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 2](#_Toc137596432)

[2. Opis wymagań funkcjonalnych i niefunkcjonalnych 3](#_Toc137596433)

[3. Model fizyczny lub schemat bazy danych w postaci graficznej 4](#_Toc137596434)

[4. Skrypt tworzenia bazy danych 5](#_Toc137596435)

[5. Wykaz utworzonych tabel fizycznych 5](#_Toc137596436)

[6. Skrypt tworzenia tabel i referencji między nimi 6](#_Toc137596437)

[7. Wykaz zaimplementowanych indeksów 9](#_Toc137596438)

[8. Skrypt tworzenia indeksów 10](#_Toc137596439)

[9. Skrypt ładujący przykładowe dane do tabel 12](#_Toc137596440)

[10. Wykaz utworzonych widoków 15](#_Toc137596441)

[11. Skrypt tworzenia widoków 15](#_Toc137596442)

[12. Wykaz utworzonych procedur składowanych 23](#_Toc137596443)

[13. Skrypt tworzenia procedur składowanych 23](#_Toc137596444)

[14. Wykaz utworzonych funkcji użytkownika 39](#_Toc137596445)

[15. Skrypt tworzenia funkcji definiowanych przez użytkownika 39](#_Toc137596446)

[16. Wykaz utworzonych wyzwalaczy 49](#_Toc137596447)

[17. Skrypt tworzenia wyzwalaczy 49](#_Toc137596448)

[18. Wykaz obiektów wartości domyślnej i reguł 55](#_Toc137596449)

[19. Skrypt tworzenia obiektów wartości domyślnej i reguł 55](#_Toc137596450)

[20. Schemat zaprojektowanego systemu przedstawiający punkty wejścia i wyjścia 57](#_Toc137596451)

[21. Wnioski i podsumowanie 67](#_Toc137596452)

[Spis ilustracji 68](#_Toc137596453)

# 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

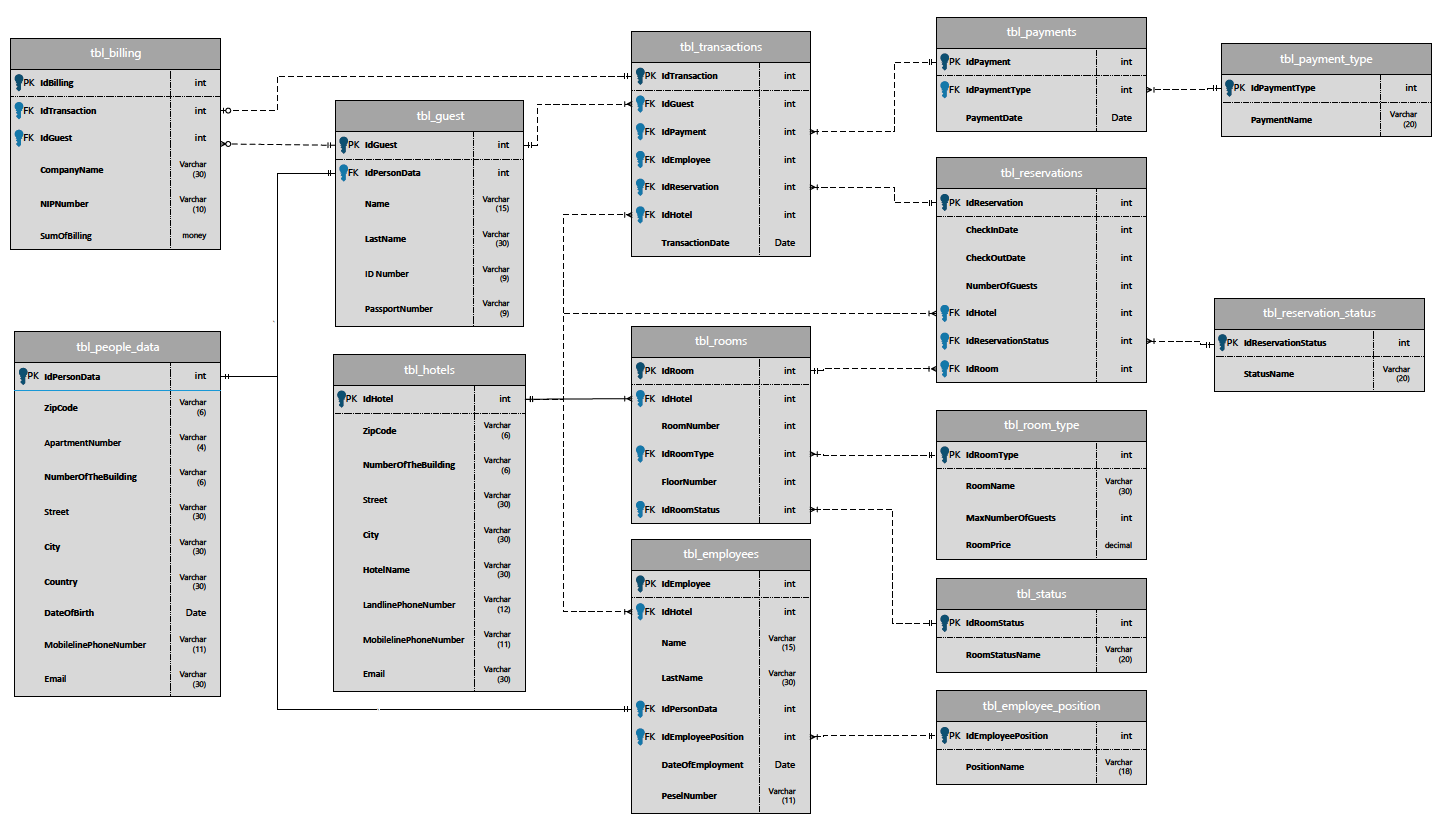
# Model fizyczny lub schemat bazy danych w postaci graficznej

Rys. 1. Model logiczny bazy danych

Obraz zawierający diagram

Opis wygenerowany automatycznie

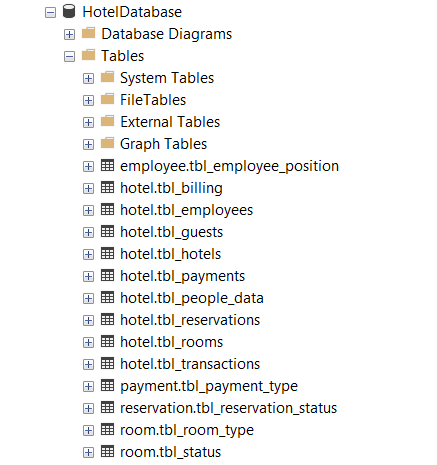
Rys. 2. Model fizyczny bazy danych



# 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,

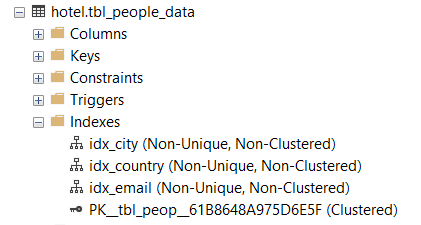
CompanyName varchar(30) NOT NULL,

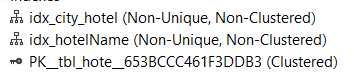
NIPNumber varchar(10) NOT NULL CHECK (dbo.uf\_IsNIPNumberOK(NIPNumber) = 1),

SumOfBilling money Null);

GO

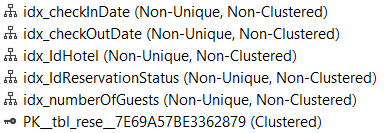
# Wykaz zaimplementowanych indeksów

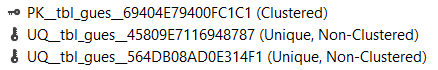


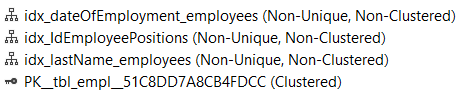


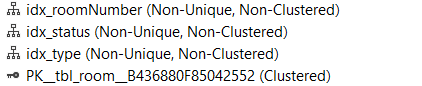
Obraz zawierający tekst

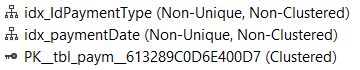
Opis wygenerowany automatycznie

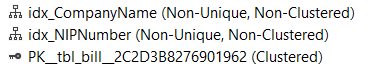












# Skrypt tworzenia indeksów

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'),

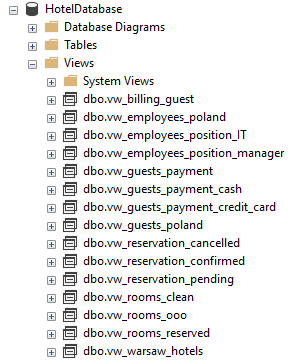
(4, 4, 'LMN Enterprises', '8642097532'),

(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

--- VIEW DEFINITION

--- vw\_warsaw\_hotels

-- Wyświetlanie informacji o hotelach w Warszawie.

-- Tworzony widok, który zawiera nazwy hoteli, miasto, ulicę, numer budynku oraz kod pocztowy.

-- Kolumny:

-- - HotelName: Nazwa hotelu.

-- - City: Miasto, w którym znajduje się hotel.

-- - Street: Nazwa ulicy, na której znajduje się hotel.

-- - NumberOfTheBuilding: Numer budynku hotelowego.

-- - ZipCode: Kod pocztowy hotelu.

-- Przykład użycia:

-- SELECT \* FROM vw\_warsaw\_hotels;

-- Wynik działania:

-- Zwracane są informacje o hotelach znajdujących się w Warszawie.

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

--- VIEW DEFINITION

--- vw\_employees\_poland

-- Wyświetlanie informacji o pracownikach z Polski.

-- Tworzony widok, który zawiera imię, nazwisko, numer PESEL, data urodzenia, adres e-mail oraz kraj zatrudnienia.

-- Kolumny:

-- - Name: Imię pracownika.

-- - LastName: Nazwisko pracownika.

-- - PeselNumber: Numer PESEL pracownika.

-- - DateOfBirth: Data urodzenia pracownika.

-- - Email: Adres e-mail pracownika.

-- - Country: Kraj zatrudnienia pracownika.

-- Przykład użycia:

-- SELECT \* FROM vw\_employees\_poland;

-- Wynik działania:

-- Zwracane są informacje o pracownikach zatrudnionych w Polsce.

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

--- VIEW DEFINITION

--- vw\_guests\_poland

-- Wyświetlanie informacji o gościach z Polski.

-- Tworzony widok, który zawiera imię, nazwisko, numer paszportu, data urodzenia, adres e-mail oraz kraj pochodzenia.

-- Kolumny:

-- - Name: Imię gościa.

-- - LastName: Nazwisko gościa.

-- - PassportNumber: Numer paszportu gościa.

-- - DateOfBirth: Data urodzenia gościa.

-- - Email: Adres e-mail gościa.

-- - Country: Kraj pochodzenia gościa.

-- Przykład użycia:

-- SELECT \* FROM vw\_guests\_poland;

-- Wynik działania:

-- Zwracane są informacje o gościach pochodzących z Polski.

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

--- VIEW DEFINITION

--- vw\_employees\_position\_IT

-- Wyświetlanie informacji o pracownikach zajmujących stanowisko IT Specialist.

-- Tworzony widok, który zawiera imię, nazwisko oraz nazwę stanowiska.

-- Kolumny:

-- - Name: Imię pracownika.

-- - LastName: Nazwisko pracownika.

-- - PositionName: Nazwa stanowiska.

-- Przykład użycia:

-- SELECT \* FROM vw\_employees\_position\_IT;

-- Wynik działania:

-- Zwracane są informacje o pracownikach zajmujących stanowisko IT Specialist.

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

--- VIEW DEFINITION

--- vw\_employees\_position\_manager

-- Wyświetlanie informacji o pracownikach zajmujących stanowisko Manager.

-- Tworzony widok, który zawiera imię, nazwisko oraz nazwę stanowiska.

-- Kolumny:

-- - Name: Imię pracownika.

-- - LastName: Nazwisko pracownika.

-- - PositionName: Nazwa stanowiska.

-- Przykład użycia:

-- SELECT \* FROM vw\_employees\_position\_manager;

-- Wynik działania:

-- Zwracane są informacje o pracownikach zajmujących stanowisko 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

--- VIEW DEFINITION

--- vw\_rooms\_reserved

-- Wyświetlanie informacji o zarezerwowanych pokojach.

-- Tworzony widok, który zawiera nazwę pokoju, cenę, numer pokoju, numer piętra, maksymalną liczbę gości oraz status pokoju.

-- Kolumny:

-- - RoomName: Nazwa pokoju.

-- - RoomPrice: Cena pokoju.

-- - RoomNumber: Numer pokoju.

-- - FloorNumber: Numer piętra, na którym znajduje się pokój.

-- - MaxNumberOfGuests: Maksymalna liczba gości, jaką pokój może pomieścić.

-- - RoomStatusName: Status pokoju.

-- Przykład użycia:

-- SELECT \* FROM vw\_rooms\_reserved;

-- Wynik działania:

-- Zwracane są informacje o zarezerwowanych pokojach.

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

--- VIEW DEFINITION

--- vw\_rooms\_clean

-- Wyświetlanie informacji o czystych pokojach.

-- Tworzony widok, który zawiera nazwę pokoju, cenę, numer pokoju, numer piętra, maksymalną liczbę gości oraz status pokoju.

-- Kolumny:

-- - RoomName: Nazwa pokoju.

-- - RoomPrice: Cena pokoju.

-- - RoomNumber: Numer pokoju.

-- - FloorNumber: Numer piętra, na którym znajduje się pokój.

-- - MaxNumberOfGuests: Maksymalna liczba gości, jaką pokój może pomieścić.

-- - RoomStatusName: Status pokoju.

-- Przykład użycia:

-- SELECT \* FROM vw\_rooms\_clean;

-- Wynik działania:

-- Zwracane są informacje o czystych pokojach.

CREATE VIEW vw\_rooms\_clean

AS

SELECT rt.RoomName,

rt.RoomPrice,

r.RoomNumber,

r.FloorNumber,

rt.MaxNumberOfGuests,

rs.RoomStatusName

[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

--- VIEW DEFINITION

--- vw\_rooms\_ooo

-- Wyświetlanie informacji o pokojach "out of order" (poza użytkiem).

-- Tworzony widok, który zawiera nazwę pokoju, cenę, numer pokoju, numer piętra, maksymalną liczbę gości oraz status pokoju.

-- Kolumny:

-- - RoomName: Nazwa pokoju.

-- - RoomPrice: Cena pokoju.

-- - RoomNumber: Numer pokoju.

-- - FloorNumber: Numer piętra, na którym znajduje się pokój.

-- - MaxNumberOfGuests: Maksymalna liczba gości, jaką pokój może pomieścić.

-- - RoomStatusName: Status pokoju.

-- Przykład użycia:

-- SELECT \* FROM vw\_rooms\_ooo;

-- Wynik działania:

-- Zwracane są informacje o pokojach "out of order" (poza użytkiem).

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

--- VIEW DEFINITION

--- vw\_reservation\_pending

-- Wyświetlanie oczekujących rezerwacji.

-- Tworzony widok, który zawiera nazwę hotelu, numer pokoju oraz status rezerwacji.

-- Kolumny:

-- - HotelName: Nazwa hotelu.

-- - RoomNumber: Numer pokoju.

-- - StatusName: Status rezerwacji.

-- Przykład użycia:

-- SELECT \* FROM vw\_reservation\_pending;

-- Wynik działania:

-- Zwracane są informacje o oczekujących rezerwacjach.

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

--- VIEW DEFINITION

--- vw\_reservation\_confirmed

-- Wyświetlanie potwierdzonych rezerwacji.

-- Tworzony widok, który zawiera nazwę hotelu, numer pokoju oraz status rezerwacji.

-- Kolumny:

-- - HotelName: Nazwa hotelu.

-- - RoomNumber: Numer pokoju.

-- - StatusName: Status rezerwacji.

-- Przykład użycia:

-- SELECT \* FROM vw\_reservation\_confirmed;

-- Wynik działania:

-- Zwracane są informacje o potwierdzonych rezerwacjach.

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

--- VIEW DEFINITION

--- vw\_reservation\_cancelled

-- Wyświetlanie anulowanych rezerwacji.

-- Tworzony widok, który zawiera nazwę hotelu, numer pokoju oraz status rezerwacji.

-- Kolumny:

-- - HotelName: Nazwa hotelu.

-- - RoomNumber: Numer pokoju.

-- - StatusName: Status rezerwacji.

-- Przykład użycia:

-- SELECT \* FROM vw\_reservation\_cancelled;

-- Wynik działania:

-- Zwracane są informacje o anulowanych rezerwacjach.

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

--- VIEW DEFINITION

--- vw\_billing\_guest

-- Wyświetlanie danych dotyczących rozliczeń dla gości.

-- Tworzony widok, który zawiera informacje o gościach oraz dane rozliczeniowe.

-- Kolumny:

-- - Name: Imię gościa.

-- - LastName: Nazwisko gościa.

-- - PassportNumber: Numer paszportu gościa.

-- - CompanyName: Nazwa firmy rozliczeniowej.

-- - NIPNumber: Numer NIP firmy rozliczeniowej.

-- Przykład użycia:

-- SELECT \* FROM vw\_billing\_guest;

-- Wynik działania:

-- Zwracane są dane dotyczące rozliczeń dla gości.

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

--- VIEW DEFINITION

--- vw\_guests\_payment

-- Wyświetlanie danych dotyczących płatności gości.

-- Tworzony widok, który zawiera informacje o gościach oraz danych dotyczących płatności.

-- Kolumny:

-- - Name: Imię gościa.

-- - LastName: Nazwisko gościa.

-- - TransactionDate: Data transakcji.

-- - PaymentName: Nazwa metody płatności.

-- Przykład użycia:

-- SELECT \* FROM vw\_guests\_payment;

-- Wynik działania:

-- Zwracane są informacje dotyczące płatności gości.

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

--- VIEW DEFINITION

--- vw\_guests\_payment\_cash

-- Wyświetlanie danych dotyczących płatności gości dokonanych gotówką.

-- Tworzony widok, który zawiera informacje o gościach oraz danych dotyczących płatności wykonanych gotówką.

-- Kolumny:

-- - Name: Imię gościa.

-- - LastName: Nazwisko gościa.

-- - TransactionDate: Data transakcji.

-- - PaymentName: Nazwa metody płatności (gotówka).

-- Przykład użycia:

-- SELECT \* FROM vw\_guests\_payment\_cash;

-- Wynik działania:

-- Zwracane są informacje dotyczące płatności gości dokonanych gotówką.

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

--- VIEW DEFINITION

--- vw\_guests\_payment\_credit\_card

-- Wyświetlanie danych dotyczących płatności gości dokonanych kartą kredytową.

-- Tworzony widok, który zawiera informacje o gościach oraz danych dotyczących płatności wykonanych kartą kredytową.

-- Kolumny:

-- - Name: Imię gościa.

-- - LastName: Nazwisko gościa.

-- - TransactionDate: Data transakcji.

-- - PaymentName: Nazwa metody płatności (karta kredytowa).

-- Przykład użycia:

-- SELECT \* FROM vw\_guests\_payment\_credit\_card;

-- Wynik działania:

-- Zwracane są informacje dotyczące płatności gości dokonanych kartą kredytową.

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

# Wykaz utworzonych procedur składowanych

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

# Skrypt tworzenia procedur składowanych

----Procedura up\_AddBilling została utworzona w celu dodania nowych rachunków

--parametry wejściowe:

--@TransactionId INT,

--@GuestId INT,

--@CompanyName VARCHAR(30),

--@NIPNumber VARCHAR(10),

--@SumOfBilling MONEY

CREATE PROCEDURE up\_AddBilling

(

@TransactionId INT,

@GuestId INT,

@CompanyName VARCHAR(30),

@NIPNumber VARCHAR(10),

@SumOfBilling MONEY

)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

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

VALUES (@TransactionId, @GuestId, @CompanyName, @NIPNumber, @SumOfBilling);

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

SELECT @ErrorMessage = ERROR\_MESSAGE();

RAISERROR('Wystąpił błąd podczas dodawania faktury: %s', 16, 1, @ErrorMessage);

END CATCH;

END

--Przykład użycia

DECLARE @TransactionId INT;

DECLARE @GuestId INT;

DECLARE @CompanyName VARCHAR(30);

DECLARE @NIPNumber VARCHAR(10);

DECLARE @SumOfBilling MONEY;

--Procedura up\_AddEmployee została utworzona lub zmodyfikowana w celu dodania pracownika do bazy danych.--

--parametry wejściowe:

--@FirstName (varchar(15)): Przechowuje imię pracownika.

--@LastName (varchar(30)): Przechowuje nazwisko pracownika.

--@EmployeePosition (varchar(18)): Przechowuje stanowisko pracownika.

--@DateOfEmployment (date): Przechowuje datę zatrudnienia pracownika.

--@HotelId (int): Przechowuje identyfikator hotelu, w którym pracownik jest zatrudniony.

--@ZipCode (varchar(6)): Przechowuje kod pocztowy miejsca zamieszkania pracownika.

--@ApartmentNumber (varchar(4)): Przechowuje numer mieszkania pracownika.

--@NumberOfTheBuilding (varchar(6)): Przechowuje numer budynku pracownika.

--@Street (varchar(30)): Przechowuje nazwę ulicy, na której mieszka pracownik.

--@City (varchar(30)): Przechowuje nazwę miasta, w którym mieszka pracownik.

--@Country (varchar(30)): Przechowuje nazwę kraju, w którym mieszka pracownik.

--@DateOfBirth (date): Przechowuje datę urodzenia pracownika.

--@MobilePhoneNumber (varchar(11)): Przechowuje numer telefonu komórkowego pracownika.

--@Email (varchar(30)): Przechowuje adres e-mail pracownika.

--@PeselNumber (varchar(11)): Przechowuje numer PESEL pracownika.--

--

CREATE OR ALTER PROCEDURE up\_AddEmployee

@FirstName varchar(15),

@LastName varchar(30),

@EmployeePosition varchar(18),

@DateOfEmployment date,

@HotelId int,

@ZipCode varchar(6),

@ApartmentNumber varchar(4),

@NumberOfTheBuilding varchar(6),

@Street varchar(30),

@City varchar(30),

@Country varchar(30),

@DateOfBirth date,

@MobilePhoneNumber varchar(11),

@Email varchar(30),

@PeselNumber varchar(11)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

BEGIN TRANSACTION;

DECLARE @PersonDataId int;

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

VALUES (@ZipCode, @ApartmentNumber, @NumberOfTheBuilding, @Street, @City, @Country, @DateOfBirth, @MobilePhoneNumber, @Email);

SET @PersonDataId = SCOPE\_IDENTITY();

DECLARE @EmployeePositionId int;

SELECT @EmployeePositionId = IdEmployeePosition FROM employee.tbl\_employee\_position WHERE PositionName = @EmployeePosition;

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

VALUES (@PersonDataId, @FirstName, @LastName, @EmployeePositionId, @DateOfEmployment, @HotelId, @PeselNumber);

COMMIT;

END TRY

BEGIN CATCH

IF @@TRANCOUNT > 0

ROLLBACK;

DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();

DECLARE @ErrorSeverity INT = ERROR\_SEVERITY();

DECLARE @ErrorState INT = ERROR\_STATE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH;

END;

--Przykład użycia

EXECUTE up\_AddEmployee

@FirstName = 'John', @LastName = 'Doe',

@EmployeePosition = 'Receptionist', @DateOfEmployment = '1976-11-12',

@HotelId = 1, @ZipCode = '12-345',

@ApartmentNumber = '12', @NumberOfTheBuilding = '45',

@Street = 'Black Street', @City = 'Lodz',

@Country = 'Poland', @DateOfBirth = '1985-01-01',

@MobilePhoneNumber = '48777777777',

@Email = 'example@example.com',

@PeselNumber = '76111283795';

--Procedura up\_AddGuest została utworzona lub zmodyfikowana w celu dodania gościa do bazy danych.

--parametry wejściowe:

--@Name (varchar(15)): Przechowuje imię gościa.

--@LastName (varchar(30)): Przechowuje nazwisko gościa.

--@IDNumber (varchar(9)): Przechowuje numer dowodu tożsamości gościa.

--@PassportNumber (varchar(9)): Przechowuje numer paszportu gościa.

--@ZipCode (varchar(6)): Przechowuje kod pocztowy miejsca zamieszkania gościa.

--@ApartmentNumber (varchar(4)): Przechowuje numer mieszkania gościa.

--@NumberOfTheBuilding (varchar(6)): Przechowuje numer budynku gościa.

--@Street (varchar(30)): Przechowuje nazwę ulicy, na której mieszka gość.

--@City (varchar(30)): Przechowuje nazwę miasta, w którym mieszka gość.

--@Country (varchar(30)): Przechowuje nazwę kraju, w którym mieszka gość.

--@DateOfBirth (date): Przechowuje datę urodzenia gościa.

--@MobilelinePhoneNumber (varchar(11)): Przechowuje numer telefonu komórkowego gościa.

--@Email (varchar(30)): Przechowuje adres e-mail gościa.--

USE HotelDatabase

GO

CREATE OR ALTER PROCEDURE up\_AddGuest

(

@Name varchar(15),

@LastName varchar(30),

@IDNumber varchar(9),

@PassportNumber varchar(9),

@ZipCode varchar(6),

@ApartmentNumber varchar(4),

@NumberOfTheBuilding varchar(6),

@Street varchar(30),

@City varchar(30),

@Country varchar(30),

@DateOfBirth date,

@MobilelinePhoneNumber varchar(11),

@Email varchar(30)

)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

DECLARE @IdPersonData int;

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

VALUES (@ZipCode, @ApartmentNumber, @NumberOfTheBuilding, @Street, @City, @Country, @DateOfBirth, @MobilelinePhoneNumber, @Email);

SET @IdPersonData = SCOPE\_IDENTITY();

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

VALUES (@IdPersonData, @Name, @LastName, @IDNumber, @PassportNumber);

SELECT 'Guest added successfully.' AS Result;

END TRY

BEGIN CATCH

SELECT ERROR\_MESSAGE() AS ErrorMessage;

END CATCH;

END

GO

----Procedura up\_AddHotel została utworzona lub zmodyfikowana w celu dodania nowego hotelu do bazy danych.

--parametry wejściowe:

--@HotelName (VARCHAR(30)): Przechowuje nazwę hotelu.

--@CityName (VARCHAR(50)): Przechowuje nazwę miasta, w którym znajduje się hotel.

--@StreetAddress (VARCHAR(100)): Przechowuje adres ulicy hotelu.

--@PhoneNumber (VARCHAR(20)): Przechowuje numer telefonu hotelowego.

--@ZipCode (VARCHAR(6)): Przechowuje kod pocztowy hotelu.

--@NumberOfTheBuilding (VARCHAR(6)): Przechowuje numer budynku hotelowego.

CREATE OR ALTER PROCEDURE up\_AddHotel

@HotelName VARCHAR(30),

@CityName VARCHAR(50),

@StreetAddress VARCHAR(100),

@PhoneNumber VARCHAR(20),

@ZipCode VARCHAR(6),

@NumberOfTheBuilding varchar(6)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

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

VALUES (@HotelName, @CityName, @StreetAddress, @PhoneNumber, @ZipCode, @NumberOfTheBuilding);

SELECT 'Hotel added successfully.' AS Result;

END TRY

BEGIN CATCH

PRINT ERROR\_MESSAGE();

END CATCH;

END;

--Przykład użycia

EXEC up\_AddHotel

@HotelName = 'Hotel ABC',

@CityName = 'City XYZ',

@StreetAddress = '123 Street',

@PhoneNumber = '123-456-7890',

@ZipCode = '12345',

@NumberOfTheBuilding = 'A1';

----Procedura up\_addReservation została utworzona w celu dodania nowych rezerwacji

--parametry wejściowe:

--@IdRoom INT,

--@CheckInDate DATE,

--@CheckOutDate DATE,

--@NumberOfGuests VARCHAR(3),

--@IdReservationStatus INT,

--@IdHotel INT

USE HotelDatabase;

GO

CREATE OR ALTER PROCEDURE up\_addReservation

(

@IdRoom INT,

@CheckInDate DATE,

@CheckOutDate DATE,

@NumberOfGuests VARCHAR(3),

@IdReservationStatus INT,

@IdHotel INT

)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

-- Sprawdź czy pokój jest dostępny w wybranym okresie

IF EXISTS (

SELECT 1

FROM hotel.tbl\_reservations

WHERE IdRoom = @IdRoom

AND (

(CheckInDate >= @CheckInDate AND CheckInDate < @CheckOutDate)

OR (CheckOutDate > @CheckInDate AND CheckOutDate <= @CheckOutDate)

OR (CheckInDate < @CheckInDate AND CheckOutDate > @CheckOutDate)

)

)

BEGIN

RAISERROR('Ten pokój jest zarezerwowany.', 16, 1);

RETURN;

END;

-- Dodaj nową rezerwację i zwróć pełne informacje o wstawionym wierszu

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

OUTPUT inserted.\*

VALUES (@IdRoom, @CheckInDate, @CheckOutDate, @NumberOfGuests, @IdReservationStatus, @IdHotel);

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

SELECT @ErrorMessage = ERROR\_MESSAGE();

RAISERROR('Wystąpił błąd podczas dodawania rezerwacji: %s', 16, 1, @ErrorMessage);

END CATCH;

END;

-----Przykład użycia

DECLARE @NewReservationId INT;

EXEC hotel.up\_addReservation

@IdRoom = 4,

@CheckInDate = '2023-06-15',

@CheckOutDate = '2023-06-20',

@NumberOfGuests = '2',

@IdReservationStatus = 1,

@IdHotel = 1;

SELECT \* FROM hotel.tbl\_reservations WHERE IdReservation = @NewReservationId;

--Wynik

Wyświetla dane dodane jako nowa rezerwacja

----Procedura up\_AddTransactionWithPayment została utworzona w celu dodania nowych transkacji

--parametry wejściowe:

--@ReservationId INT,

--@EmployeeId INT,

--@GuestId INT,

--@HotelId INT,

--@TransactionDate DATE,

--@PaymentTypeId INT,

--@PaymentDate DATE

CREATE OR ALTER PROCEDURE up\_AddTransactionWithPayment

(

@ReservationId INT,

@EmployeeId INT,

@GuestId INT,

@HotelId INT,

@TransactionDate DATE,

@PaymentTypeId INT,

@PaymentDate DATE

)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

DECLARE @PaymentId INT;

-- Dodaj nową płatność

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

VALUES (@PaymentTypeId, @PaymentDate);

-- Pobierz identyfikator dodanej płatności

SET @PaymentId = SCOPE\_IDENTITY();

-- Dodaj nową transakcję z powiązaną płatnością

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

VALUES (@ReservationId, @PaymentId, @EmployeeId, @GuestId, @HotelId, @TransactionDate);

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

SELECT @ErrorMessage = ERROR\_MESSAGE();

RAISERROR('Wystąpił błąd podczas dodawania transakcji: %s', 16, 1, @ErrorMessage);

END CATCH;

END;

----Przykład użycia

DECLARE @ReservationId INT;

DECLARE @EmployeeId INT;

DECLARE @GuestId INT;

DECLARE @HotelId INT;

DECLARE @TransactionDate DATE;

DECLARE @PaymentTypeId INT;

DECLARE @PaymentDate DATE;

-- Ustaw odpowiednie wartości dla parametrów

SET @ReservationId = 13;

SET @EmployeeId = 1;

SET @GuestId = 1;

SET @HotelId = 1;

SET @TransactionDate = GETDATE();

SET @PaymentTypeId = 1; -- Identyfikator typu płatności

SET @PaymentDate = GETDATE();

-- Wywołaj procedurę

EXEC up\_AddTransactionWithPayment @ReservationId, @EmployeeId, @GuestId, @HotelId, @TransactionDate, @PaymentTypeId, @PaymentDate;

----Wynik

Commands completed successfully.

----Procedura up\_ChangeRoomStatus została utworzona lub zmodyfikowana w celu zmiany exec sp\_helptext up\_SetTriggerOFF

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

--- PROCEDURE DEFINITION

--- up\_SetTriggerOFF (@TableName varchar(50))

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

--wyłącza wyzwalacz dla podanej tabeli słownikowej

--

--parametry wejściowe:

--@TableName - nazwa tabeli słownikowej

--

/\*

exec up\_SetTriggerOFF @TableName='tbl\_status'

--

--Wynik działania:

--Wyzwalacz trg\_EmployeePositionEditBlock zostaje wyłączony

--Komunikat:

--Triger dla tabeli tbl\_status został wyłączony

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

\*/

CREATE OR ALTER PROCEDURE up\_SetTriggerOFF

@TableName VARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

IF @TableName IN ('employee.tbl\_employee\_position', 'tbl\_employee\_position', '[employee].[tbl\_employee\_position]')

BEGIN

DISABLE TRIGGER trg\_EmployeePositionEditBlock ON tbl\_employee\_position;

PRINT 'Triger dla tabeli ' + @TableName + ' został wyłączony.';

END

ELSE IF @TableName IN ('tbl\_status', '[room.tbl\_status', '[room].[tbl\_status]')

BEGIN

DISABLE TRIGGER trg\_RoomStatusEditBlock ON tbl\_status;

PRINT 'Triger dla tabeli ' + @TableName + ' został wyłączony.';

END

ELSE IF @TableName IN ('tbl\_reservation\_status', 'reservation.tbl\_reservation\_status', '[reservation].[tbl\_reservation\_status]')

BEGIN

DISABLE TRIGGER trg\_ReservationStatusEditBlock ON tbl\_reservation\_status;

PRINT 'Triger dla tabeli ' + @TableName + ' został wyłączony.';

END

ELSE IF @TableName IN ('tbl\_room\_type', 'room.tbl\_room\_type', '[room].[tbl\_room\_type]')

BEGIN

DISABLE TRIGGER trg\_RoomTypeEditBlock ON [room].[tbl\_room\_type];

PRINT 'Triger dla tabeli ' + @TableName + ' został wyłączony.';

END

ELSE IF @TableName IN ('tbl\_payment\_type', 'payment.tbl\_payment\_type', '[payment].[tbl\_payment\_type]')

BEGIN

DISABLE TRIGGER trg\_PaymentTypeEditBlock ON [payment].[tbl\_payment\_type];

PRINT 'Triger dla tabeli ' + @TableName + ' został wyłączony.';

END

ELSE

BEGIN

RAISERROR('Nieprawidłowa nazwa tabeli.', 16, 1);

RETURN;

END;

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorState = ERROR\_STATE();

RAISERROR('Wystąpił błąd podczas wyłączania trigerów: %s', @ErrorState, 1, @ErrorMessage);

END CATCH;

END;

statusu pokoju na podstawie numeru pokoju (@RoomNumber) i nowego statusu (@NewStatus).

--parametry wejściowe:`

---@RoomNumber (INT): Przechowuje numer pokoju, dla którego ma zostać zmieniony status.

---@NewStatus (INT): Przechowuje nowy identyfikator statusu pokoju.

---Zwrócono wartość: (1 row affected)

CREATE OR ALTER PROCEDURE up\_ChangeRoomStatus

@RoomNumber INT,

@NewStatus INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

UPDATE [hotel].[tbl\_rooms]

SET IdRoomStatus = @NewStatus

WHERE RoomNumber = @RoomNumber

AND EXISTS (

SELECT 1

FROM [room].[tbl\_status]

WHERE IdRoomStatus = @NewStatus

);

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

SELECT @ErrorMessage = ERROR\_MESSAGE();

RAISERROR('Wystąpił błąd podczas zmiany statusu pokoju: %s', 16, 1, @ErrorMessage);

END CATCH;

END;

--Przykład użycia

EXEC up\_ChangeRoomStatus @RoomNumber = 101, @NewStatus = 2;

----Procedura up\_DeleteEmployee została utworzona w celu usunięcia pracownika z bazy danych na podstawie identyfikatora pracownika (@EmployeeId).

--parametry wejściowe:

--@EmployeeId (int): Przechowuje identyfikator pracownika, który ma zostać usunięty.

CREATE OR ALTER PROCEDURE up\_DeleteEmployee

@EmployeeId INT

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

-- Usunięcie powiązanych rekordów w tabeli tbl\_transactions

DELETE FROM hotel.tbl\_transactions

WHERE IdEmployee = @EmployeeId;

-- Usunięcie powiązanych rekordów w tabeli tbl\_guests

DELETE FROM hotel.tbl\_guests

WHERE IdPersonData = (SELECT IdPersonData FROM hotel.tbl\_employees WHERE hotel.tbl\_employees.IdEmployee = @EmployeeId);

-- Usunięcie pracownika z tabeli tbl\_employees

DELETE FROM hotel.tbl\_employees

WHERE hotel.tbl\_employees.IdEmployee = @EmployeeId;

-- Usunięcie danych osobowych pracownika z tabeli tbl\_people\_data

DELETE FROM hotel.tbl\_people\_data

WHERE IdPersonData = (SELECT IdPersonData FROM hotel.tbl\_employees WHERE hotel.tbl\_employees.IdEmployee = @EmployeeId);

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

SELECT @ErrorMessage = ERROR\_MESSAGE();

RAISERROR('Wystąpił błąd podczas usuwania pracownika: %s', 16, 1, @ErrorMessage);

END CATCH;

END;

--Przykład użycia

EXEC up\_DeleteEmployee

@EmployeeId = 123;

----Procedura up\_DeleteHotel została utworzona lub zmodyfikowana w celu usunięcia hotelu z bazy danych na podstawie nazwy hotelu (@HotelName).

--parametry wejściowe:

--@HotelName (VARCHAR(100)): Przechowuje nazwę hotelu, który ma zostać usunięty.

CREATE OR ALTER PROCEDURE up\_DeleteHotel

@HotelName VARCHAR(100)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

IF NOT EXISTS (

SELECT 1

FROM hotel.tbl\_hotels

WHERE HotelName = @HotelName

)

BEGIN

RAISERROR('Nie znaleziono hotelu o podanej nazwie.', 16, 1);

RETURN;

END;

DELETE FROM hotel.tbl\_hotels

WHERE HotelName = @HotelName;

SELECT 'Hotel został pomyślnie usunięty.' AS Result;

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorState = ERROR\_STATE();

RAISERROR('Wystąpił błąd podczas usuwania hotelu: %s', @ErrorState, 1, @ErrorMessage);

END CATCH;

END;

--Przykład użycia

EXEC up\_DeleteHotel

@HotelName = 'Hotel ABC';

--Procedura up\_UpdateEmployeePositionByPesel została utworzona w celu aktualizacji stanowiska pracownika na podstawie numeru PESEL..

--parametry wejściowe:

--@PeselNumber (varchar(11)): Przechowuje numer PESEL pracownika, którego stanowisko ma zostać zaktualizowane.

--@NewPosition (varchar(100)): Przechowuje nazwę nowego stanowiska, na które ma zostać zaktualizowane.

CREATE PROCEDURE up\_UpdateEmployeePositionByPesel

@PeselNumber VARCHAR(11),

@NewPosition VARCHAR(100)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

UPDATE hotel.tbl\_employees

SET IdEmployeePosition = (

SELECT IdEmployeePosition

FROM HotelDatabase.employee.tbl\_employee\_position

WHERE PositionName = @NewPosition

)

WHERE PeselNumber = @PeselNumber;

IF @@ROWCOUNT > 0

BEGIN

PRINT 'Stanowisko pracownika zostało zaktualizowane.';

END

ELSE

BEGIN

PRINT 'Nie znaleziono pracownika o podanym numerze PESEL.';

END;

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorState = ERROR\_STATE();

RAISERROR('Wystąpił błąd podczas aktualizowania stanowiska pracownika: %s', @ErrorState, 1, @ErrorMessage);

END CATCH;

END;

--Przykład użycia

EXEC up\_UpdateEmployeePositionByPesel @PeselNumber = '58110187843', @NewPosition = 'Cook';

--Procedura up\_DeleteGuestByPassport została utworzona w celu usunięcia gościa z bazy danych na podstawie numeru paszportowego.

--parametry wejściowe:

--@PassportNumber (varchar(20)): Przechowuje numer paszportu gościa, który ma zostać usunięty.--/\*

CREATE PROCEDURE up\_DeleteGuestByPassport

@PassportNumber VARCHAR(20)

AS

CREATE OR ALTER PROCEDURE up\_DeleteGuest

@PassportNumber VARCHAR(9)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

-- Sprawdź, czy gość istnieje na podstawie numeru paszportowego

DECLARE @GuestId INT;

SELECT @GuestId = IdGuest

FROM hotel.tbl\_guests

WHERE PassportNumber = @PassportNumber;

IF @GuestId IS NOT NULL

BEGIN

-- Usuń powiązane wpisy z tabeli hotel.tbl\_billing

DELETE FROM hotel.tbl\_billing

WHERE IdGuest = @GuestId;

-- Usuń gościa z tabeli hotel.tbl\_guests

DELETE FROM hotel.tbl\_guests

WHERE IdGuest = @GuestId

SELECT 'Gość został pomyślnie usunięty.' AS Result;

END

ELSE

BEGIN

RAISERROR('Nie znaleziono gościa o podanym numerze paszportowym.', 16, 1);

RETURN;

END;

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorState = ERROR\_STATE();

RAISERROR('Wystąpił błąd podczas usuwania gościa: %s', @ErrorState, 1, @ErrorMessage);

END CATCH;

END;

--Przykład użycia

EXEC up\_DeleteGuestByPassport @PassportNumber = 'AB1234567';

exec sp\_helptext up\_SetTriggerON

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

--- PROCEDURE DEFINITION

--- up\_SetTriggerON (@TableName varchar(50))

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

--włącza wyzwalacz dla podanej tabeli słownikowej

--

--parametry wejściowe:

--@TableName - nazwa tabeli słownikowej

--

/\*

exec up\_SetTriggerON @TableName='tbl\_status'

--

--Wynik działania:

--Wyzwalacz trg\_EmployeePositionEditBlock zostaje włączony

--Komunikat:

--Triger dla tabeli tbl\_status został włączony

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

\*/

CREATE OR ALTER PROCEDURE up\_SetTriggerON

@TableName VARCHAR(50)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

IF @TableName IN ('employee.tbl\_employee\_position', 'tbl\_employee\_position', '[employee].[tbl\_employee\_position]')

BEGIN

ENABLE TRIGGER trg\_EmployeePositionEditBlock ON tbl\_employee\_position;

PRINT 'Triger dla tabeli ' + @TableName + ' został włączony.';

END

ELSE IF @TableName IN ('tbl\_status', '[room.tbl\_status', '[room].[tbl\_status]')

BEGIN

ENABLE TRIGGER trg\_RoomStatusEditBlock ON tbl\_status;

PRINT 'Triger dla tabeli ' + @TableName + ' został włączony.';

END

ELSE IF @TableName IN ('tbl\_reservation\_status', 'reservation.tbl\_reservation\_status', '[reservation].[tbl\_reservation\_status]')

BEGIN

ENABLE TRIGGER trg\_ReservationStatusEditBlock ON tbl\_reservation\_status;

PRINT 'Triger dla tabeli ' + @TableName + ' został włączony.';

END

ELSE IF @TableName IN ('tbl\_room\_type', 'room.tbl\_room\_type', '[room].[tbl\_room\_type]')

BEGIN

ENABLE TRIGGER trg\_RoomTypeEditBlock ON [room].[tbl\_room\_type];

PRINT 'Triger dla tabeli ' + @TableName + ' został włączony.';

END

ELSE IF @TableName IN ('tbl\_payment\_type', 'payment.tbl\_payment\_type', '[payment].[tbl\_payment\_type]')

BEGIN

ENABLE TRIGGER trg\_PaymentTypeEditBlock ON [payment].[tbl\_payment\_type];

PRINT 'Triger dla tabeli ' + @TableName + ' został włączony.';

END

ELSE

BEGIN

RAISERROR('Nieprawidłowa nazwa tabeli.', 16, 1);

RETURN;

END;

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorState = ERROR\_STATE();

RAISERROR('Wystąpił błąd podczas włączania trigerów: %s', @ErrorState, 1, @ErrorMessage);

END CATCH;

END;

----Procedura up\_UpdateEmployee została utworzona lub zmodyfikowana w celu aktualizacji informacji o pracowniku na podstawie identyfikatora pracownika (@EmployeeId).

--parametry wejściowe:

--@EmployeeId (int): Przechowuje identyfikator pracownika, dla którego mają zostać zaktualizowane informacje.

--@FirstName (varchar(15)): Przechowuje nowe imię pracownika.

--@LastName (varchar(30)): Przechowuje nowe nazwisko pracownika.

--@EmployeePositionId (int): Przechowuje nowy identyfikator stanowiska pracownika.

--@DateOfEmployment (date): Przechowuje nową datę zatrudnienia pracownika.

--@HotelId (int): Przechowuje nowy identyfikator hotelu, w którym pracownik jest zatrudniony.

--@PeselNumber (varchar(11)): Przechowuje nowy numer PESEL pracownika.

CREATE OR ALTER PROCEDURE up\_UpdateEmployee

(

@EmployeeId INT,

@FirstName VARCHAR(15),

@LastName VARCHAR(30),

@EmployeePositionId INT,

@DateOfEmployment DATE,

@HotelId INT,

@PeselNumber VARCHAR(11)

)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

UPDATE hotel.tbl\_employees

SET

[Name] = @FirstName,

LastName = @LastName,

IdEmployeePosition = @EmployeePositionId,

DateOfEmployment = @DateOfEmployment,

IdHotel = @HotelId,

PeselNumber = @PeselNumber

WHERE

IdEmployee = @EmployeeId;

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorState = ERROR\_STATE();

RAISERROR('Wystąpił błąd podczas aktualizowania pracownika: %s', @ErrorState, 1, @ErrorMessage);

END CATCH;

END;

--Przykład użycia

EXEC up\_UpdateEmployee

@EmployeeId = 1,

@FirstName = 'John',

@LastName = 'Doe',

@EmployeePositionId = 2,

@DateOfEmployment = '2022-01-01',

@HotelId = 3,

@PeselNumber = '12345678901';

----Procedura up\_UpdateReservation została utworzona w celu dodania nowych transkacji

--parametry wejściowe:

--@IdReservation INT,

--@CheckInDate DATE,

--@CheckOutDate DATE,

--@NumberOfGuests VARCHAR(3)

CREATE PROCEDURE up\_UpdateReservation

@IdReservation INT,

@CheckInDate DATE,

@CheckOutDate DATE,

@NumberOfGuests VARCHAR(3)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

UPDATE hotel.tbl\_reservations

SET CheckInDate = @CheckInDate,

CheckOutDate = @CheckOutDate,

NumberOfGuests = @NumberOfGuests

WHERE IdReservation = @IdReservation;

IF @@ROWCOUNT > 0

BEGIN

PRINT 'Rezerwacja została zaktualizowana.';

END

ELSE

BEGIN

PRINT 'Nie znaleziono rezerwacji o podanym identyfikatorze.';

END;

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorState = ERROR\_STATE();

RAISERROR('Wystąpił błąd podczas aktualizowania rezerwacji: %s', @ErrorState, 1, @ErrorMessage);

END CATCH;

END;

----Przykład użycia

EXEC up\_UpdateReservation

@IdReservation = 1,

@CheckInDate = '2023-06-20',

@CheckOutDate = '2023-06-25',

@NumberOfGuests = '2';

----Wynik

Commands completed successfully.

--Procedura up\_UpdateReservationStatusByRoom została utworzona w celu aktualizacji statusu rezerwacji dla pokoju na podstawie numeru pokoju...

--parametry wejściowe:

--@RoomNumber (int): Przechowuje numer pokoju, dla którego ma zostać zaktualizowany status rezerwacji.

--@NewStatusName (varchar(100)): Przechowuje nazwę nowego statusu rezerwacji, na który ma zostać zaktualizowany.

--Zwrócono wartość: Commands completed successfully.

CREATE PROCEDURE up\_UpdateReservationStatusByRoom

@RoomNumber INT,

@NewStatusName VARCHAR(100)

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRY

UPDATE [HotelDatabase].[hotel].[tbl\_reservations]

SET IdReservationStatus = (SELECT IdReservationStatus FROM [HotelDatabase].[reservation].[tbl\_reservation\_status] WHERE StatusName = @NewStatusName)

WHERE IdRoom IN (SELECT IdRoom FROM [HotelDatabase].[hotel].[tbl\_rooms] WHERE RoomNumber = @RoomNumber);

IF @@ROWCOUNT > 0

BEGIN

PRINT 'Status rezerwacji został zaktualizowany dla pokoju o numerze ' + CAST(@RoomNumber AS VARCHAR(10)) + '.';

END

ELSE

BEGIN

PRINT 'Nie znaleziono rezerwacji dla pokoju o numerze ' + CAST(@RoomNumber AS VARCHAR(10)) + '.';

END;

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorState = ERROR\_STATE();

RAISERROR('Wystąpił błąd podczas aktualizowania statusu rezerwacji: %s', @ErrorState, 1, @ErrorMessage);

END CATCH;

END;

--Przykład użycia

EXEC up\_UpdateReservationStatusByRoom @RoomNumber = 101, @NewStatusName = 'Confirmed';

# 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\_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

INNER JOIN [hotel].[tbl\_transactions] AS t ON g.IdGuest = t.IdGuest

) AS p

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

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

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

)

GO

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

* 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;

IF EXISTS (SELECT 1 FROM [hotel].[tbl\_reservations] WHERE [IdReservation] = @id\_reservation)

BEGIN

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

END

else BEGIN

SET @CheckInDate = NULL;

END;

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;

IF EXISTS (SELECT 1 FROM [hotel].[tbl\_reservations] WHERE [IdReservation] = @id\_reservation)

BEGIN

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

END

else BEGIN

SET @CheckOutDate = NULL;

END;

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;

IF EXISTS (SELECT 1 FROM [hotel].[tbl\_billing] WHERE [IdBilling] = @id\_billing)

BEGIN

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);

END

BEGIN

SET @id = NULL;

PRINT("Nieprawidłowe Id gościa.")

END;

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;

IF EXISTS (SELECT 1 FROM [hotel].[tbl\_rooms] WHERE [IdRoom] = @id\_room)

BEGIN

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

);

END

ELSE

BEGIN

SET @price = NULL;

END

return @price

ENDexec 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) <> CAST(SUBSTRING(@NIPNumber, 10, 1) AS tinyint)

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

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

\*/

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)

DECLARE @result tinyint = 0

IF ISNUMERIC(@pesel) = 0

RETURN @result

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

BEGIN

SET @result = 1

END

RETURN @result

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

AS

BEGIN

DECLARE @is\_free bit, @CheckInDate date, @CheckOutDate date, @status varchar(100);

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 IS NULL OR @status = 'Cancelled' OR @CheckOutDate <= GETDATE() OR @CheckInDate >= GETDATE())

SET @is\_free = 1;

ELSE

SET @is\_free = 0;

RETURN @is\_free;

END;

GO

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

AS

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

AS

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

ELSE

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

AS

BEGIN

DECLARE @SumOfBilling money = 0

IF EXISTS (SELECT 1 FROM [hotel].[tbl\_transactions] WHERE [IdTransaction] = @id\_transaction)

BEGIN

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

END

ELSE

BEGIN

SET @SumOfBilling = NULL

END

RETURN @SumOfBilling

END;

GO

# 16. Wykaz utworzonych wyzwalaczy













# 17. Skrypt tworzenia wyzwalaczy

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

--- TRIGGER DEFINITION

--- trgtbl\_ChangeRoomStatus

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

--zmiana statusu pokoju

--wywoływany podczas dodawania lub edycji rekordu danych w tabeli tbl\_reservations

--

--parametry wyjściowe/zwracane wartości:

--zmiana statusu pokoju na wolny/zajęty

--

/\*

INSERT INTO [hotel].[tbl\_reservations]

([IdRoom]

,[CheckInDate]

,[CheckOutDate]

,[NumberOfGuests]

,[IdReservationStatus]

,[IdHotel])

VALUES (1,'14.06.2023','17.06.2023', 2, 2, 1)

--

--Wynik działania

--Dane zostaną dodane do tabeli tbl\_reservations, status pokoju o Id=1 zmieni się na zajęty

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

\*/

USE HotelDatabase;

GO

CREATE OR ALTER TRIGGER trgtbl\_ChangeRoomStatus

ON hotel.tbl\_reservations

AFTER INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

DECLARE @IdReservationStatus int;

DECLARE @ReservationStatus varchar(20);

DECLARE @IdRoom int;

SELECT TOP 1 @IdReservationStatus = i.IdReservationStatus,

@IdRoom = i.IdRoom

FROM inserted i;

SELECT @ReservationStatus = s.StatusName

FROM reservation.tbl\_reservation\_status s

WHERE s.IdReservationStatus = @IdReservationStatus;

BEGIN TRY

IF @ReservationStatus = 'Checked In'

BEGIN

UPDATE r

SET r.IdRoomStatus = s.IdRoomStatus

FROM hotel.tbl\_rooms r

JOIN room.tbl\_status s ON r.IdRoomStatus = s.IdRoomStatus

WHERE r.IdRoom = @IdRoom AND s.RoomStatusName = 'occupied';

END

ELSE IF @ReservationStatus = 'Checked Out'

BEGIN

UPDATE r

SET r.IdRoomStatus = s.IdRoomStatus

FROM hotel.tbl\_rooms r

JOIN room.tbl\_status s ON r.IdRoomStatus = s.IdRoomStatus

WHERE r.IdRoom = @IdRoom AND s.RoomStatusName = 'clean';

END;

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(MAX);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT @ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);

END CATCH;

END;

GO

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

--- TRIGGER DEFINITION

--- trgtbl\_EmployeePositionEditBlock

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

--blokowanie edycji tabeli słownikowej: Employee\_Position

--wywoływany podczas próby wstawiania, edycji lub usuwania rekordów z tabeli

--

--parametry wyjściowe/zwracane wartości:

--zmiana nie zostanie wprowadzona

--

/\*

USE [HotelDatabase]

GO

DELETE FROM [employee].[tbl\_employee\_position]

GO

--

--Wynik działania

--Dane nie zostaną usunięte z tabeli

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

\*/

use HotelDatabase

go

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

INSTEAD OF INSERT, UPDATE, DELETE

AS

RAISERROR ('Edycja tabeli employee.tbl\_employee\_position jest niedozwolona.', 16, 1);

RETURN;

GO

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

--- TRIGGER DEFINITION

--- trgtbl\_PaymentTypeEditBlock

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

--blokowanie edycji tabeli słownikowej: payment.tbl\_payment\_type

--wywoływany podczas próby wstawiania, edycji lub usuwania rekordów z tabeli

--

--parametry wyjściowe/zwracane wartości:

--zmiana nie zostanie wprowadzona

--

/\*

USE [HotelDatabase]

GO

DELETE FROM payment.tbl\_payment\_type

GO

--

--Wynik działania

--Dane nie zostaną usunięte z tabeli

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

\*/

use HotelDatabase

go

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

INSTEAD OF INSERT, UPDATE, DELETE

AS

RAISERROR ('Edycja tabeli payment.tbl\_payment\_type jest niedozwolona.', 16, 1);

RETURN;

GO

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

--- TRIGGER DEFINITION

--- trgtbl\_ReservationDateBlock

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

--blokowanie wprowadzenia daty wymeldowania przed datą zameldowania oraz wprowadzenia rezerwacji z datą wcześniejszą niż data dzisiejsza

--wywoływany w trakcie wprowadzania/edycji daty zameldowania i daty wymeldowania

--

--parametry wyjściowe/zwracane wartości:

--podczas edycji dat na wcześniejsze niż obecna lub próby edycji daty wymeldowania na wcześniejszą niż zameldowania zmiany zostają zablokowane

--

/\*

USE [HotelDatabase]

GO

INSERT INTO [hotel].[tbl\_reservations]

([IdRoom]

,[CheckInDate]

,[CheckOutDate]

,[NumberOfGuests]

,[IdReservationStatus]

,[IdHotel])

VALUES (1,'01.03.2012','01.04.2023', 2, 1, 1)

GO

--

--Wynik działania

--Rekord nie zostaje dodany do tabeli.

--Komunikat:

--Msg 50000, Level 16, State 1, Procedure trgtbl\_ReservationDateBlock, Line 33 [Batch Start Line 1340]

Data wymeldowania nie może być wcześniejsza niż obecna data.

Msg 3609, Level 16, State 1, Line 1341

The transaction ended in the trigger. The batch has been aborted.

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

\*/

USE HotelDatabase;

GO

CREATE OR ALTER TRIGGER trgtbl\_ReservationDateBlock

ON hotel.tbl\_reservations

INSTEAD OF UPDATE, INSERT

AS

BEGIN

declare @IdReservation int;

declare @IdRoom int;

declare @CheckInDate Date;

declare @CheckOutDate Date;

declare @NumberOfGuest int;

declare @IdReservationStatus int;

declare @IdHotel int;

IF EXISTS (

SELECT 1

FROM inserted i

WHERE i.CheckInDate > i.CheckOutDate

)

BEGIN

RAISERROR('Data zameldowania nie może być późniejsza niż data wymeldowania.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END;

IF EXISTS (

SELECT 1

FROM inserted i

WHERE i.CheckOutDate < GETDATE()

)

BEGIN

RAISERROR('Data wymeldowania nie może być wcześniejsza niż obecna data.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END;

IF EXISTS (

SELECT 1

FROM inserted i

WHERE i.CheckInDate < CONVERT(DATE, GETDATE())

)

BEGIN

RAISERROR('Data zameldowania nie może być wcześniejsza niż dzisiejsza data.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END;

IF EXISTS (SELECT \* FROM deleted)

BEGIN

Select @IdReservation=IdReservation, @IdRoom=IdRoom,

@CheckInDate=CheckInDate, @CheckOutDate=CheckOutDate, @NumberOfGuest=NumberOfGuests,

@IdReservationStatus=IdReservationStatus, @IdHotel=IdHotel

from inserted

update hotel.tbl\_reservations set IdRoom=@IdRoom,

CheckInDate=@CheckInDate, CheckOutDate=@CheckOutDate,

NumberOfGuests=@NumberOfGuest, IdReservationStatus=@IdReservationStatus,

IdHotel=@IdHotel where IdReservation=@IdReservation

END;

ElSE IF EXISTS (SELECT \* FROM inserted)

BEGIN

Select @IdRoom=IdRoom,@CheckInDate=CheckInDate,

@CheckOutDate=CheckOutDate, @NumberOfGuest=NumberOfGuests,

@IdReservationStatus=IdReservationStatus, @IdHotel=IdHotel

from inserted

Insert into hotel.tbl\_reservations(IdRoom, CheckInDate,CheckOutDate, NumberOfGuests, IdReservationStatus, IdHotel)

values(@IdRoom, @CheckInDate, @CheckOutDate, @NumberOfGuest, @IdReservationStatus, @IdHotel)

END;

END;

GO

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

--- TRIGGER DEFINITION

--- trgtbl\_ReservationStatusEditBlock

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

--blokowanie edycji tabeli słownikowej: reservation.tbl\_reservation\_status

--wywoływany podczas próby wstawiania, edycji lub usuwania rekordów z tabeli

--

--parametry wyjściowe/zwracane wartości:

--zmiana nie zostanie wprowadzona

--

/\*

USE [HotelDatabase]

GO

DELETE FROM reservation.tbl\_reservation\_status

GO

--

--Wynik działania

--Dane nie zostaną usunięte z tabeli

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

\*/

use HotelDatabase

go

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

INSTEAD OF INSERT, UPDATE, DELETE

AS

RAISERROR('Edycja tabeli "tbl\_reservation\_status" jest niedozwolona.', 16, 1);

RETURN;

GO

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

--- TRIGGER DEFINITION

--- trgtbl\_RoomStatusEditdBlock

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

--blokowanie edycji tabeli słownikowej: [room].[tbl\_status]

--wywoływany podczas próby wstawiania, edycji lub usuwania rekordów z tabeli

--

--parametry wyjściowe/zwracane wartości:

--zmiana nie zostanie wprowadzona

--

/\*

USE [HotelDatabase]

GO

DELETE FROM [room].[tbl\_status]

GO

--

--Wynik działania

--Dane nie zostaną usunięte z tabeli

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

\*/

use HotelDatabase

go

CREATE TRIGGER trgtbl\_RoomStatusEditdBlock ON room.tbl\_status

INSTEAD OF INSERT, UPDATE, DELETE

AS

RAISERROR('Edycja tabeli "tbl\_status" jest niedozwolona.', 16, 1);

RETURN;

GO

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

--- TRIGGER DEFINITION

--- trgtbl\_RoomTypeEditBlock

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

--blokowanie edycji tabeli słownikowej: room.tbl\_room\_type

--wywoływany podczas próby wstawiania, edycji lub usuwania rekordów z tabeli

--

--parametry wyjściowe/zwracane wartości:

--zmiana nie zostanie wprowadzona

--

/\*

USE [HotelDatabase]

GO

DELETE FROM room.tbl\_room\_type

GO

--

--Wynik działania

--Dane nie zostaną usunięte z tabeli

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

\*/

use HotelDatabase

go

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

INSTEAD OF INSERT, UPDATE, DELETE

AS

RAISERROR('Edycja tabeli "tbl\_room\_type" jest niedozwolona.', 16, 1);

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. Schemat zaprojektowanego systemu przedstawiający punkty wejścia i wyjścia

Rys. 3. Schemat dla funkcji uf\_SearchGuestByNameOrLastName

Obraz zawierający tekst, diagram, Równolegle, linia

Opis wygenerowany automatycznie

Rys. 4. Schemat dla funkcji IsRoomFree

Obraz zawierający tekst, linia, diagram, Wykres

Opis wygenerowany automatycznie

Rys. 5. Schemat dla funkcji uf\_NumberOfGuestReservations

Obraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatycznie

Rys. 6. Schemat dla funkcji uf\_getRoomPrice

Obraz zawierający tekst, paragon, diagram, zrzut ekranu

Opis wygenerowany automatycznie

Rys. 7. Schemat dla funkcji uf\_getGuestId

Obraz zawierający tekst, zrzut ekranu, linia, diagram

Opis wygenerowany automatycznie

Rys. 8. Schemat dla funkcji uf\_getCheckOutDate

Obraz zawierający tekst, zrzut ekranu, linia, numer

Opis wygenerowany automatycznie

Rys. 9. Schemat dla funkcji uf\_getCheckInDate

Obraz zawierający tekst, zrzut ekranu, linia, numer

Opis wygenerowany automatycznie

Rys. 10. Schemat dla funkcji uf\_LengthOfTheVisit

Obraz zawierający tekst, zrzut ekranu, numer, linia

Opis wygenerowany automatycznie

Rys. 11. Schemat dla funkcji uf\_setRoomStatusReserved

Obraz zawierający tekst, zrzut ekranu, numer, linia

Opis wygenerowany automatycznie

Rys. 12. Schemat dla funkcji uf\_SumOfBilling

Obraz zawierający diagram, tekst, linia, Równolegle

Opis wygenerowany automatycznie

Rys. 13. Schemat dla procedury up\_AddEmployee

Obraz zawierający tekst, diagram, zrzut ekranu, numer

Opis wygenerowany automatycznie

Rys. 14. Schemat dla procedury up\_AddGuest

Obraz zawierający tekst, zrzut ekranu, diagram, numer

Opis wygenerowany automatycznie

Rys. 15. Schemat dla procedury up\_DeleteGuestByPassport

Obraz zawierający tekst, zrzut ekranu, diagram, numer

Opis wygenerowany automatycznie

Rys. 16. Schemat dla procedury up\_UpdateEmployeePositionByPesel

Obraz zawierający tekst, zrzut ekranu, numer, diagram

Opis wygenerowany automatycznie

Rys. 17. Schemat dla procedury up\_UpdateReservationStatusByRoom

Obraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatycznie

Rys. 18. Schemat dla procedury up\_DeleteEmployee

Obraz zawierający tekst, zrzut ekranu, Czcionka, diagram

Opis wygenerowany automatycznie

Rys. 19. Schemat dla procedury up\_UpdateEmployee

Obraz zawierający tekst, zrzut ekranu, diagram, numer

Opis wygenerowany automatycznie

Rys. 20. Schemat dla procedury up\_ChangeRoomStatus

Obraz zawierający tekst, zrzut ekranu, diagram, Równolegle

Opis wygenerowany automatycznie

Rys. 21. Schemat dla procedury up\_addReservation

Obraz zawierający tekst, zrzut ekranu, diagram, Równolegle

Opis wygenerowany automatycznie

Rys. 22. Schemat dla procedury up\_SetTriggerOFF

Obraz zawierający tekst, diagram, Równolegle, linia

Opis wygenerowany automatycznie

Rys. 23. Schemat dla procedury up\_SetTriggerON

Obraz zawierający tekst, diagram, Równolegle, linia

Opis wygenerowany automatycznie

Rys. 24. Schemat dla procedury up\_AddTransactionWithPayment

Obraz zawierający tekst, diagram, Plan, Rysunek techniczny

Opis wygenerowany automatycznie

Rys. 25. Schemat dla procedury up\_AddBilling

Obraz zawierający tekst, diagram, zrzut ekranu, numer

Opis wygenerowany automatycznie

Rys. 26. Schemat dla procedury up\_UpdateReservation

Obraz zawierający tekst, zrzut ekranu, linia, diagram

Opis wygenerowany automatycznie

# 21. 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.

# Spis ilustracji

[Rys. 1. Model logiczny bazy danych 4](#_Toc137596327)

[Rys. 2. Model fizyczny bazy danych 4](#_Toc137596328)

[Rys. 3. Schemat dla funkcji uf\_SearchGuestByNameOrLastName 57](#_Toc137596329)

[Rys. 4. Schemat dla funkcji IsRoomFree 57](#_Toc137596330)

[Rys. 5. Schemat dla funkcji uf\_NumberOfGuestReservations 58](#_Toc137596331)

[Rys. 6. Schemat dla funkcji uf\_getRoomPrice 58](#_Toc137596332)

[Rys. 7. Schemat dla funkcji uf\_getGuestId 58](#_Toc137596333)

[Rys. 8. Schemat dla funkcji uf\_getCheckOutDate 59](#_Toc137596334)

[Rys. 9. Schemat dla funkcji uf\_getCheckInDate 59](#_Toc137596335)

[Rys. 10. Schemat dla funkcji uf\_LengthOfTheVisit 59](#_Toc137596336)

[Rys. 11. Schemat dla funkcji uf\_setRoomStatusReserved 59](#_Toc137596337)

[Rys. 12. Schemat dla funkcji uf\_SumOfBilling 60](#_Toc137596338)

[Rys. 13. Schemat dla procedury up\_AddEmployee 60](#_Toc137596339)

[Rys. 14. Schemat dla procedury up\_AddGuest 61](#_Toc137596340)

[Rys. 15. Schemat dla procedury up\_DeleteGuestByPassport 61](#_Toc137596341)

[Rys. 16. Schemat dla procedury up\_UpdateEmployeePositionByPesel 62](#_Toc137596342)

[Rys. 17. Schemat dla procedury up\_UpdateReservationStatusByRoom 62](#_Toc137596343)

[Rys. 18. Schemat dla procedury up\_DeleteEmployee 63](#_Toc137596344)

[Rys. 19. Schemat dla procedury up\_UpdateEmployee 63](#_Toc137596345)

[Rys. 20. Schemat dla procedury up\_ChangeRoomStatus 64](#_Toc137596346)

[Rys. 21. Schemat dla procedury up\_addReservation 64](#_Toc137596347)

[Rys. 22. Schemat dla procedury up\_SetTriggerOFF 65](#_Toc137596348)

[Rys. 23. Schemat dla procedury up\_SetTriggerON 65](#_Toc137596349)

[Rys. 24. Schemat dla procedury up\_AddTransactionWithPayment 66](#_Toc137596350)

[Rys. 25. Schemat dla procedury up\_AddBilling 66](#_Toc137596351)

[Rys. 26. Schemat dla procedury up\_UpdateReservation 67](#_Toc137596352)