazki = @ID_Ksiazki;
END
GO

-- Usuń książkę
CREATE PROCEDURE UsunKsiazke
    @ID_Ksiazki INT
AS
BEGIN
    DELETE FROM Ksiazka WHERE ID_Ksiazki = @ID_Ksiazki;
END
GO
```

Procedury dla klasy: Wypozyczenie

```
-- Zarejestruj wypożyczenie
CREATE PROCEDURE ZarejestrujWypozyczenie
    @ID_Wypozyczenia INT,
    @ID_Czytelnika INT,
    @ID_Ksiazki INT,
    @Data_Wypozyczenia DATE,
    @Termin_Zwrotu DATE,
    @Status NVARCHAR(20)
AS
BEGIN
    INSERT INTO Wypozyczenie (ID_Wypozyczenia, ID_Czytelnika, ID_Ksiazki,
Data_Wypozyczenia, Termin_Zwrotu, Status)
    VALUES (@ID_Wypozyczenia, @ID_Czytelnika, @ID_Ksiazki, @Data_Wypozyczenia,
@Termin_Zwrotu, @Status);
END
GO

-- Zaktualizuj status wypożyczenia
CREATE PROCEDURE ZaktualizujStatusWypozyczenia
    @ID_Wypozyczenia INT,
    @NowyStatus NVARCHAR(20)
AS
BEGIN
    UPDATE Wypozyczenie
    SET Status = @NowyStatus
    WHERE ID_Wypozyczenia = @ID_Wypozyczenia;
END
GO

-- Zarejestruj zwrot
CREATE PROCEDURE ZarejestrujZwrot
```

```
    @ID_Wypozyczenia INT,
    @Data_Zwrotu DATE
AS
BEGIN
    UPDATE Wypozyczenie
    SET Data_Zwrotu = @Data_Zwrotu, Status = 'Zwrócona'
    WHERE ID_Wypozyczenia = @ID_Wypozyczenia;

    DECLARE @ID_Ksiazki INT;
    SELECT @ID_Ksiazki = ID_Ksiazki FROM Wypozyczenie WHERE ID_Wypozyczenia =
@ID_Wypozyczenia;

    UPDATE Ksiazka
    SET Status = 'Dostępna'
    WHERE ID_Ksiazki = @ID_Ksiazki;
END
GO

-- Usuń wypożyczenie
CREATE PROCEDURE UsunWypozyczenie
    @ID_Wypozyczenia INT
AS
BEGIN
    DELETE FROM Wypozyczenie WHERE ID_Wypozyczenia = @ID_Wypozyczenia;
END
GO
```

7. Wprowadzenie danych przykładowych

```
--Biblioteka
INSERT INTO Biblioteka (ID_Biblioteki, Nazwa, Adres, Kierownik)
VALUES
(1, 'Biblioteka Główna', 'ul. Książkowa 1, 00-001 Warszawa', 'Anna Kowalska');

-- Czytelnik
INSERT INTO Czytelnik (ID_Czytelnika, Imie, Nazwisko, PESEL, Adres,
Data_Rejestracji, Status)
VALUES
(1, 'Jan', 'Nowak', '90010112345', 'ul. Czytelnicza 5, 00-002 Warszawa', '2023-05-
10', 'Aktywny'),
(2, 'Maria', 'Wiśniewska', '85031567890', 'ul. Biblioteczna 2, 00-003 Warszawa',
'2024-01-15', 'Aktywny');

--Ksiazka
INSERT INTO Ksiazka (ID_Ksiazki, Tytul, Autor, ISBN, Rok_Wydania, Status)
VALUES
(1, 'Lalka', 'Bolesław Prus', '9788373271890', 2010, 'Dostępna'),
(2, 'Pan Tadeusz', 'Adam Mickiewicz', '9788373271883', 2005, 'Dostępna'),
(3, 'Zbrodnia i kara', 'Fiodor Dostojewski', '9788373271876', 2012,
'Zarezerwowana');

--Wypozyczenie
INSERT INTO Wypozyczenie (ID_Wypozyczenia, ID_Czytelnika, ID_Ksiazki,
Data_Wypozyczenia, Termin_Zwrotu, Data_Zwrotu, Status)
VALUES
(1, 1, 1, '2024-04-01', '2024-04-15', '2024-04-14', 'Zwrócona'),
(2, 2, 2, '2024-05-01', '2024-05-20', NULL, 'W trakcie');

--Rezerwacja (dzisiejsza lub przyszła data!)
```

```
INSERT INTO Rezerwacja (ID_Rezerwacji, ID_Czytelnika, ID_Ksiazki, Data_Rezerwacji,
Status_Rezerwacji)
VALUES
(1, 1, 3, CAST(GETDATE() AS DATE), 'Aktywna'),
(2, 2, 1, DATEADD(DAY, 1, GETDATE()), 'Aktywna');
```

8. Definicja i implementacja zasad ochrony bazy danych

Instrukcja tworzenia kopii zapasowej i zasad bezpieczeństwa dla bazy danych Biblioteka:

Krok 1: Przygotowanie do tworzenia kopii zapasowej

- **Zalogowanie się do serwera SQL**
  Użyj SQL Server Management Studio (SSMS) lub połącz się ze swoim serwerem SQL z poziomu terminala lub skryptu.
- **Sprawdzenie uprawnień**
  Upewnij się, że masz przydzielone uprawnienia BACKUP DATABASE.

Krok 2: Tworzenie kopii zapasowej bazy danych Biblioteka

- **Wybór bazy danych**

```
USE Biblioteka;
GO
```

- **Tworzenie pełnej kopii zapasowej**

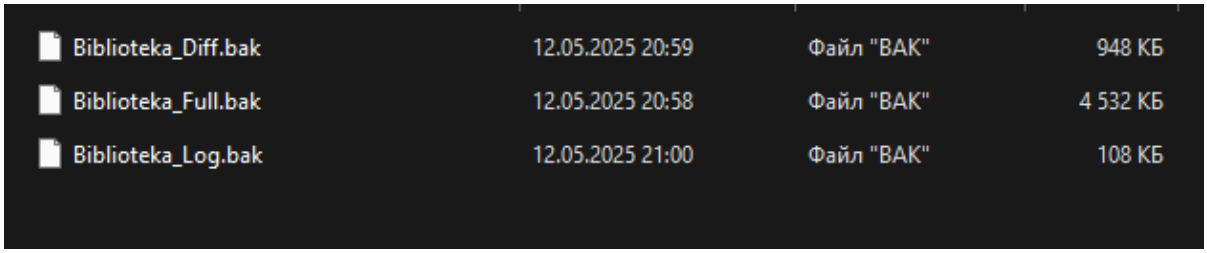
```
BACKUP DATABASE Biblioteka
TO DISK = 'C:\Backup\Biblioteka_Full.bak'
WITH FORMAT,
     MEDIANAME = 'BibliotekaBackup',
     NAME = 'Pelna kopia zapasowa bazy Biblioteka';
GO
```

- **Tworzenie różnicowej kopii zapasowej**

```
BACKUP DATABASE Biblioteka
TO DISK = 'C:\Backup\Biblioteka_Diff.bak'
WITH DIFFERENTIAL,
     MEDIANAME = 'BibliotekaBackup',
     NAME = 'Roznicowa kopia zapasowa bazy Biblioteka';
GO
```

- **Tworzenie kopii zapasowej dziennika transakcji**

```
BACKUP LOG Biblioteka
TO DISK = 'C:\Backup\Biblioteka_Log.bak'
WITH NOFORMAT,
     MEDIANAME = 'BibliotekaBackup',
     NAME = 'Kopia zapasowa dziennika transakcji bazy Biblioteka';
GO
```



Krok 3: Automatyzacja kopii zapasowych

- **Tworzenie zadania w SQL Server Agent**

```
EXEC msdb.dbo.sp_add_job @job_name = N'Backup Biblioteka - Pelna';

EXEC msdb.dbo.sp_add_jobstep
    @job_name = N'Backup Biblioteka - Pelna',
    @step_name = N'BackupFullStep',
    @subsystem = N'TSQL',
    @command = N'BACKUP DATABASE Biblioteka TO DISK =
''C:\Backup\Biblioteka_Full.bak'' WITH FORMAT;',
    @on_success_action = 1,
    @on_fail_action = 2;

EXEC msdb.dbo.sp_add_schedule
    @schedule_name = N'WeeklyFullBackup',
    @freq_type = 4,  -- Weekly
    @freq_interval = 1,
    @active_start_time = 010000; -- 01:00 AM

EXEC msdb.dbo.sp_attach_schedule
    @job_name = N'Backup Biblioteka - Pelna',
    @schedule_name = N'WeeklyFullBackup';

EXEC msdb.dbo.sp_add_jobserver @job_name = N'Backup Biblioteka - Pelna';
```

Zasady Ochrony Bazy Danych Biblioteka:

- **Regularne kopie zapasowe**
  - Pełne co tydzień
  - Różnicowe codziennie
  - Logi transakcyjne co godzinę
- **Bezpieczne przechowywanie**
  - Oddzielny serwer lub zaszyfrowany nośnik
  - Szyfrowanie kopii zapasowych
- **Ograniczony dostęp**
  - Tylko administratorzy mają dostęp do funkcji backupu i restore
  - Użytkownicy mają tylko niezbędne uprawnienia
- **Testy odzyskiwania**
  - Regularne testy restore z kopii zapasowej
- **Logowanie i monitoring**

- Logowanie operacji backupu i restore
- Powiadomienia o niepowodzeniu

Kontrola dostępu i bezpieczeństwo danych:

- **Tworzenie ról i uprawnień**

```
CREATE ROLE Rola_Bibliotekarza;
CREATE ROLE Rola_Administratora;

GRANT SELECT, INSERT, UPDATE ON Ksiazka TO Rola_Bibliotekarza;
GRANT SELECT, INSERT, UPDATE ON Czytelnik TO Rola_Bibliotekarza;
GRANT SELECT, INSERT, UPDATE, DELETE ON Rezerwacja TO Rola_Bibliotekarza;

GRANT CONTROL ON DATABASE::Biblioteka TO Rola_Administratora;
```

- **Przypisywanie użytkowników do ról**

```
ALTER ROLE Rola_Bibliotekarza ADD MEMBER Uzytkownik_Bibliotekarz;
ALTER ROLE Rola_Administratora ADD MEMBER Admin_Biblioteki;
```

- **Szyfrowanie i audyt**
- Używanie SSL/TLS do połączeń
- Audyt logowań i operacji DML/DCL
- Zgodność z RODO (np. anonimizacja PESEL)