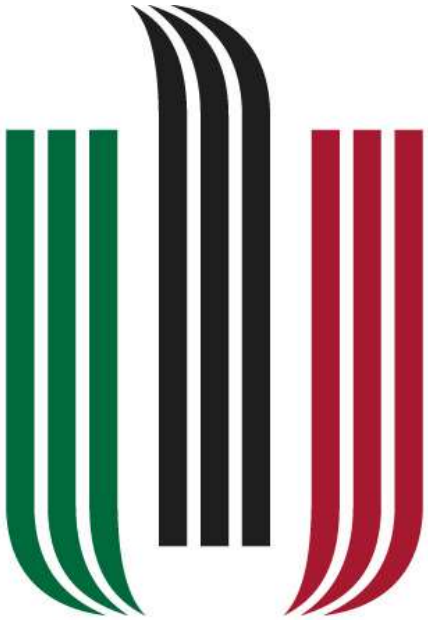


AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

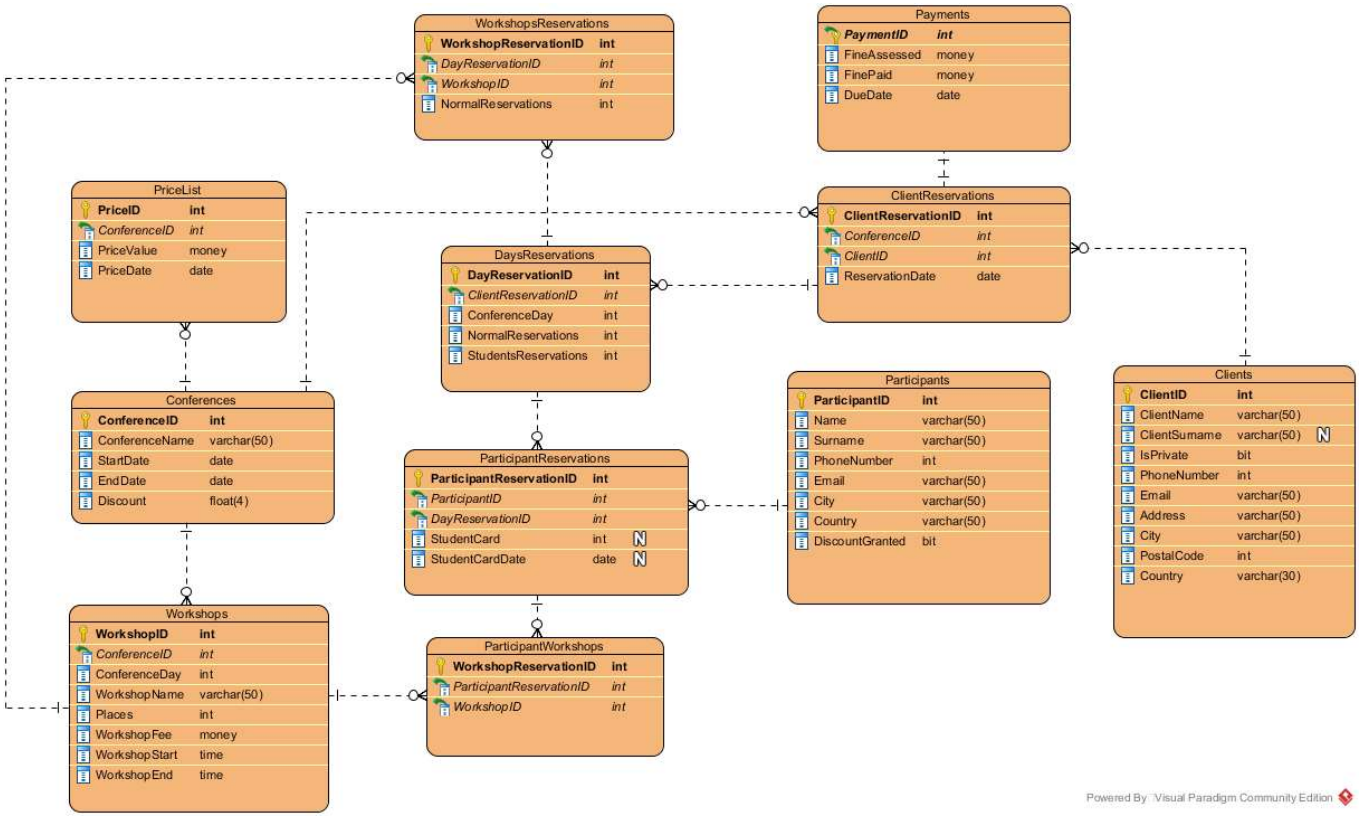
Laboratorium Podstaw Baz Danych
Dokumentacja Systemu zarządzania konferencjami

Miłosz Mandowski
Michał Śledź

Ogólne informacje

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

Schemat bazy danych



Tabele

Clients - tabela zawierająca informacje o klientach. Zawiera pola

- **ClientID - identyfikator klienta, unikatowy, zaczyna się od 1, inkrementowany o 1**
- **ClientName - nazwa klienta. W przypadku firm nazwa firmy, w przypadku osoby prywatnej imię**
- **ClientSurname - nazwisko klienta. W przypadku frim pozostaje wartość `NULL`, w przypadku osoby prywatnej nazwisko**
- **IsPrivate - informacja czy klientem jest firma czy osoba prywatna. Typ bitowy.**
- **PhoneNumber - numer telefonu. Musi posiadać 9 cyfr, a pierwsza nie może być zerem**
- **Email - adres email klienta**
- **Address - adres klienta**
- **City - miasto klienta**
- **PostalCode - kod pocztowy**
- **Country - państwo klienta**

```
CREATE TABLE Clients (
  ClientID int IDENTITY(1,1) NOT NULL,
  ClientName varchar(50) NOT NULL,
  ClientSurname varchar(50) NULL,
  IsPrivate bit NOT NULL DEFAULT 0,
  PhoneNumber int NOT NULL CHECK (PhoneNumber like '[1-9][0-9][0-9][0-9][0-9][0-9][
0-9][0-9][0-9]'),
  Email varchar(50) NOT NULL,
  Address varchar(50) NOT NULL,
  City varchar(50) NOT NULL,
  PostalCode int NOT NULL,
  Country varchar(30) NOT NULL,
  PRIMARY KEY (ClientID));
```

Conferences - tabela zawierająca informacje o konferencjach. Zawiera pola

- **ConferenceID - identyfikator konferencji, unikatowy, zaczyna się od 1, inkrementowany o 1**
- **ConferenceName - temat konferencji**
- **StartDate - data rozpoczęcia konferencji w formacie `yyyy-mm-dd`**
- **EndDate - data zakończenia konferencji w formacie `yyyy-mm-dd`**
- **Discount - zniżka dla studentów. Domyślnie wartość ustawiona na 0. Musi być mniejsza lub równa 1.**

```
CREATE TABLE Conferences (
  ConferenceID int IDENTITY(1,1) NOT NULL,
  ConferenceName varchar(50) NOT NULL,
  StartDate date NOT NULL,
  EndDate date NOT NULL,
  Discount float(4) NOT NULL DEFAULT 0 CHECK (Discount <= 1),
  PRIMARY KEY (ConferenceID));
```

```
CREATE TABLE ClientReservations (
  ClientReservationID int IDENTITY(1,1) NOT NULL,
  ConferenceID int NOT NULL,
  ClientID int NOT NULL,
  ReservationDate date NOT NULL DEFAULT Convert(date, getdate()),
  PRIMARY KEY (ClientReservationID));
```

```
CREATE TABLE DaysReservations (
    DayReservationID int IDENTITY(1,1) NOT NULL,
    ClientReservationID int NOT NULL,
    ConferenceDay int NOT NULL,
    NormalReservations int NOT NULL CHECK (NormalReservations > 0),
    StudentsReservations int NOT NULL DEFAULT 0,
    PRIMARY KEY (DayReservationID));
```

```
CREATE TABLE ParticipantReservations (
    ParticipantReservationID int IDENTITY(1,1) NOT NULL,
    ParticipantID int NOT NULL,
    DayReservationID int NOT NULL,
    StudentCard int NULL,
    StudentCardDate date NULL,
    PRIMARY KEY (ParticipantReservationID));
```

```
CREATE TABLE Participants (
    ParticipantID int IDENTITY(1,1) NOT NULL,
    Name varchar(50) NOT NULL,
    Surname varchar(50) NOT NULL,
    PhoneNumber int NOT NULL CHECK (PhoneNumber like '[1-9][0-9][0-9][0-9][0-9][0-9
][0-9][0-9][0-9]'),
    Email varchar(50) NOT NULL,
    City varchar(50) NOT NULL,
    Country varchar(50) NOT NULL,
    DiscountGranted bit NOT NULL DEFAULT 0,
    PRIMARY KEY (ParticipantID));
```

```
CREATE TABLE ParticipantWorkshops (
    WorkshopReservationID int IDENTITY(1,1) NOT NULL,
    ParticipantReservationID int NOT NULL,
    WorkshopID int NOT NULL,
```

```
    PRIMARY KEY (WorkshopReservationID));
```

```
CREATE TABLE Payments (
    PaymentID int IDENTITY(1,1) NOT NULL,
    FineAssessed money NOT NULL,
    FinePaid money NOT NULL DEFAULT 0,
    DueDate date NOT NULL,
    PRIMARY KEY (PaymentID));
```

```
CREATE TABLE PriceList (
    PriceID int IDENTITY(1,1) NOT NULL,
    ConferenceID int NOT NULL,
    PriceValue money NOT NULL,
    PriceDate date NOT NULL,
    PRIMARY KEY (PriceID));
```

```
CREATE TABLE Workshops (
    WorkshopID int IDENTITY(1,1) NOT NULL,
    ConferenceID int NOT NULL,
    ConferenceDay int NOT NULL,
    WorkshopName varchar(50) NOT NULL,
    Places int NOT NULL CHECK (Places >= 0),
    WorkshopFee money NOT NULL,
    WorkshopStart time NOT NULL,
    WorkshopEnd time NOT NULL,
    PRIMARY KEY (WorkshopID));
```

```
CREATE TABLE WorkshopsReservations (
    WorkshopReservationID int IDENTITY(1,1) NOT NULL,
    DayReservationID int NOT NULL,
    WorkshopID int NOT NULL,
    NormalReservations int NOT NULL CHECK (NormalReservations > 0),
    PRIMARY KEY (WorkshopReservationID));
```

```
-- ========================================
-- Creating Keys
-- ========================================

ALTER TABLE WorkshopsReservations
```

```
    ADD CONSTRAINT FKWorkshopsResToDaysRes
    FOREIGN KEY (DayReservationID) REFERENCES DaysReservations (DayReservationID);

ALTER TABLE WorkshopsReservations
    ADD CONSTRAINT FKWorkshopsResToWorkshops
    FOREIGN KEY (WorkshopID) REFERENCES Workshops (WorkshopID);

ALTER TABLE ParticipantReservations
    ADD CONSTRAINT FKParticipantResToDaysRes
    FOREIGN KEY (DayReservationID) REFERENCES DaysReservations (DayReservationID);

ALTER TABLE ParticipantReservations
    ADD CONSTRAINT FKParticipantResToParticipants
    FOREIGN KEY (ParticipantID) REFERENCES Participants (ParticipantID);

ALTER TABLE ParticipantWorkshops
    ADD CONSTRAINT FKParticipantWorksToParticipantRes
    FOREIGN KEY (ParticipantReservationID) REFERENCES ParticipantReservations (Part
icipantReservationID);

ALTER TABLE ParticipantWorkshops
    ADD CONSTRAINT FKParticipantWorksToWorkshops
    FOREIGN KEY (WorkshopID) REFERENCES Workshops (WorkshopID);

ALTER TABLE ClientReservations
    ADD CONSTRAINT FKClientResToClients
    FOREIGN KEY (ClientID) REFERENCES Clients (ClientID);

ALTER TABLE ClientReservations
    ADD CONSTRAINT FKClientResToConferences
    FOREIGN KEY (ConferenceID) REFERENCES Conferences (ConferenceID);

ALTER TABLE DaysReservations
    ADD CONSTRAINT FKDaysResToClientRes
    FOREIGN KEY (ClientReservationID) REFERENCES ClientReservations (ClientReservat
ionID);

ALTER TABLE Payments
    ADD CONSTRAINT FKPaymentsToClientRes
    FOREIGN KEY (PaymentID) REFERENCES ClientReservations (ClientReservationID);
```

```
ALTER TABLE PriceList
    ADD CONSTRAINT FKPriceListToConferences
    FOREIGN KEY (ConferenceID) REFERENCES Conferences (ConferenceID);

ALTER TABLE Workshops
    ADD CONSTRAINT FKWorkshopsToConferences
    FOREIGN KEY (ConferenceID) REFERENCES Conferences (ConferenceID);
```

```
-- ========================================
-- Drop code
-- ========================================

ALTER TABLE WorkshopsReservations
    DROP CONSTRAINT FKWorkshopsResToDaysRes;

ALTER TABLE WorkshopsReservations
    DROP CONSTRAINT FKWorkshopsResToWorkshops;

ALTER TABLE ParticipantReservations
    DROP CONSTRAINT FKParticipantResToDaysRes;

ALTER TABLE ParticipantReservations
    DROP CONSTRAINT FKParticipantResToParticipants;

ALTER TABLE ParticipantWorkshops
    DROP CONSTRAINT FKParticipantWorksToParticipantRes;

ALTER TABLE ParticipantWorkshops
    DROP CONSTRAINT FKParticipantWorksToWorkshops;

ALTER TABLE ClientReservations
    DROP CONSTRAINT FKClientResToClients;

ALTER TABLE ClientReservations
    DROP CONSTRAINT FKClientResToConferences;

ALTER TABLE DaysReservations
    DROP CONSTRAINT FKDaysResToClientRes;
```

```
ALTER TABLE Payments
    DROP CONSTRAINT FKPaymentsToClientRes;

ALTER TABLE PriceList
    DROP CONSTRAINT FKPriceListToConferences;

ALTER TABLE Workshops
    DROP CONSTRAINT FKWorkshopsToConferences;

DROP TABLE ClientReservations;
DROP TABLE Clients;
DROP TABLE Conferences;
DROP TABLE DaysReservations;
DROP TABLE ParticipantReservations;
DROP TABLE Participants;
DROP TABLE ParticipantWorkshops;
DROP TABLE Payments;
DROP TABLE PriceList;
DROP TABLE Workshops;
DROP TABLE WorkshopsReservations;
```

Triggery

- IsPrivateRequireSurname
- NoPrivateNoSurname
- ExistingClient

IsPrivateRequireSurname - sprawdza czy podano nazwisko jeżeli dodany lub zaktualizowany klient jest osobą prywatną

```
CREATE TRIGGER IsPrivateRequireSurname
ON Clients
AFTER INSERT, UPDATE
AS
BEGIN
  IF (SELECT IsPrivate FROM inserted) = 1 AND (SELECT ClientSurname FROM inser
ted) IS NULL
  BEGIN
      ;THROW 51000, 'Dla prywatnego klienta należy podać nazwisko.', 1;
      ROLLBACK TRANSACTION
```

```
  END
END
GO
```

NoPrivateNoSurname - sprawdza czy podano nazwisko jeżeli klient nie jest osobą prywatną. Jeżeli podano, zwraca wyjątek i nazwisko trzeba usunąć.

```
CREATE TRIGGER NoPrivateNoSurname
ON Clients
AFTER INSERT, UPDATE
AS
BEGIN
  IF (SELECT IsPrivate FROM inserted) = 0 AND (SELECT ClientSurname FROM inser
ted) IS NOT NULL
  BEGIN
      ;THROW 51000, 'Dla firmowego klienta nie należy podawać nazwiska.', 1;
      ROLLBACK TRANSACTION
  END
END
GO
```

ExistingClient - sprawdza czy wprowadzany klient jest już zapisany

```
CREATE TRIGGER ExistingClient
ON Clients
AFTER INSERT
AS
BEGIN
  IF (SELECT COUNT(*) FROM inserted as i
      inner join clients as c
      on i.ClientID = c.ClientID
  ) > 0
  BEGIN
      ;THROW 51000, 'Podany klient juz istnieje', 1;
      ROLLBACK TRANSACTION
  END
END
GO
```