DOKUMENTACJA PROJEKTU

Krystian Madej, Marcin Walendzik, Błażej Kapkowski

Spis treści

Spis treści **3**
Funkcje oraz użytkownicy systemu **7**
Diagram bazy danych **10**
Opisy tabel **11**
Tabela applications 11
Tabela carts 12
Tabela cart_items 14
Tabela cities 15
Tabela countries 16
Tabela courses 17
Tabela courses_modules 18
Zawiera połączenia kursu z jego modułami 18
Klucz główny: product_id, module_id
Klucz obcy: product_id (z courses), module_id (z modules)

product_id - ID kursu
module_id - ID modułu 18
Tabela customers 20
Tabela diplomas 22
Tabela employees 24
Tabela employees_reviews 26
Tabela exam_grades 28
Tabela exams 31
Tabela final_grades 32
Tabela grades 34
Tabela group_members 36
Tabela groups 38
Tabela internships 40
Tabela languages 41
Tabela meeting_participants 42
Tabela meetings 44
record 45
Tabela meetings_online 46
Tabela meetings_stationary 48
Tabela modules 49
Tabela modules_categories 50
Tabela modules_groups 52
Tabela order_items 54
Tabela orders 56
Tabela payments 58

Tabela presence 60
Tabela products 62
Tabela recordings 63
Tabela resources 64
Tabela roles 65
Tabela rooms 66
Tabela studies 67
Tabela studies_groups 68
Tabela studies_subjects 69
Tabela subjects 71
Tabela syllabuses 72
Tabela theme_categories 73
Tabela translators 74
Tabela translators_languages 76
Tabela users 78
Tabela webinars 80
Widoki **82**
Najpopularniejsze kierunki (BK) 82
Najbardziej opłacalne kierunki (BK) 82
Studenci, którzy nie ukończyli kursu/studium (kierunek) (BK) 83
Absolwenci studium/kurs (BK) 84
Najmniej zdawalny kierunek (BK) 85
Najpopularniejsze języki (całość) (KM) 85
Najpopularniejsze języki (rocznie) (KM) 86
Wolne sale (KM) 87
Oceny prowadzących (całość) (KM) 89
Oceny prowadzących (rocznie) (KM) 89
Osoby zapisane na przyszłe wydarzenia (lista) (KM) 90
Osoby zapisane na przyszłe wydarzenia (liczba) (KM) 91
Raport dłużników (MW) 92
Zaakceptowane wnioski (MW) 93
Raport bilokacji (MW) 94
Zajęcia w danym tygodniu (MW) 95
Zajęcia w danym miesiącu (MW) 95
Obecność każdego ucznia (MW) 96
Procedury **97**
Procedura create_application (KM) 97
Procedura accept_application (KM) 97
Procedura create_cart (KM) 98
Procedura add_item_to_cart (KM) 98
Procedura remove_item_to_cart (KM) 99
Procedura remove_cart (KM) 100
Procedura add_city (KM) 100

Procedura add_country (KM) 101
Procedura create_course (KM) 101
Procedura create_customer (KM) 102
Procedura create_diploma (KM) 103
Procedura create_employee (KM) 104
Procedura add_employee_review (KM) 104
Procedura add_grade (KM) 105
Procedura change_grade (KM) 106
Procedura create_exam (KM) 107
Procedura set_exam_grades (KM) 108
Procedura set_final_grades (KM) 109
Procedura add_to_group (KM) 110
Procedura add_to_group_meetings (KM) 111
Procedura remove_from_group (KM) 112
Procedura change_group_cooridinator (KM) 112
Procedura create_group (KM) 113
Procedura find_group_to_add_to(KM) 114
Procedura create_studies (KM) 114
Procedura create_studies_groups (KM) 115
Procedura pass_internship (MW) 116
Procedura add_language (MW) 117
Procedura create_meeting (MW) 118
Procedura change_meeting_date (MW) 119
Procedura create_meeting_stationary (MW) 120
Procedura create_online_meeting (MW) 121
Procedura create_module (MW) 121
Procedura add_module_category (MW) 122
Procedura add_module_to_group (MW) 123
Procedura create_order (MW) 123
Procedura create_order_from_cart (MW) 124
Procedura create_payment (MW) 125
Procedura pay_order (MW) 126
Procedura create_presence (MW) 127
Procedura set_presence (MW) 127
Procedura create_product (MW) 128
Procedura add_recording (BK) 128
Procedura add_resource (BK) 130
Procedura create_role (BK) 130
Procedura add_room (BK) 131
Procedura add_studies (BK) 131
Procedura add_studies_group (BK) 132
Procedura add_studies_subject (BK) 133
Procedura add_syllabus (BK) 133

Procedura create_subject (BK) 134
Procedura create_theme_category (BK) 135
Procedura add_translator (BK) 135
Procedura add_translator_language (BK) 136
Procedura create_user (BK) 136
Procedura activate/deactivate_user (BK) 137
Procedura create_webinar (BK) 138
Procedura add_participant 139
Procedura add_product_to_order 139
Procedura add_subject_to_studies 140
Procedura room_is_available 140
Triggery **142**
Trigger add_to_group_after_payment (KM) 142
Trigger add_diplom (BK) 144
Uprawnienia **145**
Student 145
Ćwiczeniowiec/ wykładowca 146
Koordynator 146
Księgowa 147
Dyrektor 147
Admin 147

Funkcje oraz użytkownicy systemu

Użytkownicy systemu:

- Anonimowy (niezalogowany)
- Student
- Koordynator roku
- Ćwiczeniowiec
- Koordynator kierunku/przedmiotu/kursu/webinary
- Prowadzący kurs/webinar
- Tłumacz
- Księgowa
- Dyrektor
- Administrator

Koszyk (Dostępny dla wszystkich):

- Tworzenie nowego koszyka
- Usuwanie koszyka
- Dodawanie rzeczy do koszyka
- Usuwanie rzeczy z koszyka
- Pobieranie wartości koszyka

Podstawowe interakcje (Dostępne dla wszystkich):

- Lista prowadzonych kursów/webinarów/studiów
- Dane kontaktowe do szkoły
- Logowanie
- Zakładanie konta
- Zadawanie pytań?

Student:

- Zapisywanie się na kurs itd
- Wypisywanie się z kursu itd
- Opłacenie kursu itd
- Historia opłat
- Wniosek o odroczenie płatności
- Wniosek o rabat dla stałego klienta
- Wniosek o brak wymogu wpisowego, ze względu na np. średnią
- Sprawdzenie opłaconych kursów
- Sprawdzenie deadlinów opłat
- Przypomnienia mailowe o opłaceniu kursów
- Sprawdzenie własnej frekwencji
- Zapisanie się na odrabianie zajęć
- Wypisanie się z odrabiania zajęć
- Sprawdzenie kolizji
- Generowanie planu zajęć
- Lista emaili do prowadzących, współuczestników grup

- Sprawdzanie polecanych firm
- Wgrywanie certyfikatów z praktyk
- Dostęp do materiałów przesłanych przez prowadzącego
- Ocenianie prowadzących

Koordynator roku:
- Ustalanie harmonogramu zajęć dla studentów

Ćwiczeniowcy:
- Sprawdzanie frekwencji
- Akceptowanie uczestników ‘z zewnątrz’
- Dodawanie/zmiana ocen
- Lista emaili do swoich grup
- Lista studentów z własnych grup
- Udostępnianie materiałów
- Generowanie planu zajęć
- Dodawanie tłumacza
- Lista osób zapisanych na przyszłe wydarzenie

Koordynator przedmiotu:
- Ustalanie Sylabusu
- Lista studentów na przedmiocie
- Raport o ocenach
- Raport o frekwencji
- Raport o ocenach ćwiczeniowców
- Raport o zdawalności
- Ustalanie czy jest zdalnie, hybrydowo czy stacjonarnie
- Ustalanie terminów egzaminów
- Wpisywanie ocen za egzaminy
- Zmiany w harmonogramie z przypadków losowych
- Zaliczanie praktyk

Prowadzący kurs/webinar:
- Wrzucanie nagrań
- Wrzucanie materiałów
- Dodawanie tłumacza
- Raport o oglądalności
- Raport o obecności
- Sprawdzanie obecności (stacjonarnie)
- Ustalanie harmonogramu
- Lista osób zapisanych na przyszłe wydarzenie
- Lista osób w grupie

Tłumacz:
- Generowanie harmonogramu
- Dostęp do spotkań na platformie

Księgowa:

- Raporty finansowe
- Raport o przychodach z kursu, studium itd
- Drukowanie i wysyłanie dyplomów
- Lista studentów i ich adresy

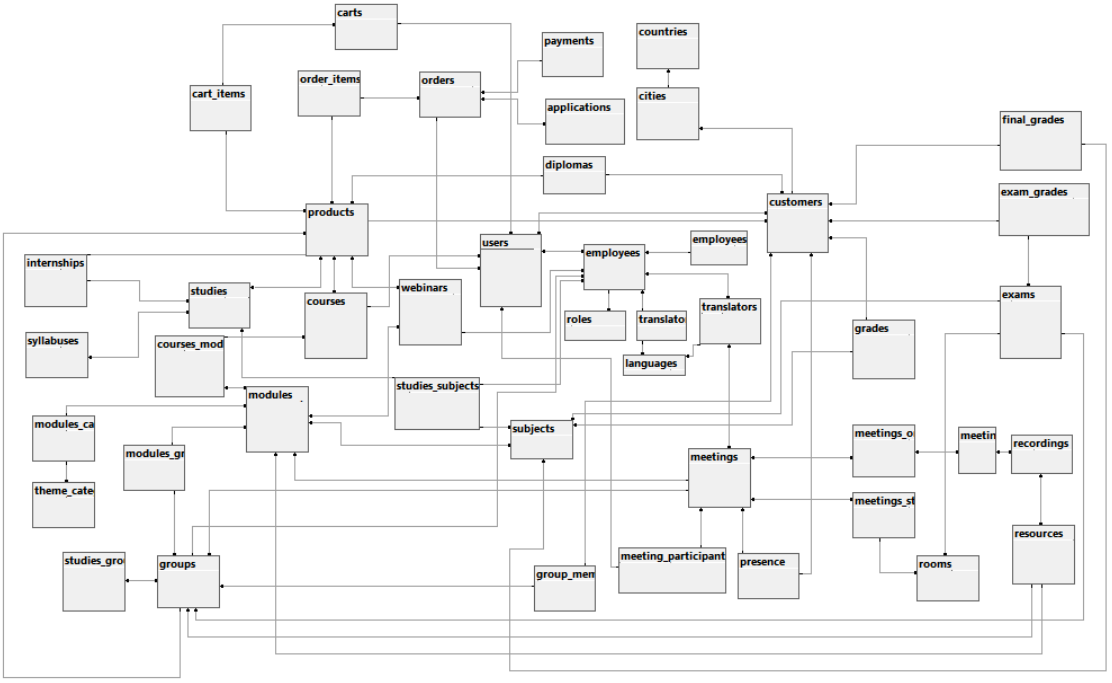
Dyrektor:

- Akceptowanie wniosków dot. płatności
- Lista stałych klientów
- Lista dłużników
- Lista studentów wg średniej
- Tworzenie nowych kierunków, kursów itd i ustalanie ich cen
- Zmiana cen
- Tworzenie nowych przedmiotów
- Zakańczanie działania kursu itd
- Tworzenie dyplomów
- Zatrudnianie pracowników
- Zwalnianie pracowników

Administrator:

- Dodawanie użytkowników pracowników
- Zmiana typu kont, uprawnień
- Blokowanie kont
- Tworzenie kopii zapasowych
- Kasowanie webinarów

Diagram bazy danych



Krystian Madej, Marcin Walendzik, Błażej Kapkowski

Opisy tabel

Tabela applications

Zawiera informacje o złożonych wnioskach dotyczących zamówień

Klucz główny: application_id
Klucz obcy: order_id (z orders)
Defaulty: date - getdate()
Unique: order_id

application_id - ID wniosku, not null
order_id - ID zamówienia, not null
date - data złożenia wniosku, null
accepted - czy zaakceptowany (gdy null, to nierozpatrzony), null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[applications]    Script Date: 03.01.2024
13:53:05 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[applications](
	[application_id] [int] NOT NULL,
	[order_id] [int] NOT NULL,
	[date] [datetime] NULL,
	[accepted] [bit] NULL,
 CONSTRAINT [PK_applications] PRIMARY KEY CLUSTERED
(
	[application_id] 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],
 CONSTRAINT [UQ_applications_order_id] UNIQUE NONCLUSTERED
(
	[order_id] 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].[applications] ADD  CONSTRAINT [DF_applications_date]
DEFAULT (getdate()) FOR [date]
GO
ALTER TABLE [dbo].[applications]  WITH CHECK ADD  CONSTRAINT
[FK_applications_orders] FOREIGN KEY([order_id])
REFERENCES [dbo].[orders] ([order_id])
GO
ALTER TABLE [dbo].[applications] CHECK CONSTRAINT
[FK_applications_orders]
GO
```

Tabela carts

Zawiera informacje o koszyku

Klucz główny: cart_id
Klucz obcy: user_id (z users)

cart_id - ID koszyka, not null
user_id - ID użytkownika, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[carts]    Script Date: 03.01.2024 13:20:00
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[carts](
	[cart_id] [int] NOT NULL,
	[user_id] [int] NOT NULL,
 CONSTRAINT [PK_carts] PRIMARY KEY CLUSTERED
(
	[cart_id] 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].[carts]  WITH CHECK ADD  CONSTRAINT [FK_carts_users]
FOREIGN KEY([user_id])
REFERENCES [dbo].[users] ([user_id])
GO
ALTER TABLE [dbo].[carts] CHECK CONSTRAINT [FK_carts_users]
GO
```

Tabela cart_items

Zawiera informacje o przedmiotach w koszyku

Klucz główny: cart_id, product_id
Klucz obcy: product_id (z products)

cart_id - ID koszyka, not null
product_id - ID produktu, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[cart_items]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[cart_items](
	[cart_id] [int] NOT NULL,
	[product_id] [int] NOT NULL,
 CONSTRAINT [PK_cart_items] PRIMARY KEY CLUSTERED
(
	[cart_id] ASC,
	[product_id] 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].[cart_items]  WITH CHECK ADD  CONSTRAINT
[FK_cart_items_carts] FOREIGN KEY([cart_id])
REFERENCES [dbo].[carts] ([cart_id])
GO
ALTER TABLE [dbo].[cart_items] CHECK CONSTRAINT [FK_cart_items_carts]
GO
ALTER TABLE [dbo].[cart_items]  WITH CHECK ADD  CONSTRAINT
[FK_cart_items_products] FOREIGN KEY([product_id])
REFERENCES [dbo].[products] ([product_id])
GO
ALTER TABLE [dbo].[cart_items] CHECK CONSTRAINT [FK_cart_items_products]
GO
```

Tabela cities

Zawiera informacje o miastach

Klucz główny: city_id
Klucz obcy: country_id (z countries)

city_id - ID miasta, not null
name - nazwa miasta, not null
country_id - ID kraju, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[cities]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[cities](
	[city_id] [int] NOT NULL,
	[name] [varchar](50) NOT NULL,
	[country_id] [int] NOT NULL,
 CONSTRAINT [PK_cities] PRIMARY KEY CLUSTERED
(
	[city_id] 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].[cities]  WITH CHECK ADD  CONSTRAINT
[FK_cities_countries] FOREIGN KEY([country_id])
REFERENCES [dbo].[countries] ([country_id])
GO
ALTER TABLE [dbo].[cities] CHECK CONSTRAINT [FK_cities_countries]
GO
```

Tabela countries

Zawiera informacje o krajach

Klucz główny: country_id

country_id - ID kraju, not null
name - nazwa kraju, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[countries]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[countries](
	[country_id] [int] NOT NULL,
	[name] [nvarchar](50) NOT NULL,
 CONSTRAINT [PK_countries] PRIMARY KEY CLUSTERED
(
	[country_id] 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
```

Tabela courses

Zawiera informacje o dostępnych kursach

Klucz główny: product_id
Klucz obcy: product_id (z products), module_id (z modules), lecturer_id (z users)
Checki: end_date musi być po start_date
Unique: module_id

product_id - ID produktu, not null
lecturer_id - ID wykładowcy, not null
start_date - Data rozpoczęcia kursu, not null
end_data - Data zakończenia kursu, not null

```
USE [u_kmadej]
GO

/****** Object:  Table [dbo].[courses]    Script Date: 17.01.2024
13:54:32 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[courses](
	[product_id] [int] NOT NULL,
	[lecturer_id] [int] NOT NULL,
	[start_date] [date] NOT NULL,
	[end_date] [date] NOT NULL,
 CONSTRAINT [PK_courses_1] PRIMARY KEY CLUSTERED
(
	[product_id] 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].[courses]  WITH CHECK ADD  CONSTRAINT
[FK_courses_products] FOREIGN KEY([product_id])
REFERENCES [dbo].[products] ([product_id])
GO

ALTER TABLE [dbo].[courses] CHECK CONSTRAINT [FK_courses_products]
```

```
GO

ALTER TABLE [dbo].[courses]  WITH CHECK ADD  CONSTRAINT
[FK_courses_users] FOREIGN KEY([lecturer_id])
REFERENCES [dbo].[users] ([user_id])
GO

ALTER TABLE [dbo].[courses] CHECK CONSTRAINT [FK_courses_users]
GO

ALTER TABLE [dbo].[courses]  WITH CHECK ADD  CONSTRAINT
[CK_courses_dates] CHECK  (([start_date]<[end_date]))
GO

ALTER TABLE [dbo].[courses] CHECK CONSTRAINT [CK_courses_dates]
GO
```

Tabela courses_modules

Zawiera połączenia kursu z jego modułami

Klucz główny: product_id, module_id
Klucz obcy: product_id (z courses), module_id (z modules)

product_id - ID kursu
module_id - ID modułu

```
USE [u_kmadej]
GO

/****** Object:  Table [dbo].[courses_modules]    Script Date:
17.01.2024 13:57:26 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[courses_modules](
      [product_id] [int] NOT NULL,
      [module_id] [int] NOT NULL,
 CONSTRAINT [PK_courses_modules] PRIMARY KEY CLUSTERED
```

```
(
      [product_id] ASC,
      [module_id] 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],
 CONSTRAINT [UQ_courses_modules] UNIQUE NONCLUSTERED
(
      [module_id] 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].[courses_modules]  WITH CHECK ADD  CONSTRAINT
[FK_courses_modules_courses] FOREIGN KEY([product_id])
REFERENCES [dbo].[courses] ([product_id])
GO

ALTER TABLE [dbo].[courses_modules] CHECK CONSTRAINT
[FK_courses_modules_courses]
GO

ALTER TABLE [dbo].[courses_modules]  WITH CHECK ADD  CONSTRAINT
[FK_courses_modules_modules] FOREIGN KEY([module_id])
REFERENCES [dbo].[modules] ([module_id])
GO

ALTER TABLE [dbo].[courses_modules] CHECK CONSTRAINT
[FK_courses_modules_modules]
GO
```

Tabela customers

Zawiera dodatkowe informacje o klientach

Klucz główny: user_id
Klucz obcy: user_id (z users), city_id (z cities)

user_id - ID użytkownika, not null
city_id - ID miasta zamieszkania klienta, not null
street - ulica zamieszkania klienta, not null
zip_code - kod pocztowy, not null

```
USE [u_kmadej]
GO

/****** Object:  Table [dbo].[customers]    Script Date: 16.01.2024
17:36:31 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[customers](
	[user_id] [int] NOT NULL,
	[city_id] [int] NOT NULL,
	[street] [varchar](50) NOT NULL,
	[zip_code] [varchar](20) NOT NULL,
 CONSTRAINT [PK_customers] PRIMARY KEY CLUSTERED
(
	[user_id] 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].[customers]  WITH CHECK ADD  CONSTRAINT
[FK_customers_cities] FOREIGN KEY([city_id])
REFERENCES [dbo].[cities] ([city_id])
GO

ALTER TABLE [dbo].[customers] CHECK CONSTRAINT [FK_customers_cities]
GO
```

```
ALTER TABLE [dbo].[customers]  WITH CHECK ADD  CONSTRAINT
[FK_customers_users] FOREIGN KEY([user_id])
REFERENCES [dbo].[users] ([user_id])
GO

ALTER TABLE [dbo].[customers] CHECK CONSTRAINT [FK_customers_users]
GO
```

Tabela diplomas

Zawiera informacje potrzebne do tworzenia dyplomów

Klucz główny: diploma_id
Klucz obcy: user_id (z customers), product_id (z products)
Unique: product_id, user_id

diploma_id - ID dyplomu, not null
user_id - ID użytkownika, not null
product_id - ID produktu, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[diplomas]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[diplomas](
	[diploma_id] [int] NOT NULL,
	[user_id] [int] NOT NULL,
	[product_id] [int] NOT NULL,
 CONSTRAINT [PK_diplomas] PRIMARY KEY CLUSTERED
(
	[diploma_id] 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],
 CONSTRAINT [IX_diplomas] UNIQUE NONCLUSTERED
(
	[product_id] ASC,
	[user_id] 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].[diplomas]  WITH CHECK ADD  CONSTRAINT
[FK_diplomas_customers] FOREIGN KEY([user_id])
REFERENCES [dbo].[customers] ([user_id])
GO
ALTER TABLE [dbo].[diplomas] CHECK CONSTRAINT [FK_diplomas_customers]
```

```
GO
ALTER TABLE [dbo].[diplomas]  WITH CHECK ADD  CONSTRAINT
[FK_diplomas_products] FOREIGN KEY([product_id])
REFERENCES [dbo].[products] ([product_id])
GO
ALTER TABLE [dbo].[diplomas] CHECK CONSTRAINT [FK_diplomas_products]
GO
```

Tabela employees

Zawiera dodatkowe informacje o pracownikach

Klucz główny: user_id
Klucz obcy: user_id (z users), role_id (z roles)

user_id - ID użytkownika, not null
role_id - ID roli pracownika, not null

//koordynatora rozpoznajemy po tym czym ma przypisaną jakąś grupę

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[employees]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[employees](
	[user_id] [int] NOT NULL,
	[role_id] [int] NOT NULL,
 CONSTRAINT [PK_employees] PRIMARY KEY CLUSTERED
(
	[user_id] 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],
 CONSTRAINT [IX_employees] UNIQUE NONCLUSTERED
(
	[role_id] 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_roles] FOREIGN KEY([role_id])
REFERENCES [dbo].[roles] ([role_id])
GO
ALTER TABLE [dbo].[employees] CHECK CONSTRAINT [FK_employees_roles]
GO
```

```
ALTER TABLE [dbo].[employees]  WITH CHECK ADD  CONSTRAINT
[FK_employees_users] FOREIGN KEY([user_id])
REFERENCES [dbo].[users] ([user_id])
GO
ALTER TABLE [dbo].[employees] CHECK CONSTRAINT [FK_employees_users]
GO
```

Tabela employees_reviews

Zawiera opinie o pracownikach.

Klucz główny: reviewed_id, date
Klucz obcy: reviewed_id (z employees)
Checki: rating - wartość pomiędzy 1 a 10

reviewed_id - ID opiniowanego, not null
date - Data wystawienia opinii, not null
review - Komentarz do opinii, not null
rating - Ocena pracownika, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[employees_reviews]    Script Date:
03.01.2024 13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[employees_reviews](
	[reviewed_id] [int] NOT NULL,
	[date] [datetime] NOT NULL,
	[review] [ntext] NOT NULL,
	[rating] [int] NOT NULL,
 CONSTRAINT [PK_employees_reviews] PRIMARY KEY CLUSTERED
(
	[reviewed_id] ASC,
	[date] 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].[employees_reviews]  WITH CHECK ADD  CONSTRAINT
[FK_employees_reviews_employees] FOREIGN KEY([reviewed_id])
REFERENCES [dbo].[employees] ([user_id])
GO
ALTER TABLE [dbo].[employees_reviews] CHECK CONSTRAINT
[FK_employees_reviews_employees]
GO
ALTER TABLE [dbo].[employees_reviews]  WITH CHECK ADD  CONSTRAINT
[CK_employees_reviews_rating] CHECK  (([rating]>=(1) AND
[rating]<=(10)))
```

```
GO
ALTER TABLE [dbo].[employees_reviews] CHECK CONSTRAINT
[CK_employees_reviews_rating]
GO
```

Tabela exam_grades

Zawiera informacje o ocenach z egzaminu. Przechowuje ocenę końcową studenta z danego przedmiotu

Klucz główny: exam_id, user_id
Klucz obcy: exam_id (z exams), user_id (z customers)
Checki: grade - jedna z wartości (2, 3, 3.5, 4, 4.5, 5)

exam_id - ID egzaminu, not null
user_id - ID użytkownika, not null
grade - ocena, null oznacza niepodejście do egzaminu, null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[exam_grades]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[exam_grades](
	[exam_id] [int] NOT NULL,
	[user_id] [int] NOT NULL,
	[grade] [real] NULL,
 CONSTRAINT [PK_exam_grades] PRIMARY KEY CLUSTERED
(
	[exam_id] ASC,
	[user_id] 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].[exam_grades]  WITH CHECK ADD  CONSTRAINT
[FK_exam_grades_customers] FOREIGN KEY([user_id])
REFERENCES [dbo].[customers] ([user_id])
GO
ALTER TABLE [dbo].[exam_grades] CHECK CONSTRAINT
[FK_exam_grades_customers]
GO
ALTER TABLE [dbo].[exam_grades]  WITH CHECK ADD  CONSTRAINT
[FK_exam_grades_exams] FOREIGN KEY([exam_id])
REFERENCES [dbo].[exams] ([exam_id])
```

```
GO
ALTER TABLE [dbo].[exam_grades] CHECK CONSTRAINT [FK_exam_grades_exams]
GO
ALTER TABLE [dbo].[exam_grades]  WITH CHECK ADD  CONSTRAINT
[CK_exam_grades] CHECK  (([grade]=(5.0) OR [grade]=(4.5) OR
[grade]=(4.0) OR [grade]=(3.5) OR [grade]=(3.0) OR [grade]=(2.0)))
GO
ALTER TABLE [dbo].[exam_grades] CHECK CONSTRAINT [CK_exam_grades]
GO
```

Tabela exams

Zawiera informacje o egzaminach

Klucz główny: exam_id
Klucz obcy: group_id (z groups), module_id (z modules), room_id (z rooms)
Checki: term - jedna z wartości (1, 2, 3)
Unique: group_id, module_id, term

exam_id - ID egzaminu, not null
group_id - ID grupy, not null
module_id - ID moudułu, not null
date - data egzaminu, not null
room_id - ID pokoju, not null
term - termin egzaminu, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[exams]    Script Date: 03.01.2024 13:20:00
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[exams](
	[exam_id] [int] NOT NULL,
	[group_id] [int] NOT NULL,
	[module_id] [int] NOT NULL,
	[start_time] [datetime] NOT NULL,
	[term] [int] NOT NULL,
	[room_id] [int] NOT NULL,
	[end_time] [datetime] NOT NULL,
 CONSTRAINT [PK_exams] PRIMARY KEY CLUSTERED
(
	[exam_id] 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],
 CONSTRAINT [IX_exams] UNIQUE NONCLUSTERED
(
	[group_id] ASC,
	[module_id] ASC,
	[term] 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].[exams]  WITH CHECK ADD  CONSTRAINT [FK_exams_groups]
FOREIGN KEY([group_id])
REFERENCES [dbo].[groups] ([group_id])
GO
ALTER TABLE [dbo].[exams] CHECK CONSTRAINT [FK_exams_groups]
GO
ALTER TABLE [dbo].[exams]  WITH CHECK ADD  CONSTRAINT [FK_exams_modules]
FOREIGN KEY([module_id])
REFERENCES [dbo].[subjects] ([module_id])
GO
ALTER TABLE [dbo].[exams] CHECK CONSTRAINT [FK_exams_modules]
GO
ALTER TABLE [dbo].[exams]  WITH CHECK ADD  CONSTRAINT [FK_exams_rooms]
FOREIGN KEY([room_id])
REFERENCES [dbo].[rooms] ([room_id])
GO
ALTER TABLE [dbo].[exams] CHECK CONSTRAINT [FK_exams_rooms]
GO
ALTER TABLE [dbo].[exams]  WITH CHECK ADD  CONSTRAINT [CK_exams_dates]
CHECK  (([start_time]<[end_time]))
GO
ALTER TABLE [dbo].[exams] CHECK CONSTRAINT [CK_exams_dates]
GO
ALTER TABLE [dbo].[exams]  WITH CHECK ADD  CONSTRAINT [CK_exams_term]
CHECK  (([term]=(1) OR [term]=(2) OR [term]=(3)))
GO
ALTER TABLE [dbo].[exams] CHECK CONSTRAINT [CK_exams_term]
GO
```

Tabela final_grades

Zawiera informacje o ocenach końcowych

Klucz główny: date, user_id, module_id
Klucz obcy: user_id (z customers), module_id (z modules)
Checki: value - jedna z wartości (2, 3, 3.5, 4, 4.5, 5)

date - data wpisania oceny, not null
user_id - ID użytkownika, not null
module_id - ID modułu, not null
value - wartość oceny, null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[final_grades]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[final_grades](
	[user_id] [int] NOT NULL,
	[module_id] [int] NOT NULL,
	[date] [datetime] NOT NULL,
	[value] [real] NULL,
 CONSTRAINT [PK_final_grades] PRIMARY KEY CLUSTERED
(
	[user_id] ASC,
	[module_id] ASC,
	[date] 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].[final_grades]  WITH CHECK ADD  CONSTRAINT
[FK_final_grades_customers] FOREIGN KEY([user_id])
REFERENCES [dbo].[customers] ([user_id])
GO
ALTER TABLE [dbo].[final_grades] CHECK CONSTRAINT
[FK_final_grades_customers]
GO
ALTER TABLE [dbo].[final_grades]  WITH CHECK ADD  CONSTRAINT
```

```
[FK_final_grades_modules] FOREIGN KEY([module_id])
REFERENCES [dbo].[subjects] ([module_id])
GO
ALTER TABLE [dbo].[final_grades] CHECK CONSTRAINT
[FK_final_grades_modules]
GO
ALTER TABLE [dbo].[final_grades]  WITH CHECK ADD  CONSTRAINT
[CK_final_grades] CHECK  (([value]=(5.0) OR [value]=(4.5) OR
[value]=(4.0) OR [value]=(3.5) OR [value]=(3.0) OR [value]=(2.0)))
GO
ALTER TABLE [dbo].[final_grades] CHECK CONSTRAINT [CK_final_grades]
GO
```

Tabela grades

Zawiera informacje o ocenach. Przechowuje ocenę cząstkową jaką dostał student w danym dniu z danego przedmiotu

Klucz główny: date, user_id, module_id
Klucz obcy: user_id (z customers), module_id (z modules)
Checki: value jedna z wartości (2, 3, 3.5, 4, 4.5, 5)

date - data wpisania oceny, not null
user_id - ID użytkownika, not null
module_id - ID modułu, not null
value - wartość oceny, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[grades]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[grades](
	[date] [datetime] NOT NULL,
	[user_id] [int] NOT NULL,
	[module_id] [int] NOT NULL,
	[value] [real] NOT NULL,
 CONSTRAINT [PK_grades] PRIMARY KEY CLUSTERED
(
	[date] ASC,
	[user_id] ASC,
	[module_id] 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].[grades]  WITH CHECK ADD  CONSTRAINT
[FK_grades_customers] FOREIGN KEY([user_id])
REFERENCES [dbo].[customers] ([user_id])
GO
ALTER TABLE [dbo].[grades] CHECK CONSTRAINT [FK_grades_customers]
GO
```

```
ALTER TABLE [dbo].[grades]  WITH CHECK ADD  CONSTRAINT
[FK_grades_subjects] FOREIGN KEY([module_id])
REFERENCES [dbo].[subjects] ([module_id])
GO
ALTER TABLE [dbo].[grades] CHECK CONSTRAINT [FK_grades_subjects]
GO
ALTER TABLE [dbo].[grades]  WITH CHECK ADD  CONSTRAINT [CK_grades] CHECK
(([value]=(5.0) OR [value]=(4.5) OR [value]=(4.0) OR [value]=(3.5) OR
[value]=(3.0) OR [value]=(2.0)))
GO
ALTER TABLE [dbo].[grades] CHECK CONSTRAINT [CK_grades]
GO
```

Tabela group_members

Zawiera informacje o członkach grupy

Klucz główny: group_id, user_id
Klucz obcy: group_id (z groups), user_id (z users)

group_id - ID grupy, not null
user_id - ID użytkownika, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[group_members]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
```

```
GO
CREATE TABLE [dbo].[group_members](
      [group_id] [int] NOT NULL,
      [user_id] [int] NOT NULL,
 CONSTRAINT [PK_group_members] PRIMARY KEY CLUSTERED
(
      [group_id] ASC,
      [user_id] 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].[group_members]  WITH CHECK ADD  CONSTRAINT
[FK_group_members_groups] FOREIGN KEY([group_id])
REFERENCES [dbo].[groups] ([group_id])
GO
ALTER TABLE [dbo].[group_members] CHECK CONSTRAINT
[FK_group_members_groups]
GO
ALTER TABLE [dbo].[group_members]  WITH CHECK ADD  CONSTRAINT
[FK_group_members_users] FOREIGN KEY([user_id])
REFERENCES [dbo].[customers] ([user_id])
GO
ALTER TABLE [dbo].[group_members] CHECK CONSTRAINT
[FK_group_members_users]
GO
```

Tabela groups

Zawiera informacje o grupach

Klucz główny: group_id
Klucz obcy: coordinator_id (z users)

group_id - ID grupy, not null
coordinator_id - ID koordynatora, not null
product_id - ID produktu, do którego jest przypisana grupa, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[groups]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[groups](
	[group_id] [int] NOT NULL,
	[coordinator_id] [int] NOT NULL,
	[product_id] [int] NOT NULL,
 CONSTRAINT [PK_groups] PRIMARY KEY CLUSTERED
(
	[group_id] 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].[groups]  WITH CHECK ADD  CONSTRAINT
[FK_groups_products] FOREIGN KEY([product_id])
REFERENCES [dbo].[products] ([product_id])
GO
ALTER TABLE [dbo].[groups] CHECK CONSTRAINT [FK_groups_products]
GO
ALTER TABLE [dbo].[groups]  WITH CHECK ADD  CONSTRAINT [FK_groups_users]
FOREIGN KEY([coordinator_id])
REFERENCES [dbo].[employees] ([user_id])
GO
ALTER TABLE [dbo].[groups] CHECK CONSTRAINT [FK_groups_users]
GO
```

Tabela internships

Zawiera informacje o praktykach

Klucz główny: user_id, studies_id
Klucz obcy: studies_id (z studies), user_id (z customers)

user_id - ID użytkownika, not null
studies_id - ID studiów, not null
passed - określa czy zaliczona praktyki, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[internships]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[internships](
      [user_id] [int] NOT NULL,
      [studies_id] [int] NOT NULL,
      [passed] [bit] NOT NULL,
 CONSTRAINT [PK_internships] PRIMARY KEY CLUSTERED
(
      [user_id] ASC,
      [studies_id] 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_customers] FOREIGN KEY([user_id])
REFERENCES [dbo].[customers] ([user_id])
GO
ALTER TABLE [dbo].[internships] CHECK CONSTRAINT
[FK_internships_customers]
GO
ALTER TABLE [dbo].[internships]  WITH CHECK ADD  CONSTRAINT
[FK_internships_studies] FOREIGN KEY([studies_id])
REFERENCES [dbo].[studies] ([product_id])
GO
ALTER TABLE [dbo].[internships] CHECK CONSTRAINT
[FK_internships_studies]
```

```
GO
```

Tabela languages

Klucz główny: language_id
Unique: language_id

language_id - ID języka, not null
language_name - nazwa języka, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[languages]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[languages](
	[language_id] [int] NOT NULL,
	[language_name] [varchar](50) NOT NULL,
 CONSTRAINT [PK_languages] PRIMARY KEY CLUSTERED
(
	[language_id] 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],
 CONSTRAINT [UQ_languages_name] UNIQUE NONCLUSTERED
(
	[language_id] 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
```

Tabela meeting_participants

Zawiera informacje o członkach każdego spotkania

Klucz główny: meeting_id
Klucz obcy: meeting_id (z meetings), user_id (z users)

meeting_id - ID spotkania, not null
user_id - ID użytkownika, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[meeting_participants]    Script Date:
03.01.2024 13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[meeting_participants](
	[meeting_id] [int] NOT NULL,
	[user_id] [int] NOT NULL,
 CONSTRAINT [PK_meeting_participants] PRIMARY KEY CLUSTERED
(
	[meeting_id] ASC,
	[user_id] 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].[meeting_participants]  WITH CHECK ADD  CONSTRAINT
[FK_meeting_participants_meetings] FOREIGN KEY([meeting_id])
REFERENCES [dbo].[meetings] ([meeting_id])
GO
ALTER TABLE [dbo].[meeting_participants] CHECK CONSTRAINT
[FK_meeting_participants_meetings]
GO
ALTER TABLE [dbo].[meeting_participants]  WITH CHECK ADD  CONSTRAINT
[FK_meeting_participants_users] FOREIGN KEY([user_id])
REFERENCES [dbo].[users] ([user_id])
GO
ALTER TABLE [dbo].[meeting_participants] CHECK CONSTRAINT
[FK_meeting_participants_users]
GO
```

Tabela meetings

Zawiera informacje o spotkaniach

Klucz główny: meeting_id
Klucz obcy: module_id (z modules), meeting_id (z meetings_online), meeting_id (z translators)
Checki: początek spotkania jest wcześniej niż zakończenie

meeting_id - ID spotkania, not null
start_date - data rozpoczęcia spotkania, not null
end_date - data zakończenia spotkania, not null
module_id - ID modułu, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[meetings]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[meetings](
	[meeting_id] [int] NOT NULL,
	[start_date] [datetime] NOT NULL,
	[end_date] [datetime] NOT NULL,
	[module_id] [int] NOT NULL,
 CONSTRAINT [PK_meetings] PRIMARY KEY CLUSTERED
(
	[meeting_id] 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].[meetings]  WITH CHECK ADD  CONSTRAINT
[FK_meetings_modules] FOREIGN KEY([module_id])
REFERENCES [dbo].[modules] ([module_id])
GO
ALTER TABLE [dbo].[meetings] CHECK CONSTRAINT [FK_meetings_modules]
GO
ALTER TABLE [dbo].[meetings]  WITH CHECK ADD  CONSTRAINT
[CK_meetings_dates] CHECK  (([start_date]<[end_date]))
```

```
GO
ALTER TABLE [dbo].[meetings] CHECK CONSTRAINT [CK_meetings_dates]
GO
```

record

Tabela meetings_online

Zawiera informacje o spotkaniach online

Klucz główny: meeting_id
Klucz obcy: meeting_id (z meetings), recording_id (z recordings)
Checki: początek spotkania jest wcześniej niż zakończenie
Unique: recording_id

meeting_id - ID spotkania, not null
link - link do spotkania online, not null
recording_id - ID nagrania, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[meetings_online]    Script Date:
03.01.2024 13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[meetings_online](
	[meeting_id] [int] NOT NULL,
	[link] [varchar](200) NOT NULL,
	[recording_id] [int] NOT NULL,
 CONSTRAINT [PK_meetings_online] PRIMARY KEY CLUSTERED
(
	[meeting_id] 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],
 CONSTRAINT [IX_meetings_online] UNIQUE NONCLUSTERED
(
	[recording_id] 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].[meetings_online]  WITH CHECK ADD  CONSTRAINT
[FK_meetings_online_meetings] FOREIGN KEY([meeting_id])
REFERENCES [dbo].[meetings] ([meeting_id])
```

```
GO
ALTER TABLE [dbo].[meetings_online] CHECK CONSTRAINT
[FK_meetings_online_meetings]
GO
ALTER TABLE [dbo].[meetings_online]  WITH CHECK ADD  CONSTRAINT
[FK_meetings_online_meetings_online] FOREIGN KEY([recording_id])
REFERENCES [dbo].[recordings] ([resource_id])
GO
ALTER TABLE [dbo].[meetings_online] CHECK CONSTRAINT
[FK_meetings_online_meetings_online]
GO
```

Tabela meetings_stationary

Zawiera informacje o spotkaniach stacjonarnie

Klucz główny: meeting_id
Klucz obcy: meeting_id (z meetings), room_id (z rooms)

meeting_id - ID spotkania, not null
room_id - ID pokoju. not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[meetings_stationary]    Script Date:
03.01.2024 13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[meetings_stationary](
	[meeting_id] [int] NOT NULL,
	[room_id] [int] NOT NULL,
 CONSTRAINT [PK_meetings_stationary] PRIMARY KEY CLUSTERED
(
	[meeting_id] 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].[meetings_stationary]  WITH CHECK ADD  CONSTRAINT
[FK_meetings_stationary_meetings] FOREIGN KEY([meeting_id])
REFERENCES [dbo].[meetings] ([meeting_id])
GO
ALTER TABLE [dbo].[meetings_stationary] CHECK CONSTRAINT
[FK_meetings_stationary_meetings]
GO
ALTER TABLE [dbo].[meetings_stationary]  WITH CHECK ADD  CONSTRAINT
[FK_meetings_stationary_rooms] FOREIGN KEY([room_id])
REFERENCES [dbo].[rooms] ([room_id])
GO
ALTER TABLE [dbo].[meetings_stationary] CHECK CONSTRAINT
[FK_meetings_stationary_rooms]
GO
```

Tabela modules

Zawiera informacje o typach modułów - części lub całości produktu dostępnego do kupna

Klucz główny: module_id

module_id - ID modułu, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[modules]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[modules](
	[module_id] [int] NOT NULL,
 CONSTRAINT [PK_modules] PRIMARY KEY CLUSTERED
(
	[module_id] 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
```

Tabela modules_categories

Zawiera informacje o kategoriach modelów. Jeden przedmiot jest modułem

Klucz główny: module_id, category_id
Klucz obcy: module_id (z modules), category_id (z theme_categories)

module_id - ID modułu, not null
category_id - ID kategorii, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[modules_categories]    Script Date:
03.01.2024 13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[modules_categories](
	[module_id] [int] NOT NULL,
	[category_id] [int] NOT NULL,
 CONSTRAINT [PK_modules_categories] PRIMARY KEY CLUSTERED
(
	[module_id] ASC,
	[category_id] 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].[modules_categories]  WITH CHECK ADD  CONSTRAINT
[FK_modules_categories_modules] FOREIGN KEY([module_id])
REFERENCES [dbo].[modules] ([module_id])
GO
ALTER TABLE [dbo].[modules_categories] CHECK CONSTRAINT
[FK_modules_categories_modules]
GO
ALTER TABLE [dbo].[modules_categories]  WITH CHECK ADD  CONSTRAINT
[FK_modules_categories_theme_categories] FOREIGN KEY([category_id])
REFERENCES [dbo].[theme_categories] ([category_id])
GO
ALTER TABLE [dbo].[modules_categories] CHECK CONSTRAINT
[FK_modules_categories_theme_categories]
GO
```

Tabela modules_groups

Zawiera informacje o grupach danych modułów

Klucz główny: module_id, group_id
Klucz obcy: module_id (z modules), group_id (z groups)

module_id - ID modułu, not null
group_id - ID grupy, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[modules_groups]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[modules_groups](
	[module_id] [int] NOT NULL,
	[group_id] [int] NOT NULL,
 CONSTRAINT [PK_modules_groups] PRIMARY KEY CLUSTERED
(
	[module_id] ASC,
	[group_id] 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].[modules_groups]  WITH CHECK ADD  CONSTRAINT
[FK_modules_groups_groups] FOREIGN KEY([group_id])
REFERENCES [dbo].[groups] ([group_id])
GO
ALTER TABLE [dbo].[modules_groups] CHECK CONSTRAINT
[FK_modules_groups_groups]
GO
ALTER TABLE [dbo].[modules_groups]  WITH CHECK ADD  CONSTRAINT
[FK_modules_groups_modules] FOREIGN KEY([module_id])
REFERENCES [dbo].[modules] ([module_id])
GO
ALTER TABLE [dbo].[modules_groups] CHECK CONSTRAINT
[FK_modules_groups_modules]
GO
```

Tabela order_items

Zawiera informacje o zamówionych przedmiotach

Klucz główny: order_id, product_id
Klucz obcy: order_id (z orders), product_id (z products)

order_id - ID zamówienia, not null
product_id - ID produktu, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[order_items]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[order_items](
      [order_id] [int] NOT NULL,
      [product_id] [int] NOT NULL,
 CONSTRAINT [PK_order_items] PRIMARY KEY CLUSTERED
(
      [order_id] ASC,
      [product_id] 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].[order_items]  WITH CHECK ADD  CONSTRAINT
[FK_order_items_orders] FOREIGN KEY([order_id])
REFERENCES [dbo].[orders] ([order_id])
GO
ALTER TABLE [dbo].[order_items] CHECK CONSTRAINT [FK_order_items_orders]
GO
ALTER TABLE [dbo].[order_items]  WITH CHECK ADD  CONSTRAINT
[FK_order_items_products] FOREIGN KEY([product_id])
REFERENCES [dbo].[products] ([product_id])
GO
ALTER TABLE [dbo].[order_items] CHECK CONSTRAINT
[FK_order_items_products]
GO
```

Tabela orders

Zawiera informacje o zamówieniach

Klucz główny: order_id
Klucz obcy: user_id (z users)
Default: order_date - data dzisiejsza

order_id - ID zamówienia, not null
user_id - ID użytkownika, not null
order_date - data zamówienia, null oznacza dzisiaj

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[orders]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[orders](
	[order_id] [int] NOT NULL,
	[user_id] [int] NOT NULL,
	[order_date] [datetime] NULL,
 CONSTRAINT [PK_orders] PRIMARY KEY CLUSTERED
(
	[order_id] 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].[orders] ADD  CONSTRAINT [DF_orders_order_date]
DEFAULT (getdate()) FOR [order_date]
GO
ALTER TABLE [dbo].[orders]  WITH CHECK ADD  CONSTRAINT [FK_orders_users]
FOREIGN KEY([user_id])
REFERENCES [dbo].[users] ([user_id])
GO
ALTER TABLE [dbo].[orders] CHECK CONSTRAINT [FK_orders_users]
GO
```

Tabela payments

Zawiera informacje o płatnościach

Klucz główny: payment_id
Klucz obcy: order_id (z orders)
Checki: payment_date i value są jednocześnie nullami albo jednocześnie nie są

payment_id - ID płatności, not null
order_id - ID zamówienia, not null
due_date - ostateczny termin zapłaty, not null
payment_date - data zapłacenia, null oznacza brak płatności
value - wartość płatności, null oznacza brak płatności

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[payments]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[payments](
	[payment_id] [int] NOT NULL,
	[order_id] [int] NOT NULL,
	[due_date] [datetime] NOT NULL,
	[payment_date] [datetime] NULL,
	[value] [money] NULL,
 CONSTRAINT [PK_payments] PRIMARY KEY CLUSTERED
(
	[payment_id] 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].[payments]  WITH CHECK ADD  CONSTRAINT
[FK_payments_orders] FOREIGN KEY([order_id])
REFERENCES [dbo].[orders] ([order_id])
GO
ALTER TABLE [dbo].[payments] CHECK CONSTRAINT [FK_payments_orders]
GO
ALTER TABLE [dbo].[payments]  WITH CHECK ADD  CONSTRAINT
[CK_payments_xnor] CHECK  (([payment_date] IS NULL AND [value] IS NULL
```

```
OR [payment_date] IS NOT NULL AND [value] IS NOT NULL))
GO
ALTER TABLE [dbo].[payments] CHECK CONSTRAINT [CK_payments_xnor]
GO
```

Tabela presence

Zawiera informacje o obecności

Klucz główny: meeting_id, user_id
Klucz obcy: meeting_id (z meetings), user_id (z customers)

meeting_id - ID spotkania, not null
user_id - ID użytkownika, not null
is_present - określa czy użytkownik był obecny, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[presence]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[presence](
	[meeting_id] [int] NOT NULL,
	[user_id] [int] NOT NULL,
	[is_present] [bit] NOT NULL,
 CONSTRAINT [PK_presence] PRIMARY KEY CLUSTERED
(
	[meeting_id] ASC,
	[user_id] 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].[presence]  WITH CHECK ADD  CONSTRAINT
[FK_presence_meetings] FOREIGN KEY([meeting_id])
REFERENCES [dbo].[meetings] ([meeting_id])
GO
ALTER TABLE [dbo].[presence] CHECK CONSTRAINT [FK_presence_meetings]
GO
ALTER TABLE [dbo].[presence]  WITH CHECK ADD  CONSTRAINT
[FK_presence_users] FOREIGN KEY([user_id])
REFERENCES [dbo].[customers] ([user_id])
GO
ALTER TABLE [dbo].[presence] CHECK CONSTRAINT [FK_presence_users]
GO
```

Tabela products

Zawiera informacje o dostępnych produktach

Klucz główny: product_id
Klucz obcy: product_id (z courses), product_id (z webinars), product_id (z studies)
Default: cena jest równa 0

product_id - ID produktu, not null
price - cena produktu, null oznacza darmowy
product_name - nazwa produktu, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[products]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[products](
	[product_id] [int] NOT NULL,
	[product_name] [varchar](50) NOT NULL,
	[price] [money] NULL,
 CONSTRAINT [PK_products] PRIMARY KEY CLUSTERED
(
	[product_id] 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].[products] ADD  CONSTRAINT [DF_products_price]
DEFAULT ((0)) FOR [price]
GO
```

Tabela recordings

Zawiera informacje o nagraniach spotkań online

Klucz główny: resource_id
Klucz obcy: resource_id (z meetings_online), resource_id (z resources)

resource_id - ID nagrania, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[recordings]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[recordings](
	[resource_id] [int] NOT NULL,
 CONSTRAINT [PK_recordings] PRIMARY KEY CLUSTERED
(
	[resource_id] 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].[recordings]  WITH CHECK ADD  CONSTRAINT
[FK_recordings_resources] FOREIGN KEY([resource_id])
REFERENCES [dbo].[resources] ([resource_id])
GO
ALTER TABLE [dbo].[recordings] CHECK CONSTRAINT
[FK_recordings_resources]
GO
```

Tabela resources

Zawiera informacje o dostępności materiałów

Klucz główny: resource_id
Klucz obcy: resource_id (z recordings), module_id (z modules)
Default: add_date jest ustawiona na dzisiaj

resource_id - ID nagrania, not null
link - link do nagrania, not null
module_id - ID modułu, not null
add_date - data dodania nagrania, null to data dzisiejsza

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[resources]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[resources](
	[resource_id] [int] NOT NULL,
	[link] [varchar](200) NOT NULL,
	[module_id] [int] NOT NULL,
	[add_date] [datetime] NULL,
 CONSTRAINT [PK_resources] PRIMARY KEY CLUSTERED
(
	[resource_id] 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].[resources] ADD  CONSTRAINT [DF_resources_add_date]
DEFAULT (getdate()) FOR [add_date]
GO
ALTER TABLE [dbo].[resources]  WITH CHECK ADD  CONSTRAINT
[FK_resources_modules] FOREIGN KEY([module_id])
REFERENCES [dbo].[modules] ([module_id])
GO
ALTER TABLE [dbo].[resources] CHECK CONSTRAINT [FK_resources_modules]
GO
```

Tabela roles

Zawiera dodatkowe informacje o rolach pracowników

Klucz główny: role_id
Klucz obcy: role_id (z employees)

role_id - ID roli pracownika, not null
name - nazwa roli pracownika, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[roles]    Script Date: 03.01.2024 13:20:00
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[roles](
	[role_id] [int] NOT NULL,
	[name] [varchar](50) NOT NULL,
 CONSTRAINT [PK_roles] PRIMARY KEY CLUSTERED
(
	[role_id] 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
```

Tabela rooms

Zawiera informacje o pokojach

Klucz główny: room_id

number - numer pokoju, not null
room_id - ID pokoju, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[rooms]    Script Date: 03.01.2024 13:20:00
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[rooms](
	[room_id] [int] NOT NULL,
	[number] [varchar](10) NOT NULL,
 CONSTRAINT [PK_rooms] PRIMARY KEY CLUSTERED
(
	[room_id] 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
```

Tabela studies

Zawiera informacje o cenach produktów dla osób z zewnątrz

Klucz główny: product_id
Klucz obcy: product_id (z products)

product_id - ID produktu, not null
price_for_outsiders - cena dla osób z zewnątrz, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[studies]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[studies](
	[product_id] [int] NOT NULL,
	[price_for_outsiders] [money] NOT NULL,
 CONSTRAINT [PK_studies] PRIMARY KEY CLUSTERED
(
	[product_id] 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].[studies]  WITH CHECK ADD  CONSTRAINT
[FK_studies_products] FOREIGN KEY([product_id])
REFERENCES [dbo].[products] ([product_id])
GO
ALTER TABLE [dbo].[studies] CHECK CONSTRAINT [FK_studies_products]
GO
```

Tabela studies_groups

Zawiera informacje o grupach danych przedmiotów

Klucz główny: group_id
Klucz obcy: group_id (z groups)

group_id - ID grupy, not null
start_date - data rozpoczęcia, not null
end_date - data zakończenia, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[studies_groups]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[studies_groups](
	[group_id] [int] NOT NULL,
	[start_date] [date] NOT NULL,
	[end_date] [date] NOT NULL,
 CONSTRAINT [PK_studies_groups] PRIMARY KEY CLUSTERED
(
	[group_id] 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].[studies_groups]  WITH CHECK ADD  CONSTRAINT
[FK_studies_groups_groups] FOREIGN KEY([group_id])
REFERENCES [dbo].[groups] ([group_id])
GO
ALTER TABLE [dbo].[studies_groups] CHECK CONSTRAINT
[FK_studies_groups_groups]
GO
```

Tabela studies_subjects

Zawiera informacje o rodzajach przedmiotów

Klucz główny: product_id, module_id
Klucz obcy: product_id (z studies), module_id (z modules), master_id (z employees)

product_id - ID produktu, not null
module_id - ID modułu, not null
master_id - ID prowadzącego,not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[studies_subjects]    Script Date:
03.01.2024 13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[studies_subjects](
	[product_id] [int] NOT NULL,
	[module_id] [int] NOT NULL,
	[master_id] [int] NOT NULL,
 CONSTRAINT [PK_studies_subjects] PRIMARY KEY CLUSTERED
(
	[product_id] ASC,
	[module_id] 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].[studies_subjects]  WITH CHECK ADD  CONSTRAINT
[FK_studies_subjects_employees] FOREIGN KEY([master_id])
REFERENCES [dbo].[employees] ([user_id])
GO
ALTER TABLE [dbo].[studies_subjects] CHECK CONSTRAINT
[FK_studies_subjects_employees]
GO
ALTER TABLE [dbo].[studies_subjects]  WITH CHECK ADD  CONSTRAINT
[FK_studies_subjects_studies] FOREIGN KEY([product_id])
REFERENCES [dbo].[studies] ([product_id])
GO
ALTER TABLE [dbo].[studies_subjects] CHECK CONSTRAINT
```

```
[FK_studies_subjects_studies]
GO
ALTER TABLE [dbo].[studies_subjects]  WITH CHECK ADD  CONSTRAINT
[FK_studies_subjects_subjects] FOREIGN KEY([module_id])
REFERENCES [dbo].[subjects] ([module_id])
GO
ALTER TABLE [dbo].[studies_subjects] CHECK CONSTRAINT
[FK_studies_subjects_subjects]
GO
```

Tabela subjects

Zawiera informacje o rodzajach przedmiotów

Klucz główny: module_id
Klucz obcy: module_id (z modules)

module_id - ID modułu, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[subjects]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[subjects](
	[module_id] [int] NOT NULL,
 CONSTRAINT [PK_subjects] PRIMARY KEY CLUSTERED
(
	[module_id] 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].[subjects]  WITH CHECK ADD  CONSTRAINT
[FK_subjects_modules] FOREIGN KEY([module_id])
REFERENCES [dbo].[modules] ([module_id])
GO
ALTER TABLE [dbo].[subjects] CHECK CONSTRAINT [FK_subjects_modules]
GO
```

Tabela syllabuses

Zawiera informacje o sylabusie

Klucz główny: studies_id
Klucz obcy: studies_id (z studies)

studies_id - ID przedmiotu, not null
link - link do sylabusu, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[syllabuses]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[syllabuses](
	[studies_id] [int] NOT NULL,
	[link] [varchar](255) NOT NULL,
 CONSTRAINT [PK_syllabuses] PRIMARY KEY CLUSTERED
(
	[studies_id] 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].[syllabuses]  WITH CHECK ADD  CONSTRAINT
[FK_syllabuses_studies] FOREIGN KEY([studies_id])
REFERENCES [dbo].[studies] ([product_id])
GO
ALTER TABLE [dbo].[syllabuses] CHECK CONSTRAINT [FK_syllabuses_studies]
GO
```

Tabela theme_categories

Zawiera nazwy kategorii

Klucz główny: category_id

category_id - ID kategorii, not null
name - nazwa kategorii, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[theme_categories]    Script Date:
03.01.2024 13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[theme_categories](
	[category_id] [int] NOT NULL,
	[name] [varchar](50) NOT NULL,
 CONSTRAINT [PK_theme_categories] PRIMARY KEY CLUSTERED
(
	[category_id] 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
```

Tabela translators

Zawiera informacje o tłumaczach

Klucz główny: meeting_id
Klucz obcy: user_id (z employees), meeting_id (z meetings)
Unique: user_id

user_id - ID użytkownika, not null
meeting_id - ID spotkania, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[translators]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[translators](
	[user_id] [int] NOT NULL,
	[meeting_id] [int] NOT NULL,
 CONSTRAINT [PK_translators] PRIMARY KEY CLUSTERED
(
	[meeting_id] 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],
 CONSTRAINT [IX_translators] UNIQUE NONCLUSTERED
(
	[user_id] 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].[translators]  WITH CHECK ADD  CONSTRAINT
[FK_translators_employees] FOREIGN KEY([user_id])
REFERENCES [dbo].[employees] ([user_id])
GO
ALTER TABLE [dbo].[translators] CHECK CONSTRAINT
[FK_translators_employees]
GO
ALTER TABLE [dbo].[translators]  WITH CHECK ADD  CONSTRAINT
[FK_translators_meetings] FOREIGN KEY([meeting_id])
```

```
REFERENCES [dbo].[meetings] ([meeting_id])
GO
ALTER TABLE [dbo].[translators] CHECK CONSTRAINT
[FK_translators_meetings]
GO
```

Tabela translators_languages

Zawiera id tłumacza oraz języka, w którym tłumaczy

Klucz główny: translator_id, language_id
Klucz obcy: translator_id (z employees), language_id (z translators_languages)

translator_id - id tłumacza, not null
language_id - id języka, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[translators_languages]    Script Date:
03.01.2024 13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[translators_languages](
      [translator_id] [int] NOT NULL,
      [language_id] [int] NOT NULL,
 CONSTRAINT [PK_translators_languages] PRIMARY KEY CLUSTERED
(
      [translator_id] ASC,
      [language_id] 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].[translators_languages]  WITH CHECK ADD  CONSTRAINT
[FK_translators_languages_languages] FOREIGN KEY([language_id])
REFERENCES [dbo].[languages] ([language_id])
GO
ALTER TABLE [dbo].[translators_languages] CHECK CONSTRAINT
[FK_translators_languages_languages]
GO
ALTER TABLE [dbo].[translators_languages]  WITH CHECK ADD  CONSTRAINT
[FK_translators_languages_translators] FOREIGN KEY([translator_id])
REFERENCES [dbo].[employees] ([user_id])
GO
ALTER TABLE [dbo].[translators_languages] CHECK CONSTRAINT
[FK_translators_languages_translators]
GO
```

Tabela users

Zawiera dane wspólne dla wszystkich użytkowników

Klucz główny: username
Klucz obcy: user_id (z customers), user_id (z employees)
Checki: phone, który zawiera znaki 0-9
Unique: username

user_id - ID użytkownika, not null
firstname - imię użytkownika, not null
lastname - nazwisko użytkownika, not null
username - nazwa użytkownika, not null
phone - numer telefonu
is_active - przechowuje informacje czy użytkownik jest aktywny, wartość 0 - nie, wartość 1 - tak

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[users]    Script Date: 03.01.2024 13:20:00
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[users](
	[user_id] [int] IDENTITY(1,1) NOT NULL,
	[firstname] [varchar](50) NOT NULL,
	[lastname] [varchar](50) NOT NULL,
	[username] [varchar](50) NOT NULL,
	[phone] [varchar](20) NULL,
	[is_active] [bit] NOT NULL,
 CONSTRAINT [PK_users] PRIMARY KEY CLUSTERED
(
	[user_id] 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],
 CONSTRAINT [UQ_users_username] UNIQUE NONCLUSTERED
(
	[username] 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_phone]
CHECK  ((NOT [phone] like '%[^0-9]%'))
GO
ALTER TABLE [dbo].[users] CHECK CONSTRAINT [CK_users_phone]
GO
```

Tabela webinars

Zawiera informacje o dostępnych webinarach

Klucz główny: product_id
Klucz obcy: product_id (z products), module_id (z modules), lecturer_id (z users)
Checki: data rozpoczęcia musi być chronologicznie wcześniejsza niż data zakończenia
Unique: module_id

product_id - ID produktu, not null
module_id - ID modułu, not null
lecturer_id - ID wykładowcy, not null
start_date - data rozpoczęcia, not null
end_date - data zakończenia, not null

```
USE [u_kmadej]
GO
/****** Object:  Table [dbo].[webinars]    Script Date: 03.01.2024
13:20:00 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[webinars](
	[product_id] [int] NOT NULL,
	[module_id] [int] NOT NULL,
	[lecturer_id] [int] NOT NULL,
	[start_date] [datetime] NOT NULL,
	[end_date] [datetime] NOT NULL,
 CONSTRAINT [PK_webinars] PRIMARY KEY CLUSTERED
(
	[product_id] 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],
 CONSTRAINT [IX_webinars] UNIQUE NONCLUSTERED
(
	[module_id] 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].[webinars]  WITH CHECK ADD  CONSTRAINT
```

```
[FK_webinars_modules] FOREIGN KEY([module_id])
REFERENCES [dbo].[modules] ([module_id])
GO
ALTER TABLE [dbo].[webinars] CHECK CONSTRAINT [FK_webinars_modules]
GO
ALTER TABLE [dbo].[webinars]  WITH CHECK ADD  CONSTRAINT
[FK_webinars_products] FOREIGN KEY([product_id])
REFERENCES [dbo].[products] ([product_id])
GO
ALTER TABLE [dbo].[webinars] CHECK CONSTRAINT [FK_webinars_products]
GO
ALTER TABLE [dbo].[webinars]  WITH CHECK ADD  CONSTRAINT
[FK_webinars_users] FOREIGN KEY([lecturer_id])
REFERENCES [dbo].[employees] ([user_id])
GO
ALTER TABLE [dbo].[webinars] CHECK CONSTRAINT [FK_webinars_users]
GO
ALTER TABLE [dbo].[webinars]  WITH CHECK ADD  CONSTRAINT
[CK_webinars_dates] CHECK  (([start_date]<[end_date]))
GO
ALTER TABLE [dbo].[webinars] CHECK CONSTRAINT [CK_webinars_dates]
GO
```

Widoki

Najpopularniejsze kierunki (BK)

Zlicza najpopularniejsze kierunki studiów, na które uczęszcza najwięcej osób

product_id - ID produktu
product_name - nazwa produktu
total_students - ilość studentów

```
CREATE VIEW most_popular_products AS
SELECT p.product_id, p.product_name, COUNT(gm.user_id) AS total_students
FROM products p
INNER JOIN groups g ON p.product_id = g.product_id
INNER JOIN group_members gm ON g.group_id = gm.group_id
GROUP BY p.product_id, p.product_name
ORDER BY total_students DESC
GO
```

Najbardziej opłacalne kierunki (BK)

Oblicza które kierunki dają największy zysk

product_id - ID produktu
product_name - nazwa produktu
total_revenue - suma zarobków z deneko kierunku

```
CREATE VIEW most_profitable_products AS
SELECT p.product_id, p.product_name,
       SUM(p.price * gm.total_users) AS total_revenue
FROM products p
INNER JOIN (
    SELECT g.product_id, g.group_id,
           COUNT(gm.user_id) AS total_users
    FROM groups g
    INNER JOIN group_members gm ON g.group_id = gm.group_id
    GROUP BY g.product_id, g.group_id
) gm ON p.product_id = gm.product_id
GROUP BY p.product_id, p.product_name
ORDER BY total_revenue DESC
```

```
GO
```

Studenci, którzy nie ukończyli kursu/studium (kierunek) (BK)

Wyświetla studentów, którzy nie ukończyli studiów

firstname, lastname - imię i nazwisko
module_id - ID modułu z którego nie zdali

```
CREATE VIEW not_passed_students AS
SELECT u.firstname, u.lastname, f.module_id
FROM users u
INNER JOIN customers c ON u.user_id = c.user_id
INNER JOIN final_grades f ON c.user_id = f.user_id
INNER JOIN subjects s ON f.module_id = s.module_id
WHERE f.value = 2
GO
```

Absolwenci studium/kurs (BK)

Wyświetla studentów, którzy zaliczyli studia

firstname, lastname - imię i nazwisko
module_id - ID zaliczonego modułu

```
CREATE VIEW graduates AS
SELECT u.firstname, u.lastname, f.module_id
FROM users u
INNER JOIN customers c ON u.user_id = c.user_id
INNER JOIN final_grades f ON c.user_id = f.user_id
INNER JOIN subjects s ON f.module_id = s.module_id
WHERE f.value > 2
GO
```

Najmniej zdawalny kierunek (BK)

Pokazuje na których kierunkach najwięcej osób nie zalicza kursu

module_id - ID modułu
fail_count - ilość osób, która oblała

```
CREATE VIEW most_failed_major AS
SELECT f.module_id, COUNT(*) AS fail_count
FROM final_grades f
WHERE f.value = 2
GROUP BY f.module_id
ORDER BY fail_count DESC
GO
```

Najpopularniejsze języki (całość) (KM)

Zlicza najczęściej tłumaczone języki podczas spotkań, bez ograniczeń czasowych

Język - nazwa języka
Liczba spotkań - zliczona łączna liczba spotkań

```
CREATE VIEW languages_usage_all_time AS
SELECT l.language_name AS Język, COUNT(t.meeting_id) AS [Liczba spotkań]
FROM dbo.languages AS l
INNER JOIN dbo.translators_languages AS tl ON l.language_id =
tl.language_id
INNER JOIN dbo.employees AS e ON tl.translator_id = e.user_id
INNER JOIN dbo.translators AS t ON e.user_id = t.user_id
GROUP BY l.language_name
GO
```

Najpopularniejsze języki (rocznie) (KM)

Zlicza najczęściej tłumaczone języki podczas spotkań, z podziałem na lata

Język - nazwa języka
Rok - rok
Liczba spotkań - zliczona liczba spotkań w roku

```
CREATE VIEW languages_usage_yearly AS
SELECT l.language_name AS Język, YEAR(m.start_date) AS Rok,
COUNT(t.meeting_id) AS [Liczba spotkań]
FROM dbo.languages AS l
INNER JOIN dbo.translators_languages AS tl ON l.language_id =
tl.language_id
INNER JOIN dbo.employees AS e ON tl.translator_id = e.user_id
INNER JOIN dbo.translators AS t ON e.user_id = t.user_id
INNER JOIN dbo.meetings AS m ON m.meeting_id = t.meeting_id
GROUP BY l.language_name, YEAR(m.start_date)
GO
```

Wolne sale (KM)

Zwraca sale oraz przedziały czasowe, w których są one wolne.

Numer sali - numer sali
ID sali - ID sali
Początek - początek wolnego zakresu
Koniec - koniec wolnego zakresu

```
USE [u_kmadej]
GO

/****** Object:  View [dbo].[rooms_availability]    Script Date:
24.01.2024 00:04:52 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


CREATE VIEW [dbo].[rooms_availability] AS
with future_events as (
      select
            r.room_id 'ID Sali',
            r.number 'Numer Sali',
            m.start_date 'Czas Rozpoczęcia',
            m.end_date 'Czas Zakończenia'
      from
            rooms r
            join meetings_stationary ms
                  on r.room_id = ms.room_id
            join meetings m
                  on ms.meeting_id = m.meeting_id
      where getdate() < m.start_date
      union all
      select
            r.room_id 'ID Sali',
            r.number 'Numer Sali',
            e.start_time 'Czas Rozpoczęcia',
            e.end_time 'Czas Zakończenia'
      from
            rooms r
            join exams e
                  on e.room_id = r.room_id
```

```
	where getdate() < e.start_time
)

select * from (
	select
		r.number 'Numer Sali',
		r.room_id 'ID Sali',
		getdate() 'Początek',
		isnull((
			select top 1
				fe2.[Czas Rozpoczęcia]
			from
				future_events fe2
			where fe2.[ID Sali] = r.room_id
			order by fe2.[Czas Rozpoczęcia]
		), convert(datetime, '9999-12-31T23:59:59.997', 126))
'Koniec'
	from
		dbo.rooms r
	union
	select
		fe.[Numer Sali],
		fe.[ID Sali],
		fe.[Czas Zakończenia] 'Początek',
		case
			when lead(fe.[ID Sali]) over(order by fe.[ID Sali],
fe.[Czas Rozpoczęcia]) = fe.[ID Sali] then
				lead(fe.[Czas Rozpoczęcia]) over(order by fe.[ID
Sali], fe.[Czas Rozpoczęcia])
			else
				convert(datetime, '9999-12-31T23:59:59.997', 126)
		end 'Koniec'
	from
		future_events fe
) wtf_is_this
where wtf_is_this.Początek <> wtf_is_this.Koniec
GO
```

Oceny prowadzących (całość) (KM)

Zwraca średnią ocenę osób zatrudnionych, bez ograniczeń czasowych.

Imię - imię ocenianego prowadzącego
Nazwisko - nazwisko ocenianego prowadzącego
ID - ID ocenianego prowadzącego
Średnia ocena - średnia ocena prowadzącego w zakresie 1-10

```
CREATE VIEW employees_reviews_all_time AS
SELECT u.firstname AS Imię, u.lastname AS Nazwisko, u.user_id AS ID,
AVG(er.rating) AS [Średnia ocena]
FROM     dbo.employees_reviews AS er INNER JOIN
                  dbo.employees AS e ON e.user_id = er.reviewed_id INNER
JOIN
                  dbo.users AS u ON u.user_id = e.user_id
GROUP BY u.user_id, u.firstname, u.lastname
GO
```

Oceny prowadzących (rocznie) (KM)

Zwraca średnią ocenę osób zatrudnionych, z podziałem na lata.

Imię - imię ocenianego prowadzącego
Nazwisko - nazwisko ocenianego prowadzącego
ID - ID ocenianego prowadzącego
Rok - rok dla którego ocena była obliczona
Średnia ocena - średnia ocena prowadzącego w zakresie 1-10

```
CREATE VIEW employees_reviews_yearly AS
SELECT u.firstname AS Imię, u.lastname AS Nazwisko, u.user_id AS ID,
YEAR(er.date) AS Rok, AVG(er.rating) AS [Średnia ocena]
FROM     dbo.employees_reviews AS er INNER JOIN
                  dbo.employees AS e ON e.user_id = er.reviewed_id INNER
JOIN
                  dbo.users AS u ON u.user_id = e.user_id
GROUP BY u.user_id, u.firstname, u.lastname, YEAR(er.date)
GO
```

Osoby zapisane na przyszłe wydarzenia (lista) (KM)

Zwraca listę osób zapisanych na przyszłe wydarzenia.

Imię - imię osoby zapisanej na wydarzenie
Nazwisko - nazwisko osoby zapisanej
ID - id osoby zapisanej
ID Spotkania - id spotkania na które osoba jest zapisana
Czas Rozpoczęcia
Czas Zakończenia

```
CREATE VIEW future_meetings_participants AS
SELECT u.firstname AS Imię, u.lastname AS Nazwisko, u.user_id AS ID,
m.meeting_id AS [ID Spotkania], m.start_date AS [Czas Rozpoczęcia],
m.end_date AS [Czas Zakończenia]
FROM     dbo.meetings AS m INNER JOIN
                  dbo.meeting_participants AS mp ON mp.meeting_id =
m.meeting_id INNER JOIN
                  dbo.users AS u ON mp.user_id = u.user_id INNER JOIN
                  dbo.customers AS c ON u.user_id = c.user_id
WHERE  (GETDATE() < m.start_date)
GO
```

Osoby zapisane na przyszłe wydarzenia (liczba) (KM)

Zwraca liczbę osób zapisanych na przyszłe wydarzenia.

ID Spotkania - id spotkania na które osoba jest zapisana
Czas Rozpoczęcia
Czas Zakończenia
Liczba Zapisanych Uczestników - łączna liczba uczestników

```
CREATE VIEW future_meetings_participants_count AS
SELECT m.meeting_id AS [ID Spotkania], m.start_date AS [Czas
Rozpoczęcia], m.end_date AS [Czas Zakończenia], COUNT(mp.user_id) AS
[Liczba Zapisanych Uczestników]
FROM     dbo.meetings AS m INNER JOIN
                  dbo.meeting_participants AS mp ON mp.meeting_id =
m.meeting_id INNER JOIN
                  dbo.users AS u ON mp.user_id = u.user_id INNER JOIN
                  dbo.customers AS c ON u.user_id = c.user_id
WHERE  (GETDATE() < m.start_date)
GROUP BY m.meeting_id, m.start_date, m.end_date
GO
```

Raport dłużników (MW)

Zwraca dane użytkownika oraz wylicza dług z konkretnych zamówień

ID - id użytkownika
Imię i nazwisko
Dług - dodatnia różnica między kosztami zamówienia a opłatą (bądź jej brakiem)

```
CREATE view [dbo].[debts] as (
	SELECT u.user_id AS ID, u.firstname + ' ' + u.lastname AS 'Imie i
nazwisko',
		t1.order_id, t1.price1 - COALESCE(t2.price2, 0) AS Dlug
	FROM (
		SELECT o.user_id AS id, o.order_id, SUM(p.price) AS price1
		FROM orders o
		INNER JOIN order_items oi ON o.order_id = oi.order_id
		INNER JOIN products p ON oi.product_id = p.product_id
		GROUP BY o.user_id, o.order_id
	) t1
	LEFT JOIN (
		SELECT p.order_id, SUM(p.value) AS price2
		FROM payments p
		GROUP BY p.order_id
	) t2 ON t1.order_id = t2.order_id
	INNER JOIN users u ON t1.id = u.user_id
	WHERE t1.price1 - COALESCE(t2.price2, 0) > 0
)
GO
```

Zaakceptowane wnioski (MW)

Zwraca dane użytkownika oraz numery zamówień na które przyjęto wniosek

ID - id użytkownika
Imię i nazwisko
order_id - zamówienie, na które złożono wniosek

```
CREATE VIEW accepted_applications AS
select u.user_id as ID, u.firstname + ' ' + u.lastname as 'Imie i
nazwisko', o.order_id
from users u
inner join orders o
on u.user_id = o.user_id
inner join applications a
on o.order_id = a.order_id
where a.accepted = 1
GO
```

Raport bilokacji (MW)

Zwraca dane użytkownika (id, imię i nazwisko) oraz dania spotkań, które pokrywają się

ID - id użytkownika
Imię i nazwisko
meeting_id - id kolidującego spotkania
start_date - data rozpoczęcia
end_date - data zakończenia

```
CREATE view [dbo].[bilocations] as
select u.user_id as ID, u.firstname + ' ' + u.lastname as 'Imie i
nazwisko',
      m.meeting_id, m.start_date, m.end_date
from users u
inner join meeting_participants mp
on u.user_id = mp.user_id
inner join meetings m
on mp.meeting_id = m.meeting_id
where exists
        (select 1
        from meeting_participants mp2
        inner join meetings m2
            on mp2.meeting_id = m2.meeting_id
        where
            mp2.user_id = mp.user_id
                  and m.meeting_id <> m2.meeting_id
            and ((m2.start_date between m.start_date and m.end_date)
                or (m2.end_date between m.start_date and m.end_date)
                or (m.start_date between m2.start_date and m2.end_date)
                or (m.end_date between m2.start_date and m2.end_date)))

GO
```

Zajęcia w danym tygodniu (MW)

Zwraca grupy, które mają spotkania w danym tygodniu

group_id - id grupy
start_date - data rozpoczęcia spotkania

```
CREATE VIEW current_week_meetings AS
select mg.group_id, mt.start_date
from modules_groups mg
inner join meetings mt
on mg.module_id = mt.module_id
where year(mt.start_date) = year(getdate()) and datepart(week,
mt.start_date) = datepart(week, getdate())
GO
```

Zajęcia w danym miesiącu (MW)

Zwraca grupy, które mają spotkania w danym miesiącu

group_id - id grupy
start_date - data rozpoczęcia spotkania

```
CREATE VIEW current_month_meetings AS
select mg.group_id, mt.start_date
from modules_groups mg
inner join meetings mt
on mg.module_id = mt.module_id
where year(mt.start_date) = year(getdate()) and datepart(month,
mt.start_date) = datepart(month, getdate())
GO
```

Obecność każdego ucznia (MW)

Zwraca dane używkownika (id, imię i nazwisko) oraz wylicza procent obecności na podstawie ilości odbytych spotkań

module_id - id modułu
user_id - id użytkownika
Imię i nazwisko
Obecność - procent obecności na danym module

```
CREATE view [dbo].[users_presence] as
      SELECT m.module_id, p.user_id, u.firstname + ' ' + u.lastname AS
'Imie i nazwisko',
            CAST(SUM(CAST(p.is_present AS INT)) * 100.0 /
COUNT(m.meeting_id) AS DECIMAL(10, 2)) AS Obecnosc
      FROM meetings m
      INNER JOIN presence p ON m.meeting_id = p.meeting_id
      INNER JOIN users u ON p.user_id = u.user_id
      GROUP BY m.module_id, p.user_id, u.firstname, u.lastname;
GO
```

Procedury

Procedura create_application (KM)

Procedura create_application tworzy nowe zgłoszenie, wstawiając nowy wiersz do tabeli applications i zwraca identyfikator tego zgłoszenia.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_application](
   @application_id int output,
   @order_id int
) as begin
   set nocount on
   insert into applications(
      order_id,
      date,
      accepted
   )
   values(
      @order_id,
      getdate(),
      null
   )
   set @application_id = @@identity
end
```

Procedura accept_application (KM)

Procedura accept_application akceptuje zgłoszenie, zmieniając status accepted w tabeli applications na 1 dla konkretnego zgłoszenia o określonym identyfikatorze.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[accept_application](
   @application_id int
) as begin
   set nocount on
   update applications set
      accepted = 1
   where application_id = @application_id
end
```

Procedura create_cart (KM)

Procedura create_cart tworzy nowy koszyk, wstawiając nowy wiersz do tabeli carts i zwraca identyfikator tego koszyka.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_cart](
   @cart_id int output,
   @user_id int
) as begin
   insert into carts(
      user_id
   )
   values(
      @user_id
   )

   set @cart_id = @@identity
end
```

Procedura add_item_to_cart (KM)

Procedura add_item_to_cart dodaje nowy produkt do koszyka, wstawiając nowy wiersz do tabeli cart_items.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_item_to_cart](
   @cart_id int,
   @product_id int
) as begin
   insert into cart_items(
      cart_id,
      product_id
   )
   values(
      @cart_id,
      @product_id
   )
end
```

Procedura remove_item_to_cart (KM)

Procedura remove_item_to_cart usuwa produkt o określonym identyfikatorze (product_id) z koszyka o określonym identyfikatorze (cart_id).

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[remove_item_from_cart](
   @cart_id int,
   @product_id int
) as begin
   delete from cart_items
   where
      cart_id = @cart_id
      and product_id = @product_id
end
```

Procedura remove_cart (KM)

Procedura remove_cart usuwa koszyk o określonym identyfikatorze (cart_id) wraz z jego zawartością (elementami koszyka w tabeli cart_items).

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[remove_cart](
   @cart_id int
) as begin
   delete from cart_items
   where
      cart_id = @cart_id
   delete from carts
   where cart_id = @cart_id
end
```

Procedura add_city (KM)

Procedura add_city dodaje nowe miasto, wstawiając nowy wiersz do tabeli cities, i zwraca identyfikator tego miasta.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_city](
   @city_id int output,
   @city_name varchar(50),
   @country_id int
) as begin
   insert into cities(
      name,
      country_id
   )
   values(
      @city_name,
      @country_id
   )
```

```
   set @city_id = @@identity
end
```

Procedura add_country (KM)

Procedura add_country dodaje nowy kraj, wstawiając nowy wiersz do tabeli countries, i zwraca identyfikator tego kraju.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_country](
   @country_id int output,
   @country_name nvarchar(50)
) as begin
   insert into countries(
      name
   )
   values(
      @country_name
   )

   set @country_id = @@identity
end
```

Procedura create_course (KM)

Procedura create_course tworzy nowy kurs w bazie danych, włączając utworzenie związanych z nim produktu, grupy i informacji o kursie.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_course](
   @product_id int output,
   @group_id int output,
   @course_name varchar(50),
```

```
   @lecturer_id int,
   @start_date date,
   @end_date date,
   @price money
) as begin
   exec create_product @product_id output, @course_name, @price

   insert into courses(
      product_id,
      lecturer_id,
      start_date,
      end_date
   )
   values(
      @product_id,
      @lecturer_id,
      @start_date,
      @end_date
   )

   exec create_group @group_id output, @lecturer_id, @product_id
end
```

Procedura create_customer (KM)

Procedura create_customer tworzy nowego klienta, co uwzględnia utworzenie związanego z nim użytkownika, koszyka i informacji o kliencie.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_customer](
   @user_id int output,
   @firstname varchar(50),
   @lastname varchar(50),
   @username varchar(50),
   @phone varchar(15),
   @city_id int,
   @street varchar(50),
   @zip_code varchar(20),
   @is_active BIT = 1
```

```
   ) as begin
      declare @cart_id int
      exec create_user @user_id output, @firstname, @lastname,
@username, @phone, @is_active
      exec create_cart @cart_id output, @user_id
      insert into customers(
         user_id,
         city_id,
         street,
         zip_code
      )
      values(
         @user_id,
         @city_id,
         @street,
         @zip_code
      )
   end
```

Procedura create_diploma (KM)

Procedura create_diploma tworzy nowy dyplom, wstawiając nowy wiersz do tabeli diplomas, i zwraca identyfikator tego dyplomu.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_diploma](
   @diploma_id int output,
   @customer_id int,
   @product_id int
) as begin
   insert into diplomas(
      user_id,
      product_id
   )
   values(
      @customer_id,
      @product_id
   )

   set @diploma_id = @@identity
```

```
end
```

Procedura create_employee (KM)

Procedura ta tworzy nowego pracownika, co uwzględnia utworzenie związanego z nim użytkownika i informacji o pracowniku.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_employee](
   @user_id int output,
   @firstname varchar(50),
   @lastname varchar(50),
   @username varchar(50),
   @phone varchar(15),
   @role_id int,
   @is_active BIT = 1
) as begin
   exec create_user @user_id output, @firstname, @lastname,
@username, @phone, @is_active
   insert into employees(
      user_id,
      role_id
   )
   values(
      @user_id,
      @role_id
   )
end
```

Procedura add_employee_review (KM)

Procedura ta dodaje recenzję pracownika do tabeli employees_reviews, rejestrując ocenę, treść recenzji oraz datę jej dodania.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_employee_review](
   @employee_id int,
   @rating int,
   @review ntext
) as begin
   insert into employees_reviews(
      reviewed_id,
      date,
      review,
      rating
   )
   values(
      @employee_id,
      getdate(),
      @review,
      @rating
   )
end
```

Procedura add_grade (KM)

Procedura ta dodaje ocenę do tabeli grades, rejestrując użytkownika, moduł, datę dodania oraz wartość oceny. Jeśli parametr @value nie zostanie przekazany, wartość oceny zostanie ustawiona na wartość domyślną (null).

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_grade](
   @user_id int,
   @module_id int,
   @value real = null
) as begin
   insert into grades(
      user_id,
      module_id,
      date,
      value
   )
   values(
      @user_id,
      @module_id,
```

```
        getdate(),
        @value
    )
end
```

Procedura change_grade (KM)

Procedura ta zmienia wartość oceny dla danego użytkownika, modułu i daty, jeśli ocena o podanych kryteriach istnieje. W przeciwnym razie, generuje błąd informujący o braku oceny o podanych kryteriach.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[change_grade](
    @user_id int,
    @module_id int,
    @date datetime,
    @value real = null
) as begin
    if exists(
        select
            *
        from grades g
        where
            g.date = @date
            and g.user_id = @user_id
            and g.module_id = @module_id
    ) begin
        update grades set
            date = @date,
            user_id = @user_id,
            module_id = @module_id,
            value = @value
        where
            grades.date = @date
            and grades.user_id = @user_id
            and grades.module_id = @module_id
    end
    else begin
        raiserror('Nie znaleziono oceny.', 11, 1)
    end
```

```
end
```

Procedura create_exam (KM)

Procedura ta tworzy nowy egzamin, sprawdzając dostępność sali oraz istnienie egzaminu o podanych kryteriach, i zgłaszając odpowiednie błędy w przypadku niespełnienia tych warunków.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_exam](
   @exam_id int output,
   @group_id int,
   @module_id int,
   @start_time datetime,
   @end_time datetime,
   @term int,
   @room_id int
) as begin
   if not exists(
      select
         *
      from exams
      where
         group_id = @group_id
         and module_id = @module_id
         and term = @term
   ) begin
      declare @room_available bit
      exec room_is_available @room_id, @start_time, @end_time,
@room_available
      if @room_available = 1 begin
         insert into exams(
            group_id,
            module_id,
            start_time,
            end_time,
            term,
            room_id
         )
         values(
```

```
            @group_id,
            @module_id,
            @start_time,
            @end_time,
            @term,
            @room_id
         )
      end
      else begin
         raiserror('Sala nie jest dostępna w podanym przedziale
czasowym', 11, 1)
      end
   end
   else begin
      raiserror('Egzamin o danym terminie już istnieje', 11, 1)
   end
end
```

Procedura set_exam_grades (KM)

Procedura ta ustawia ocenę dla danego użytkownika i egzaminu w tabeli exam_grades, dodając nowy wpis, jeśli ocena nie istnieje, lub aktualizując istniejącą ocenę.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[set_exam_grade](
   @exam_id int,
   @user_id int,
   @grade real = null
) as begin
   if not exists(
      select
         *
      from exam_grades eg
      where
         eg.user_id = @user_id
         and eg.exam_id = @exam_id
   ) begin
      insert into exam_grades(
```

```
            exam_id,
            user_id,
            grade
        )
        values(
            @exam_id,
            @user_id,
            @grade
        )
    end
    else begin
        update exam_grades set
            grade = @grade
        where
            exam_grades.exam_id = @exam_id
            and exam_grades.user_id = @user_id
    end
end
```

Procedura set_final_grades (KM)

Procedura ta ustawia ocenę końcową dla danego użytkownika i modułu w tabeli final_grades, dodając nowy wpis, jeśli ocena końcowa nie istnieje, lub aktualizując istniejącą ocenę końcową.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[set_final_grade](
    @user_id int,
    @module_id int,
    @value real = null
) as begin
    if not exists(
        select
            *
        from final_grades fg
        where
            fg.user_id = @user_id
            and fg.module_id = @module_id
    ) begin
        insert into final_grades(
            user_id,
```

```
            module_id,
            date,
            value
        )
        values(
            @user_id,
            @module_id,
            getdate(),
            @value
        )
    end
    else begin
        update final_grades set
            date = getdate(),
            value = @value
        where
            final_grades.user_id = @user_id
            and final_grades.module_id = module_id
    end
end
```

Procedura add_to_group (KM)

Procedura ta dodaje użytkownika do określonej grupy w bazie danych, tworząc wpis w tabeli group_members.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_to_group](
    @group_id int,
    @user_id int
) as begin
    insert into group_members(
        group_id,
        user_id
    )
    values(
        @group_id,
        @user_id
    )

    exec add_to_group_meetings @group_id, @user_id
```

```
end
```

Procedura add_to_group_meetings (KM)

Procedura ta dodaje użytkownika do wszystkich przyszłych spotkań danej grupy. Jeżeli użytkownik nie należy do podanej grupy, rzuca błąd.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

create procedure [dbo].[add_to_group_meetings] (
   @group_id int,
   @user_id int
) as begin
   if not exists (
      select
         *
      from
         group_members gm
      where
         gm.group_id = @group_id
         and gm.user_id = @user_id
   ) begin
      raiserror('Użytkownik nie należy do podanej grupy!', 1, 11)
   end

   declare add_to_group_meetings_c cursor for
      select m.meeting_id
      from meetings m
      where m.group_id = @group_id and m.start_date >= getdate()

   declare @meeting_id int

   open add_to_group_meetings_c
   fetch next from add_to_group_meetings_c  into @meeting_id
   while @@fetch_status = 0 begin
      exec create_presence @meeting_id, @user_id
      exec add_participant @meeting_id, @user_id

      fetch next from add_to_group_meetings_c  into @meeting_id
```

```
    end
    close add_to_group_meetings_c
    deallocate add_to_group_meetings_c
end
GO
```

Procedura remove_from_group (KM)

Procedura ta usuwa użytkownika z określonej grupy w bazie danych, usuwając odpowiedni wpis z tabeli group_members.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[remove_from_group](
    @user_id int,
    @group_id int
) as begin
    delete from group_members
    where
        group_id = @group_id
        and user_id = @user_id
end
```

Procedura change_group_cooridinator (KM)

Procedura ta zmienia koordynatora dla określonej grupy w bazie danych, aktualizując odpowiednie pole w tabeli groups.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[change_group_coordinator](
    @group_id int,
    @coordinator_id int
) as begin
```

```
   update groups set
      coordinator_id = @coordinator_id
   where group_id = @group_id
end
```

Procedura create_group (KM)

Procedura ta tworzy nową grupę w bazie danych, przypisując koordynatora i produkt do grupy. Sprawdza też, czy użytkownik o identyfikatorze @coordinator_id istnieje w tabeli employees. Jeśli nie, procedura rzuca błąd.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_group](
   @group_id int output,
   @coordinator_id int,
   @product_id int
) as begin
   if exists (
      select
         *
      from employees
      where user_id = @coordinator_id
   ) begin

      insert into groups(
         coordinator_id,
         product_id
      )
      values(
         @coordinator_id,
         @product_id
      )

      set @group_id = @@identity
   end
   else begin
      raiserror('Podany użytkownik nie jest pracownikiem', 11, 1)
   end
end
```

Procedura find_group_to_add_to(KM)

Procedura ta znajduje grupę, której należy dodać użytkownika, który opłacił zamówiony produkt.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

create procedure [dbo].[find_group_to_add_to] (
   @group_id int output,
   @product_id int
) as begin
   if @product_id in (select s.product_id from studies s) begin
      set @group_id = (
         select top 1
            g.group_id
         from
            groups g
            join studies_groups sg on
               sg.group_id = g.group_id
         where
            g.product_id = @product_id
            and cast(sg.start_date as datetime) >= getdate()
         order by cast(sg.start_date as datetime) asc
      )
   end
   else begin
      set @group_id = (
         select
            g.group_id
         from groups g
         where g.product_id = @product_id
      )
   end
end
GO
```

Procedura create_studies (KM)

Procedura ta tworzy nowy produkt (korzystając z innej procedury create_product) i dodaje informacje o tym produkcie do tabeli studies.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_studies](
   @product_id int output,
   @price money,
   @price_for_outsiders money,
   @product_name varchar(50)
   ) as
      begin
         exec create_product @product_id output, @product_name,
@price;
         insert into studies (
            product_id,
            price_for_outsiders
         )
         values (
            @product_id,
            @price_for_outsiders
         )
      end
GO
```

Procedura create_studies_groups (KM)

Procedura ta tworzy nową grupę studencką poprzez wywołanie procedury create_group z określonymi parametrami. Dodaje informacje o utworzonej grupie do tabeli studies_groups. Przypisuje moduły do utworzonej grupy na podstawie danych z tabeli studies_subjects. Wykorzystuje procedurę add_module_to_group.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_studies_groups] (
   @start_date date,
   @end_date date,
   @coordinator_id int,
   @product_id int
   ) as
```

```
      begin
         declare @group_id int;
         exec create_group @group_id output, @coordinator_id,
@product_id;
         insert into studies_groups(
            group_id,
            start_date,
            end_date
         )
         values (
            @group_id,
            @start_date,
            @end_date
         )

         declare create_studies_groups_c cursor for
            select ss.module_id
            from studies_subjects ss
            inner join studies s
            on ss.product_id = s.product_id
            where s.product_id = 7

            declare @id int;

             open create_studies_groups_c;
            fetch next from create_studies_groups_c into @id;

            while @@fetch_status = 0
            begin
               exec add_module_to_group @id, @group_id;
               fetch next from create_studies_groups_c into @id;
            end

            close create_studies_groups_c
            deallocate create_studies_groups_c

      end
GO
```

Procedura pass_internship (MW)

Procedura ta umożliwia oznaczenie zaliczenia praktyk przez danego użytkownika w kontekście określonych studiów.

```
USE [u_kmadej]
```

```
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[pass_internship] (
   @user_id int,
   @studies_id int
   ) as
      begin
         if not exists (
            select * from internships
            where user_id = @user_id and studies_id = @studies_id
         )
            begin;
               raiserror('Not found ', 11, 1)
            end
         update internships
         set passed = 1
         where user_id = @user_id and studies_id = @studies_id
      end
```

Procedura add_language (MW)

Procedura ta umożliwia dodanie nowego języka do tabeli, przy czym sprawdza, czy taki język już istnieje w bazie danych. Jeśli już istnieje, procedura nie wykonuje dodawania i informuje o tym poprzez rzucenie błędu.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_language] (
   @language_id int output,
   @language_name nvarchar(50)
   ) as
      begin
         if exists (
            select * from languages
            where language_name= @language_name
         )
            begin;
               throw 50000, 'Language already in table', 1
```

```
            end
         insert into languages (
            language_name
         )
         values (
            @language_name
         )
         set @language_id = @@identity
      end
```

Procedura create_meeting (MW)

Procedura ta tworzy nowe spotkanie, a następnie dla każdego uczestnika grupy wykonuje dwie procedury, które dodają obecność i uczestnika do spotkania.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_meeting]
   (
      @meeting_id int output,
      @start_date datetime,
      @end_date datetime,
      @module_id int,
      @group_id int
   ) as
      begin
            insert into meetings(
               start_date,
               end_date,
               module_id,
               group_id
            )
            values (
               @start_date,
               @end_date,
               @module_id,
               @group_id
            )
            set @meeting_id = @@identity

            declare create_meeting_c cursor for
```

```
            select gm.user_id
            from group_members gm
            where gm.group_id = @group_id

            declare @id int;

             open create_meeting_c;
            fetch next from create_meeting_c into @id;

            while @@fetch_status = 0
            begin
               exec create_presence @meeting_id, @id;
               exec add_participant @meeting_id, @id;
               fetch next from create_meeting_c into @id;
            end

            close create_meeting_c
            deallocate create_meeting_c

      end
```

Procedura change_meeting_date (MW)

Procedura ta umożliwia dostosowanie daty i czasu istniejącego spotkania do nowych wartości. Jeśli podane spotkanie nie istnieje, rzuca błąd.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[change_meeting_date] (
   @meeting_id int,
   @start_date datetime,
   @end_date datetime,
   @module_id int
   ) as
      begin
         if not exists (
            select * from meetings
            where meeting_id = @meeting_id
         )
            begin;
               raiserror('Meeting not found ', 11, 1)
```

```
            end
         update meetings
         set start_date = @start_date,
            end_date = end_date
         where meeting_id = @meeting_id
      end
```

Procedura create_meeting_stationary (MW)

Procedura ta umożliwia tworzenie spotkań stacjonarnych, kontrolując jednocześnie dostępność pokoju w danym przedziale czasowym.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE procedure [dbo].[create_stationary_meeting] (
      @meeting_id int output,
      @room_id int,
      @start_date datetime,
      @end_date datetime,
      @module_id int,
      @group_id int
   ) as
      begin
            declare @is_available bit
            exec room_is_available @room_id, @start_date,
@end_date, @is_available output;

            if @is_available = 1
               begin
                  declare @meeting_id_temp int
                  exec create_meeting @meeting_id output,
@start_date, @end_date, @module_id, @group_id;

                  insert into meetings_stationary(
                     meeting_id,
                     room_id
                  )
                  values (
                     @meeting_id,
                     @room_id
```

```
                )
            end
        else
            begin
                raiserror('Room not available ', 11, 1)
            end
    end
```

Procedura create_online_meeting (MW)

Procedura ta umożliwia tworzenie spotkań online, zapisując informacje o nich w odpowiedniej tabeli.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_online_meeting] (
    @meeting_id int output,
    @link varchar(200),
    @start_date datetime,
    @end_date datetime,
    @module_id int,
    @group_id int
) as
    begin
        exec create_meeting @meeting_id output, @start_date,
@end_date, @module_id, @group_id;

        insert into meetings_online(
            meeting_id,
            link
        )
        values (
            @meeting_id,
            @link
        )
    end
```

Procedura create_module (MW)

Procedura ta pozwala na tworzenie modułów w bazie danych.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE procedure [dbo].[create_module]
   (
      @module_id int output
   ) as
      begin
            insert into modules default values
            set @module_id = @@identity
      end
```

Procedura add_module_category (MW)

Procedura ta pozwala na przypisanie kategorii do konkretnego modułu w systemie.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_module_category] (
   @module_id int,
   @category_id int
   ) as
      begin
            insert into modules_categories(
               module_id,
               category_id
            )
            values (
               @module_id,
               @category_id
            )
      end
```

Procedura add_module_to_group (MW)

Procedura ta umożliwia przypisanie modułu do konkretnej grupy w systemie.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_module_to_group] (
   @module_id int,
   @group_id int
   ) as
      begin
         insert into modules_groups(
            module_id,
            group_id
         )
         values (
            @module_id,
            @group_id
         )
      end
```

Procedura create_order (MW)

Procedura create_order tworzy nowe zamówienie dla danego użytkownika, zapisując informacje o zamówieniu w bazie danych i przypisując nowemu zamówieniu unikalne ID.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_order] (
   @order_id int output,
   @user_id int
   ) as
      begin
         insert into orders (
            user_id,
            order_date
         )
         values (
```

```
            @user_id,
            getdate()
        )
        set @order_id = @@identity
    end
```

Procedura create_order_from_cart (MW)

Procedura ta umożliwia użytkownikowi złożenie zamówienia na podstawie produktów znajdujących się w jego koszyku. Produkty te zostaną przeniesione do zamówienia, a z koszyka zostaną usunięte.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_order_from_cart] (
   @order_id int output,
   @cart_id int
   ) as
      begin
      if exists (
         select * from cart_items where cart_id = @cart_id
      ) begin
         declare @user_id int = (
            select user_id from carts where cart_id = @cart_id
         )


         exec create_order @order_id output, @user_id;


         declare create_order_from_cart_c cursor for
         select product_id
         from cart_items
         where cart_id = @cart_id


         declare @id int;


         open create_order_from_cart_c ;
         fetch next from create_order_from_cart_c  into @id;
```

```
        while @@fetch_status = 0
        begin
           exec add_product_to_order @order_id, @id;
           fetch next from create_order_from_cart_c  into @id;
        end


        close create_order_from_cart_c
        deallocate create_order_from_cart_c


        exec remove_cart @cart_id
     end
     else begin
        raiserror('Podany koszyk jest pusty!', 1, 11)
     end
  end
GO
```

Procedura create_payment (MW)

Procedura create_payment tworzy nowy rekord płatności dla danego zamówienia, zapisując informacje o płatności w bazie danych i przypisując nowej płatności unikalne ID. due_date (termin płatności) jest ustawiany na 14 dni od daty utworzenia płatności.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_payment] (
   @payment_id int output,
   @order_id int,
   @payment_date datetime,
   @value money
   ) as
      begin
         insert into payments(
            order_id,
            due_date,
            payment_date,
            value
         )
```

```
        values (
           @order_id,
           dateadd(day, 14, getdate()),
           @payment_date,
           @value
        )
        set @payment_id = @@identity
     end
```

Procedura pay_order (MW)

Procedura ta umożliwia klientowi dokonanie płatności dla konkretnego zamówienia, aktualizując wartość płatności i datę płatności w bazie danych. W przypadku, gdy płatność o podanym ID nie istnieje, procedura zgłasza błąd.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[pay_order] (
   @payment_id int,
   @order_id int,
   @due_date datetime,
   @payment_date datetime,
   @value money
   ) as
      begin
         if not exists (
            select * from payments
            where payment_id = @payment_id and order_id =
@order_id
         )
            begin;
               raiserror('Payment not found ', 11, 1)
            end
         update payments
         set value = value + @value,
            payment_date = @payment_date
         where payment_id = @payment_id and order_id = @order_id
      end
```

Procedura create_presence (MW)

Procedura ta pozwala na inicjalizację wpisu obecności dla danego użytkownika w określonym spotkaniu. Wartość is_present jest początkowo ustawiana na 0.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_presence](
   @meeting_id int,
   @user_id int
   ) as
      begin
         insert into presence (
            meeting_id,
            user_id,
            is_present
         )
         values (
            @meeting_id,
            @user_id,
            0
         )
      end
```

Procedura set_presence (MW)

Procedura ta umożliwia zmianę statusu obecności użytkownika na danym spotkaniu. Jeśli użytkownika nie ma na liście obecności, generowany jest błąd. W przeciwnym razie status obecności jest aktualizowany zgodnie z podanym parametrem @is_present.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[set_presence] (
   @meeting_id int,
   @user_id int,
   @is_present bit
   ) as
      begin
```

```
        if not exists (
           select * from presence
           where user_id = @user_id and meeting_id = @meeting_id
        )
           begin;
              raiserror('Student not on a present list ', 11, 1)
           end
        update presence
        set is_present = 1
        where user_id = @user_id and meeting_id = @meeting_id
     end
```

Procedura create_product (MW)

Procedura ta umożliwia dodanie nowego produktu do bazy danych, przypisując mu automatycznie generowane ID.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_product]
   (
      @product_id int output,
      @product_name varchar(50),
      @price money
   ) as
      begin
            insert into products(
               product_name,
               price
            )
            values (
               @product_name,
               @price
            )
            set @product_id = @@identity
      en
```

Procedura add_recording (BK)

Procedura add_recording dodaje nowe nagranie do bazy danych, tworząc zasób, przypisując go do spotkania i zwracając jego ID

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[add_recording]
   @resource_id INT output,
   @meeting_id INT,
   @link NVARCHAR(MAX),
   @module_id INT,
   @add_date DATETIME = NULL
AS
BEGIN
   SET NOCOUNT ON;

  exec add_resource @link, @module_id, @add_date
  INSERT INTO recordings (resource_id)
  VALUES (@resource_id);
  INSERT INTO meetings_recordings (meeting_id, recording_id)
  VALUES (@meeting_id, @resource_id);
  SET @resource_id = @@identity
END
```

Procedura add_resource (BK)

Procedura add_resource dodaje nowy zasób do bazy danych, przypisując mu link, ID modułu i opcjonalną datę dodania, a następnie zwraca ID tego zasobu

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[add_resource]
   @resource_id INT output,
   @link NVARCHAR(MAX),
   @module_id INT,
   @add_date DATETIME = NULL
AS
BEGIN
   SET NOCOUNT ON;

  INSERT INTO resources (link, module_id, add_date)
  VALUES (@link, @module_id, COALESCE(@add_date, GETDATE()));
  SET @resource_id = @@identity
END
```

Procedura create_role (BK)

Procedura create_role dodaje nową rolę do bazy danych, przypisując jej nazwę, a następnie zwraca ID tej roli

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[create_role]
   @role_id INT output,
   @name NVARCHAR(50)
AS
BEGIN
   SET NOCOUNT ON;

   BEGIN
       INSERT INTO roles (name)
```

```
        VALUES (@name)
       set @role_id = @@identity

   END
END
```

Procedura add_room (BK)

Procedura add_room dodaje nowe pomieszczenie do bazy danych, przypisując mu numer, a następnie zwraca ID tego pomieszczenia

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[add_room]
   @room_id INT output,
   @number varchar(10)
AS
BEGIN
   SET NOCOUNT ON;

   BEGIN
       INSERT INTO rooms (number)
       VALUES (@number)
      set @room_id = @@identity

   END

END
```

Procedura add_studies (BK)

Procedura add_studies dodaje nowe studia do bazy danych, tworząc produkt studiów, moduł studiów i przypisując im odpowiednie informacje

```

USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE PROCEDURE [dbo].[add_studies]
   @product_id INT output,
   @price_for_outsiders money,
   @module_id INT,
   @course_name varchar(50),
   @price money
AS
BEGIN
   SET NOCOUNT ON;

   exec create_product @product_id output, @course_name, @price
   exec create_module @module_id output

   INSERT INTO studies (product_id, price_for_outsiders)
   VALUES (@product_id,@price_for_outsiders)

END
```

Procedura add_studies_group (BK)

Procedura add_studies_group dodaje nową grupę studiów do bazy danych, przypisując jej daty rozpoczęcia i zakończenia, a następnie zwraca ID tej grupy studiów

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[add_studies_group]
   @group_id INT output,
   @start_date DATE,
   @end_date DATE
AS
BEGIN
   SET NOCOUNT ON;

   INSERT INTO studies_groups (start_date, end_date)
   VALUES (@start_date, @end_date)
   set @group_id = @@identity
END
```

Procedura add_studies_subject (BK)

Procedura add_studies_subject tworzy moduł studiów, przedmiot studiów, grupę studencką, łączy moduł z grupą, a następnie dodaje informacje o przedmiocie studiów do odpowiedniej tabeli

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_studies_subject] (
   @module_id int output,
   @product_id int,
   @master_id int,
   @subject_name varchar(50)
   ) as
      begin
         declare @group_id int;
         exec create_module @module_id output;
         exec create_subject @module_id, @subject_name;
         exec create_group @group_id output, @master_id,
@product_id;
         exec add_module_to_group @module_id, @group_id;
         insert into studies_subjects (
            module_id,
            product_id,
            master_id
         )
         values (
            @module_id,
            @product_id,
            @master_id
         )
      end
```

Procedura add_syllabus (BK)

Procedura add_syllabus umożliwia dodanie syllabusa do bazy danych, przypisując go konkretnym studiom

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[add_syllabus]
   @studies_id INT,
   @link NVARCHAR(200)
AS
BEGIN
   SET NOCOUNT ON;

   INSERT INTO syllabuses (studies_id, link)
   VALUES (@studies_id, @link)
END
```

Procedura create_subject (BK)

Procedura create_subject tworzy nowy przedmiot w kontekście określonego modułu, przy użyciu innej procedury do utworzenia modułu

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[create_subject] (
   @module_id int output,
   @subject_name varchar(50)
   ) as
      begin
         exec create_module @module_id output;
         insert into subjects (
            module_id,
            subject_name
         )
         values (
            @module_id,
            @subject_name
         )
      end
```

Procedura create_theme_category (BK)

Procedura create_theme_category dodaje nową kategorię przedmiotu do bazy danych, przypisując jej nazwę, a następnie zwraca ID tej kategorii

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[create_theme_category]
   @category_id INT output,
   @name NVARCHAR(100)
AS
BEGIN
   SET NOCOUNT ON;

   INSERT INTO theme_categories (name)
   VALUES (@name)
   set @category_id = @@identity
END
```

Procedura add_translator (BK)

Procedura add_translator służy do dodawania informacji o tłumaczu do bazy danych, określając użytkownika, spotkanie i język tłumaczenia.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[add_translator]
   @user_id INT,
   @meeting_id INT,
   @language_id INT
AS
BEGIN
   SET NOCOUNT ON;

   INSERT INTO translators (user_id, meeting_id, language_id)
   VALUES (@user_id, @meeting_id, @language_id)

END
```

Procedura add_translator_language (BK)

Procedura add_translator_language służy do dodawania informacji o języku, którym posługuje się tłumacz

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[add_translator_language]
   @translator_id INT,
   @language_id INT
AS
BEGIN
   SET NOCOUNT ON;

   INSERT INTO translators_languages (translator_id, language_id)
   VALUES (@translator_id, @language_id)

END
```

Procedura create_user (BK)

Procedura create_user dodaje nowego użytkownika do bazy danych, przypisując mu imię, nazwisko, nazwę użytkownika, numer telefonu i określając jego aktywność.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[create_user]
   @user_id INT output,
   @firstname VARCHAR(50),
   @lastname VARCHAR(50),
   @username VARCHAR(50),
   @phone VARCHAR(15),
   @is_active BIT
AS
BEGIN
```

```
    SET NOCOUNT ON;

    INSERT INTO users (firstname, lastname, username, phone,
is_active)
    VALUES (@firstname, @lastname, @username, @phone, @is_active);
    set @user_id = @@identity
END
```

Procedura activate/deactivate_user (BK)

Procedura activate_user zmienia stan aktywacji użytkownika na 1 (czyli "aktywowany"), a procedura deactivate_user na 0 czyli "nieaktywny") w bazie danych, na podstawie przekazanego ID użytkownika.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[activate_user]
    @user_id NVARCHAR(50)
AS
BEGIN
    UPDATE users
    SET is_active = 1
    WHERE user_id = @user_id;
END
```

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[deactivate_user]
    @user_id NVARCHAR(50)
AS
BEGIN
    UPDATE users
    SET is_active = 0
    WHERE user_id = @user_id;
END
```

Procedura create_webinar (BK)

Procedura create_webinar tworzy nowy webinar w bazie danych, przypisując mu informacje o produkcie, module, grupie, prowadzącym, spotkaniu online (webinarze) oraz ustalając daty i link. Wywołuje procedurę create_product w celu utworzenia nowego produktu (webinaru) i przechwycenia jego ID. Wywołuje procedurę create_module w celu utworzenia nowego modułu i przechwycenia jego ID. Wywołuje procedurę create_group w celu utworzenia nowej grupy i przechwycenia jej ID. Wywołuje procedurę add_module_to_group w celu przypisania modułu do grupy. Wywołuje procedurę create_online_meeting w celu utworzenia spotkania online (webinaru) i przypisania mu odpowiednich danych.

```
USE [u_kmadej]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[create_webinar]
   @product_id INT output,
   @module_id INT output,
   @group_id int output,
   @meeting_id int output,
   @start_time datetime,
   @end_time datetime,
   @webinar_name varchar(50),
   @link varchar(50),
   @lecturer_id INT,
   @price money
AS
BEGIN
   SET NOCOUNT ON;

   exec create_product @product_id output, @webinar_name, @price
   exec create_module @module_id output

   INSERT INTO webinars (product_id, module_id, lecturer_id)
   VALUES (@product_id, @module_id, @lecturer_id)

   exec create_group @group_id output, @lecturer_id, @product_id
   exec add_module_to_group @module_id, @group_id

   exec create_online_meeting @meeting_id output, @link,
@start_time, @end_time, @module_id, @group_id
END
```

Procedura add_participant

```
USE [u_kmadej]
GO
/****** Object:  StoredProcedure [dbo].[add_participant]    Script
Date: 24.01.2024 13:36:49 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_participant]
 (
    @meeting_id int,
    @user_id int
 ) as
    begin
          insert into meeting_participants(
             meeting_id,
             user_id
          )
          values (
             @meeting_id,
             @user_id
          )
    end
```

Procedura add_product_to_order

```
USE [u_kmadej]
GO
/****** Object:  StoredProcedure [dbo].[add_product_to_order]
Script Date: 24.01.2024 13:38:53 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[add_product_to_order] (
  @order_id int,
  @product_id int
  ) as
     begin
        insert into order_items(
           product_id,
           order_id
        )
```

```
        values (
           @product_id,
           @order_id
        )
     end
```

Procedura add_subject_to_studies

```
USE [u_kmadej]
GO
/****** Object:  StoredProcedure [dbo].[add_subject_to_studies]
Script Date: 24.01.2024 13:39:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER procedure [dbo].[add_subject_to_studies] (
   @module_id int,
   @product_id int,
   @master_id int
   ) as
      begin
         declare @group_id int;
         exec create_group @group_id output, @master_id,
@product_id;
         exec add_module_to_group @module_id, @group_id;
         insert into studies_subjects (
            module_id,
            product_id,
            master_id
         )
         values (
            @module_id,
            @product_id,
            @master_id
         )
      end
```

Procedura room_is_available

```
USE [u_kmadej]
```

```
GO
/****** Object:  StoredProcedure [dbo].[room_is_available]
Script Date: 24.01.2024 13:41:26 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[room_is_available] (
   @room_id int,
   @start_date datetime,
   @end_date datetime,
   @is_available bit output
   ) as
   begin
      if exists (
         select *
         from rooms_availability
         where [ID Sali] = @room_id
         and [Początek] <= @start_date
         and [Koniec] >= @end_date
      )
      begin
         set @is_available = 1
      end
      else begin
         set @is_available = 0
      end
   end
```

Triggery

Trigger add_to_group_after_payment (KM)

Trigger sprawdza czy po dokonaniu płatności zamówienie zostało w pełni opłacone. Jeżeli tak, to znajduje grupy przypisane do produktów w zamówieniu i dodaje do nich użytkownika.

```
USE [u_kmadej]
GO


/****** Object:  Trigger [dbo].[add_to_group_after_payment]
Script Date: 24.01.2024 13:34:29 ******/
SET ANSI_NULLS ON
GO


SET QUOTED_IDENTIFIER ON
GO


CREATE trigger [dbo].[add_to_group_after_payment]
on [dbo].[payments] after insert
as begin
   declare @order_id int
   set @order_id = (
      select i.order_id from inserted i
   )


   declare @paid int
   set @paid = (
      select sum(p.value) from payments p where p.order_id =
@order_id group by p.order_id
   )


   declare @order_value int
   set @order_value = (
      select
         sum(p.price)
      from
```

```
         orders o
         join order_items oi on
            o.order_id = oi.order_id
         join products p on
            p.product_id = oi.product_id
      where o.order_id = @order_id
      group by o.order_id
   )

   if @paid >= @order_value begin
      declare @user_id int
      set @user_id = (
         select o.user_id
         from
            inserted i
            join orders o on
               o.order_id = i.order_id
      )
      declare @product_id int


      declare add_to_group_after_payment_c cursor for
         select
            oi.product_id
         from order_items oi
         where oi.order_id = @order_id
      open add_to_group_after_payment_c
      fetch next from add_to_group_after_payment_c into
@product_id
      while @@fetch_status = 0 begin
         declare @group_id int
         exec find_group_to_add_to @group_id output, @product_id


         exec add_to_group @group_id, @user_id


         fetch next from add_to_group_after_payment_c into
@product_id
      end
      close add_to_group_after_payment_c
      deallocate add_to_group_after_payment_c
   end
end
GO
```

```
ALTER TABLE [dbo].[payments] ENABLE TRIGGER
[add_to_group_after_payment]
GO
```

Trigger add_diplom (BK)

Po zakończeniu kursu lub studium generuje dyplom dla uczestnika.

```
CREATE TRIGGER add_diplom
AFTER INSERT ON final_grades
FOR EACH ROW
BEGIN
   DECLARE grade_value_var REAL;

   SELECT NEW.value INTO grade_value_var;

   IF grade_value_var > 2 THEN
       IF NOT EXISTS (
           SELECT 1
           FROM diplomas d
           INNER JOIN courses c ON d.product_id = c.product_id
           WHERE d.user_id = NEW.user_id AND c.module_id =
NEW.module_id
       ) THEN
           INSERT INTO diplomas (user_id, product_id)
           SELECT NEW.user_id, c.product_id
           FROM courses c
           WHERE c.module_id = NEW.module_id;
       END IF;
   END IF;
END;
```

Uprawnienia

Student

```
create role student

grant select on bilocations to student
grant select on current_week_meetings to student
grant select on current_month_meetings to student
grant select on presence to student

grant execute on create_application to student
grant execute on create_cart to student
grant execute on add_item_to_cart to student
grant execute on remove_item_to_cart to student
grant execute on add_employee_review to student
grant execute on create_payment to student
grant execute on create_order_from_cart to student
```

Ćwiczeniowiec/ wykładowca

```
create role teacher

grant select on not_passed_students to teacher
grant select on graduates to teacher
grant select on rooms_availability to teacher
grant select on current_week_meetings to teacher
grant select on current_month_meetings to teacher
grant select on presence to teacher

grant execute on add_grade to teacher
grant execute on change_grade to teacher
grant execute on set_exam_grades to teacher
grant execute on set_final_grades to teacher
grant execute on pass_internship to teacher
grant execute on add_recording to teacher
grant execute on add_resource to teacher
grant execute on set_presence to teacher
grant execute on change_meeting_date to teacher
grant execute on create_meeting_stationary to teacher
grant execute on create_meeting_online to teacher
```

Koordynator

```
create role coordinator

grant select on not_passed_students to coordinator
grant select on graduates to coordinator
grant select on rooms_availability to coordinator
grant select on employees_reviews_all_time to coordinator
grant select on employees_reviews_yearly to coordinator
grant select on future_meetings_participants to coordinator
grant select on presence to coordinator

grant execute on add_syllabus to coordinator
grant execute on create_course to coordinator
grant execute on create_studies to coordinator
grant execute on create_webinar to coordinator
grant execute on create_subject to coordinator
grant execute on create_exam to coordinator
```

Księgowa

```
create role accountant

grant select on debts to accountant
grant select on accepted_applications to accountant

grant select on most_popular_products to accountant
grant select on most_profitable_products to accountant
```

Dyrektor

```
create role director

grant select on most_popular_products to director
grant select on most_profitable_products to director
grant select on graduates to director
grant select on most_failed_major to director
grant select on languages_usage_all_time to director
grant select on languages_usage_yearly to director
grant select on employees_reviews_all_time to director
grant select on employees_reviews_yearly to director
grant select on future_meetings_participants_count to director


grant execute on accept_application to director
grant execute on create_diploma to director
grant execute on create_employee to director
grant execute on change_group_coordinator to director
grant execute on create_role to director
grant execute on add_translator to director
```

Admin

```
create role admin
grant all privileges ON u_kmadej.dbo TO admin
```