Uniwersytet Jagielloński

Bazy danych

Projekt zaliczeniowy

Baza danych szkoły średniej

Paweł Gmerek Katarzyna Płoskonka

1 Założenia wstępne projektu

1.1 Cel projektu

Projekt ma na celu realizować podstawowe funkcje bazy danych używanej do zarządzania szkołą. Gromadzi ona dane

1. **Uczniów, Rodziców oraz Nauczycieli:** Dane takie jak imię, nazwisko, czy numer telefonu. Warto również dodać, każda osoba dziedziczy po encji *Account* która nadaje hasło i login każdemu użytkownikowi
2. **Ocen:** Został zaimplementowany system zarządzania ocenami uczniów, który obejmuje:
   - Oceny ze sprawdzianów
   - Oceny za aktywność (plusy)
   - Naganne oceny z zachowania za uwagi
3. **Biblioteki szkolnej:** W naszej bazie danych znajduje się funkcjonujący system bibliotekii szkolnej w której każdy uczeń może wypożyczać oraz oddawać książki. Baza danych śledzi ilość dostępnych książek oraz nalicza odpowiednią karę po przekroczeniu terminu wypożyczenia
4. **Przedmiotów:** Baza została podzielona na tabele przedmiotów, klas oraz lekcji co pozwala na układanie planów dla każdej klasy z osobna.

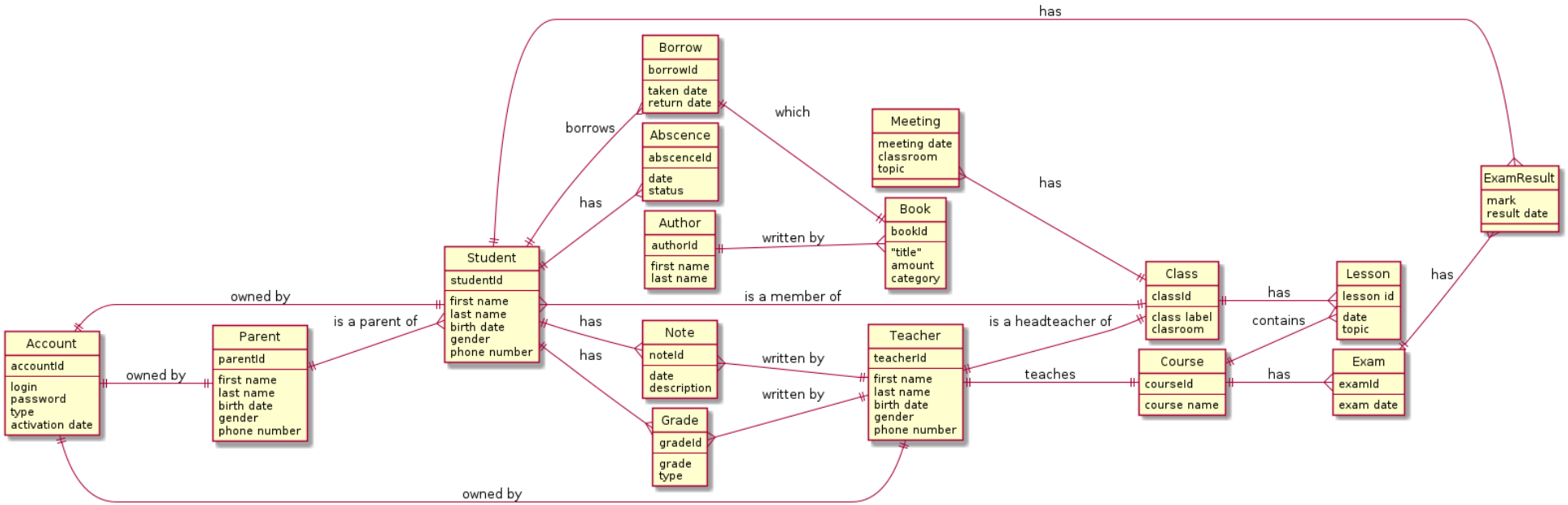
1.2 Ograniczenia przyjęte przy projektowaniu

Podczas projektowania bazy danych zdecydowaliśmy się na pewne ograniczenia, między innymi

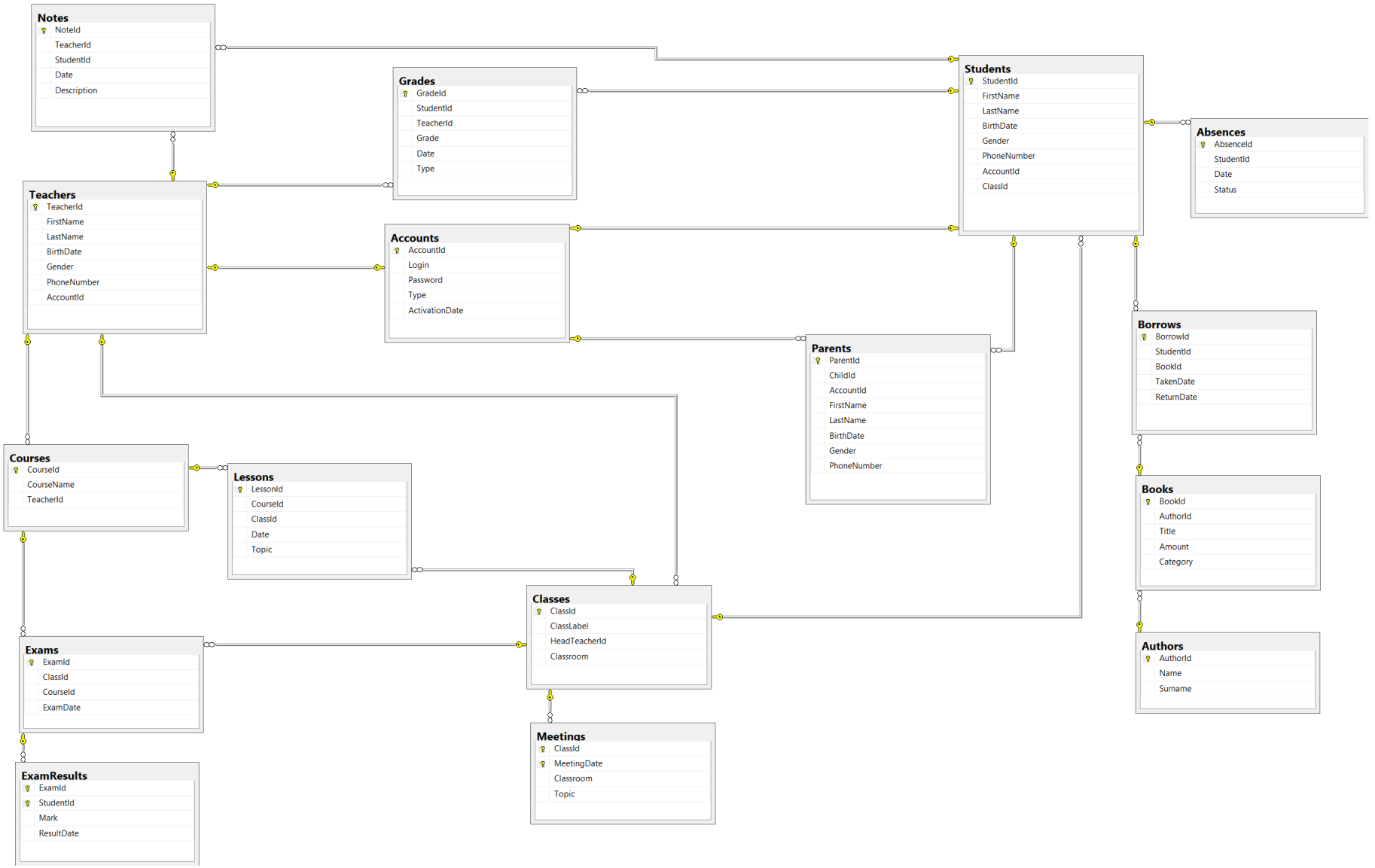
- Każdy nauczyciel uczy dokładnie jednego przedmiotu
- Każdy rodzic ma dokładnie jedno dziecko
- Każda książka ma tylko jednego autora

2 Struktura bazy

2.1 Diagram związków encji



2.2 Schemat relacyjny bazy



2.3 Dodatkowe więzy integralności

W bazie znajdują się dodatkowe więzy integralności takie jak

- **is_password_valid** - sprawdzający czy długość hasła przekracza 7 znaków
- **is_student_phone_valid** - sprawdzający czy został wprowadzony prawidłowy numer telefonu przez ucznia
- **is_teacher_phone_valid** - tak samo tylko sprawdzający numer telefonu nauczyciela
- **is_mark_valid** - sprawdzający czy ocena jest w przedziale [1, 6]

3 Opis stworzonych widoków

3.1 Lista autorów wraz z ilością ich wypożyczonych książek

```
01 | CREATE VIEW authors_with_books_borrows AS
02 |     SELECT A.Name [Imie], A.Surname [Nazwisko], B.Title [Tytul], COUNT(Borrows.BookId) [Liczba wypozyczen]
03 |     FROM Borrows
04 |     RIGHT JOIN Books B ON Borrows.BookID = B.BookId
05 |     JOIN Authors A ON A.AuthorId = B.AuthorId
06 |     GROUP BY A.Name, A.Surname, B.Title
```

3.2 Lista zagrożonych uczniów

```
01 | CREATE VIEW endangered_students AS
02 |     SELECT S.FirstName [Imie], S.LastName [Nazwisko],C.CourseName [Przedmiot], ROUND(AVG(Mark),2) [srednia]
03 |     FROM ExamResults ER
04 |     JOIN Students S ON S.StudentId = ER.StudentId
05 |     JOIN Exams E ON ER.ExamId = E.ExamId
06 |     JOIN Courses C ON E.CourseId = C.CourseId
07 |     GROUP BY S.FirstName, S.LastName, C.CourseName
08 |     HAVING ROUND(AVG(Mark),2) < 2.0
```

3.3 Lista uczniów wraz z ich średnią

```
01 | CREATE VIEW students_average_grades AS
02 |     SELECT S.FirstName [Imie], S.LastName [Nazwisko], ROUND(AVG(Grade), 2) [Srednia]
03 |     FROM Students S
04 |     LEFT JOIN Grades G ON S.StudentId = G.StudentId
05 |     GROUP BY S.FirstName, S.LastName
```

3.4 Wyróżnieni uczniowie

```
01 | CREATE VIEW students_with_honours AS
02 |     SELECT S.FirstName [Imie], S.LastName [Nazwisko], ROUND(AVG(Grade), 2)[U+FFFD]rednia
03 |     FROM Grades G
04 |     JOIN Students S ON S.StudentId = G.StudentId
05 |     GROUP BY S.FirstName, S.LastName
06 |     HAVING ROUND(AVG(Grade), 2) >= 4.75
```

3.5 Zestawienie nauczycieli z ilością ich przepracowanych godzin

```
01 | CREATE VIEW hours_worked_per_teacher AS
02 |     SELECT T.FirstName Imie, T.LastName Nazwisko, SUM(P.hours) [Przepracowane godziny]
03 |     FROM Teachers T
04 |     JOIN Courses C ON T.TeacherId = C.TeacherId
05 |     JOIN (SELECT CourseId, COUNT(*) hours FROM Lessons GROUP BY CourseId) P ON P.CourseId = C.CourseId
06 |     GROUP BY T.FirstName, T.LastName
```

4 Opis stworzonych funkcji

4.1 Funkcja przyjmująca ID ucznia i zwracająca jego nieusprawiedliwione nieobecności

Kod źródłowy w pliku **get-absences-by-id.sql**. Funkcja przyjmująca ID ucznia i zwracająca jego wszystkie nieusprawiedliwione obecności.

```sql
CREATE FUNCTION get_absences (
    @StudentId INT
)
RETURNS TABLE
AS
RETURN
    SELECT
        S.FirstName,
        S.LastName,
        A.Date
    FROM Students S
    JOIN Absences A
    ON A.StudentId = S.StudentId
    WHERE @StudentId = S.StudentId AND A.Status = 'Nieusprawiedliwione'
```

4.2 Funkcja get_lessons

Kod źródłowy w pliku **get-lessons-by-course.sql**. Funkcja przyjmująca ID kursu i zwracająca lekcje z tego kursu w porządku od najnowszej do najstarszej.

```sql
CREATE FUNCTION get_lessons (
    @CourseId INT
)
RETURNS TABLE
AS
RETURN
    SELECT
        L.ClassLabel,
        L.Date,
        L.Topic
    FROM Lessons L
    JOIN Courses C ON C.CourseId = L.CourseId
    WHERE @CourseId = C.CourseId
    ORDER BY L.Date DESC
```

4.3 Funkcja get_schedule

Kod źródłowy w pliku **get_lesson_schedule.sql**. Funkcja przyjmująca ID klasy i zwracająca jej tygodniowy plan lekcji w formie tabeli, której kolumnami są dni tygodnia a wierszami planowane lekcje w danym dniu.

```
01 | CREATE FUNCTION get_schedule (
02 |     @ClassId
03 | )
04 | RETURNS TABLE
05 | AS
06 | RETURN
07 |     SELECT
08 |         H.Hours [*],
09 |         P.CourseName [Poniedzialek],
10 |         T.CourseName [Wtorek],
11 |         W.CourseName [Sroda],
12 |         TH.CourseName [Czwartek],
13 |         F.CourseName [Piatek]
14 |     FROM (SELECT DISTINCT FORMAT(Date,'HH:mm') Hours From Lessons) H
15 |                 LEFT JOIN (SELECT DISTINCT C.CourseName, FORMAT(Date,'HH:mm') Hours
16 |                         FROM Lessons L
17 |                         JOIN Courses C ON C.CourseId = L.CourseId
18 |                         WHERE ClassId = @CourseId AND DATENAME(DW, L.Date) = 'Monday' ) P ON H.Hours = P.Hours
19 |                 LEFT JOIN (SELECT DISTINCT C.CourseName, FORMAT(Date,'HH:mm') Hours
20 |                         FROM Lessons L
21 |                         JOIN Courses C ON C.CourseId = L.CourseId
22 |                         WHERE ClassId = @CourseId AND DATENAME(DW, L.Date) = 'Tuesday' ) T ON H.Hours = T.Hours
23 |                 LEFT JOIN (SELECT DISTINCT C.CourseName, FORMAT(Date,'HH:mm') Hours
24 |                         FROM Lessons L
25 |                         JOIN Courses C ON C.CourseId = L.CourseId
26 |                         WHERE ClassId =@CourseId AND DATENAME(DW, L.Date) = 'Wednesday' ) W ON H.Hours = W.Hours
27 |                 LEFT JOIN (SELECT DISTINCT C.CourseName, FORMAT(Date,'HH:mm') Hours
28 |                         FROM Lessons L
29 |                         JOIN Courses C ON C.CourseId = L.CourseId
30 |                         WHERE ClassId = @CourseId AND DATENAME(DW, L.Date) = 'Thursday' ) TH ON H.Hours = TH.Hours
31 |                 LEFT JOIN (SELECT DISTINCT C.CourseName, FORMAT(Date,'HH:mm') Hours
32 |                         FROM Lessons L
33 |                         JOIN Courses C ON C.CourseId = L.CourseId
34 |                         WHERE ClassId = @CourseId AND DATENAME(DW, L.Date) = 'Friday' ) F ON H.Hours = F.Hours
```

4.4 Funkcja get_overdue_books

Kod źródłowy w pliku **get_overdue_books_by_id.sql**. Funkcja przyjmująca ID ucznia i zwracająca listę książek, które zostały przez niego wypożyczone, ale jeszcze nie zostały zwrócone.

```
01 | CREATE FUNCTION get_overdue_books (
02 |     @StudentId INT
03 | )
04 | RETURNS TABLE
05 | AS
06 | RETURN
07 |     SELECT S.FirstName [Imi[U+FFFD], S.LastName [Nazwisko], B.TakenDate [Data wypozyczenia], Books.Title [Ty[U+FFFD]
08 |     FROM Students S
09 |     JOIN Borrows B ON S.StudentId = B.StudentId
10 |     JOIN Books ON B.BookId = Books.BookId
11 |     WHERE @StudentId = S.StudentId AND B.ReturnDate IS NULL
```

4.5 Funkcja get_overdue_exams

Kod źródłowy w pliku **get_overdue_exams_by_teacher_id.sql**. Funkcja przyjmująca ID nauczyciela i zwracająca listę sprawdzianów przeprowadzonych przez niego, które nie zostały jeszcze ocenione (nie ma żadnych dostępnych wyników tych sprawdzianów).

```
01 | CREATE FUNCTION get_overdue_exams (
02 |     @TeacherId INT
03 | )
04 | RETURNS TABLE
05 | AS
06 | RETURN
07 |     SELECT DISTINCT Classes.ClassLabel [Klasa], Courses.CourseName [Kurs], Exams.ExamDate [Data sprawdzianu]
08 |     FROM Exams
09 |     JOIN Classes ON Exams.ClassId = Classes.ClassId
10 |     JOIN Courses ON Exams.CourseId = Courses.CourseId
11 |     WHERE @TeacherId = Courses.TeacherId AND NOT EXISTS (SELECT * FROM ExamResults WHERE Exams.ExamId = ExamResults.
       ExamId)
```

5 Opis stworzonych wyzwalaczy

5.1 Wyzwalacz grade_for_activity

Kod źródłowy w pliku **tr_grade_for_activity.sql**. Po wpisaniu uczniowi piątego plusa automatycznie wystawiana jest mu ocena bardzo dobra z aktywności.

```
01 | CREATE TRIGGER grade_for_activity ON Grades
02 |     AFTER INSERT
03 |         IF((SELECT Description FROM inserted) = '+'
04 |         AND (SELECT count(*) FROM Grades WHERE StudentId = (SELECT StudentId FROM inserted)
05 |         AND Description = '+')%5 = 0)
06 |         BEGIN
07 |             INSERT INTO Grades((SELECT StudentId from inserted),
08 |                                (SELECT TeacherId from inserted),
09 |                                 5, GETDATE(), 'Aktywnosc')
10 |         END
11 |     END
```

5.2 Wyzwalacz grade_from_exam

Kod źródłowy w pliku **tr_grade_for_exam.sql**. Po wpisaniu wyniku z egzaminu danego ucznia (operacja wstawiania w ExamResults) automatycznie wystawiana jest mu ocena do tabeli Grades.

```
01 | CREATE TRIGGER grade_from_exam ON ExamResults
02 |     FOR INSERT AS
03 |         INSERT INTO Grades VALUES ((SELECT StudentId FROM inserted), (
04 |             SELECT T.TeacherId FROM Teachers
05 |             JOIN Courses C ON C.TeacherId = T.TeacherId
06 |             JOIN Exams E ON E.CourseId = C.CourseId
07 |             JOIN ExamResults ER ON ER.ExamId = (SELECT ExamId FROM inserted)),
08 |             (SELECT Mark FROM inserted),
09 |             (SELECT ResultDate from inserted),
10 |             'Sprawdzian')
11 |     END
```

5.3 Wyzwalacz tr_grade_for_notes

Kod źródłowy w pliku **tr_grade_for_notes.sql**. Po wystawieniu uczniowi trzeciej uwagi automatycznie jest mu wystawiana ocena niedostateczna.

```
01 | CREATE TRIGGER tr_grade_for_notes ON Notes
02 |     AFTER INSERT AS
03 |         IF(SELECT count(*) FROM Notes WHERE StudentId = (SELECT StudentId FROM inserted)) % 3 = 0
04 |         BEGIN
05 |             INSERT INTO Grades VALUES
06 |                 ((SELECT StudentId FROM inserted),
07 |                 (SELECT TeacherId FROM inserted),1, GETDATE(), 'Trzy uwagi')
08 |         END
09 |     END
```

5.4 Wyzwalacz absences_for_note

Kod źródłowy w pliku **tr_absences_for_note.sql**. Po wystawieniu uczniowi piątej nieusprawiedliwionej nieobecności na zajęciach, automatycznie wystawiana jest mu uwaga od wychowawcy za nieobecności.

```
01 | CREATE TRIGGER absences_for_note ON Absences
02 |     AFTER INSERT AS
03 |         IF ((SELECT Status FROM inserted) = 'Nieusprawiedliwione' AND
04 |         (SELECT count(*) FROM Absences WHERE StudentId = (SELECT StudentId FROM inserted) AND Status = '
       Nieusprawiedliwione')%5 = 0)
05 |             BEGIN
06 |                 DECLARE @TeacherId INT
07 |                 DECLARE @ClassId INT
08 |                 SET @ClassId = (SELECT ClassId FROM Students WHERE Students.StudentId = (SELECT StudentID FROM
       inserted))
09 |                 SET @TeacherId = (SELECT HeadTeacherId FROM Classes WHERE Classes.ClassId = @ClassId)
10 |                 INSERT INTO Notes VALUES
11 |                     (@TeacherId, (SELECT StudentId FROM inserted), GETDATE(), 'Piec nieusprawiedliwionych nieobecnosci
       ')
12 |             END
```

5.5 Wyzwalacz exam_announcement

Kod źródłowy w pliku **tr_exam_announcement.sql**. Przy dodawaniu sprawdzianu wyzwalacz sprawdza, czy został zapowiedziany z co najmniej siedmiodniowym wyprzedzeniem. Jeśli nie, dopuszcza dodanie sprawdzianu, ale wyświetla informację, że został zapowiedziany za późno wraz z najbliższą możliwą datą sprawdzianu.

```
01 | CREATE TRIGGER exam_announcement ON Exams
02 |     FOR INSERT, UPDATE AS
03 |     IF ((SELECT ExamDate FROM inserted) < GETDATE() + 7)
04 |         PRINT 'Sprawdzian zostal zapowiedziany za pozno. Najblizszy mozliwy termin sprawdzianu to: ' + CONVERT(VARCHAR
       (100), GETDATE() + 7)
```

6 Opis stworzonych procedur składowanych

6.1 Procedura add_grade_to_student

Kod źródłowy w pliku **add_grade_to_student.sql**. Procedura wystawiająca ocenę uczniowi. Nauczyciel aby to zrobić musi się zalogować loginem i hasłem oraz podać ID ucznia, ocenę i jej typ. Procedura wyświetli błąd jeśli login bądź hasło są nieprawidłowe.

```
01 | CREATE PROCEDURE add_grade_to_student @AccountLogin NVARCHAR(50), @Password NVARCHAR(50), @StudentId INTEGER, @Grade
       FLOAT, @GradeType NVARCHAR(50) AS
02 |     DECLARE @TeacherId INT
03 |     IF EXISTS(SELECT TeacherId FROM Teachers T JOIN Accounts A ON T.AccountId = A.AccountId WHERE A.Login =
       @AccountLogin AND A.Password = @Password) BEGIN
04 |         SET @TeacherId = (SELECT TeacherId FROM Teachers T JOIN Accounts A ON T.AccountId = A.AccountId WHERE A.Login
       = @AccountLogin AND A.Password = @Password)
05 |         BEGIN TRANSACTION
06 |             INSERT INTO Grades VALUES
07 |                 (@TeacherId, @StudentId, @Grade, GETDATE(), @GradeType)
08 |         COMMIT TRANSACTION
09 |     END
10 |     ELSE BEGIN
11 |         PRINT N'Nie ma nauczyciela o takim hasle/loginie';
12 |     END
```

6.2 Procedura add_lesson

Kod źródłowy w pliku **add_lesson.sql**. Za pomocą tej procedury nauczyciel może zapisać w dzienniku lekcję wraz z jej tematem. Aby to zrobić musi zalogować się loginem i hasłem oraz podać nazwę kursu, klasy, datę i temat lekcji. Procedura wyświetli błąd jeśli login bądź hasło nauczyciela są nieprawidłowe lub nie istnieje podany kurs lub klasa.

```
01 | CREATE PROCEDURE add_lesson @AccountLogin NVARCHAR(50), @Password NVARCHAR(50), @CourseName NVARCHAR(50), @ClassLabel
       NVARCHAR(2), @Date DATE, @Topic NVARCHAR(150)
02 | AS
03 |     IF EXISTS(SELECT TeacherId FROM Teachers T JOIN Accounts A ON T.AccountId = A.AccountId WHERE A.Login =
       @AccountLogin AND A.Password = @Password)
04 |     AND EXISTS(SELECT CourseId FROM Courses WHERE CourseName = @CourseName)
05 |     AND EXISTS(SELECT ClassId FROM Classes WHERE ClassLabel = @ClassLabel)
06 |     BEGIN
07 |         DECLARE @CourseId INT, @ClassId INT
08 |         SET @CourseId = (SELECT CourseId FROM Courses WHERE CourseName = @CourseName)
09 |         SET @ClassId = (SELECT ClassId FROM Classes WHERE ClassLabel = @ClassLabel)
10 |
11 |         BEGIN TRANSACTION
12 |             INSERT INTO Lessons VALUES
13 |                 (@CourseId, @ClassId, @Date, @Topic)
14 |         COMMIT TRANSACTION
15 |     END
16 |     ELSE BEGIN
17 |         PRINT N'Co najmniej jeden z podanych parametrow jest nieprawidlowy';
18 |     END
```

6.3 Procedura borrow_book

Kod źródłowy w pliku **borrow_book.sql**. Za pomocą tej procedury można wypożyczyć książkę. Aby to zrobić należy zalogować się loginem i hasłem i podać tytuł książki. Jeśli na stanie biblioteki jest wystarczająca liczba książek, wykonuje się transakcja, która zmniejsza stan wypożyczanej książki o 1. Jeśli nie ma wystarczającej liczby książek lub nie istnieje użytkownik o podanym loginie i haśle, procedura wypisuje błąd.

```
01 | CREATE PROCEDURE borrow_book @AccountLogin nvarchar(30), @Password nvarchar(30), @Title nvarchar(30)
02 | AS
03 |     IF EXISTS(SELECT AccountId FROM Accounts WHERE Login = @AccountLogin AND Password = @Password) BEGIN
04 |             BEGIN TRANSACTION
05 |                 IF (SELECT Amount FROM Books WHERE Title = @Title) > 0 BEGIN
06 |                     INSERT INTO Borrows VALUES
07 |                         ((SELECT StudentId FROM Students S JOIN Accounts A ON S.AccountId = A.AccountId WHERE A.Login
       = @AccountLogin),
08 |                         (SELECT BookId FROM Books WHERE Title = @Title),
09 |                         GETDATE(),
10 |                                             NULL
11 |                         )
12 |                     UPDATE Books
13 |                         SET Amount = (SELECT Amount FROM Books WHERE Title = @Title) - 1
14 |                         WHERE Title = @Title
15 |                     END
16 |                 ELSE PRINT N'Nie wystarczajaca liczba ksiazek w bibliotece o podanym tytule';
17 |             COMMIT TRANSACTION
18 |     END
19 |     ELSE BEGIN
20 |             PRINT N'Nie ma uzytkownika o takim hasle/loginie';
21 |     END
```

6.4 Procedura justify_absence

Kod źródłowy w pliku **justify_absence.sql**. Procedura usprawiedliwia nieobecność ucznia. Aby to zrobić należy zalogować się loginem i hasłem oraz podać ID nieobecnośći.

```
01 | CREATE PROCEDURE justify_absence @AccountLogin nvarchar(30), @Password nvarchar(30), @AbsenceId INTEGER
02 | AS
03 |     IF EXISTS(SELECT AccountId FROM Accounts WHERE Login = @AccountLogin AND Password = @Password) BEGIN
04 |         UPDATE Absences
05 |             SET Status = 'Usprawiedliwione'
06 |         WHERE AbsenceId = @AbsenceId
07 |     END
```

6.5 Procedura move_student_to_class

Kod źródłowy w pliku **move_student_to_class.sql**. Za pomocą tej procedury nauczyciel może przenieść ucznia do innej klasy. Aby to zrobić musi zalogować się loginem i hasłem, podać imię i nazwisko ucznia oraz nazwę klasy, do której ma zostać przeniesiony. Jeśli uczeń jest już w podanej klasie bądź co najmniej jeden z parametrów jest nieprawidłowy, procedura wyświetli błąd.

```
01 | CREATE PROCEDURE move_student_to_class @AccountLogin NVARCHAR(50), @Password NVARCHAR(50), @StudentFirstName NVARCHAR
       (50), @StudentLastName NVARCHAR(50), @ClassLabel NVARCHAR(2)
02 | AS
03 |     IF EXISTS(SELECT TeacherId FROM Teachers T JOIN Accounts A ON T.AccountId = A.AccountId WHERE A.Login =
       @AccountLogin AND A.Password = @Password)
04 |     AND EXISTS(SELECT StudentId FROM Students WHERE FirstName = @StudentFirstName AND LastName = @StudentLastName)
05 |     AND EXISTS(SELECT ClassId FROM Classes WHERE ClassLabel = @ClassLabel)
06 |     BEGIN
07 |         Declare @ClassId INT
08 |         SET @ClassId = (SELECT ClassId FROM Classes WHERE ClassLabel = @ClassLabel)
09 |
10 |         IF NOT EXISTS(SELECT StudentId FROM Students WHERE FirstName = @StudentFirstName AND LastName =
       @StudentLastName AND ClassId = @ClassID)
11 |         BEGIN
12 |             BEGIN TRANSACTION
13 |                 UPDATE Students
14 |                 SET ClassId = @ClassId
15 |                 WHERE FirstName = @StudentFirstName AND LastName = @StudentLastName
16 |             COMMIT TRANSACTION
17 |         END
18 |         ELSE BEGIN
19 |             PRINT N'Uczen nalezy juz do tej klasy!'
20 |         END
21 |     END
22 |     ELSE BEGIN
23 |         PRINT N'Co najmniej jeden z podanych parametrow jest nieprawidlowy';
24 |     END
```

6.6 Procedura return_book

Kod źródłowy w pliku **return_book.sql**. Za pomocą tej procedury można zwrócić książkę. Aby to zrobić należy zalogować się loginem i hasłem oraz podać tytuł zwracanej książki. Jeśli książka została oddana po terminie, naliczana jest kara i komunikat o niej pojawia się na ekranie.

```
01 | CREATE PROCEDURE return_book @AccountLogin nvarchar(30), @Password nvarchar(30), @Title nvarchar(30)
02 | AS
03 |     IF EXISTS(SELECT AccountId FROM Accounts WHERE Login = @AccountLogin AND Password = @Password) BEGIN
04 |             BEGIN TRANSACTION
05 |                                     UPDATE Borrows
06 |                                             SET ReturnDate = GETDATE()
07 |                                     FROM Borrows B
08 |                                     JOIN Students S ON B.StudentId = S.StudentId
09 |                                     JOIN Accounts A ON A.AccountId = S.AccountId
10 |                                     JOIN Books Bo on Bo.BookId = B.BookId
11 |                                     WHERE @AccountLogin = A.Login AND Bo.Title = @Title
12 |
13 |                     DECLARE @duration INT
14 |                             DECLARE @penalty_amount  FLOAT
15 |                             DECLARE @TakenDate DATE
16 |                             DECLARE @ReturnDate DATE
17 |
18 |                             SELECT @TakenDate = (SELECT TakenDate FROM Borrows B
19 |                                             JOIN Students S ON B.StudentId = S.StudentId
20 |                                             JOIN Accounts A ON A.AccountId = S.AccountId
21 |                                             JOIN Books Bo on Bo.BookId = B.BookId
22 |                                             WHERE @AccountLogin = A.Login AND Bo.Title = @Title)
23 |
24 |                             SET @ReturnDate = GETDATE()
25 |                             SET @duration  = DATEDIFF(DAY, @TakenDate, @ReturnDate)
26 |                             SET @penalty_amount = (@duration - 28) * 2.7
27 |
28 |                 IF (@duration > 28)
29 |                         BEGIN
30 |                         PRINT N'Kara do zaplaty za opoznienie: ' + CONVERT(varchar(100),@wynik)
31 |                         END
32 |                 ELSE
33 |                         BEGIN
34 |                         PRINT N'Ksiazka oddana w terminie'
```

```
35 |
36 |                     UPDATE Books
37 |                         SET Amount = (SELECT Amount FROM Books WHERE Title = @Title) + 1
38 |                         WHERE Title = @Title
39 |             COMMIT TRANSACTION
40 |     END
41 |     ELSE BEGIN
42 |             PRINT N'Nie ma uzytkownika o takim hasle/loginie';
43 |         END
```

7 Skrypt tworzący bazę danych

7.1 Skrypt tworzący tabele

```
01 | IF OBJECT_ID('ExamResults', 'U') IS NOT NULL DROP TABLE ExamResults
02 | IF OBJECT_ID('Lessons', 'U') IS NOT NULL DROP TABLE Lessons
03 | IF OBJECT_ID('Notes', 'U') IS NOT NULL DROP TABLE Notes
04 | IF OBJECT_ID('Absences', 'U') IS NOT NULL DROP TABLE Absences
05 | IF OBJECT_ID('Meetings', 'U') IS NOT NULL DROP TABLE Meetings
06 | IF OBJECT_ID('Exams', 'U') IS NOT NULL DROP TABLE Exams
07 | IF OBJECT_ID('Borrows', 'U') IS NOT NULL DROP TABLE Borrows
08 | IF OBJECT_ID('Books', 'U') IS NOT NULL DROP TABLE Books
09 | IF OBJECT_ID('Authors', 'U') IS NOT NULL DROP TABLE Authors
10 | IF OBJECT_ID('Parents', 'U') IS NOT NULL DROP TABLE Parents
11 | IF OBJECT_ID('Grades', 'U') IS NOT NULL DROP TABLE Grades
12 | IF OBJECT_ID('Students', 'U') IS NOT NULL DROP TABLE Students
13 | IF OBJECT_ID('Classes', 'U') IS NOT NULL DROP TABLE Classes
14 | IF OBJECT_ID('Courses', 'U') IS NOT NULL DROP TABLE Courses
15 | IF OBJECT_ID('Teachers', 'U') IS NOT NULL DROP TABLE Teachers
16 | IF OBJECT_ID('Accounts', 'U') IS NOT NULL DROP TABLE Accounts
17 |
18 | GO
19 |
20 | CREATE TABLE Accounts (
21 |     AccountId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
22 |     Login NVARCHAR(50) NOT NULL,
```

```
23 |     Password NVARCHAR(50) NOT NULL,
24 |     Type NVARCHAR(1) NOT NULL,
25 |     ActivationDate DATE NOT NULL
26 | )
27 | GO
28 |
29 | CREATE TABLE Teachers (
30 |     TeacherId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
31 |     FirstName NVARCHAR(50) NOT NULL,
32 |     LastName NVARCHAR(50) NOT NULL,
33 |     BirthDate DATE NOT NULL,
34 |     Gender NVARCHAR(1) NOT NULL,
35 |     PhoneNumber VARCHAR(9) NOT NULL,
36 |     AccountId INTEGER NOT NULL UNIQUE,
37 |     CONSTRAINT TeachersFK FOREIGN KEY(AccountId) REFERENCES Accounts(AccountId)
38 |     ON UPDATE CASCADE
39 |     ON DELETE CASCADE,
40 | )
41 | GO
42 |
43 | CREATE TABLE Courses (
44 |     CourseId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
45 |     CourseName NVARCHAR(50) NOT NULL,
46 |     TeacherId INTEGER NOT NULL
47 |     CONSTRAINT CoursesFK FOREIGN KEY(TeacherId) REFERENCES Teachers(TeacherId)
48 |     ON UPDATE CASCADE
49 |     ON DELETE CASCADE
50 | )
51 | GO
52 |
53 | CREATE TABLE Classes (
54 |     ClassId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
55 |     ClassLabel NVARCHAR(2) NOT NULL,
56 |     HeadTeacherId INTEGER NULL,
57 |     Classroom INTEGER NOT NULL,
58 |     CONSTRAINT HeadTeacherFK FOREIGN KEY(HeadTeacherId) REFERENCES Teachers(TeacherId)
59 | )
60 | GO
61 |
```

```
CREATE TABLE Students (
    StudentId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    BirthDate DATE NOT NULL,
    Gender NVARCHAR(1) NOT NULL,
    PhoneNumber VARCHAR(9) NOT NULL,
    AccountId INTEGER NOT NULL UNIQUE,
    ClassId INTEGER NOT NULL
    CONSTRAINT StudentsFK FOREIGN KEY(AccountId) REFERENCES Accounts(AccountId)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
    CONSTRAINT StudentClassFK FOREIGN KEY(ClassId) REFERENCES Classes(ClassId)
)
GO
CREATE TABLE Grades (
    GradeId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
    StudentId INTEGER NOT NULL,
    TeacherId INTEGER NOT NULL,
    Grade FLOAT,
    Date DATE NOT NULL,
    Type NVARCHAR(50),
    CONSTRAINT StudentGradeFK FOREIGN KEY(StudentId) REFERENCES Students(StudentId),
    CONSTRAINT TeacherGradeFK FOREIGN KEY(TeacherId) REFERENCES Teachers(TeacherId),

)
GO
CREATE TABLE Parents (
    ParentId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
    ChildId INTEGER NOT NULL,
    AccountId INTEGER NOT NULL,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    BirthDate DATE NOT NULL,
    Gender NVARCHAR(1) NOT NULL,
    PhoneNumber VARCHAR(9) NOT NULL,
    CONSTRAINT ParentFk FOREIGN KEY(AccountId) REFERENCES Accounts(AccountId),
    CONSTRAINT ChildFK FOREIGN KEY(ChildId) REFERENCES Students(StudentId)
    ON UPDATE CASCADE
```

```
101 |     ON DELETE CASCADE
102 | )
103 | GO
104 |
105 | CREATE TABLE Meetings (
106 |     ClassId INTEGER NOT NULL,
107 |     MeetingDate DATE NOT NULL,
108 |     Classroom INTEGER NOT NULL,
109 |     Topic NVARCHAR(50) NOT NULL,
110 |         CONSTRAINT MeetingPK PRIMARY KEY(ClassId, MeetingDate),
111 |     CONSTRAINT MeetingFK FOREIGN KEY(ClassId) REFERENCES Classes(ClassId)
112 | )
113 | GO
114 |
115 | CREATE TABLE Exams (
116 |     ExamId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
117 |     ClassId INTEGER NOT NULL,
118 |     CourseId INTEGER NOT NULL,
119 |     ExamDate DATE NOT NULL,
120 |     CONSTRAINT ExamsFK FOREIGN KEY(ClassId) REFERENCES Classes(ClassId),
121 |     CONSTRAINT CourseFK FOREIGN KEY(CourseId) REFERENCES Courses(CourseId)
122 | )
123 |
124 | GO
125 | CREATE TABLE ExamResults (
126 |     ExamId INTEGER NOT NULL,
127 |     StudentId INTEGER NOT NULL,
128 |     Mark FLOAT NOT NULL,
129 |     ResultDate DATE NOT NULL,
130 |         CONSTRAINT ExamResultsPK PRIMARY KEY(ExamId, StudentId),
131 |     CONSTRAINT ExamsResultsFK FOREIGN KEY(ExamId) REFERENCES Exams(ExamId),
132 | )
133 | GO
134 |
135 | CREATE TABLE Authors (
136 |     AuthorId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
137 |     Name NVARCHAR(50) NOT NULL,
138 |     Surname NVARCHAR(50) NOT NULL,
139 | )
```

```
140 | GO
141 |
142 | CREATE TABLE Books (
143 |     BookId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
144 |     AuthorId INTEGER NOT NULL,
145 |     Title NVARCHAR(50) NOT NULL,
146 |     Amount INTEGER NOT NULL,
147 |     Category NVARCHAR(50) NOT NULL,
148 |     CONSTRAINT AuthorFK FOREIGN KEY(AuthorId) REFERENCES Authors(AuthorId)
149 | )
150 | GO
151 |
152 | CREATE TABLE Borrows (
153 |     BorrowId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
154 |     StudentId INTEGER NOT NULL,
155 |     BookId INTEGER NOT NULL,
156 |     TakenDate DATE NOT NULL,
157 |     ReturnDate DATE,
158 |     CONSTRAINT BookFK FOREIGN KEY(BookId) REFERENCES Books(BookId),
159 |     CONSTRAINT StudentBorrowsFK FOREIGN KEY(StudentId) REFERENCES Students(StudentId)
160 | )
161 | GO
162 |
163 | CREATE TABLE Absences (
164 |     AbsenceId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
165 |     StudentId INTEGER NOT NULL,
166 |     Date DATE NOT NULL,
167 |     Status NVARCHAR(50) NOT NULL,
168 |     CONSTRAINT AbsentStudentFK FOREIGN KEY(StudentId) REFERENCES Students(StudentId)
169 |     ON UPDATE CASCADE
170 |     ON DELETE CASCADE
171 | )
172 | GO
173 |
174 | CREATE TABLE Notes (
175 |     NoteId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
176 |     TeacherId INTEGER NOT NULL,
177 |     StudentId INTEGER NOT NULL,
178 |     Date DATE NOT NULL,
```

```
    Description NVARCHAR(150) NOT NULL,
    CONSTRAINT NotingTeacherId FOREIGN KEY(TeacherId) REFERENCES Teachers(TeacherId),
    CONSTRAINT NotedStudentId FOREIGN KEY(StudentId) REFERENCES Students(StudentId)
)
GO

CREATE TABLE Lessons (
    LessonId INTEGER NOT NULL PRIMARY KEY IDENTITY(1,1),
    CourseId INTEGER NOT NULL,
    ClassId INTEGER NOT NULL,
    Date DATETIME NOT NULL,
    Topic NVARCHAR(150) NOT NULL,
    CONSTRAINT LessonCourseId FOREIGN KEY(CourseId) REFERENCES Courses(CourseId),
    CONSTRAINT LessonClassId FOREIGN KEY(ClassId) REFERENCES Classes(ClassId)
)
GO

ALTER TABLE Accounts
ADD CONSTRAINT is_password_valid CHECK(LEN(Password) > 7)

ALTER TABLE Students
ADD CONSTRAINT is_student_phone_valid CHECK(ISNUMERIC(PhoneNumber) = 1 AND LEN(PhoneNumber) = 9)

ALTER TABLE Teachers
ADD CONSTRAINT is_teacher_phone_valid CHECK(ISNUMERIC(PhoneNumber) = 1 AND LEN(PhoneNumber) = 9)

ALTER TABLE ExamResults
ADD CONSTRAINT is_mark_valid CHECK(Mark BETWEEN 1 AND 6)

GO
```

7.2 Skrypt wstawiający przykładowe wiersze do tabeli

```
INSERT INTO Accounts VALUES
    ('RobHol','lIgtDecE','U','2021-01-19'),
    ('JohNew','hezAUXgq','U','2021-01-19'),
    ('DebTuc','imsxATRd','U','2021-01-19'),
```

```
05 |       ('MelGil','YCTiiVGg','U','2021-01-19'),
06 |       ('LilAug','mQIYZcEo','U','2021-01-19'),
07 |       ('BevFor','eleWrByw','U','2021-01-19'),
08 |       ('MatFri','QdYCjvpv','U','2021-01-19'),
09 |       ('EthRem','mQjFdKpr','U','2021-01-19'),
10 |       ('WilAbr','ZsyWzNPP','U','2021-01-19'),
11 |       ('ChrGet','UIkXtlNM','U','2021-01-19'),
12 |       ('AarAnd','WMzLSrRa','U','2021-01-19'),
13 |       ('GinMcd','HRVbSpos','U','2021-01-19'),
14 |       ('DavAls','zUaytznH','U','2021-01-19'),
15 |       ('DarTer','upSmfsIv','U','2021-01-19'),
16 |       ('DiaTuc','gLFLxKdA','U','2021-01-19'),
17 |       ('EugGam','JeTwRoXl','U','2021-01-19'),
18 |       ('LisRog','jpFZiYWX','U','2021-01-19'),
19 |       ('JoeRam','VifwdSpq','U','2021-01-19'),
20 |       ('RobWas','YnWKCEll','U','2021-01-19'),
21 |       ('JasWhi','pcjYxXbL','U','2021-01-19'),
22 |       ('ClaHor','cmwZRmfH','U','2021-01-19'),
23 |       ('JenZoo','WmSuzDzK','U','2021-01-19'),
24 |       ('ThoSol','ySRjeMuX','U','2021-01-19'),
25 |       ('ClaNol','TkQLLDyE','U','2021-01-19'),
26 |       ('JosOrt','IjHqnBup','U','2021-01-19'),
27 |       ('BarCox','KmLYdxkl','U','2021-01-19'),
28 |       ('DelRiv','sKBBPhKW','U','2021-01-19'),
29 |       ('EmiDun','bEiPbuoF','U','2021-01-19'),
30 |       ('AngAbs','KbbucMEU','U','2021-01-19'),
31 |       ('PatMur','seLfEfRT','N','2021-01-19'),
32 |       ('DanSam','qDwMgJQG','N','2021-01-19'),
33 |       ('KevHow','OLsdcVsG','N','2021-01-19'),
34 |       ('RobHolRodzic','lIgtDecE','P','2021-01-19'),
35 |       ('JohNewRodzic','hezAUXgq','P','2021-01-19'),
36 |       ('DebTucRodzic','imsxATRd','P','2021-01-19'),
37 |       ('MelGilRodzic','YCTiiVGg','P','2021-01-19'),
38 |       ('LilAugRodzic','mQIYZcEo','P','2021-01-19'),
39 |       ('BevForRodzic','eleWrByw','P','2021-01-19'),
40 |       ('MatFriRodzic','QdYCjvpv','P','2021-01-19'),
41 |       ('EthRemRodzic','mQjFdKpr','P','2021-01-19'),
42 |       ('WilAbrRodzic','ZsyWzNPP','P','2021-01-19'),
43 |       ('ChrGetRodzic','UIkXtlNM','P','2021-01-19'),
```

```
    ('AarAndRodzic','WMzLSrRa','P','2021-01-19'),
    ('GinMcdRodzic','HRVbSpos','P','2021-01-19'),
    ('DavAlsRodzic','zUaytznH','P','2021-01-19'),
    ('DarTerRodzic','upSmfsIv','P','2021-01-19'),
    ('DiaTucRodzic','gLFLxKdA','P','2021-01-19'),
    ('EugGamRodzic','JeTwRoXl','P','2021-01-19'),
    ('LisRogRodzic','jpFZiYWX','P','2021-01-19'),
    ('JoeRamRodzic','VifwdSpq','P','2021-01-19'),
    ('RobWasRodzic','YnWKCEll','P','2021-01-19'),
    ('JasWhiRodzic','pcjYxXbL','P','2021-01-19'),
    ('ClaHorRodzic','cmwZRmfH','P','2021-01-19'),
    ('JenZooRodzic','WmSuzDzK','P','2021-01-19'),
    ('ThoSolRodzic','ySRjeMuX','P','2021-01-19'),
    ('ClaNolRodzic','TkQLLDyE','P','2021-01-19'),
    ('JosOrtRodzic','IjHqnBup','P','2021-01-19'),
    ('BarCoxRodzic','KmLYdxkl','P','2021-01-19'),
    ('DelRivRodzic','sKBBPhKW','P','2021-01-19'),
    ('EmiDunRodzic','bEiPbuoF','P','2021-01-19'),
    ('AngAbsRodzic','KbbucMEU','P','2021-01-19')

INSERT INTO Teachers VALUES
    ('William','Bell','2000-01-21','F','755474976',30),
    ('Jacqueline','Rodriguez','2000-07-13','F','247986868',31),
    ('Eileen','Chadwick','2000-07-21','M','485003854',32)

INSERT INTO Classes VALUES
    ('1A',1,'0068'),
    ('1B',2,'0071'),
    ('2A',3,'0083')


INSERT INTO Students VALUES
    ('Christina','Larry','2000-01-16','F','926764505',1,3),
    ('Kelsey','Duran','2000-03-12','M','921766675',2,3),
    ('Johnny','Jordan','2000-02-14','F','906069985',3,3),
    ('Karen','Johnson','2000-06-12','F','931128982',4,1),
    ('Micheal','Tricarico','2000-08-16','M','982933657',5,3),
    ('Frank','Rosmarin','2000-06-18','F','958965596',6,1),
    ('Hubert','Jones','2000-07-16','F','976301309',7,3),
```

```
83 |      ('Michael','Stiger','2000-08-11','M','960535163',8,2),
84 |      ('Amy','Carrell','2000-05-13','M','914195220',9,3),
85 |      ('Richard','Liang','2000-04-12','M','960401300',10,1),
86 |      ('Cynthia','Nelson','2000-01-10','F','962064977',11,2),
87 |      ('Randy','Lalonde','2000-06-27','M','902683026',12,3),
88 |      ('Stephen','Herring','2000-04-15','M','972763261',13,1),
89 |      ('Mildred','Conner','2000-05-17','M','946939858',14,2),
90 |      ('Maryann','Vaden','2000-07-21','F','922227016',15,3),
91 |      ('Glenn','Forman','2000-01-12','M','989203607',16,3),
92 |      ('Paul','Morgan','2000-07-21','M','934712244',17,3),
93 |      ('Mara','Fabrizio','2000-03-13','F','994004441',18,3),
94 |      ('Cedric','Whitted','2000-02-11','F','946358566',19,1),
95 |      ('Martha','Clayton','2000-08-26','F','926994131',20,2),
96 |      ('Winford','Bissell','2000-02-18','M','980312277',21,3),
97 |      ('Frances','Johnson','2000-06-27','M','974116618',22,2),
98 |      ('Wilma','Carter','2000-06-15','F','994907603',23,1),
99 |      ('Kristin','Romero','2000-08-21','M','972808291',24,2),
100 |     ('Orville','Payne','2000-06-23','F','984904403',25,3),
101 |     ('Vincent','Wahid','2000-05-18','F','983674947',26,3),
102 |     ('Renee','Mcnamara','2000-04-19','F','914481600',27,2),
103 |     ('George','Geralds','2000-05-26','F','909289029',28,1),
104 |     ('Manuel','Bennett','2000-08-16','F','979776960',29,1)
105 |
106 | INSERT INTO Notes VALUES
107 |     (1,29,'2015-05-16','Granie na telefonie podczas lekcji'),
108 |     (1,7,'2015-05-26','Palenie papierosow na przerwach'),
109 |     (2,26,'2015-07-25','Palenie papierosow na przerwach'),
110 |     (1,11,'2015-07-18','Brak mundurka'),
111 |     (3,26,'2015-02-11','Brak mundurka'),
112 |     (3,13,'2015-03-21','Palenie papierosow na przerwach'),
113 |     (3,22,'2015-04-11','Ciagle rozmawianie na lekcjach'),
114 |     (2,20,'2015-01-27','Palenie papierosow na przerwach'),
115 |     (2,19,'2015-05-20','Granie na telefonie podczas lekcji'),
116 |     (2,9,'2015-05-10','Brak mundurka')
117 |
118 | INSERT INTO Parents VALUES
119 |     (1,33,'James','Freeman','1980-05-13','F','733336469'),
120 |     (2,34,'Joan','Rolf','1980-02-18','F','535335222'),
121 |     (3,35,'Alfonso','Townsend','1980-07-19','F','649165343'),
```

```
122 |      (4,36,'Brian','Dipietro','1980-05-22','F','603599173'),
123 |      (5,37,'Cathy','Naranjo','1980-06-11','F','92071034'),
124 |      (6,38,'Darla','Harvey','1980-04-22','F','315201980'),
125 |      (7,39,'Oralia','Tovar','1980-02-12','F','971907517'),
126 |      (8,40,'Kristen','Amorin','1980-05-19','M','579341182'),
127 |      (9,41,'Donna','Marshall','1980-06-14','F','231927076'),
128 |      (10,42,'Amanda','Hill','1980-07-10','M','861035749'),
129 |      (11,43,'Pa','Kroes','1980-02-27','M','530080434'),
130 |      (12,44,'Twila','Choute','1980-07-23','F','821061365'),
131 |      (13,45,'Elizabeth','Kio','1980-04-19','M','442987770'),
132 |      (14,46,'Eddie','Green','1980-08-20','F','400235401'),
133 |      (15,47,'Thomas','Bell','1980-04-16','F','588651471'),
134 |      (16,48,'Ruben','Hedrick','1980-06-20','F','178314659'),
135 |      (17,49,'Mandy','Fennimore','1980-08-10','F','875305310'),
136 |      (18,50,'Nina','Rankin','1980-07-12','M','481181042'),
137 |      (19,51,'Chad','Jimison','1980-05-11','M','898240333'),
138 |      (20,52,'Russell','Lowery','1980-08-27','F','725327723'),
139 |      (21,53,'Paul','Altman','1980-02-13','F','930414146'),
140 |      (22,54,'Carin','Roberts','1980-03-15','F','681882320'),
141 |      (23,55,'Ana','Cota','1980-01-15','M','334661896'),
142 |      (24,56,'Soledad','Spencer','1980-04-14','M','203779501'),
143 |      (25,57,'Ronald','Crouch','1980-05-22','M','277528784'),
144 |      (26,58,'Nelson','Green','1980-04-26','M','993133843'),
145 |      (27,59,'James','Barboza','1980-02-16','M','176200241'),
146 |      (28,60,'Wayne','Keister','1980-04-13','F','800753469'),
147 |      (29,61,'Helen','Lundblad','1980-07-11','F','63851641')
148 |
149 |  INSERT INTO Meetings VALUES
150 |      (1,'2021-01-03','0068','Spotkanie po nowym roku'),
151 |      (1,'2021-10-03','0068','Wywiadowka'),
152 |      (2,'2021-01-03','0071','Spotkanie po nowym roku'),
153 |      (2,'2021-10-03','0071','Wywiadowka'),
154 |      (3,'2021-01-03','0083','Spotkanie po nowym roku'),
155 |      (3,'2021-10-03','0083','Wywiadowka')
156 |
157 |  INSERT INTO Courses VALUES
158 |      ('Matematyka', 1),
159 |      ('Fizyka', 1),
160 |      ('Polski',2),
```

```
161 |      ('Historia',2),
162 |      ('Angielski',3),
163 |      ('Niemiecki', 3),
164 |      ('Aktywnosc', 1)
165 |
166 |  INSERT INTO Lessons VALUES
167 |      (1,1,'2020-12-15 09:15','Funkcja liniowa'),
168 |      (1,2,'2020-12-15 10:00','Funkcja liniowa'),
169 |      (1,3,'2020-12-15 10:45','Funkcja liniowa'),
170 |      (1,1,'2020-12-22 09:15','Funkcja kwadratowa'),
171 |      (1,2,'2020-12-22 10:00','Funkcja kwadratowa'),
172 |      (1,3,'2020-12-22 10:45','Funkcja kwadratowa'),
173 |      (1,1,'2021-01-05 09:15','Kombinatoryka'),
174 |      (1,2,'2021-01-05 10:00','Kombinatoryka'),
175 |      (1,3,'2021-01-05 10:45','Kombinatoryka'),
176 |      (2,1,'2020-12-14 09:15','1 zasada dynamiki'),
177 |      (2,2,'2020-12-14 10:00','1 zasada dynamiki'),
178 |      (2,3,'2020-12-14 10:45','1 zasada dynamiki'),
179 |      (2,1,'2020-12-21 09:15','Rownanie ruchu'),
180 |      (2,2,'2020-12-21 10:00','Rownanie ruchu'),
181 |      (2,3,'2020-12-21 10:45','Rownanie ruchu'),
182 |      (2,1,'2021-01-04 09:15','Wahadlo matematyczne i jego zastosowania'),
183 |      (2,2,'2021-01-04 10:00','Wahadlo matematyczne i jego zastosowania'),
184 |      (2,3,'2021-01-04 10:45','Wahadlo matematyczne i jego zastosowania'),
185 |      (3,1,'2020-12-14 8:00','Omawianie Dziadow III'),
186 |      (3,2,'2020-12-14 9:00','Omawianie Dziadow III'),
187 |      (3,3,'2020-12-14 10:00','Omawianie Dziadow III'),
188 |      (3,1,'2020-12-21 8:00','Mickiewicz - zycie i tworczosc'),
189 |      (3,2,'2020-12-21 9:00','Mickiewicz - zycie i tworczosc'),
190 |      (3,3,'2020-12-21 10:00','Mickiewicz - zycie i tworczosc'),
191 |      (3,1,'2021-01-04 8:00','Romantyzm'),
192 |      (3,2,'2021-01-04 9:00','Romantyzm'),
193 |      (3,3,'2021-01-04 10:00','Romantyzm'),
194 |      (4,1,'2020-12-13 8:00','I wojna swiatowa'),
195 |      (4,2,'2020-12-13 9:00','I wojna swiatowa'),
196 |      (4,3,'2020-12-13 10:00','I wojna swiatowa'),
197 |      (4,1,'2020-12-20 8:00','Okres miedzywojenny'),
198 |      (4,2,'2020-12-20 9:00','Okres miedzywojenny'),
199 |      (4,3,'2020-12-20 10:00','Okres miedzywojenny'),
```

```
200 |      (4,1,'2021-01-03 8:00','II wojna swiatowa'),
201 |      (4,2,'2021-01-03 9:00','II wojna swiatowa'),
202 |      (4,3,'2021-01-03 10:00','II wojna swiatowa'),
203 |      (5,1,'2020-12-13 9:00','Grammar - p.27'),
204 |      (5,2,'2020-12-13 10:00','Grammar - p.27'),
205 |      (5,3,'2020-12-13 11:00','Grammar - p.27'),
206 |      (5,1,'2020-12-20 9:00','Speaking and Listening'),
207 |      (5,2,'2020-12-20 10:00','Speaking and Listening'),
208 |      (5,3,'2020-12-20 11:00','Speaking and Listening'),
209 |      (5,1,'2021-01-03 9:00','Presentations'),
210 |      (5,2,'2021-01-03 10:00','Presentations'),
211 |      (5,3,'2021-01-03 11:00','Presentations'),
212 |      (6,1,'2020-12-13 10:00','Flexion von Pronomen'),
213 |      (6,2,'2020-12-13 11:00','Flexion von Pronomen'),
214 |      (6,3,'2020-12-13 12:00','Flexion von Pronomen'),
215 |      (6,1,'2020-12-20 10:00','Sprechen und Zuhoren'),
216 |      (6,2,'2020-12-20 11:00','Sprechen und Zuhoren'),
217 |      (6,3,'2020-12-20 12:00','Sprechen und Zuhoren'),
218 |      (6,1,'2021-01-03 10:00','Hausaufgaben besprechen'),
219 |      (6,2,'2021-01-03 11:00','Hausaufgaben besprechen'),
220 |      (6,3,'2021-01-03 12:00','Hausaufgaben besprechen')
221 |
222 |  INSERT INTO Exams VALUES
223 |      (1,1,'2021-01-12'),
224 |      (1,3,'2021-01-19'),
225 |      (2,1,'2021-01-11'),
226 |      (2,3,'2021-01-18'),
227 |      (3,1,'2021-01-10'),
228 |      (3,3,'2021-01-17')
229 |
230 |  INSERT INTO ExamResults VALUES
231 |      (1,4,1,'2021-01-19'),
232 |      (1,6,1.5,'2021-01-19'),
233 |      (1,10,2,'2021-01-19'),
234 |      (1,13,3,'2021-01-19'),
235 |      (1,19,4,'2021-01-19'),
236 |      (1,23,5,'2021-01-19'),
237 |      (1,28,6,'2021-01-19'),
238 |      (1,29,2,'2021-01-19'),
```

```
239 |       (2,4,1,'2021-01-19'),
240 |       (2,6,1.5,'2021-01-19'),
241 |       (2,10,2,'2021-01-19'),
242 |       (2,13,3,'2021-01-19'),
243 |       (2,19,4,'2021-01-19'),
244 |       (2,23,5,'2021-01-19'),
245 |       (2,28,6,'2021-01-19'),
246 |       (2,29,2,'2021-01-19'),
247 |       (3,8,1,'2021-01-18'),
248 |       (3,11,1.5,'2021-01-18'),
249 |       (3,14,2,'2021-01-18'),
250 |       (3,20,3,'2021-01-18'),
251 |       (3,22,4,'2021-01-18'),
252 |       (3,24,5,'2021-01-18'),
253 |       (3,27,6,'2021-01-18'),
254 |       (4,8,1,'2021-01-18'),
255 |       (4,11,1.5,'2021-01-18'),
256 |       (4,14,2,'2021-01-18'),
257 |       (4,20,3,'2021-01-18'),
258 |       (4,22,4,'2021-01-18'),
259 |       (4,24,5,'2021-01-18'),
260 |       (4,27,6,'2021-01-18'),
261 |       (5,1,1,'2021-01-17'),
262 |       (5,2,1.5,'2021-01-17'),
263 |       (5,3,2,'2021-01-17'),
264 |       (5,5,3,'2021-01-17'),
265 |       (5,7,4,'2021-01-17'),
266 |       (5,9,5,'2021-01-17'),
267 |       (5,12,6,'2021-01-17'),
268 |       (5,15,1,'2021-01-17'),
269 |       (5,16,1.5,'2021-01-17'),
270 |       (5,17,2,'2021-01-17'),
271 |       (5,18,3,'2021-01-17'),
272 |       (5,21,4,'2021-01-17'),
273 |       (5,25,5,'2021-01-17'),
274 |       (5,26,6,'2021-01-17'),
275 |       (6,1,1,'2021-01-17'),
276 |       (6,2,1.5,'2021-01-17'),
277 |       (6,3,2,'2021-01-17'),
```

```
278 |      (6,5,3,'2021-01-17'),
279 |      (6,7,4,'2021-01-17'),
280 |      (6,9,5,'2021-01-17'),
281 |      (6,12,6,'2021-01-17'),
282 |      (6,15,1,'2021-01-17'),
283 |      (6,16,1.5,'2021-01-17'),
284 |      (6,17,2,'2021-01-17'),
285 |      (6,18,3,'2021-01-17'),
286 |      (6,21,4,'2021-01-17'),
287 |      (6,25,5,'2021-01-17'),
288 |      (6,26,6,'2021-01-17')
289 |
290 |  INSERT INTO Absences VALUES
291 |      (14,'2020-03-19','Nieusprawiedliwione'),
292 |      (7,'2020-09-11','Usprawiedliwione'),
293 |      (13,'2021-09-27','Usprawiedliwione'),
294 |      (19,'2020-05-10','Usprawiedliwione'),
295 |      (6,'2020-03-16','Usprawiedliwione'),
296 |      (19,'2020-02-11','Nieusprawiedliwione'),
297 |      (2,'2020-09-22','Nieusprawiedliwione'),
298 |      (22,'2020-04-27','Nieusprawiedliwione'),
299 |      (22,'2021-04-18','Usprawiedliwione')
300 |
301 |  INSERT INTO Authors VALUES
302 |      ('Adam','Mickiewicz'),
303 |      ('Juliusz', 'Slowacki'),
304 |      ('Ernest','Hemingway'),
305 |      ('Fiodor','Dostojewski'),
306 |      ('Henryk', 'Sienkiewicz'),
307 |      ('Thomas', 'Cormen')
308 |
309 |  INSERT INTO Books VALUES
310 |      (1, 'Dziady III',10, 'Dramat'),
311 |      (1, 'Pan Tadeusz',20,'Epopeja'),
312 |      (2, 'Kordian', 20, 'Dramat'),
313 |      (3, 'Stary czlowiek i morze', 5, 'Powiesc'),
314 |      (4, 'Zbrodnia i Kara', 15, 'Literatura rosyjska'),
315 |      (4, 'Bracaia Karamazow', 3, 'Literatura rosyjska'),
316 |      (5, 'Ogniem i mieczem', 10, 'Powiesc historyczna'),
```

```
317 |      (5, 'Quo vaids',11, 'Powiesc'),
318 |      (6, 'Introduction to Algorithms', 1, 'Algorytmy i struktury danych')
319 |
320 |  INSERT INTO Grades VALUES
321 |      (1, 1, 5.0, '2020-12-12', 'aktywnosc'),
322 |      (1, 2, 4.5, '2021-01-02', 'kartkowka'),
323 |      (1, 1, 5.0, '2020-10-13', 'sprawdzian'),
324 |      (1, 1, 5.0, '2020-03-19', 'sprawdzian'),
325 |      (2, 1, 4.5, '2020-06-12', 'aktywnosc'),
326 |      (2, 2, 5.0, '2021-02-10', 'sprawdzian'),
327 |      (2, 3, 4.5, '2021-01-02', 'kartkowka'),
328 |      (2, 3, 6.0, '2020-12-25', 'konkurs'),
329 |      (3, 1, 3.0, '2020-11-11', 'aktywnosc'),
330 |      (3, 2, 1.0, '2020-09-01', 'zadanie domowe')
331 |
332 |
333 |  INSERT INTO Borrows VALUES
334 |      (23, 8, '2020-12-12', NULL),
335 |      (12, 3, '2020-12-14', '2021-01-06'),
336 |      (1, 1, '2021-01-01', NULL),
337 |      (4, 4, '2020-11-01', NULL),
338 |      (20, 9, '2020-10-01', NULL),
339 |      (2, 2, '2021-01-09', '2021-01-10'),
340 |      (23, 5, '2020-12-12', '2020-12-20')
```

8 Typowe zapytania

```
01 | SELECT * FROM Students
02 | SELECT * FROM Grades WHERE Grade = 5.0
03 | SELECT Grade, S.FirstName, S.LastName FROM Grades JOIN Students S ON Grades.StudentId = S.StudentId
04 | SELECT ClassLabel, FirstName, LastName FROM Students S JOIN Classes C ON C.ClassId = S.ClassId ORDER BY ClassLabel ASC
05 | SELECT P.FirstName, P.LastName, S.FirstName [Child First Name], S.LastName [Child Last Name] FROM Parents P JOIN
       Students S ON ChildId = StudentId
06 | SELECT * FROM Books WHERE Amount > 0
07 | SELECT * FROM ExamResults WHERE Mark = 1
08 | SELECT * FROM Teachers WHERE EXISTS(SELECT * FROM Classes WHERE TeacherId = HeadTeacherId)
09 | SELECT FirstName, LastName, Login FROM Teachers T JOIN Accounts A ON A.AccountId = T.AccountId
10 | SELECT * FROM Meetings
```

9 Strategia pielęgnacyjna

Zostały zaimplementowane procedury tworzące kopię zapasową bazy danych na dysku oraz przywracające bazę danych z kopii zapasowej.