SYSTEM REZERWACJI HOTELOWEJ

*DOKUMENTACJA BAZY DANYCH*



Skład grupy:

Natalia Owczarek  
Dominika Sybilska  
Barbara Wolska  
Olga Sadokierska  
Izabela Najder

Spis treści

[1. Opis tła projektowanego i implementowanego zagadnienia 3](#_Toc129686240)

[2. Opis wymagań funkcjonalnych i niefunkcjonalnych 3](#_Toc129686241)

[3. Model fizyczny lub schemat bazy danych w postaci graficznej 4](#_Toc129686242)

[4. Skrypt tworzenia bazy danych 5](#_Toc129686243)

[5. Wykaz utworzonych tabel fizycznych 5](#_Toc129686244)

[6. Skrypt tworzenia tabel i referencji między nimi 5](#_Toc129686245)

[7. Wykaz zaimplementowanych indeksów 5](#_Toc129686246)

[8. Skrypt tworzenia indeksów 9](#_Toc129686247)

[9. Skrypt ładujący przykładowe dane do tabel 11](#_Toc129686248)

[10. Wykaz utworzonych widoków 14](#_Toc129686249)

[11. Skrypt tworzenia widoków 14](#_Toc129686250)

[12. Wykaz utworzonych procedur składowanych 18](#_Toc129686251)

[13. Skrypt tworzenia procedur składowanych 18](#_Toc129686252)

[14. Wykaz utworzonych funkcji użytkownika 18](#_Toc129686253)

[15. Skrypt tworzenia funkcji definiowanych przez użytkownika 18](#_Toc129686254)

[16. Wykaz utworzonych wyzwalaczy 19](#_Toc129686255)

[17. Skrypt tworzenia wyzwalaczy 19](#_Toc129686256)

[18. Wykaz obiektów wartości domyślnej i reguł 20](#_Toc129686257)

[19. Skrypt tworzenia obiektów wartości domyślnej i reguł 20](#_Toc129686258)

[20. Wykaz typów danych tworzonych przez użytkownika 20](#_Toc129686259)

[21. Skrypt tworzenia typów danych definiowanych przez użytkownika 20](#_Toc129686260)

[22. Schemat zaprojektowanego systemu przedstawiający punkty wejścia i wyjścia 20](#_Toc129686261)

[23. Wnioski i podsumowanie 20](#_Toc129686262)

[24. Wykaz/spis ilustracji 21](#_Toc129686263)

[25. Wykaz/spis tabel 21](#_Toc129686264)

# Opis tła projektowanego i implementowanego zagadnienia

Przedmiotem naszego projektu jest sieć hoteli o nazwie „Hotel D&B&N”, która dąży do osiągnięcia wysokiej pozycji na rynku hotelarskim. Przedstawiona w projekcie sieć hotelarska posiada swoje placówki w kilku lokalizacjach na terenie Polski. „Hotel D&B&N” dba o satysfakcje klientów, dlatego w swojej ofercie posiada różnorodne pokoje, takie jak studio, wieloosobowe, czy Superior. System bazy danych oferuje możliwość rezerwacji pokoju w wybranej lokacji i dacie. System zawiera również informacje na temat pracowników sieci hoteli.

Baza danych zawiera kompleksowe informacje na temat placówek hotelu oraz dostępnych pokoi wraz z ich statusem i typem. System pozwala na dodawanie nowych rezerwacji i przechowywanie informacji na temat zrealizowanych i odwołanych rezerwacji. System zawiera również wszelkie informacje na temat gości oraz ich płatności. System pomaga śledzić transakcje, dzięki zbieraniu danych na temat płatności oraz sposobach płatności, dzięki czemu umożliwia szczegółowe kontrolowanie i tworzenie danych do faktur.

Wdrożyliśmy funkcjonalność umożliwiającą składanie rezerwacji z wyprzedzeniem, a także przypisywanie wielu rezerwacji do wielu klientów. Ponadto, istnieje możliwość zbierania informacji potrzebnych do wystawienia faktur, co usprawni proces rozliczeń i ułatwi obsługę klientów. Dzięki tym elementom pracownik jest w stanie lepiej zarządzać rezerwacjami i świadczyć jeszcze lepsze usługi dla klientów.

Wszystkie elementy wymienione powyżej są integralnymi częściami bazy danych sieci hoteli "Hotel D&B&N", która działa na potrzeby zarządzania hotelem, zapewniając kompleksową obsługę klientów oraz usprawniając procesy wewnętrzne.

# Opis wymagań funkcjonalnych i niefunkcjonalnych

Wymagania funkcjonalne:

* Dodawanie rezerwacji
* Dodawanie wypożyczenia
* Dodawanie faktury
* Modyfikacja faktury
* Usunięcie faktury
* Dodawanie klientów
* Modyfikacja klienta
* Dodawanie pokoju
* Modyfikacja pokoju
* Usunięcie pokoju
* Dodanie płatności
* Modyfikacja płatności
* Dodawanie klientów
* Modyfikacja klientów

Wymagania niefunkcjonalne:

* Brak możliwości usunięcia placówki hotelu
* Brak możliwości usunięcia rezerwacji pokoju
* Nie ma możliwości usunięcia faktury z bieżącego roku
* Nie można zarezerwować już zajętego pokoju
* Nie można usunąć płatności z bieżącego roku

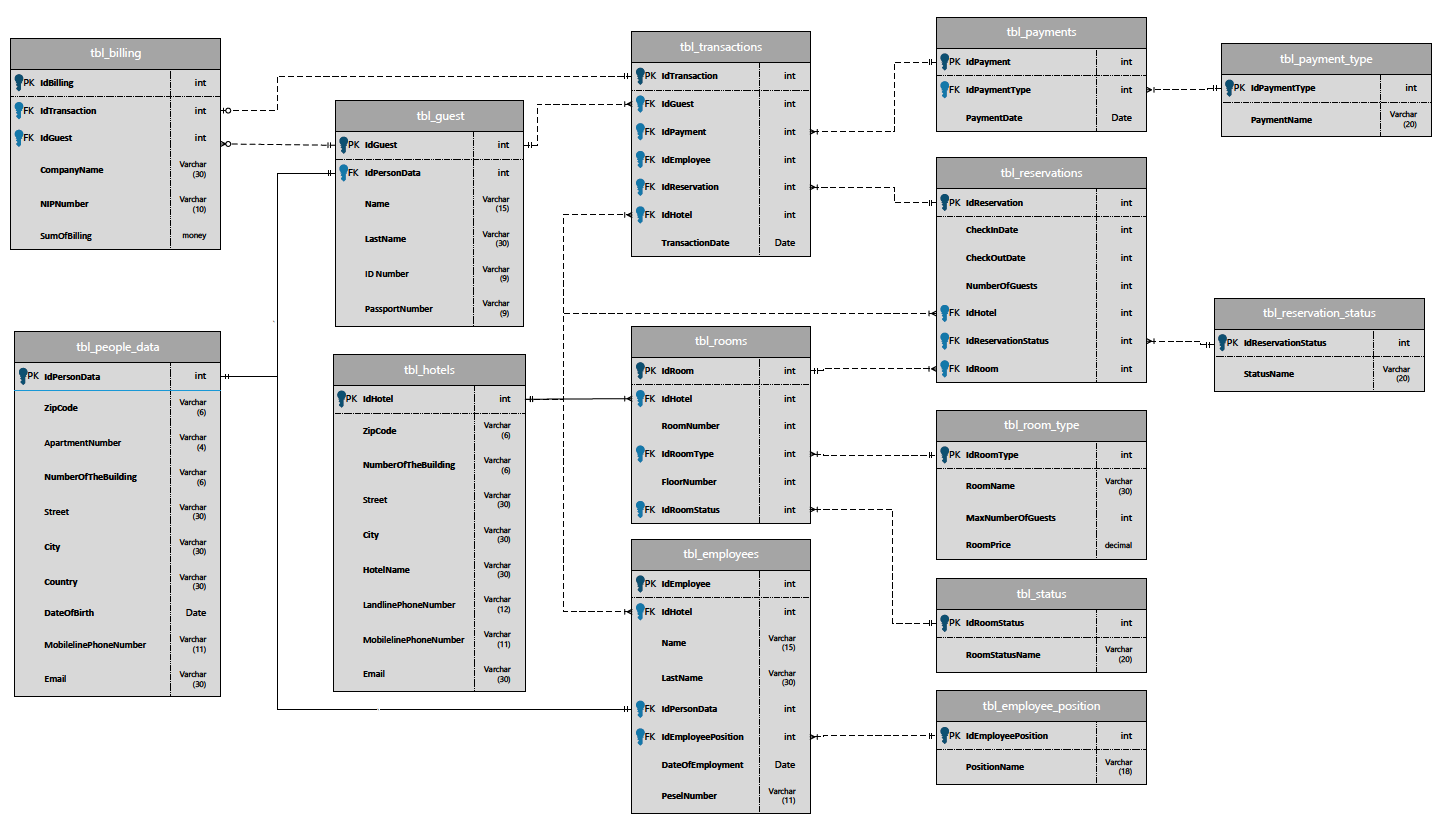
# Model fizyczny lub schemat bazy danych w postaci graficznej

Model logiczny

Obraz zawierający diagram

Opis wygenerowany automatycznie

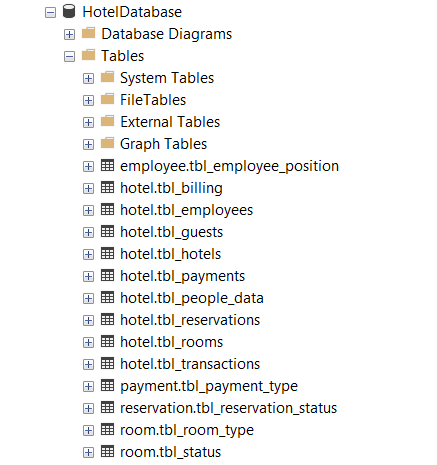
Model fizyczny



# Skrypt tworzenia bazy danych

CREATE DATABASE HotelDatabase;

# Wykaz utworzonych tabel fizycznych



# Skrypt tworzenia tabel i referencji między nimi

USE HotelDatabase

GO

CREATE SCHEMA hotel

GO

CREATE SCHEMA room

GO

CREATE SCHEMA employee

GO

CREATE SCHEMA payment

GO

CREATE SCHEMA reservation

GO

USE HotelDatabase

GO

CREATE TABLE room.tbl\_status

(

IdRoomStatus int NOT NULL IDENTITY(1,1) PRIMARY KEY,

RoomStatusName varchar(20) NOT NULL

)

ON Dictionaries;

GO

CREATE TABLE room.tbl\_room\_type

(

IdRoomType int NOT NULL IDENTITY(1,1) PRIMARY KEY,

RoomName varchar(30) NOT NULL,

MaxNumberOfGuests int NOT NULL,

RoomPrice decimal NOT NULL,

)

ON Dictionaries;

CREATE TABLE employee.tbl\_employee\_position

(

IdEmployeePosition int NOT NULL IDENTITY(1,1) PRIMARY KEY,

PositionName varchar(18) NOT NULL,

)

ON Dictionaries;

CREATE TABLE payment.tbl\_payment\_type

(

IdPaymentType int NOT NULL IDENTITY(1,1) PRIMARY KEY,

PaymentName varchar(20) NOT NULL,

)

ON Dictionaries;

CREATE TABLE reservation.tbl\_reservation\_status

(

IdReservationStatus int NOT NULL IDENTITY(1,1) PRIMARY KEY,

StatusName varchar(20) NOT NULL,

)

ON Dictionaries;

CREATE TABLE hotel.tbl\_hotels

(

IdHotel int NOT NULL IDENTITY(1,1) PRIMARY KEY,

ZipCode varchar(6) NOT NULL,

NumberOfTheBuilding varchar(6) NOT NULL,

Street varchar(30) NULL,

City varchar(30) NOT NULL,

HotelName varchar(30) NOT NULL,

LandlinePhoneNumber varchar(12) NULL,

MobilelinePhoneNumber varchar(11) NULL,

Email varchar(30) NULL,

);

GO

CREATE TABLE hotel.tbl\_rooms

(

IdRoom int NOT NULL IDENTITY(1,1) PRIMARY KEY,

IdRoomType int NOT NULL FOREIGN KEY REFERENCES room.tbl\_room\_type(IdRoomType) ON UPDATE CASCADE,

IdRoomStatus int NOT NULL FOREIGN KEY REFERENCES room.tbl\_status(IdRoomStatus) ON UPDATE CASCADE,

RoomNumber int NOT NULL,

FloorNumber int NOT NULL,

IdHotel int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_hotels(IdHotel) ON UPDATE CASCADE,

);

GO

CREATE TABLE hotel.tbl\_reservations

(

IdReservation int NOT NULL IDENTITY(1,1) PRIMARY KEY,

IdRoom int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_rooms(IdRoom),

CheckInDate date NOT NULL,

CheckOutDate date NOT NULL,

NumberOfGuests varchar(3) NOT NULL,

IdReservationStatus int NOT NULL FOREIGN KEY REFERENCES reservation.tbl\_reservation\_status(IdReservationStatus),

IdHotel int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_hotels(IdHotel),

);

GO

CREATE TABLE hotel.tbl\_people\_data

(

IdPersonData int NOT NULL IDENTITY(1,1) PRIMARY KEY,

ZipCode varchar(6) NOT NULL,

ApartmentNumber varchar(4) NULL,

NumberOfTheBuilding varchar(6) NOT NULL,

Street varchar(30) NULL,

City varchar(30) NOT NULL,

Country varchar(30) NOT NULL,

DateOfBirth date NOT NULL CHECK ((YEAR(GETDATE()) - YEAR(DateOfBirth)) >= 18),

MobilelinePhoneNumber varchar(11) NULL,

Email varchar(30) NULL,

);

GO

CREATE TABLE hotel.tbl\_guests

(

IdGuest int NOT NULL IDENTITY(1,1) PRIMARY KEY,

IdPersonData int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_people\_data(IdPersonData) ON UPDATE CASCADE,

[Name] varchar(15) NOT NULL,

LastName varchar(30) NOT NULL,

IDNumber varchar(9) NOT NULL UNIQUE,

PassportNumber varchar(9) NOT NULL UNIQUE,

);

GO

CREATE TABLE hotel.tbl\_employees

(

IdEmployee int NOT NULL IDENTITY(1,1) PRIMARY KEY,

IdPersonData int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_people\_data(IdPersonData) ON UPDATE CASCADE,

[Name] varchar(15) NOT NULL,

LastName varchar(30) NOT NULL,

IdEmployeePosition int NOT NULL FOREIGN KEY REFERENCES employee.tbl\_employee\_position(IdEmployeePosition) ON UPDATE CASCADE,

DateOfEmployment date NOT NULL CHECK (DateOfEmployment <= GETDATE()),

IdHotel int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_hotels(IdHotel) ON UPDATE CASCADE,

PeselNumber varchar(11) NOT NULL CHECK (dbo.uf\_IsPeselOK(PeselNumber) = 1)

);

GO

CREATE TABLE hotel.tbl\_payments

(

IdPayment int NOT NULL IDENTITY(1,1) PRIMARY KEY,

IdPaymentType int NOT NULL FOREIGN KEY REFERENCES payment.tbl\_payment\_type(IdPaymentType) ON UPDATE CASCADE,

PaymentDate date NOT NULL

);

GO

CREATE TABLE hotel.tbl\_transactions

(

IdTransaction int NOT NULL IDENTITY(1,1) PRIMARY KEY,

IdReservation int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_reservations(IdReservation),

IdPayment int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_payments(IdPayment),

IdEmployee int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_employees(IdEmployee),

IdGuest int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_guests(IdGuest),

IdHotel int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_hotels(IdHotel),

TransactionDate Date NOT NULL

);

GO

CREATE TABLE hotel.tbl\_billing

(

IdBilling int NOT NULL IDENTITY(1,1) PRIMARY KEY,

IdTransaction int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_transactions(IdTransaction) ON UPDATE CASCADE,

IdGuest int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_guests(IdGuest) ON UPDATE CASCADE,

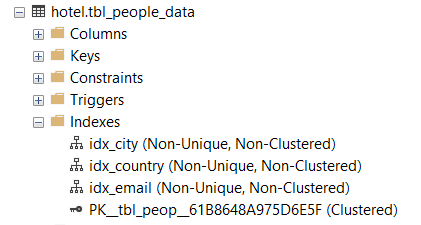
CompanyName varchar(30) NOT NULL,

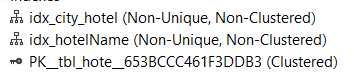
NIPNumber varchar(10) NOT NULL CHECK (dbo.uf\_IsNIPNumberOK(NIPNumber) = 1),

SumOfBilling money Null);

GO

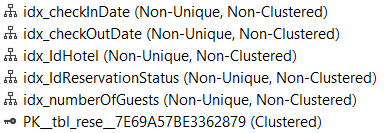
# Wykaz zaimplementowanych indeksów

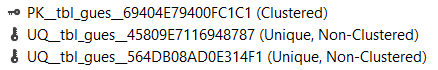


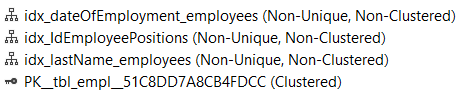


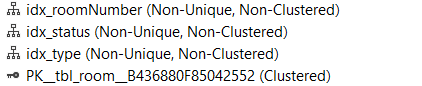
Obraz zawierający tekst

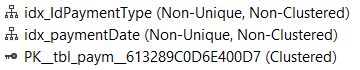
Opis wygenerowany automatycznie

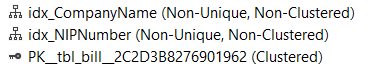












# Skrypt tworzenia indeksów

CREATE NONCLUSTERED INDEX [idx\_email] ON hotel.tbl\_people\_data

(

[Email] ASC

)

CREATE NONCLUSTERED INDEX [idx\_country] ON hotel.tbl\_people\_data

(

[Country] ASC

)

CREATE NONCLUSTERED INDEX [idx\_city] ON hotel.tbl\_people\_data

(

[City] ASC

)

CREATE NONCLUSTERED INDEX [idx\_city\_hotel] ON hotel.tbl\_hotels

(

[City] ASC

)

CREATE NONCLUSTERED INDEX [idx\_hotelName] ON hotel.tbl\_hotels

(

[HotelName] ASC

)

CREATE NONCLUSTERED INDEX [idx\_TransactionDate] ON hotel.tbl\_transactions

(

[TransactionDate] ASC

)

CREATE NONCLUSTERED INDEX [idx\_idReservation] ON hotel.tbl\_transactions

(

[IdReservation] ASC

)

CREATE NONCLUSTERED INDEX [idx\_IdPayment] ON hotel.tbl\_transactions

(

[IdPayment] ASC

)

CREATE NONCLUSTERED INDEX [idx\_checkInDate] ON hotel.tbl\_reservations

(

[CheckInDate] ASC

)

CREATE NONCLUSTERED INDEX [idx\_checkOutDate] ON hotel.tbl\_reservations

(

[CheckOutDate] ASC

)

CREATE NONCLUSTERED INDEX [idx\_numberOfGuests] ON hotel.tbl\_reservations

(

[NumberOfGuests] ASC

)

CREATE NONCLUSTERED INDEX [idx\_IdReservationStatus] ON hotel.tbl\_reservations

(

[IdReservationStatus] ASC

)

CREATE NONCLUSTERED INDEX [idx\_IdHotel] ON hotel.tbl\_reservations

(

[IdHotel] ASC

)

CREATE NONCLUSTERED INDEX [idx\_lastName] ON hotel.tbl\_guests

(

[LastName] ASC

)

CREATE NONCLUSTERED INDEX [idx\_lastName\_employees] ON hotel.tbl\_employees

(

[LastName] ASC

)

CREATE NONCLUSTERED INDEX [idx\_IdEmployeePositions] ON hotel.tbl\_employees

(

[IdEmployeePosition] ASC

)

CREATE NONCLUSTERED INDEX [idx\_dateOfEmployment\_employees] ON hotel.tbl\_employees

(

[DateOfEmployment] ASC

)

CREATE NONCLUSTERED INDEX [idx\_roomNumber] ON hotel.tbl\_rooms

(

[RoomNumber] ASC

)

CREATE NONCLUSTERED INDEX [idx\_status] ON hotel.tbl\_rooms

(

[IdRoomStatus] ASC

)

CREATE NONCLUSTERED INDEX [idx\_type] ON hotel.tbl\_rooms

(

[IdRoomType] ASC

)

CREATE NONCLUSTERED INDEX [idx\_paymentDate] ON hotel.tbl\_payments

(

[PaymentDate] ASC

)

CREATE NONCLUSTERED INDEX [idx\_IdPaymentType] ON hotel.tbl\_payments

(

[IdPaymentType] ASC

)

CREATE NONCLUSTERED INDEX [idx\_NIPNumber] ON hotel.tbl\_billing

(

[NIPNumber] ASC

)

CREATE NONCLUSTERED INDEX [idx\_CompanyName] ON hotel.tbl\_billing

(

[CompanyName] ASC

)

# Skrypt ładujący przykładowe dane do tabel

INSERT INTO room.tbl\_status (RoomStatusName) VALUES

('clean'),

('occupied'),

('maintenance'),

('out of order'),

('reserved');

INSERT INTO room.tbl\_room\_type (RoomName, MaxNumberOfGuests, RoomPrice) VALUES

('Single Room', 1, 100.00),

('Double Room', 2, 150.00),

('Twin Room', 2, 150.00),

('Triple Room', 3, 200.00),

('Quad Room', 4, 250.00),

('Deluxe Double Room', 2, 300.00),

('Deluxe Twin Room', 2, 300.00),

('Junior Suite', 2, 400.00),

('Executive Suite', 2, 500.00),

('Presidential Suite', 2, 1000.00);

INSERT INTO employee.tbl\_employee\_position (PositionName)

VALUES ('Manager'), ('Receptionist'), ('Cleaner'), ('Cook'), ('Waiter'), ('Bartender'), ('Security Guard'), ('IT Specialist'), ('Accountant'), ('HR Specialist');

INSERT INTO payment.tbl\_payment\_type (PaymentName) VALUES

('Cash'),

('Credit Card'),

('Debit Card'),

('Bank Transfer'),

('PayPal'),

('Google Pay'),

('Apple Pay'),

('Bitcoin'),

('Ethereum'),

('Dogecoin');

INSERT INTO reservation.tbl\_reservation\_status (StatusName) VALUES

('Pending'),

('Confirmed'),

('Checked In'),

('Checked Out'),

('Cancelled'),

('No Show'),

('Expired'),

('Invoiced'),

('Paid'),

('Partially Paid');

INSERT INTO hotel.tbl\_hotels (ZipCode, NumberOfTheBuilding, Street, City, HotelName, LandlinePhoneNumber, MobilelinePhoneNumber, Email)

VALUES

('00-001', '1', 'First Street', 'Warsaw', 'The First Hotel', NULL, '48678901244', 'info@thefirsthotel.com'),

('00-002', '2', 'Second Street', 'Lodz', 'The Second Hotel', NULL, '48678901234', 'info@thesecondhotel.com'),

('00-003', '3', 'Third Street', 'Poznan', 'The Third Hotel', NULL, '48500123458', 'info@thethirdhotel.com'),

('00-004', '4', 'Fourth Street', 'Warsaw', 'The Fourth Hotel', NULL, '48500123459', 'info@thefourthhotel.com'),

('00-005', '5', 'Fifth Street', 'Gdansk', 'The Fifth Hotel', NULL, '48500123450', 'info@thefifthhotel.com'),

('00-006', '6', 'Sixth Street', 'Szczecin', 'The Sixth Hotel', NULL, '48500123451', 'info@thesixthhotel.com'),

('00-007', '7', 'Seventh Street', 'Gdynia', 'The Seventh Hotel', NULL, '48500123452', 'info@theseventhhotel.com'),

('00-008', '8', 'Eighth Street', 'Sopot', 'The Eighth Hotel', NULL, '48500123453', 'info@theeighthhotel.com'),

('00-009', '9', 'Ninth Street', 'Zakopane', 'The Ninth Hotel', NULL, '48500123454', 'info@theninthhotel.com'),

('00-010', '10', 'Tenth Street', 'Cracow', 'The Tenth Hotel', NULL, '48500123455', 'info@thetenthhotel.com');

INSERT INTO hotel.tbl\_people\_data (ZipCode, ApartmentNumber, NumberOfTheBuilding, Street, City, Country, DateOfBirth, MobilelinePhoneNumber, Email)

VALUES

('12-345', '12', '45', 'Black Street', 'Lodz', 'Poland', '1985-01-01', '48777777777', 'example1@example.com'),

('23-456', NULL, '34', 'Blue Street', 'Berlin', 'Germany', '1990-02-12', '48555555555', 'example2@example.com'),

('34-567', '22', '5', 'Orange Street', 'Galway', 'Ireland', '1988-08-15', '48666666666', 'example3@example.com'),

('45-678', '33', '11', 'White Street', 'Warsaw', 'Poland', '1977-05-03', '48555555566', 'example4@example.com'),

('56-789', '44', '9', 'Brown Street', 'Glasgow', 'Scotland', '1995-12-25', '48777777678', 'example5@example.com'),

('67-890', NULL, '2', 'Yellow Street', 'Gdansk', 'Poland', '1992-07-09', '48666666666', 'example6@example.com'),

('78-901', '7', '14', 'Color Street', 'Copenhagen', 'Denmark', '1981-04-22', '48777777777', 'example7@example.com'),

('78-901', '7', '14', 'Red Street', 'Oslo', 'Norway', '1999-05-03', '48777777778', 'example8@example.com'),

('78-901', '7', '14', 'Green Street', 'Poznan', 'Poland', '2001-07-15', '48777777779', 'example9@example.com'),

('78-901', '7', '14', 'Violet Street', 'Lodz', 'Poland', '2002-11-16', '48777777776', 'example10@example.com');

INSERT INTO hotel.tbl\_rooms (IdRoomType, IdRoomStatus, RoomNumber, FloorNumber, IdHotel)

VALUES (1, 1, 101, 1, 1),

(2, 1, 102, 2, 9),

(3, 2, 103, 3, 3),

(4, 2, 104, 4, 4),

(5, 3, 201, 1, 1),

(6, 3, 202, 2, 2),

(7, 4, 203, 3, 2),

(8, 4, 204, 4, 1),

(9, 5, 301, 3, 8),

(10, 5, 302, 3, 1);

INSERT INTO hotel.tbl\_reservations (IdRoom, CheckInDate, CheckOutDate, NumberOfGuests, IdReservationStatus, IdHotel)

VALUES (1, '2023-04-01', '2023-04-05', '2', 1, 1),

(2, '2023-04-02', '2023-04-06', '3', 2, 2),

(3, '2023-04-01', '2023-04-03', '1', 3, 3),

(4, '2023-04-07', '2023-04-10', '2', 4, 4),

(5, '2023-04-15', '2023-04-18', '1', 5, 5),

(6, '2023-04-10', '2023-04-14', '3', 6, 6),

(7, '2023-04-12', '2023-04-15', '2', 7, 7),

(8, '2023-04-20', '2023-04-23', '1', 8, 8),

(9, '2023-04-21', '2023-04-25', '4', 9, 9),

(10, '2023-04-30', '2023-05-03', '2', 10, 10);

INSERT INTO hotel.tbl\_guests (IdPersonData, Name, LastName, IDNumber, PassportNumber)

VALUES

(1, 'John', 'Doe', 'ABC123456', 'AB1234567'),

(2, 'Jane', 'Doe', 'DEF987654', 'DE7654321'),

(3, 'David', 'Smith', 'GHI456789', 'GH9876543'),

(4, 'Emily', 'Johnson', 'JKL234567', 'JK4321098'),

(5, 'Michael', 'Brown', 'MNO876543', 'MN6543210'),

(6, 'Sophia', 'Garcia', 'PQR345678', 'PQ0987654'),

(7, 'Robert', 'Taylor', 'STU912345', 'ST8765432'),

(8, 'Olivia', 'Wilson', 'VWX678901', 'VW7654321'),

(9, 'William', 'Martinez', 'YZA234567', 'YZ1234567'),

(10, 'Ava', 'Anderson', 'BCD890123', 'BC0987654');

INSERT INTO hotel.tbl\_employees (IdPersonData, Name, LastName, IdEmployeePosition, DateOfEmployment, IdHotel, PeselNumber)

VALUES

(1, 'Anna', 'Nowak', 1, '2022-01-01', 1, '58110187843'),

(2, 'Jan', 'Kowalski', 2, '2021-06-01', 2, '92061115432'),

(3, 'Katarzyna', 'Wójcik', 3, '2020-03-01', 3, '55090228281'),

(4, 'Piotr', 'Kowalczyk', 4, '2022-02-15', 1, '86112485874'),

(5, 'Marta', 'Lewandowska', 5, '2021-09-01', 2, '74092062244'),

(6, 'Michał', 'Szymański', 6, '2022-01-15', 3, '54042475931'),

(7, 'Kamil', 'Jankowski', 7, '2021-11-01', 1, '52102883577'),

(8, 'Karolina', 'Wójcik', 8, '2022-03-01', 2, '04301014645'),

(9, 'Adam', 'Kaczmarek', 9, '2021-08-01', 3, '56082015957'),

(10, 'Izabela', 'Dąbrowska', 10, '2022-02-01', 1, '66031247728');

INSERT INTO hotel.tbl\_payments (IdPaymentType, PaymentDate)

VALUES

(1, '2022-01-01'),

(2, '2022-02-03'),

(3, '2022-01-15'),

(4, '2022-02-07'),

(5, '2022-03-10'),

(6, '2022-04-21'),

(7, '2022-03-18'),

(8, '2022-02-28'),

(9, '2022-05-05'),

(10, '2022-04-12');

INSERT INTO hotel.tbl\_transactions (IdReservation, IdPayment, IdEmployee, IdGuest, IdHotel, TransactionDate)

VALUES (1, 1, 2, 3, 1, '2023-03-14'),

(2, 2, 3, 4, 1, '2023-03-15'),

(3, 3, 1, 5, 2, '2023-03-16'),

(4, 4, 2, 6, 2, '2023-03-17'),

(5, 5, 3, 7, 3, '2023-03-18'),

(6, 6, 4, 8, 3, '2023-03-19'),

(7, 7, 1, 9, 4, '2023-03-20'),

(8, 8, 2, 10, 4, '2023-03-21'),

(9, 9, 3, 1, 5, '2023-03-22'),

(10, 10, 4, 2, 5, '2023-03-23');

INSERT INTO hotel.tbl\_billing (IdGuest, IdTransaction, CompanyName, NIPNumber)

VALUES

(1, 1, 'ABC Company', '1234567890'),

(2, 2, 'XYZ Corporation', '0987654325'),

(3, 3, 'PQR Industries', '1357902463'),

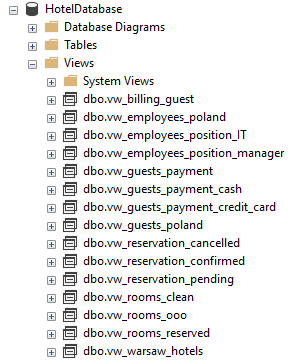
(4, 4, 'LMN Enterprises', '8642097532'),

(5, 5, 'DEF Inc.', '2468013578'),

(6, 6, 'GHI Corporation', '9753108647'),

(7, 7, 'JKL Industries', '7049182534');

# Wykaz utworzonych widoków



# Skrypt tworzenia widoków

-- Tworzenie widoku vw\_warsaw\_hotels

CREATE VIEW vw\_warsaw\_hotels

AS

SELECT

h.HotelName,

h.City,

h.Street,

h.NumberOfTheBuilding,

h.ZipCode

FROM

[hotel].[tbl\_hotels] h

WHERE

h.City = 'Warsaw';

GO

-- Tworzenie widoku vw\_employees\_poland

CREATE VIEW vw\_employees\_poland

AS

SELECT

e.Name,

e.LastName,

e.PeselNumber,

p.DateOfBirth,

p.Email,

p.Country

FROM

[hotel].[tbl\_people\_data] p, [hotel].[tbl\_employees] e

WHERE

p.Country = 'Poland' AND p.IdPersonData = e.IdPersonData;

GO

-- Tworzenie widoku vw\_guests\_poland

CREATE VIEW vw\_guests\_poland

AS

SELECT

g.Name,

g.LastName,

g.PassportNumber,

p.DateOfBirth,

p.Email,

p.Country

FROM

[hotel].[tbl\_people\_data] p, [hotel].[tbl\_guests] g

WHERE

p.Country = 'Poland' AND p.IdPersonData = g.IdPersonData;

GO

-- Tworzenie widoku vw\_employees\_position\_IT

CREATE VIEW vw\_employees\_position\_IT

AS

SELECT

e.Name,

e.LastName,

ep.PositionName

FROM

[hotel].[tbl\_employees] e, [employee].[tbl\_employee\_position] ep

WHERE

ep.PositionName = 'IT Specialist' AND e.IdEmployeePosition = ep.IdEmployeePosition;

GO

-- Tworzenie widoku vw\_employees\_position\_manager

CREATE VIEW vw\_employees\_position\_manager

AS

SELECT

e.Name,

e.LastName,

ep.PositionName

FROM

[hotel].[tbl\_employees] e, [employee].[tbl\_employee\_position] ep

WHERE

ep.PositionName = 'Manager' AND e.IdEmployeePosition = ep.IdEmployeePosition;

GO

-- Tworzenie widoku vw\_rooms\_reserved

CREATE VIEW vw\_rooms\_reserved

AS

SELECT

rt.RoomName,

rt.RoomPrice,

r.RoomNumber,

r.FloorNumber,

rt.MaxNumberOfGuests,

rs.RoomStatusName

FROM

[hotel].[tbl\_rooms] r, [room].[tbl\_room\_type] rt, [room].[tbl\_status] rs

WHERE

rs.RoomStatusName = 'reserved' AND rs.IdRoomStatus = r.IdRoomStatus AND rt.IdRoomType = r.IdRoomType;

GO

-- Tworzenie widoku vw\_rooms\_clean

CREATE VIEW vw\_rooms\_clean

AS

SELECT

rt.RoomName,

rt.RoomPrice,

r.RoomNumber,

r.FloorNumber,

rt.MaxNumberOfGuests,

rs.RoomStatusName

FROM

[hotel].[tbl\_rooms] r, [room].[tbl\_room\_type] rt, [room].[tbl\_status] rs

WHERE

rs.RoomStatusName = 'clean' AND rs.IdRoomStatus = r.IdRoomStatus AND rt.IdRoomType = r.IdRoomType;

GO

-- Tworzenie widoku vw\_rooms\_ooo

CREATE VIEW vw\_rooms\_ooo

AS

SELECT

rt.RoomName,

rt.RoomPrice,

r.RoomNumber,

r.FloorNumber,

rt.MaxNumberOfGuests,

rs.RoomStatusName

FROM

[hotel].[tbl\_rooms] r, [room].[tbl\_room\_type] rt, [room].[tbl\_status] rs

WHERE

rs.RoomStatusName = 'out of order' AND rs.IdRoomStatus = r.IdRoomStatus AND rt.IdRoomType = r.IdRoomType;

GO

-- Tworzenie widoku vw\_reservation\_pending

CREATE VIEW vw\_reservation\_pending

AS

SELECT

h.HotelName,

roo.RoomNumber,

rs.StatusName

FROM

[hotel].[tbl\_reservations] res, [reservation].[tbl\_reservation\_status] rs, [hotel].[tbl\_rooms] roo, [hotel].[tbl\_hotels] h

WHERE

rs.StatusName = 'Pending' AND rs.IdReservationStatus = res.IdReservationStatus AND res.IdRoom = roo.IdRoom AND res.IdHotel = h.IdHotel;

GO

-- Tworzenie widoku vw\_reservation\_confirmed

CREATE VIEW vw\_reservation\_confirmed

AS

SELECT

h.HotelName,

roo.RoomNumber,

rs.StatusName

FROM

[hotel].[tbl\_reservations] res, [reservation].[tbl\_reservation\_status] rs, [hotel].[tbl\_rooms] roo, [hotel].[tbl\_hotels] h

WHERE

rs.StatusName = 'Confirmed' AND rs.IdReservationStatus = res.IdReservationStatus AND res.IdRoom = roo.IdRoom AND res.IdHotel = h.IdHotel;

GO

-- Tworzenie widoku vw\_reservation\_cancelled

CREATE VIEW vw\_reservation\_cancelled

AS

SELECT

h.HotelName,

roo.RoomNumber,

rs.StatusName

FROM

[hotel].[tbl\_reservations] res, [reservation].[tbl\_reservation\_status] rs, [hotel].[tbl\_rooms] roo, [hotel].[tbl\_hotels] h

WHERE

rs.StatusName = 'Cancelled' AND rs.IdReservationStatus = res.IdReservationStatus AND res.IdRoom = roo.IdRoom AND res.IdHotel = h.IdHotel;

GO

-- Tworzenie widoku vw\_billing\_guest

CREATE VIEW vw\_billing\_guest

AS

SELECT

g.Name,

g.LastName,

g.PassportNumber,

b.CompanyName,

b.NIPNumber

FROM

[hotel].[tbl\_guests] g, [hotel].[tbl\_billing] b

WHERE

g.IdGuest = b.IdGuest;

GO

-- Tworzenie widoku vw\_guests\_payment

CREATE VIEW vw\_guests\_payment

AS

SELECT

g.Name,

g.LastName,

t.TransactionDate,

pt.PaymentName

FROM

[hotel].[tbl\_payments] p, [payment].[tbl\_payment\_type] pt, [hotel].[tbl\_transactions] t, [hotel].[tbl\_guests] g

WHERE

p.IdPaymentType = pt.IdPaymentType AND t.IdPayment = p.IdPayment AND t.IdGuest = g.IdGuest;

GO

-- Tworzenie widoku vw\_guests\_payment\_cash

CREATE VIEW vw\_guests\_payment\_cash

AS

SELECT

g.Name,

g.LastName,

t.TransactionDate,

pt.PaymentName

FROM

[hotel].[tbl\_payments] p, [payment].[tbl\_payment\_type] pt, [hotel].[tbl\_transactions] t, [hotel].[tbl\_guests] g

WHERE

pt.PaymentName = 'Cash' AND p.IdPaymentType = pt.IdPaymentType AND t.IdPayment = p.IdPayment AND t.IdGuest = g.IdGuest;

GO

-- Tworzenie widoku vw\_guests\_payment\_credit\_card

CREATE VIEW vw\_guests\_payment\_credit\_card

AS

SELECT

g.Name,

g.LastName,

t.TransactionDate,

pt.PaymentName

FROM

[hotel].[tbl\_payments] p, [payment].[tbl\_payment\_type] pt, [hotel].[tbl\_transactions] t, [hotel].[tbl\_guests] g

WHERE

pt.PaymentName = 'Credit Card' AND p.IdPaymentType = pt.IdPaymentType AND t.IdPayment = p.IdPayment AND t.IdGuest = g.IdGuest;

GO

# 12. Wykaz utworzonych procedur składowanych

# 13. Skrypt tworzenia procedur składowanych

# 14. Wykaz utworzonych funkcji użytkownika

Obraz zawierający tekst

Opis wygenerowany automatycznie

# 15. Skrypt tworzenia funkcji definiowanych przez użytkownika

* 1. Funkcje tabelaryczne

exec sp\_helptext uf\_SearchGuestByNameOrLastName

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

--- FUNCTION DEFINITION

--- uf\_SearchGuestByNameOrLastName (@NameOrSurname varchar(60))

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

-- wyszukiwanie informacji o gościu na podstawie fragmentu imienia lub nazwiksa

--

-- parametry wejściowe:

-- @NameOrSurname - wyszukiwane imię lub nazwisko gościa

--

--parametry wyjściowe/zwracane wartości:

--zwraca tabelę z informacjami o gościu, który został wyszukany

--

/\*

use HotelDatabase

go

Select uf\_SearchGuestByNameOrLastName ('Doe')

--

--Wynik działania

--Zwraca 2 rekordy tabeli, ponieważ w bazie znajdują się 2 osoby o nazwisku 'Doe'

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

\*/

use HotelDatabase

go

CREATE OR ALTER FUNCTION uf\_SearchGuestByNameOrLastName(@NameOrSurname varchar(60))

RETURNS TABLE

AS

RETURN(

SELECT g.IdGuest,g.IDNumber, g.LastName, g.Name, g.PassportNumber, pd.\*

FROM [hotel].[tbl\_guests] as g inner join [hotel].[tbl\_people\_data] as pd

on g.IdPersonData=pd.IdPersonData

WHERE LOWER([Name])+' '+LOWER([LastName]) LIKE LOWER('%'+@NameOrSurname+'%'))

GO

exec sp\_helptext uf\_NumberOfGuestReservations

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

--- FUNCTION DEFINITION

--- uf\_NumberOfGuestReservations (@NameOrLastName varchar(50))

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

-- wyszukiwanie liczby rezerwacji na podstawie fragmentu imienia lub nazwiska gościa

--

-- parametry wejściowe:

-- @NameOrLastName varchar(50) - fragment imienia lub nazwiska

--

--parametry wyjściowe/zwracane wartości:

-- Zwraca tabelę wyników

--

/\*

use HotelDatabase

go

Select uf\_NumberOfGuestReservations('Doe')

--

--Wynik działania

--Wynikiem działania są 2 wiersze, ponieważ w bazie znajdują się 2 osoby o nazwisku 'Doe'

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

\*/

use HotelDatabase

go

CREATE OR ALTER FUNCTION uf\_NumberOfGuestReservations

(@NameOrLastName varchar(50))

RETURNS TABLE

AS

RETURN(

SELECT p.Name, p.LastName, p.IdGuest, COUNT(r.IdReservation) AS NumberOfReservation

FROM (SELECT g.Name, g.LastName, g.IdGuest, t.IdReservation

FROM [hotel].[tbl\_guests] as g, [hotel].[tbl\_transactions] as t

WHERE g.IdGuest = t.IdGuest) as p, [hotel].[tbl\_reservations] as r

Where r.[IdReservation]=p.[IdReservation]

and LOWER(p.Name) + ' '+LOWER(p.LastName) LIKE LOWER('%'+@NameOrLastName+'%')

GROUP BY p.IdGuest, p.Name, p.LastName)

GO

* 1. Funkcje skalarne

exec sp\_helptext uf\_getCheckInDate

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

--- FUNCTION DEFINITION

--- uf\_getCheckInDate (@id\_reservation int)

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

-- wyszukiwanie daty zameldowania na podstawie numeru rezerwacji

--

-- parametry wejściowe:

-- @id\_reservation - numer rezerwacji

--

--parametry wyjściowe/zwracane wartości:

-- @CheckInDate - data zameldowania

--

/\*

use HotelDatabase

go

Select dbo.uf\_getCheckInDate(1)

--

--Wynik działania

--Wynikiem działania jest data zameldowania przypisana do pierwszej rezerwacji '2023-04-01'

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

\*/

use HotelDatabase

go

CREATE FUNCTION uf\_getCheckInDate(@id\_reservation int)

RETURNS date

BEGIN

DECLARE @CheckInDate date

SET @CheckInDate = (SELECT CheckInDate FROM [hotel].[tbl\_reservations] WHERE [IdReservation] = @id\_reservation)

RETURN @CheckInDate

END

GO

exec sp\_helptext uf\_getCheckOutDate

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

--- FUNCTION DEFINITION

--- uf\_getCheckOutDate (@id\_reservation int)

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

-- wyszukiwanie daty wymeldowania na podstawie numeru rezerwacji

--

-- parametry wejściowe:

-- @id\_reservation - numer rezerwacji

--

--parametry wyjściowe/zwracane wartości:

-- @CheckOutDate - data wymeldowania

--

/\*

use HotelDatabase

go

Select dbo.uf\_getCheckOutDate(1)

--

--Wynik działania

--Wynikiem działania jest data wymeldowania przypisana do pierwszej rezerwacji: '2023-04-05'

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

\*/

use HotelDatabase

go

CREATE OR ALTER FUNCTION uf\_getCheckOutDate(@id\_reservation int)

RETURNS date

BEGIN

DECLARE @CheckOutDate date

SET @CheckOutDate = (SELECT CheckOutDate FROM [hotel].[tbl\_reservations] WHERE [IdReservation] = @id\_reservation)

RETURN @CheckOutDate

END

GO

exec sp\_helptext uf\_getGuestId

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

--- FUNCTION DEFINITION

--- uf\_getGuestId (@id\_billing int)

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

-- wyszukiwanie gościa na podstawie numeru faktury

--

-- parametry wejściowe:

-- @id\_billing - numer faktury

--

--parametry wyjściowe/zwracane wartości:

-- @id - numer gościa

--

/\*

use HotelDatabase

go

Select LastName from hotel.tbl\_guests

where IdGuest = uf\_getGuestId (1)

--

--Wynik działania

--Wynikiem działania jest 'Doe'

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

\*/

use HotelDatabase

go

CREATE OR ALTER FUNCTION uf\_getGuestId(@id\_billing int)

RETURNS int

BEGIN

DECLARE @id int

SET @id = (SELECT [IdPersonData]

FROM [hotel].[tbl\_billing] as b

INNER JOIN [hotel].[tbl\_guests] as g

ON b.IdGuest = g.IdGuest

WHERE IdBilling=@id\_billing)

RETURN @id

END

GO

exec sp\_helptext uf\_getRoomPrice

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

--- FUNCTION DEFINITION

--- uf\_getRoomPrice(@id\_room int)

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

--wyszukuje cenę pokoju na podstawie jego numeru

--

--parametry wejściowe:

--@id\_room - numer pokoju

--

--parametry wyjściowe/zwracane wartości:

--money @price - cena wybranegp pokoju za jednodniowy pobyt

--

/\*

Select [dbo].[uf\_getRoomPrice](1);

--

--Wynik działania

--Cena pokoju pierwszego wynosi 100.00, ponieważ jest to pokój typu 'Single Room'

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

\*/

use HotelDatabase

go

CREATE OR ALTER FUNCTION uf\_getRoomPrice(@id\_room int)

RETURNS money

BEGIN

DECLARE @price money

SET @price = (SELECT [RoomPrice] FROM [room].[tbl\_room\_type] as rt

INNER JOIN [hotel].[tbl\_rooms] as r

ON rt.[IdRoomType] = r.[IdRoomType]

WHERE r.[IdRoom] = @id\_room)

RETURN @price

END

GO

exec sp\_helptext uf\_IsNIPNumberOK

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

--- FUNCTION DEFINITION

--- uf\_IsNIPNumberOK (@NIPNumber varchar(10))

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

--walidacja poprawności numeru NIP

--wywoływana w trakcie dodawania nowej faktury

--

--parametry wejściowe:

--@NIPNumber - numer NIP firmy

--

--parametry wyjściowe/zwracane wartości:

--wartość=1 - numer NIP jest poprawny

--wartość 0 - numer NIP niepoprawny

--

/\*

CREATE TABLE hotel.tbl\_billing(

IdBilling int NOT NULL IDENTITY(1,1) PRIMARY KEY,

IdGuest int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_guests(IdGuest) ON UPDATE CASCADE,

CompanyName varchar(30) NOT NULL,

NIPNumber varchar(10) NOT NULL CHECK (dbo.uf\_IsNIPNumberOK(NIPNumber) = 1),

);

GO

--

--Wynik działania

--Stworzona tabela pozwala tylko na wprowadzenie faktur z poprawnym numerem NIP

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

\*/

use HotelDatabase

go

CREATE OR ALTER FUNCTION dbo.uf\_IsNIPNumberOK(

@NIPNumber varchar(10)

)

RETURNS tinyint

AS

BEGIN

DECLARE @wagi AS TABLE(pozycja tinyint NOT NULL,

waga tinyint NOT NULL)

IF ISNUMERIC(@NIPNumber) = 0 RETURN 0

INSERT INTO @wagi(pozycja,waga)

VALUES (1,6),(2,5),(3,7),(4,2),(5,3),

(6,4),(7,5),(8,6),(9,7)

IF (SELECT SUM(CAST(SUBSTRING(@NIPNumber, pozycja, 1) AS tinyint) \* waga) % 10

FROM @wagi) <> SUBSTRING(@NIPNumber,10,1)

RETURN 0

RETURN 1

END

GO

exec sp\_helptext uf\_IsPeselOK

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

--- FUNCTION DEFINITION

--- up\_IsPeselOK (@pesel varchar(11))

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

--walidacja poprawności numeru pesel

--wywoływana w trakcie dodawania nowego pracownika

--

--parametry wejściowe:

--@pesel-numer pesel pracownika

--

--parametry wyjściowe/zwracane wartości:

--wartość=1 - pesel pracownika jest poprawny

--wartość 0 - pesel pracownika niepoprawny

--

/\*

--Przykład użycia

CREATE TABLE dbo.employees(

IdEmployee int NOT NULL IDENTITY(1,1) PRIMARY KEY,

IdPersonData int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_people\_data(IdPersonData) ON UPDATE CASCADE,

[Name] varchar(15) NOT NULL,

LastName varchar(30) NOT NULL,

IdEmployeePosition int NOT NULL FOREIGN KEY REFERENCES employee.tbl\_employee\_position(IdEmployeePosition) ON UPDATE CASCADE,

DateOfEmployment date NOT NULL CHECK (DateOfEmployment <= GETDATE()),

IdHotel int NOT NULL FOREIGN KEY REFERENCES hotel.tbl\_hotels(IdHotel) ON UPDATE CASCADE,

Pesel varchar(11) NOT NULL CHECK (dbo.uf\_IsPeselOK(Pesel) = 1)

)

GO

--

--Wynik działania

--Stworzona tabela pozwala tylko na wprowadzenie pracowników z poprawnym numerem pesel

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

\*/

use HotelDatabase

go

CREATE OR ALTER FUNCTION dbo.uf\_IsPeselOK(

@pesel varchar(11)

)

RETURNS tinyint

AS

BEGIN

DECLARE @wagi AS TABLE(pozycja tinyint NOT NULL,

waga tinyint NOT NULL)

IF ISNUMERIC(@pesel) = 0 RETURN 0

INSERT INTO @wagi(pozycja,waga)

VALUES (1,1),(2,3),(3,7),(4,9),(5,1),

(6,3),(7,7),(8,9),(9,1),(10,3),(11,1)

IF (SELECT SUM(CAST(SUBSTRING(@pesel, pozycja, 1) AS tinyint) \* waga) % 10

FROM @wagi) <> 0

RETURN 0

RETURN 1

END

GO

exec sp\_helptext uf\_IsRoomFree

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

--- FUNCTION DEFINITION

--- uf\_IsRoomFree (@id\_room int)

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

-- weryfikacja statusu pokoju o podanym numerze

--

-- parametry wejściowe:

-- @id\_room - numer pokoju

--

--parametry wyjściowe/zwracane wartości:

-- @Is\_free

-- 1 - pokój jest wolny/dostępny do rezerwacji

-- 0 - pokój zajęty/niedostępny

--

/\*

use HotelDatabase

go

Select uf\_IsRoomFree (1)

--

--Wynik działania

--Wynikiem działania jest

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

\*/

use HotelDatabase

go

CREATE OR ALTER FUNCTION uf\_IsRoomFree(@id\_room int)

RETURNS bit

BEGIN

DECLARE @is\_free bit, @CheckInDate date, @CheckOutDate date, @status varchar

SELECT @CheckInDate=r.[CheckInDate], @CheckOutDate=r.CheckOutDate, @status=s.StatusName

FROM [hotel].[tbl\_reservations] as r

INNER JOIN [reservation].[tbl\_reservation\_status] as s

ON r.[IdReservationStatus]=s.IdReservationStatus

WHERE r.[IdRoom]=@id\_room

IF (@status <> 'Cancelled' and @CheckOutDate<=GETDATE() and @CheckInDate>=GETDATE())

SET @is\_free = 0

ELSE

SET @is\_free=1

RETURN @is\_free

END

exec sp\_helptext uf\_LengthOfVisit

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

--- FUNCTION DEFINITION

--- uf\_LengthOfVisit (@id\_Reservation int)

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

-- obliczanie długości pobytu na podstawie dat zameldowania i wymeldowania dla podanego numeru rezerwacji

--

-- parametry wejściowe:

-- @id\_Reservation - numer rezerwacji

--

--parametry wyjściowe/zwracane wartości:

-- @days - liczba dni

--

/\*

use HotelDatabase

go

Select uf\_LengthOfVisit (1)

--

--Wynik działania

--Wynikiem działania jest 'Doe'

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

\*/

use HotelDatabase

go

CREATE OR ALTER FUNCTION uf\_LengthOfTheVisit(@id\_Reservation int)

Returns int

BEGIN

DECLARE @days int, @CheckInDate date, @CheckOutDate date

SELECT @CheckInDate = r.CheckInDate, @CheckOutDate = r.CheckOutDate

FROM [hotel].[tbl\_reservations] AS r

JOIN [hotel].[tbl\_transactions] AS t ON r.IdReservation = t.IdReservation

WHERE r.IdReservation = @id\_Reservation

SET @days = DATEDIFF(day,@CheckInDate,@CheckOutDate)

RETURN @days

END

GO

exec sp\_helptext uf\_setRoomStatusReserved

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

--- FUNCTION DEFINITION

--- uf\_setRoomStatusReserved (@id\_room int)

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

-- ustawia status pokoju na 'Reserved' jeżeli data dzisiejsza mieści się w ramach dokonanej rezerwacji

-- funkcja nie uwzględnia rezerwacji anulowanych (status rezerwacji 'Cancelled')

--

-- parametry wejściowe:

-- @id\_room - numer pokoju

--

--parametry wyjściowe/zwracane wartości:

-- @isRoomReserved

-- 1 - pokój zarezerwowany

-- 0 - pokój niezarezerwowany

/\*

use HotelDatabase

go

Select uf\_setRoomStatusReserved (1)

--

--Wynik działania

--Wynikiem działania jest 0, ponieważ dzisiejsza data nie mieści się w przedziale dat na którą została dokonana jedyna rezerwacja

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

\*/

use HotelDatabase

go

CREATE OR ALTER FUNCTION uf\_setRoomStatusReserved(@id\_room int)

RETURNS bit

BEGIN

DECLARE @isRoomReserved bit, @CheckInDate date, @CheckOutDate date, @status varchar

Select @CheckInDate = [CheckInDate], @CheckOutDate = [CheckOutDate]

from [hotel].[tbl\_reservations]

where [IdRoom]=@id\_room

select @status=[StatusName]

from [reservation].[tbl\_reservation\_status] as s

inner join [hotel].[tbl\_reservations] as r

on s.[IdReservationStatus]=r.[IdReservationStatus]

where r.[IdRoom]=@id\_room

If @CheckInDate<GETDATE() and @CheckOutDate>GETDATE() and @status<>'Cancelled'

set @isRoomReserved=1

set @isRoomReserved=0

RETURN @isRoomReserved

END

GO

exec sp\_helptext uf\_SumOfBilling

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

--- FUNCTION DEFINITION

--- uf\_SumOfBilling (@id\_transaction int)

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

-- obliczanie łącznej kwoty faktury na podstawie numeru transakcji

-- wykorzystuje funkcje uf\_LengthOfTheVisit i uf\_getRoomPrice do wyznaczenia liczby dni rezerwacji i ceny pokoju

--

-- parametry wejściowe:

-- @id\_transaction - numer transakcji

--

--parametry wyjściowe/zwracane wartości:

-- @SumOfBilling - łączna kwota faktury

--

/\*

use HotelDatabase

go

Select uf\_SumOfBilling (1)

--

--Wynik działania

--Wynikiem działania jest 400, ponieważ cena za dzień w pokoju 1 wynosi 100, a rezerwacji dokonano na 4 dni

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

\*/

use HotelDatabase

go

CREATE OR ALTER FUNCTION uf\_SumOfBilling(@id\_transaction int)

RETURNS money

BEGIN

DECLARE @SumOfBilling money = 0

SELECT @SumOfBilling +=[dbo].[uf\_LengthOfTheVisit](r.IdReservation) \* [dbo].[uf\_getRoomPrice](r.IdRoom)

FROM [hotel].[tbl\_transactions] AS t

JOIN [hotel].[tbl\_reservations] AS r ON t.IdReservation = r.IdReservation

WHERE t.IdTransaction = @id\_transaction

RETURN @SumOfBilling

END

# 16. Wykaz utworzonych wyzwalaczy













# 17. Skrypt tworzenia wyzwalaczy

use HotelDatabase

go

CREATE or ALTER TRIGGER trg\_ReservationDateBlock ON hotel.tbl\_reservations

INSTEAD OF UPDATE, INSERT

AS

IF (select CheckInDate from hotel.tbl\_reservations)>(select CheckOutDate from hotel.tbl\_reservations)

RETURN;

IF (select CheckInDate from hotel.tbl\_reservations)<GETDATE()

IF (select CheckOutDate from hotel.tbl\_reservations)<GETDATE()

RETURN;

GO

use HotelDatabase

go

CREATE TRIGGER trg\_EmployeePositionEditBlock ON employee.tbl\_employee\_position

INSTEAD OF INSERT, UPDATE, DELETE

AS

RETURN;

GO

use HotelDatabase

go

CREATE TRIGGER trg\_RoomStatusEditBlock ON room.tbl\_status

INSTEAD OF INSERT, UPDATE, DELETE

AS

RETURN;

GO

use HotelDatabase

go

CREATE TRIGGER trg\_PaymentTypeEditBlock ON payment.tbl\_payment\_type

INSTEAD OF INSERT, UPDATE, DELETE

AS

RETURN;

GO

use HotelDatabase

go

CREATE TRIGGER trg\_RoomTypeEditBlock ON room.tbl\_room\_type

INSTEAD OF INSERT, UPDATE, DELETE

AS

RETURN;

GO

use HotelDatabase

go

CREATE TRIGGER trg\_ReservationStatusEditBlock ON reservation.tbl\_reservation\_status

INSTEAD OF INSERT, UPDATE, DELETE

AS

RETURN;

GO

# 18. Wykaz obiektów wartości domyślnej i reguł

1. Wartości domyślne

Obraz zawierający tekst

Opis wygenerowany automatycznie

1. Reguły

Obraz zawierający stół

Opis wygenerowany automatycznie

# 19. Skrypt tworzenia obiektów wartości domyślnej i reguł

1. Wartości domyślne

CREATE DEFAULT df\_EmploymentDate AS GETDATE()

EXEC sp\_bindefault df\_EmploymentDate, 'hotel.tbl\_employees.DateOfEmployment'

CREATE DEFAULT df\_TransactionDate AS GETDATE()

EXEC sp\_bindefault df\_TransactionDate, 'hotel.tbl\_transactions.TransactionDate'

CREATE DEFAULT df\_MobilePhone AS '48000000000'

EXEC sp\_bindefault df\_MobilePhone, 'hotel.tbl\_people\_data.MobilelinePhoneNumber'

EXEC sp\_bindefault df\_MobilePhone, 'hotel.tbl\_hotels.MobilelinePhoneNumber'

CREATE DEFAULT df\_email AS 'default.mail@default.com'

EXEC sp\_bindefault df\_email, 'hotel.tbl\_people\_data.Email'

EXEC sp\_bindefault df\_MobilePhone, 'hotel.tbl\_hotels.Email'

CREATE DEFAULT df\_country AS 'Poland'

EXEC sp\_bindefault df\_email, 'hotel.tbl\_people\_data.Country'

CREATE DEFAULT df\_PaymentDate AS GETDATE()

EXEC sp\_bindefault df\_PaymentDate, 'hotel.tbl\_payments.PaymentDate'

CREATE DEFAULT df\_LandlinePhoneNumber AS '(22) 123 45 67'

EXEC sp\_bindefault df\_LandlinePhoneNumber, 'hotel.tbl\_hotels.LandlinePhoneNumber'

1. Reguły

CREATE RULE rl\_ZipCode as @ZipCode LIKE '[0-9][0-9]-[0-9][0-9][0-9]';

EXEC sp\_bindrule rl\_ZipCode, 'hotel.tbl\_people\_data.ZipCode'

EXEC sp\_bindrule rl\_ZipCode, 'hotel.tbl\_hotels.ZipCode'

CREATE RULE rl\_Email as @email LIKE '%[A-Z0-9][@][A-Z0-9]%[.][A-Z0-9]%'

OR @email LIKE NULL;

EXEC sp\_bindrule rl\_Email, 'hotel.tbl\_people\_data.Email'

EXEC sp\_bindrule rl\_Email, 'hotel.tbl\_hotels.Email'

CREATE RULE rl\_DateOfBirth AS (@DateOfBirth < CAST(GETDATE() AS DATE));

EXEC sp\_bindrule rl\_DateOfBirth, 'hotel.tbl\_people\_data.DateOfBirth'

CREATE RULE rl\_EmploymentDate AS (@EmploymentDate <= CAST(GETDATE() AS DATE));

EXEC sp\_bindrule rl\_EmploymentDate, 'hotel.tbl\_employees.DateOfEmployment'

CREATE RULE rl\_TransactionDate AS (@TransactionDate <= CAST(GETDATE() AS DATE));

EXEC sp\_bindrule rl\_TransactionDate,'hotel.tbl\_transactions.TransactionDate'

CREATE RULE rl\_PaymentDate AS (@PaymentDate <= CAST(GETDATE() AS DATE));

EXEC sp\_bindrule rl\_PaymentDate, 'hotel.tbl\_payments.PaymentDate'

CREATE RULE rl\_MobilePhone as @MobilePhone LIKE '48[5-8][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'

or @MobilePhone LIKE NULL;

EXEC sp\_bindrule rl\_MobilePhone, 'hotel.tbl\_people\_data.MobilelinePhoneNumber'

EXEC sp\_bindrule rl\_MobilePhone, 'hotel.tbl\_hotels.MobilelinePhoneNumber'

CREATE RULE rl\_IDNumber as @IDNumber LIKE '[A-Z][A-Z][A-Z][0-9][0-9][0-9][0-9][0-9][0-9]';

EXEC sp\_bindrule rl\_IDNumber, 'hotel.tbl\_guests.IDNumber'

CREATE RULE rl\_PassportNumber as @PassportNumber LIKE '[A-Z][A-Z][0-9][0-9][0-9][0-9][0-9][0-9][0-9]';

EXEC sp\_bindrule rl\_PassportNumber, 'hotel.tbl\_guests.PassportNumber'

CREATE RULE rl\_LandlinePhoneNumber as @LandlinePhoneNumber LIKE '([1-9][0-9]) [0-9][0-9][0-9] [0-9][0-9][0-9]'

OR @LandlinePhoneNumber LIKE NULL;

EXEC sp\_bindrule rl\_LandlinePhoneNumber, 'hotel.tbl\_hotels.LandlinePhoneNumber'

# 20. Wykaz typów danych tworzonych przez użytkownika

# 21. Skrypt tworzenia typów danych definiowanych przez użytkownika

# 22. Schemat zaprojektowanego systemu przedstawiający punkty wejścia i wyjścia

# 23. Wnioski i podsumowanie

Baza danych sieci hoteli "Hotel D&B&N" została zaprojektowana zgodnie z zasadami projektowania baz danych, które zapewniają efektywne i niezawodne funkcjonowanie systemu. Baza ta spełnia wszystkie wymagania i funkcjonalności, które zostały wymienione wcześniej. Dodatkowo, wypełniono ją przykładowymi danymi, co pozwala na przetestowanie jej działania i weryfikację poprawności wykonanych implementacji. Dzięki temu użytkownicy systemu mogą mieć pewność, że dane, które są w niej przechowywane, są dokładne i aktualne, co wpływa na jakość świadczonych usług i zadowolenie klientów.

Baza danych sieci hoteli "Hotel D&B&N" została wyposażona w wiele widoków, które usprawniają proces analizy danych klientów, rezerwacji oraz pokoi. Dzięki nim użytkownicy mogą łatwo przeglądać i monitorować dane z różnych perspektyw, co przyspiesza podejmowanie decyzji biznesowych. Ponadto, baza ta umożliwia dodawanie i modyfikowanie danych związanych z klientami, rezerwacjami, pracownikami i placówkami hotelu, co zwiększa elastyczność i dostosowanie systemu do potrzeb użytkowników. Baza ta posiada także wiele funkcjonalności ułatwiających pracę z danymi, w tym zarządzanie fakturami i płatnościami, co usprawnia proces rozliczeń i ułatwia obsługę klientów.

Projekt sieci hoteli został stworzony z myślą o obsłudze wszystkich operacji związanych z prowadzeniem hoteli, tak aby był w pełni funkcjonalny i dostosowany do potrzeb użytkowników. Obejmuje on kompleksowe rozwiązania dla zarządzania rezerwacjami, obsługą klientów, finansami, pracownikami i innymi aspektami związanymi z funkcjonowaniem sieci hoteli. Dzięki temu użytkownicy systemu mogą łatwo zarządzać i monitorować procesy biznesowe, co przyczynia się do poprawy efektywności i rentowności biznesu.

# 24. Wykaz/spis ilustracji

# 25. Wykaz/spis tabel