DOKUMENTACJA PROJEKTU

*Krystian Madej, Marcin Walendzik, Błażej Kapkowski*

# Spis treści

[**Spis treści 3**](#_bbd0cq17zwvv)

[**Funkcje oraz użytkownicy systemu 7**](#_lxf86jqx8qby)

[**Diagram bazy danych 10**](#_2gixz06igd5f)

[**Opisy tabel 11**](#_knozfjs9238x)

[Tabela applications 11](#_4e653bg3u3ff)

[Tabela carts 12](#_6y0w1s7qp4q2)

[Tabela cart\_items 14](#_jv9yqei0ua1q)

[Tabela cities 15](#_w8w6q1v5c9zz)

[Tabela countries 16](#_q7316swejbd)

[Tabela courses 17](#_rvscah53xnwu)

[Tabela courses\_modules 18](#_1ns9xwezipqu)

[Zawiera połączenia kursu z jego modułami 18](#_x1iuossezfst)

[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](#_kcnwehccuyhc)

[Tabela customers 20](#_7njr11mdpufc)

[Tabela diplomas 22](#_tqp9tqxiyw5z)

[Tabela employees 24](#_yn75svje0zia)

[Tabela employees\_reviews 26](#_4q49m8m1oh6t)

[Tabela exam\_grades 28](#_bfu3bn3dl4dg)

[Tabela exams 31](#_r9to9tp6o46j)

[Tabela final\_grades 32](#_ve1upcizlz02)

[Tabela grades 34](#_2o0rzkj7uag9)

[Tabela group\_members 36](#_804ofs9kvquq)

[Tabela groups 38](#_lz15z9s6gw1c)

[Tabela internships 40](#_1t9vgovx2nix)

[Tabela languages 41](#_amg1s3h5mtyc)

[Tabela meeting\_participants 42](#_64ctj37w7so1)

[Tabela meetings 44](#_aau77zeo19ej)

[record 45](#_eljp60i33hmu)

[Tabela meetings\_online 46](#_s0zir2iyrlyg)

[Tabela meetings\_stationary 48](#_5vhtd3jtmps6)

[Tabela modules 49](#_pz8ybdq87vke)

[Tabela modules\_categories 50](#_bg78xermzkqe)

[Tabela modules\_groups 52](#_jlgz7ykuhgzg)

[Tabela order\_items 54](#_p38hycd0vqj5)

[Tabela orders 56](#_tprhygy7541b)

[Tabela payments 58](#_l2smxaqkf14s)

[Tabela presence 60](#_p8mwq6wxco4j)

[Tabela products 62](#_6xk3crog7edz)

[Tabela recordings 63](#_qyw3qonc8no)

[Tabela resources 64](#_i14wtz8e8un9)

[Tabela roles 65](#_dpxcgwz9cnpi)

[Tabela rooms 66](#_3su7dbx53xev)

[Tabela studies 67](#_67pqr615g4g6)

[Tabela studies\_groups 68](#_vd6adlx0q1l6)

[Tabela studies\_subjects 69](#_hb2tz7q1alwy)

[Tabela subjects 71](#_d77i21okfs5g)

[Tabela syllabuses 72](#_15ue61q2v4wx)

[Tabela theme\_categories 73](#_f0i3i3q4d39a)

[Tabela translators 74](#_2zwkxlie65j2)

[Tabela translators\_languages 76](#_eccmxk5d70mz)

[Tabela users 78](#_d6agx11a3ce)

[Tabela webinars 80](#_w9xmr8ldgkzs)

[**Widoki 82**](#_hsnwz3as2bop)

[Najpopularniejsze kierunki (BK) 82](#_u90j1oong702)

[Najbardziej opłacalne kierunki (BK) 82](#_eovznkwh2ssb)

[Studenci, którzy nie ukończyli kursu/studium (kierunek) (BK) 83](#_nmeh40tv2w5m)

[Absolwenci studium/kurs (BK) 84](#_invp69sqr8i)

[Najmniej zdawalny kierunek (BK) 85](#_61x7v85ggwwx)

[Najpopularniejsze języki (całość) (KM) 85](#_7evdye8ry0wu)

[Najpopularniejsze języki (rocznie) (KM) 86](#_gcvn6p2r92p0)

[Wolne sale (KM) 87](#_4lu7is905zz1)

[Oceny prowadzących (całość) (KM) 89](#_jb4dpy1ld84y)

[Oceny prowadzących (rocznie) (KM) 89](#_63cvkjdeprhv)

[Osoby zapisane na przyszłe wydarzenia (lista) (KM) 90](#_otgx7ihnhlgv)

[Osoby zapisane na przyszłe wydarzenia (liczba) (KM) 91](#_c2eg72srn7jz)

[Raport dłużników (MW) 92](#_vsp2dzyeym9w)

[Zaakceptowane wnioski (MW) 93](#_3hfl6glxoll0)

[Raport bilokacji (MW) 94](#_6rlwu8vphoad)

[Zajęcia w danym tygodniu (MW) 95](#_owcrjf7la4x2)

[Zajęcia w danym miesiącu (MW) 95](#_dsmtg1jrciwc)

[Obecność każdego ucznia (MW) 96](#_aqj1qf23kk1x)

[**Procedury 97**](#_n743nlw37jx0)

[Procedura create\_application (KM) 97](#_6wzeku1gizwx)

[Procedura accept\_application (KM) 97](#_d59l3rujwzfs)

[Procedura create\_cart (KM) 98](#_qj0b38w61wvx)

[Procedura add\_item\_to\_cart (KM) 98](#_vb8i4rfjipgq)

[Procedura remove\_item\_to\_cart (KM) 99](#_25ox7qizr4vi)

[Procedura remove\_cart (KM) 100](#_ikt6r4pkx3gh)

[Procedura add\_city (KM) 100](#_6x2vhcfikeo4)

[Procedura add\_country (KM) 101](#_8lfztdjlt5yj)

[Procedura create\_course (KM) 101](#_m138t8kgquhb)

[Procedura create\_customer (KM) 102](#_kxjrhlf5pvl3)

[Procedura create\_diploma (KM) 103](#_vyw7e7dxaco0)

[Procedura create\_employee (KM) 104](#_lvlzsd7zur9s)

[Procedura add\_employee\_review (KM) 104](#_reosnn3xxy9c)

[Procedura add\_grade (KM) 105](#_7eao4ebx0my4)

[Procedura change\_grade (KM) 106](#_4p51dfn3anb6)

[Procedura create\_exam (KM) 107](#_p0khfl3u1zrt)

[Procedura set\_exam\_grades (KM) 108](#_ra14bd8jleib)

[Procedura set\_final\_grades (KM) 109](#_yuv14g8vah7s)

[Procedura add\_to\_group (KM) 110](#_31l28z403k2v)

[Procedura add\_to\_group\_meetings (KM) 111](#_pxhoe9r58304)

[Procedura remove\_from\_group (KM) 112](#_ctc6dzfhjq1t)

[Procedura change\_group\_cooridinator (KM) 112](#_tgl9xx5xaw8z)

[Procedura create\_group (KM) 113](#_gd60opfqvwu)

[Procedura find\_group\_to\_add\_to(KM) 114](#_b4ppp0mitxh)

[Procedura create\_studies (KM) 114](#_pisakx3o6o6z)

[Procedura create\_studies\_groups (KM) 115](#_z7gd2atuq82q)

[Procedura pass\_internship (MW) 116](#_iwcu3x8gdi2b)

[Procedura add\_language (MW) 117](#_ken8eteje1ht)

[Procedura create\_meeting (MW) 118](#_o8bj1jnih67s)

[Procedura change\_meeting\_date (MW) 119](#_t3ig10g9kvkq)

[Procedura create\_meeting\_stationary (MW) 120](#_br6irmyr1w5o)

[Procedura create\_online\_meeting (MW) 121](#_umg3y8tvm49y)

[Procedura create\_module (MW) 121](#_6d205wurm5m8)

[Procedura add\_module\_category (MW) 122](#_izysqymqj8z2)

[Procedura add\_module\_to\_group (MW) 123](#_k4hvpkuzx8ol)

[Procedura create\_order (MW) 123](#_hah4jkdankmc)

[Procedura create\_order\_from\_cart (MW) 124](#_is6f2u8ogfni)

[Procedura create\_payment (MW) 125](#_4l8cfe7irldd)

[Procedura pay\_order (MW) 126](#_pb1uqb21uus6)

[Procedura create\_presence (MW) 127](#_kde1y0pwaha7)

[Procedura set\_presence (MW) 127](#_su7r11eiofyc)

[Procedura create\_product (MW) 128](#_dlyvrlmb0vc)

[Procedura add\_recording (BK) 128](#_qtdv13kfym29)

[Procedura add\_resource (BK) 130](#_69flzhv5v7u9)

[Procedura create\_role (BK) 130](#_t8rkvht8hej3)

[Procedura add\_room (BK) 131](#_8t7lfxu46f4r)

[Procedura add\_studies (BK) 131](#_j8jl7int1byv)

[Procedura add\_studies\_group (BK) 132](#_9k4pp3utm1pa)

[Procedura add\_studies\_subject (BK) 133](#_vxjouyp8g7h)

[Procedura add\_syllabus (BK) 133](#_41axzn97djmd)

[Procedura create\_subject (BK) 134](#_w56azdiubaz)

[Procedura create\_theme\_category (BK) 135](#_gh57ll9xxo4g)

[Procedura add\_translator (BK) 135](#_ldwlp5kcu83j)

[Procedura add\_translator\_language (BK) 136](#_19l56beg9ccj)

[Procedura create\_user (BK) 136](#_rdgn2uyegui3)

[Procedura activate/deactivate\_user (BK) 137](#_iqo8zeodj1o)

[Procedura create\_webinar (BK) 138](#_4ddz47l6nv1b)

[Procedura add\_participant 139](#_prbfqtkrftq1)

[Procedura add\_product\_to\_order 139](#_qo7ehy9buwdo)

[Procedura add\_subject\_to\_studies 140](#_asu5fsylk9nq)

[Procedura room\_is\_available 140](#_f0exg0nt58xc)

[**Triggery 142**](#_htyrj615jcsi)

[Trigger add\_to\_group\_after\_payment (KM) 142](#_nn6fh3mkxlar)

[Trigger add\_diplom (BK) 144](#_o6z656va3r3r)

[**Uprawnienia 145**](#_u8bxzj4hcwyz)

[Student 145](#_ykkgod2k52po)

[Ćwiczeniowiec/ wykładowca 146](#_ow9mb3ylv90u)

[Koordynator 146](#_hnz0i08yyf90)

[Księgowa 147](#_d8i0kren99gd)

[Dyrektor 147](#_6rxwo8pao8jz)

[Admin 147](#_4qc3dhwq9wn4)

# 

# 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

# 

# 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 |
| --- |