PODSTAWY BAZ DANYCH

Dokumentacja do projektu „Konferencje”

Autorzy:

Dawid Kulma
Łukasz Stanik

Prowadzący:

dr inż. Robert Marcjan

2017/2018

1. Opis funkcji sytemu

Baza danych obsługuje system informatyczny firmy organizującej konferencje. Konferencje mogą trwać kilka dni. Klientami przedsiębiorstwa są zarówno firmy jak i osoby fizyczne, którzy zgłaszają uczestników konferencji. Uczestnikami mogą być tylko osoby fizyczne. Z konferencją mogą być związane warsztaty na które uczestnicy konferencji mogą się dodatkowo rejestrować. W danej chwili czasu uczestnik może być zarejestrowany maksymalnie na jeden warsztat, co jest sprawdzane przez bazę danych. Warsztaty mogą być płatne i darmowe. Opłaty za rezerwacje uczestnictwa w konferencji są progowe – zależą od terminu rezerwacji, natomiast opłaty za uczestnictwo w warsztatach jest stałe. Jest możliwość wprowadzenia zniżki dla studentów.

1.1 OPIS FUNKCJI Z PUNKTU WIDZENIA UŻYTKOWNIKÓW

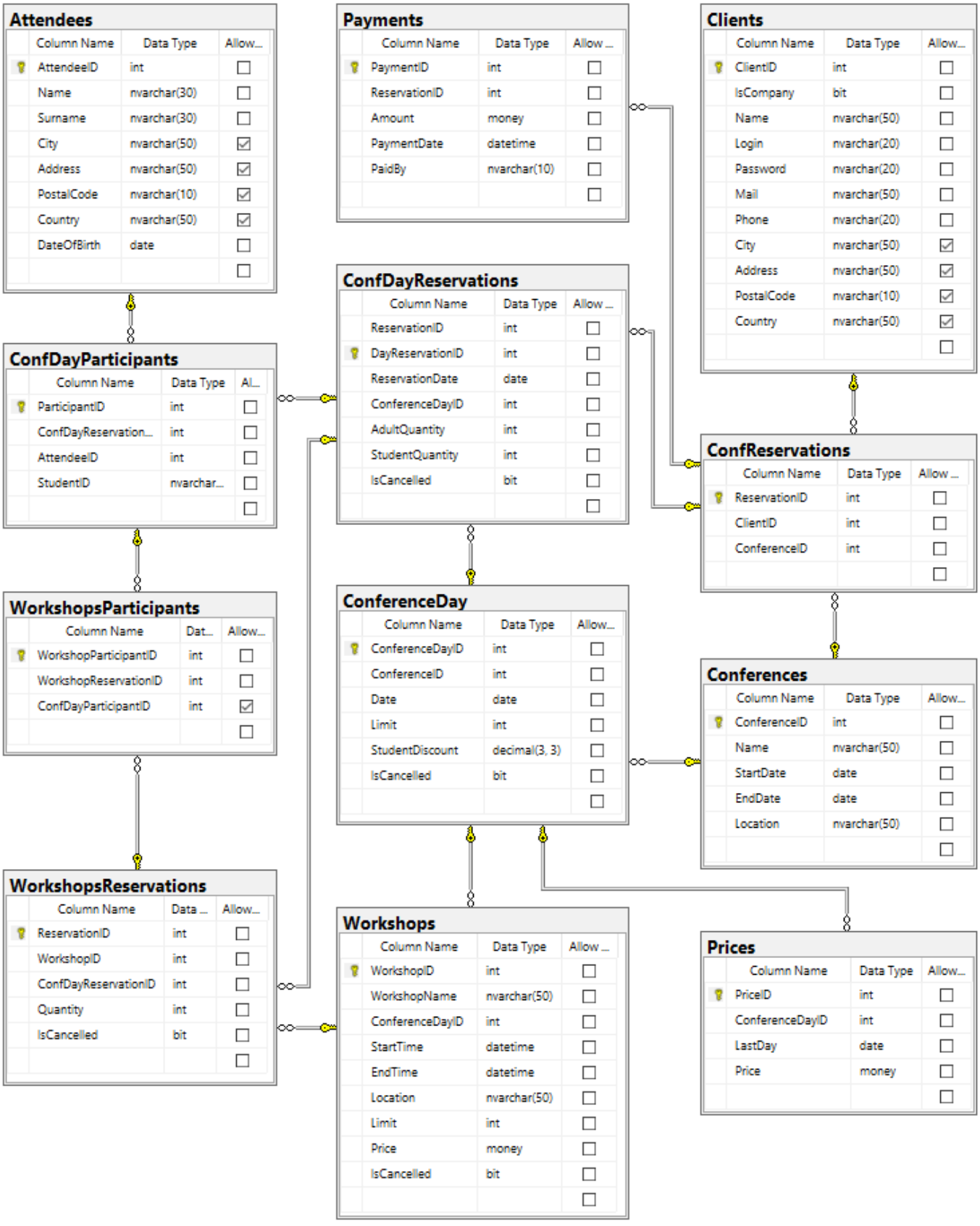
- Klient
  o Rezerwacja pewnej ilości miejsc na dane dni danych konferencji (pod warunkiem że są wolne miejsca)
  o Wyświetlenie wartości należności wobec organizatora konferencji
  o Anulowanie rezerwacji na konferencje
  o Usuwanie swoich uczestników z konferencji
  o Rezerwacja pewnej ilości miejsc na poszczególne warsztaty (pod warunkiem że są wolne miejsca)
  o Rejestrowanie uczestników na warsztaty
  o Anulowanie rezerwacji na warsztaty
  o Usuwanie swoich uczestników z warsztatów

- Uczestnik
  o Rejestrowanie na dane dni konferencji
  o Rezerwacja miejsca na warsztatach (pod warunkiem że uczestnik jest zarejestrowany na dany dzień konferencji, są wolne miejsca i nie jest zapisany na inny warsztat o tej samej godzinie)

- Organizator
  o Tworzenie konferencji
  o Tworzenie warsztatów
  o Wprowadzanie stawek za uczestnictwo w konferencji
  o Wprowadzenie informacji o dokonanej płatności przez klienta
  o Sporządzenie listy nadchodzących konferencji
  o Sporządzenie listy nadchodzących warsztatów
  o Sporządzenie listy identyfikatorów dla uczestników danej konferencji
  o Sporządzenie raportu należności wszystkich klientów firmy
  o Sporządzenie listy klientów, zarejestrowanych na daną konferencję, którzy nie zarejestrowali jeszcze wszystkich zadeklarowanych uczestników

o Sporządzenie listy najpopularniejszych konferencji
o Sporządzenie listy najpopularniejszych warsztatów
o Anulowanie wszystkich nadmiarowych rezerwacji na konferencje, na 2 tygodnie przed rozpoczęciem
o Anulowanie nieopłaconych rezerwacji na warsztaty, po tygodniu od rejestracji

2. Schemat bazy danych



3. Tabele

Attendees

Tabela zawiera informacje o uczestnikach konferencji: imię, nazwisko, miasto, adres, kod pocztowy, państwo, data urodzenia.. Date adresowe są opcjonalne mogą nie zostać podane.

```
CREATE TABLE [dbo].[Attendees](
	[AttendeeID] [int] IDENTITY(1,1) NOT NULL,
	[Name] [nvarchar](30) NOT NULL,
	[Surname] [nvarchar](30) NOT NULL,
	[City] [nvarchar](50) NULL,
	[Address] [nvarchar](50) NULL,
	[PostalCode] [nvarchar](10) NULL,
	[Country] [nvarchar](50) NULL,
	[DateOfBirth] [date] NOT NULL,
 CONSTRAINT [PK_Attendeees] PRIMARY KEY CLUSTERED
(
	[AttendeeID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Clients

Tabela zawiera inforamcje o klientach: nazwę, login, hasło, mail, telefon, miasto, adres, kod pocztowy, kraj zamieszkania. Dane adresowe są opcjonalne I nie musza być podawane.

```
CREATE TABLE [dbo].[Clients](
	[ClientID] [int] IDENTITY(1,1) NOT NULL,
	[IsCompany] [bit] NOT NULL,
	[Name] [nvarchar](50) NOT NULL,
	[Login] [nvarchar](20) NOT NULL,
	[Password] [nvarchar](20) NOT NULL,
	[Mail] [nvarchar](50) NOT NULL,
	[Phone] [nvarchar](20) NOT NULL,
	[City] [nvarchar](50) NULL,
	[Address] [nvarchar](50) NULL,
	[PostalCode] [nvarchar](10) NULL,
	[Country] [nvarchar](50) NULL,
 CONSTRAINT [PK_Clients] PRIMARY KEY CLUSTERED
(
	[ClientID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

ConfDayParticipants

Tabela zawiera informacje o uczestnikach dnia konferencji, wiąże tavbelę Attenddeea z tabelą ConfDayReservation

```
CREATE TABLE [dbo].[ConfDayParticipants](
	[ParticipantID] [int] IDENTITY(1,1) NOT NULL,
	[ConfDayReservationID] [int] NOT NULL,
	[AttendeeID] [int] NOT NULL,
	[StudentID] [nvarchar](10) NOT NULL,
 CONSTRAINT [PK_ConfDayParticipants] PRIMARY KEY CLUSTERED
(
	[ParticipantID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ConfDayParticipants]  WITH CHECK ADD  CONSTRAINT
[FK_ConfDayParticipants_Attendees] FOREIGN KEY([AttendeeID])
REFERENCES [dbo].[Attendees] ([AttendeeID])
GO

ALTER TABLE [dbo].[ConfDayParticipants] CHECK CONSTRAINT
[FK_ConfDayParticipants_Attendees]
GO

ALTER TABLE [dbo].[ConfDayParticipants]  WITH CHECK ADD  CONSTRAINT
[FK_ConfDayParticipants_ConfDayReservations] FOREIGN KEY([ConfDayReservationID])
REFERENCES [dbo].[ConfDayReservations] ([DayReservationID])
GO

ALTER TABLE [dbo].[ConfDayParticipants] CHECK CONSTRAINT
[FK_ConfDayParticipants_ConfDayReservations]
GO
```

ConfDayReservations

Tabela informacje o rezerwacjach na poszczególne dni konferencji. DayReservationID to klucz główny, ReservationID klucz obcy do tabeli ConfReservations, ConferenceDayID to klucz obcy do tabeli ConferenceDay, ReservationDate to data dokonania rezerwacji na dany dzień konferencji, AdultQuantity ilość miejsc dla dorosłych zarezerwowana na dzień konferencji, StudentQuantity to ilość miejsc dla studentów, IsCancelled to wartość bitowa oznaczająca czy dany dzień konferncji został odwołany czy nie.

```
CREATE TABLE [dbo].[ConfDayReservations](
	[ReservationID] [int] NOT NULL,
	[DayReservationID] [int] IDENTITY(1,1) NOT NULL,
	[ReservationDate] [date] NOT NULL,
	[ConferenceDayID] [int] NOT NULL,
	[AdultQuantity] [int] NOT NULL,
	[StudentQuantity] [int] NOT NULL,
	[IsCancelled] [bit] NOT NULL,
 CONSTRAINT [PK_ConfReservations] PRIMARY KEY CLUSTERED
(
	[DayReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ConfDayReservations]  WITH CHECK ADD  CONSTRAINT
[FK_ConfDayReservations_ConferenceDay] FOREIGN KEY([ConferenceDayID])
REFERENCES [dbo].[ConferenceDay] ([ConferenceDayID])
GO

ALTER TABLE [dbo].[ConfDayReservations] CHECK CONSTRAINT
[FK_ConfDayReservations_ConferenceDay]
GO

ALTER TABLE [dbo].[ConfDayReservations]  WITH CHECK ADD  CONSTRAINT
[FK_ConfDayReservations_ConfReservations] FOREIGN KEY([ReservationID])
REFERENCES [dbo].[ConfReservations] ([ReservationID])
GO

ALTER TABLE [dbo].[ConfDayReservations] CHECK CONSTRAINT
[FK_ConfDayReservations_ConfReservations]
GO

ALTER TABLE [dbo].[ConfDayReservations]  WITH CHECK ADD  CONSTRAINT [CHK_Quantity]
CHECK  ((([AdultQuantity]+[StudentQuantity])>(0)))
GO

ALTER TABLE [dbo].[ConfDayReservations] CHECK CONSTRAINT [CHK_Quantity]
GO
```

ConferenceDay

Tabela zawiera informacje na temat dnia konferencji: dacie, limicie miejsc, wysokości zniżki studenckiej i o tym czy dany dzień konferencji został odwołany. ConferenceDayID to klucz główny a ConferenceID to klucz obcy do tabeli Conferences.

```
CREATE TABLE [dbo].[ConferenceDay](
	[ConferenceDayID] [int] IDENTITY(1,1) NOT NULL,
	[ConferenceID] [int] NOT NULL,
	[Date] [date] NOT NULL,
	[Limit] [int] NOT NULL,
	[StudentDiscount] [decimal](3, 3) NOT NULL,
	[IsCancelled] [bit] NOT NULL,
 CONSTRAINT [PK_ConferenceDay] PRIMARY KEY CLUSTERED
(
	[ConferenceDayID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ConferenceDay]  WITH CHECK ADD  CONSTRAINT
[FK_ConferenceDay_Conferences] FOREIGN KEY([ConferenceID])
REFERENCES [dbo].[Conferences] ([ConferenceID])
GO

ALTER TABLE [dbo].[ConferenceDay] CHECK CONSTRAINT [FK_ConferenceDay_Conferences]
GO

ALTER TABLE [dbo].[ConferenceDay]  WITH CHECK ADD  CONSTRAINT [CHK_Limit] CHECK
(([Limit]>(0)))
GO

ALTER TABLE [dbo].[ConferenceDay] CHECK CONSTRAINT [CHK_Limit]
GO

ALTER TABLE [dbo].[ConferenceDay]  WITH CHECK ADD  CONSTRAINT [CHK_StudentDiscount]
CHECK  (([StudentDiscount]>=(0) AND [StudentDiscount]<=(1)))
GO

ALTER TABLE [dbo].[ConferenceDay] CHECK CONSTRAINT [CHK_StudentDiscount]
GO
```

Conferences

Tabela zawiera informacje o konferencjach: nazwie, dacie rozpoczęcia i zakończenia oraz lokalizacji. ConferenceID to klucz główny. Data rozpoczęcia musi być wcześniejsza niż data zakończenia.

```
CREATE TABLE [dbo].[Conferences](
	[ConferenceID] [int] IDENTITY(1,1) NOT NULL,
	[Name] [nvarchar](50) NOT NULL,
	[StartDate] [date] NOT NULL,
	[EndDate] [date] NOT NULL,
	[Location] [nvarchar](50) NOT NULL,
 CONSTRAINT [PK_Conferences] PRIMARY KEY CLUSTERED
(
	[ConferenceID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Conferences]  WITH CHECK ADD  CONSTRAINT [CHK_EndDate] CHECK
(([EndDate]>=[StartDate]))
GO

ALTER TABLE [dbo].[Conferences] CHECK CONSTRAINT [CHK_EndDate]
GO
```

ConfReservation

Tabela wwiążąca: ReservationID to klucz główny, ClientID to klucz obcy do tabeli Clients, a ConferenceID to klucz obcy do tabeli Conferences.

```
CREATE TABLE [dbo].[ConfReservations](
	[ReservationID] [int] IDENTITY(1,1) NOT NULL,
	[ClientID] [int] NOT NULL,
	[ConferenceID] [int] NOT NULL,
 CONSTRAINT [PK_ConfReservations_1] PRIMARY KEY CLUSTERED
(
	[ReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ConfReservations]  WITH CHECK ADD  CONSTRAINT
[FK_ConfReservations_Clients] FOREIGN KEY([ClientID])
REFERENCES [dbo].[Clients] ([ClientID])
GO

ALTER TABLE [dbo].[ConfReservations] CHECK CONSTRAINT [FK_ConfReservations_Clients]
GO

ALTER TABLE [dbo].[ConfReservations]  WITH CHECK ADD  CONSTRAINT
[FK_ConfReservations_Conferences] FOREIGN KEY([ConferenceID])
REFERENCES [dbo].[Conferences] ([ConferenceID])
GO

ALTER TABLE [dbo].[ConfReservations] CHECK CONSTRAINT
[FK_ConfReservations_Conferences]
GO
```

Payments

Tabela zawiera informacje o płatnościach dokonanych prze klientów: ReservationID określa, której rezerwacji dotyczy płatność, Amount oznacza kwotę transakcji, PaymentDate oznacza datę dokonania wpłaty a pole PaidBy określa rodzaj płatności.

```
CREATE TABLE [dbo].[Payments](
	[PaymentID] [int] IDENTITY(1,1) NOT NULL,
	[ReservationID] [int] NOT NULL,
	[Amount] [money] NOT NULL,
	[PaymentDate] [datetime] NOT NULL,
	[PaidBy] [nvarchar](10) NOT NULL,
 CONSTRAINT [PK_Payments] PRIMARY KEY CLUSTERED
(
	[PaymentID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Payments]  WITH CHECK ADD  CONSTRAINT
[FK_Payments_ConfReservations] FOREIGN KEY([ReservationID])
REFERENCES [dbo].[ConfReservations] ([ReservationID])
GO

ALTER TABLE [dbo].[Payments] CHECK CONSTRAINT [FK_Payments_ConfReservations]
GO

ALTER TABLE [dbo].[Payments]  WITH CHECK ADD  CONSTRAINT [CHK_Amount] CHECK
(([Amount]>(0)))
GO

ALTER TABLE [dbo].[Payments] CHECK CONSTRAINT [CHK_Amount]
GO
```

Prices

Tabela zawiera informacje o progach cenowych na dni konferencji: LastDay oznacza ostatni dzień obowiązywania ceny Price. PriceID to klucz główny, a CnferenceDayID to klucz obcy do tabeli ConferenceDay.

```
CREATE TABLE [dbo].[Prices](
	[PriceID] [int] IDENTITY(1,1) NOT NULL,
	[ConferenceDayID] [int] NOT NULL,
	[LastDay] [date] NOT NULL,
	[Price] [money] NOT NULL,
 CONSTRAINT [PK_Prices] PRIMARY KEY CLUSTERED
(
	[PriceID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Prices]  WITH CHECK ADD  CONSTRAINT [FK_Prices_ConferenceDay]
FOREIGN KEY([ConferenceDayID])
REFERENCES [dbo].[ConferenceDay] ([ConferenceDayID])
GO

ALTER TABLE [dbo].[Prices] CHECK CONSTRAINT [FK_Prices_ConferenceDay]
GO

ALTER TABLE [dbo].[Prices]  WITH CHECK ADD  CONSTRAINT [CHK_Price] CHECK
(([Price]>(0)))
GO

ALTER TABLE [dbo].[Prices] CHECK CONSTRAINT [CHK_Price]
GO
```

Workshops

Tabela zawiera informacje na teamt warsztatów: nazwie, czasie rozpoczęcia i zakończenia, lokalizacji, limicie miejsc, cenie i o tym czy zostały odwołane. WorkshopID to klucz główny, ConferenceDayID to klucz obcy do tabeli Conference Day

```
CREATE TABLE [dbo].[Workshops](
	[WorkshopID] [int] IDENTITY(1,1) NOT NULL,
	[WorkshopName] [nvarchar](50) NOT NULL,
	[ConferenceDayID] [int] NOT NULL,
	[StartTime] [datetime] NOT NULL,
	[EndTime] [datetime] NOT NULL,
	[Location] [nvarchar](50) NOT NULL,
	[Limit] [int] NOT NULL,
	[Price] [money] NOT NULL,
	[IsCancelled] [bit] NOT NULL,
 CONSTRAINT [PK_Workshops] PRIMARY KEY CLUSTERED
(
	[WorkshopID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT [FK_Workshops_ConferenceDay]
FOREIGN KEY([ConferenceDayID])
REFERENCES [dbo].[ConferenceDay] ([ConferenceDayID])
GO

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [FK_Workshops_ConferenceDay]
GO

ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT [CHK_WorkshopLimit] CHECK
(([Limit]>(0)))
GO

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [CHK_WorkshopLimit]
GO

ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT [CHK_WorkshopPrice] CHECK
(([Price]>=(0)))
GO

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [CHK_WorkshopPrice]
GO

ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT [CHK_WorkshopTime] CHECK
(([EndTime]>[StartTime]))
GO

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [CHK_WorkshopTime]
GO
```

WorkshopParticipants

Tabela zawiera informacje o uczestnikach warsztatów: WorkshopParticipantID to klucz główny. WorkshopReservationID to klucz obcy to tabeli WorkshopReservation a ConfDayParticipantID to klucz obcy do tabeli ConfDayParticipants

```
CREATE TABLE [dbo].[WorkshopsParticipants](
	[WorkshopParticipantID] [int] IDENTITY(1,1) NOT NULL,
	[WorkshopReservationID] [int] NOT NULL,
	[ConfDayParticipantID] [int] NULL,
 CONSTRAINT [PK_WorkshopsParticipants] PRIMARY KEY CLUSTERED
(
	[WorkshopParticipantID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[WorkshopsParticipants]  WITH CHECK ADD  CONSTRAINT
[FK_WorkshopsParticipants_ConfDayParticipants] FOREIGN KEY([ConfDayParticipantID])
REFERENCES [dbo].[ConfDayParticipants] ([ParticipantID])
GO

ALTER TABLE [dbo].[WorkshopsParticipants] CHECK CONSTRAINT
[FK_WorkshopsParticipants_ConfDayParticipants]
GO

ALTER TABLE [dbo].[WorkshopsParticipants]  WITH CHECK ADD  CONSTRAINT
[FK_WorkshopsParticipants_WorkshopsReservations] FOREIGN KEY([WorkshopReservationID])
REFERENCES [dbo].[WorkshopsReservations] ([ReservationID])
GO

ALTER TABLE [dbo].[WorkshopsParticipants] CHECK CONSTRAINT
[FK_WorkshopsParticipants_WorkshopsReservations]
GO
```

WorkShopReservation

Tabela zawiera informacje o rezerwacji warsztatów: ReservationID to klucz główny, WorkshopID określa których warsztatów dotyczy rezerwacja, ConfDayReservationID określa z którym dniem konferencji są związane warsztaty. Quantity oznacza ilość osób na rezerwację, IsCancelled określa czy dana rezerwacja została anulowana.

```
CREATE TABLE [dbo].[WorkshopsReservations](
	[ReservationID] [int] IDENTITY(1,1) NOT NULL,
	[WorkshopID] [int] NOT NULL,
	[ConfDayReservationID] [int] NOT NULL,
	[Quantity] [int] NOT NULL,
	[IsCancelled] [bit] NOT NULL,
 CONSTRAINT [PK_WorkshopsReservations] PRIMARY KEY CLUSTERED
(
	[ReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[WorkshopsReservations]  WITH CHECK ADD  CONSTRAINT
[FK_WorkshopsReservations_ConfDayReservations] FOREIGN KEY([ConfDayReservationID])
REFERENCES [dbo].[ConfDayReservations] ([DayReservationID])
GO

ALTER TABLE [dbo].[WorkshopsReservations] CHECK CONSTRAINT
[FK_WorkshopsReservations_ConfDayReservations]
GO

ALTER TABLE [dbo].[WorkshopsReservations]  WITH CHECK ADD  CONSTRAINT
[FK_WorkshopsReservations_Workshops] FOREIGN KEY([WorkshopID])
REFERENCES [dbo].[Workshops] ([WorkshopID])
GO

ALTER TABLE [dbo].[WorkshopsReservations] CHECK CONSTRAINT
[FK_WorkshopsReservations_Workshops]
GO

ALTER TABLE [dbo].[WorkshopsReservations]  WITH CHECK ADD  CONSTRAINT
[CHK_WorkshopQuantity] CHECK  (([Quantity]>(0)))
GO

ALTER TABLE [dbo].[WorkshopsReservations] CHECK CONSTRAINT [CHK_WorkshopQuantity]
GO
```

4. Triggery

AddParticipantToConferenceDayTrigger

Blokuje dodanie uczestnika dnia konferencji, jeśli zgłoszono już zadeklarowaną ilość uczestników (z podziałem na dorosłych i studentów)

```
CREATE TRIGGER [dbo].[AddParticipantToConferenceDayTrigger]
   ON  [dbo].[ConfDayParticipants]
   AFTER INSERT
AS
BEGIN
	SET NOCOUNT ON;

	DECLARE @ConfDayReservationID AS int = (
		SELECT ConfDayReservationID
		FROM inserted)

	DECLARE @StudentID AS int = (SELECT StudentID FROM inserted)

	IF (@StudentID = '')
	BEGIN
		DECLARE @MyAdultLimit AS int = (
			SELECT AdultQuantity
			FROM ConfDayReservations
			WHERE DayReservationID = @ConfDayReservationID)

		DECLARE @AdultsAlreadyDeclared AS int = (
			SELECT COUNT(ParticipantID)
			FROM ConfDayParticipants
			WHERE ConfDayReservationID = @ConfDayReservationID
				AND StudentID = '')

		IF (@AdultsAlreadyDeclared=@MyAdultLimit)
		BEGIN
			RAISERROR ('ERROR - You have already added declared number of
			adult participants',-1,-1)
			ROLLBACK TRANSACTION
		END
	END

	IF (@StudentID <> '')
	BEGIN
		DECLARE @MyStudentLimit AS int = (
			SELECT StudentQuantity
			FROM ConfDayReservations
			WHERE DayReservationID = @ConfDayReservationID)

		DECLARE @StudentsAlreadyDeclared AS int = (
			SELECT COUNT(ParticipantID)
			FROM ConfDayParticipants
			WHERE ConfDayReservationID = @ConfDayReservationID
				AND StudentID <> '')

		IF (@StudentsAlreadyDeclared=@MyStudentLimit)
		BEGIN
			RAISERROR ('ERROR - You have already added declared number of
			students',-1,-1)
			ROLLBACK TRANSACTION
		END
	END

END
```

AddNewConfDayReservationTrigger

Blokuje dodanie rezerwacji na dany dzień, jeśli nie ma odpowiednio dużo wolnych miejsc na niego lub jeśli istnieje już rezerwacja danego dnia dla podanej rezerwacji konferencji.

```
CREATE TRIGGER [dbo].[AddNewConfDayReservationTrigger]
   ON  [dbo].[ConfDayReservations]
   AFTER INSERT,UPDATE
AS
BEGIN
	SET NOCOUNT ON;

	DECLARE @ConferenceDayID AS int = (SELECT ConferenceDayID FROM inserted)

	DECLARE @Quantity AS int = (SELECT AdultQuantity+StudentQuantity FROM inserted)

	IF (@Quantity>dbo.FreePlacesLeftForDay(@ConferenceDayID))
	BEGIN
		RAISERROR('ERROR - There is no enough places for this conference day',-
		1,-1)
		ROLLBACK TRANSACTION
	END

	DECLARE @ReservationID AS int =(
		SELECT ReservationID
		FROM inserted)

	DECLARE @ExistingDayReservation AS INT = (
		SELECT DayReservationID
		FROM dbo.ConfDayReservations
		WHERE ReservationID=@ReservationID
			AND ConferenceDayID=@ConferenceDayID)

	IF (@ExistingDayReservation IS NOT NULL)
	BEGIN
		RAISERROR('ERROR - You have already done reservation for this day. Now
		you can only update that reservation.',-1,-1)
		ROLLBACK TRANSACTION
	END

END
```

CancelConfDayReservationTrigger

Anuluje wszystkie rezerwacje na warsztaty powiązane z anulowaną rezerwacją na dzień.

```
CREATE TRIGGER [dbo].[CancelConfDayReservationTrigger] ON [dbo].[ConfDayReservations]
FOR UPDATE AS
BEGIN
	IF UPDATE(IsCancelled)
	BEGIN
		IF ((SELECT IsCancelled FROM inserted) = 1)
		BEGIN
			DECLARE @ConfDayReservationID AS int = (
				SELECT DayReservationID
				FROM inserted)

			UPDATE WorkshopsReservations SET IsCancelled = 1 WHERE
			ConfDayReservationID = @ConfDayReservationID
		END
	END
END
```

CancelConferenceDayTrigger

Anuluje wszystkie rezerwacje na odwoływany dzień konferencji.

```
CREATE TRIGGER [dbo].[CancelConferenceDayTrigger]
   ON [dbo].[ConferenceDay]
   AFTER UPDATE
AS
BEGIN
	-- SET NOCOUNT ON added to prevent extra result sets from
	-- interfering with SELECT statements.
	SET NOCOUNT ON;

	DECLARE @ConferenceDayID AS int = (SELECT ConferenceDayID FROM inserted)
	UPDATE ConfDayReservations SET IsCancelled = 1 WHERE ConferenceDayID =
	@ConferenceDayID

END
```

AddNewWorkshopTrigger

Blokuje dodanie warsztatu, który miałby więcej miejsc niż jest to przewidziane na dany dzień konferencji.

```
CREATE TRIGGER [dbo].[AddNewWorkshopTrigger]
   ON  [dbo].[Workshops]
   AFTER INSERT
AS
BEGIN
       SET NOCOUNT ON;

    DECLARE @WorkshopLimit AS int = (
              SELECT Limit
              FROM inserted)

       DECLARE @ConferenceDayID AS int = (
              SELECT ConferenceDayID
              FROM inserted)

       DECLARE @DayLimit AS int = (
              SELECT Limit
              FROM ConferenceDay
              WHERE ConferenceDayID = @ConferenceDayID)

       IF (@WorkshopLimit<@DayLimit)

       BEGIN
              RAISERROR('ERROR - You tried to add a workshop that have more places
              than conference day limit is',-1,-1)
              ROLLBACK TRANSACTION
       END

       DECLARE @ConferenceDay AS date = (
              SELECT Date
              FROM ConferenceDay
              WHERE ConferenceDayID = @ConferenceDayID)

       DECLARE @WorkshopDay AS date = (
              SELECT CONVERT(DATE,StartTime)
              FROM inserted)

       IF (@ConferenceDay <> @WorkshopDay)
       BEGIN
              RAISERROR('ERROR - The conference day and workshop day do not match',
              -1,-1)
              ROLLBACK TRANSACTION
       END

END
```

CancelWorkshopTrigger

Anuluje wszystkie rezerwacje na odwoływany warsztat.

```
CREATE TRIGGER [dbo].[CancelWorkshopTrigger]
   ON [dbo].[Workshops]
   AFTER UPDATE
AS
BEGIN

	SET NOCOUNT ON;

	DECLARE @WorkshopID AS int = (SELECT WorkshopID FROM inserted)
	UPDATE WorkshopsReservations SET IsCancelled = 1 WHERE WorkshopID = @WorkshopID

END
```

AddParticipantToWorkshopTrigger

Blokuje dodanie uczestnika warsztatu, jeśli zgłoszono już zadeklarowaną ilość uczestników.

```
CREATE TRIGGER [dbo].[AddParticipantToWorkshopTrigger]
   ON  [dbo].[WorkshopsParticipants]
   AFTER INSERT
AS
BEGIN
	SET NOCOUNT ON;

	DECLARE @WorkshopReservationID AS int = (
		SELECT WorkshopReservationID
		FROM inserted)

	DECLARE @MyLimit AS int = (
		SELECT Quantity
		FROM WorkshopsReservations
		WHERE ReservationID=@WorkshopReservationID)

	DECLARE @AlreadyDeclared AS int = (
		SELECT COUNT(WorkshopParticipantID)
		FROM WorkshopsParticipants
		WHERE WorkshopReservationID=@WorkshopReservationID)

	IF (@AlreadyDeclared=@MyLimit)
	BEGIN
		RAISERROR('ERROR - You have already added declared number of this
		workshop participants',-1,-1)
		ROLLBACK TRANSACTION
	END


END
```

AddNewWorkshopReservationTrigger

Blokuje dodanie rezerwacji na warsztat jeśli nie ma dostatecznie dużo wolnych miejsc na niego, lub jeżeli deklarowana ilość jest większa niż zarezerwowana ilość miejsc na dzień konferencji, w którym odbywa się warsztat.

```
CREATE TRIGGER [dbo].[AddNewWorkshopReservationTrigger]
   ON  [dbo].[WorkshopsReservations]
   AFTER INSERT,UPDATE
AS
BEGIN

	SET NOCOUNT ON;

	DECLARE @WorkshopID AS int = (SELECT WorkshopID FROM inserted)

	DECLARE @Quantity AS int = (SELECT Quantity FROM inserted)

	IF (@Quantity>dbo.FreePlacesLeftForWorkshop(@WorkshopID))
	BEGIN
		RAISERROR('ERROR - There is no enough places for this workshop',-1,-1)
		ROLLBACK TRANSACTION
	END

	DECLARE @ConfDayReservationID AS int = (SELECT ConfDayReservationID FROM
	inserted)

	DECLARE @DayResQuantity AS int = (
		SELECT (AdultQuantity+StudentQuantity)
		FROM ConfDayReservations
		WHERE DayReservationID = @ConfDayReservationID)

	IF (@Quantity>@DayResQuantity)
	BEGIN
		RAISERROR('ERROR - There is no enough places on your day reservation',-
		1,-1)
		ROLLBACK TRANSACTION
	END

END
```

AddNewConferenceDayTrigger

Blokuje zadeklarowanie nowego dnia konferencji, jeżeli podana data nie występuje podczas trwania konferencji lub jeżeli dany dzień został już wcześniej zadeklarowany.

```
CREATE TRIGGER AddNewConferenceDayTrigger
   ON  ConferenceDay
   AFTER INSERT
AS
BEGIN

	SET NOCOUNT ON;

	DECLARE @ConferenceID AS int = (
		SELECT ConferenceID
		FROM inserted)

	DECLARE @Date AS date = (
		SELECT Date
		FROM inserted)

	DECLARE @StartDate AS DATE
	DECLARE @EndDate AS DATE
	SET @StartDate = (SELECT StartDate FROM Conferences WHERE ConferenceID =
	@ConferenceID)
	SET @EndDate = (SELECT EndDate FROM Conferences WHERE ConferenceID =
	@ConferenceID)

	IF (@Date < @StartDate OR @Date > @EndDate)
	BEGIN
		RAISERROR ('ERROR - The day is not during this conference',-1,-1)
		ROLLBACK TRANSACTION
	END


	DECLARE @ExistingDay DATE = (
		SELECT Date
		FROM ConferenceDay
		WHERE ConferenceID = @ConferenceID AND Date = @Date
		GROUP BY Date)
	IF (@ExistingDay IS NOT NULL)
	BEGIN
		RAISERROR('ERROR - This day has been already declared',-1,-1)
		ROLLBACK TRANSACTION
	END


END
```

AddNewPriceTrigger

Blokuje zadeklarowanie nowego progu cenowego jeżeli ostatni dzień obowiązywania progu nie mieści się w zakresie od daty wprowadzenia do dnia konferencji albo jeżeli na dany *ostatni dzień* został już wyznaczony inny próg cenowy.

```
CREATE TRIGGER AddNewPriceTrigger
   ON  Prices
   AFTER INSERT
AS
BEGIN

	SET NOCOUNT ON;

	DECLARE @ConferenceDayID AS int = (
		SELECT ConferenceDayID
		FROM inserted)

	DECLARE @ExistingDay AS date = (
		SELECT Date
		FROM dbo.ConferenceDay
		WHERE ConferenceDayID = @ConferenceDayID)
	IF (@ExistingDay IS NULL)
	BEGIN
		;THROW 52000,'ERROR - There is no such ConferenceDayID',1
	END

	DECLARE @LastDay AS date = (
		SELECT LastDay
		FROM inserted)

	IF (@LastDay > @ExistingDay OR @LastDay < CONVERT(DATE,GETDATE()))
	BEGIN
		;THROW 52000,'ERROR - The last day shoud be between today and conference
day',1
	END

	DECLARE @ExistingPrice AS INT = (
		SELECT PriceID
		FROM dbo.Prices
		WHERE LastDay=@LastDay)
	IF (@ExistingDay IS NOT NULL)
	BEGIN
		;THROW 52000,'ERROR - There is price already declared this day',1
	END

END
```

5. Indeksy

Przy tworzeniu indeksów przyjęto założenia, aby indeksować wartości często wyszukiwane oraz te, do których odwołują się np. procedury lub funkcje.

ConfDayParticipants_ConfDayReservationID_Index

```
CREATE NONCLUSTERED INDEX [ConfDayParticipants_ConfDayReservationID_Index] ON [dbo].
[ConfDayParticipants]
(
	[ConfDayReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
```

ConfDayReservations_ReservationID_Index

```
CREATE NONCLUSTERED INDEX [ConfDayReservations_ReservationID_Index] ON [dbo].
[ConfDayReservations]
(
	[ReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
```

ConferenceDay_ConferenceID_Index

```
CREATE NONCLUSTERED INDEX [ConferenceDay_ConferenceID_Index] ON
[dbo].[ConferenceDay]
(
	[ConferenceID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
```

ConfReservations_ClientID_Index

```
CREATE NONCLUSTERED INDEX [ConfReservations_ClientID_Index] ON [dbo].
[ConfReservations]
(
	[ClientID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
```

Payments_ReservationID_Index

```
CREATE NONCLUSTERED INDEX [Payments_ReservationID_Index] ON [dbo].[Payments]
(
	[ReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
```

Prices_ConferenceDayID_Index

```
CREATE NONCLUSTERED INDEX [Prices_ConferenceDayID_Index] ON [dbo].[Prices]
(
	[ConferenceDayID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
```

Workshops_ConferenceDayID_Index

```
CREATE NONCLUSTERED INDEX [Workshops_ConferenceDayID_Index] ON [dbo].[Workshops]
(
	[ConferenceDayID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
```

WorkshopsParticipants_WorkshopReservationID_Index

```
CREATE NONCLUSTERED INDEX [WorkshopsParticipants_WorkshopReservationID_Index] ON
[dbo].[WorkshopsParticipants]
(
	[WorkshopReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
```

WorkshopsReservation_WorkshopID_Index

```
CREATE NONCLUSTERED INDEX [WorkshopsReservation_WorkshopID_Index] ON [dbo].
[WorkshopsReservations]
(
	[WorkshopID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
```

WorkshopsReservations_ConfDayReservationID_Index

```
CREATE NONCLUSTERED INDEX [WorkshopsReservations_ConfDayReservationID_Index] ON [dbo].
[WorkshopsReservations]
(
	[ConfDayReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
```

6. Widoki

Customer Stats

Widok zwraca ilość konferencji zarezerwowanych przez każdego klienta

```
CREATE VIEW [dbo].[CustomersStats] AS

SELECT c.ClientID,
       c.Name,
       COUNT(*) as NumberOfReservations
FROM Clients AS c INNER JOIN ConfReservations AS cr
ON c.ClientID = cr.ClientID

GROUP BY c.ClientID, c.Name
GO
```

PopularConferences

Widok porządkuje konferencję po ID i nazwie podliczając liczbe chetnych zgłoszonych na odpowiednią konferencję

```
CREATE VIEW [dbo].[PopularConferences]
AS

       SELECT c.ConferenceID,
              c.Name,
              SUM(cdr.AdultQuantity+cdr.StudentQuantity) AS [Reserved Places]
       FROM ConferenceDay AS cd
       INNER JOIN Conferences as c
                  ON cd.ConferenceID = c.ConferenceID
       INNER JOIN ConfDayReservations AS cdr
                  ON cd.ConferenceDayID = cdr.ConferenceDayID
       WHERE cdr.IsCancelled = 0
       GROUP BY c.ConferenceID, c.Name
GO
```

PopularWorkshops

Widok porządkuje warsztaty po ID i nazwie podliczając liczbe chetnych zgłoszonych na odpowiedni warsztat.

```
CREATE VIEW [dbo].[PopularWorkshops]
AS
       SELECT w.WorkshopID,
              w.WorkshopName,
              SUM(wr.Quantity) AS [Reserved Places]
       FROM Workshops AS w
       INNER JOIN WorkshopsReservations AS wr
                  ON w.WorkshopID = wr.WorkShopID
       WHERE wr.IsCancelled = 0
       GROUP BY w.WorkshopID, w.WorkshopName
GO
```

Widoki jako procedury z parametrem

ShowConfDayParticipants

Prezentuje listę osobową na dany dzień konferencji

```
CREATE PROCEDURE [dbo].[ShowConfDayParticipants]
	@ConfDayId INT
AS
BEGIN
	SET NOCOUNT ON;
	SELECT a.Name, a.Surname
	FROM ConferenceDay as cd
	INNER JOIN  ConfDayReservations AS cdr
		    ON cd.ConferenceDayID = cdr.ConferenceDayID
	INNER JOIN ConfDayParticipants AS cdp
		    ON cdr.DayReservationID = cdp.ConfDayReservationID
	INNER JOIN Attendees AS a
		    ON cdp.AttendeeID = a.AttendeeID
	WHERE cd.ConferenceDayID = @ConfDayId AND cdr.IsCancelled = 0
END
GO
```

ShowWorkshopParticipants

Prezentuje listę osobowa na dany warsztat

```
CREATE PROCEDURE [dbo].[ShowWorkshopParticipants]
	@WorkshopId INT
AS
BEGIN
	SET NOCOUNT ON;
	SELECT a.Name, a.Surname
	FROM Workshops as w
	INNER JOIN  WorkshopsReservations AS wr
		    ON w.WorkshopID = wr.WorkshopID
	INNER JOIN WorkshopsParticipants AS wp
		   ON wr.ReservationID = wp.WorkshopReservationID
	INNER JOIN ConfDayParticipants AS cdp
		   ON wp.ConfDayParticipantID = cdp.ParticipantID
	INNER JOIN Attendees AS a
		   ON cdp.AttendeeID = a.AttendeeID
	WHERE w.WorkshopID = @WorkshopId AND wr.IsCancelled = 0
END
GO
```

ShowWorkshopsDuringConference

Prezentuje warsztaty podczas danej konferencji

```
CREATE PROCEDURE [dbo].[ShowWorkshopsDuringConference]
(
	@ConferenceId INT
)
AS
BEGIN
	SELECT *
		FROM Workshops
		INNER JOIN ConferenceDay
			ON Workshops.ConferenceDayID = ConferenceDay.ConferenceDayID
		INNER JOIN Conferences
			ON Conferences.ConferenceID = ConferenceDay.ConferenceID
		WHERE Conferences.ConferenceID = @ConferenceId;
END
GO
```

ShowWorkshopsDuringConferenceDay

Wyświetla warsztaty podczas danego dnia konferencji

```
CREATE PROCEDURE [dbo].[ShowWorkshopsDuringConferenceDay]
(
	@ConferenceDayId INT
)
AS
BEGIN
	SELECT *
		FROM Workshops
		INNER JOIN ConferenceDay
			ON Workshops.ConferenceDayID = ConferenceDay.ConferenceDayID
		WHERE ConferenceDay.ConferenceDayID = @ConferenceDayId;

END
GO
```

ShowPaymentStatusForConference

Widok prezentujący na jakim etapie, jeśli chodzi o opłacanie rezerwacji, znajduje się każda rezerwacja na podaną konferencję. Jako parametr przyjmuje on ID konferencji, dla której chcemy wygenerować raport statusowy.

```
CREATE PROCEDURE [dbo].[ShowPaymentStatusForConference]
	@ConferenceID int
AS
BEGIN

	SET NOCOUNT ON;

	SELECT ReservationID, ClientID, Name, ISNULL(SUM(Amount),0) AS Paid,
	dbo.ConferencePrice(ReservationID) AS ToPay, ToPay-Paid AS StillToPay
	FROM ConfReservation AS CR
	INNER JOIN Payments AS P ON P.ReservationID = CR.ReservationID
	WHERE ConferenceID = @ConferenceID

END
GO
```

7. Procedury modyfikujące

AddNewAttendee

Procedura dodająca nowego uczestnika. Jako argumenty przyjmuje dane uczestnika.

```
CREATE PROCEDURE [dbo].[AddNewAttendee]

	@Name VARCHAR(30),
	@Surname VARCHAR(30),
	@City VARCHAR(50),
	@Address VARCHAR(50),
	@PostalCode VARCHAR(10),
	@Country VARCHAR(50),
	@DateOfBirth DATE
	AS
BEGIN

	SET NOCOUNT ON;

	INSERT INTO dbo.Attendees
	(
	    Name,
	    Surname,
	    City,
	    Address,
	    PostalCode,
	    Country,
	    DateOfBirth
	)
	VALUES
	(   @Name,
	    @Surname,
	    @City,
	    @Address,
```

```
            @PostalCode,
            @Country,
            @DateOfBirth
            )

    END
```

AddNewClient

Procedura dodająca nowego klienta. Jako argumenty przyjmuje dane klienta.

```
ALTER PROCEDURE [dbo].[AddNewClient]

        @IsCompany BIT,
        @Name NVARCHAR(50),
        @Login NVARCHAR(20),
        @Password NVARCHAR(20),
        @Mail NVARCHAR(50),
        @Phone NVARCHAR(50),
        @City NVARCHAR(50),
        @Address NVARCHAR(50),
        @PostalCode NVARCHAR(10),
        @Country NVARCHAR(50)

        AS
BEGIN

        SET NOCOUNT ON;

        INSERT INTO dbo.Clients
        (
            IsCompany,
            Name,
            Login,
            Password,
            Mail,
            Phone,
            City,
            Address,
            PostalCode,
            Country
        )
        VALUES
        (   @IsCompany,
            @Name,
            @Login,
            @Password,
            @Mail,
            @Phone,
            @City,
            @Address,
            @PostalCode,
            @Country
            )

    END
```

AddNewConference

Procedura dodająca nową konferencję. Jako parametry przyjmuje nazwę konferencji, daty – rozpoczęcia i zakończenia – oraz lokalizację.

```
ALTER PROCEDURE [dbo].[AddNewConference]

	@Name NVARCHAR(50),
	@StartDate DATE,
	@EndDate DATE,
	@Location NVARCHAR(50)
AS
BEGIN
	SET NOCOUNT ON;

	IF (@StartDate > @EndDate)
	BEGIN
		RAISERROR('StartDate must be earlier than EndDate',-1,-1)
	END

	INSERT INTO Conferences (Name, StartDate, EndDate, Location)
	VALUES (@Name, @StartDate, @EndDate, @Location)
END
```

AddNewConferenceDay

Procedura dodająca nowy dzień danej konferencji. Jako parametry przyjmuje ID konferencji, datę, limit miejsc i wysokość zniżki studenckiej. Wykona się tylko wtedy, gdy podana data rzeczywiście jest podczas danej konferencji.

```
ALTER PROCEDURE [dbo].[AddNewConferenceDay]

	@ConferenceID INT,
	@Date DATE,
	@Limit INT,
	@StudentDiscount DECIMAL(3,3)
AS
BEGIN

	SET NOCOUNT ON;

	IF (@Limit <= 0)
	BEGIN
		THROW 52000,'ERROR - Limit must be greater than 0',1
	END

	DECLARE @ExistingConf INT = (
		SELECT ConferenceID
		FROM Conferences
		WHERE ConferenceID=@ConferenceID)
	IF (@ExistingConf IS NULL)
	BEGIN
		THROW 52000,'ERROR - There is no such conference',1
	END

	DECLARE @StartDate AS DATE
	DECLARE @EndDate AS DATE
	SET @StartDate = (SELECT StartDate FROM Conferences WHERE ConferenceID =
	@ConferenceID)
```

```
	SET @EndDate = (SELECT EndDate FROM Conferences WHERE ConferenceID =
	@ConferenceID)

	IF (@Date < @StartDate OR @Date > @EndDate)
	BEGIN
		THROW 52000,'ERROR - The day is not during this conference',1
	END


	DECLARE @ExistingDay DATE = (
		SELECT Date
		FROM ConferenceDay
		WHERE ConferenceID = @ConferenceID AND Date = @Date
		GROUP BY Date)
	IF (@ExistingDay IS NOT NULL)
	BEGIN
		;THROW 52000,'ERROR - This day has been already declared',1
	END

	IF (@StudentDiscount IS NULL OR @StudentDiscount < 0 OR @StudentDiscount > 1)
	BEGIN
		;THROW 52000,'ERROR - Student discount must be a decimal value from
		[0,1] scope',1
	END

	BEGIN
		INSERT INTO ConferenceDay (ConferenceID, Date, Limit, StudentDiscount,
		IsCancelled)
		VALUES (@ConferenceID, @Date, @Limit, @StudentDiscount, 0)
	END

END
```

AddNewDayReservation

Procedura dodająca rezerwację na dany dzień konferencji. Jako parametry przyjmuje ID rezerwacji, interesującą nas datę oraz ilości jakie chcemy zarezerwować – osobno dorosłych oraz studentów.

```
ALTER PROCEDURE [dbo].[AddNewDayReservation]

	@ReservationID INT,
	@Date DATE,
	@AdultQuantity INT,
	@StudentQuantity INT

AS
BEGIN

	SET NOCOUNT ON;

		IF(@ReservationID IS NULL)
	BEGIN
		;THROW 52000,'ERROR - You have to make a conference reservation first',1
	END

	DECLARE @ExistingReservation INT = (
		SELECT ReservationID
		FROM ConfReservations
		WHERE ReservationID=@ReservationID)
```

```
	IF(@ExistingReservation IS NULL)
	BEGIN
		;THROW 52000,'ERROR - Invalid ReservationID',1
	END

	IF (@AdultQuantity + @StudentQuantity <=0)
	BEGIN
		;THROW 52000,'ERROR - You have to reserve at least 1 place',1
	END

	DECLARE @ConferenceDayID AS INT = (
		SELECT ConferenceDayID
		FROM ConferenceDay
		INNER JOIN dbo.Conferences ON Conferences.ConferenceID =
		ConferenceDay.ConferenceID
		INNER JOIN dbo.ConfReservations ON ConfReservations.ConferenceID =
		Conferences.ConferenceID
		WHERE ConfReservations.ReservationID=@ReservationID
			AND Date = @Date)

	IF(@ConferenceDayID IS NULL)
	BEGIN
		;THROW 52000,'ERROR - There is no such conference day during the
		conference on this reservation',1
	END

	DECLARE @ExistingDayReservation AS INT = (
		SELECT DayReservationID
		FROM dbo.ConfDayReservations
		WHERE ReservationID=@ReservationID
			AND ConferenceDayID=@ConferenceDayID)

	IF (@ExistingDayReservation IS NOT NULL)
	BEGIN
		;THROW 52000,'ERROR - You have already done reservation for this day.
		Now you can only update that reservation.',1
	END

	DECLARE @FreePlaces AS INT = dbo.FreePlacesLeftForDay(@ConferenceDayID)
	IF (@AdultQuantity+@StudentQuantity>@FreePlaces)
	BEGIN
		;THROW 52000,'ERROR - Sorry, there is no enough places for the day',1
	END

	INSERT INTO dbo.ConfDayReservations
	(
	    ReservationID,
	    ReservationDate,
	    ConferenceDayID,
	    AdultQuantity,
	    StudentQuantity,
	    IsCancelled
	)
	VALUES
	(   @ReservationID,         -- ReservationID - int
	    GETDATE(), -- ReservationDate - date
	    @ConferenceDayID,         -- ConferenceDayID - int
	    @AdultQuantity,         -- AdultQuantity - int
	    @StudentQuantity,         -- StudentQuantity - int
	    0       -- IsCancelled - bit
	    )
END
```

AddNewPayment

Procedura dodająca nową płatność. Jako parametry przyjmuje ID rezerwacji, kwotę wpłaty oraz rodzaj płatności.

```
ALTER PROCEDURE [dbo].[AddNewPayment]

	@ReservationID int,
	@Amount money,
	@PaidBy nvarchar(10)
	AS
BEGIN

	SET NOCOUNT ON;

	IF (@Amount = 0)
	BEGIN
		;THROW 52000,'ERROR - Amount must be greater than 0',1
	END

	INSERT INTO Payments (ReservationID, Amount, PaymentDate, PaidBy)
	VALUES (@ReservationID, @Amount, GETDATE(), @PaidBy)

  END
```

AddNewPrice

Procedura dodająca nowy próg cenowy. Jako parametry przyjmuje ID dnia konferencji, ostatni dzień obowiązywania nowego progu cenowego oraz jego wysokość.

```
ALTER PROCEDURE [dbo].[AddNewPrice]
	@ConferenceDayID INT,
	@LastDay DATE,
	@Price MONEY
	AS
BEGIN
	SET NOCOUNT ON;

	DECLARE @ExistingDay AS date = (
		SELECT Date
		FROM dbo.ConferenceDay
		WHERE ConferenceDayID = @ConferenceDayID)
	IF (@ExistingDay IS NULL)
	BEGIN
		;THROW 52000,'ERROR - There is no such ConferenceDayID',1
	END

	IF (@LastDay > @ExistingDay OR @LastDay < CONVERT(DATE,GETDATE()))
	BEGIN
		;THROW 52000,'ERROR - The last day shoud be between today and conference
		day',1
	END

	DECLARE @ExistingPrice AS INT = (
		SELECT PriceID
		FROM dbo.Prices
		WHERE LastDay=@LastDay)
	IF (@ExistingDay IS NOT NULL)
	BEGIN
		;THROW 52000,'ERROR - There is price already declared this day',1
```

```
        END

        INSERT INTO dbo.Prices
        (
            ConferenceDayID,
            LastDay,
            Price
        )
        VALUES
        (   @ConferenceDayID,         -- ConferenceDayID - int
            @LastDay, -- LastDay - date
            @Price       -- Price - money
            )
  END
```

AddNewReservation

Procedura dodająca nową rezerwację. Jako parametry przyjmuje ID klienta oraz interesującej nas konferencji.

```
ALTER PROCEDURE [dbo].[AddNewReservation]

        @ClientID INT,
        @ConferenceID int
AS
BEGIN

        SET NOCOUNT ON;

        DECLARE @ExistingConference AS INT = (
               SELECT ConferenceID
               FROM dbo.Conferences
               WHERE ConferenceID = @ConferenceID)
        IF (@ExistingConference IS NULL)
        BEGIN
               ;THROW 52000,'ERROR - There is no such conference',1
        END

        INSERT INTO dbo.ConfReservations
        (
            ClientID,
               ConferenceID
        )
        VALUES
        (      @ClientID,
               @ConferenceID
            )


END
```

AddNewWorkshop

Procedura dodająca nowy warsztat. Jako parametry przyjmuje nazwę warsztatu, ID konferencji, czasy – rozpoczęcia zakończenia, lokalizację, limit miejsc i opcjonalnie cenę. Wykona się tylko wtedy, gdy podane godziny trwania należą do któregoś z zadeklarowanych dni konferencji.

```
ALTER PROCEDURE [dbo].[AddNewWorkshop]

	@WorkshopName NVARCHAR(50),
	@ConferenceID INT,
	@StartTime DATETIME,
	@EndTime DATETIME,
	@Location NVARCHAR(50),
	@Limit INT,
	@Price MONEY

	AS
BEGIN

	SET NOCOUNT ON;

	IF (@Limit<=0)
	BEGIN
		;THROW 52000,'ERROR - Limit must be greater than 0',1
	END

	IF (@Price IS NULL)
	BEGIN
		SET @Price = 0
	END

	DECLARE @ExistingConf INT = (
		SELECT ConferenceID
		FROM Conferences
		WHERE ConferenceID=@ConferenceID)
	IF (@ExistingConf IS NULL)
	BEGIN
		;THROW 52000,'ERROR - There is no such conference',1
	END

	IF (@StartTime>=@EndTime)
	BEGIN
		;THROW 52000,'ERROR - EndTime must be later than StartTime',1
	END

	IF(CONVERT(DATE,@StartTime)<>CONVERT(DATE,@EndTime))
	BEGIN
		;THROW 52000,'ERROR - You are trying to schedule a workshop that lasts
		more than one day!',1
	END

	DECLARE @ConferenceDayID AS INT = (
		SELECT ConferenceDayID
		FROM dbo.ConferenceDay
		WHERE ConferenceID = @ConferenceID AND CONVERT(DATE,@StartTime) = Date)

	IF (@ConferenceDayID IS NULL)
	BEGIN
		;THROW 52000,'ERROR - Entered time is not during this conference',1
	END
```

```
	DECLARE @ConfDayLimit AS INT = (
		SELECT Limit
		FROM dbo.ConferenceDay
		WHERE ConferenceDayID=@ConferenceDayID)
	IF(@Limit>@ConfDayLimit)
	BEGIN
		;THROW 52000,'WARNING - You are trying to add a workshop with limit
		greater than conference day limit. Aborted.',1
	END

	INSERT INTO dbo.Workshops
	(
	    WorkshopName,
	    ConferenceDayID,
	    StartTime,
	    EndTime,
	    Location,
	    Limit,
	    Price,
	    IsCancelled
	)
	VALUES
	(   @WorkshopName,       -- WorkshopName - nvarchar(50)
	    @ConferenceDayID,         -- ConferenceDayID - int
	    @StartTime, -- StartTime - datetime
	    @EndTime, -- EndTime - datetime
	    @Location,       -- Location - nvarchar(50)
	    @Limit,         -- Limit - int
	    @Price,      -- Price - money
	    0       -- IsCancelled - bit
	    )

	END
```

AddNewWorkshopReservation

Procedura dodająca nową rezerwację miejsc na warsztacie. Jako parametry przyjmuje ID rezerwacji oraz warsztaty, a także ilość miejsc, którą chciałoby się zarezerwować. Wykona się tylko wtedy, gdy podany warsztat odbywa się podczas konferencji, na którą dana rezerwacja została złożona.

```
ALTER PROCEDURE [dbo].[AddNewWorkshopReservation]

	@ReservationID INT,
	@WorkshopID INT,
	@Quantity INT
	AS
BEGIN

	SET NOCOUNT ON;

	IF(@ReservationID IS NULL)
	BEGIN
		;THROW 52000,'ERROR - You have to make a conference reservation first',1
	END

	DECLARE @ExistingReservation INT = (
		SELECT ReservationID
		FROM ConfReservations
		WHERE ReservationID=@ReservationID)
```

```
    IF(@ExistingReservation IS NULL)
    BEGIN
           ;THROW 52000,'ERROR - Invalid ReservationID',1
    END

    DECLARE @ExistingWorkshop INT = (
           SELECT WorkshopID
           FROM dbo.Workshops
           WHERE WorkshopID=@WorkshopID)
    IF(@ExistingWorkshop IS NULL)
    BEGIN
           ;THROW 52000,'ERROR - There is no such workshop',1
    END

    IF (@Quantity <=0)
    BEGIN
           ;THROW 52000,'ERROR - You have to reserve at least 1 place',1
    END

    DECLARE @WorkshopDay AS DATE = (
           SELECT CONVERT(DATE,StartTime)
           FROM dbo.Workshops
           WHERE WorkshopID=@WorkshopID)

    DECLARE @ConfDayReservationID AS INT = (
           SELECT DayReservationID
           FROM dbo.ConfDayReservations
           INNER JOIN dbo.ConferenceDay ON ConferenceDay.ConferenceDayID =
           ConfDayReservations.ConferenceDayID
           WHERE ReservationID=@ReservationID
                  AND dbo.ConferenceDay.Date=@WorkshopDay)

    IF (@ConfDayReservationID IS NULL)
    BEGIN
           ;THROW 52000,'ERROR - you do not have a Day reservation for a day of
           this workshop on the Conference reservation',1
    END

    DECLARE @ConferenceDayID AS INT = (
           SELECT dbo.ConfDayReservations.ConferenceDayID
           FROM dbo.ConfDayReservations
           WHERE DayReservationID=@ConfDayReservationID)

    DECLARE @FreePlaces AS INT = dbo.FreePlacesLeftForWorkshop(@WorkshopID)

    IF(@Quantity>@FreePlaces)
    BEGIN
           ;THROW 52000,'ERROR - Sorry, there is no enough places for this
           workshop',1
    END

    DECLARE @DeclaredForThisDay AS INT = (
           SELECT ISNULL(SUM(AdultQuantity + StudentQuantity),0)
           FROM dbo.ConfDayReservations
           WHERE ConferenceDayID=@ConferenceDayID)

IF (@Quantity>@DeclaredForThisDay)
    BEGIN
           RAISERROR('WARNING - You will reserve for this workshop more places than
           you have reserved for a day of conference',-1,-1)
    END
```

```
	INSERT INTO dbo.WorkshopsReservations
	(
	    WorkshopID,
	    ConfDayReservationID,
	    Quantity,
	    IsCancelled
	)
	VALUES
	(   @WorkshopID,   -- WorkshopID - int
	    @ConfDayReservationID,   -- ConfDayReservationID - int
	    @Quantity,   -- Quantity - int
	    0 -- IsCancelled - bit
	    )

END
```

AddNewParticipantToConferenceDay

Procedura przypisująca nowego uczestnika do danej rezerwacji na dzień konferencji. Jako parametry przyjmuje ID uczestnika, ID rezerwacji na dzień konferencji oraz nr legitymacji studenckiej.

```
ALTER PROCEDURE [dbo].[AddParticipantToConferenceDay]

	@AttendeeID int,
	@ConfDayReservationID int,
	@StudentID nvarchar(10) = ''
	AS
BEGIN

	SET NOCOUNT ON;

	DECLARE @IsCancelled AS int = (
		SELECT IsCancelled
		FROM ConfDayReservations
		WHERE DayReservationID = @ConfDayReservationID)

	IF (@IsCancelled = 1)
	BEGIN
		;THROW 52000,'ERROR - this day reservation has been already cancelled',1
	END

	DECLARE @ExistingParticipant AS int = (
		SELECT ParticipantID
		FROM ConfDayParticipants
		WHERE AttendeeID = @AttendeeID)

	IF (@ExistingParticipant IS NOT NULL)
	BEGIN
		;THROW 52000,'ERROR - This attendee has been already enrolled to this
		conference day',1
	END

	IF (@StudentID = '')
	BEGIN
		DECLARE @MyAdultLimit AS int = (
			SELECT AdultQuantity
			FROM ConfDayReservations
			WHERE DayReservationID = @ConfDayReservationID)

		DECLARE @AdultsAlreadyDeclared AS int = (
```

```
                        SELECT COUNT(ParticipantID)
                        FROM ConfDayParticipants
                        WHERE ConfDayReservationID = @ConfDayReservationID
                                AND StudentID = '')

                IF (@AdultsAlreadyDeclared=@MyAdultLimit)
                BEGIN
                        ;THROW 52000,'ERROR - You have already added declared number of
                        adult participants',1
                END
        END

        IF (@StudentID <> '')
        BEGIN
                DECLARE @MyStudentLimit AS int = (
                        SELECT StudentQuantity
                        FROM ConfDayReservations
                        WHERE DayReservationID = @ConfDayReservationID)

                DECLARE @StudentsAlreadyDeclared AS int = (
                        SELECT COUNT(ParticipantID)
                        FROM ConfDayParticipants
                        WHERE ConfDayReservationID = @ConfDayReservationID
                                AND StudentID <> '')

                IF (@StudentsAlreadyDeclared=@MyStudentLimit)
                BEGIN
                        ;THROW 52000,'ERROR - You have already added declared number of
                        students',1
                END
        END

        INSERT INTO ConfDayParticipants (ConfDayReservationID, AttendeeID, StudentID)
        VALUES (@ConfDayReservationID, @AttendeeID, @StudentID)
END
```

AddNewParticipantToWorkshop

Procedura przypisująca nowego uczestnika do danej rezerwacji na warsztat. Jako parametry przyjmuje ID uczestnika oraz ID rezerwacji na warsztat. Wykona się tylko wtedy, gdy dany uczestnik jest zapisany na dany dzień konferencji oraz nie bierze udziału w żadnym innym warsztacie w tym czasie.

```
ALTER PROCEDURE [dbo].[AddParticipantToWorkshop]

        @AttendeeID int,
        @WorkshopReservationID int
        AS
BEGIN

        SET NOCOUNT ON;

        DECLARE @IsCancelled AS int = (
                SELECT IsCancelled
                FROM WorkshopsReservations
                WHERE ReservationID = @WorkshopReservationID)

        IF (@IsCancelled = 1)
        BEGIN
```

```
        ;THROW 52000,'ERROR - this day reservation has been already cancelled',1
    END

    DECLARE @MyLimit AS int = (
        SELECT Quantity
        FROM WorkshopsReservations
        WHERE ReservationID=@WorkshopReservationID)

    DECLARE @AlreadyDeclared AS int = (
        SELECT COUNT(WorkshopParticipantID)
        FROM WorkshopsParticipants
        WHERE WorkshopReservationID=@WorkshopReservationID)

    IF (@AlreadyDeclared=@MyLimit)
    BEGIN
        ;THROW 52000,'ERROR - You have already added declared number of this
        workshop participants',1
    END

DECLARE @ConfDayReservationID AS int = (
        SELECT ConfDayReservationID
        FROM WorkshopsReservations
        WHERE ReservationID = @WorkshopReservationID)

    DECLARE @ConfDayParticipantID AS int = (
        SELECT ParticipantID
        FROM ConfDayParticipants
        WHERE ConfDayReservationID=@ConfDayReservationID
            AND AttendeeID=@AttendeeID)

    DECLARE @ExistingParticipant AS int = (
        SELECT WorkshopParticipantID
        FROM WorkshopsParticipants
        WHERE ConfDayParticipantID = @ConfDayParticipantID)

    IF (@ExistingParticipant IS NOT NULL)
    BEGIN
        ;THROW 52000,'ERROR - This attendee has been already enrolled to this
        workshop',1
    END

    DECLARE @WorkshopID AS int = (
        SELECT WorkshopID
        FROM WorkshopsReservations
        WHERE ReservationID = @WorkshopReservationID)

    DECLARE @ThisWorkshopStartTime as Datetime = (
        SELECT StartTime
        FROM Workshops
        WHERE WorkshopID = @WorkshopID)

    DECLARE @ThisWorkshopEndTime as Datetime = (
        SELECT EndTime
        FROM Workshops
        WHERE WorkshopID = @WorkshopID)

    DECLARE @Collisions AS int = (
        SELECT COUNT(WorkshopParticipantID)
        FROM WorkshopsParticipants as WP
        INNER JOIN WorkshopsReservations as WR ON WP.WorkshopReservationID =
        WR.ReservationID
        INNER JOIN Workshops as W ON W.WorkshopID = WR.WorkshopID
```

```
            WHERE ConfDayParticipantID=@ConfDayParticipantID
                  AND ((EndTime>@ThisWorkshopStartTime AND
                  EndTime<=@ThisWorkshopEndTime)
                  OR (@ThisWorkshopEndTime>StartTime AND
                  @ThisWorkshopEndTime<=EndTime)))

      IF (@Collisions <> 0)
      BEGIN
            ;THROW 52000,'ERROR - The attendee takes part in another workshop at
            this time',1
      END

      INSERT INTO WorkshopsParticipants (WorkshopReservationID,ConfDayParticipantID)
      VALUES (@WorkshopReservationID,@ConfDayParticipantID)

END
```

CancelConfDayReservation

Procedura anuluje rezerwację na dany dzień konferencji. Jako parametr przyjmuje ID rezerwacji na dzień.

```
ALTER PROCEDURE [dbo].[CancelConfDayReservation]

      @ConfDayReservationID int
      AS
BEGIN

      SET NOCOUNT ON;

      UPDATE ConfDayReservations SET IsCancelled = 1 WHERE DayReservationID =
      @ConfDayReservationID
END
```

CancelConferenceDay

Procedura odwołuje dany dzień konferencji. Jako parameter przyjmuje ID odwoływanego dnia. Odwołuje również wszystkie rezerwacje złożone na ten dzień.

```
ALTER PROCEDURE [dbo].[CancelConferenceDay]

      @ConferenceDayID int
AS
BEGIN

      SET NOCOUNT ON;

      UPDATE ConferenceDay SET IsCancelled = 1 WHERE ConferenceDayID =
      @ConferenceDayID
      UPDATE ConfDayReservations SET IsCancelled = 1 WHERE ConferenceDayID =
      @ConferenceDayID
END
```

CancelWorkshop

Procedura odwołuje dany warsztat. Jako parametr przyjmuje ID odwoływanego warsztatu. Odwołuje również wszystkie rezerwacje złożone na ten warsztat.

```
ALTER PROCEDURE [dbo].[CancelWorkshop]


	@WorkshopID int
AS
BEGIN

	SET NOCOUNT ON;

	UPDATE Workshops SET IsCancelled = 1 WHERE WorkshopID = @WorkshopID
	UPDATE WorkshopsReservations SET IsCancelled = 1 WHERE WorkshopID = @WorkshopID

  END
```

CancelWorkshopReservation

Procedura odwołuje daną rezerwację na warsztat. Jako parameter przyjmuje ID rezerwacji na warsztat.

```
ALTER PROCEDURE [dbo].[CancelWorkshopReservation]
@WorkshopReservationID int
AS
BEGIN

	SET NOCOUNT ON;

	UPDATE WorkshopsReservations SET IsCancelled = 1 WHERE ReservationID =
	@WorkshopReservationID
 END
```

ChangeConfDayReservationAdultQuantity

Procedura zmienia zadeklarowaną ilość osób dorosłych w rezerwacji na dany dzień. Jako parametry przyjmuje ID rezerwacji na dany dzień oraz nową deklarację ilości dorosłych. Procedura wykona się jeżeli tylko jest to możliwe, tzn.: jeżeli nowa ilość nie przekracza dostępnej ilości wolnych miejsc oraz jeżeli nie jest ona niższa niż ilość już zapisanych dorosłych uczestników.

```
ALTER PROCEDURE [dbo].[ChangeConfDayReservationAdultQuantity]

	@ConfDayReservationID int,
	@NewAdultQuantity int
	AS
BEGIN

	SET NOCOUNT ON;

	IF (@NewAdultQuantity <=0)
	BEGIN
		;THROW 52000,'ERROR - Quantity must be greater than 0',1
	END

	DECLARE @CurrentAdultQuantity AS int = (
```

```
		SELECT AdultQuantity
		FROM ConfDayReservations
		WHERE DayReservationID = @ConfDayReservationID)

	IF (@NewAdultQuantity=@CurrentAdultQuantity)
	BEGIN
		;THROW 52000,'WARNING - The new quantity is the same as the current',1
	END

	IF (@NewAdultQuantity>@CurrentAdultQuantity)
	BEGIN
		DECLARE @CurrentStudentQuantity AS int = (
			SELECT StudentQuantity
			FROM ConfDayReservations
			WHERE DayReservationID = @ConfDayReservationID)

		DECLARE @ConferenceDayID AS int = (
			SELECT ConferenceDayID
			FROM ConfDayReservations
			WHERE DayReservationID = @ConfDayReservationID)

		DECLARE @FreePlacesLeft AS int =
		dbo.FreePlacesLeftForDay(@ConferenceDayID)

		IF (@NewAdultQuantity-@CurrentAdultQuantity>@FreePlacesLeft)
		BEGIN
			;THROW 52000,'ERROR - There is no enough free places for this
			day',1
		END
	END

	IF (@NewAdultQuantity<@CurrentAdultQuantity)
	BEGIN
		DECLARE @AlreadyAddedAdults AS int = (
			SELECT COUNT(ParticipantID)
			FROM ConfDayParticipants
			WHERE ConfDayReservationID = @ConfDayReservationID
				AND StudentID = '')

		IF (@NewAdultQuantity<@AlreadyAddedAdults)
		BEGIN
			;THROW 52000,'ERROR - You have already added more adult
			participant than new quantity is',1
		END
	END

	UPDATE ConfDayReservations SET AdultQuantity = @NewAdultQuantity WHERE
	DayReservationID = @ConfDayReservationID

END
```

ChangeConfDayReservationStudentQuantity

Procedura zmienia zadeklarowaną ilość studentów w rezerwacji na dany dzień. Jako parametry przyjmuje ID rezerwacji na dany dzień oraz nową deklarację ilości studentów. Procedura wykona się jeżeli tylko jest to możliwe, tzn.: jeżeli nowa ilość nie przekracza dostępnej ilości wolnych miejsc oraz jeżeli nie jest ona niższa niż ilość już zapisanych studentów.

```
ALTER PROCEDURE [dbo].[ChangeConfDayReservationStudentQuantity]

	@ConfDayReservationID int,
	@NewStudentQuantity int
	AS
BEGIN

	SET NOCOUNT ON;

	IF (@NewStudentQuantity <=0)
	BEGIN
		;THROW 52000,'ERROR - Quantity must be greater than 0',1
	END

	DECLARE @CurrentStudentQuantity AS int = (
		SELECT StudentQuantity
		FROM ConfDayReservations
		WHERE DayReservationID = @ConfDayReservationID)

	IF (@NewStudentQuantity=@CurrentStudentQuantity)
	BEGIN
		;THROW 52000,'WARNING - The new quantity is the same as the current',1
	END

	IF (@NewStudentQuantity>@CurrentStudentQuantity)
	BEGIN
		DECLARE @CurrentAdultQuantity AS int = (
			SELECT AdultQuantity
			FROM ConfDayReservations
			WHERE DayReservationID = @ConfDayReservationID)

		DECLARE @ConferenceDayID AS int = (
			SELECT ConferenceDayID
			FROM ConfDayReservations
			WHERE DayReservationID = @ConfDayReservationID)

		DECLARE @FreePlacesLeft AS int =
		dbo.FreePlacesLeftForDay(@ConferenceDayID)

		IF (@NewStudentQuantity-@CurrentStudentQuantity>@FreePlacesLeft)
		BEGIN
			;THROW 52000,'ERROR - There is no enough free places for this
			day',1
		END
	END

	IF (@NewStudentQuantity<@CurrentStudentQuantity)
	BEGIN
		DECLARE @AlreadyAddedStudents AS int = (
			SELECT COUNT(ParticipantID)
			FROM ConfDayParticipants
			WHERE ConfDayReservationID = @ConfDayReservationID
				AND StudentID <> '')

		IF (@NewStudentQuantity<@AlreadyAddedStudents)
		BEGIN
			;THROW 52000,'ERROR - You have already added more students than
			new quantity is',1
		END
	END
```

```
	UPDATE ConfDayReservations SET StudentQuantity = @NewStudentQuantity WHERE
	DayReservationID = @ConfDayReservationID

END
```

ChangeConferenceDayLimit

Procedura zmienia limit osób na dany dzień konferencji. Jako parametry przyjmuje ID dnia konferencji oraz nowy limit. Procedura wykona się jeśli tylko jest to możliwe, tzn. jeżeli nowy limit nie jest mniejszy niż ilość osób już zapisanych na dany dzień konferencji.

```
ALTER PROCEDURE [dbo].[ChangeConferenceDayLimit]

	@ConferenceDayID int,
	@NewLimit int
	AS
BEGIN

	SET NOCOUNT ON;

	IF(@NewLimit<=0)
	BEGIN
		;THROW 52000,'ERROR - Limit must be greater than 0',1
	END

	IF(dbo.DeclaredPlacesForConfDay(@ConferenceDayID)>@NewLimit)
	BEGIN
		;THROW 52000,'ERROR - There is already declared more places for this day
		than new limit is',1
	END

	UPDATE ConferenceDay SET Limit = @NewLimit WHERE
	ConferenceDayID=@ConferenceDayID
END
```

ChangeWorkshopLimit

Procedura zmienia limit osób na dany warsztat. Jako parametry przyjmuje ID warsztatu oraz nowy limit. Procedura wykona się jeśli tylko jest to możliwe, tzn. jeżeli nowy limit nie jest mniejszy niż ilość osób już zapisanych na dany warsztat.

```
ALTER PROCEDURE [dbo].[ChangeWorkshopLimit]

	@WorkshopID int,
	@NewLimit int
	AS
BEGIN

	SET NOCOUNT ON;

	IF(@NewLimit<=0)
	BEGIN
		;THROW 52000,'ERROR - Limit must be greater than 0',1
	END

	IF(dbo.DeclaredPlacesForWorkshop(@WorkshopID)>@NewLimit)
	BEGIN
		;THROW 52000,'ERROR - There is already declared more places for this day
		than new limit is',1
	END
```

```
	UPDATE Workshops SET Limit = @NewLimit WHERE WorkshopID = @WorkshopID
END
```

ChangeWorkshopReservationQuantity

Procedura zmienia zadeklarowaną ilość osób w rezerwacji na dany warsztat.. Jako parametry przyjmuje ID rezerwacji na dany warsztat oraz nową deklarację ilości osób. Procedura wykona się jeżeli tylko jest to możliwe, tzn.: jeżeli nowa ilość nie przekracza dostępnej ilości wolnych miejsc oraz jeżeli nie jest ona niższa niż ilość osób już zapisanych na warsztat.

```
ALTER PROCEDURE [dbo].[ChangeWorkshopReservationQuantity]

	@WorkshopReservationID int,
	@NewQuantity int
	AS
BEGIN

	SET NOCOUNT ON;

	IF (@NewQuantity <=0)
	BEGIN
		;THROW 52000,'ERROR - Quantity must be greater than 0',1
	END

	DECLARE @CurrentQuantity AS int = (
		SELECT Quantity
		FROM WorkshopsReservations
		WHERE ReservationID = @WorkshopReservationID)

	IF (@NewQuantity=@CurrentQuantity)
	BEGIN
		;THROW 52000,'WARNING - The new quantity is the same as the current',1
	END

	IF (@NewQuantity>@CurrentQuantity)
	BEGIN

		DECLARE @WorkshopID AS int = (
			SELECT WorkshopID
			FROM WorkshopsReservations
			WHERE ReservationID = @WorkshopReservationID)

		DECLARE @FreePlacesLeft AS int =
		dbo.FreePlacesLeftForWorkshop(@WorkshopID)

		IF (@NewQuantity-@CurrentQuantity>@FreePlacesLeft)
		BEGIN
			;THROW 52000,'ERROR - There is no enough free places for this
			day',1
		END
	END

	IF (@NewQuantity<@CurrentQuantity)
	BEGIN
		DECLARE @AlreadyAddedParticipants AS int = (
			SELECT COUNT(WorkshopParticipantID)
			FROM WorkshopsParticipants
			WHERE WorkshopReservationID = @WorkshopReservationID)

		IF (@NewQuantity<@AlreadyAddedParticipants)
		BEGIN
```

```
                ;THROW 52000,'ERROR - You have already added more adult
                participant than new quantity is',1
        END
    END

    UPDATE WorkshopsReservations SET Quantity = @NewQuantity WHERE ReservationID =
    @WorkshopReservationID

END
```

9. Funkcje

DeclaredPlacesForConfDay

Funkcja zwraca ilość zarezerwowanych miejsc na dzień konferencji, którego ID zostanie podane jako parametr.

```
CREATE FUNCTION [dbo].[DeclaredPlacesForConfDay]
(
    @ConferenceDayID int
)
RETURNS int
AS
BEGIN

    RETURN (
        SELECT ISNULL(SUM(AdultQuantity+StudentQuantity),0)
        FROM ConfDayReservations
        WHERE ConferenceDayID=@ConferenceDayID
            AND IsCancelled = 0)

END
```

DeclaredPlacesForWorkshop

Funkcja zwraca ilość zarezerwowanych miejsc na warsztat, którego ID zostanie podane jako parametr.

```
CREATE FUNCTION [dbo].[DeclaredPlacesForWorkshop]
(
    @WorkshopID int
)
RETURNS int
AS
BEGIN

    RETURN (
        SELECT ISNULL(SUM(Quantity),0)
        FROM WorkshopsReservations
        WHERE WorkshopID = @WorkshopID
            AND IsCancelled = 0)

END
```

FreePlacesLeftForDay

Funkcja zwraca ilość wolnych miejsc na dzień konferencji, którego ID zostanie podane jako parametr.

```
CREATE FUNCTION [dbo].[FreePlacesLeftForDay]
(
	-- Add the parameters for the function here
	@ConferenceDayID INT
)
RETURNS INT
AS
BEGIN
	DECLARE @Limit INT = (
		SELECT Limit
		FROM dbo.ConferenceDay
		WHERE ConferenceDayID=@ConferenceDayID)

	DECLARE @AlreadyBooked INT = (
		SELECT ISNULL(SUM(AdultQuantity)+SUM(StudentQuantity),0)
		FROM dbo.ConfDayReservations
		WHERE ConferenceDayID=@ConferenceDayID AND IsCancelled=0)

	RETURN @Limit - @AlreadyBooked

END
```

FreePlacesLeftForWorkshop

Funkcja zwraca ilość wolnych miejsc na warsztat, którego ID zostanie podane jako parametr.

```
CREATE FUNCTION [dbo].[FreePlacesLeftForWorkshop]
(
	-- Add the parameters for the function here
	@WorkshopID INT
)
RETURNS INT
AS
BEGIN
	DECLARE @Limit INT = (
		SELECT Limit
		FROM dbo.Workshops
		WHERE WorkshopID=@WorkshopID)

	DECLARE @AlreadyBooked INT = (
		SELECT ISNULL(SUM(Quantity),0)
		FROM dbo.WorkshopsReservations
		WHERE WorkshopID=@WorkshopID AND IsCancelled=0)

	RETURN @Limit - @AlreadyBooked

END
GO
```

ConferenceDayPrice

Funkcja zwraca koszt udziału w dniu konferencji dla danego ID rezerwacji dnia.

```
CREATE FUNCTION [dbo].[ConferenceDayPrice]
(
	@ConfDayReservationID int
)
RETURNS money
AS
BEGIN

	DECLARE @ConferenceDayID AS int = (
		SELECT ConferenceDayID
		FROM ConfDayReservations
		WHERE DayReservationID = @ConfDayReservationID)

	DECLARE @StudentDiscount AS int = (
		SELECT StudentDiscount
		FROM ConferenceDay
		WHERE ConferenceDayID = @ConferenceDayID)

	DECLARE @ReservationDate AS date = (
		SELECT ReservationDate
		FROM ConfDayReservations
		WHERE DayReservationID = @ConfDayReservationID)

	DECLARE @MyLastDay AS date = (
		SELECT MIN(LastDay)
		FROM Prices
		WHERE LastDay>@ReservationDate
			AND ConferenceDayID = @ConferenceDayID)

	DECLARE @Price AS money = (
		SELECT Price
		FROM Prices
		WHERE LastDay = @MyLastDay
			AND ConferenceDayID = @ConferenceDayID)

	DECLARE @AdultQuantity AS int = (
		SELECT AdultQuantity
		FROM ConfDayReservations
		WHERE DayReservationID = @ConfDayReservationID)

	DECLARE @StudentQuantity AS int = (
		SELECT StudentQuantity
		FROM ConfDayReservations
		WHERE DayReservationID = @ConfDayReservationID)

	RETURN  ISNULL(@AdultQuantity*@Price + @StudentQuantity*@Price*(1-
	@StudentDiscount),0)

END
GO
```

WorkshopsPrice

Funkcja zwraca koszt udziału w warsztatach podczas jednego dnia konferencji dla danego ID rezerwacji dnia.

```
CREATE FUNCTION [dbo].[WorkshopsPrice]
(
	@ConfDayReservationID int
)
RETURNS money
AS
BEGIN

	RETURN (SELECT SUM(ISNULL(Quantity*Price,0))
			FROM WorkshopsReservations as WR
			INNER JOIN Workshops as W ON W.WorkshopID = WR.WorkshopID
			WHERE ConfDayReservationID = @ConfDayReservationID)

END
GO
```

ConferencePrice

Funkcja zwraca koszt udziału w całej konferencji dla danego ID rezerwacji konferencji.

```
CREATE FUNCTION [dbo].[ConferencePrice]
(
	@ReservationID int
)
RETURNS money
AS
BEGIN

			RETURN (SELECT SUM(dbo.ConferenceDayPrice(DayReservationID)
			+dbo.WorkshopsPrice(DayReservationID))
			FROM ConfDayReservations
			WHERE ReservationID = @ReservationID)

END
GO
```

10. Generator danych

Do wygenerowania danych posłużono się generatorem RedGate SQL Data Generator *(https://www.red-gate.com/products/sql-development/sql-data-generator/index).*

Wartości zostały wygenerowane ze zbiorów, zgodnych z założeniami projektu. Ponadto podczas generowania danych włączone zostały wszystkie triggery, co wraz z checkami w tabelach zapewniło spójność otrzymanych danych.

Przyjęto odpowiednie założenia co do ilości rekordów w poszczególnych tabelach:

Conferences – 75 rekordów (3 lata = 36 miesięcy po ok. 2 konferencje w miesiącu)

ConferenceDay – 200 rekordów (75 konferencji po 2-3 dni na każdą)

Prices – 500 rekordów (200 dni po 2-3 progi cenowe)

Workshops – 800 rekordów (200 dni, średnio po 4 warsztaty na każdy)

WorkshopsParticipants – 32 000 rekordów (800 warsztatów, średnio po 40 uczestników)

WorkshopsReservations – 6 400 rekordów (32 000 uczestników, średnio po 5 uczestników na jedną rezerwację)

ConfDayParticipants – 30 000 rekordów (200 dni, średnio po 150 osób na dzień - to aproksymuje do ok. 200 uczestników na każdą konferencję)

ConfDayReservations – 1 500 rekordów (30 000 uczestników, średnio po 20 na jedną rezerwację)

ConfReservations – 750 rekordów (1 500 rezerwacji dni, średnio po 2 dni na jedną rezerwację konferencji)

Payments – 2000 rekordów (750 rezerwacji, po 2-3 płatności na rezerwację)

Attendees – 3000 rekordów (30 000 uczestników dni konferencji, każdy uczestnik uczestniczy średnio w 10 dniach)

Clients – 200 (750 rezerwacji, każdy klient składa średnio ok. 4 rezerwacji)

11. Role użytkowników

-administrator (pełny dostęp do bazy)

- obsługa - dostęp do widoków i procedur informacyjnych

- właściciel firmy - może tworzyć konferencje i warsztaty, usuwać/anulować własne wydarzenia, modyfikować ich dane, dostęp do wszystkich widoków i procedur związanych z utworzonymi konferencjami.

-klient - może robić/anulować rezerwacje na konferencje i warsztaty, zgłaszać dowolną ilość osób na nie, korzystać z funkcji mówiącej o jego płatnościach, użyć funkcji zwracającej informację ile jeszcze pozostało do zapłaty za daną konferencję

-uczestnik - może sprawdzać na co jest zapisany, może stać się klientem, korzystać z procedury do edycji własnych danych