
# „System zarządzania konferencjami”

Projekt i Implementacja systemu bazodanowego.

**Wykonawcy:**

Anna Baran

Karolina Biela

Projekt realizowany w
ramach przedmiotu „Podstawy Baz Danych”

## Opis:

Projekt dotyczy firmy organizującej konferencje i warsztaty zarówno dla osób prywatnych jak i firm. Każda konferencja trwa jeden lub kilka dni, podczas których każdy z uczestników może wziąć udział w warsztatach. Ze względu na wcześniejszą rezerwacją oraz prawu skorzystania z ulg przysługującym studentom, funkcjonuje system zniżek.

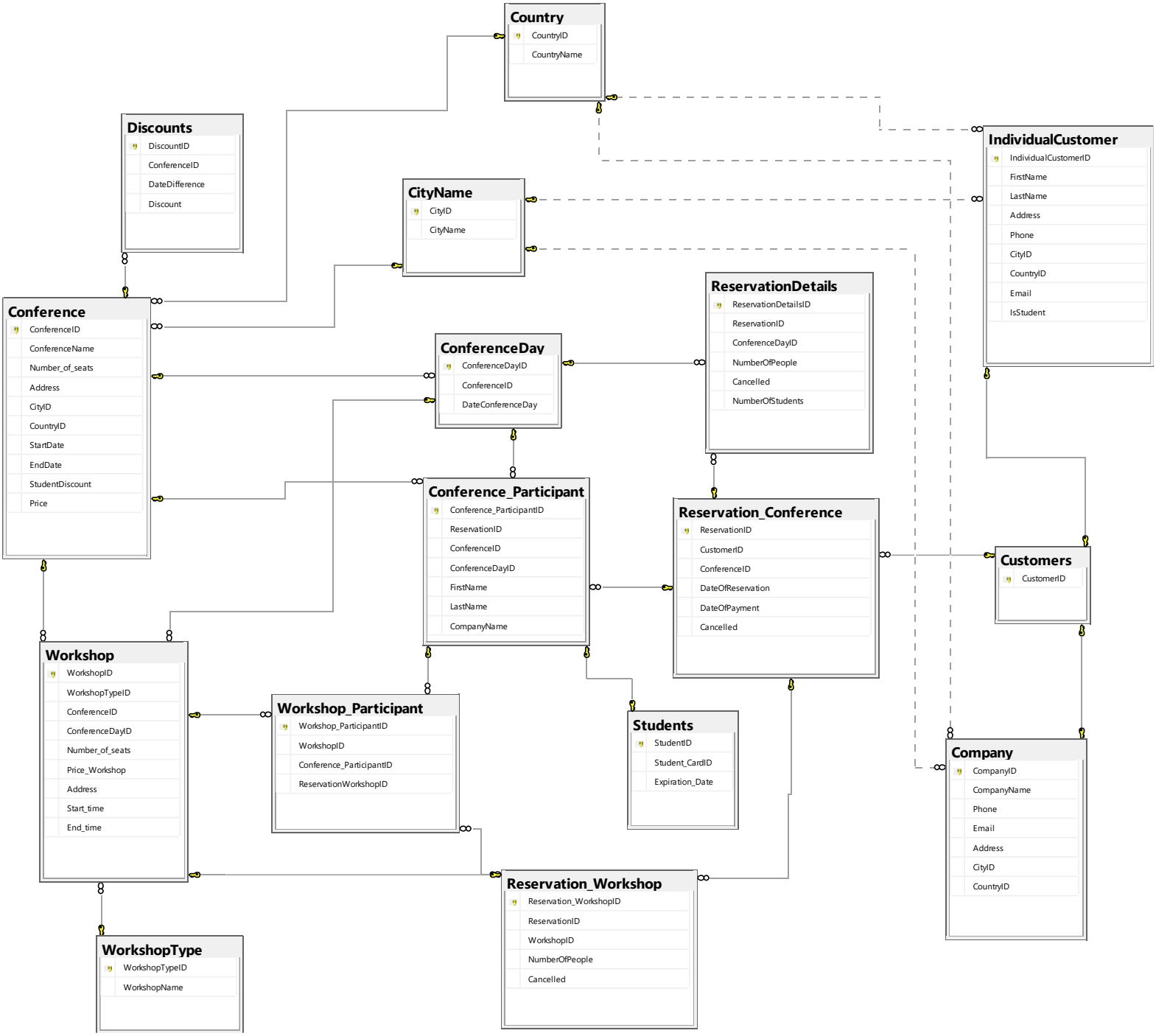
## Opis funkcji systemu wraz z informacją, co jaki użytkownik może wykonywać w systemie

- Administrator
  - Zarządzenie bazą z wszystkimi uprawnieniami

- Właściciel firmy
  - Pełny dostęp do danych oraz widoków
- Pracownik
  - Tworzenie i dostęp do konferencji i warsztatów
  - Obsługa klientów
  - Wprowadzanie danych dotyczących opłat oraz zniżek

- Klient
  - rejestracja na konferencję (wybór dni i warsztatów, podanie danych)
    - Indywidualny
    - Firma
      - podanie listy uczestników;
      - rezerwacja określonej liczby miejsc na dane dni i warsztaty
  - uiszczenie opłaty

# Schemat bazy:



# Tabele

| Name |
| --- |
| **dbo.CityName** |
| **dbo.Company** |
| **dbo.Conference** |
| **dbo.Conference_Participant** |
| **dbo.ConferenceDay** |
| **dbo.Country** |
| **dbo.Customers** |
| **dbo.Discounts** |
| **dbo.IndividualCustomer** |
| **dbo.Reservation_Conference** |
| **dbo.Reservation_Workshop** |
| **dbo.ReservationDetails** |
| **dbo.Students** |
| **dbo.Workshop** |
| **dbo.Workshop_Participant** |
| **dbo.WorkshopType** |

## *[dbo].[CityName]*

Tabela reprezentująca nazwy miast w bazie danych. Używana jako słownik w pozostałych tabelach. Ilość rekordów:1001

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | CityID | int | 4 | False |
| | CityName | nvarchar(50) | 100 | False |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_CityName | CityID | True |

### SQL Script

```
CREATE TABLE [dbo].[CityName]

(

[CityID] [int] NOT NULL,

[CityName] [nvarchar] (50) COLLATE Polish_CI_AS NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[CityName] ADD CONSTRAINT [PK_CityName] PRIMARY KEY CLUSTERED
([CityID]) ON [PRIMARY]

GO
```

### Używane przez

[dbo].[Company]

[dbo].[Conference]

[dbo].[IndividualCustomer]

[dbo].[AddConference]

[dbo].[AddCustomerCompany]

[dbo].[AddCustomerIndividual]

[dbo].[ChangeCompanyData]

[dbo].[ChangeIndividualCustomerData]

## *[dbo].[Company]*

Tabela reprezentująca Klienta Firmowego w bazie danych. Ilość rekordów:3001

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | CompanyID | int | 4 | False |
| | CompanyName | nvarchar(50) | 100 | False |
| | Phone | nvarchar(50) | 100 | False |
| | Email | nvarchar(50) | 100 | False |
| | Address | nvarchar(50) | 100 | False |
| **FK** | CityID | int | 4 | False |
| **FK** | CountryID | int | 4 | False |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_Company_1 | CompanyID | True |
| | IX_Company | CompanyID | True |
| | CompanyName | CompanyName | |

### Foreign Keys

| Name | No Check | Columns |
|---|---|---|
| **FK_Company_CityName** | True | CityID->[dbo].[CityName].[CityID] |
| **FK_Company_Country** | True | CountryID->[dbo].[Country].[CountryID] |
| **FK_Company_Customers** | | CompanyID->[dbo].[Customers].[CustomerID] |

## SQL Script

```
CREATE TABLE [dbo].[Company]

([CompanyID] [int] NOT NULL,

[CompanyName] [nvarchar] (50) COLLATE Polish_CI_AS NOT NULL,

[Phone] [nvarchar] (50) COLLATE Polish_CI_AS NOT NULL,

[Email] [nvarchar] (50) COLLATE Polish_CI_AS NOT NULL,

[Address] [nvarchar] (50) COLLATE Polish_CI_AS NOT NULL,

[CityID] [int] NOT NULL,

[CountryID] [int] NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Company] ADD CONSTRAINT [PK_Company_1] PRIMARY KEY CLUSTERED
([CompanyID]) ON [PRIMARY]

GO

CREATE UNIQUE NONCLUSTERED INDEX [IX_Company] ON [dbo].[Company] ([CompanyID]) ON
[PRIMARY]

GO

CREATE NONCLUSTERED INDEX [CompanyName] ON [dbo].[Company] ([CompanyName]) ON
[PRIMARY]

GO

ALTER TABLE [dbo].[Company] WITH NOCHECK ADD CONSTRAINT [FK_Company_CityName]
FOREIGN KEY ([CityID]) REFERENCES [dbo].[CityName] ([CityID])

GO

ALTER TABLE [dbo].[Company] WITH NOCHECK ADD CONSTRAINT [FK_Company_Country]
FOREIGN KEY ([CountryID]) REFERENCES [dbo].[Country] ([CountryID])

GO

ALTER TABLE [dbo].[Company] ADD CONSTRAINT [FK_Company_Customers] FOREIGN KEY
([CompanyID]) REFERENCES [dbo].[Customers] ([CustomerID])

GO

ALTER TABLE [dbo].[Company] NOCHECK CONSTRAINT [FK_Company_CityName]

GO

ALTER TABLE [dbo].[Company] NOCHECK CONSTRAINT [FK_Company_Country]

GO
```

## Korzysta z

[dbo].[CityName]

[dbo].[Country]

[dbo].[Customers]

## Używane przez

[dbo].[CancelledReservations]

[dbo].[GeneratorCompany]

[dbo].[GeneratorConference_participantCompany]

[dbo].[GeneratorReservationDetailsCompany]

[dbo].[GeneratorReservationsIDCompany]

[dbo].[GeneratorWorkshopKompania]

[dbo].[AddConferenceParticipant]

[dbo].[AddCustomerCompany]

[dbo].[ChangeCompanyData]

[dbo].[Conference_Participants]

## *[dbo].[Conference]*

Tabela reprezentująca Konferencje w bazie danych. Ilość rekordów:102

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | ConferenceID | int | 4 | False |
| | ConferenceName | nvarchar(50) | 100 | False |
| | Number_of_seats | int | 4 | True |
| | Address | varchar(50) | 50 | True |
| **FK** | CityID | int | 4 | False |
| **FK** | CountryID | int | 4 | False |
| | StartDate | date | 3 | True |
| | EndDate | date | 3 | True |
| | StudentDiscount | real | 4 | True |
| | Price | money | 8 | False |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_Conference | ConferenceID | True |
| | IX_Conference | ConferenceName | |
| | IX_Conference_1 | CountryID | |
| | IX_Conference_2 | CityID | |

### Foreign Keys

| Name | Columns |
|---|---|
| **FK_Conference_CityName** | CityID->[dbo].[CityName].[CityID] |
| **FK_Conference_Country** | CountryID->[dbo].[Country].[CountryID] |

## SQL Script

```
CREATE TABLE [dbo].[Conference]

(

[ConferenceID] [int] NOT NULL,

[ConferenceName] [nvarchar] (50) COLLATE Polish_CI_AS NOT NULL,

[Number_of_seats] [int] NULL,

[Address] [varchar] (50) COLLATE Polish_CI_AS NULL,

[CityID] [int] NOT NULL,

[CountryID] [int] NOT NULL,

[StartDate] [date] NULL,

[EndDate] [date] NULL,

[StudentDiscount] [real] NULL,

[Price] [money] NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Conference] ADD CONSTRAINT [PK_Conference] PRIMARY KEY CLUSTERED
([ConferenceID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [IX_Conference_2] ON [dbo].[Conference] ([CityID]) ON
[PRIMARY]

GO

CREATE NONCLUSTERED INDEX [IX_Conference] ON [dbo].[Conference] ([ConferenceName])
ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [IX_Conference_1] ON [dbo].[Conference] ([CountryID]) ON
[PRIMARY]

GO

ALTER TABLE [dbo].[Conference] ADD CONSTRAINT [FK_Conference_CityName] FOREIGN KEY
([CityID]) REFERENCES [dbo].[CityName] ([CityID])

GO

ALTER TABLE [dbo].[Conference] ADD CONSTRAINT [FK_Conference_Country] FOREIGN KEY
([CountryID]) REFERENCES [dbo].[Country] ([CountryID])

GO
```

## Korzysta z

[dbo].[CityName]

[dbo].[Country]

## Używane przez

[dbo].[Conference_Participant]

[dbo].[ConferenceDay]

[dbo].[Discounts]

[dbo].[Workshop]

[dbo].[ConferenceDayParticipants]

[dbo].[ConferenceParticipants]

[dbo].[PopularConferences]

[dbo].[PriceOfReservation]

[dbo].[WorkshopsInConferenceList]

[dbo].[WorkshopsInDayList]

[dbo].[AddConference]

[dbo].[AddConferenceDay]

[dbo].[AddDiscount]

[dbo].[AddReservation_Conference]

[dbo].[AddReservationDetails]

[dbo].[AddWorkshop]

[dbo].[ConferenceDayAvailablePlaces]

[dbo].[ConferenceDayFreePlaces]

[dbo].[DateDifferenceDiscount]

[dbo].[GetDiscount]

[dbo].[Reservation_ConferenceValue]

[dbo].[StudentDiscount]

## *[dbo].[Conference_Participant]*

Tabela reprezentująca uczestnika konferencji w danym dniu. Ilość rekordów: 106327

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | Conference_ParticipantID | int | 4 | False |
| **FK** | ReservationID | int | 4 | False |
| **FK** | ConferenceID | int | 4 | False |
| **FK** | ConferenceDayID | int | 4 | False |
| | FirstName | nvarchar(50) | 100 | False |
| | LastName | nvarchar(50) | 100 | False |
| | CompanyName | nvarchar(50) | 100 | True |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_Conference_Participant | Conference_ParticipantID | True |
| | ConferenceID | ConferenceID | |
| | LastName | LastName | |

### Foreign Keys

| Name | Columns |
|---|---|
| **FK_Conference_Participant_-Conference** | ConferenceID->[dbo].[Conference].[ConferenceID] |
| **FK_Conference_Participant_-ConferenceDay** | ConferenceDayID->[dbo].[ConferenceDay].[ConferenceDayID] |
| **FK_Conference_Participant_-Reservation_Conference** | ReservationID->[dbo].[Reservation_Conference].[ReservationID] |

## SQL Script

```
CREATE TABLE [dbo].[Conference_Participant]

(

[Conference_ParticipantID] [int] NOT NULL,

[ReservationID] [int] NOT NULL,

[ConferenceID] [int] NOT NULL,

[ConferenceDayID] [int] NOT NULL,

[FirstName] [nvarchar] (50) COLLATE Polish_CI_AS NOT NULL,

[LastName] [nvarchar] (50) COLLATE Polish_CI_AS NOT NULL,

[CompanyName] [nvarchar] (50) COLLATE Polish_CI_AS NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Conference_Participant] ADD CONSTRAINT [PK_Conference_-
Participant] PRIMARY KEY CLUSTERED  ([Conference_ParticipantID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [ConferenceID] ON [dbo].[Conference_Participant]
([ConferenceID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [LastName] ON [dbo].[Conference_Participant] ([LastName])
ON [PRIMARY]

GO

ALTER TABLE [dbo].[Conference_Participant] ADD CONSTRAINT [FK_Conference_-
Participant_Conference] FOREIGN KEY ([ConferenceID]) REFERENCES [dbo].[Conference]
([ConferenceID])

GO

ALTER TABLE [dbo].[Conference_Participant] ADD CONSTRAINT [FK_Conference_-
Participant_ConferenceDay] FOREIGN KEY ([ConferenceDayID]) REFERENCES
[dbo].[ConferenceDay] ([ConferenceDayID])

GO

ALTER TABLE [dbo].[Conference_Participant] ADD CONSTRAINT [FK_Conference_-
Participant_Reservation_Conference] FOREIGN KEY ([ReservationID]) REFERENCES
[dbo].[Reservation_Conference] ([ReservationID])

GO
```

## Korzysta z

[dbo].[Conference]

[dbo].[ConferenceDay]

[dbo].[Reservation_Conference]

## Używane przez

[dbo].[Students]

[dbo].[Workshop_Participant]

[dbo].[ConferenceDayParticipants]

[dbo].[ConferenceParticipants]

[dbo].[GeneratorParticipants_IndividualStudent]

[dbo].[GeneratorWorkshopParticipantsAdding]

[dbo].[WorkshopParticipants]

[dbo].[AddConferenceParticipant]

[dbo].[AddReservationDetails]

[dbo].[DayParticipantsList]

[dbo].[ListOfWorkshopParticipants]

## *[dbo].[ConferenceDay]*

Tabela reprezentująca określony Dzień Konferencji w bazie danych. Ilość rekordów: 379

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | ConferenceDayID | int | 4 | False |
| **FK** | ConferenceID | int | 4 | True |
| | DateConferenceDay | date | 3 | True |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_ConferenceDay | ConferenceDayID | True |
| | ConferenceID | ConferenceID | |

### Foreign Keys

| Name | Columns |
|---|---|
| **FK_ConferenceDay_Conference** | ConferenceID->[dbo].[Conference].[ConferenceID] |

## SQL Script

```
CREATE TABLE [dbo].[ConferenceDay]

(

[ConferenceDayID] [int] NOT NULL,

[ConferenceID] [int] NULL,

[DateConferenceDay] [date] NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[ConferenceDay] ADD CONSTRAINT [PK_ConferenceDay] PRIMARY KEY
CLUSTERED  ([ConferenceDayID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [ConferenceID] ON [dbo].[ConferenceDay] ([ConferenceID])
ON [PRIMARY]

GO

ALTER TABLE [dbo].[ConferenceDay] ADD CONSTRAINT [FK_ConferenceDay_Conference]
FOREIGN KEY ([ConferenceID]) REFERENCES [dbo].[Conference] ([ConferenceID])

GO
```

## Korzysta z

[dbo].[Conference]

## Używane przez

[dbo].[Conference_Participant]

[dbo].[ReservationDetails]

[dbo].[Workshop]

[dbo].[ConferenceDayParticipants]

[dbo].[GeneratorConference_participantCompany]

[dbo].[GeneratorParticipants_Individual]

[dbo].[GeneratorReservationDetailsCompany]

[dbo].[GeneratorReservationDetailsIndividual]

[dbo].[GeneratorReservationsIDCompany]

[dbo].[GeneratorReservationsIDIndywidual]

[dbo].[WorkshopParticipants]

[dbo].[WorkshopsInDayList]

[dbo].[AddConferenceDay]

[dbo].[AddReservationDetails]

[dbo].[CollisionsInWorkshops]

[dbo].[ConferenceDayAvailablePlaces]

[dbo].[ConferenceDayFreePlaces]

[dbo].[StudentDiscount]

## *[dbo].[Country]*

Tabela reprezentująca kraje, w których odbywają się konferencje oraz z których pochodzą klienci. Ilość rekordów: 207

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | CountryID | int | 4 | False |
| | CountryName | nvarchar(50) | 100 | False |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_Country | CountryID | True |
| | IX_Country | CountryID | True |
| | IX_Country_1 | CountryName | True |

### SQL Script

```
CREATE TABLE [dbo].[Country]

(

[CountryID] [int] NOT NULL,

[CountryName] [nvarchar] (50) COLLATE Polish_CI_AS NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Country] ADD CONSTRAINT [PK_Country] PRIMARY KEY CLUSTERED
([CountryID]) ON [PRIMARY]

GO

CREATE UNIQUE NONCLUSTERED INDEX [IX_Country] ON [dbo].[Country] ([CountryID]) ON
[PRIMARY]

GO

CREATE UNIQUE NONCLUSTERED INDEX [IX_Country_1] ON [dbo].[Country] ([CountryName])
ON [PRIMARY]

GO
```

## Używane przez

[dbo].[Company]

[dbo].[Conference]

[dbo].[IndividualCustomer]

[dbo].[AddConference]

[dbo].[AddCustomerCompany]

[dbo].[AddCustomerIndividual]

[dbo].[ChangeCompanyData]

[dbo].[ChangeIndividualCustomerData]

## *[dbo].[Customers]*

Tabela reprezentująca numery ID Klienta Indywidualnego oraz firmowego. Ilość rekordów: 8003

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | CustomerID | int | 4 | False |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_Customers | CustomerID | True |

### SQL Script

```
CREATE TABLE [dbo].[Customers]

(

[CustomerID] [int] NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Customers] ADD CONSTRAINT [PK_Customers] PRIMARY KEY CLUSTERED
([CustomerID]) ON [PRIMARY]

GO
```

## Używane przez

[dbo].[Company]

[dbo].[IndividualCustomer]

[dbo].[Reservation_Conference]

[dbo].[GeneratorCompany]

[dbo].[AddCustomerCompany]

[dbo].[AddCustomerIndividual]

[dbo].[AddReservation_Conference]

[dbo].[AddReservationWorkshop]

[dbo].[AddStudent]

[dbo].[Conference_Participants]

[dbo].[StudentDiscount]

## *[dbo].[Discounts]*

Tabela zawierająca progi zniżkowe dla każdej konferencji. Wysokość zniżki zależy od tego,o ile przed rozpoczęciem konferencji klient dokonał rezerwacji. Ilość rekordów: 303

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | DiscountID | int | 4 | False |
| **FK** | ConferenceID | int | 4 | False |
| | DateDifference | int | 4 | False |
| | Discount | real | 4 | False |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_Discounts | DiscountID | True |
| | IX_Discounts | ConferenceID | |

### Foreign Keys

| Name | Columns |
|---|---|
| **FK_Discounts_Conference** | ConferenceID->[dbo].[Conference].[ConferenceID] |

## SQL Script

```
CREATE TABLE [dbo].[Discounts]

(

[DiscountID] [int] NOT NULL,

[ConferenceID] [int] NULL,

[DateDifference] [int] NULL,

[Discount] [real] NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Discounts] ADD CONSTRAINT [PK_Discounts] PRIMARY KEY CLUSTERED
([DiscountID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [IX_Discounts] ON [dbo].[Discounts] ([ConferenceID]) ON
[PRIMARY]

GO

ALTER TABLE [dbo].[Discounts] ADD CONSTRAINT [FK_Discounts_Conference] FOREIGN KEY
([ConferenceID]) REFERENCES [dbo].[Conference] ([ConferenceID])

GO
```

## Korzysta z

[dbo].[Conference]

## Używane przez

[dbo].[PriceOfReservation]

[dbo].[AddDiscount]

[dbo].[GetDiscount]

## *[dbo].[IndividualCustomer]*

Tabela reprezentująca Klienta Indywidualnego. Ilość rekordów: 4001

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | IndividualCustomerID | int | 4 | False |
| | FirstName | nvarchar(50) | 100 | False |
| | LastName | nvarchar(50) | 100 | False |
| | Address | varchar(50) | 50 | False |
| | Phone | nvarchar(50) | 100 | False |
| **FK** | CityID | int | 4 | False |
| **FK** | CountryID | int | 4 | False |
| | Email | nvarchar(50) | 100 | False |
| | IsStudent | bit | 1 | True |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_IndividualCustomer | IndividualCustomerID | True |

### Foreign Keys

| Name | No Check | Columns |
|---|---|---|
| **FK_IndividualCustomer_CityName** | True | CityID->[dbo].[CityName].[CityID] |
| **FK_IndividualCustomer_Country** | True | CountryID->[dbo].[Country].[CountryID] |
| **FK_IndividualCustomer_Customers** | | IndividualCustomerID->[dbo].[Customers].[CustomerID] |

## SQL Script

```
CREATE TABLE [dbo].[IndividualCustomer]

(

[IndividualCustomerID] [int] NOT NULL,

[FirstName] [nvarchar] (50) NOT NULL,

[LastName] [nvarchar] (50) NOT NULL,

[Address] [varchar] (50) NOT NULL,

[Phone] [nvarchar] (50) NOT NULL,

[CityID] [int] NOT NULL,

[CountryID] [int] NOT NULL,

[Email] [nvarchar] (50) NOT NULL,

[IsStudent] [bit] NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[IndividualCustomer] ADD CONSTRAINT [PK_IndividualCustomer]
PRIMARY KEY CLUSTERED  ([IndividualCustomerID]) ON [PRIMARY]

GO

ALTER TABLE [dbo].[IndividualCustomer] WITH NOCHECK ADD CONSTRAINT [FK_Individual-
Customer_CityName] FOREIGN KEY ([CityID]) REFERENCES [dbo].[CityName] ([CityID])

GO

ALTER TABLE [dbo].[IndividualCustomer] WITH NOCHECK ADD CONSTRAINT [FK_Individual-
Customer_Country] FOREIGN KEY ([CountryID]) REFERENCES [dbo].[Country] ([Country-
ID])

GO

ALTER TABLE [dbo].[IndividualCustomer] ADD CONSTRAINT [FK_IndividualCustomer_-
Customers] FOREIGN KEY ([IndividualCustomerID]) REFERENCES [dbo].[Customers]
([CustomerID])

GO

ALTER TABLE [dbo].[IndividualCustomer] NOCHECK CONSTRAINT [FK_IndividualCustomer_-
CityName]

GO

ALTER TABLE [dbo].[IndividualCustomer] NOCHECK CONSTRAINT [FK_IndividualCustomer_-
Country]

GO
```

## Korzysta z

[dbo].[CityName]

[dbo].[Country]

[dbo].[Customers]

## Używane przez

[dbo].[CancelledReservations]

[dbo].[GeneratorCompany]

[dbo].[GeneratorParticipants_Individual]

[dbo].[GeneratorParticipants_IndividualStudent]

[dbo].[GeneratorReservationDetailsIndividual]

[dbo].[GeneratorReservationsIDIndywidual]

[dbo].[GeneratorWorkshopIndywidualny]

[dbo].[AddConferenceParticipant]

[dbo].[AddCustomerIndividual]

[dbo].[AddReservationDetails]

[dbo].[ChangeIndividualCustomerData]

## *[dbo].[Reservation_Conference]*

Tabela zawierająca rezerwacje poszczególnych klientów na daną konferencję. Ilość rekordów: 6004

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | ReservationID | int | 4 | False |
| **FK** | CustomerID | int | 4 | False |
| | ConferenceID | int | 4 | False |
| | DateOfReservation | datetime | 8 | False |
| | DateOfPayment | datetime | 8 | True |
| | Cancelled | bit | 1 | False |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_Reservation_Conference | ReservationID | True |
| | CustomerID | CustomerID | |
| | IX_Reservation_Conference | ConferenceID | |

### Foreign Keys

| Name | Columns |
|---|---|
| **FK_Reservation_Conference_Customers** | CustomerID->[dbo].[Customers].[CustomerID] |

## SQL Script

```
CREATE TABLE [dbo].[Reservation_Conference]

(

[ReservationID] [int] NOT NULL,

[CustomerID] [int] NOT NULL,

[ConferenceID] [int] NOT NULL,

[DateOfReservation] [datetime] NOT NULL,

[DateOfPayment] [datetime] NULL,

[Cancelled] [bit] NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Reservation_Conference] ADD CONSTRAINT [PK_Reservation_-
Conference] PRIMARY KEY CLUSTERED  ([ReservationID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [IX_Reservation_Conference] ON [dbo].[Reservation_-
Conference] ([ConferenceID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [CustomerID] ON [dbo].[Reservation_Conference]
([CustomerID]) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Reservation_Conference] ADD CONSTRAINT [FK_Reservation_-
Conference_Customers] FOREIGN KEY ([CustomerID]) REFERENCES [dbo].[Customers]
([CustomerID])

GO
```

## Korzysta z

[dbo].[Customers]

## Używane przez

[dbo].[Conference_Participant]

[dbo].[Reservation_Workshop]

[dbo].[ReservationDetails]

[dbo].[CancelledReservations]

[dbo].[ConferenceDayParticipants]

[dbo].[ConferenceParticipants]

[dbo].[ConferenceToCancelled]

[dbo].[GeneratorConference_participantCompany]

[dbo].[GeneratorParticipants_Individual]

[dbo].[GeneratorParticipants_IndividualStudent]

[dbo].[GeneratorReservationDetailsCompany]

[dbo].[GeneratorReservationDetailsIndividual]

[dbo].[GeneratorReservationsIDCompany]

[dbo].[GeneratorReservationsIDIndywidual]

[dbo].[GeneratorWorkshopIndywidualny]

[dbo].[GeneratorWorkshopKompania]

[dbo].[NumberOfCustomerReservations]

[dbo].[PopularConferences]

[dbo].[PriceOfReservation]

[dbo].[RegularCustomers]

[dbo].[AddConferenceParticipant]

[dbo].[AddReservation_Conference]

[dbo].[AddReservationDetails]

[dbo].[CancelReservation_Conference]

[dbo].[UpdatePaymentDate]

[dbo].[Conference_Participants]

[dbo].[DateDifferenceDiscount]

[dbo].[DayParticipantsList]

[dbo].[ListOfWorkshopParticipants]

[dbo].[Reservation_ConferenceValue]

[dbo].[Reservation_WorkshopValue]

[dbo].[StudentDiscount]

## *[dbo].[Reservation_Workshop]*

Tabela zawierająca rezerwacje poszczególnych warsztatów dla danej rezerwacji konferencji.
Ilość rekordów: 4952

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | Reservation_WorkshopID | int | 4 | False |
| **FK** | ReservationID | int | 4 | False |
| **FK** | WorkshopID | int | 4 | False |
| | NumberOfPeople | int | 4 | False |
| | Cancelled | bit | 1 | False |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_Reservation_Workshop | Reservation_WorkshopID | True |
| | ReservationID | ReservationID | |
| | WorkshopID | WorkshopID | |

### Foreign Keys

| Name | Columns |
|---|---|
| **FK_Reservation_Workshop_Reservation_-Conference** | ReservationID->[dbo].[Reservation_-Conference].[ReservationID] |
| **FK_Reservation_Workshop_Workshop** | WorkshopID->[dbo].[Workshop].[WorkshopID] |

## SQL Script

```
CREATE TABLE [dbo].[Reservation_Workshop]

(

[Reservation_WorkshopID] [int] NOT NULL,

[ReservationID] [int] NOT NULL,

[WorkshopID] [int] NOT NULL,

[NumberOfPeople] [int] NOT NULL,

[Cancelled] [bit] NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Reservation_Workshop] ADD CONSTRAINT [PK_Reservation_Workshop]
PRIMARY KEY CLUSTERED  ([Reservation_WorkshopID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [ReservationID] ON [dbo].[Reservation_Workshop]
([ReservationID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [WorkshopID] ON [dbo].[Reservation_Workshop] ([Workshop-
ID]) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Reservation_Workshop] ADD CONSTRAINT [FK_Reservation_Workshop_-
Reservation_Conference] FOREIGN KEY ([ReservationID]) REFERENCES
[dbo].[Reservation_Conference] ([ReservationID])

GO

ALTER TABLE [dbo].[Reservation_Workshop] ADD CONSTRAINT [FK_Reservation_Workshop_-
Workshop] FOREIGN KEY ([WorkshopID]) REFERENCES [dbo].[Workshop] ([WorkshopID])

GO
```

## Korzysta z

[dbo].[Reservation_Conference]

[dbo].[Workshop]

## Używane przez

[dbo].[Workshop_Participant]

[dbo].[GeneratorWorkshopParticipantsAdding]

[dbo].[PopularWorkshops]

[dbo].[AddReservationWorkshop]

[dbo].[FreePlacesForWorkshop]

[dbo].[ListOfWorkshopParticipants]

[dbo].[Reservation_WorkshopValue]

## *[dbo].[ReservationDetails]*

Tabela zawierająca szczegóły rezerwacji konferencji danego klienta. Na każdy dzień konferencji stworzony zostaje osobny rekord. Ilość rekordów: 21644

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | ReservationDetailsID | int | 4 | False |
| **FK** | ReservationID | int | 4 | False |
| **FK** | ConferenceDayID | int | 4 | False |
| | NumberOfPeople | int | 4 | True |
| | Cancelled | bit | 1 | True |
| | NumberOfStudents | int | 4 | True |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_ReservationDetails | ReservationDetailsID | True |
| | IX_ReservationDetails | ReservationID | |
| | IX_ReservationDetails_1 | ConferenceDayID | |

### Triggers

| Name | ANSI Nulls On | Quoted Identifier On | On |
|---|---|---|---|
| **MoreSeatsThanPeople** | True | True | After Insert |

### Foreign Keys

| Name | Columns |
|---|---|
| **FK_ReservationDetails_ConferenceDay** | ConferenceDayID->[dbo].[ConferenceDay].[ConferenceDayID] |
| **FK_ReservationDetails_Reservation_-Conference** | ReservationID->[dbo].[Reservation_Conference].[ReservationID] |

## SQL Script

```
CREATE TABLE [dbo].[ReservationDetails]

(

[ReservationDetailsID] [int] NOT NULL,

[ReservationID] [int] NOT NULL,

[ConferenceDayID] [int] NOT NULL,

[NumberOfPeople] [int] NULL,

[Cancelled] [bit] NULL,

[NumberOfStudents] [int] NULL

) ON [PRIMARY]

GO

--sprawdza, czy ilość miejsc dostępnych danego dnia konferencji nie jest mniejsza
od ilości miejsc zarezerwowanych w tworzonej rezerwacji.

CREATE TRIGGER [dbo].[MoreSeatsThanPeople] on [dbo].[ReservationDetails]

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON

DECLARE @reserved int;

DECLARE @free int;

SET @reserved = (SELECT NumberOfPeople FROM INSERTED);

SET @free = dbo.ConferenceDayFreePlaces((SELECT ConferenceDayID FROM INSERTED)) +
@reserved;

IF(@free < @reserved)

BEGIN

RAISERROR('Only %d places are available', 16, 1, @reserved, @free)

ROLLBACK TRANSACTION

END

END

GO

ALTER TABLE [dbo].[ReservationDetails] ADD CONSTRAINT [PK_ReservationDetails]
PRIMARY KEY CLUSTERED  ([ReservationDetailsID]) ON [PRIMARY]
```

```
GO

CREATE NONCLUSTERED INDEX [IX_ReservationDetails_1] ON [dbo].[ReservationDetails]
([ConferenceDayID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [IX_ReservationDetails] ON [dbo].[ReservationDetails]
([ReservationID]) ON [PRIMARY]

GO

ALTER TABLE [dbo].[ReservationDetails] ADD CONSTRAINT [FK_ReservationDetails_-
ConferenceDay] FOREIGN KEY ([ConferenceDayID]) REFERENCES [dbo].[ConferenceDay]
([ConferenceDayID])

GO

ALTER TABLE [dbo].[ReservationDetails] ADD CONSTRAINT [FK_ReservationDetails_-
Reservation_Conference] FOREIGN KEY ([ReservationID]) REFERENCES
[dbo].[Reservation_Conference] ([ReservationID])

GO
```

## Korzysta z

[dbo].[ConferenceDay]

[dbo].[Reservation_Conference]

## Używane przez

[dbo].[GeneratorConference_participantCompany]

[dbo].[GeneratorWorkshopIndywidualny]

[dbo].[GeneratorWorkshopKompania]

[dbo].[PopularConferences]

[dbo].[PriceOfReservation]

[dbo].[AddReservationDetails]

[dbo].[ConferenceDayAvailablePlaces]

[dbo].[ConferenceDayFreePlaces]

[dbo].[DayParticipantsList]

[dbo].[StudentDiscount]

## *[dbo].[Students]*

Tabela zawierająca dane studenta - uczestnika konferencji. Ilość rekordów: 5495

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK,FK** | StudentID | int | 4 | False |
| | Student_CardID | int | 4 | True |
| | Expiration_Date | date | 3 | True |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_Students | StudentID | True |
| | Student_CardID | Student_CardID | True |

### Foreign Keys

| Name | Columns |
|---|---|
| **FK_Students_Conference_Participant** | StudentID->[dbo].[Conference_Participant].[Conference_ParticipantID] |

## SQL Script

```
CREATE TABLE [dbo].[Students]

(

[StudentID] [int] NOT NULL,

[Student_CardID] [int] NULL,

[Expiration_Date] [date] NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Students] ADD CONSTRAINT [PK_Students] PRIMARY KEY CLUSTERED
([StudentID]) ON [PRIMARY]

GO

CREATE UNIQUE NONCLUSTERED INDEX [Student_CardID] ON [dbo].[Students] ([Student_-
CardID]) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Students] ADD CONSTRAINT [FK_Students_Conference_Participant]
FOREIGN KEY ([StudentID]) REFERENCES [dbo].[Conference_Participant] ([Conference_-
ParticipantID])

GO
```

## Korzysta z

[dbo].[Conference_Participant]

## Używane przez

[dbo].[AddConferenceParticipant]

[dbo].[AddStudent]

[dbo].[StudentDiscount]

## *[dbo].[Workshop]*

Tabela zawierająca informacje dotyczące konkretnego warsztatu. Ilość rekordów: 1134

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | WorkshopID | int | 4 | False |
| **FK** | WorkshopTypeID | int | 4 | False |
| **FK** | ConferenceID | int | 4 | False |
| **FK** | ConferenceDayID | int | 4 | False |
| | Number_of_seats | int | 4 | True |
| | Price_Workshop | money | 8 | True |
| | Address | nvarchar(50) | 100 | True |
| | Start_time | datetime | 8 | True |
| | End_time | datetime | 8 | True |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_Workshop | WorkshopID | True |
| | ConferenceDayID | ConferenceDayID | |
| | ConferenceID | ConferenceID | |
| | IX_Workshop | WorkshopTypeID | |

### Foreign Keys

| Name | Columns |
|---|---|
| **FK_Workshop_Conference** | ConferenceID->[dbo].[Conference].[ConferenceID] |
| **FK_Workshop_ConferenceDay** | ConferenceDayID->[dbo].[ConferenceDay].[ConferenceDayID] |
| **FK_Workshop_WorkshopType** | WorkshopTypeID->[dbo].[WorkshopType].[WorkshopTypeID] |

## SQL Script

```
CREATE TABLE [dbo].[Workshop]

([WorkshopID] [int] NOT NULL,

[WorkshopTypeID] [int] NOT NULL,

[ConferenceID] [int] NOT NULL,

[ConferenceDayID] [int] NOT NULL,

[Number_of_seats] [int] NULL,

[Price_Workshop] [money] NULL,

[Address] [nvarchar] (50) COLLATE Polish_CI_AS NULL,

[Start_time] [datetime] NULL,

[End_time] [datetime] NULL ) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Workshop] ADD CONSTRAINT [PK_Workshop] PRIMARY KEY CLUSTERED
([WorkshopID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [ConferenceDayID] ON [dbo].[Workshop] ([ConferenceDayID])
ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [ConferenceID] ON [dbo].[Workshop] ([ConferenceID]) ON
[PRIMARY]

GO

CREATE NONCLUSTERED INDEX [IX_Workshop] ON [dbo].[Workshop] ([WorkshopTypeID]) ON
[PRIMARY]

GO

ALTER TABLE [dbo].[Workshop] ADD CONSTRAINT [FK_Workshop_Conference] FOREIGN KEY
([ConferenceID]) REFERENCES [dbo].[Conference] ([ConferenceID])

GO

ALTER TABLE [dbo].[Workshop] ADD CONSTRAINT [FK_Workshop_ConferenceDay] FOREIGN KEY
([ConferenceDayID]) REFERENCES [dbo].[ConferenceDay] ([ConferenceDayID])

GO

ALTER TABLE [dbo].[Workshop] ADD CONSTRAINT [FK_Workshop_WorkshopType] FOREIGN KEY
([WorkshopTypeID]) REFERENCES [dbo].[WorkshopType] ([WorkshopTypeID])

GO
```

## Korzysta z

[dbo].[Conference]

[dbo].[ConferenceDay]

[dbo].[WorkshopType]

## Używane przez

[dbo].[Reservation_Workshop]

[dbo].[Workshop_Participant]

[dbo].[GeneratorWorkshopIndywidualny]

[dbo].[GeneratorWorkshopKompania]

[dbo].[GeneratorWorkshopParticipantsAdding]

[dbo].[PopularWorkshops]

[dbo].[WorkshopParticipants]

[dbo].[WorkshopsInConferenceList]

[dbo].[WorkshopsInDayList]

[dbo].[AddWorkshop]

[dbo].[CollisionsInWorkshops]

[dbo].[FreePlacesForWorkshop]

[dbo].[ListOfWorkshopParticipants]

[dbo].[Reservation_WorkshopValue]

## *[dbo].[Workshop_Participant]*

Tabela zawierająca dane dotyczace uczestników warsztatów. Ilość rekordów: 15189

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | Workshop_ParticipantID | int | 4 | False |
| **FK** | WorkshopID | int | 4 | False |
| **FK** | Conference_ParticipantID | int | 4 | False |
| **FK** | ReservationWorkshopID | int | 4 | False |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_Workshop_Participant | Workshop_ParticipantID | True |
| | WorkshopID | WorkshopID | |

### Foreign Keys

| Name | Columns |
|---|---|
| **FK_Workshop_Participant_Conference_-Participant** | Conference_ParticipantID->[dbo].[Conference_-Participant].[Conference_ParticipantID] |
| **FK_Workshop_Participant_Reservation_-Workshop** | ReservationWorkshopID->[dbo].[Reservation_-Workshop].[Reservation_WorkshopID] |
| **FK_Workshop_Participant_Workshop** | WorkshopID->[dbo].[Workshop].[WorkshopID] |

## SQL Script

```
CREATE TABLE [dbo].[Workshop_Participant]

(

[Workshop_ParticipantID] [int] NOT NULL,

[WorkshopID] [int] NOT NULL,

[Conference_ParticipantID] [int] NOT NULL,

[ReservationWorkshopID] [int] NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Workshop_Participant] ADD CONSTRAINT [PK_Workshop_Participant]
PRIMARY KEY CLUSTERED  ([Workshop_ParticipantID]) ON [PRIMARY]

GO

CREATE NONCLUSTERED INDEX [WorkshopID] ON [dbo].[Workshop_Participant] ([Workshop-
ID]) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Workshop_Participant] ADD CONSTRAINT [FK_Workshop_Participant_-
Conference_Participant] FOREIGN KEY ([Conference_ParticipantID]) REFERENCES
[dbo].[Conference_Participant] ([Conference_ParticipantID])

GO

ALTER TABLE [dbo].[Workshop_Participant] ADD CONSTRAINT [FK_Workshop_Participant_-
Reservation_Workshop] FOREIGN KEY ([ReservationWorkshopID]) REFERENCES
[dbo].[Reservation_Workshop] ([Reservation_WorkshopID])

GO

ALTER TABLE [dbo].[Workshop_Participant] ADD CONSTRAINT [FK_Workshop_Participant_-
Workshop] FOREIGN KEY ([WorkshopID]) REFERENCES [dbo].[Workshop] ([WorkshopID])

GO
```

## Korzysta z

[dbo].[Conference_Participant]

[dbo].[Reservation_Workshop]

[dbo].[Workshop]

## Używane przez

[dbo].[WorkshopParticipants]

## *[dbo].[WorkshopType]*

Tabela zawierajaca nazwy warsztatów. Ilość rekordów: 33

### Columns

| Key | Name | Data Type | Max Length (Bytes) | Allow Nulls |
|---|---|---|---|---|
| **PK** | WorkshopTypeID | int | 4 | False |
| | WorkshopName | nvarchar(50) | 100 | False |

### Indexes

| Key | Name | Key Columns | Unique |
|---|---|---|---|
| **PK** | PK_WorkshopType | WorkshopTypeID | True |

### SQL Script

```
CREATE TABLE [dbo].[WorkshopType]

(

[WorkshopTypeID] [int] NOT NULL,

[WorkshopName] [nvarchar] (50) COLLATE Polish_CI_AS NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[WorkshopType] ADD CONSTRAINT [PK_WorkshopType] PRIMARY KEY
CLUSTERED  ([WorkshopTypeID]) ON [PRIMARY]

GO
```

### Używane przez

[dbo].[Workshop]

[dbo].[PopularWorkshops]

[dbo].[WorkshopParticipants]

[dbo].[WorkshopsInConferenceList]

[dbo].[WorkshopsInDayList]

[dbo].[AddWorkshop]

# Procedury

| Name |
| --- |
| **dbo.AddConference** |
| **dbo.AddConferenceDay** |
| **dbo.AddConferenceParticipant** |
| **dbo.AddCustomerCompany** |
| **dbo.AddCustomerIndividual** |
| **dbo.AddDiscount** |
| **dbo.AddReservation_Conference** |
| **dbo.AddReservationDetails** |
| **dbo.AddReservationWorkshop** |
| **dbo.AddStudent** |
| **dbo.AddWorkshop** |
| **dbo.AddWorkshopParticipant** |
| **dbo.CancelReservation_Conference** |
| **dbo.ChangeCompanyData** |
| **dbo.ChangeIndividualCustomerData** |
| **dbo.UpdatePaymentDate** |

## *[dbo].[AddConference]*

Procedura dodająca nową konferencje wraz z jej specyfikacjami.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@Name** | nvarchar(50) | 100 |
| **@StartDate** | date | 3 |
| **@EndDate** | date | 3 |
| **@NUmberOfSeats** | int | 4 |
| **@StudentDiscount** | real | 4 |
| **@Address** | varchar(50) | 50 |
| **@CountryName** | varchar(50) | 50 |
| **@CityName** | varchar(50) | 50 |
| **@Price** | money | 8 |

### SQL Script

```
CREATE PROCEDURE [dbo].[AddConference]
    @Name NVARCHAR(50),
    @StartDate DATE,
    @EndDate DATE,
    @NUmberOfSeats INT,
    @StudentDiscount REAL,
    @Address VARCHAR(50),
    @CountryName VARCHAR(50),
    @CityName VARCHAR(50),
    @Price MONEY
    AS
    BEGIN
    SET NOCOUNT ON;
        IF ( @StartDate > @EndDate)
        BEGIN;
            THROW 51000, 'EndDate should not be earlier than StartDate.', 1
        END
```

```
        IF ( @StudentDiscount < 0 OR @StudentDiscount > 1)
        BEGIN ;
            THROW 51000, 'The discount must be between 0 and 1.',1
        END

        IF ( @StartDate < GETDATE())
        BEGIN;
            THROW 51000, 'Conference cannot start in the past.', 1
        END

        IF @CountryName NOT IN (SELECT CountryName FROM Country)
BEGIN
DECLARE @CountryNewID INT
SET  @CountryNewID = (SELECT TOP 1 CountryID
                    FROM Country
                     ORDER BY CountryID DESC)+1
INSERT INTO Country(CountryID,CountryName)
VALUES( @CountryNewID,@CountryName)
END

IF @CityName NOT IN (SELECT CityName FROM CityName)
BEGIN
DECLARE @CityNewID INT
SET  @CityNewID = (SELECT TOP 1 CityID
                    FROM CityName
                     ORDER BY CityID DESC)+1
INSERT INTO CityName(CityID,CityName)
VALUES( @CityNewID,@CityName)
END
```

```
        DECLARE @CountryID INT
        SET @CountryID = (
            SELECT CountryID
            FROM Country
            WHERE CountryName = @CountryName
        )
        DECLARE @CityID INT
        SET @CityID = (
            SELECT CityID
            FROM CityName
            WHERE CityName = @CityName
        )
        DECLARE @ConferenceID INT
        SET @ConferenceID = (SELECT TOP 1 ConferenceID
                             FROM Conference
                             ORDER BY conferenceid DESC) + 1
    INSERT INTO Conference(ConferenceID, ConferenceName, Number_of_seats,
Address,CityID,CountryID,StartDate,EndDate, StudentDiscount, Price)
    VALUES (@ConferenceID, @Name, @NumberOfSeats, @Address, @CityID, @CountryID,
@StartDate, @EndDate, @StudentDiscount, @Price)
END


GO
```

**Używa**

[dbo].[CityName]

[dbo].[Conference]

[dbo].[Country]

## *[dbo].[AddConferenceDay]*

Procedura dodająca dzień konferencji dla konkretnej konferencji.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@ConferenceID** | int | 4 |

### SQL Script

```
CREATE PROCEDURE [dbo].[AddConferenceDay]
    @ConferenceID int


    AS
    BEGIN
    SET NOCOUNT ON;


        DECLARE @StartDate date
        DECLARE @EndDate date


        SET @EndDate = (
        SELECT EndDate from
        Conference
        WHERE ConferenceID=@ConferenceID)


        SET @StartDate = (
        SELECT StartDate from
        Conference
        WHERE ConferenceID=@ConferenceID)


        DECLARE @Day int
        DECLARE @Date date
```

```
        SET @Date = @StartDate

        DECLARE  @ConferenceDayID int


        WHILE (@Date<=@EndDate)

        BEGIN

        SET @Day = day(@Date)

        SET @ConferenceDayID =concat(@Day,0,@ConferenceID)

        INSERT INTO ConferenceDay(ConferenceDayID,ConferenceID,DateConferenceDay)

        VALUES (@ConferenceDayID,@ConferenceID,@Date)

        SET @Date = DATEADD(day,1,@Date)

        END

    END

GO
```

### Używa

[dbo].[Conference]

[dbo].[ConferenceDay]

## *[dbo].[AddConferenceParticipant]*

Procedura dodająca uczestnika konferencji.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@ReservationID** | int | 4 |
| **@ConferenceDayID** | int | 4 |
| **@FirstName** | nvarchar(50) | 100 |
| **@LastName** | nvarchar(50) | 100 |
| **@StudentCardID** | int | 4 |
| **@Expiration** | date | 3 |
| **@IsStudent** | bit | 1 |

### SQL Script

```
CREATE PROCEDURE [dbo].[AddConferenceParticipant]

@ReservationID INT,

@ConferenceDayID INT,

@FirstName NVARCHAR(50),

@LastName NVARCHAR(50),

@StudentCardID INT,

@Expiration DATE,

@IsStudent BIT

AS

BEGIN


SET NOCOUNT ON;


DECLARE @CompanyName NVARCHAR(50)

DECLARE @CustomerID INT


SET @CustomerID = (SELECT CustomerID FROM Reservation_Conference

                    WHERE ReservationID = @ReservationID)
```

```
IF (@CustomerID IN (SELECT IndividualCustomerID FROM IndividualCustomer))
    BEGIN;
        SET @CompanyName = NULL
        SET @IsStudent = (SELECT IndividualCustomerID FROM IndividualCustomer
                          WHERE IndividualCustomerID = @CustomerID)
    END
ELSE
    BEGIN;
        SET @CompanyName = (SELECT CompanyName FROM Company
                          WHERE CompanyID = @CustomerID)
    END

DECLARE @ConferenceParticipantID INT
SET @ConferenceParticipantID = (SELECT TOP 1 Conference_ParticipantID FROM
Conference_Participant
                               ORDER BY Conference_ParticipantID DESC) + 1

DECLARE @ConferenceID INT
SET @ConferenceID = (SELECT ConferenceID FROM Reservation_Conference
                    WHERE ReservationID = @ReservationID)

DECLARE @AlreadyRegistered INT
SET @AlreadyRegistered = (SELECT COUNT(@ConferenceParticipantID) FROM
dbo.Conference_Participant
                          WHERE ReservationID = @ReservationID
                          AND ConferenceDayID = @ConferenceDayID)

DECLARE @ReservationDetailsID INT
SET @ReservationDetailsID = (SELECT @ReservationDetailsID FROM dbo.Reservation-
Details
                        WHERE ConferenceDayID = ConferenceDayID
```

```
                            AND ReservationID = @ReservationID)

IF ((SELECT NumberOfPeople FROM dbo.ReservationDetails

    WHERE ReservationDetailsID = @ReservationDetailsID) <= @AlreadyRegistered )

    BEGIN;

        THROW 51000, 'You cannot add another participant because you have already registered
all of reserved places', 0

    END



INSERT INTO Conference_Participant(Conference_ParticipantID, ReservationID, ConferenceID,
FirstName, LastName, Companyname)

VALUES(@ConferenceParticipantID, @ReservationID, @ConferenceID, @FirstName, @LastName,
@CompanyName)



IF (@IsStudent =1 AND @Expiration IS NULL)

    BEGIN;

        THROW 51000, 'If participant is a student you need to add the epiration date of
Student ID.', 0

    END



    IF (@IsStudent =1 AND @StudentCardID IS NULL)

    BEGIN;

        THROW 51000, 'If participant is a student you need to add Student ID number.', 0

    END



IF(@IsStudent =1 AND @StudentCardID IS NOT NULL AND @Expiration IS NOT NULL)

BEGIN;

    INSERT INTO Students(StudentID, Student_CardID, Expiration_Date)
VALUES(@ConferenceParticipantID, @StudentCardID, @Expiration)

END

END

GO
```

**Używa**

[dbo].[Company]

[dbo].[Conference_Participant]

[dbo].[IndividualCustomer]

[dbo].[Reservation_Conference]

[dbo].[ReservationDetails]

[dbo].[Students]

## *[dbo].[AddCustomerCompany]*

Procedura dodająca klienta, który jest firmą.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@CompanyName** | nvarchar(50) | 100 |
| **@Phone** | nvarchar(50) | 100 |
| **@Email** | nvarchar(50) | 100 |
| **@Address** | nvarchar(50) | 100 |
| **@CityName** | nvarchar(50) | 100 |
| **@CountryName** | nvarchar(50) | 100 |

### SQL Script

```
CREATE PROCEDURE [dbo].[AddCustomerCompany]

@CompanyName NVARCHAR(50),@Phone NVARCHAR(50) ,@Email NVARCHAR(50), @Address
NVARCHAR(50),@CityName NVARCHAR(50),@CountryName NVARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @CustomerID int

        set  @CustomerID = (Select top 1 CustomerID

                            from dbo.Customers

                           order by CustomerID desc) + 1

IF @CompanyName IN (SELECT Companyname FROM dbo.Company)

BEGIN;

       THROW 52000, 'This Company already exists in our database.', 1

END


IF @CountryName NOT IN (SELECT CountryName FROM Country)

BEGIN

DECLARE @CountryNewID INT

SET  @CountryNewID = (SELECT TOP 1 CountryID

                     FROM Country
```

```
                    ORDER BY CountryID DESC)+1
INSERT INTO Country(CountryID,CountryName)
VALUES( @CountryNewID,@CountryName)
END
IF @CityName NOT IN (SELECT CityName FROM CityName)
BEGIN
DECLARE @CityNewID INT
SET  @CityNewID = (SELECT TOP 1 CityID
                    FROM CityName
                     ORDER BY CityID DESC)+1
INSERT INTO CityName(CityID,CityName)
VALUES( @CityNewID,@CityName)
END
DECLARE @CountryID INT
        SET @CountryID = (
            SELECT CountryID
            FROM Country
            WHERE CountryName = @CountryName
        )
        DECLARE @CityID INT
        SET @CityID = (
            SELECT CityID
            FROM CityName
            WHERE CityName = @CityName
        )
INSERT INTO Customers(CustomerID)
VALUES(@CustomerID)

INSERT INTO Company(CompanyID,CompanyName,Phone,Email,Address,CityID,CountryID)
VALUES(@CustomerID,@CompanyName ,@Phone,@Email,@Address,@CityID,@CountryID)

END
GO
```

**Używa**

[dbo].[CityName]

[dbo].[Company]

[dbo].[Country]

[dbo].[Customers]

## *[dbo].[AddCustomerIndividual]*

Procedura dodająca klienta indywidualnego.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@IsStudent** | bit | 1 |
| **@FirstName** | nvarchar(50) | 100 |
| **@LastName** | nvarchar(50) | 100 |
| **@Address** | nvarchar(50) | 100 |
| **@Phone** | nvarchar(50) | 100 |
| **@CityName** | nvarchar(50) | 100 |
| **@CountryName** | nvarchar(50) | 100 |
| **@Email** | nvarchar(50) | 100 |

### SQL Script

```
CREATE PROCEDURE [dbo].[AddCustomerIndividual]

@IsStudent BIT, @FirstName NVARCHAR(50),@LastName NVARCHAR(50), @Address
NVARCHAR(50),@Phone NVARCHAR(50) ,@CityName NVARCHAR(50),@CountryName
NVARCHAR(50),@Email NVARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @CustomerID int

        set  @CustomerID = (Select top 1 CustomerID

                            from Customers

                            order by CustomerID desc) + 1

IF @CountryName NOT IN (SELECT CountryName FROM Country)

BEGIN

DECLARE @CountryNewID INT

SET  @CountryNewID = (SELECT TOP 1 CountryID

                    FROM Country

                     ORDER BY CountryID DESC)+1

INSERT INTO Country(CountryID,CountryName)

VALUES( @CountryNewID,@CountryName)
```

```
END


IF @CityName NOT IN (SELECT CityName FROM CityName)
BEGIN
DECLARE @CityNewID INT
SET  @CityNewID = (SELECT TOP 1 CityID
                    FROM CityName
                     ORDER BY CityID DESC)+1
INSERT INTO CityName(CityID,CityName)
VALUES( @CityNewID,@CityName)
END
DECLARE @CountryID INT
        SET @CountryID = (
            SELECT CountryID
            FROM Country
            WHERE CountryName = @CountryName
        )
        DECLARE @CityID INT
        SET @CityID = (
            SELECT CityID
            FROM CityName
            WHERE CityName = @CityName
        )
    DECLARE @IndividualCustomerID INT
        SET @IndividualCustomerID = (
        SELECT TOP 1 CustomerID
        FROM Customers
        ORDER BY CustomerID DESC )+1



INSERT INTO Customers(CustomerID)
```

```
VALUES(@CustomerID)

INSERT INTO IndividualCustomer(IndividualCustomerID,FirstName,Last-
Name,Address,Phone,CityID,CountryID,Email)

VALUES( @IndividualCustomerID,@FirstName ,@LastName,@Address,@Phone,@City-
ID,@CountryID,@Email)

END

GO
```

### Używa

[dbo].[CityName]

[dbo].[Country]

[dbo].[Customers]

[dbo].[IndividualCustomer]

## *[dbo].[AddDiscount]*

Procedura dodającą zniżki do tabeli Discounts dla konkretnej konferencji.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@ConferenceID** | int | 4 |
| **@DateDifference1** | int | 4 |
| **@Discount1** | real | 4 |
| **@DateDifference2** | int | 4 |
| **@Discount2** | real | 4 |
| **@DateDifference3** | int | 4 |
| **@Discount3** | real | 4 |

### SQL Script

```
CREATE PROCEDURE [dbo].[AddDiscount]

@ConferenceID INT,

@DateDifference1 INT,

@Discount1 REAL,

@DateDifference2 INT,

@Discount2 REAL,

@DateDifference3 INT,

@Discount3 REAL

AS

BEGIN

SET NOCOUNT ON;

    IF @DateDifference1<0 OR @DateDifference2<0 OR @DateDifference3<0

    BEGIN;

            THROW 52000, 'Date difference need to be positive .', 1

    END


    IF @Discount1>1 OR @Discount2>1 OR @Discount3>1
```

```
    BEGIN;
        THROW 52000, 'Discount need to be smaller than 1 .', 1
    END

    IF @ConferenceID NOT IN (SELECT ConferenceID FROM dbo.Conference)
    BEGIN;
    THROW 52000, 'There is not such ConferenceID in Conference Table.', 1
    END

DECLARE @DiscountID1 INT
SET @DiscountID1=(select top 1 DiscountID
                        from dbo.Discounts
                        order by DiscountID desc) + 1
INSERT INTO dbo.Discounts( DiscountID ,ConferenceID ,DateDifference ,Discount)
VALUES  ( @DiscountID1,@ConferenceID,@DateDifference1,@Discount1)
INSERT INTO dbo.Discounts( DiscountID ,ConferenceID ,DateDifference ,Discount)
VALUES  ( @DiscountID1+1,@ConferenceID,@DateDifference2,@Discount2)

INSERT INTO dbo.Discounts( DiscountID ,ConferenceID ,DateDifference ,Discount)
VALUES  ( @DiscountID1+2,@ConferenceID,@DateDifference3,@Discount3)
END
GO
```

**Używa**

[dbo].[Conference]

[dbo].[Discounts]

## *[dbo].[AddReservation_Conference]*

Procedura dodająca reserwacje na konkretną konferencje.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@ConferenceID** | int | 4 |
| **@CustomerID** | int | 4 |

### SQL Script

```
Create PROCEDURE [dbo].[AddReservation_Conference]

@ConferenceID int,

@CustomerID int



AS

BEGIN

SET NOCOUNT ON;



-- SPRAWDZANIE POPRAWNOŚCI DANYCH

Declare @ConferenceIDCheck int

set @ConferenceIDCheck = (select ConferenceID from conference

                        where conferenceID = @COnferenceID)

IF ( @ConferenceIDCheck is NUll)

        BEGIN;

            THROW 52000, 'There is no such conference.', 1

        END



Declare @CustomerIDCheck int

set @CustomerIDCheck = (select CustomerID from Customers

                        where CustomerID = @CustomerID)
```

```
IF ( @CustomerIDCheck is NUll)

        BEGIN;

            THROW 52000, 'This Customer does not exist', 1

        END

-----------------------------------------------

DECLARE @ReservationID int

set @ReservationID = (select top 1 ReservationID

                        from Reservation_Conference

                        order by ReservationID desc) + 1


INSERT INTO Reservation_Conference(ReservationID, CustomerID,ConferenceID,DateOf-
Reservation,DateOfPayment, Cancelled)

VALUES(@ReservationID, @CustomerID,@ConferenceID, GETDATE(), NUll, 0)

END

GO
```

**Używa**

[dbo].[Conference]

[dbo].[Customers]

[dbo].[Reservation_Conference]

## *[dbo].[AddReservationDetails]*

Procedura dodająca dane dotyczące konkretnej rezerwacji takie jak: liczba miejsc na dany dzień konferencji oraz liczba studentów.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@ReservationID** | int | 4 |
| **@NumberOfDay** | int | 4 |
| **@NumberOfPeople** | int | 4 |
| **@NumberOfStudents** | int | 4 |

### SQL Script

```
CREATE procedure [dbo].[AddReservationDetails]
@ReservationID int,
@NumberOfDay int,
@NumberOfPeople int,
@NumberOfStudents int
AS
BEGIN
SET NOCOUNT ON;

--Conference id wyciągam z rezarwacji. przyda mi się 2 razy
Declare @ConferenceID int
SET @ConferenceID =(select ConferenceID
    from Reservation_Conference
    where ReservationID = @ReservationID)



--
```

```
DECLARE @ConferenceDayID int

SET @ConferenceDayID = (select ConferenceDayID

      from ConferenceDay

      where ConferenceID = @ConferenceID

      and DateConferenceDay = (select DateAdd(DD,  @NumberOfDay -1, startDate)

            from Conference

            where ConferenceID = @ConferenceID))


--dodawanie indywidualnego klienta jako participanta conf

DECLARE @CustomerID int

SET @CustomerID = (select CustomerID from Reservation_Conference

     where ConferenceID = @ConferenceID)

if (@CustomerID in (select IndividualCustomerID from IndividualCustomer

    where @ConferenceID = IndividualCustomerID) AND @NumberOfPeople>1)

 BEGIN;

 THROW 52000, 'Individual Customer can not be multiplied, unless he or she is a Holy Trinity
then

 Individual Customer can be three people at once.', 1

END


--SPRAWDZANIE DANYCH

Declare @PeopleInConference int

SET @PeopleInConference = (dbo.ConferenceDayFreePlaces(@ConferenceDayID))

Declare @EndDate date

Set @EndDate = (select EndDate from Conference

    where ConferenceID = @ConferenceID)

IF @NumberOfPeople < @NumberOfStudents

 Begin;

  THROW 52000, 'Students are people too! Number of students must be smaller than number of
people.', 1

 END

IF @NumberOfPeople > (select Number_Of_Seats from Conference

      where conferenceID = @ConferenceID)

 Begin;
```

```
  THROW 52000, 'Number of people must be smaller than number of seats in conference.', 1
 END


IF ( @EndDate < (select DateAdd(DD,  @NumberOfDay -1, startDate)
            from Conference
            where ConferenceID = @ConferenceID))
 BEGIN;
  THROW 51000, 'There is less days in a conference!', 1
 END


IF ( @NumberOfPeople > @PeopleInConference)
 BEGIN;
  THROW 51000, 'Number of seats you want to reserve is bigger than number of places left for
this conference.', 1
 END


Declare @ReservationIDCheck int
set @ReservationIDCheck = (select ReservationID from Reservation_Conference
       where ReservationID = @ReservationID)
IF ( @ReservationIDCheck is NUll)
 BEGIN;
  THROW 51000, 'There is no such conference.', 1
 END
-----------------------------------------------------
DECLARE @ReservationDetailsID  int
SET @ReservationDetailsID = (select top 1 ReservationDetailsID
      from ReservationDetails
      order by ReservationDetailsID  desc) + 1


INSERT INTO ReservationDetails(ReservationDetailsID,ReservationID,ConferenceDayID,NumberOf-
People,Cancelled)
VALUES( @ReservationDetailsID, @ReservationID, @ConferenceDayID,@NumberOfPeople, 0)
END
GO
```

**Używa**

[dbo].[Conference]

[dbo].[ConferenceDay]

[dbo].[IndividualCustomer]

[dbo].[Reservation_Conference]

[dbo].[ReservationDetails]

[dbo].[ConferenceDayFreePlaces]

## *[dbo].[AddReservationWorkshop]*

Procedura dodająca rezerwacje na konkretny warsztat odbywający się w zarezerwowanej już konferencji.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@NumberOfPeople** | int | 4 |
| **@ReservationID** | int | 4 |
| **@WorkshopID** | int | 4 |

### SQL Script

```
CREATE PROCEDURE [dbo].[AddReservationWorkshop]

@NumberOfPeople INT,

@ReservationID INT,

@WorkshopID INT




AS

BEGIN

SET NOCOUNT ON;


IF (@ReservationID NOT IN (SELECT ReservationID FROM dbo.Reservation_Conference))

    BEGIN;

        THROW 52000, 'There is no reservation with this ID.', 1

    END


DECLARE @ReservationDetailsID INT

SET @ReservationDetailsID = (SELECT ReservationDetailsID FROM dbo.Reservation-
Details

                            WHERE ConferenceDayID = (SELECT ConferenceDayID FROM
dbo.Workshop

                                                     WHERE WorkshopID =
@WorkshopID)
```

```
                            AND @ReservationID = ReservationID)



IF @ReservationDetailsID IS NULL

    BEGIN;

        THROW 52000, 'There is no reservation on this day.', 1

    END


DECLARE @CustomerID INT

SET @CustomerID = (SELECT CustomerID FROM dbo.Reservation_Conference)


IF (@CustomerID IN (SELECT IndividualCustomerID FROM dbo.IndividualCustomer) AND
@NumberOfPeople > 1)

    BEGIN;

        THROW 52000, 'Individual customer can reserve only one place in workshop.',
1

    END


DECLARE @NumberOfPeopleForDay int

SET @NumberOfPeopleForDay=(SELECT NumberOfPeople FROM dbo.ReservationDetails

                    WHERE ReservationDetailsID=@ReservationDetailsID)


IF @NumberOfPeople>@NumberOfPeopleForDay

    BEGIN;

        THROW 52000, 'You do not booked enough seats for that day.', 1

    END




DECLARE @TakenPlaces INT

SET @TakenPlaces = (SELECT SUM(NumberOfPeople) FROM Reservation_Workshop AS rw

                            WHERE rw.workshopID = @workshopID)
```

```
DECLARE @AllPlaces INT
SET @AllPlaces = (SELECT W.number_of_seats FROM Workshop w
                    WHERE @WorkshopID = w.WorkshopID)

DECLARE @FreePlaces INT
SET @FreePlaces = @AllPlaces - @TakenPlaces
IF (@freePlaces < @NumberOfPeople)
    BEGIN;
        THROW 52000, 'Number of free places is smaller than number of people you
want to register for this workshop.', 1
    END
-----------------------------------------------
DECLARE @ReservationWorkshopID INT
SET @ReservationWorkshopID = (SELECT TOP 1 Reservation_WorkshopID
                        FROM Reservation_Workshop
                        ORDER BY Reservation_WorkshopID DESC) + 1

INSERT INTO Reservation_Workshop(Reservation_WorkshopID, ReservationID,Workshop-
ID,NumberOfPeople, Cancelled)
VALUES(@ReservationWorkshopID,@ReservationID,@WorkshopID, @NumberOfPeople, 0)
END
GO
```

**Używa**

[dbo].[IndividualCustomer]

[dbo].[Reservation_Conference]

[dbo].[Reservation_Workshop]

[dbo].[ReservationDetails]

[dbo].[Workshop]

## *[dbo].[AddWorkshop]*

Procedura dodająca warsztat wraz z jego dokładnymi specyfikacjami

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@ConferenceID** | int | 4 |
| **@WorkshopName** | nvarchar(50) | 100 |
| **@ConferenceDayID** | int | 4 |
| **@Number_of_seats** | int | 4 |
| **@Price_Workshop** | money | 8 |
| **@Address** | nvarchar(50) | 100 |
| **@Start_time** | datetime | 8 |
| **@End_time** | datetime | 8 |

### SQL Script

```
CREATE PROCEDURE [dbo].[AddWorkshop]

    @ConferenceID int,

    @WorkshopName nvarchar(50),

    @ConferenceDayID int,

    @Number_of_seats int,

    @Price_Workshop money,

    @Address nvarchar(50),

    @Start_time datetime,

    @End_time datetime


    AS

    BEGIN

    SET NOCOUNT ON;

        IF ( @Start_time > @End_time)

        BEGIN;

            THROW 51000, 'EndDate should not be earlier than StartDate.', 1

        END
```

```
        Declare @StartDate date
        set @startDate = (select StartDate from Conference
                          where ConferenceID = @ConferenceID)
        Declare @EndDate Date
        set @endDate = (select EndDate from Conference
                          where ConferenceID = @ConferenceID)
        IF ( @Start_time < @StartDate)
        Declare @NumberOfPeopleInConference int
        set @NumberOfPeopleInConference = (select Number_of_seats from Conference
                                           where ConferenceID = @ConferenceID)
        IF ( @Start_time < @StartDate)
        BEGIN;
            THROW 51000, 'Start_time should be contained in Conference time.', 1
        END


        IF ( @EndDate < @End_time)
        BEGIN;
            THROW 51000, 'End_time should be contained in Conference time.', 1
        END


        IF ( @NumberOfPeopleInConference < @Number_of_seats)
        BEGIN;
            THROW 51000, 'Number of people on workshop has to be smaller than
number of people on conference', 1
        END


        IF @WorkshopName not in (select WorkshopName from WorkshopType)
        BEGIN
        DECLARE @WorkshopNewTypeID int
        SET @WorkshopNewTypeID = (Select top 1 WorkshopTypeID
```

```
                            from WorkshopType
                            order by WorkshopTypeID desc) + 1
        INSERT INTO WorkshopType(WorkshopTypeID ,WorkshopName)
        VALUES (@WorkshopNewTypeID,@WorkshopName)
        END
        DECLARE @WorkshopTypeID int
        SET  @WorkshopTypeID = (SELECT WorkshopTypeID
                                FROM WorkshopType
                                WHERE @WorkshopName = WorkshopName)



        DECLARE @WorkshopID int
        set @WorkshopID = (Select top 1 WorkshopID
                            from Workshop
                            order by WorkshopID desc) + 1
    INSERT INTO Workshop(WorkshopID, WorkshopTypeID,ConferenceID,ConferenceDayID,
Number_of_seats,Price_Workshop, Address,Start_time,End_time)
    VALUES (@WorkshopID, @WorkshopTypeID, @ConferenceID,@ConferenceDay-
ID,@Number_of_seats,@Price_Workshop, @Address,@Start_time, @End_time )
END
GO
```

**Używa**

[dbo].[Conference]

[dbo].[Workshop]

[dbo].[WorkshopType]

## *[dbo].[AddWorkshopParticipant]*

Procedeura dodająca uczestnika konkretnego warsztatu.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@ReservationWorkshopID** | int | 4 |
| **@ConferenceParticipantID** | int | 4 |

### SQL Script

```
CREATE PROCEDURE [dbo].[AddWorkshopParticipant]
    @ReservationWorkshopID INT,
    @ConferenceParticipantID INT
AS
BEGIN
SET NOCOUNT ON;
    DECLARE @Conference_ParticipantID int
    DECLARE @Start_Time DATETIME
    DECLARE @End_Time DATETIME
    DECLARE @Work_shopID INT



    IF @ReservationWorkshopID NOT IN (SELECT Reservation_WorkshopID FROM
dbo.Reservation_Workshop)
        BEGIN;
            THROW 51000, 'This reservation of workshop does not exist', 1
        END


    IF @ConferenceParticipantID NOT IN (SELECT Conference_ParticipantID FROM
dbo.Conference_Participant)
```

```
        BEGIN;
            THROW 52000, 'This conference participant does not exists', 1
        END

    DECLARE @WorkshopID INT
    SET @WorkshopID = (SELECT WorkshopID FROM dbo.Reservation_Workshop
                       WHERE Reservation_WorkshopID = @ReservationWorkshopID)

    DECLARE @WorkshopParticipantID INT
    SET @WorkshopParticipantID = (SELECT TOP 1 Workshop_ParticipantID FROM
dbo.Workshop_Participant
                                  ORDER BY Workshop_ParticipantID DESC) + 1

-- sprawdzanie czy ktoś nie próbuje dodać wiecej uczestników niz zareklarował w
rezerwacji warsztatu
    DECLARE @AlreadyRegistered INT
    SET @AlreadyRegistered = (SELECT SUM(Workshop_ParticipantID) FROM
dbo.Workshop_Participant WP
                              INNER JOIN dbo.Reservation_Workshop Rw
                              ON rw.Reservation_WorkshopID = wp.ReservationWorkshopID
                              WHERE rw.Reservation_WorkshopID = @ReservationWorkshop-
ID)

    IF @AlreadyRegistered >= (SELECT NumberOfPeople FROM dbo.Reservation_Workshop
                              WHERE Reservation_WorkshopID= @ReservationWorkshopID)
    BEGIN;
        THROW 52000, 'You have already registered all participans from this
reservation.', 1
    END

-- sprawzdanie czy warsztaty nie kolidują danemu uczestnikowi
    DECLARE @StartTime DATETIME
```

```
    SET @StartTime = (SELECT Start_time FROM dbo.Workshop
                        WHERE WorkshopID = @WorkshopID)


    DECLARE @EndTime DATETIME
    SET @EndTime = (SELECT End_time FROM dbo.Workshop
                        WHERE WorkshopID = @WorkshopID)


    DECLARE @Collision BIT
    SET @Collision = 0


/*deklarujemy kursor*/
DECLARE moj_kursor CURSOR LOCAL FOR
/*deklarujemy zrodlo kursora*/
SELECT Conference_ParticipantID, wp.WorkshopID, w.Start_Time, w.End_Time FROM
dbo.Workshop w
INNER JOIN dbo.Workshop_Participant wp
ON wp.WorkshopID = w.WorkshopID
WHERE Conference_ParticipantID = 3698
AND wp.WorkshopID = w.WorkshopID
/*otwieramy kursor*/
OPEN moj_kursor
/*przypisujemy pierwszy wiersz do zmiennych o typach zgodnych z kolumnami w
zrodle*/
FETCH NEXT FROM moj_kursor INTO @Conference_ParticipantID, @Work_shopID, @Start_-
Time,@End_Time
/*jezeli przypisanie sie powiodlo wchodzimy w petle*/
WHILE @@FETCH_STATUS = 0
BEGIN
/*przykladowa czynnosc na danych*/
IF (@StartTime < @End_Time OR @EndTime > @Start_Time)
    BEGIN;
        THROW 51000, 'This workshop collides with the one you have already
registered for.', 1
```

```
    END
/*przypisanie kolejnego wiersza*/
  FETCH NEXT FROM moj_kursor INTO @Conference_ParticipantID, @WorkshopID, @Start_-
Time,@End_Time
END
/*zamkniecie kursora*/
CLOSE moj_kursor
/*zwolnienie kursora*/
DEALLOCATE moj_kursor


INSERT INTO dbo.Workshop_Participant
        ( Workshop_ParticipantID ,
          WorkshopID ,
          Conference_ParticipantID ,
          ReservationWorkshopID
        )
VALUES  ( @WorkshopParticipantID , -- Workshop_ParticipantID - int
          @WorkshopID , -- WorkshopID - int
          @ConferenceParticipantID , -- Conference_ParticipantID - int
          @ReservationWorkshopID  -- ReservationWorkshopID - int
        )
END
GO
```

**Używa**

[dbo].[Conference_Participant]

[dbo].[Reservation_Workshop]

[dbo].[Workshop]

[dbo].[Workshop_Participant]

## *[dbo].[CancelReservation_Conference]*

Procedura anulowania rezerwacji.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@Reservation_ConferenceID** | int | 4 |

### SQL Script

```
Create PROCEDURE [dbo].[CancelReservation_Conference]

@Reservation_ConferenceID int

AS

BEGIN

SET NOCOUNT ON;

--SPRAWDZANIE DANYCH

Declare @ConfIDCheck int

set @ConfIDCheck = (select ReservationID from Reservation_Conference

                    where ReservationID = @Reservation_ConferenceID)

IF ( @ConfIDCheck is NULL)

        BEGIN;

            THROW 51000, 'There is no such conference',1

        END

-------------------------------------

UPDATE Reservation_Conference

SET Cancelled = 1

WHERE ReservationID = @Reservation_ConferenceID

END

GO
```

### Używa

[dbo].[Reservation_Conference]

## *[dbo].[ChangeCompanyData]*

Procedura zmieniająca specyfikacje dotyczące konkretnej konferencji.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@ComanyID** | int | 4 |
| **@NewCompanyName** | nvarchar(50) | 100 |
| **@NewPhone** | nvarchar(50) | 100 |
| **@NewEmail** | nvarchar(50) | 100 |
| **@NewAddress** | nvarchar(50) | 100 |
| **@NewCity** | nvarchar(50) | 100 |
| **@NewCountry** | nvarchar(50) | 100 |

### SQL Script

```
CREATE PROCEDURE [dbo].[ChangeCompanyData]
@ComanyID INT,
@NewCompanyName NVARCHAR(50),
@NewPhone NVARCHAR(50),
@NewEmail NVARCHAR(50),
@NewAddress NVARCHAR(50),
@NewCity NVARCHAR(50),
@NewCountry NVARCHAR(50)
AS
BEGIN
SET NOCOUNT ON;
IF @NewCity IS NOT NULL
    BEGIN;
        DECLARE @CountryID INT
        SET @CountryID = (SELECT CountryID FROM dbo.Country
                        WHERE CountryName LIKE @NewCountry)
```

```
        UPDATE dbo.Company
        SET CountryID = @CountryID
        WHERE CompanyID = @ComanyID

    END


IF @NewCountry IS NOT NULL
    BEGIN;
        DECLARE @CityID INT
        SET @CityID = (SELECT @CityID FROM dbo.CityName
                       WHERE CityName LIKE @NewCity)

        UPDATE dbo.Company
        SET CityID = @CityID
        WHERE CompanyID = @ComanyID
    END

IF @NewCompanyName IS NOT NULL
    BEGIN;
        UPDATE dbo.Company
        SET CompanyName= @NewCompanyName
        WHERE CompanyID = @ComanyID
    END

IF @NewPhone IS NOT NULL
    BEGIN;
        UPDATE dbo.Company
        SET Phone= @NewPhone
        WHERE CompanyID = @ComanyID
    END
```

```
IF @NewEmail IS NOT NULL
    BEGIN;
        UPDATE dbo.Company
        SET Email= @NewEmail
        WHERE CompanyID = @ComanyID
    END

IF @NewAddress IS NOT NULL
    BEGIN;
        UPDATE dbo.Company
        SET Address= @NewAddress
        WHERE CompanyID = @ComanyID
    END
END

GO
```

**Używa**

[dbo].[CityName]

[dbo].[Company]

[dbo].[Country]

## *[dbo].[ChangeIndividualCustomerData]*

Procedura zmieniająca dane dotyczące klienta indywidualnego.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@IndividualCustomerID** | int | 4 |
| **@NewFirstName** | varchar(50) | 50 |
| **@NewLastName** | varchar(50) | 50 |
| **@NewPhone** | nvarchar(50) | 100 |
| **@NewEmail** | nvarchar(50) | 100 |
| **@NewAddress** | nvarchar(50) | 100 |
| **@NewCity** | nvarchar(50) | 100 |
| **@NewCountry** | nvarchar(50) | 100 |
| **@NewIsStudent** | bit | 1 |

### SQL Script

```
CREATE PROCEDURE [dbo].[ChangeIndividualCustomerData]
@IndividualCustomerID INT,
@NewFirstName VARCHAR(50),
@NewLastName VARCHAR(50),
@NewPhone NVARCHAR(50),
@NewEmail NVARCHAR(50),
@NewAddress NVARCHAR(50),
@NewCity NVARCHAR(50),
@NewCountry NVARCHAR(50),
@NewIsStudent BIT

AS
BEGIN
SET NOCOUNT ON;
```

```
IF @NewCity IS NOT NULL
    BEGIN;
        DECLARE @CountryID INT
        SET @CountryID = (SELECT CountryID FROM dbo.Country
                        WHERE CountryName LIKE @NewCountry)

        UPDATE IndividualCustomer
        SET CountryID = @CountryID
        WHERE IndividualCustomerID = @IndividualCustomerID

    END


IF @NewCountry IS NOT NULL
    BEGIN;
        DECLARE @CityID INT
        SET @CityID = (SELECT @CityID FROM dbo.CityName
                        WHERE CityName LIKE @NewCity)

        UPDATE IndividualCustomer
        SET CityID = @CityID
        WHERE IndividualCustomerID = @IndividualCustomerID
    END

IF @NewFirstName IS NOT NULL
    BEGIN;
        UPDATE IndividualCustomer
        SET FirstName = @NewFirstName
        WHERE IndividualCustomerID = @IndividualCustomerID
    END
```

```
IF @NewLastName IS NOT NULL
    BEGIN;
        UPDATE IndividualCustomer
        SET LastName = @NewLastName
        WHERE IndividualCustomerID = @IndividualCustomerID
    END

IF @NewPhone IS NOT NULL
    BEGIN;
        UPDATE IndividualCustomer
        SET Phone= @NewPhone
        WHERE IndividualCustomerID = @IndividualCustomerID
    END

IF @NewEmail IS NOT NULL
    BEGIN;
        UPDATE IndividualCustomer
        SET Email= @NewEmail
        WHERE IndividualCustomerID = @IndividualCustomerID
    END

IF @NewAddress IS NOT NULL
    BEGIN;
        UPDATE IndividualCustomer
        SET Address= @NewAddress
        WHERE IndividualCustomerID = @IndividualCustomerID
    END
```

```
IF @NewIsStudent IS NOT NULL

    BEGIN;

        UPDATE IndividualCustomer

        SET Address= IsStudent

        WHERE IndividualCustomerID = @IndividualCustomerID

    END

END


GO
```

**Używa**

[dbo].[CityName]

[dbo].[Country]

[dbo].[IndividualCustomer]

# *[dbo].[UpdateConferenceNumberOfSeats]*

Procedura aktualizująca liczbę miejsc na konferencji

## Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@NumberOfSeats** | int | 4 |
| **@ConferenceID** | int | 4 |

## SQL Script

```
CREATE PROCEDURE [dbo].[UpdateConferenceNumberOfSeats]
@NumberOfSeats INT,
@ConferenceID INT
AS
BEGIN
SET NOCOUNT ON;

UPDATE dbo.Conference
SET Number_of_seats = @NumberOfSeats
WHERE ConferenceID = @ConferenceID

END

GO
```

## Uses

[dbo].[Conference]

## *[dbo].[UpdatePaymentDate]*

Procedura uaktualnia date dokonania wpłaty.

### Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@DateOfPayment** | datetime | 8 |
| **@Reservation_ConferenceID** | int | 4 |

### SQL Script

```
CREATE PROCEDURE [dbo].[UpdatePaymentDate]
@DateOfPayment DATETIME,
@Reservation_ConferenceID INT


AS
BEGIN
SET NOCOUNT ON;
UPDATE dbo.Reservation_Conference
SET DateOfPayment = @DateOfPayment
WHERE ReservationID = @Reservation_ConferenceID




END
GO
```

### Używa

[dbo].[Reservation_Conference]

# *[dbo].[UpdateWorkshopNumberOfSeats]*

## Parameters

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **@NumberOfSeats** | int | 4 |
| **@WorkshopID** | int | 4 |

## SQL Script

```
CREATE PROCEDURE [dbo].[UpdateWorkshopNumberOfSeats]
@NumberOfSeats INT,
@WorkshopID INT
AS
BEGIN
SET NOCOUNT ON;

UPDATE dbo.Workshop
SET Number_of_seats = @NumberOfSeats
WHERE WorkshopID = @WorkshopID

END

GO
```

## Uses

[dbo].[Workshop]

# Views

| Name |
| --- |
| **dbo.CancelledReservations** |
| **dbo.ConferenceDayParticipants** |
| **dbo.ConferenceParticipants** |
| **dbo.ConferenceToCancelled** |
| **dbo.CustomersToCall** |
| **dbo.GeneratorCompany** |
| **dbo.GeneratorConference_participantCompany** |
| **dbo.GeneratorParticipants_Individual** |
| **dbo.GeneratorParticipants_IndividualStudent** |
| **dbo.GeneratorReservationDetailsCompany** |
| **dbo.GeneratorReservationDetailsIndividual** |
| **dbo.GeneratorReservationsIDCompany** |
| **dbo.GeneratorReservationsIDIndywidual** |
| **dbo.GeneratorWorkshop** |
| **dbo.GeneratorWorkshopIndywidualny** |
| **dbo.GeneratorWorkshopKompania** |
| **dbo.GeneratorWorkshopParticipantsAdding** |
| **dbo.NumberOfCustomerReservations** |
| **dbo.PopularConferences** |
| **dbo.PopularWorkshops** |
| **dbo.PriceOfReservation** |
| **dbo.RegularCustomers** |
| **dbo.ValueOfReservation** |
| **dbo.WorkshopParticipants** |
| **dbo.WorkshopsInConferenceList** |
| **dbo.WorkshopsInDayList** |

## *[dbo].[CancelledReservations]*

Widok pokazujący odwołane rezerwacje.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **ReservationID** | int | 4 |
| **Customer Name** | nvarchar(101) | 202 |

### SQL Script

```
CREATE VIEW [dbo].[CancelledReservations] AS
SELECT RC.ReservationID, IC.FirstName + ' ' + IC.LastName AS [Customer Name]
FROM Reservation_Conference as RC
JOIN IndividualCustomer as IC
ON IC.IndividualCustomerID = RC.CustomerID
WHERE RC.Cancelled = 1

UNION

SELECT RC.ReservationID, C.CompanyName AS [Customer Name]
FROM Reservation_Conference as RC
JOIN Company as C
ON C.CompanyID = RC.CustomerID
WHERE RC.Cancelled = 1
GO
```

### Używa

[dbo].[Company]

[dbo].[IndividualCustomer]

[dbo].[Reservation_Conference]

## *[dbo].[ConferenceDayParticipants]*

Widok pokazujący liste uczestników danego dnia konferencji.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **ConferenceID** | int | 4 |
| **ConferenceDayID** | int | 4 |
| **Number of Conference Day** | int | 4 |
| **ConferenceName** | nvarchar(50) | 100 |
| **FirstName** | nvarchar(50) | 100 |
| **Lastname** | nvarchar(50) | 100 |
| **Companyname** | nvarchar(50) | 100 |

### SQL Script

```
CREATE VIEW [dbo].[ConferenceDayParticipants]

AS

SELECT c.ConferenceID, cp.ConferenceDayID, DATEDIFF(dd, c.StartDate,CD.Date-
ConferenceDay)+1 AS [Number of Conference Day], c.ConferenceName,cp.FirstName,
cp.Lastname, cp.Companyname

FROM Conference_Participant AS cp

INNER JOIN dbo.ConferenceDay CD

ON CP.ConferenceDayID = CD.ConferenceDayID

INNER JOIN Conference AS c

ON c.ConferenceID = cp.ConferenceID

INNER JOIN dbo.Reservation_Conference RC

ON Cp.ReservationID = RC.ReservationID

WHERE RC.Cancelled = 0

AND c.ConferenceID=cp.ConferenceID

AND cd.ConferenceDayID = cp.conferenceDayID

GO
```

**Używa**

[dbo].[Conference]

[dbo].[Conference_Participant]

[dbo].[ConferenceDay]

[dbo].[Reservation_Conference]

## *[dbo].[ConferenceParticipants]*

Widok pokazujący liste uczestników danej konferencji.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **conferenceID** | int | 4 |
| **ConferenceName** | nvarchar(50) | 100 |
| **FirstName** | nvarchar(50) | 100 |
| **Lastname** | nvarchar(50) | 100 |
| **Companyname** | nvarchar(50) | 100 |

### SQL Script

```
CREATE VIEW [dbo].[ConferenceParticipants]
AS
SELECT c.conferenceID, c.ConferenceName,cp.FirstName, cp.Lastname, cp.Companyname
FROM Conference_Participant AS cp
INNER JOIN Conference AS c
ON c.ConferenceID = cp.ConferenceID
INNER JOIN dbo.Reservation_Conference RC
ON Cp.ReservationID = RC.ReservationID
WHERE RC.Cancelled = 0
AND c.ConferenceID=cp.ConferenceID
GO
```

### Używa

[dbo].[Conference]

[dbo].[Conference_Participant]

[dbo].[Reservation_Conference]

## *[dbo].[ReservationConferenceToCancelled]*

Widok pokazujący rezerwacje, które powinny być anulowane ze względu na brak dokonania wpłaty do 7 dni po dokonaniu rezerwacji.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **ReservationID** | int | 4 |
| **CustomerID** | int | 4 |
| **DateDifference** | int | 4 |

### SQL Script

```
CREATE VIEW [dbo].[ReservationConferenceToCancelled] as


SELECT rc.ReservationID, rc.CustomerID,DATEDIFF(day,rc.DateOfReservation,
GETDATE())as DateDifference

FROM Reservation_Conference as rc

WHERE DATEDIFF(day,rc.DateOfReservation, GETDATE()) > 7 and rc.DateOfPayment is
null

GO
```

### Używa

[dbo].[Reservation_Conference]

## *[dbo].[CustomersToCall]*

Widok pokazujący liste klientów, którzy 14 dni przed konferencją na którą złożyli rezerwacje nie podali list imiennych uczestników biorących w niej udział.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **reservationID** | int | 4 |
| **CustomerID** | int | 4 |
| **Number of Missing Participants** | int | 4 |
| **Phone** | nvarchar(50) | 100 |

### SQL Script

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

SELECT rc.reservationID, CustomerID, dbo.NumberOfPeopleIn-
Reservation(rc.ReservationID) - dbo.NumberOfParticipantsBy-
Reservation(rc.ReservationID) AS [Number of Missing Participants], Phone

FROM dbo.Reservation_Conference AS rc

INNER JOIN Conference AS c

on c.ConferenceID = rc.ConferenceID

INNER JOIN Company AS Comp

ON comp.CompanyID = RC.CustomerID

WHERE DATEADD(dd, 14, GETDATE()) >  c.StartDate

AND Startdate > GETDATE()

AND dbo.NumberOfPeopleInReservation(rc.ReservationID) - dbo.NumberOfParticipantsBy-
Reservation(rc.ReservationID) <> 0


GO
```

**Używa**

[dbo].[NumberOfParticipantsByReservation]

[dbo].[NumberOfPeopleInReservation]

[dbo].[Company]

[dbo].[Conference]

[dbo].[Reservation_Conference]

## *[dbo].[NumberOfCustomerReservations]*

Widok pokazujący liczbę rezerwacji dokonanych przez klientów.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **CustomerID** | int | 4 |
| **NumberOfReservations** | int | 4 |

### SQL Script

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

select CustomerID, count(customerid) as NumberOfReservations from Reservation_-
Conference

group by customerid

GO
```

### Używa

[dbo].[Reservation_Conference]

## *[dbo].[PopularConferences]*

Widok pokazujący liste najpopularniejszych konferencji.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **ConferenceName** | nvarchar(50) | 100 |
| **Number of people** | int | 4 |

### SQL Script

```
CREATE VIEW [dbo].[PopularConferences]
AS
SELECT TOP 100 C.ConferenceName, SUM(NumberOfPeople) AS [Number of people]
FROM Conference AS C
JOIN Reservation_Conference AS RC ON C.ConferenceID = RC.ConferenceID
JOIN ReservationDetails AS RD ON RC.ReservationID = RD.ReservationID

WHERE RC.Cancelled LIKE 0
GROUP BY C.ConferenceName
ORDER BY [Number of people] DESC

GO
```

### Używa

[dbo].[Conference]

[dbo].[Reservation_Conference]

[dbo].[ReservationDetails]

## *[dbo].[PopularWorkshops]*

Widok pokazujący liste najpopularniejszych warsztatów.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **WorkshopName** | nvarchar(50) | 100 |
| **Number of WS participants** | int | 4 |

### SQL Script

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

SELECT TOP 100  WT.WorkshopName, SUM(RW.NumberOfPeople) AS [Number of WS
participants]

FROM Reservation_Workshop AS RW

INNER JOIN Workshop AS W

ON RW.WorkshopID = W.WorkshopID

INNER JOIN WorkshopType AS WT

ON WT.WorkshopTypeID = W.WorkshopID

GROUP BY WT.WorkshopName

ORDER BY [Number of WS participants] DESC

GO
```

### Używa

[dbo].[Reservation_Workshop]

[dbo].[Workshop]

[dbo].[WorkshopType]

## *[dbo].[RegularCustomers]*

Widok zwracający liste stałych klientów.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **CustomerID** | int | 4 |
| **Number of Conferences** | int | 4 |

### SQL Script

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


SELECT TOP 100 CustomerID, COUNT(CustomerID) AS [Number of Conferences] FROM
dbo.Reservation_Conference

GROUP BY CustomerID

ORDER BY COUNT(CustomerID) DESC

GO
```

### Używa

[dbo].[Reservation_Conference]

## *[dbo].[ValueOfReservation]*

Widok pokazujący należność do zapłacenia za daną rezerwacje uwzględniając wszystkie zniżki.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **ReservationID** | int | 4 |
| **Value** | money | 8 |

### SQL Script

```
CREATE VIEW [dbo].[ValueOfReservation]
AS
SELECT ReservationID, dbo.ReservationValue(ReservationID) AS Value
FROM dbo.Reservation_Conference
GO
```

### Używa

[dbo].[ReservationValue]

[dbo].[Reservation_Conference]

## *[dbo].[WorkshopParticipants]*

Widok pokazujący lsite uczestników wraz z danymi danego warsztatu.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **ConferenceDayID** | int | 4 |
| **WorkshopID** | int | 4 |
| **WorkshopName** | nvarchar(50) | 100 |
| **Workshop_ParticipantID** | int | 4 |
| **Firstname** | nvarchar(50) | 100 |
| **Lastname** | nvarchar(50) | 100 |
| **Companyname** | nvarchar(50) | 100 |

### SQL Script

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

(

    SELECT CD.ConferenceDayID,W.WorkshopID, WT.WorkshopName, WP.Workshop_-
ParticipantID, CP.Firstname, CP.Lastname, CP.Companyname

    FROM Workshop_Participant AS WP

    JOIN Conference_Participant AS CP

    ON CP.Conference_ParticipantID = WP.Conference_ParticipantID

    JOIN Workshop AS W

    ON W.WorkshopID = WP.WorkshopID

    JOIN ConferenceDay AS CD

    ON CD.ConferenceDayID = W.ConferenceDayID

    JOIN WorkshopType AS WT

    ON WT.WorkshopTypeID = W.WorkshopID

    WHERE CP.Conference_ParticipantID = WP.Conference_ParticipantID

    AND WP.WorkshopID = W.WorkshopID


)

GO
```

**Używa**

[dbo].[Conference_Participant]

[dbo].[ConferenceDay]

[dbo].[Workshop]

[dbo].[Workshop_Participant]

[dbo].[WorkshopType]

## *[dbo].[WorkshopsInConferenceList]*

Widok zwracający liste dostępnych warsztatów na daną konferencje.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **Conference Name** | nvarchar(50) | 100 |
| **Workshop Name** | nvarchar(50) | 100 |

### SQL Script

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

(

    SELECT C.ConferenceName AS [Conference Name] , WT.WorkshopName  AS [Workshop
Name]

    FROM Workshop as W

    JOIN WorkshopType AS WT

    ON W.WorkshopID = WT.WorkshopTypeID

    JOIN Conference    AS C

    ON W.ConferenceID = C.ConferenceID

    WHERE C.ConferenceID = W.ConferenceID

    )

GO
```

### Używa

[dbo].[Conference]

[dbo].[Workshop]

[dbo].[WorkshopType]

## *[dbo].[WorkshopsInDayList]*

Widok zwracający listę warsztatów na dany dzień konferencji.

### Columns

| Name | Data Type | Max Length (Bytes) |
|---|---|---|
| **ConferenceDayID** | int | 4 |
| **ConferenceName** | nvarchar(50) | 100 |
| **Number of Conference Day** | int | 4 |
| **Workshop Name** | nvarchar(50) | 100 |

### SQL Script

```
CREATE VIEW [dbo].[WorkshopsInDayList]

AS

    SELECT CD.ConferenceDayID, C.ConferenceName, abs(DATEDIFF(DD, StartDate, Date-
ConferenceDay))+1 as [Number of Conference Day], WT.WorkshopName  AS [Workshop
Name]

    FROM Workshop as W

    JOIN WorkshopType AS WT

    ON W.WorkshopID = WT.WorkshopTypeID

    JOIN Conference    AS C

    ON W.ConferenceID = C.ConferenceID

    JOIN ConferenceDay AS CD

    ON CD.ConferenceDayID = W.ConferenceDayID

    WHERE W.ConferenceDayID = CD.ConferenceDayID

GO
```

### Używa

[dbo].[Conference]

[dbo].[ConferenceDay]

[dbo].[Workshop]

# Funkcje

| Name |
| --- |
| **dbo.CollisionsInWorkshops** |
| **dbo.ConferenceDayAvailablePlaces** |
| **dbo.ConferenceDayFreePlaces** |
| **dbo.FreePlacesForWorkshop** |
| **dbo.GetDiscount** |
| **dbo.NumberOfParticipantsByReservation** |
| **dbo.NumberOfPeopleInReservation** |
| **dbo.Reservation_ConferenceValue** |
| **dbo.Reservation_WorkshopValue** |
| **dbo.ReservationValue** |
| **dbo.StudentDiscount** |

## *[dbo].[CollisionsInWorkshops]*

Funckja zwraca 1 w przypadku, gdy dane warsztaty odbywają się w tym samym czasie i 0 w przeciwnym przypadku.

### SQL Script

```
--Funkcja workshopsCollision - zwraca 1 w przypadku, gdy dane warsztaty odbywają
się w tym samym czasie i 0 w przeciwnym przypadku.

CREATE FUNCTION [dbo].[CollisionsInWorkshops] (@workshopID1 int, @workshopID2 int)

RETURNS bit

AS

BEGIN

DECLARE @start_time1 time(7);

DECLARE @end_time1 time(7);

DECLARE @date1 date;

DECLARE @start_time2 time(7);

DECLARE @end_time2 time(7);

DECLARE @date2 date;

DECLARE @collision bit;


SET @start_time1 = (SELECT Start_time FROM Workshop WHERE workshopID = @workshop-
ID1)

SET @end_time1 = (SELECT end_time FROM Workshop WHERE workshopID = @workshopID1)

SET @date1 = (SELECT DateConferenceDay FROM ConferenceDay AS cd

                INNER JOIN Workshop AS w

                ON w.ConferenceDayID = cd.ConferenceDayID

                where w.WorkshopID = @workshopID1)


SET @start_time2 = (SELECT start_time FROM Workshop WHERE workshopID = @workshop-
ID2)

SET @end_time2 = (SELECT end_time FROM Workshop WHERE workshopID = @workshopID2)

SET @date2 = (SELECT DateConferenceDay FROM ConferenceDay AS cd
```

```
                INNER JOIN Workshop AS w
                ON w.ConferenceDayID = cd.ConferenceDayID
                WHERE w.WorkshopID = @workshopID2)


IF((@date1 = @date2) AND (@start_time1 < @end_time2 OR @start_time2 < @end_time1))
SET @collision = 1
ELSE
SET @collision = 0
RETURN @collision
END
GO
```

**Używa**

[dbo].[ConferenceDay]

[dbo].[Workshop]

## *[dbo].[ConferenceDayFreePlaces]*

Funckja zwraca ilość wolnych miejsc na dany dzień konferencji.

### SQL Script

```
CREATE FUNCTION [dbo].[ConferenceDayFreePlaces] (@ConferenceDayID int)

RETURNS int

AS

BEGIN

DECLARE @Number_of_seats int;

SET @Number_of_seats = (SELECT c.Number_of_seats FROM Conference as c

inner join ConferenceDay as cd

on c.ConferenceID = cd.ConferenceID

where cd.ConferenceDayID = @ConferenceDayID)


DECLARE @takenPlaces int;

SET @takenPlaces = (SELECT SUM(rd.NumberOfPeople) FROM ReservationDetails as rd
WHERE rd.ConferenceDayID = @ConferenceDayID)

RETURN (@Number_of_seats - @takenPlaces);

END

GO
```

### Używa

[dbo].[Conference]

[dbo].[ConferenceDay]

[dbo].[ReservationDetails]

### Używane przez

[dbo].[AddReservationDetails]

## *[dbo].[FreePlacesForWorkshop]*

Funckja zwracająca ilość wolnych miejsc na dany warsztat.

### SQL Script

```
CREATE FUNCTION [dbo].[FreePlacesForWorkshop] (@workshopID int)

RETURNS int

AS

BEGIN

DECLARE @allPlaces int;

SET @allPlaces = (SELECT Number_of_seats FROM Workshop WHERE workshopID =
@workshopID)

DECLARE @takenPlaces int;

SET @takenPlaces = (SELECT SUM(NumberOfPeople) FROM Reservation_Workshop WHERE
workshopID = @workshopID)

RETURN (@allPlaces - @takenPlaces);

END

GO
```

### Używa

[dbo].[Reservation_Workshop]

[dbo].[Workshop]

## *[dbo].[GetDiscount]*

Funckja zwracająca wysokość zniżki ze względu na różnice dni pomiędzy datą rezerwacji a data rozpoczęcia konferencji.

### SQL Script

```
CREATE FUNCTION [dbo].[GetDiscount] (@ReservationID INT)
RETURNS REAL
AS
BEGIN
DECLARE @Disc REAL
SET @Disc = (SELECT TOP 1 Discount FROM Discounts
                    INNER JOIN Conference AS c
                    ON c.ConferenceId = Discounts.ConferenceID
                    INNER JOIN dbo.Reservation_Conference RC
                    ON RC.ConferenceID = c.ConferenceID
                    WHERE RC.ReservationID = @ReservationID
                    And DATEDIFF(dd,RC.DateOfReservation,c.StartDate ) >
Discounts.DateDifference)
RETURN @Disc
END

GO
```

### Używa

[dbo].[Conference]

[dbo].[Discounts]

[dbo].[Reservation_Conference]

### Używane przez

[dbo].[Reservation_ConferenceValue]

## *[dbo].[NumberOfParticipantsByReservation]*

Funckja zwraca ilość osób zdeklarowanych poprzez dane osobowe na dany dzień konferencji.

### SQL Script

```
CREATE FUNCTION [dbo].[NumberOfParticipantsByReservation] (@ReservationId INT)

RETURNS INT

AS

BEGIN

DECLARE @NumberOfParticipants int

SET @NumberOfParticipants = (SELECT COUNT(Conference_participantID) FROM
dbo.Conference_Participant

                                WHERE ReservationID = @ReservationId)

RETURN @NumberOfParticipants

END


GO
```

### Używa

[dbo].[Conference_Participant]

### Używane przez

[dbo].[CustomersToCall]

## *[dbo].[NumberOfPeopleInReservation]*

Funkcja zwracająca ilość zarezerwowanych miejsc dla danej rezerwacji.

### SQL Script

```
CREATE FUNCTION [dbo].[NumberOfPeopleInReservation] (@ReservationId INT)
RETURNS INT
AS
BEGIN
DECLARE @NumberOfPeople INT
SET @NumberOfPeople = (SELECT SUM(NumberOfPeople) FROM dbo.ReservationDetails
                                WHERE ReservationID = @ReservationId)
RETURN @NumberOfPeople
END


GO
```

### Używa

[dbo].[ReservationDetails]

### Używane przez

[dbo].[CustomersToCall]

## *[dbo].[Reservation_ConferenceValue]*

Funkcja zwracająca należność dla rezerwacji .

### SQL Script

```
CREATE FUNCTION [dbo].[Reservation_ConferenceValue] (@ReservationID int)

RETURNS MONEY --wynikiem funkcji jest cena tylko za rezerwacje konferencji(bez
warsztatów)

AS

BEGIN

--ustalamy liczbę ludzi którzy nie są studentami i z tego tytułu nie przysłuje im
zniżka

DECLARE @NumberOfRegularPeople int

SET @NumberOfRegularPeople = (SELECT SUM(rd.NumberOfPeople-rd.NumberOfStudents) AS
NOT_Students

                              FROM dbo.ReservationDetails rd

                             WHERE rd.ReservationID=@ReservationID

                             GROUP BY rd.ReservationID )

--ustalamy liczbę studentów

DECLARE @NumberOfStudents INT

SET @NumberOfStudents=(SELECT SUM(NumberOfStudents) FROM dbo.ReservationDetails

                       WHERE ReservationID=@ReservationID)

--cena za dzień konferencji

DECLARE @Price money

SET @Price=(SELECT c.Price FROM dbo.Conference c

            WHERE ConferenceID=(SELECT ConferenceID FROM dbo.Reservation_Conference

                                WHERE ReservationID=@ReservationID))

--wartosc rezerwacji bez uwzględniania zniżki dniowej

DECLARE @Value MONEY

SET @Value=@Price*(@NumberOfRegularPeople+(1-dbo.StudentDiscount(@Reservation-
ID))*@NumberOfStudents)
```

```
--finalna wartość zamówienia konferencji po uwzględnienu zniżki dniowej

DECLARE @ValueAfterDiscount MONEY

SET @ValueAfterDiscount=(1-dbo.GetDiscount(@ReservationID))*@Value



RETURN(@ValueAfterDiscount);

END

GO
```

### Używa

[dbo].[GetDiscount]

[dbo].[StudentDiscount]

[dbo].[Conference]

[dbo].[Reservation_Conference]

[dbo].[ReservationDetails]

### Używane przez

[dbo].[ReservationValue]

## *[dbo].[Reservation_WorkshopValue]*

Funckja zwracająca należność za rezerwacje warsztatu.

### SQL Script

```
CREATE FUNCTION [dbo].[Reservation_WorkshopValue] (@ReservationID int)

RETURNS money

AS

BEGIN



DECLARE @Licznik INT

SET @Licznik= (SELECT count(ReservationID) FROM reservation_workshop WHERE Reservation-
ID=@ReservationID)

DECLARE @Reservation_WorkshopID INT

DECLARE @Value1 INT

SET @Reservation_WorkshopID=0

SET @Value1=0

DECLARE @Value INT

SET @Value=0



WHILE @Licznik>0

BEGIN

DECLARE @WorkshopID INT

SET @WorkshopID=(SELECT TOP 1 WorkshopID FROM dbo.Reservation_Workshop

                WHERE ReservationID=@ReservationID AND Reservation_WorkshopID>@Reservation_-
WorkshopID)

DECLARE @Price int

SET @Price=(SELECT Price_Workshop FROM Workshop WHERE WorkshopID=@WorkshopID)

DECLARE @NumberOfPeople INT

SET @NumberOfPeople=(SELECT TOP 1 NumberOfPeople FROM dbo.Reservation_Workshop

                     WHERE ReservationID=@ReservationID AND WorkshopID=@WorkshopID

                     AND Reservation_WorkshopID>@Reservation_WorkshopID)



SET @Reservation_WorkshopID=(SELECT TOP 1 Reservation_WorkshopID FROM dbo.Reservation_Workshop

                             WHERE ReservationID=@ReservationID AND Reservation_Workshop-
ID>@Reservation_WorkshopID)
```

```
SET @Value1=@Price*@NumberOfPeople

SET @Value=@Value+@Value1

SET @Licznik=@Licznik-1



END



RETURN(@Value);

END

GO
```

### Używa

[dbo].[Reservation_Workshop]

[dbo].[Workshop]

### Używane przez

[dbo].[ReservationValue]

## *[dbo].[ReservationValue]*

Funkcja zwracająca finalną należność za całość dokonanej rezerwacji.

### SQL Script

```
--ReservationValue

CREATE FUNCTION [dbo].[ReservationValue] (@ReservationID int)

RETURNS money

AS

BEGIN

RETURN(dbo.Reservation_ConferenceValue(@ReservationID) + dbo.Reservation_Workshop-
Value(@ReservationID));

END

GO
```

### Używa

[dbo].[Reservation_ConferenceValue]

[dbo].[Reservation_WorkshopValue]

### Używane przez

[dbo].[PriceOfReservation]

[dbo].[ValueOfReservation]

## *[dbo].[StudentDiscount]*

Funckja zwracająca wysokość zniżki studenckiej zależnie od ID konferencji.

### SQL Script

```
CREATE FUNCTION [dbo].[StudentDiscount] (@ReservationID INT)
RETURNS REAL
AS
BEGIN

DECLARE @SDiscount REAL;
DECLARE @CustomerID INT;

SET @SDiscount = (SELECT studentDiscount FROM Conference c
                    INNER JOIN dbo.Reservation_Conference AS rc
                    ON Rc.ConferenceID = c.ConferenceID
                    WHERE rc.ReservationID = @ReservationID)
RETURN @SDiscount
END

GO
```

### Używa

[dbo].[Conference]

[dbo].[Reservation_Conference]

### Używane przez

[dbo].[Reservation_ConferenceValue]

## Role w systemie

Zalecamy stworzenie następujących ról:

**administrator** - osoba zaznajomiona z językiem SQL, która może rozwijać i udoskonalać bazę danych. Jest także odpowiedzialna za obsługę zdarzeń losowych.
- dostęp do wszystkich procedur oraz widoków.

**pracownik firmy** - osoba obsługująca zamówienia, która będzie się kontaktowała z klientem i pomagała mu w przypadku problemów z rejestracją.
- procedury:
  - dbo.AddReservationDetails
  - dbo.UpdatePaymentDate
  - dbo.AddDiscount
  - dbo.CancelReservation_Conference
  - dbo.AddConference
  - dbo.AddConferenceDay
  - dbo.AddConferenceParticipant
  - dbo.AddWorkshop
  - dbo.AddStudent

- widoki:
  - dbo.NumberOfCustomerReservations
  - dbo.CancelledReservations
  - dbo.ConferenceDayParticipants
  - dbo.ConferenceParticipants
  - dbo.ConferenceToCancelled
  - dbo.CustomersToCall
  - dbo.PopularConferences
  - dbo.PopularWorkshops
  - dbo.RegularCustomers
  - dbo.ValueOfReservation
  - dbo.WorkshopParticipants
  - dbo.WorkshopsInConferenceList

**klient -** osoba składająca zamówienia oraz dodająca uczestników
- procedury:
  - dbo.AddReservation_Conference
  - dbo.AddReservationWorkshop
  - dbo.AddReservationDetails
  - dbo.ChangeIndividualCustomerData
  - dbo.ChangeCompanyData
  - dbo.CancelReservation_Conference
  - dbo.AddCustomerIndividual

- dbo.AddCustomerCompany
- dbo.CancelReservation_Conference
- dbo.AddConferenceParticipant
- dbo.AddWorkshopParticipant
- dbo.AddStudent

o widoki:

- dbo.ConferenceParticipants
- dbo.WorkshopsInDayList
- dbo.WorkshopsInConferenceList
- dbo.ValueOfReservation

## Generator

Do generowania danych wykorzystałyśmy program RedGate, napisane przez nas widoki oraz wykorzystałyśmy wbudowane funkcje Microsoft SQL Management Studio.

### *Widoki wykorzystywane do generowania danych:*

[dbo].[GeneratorCompany]

SQL Script

```
CREATE VIEW [dbo].[GeneratorCompany]
as
SELECT CustomerID FROM dbo.Customers
WHERE CustomerID IN (SELECT companyid FROM dbo.Company)
GO
```

## [dbo].[GeneratorConference_participantCompany]

### SQL Script

```
CREATE VIEW [dbo].[GeneratorConference_participantCompany]

AS

SELECT rc.reservationID,rc.ConferenceID,cd.ConferenceDayID,(SELECT c.companyname FROM
dbo.Company c WHERE rc.CustomerID=c.CompanyID) AS CompanyName,(SELECT SUM(numberofpeople) FROM
dbo.ReservationDetails rd1 WHERE rd1.ReservationID=rc.ReservationID) AS number

FROM dbo.Reservation_Conference rc

INNER JOIN dbo.ConferenceDay cd

ON cd.ConferenceID = RC.ConferenceID

WHERE rc.CustomerID IN (SELECT companyid FROM dbo.Company) AND rc.ReservationID IN (SELECT
ReservationID FROM dbo.ReservationDetails) GO
```

## [dbo].[GeneratorParticipants_Individual]

### SQL Script

```


CREATE VIEW [dbo].[GeneratorParticipants_Individual] AS



SELECT ROW_NUMBER() OVER( ORDER BY rc.ReservationID )AS Conference_ParticipantID
,rc.ReservationID ,rc.ConferenceID,cd.ConferenceDayID,ic.FirstName,ic.LastName,NULL AS
CompanyName

FROM Reservation_Conference AS RC

INNER JOIN dbo.IndividualCustomer ic

ON rc.CustomerID=ic.IndividualCustomerID

INNER JOIN dbo.ConferenceDay cd

ON cd.ConferenceID = RC.ConferenceID

WHERE rc.CustomerID IN (SELECT ic.IndividualCustomerID FROM dbo.IndividualCustomer)

GO
```

## [dbo].[GeneratorParticipants_IndividualStudent]

### SQL Script

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



SELECT cp.Conference_ParticipantID AS StudentID,cp.Conference_ParticipantID+15000 AS Student_-
CardID, GETDATE() AS Expiration_Date

FROM dbo.Conference_Participant cp

INNER JOIN dbo.Reservation_Conference rc

ON rc.ReservationID = cp.ReservationID

WHERE rc.CustomerID IN (SELECT IndividualCustomerID FROM dbo.IndividualCustomer WHERE Is-
Student=1)

GO
```

## [dbo].[GeneratorReservationDetailsCompany]

### SQL Script

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

SELECT ROW_NUMBER() OVER( ORDER BY rc.ReservationID ) AS ReservationDetailsID,

rc.ReservationID,cd.ConferenceDayID, 2 AS NumberOfPeople,0 AS Cancelled,0 AS NumberOfStudents

FROM dbo.Reservation_Conference rc

JOIN dbo.ConferenceDay cd

ON cd.ConferenceID = rc.ConferenceID

WHERE CustomerID IN (SELECT companyId FROM dbo.Company)



GO
```

## [dbo].[GeneratorReservationDetailsIndividual]

### SQL Script

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

SELECT ROW_NUMBER() OVER( ORDER BY rc.ReservationID ) AS ReservationDetailsID,

rc.ReservationID,cd.ConferenceDayID, 2 AS NumberOfPeople,0 AS Cancelled,0 AS NumberOfStudents

FROM dbo.Reservation_Conference rc

JOIN dbo.ConferenceDay cd

ON cd.ConferenceID = rc.ConferenceID

WHERE CustomerID IN (SELECT IndividualCustomerID  FROM dbo.IndividualCustomer)

GO
```

## [dbo].[GeneratorReservationsIDCompany]

### SQL Script

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

SELECT ROW_NUMBER() OVER( ORDER BY rc.ReservationID )+13222 AS ReservationDetailsID,
RC.ReservationID,cd.ConferenceDayID,5 AS NumberOfPeople, 0 AS Cancelled,2 AS NumberOfStudents

FROM Reservation_Conference AS RC

INNER JOIN dbo.ConferenceDay cd

ON cd.ConferenceID = RC.ConferenceID

WHERE RC.CustomerID IN (SELECT CompanyID FROM dbo.Company)

GO
```

## [dbo].[GeneratorReservationsIDIndywidual]

### SQL Script

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

SELECT ROW_NUMBER() OVER( ORDER BY rc.ReservationID ) AS ReservationDetailsID, RC.Reservation-
ID,cd.ConferenceDayID,'1' AS NumberOfPeople,'0' AS Cancelled, ic.IsStudent AS NumberofStudents



FROM Reservation_Conference AS RC

INNER JOIN dbo.ConferenceDay cd

ON cd.ConferenceID = RC.ConferenceID

INNER JOIN dbo.IndividualCustomer ic

ON ic.IndividualCustomerID=rc.CustomerID

WHERE RC.CustomerID IN  (SELECT IndividualCustomerID FROM dbo.IndividualCustomer ic WHERE
rc.CustomerID=ic.IndividualCustomerID) AND

rc.ConferenceID IN (SELECT cd.ConferenceDayID FROM dbo.ConferenceDay cd WHERE cd.Conference-
ID=rc.ConferenceID)

GO
```

## [dbo].[GeneratorWorkshop]

### SQL Script

```
CREATE VIEW [dbo].[GeneratorWorkshop]

AS

SELECT ConferenceID,ConferenceDayID,DateConferenceDay FROM dbo.ConferenceDay

GO
```

## [dbo].[GeneratorWorkshopIndywidualny]

### SQL Script

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

SELECT DISTINCT RC.ReservationID,W.WorkshopID, '1' AS NumberOfPeople,'0' AS Cancelled

FROM Reservation_Conference AS RC

INNER JOIN Workshop AS w

ON W.ConferenceID = RC.ConferenceID

INNER JOIN dbo.IndividualCustomer ic

ON ic.IndividualCustomerID = rc.CustomerID

WHERE RC.CustomerID IN  (SELECT IndividualCustomerID FROM dbo.IndividualCustomer) AND

rc.ConferenceID IN (SELECT ConferenceDayID FROM dbo.ReservationDetails WHERE Reservation-
ID=rc.ReservationID)

and RC.reservationID BETWEEN 2000 AND 4000

GO
```

## [dbo].[GeneratorWorkshopFirma]

### SQL Script

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

SELECT DISTINCT RC.ReservationID, W.WorkshopID

FROM Reservation_Conference AS RC

INNER JOIN dbo.ReservationDetails RD

ON RD.ReservationID = RC.ReservationID
```

```
INNER JOIN Workshop AS w

ON W.ConferenceID = RC.ConferenceID

INNER JOIN dbo.Company ic

ON ic.CompanyID = rc.CustomerID

WHERE RC.CustomerID IN  (SELECT CompanyID FROM dbo.Company) AND

rc.ConferenceID IN (SELECT ConferenceDayID FROM dbo.ReservationDetails WHERE Reservation-
ID=rc.ReservationID)

AND RC.ReservationID > 5000

GO
```

## [dbo].[GeneratorWorkshopParticipantsAdding]

### SQL Script

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


SELECT rw.workshopid,rw.ReservationID,rw.Reservation_WorkshopID,cp.Conference_ParticipantID

FROM dbo.Reservation_Workshop rw

INNER JOIN dbo.Conference_Participant cp

ON cp.ReservationID=rw.ReservationID

WHERE rw.WorkshopID IN ( SELECT WorkshopID FROM dbo.Workshop WHERE ConferenceDay-
ID=cp.ConferenceDayID)

GO
```