Bazy-Danych-Projekt-2023/2024

Systemy Baz Danych 2023/2024 – projekt systemu bazodanowego dla firmy oferującej kursy i szkolenia

Autorzy:

Piotr Śmiałek
Robert Zuziak
Hubert Tułacz

Prowadzący

dr inż. Robert Marcjan

Funkcje użytkowników

Użytkownik anonimowy (gość):

- Przeglądanie dostępnych webinariów.
- Przeglądanie dostępnych kursów.
- Przeglądanie dostępnych studiów.
- Przeglądanie dostępnych informacji o wykładowcach.
- Przeglądanie dostępnych terminów i miejsc spotkań stacjonarnych.
- Rejestracja na darmowe webinaria.
- Przeglądanie nagrań webinariów dostępnych publicznie.
- Możliwość założenia konta

Użytkownik zarejestrowany:

- Logowanie do systemu.
- Przeglądanie dostępnych webinariów.
- Przeglądanie kursów.
- Przeglądanie studiów.
- Przeglądanie informacji o wykładowcach.
- Rejestracja na płatne webinaria.
- Zapisywanie się na kursy (wybór terminów i formy zajęć).
- Zapisywanie się na studia (wybór specjalizacji).
- Przeglądanie własnych zapisów i historii uczestnictwa.
- Przeglądanie informacji o płatnościach.
- Odrabianie nieobecności na zajęciach (jeśli to możliwe).

Wykładowca/Nauczyciel:

Zarządzanie Kursami/Spotkaniami:

- Dodawanie nowych kursów, webinarów i studiów do systemu.
- Zarządzanie terminami i miejscami spotkań stacjonarnych.
- Aktualizacja informacji o programach nauczania (sylabusach).
- Przypisywanie uczestników do kursów i studiów.

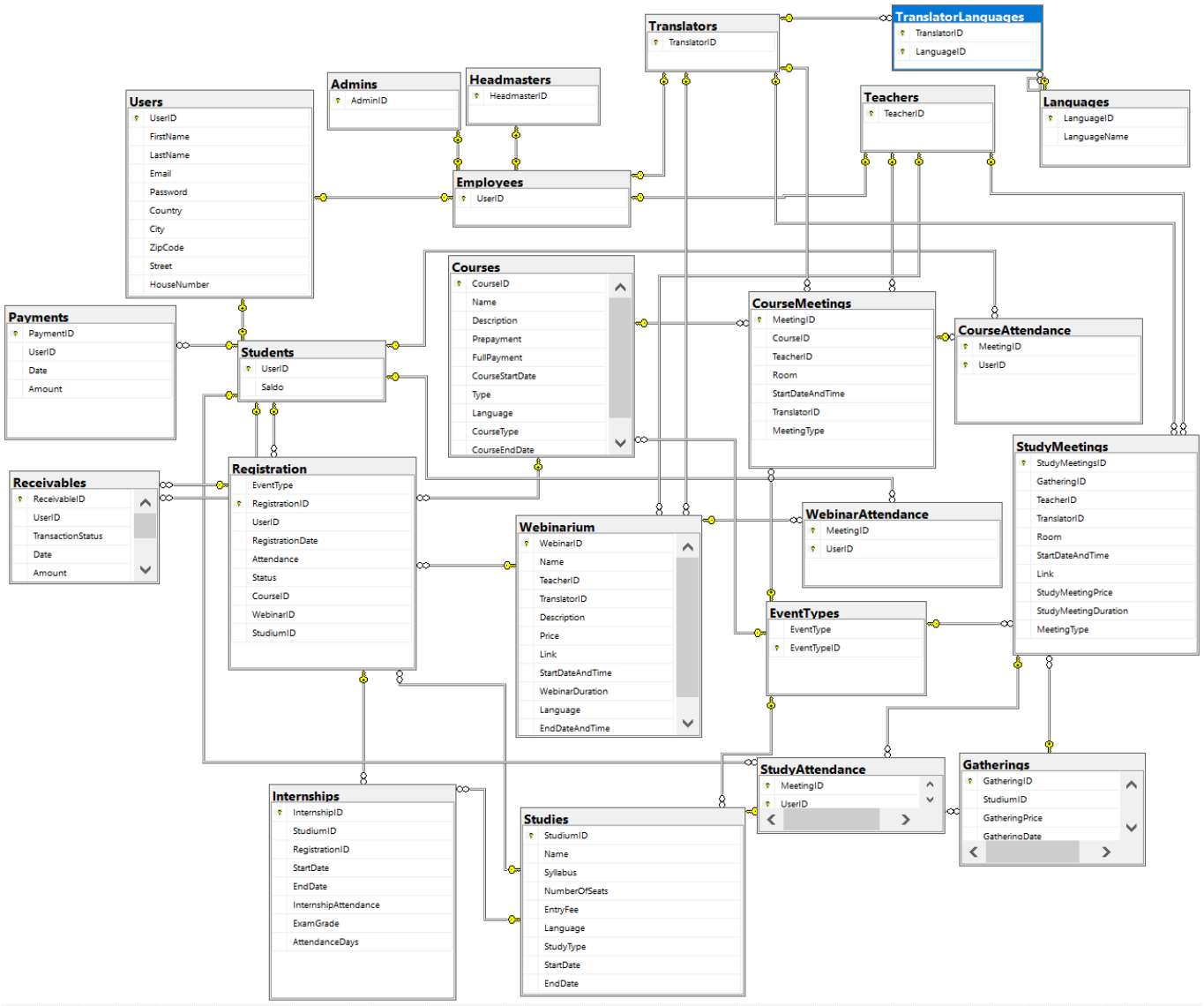
Zarządzanie Ocenami i Frekwencją:

- Wprowadzanie ocen dla uczestników kursów.
- Zaznaczanie obecności na spotkaniach stacjonarnych i online.
- Generowanie raportów dotyczących frekwencji i ocen.

Administrator systemu:

- Dodawanie, edytowanie i usuwanie webinariów.
- Dodawanie, edytowanie i usuwanie kursów.
- Dodawanie, edytowanie i usuwanie studiów.
- Zarządzanie listą wykładowców.
- Zarządzanie terminami i miejscami spotkań stacjonarnych.
- Zarządzanie użytkownikami (edycja danych, blokowanie, usuwanie).

- Przeglądanie raportów finansowych.
- Generowanie listy "dłużników".
- Generowanie raportu dotyczącego liczby zapisanych osób na przyszłe wydarzenia.
- Generowanie raportu dotyczącego frekwencji na zakończonych wydarzeniach.
- Generowanie listy obecności dla każdego szkolenia.
- Generowanie raportu bilokacji.
- Zarządzanie rolami i uprawnieniami użytkowników.
- Dodawanie nowych użytkowników.
- Edycja treści i opisów kursów, studiów, webinariów.



Tabele:

Users - tabela zawiera wszystkich użytkowników w systemie (studentów i pracowników)

- UserID (PK)- Identyfikator użytkownika
- Firstname - imię użytkownika
- LastName - nazwisko użytkownika
- Address - adres użytkownika
- Email - email użytkownika
- Password - hasło użytkownika

```
CREATE TABLE [dbo].[Users](
    [UserID] [int] IDENTITY(1,1) NOT NULL,
    [FirstName] [nchar](20) NOT NULL,
    [LastName] [nchar](20) NOT NULL,
    [Email] [nchar](30) NOT NULL,
    [Password] [nchar](30) NOT NULL,
    [Country] [nchar](20) NOT NULL,
    [City] [nchar](20) NOT NULL,
    [ZipCode] [nchar](20) NOT NULL,
    [Street] [nchar](30) NOT NULL,
    [HouseNumber] [nchar](10) NOT NULL,
 CONSTRAINT [PK_Users] PRIMARY KEY CLUSTERED
(
    [UserID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Users]  WITH CHECK ADD  CONSTRAINT [CK_Users] CHECK  (([Email]
like '%%@%.%'))
GO

ALTER TABLE [dbo].[Users] CHECK CONSTRAINT [CK_Users]
GO
```

Employees -zawiera pracowników z przydzieloną rolą pracownik/administrator

- UserID (Fk) - identyfikator w tabeli Users
- Role - rola, admin/teacher/tłumacz
- Language - jezyk w jakim posługuje sie nauczyciel/tłumacz

```
CREATE TABLE [dbo].[Employees](
    [UserID] [int] NOT NULL,
    [Role] [nchar](10) NOT NULL,
    [Language] [nchar](20) NULL,
 CONSTRAINT [PK_Employees] PRIMARY KEY CLUSTERED
(
    [UserID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Employees]  WITH CHECK ADD  CONSTRAINT [FK_Employees_Users]
FOREIGN KEY([UserID])
REFERENCES [dbo].[Users] ([UserID])
GO
```

```
ALTER TABLE [dbo].[Employees] CHECK CONSTRAINT [FK_Employees_Users]
GO

ALTER TABLE [dbo].[Employees]  WITH CHECK ADD  CONSTRAINT [CK_Employees] CHECK
(([Role] like 'Teacher%' OR [Role] like 'Translator%' OR [Role] like
'Administrator%'))
GO

ALTER TABLE [dbo].[Employees] CHECK CONSTRAINT [CK_Employees]
GO
```

Students - tabela zawiera tylko studentów z tabeli Users

- UserID - Identyfikator w tabeli Users

```
CREATE TABLE [dbo].[Students](
    [UserID] [int] NOT NULL,
 CONSTRAINT [PK_Students] PRIMARY KEY CLUSTERED
(
    [UserID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Students]  WITH CHECK ADD  CONSTRAINT [FK_Students_Users]
FOREIGN KEY([UserID])
REFERENCES [dbo].[Users] ([UserID])
GO

ALTER TABLE [dbo].[Students] CHECK CONSTRAINT [FK_Students_Users]
GO
```

EventTypes- słownik mapujący EventType na EventTypeID stationary - 1 hybrid - 2 online - 3

```
USE [u_pismiale]
GO

/****** Object:  Table [dbo].[EventTypes]    Script Date: 14.01.2024 22:59:40
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[EventTypes](
```

```
    [EventType] [varchar](50) NOT NULL,
    [EventTypeID] [int] NOT NULL,
 CONSTRAINT [PK_EventTypes] PRIMARY KEY CLUSTERED
(
    [EventTypeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
```

Courses - zawiera kursy oraz informacje o nich

- CoursesID (Pk) - klucz główny kursu
- Name - nazwa
- Description - tekstowy opis kursu
- Prepayment - zaliczka przy zapisie
- Full Payment - dopłata całości kwoty (z wyłączeniem zaliczki)
- CourseStartDate - data rozpoczęcia kursu
- NumberOfSeats - liczba miejsc na kurs
- Language - język w jakim jest prowadzony kurs
- CourseType - 1/2/3 (używane w słowniku EventTypes)
- CourseEndDate - data zakończenia kursu

```
USE [u_pismiale]
GO

/****** Object:  Table [dbo].[Courses]    Script Date: 14.01.2024 23:03:54 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Courses](
    [CourseID] [int] NOT NULL,
    [Name] [varchar](50) NOT NULL,
    [Description] [text] NOT NULL,
    [Prepayment] [money] NOT NULL,
    [FullPayment] [money] NOT NULL,
    [CourseStartDate] [datetime] NOT NULL,
    [Type] [varchar](50) NOT NULL,
    [Language] [varchar](50) NULL,
    [CourseType] [int] NULL,
    [CourseEndDate] [datetime] NULL,
 CONSTRAINT [PK_Courses] PRIMARY KEY CLUSTERED
(
    [CourseID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
```

```
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[Courses]  WITH CHECK ADD  CONSTRAINT [FK_Courses_EventTypes]
FOREIGN KEY([CourseType])
REFERENCES [dbo].[EventTypes] ([EventTypeID])
GO

ALTER TABLE [dbo].[Courses] CHECK CONSTRAINT [FK_Courses_EventTypes]
GO

ALTER TABLE [dbo].[Courses]  WITH CHECK ADD  CONSTRAINT [CK_Courses] CHECK
(([Type]='hybrid' OR [Type]='on-line' OR [Type]='stationary'))
GO

ALTER TABLE [dbo].[Courses] CHECK CONSTRAINT [CK_Courses]
GO
```

Registration - tabela zawiera rejestracje dla każdego studenta. Student może mieć wiele rejestracji (może uczęszczać na wiele eventów)

- RegistrationID (PK) - identyfikator rejestracji
- EventType - rodzaj wydarzenia na który zapisał się student
- EventID - identyfikator wydarzenia
- UserID (PK)- Identyfikator użytkownika
- RegistrationDate - data zarejestrowania na dane wydarzenie
- Attendance - procentowa wartość obecności na danym wydarzeniu
- Status - czy student zakończył, jest w trakcie lub nie ukończył wydarzenia

```
CREATE TABLE [dbo].[Registration](
    [EventType] [text] NOT NULL,
    [RegistrationID] [int] NOT NULL,
    [UserID] [int] NOT NULL,
    [RegistrationDate] [date] NOT NULL,
    [Attendance] [float] NOT NULL,
    [Status] [nchar](20) NOT NULL,
    [CourseID] [int] NULL,
    [WebinarID] [int] NULL,
    [StudiumID] [int] NULL,
 CONSTRAINT [PK_Registration] PRIMARY KEY CLUSTERED
(
    [RegistrationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[Registration]  WITH CHECK ADD  CONSTRAINT
[FK_Registration_Courses1] FOREIGN KEY([CourseID])
```

```
REFERENCES [dbo].[Courses] ([CourseID])
GO

ALTER TABLE [dbo].[Registration] CHECK CONSTRAINT [FK_Registration_Courses1]
GO

ALTER TABLE [dbo].[Registration]  WITH CHECK ADD  CONSTRAINT
[FK_Registration_Students] FOREIGN KEY([UserID])
REFERENCES [dbo].[Students] ([UserID])
GO

ALTER TABLE [dbo].[Registration] CHECK CONSTRAINT [FK_Registration_Students]
GO

ALTER TABLE [dbo].[Registration]  WITH CHECK ADD  CONSTRAINT
[FK_Registration_Studies1] FOREIGN KEY([StudiumID])
REFERENCES [dbo].[Studies] ([StudiumID])
GO

ALTER TABLE [dbo].[Registration] CHECK CONSTRAINT [FK_Registration_Studies1]
GO

ALTER TABLE [dbo].[Registration]  WITH CHECK ADD  CONSTRAINT
[FK_Registration_Webinarium1] FOREIGN KEY([WebinarID])
REFERENCES [dbo].[Webinarium] ([WebinarID])
GO

ALTER TABLE [dbo].[Registration] CHECK CONSTRAINT [FK_Registration_Webinarium1]
GO

ALTER TABLE [dbo].[Registration]  WITH CHECK ADD  CONSTRAINT [CK_Registration]
CHECK  (([Status]='completed' OR [Status]='pending'))
GO

ALTER TABLE [dbo].[Registration] CHECK CONSTRAINT [CK_Registration]
GO
```

CourseMeetings - zawiera wszystkie spotkania w ramach jednego kursu

- MeetingID - klucz główny identyfikator każdego spotkania
- CourseID - identyfikator kursu do którego należy spotkanie
- TeacherID - identyfikator nauczyciela prowadzącego spotkanie
- Room - sala w jakiej odbywa się spotkanie
- StartDateAndTime - data i godzina odbycia się zajęć
- Type - czy spotkanie jest stacjonarne/zdalne/zdalnie asynchronicznie
- Link - link do zewnętrznego komunikatora (jeżeli stacjonarnie to NULL)

```
CREATE TABLE [dbo].[CourseMeetings](
    [MeetingID] [int] NOT NULL,
    [CourseID] [int] NOT NULL,
    [TeacherID] [int] NOT NULL,
```

```
    [Room] [nchar](10) NULL,
    [StartDateAndTime] [datetime] NOT NULL,
    [Type] [nchar](10) NOT NULL,
 CONSTRAINT [PK_CourseMeetings] PRIMARY KEY CLUSTERED
(
    [MeetingID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[CourseMeetings]  WITH CHECK ADD  CONSTRAINT
[FK_CourseMeetings_Courses] FOREIGN KEY([CourseID])
REFERENCES [dbo].[Courses] ([CourseID])
GO

ALTER TABLE [dbo].[CourseMeetings] CHECK CONSTRAINT [FK_CourseMeetings_Courses]
GO

ALTER TABLE [dbo].[CourseMeetings]  WITH CHECK ADD  CONSTRAINT
[FK_CourseMeetings_Employees] FOREIGN KEY([TeacherID])
REFERENCES [dbo].[Employees] ([UserID])
GO

ALTER TABLE [dbo].[CourseMeetings] CHECK CONSTRAINT [FK_CourseMeetings_Employees]
GO

ALTER TABLE [dbo].[CourseMeetings]  WITH CHECK ADD  CONSTRAINT [CK_CourseMeetings]
CHECK  (([Type]='hybrid' OR [Type]='on-line' OR [Type]='stationary'))
GO

ALTER TABLE [dbo].[CourseMeetings] CHECK CONSTRAINT [CK_CourseMeetings]
GO
```

CourseAttendance - Zawiera informacje o obecności studenta na zajęciach kursu

- MeetingID - identyfikator kursu
- UserID - identyfikator studenta

```
CREATE TABLE [dbo].[CourseAttendance](
    [MeetingID] [int] NOT NULL,
    [UserID] [int] NOT NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[CourseAttendance]  WITH CHECK ADD  CONSTRAINT
[FK_CourseAttendance_CourseMeetings] FOREIGN KEY([MeetingID])
REFERENCES [dbo].[CourseMeetings] ([MeetingID])
GO
```

```
ALTER TABLE [dbo].[CourseAttendance] CHECK CONSTRAINT
[FK_CourseAttendance_CourseMeetings]
GO

ALTER TABLE [dbo].[CourseAttendance]  WITH CHECK ADD  CONSTRAINT
[FK_CourseAttendance_Students] FOREIGN KEY([UserID])
REFERENCES [dbo].[Students] ([UserID])
GO

ALTER TABLE [dbo].[CourseAttendance] CHECK CONSTRAINT
[FK_CourseAttendance_Students]
GO
```

Internships - zawiera informacje o praktykach studentów

- InternshipID - klucz główny identyfikatora praktyki
- StudiumID - identyfikator studiów do których częścią jest dana praktyka
- RegistrationID - identyfikator rejestracji na dane praktyki
- StartDate - data rozpoczęcia praktyk
- EndDate - data zakończenia praktyk
- AttendanceDays - liczba dni na których student był obecy (od 0 do 14)
- ExamGrade - ocena z egzaminu

```
CREATE TABLE [dbo].[Internships](
    [InternshipID] [int] NOT NULL,
    [StudiumID] [int] NOT NULL,
    [RegistrationID] [int] NOT NULL,
    [StartDate] [date] NULL,
    [EndDate] [date] NULL,
    [InternshipAttendance] [float] NULL,
    [ExamGrade] [float] NULL,
    [AttendanceDays] [int] NULL,
 CONSTRAINT [PK_Internships] PRIMARY KEY CLUSTERED
(
    [InternshipID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Internships]  WITH CHECK ADD  CONSTRAINT
[FK_Internships_Registration] FOREIGN KEY([RegistrationID])
REFERENCES [dbo].[Registration] ([RegistrationID])
GO

ALTER TABLE [dbo].[Internships] CHECK CONSTRAINT [FK_Internships_Registration]
GO

ALTER TABLE [dbo].[Internships]  WITH CHECK ADD  CONSTRAINT
[FK_Internships_Studies] FOREIGN KEY([StudiumID])
```

```
REFERENCES [dbo].[Studies] ([StudiumID])
GO

ALTER TABLE [dbo].[Internships] CHECK CONSTRAINT [FK_Internships_Studies]
GO

ALTER TABLE [dbo].[Internships]  WITH CHECK ADD  CONSTRAINT [CK_Internships] CHECK
(([ExamGrade]>=(2) AND [ExamGrade]<=(5)))
GO

ALTER TABLE [dbo].[Internships] CHECK CONSTRAINT [CK_Internships]
GO

ALTER TABLE [dbo].[Internships]  WITH CHECK ADD  CONSTRAINT [CK_Internships_1]
CHECK  (([AttendanceDays]>=(0) AND [AttendanceDays]<=(14)))
GO

ALTER TABLE [dbo].[Internships] CHECK CONSTRAINT [CK_Internships_1]
GO

ALTER TABLE [dbo].[Internships]  WITH CHECK ADD  CONSTRAINT [CK_Internships_2]
CHECK  (([InternshipAttendance]>=(0) AND [InternshipAttendance]<=(100)))
GO

ALTER TABLE [dbo].[Internships] CHECK CONSTRAINT [CK_Internships_2]
GO
```

Receivables - tabela zawierająca wszystkie należności

- ReceivableID (PK) - identyfikator należności
- UserID (FK) - identyfikator studenta do którego przypisana jest należność
- TransactionStatus - informacja czy należność została uregulowana
- Date - data powstania należności
- Amount - wartość należności
- DueDate - data do której należność ma być uregulowana
- RegistrationID(FK) - identyfikator rejestracji z której pochodzi należność

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Receivables](
    [ReceivableID] [int] NOT NULL,
    [UserID] [int] NULL,
    [TransactionStatus] [nchar](10) NULL,
    [Date] [date] NULL,
    [Amount] [money] NULL,
    [DueDate] [date] NULL,
    [RegistrationID] [int] NULL
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[Receivables] ADD PRIMARY KEY CLUSTERED
(
    [ReceivableID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
GO
ALTER TABLE [dbo].[Receivables]  WITH CHECK ADD FOREIGN KEY([RegistrationID])
REFERENCES [dbo].[Registration] ([RegistrationID])
GO
ALTER TABLE [dbo].[Receivables]  WITH CHECK ADD FOREIGN KEY([UserID])
REFERENCES [dbo].[Students] ([UserID])
GO
```

Payments - tabela zawierająca wszystkie wpłaty

- PaymentID (PK) - identyfikator płatności
- UserID (FK) - identyfikator użytkownika który dokonał płatności
- Date - data dokonania płatności
- Amount - wartość płatności

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Payments](
    [PaymentID] [int] IDENTITY(1,1) NOT NULL,
    [UserID] [int] NULL,
    [Date] [date] NULL,
    [Amount] [money] NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Payments] ADD PRIMARY KEY CLUSTERED
(
    [PaymentID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
ON [PRIMARY]
GO
ALTER TABLE [dbo].[Payments]  WITH CHECK ADD FOREIGN KEY([UserID])
REFERENCES [dbo].[Students] ([UserID])
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TRIGGER [dbo].[UpdateSaldoAfterPayment]
ON [dbo].[Payments]
AFTER INSERT
AS
BEGIN
```

```
    SET NOCOUNT ON;

    -- Aktualizuj saldo dla każdego nowego wpisu w tabeli Payments
    UPDATE s
    SET s.Saldo = s.Saldo + i.Amount
    FROM dbo.Students s
    INNER JOIN inserted i ON s.UserID = i.UserID;
END;
GO
ALTER TABLE [dbo].[Payments] ENABLE TRIGGER [UpdateSaldoAfterPayment]
GO
```

Webinarium - tabela zawierająca wszystkie webinary w systemie

- WebinarID (PK) - identyfikator webinaru
- TeacherID (FK) - identyfikator nauczyciela prowadzącego zajęcia
- TranslatorID (FK) - identyfikator tłumacza
- Name - nazwa webinaru
- Descryption - opis webinaru
- Price - cena webinaru (jeśli free to 0)
- Type (free or not) - darmowy czy płatny
- Link (to external website) - link do webinaru
- StartDateAndTime - dokładna data starru webinaru
- ExpirationDate - data wygaśnięcia webinaru
- WebinariumDuration - czas trwania webinaru

```
CREATE TABLE [dbo].[Webinarium](
    [WebinarID] [int] NOT NULL,
    [Name] [char](30) NOT NULL,
    [TeacherID] [int] NULL,
    [TranslatorID] [int] NULL,
    [Description] [text] NULL,
    [Price] [money] NOT NULL,
    [Link] [text] NULL,
    [StartDateAndTime] [datetime] NULL,
    [WebinarDuration] [time](7) NULL,
    [Language] [nchar](30) NULL,
 CONSTRAINT [PK_Webinarium] PRIMARY KEY CLUSTERED
(
    [WebinarID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[Webinarium]  WITH CHECK ADD  CONSTRAINT
[FK_Webinarium_Employees1] FOREIGN KEY([TranslatorID])
REFERENCES [dbo].[Employees] ([UserID])
```

```
GO

ALTER TABLE [dbo].[Webinarium] CHECK CONSTRAINT [FK_Webinarium_Employees1]
GO

ALTER TABLE [dbo].[Webinarium]  WITH CHECK ADD  CONSTRAINT
[FK_Webinarium_Employees2] FOREIGN KEY([TeacherID])
REFERENCES [dbo].[Employees] ([UserID])
GO

ALTER TABLE [dbo].[Webinarium] CHECK CONSTRAINT [FK_Webinarium_Employees2]
GO
```

WebinarAttendance - tabela zawierająca informacje o użytkownikach obecnych na webinarze

- MeetingID (FK) - identyfikator webinaru
- UserID (FK) - identyfikator uzytkownika

```
CREATE TABLE [dbo].[Webinarium](
    [WebinarID] [int] NOT NULL,
    [Name] [char](30) NOT NULL,
    [TeacherID] [int] NULL,
    [TranslatorID] [int] NULL,
    [Description] [text] NULL,
    [Price] [money] NOT NULL,
    [Link] [text] NULL,
    [StartDateAndTime] [datetime] NULL,
    [WebinarDuration] [time](7) NULL,
    [Language] [nchar](30) NULL,
 CONSTRAINT [PK_Webinarium] PRIMARY KEY CLUSTERED
(
    [WebinarID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[Webinarium]  WITH CHECK ADD  CONSTRAINT
[FK_Webinarium_Employees1] FOREIGN KEY([TranslatorID])
REFERENCES [dbo].[Employees] ([UserID])
GO

ALTER TABLE [dbo].[Webinarium] CHECK CONSTRAINT [FK_Webinarium_Employees1]
GO

ALTER TABLE [dbo].[Webinarium]  WITH CHECK ADD  CONSTRAINT
[FK_Webinarium_Employees2] FOREIGN KEY([TeacherID])
REFERENCES [dbo].[Employees] ([UserID])
GO
```

```
ALTER TABLE [dbo].[Webinarium] CHECK CONSTRAINT [FK_Webinarium_Employees2]
GO
```

StudyMeetings - Zawiera informacje o zajęciach w ramach jednego kierunku

- StudyMeetingsID - Identyfikator spotkania
- GatheringID (FK) - identyfikator zjazdu
- TeacherID (FK) - identyfikator nauczyciela prowadzącego zajęcia
- TranslatorID (FK) - identyfikator tłumacza
- Room - numer sali w której odbywają się zajęcia
- StartDateAndTime - data i godzina o której odbędą się zajęcia
- Type - typ stacjonarne/zdalne/zdalne asynchroniczne
- Link - link do zewnętrznego komunikatora (jeżeli stacjonarne to NULL)
- StudyMeetingPrice - koszt zjazdu
- StudyMeetingDuration - czas trwania spotkania

```
CREATE TABLE [dbo].[StudyMeetings](
    [StudyMeetingsID] [int] NOT NULL,
    [GatheringID] [int] NULL,
    [TeacherID] [int] NULL,
    [TranslatorID] [int] NULL,
    [Room] [nchar](20) NULL,
    [StartDateAndTime] [datetime] NULL,
    [Type] [nchar](20) NULL,
    [Link] [text] NULL,
    [StudyMeetingPrice] [money] NULL,
    [StudyMeetingDuration] [time](7) NULL,
 CONSTRAINT [PK_StudyMeetings] PRIMARY KEY CLUSTERED
(
    [StudyMeetingsID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT
[FK_StudyMeetings_Employees] FOREIGN KEY([TeacherID])
REFERENCES [dbo].[Employees] ([UserID])
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [FK_StudyMeetings_Employees]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT
[FK_StudyMeetings_Employees1] FOREIGN KEY([TranslatorID])
REFERENCES [dbo].[Employees] ([UserID])
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [FK_StudyMeetings_Employees1]
```

```
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT
[FK_StudyMeetings_Gatherings] FOREIGN KEY([GatheringID])
REFERENCES [dbo].[Gatherings] ([GatheringID])
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [FK_StudyMeetings_Gatherings]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT [CK_StudyMeetings]
CHECK  (([Type] like 'stationary%' OR [Type] like 'on-line%' OR [Type] like
'hybrid%'))
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [CK_StudyMeetings]
GO
```

StudyAttendance - Zawiera informacje o obecności studenta na zajęciach na studiach

- MeetingID - identyfikator webinaru
- UserID - identyfikator studenta

```
CREATE TABLE [dbo].[StudyMeetings](
    [StudyMeetingsID] [int] NOT NULL,
    [GatheringID] [int] NULL,
    [TeacherID] [int] NULL,
    [TranslatorID] [int] NULL,
    [Room] [nchar](20) NULL,
    [StartDateAndTime] [datetime] NULL,
    [Type] [nchar](20) NULL,
    [Link] [text] NULL,
    [StudyMeetingPrice] [money] NULL,
    [StudyMeetingDuration] [time](7) NULL,
 CONSTRAINT [PK_StudyMeetings] PRIMARY KEY CLUSTERED
(
    [StudyMeetingsID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT
[FK_StudyMeetings_Employees] FOREIGN KEY([TeacherID])
REFERENCES [dbo].[Employees] ([UserID])
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [FK_StudyMeetings_Employees]
GO
```

```
ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT
[FK_StudyMeetings_Employees1] FOREIGN KEY([TranslatorID])
REFERENCES [dbo].[Employees] ([UserID])
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [FK_StudyMeetings_Employees1]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT
[FK_StudyMeetings_Gatherings] FOREIGN KEY([GatheringID])
REFERENCES [dbo].[Gatherings] ([GatheringID])
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [FK_StudyMeetings_Gatherings]
GO

ALTER TABLE [dbo].[StudyMeetings]  WITH CHECK ADD  CONSTRAINT [CK_StudyMeetings]
CHECK  (([Type] like 'stationary%' OR [Type] like 'on-line%' OR [Type] like
'hybrid%'))
GO

ALTER TABLE [dbo].[StudyMeetings] CHECK CONSTRAINT [CK_StudyMeetings]
GO
```

Studies - zawiera informacje dotyczące kierunków studiów

- StudiumID - klucz główny identyfikatora studiów
- Name - nazwa kierunku
- Syllabus - opis studiów
- NumberOfSeats - liczba miejsc na dany kierunek
- EntryFee - wpisowe na studia
- Language - język w jakim prowadzone są studia
- StudyType - 1/2/3 (używane w słowniku EventTypes)
- StartDate
-

```
CREATE TABLE [dbo].[Studies](
    [StudiumID] [int] NOT NULL,
    [Name] [nchar](50) NOT NULL,
    [Syllabus] [text] NOT NULL,
    [NumberOfSeats] [int] NULL,
    [EntryFee] [money] NULL,
    [Language] [nchar](30) NULL,
 CONSTRAINT [PK_Studies] PRIMARY KEY CLUSTERED
(
    [StudiumID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
```

```
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Gatherings - zjazdy w ramach jednego kierunku studiów

- GatheringID (Pk) - identyfikator zjazdu
- StudiumID (FK) - identyfikator studiów
- GatheringPrice - cena zjazdu
- GatheringDate - data zjazdu

```
CREATE TABLE [dbo].[Gatherings](
    [GatheringID] [int] NOT NULL,
    [StudiumID] [int] NULL,
    [GatheringPrice] [money] NULL,
    [GatheringDate] [date] NULL,
 CONSTRAINT [PK_Gatherings] PRIMARY KEY CLUSTERED
(
    [GatheringID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Gatherings]  WITH CHECK ADD  CONSTRAINT [FK_Gatherings_Studies]
FOREIGN KEY([StudiumID])
REFERENCES [dbo].[Studies] ([StudiumID])
GO

ALTER TABLE [dbo].[Gatherings] CHECK CONSTRAINT [FK_Gatherings_Studies]
GO
```

Widoki:

TeachersInfo Wyświetla informacje o nauczycielach w systemie.

```
CREATE VIEW [dbo].[TeachersInfo]
AS
SELECT dbo.Users.UserID, dbo.Users.FirstName, dbo.Users.LastName
FROM   dbo.Employees INNER JOIN
             dbo.Users ON dbo.Employees.UserID = dbo.Users.UserID
WHERE (dbo.Employees.Role LIKE 'Teacher')
```

StudentsInfo Wyświetla informacje o studentach w systemie.

```
CREATE VIEW [dbo].[StudentsInfo]
AS
SELECT        dbo.Students.UserID AS Expr1, dbo.Users.*
FROM            dbo.Students INNER JOIN
                         dbo.Users ON dbo.Students.UserID = dbo.Users.UserID
```

StudentsNotEnrolled Wyświetla informacje o studentach, którzy nie zarejestrowali się na żaden kurs/webinarium/studia.

```
CREATE VIEW [dbo].[StudentsNotEnrolled]
AS
SELECT dbo.Users.UserID, dbo.Users.FirstName, dbo.Users.LastName
FROM   dbo.Users INNER JOIN
             dbo.Students ON dbo.Users.UserID = dbo.Students.UserID LEFT OUTER
JOIN
             dbo.Registration ON dbo.Students.UserID = dbo.Registration.UserID
WHERE (dbo.Registration.UserID IS NULL)
```

WebinarsInfo Wyświetla informacje o dostępnych webinarach. Pokazuje cenę, sylabus, datę, czas trwania i język w jakim będzie prowadzony webinar.

```
CREATE VIEW [dbo].[WebinarsInfo]
AS
SELECT   Name, Description, Price, StartDateAndTime, WebinarDuration, Language
FROM         dbo.Webinarium
```

CoursesInfo Wyświetla informacje o dostępnych kursach. Pokazuje cenę za cały kurs, zaliczkę, date kursu, rodzaj kursu i język w jakim będzie prowadzony.

```
CREATE VIEW [dbo].[CoursesInfo]
AS
SELECT   Name, Description, Prepayment, FullPayment, CourseStartDate, Type,
Language
FROM         dbo.Courses
```

StudiesInfo Wyświetla informacje o dostępnych kierunkach studiów. Pokazuje nazwe, syllabus, liczbe miejsc, wpisowe, język w jakich są prowadzone.

```
CREATE VIEW [dbo].[StudiesInfo]
AS
SELECT        Name, Syllabus, NumberOfSeats, EntryFee, Language
FROM            dbo.Studies
```

AttendancePerCourseMeeting Wyświetla statystyki frekwencji na dany course meeting. Pokazuje nazwe kursu, id kursu, id meetingu, frekwencja(jako procent), liczba obecnych, liczba zarejestrowanych.

```
SELECT dbo.Courses.Name, CAPM.CourseID, CAPM.MeetingID, CASE WHEN
CNRU.NumberRegistered = 0 THEN NULL ELSE CAST(CAPM.NumberAttended AS FLOAT) /
CNRU.NumberRegistered END AS Attendance, CAPM.NumberAttended,
CNRU.NumberRegistered
FROM   dbo.CourseAttendancePerMeeting() AS CAPM INNER JOIN
            dbo.CourseNumberRegisteredUsers() AS CNRU ON CAPM.CourseID =
CNRU.CourseID INNER JOIN
            dbo.Courses ON CAPM.CourseID = dbo.Courses.CourseID
```

AttendancePerEndedCourse Wyświetla statystyki frekwencji na dany zakończony kurs. Pokazuje id kursu, nazwę kursu, średnią frekwencję.

```
SELECT dbo.Courses.CourseID, dbo.Courses.Name,
AVG(dbo.AttendancePerCourseMeeting.Attendance) AS AverageAttendance
FROM   dbo.AttendancePerCourseMeeting INNER JOIN
            dbo.Courses ON dbo.AttendancePerCourseMeeting.CourseID =
dbo.Courses.CourseID
WHERE (dbo.Courses.CourseEndDate < GETDATE())
GROUP BY dbo.Courses.CourseID, dbo.Courses.Name
```

AttendancePerEndedWebinar Wyświetla statystyki frekwencji na dany zakończony webinar. Pokazuje id webinaru, nazwe, frekwencję.

```
SELECT dbo.Webinarium.WebinarID, dbo.Webinarium.Name, CASE WHEN NumberRegistered =
0 THEN NULL ELSE CAST(WAPM.NumberAttended AS FLOAT) / NumberRegistered END AS
Attendance
FROM   dbo.WebinariumAttendancePerMeeting() AS WAPM INNER JOIN
            dbo.Webinarium ON WAPM.WebinarID = dbo.Webinarium.WebinarID LEFT
OUTER JOIN
                (SELECT WebinarID, COUNT(UserID) AS NumberRegistered
                FROM    dbo.Registration
                GROUP BY WebinarID) AS Subquery ON dbo.Webinarium.WebinarID =
Subquery.WebinarID
WHERE (dbo.Webinarium.EndDateAndTime < GETDATE())
```

BilocationReport Wyświetla raport bilokacji czyli wszystkich użytkowników, którym nakladają sie realizowane szkolenia. Widok zwraca funkcje której sygnatura jest podana dalej.

```
SELECT UserID
FROM dbo.GetBilocationReport() AS getReport
```

Procedury:

AddUser Procedura dodaje użytkownika do systemu.

```
CREATE PROCEDURE [dbo].[AddUser]
    @FirstName nchar(20),
    @LastName nchar(20),
    @Email nchar(30),
    @Password nchar(30),
    @Country nchar(20),
    @City nchar(20),
    @ZipCode nchar(20),
    @Street nchar(30),
    @HouseNumber nchar(10)
AS
BEGIN
    INSERT INTO [dbo].[Users] (
        [FirstName],
        [LastName],
        [Email],
        [Password],
        [Country],
        [City],
        [ZipCode],
        [Street],
        [HouseNumber]
    )
    VALUES (
        @FirstName,
        @LastName,
        @Email,
        @Password,
        @Country,
        @City,
        @ZipCode,
        @Street,
        @HouseNumber
    )
END
```

Triggery:

shoping cart zamiana nazwy zmiana polaczenia na odwrotne (bez registration iD)

1. Wychodzimy z tabeli employees na Translators i Teachers relacja jeden do jednego
2. tworzymy tabele Languages z kolunnami kod, jezyk i laczymy z Translators i Teachers (jeden nauczyciel/translator moze znac wiele jezykow)
3. Tworzym slownik dla (on-line, hybrid i stationary) i łaczmy z tabelami Courses, Wbinarium i Studies Type w powyzszych tabelach zaweira kod ze słownika

4. dodac id do pending