Podstawy Baz Danych

System zarządzania konferencjami

Damian Sobolewski, Tomasz Skrzek

Spis treści

[Wprowadzenie 4](#_Toc535866158)

[Cel 4](#_Toc535866159)

[Aktorzy 4](#_Toc535866160)

[Administrator 4](#_Toc535866161)

[Pracownik firmy 4](#_Toc535866162)

[Klient biznesowy 4](#_Toc535866163)

[Uczestnik 4](#_Toc535866164)

[Schemat bazy 5](#_Toc535866165)

[Tabele 6](#_Toc535866166)

[Conferences 6](#_Toc535866167)

[ConferenceDays 7](#_Toc535866168)

[ConferenceCosts 8](#_Toc535866169)

[Workshops 9](#_Toc535866170)

[ConferenceBooking 10](#_Toc535866171)

[ConferenceDayBooking 11](#_Toc535866172)

[WorkshopBooking 12](#_Toc535866173)

[Payments 13](#_Toc535866174)

[Participants 14](#_Toc535866175)

[DayParticipants 15](#_Toc535866176)

[WorkshopParticipants 16](#_Toc535866177)

[Clients 17](#_Toc535866178)

[Indeksy 18](#_Toc535866179)

[Widoki 19](#_Toc535866180)

[View\_MostPopularWorkshops 19](#_Toc535866181)

[View\_MostPopularConferences 20](#_Toc535866182)

[View\_MostPopularConferencesByStudents 21](#_Toc535866183)

[View\_MostProfitableConference 22](#_Toc535866184)

[View\_MostProfitableWorkshops 23](#_Toc535866185)

[View\_WorkshopsFreePlaces 24](#_Toc535866186)

[View\_ConferenceFreePlaces 25](#_Toc535866187)

[View\_AvailableConferenceDays 26](#_Toc535866188)

[View\_AvailableWorkshops 27](#_Toc535866189)

[View\_ClientsActivity 28](#_Toc535866190)

[View\_MostProfitableClients 29](#_Toc535866191)

[Procedury 30](#_Toc535866192)

[Dodające 30](#_Toc535866193)

[Procedure\_AddConferenceDay 30](#_Toc535866194)

[Procedure\_AddWorkshop 31](#_Toc535866195)

[Procedure\_AddConferenceCost 32](#_Toc535866196)

[Procedure\_AddConferenceBooking 33](#_Toc535866197)

[Procedure\_AddConferenceDayBooking 34](#_Toc535866198)

[Procedure\_AddWorkshopBooking 35](#_Toc535866199)

[Procedure\_AddDayParticipant 36](#_Toc535866200)

[Procedure\_AddWorkshopParticipant 37](#_Toc535866201)

[Procedure\_AddParticipant 38](#_Toc535866202)

[Procedure\_AddClient 39](#_Toc535866203)

[Aktualizujące 40](#_Toc535866204)

[Procedure\_RemoveConference 40](#_Toc535866205)

[Procedure\_UpdateConferenceDetails 41](#_Toc535866206)

[Procedure\_UpdateWorkshopDetails 43](#_Toc535866207)

[Procedure\_CancelConferenceBooking 45](#_Toc535866208)

[Procedure\_CancelConferenceDayBooking 46](#_Toc535866209)

[Procedure\_UpdateWorkshopNumberOfParticipants 47](#_Toc535866210)

[Procedure\_UpdateConferenceDayNumberOfParticipants 48](#_Toc535866211)

[Procedure\_CancelConferenceBookingWithoutPayingAfterSevenDays 49](#_Toc535866212)

[Wyświetlające 50](#_Toc535866213)

[Procedure\_ShowConferenceDaysAmountOfParticipants 50](#_Toc535866214)

[Procedure\_ShowListOfEventsOfConference 51](#_Toc535866215)

[Funkcje 52](#_Toc535866216)

[Function\_FreeDayPlaces 52](#_Toc535866217)

[Function\_FreeWorkshopPlaces 53](#_Toc535866218)

[Function\_BookingFreeStudentPlaces 54](#_Toc535866219)

[Function\_DaysOfConference 55](#_Toc535866220)

[Function\_ConferenceDayParticipants 56](#_Toc535866221)

[Function\_WorkshopsPerConference 57](#_Toc535866222)

[Function\_WorkshopDate 58](#_Toc535866223)

[Function\_BookingDaysCost 59](#_Toc535866224)

[Function\_BookingWorkshopCost 60](#_Toc535866225)

[Function\_TotalBookingCost 61](#_Toc535866226)

[Function\_WorkshopListForParticipant 62](#_Toc535866227)

[Function\_ConferencesDaysListForParticipant 63](#_Toc535866228)

[Function\_ClientsOrdersList 64](#_Toc535866229)

[Triggery 65](#_Toc535866230)

[Trigger\_TooFewFreePlacesForDayBooking 65](#_Toc535866231)

[Trigger\_TooFewFreePlacesForWorkshopBooking 66](#_Toc535866232)

[Trigger\_LessPlacesForDayThanForWorkshop 67](#_Toc535866233)

[Trigger\_NotEnoughBookedPlacesForDay 68](#_Toc535866234)

[Trigger\_NotEnoughBookedPlacesForWorkshop 69](#_Toc535866235)

[Trigger\_TooFewPlacesAfterDecreasingDayCapacity 70](#_Toc535866236)

[Trigger\_TooFewPlacesAfterDecreasingWorkshopCapacity 71](#_Toc535866237)

[Trigger\_BookingDayInDifferentConference 72](#_Toc535866238)

[Trigger\_BookingDayAlreadyExists 73](#_Toc535866239)

[Trigger\_BookingWorkshopInDifferentDay 74](#_Toc535866240)

[Trigger\_ArePriceThresholdsMonotonous 75](#_Toc535866241)

[Generator danych 76](#_Toc535866242)

# Wprowadzenie

## Cel

Tworzona baza danych wspomagania firmy organizującej konferencji. Firma organizuje konferencje, które mogą być jedno- lub kilkudniowe. Klienci powinni móc rejestrować się na konferencje za pomocą systemu www. Klientami mogą być zarówno indywidualne osoby jak i firmy, natomiast uczestnikami konferencji są osoby (firma nie musi podawać od razu przy rejestracji listy uczestników - może zarezerwować odpowiednią ilość miejsc na określone dni oraz na warsztaty, natomiast na 2 tygodnie przed rozpoczęciem musi te dane uzupełnić - a jeśli sama nie uzupełni do tego czasu, to pracownicy dzwonią do firmy i ustalają takie informacje). Każdy uczestnik konferencji otrzymuje identyfikator imienny (+ ew. informacja o firmie na nim). Dla konferencji kilkudniowych, uczestnicy mogą rejestrować się na dowolne z tych dni.

## Aktorzy

### Administrator

Osoba znająca się na bazach danych, potrafiąca obsłużyć błędy oraz w miarę możliwości poprawiająca i rozszerzająca działanie bazy. Ma dostęp do wszystkich procedur i widoków.

### Pracownik firmy

Osoba odpowiadająca za przyjmowanie zamówień od klientów biznesowych, egzekwowanie opłat za zamówione konferencje oraz pomagająca w przypadku problemów z rejestracją.

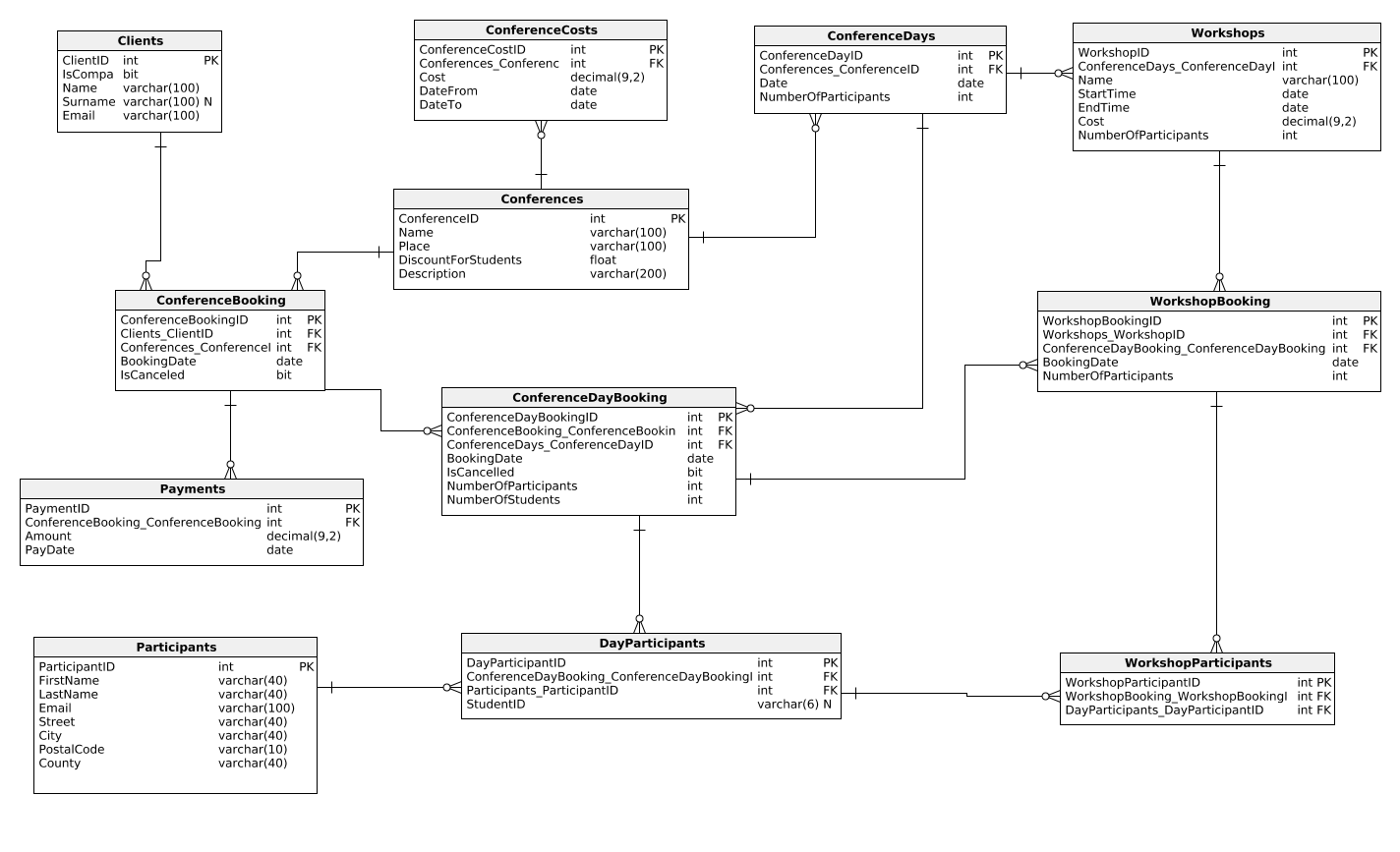
### Klient biznesowy

Osoba składająca zamówienie oraz dodająca uczestników do poszczególnych konferencji i warsztatów za pomocą zewnętrznego serwisu.

### Uczestnik

Osoba uczestnicząca w konferencji oraz wybranych warsztatach. Zostaje ona wprowadzona do bazy za pośrednictwem Klienta Biznesowego. Nie posiada dostępu do żadnych funkcji w systemie.

# Schemat bazy



# Tabele

## Conferences

**CREATE TABLE Conferences (**

**ConferenceID INT NOT NULL IDENTITY,**

**Name VARCHAR(100) NOT NULL,**

**Place VARCHAR(100) NOT NULL,**

**DiscountForStudents FLOAT NOT NULL DEFAULT 0,**

**Description VARCHAR(200) NOT NULL,**

**CONSTRAINT ProperDiscountForStudents CHECK (**

**DiscountForStudents >= 0**

**AND DiscountForStudents <= 100**

**),**

**CONSTRAINT ProperDescription CHECK (LEN(Description) > 5),**

**CONSTRAINT Conferences\_pk PRIMARY KEY (ConferenceID)**

**);**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **ConferenceID** | **INT** | **PK** |
| **Name** | **VARCHAR(100)** |  |
| **Place** | **VARCHAR(100)** |  |
| **DiscountForStudents** | **FLOAT** |  |
| **Description** | **VARCHAR(100)** |  |

## ConferenceDays

**CREATE TABLE ConferenceDays (**

**ConferenceDayID INT NOT NULL IDENTITY,**

**Conferences\_ConferenceID INT NOT NULL,**

**DATE DATE NOT NULL,**

**NumberOfParticipants INT NOT NULL,**

**CONSTRAINT PositiveNumberOfConferenceParticipants CHECK (NumberOfParticipants > 0),**

**CONSTRAINT ConferenceDays\_pk PRIMARY KEY (ConferenceDayID)**

**);**

**-- Reference: Conferences\_ConferenceDays (table: ConferenceDays)**

**ALTER TABLE ConferenceDays ADD CONSTRAINT Conferences\_ConferenceDays FOREIGN KEY (Conferences\_ConferenceID) REFERENCES Conferences (ConferenceID)**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **ConferenceDayID** | **INT** | **PK** |
| **Conferences\_ConferenceID** | **INT** | **FK** |
| **DATE** | **DATE** |  |
| **NumberOfParticipants** | **INT** | **MUST BE POSITIVE** |

## ConferenceCosts

**CREATE TABLE ConferenceCosts (**

**ConferenceCostID INT NOT NULL IDENTITY,**

**Conferences\_ConferenceID INT NOT NULL,**

**Cost DECIMAL(9, 2) NOT NULL,**

**DateFrom DATE NOT NULL,**

**DateTo DATE NOT NULL,**

**CONSTRAINT ProperDayDifferance CHECK (DateFrom <= DateTo),**

**CONSTRAINT NonnegativeCostConf CHECK (Cost >= 0),**

**CONSTRAINT ConferenceCosts\_pk PRIMARY KEY (ConferenceCostID)**

**);**

**-- Reference: Conferences\_ConferenceDay (table: ConferenceCosts)**

**ALTER TABLE ConferenceCosts ADD CONSTRAINT Conferences\_ConferenceDay FOREIGN KEY (Conferences\_ConferenceID) REFERENCES Conferences (ConferenceID)**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **ConferenceCostID** | **INT** | **PK** |
| **Conferences\_ConferenceID** | **INT** | **FK** |
| **Cost** | **DECIMAL(9,2)** | **Cost >= 0** |
| **DateFrom** | **DATE** | **DateFrom <= DateTo** |
| **DateTo** | **DATE** | **DateFrom <= DateTo** |

## Workshops

**CREATE TABLE Workshops (**

**WorkshopID INT NOT NULL IDENTITY,**

**ConferenceDays\_ConferenceDayID INT NOT NULL,**

**Name VARCHAR(100) NOT NULL,**

**StartTime DATE NOT NULL,**

**EndTime DATE NOT NULL,**

**Cost DECIMAL(9, 2) NOT NULL DEFAULT 0,**

**NumberOfParticipants INT NOT NULL DEFAULT 10,**

**CONSTRAINT NonnegativeCostWorkshop CHECK (Cost >= 0),**

**CONSTRAINT ProperTimeDifferance CHECK (StartTime <= EndTime),**

**CONSTRAINT PosiviteWorkshopParticipants CHECK (NumberOfParticipants > 0),**

**CONSTRAINT Workshops\_pk PRIMARY KEY (WorkshopID)**

**);**

**-- Reference: Workshops\_ConferenceDays (table: Workshops)**

**ALTER TABLE Workshops ADD CONSTRAINT Workshops\_ConferenceDays FOREIGN KEY (ConferenceDays\_ConferenceDayID) REFERENCES ConferenceDays (ConferenceDayID)**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **WorkshopID** | **INT** | **PK** |
| **ConferenceDays\_ConferenceID** | **INT** | **FK** |
| **Name** | **VARCHAR(100)** |  |
| **StartTime** | **DATE** | **StartTime <= EndTime** |
| **EndTime** | **DATE** | **StartTime <= EndTime** |
| **Cost** | **DECIMAL(9,2)** | **Cost >= 0** |
| **NumberOfParticipants** | **INT** | **MUST BE POSITIVE** |

## ConferenceBooking

**CREATE TABLE ConferenceBooking (**

**ConferenceBookingID INT NOT NULL IDENTITY,**

**Clients\_ClientID INT NOT NULL,**

**Conferences\_ConferenceID INT NOT NULL,**

**BookingDate DATE NOT NULL DEFAULT getdate(),**

**IsCanceled BIT NOT NULL DEFAULT 0,**

**CONSTRAINT ConferenceBooking\_pk PRIMARY KEY (ConferenceBookingID)**

**);**

**-- Reference: BookingConference\_Clients (table: ConferenceBooking)**

**ALTER TABLE ConferenceBooking ADD CONSTRAINT BookingConference\_Clients FOREIGN KEY (Clients\_ClientID) REFERENCES Clients (ClientID)**

**-- Reference: BookingConference\_Conferences (table: ConferenceBooking)**

**ALTER TABLE ConferenceBooking ADD CONSTRAINT BookingConference\_Conferences FOREIGN KEY (Conferences\_ConferenceID) REFERENCES Conferences (ConferenceID)**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **ConferenceBookingID** | **INT** | **PK** |
| **Clients\_ClientID** | **INT** | **FK** |
| **Conferences\_ConferenceID** | **INT** | **FK** |
| **BookingDate** | **DATE** |  |
| **IsCanceled** | **BIT** |  |

## ConferenceDayBooking

**CREATE TABLE ConferenceDayBooking (**

**ConferenceDayBookingID INT NOT NULL IDENTITY,**

**ConferenceBooking\_ConferenceBookingID INT NOT NULL,**

**ConferenceDays\_ConferenceDayID INT NOT NULL,**

**BookingDate DATE NOT NULL DEFAULT getdate(),**

**IsCancelled BIT NOT NULL DEFAULT 0,**

**NumberOfParticipants INT NOT NULL,**

**NumberOfStudents INT NOT NULL DEFAULT 0,**

**CONSTRAINT PositiveNumberOfParticipants CHECK (NumberOfParticipants > 0),**

**CONSTRAINT NonnegativeNumberOfStudents CHECK (NumberOfStudents >= 0),**

**CONSTRAINT ProperNumberOfStudents CHECK (NumberOfParticipants >= NumberOfStudents),**

**CONSTRAINT ConferenceDayBooking\_pk PRIMARY KEY (ConferenceDayBookingID)**

**);**

**-- Reference: ConferenceDayBooking\_ConferenceBooking (table: ConferenceDayBooking)**

**ALTER TABLE ConferenceDayBooking ADD CONSTRAINT ConferenceDayBooking\_ConferenceBooking FOREIGN KEY (ConferenceBooking\_ConferenceBookingID) REFERENCES ConferenceBooking (ConferenceBookingID)**

**-- Reference: ConferenceDayBooking\_ConferenceDays (table: ConferenceDayBooking)**

**ALTER TABLE ConferenceDayBooking ADD CONSTRAINT ConferenceDayBooking\_ConferenceDays FOREIGN KEY (ConferenceDays\_ConferenceDayID) REFERENCES ConferenceDays (ConferenceDayID)**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **ConferenceDayBookingID** | **INT** | **PK** |
| **ConferenceBooking\_ConferenceBookingID** | **INT** | **FK** |
| **ConferenceDays\_ConferenceDayID** | **INT** | **FK** |
| **BookingDate** | **DATE** |  |
| **IsCanceled** | **BIT** |  |
| **NumberOfParticipants** | **INT** | **POSITIVE AND NO SMALLER THAN NUMBEROFSTUDENTS** |
| **NumberOfStudents** | **INT** | **POSITIVE** |

## WorkshopBooking

**CREATE TABLE WorkshopBooking (**

**WorkshopBookingID INT NOT NULL IDENTITY,**

**Workshops\_WorkshopID INT NOT NULL,**

**ConferenceDayBooking\_ConferenceDayBookingID INT NOT NULL,**

**BookingDate DATE NOT NULL DEFAULT getdate(),**

**NumberOfParticipants INT NOT NULL,**

**CONSTRAINT PositiveWorkshopNumberOfParticipants CHECK (NumberOfParticipants > 0),**

**CONSTRAINT WorkshopBooking\_pk PRIMARY KEY (WorkshopBookingID)**

**);**

**-- Reference: WorkshopBooking\_ConferenceDayBooking (table: WorkshopBooking)**

**ALTER TABLE WorkshopBooking ADD CONSTRAINT WorkshopBooking\_ConferenceDayBooking FOREIGN KEY (ConferenceDayBooking\_ConferenceDayBookingID) REFERENCES ConferenceDayBooking (ConferenceDayBookingID)**

**-- Reference: WorkshopBooking\_Workshops (table: WorkshopBooking)**

**ALTER TABLE WorkshopBooking ADD CONSTRAINT WorkshopBooking\_Workshops FOREIGN KEY (Workshops\_WorkshopID) REFERENCES Workshops (WorkshopID)**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **WorkshopBookingID** | **INT** | **PK** |
| **Workshops\_WorkshopID** | **INT** | **FK** |
| **ConferenceDayBooking\_ConferenceDayBookingID** | **INT** | **FK** |
| **BookingDate** | **DATE** |  |
| **NumberOfParticipants** | **BIT** | **POSITIVE** |

## Payments

**CREATE TABLE Payments (**

**PaymentID INT NOT NULL IDENTITY,**

**ConferenceBooking\_ConferenceBookingID INT NOT NULL,**

**Amount DECIMAL(9, 2) NOT NULL,**

**PayDate DATE NOT NULL DEFAULT getdate(),**

**CONSTRAINT PositiveValue CHECK (Amount > 0),**

**CONSTRAINT Payments\_pk PRIMARY KEY (PaymentID)**

**);**

**-- Reference: Payments\_ConferenceBooking (table: Payments)**

**ALTER TABLE Payments ADD CONSTRAINT Payments\_ConferenceBooking FOREIGN KEY (ConferenceBooking\_ConferenceBookingID) REFERENCES ConferenceBooking (ConferenceBookingID)**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **PaymentID** | **INT** | **PK** |
| **ConferenceBooking\_ConferenceBookingID** | **INT** | **FK** |
| **Amount** | **DECIMAL(9,2)** | **POSITIVE** |
| **PayDate** | **DATE** |  |

## Participants

**CREATE TABLE Participants (**

**ParticipantID INT NOT NULL IDENTITY,**

**FirstName VARCHAR(40) NOT NULL,**

**LastName VARCHAR(40) NOT NULL,**

**Email VARCHAR(100) NOT NULL,**

**Street VARCHAR(40) NOT NULL,**

**City VARCHAR(40) NOT NULL,**

**PostalCode VARCHAR(10) NOT NULL,**

**Country VARCHAR(40) NOT NULL,**

**CONSTRAINT Participants\_pk PRIMARY KEY (ParticipantID)**

**);**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **ParticipantID** | **INT** | **PK** |
| **FirstName** | **VARCHAR(40)** |  |
| **LastName** | **VARCHAR(40)** |  |
| **Email** | **VARCHAR(100)** |  |
| **Street** | **VARCHAR(40)** |  |
| **City** | **VARCHAR(40)** |  |
| **PostalCode** | **VARCHAR(10)** |  |
| **Country** | **VARCHAR(40)** |  |

## DayParticipants

**CREATE TABLE DayParticipants (**

**DayParticipantID INT NOT NULL IDENTITY,**

**ConferenceDayBooking\_ConferenceDayBookingID INT NOT NULL,**

**Participants\_ParticipantID INT NOT NULL,**

**StudentID VARCHAR(6) NULL DEFAULT NULL,**

**CONSTRAINT DayParticipants\_pk PRIMARY KEY (DayParticipantID)**

**);**

**-- Reference: DayParticipants\_ConferenceDayBooking (table: DayParticipants)**

**ALTER TABLE DayParticipants ADD CONSTRAINT DayParticipants\_ConferenceDayBooking FOREIGN KEY (ConferenceDayBooking\_ConferenceDayBookingID) REFERENCES ConferenceDayBooking (ConferenceDayBookingID)**

**-- Reference: Participants\_DayParticipants (table: DayParticipants)**

**ALTER TABLE DayParticipants ADD CONSTRAINT Participants\_DayParticipants FOREIGN KEY (Participants\_ParticipantID) REFERENCES Participants (ParticipantID)**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **DayParticipantID** | **INT** | **PK** |
| **ConferenceDayBooking\_ConferenceDayBookingID** | **INT** | **FK** |
| **Participants\_ParticipantID** | **INT** | **FK** |
| **StudentID** | **VARCHAR(6)** |  |

## WorkshopParticipants

**CREATE TABLE WorkshopParticipants (**

**WorkshopParticipantID INT NOT NULL IDENTITY,**

**WorkshopBooking\_WorkshopBookingID INT NOT NULL,**

**DayParticipants\_DayParticipantID INT NOT NULL,**

**CONSTRAINT WorkshopParticipants\_pk PRIMARY KEY (WorkshopParticipantID)**

**);**

**-- Reference: DayParticipants\_WorkshopParticipants (table: WorkshopParticipants)**

**ALTER TABLE WorkshopParticipants ADD CONSTRAINT DayParticipants\_WorkshopParticipants FOREIGN KEY (DayParticipants\_DayParticipantID) REFERENCES DayParticipants (DayParticipantID)**

**-- Reference: WorkshopParticipants\_WorkshopBooking (table: WorkshopParticipants)**

**ALTER TABLE WorkshopParticipants ADD CONSTRAINT WorkshopParticipants\_WorkshopBooking FOREIGN KEY (WorkshopBooking\_WorkshopBookingID) REFERENCES WorkshopBooking (WorkshopBookingID)**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **WorkshopParticipantID** | **INT** | **PK** |
| **WorkshopBooking\_WorkshopBookingID** | **INT** | **FK** |
| **DayParticipants\_DayParticipantID** | **INT** | **FK** |

## Clients

**CREATE TABLE Clients (**

**ClientID INT NOT NULL IDENTITY,**

**IsCompany BIT NOT NULL,**

**Name VARCHAR(100) NOT NULL,**

**Surname VARCHAR(100) NULL,**

**Email VARCHAR(100) NOT NULL,**

**CONSTRAINT Clients\_pk PRIMARY KEY (ClientID)**

**);**

|  |  |  |
| --- | --- | --- |
| **Nazwa kolumny** | **Typ danych** | **Inne informacje** |
| **ClientID** | **INT** | **PK** |
| **IsCompany** | **BIT** |  |
| **Name** | **VARCHAR(100)** |  |
| **Surname** | **VARCHAR(100)** |  |
| **Email** | **VARCHAR(100)** |  |

# Indeksy

**CREATE INDEX Client ON Clients (Surname ASC)**

**CREATE INDEX Client\_ConferenceBooking ON ConferenceBooking (Clients\_ClientID ASC)**

**CREATE INDEX ConferenceBook ON ConferenceDayBooking (ConferenceDayBookingID ASC)**

**CREATE INDEX ConferenceDay ON ConferenceDayBooking (ConferenceDays\_ConferenceDayID ASC)**

**CREATE INDEX NumberOfParticipantsInConferenceDay ON ConferenceDays (NumberOfParticipants ASC)**

**CREATE INDEX Conference ON ConferenceDays (Conferences\_ConferenceID ASC)**

**CREATE INDEX Participant ON DayParticipants (DayParticipantID ASC)**

**CREATE INDEX ConferenceDayBook ON WorkshopBooking (ConferenceDayBooking\_ConferenceDayBookingID ASC)**

**CREATE INDEX DayParticipant ON WorkshopParticipants (DayParticipants\_DayParticipantID ASC)**

**CREATE INDEX NumberOfParticipantsInWorkshop ON Workshops (NumberOfParticipants ASC)**

**CREATE INDEX ConferenceDayID ON Workshops (ConferenceDays\_ConferenceDayID ASC)**

# Widoki

## View\_MostPopularWorkshops

Wyświetla najpopularniejsze warsztaty

**CREATE VIEW view\_MostPopularWorkshops**

**AS**

**SELECT TOP 10 conf.name AS ConferenceName,**

**w.workshopid,**

**w.name,**

**isnull(sum(wb.NumberOfParticipants), 0) AS popularnosc**

**FROM Workshops AS w**

**INNER JOIN conferencedays AS cd ON cd.conferencedayid = w.ConferenceDays\_ConferenceDayID**

**INNER JOIN conferences AS conf ON conf.conferenceid = cd.Conferences\_ConferenceID**

**INNER JOIN WorkshopBooking AS wb ON wb.Workshops\_WorkshopID = w.WorkshopID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDayBookingID = wb.ConferenceDayBooking\_ConferenceDayBookingID**

**AND cdb.IsCancelled = 0**

**GROUP BY w.workshopid,**

**w.name,**

**conf.name**

**ORDER BY popularnosc DESC**

## View\_MostPopularConferences

Wyświetla najpopularniejsze konferencje

**CREATE VIEW view\_MostPopularConferences**

**AS**

**SELECT TOP 10 c.ConferenceID**

**,c.Name**

**,isnull(sum(cdb.NumberOfParticipants), 0) AS popularity**

**FROM Conferences AS c**

**INNER JOIN ConferenceDays AS cd ON cd.Conferences\_ConferenceID = c.ConferenceID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDays\_ConferenceDayID = cd.ConferenceDayID**

**AND cdb.IsCancelled = 0**

**GROUP BY c.ConferenceID**

**,c.Name**

**ORDER BY popularity DESC**

## View\_MostPopularConferencesByStudents

Wyświetla najpopularniejsze konferencje wśród studentów

**CREATE VIEW view\_MostPopularConferencesByStudents**

**AS**

**SELECT TOP 10 c.ConferenceID**

**,c.Name**

**,isnull(sum(cdb.NumberOfStudents), 0) AS popularity**

**FROM Conferences AS c**

**INNER JOIN ConferenceDays AS cd ON cd.Conferences\_ConferenceID = c.ConferenceID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDays\_ConferenceDayID = cd.ConferenceDayID**

**AND cdb.IsCancelled = 0**

**GROUP BY c.ConferenceID**

**,c.Name**

**ORDER BY popularity DESC**

## View\_MostProfitableConference

Wyświetla najbardziej zyskowne konferencje

**CREATE VIEW view\_MostProfitableConference**

**AS**

**SELECT TOP 10 c.ConferenceID**

**,c.Name**

**,isnull(sum(p.Amount), 0) AS Profit**

**FROM Conferences AS c**

**INNER JOIN ConferenceBooking AS cb ON cb.Conferences\_ConferenceID = c.ConferenceID**

**AND cb.IsCanceled = 0**

**INNER JOIN Payments AS p ON p.ConferenceBooking\_ConferenceBookingID = cb.ConferenceBookingID**

**GROUP BY c.ConferenceID**

**,c.Name**

**ORDER BY Profit DESC**

## View\_MostProfitableWorkshops

Wyświetla najbardziej zyskowne warsztaty

**CREATE VIEW view\_MostProfitableWorkshops**

**AS**

**SELECT TOP 10 c.name AS NazwaKonferencji,**

**w.name,**

**w.cost \* w.NumberOfParticipants AS Profit**

**FROM Workshops AS w**

**INNER JOIN ConferenceDays AS cd ON cd.ConferenceDayID = w.ConferenceDays\_ConferenceDayID**

**INNER JOIN Conferences AS c ON c.ConferenceID = cd.Conferences\_ConferenceID**

**INNER JOIN WorkshopBooking AS wb ON wb.Workshops\_WorkshopID = w.WorkshopID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDayBookingID = wb.ConferenceDayBooking\_ConferenceDayBookingID**

**AND cdb.IsCancelled = 0**

**ORDER BY Profit DESC**

## View\_WorkshopsFreePlaces

Wyświetla informacje o miejscach na poszczególne warsztaty

**CREATE VIEW view\_WorkshopsFreePlaces**

**AS**

**SELECT c.name AS ConferenceName**

**,w.workshopid**

**,w.name**

**,w.numberofparticipants AS Miejsca**

**,isnull(SUM(wb.NumberOfParticipants), 0) AS Zajete**

**,w.numberofparticipants - isnull(SUM(wb.NumberOfParticipants), 0) AS WolneMiejsca**

**FROM workshops AS w**

**INNER JOIN WorkshopBooking AS wb ON wb.workshops\_workshopid = w.workshopid**

**INNER JOIN conferencedays AS cd ON cd.conferencedayid = w.conferencedays\_conferencedayid**

**INNER JOIN conferences AS c ON c.conferenceid = cd.conferences\_conferenceid**

**INNER JOIN ConferenceBooking AS cb ON c.ConferenceID = cb.Conferences\_ConferenceID**

**AND cb.IsCanceled = 0**

**GROUP BY w.workshopid**

**,w.name**

**,w.numberofparticipants**

**,c.name**

## View\_ConferenceFreePlaces

Wyświetla informacje o miejscach na poszczególne konferencje

**CREATE VIEW view\_ConferenceFreePlaces**

**AS**

**SELECT c.ConferenceID**

**,c.name**

**,cd.DATE**

**,isnull(sum(cdb.NumberOfParticipants), 0) AS Zajete**

**,cd.NumberOfParticipants AS Miejsca**

**,cd.NumberOfParticipants - isnull(sum(cdb.NumberOfParticipants), 0) AS WolneMiejsca**

**FROM Conferences AS c**

**INNER JOIN ConferenceDays AS cd ON cd.Conferences\_ConferenceID = c.ConferenceID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDays\_ConferenceDayID = cd.ConferenceDayID**

**AND cdb.IsCancelled = 0**

**GROUP BY c.conferenceid, c.conferencedayid**

**,c.Name**

**,cd.DATE**

**,cd.NumberOfParticipants**

## View\_AvailableConferenceDays

Wyświetla informacje o miejscach na konferencje mające wolne miejsca

**CREATE VIEW view\_AvailableConferenceDays**

**AS**

**SELECT cdp.\***

**FROM VIEW\_ConferenceFreePlaces cdp**

**WHERE WolneMiejsca > 0**

## View\_AvailableWorkshops

Wyświetla informacje o miejscach na warsztaty mające wolne miejsca

**CREATE VIEW view\_AvailableWorkshops**

**AS**

**SELECT w.\***

**FROM view\_WorkshopsFreePlaces w**

**WHERE WolneMiejsca > 0**

## View\_ClientsActivity

Wyświetla informacje o aktywności klientów, ich zamówieniach oraz ich wartości

**CREATE VIEW view\_ClientActivity**

**AS**

**SELECT c.\***

**,isnull((**

**SELECT COUNT(\*)**

**FROM conferencebooking AS cb**

**WHERE cb.clients\_clientid = c.ClientID**

**AND cb.IsCanceled = 0**

**), 0) AS Bookings**

**,isnull((**

**SELECT SUM(p.amount)**

**FROM conferencebooking AS cb**

**INNER JOIN payments AS p ON cb.ConferenceBookingID = p.ConferenceBooking\_ConferenceBookingID**

**WHERE cb.clients\_clientid = c.ClientID**

**AND cb.IsCanceled = 0**

**), 0) AS Zaplacono**

**,isnull((**

**SELECT isnull(sum(cc.Cost \* cd.NumberOfParticipants), 0)**

**FROM ConferenceBooking AS cb**

**INNER JOIN Conferences AS cf ON cf.ConferenceID = cb.Conferences\_ConferenceID**

**INNER JOIN ConferenceCosts AS cc ON cc.Conferences\_ConferenceID = cf.ConferenceID**

**INNER JOIN ConferenceDays AS cd ON cd.Conferences\_ConferenceID = cf.ConferenceID**

**WHERE cb.Clients\_ClientID = c.ClientID**

**AND cb.IsCanceled = 0**

**), 0) AS DoZaplaty**

**FROM Clients AS c**

## View\_MostProfitableClients

Wyświetla informacje o klientach od których firma zarobiła najwiecej

**CREATE VIEW view\_MostProfitableClients**

**AS**

**SELECT TOP 10 c.\***

**FROM view\_ClientsActivity c**

**ORDER BY Payments**

# Procedury

## Dodające

### Procedure\_AddConferenceDay

Procedura dodaje dzień do konferencji.

**CREATE PROCEDURE PROCEDURE\_AddConferenceDay (**

**@ConferenceID INT,**

**@Date DATE,**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**INSERT INTO ConferenceDays (**

**Conferences\_ConferenceID,**

**DATE,**

**NumberOfParticipants**

**)**

**VALUES (**

**@ConferenceID,**

**@Date,**

**@NumberOfParticipants**

**);**

**END**

### Procedure\_AddWorkshop

Procedura dodaje warsztat do konferencji.

**CREATE PROCEDURE PROCEDURE\_AddWorkshop (**

**@ConferenceDayID INT,**

**@Name VARCHAR(100),**

**@StartTime DATE,**

**@EndTime DATE,**

**@Cost DECIMAL(9, 2),**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**INSERT INTO Workshops (**

**ConferenceDays\_ConferenceDayID,**

**Name,**

**StartTime,**

**EndTime,**

**Cost,**

**NumberOfParticipants**

**)**

**VALUES (**

**@ConferenceDayID,**

**@Name,**

**@StartTime,**

**@EndTime,**

**@Cost,**

**@NumberOfParticipants**

**);**

**END**

### Procedure\_AddConferenceCost

Procedura dodaje koszt konferencji na ustalone dni.

**CREATE PROCEDURE PROCEDURE\_AddConferenceCost (**

**@ConferenceID INT,**

**@Cost DECIMAL(9, 2),**

**@DateFrom DATE,**

**@DateTo DATE**

**)**

**AS**

**BEGIN**

**INSERT INTO ConferenceCosts (**

**Conferences\_ConferenceID,**

**Cost,**

**DateFrom,**

**DateTo**

**)**

**VALUES (**

**@ConferenceID,**

**@Cost,**

**@DateFrom,**

**@DateTo**

**);**

**END**

### Procedure\_AddConferenceBooking

Procedura dodaje rezerwację klienta na konferencję.

**CREATE PROCEDURE PROCEDURE\_AddConferenceBooking (**

**@ConferenceID INT,**

**@ClientID INT,**

**@BookingDate DATE**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Conferences**

**WHERE @ConferenceID = ConferenceID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono konferencji', 1**

**END**

**ELSE**

**BEGIN**

**IF @BookingDate IS NULL**

**BEGIN**

**INSERT INTO ConferenceBooking (**

**Conferences\_ConferenceID,**

**Clients\_ClientID**

**)**

**VALUES (**

**@ConferenceID,**

**@ClientID**

**);**

**END**

**ELSE**

**BEGIN**

**INSERT INTO ConferenceBooking (**

**Conferences\_ConferenceID,**

**BookingDate,**

**Clients\_ClientID**

**)**

**VALUES (**

**@ConferenceID,**

**@BookingDate,**

**@ClientID**

**);**

**END**

**END**

**END**

### Procedure\_AddConferenceDayBooking

Procedura dodaje rezerwację klienta na określony dzień konferencji.

**CREATE PROCEDURE PROCEDURE\_AddConferenceDayBooking (**

**@ConferenceDayID INT,**

**@ConferenceBookingID INT,**

**@NumberOfParticipants INT,**

**@NumberOfStudents INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceDays**

**WHERE @ConferenceDayID = ConferenceDayID**

**)**

**OR NOT EXISTS (**

**SELECT \***

**FROM ConferenceBooking**

**WHERE @ConferenceBookingID = ConferenceBookingID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceDayID lub ConferenceBookingID', 1**

**END**

**ELSE**

**BEGIN**

**IF @NumberOfStudents IS NULL**

**BEGIN**

**INSERT INTO ConferenceDayBooking (**

**ConferenceDays\_ConferenceDayID,**

**ConferenceBooking\_ConferenceBookingID,**

**NumberOfParticipants**

**)**

**VALUES (**

**@ConferenceDayID,**

**@ConferenceBookingID,**

**@NumberOfParticipants**

**);**

**END**

**ELSE**

**BEGIN**

**INSERT INTO ConferenceDayBooking (**

**ConferenceDays\_ConferenceDayID,**

**ConferenceBooking\_ConferenceBookingID,**

**NumberOfParticipants,**

**NumberOfStudents**

**)**

**VALUES (**

**@ConferenceDayID,**

**@ConferenceBookingID,**

**@NumberOfParticipants,**

**@NumberOfStudents**

**);**

**END**

**END**

**END**

### Procedure\_AddWorkshopBooking

Procedura dodaje rezerwację klienta na warsztat.

**CREATE PROCEDURE PROCEDURE\_AddWorkshopBooking (**

**@WorkshopID INT,**

**@ConferenceDayBookingID INT,**

**@BookingDate DATE,**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Workshops**

**WHERE @WorkshopID = WorkshopID**

**)**

**OR NOT EXISTS (**

**SELECT \***

**FROM ConferenceDayBooking**

**WHERE @ConferenceDayBookingID = ConferenceDayBookingID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono WorkshopID lub ConferenceDayBookingID', 1**

**END**

**ELSE**

**BEGIN**

**IF @BookingDate IS NULL**

**BEGIN**

**INSERT INTO WorkshopBooking (**

**Workshops\_WorkshopID,**

**ConferenceDayBooking\_ConferenceDayBookingID,**

**NumberOfParticipants**

**)**

**VALUES (**

**@WorkshopID,**

**@ConferenceDayBookingID,**

**@NumberOfParticipants**

**);**

**END**

**ELSE**

**BEGIN**

**INSERT INTO WorkshopBooking (**

**Workshops\_WorkshopID,**

**ConferenceDayBooking\_ConferenceDayBookingID,**

**BookingDate,**

**NumberOfParticipants**

**)**

**VALUES (**

**@WorkshopID,**

**@ConferenceDayBookingID,**

**@BookingDate,**

**@NumberOfParticipants**

**);**

**END**

**END**

**END**

### Procedure\_AddDayParticipant

Procedura dodaje osobę do rezerwacji dnia konferencji.

**CREATE PROCEDURE PROCEDURE\_AddDayParticipant (**

**@ConferenceDayBookingID INT,**

**@ParticipantID INT,**

**@StudentID VARCHAR(6)**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceDayBooking**

**WHERE @ConferenceDayBookingID = ConferenceDayBookingID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceDayBookingID', 1**

**END**

**ELSE**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Participants**

**WHERE @ParticipantID = ParticipantID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ParticipantID', 1**

**END**

**ELSE**

**BEGIN**

**INSERT INTO DayParticipants (**

**ConferenceDayBooking\_ConferenceDayBookingID,**

**Participants\_ParticipantID,**

**StudentID**

**)**

**VALUES (**

**@ConferenceDayBookingID,**

**@ParticipantID,**

**@StudentID**

**);**

**END**

**END**

**END**

### Procedure\_AddWorkshopParticipant

Procedura dodaje osobę do rezerwacji warsztatu.

**CREATE PROCEDURE PROCEDURE\_AddWorkshopParticipant (**

**@WorkshopBookingID INT,**

**@DayParticipantID INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM WorkshopBooking**

**WHERE @WorkshopBookingID = WorkshopBookingID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono WorkshopBookingID', 1**

**END**

**ELSE**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Participants**

**WHERE @DayParticipantID = ParticipantID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ParticipantID', 1**

**END**

**ELSE**

**BEGIN**

**INSERT INTO WorkshopParticipants (**

**WorkshopBooking\_WorkshopBookingID,**

**DayParticipants\_DayParticipantID**

**)**

**VALUES (**

**@WorkshopBookingID,**

**@DayParticipantID**

**);**

**END**

**END**

**END**

### Procedure\_AddParticipant

Procedura dodaje uczestnika.

**CREATE PROCEDURE PROCEDURE\_AddParticipant (**

**@FirstName VARCHAR(40),**

**@LastName VARCHAR(40),**

**@Email VARCHAR(100),**

**@Street VARCHAR(40),**

**@City VARCHAR(40),**

**@PostalCode VARCHAR(10),**

**@Country VARCHAR(40)**

**)**

**AS**

**BEGIN**

**INSERT INTO Participants (**

**FirstName,**

**LastName,**

**Email,**

**Street,**

**City,**

**PostalCode,**

**County**

**)**

**VALUES (**

**@FirstName,**

**@LastName,**

**@Email,**

**@Street,**

**@City,**

**@PostalCode,**

**@Country**

**);**

**END**

### Procedure\_AddClient

Procedura dodaje klienta.

**CREATE PROCEDURE PROCEDURE\_AddClient (**

**@IsCompany BIT,**

**@Name VARCHAR(100),**

**@Surname VARCHAR(100),**

**@Email VARCHAR(100)**

**)**

**AS**

**BEGIN**

**INSERT INTO Clients (**

**IsCompany,**

**Name,**

**Surname,**

**Email**

**)**

**VALUES (**

**@IsCompany,**

**@Name,**

**@Surname,**

**@Email**

**);**

**END**

## Aktualizujące

### Procedure\_RemoveConference

Procedura usuwa konferencję.

**CREATE PROCEDURE PROCEDURE\_RemoveConference (@ConferenceID INT)**

**AS**

**IF NOT EXISTS (**

**SELECT \***

**FROM Conferences**

**WHERE @ConferenceID = ConferenceID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono konferencji', 1**

**END**

**ELSE**

**BEGIN**

**DELETE**

**FROM Conferences**

**WHERE Conferences.ConferenceID = @ConferenceID;**

**END**

### Procedure\_UpdateConferenceDetails

Procedura aktualizuje dane konferencji.

**CREATE PROCEDURE PROCEDURE\_UpdateConferenceDetails (**

**@ConferenceID INT,**

**@Name VARCHAR(100),**

**@Place VARCHAR(100),**

**@DiscountForStudents FLOAT,**

**@Description VARCHAR(200)**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Conferences**

**WHERE @ConferenceID = ConferenceID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono konferencji', 1**

**END**

**ELSE**

**BEGIN**

**IF @Name IS NOT NULL**

**BEGIN**

**UPDATE Conferences**

**SET Name = @Name**

**WHERE ConferenceID = @ConferenceID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'ConferenceID is null', 1**

**END**

**IF @DiscountForStudents IS NOT NULL**

**BEGIN**

**UPDATE Conferences**

**SET DiscountForStudents = @DiscountForStudents**

**WHERE ConferenceID = @ConferenceID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'DiscountForStudents is null', 1**

**END**

**IF @Place IS NOT NULL**

**BEGIN**

**UPDATE Conferences**

**SET Place = @Place**

**WHERE ConferenceID = @ConferenceID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'Place is null', 1**

**END**

**IF @Description IS NOT NULL**

**BEGIN**

**UPDATE Conferences**

**SET Description = @Description**

**WHERE ConferenceID = @ConferenceID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'Description is null', 1**

**END**

**END**

**END**

### Procedure\_UpdateWorkshopDetails

Procedura aktualizuje dane warsztatu.

**CREATE PROCEDURE PROCEDURE\_UpdateWorkshopDetails (**

**@WorkshopID INT,**

**@Name VARCHAR(100),**

**@StartTime DATE,**

**@EndTime DATE,**

**@Cost DECIMAL(9, 2),**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Workshops**

**WHERE @WorkshopID = WorkshopID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono warsztatu', 1**

**END**

**ELSE**

**BEGIN**

**IF @Name IS NOT NULL**

**BEGIN**

**UPDATE Workshops**

**SET Name = @Name**

**WHERE WorkshopID = @WorkshopID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'WorkshopID is null', 1**

**END**

**IF @StartTime IS NOT NULL**

**BEGIN**

**UPDATE Workshops**

**SET StartTime = @StartTime**

**WHERE WorkshopID = @WorkshopID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'StartTime is null', 1**

**END**

**IF @EndTime IS NOT NULL**

**BEGIN**

**UPDATE Workshops**

**SET EndTime = @EndTime**

**WHERE WorkshopID = @WorkshopID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'EndTime is null', 1**

**END**

**IF @Cost IS NOT NULL**

**BEGIN**

**UPDATE Workshops**

**SET Cost = @Cost**

**WHERE WorkshopID = @WorkshopID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'Cost is null', 1**

**END**

**IF @NumberOfParticipants IS NOT NULL**

**BEGIN**

**UPDATE Workshops**

**SET NumberOfParticipants = @NumberOfParticipants**

**WHERE WorkshopID = @WorkshopID;**

**END**

**ELSE**

**BEGIN**

**THROW 50000, 'NumberOfParticipants is null', 1**

**END**

**END**

**END**

### Procedure\_CancelConferenceBooking

Procedura anuluje rezerwację klienta na konferencję.

**CREATE PROCEDURE PROCEDURE\_CancelConferenceBooking (@ConferenceBookingID INT)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceBooking**

**WHERE @ConferenceBookingID = ConferenceBookingID**

**)**

**OR NOT (**

**SELECT IsCanceled**

**FROM ConferenceBooking**

**WHERE @ConferenceBookingID = ConferenceBookingID**

**) = 0**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceBookingID lub rezerwacja została już wcześniej anulowana', 1**

**END**

**ELSE**

**BEGIN**

**UPDATE ConferenceBooking**

**SET IsCanceled = 1**

**WHERE @ConferenceBookingID = ConferenceBookingID;**

**END**

**BEGIN**

**UPDATE ConferenceDayBooking**

**SET IsCancelled = 1**

**WHERE @ConferenceBookingID = ConferenceBooking\_ConferenceBookingID;**

**END**

**END**

### Procedure\_CancelConferenceDayBooking

Procedura anuluje rezerwację klienta na dzień konferencji.

**CREATE PROCEDURE PROCEDURE\_CancelConferenceDayBooking (@ConferenceDayBookingID INT)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceDayBooking**

**WHERE @ConferenceDayBookingID = ConferenceDayBookingID**

**)**

**OR NOT (**

**SELECT IsCancelled**

**FROM ConferenceDayBooking**

**WHERE @ConferenceDayBookingID = ConferenceDayBookingID**

**) = 0**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceDayBookingID lub rezerwacja została już wcześniej anulowana', 1**

**END**

**ELSE**

**BEGIN**

**UPDATE ConferenceDayBooking**

**SET IsCancelled = 1**

**WHERE @ConferenceDayBookingID = ConferenceDayBookingID;**

**END**

**END**

### Procedure\_UpdateWorkshopNumberOfParticipants

Procedura aktualizuje ilość miejsc na warsztacie.

**CREATE PROCEDURE PROCEDURE\_UpdateWorkshopNumberOfParticipants (**

**@WorkshopID INT,**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Workshops**

**WHERE @WorkshopID = WorkshopID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono WorkshopID', 1**

**END**

**ELSE**

**BEGIN**

**UPDATE Workshops**

**SET NumberOfParticipants = @NumberOfParticipants**

**WHERE @WorkshopID = WorkshopID;**

**END**

**END**

### Procedure\_UpdateConferenceDayNumberOfParticipants

Procedura aktualizuje ilość miejsc w dniu konferencji.

**CREATE PROCEDURE PROCEDURE\_UpdateConferenceDayNumberOfParticipants (**

**@ConferenceDayID INT,**

**@NumberOfParticipants INT**

**)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceDays**

**WHERE @ConferenceDayID = ConferenceDayID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceDayID', 1**

**END**

**ELSE**

**BEGIN**

**UPDATE ConferenceDays**

**SET NumberOfParticipants = @NumberOfParticipants**

**WHERE @ConferenceDayID = ConferenceDayID;**

**END**

**END**

### Procedure\_CancelConferenceBookingWithoutPayingAfterSevenDays

Procedura anuluje niedokonane płatności w przeciągu 7 dni.

**CREATE PROCEDURE PROCEDURE\_CancelConferenceBookingWithoutPayingAfterSevenDays**

**AS**

**BEGIN**

**UPDATE ConferenceBooking**

**SET IsCanceled = 1**

**FROM (**

**SELECT \***

**FROM ConferenceBooking**

**LEFT JOIN Payments ON ConferenceBookingID = ConferenceBooking\_ConferenceBookingID**

**WHERE IsCanceled = 0**

**AND PaymentID IS NULL**

**AND DATEDIFF(DAY, BookingDate, getdate()) > 7**

**) AS a**

**WHERE ConferenceBooking.ConferenceBookingID = a.ConferenceBookingID**

**END**

**BEGIN**

**UPDATE ConferenceDayBooking**

**SET IsCancelled = 1**

**FROM (**

**SELECT ConferenceDayBookingID AS ID**

**FROM ConferenceBooking**

**INNER JOIN ConferenceDayBooking ON ConferenceBookingID = ConferenceBooking\_ConferenceBookingID**

**WHERE ConferenceBooking.IsCanceled = 1**

**AND ConferenceDayBooking.IsCancelled = 0**

**) AS b**

**WHERE ConferenceDayBookingID = b.ID**

**END**

### 

## Wyświetlające

### Procedure\_ShowConferenceDaysAmountOfParticipants

Procedura wyświetli ilość uczestników określonego dnia konferencji.

**CREATE PROCEDURE PROCEDURE\_ShowConferenceDaysAmountOfParticipants (@ConferenceID INT)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM Conferences**

**WHERE @ConferenceID = ConferenceID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceID', 1**

**END**

**ELSE**

**BEGIN**

**SELECT ConferenceDayID,**

**DATE,**

**SUM(ConferenceDayBooking.NumberOfParticipants) AS Participants,**

**SUM(NumberOfStudents) AS Students**

**FROM ConferenceDays**

**INNER JOIN ConferenceDayBooking ON Conferences\_ConferenceID = ConferenceDays\_ConferenceDayID**

**WHERE Conferences\_ConferenceID = @ConferenceID**

**AND IsCancelled = 0**

**GROUP BY ConferenceDayID,**

**DATE**

**END**

**END**

### Procedure\_ShowListOfEventsOfConference

Procedura wyświetli ilość wydarzeń określonego dnia konferencji

**CREATE PROCEDURE PROCEDURE\_ShowListOfEventsOfConference (@ConferenceDayID INT)**

**AS**

**BEGIN**

**IF NOT EXISTS (**

**SELECT \***

**FROM ConferenceDays**

**WHERE @ConferenceDayID = ConferenceDayID**

**)**

**BEGIN**

**THROW 50000, 'Nie znaleziono ConferenceDayID', 1**

**END**

**ELSE**

**BEGIN**

**SELECT WorkshopID,**

**Name,**

**StartTime,**

**EndTime,**

**Cost,**

**NumberOfParticipants**

**FROM Workshops**

**WHERE ConferenceDays\_ConferenceDayID = @ConferenceDayID**

**END**

**END**

# Funkcje

## Function\_FreeDayPlaces

Zwraca ilość wolnych miejsc na podany dzień konferencji

**CREATE FUNCTION FUNCTION\_FreeDayPlaces (@ConferenceDayID INT)**

**RETURNS INT**

**AS**

**BEGIN**

**RETURN (**

**SELECT cd.numberofparticipants - isnull(SUM(cdb.NumberOfParticipants), 0)**

**FROM conferencedays AS cd**

**LEFT JOIN ConferenceDayBooking AS cdb ON cd.conferencedayid = cdb.conferencedays\_conferencedayid**

**WHERE cd.conferencedayid = @ConferenceDayID**

**GROUP BY cd.conferencedayid**

**,cd.numberofparticipants**

**);**

**END;**

**GO**

## Function\_FreeWorkshopPlaces

Zwraca ilość wolnych miejsc na podany warsztat

**CREATE FUNCTION FUNCTION\_FreeWorkshopPlaces (@WorkShopID INT)**

**RETURNS INT**

**AS**

**BEGIN**

**RETURN (**

**SELECT w.numberofparticipants - isnull(SUM(wb.NumberOfParticipants), 0)**

**FROM Workshops AS w**

**LEFT JOIN WorkshopBooking AS wb ON wb.Workshops\_WorkshopID = w.WorkshopID**

**WHERE w.WorkshopID = @WorkShopID**

**GROUP BY w.WorkshopID**

**,w.numberofparticipants**

**);**

**END;**

**GO**

## Function\_BookingFreeStudentPlaces

Zwraca ilość wolnych miejsc dla studentów

**CREATE FUNCTION FUNCTION\_BookingFreeStudentPlaces (@ConferenceDayBookingID INT)**

**RETURNS INT**

**AS**

**BEGIN**

**RETURN (**

**SELECT cdb.NumberOfStudents - COUNT(dp.conferencedaybooking\_conferencedaybookingid)**

**FROM ConferenceDayBooking cdb**

**LEFT JOIN dayparticipants dp ON cdb.conferencedaybookingid = dp.ConferenceDayBooking\_ConferenceDayBookingID**

**AND dp.studentid IS NOT NULL**

**WHERE cdb.ConferenceDayBookingID = @ConferenceDayBookingID**

**GROUP BY cdb.ConferenceDayBookingID**

**,cdb.NumberOfStudents**

**);**

**END;**

**GO**

## Function\_DaysOfConference

Dla podanej konferencji zwraca dni, w których dana konferencja się odbywa

**CREATE FUNCTION FUNCTION\_DaysOfConference (@ConferenceID INT)**

**RETURNS @days TABLE (ConferenceDayID INT)**

**AS**

**BEGIN**

**INSERT INTO @days**

**SELECT cd.ConferenceDayID**

**FROM ConferenceDays AS cd**

**WHERE cd.Conferences\_ConferenceID = @ConferenceID**

**RETURN;**

**END**

## Function\_ConferenceDayParticipants

Dla danego dnia konferencji zwraca tabelę z danymi uczestników określonego dnia tej konferencji

**CREATE FUNCTION FUNCTION\_ConferenceDayParticipants (@ConferenceDayID INT)**

**RETURNS @Participants TABLE (**

**firstname VARCHAR(40)**

**,lastname VARCHAR(40)**

**,email VARCHAR(100)**

**,county VARCHAR(40)**

**,city VARCHAR(40)**

**,street VARCHAR(40)**

**,postalcode VARCHAR(10)**

**)**

**AS**

**BEGIN**

**INSERT INTO @Participants**

**SELECT p.firstname**

**,p.lastname**

**,p.email**

**,p.county**

**,p.city**

**,p.Street**

**,p.PostalCode**

**FROM Participants AS p**

**INNER JOIN DayParticipants AS dp ON dp.Participants\_ParticipantID = p.ParticipantID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDayBookingID = dp.ConferenceDayBooking\_ConferenceDayBookingID**

**WHERE cdb.ConferenceDays\_ConferenceDayID = @ConferenceDayID**

**AND cdb.IsCancelled = 0**

**RETURN**

**END**

**GO**

## Function\_WorkshopsPerConference

Dla danej konferencji zwraca tabelę z informacją o warsztatach odbywających się w ramach tej konferencji

**CREATE FUNCTION FUNCTION\_WorkshopsPerConference (@ConferenceID INT)**

**RETURNS @Workshop TABLE (**

**workshopID INT**

**,name VARCHAR(100)**

**,starttime DATE**

**,endtime DATE**

**,cost DECIMAL(9, 2)**

**,numberofparticipants INT**

**)**

**AS**

**BEGIN**

**INSERT INTO @Workshop**

**SELECT w.workshopid**

**,w.name**

**,w.StartTime**

**,w.EndTime**

**,w.cost**

**,w.NumberOfParticipants**

**FROM Workshops AS w**

**INNER JOIN ConferenceDays AS cd ON cd.ConferenceDayID = w.ConferenceDays\_ConferenceDayID**

**WHERE cd.Conferences\_ConferenceID = @ConferenceID**

**RETURN**

**END**

**GO**

## Function\_WorkshopDate

Dla ID warsztatu zwraca tabelę z informacjami o danym warsztacie

**CREATE FUNCTION FUNCTION\_WorkshopDate (@WorkshopID INT)**

**RETURNS @WShop TABLE (**

**workshopID INT**

**,name VARCHAR(100)**

**,startime DATE**

**,endtime DATE**

**)**

**AS**

**BEGIN**

**INSERT INTO @WShop**

**SELECT workshopid**

**,name**

**,StartTime**

**,EndTime**

**FROM Workshops**

**WHERE WorkshopID = @WorkshopID**

**RETURN**

**END**

**GO**

## Function\_BookingDaysCost

Zwraca wartość sprzedanych miejsc dla konferencji bez warsztatów

**CREATE FUNCTION FUNCTION\_BookingDaysCost (@ConferenceBookID INT)**

**RETURNS MONEY**

**AS**

**BEGIN**

**RETURN (**

**SELECT isnull(SUM((cdb.NumberOfParticipants) \* cc.cost + cdb.NumberOfStudents \* cc.cost \* (1 - c.discountforstudents)), 0)**

**FROM ConferenceDayBooking cdb**

**INNER JOIN ConferenceBooking cb ON cdb.ConferenceBooking\_ConferenceBookingID = cb.ConferenceBookingID**

**INNER JOIN conferences AS c ON cb.conferences\_conferenceid = c.conferenceid**

**INNER JOIN conferencecosts cc ON c.conferenceid = cc.conferences\_conferenceid**

**WHERE cb.ConferenceBookingID = @ConferenceBookID**

**GROUP BY cb.ConferenceBookingID**

**);**

**END;**

## Function\_BookingWorkshopCost

Zwraca wartość sprzedanych miejsc dla warsztatów danej konferencji

**CREATE FUNCTION FUNCTION\_BookingWorkshopCost (@ConferenceBookID INT)**

**RETURNS DECIMAL(9, 2)**

**AS**

**BEGIN**

**RETURN (**

**SELECT isnull(SUM(wb.numberofparticipants \* w.cost), 0)**

**FROM ConferenceBooking cb**

**LEFT JOIN conferencedaybooking cdb ON cb.conferencebookingid = cdb.ConferenceBooking\_ConferenceBookingID**

**LEFT JOIN workshopbooking wb ON cdb.conferencedaybookingid = wb.conferencedaybooking\_conferencedaybookingid**

**LEFT JOIN workshops w ON wb.workshops\_workshopid = w.workshopid**

**WHERE cb.conferencebookingid = @ConferenceBookID**

**GROUP BY cb.conferencebookingid**

**);**

**END;**

**GO**

## Function\_TotalBookingCost

Zwraca wartość sprzedanych miejsc dla danej konferencji włącznie z warsztatami

**CREATE FUNCTION FUNCTION\_TotalBookingCost (@ConferenceBookID INT)**

**RETURNS DECIMAL(9, 2)**

**AS**

**BEGIN**

**RETURN (**

**SELECT dbo.FUNCTION\_BookingDaysCost(bs.conferencebookingid) + dbo.FUNCTION\_BookingWorkshopCost(bs.conferencebookingid)**

**FROM conferencebooking bs**

**WHERE bs.conferencebookingid = @ConferenceBookID**

**);**

**END**

## Function\_WorkshopListForParticipant

Zwraca tabelę z informacją o warsztatach w jakich uczestniczyła podana osoba

**CREATE FUNCTION FUNCTION\_WorkshopListForParticipant (@Participant INT)**

**RETURNS @table TABLE (**

**workshopid INT**

**,Name VARCHAR(100)**

**)**

**AS**

**BEGIN**

**INSERT INTO @table**

**SELECT w.workshopid**

**,w.name**

**FROM participants AS p**

**INNER JOIN dayparticipants AS dp ON p.participantid = dp.participants\_participantid**

**INNER JOIN workshopparticipants AS wp ON dp.dayparticipantid = wp.dayparticipants\_dayparticipantid**

**INNER JOIN WorkshopBooking AS wb ON wp.WorkshopBooking\_WorkshopBookingID = wb.workshopbookingid**

**INNER JOIN workshops AS w ON wb.workshops\_workshopid = w.workshopid**

**WHERE p.participantid = @Participant**

**RETURN**

**END**

**GO**

## Function\_ConferencesDaysListForParticipant

Zwraca tabelę z informacją o konferencjach w jakich uczestniczyła podana osoba

**CREATE FUNCTION FUNCTION\_ConferencesDaysListForParticipant (@Participant INT)**

**RETURNS @table TABLE (**

**name VARCHAR(100)**

**,place VARCHAR(100)**

**,DATE DATE**

**)**

**AS**

**BEGIN**

**INSERT INTO @table**

**SELECT c.name**

**,c.place**

**,cd.DATE**

**FROM participants AS p**

**INNER JOIN dayparticipants AS dp ON p.participantid = dp.participants\_participantid**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDayBookingID = dp.ConferenceDayBooking\_ConferenceDayBookingID**

**INNER JOIN ConferenceDays AS cd ON cd.ConferenceDayID = cdb.ConferenceDays\_ConferenceDayID**

**INNER JOIN Conferences AS c ON c.ConferenceID = cd.Conferences\_ConferenceID**

**WHERE p.participantid = @Participant**

**RETURN**

**END**

**GO**

## Function\_ClientsOrdersList

Zwraca listę zamówień klientów

**CREATE FUNCTION FUNCTION\_ClientsOrdersList (@ClientID INT)**

**RETURNS @table TABLE (**

**conferenceid INT**

**,name VARCHAR(100)**

**,place VARCHAR(100)**

**)**

**AS**

**BEGIN**

**INSERT INTO @table**

**SELECT c.ConferenceID**

**,c.name**

**,c.place**

**FROM Conferences AS c**

**INNER JOIN ConferenceBooking AS cb ON cb.Conferences\_ConferenceID = c.ConferenceID**

**INNER JOIN Clients AS cl ON cl.ClientID = cb.Clients\_ClientID**

**WHERE cl.ClientID = @ClientID**

**RETURN**

**END**

**GO**

# Triggery

## Trigger\_TooFewFreePlacesForDayBooking

Sprawdza, czy jest wystarczająca liczba miejsc w dniu konferencji.

**CREATE TRIGGER TRIGGER\_TooFewFreePlacesForDayBooking ON ConferenceDayBooking**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**WHERE dbo.FUNCTION\_FreeDayPlaces(a.ConferenceDays\_ConferenceDayID) < 0**

**)**

**BEGIN**

**THROW 50000, 'Brak wystarczajacej liczby miejsc w dniu konferencji', 1;**

**END**

**END**

## Trigger\_TooFewFreePlacesForWorkshopBooking

Sprawdza, czy jest wystarczająca liczba miejsc na warsztacie.

**CREATE TRIGGER TRIGGER\_TooFewFreePlacesForWorkshopBooking ON WorkshopBooking**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**WHERE dbo.FUNCTION\_FreeWorkshopPlaces(a.Workshops\_WorkshopID) < 0**

**)**

**BEGIN**

**THROW 50000, 'Brak wystarczajacej liczby miejsc w warsztacie', 1;**

**END**

**END**

## Trigger\_LessPlacesForDayThanForWorkshop

Blokuje rezerwację na warsztat, jeżeli klient zarezerwował mniej miejsc na dzień niż warsztat.

**CREATE TRIGGER TRIGGER\_LessPlacesForDayThanForWorkshop ON ConferenceDayBooking**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**INNER JOIN WorkshopBooking AS b ON b.ConferenceDayBooking\_ConferenceDayBookingID = a.ConferenceDayBookingID**

**WHERE a.NumberOfParticipants < b.NumberOfParticipants**

**)**

**BEGIN**

**THROW 50000,**

**'Klient zarezerwował mniej miejsc na dzień niż na warsztat',**

**1**

**END**

**END**

## Trigger\_NotEnoughBookedPlacesForDay

Blokuje zapis uczestnika na dzień konferencji, jeżeli wszystkie miejsca od klienta są już zajęte.

**CREATE TRIGGER TRIGGER\_NotEnoughBookedPlacesForDay ON DayParticipants**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**WHERE (**

**a.StudentID IS NULL**

**AND dbo.FUNCTION\_FreeDayPlacesForStudents(a.ConferenceDayBooking\_ConferenceDayBookingID) < 0**

**)**

**OR (**

**a.StudentID IS NULL**

**AND dbo.FUNCTION\_FreeDayPlacesForParticipants(a.ConferenceDayBooking\_ConferenceDayBookingID) < 0**

**)**

**)**

**BEGIN**

**THROW 50000, 'Wszystkie miejsca klienta zostały już zarezerwowane', 1;**

**END**

**END**

## Trigger\_NotEnoughBookedPlacesForWorkshop

Blokuje zapis uczestnika na warsztat, jeżeli wszystkie zarezerwowane miejsca są już zajęte.

**CREATE TRIGGER TRIGGER\_NotEnoughBookedPlacesForWorkshop ON WorkshopParticipants**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**WHERE dbo.FUNCTION\_FreeWorkshopPlaces(a.WorkshopBooking\_WorkshopBookingID) < 0**

**)**

**BEGIN**

**THROW 50000, 'Wszystkie zarezerwowane miejsca są już zajęte', 1;**

**END**

**END**

## Trigger\_TooFewPlacesAfterDecreasingDayCapacity

Sprawdza, czy po zmniejszeniu liczby miejsc na dzień konferencji zarezerwowane miejsca mieszczą się w nowym limicie.

**CREATE TRIGGER TRIGGER\_TooFewPlacesAfterDecreasingDayCapacity ON ConferenceDays**

**AFTER UPDATE**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**LEFT JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDays\_ConferenceDayID = a.ConferenceDayID**

**GROUP BY a.ConferenceDayID,**

**a.NumberOfParticipants**

**HAVING a.NumberOfParticipants < SUM(cdb.NumberOfParticipants) + SUM(cdb.NumberOfStudents)**

**)**

**BEGIN**

**THROW 50000, 'Po zmniejszeniu liczby miejsc na dzień konferencji zarezerwowane miejsca nie mieszczą się w nowym limicie', 1;**

**END**

**END**

## Trigger\_TooFewPlacesAfterDecreasingWorkshopCapacity

Sprawdza, czy po zmniejszeniu liczby miejsc na warsztat zarezerwowane miejsca mieszczą się w nowym limicie.

**CREATE TRIGGER TRIGGER\_TooFewPlacesAfterDecreasingWorkshopCapacity ON Workshops**

**AFTER UPDATE**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**LEFT JOIN WorkshopBooking AS wb ON wb.Workshops\_WorkshopID = a.WorkshopID**

**GROUP BY a.WorkshopID,**

**a.NumberOfParticipants**

**HAVING a.NumberOfParticipants < SUM(wb.NumberOfParticipants)**

**)**

**BEGIN**

**THROW 50000, 'Po zmniejszeniu liczby miejsc na warsztat zarezerwowane miejsca nie mieszczą się w nowym limicie', 1;**

**END**

**END**

## Trigger\_BookingDayInDifferentConference

Sprawdza, czy rezerwowany jest dzień z konferencji odpowiadającej rezerwacji na konferencję.

**CREATE TRIGGER TRIGGER\_BookingDayInDifferentConference ON ConferenceDayBooking**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**INNER JOIN ConferenceDays AS cd ON cd.ConferenceDayID = a.ConferenceDays\_ConferenceDayID**

**INNER JOIN Conferences AS c1 ON c1.ConferenceID = cd.Conferences\_ConferenceID**

**INNER JOIN ConferenceBooking AS cb ON cb.ConferenceBookingID = a.ConferenceBooking\_ConferenceBookingID**

**INNER JOIN Conferences AS c2 ON c2.ConferenceID = cb.Conferences\_ConferenceID**

**WHERE c1.ConferenceID != c2.ConferenceID**

**)**

**BEGIN**

**THROW 50000, 'Klient próbuje przepisać do konferencji rezerwację dnia z innej konferencji', 1;**

**END**

**END**

## Trigger\_BookingDayAlreadyExists

Sprawdza, czy rezerwacja danego dnia konferencji już istnieje.

**CREATE TRIGGER TRIGGER\_BookingDayAlreadyExists ON ConferenceDayBooking**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**LEFT JOIN ConferenceDayBooking AS cbd ON a.ConferenceBooking\_ConferenceBookingID = cbd.ConferenceBooking\_ConferenceBookingID**

**AND a.ConferenceDays\_ConferenceDayID = cbd.ConferenceDays\_ConferenceDayID**

**WHERE a.ConferenceBooking\_ConferenceBookingID != cbd.ConferenceBooking\_ConferenceBookingID**

**)**

**BEGIN**

**THROW 50000, 'Rezerwacja danego dnia konferencji już istnieje', 1;**

**END**

**END**

## Trigger\_BookingWorkshopInDifferentDay

Sprawdza, czy rezerwowany jest warsztat z dnia odpowiadającemu rezerwacji.

**CREATE TRIGGER TRIGGER\_BookingWorkshopInDifferentDay ON WorkshopBooking**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**INNER JOIN Workshops AS w ON w.WorkshopID = a.Workshops\_WorkshopID**

**INNER JOIN ConferenceDays AS cd1 ON cd1.ConferenceDayID = w.ConferenceDays\_ConferenceDayID**

**INNER JOIN ConferenceDayBooking AS cdb ON cdb.ConferenceDayBookingID = a.ConferenceDayBooking\_ConferenceDayBookingID**

**INNER JOIN ConferenceDays AS cd2 ON cd2.ConferenceDayID = cdb.ConferenceDays\_ConferenceDayID**

**WHERE cd1.Conferences\_ConferenceID != cd2.Conferences\_ConferenceID**

**)**

**BEGIN**

**THROW 50000, 'Klient próbuje zapisać się do warsztatu z innego dnia niż jego rezerwacja', 1;**

**END**

**END**

## Trigger\_ArePriceThresholdsMonotonous

Sprawdza, czy progi cenowe konferencji są ułożone w porządku rosnącym w stosunku do czasu pozostałego do konferencji.

**CREATE TRIGGER TRIGGER\_ArePriceThresholdsMonotonous ON ConferenceCosts**

**AFTER INSERT**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**DECLARE @Cost DECIMAL(9, 2) = (**

**SELECT a.Cost**

**FROM inserted AS a**

**);**

**IF EXISTS (**

**SELECT \***

**FROM inserted AS a**

**LEFT JOIN ConferenceCosts AS cc ON a.Conferences\_ConferenceID = cc.Conferences\_ConferenceID**

**WHERE (**

**(**

**cc.DateFrom < a.DateFrom**

**AND a.DateFrom < cc.dateto**

**)**

**OR (**

**a.DateFrom < cc.DateFrom**

**AND cc.DateTo < a.DateTo**

**)**

**OR (**

**cc.DateFrom >= a.DateFrom**

**AND a.DateTo >= cc.DateTo**

**)**

**OR (**

**a.DateFrom >= cc.DateFrom**

**AND cc.DateTo >= a.DateTo**

**)**

**)**

**AND cc.ConferenceCostID != a.ConferenceCostID**

**)**

**BEGIN**

**THROW 50000,**

**'Koszt pokrywa się z istniejącymi kosztami',**

**1**

**END**

**ELSE**

**BEGIN**

**DECLARE @PreviousCost DECIMAL(9, 2) = (**

**SELECT TOP 1 a.Cost**

**FROM inserted AS a**

**INNER JOIN ConferenceCosts AS cc ON cc.ConferenceCostID = a.ConferenceCostID**

**WHERE cc.Conferences\_ConferenceID = a.Conferences\_ConferenceID**

**AND cc.DateTo < a.DateFrom**

**ORDER BY cc.DateFrom DESC**

**)**

**DECLARE @NextCost DECIMAL(9, 2) = (**

**SELECT TOP 1 a.Cost**

**FROM inserted AS a**

**INNER JOIN ConferenceCosts AS cc ON cc.ConferenceCostID = a.ConferenceCostID**

**WHERE cc.Conferences\_ConferenceID = a.Conferences\_ConferenceID**

**AND cc.DateFrom > a.DateTo**

**ORDER BY cc.DateFrom**

**)**

**IF (**

**(**

**@PreviousCost IS NOT NULL**

**AND @PreviousCost >= @Cost**

**)**

**OR (**

**@NextCost IS NOT NULL**

**AND @NextCost <= @Cost**

**)**

**)**

**BEGIN**

**THROW 50000, 'Cena nie jest w poprawnej kolejności z poprzednimi (PreviousCost = %, NextCost = %.,**

**@PreviousCost,**

**@NextCost', 1;**

**END**

**END**

**END**

# Generator danych

Dane zostały wygenerowane w programie Redgate SQL Data Generator. Podczas generowania napotkaliśmy jednak duże trudności w związku z działaniem triggerów. Program po napotkaniu jakiegokolwiek błędu przestaje generować dane w obecnie generowanej tabeli i czyści ją. Na czas generowania danych zdecydowaliśmy się wyłączyć wszystkie triggery.

Conferences

* 72 rekordów
* Zniżka studencka jest wartością losową

ConferencesDays

* 180 rekordów (na każdą konferencję średnio 2.5 dni konferencji)
* Data z zakresu od 01.01.2015r. do 31.12.2017r. (3 lata działalności)
* Liczba uczestników jest wartością losową z zakresu 300-400

ConferenceCosts

* 72 rekordów
* Koszt jest liczbą losową z zakresu 100-500

Clients

* 150 rekordów

ConferenceBooking

* 150 rekordów
* Anulowane zostało 5% rezerwacji

ConferenceDayBooking

* 3600 rekordów
* Liczba uczestników jest liczbą losową z zakresu 1-19
* Liczba studentów stanowi połowę uczestników

Participants

* 3000 rekordów

DayParticipants

* 5400 rekordów

Payments

* 150 rekordów

Workshops

* 720 rekordów (średnio 4 warsztaty w jednym dniu)
* Koszt jest liczbą losową z zakresu 100-300

WorkshopBooking

* 1200 rekordów
* Liczba uczestników jest liczbą losową z zakresu 1-10

WorkshopParticipants

* 6000 rekordów

Średni