# OPIS WARUNKÓW INTEGRALNOŚCIOWYCH

Dla każdej relacji opis zawiera:

* krótki opis,
* analizę warunków integralnościowych,
* kod implementujący ww. warunki.

**Główne założenie**: Jeśli nie jest napisane inaczej, wprowadzane wartości nie mogą być nullami. Inne dodatkowe warunki będą opisane jeśli zajdzie taka konieczność.

### Categories

Tabela przechowująca kategorie produktów oferowanych przez restauracje.

Dodatkowe warunki:

* 1. Picture może być null’em.

CREATE TABLE Categories  
(  
 CategoryID int NOT NULL IDENTITY,  
 CategoryName nvarchar(100) NOT NULL,  
 Picture image NULL,  
 CONSTRAINT Categories\_pk PRIMARY KEY (CategoryID)  
);

### Cities

Relacja także jest słownikiem.

Dodatkowe warunki:

* 1. Nazwy miast rozpoczynają się wielką literą.

CREATE TABLE Cities  
(  
 CityID int NOT NULL IDENTITY,  
 CityName nvarchar(100) NOT NULL,  
 CountryID int NOT NULL,  
 CONSTRAINT capitalCheckCities CHECK ((CityName LIKE '[A-Z]%')),  
 CONSTRAINT Cities\_pk PRIMARY KEY (CityID)  
);

### Companies

Relacja przechowuje firmy, które korzystają z usług restauracji.

CREATE TABLE Companies  
(  
 CompanyID int NOT NULL IDENTITY,  
 CompanyName nvarchar(100) NOT NULL,  
 CONSTRAINT Companies\_pk PRIMARY KEY (CompanyID)  
);

### CompanyCustomers

Przechowuje informacje o klientach firmowych.

Dodatkowe warunki:

* 1. Imię i nazwisko rozpoczynają się wielką literą.

CREATE TABLE CompanyCustomers  
(  
 CCustomerID int NOT NULL IDENTITY,  
 FirstName nvarchar(100) NOT NULL,  
 LastName nvarchar(100) NOT NULL,  
 CONSTRAINT capitalCheckCCustomers CHECK ((FirstName LIKE '[A-Z]%' AND LastName LIKE '[A-Z]%')),  
 CONSTRAINT CompanyCustomers\_pk PRIMARY KEY (CCustomerID)  
);

### Countries

Relacja jest słownikiem dla innych tabel.

Dodatkowe warunki:

* 1. Nazwy państw rozpoczynają się wielką literą

CREATE TABLE Countries  
(  
 CountryID int NOT NULL IDENTITY,  
 CountryName nvarchar(100) NOT NULL,  
 CONSTRAINT capitalCheckCountries CHECK ((CountryName LIKE '[A-Z]%')),  
 CONSTRAINT Countries\_pk PRIMARY KEY (CountryID)  
);

### Currencies

Relacja przechowuje informacje o aktualnych kursach walut dostępnych w restauracji.

Dodatkowe warunki:

* 1. RatioToPLN musi być dodatnie.

CREATE TABLE Currencies  
(  
 CurrencyName nvarchar(3) NOT NULL,  
 RatioToPLN numeric(10, 2) NOT NULL DEFAULT 1,  
 CONSTRAINT ratioCheck CHECK ((RatioToPLN > 0)),  
 CONSTRAINT Currencies\_pk PRIMARY KEY (CurrencyName)  
);

### CustomerDepartments

Relacja przechowuje dane o zależnościach między klientami firm, a departamentami, które płacą za ich posiłki.

CREATE TABLE CustomerDepartments  
(  
 CCustomersID int NOT NULL,  
 DepID int NOT NULL,  
 CONSTRAINT CustomerDepartments\_pk PRIMARY KEY (CCustomersID, DepID)  
);

### Customers

Relacja przechowująca informacje o klientach prywatnych.

Dodatkowe warunki:

* 1. E-mail może być wartością null.
  2. E-mail musi zawierać w sobie @ i “.”.
  3. Imię i nazwisko musi rozpoczynać się wielką literą.

CREATE TABLE Customers  
(  
 CustomerID int NOT NULL IDENTITY,  
 FirstName nvarchar(100) NOT NULL,  
 LastName nvarchar(100) NOT NULL,  
 Email nvarchar(100) NULL,  
 CONSTRAINT emailFormula CHECK ((Email LIKE '%@%')),  
 CONSTRAINT capitalCheckCustomers CHECK ((FirstName LIKE '[A-Z]%' AND LastName LIKE '[A-Z]%')),  
 CONSTRAINT Customers\_pk PRIMARY KEY (CustomerID)  
);

### Departments

Tabela przechowuje informacje o poszczególnych departamentach firm klienckich.

CREATE TABLE Departments  
(  
 DepID int NOT NULL IDENTITY,  
 NIP nvarchar(100) NOT NULL,  
 CityID int NOT NULL,  
 Address nvarchar(100) NOT NULL,  
 PostalCode nvarchar(20) NOT NULL,  
 CompanyID int NOT NULL,  
 CONSTRAINT Departaments\_pk PRIMARY KEY (DepID)  
);

### Discounts

Tabela przechowująca i opisująca zniżki przysługujące klientom.

Dodatkowe warunki:

* 1. DiscountValue oraz ifUsed mają warunek default, oba ustawiony na zero, z tą różnicą, że w DiscountValue oznacza ono wartość liczbową, a w ifUsed wartość false.
  2. Atrybut Until ma wartość domyślną ustawioną na 30 dni od dnia dzisiejszego
  3. Poprzez formułę CHECK ustalane są jedyne dwa możliwe typy: “Loyal Customer discount” oraz “One use discount”

CREATE TABLE Discounts  
(  
 DiscountID int NOT NULL IDENTITY,  
 CustomerID int NOT NULL,  
 DiscountType nvarchar(100) NOT NULL,  
 DiscountValue float NOT NULL DEFAULT 0,  
 IfUsed bit NOT NULL DEFAULT 0,  
 Until date NOT NULL DEFAULT GETDATE() + 30,  
 CONSTRAINT discountType CHECK ((DiscountType IN ('Loyal Customers discount', 'One use discount'))),  
 CONSTRAINT discountValue CHECK (DiscountValue BETWEEN 0 AND 1),  
 CONSTRAINT Discounts\_pk PRIMARY KEY (DiscountID)  
);

### Employees

Tabela przechowująca informacje o pracownikach.

Dodatkowe warunki:

* 1. Data urodzenia musi być realna ( > 01.01.1900) i większa niż 15 lat od dzisiejszego dnia.
  2. Data zatrudnienia ma defaultowe ustawienie na dzisiejszą datę.
  3. E-mail musi zawierać @ i “.”.
  4. SuperiorID może być wartością null.
  5. Imię i nazwisko rozpoczynają się wielką literą.

CREATE TABLE Employees  
(  
 EmployeeID int NOT NULL IDENTITY,  
 FirstName nvarchar(100) NOT NULL,  
 LastName nvarchar(100) NOT NULL,  
 BirthDate date NOT NULL,  
 CityID int NOT NULL,  
 Address nvarchar(100) NOT NULL,  
 PostalCode nvarchar(100) NOT NULL,  
 Email nvarchar(100) NOT NULL,  
 HireDate date NOT NULL DEFAULT GETDATE(),  
 JobID int NOT NULL,  
 SuperiorID int NULL,  
 CONSTRAINT Basics CHECK ((BirthDate > '1900-01-01' AND BirthDate < GETDATE() - 5475 AND Email LIKE '%@%.%' AND  
 FirstName LIKE '[A-Z]%' AND LastName LIKE '[A-Z]%')),  
 CONSTRAINT Employees\_pk PRIMARY KEY (EmployeeID)  
);

### Jobs

Słownik, przechowujący wszystkie role pracowników.

CREATE TABLE Jobs  
(  
 JobID int NOT NULL IDENTITY,  
 JobName nvarchar(max) NOT NULL,  
 JobDescription nvarchar(max) NOT NULL,  
 CONSTRAINT Jobs\_pk PRIMARY KEY (JobID)  
);

### Menu

Reprezentuje obecne menu restauracji.

Dodatkowe warunki:

* 1. SinceDate ma ustawioną domyślną wartość (warunek default) na dzisiejszą datę.

CREATE TABLE Menu  
(  
 MealID int NOT NULL,  
 SinceDate date NOT NULL DEFAULT GETDATE(),  
 CONSTRAINT Menu\_pk PRIMARY KEY (MealID)  
);

### OrderDetails

Przechowuje szczegóły każdego zamówienia (czyli siłą rzeczy jest również archiwum szczegółów o wcześniejszych zamówieniach).

Dodatkowe warunki:

* 1. Atrybut Quantity musi być dodatni, a UnitPrice nieujemny (formuła CHECK)

CREATE TABLE OrderDetails  
(  
 --ID int NOT NULL IDENTITY,  
 OrderID int NOT NULL,  
 MealID int NOT NULL,  
 Quantity int NOT NULL DEFAULT 0,  
 UnitPrice money NOT NULL DEFAULT 0,  
 CONSTRAINT pricesAndQuantities CHECK ((Quantity > 0 AND UnitPrice >= 0)),  
 CONSTRAINT OrderDetails\_pk PRIMARY KEY (MealID, OrderID)  
);

### OrderedFor

Przechowuje informacje, na których klientów firmowych składane było dane zamówienie.

CREATE TABLE OrderedFor  
(  
 OrderID int NOT NULL,  
 CCustomerID int NOT NULL,  
 CONSTRAINT OrderedFor\_pk PRIMARY KEY (OrderID, CCustomerID)  
);

### Orders

Przechowuje najważniejsze informacje o aktualnych i zrealizowanych zamówieniach.

Dodatkowe warunki:

* 1. Atrybuty ForHowLong oraz DiscountID mogą być wartościami null.
  2. SeatsNeeded musi być nieujemne.
  3. Klient jest albo firmą albo indywidualnym klientem

CREATE TABLE Orders  
(  
 OrderID int NOT NULL IDENTITY,  
 CustomerID int NULL,  
 DepID int NULL,  
 EmployeeID int NOT NULL,  
 WhenOrdered datetime NOT NULL,  
 WhenToBeDone datetime NOT NULL,  
 MealPrepared datetime NULL,  
 ForHowLong time NULL,  
 SeatsNeeded int NOT NULL,  
 DiscountID int NULL,  
 CurrencyName nvarchar(3) NOT NULL,  
 IfPaid bit NOT NULL,  
 CONSTRAINT seatsRange CHECK ((SeatsNeeded >= 0)),  
 CONSTRAINT OrderID PRIMARY KEY (OrderID),  
 CONSTRAINT ExactlyOneClient CHECK ((CustomerID is NULL AND DepID is not NULL) OR  
 (CustomerID is not NULL AND DepID is NULL)),  
 CONSTRAINT reservationSeats CHECK ((WhenOrdered = WhenToBeDone or SeatsNeeded = 0) or SeatsNeeded >= 2)  
);

### TableClasses

Zawiera klasy stolików jak np. w ogródku, koło kuchni, itd. Przy założeniu, że relacja aktualizowana jest na bieżąco, implementuje główne założenie.

CREATE TABLE TableClasses  
(  
 TClassID int NOT NULL IDENTITY,  
 TClassName nvarchar(100) NOT NULL,  
 TClassDescription nvarchar(100) NOT NULL,  
 CONSTRAINT TableClasses\_pk PRIMARY KEY (TClassID)  
);

### Tables

Przechowuje informacje o stolikach w restauracji.

Dodatkowe warunki:

* 1. SeatsNo powinno być z pomiędzy (1,100) - robocza wartość.

CREATE TABLE Tables  
(  
 TableID int NOT NULL IDENTITY,  
 SeatNo int NOT NULL,  
 TClassID int NOT NULL,  
 CONSTRAINT seatsConstraint CHECK ((SeatNo >= 0 AND SeatNo < 100) ),  
 CONSTRAINT Tables\_pk PRIMARY KEY (TableID)  
);

### TablesUsed

Przetrzymuje informacje o zajętych stolikach.

CREATE TABLE TablesUsed  
(  
 OrderID int NOT NULL,  
 TableID int NOT NULL,  
 CONSTRAINT TablesUsed\_pk PRIMARY KEY (TableID, OrderID)  
);

### Warehouse

Reprezentuje aktualny stan magazynu restauracji.

Dodatkowe warunki:

* 1. Atrybuty Pictures i Description mogą być wartościami null.
  2. UnitPrice oraz UnitsInStock muszą zawierać wartości >=0 (ustawione za pomocą formuły CHECK).

CREATE TABLE Warehouse  
(  
 MealID int NOT NULL IDENTITY,  
 UnitsInStock int NOT NULL DEFAULT 0,  
 UnitPrice money NOT NULL DEFAULT 0,  
 CategoryID int NOT NULL,  
 Description nvarchar(100) NULL,  
 Picture varbinary(max) NULL,  
 Name nvarchar(100) NOT NULL,  
 CONSTRAINT stockAndCosts CHECK ((UnitPrice >= 0 AND UnitsInStock >= 0)),  
 CONSTRAINT Warehouse\_pk PRIMARY KEY (MealID)  
);

### Variables

Przechowuje informacje o stałych liczbowych (do zniżek, rezerwacji itp.)

CREATE TABLE Variables  
(  
 VarName nvarchar(20),  
 VarValue decimal(6,2),  
 VarDescription nvarchar(100),  
 CONSTRAINT Variables\_pk PRIMARY KEY (VarName)  
);

### Klucze obce

-- foreign keys  
-- Reference: Cities\_Countries (table: Cities)  
ALTER TABLE Cities  
 ADD CONSTRAINT Cities\_Countries  
 FOREIGN KEY (CountryID)  
 REFERENCES Countries (CountryID);  
  
-- Reference: CustomerDepartments\_CompanyCustomers (table: CustomerDepartments)  
ALTER TABLE CustomerDepartments  
 ADD CONSTRAINT CustomerDepartments\_CompanyCustomers  
 FOREIGN KEY (CCustomersID)  
 REFERENCES CompanyCustomers (CCustomerID);  
  
-- Reference: CustomerDepartments\_Departaments (table: CustomerDepartments)  
ALTER TABLE CustomerDepartments  
 ADD CONSTRAINT CustomerDepartments\_Departaments  
 FOREIGN KEY (DepID)  
 REFERENCES Departments (DepID);  
  
-- Reference: Customers\_Discounts (table: Discounts)  
ALTER TABLE Discounts  
 ADD CONSTRAINT Customers\_Discounts  
 FOREIGN KEY (CustomerID)  
 REFERENCES Customers (CustomerID);  
  
-- Reference: Departaments\_Cities (table: Departments)  
ALTER TABLE Departments  
 ADD CONSTRAINT Departaments\_Cities  
 FOREIGN KEY (CityID)  
 REFERENCES Cities (CityID);  
  
-- Reference: Departaments\_Companies (table: Departments)  
ALTER TABLE Departments  
 ADD CONSTRAINT Departaments\_Companies  
 FOREIGN KEY (CompanyID)  
 REFERENCES Companies (CompanyID);  
  
-- Reference: Employees\_Cities (table: Employees)  
ALTER TABLE Employees  
 ADD CONSTRAINT Employees\_Cities  
 FOREIGN KEY (CityID)  
 REFERENCES Cities (CityID);  
  
-- Reference: Employees\_Employees (table: Employees)  
ALTER TABLE Employees  
 ADD CONSTRAINT Employees\_Employees  
 FOREIGN KEY (SuperiorID)  
 REFERENCES Employees (EmployeeID);  
  
-- Reference: Employees\_Jobs (table: Employees)  
ALTER TABLE Employees  
 ADD CONSTRAINT Employees\_Jobs  
 FOREIGN KEY (JobID)  
 REFERENCES Jobs (JobID);  
  
-- Reference: Menu\_Warehouse (table: Menu)  
ALTER TABLE Menu  
 ADD CONSTRAINT Menu\_Warehouse  
 FOREIGN KEY (MealID)  
 REFERENCES Warehouse (MealID);  
  
-- Reference: OrderDetails\_Menu (table: OrderDetails)  
ALTER TABLE OrderDetails  
 ADD CONSTRAINT OrderDetails\_Warehouse  
 FOREIGN KEY (MealID)  
 REFERENCES Warehouse (MealID);  
  
-- Reference: OrderedFor\_CompanyCustomers (table: OrderedFor)  
ALTER TABLE OrderedFor  
 ADD CONSTRAINT OrderedFor\_CompanyCustomers  
 FOREIGN KEY (CCustomerID)  
 REFERENCES CompanyCustomers (CCustomerID);  
  
-- Reference: Orders\_Currencies (table: Orders)  
ALTER TABLE Orders  
 ADD CONSTRAINT Orders\_Currencies  
 FOREIGN KEY (CurrencyName)  
 REFERENCES Currencies (CurrencyName);  
  
-- Reference: Orders\_Discounts (table: Orders)  
ALTER TABLE Orders  
 ADD CONSTRAINT Orders\_Discounts  
 FOREIGN KEY (DiscountID)  
 REFERENCES Discounts (DiscountID);  
  
-- Reference: Orders\_Employees (table: Orders)  
ALTER TABLE Orders  
 ADD CONSTRAINT Orders\_Employees  
 FOREIGN KEY (EmployeeID)  
 REFERENCES Employees (EmployeeID);  
  
-- Reference: Orders\_Departaments (table: Orders)  
ALTER TABLE Orders  
 ADD CONSTRAINT Orders\_Departments  
 FOREIGN KEY (DepID)  
 REFERENCES Departments (DepID);  
  
-- Reference: Orders\_Employees (table: Orders)  
ALTER TABLE Orders  
 ADD CONSTRAINT Orders\_Customers  
 FOREIGN KEY (CustomerID)  
 REFERENCES Customers (CustomerID);  
  
-- Reference: Orders\_OrderDetails (table: OrderDetails)  
ALTER TABLE OrderDetails  
 ADD CONSTRAINT Orders\_OrderDetails  
 FOREIGN KEY (OrderID)  
 REFERENCES Orders (OrderID) ON DELETE CASCADE ;  
  
-- Reference: Orders\_OrderedFor (table: OrderedFor)  
ALTER TABLE OrderedFor  
 ADD CONSTRAINT Orders\_OrderedFor  
 FOREIGN KEY (OrderID)  
 REFERENCES Orders (OrderID);  
  
-- Reference: Orders\_TablesUsed (table: TablesUsed)  
ALTER TABLE TablesUsed  
 ADD CONSTRAINT Orders\_TablesUsed  
 FOREIGN KEY (OrderID)  
 REFERENCES Orders (OrderID);  
  
-- Reference: TablesUsed\_Tables (table: TablesUsed)  
ALTER TABLE TablesUsed  
 ADD CONSTRAINT TablesUsed\_Tables  
 FOREIGN KEY (TableID)  
 REFERENCES Tables (TableID);  
  
-- Reference: Tables\_TableClasses (table: Tables)  
ALTER TABLE Tables  
 ADD CONSTRAINT Tables\_TableClasses  
 FOREIGN KEY (TClassID)  
 REFERENCES TableClasses (TClassID);  
  
-- Reference: Warehouse\_Categories (table: Warehouse)  
ALTER TABLE Warehouse  
 ADD CONSTRAINT Warehouse\_Categories  
 FOREIGN KEY (CategoryID)  
 REFERENCES Categories (CategoryID);