Temat: Hotel - Rezerwacja Miejsc Noclegowych

Autorzy: Wojciech Kwiatkowski, Adam Orzeł, Bartosz Lasoń, Kacper Kaleta, Marcin Marszałek

1. Zakres i krótki opis systemu

Celem projektu jest stworzenie systemu umożliwiającego ewidencjonowanie i zarządzanie rezerwacjami pokoi hotelowych.

Hotel oferuje pokoje na wynajem. Pokoje są jedno, dwu, trzy-cztero osobowe. Oferują różny stopień wyposażenia: balkon (możliwość palenia papierosów), aneks, klimatyzacja, telewizor, wanna lub prysznic. Rezerwacja może zostać poszerzona o dodatkowe usługi: sauna i/lub wypożyczenie rowerów. Hotel oferuje trzy standardy wyżywienia: śniadanie, obiadokolacja + śniadanie, all inclusive.

Zakres wynajmu wynosi od 1 doby do maksymalnie 2 tygodni. Rezerwacja musi zostac dokonana minimum na 48h przed zameldowaniem, aby system mógł zatwierdzić rezerwacje klienta. W czasie rezerwacji system będzie sprawdzał czy dany pokój jest dostępny, jeśli nie to czy jest wolny podobny pokój o szukanych wymaganiach. Jeśli klient zdecyduje się przedłużyc okres wynajmu, system sprawdza czy konkretny pokój nie został uwcześnie zarezerwowany przez innego klienta w danym terminie.

Informacje dla nas na podstawie których będziemy tworzyć rezerwacje w bazie danych.

Cenna bazowa za rozmiar pokoju:

- jednoosobowe 150 zł
- dwuosobowe 220 zł
- trzyosobowy 300 zł
- czteroosobowy 420 zł

Kategorie pokojów

- economic(bez dodatkowego wyposażenia) + 0 zł
- standard (telewizor, balkon) + 50 zł
- premium(standard + klimatyzacja, aneks) + 110 zł
- exclusive(premium + wanna) + 160 zł

Dodatkowe usługi:

- możliwoś korzystania z sauny: 20zł
- wypożyczenie roweru: 30zł

Wyżywienie za dzień:

- śniadanie: 15zł
- obiadokolacja + śniadanie: 40zł
- all inclusive: 80zł

Na podstawie wybranych parametrów pokoju oraz oferty usług wyliczany będzie koszt całkowity za pobyt klienta/ów. Jeśli czas pobytu będzie przekraczał tydzień zostanie doliczony rabat w wysokości 10 % od całkowitej kwoty pobytu. Rabat za czas można łączyć z pozostałymi rabatami.

2. Wymagania i funkcje systemu

Lista wymagań:

- wyświetlanie specyfikacji pokoju
- wyświetlanie informacji o rezerwacji
- wyświetlanie informacji o dostepnych pokojach w danych terminach
- dodawanie rezerwacji
- modyfikacja rezerwacji
- anulowanie rezerwacji
- obliczanie całkowitego kosztu pobytu

Przypadki użycia:

1. Klient chce zapoznać się z ofertą dostępnym pokoi w konkretnym terminie.
2. Klient chce zapoznać się z wyposażeniem wybranego pokoju.
3. Klient chce zapoznać z dodatkowymi usługami hotelu.
4. Klient chce zarezerwować pokój.
5. Klient chce przedłużyć pobyt.
6. Klient chce dostać kwote wynajmu.
7. Pracownik chce sprawdzić kiedy pokój zostanie zwolniony

3. Projekt bazy danych

Schemat bazy danych

Opis poszczególnych tabel

(Dla każdej tabeli opis w formie tabelki)

Nazwa tabeli: (nazwa tabeli)

- Opis: (opis tabeli, komentarz)

Tabela rezerwacje

Opis: Tabela rezerwacje przechowuje informacje o rezerwacjach dokonywanych przez klientów. Każdy rekord w tej tabeli odnosi się do konkretnej rezerwacji, zawierając szczegóły takie jak dane klienta, daty zameldowania i wymeldowania, status rezerwacji oraz ewentualny rabat.

Nazwa atrybutu	Typ	Opis/Uwagi
ID	int	Primary key, auto increment
id_klienta	int	Foreign key
data_zamelodwania	date	Data zameldowania w pokoju
data_wymeldowania	date	Data wymedlowania z pokoju
data_rezerwacji	date	Data rezerwacji pokoju
id_status	int	Foreign key
rabat	numeric	Opcjonalnie doliczany rabat przy kwocie końcowej za pobyty wyrażony w %

Tabela wyżywienie

Opis: Tabela łącząca tabelę rezerwacje z typ_wyzywienia. Odpowiada za zapis poszczególnych rezerwacji wyżywień (można zobaczyć starą cenę wyżywienia)

Nazwa atrybutu	Typ	Opis/Uwagi
id_rezerwacji	int	Primary Key, Klucz główny połączony z id_typ_wyzywienia, odnosi się do id rezerwacji
id_typ_wyzywienia	int	Primary Key, Klucz główny połączony z id_rezerwacji, odnosi się do id poszczególnego typu wyżywienia
cena_wyzywienia	money	Zapis ceny poszczególnej rezerwacji typu wyżywienia

---

Tabela typ_wyzywienia

Opis: Tabela opisująca typy wyżywienia w hotelu. Można dowolnie zmieniać ceny bez wpływu na zapisane rezerwacje wyżywienia.

Nazwa atrybutu	Typ	Opis/Uwagi
id	int	Primary Key, Auto Increment, identyfikator typu wyżywienia
opis	varchar	Opis typu wyżywienia
cena	money	Cena typu wyżywienia

Tabela klienci

Opis: Tabela zawiera podstawowe informacje o klientach id klienta , imie, nazwisko i jego numer telefonu

Nazwa atrybutu	Typ	Opis/Uwagi
id	int	primary_key
imie	var	Imie klienta
nazwisko	var	Nazwisko klienta
telefon	var	Numer telefonu klienta

Tabela statusy

Opis: Tabela zawiera id statusu i nazwę statusu na jakim jest rezerwacja np.“Odrzucona”

Nazwa atrybutu	Typ	Opis/Uwagi
id	int	primary_key
nazwa	var	Nazwa statustu na jakim jest rezerwacja

Tabela usługi

Opis: Tabela łącznikowa dla tabeli rezerwacje oraz typ_usługi. Oprócz łączenia tych tabel przez ich identyfikatory posiada także atrybut cena_usługi, która wskazuje cenę za jaką usługa została przy danej rezerwacji sprzedana.

Nazwa atrybutu	Typ	Opis/Uwagi
id_typ_uslugi	integer	Primary Key razem z atrybutem id_rezerwacji, zawiera numer identyufikujący daną rezerwację
id_rezerwacji	integer	Primary Key razem z atrybutem id_typ_usługi, zawiera numer identyfikujący daną usługę
cena_uslugi	money	Atrybut określa cenę usługi dla danej rezerwacji

---

Tabela typ_usług

Opis: Tabela zawiera podstawowe informacje o dostępnych usługach, takie jak ich identyfikator, nazwa danej usługi oraz jej cena dla klienta w danej chwili.

Nazwa atrybutu	Typ	Opis/Uwagi
id	integer	Primary Key, autoincrement, zawiera numer identyfikujący daną usługę
opis	varchar	Opisowa nazwa usługi
cena	money	Atrybut określa cenę danej usługi w danym momencie

---

Tabela rezerwacje_pokoi

Opis: Tabela rezerwacje_pokoi służy do przechowywania informacji o pokojach zarezerwowanych w ramach poszczególnych rezerwacji. Pozwala na powiązanie konkretnych pokoi z rezerwacjami oraz określenie ceny za wynajem tych pokoi.

Nazwa atrybutu	Typ	Opis/Uwagi
id_rezerwacji	int	Primary key
id_pokoju	int	Primary key
cena_pokojow	money	Cena za zarezerowane pokoje

Tabela pokoje

Opis: Tabela pokoje przechowuje informacje o poszczególnych pokojach dostępnych w obiekcie hotelowym. Każdy pokój jest przypisany do określonej kategorii i konkretnego typu pokoju.

Nazwa atrybutu	Typ	Opis/Uwagi
id	int	Primary key, Auto increment
id_kategoia	int	Numer kategorii pokoju
id_typ_pokoju	int	Numer typu pokoju

Tabela kategorie_pokoju

Opis: Tabela kategorie_pokoju przechowuje informacje o różnych kategoriach pokoi dostępnych w obiekcie hotelowym. Każda kategoria pokoju charakteryzuje się unikalnymi cechami, takimi jak obecność balkonu, aneksu kuchennego, klimatyzacji, czy telewizora.

Nazwa atrybutu	Typ	Opis/Uwagi
id	int	Primary key, autoincrement
nazwa	varchar	Nazwa kategorii
czy_balkon	BIT	Czy pokoj w danej kategorii zawiera balkon
czy_aneks	BIT	Czy pokoj w danej kategorii zawiera aneks
czy_klimatyzacja	BIT	Czy pokoj w danej kategorii zawiera klimatyzacje
czy_ telewizor	BIT	Czy pokoj w danej kategorii zawiera wanne
cena	money	Cena danej kategorii

Tabela typ_pokoju

Opis: Tabela typ_pokoju przechowuje informacje o różnych typach pokoi dostępnych w hotelu. Zawiera dane dotyczące liczby osób, które mogą przebywać w pokoju, oraz ceny za poszczególny typ pokoju.

Nazwa atrybutu	Typ	Opis/Uwagi
id	int	Primary key, autoincrement
ile_osob	nvarchar	Ilość osób
cena	money	Cena danego typu

4. Implementacja

Kod poleceń DDL

(dla każdej tabeli należy wkleić kod DDL polecenia tworzącego tabelę)

Tabela rezerwacje

```
CREATE TABLE rezerwacje (
  id integer IDENTITY(1,1) PRIMARY KEY,
  id_klienta integer,
  data_zameldowania date,
  data_wymeldowania date,
  data_rezerwacji date,
  id_status integer,
  rabat numeric CHECK (rabat >= 0 AND rabat <= 100)
);
```

Tabela wyzywienie

```
CREATE TABLE wyzywienie (
  id_rezerwacji integer,
  id_typ_wyzywienia integer,
  cena_wyzywienia money CHECK (cena_wyzywienia > 0),
  PRIMARY KEY (id_rezerwacji, id_typ_wyzywienia),
);
```

Tabela typ_wyzywienia

```
CREATE TABLE typ_wyzywienia (
  id integer IDENTITY(1,1) PRIMARY KEY,
  opis nvarchar(30),
  cena money CHECK (cena > 0)
)
```

Tabela klienci

```
CREATE TABLE klienci (
  id integer IDENTITY(1,1) PRIMARY KEY,
  imie nvarchar(12),
  nazwisko nvarchar(15),
  telefon nvarchar(15)
)
```

Tabela statusy

```
CREATE TABLE statusy (
  id integer IDENTITY(1,1) PRIMARY KEY,
  nazwa nvarchar(11)
)
```

Tabela uslugi

```
CREATE TABLE uslugi (
  id_typ_uslugi integer,
  id_rezerwacji integer,
  cena_uslug money CHECK (cena_uslug > 0),
  PRIMARY KEY (id_typ_uslugi, id_rezerwacji)
)
```

Tabela typ_uslugi

```
CREATE TABLE typ_uslugi (
  id integer IDENTITY(1,1) PRIMARY KEY,
  opis nvarchar(5),
  cena money CHECK (cena > 0)
)
```

Tabela rezerwacje_pokoi

```
CREATE TABLE rezerwacje_pokoi (
  id_rezerwacji integer,
  id_pokoju integer,
  cena_pokojow money CHECK (cena_pokojow > 0),
  PRIMARY KEY (id_rezerwacji, id_pokoju)
)
```

Tabela pokoje

```
CREATE TABLE pokoje (
  id integer IDENTITY(1,1) PRIMARY KEY,
  id_kategoria integer,
  id_typ_pokoju integer
)
```

Tabela kategorie_pokoju

```
CREATE TABLE kategorie_pokoju (
  id integer IDENTITY(1,1) PRIMARY KEY,
  nazwa nvarchar(9),
  czy_balkon BIT,
  czy_aneks BIT,
  czy_klimatyzacja BIT,
  czy_telewizor BIT,
  czy_wanna BIT,
  cena money CHECK (cena >= 0)
)
```

Tabela typ pokoju

```
CREATE TABLE typ_pokoju (
  id integer IDENTITY(1,1) PRIMARY KEY,
```

```
  ile_osob nvarchar(20),
  cena money CHECK (cena > 0)
)
```

Tabela system_log

```
CREATE TABLE [dbo].[system_log](
    [log_id] [int] IDENTITY(1,1) NOT NULL,
    [log_data] [datetime] NOT NULL,
    [typ] [nvarchar](50) NOT NULL,
    [tabela] [nvarchar](20) NOT NULL,
    [opis] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
    [log_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Związki

```
ALTER TABLE rezerwacje ADD FOREIGN KEY (id_klienta) REFERENCES klienci (id)
```

```
ALTER TABLE rezerwacje_pokoi ADD FOREIGN KEY (id_rezerwacji) REFERENCES rezerwacje (id)
```

```
ALTER TABLE rezerwacje_pokoi ADD FOREIGN KEY (id_pokoju) REFERENCES pokoje (id)
```

```
ALTER TABLE pokoje ADD FOREIGN KEY (id_kategoria) REFERENCES kategorie_pokoju (id)
```

```
ALTER TABLE rezerwacje ADD FOREIGN KEY (id_status) REFERENCES statusy (id)
```

```
ALTER TABLE wyzywienie ADD FOREIGN KEY (id_rezerwacji) REFERENCES rezerwacje (id)
```

```
ALTER TABLE wyzywienie ADD FOREIGN KEY (id_typ_wyzywienia) REFERENCES typ_wyzywienia (id)
```

```
ALTER TABLE uslugi ADD FOREIGN KEY (id_typ_uslugi) REFERENCES typ_uslugi (id)
```

```
ALTER TABLE uslugi ADD FOREIGN KEY (id_rezerwacji) REFERENCES rezerwacje (id)
```

```
ALTER TABLE pokoje ADD FOREIGN KEY (id_typ_pokoju) REFERENCES typ_pokoju (id)
```

Widoki

(dla każdego widoku należy wkleić kod polecenia definiującego widok wraz z komentarzem)

1. wyświetlanie specyfikacji pokoju

```
CREATE VIEW vw_specyfikacja_pokoju AS
SELECT p.id, k.nazwa, k.czy_balkon, k.czy_aneks, k.czy_klimatyzacja, k.czy_telewizor, k.czy_wanna, tp.ile_osob, (tp.cena + kp.cena) AS kwota
FROM pokoje as p
INNER JOIN kategorie_pokoju as k on p.id_kategoria = k.id
INNER JOIN typ_pokoju as tp on p.id_typ_pokoju = tp.id
INNER JOIN kategorie_pokoju as kp on p.id_kategoria = kp.id
```

	id	nazwa	czy_balkon	czy_aneks	czy_klimatyzacja	czy_telewizor	czy_wanna	ile_osob	kwota
1	1	Economic	0	0	0	0	0	jednoosobowe	150,00
2	2	Economic	0	0	0	0	0	dwuosobowe	220,00
3	3	Economic	0	0	0	0	0	trzyosobowy	300,00
4	4	Economic	0	0	0	0	0	czteroosobowy	420,00
5	5	Standard	1	0	0	1	0	jednoosobowe	200,00
6	6	Standard	1	0	0	1	0	dwuosobowe	270,00
7	7	Standard	1	0	0	1	0	trzyosobowy	350,00
8	8	Standard	1	0	0	1	0	czteroosobowy	470,00
9	9	Premium	1	1	1	1	0	jednoosobowe	260,00
10	10	Premium	1	1	1	1	0	dwuosobowe	330,00
11	11	Premium	1	1	1	1	0	trzyosobowy	410,00
12	12	Premium	1	1	1	1	0	czteroosobowy	530,00
13	13	Exclusive	1	1	1	1	1	jednoosobowe	310,00
14	14	Exclusive	1	1	1	1	1	dwuosobowe	380,00
15	15	Exclusive	1	1	1	1	1	trzyosobowy	460,00
16	16	Exclusive	1	1	1	1	1	czteroosobowy	580,00

2. wyświetlanie informacji o rezerwacji

```
CREATE VIEW vw_rezerwacja AS
SELECT r.id AS id_rezerwacji, r.id_klienta, r.data_zameldowania, r.data_wymeldowania, r.data_rezerwacji, r.id_status, r.rabat, k.imie, k.nazwisko, s.nazwa as status,
rp.id_pokoju, ABS((COALESCE(SUM(u.cena_uslug), 0) + COALESCE(SUM(rp.cena_pokojow), 0) + COALESCE(SUM(w.cena_wyzywienia), 0)) * DATEDIFF(day,r.data_wymeldowania,
```

```
r.data_zameldowania)) AS kwota
FROM rezerwacje AS r
LEFT JOIN uslugi AS u ON r.id = u.id_rezerwacji
LEFT JOIN rezerwacje_pokoi AS rp ON r.id = rp.id_rezerwacji
LEFT JOIN wyzywienie AS w ON r.id = w.id_rezerwacji
INNER JOIN klienci AS k ON r.id_klienta = k.id
INNER JOIN statusy AS s ON r.id_status = s.id
GROUP BY r.id, r.id_klienta, r.data_zameldowania, r.data_wymeldowania, r.data_rezerwacji, r.id_status, r.rabat, k.imie, k.nazwisko, s.nazwa, rp.id_pokoju;
```

	id_rezerwacji	id_klienta	data_zameldowania	data_wymeldowania	data_rezerwacji	id_status	rabat	imie	nazwisko	status	id_pokoju	kwota
1	1	1	2023-11-11	2023-11-15	2023-10-10	3	0	Albert	Nowak	oplacone	1	660.00
2	1	1	2023-11-11	2023-11-15	2023-10-10	3	0	Albert	Nowak	oplacone	5	860.00
3	2	2	2023-11-12	2023-11-15	2023-11-05	3	5	Bozydar	Kowal	oplacone	2	855.00
4	2	2	2023-11-12	2023-11-15	2023-11-05	3	5	Bozydar	Kowal	oplacone	6	1035.00
5	3	3	2023-11-15	2023-11-20	2023-10-30	3	0	Monika	Orzel	oplacone	3	2250.00
6	3	3	2023-11-15	2023-11-20	2023-10-30	3	0	Monika	Orzel	oplacone	7	2750.00
7	4	4	2023-12-15	2023-12-18	2023-12-14	1	0	Bartosz	Zelek	anulowane	4	1830.00
8	4	4	2023-12-15	2023-12-18	2023-12-14	1	0	Bartosz	Zelek	anulowane	8	2160.00
9	5	5	2023-12-16	2023-12-18	2023-10-12	3	0	Julia	Pajor	oplacone	9	880.00
10	5	5	2023-12-16	2023-12-18	2023-10-12	3	0	Julia	Pajor	oplacone	13	1200.00
11	6	6	2024-01-06	2024-01-07	2023-12-28	3	0	Stefania	Filipek	oplacone	10	520.00
12	6	6	2024-01-06	2024-01-07	2023-12-28	3	0	Stefania	Filipek	oplacone	14	660.00
13	7	7	2024-01-07	2024-01-10	2024-01-02	3	0	Andrzej	Jedrzejek	oplacone	11	1830.00
14	7	7	2024-01-07	2024-01-10	2024-01-02	3	0	Andrzej	Jedrzejek	oplacone	15	2340.00
15	8	8	2024-01-07	2024-01-11	2024-01-03	1	3	Bartlomiej	Matras	anulowane	12	3400.00
16	8	8	2024-01-07	2024-01-11	2024-01-03	1	3	Bartlomiej	Matras	anulowane	16	4080.00
17	9	9	2024-01-12	2024-01-18	2024-01-06	3	0	Ewa	Bukowiec	oplacone	1	900.00
18	9	9	2024-01-12	2024-01-18	2024-01-06	3	0	Ewa	Bukowiec	oplacone	6	1980.00
19	10	10	2024-02-01	2024-02-05	2024-01-26	1	0	Marta	Szyszka	anulowane	2	1080.00

---

3. Zestawienie wybranych opcji dodatkowych

```
create view [dbo].[vw_zestawienie_dodatkowych_opcji] as
select u.opis as 'dodatkowe opcje', u.cena
from typ_uslugi u
union
    select 'wyzywienie: ' + w.opis, w.cena
    from typ_wyzywienia w
GO
```

```
1   select * from vw_zestawienie_dodatkowych_opcji
2
```

Results Messages

	dodatkowe opcje	cena
1	rower	20,00
2	sauna	30,00
3	wyzywienie: all inclusive	80,00
4	wyzywienie: obiadokolacja + sniadanie	40,00
5	wyzywienie: sniadanie	15,00

Procedury/funkcje

(dla każdej procedury/funkcji należy wkleić kod polecenia definiującego procedurę wraz z komentarzem)

Procedury

Wyświetlanie dostępnych pokoi w kokretnym terminie

```
CREATE PROCEDURE p_dostępne_pokoje
@data_poczatkowa date, @data_koncowa date
AS
BEGIN

    if @data_poczatkowa < dateadd(day, 2, getdate())
        throw 50001, 'Rezerwacja musi byc wykonana conajmniej 48h przez zameldowaniem', 1;
    if @data_koncowa < @data_poczatkowa
        throw 50001, 'Błędny przedział', 1;

    SELECT id as numer_pokoju, nazwa, czy_balkon, czy_aneks, czy_klimatyzacja, czy_telewizor, czy_wanna, ile_osob, kwota
    FROM vw_specyfikacja_pokoju
    WHERE id IN (
        SELECT DISTINCT id_pokoju
        FROM vw_rezerwacja
        WHERE id_pokoju NOT IN (
            SELECT id_pokoju
            FROM vw_rezerwacja
            WHERE ((data_zameldowania < @data_koncowa AND data_wymeldowania > @data_poczatkowa)
            OR (data_zameldowania >= @data_poczatkowa AND data_zameldowania < @data_koncowa)
            OR (data_wymeldowania > @data_poczatkowa AND data_wymeldowania <= @data_koncowa)
            OR (data_zameldowania <= @data_poczatkowa AND data_wymeldowania >= @data_koncowa))
            AND status != 'anulowane'
        )
    );
END;
```

Opis: Procedura p_dostępne_pokoje służy do wyszukiwania dostępnych pokoi w określonym przedziale czasowym, sprawdzając przy tym czy podana data początkowa jest większa o 2 dni od aktualnej daty ponieważ datę rezerwacji i datę zameldowania musi dzielić 48 godzin.

Przykład 1

```
31  exec p_dostępne_pokoje '2023-12-15', '2023-12-18';
32
```

Messages

```
10:12:51 PM   Started executing query at Line 31
              Msg 50001, Level 16, State 1, Procedure p_dostępne_pokoje, Line 7
              Rezerwacja musi byc wykonana conajmniej 48h przez zameldowaniem
              Total execution time: 00:00:00.152
```

Przykład 2

```
33
34  exec p_dostępne_pokoje '2024-06-29', '2024-06-10';
35
36
```

Messages

```
10:13:38 PM   Started executing query at Line 34
              Msg 50001, Level 16, State 1, Procedure p_dostępne_pokoje, Line 9
              Błędny przedział
              Total execution time: 00:00:00.181
```

Przykład 3

```
37  exec p_dostępne_pokoje '2024-06-29', '2024-06-29';
```

Results Messages

	numer_pokoju	nazwa	czy_balkon	czy_aneks	czy_klimatyzacja	czy_telewizor	czy_wanna	ile_osob	kwota
1	1	Economic	0	0	0	0	0	jednoosobowe	150,00
2	2	Economic	0	0	0	0	0	dwuosobowe	220,00
3	3	Economic	0	0	0	0	0	trzyosobowy	300,00
4	4	Economic	0	0	0	0	0	czteroosobowy	420,00
5	5	Standard	1	0	0	1	0	jednoosobowe	200,00
6	6	Standard	1	0	0	1	0	dwuosobowe	270,00
7	7	Standard	1	0	0	1	0	trzyosobowy	350,00
8	8	Standard	1	0	0	1	0	czteroosobowy	470,00
9	9	Premium	1	1	1	1	0	jednoosobowe	260,00
10	10	Premium	1	1	1	1	0	dwuosobowe	330,00
11	11	Premium	1	1	1	1	0	trzyosobowy	410,00
12	12	Premium	1	1	1	1	0	czteroosobowy	530,00
13	13	Exclusive	1	1	1	1	1	jednoosobowe	310,00
14	14	Exclusive	1	1	1	1	1	dwuosobowe	380,00
15	15	Exclusive	1	1	1	1	1	trzyosobowy	460,00
16	16	Exclusive	1	1	1	1	1	czteroosobowy	580,00

Dodawanie rezerwacji

```
CREATE   PROCEDURE [dbo].[p_dodanie_rezerwacji]
@id_klienta int, @data_zameldowania date, @data_wymeldowania date, @id_status int, @rabat int, @idWyzywienia dbo.idWyzywienia READONLY, @idUslugi dbo.idUslugi READONLY,
@idPokoju dbo.idPokoju READONLY
as
begin
    if @data_zameldowania >= @data_wymeldowania
        throw 50001, 'Data zameldowania musi być wcześniejsza niż wymeldowania', 1;

    if @data_zameldowania < dateadd(day, 2, getdate())
        throw 50001, 'Rezerwacja musi byc wykonana conajmniej 48h przez zameldowaniem', 1;

    if ABS(DATEDIFF(day, @data_zameldowania, @data_wymeldowania))>14
        throw 50001, 'Okres rezerwacji nie może być dłuższy niż 14 dni', 1;

    if NOT EXISTS
    (SELECT ID
    FROM @idPokoju
    WHERE EXISTS
        (
        SELECT id_pokoju
            FROM vw_rezerwacja
            WHERE id_pokoju NOT IN (
                SELECT id_pokoju
                FROM vw_rezerwacja
                WHERE ((data_zameldowania < @data_wymeldowania AND data_wymeldowania > @data_zameldowania)
                OR (data_zameldowania >= @data_zameldowania AND data_zameldowania < @data_wymeldowania)
                OR (data_wymeldowania > @data_zameldowania AND data_wymeldowania <= @data_wymeldowania)
                OR (data_zameldowania <= @data_zameldowania AND data_wymeldowania >= @data_wymeldowania))
                AND status != 'anulowane'
            )
        )
    )
    throw 50001, 'Co najmniej jeden z pokoi jest już zarezerwowanych w tym okresie', 1;

    if ABS(DATEDIFF(day, @data_zameldowania, @data_wymeldowania))>7
    SET @rabat = @rabat + 10

    INSERT INTO rezerwacje
    VALUES(@id_klienta, @data_zameldowania, @data_wymeldowania, GETDATE(), @id_status, @rabat);

    declare @id_rezerwacji int
    SET @id_rezerwacji = @@IDENTITY;

    IF EXISTS (SELECT ID FROM @idWyzywienia)
    BEGIN
        INSERT INTO wyzywienie (id_rezerwacji, id_typ_wyzywienia, cena_wyzywienia)
        SELECT @id_rezerwacji, w.ID, tw.cena
        FROM @idWyzywienia w
        JOIN typ_wyzywienia tw ON w.ID = tw.ID;
    END;
```

```
    IF EXISTS (SELECT ID FROM @idUslugi)
    BEGIN
        INSERT INTO uslugi (id_typ_uslugi, id_rezerwacji, cena_uslug)
        SELECT u.ID, @id_rezerwacji, tu.cena
        FROM @idUslugi u
        JOIN typ_uslugi tu ON u.ID = tu.ID;
    END;

    IF EXISTS (SELECT ID FROM @idPokoju)
    BEGIN
        INSERT INTO rezerwacje_pokoi (id_rezerwacji, id_pokoju, cena_pokojow)
        SELECT @id_rezerwacji, p.ID, SUM(tp.cena + kp.cena) AS cena_pokojow
        FROM @idPokoju p
        JOIN typ_pokoju tp ON p.ID = tp.ID
        JOIN kategorie_pokoju kp ON tp.id = kp.ID
        GROUP BY p.ID;
    END;
end;
```

Opis: Procedura dodaje rezerwacje do tabeli rezerwacje

```
DECLARE @idPokoju dbo.idPokoju;
INSERT INTO @idPokoju VALUES (1), (2);

DECLARE @idUslugi dbo.idUslugi;
INSERT INTO @idUslugi VALUES (1), (2);

DECLARE @idWyzywienia dbo.idWyzywienia;
INSERT INTO @idWyzywienia VALUES (1), (2);

exec p_dodanie_rezerwacji 5, '2024-07-05', '2024-07-12', 3, 5, @idWyzywienia, @idUslugi, @idPokoju
```

121 %

Results Messages

	id	id_klienta	data_zameldowania	data_wymeldowania	data_rezerwacji	id_status	rabat
1	38	5	2024-07-05	2024-07-12	2024-06-28	3	5

Rezerwacja o podanym statusie

```
create or alter procedure p_rezerwacja_o_podanym_statusie
@status_rezerwacji char(11)
as
begin
    if not exists (select * from statusy where nazwa = @status_rezerwacji)
        throw 50001, 'Nie ma takiego statusu', 1;

    select *
    from vw_rezerwacja
    where status = @status_rezerwacji
end;
```

Opis: Procedura ta przy wykorzystaniu widoku vw_rezerwacja wyswietla informacje o rezerwacjach posiadajacych wybrany status.

```
1
2
3  exec p_rezerwacja_o_podanym_statusie 'anulowane'
4
5
```

Results Messages

	id_rezerwacji	id_klienta	data_zameldowania	data_wymeldowania	data_rezerwacji	id_status	rabat	imie	nazwisko	status	id_pokoju	kwota
1	5	5	2023-12-15	2023-12-18	2023-12-14	1	0	Julia	Pajor	anulowane	5	810,00
2	8	8	2024-01-07	2024-01-11	2024-01-03	1	3	Bartłomiej	Matras	anulowane	8	1880,00
3	10	10	2024-02-01	2024-02-05	2024-01-26	1	0	Marta	Szyszka	anulowane	10	1640,00

```
1
2
3  exec p_rezerwacja_o_podanym_statusie 'niekompletne'
4
5
```

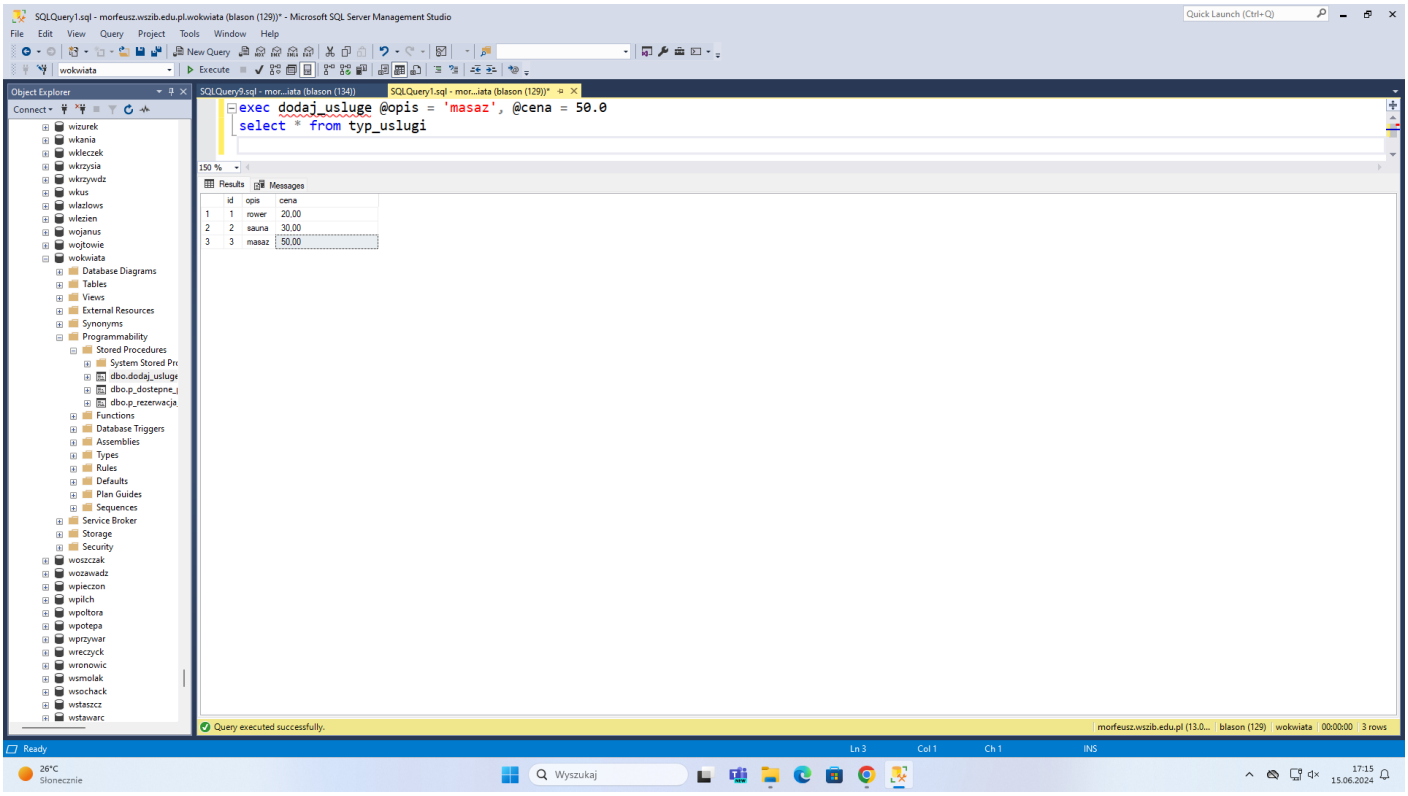
Messages

22:11:35 Started executing query at Line 3
Msg 50001, Level 16, State 1, Procedure p_rezerwacja_o_podanym_statusie, Line 6
Nie ma takiego statusu
Total execution time: 00:00:00.146

Dodawanie usługi

```
create or alter procedure [dbo].[p_dodaj_usluge]
@opis nvarchar(5),
@cena money
as
begin
insert into typ_uslugi (opis, cena)
values (@opis, @cena);
end
```

Opis: Procedura p_dodaj_usluge służy do dodawania nowej usługi do tabeli typ_uslugi. Przyjmuje dwa parametry: opis usługi oraz jej cenę.



Anulowanie zamówienia

```
CREATE PROCEDURE [dbo].[p_anuluj_zamowienie] @id_zamowienia INT
AS
BEGIN
  DECLARE @status_anulowane INT;
  DECLARE @data_zameldowania DATE;

  SELECT @status_anulowane = id
  FROM statusy
  WHERE nazwa = 'anulowane';

  IF EXISTS (SELECT * FROM rezerwacje WHERE id = @id_zamowienia)
  BEGIN
    SELECT @data_zameldowania = data_zameldowania
    FROM rezerwacje
    WHERE id = @id_zamowienia;

    IF DATEDIFF(DAY, GETDATE(), @data_zameldowania) >= 1
    BEGIN
      UPDATE rezerwacje
      SET id_status = @status_anulowane
      WHERE id = @id_zamowienia;
      PRINT 'Status zamówienia został zmieniony na anulowane.';
    END
    ELSE
    BEGIN
      PRINT 'Zamówienie może być anulowane maksymalnie 1 dzień przed datą zameldowania.';
    END
  END
  ELSE
  BEGIN
    PRINT 'Zamówienie z podanym id nie istnieje.';
  END
END;
```

Opis: Procedura p_anuluj_zamowienie jest przeznaczona do anulowania zamówienia w systemie. Jeśli zamówienie o podanym id nie istnieje zwracana jest informacja o jego braku. W przypadku gdy próba anulowania rezerwacji jest na jeden dzień przed datą zameldowania, próba zostanie odrzucona.

Anulowanie zamówienia

```
exec p_anuluj_zamowienie 29
select * from rezerwacje where id = 29
```

133 %

Results Messages

	id	id_klienta	data_zameldowania	data_wymeldowania	data_rezerwacji	id_status	rabat
1	29	1	2024-07-01	2024-07-12	2024-06-25	1	10

Próba anulowania zamówienia o błędnym id

```
exec p_anuluj_zamowienie 22
```

146 %

Messages

```
Zamówienie z podanym id nie istnieje.
```

Próba anulowania zamówienia z zbyt bliską datą zamelodania

	id	id_klienta	data_zameldowania	data_wymeldowania	data_rezerwacji	id_status	rabat
1	20	20	2024-05-20	2024-05-30	2023-05-17	2	0

```
exec p_anuluj_zamowienie 20
```

133 %

Messages

```
Zamówienie może być anulowane maksymalnie 1 dzień przed datą zameldowania.
```

Sprawdzenie kiedy dany pokoj bedzie wolny

```
CREATE PROCEDURE p_zwolniony_pokoj
    @room_id INT
AS
BEGIN
    SELECT r.id AS rezerwacja_id, r.data_wymeldowania, r.id_klienta
    FROM rezerwacje_pokoi rp
    JOIN rezerwacje r ON rp.id_rezerwacji = r.id
    WHERE rp.id_pokoju = @room_id
    ORDER BY r.data_wymeldowania DESC;
END;
```

Opis: Procedura sprawdza kiedy dany pokój zostanie zwolniony.

```
EXEC dbo.p_zwolniony_pokoj @room_id = 2;
```

99 %

Results Messages

	rezerwacja_id	data_wymeldowania	id_klienta
1	18	2024-05-08	18
2	2	2023-12-18	2

Przedluzenie rezerwacji

```
CREATE OR ALTER PROCEDURE p_przedluzenie_rezerwacji
@id_rezerwacji INTEGER, @nowa_data_koncowa DATE
AS
BEGIN
    DECLARE @data_poczatkowa DATE = (SELECT DISTINCT data_zameldowania FROM vw_rezerwacja WHERE id_rezerwacji = @id_rezerwacji);
    DECLARE @data_koncowa DATE = (SELECT DISTINCT data_wymeldowania FROM vw_rezerwacja WHERE id_rezerwacji = @id_rezerwacji);

    IF @nowa_data_koncowa <= @data_koncowa
        THROW 50001, 'Nowa data wymeldowania nie może być wcześniejsza lub równa aktualnej', 1;

    IF (SELECT id_status FROM rezerwacje WHERE id = @id_rezerwacji) = 1
        THROW 50001, 'Nie można przedłużyć anulowanej rezerwacji', 1;

    IF GETDATE() > @data_koncowa
        THROW 50001, 'Nie można przedłużyć rezerwacji, która uległa już wygaśnięciu', 1;

    IF ABS(DATEDIFF(day, @data_poczatkowa, @nowa_data_koncowa)) > 14
        THROW 50001, 'Okres rezerwacji nie może być dłuższy niż 14 dni', 1;

    IF EXISTS (
        SELECT id_pokoju
        FROM vw_rezerwacja
        WHERE id_pokoju IN (
            SELECT id_pokoju
            FROM vw_rezerwacja
            WHERE id_rezerwacji = @id_rezerwacji
        )
        AND ((data_zameldowania < @nowa_data_koncowa AND data_wymeldowania > @data_poczatkowa)
```

```
            OR (data_zameldowania >= @data_poczatkowa AND data_zameldowania < @nowa_data_koncowa)
            OR (data_wymeldowania > @data_poczatkowa AND data_wymeldowania <= @nowa_data_koncowa)
            OR (data_zameldowania <= @data_poczatkowa AND data_wymeldowania >= @nowa_data_koncowa))
        AND status != 'anulowane'
        AND id_rezerwacji != @id_rezerwacji
    )
        THROW 50001, 'Rezerwacja danego pokoju nie może zostać przedłużona. Ktoś inny zdążył już zarezerwować ten pokój w wybranym terminie', 1;
    ELSE
        UPDATE rezerwacje
        SET data_wymeldowania = @nowa_data_koncowa
        WHERE id = @id_rezerwacji;
END;
GO
```

Opis: Procedura p_przedluzenie_rezerwacji jest zaprojektowana w celu przedłużenia istniejącej rezerwacji w bazie danych. Procedura sprawdza różne warunki przed aktualizacją daty wymeldowania, aby zapewnić integralność danych i zapobiec konfliktom rezerwacji.

Przykład 1

```
select * from vw_rezerwacja where id_rezerwacji = 41

exec p_przedluzenie_rezerwacji 41, '2024-07-10'
```

Results Messages

	id_rezerwacji	id_klienta	data_zameldowania	data_wymeldowania	data_rezerwacji	id_status	rabat	imie	nazwisko	status	id_pokoju	kwota
1	41	20	2024-07-02	2024-07-06	2024-06-28	3	5	Agata	Wojcieszak	opłacone	3	1640,00
2	41	20	2024-07-02	2024-07-06	2024-06-28	3	5	Agata	Wojcieszak	opłacone	4	2320,00

```
select * from vw_rezerwacja where id_rezerwacji = 41

exec p_przedluzenie_rezerwacji 41, '2024-07-07'
```

essages

```
8:59:05 PM    Started executing query at Line 89
              Msg 512, Level 16, State 1, Procedure p_przedluzenie_rezerwacji, Line 11
              Subquery returned more than 1 value. This is not permitted when the subquery follows =, !=, <, <= , >, >= or when the subquery is used as an expression.
              (1 row affected)
              Total execution time: 00:00:00.244
```

```
select * from vw_rezerwacja where id_rezerwacji = 41

exec p_przedluzenie_rezerwacji 41, '2024-07-07'
```

Results Messages

	id_rezerwacji	id_klienta	data_zameldowania	data_wymeldowania	data_rezerwacji	id_status	rabat	imie	nazwisko	status	id_pokoju	kwota
1	41	20	2024-07-02	2024-07-07	2024-06-28	3	5	Agata	Wojcieszak	opłacone	3	2050,00
2	41	20	2024-07-02	2024-07-07	2024-06-28	3	5	Agata	Wojcieszak	opłacone	4	2900,00

Przykład 2

```
select * from vw_rezerwacja where id_rezerwacji = 41

exec p_przedluzenie_rezerwacji 41, '2024-07-06'
```

Messages

```
9:00:39 PM    Started executing query at Line 88
              Msg 50001, Level 16, State 1, Procedure p_przedluzenie_rezerwacji, Line 9
              Nowa data wymeldowania nie może być wcześniejsza lub równa aktualnej
              Total execution time: 00:00:00.339
```

Przykład 3

```
select * from vw_rezerwacja where id_rezerwacji = 41

select * from rezerwacje

exec p_anuluj_zamowienie 41

exec p_przedluzenie_rezerwacji 41, '2024-07-10'
```

Results Messages

	id_rezerwacji	id_klienta	data_zameldowania	data_wymeldowania	data_rezerwacji	id_status	rabat	imie	nazwisko	status	id_pokoju	kwota
	41	20	2024-07-02	2024-07-09	2024-06-28	1	5	Agata	Wojcieszak	anulowane	3	2870,00
	41	20	2024-07-02	2024-07-09	2024-06-28	1	5	Agata	Wojcieszak	anulowane	4	4060,00

```
select * from vw_rezerwacja where id_rezerwacji = 41

select * from rezerwacje

exec p_anuluj_zamowienie 41

exec p_przedluzenie_rezerwacji 41, '2024-07-10'
```

Messages

```
9:04:52 PM    Started executing query at Line 91
              Msg 50001, Level 16, State 1, Procedure p_przedluzenie_rezerwacji, Line 12
              Nie można przedłużyć anulowanej rezerwacji
              Total execution time: 00:00:00.177
```

Przykład 4

```
select * from vw_rezerwacja where id_rezerwacji = 18

select * from rezerwacje

exec p_anuluj_zamowienie 41

exec p_przedluzenie_rezerwacji 18, '2024-07-09'
```

Results Messages

id_rezerwacji	id_klienta	data_zameldowania	data_wymeldowania	data_rezerwacji	id_status	rabat	imie	nazwisko	status	id_pokoju	kwota
18	18	2024-05-05	2024-05-08	2024-05-01	3	0	Marcin	Franczak	opłacone	2	720,00

```
select * from vw_rezerwacja where id_rezerwacji = 18

select * from rezerwacje

exec p_anuluj_zamowienie 41

exec p_przedluzenie_rezerwacji 18, '2024-07-09'
```

Messages

```
9:06:30 PM  Started executing query at Line 92
            Msg 50001, Level 16, State 1, Procedure p_przedluzenie_rezerwacji, Line 15
            Nie można przedłużyć rezerwacji, która uległa już wygaśnięciu
            Total execution time: 00:00:00.184
```

Funkcje

Obliczanie całkowitego kosztu rezerwacji

```
CREATE FUNCTION [dbo].[f_calkowity_koszt] (@id_rezerwacji INT)
RETURNS MONEY
AS BEGIN
DECLARE @ckoszt MONEY;
DECLARE @rabat DECIMAL;

--------------------------------------------------------------------------------
DECLARE @liczba_dni INT;
 SELECT @liczba_dni = DATEDIFF(DAY, r.data_zameldowania, r.data_wymeldowania),@rabat = r.rabat
    FROM rezerwacje r
    WHERE r.id = @id_rezerwacji;
--------------------------------------------------------------------------------

 SELECT @ckoszt = ISNULL(SUM(u.cena_uslug), 0) + ISNULL(SUM(rp.cena_pokojow*@liczba_dni), 0) + ISNULL(SUM(w.cena_wyzywienia*@liczba_dni), 0)
    FROM rezerwacje r
    LEFT JOIN uslugi u ON r.id = u.id_rezerwacji
    LEFT JOIN rezerwacje_pokoi rp ON r.id = rp.id_rezerwacji
    LEFT JOIN wyzywienie w ON r.id = w.id_rezerwacji
    WHERE r.id = @id_rezerwacji;

        SET @ckoszt = @ckoszt * (1 - @rabat / 100.0);

    RETURN @ckoszt;
    END;
-----------------/*Wyświetlenie*/-----------------
GO
select dbo.calkowity_koszt(20) as calkowity_koszt /*( ) - id_rezerwacji*/
GO
```

Opis: Funkcja sumuje koszty usług, pokoi oraz wyżywienia z rezerwacji, używa do tego JOIN aby połączyć tabele rezerwacji, usługi, rezerwacje_pokoi i wyżywienia. Sumuje koluny z kosztami, jeżeli któraś z kolumn nie zawiera danych wstawia 0.

Przykład 1

```
GO
select dbo.f_calkowity_koszt(10) as calkowity_koszt
GO
```

100 % Results Messages

	calkowity_koszt
1	1640,00

Przykład 2

```
GO
select dbo.f_calkowity_koszt(1) as calkowity_koszt
GO
```

100 % Results Messages

	calkowity_koszt
1	660,00

Triggery

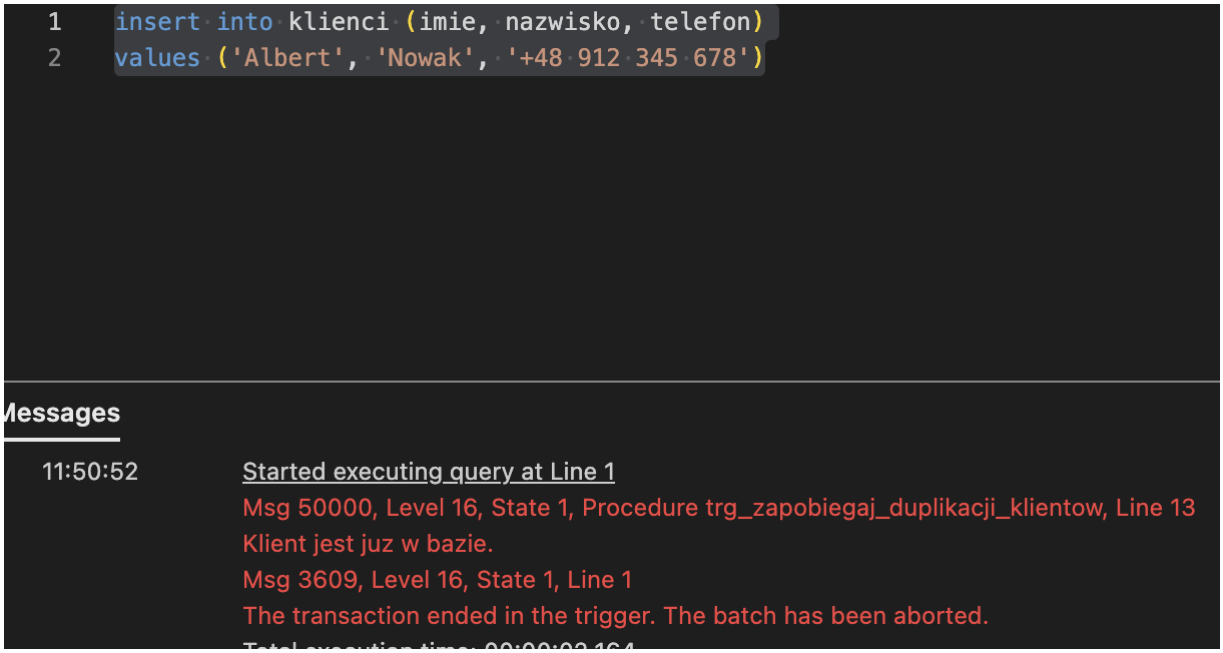
(dla każdego triggera należy wkleić kod polecenia definiującego trigger wraz z komentarzem)

trg_zapobiegaj_duplikacji_klientow

```
CREATE TRIGGER trg_zapobiegaj_duplikacji_klientow
ON klienci
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @noweImie NVARCHAR(12), @noweNazwisko NVARCHAR(15), @nowyTelefon NVARCHAR(15)
    SELECT @noweImie = i.imie, @noweNazwisko = i.nazwisko, @nowyTelefon = i.telefon
    FROM inserted i

    IF EXISTS (SELECT 1 FROM klienci
               WHERE imie = @noweImie AND nazwisko = @noweNazwisko AND telefon = @nowyTelefon)
    BEGIN
        RAISERROR ('Klient jest juz w bazie.', 16, 1)
        ROLLBACK TRANSACTION
    END
    ELSE
    BEGIN
        INSERT INTO klienci (imie, nazwisko, telefon)
        SELECT imie, nazwisko, telefon
        FROM inserted
    END
END
GO
```

Opis: Trigger ten zapewnia integralność danych w tabeli klienci, eliminując możliwość występowania zduplikowanych rekordów klientów.



---

trg_zapobiegaj_duplikacji_rezerwacje

```
CREATE   TRIGGER [dbo].[trg_zapobiegaj_duplikacji_rezerwacje]
ON [dbo].[rezerwacje]
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @nowy_id_klienta INT,
            @nowa_data_zameldowania DATE,
            @nowa_data_wymeldowania DATE,
            @nowa_data_rezerwacji DATE,
            @nowy_id_status INT,
            @nowy_rabat NUMERIC(18, 0);

    SELECT @nowy_id_klienta = i.id_klienta,
           @nowa_data_zameldowania = i.data_zameldowania,
           @nowa_data_wymeldowania = i.data_wymeldowania,
           @nowa_data_rezerwacji = i.data_rezerwacji,
           @nowy_id_status = i.id_status,
           @nowy_rabat = i.rabat
    FROM inserted i;

    IF EXISTS (SELECT 1
               FROM rezerwacje
               WHERE id_klienta = @nowy_id_klienta
                 AND data_zameldowania = @nowa_data_zameldowania
                 AND data_wymeldowania = @nowa_data_wymeldowania
                 AND data_rezerwacji = @nowa_data_rezerwacji
                 AND id_status = @nowy_id_status
                 AND rabat = @nowy_rabat)
    BEGIN
        RAISERROR('Rezerwacja już jest w bazie', 16, 1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        INSERT INTO rezerwacje (id_klienta, data_zameldowania, data_wymeldowania, data_rezerwacji, id_status, rabat)
        SELECT id_klienta, data_zameldowania, data_wymeldowania, data_rezerwacji, id_status, rabat
        FROM inserted;
    END
```

```
END;
```

Opis: Trigger zapobiega wstawianiu zduplikowanych rekordów do tabeli rezerwacje. W momencie próby dodania nowej rezerwacji, trigger ten sprawdza, czy rezerwacja o tych samych danych już istnieje w bazie danych. Jeśli taka rezerwacja już istnieje, operacja jest przerywana, a użytkownik otrzymuje komunikat o błędzie.

```
INSERT INTO rezerwacje (id_klienta, data_zameldowania, data_wymeldowania, data_rezerwacji, id_status, rabat)
VALUES (13, '2024-03-02', '2024-03-05', '2024-02-24', 3, 0);
```

```
133 %
Messages
Msg 50000, Level 16, State 1, Procedure trg_zapobiegaj_duplikacji_rezerwacje, Line 30 [Batch Start Line 13]
Rezerwacja już jest w bazie
Msg 3609, Level 16, State 1, Line 14
The transaction ended in the trigger. The batch has been aborted.
```

---

trg_zapobiegaj_duplikacji_typ_uslugi

```
CREATE   TRIGGER [dbo].[trg_zapobiegaj_duplikacji_typ_uslugi]
ON [dbo].[typ_uslugi]
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @nowy_opis NVARCHAR(5),
            @nowa_cena MONEY;
    SELECT @nowy_opis = i.opis,
           @nowa_cena = i.cena
    FROM inserted i;
    IF EXISTS (SELECT 1
               FROM typ_uslugi
               WHERE opis = @nowy_opis
                 AND cena = @nowa_cena)
    BEGIN
        RAISERROR('Typ usługi już jest w bazie', 16, 1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        INSERT INTO typ_uslugi (opis, cena)
        SELECT opis, cena
        FROM inserted;
    END
END;
```

Opis: Trigger zapobiega wstawianiu zduplikowanych rekordów do tabeli typ_uslugi. W momencie próby dodania nowego typu usługi, trigger ten sprawdza, czy usługa o tym samym opisie i cenie już istnieje w bazie danych. Jeśli taka usługa już istnieje, operacja jest przerywana, a użytkownik otrzymuje komunikat o błędzie.

```
INSERT INTO typ_uslugi (opis, cena)
VALUES ('rower', 20.0000);
```

```
3 %
Messages
Msg 50000, Level 16, State 1, Procedure trg_zapobiegaj_duplikacji_typ_uslugi, Line 16 [Batch Start Line 15]
Typ usługi już jest w bazie
Msg 3609, Level 16, State 1, Line 16
The transaction ended in the trigger. The batch has been aborted.
```

---

trg_zapobiegaj_duplikacji_rezerwacje_pokoi

```
CREATE   TRIGGER [dbo].[trg_zapobiegaj_duplikacji_rezerwacje_pokoi]
ON [dbo].[rezerwacje_pokoi]
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @nowy_id_rezerwacji INT,
            @nowy_id_pokoju INT;

    SELECT @nowy_id_rezerwacji = i.id_rezerwacji,
           @nowy_id_pokoju = i.id_pokoju
    FROM inserted i;

    IF EXISTS (SELECT 1
               FROM rezerwacje_pokoi
               WHERE id_rezerwacji = @nowy_id_rezerwacji
                 AND id_pokoju = @nowy_id_pokoju)
    BEGIN
        RAISERROR('Rezerwacja pokoji jest ju w bazie', 16, 1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        INSERT INTO rezerwacje_pokoi (id_rezerwacji, id_pokoju, cena_pokojow)
        SELECT id_rezerwacji, id_pokoju, cena_pokojow
        FROM inserted;
    END
END;
```

Opis: Trigger zapobiega wstawianiu zduplikowanych rekordów do tabeli rezerwacje_pokoi. W momencie próby dodania nowej rezerwacji pokoju, trigger ten sprawdza, czy już istnieje rekord powiązany z daną rezerwacją i pokojem. Jeśli taki rekord już istnieje, operacja jest przerywana, a użytkownik otrzymuje komunikat o błędzie.

```
INSERT INTO rezerwacje_pokoi (id_rezerwacji, id_pokoju, cena_pokojow)
VALUES (17, 1, 150);
```

```
Messages
Msg 50000, Level 16, State 1, Procedure trg_zapobiegaj_duplikacji_rezerwacje_pokoi, Line 18 [Batch Start Line 15]
Rezerwacja pokoji jest ju w bazie
Msg 3609, Level 16, State 1, Line 16
The transaction ended in the trigger. The batch has been aborted.
```

---

trigger do typ_pokoju

```
CREATE   TRIGGER [dbo].[trg_zapobiegaj_duplikacji_typ_pokoju]
ON [dbo].[typ_pokoju]
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @nowy_opis NVARCHAR(20),
            @nowa_cena MONEY;
    SELECT @nowy_opis = i.ile_osob,
           @nowa_cena = i.cena
    FROM inserted i;
    IF EXISTS (SELECT 1
               FROM typ_pokoju
               WHERE ile_osob = @nowy_opis
                 AND cena = @nowa_cena)
    BEGIN
        RAISERROR('Typ pokoju już jest w bazie', 16, 1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        INSERT INTO typ_pokoju(ile_osob, cena)
        SELECT ile_osob, cena
        FROM inserted;
    END
END;
```

Opis: Trigger zapobiega wstawianiu zduplikowanych rekordów do tabeli typ_pokoju. W momencie próby dodania nowego typu pokoju, trigger ten sprawdza, czy już istnieje rekord powiązany z danym typem pokoju. Jeśli taki rekord już istnieje, operacja jest przerywana, a użytkownik otrzymuje komunikat o błędzie.

```
insert into typ_pokoju(ile_osob, cena)
values ('jednoosobowe', 150.00)
```

```
Messages
Msg 50000, Level 16, State 1, Procedure trg_zapobiegaj_duplikacji_typ_pokoju, Line 16 [Batch Start Line 34]
Typ pokoju już jest w bazie
Msg 3609, Level 16, State 1, Line 35
The transaction ended in the trigger. The batch has been aborted.

Completion time: 2024-06-28T19:41:25.3341336+02:00
```

Trigger dla całej bazy

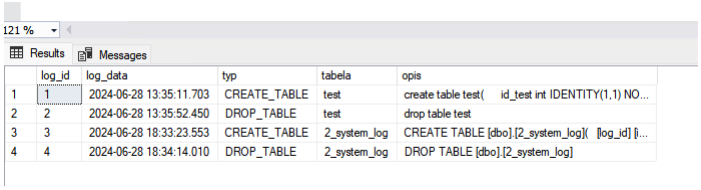
```
CREATE TRIGGER [trg_logi_bazy]
ON DATABASE
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE,
    CREATE_PROCEDURE, ALTER_PROCEDURE, DROP_PROCEDURE,
    CREATE_FUNCTION, ALTER_FUNCTION, DROP_FUNCTION
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @log_data XML;
    DECLARE @typ NVARCHAR(50);
    DECLARE @tabela NVARCHAR(20);
    DECLARE @opis NVARCHAR(MAX);

    SET @log_data = EVENTDATA();
    SET @typ = @log_data.value('(/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(50)');
    SET @tabela = @log_data.value('(/EVENT_INSTANCE/ObjectName)[1]', 'NVARCHAR(20)');
    SET @opis = @log_data.value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'NVARCHAR(MAX)');

    INSERT INTO system_log (typ, tabela, opis)
    VALUES (@typ, @tabela, @opis);
END;
GO
```

Opis: Trigger wpisuje logi bazy danych do tabeli system_log, którą trzeba było dodać na potrzeby tego triggera



121 %

Results Messages

	log_id	log_data	typ	tabela	opis
1	1	2024-06-28 13:35:11.703	CREATE_TABLE	test	create table test(id_test int IDENTITY(1,1) NO...
2	2	2024-06-28 13:35:52.450	DROP_TABLE	test	drop table test
3	3	2024-06-28 18:33:23.553	CREATE_TABLE	2_system_log	CREATE TABLE [dbo].[2_system_log]([log_id] [i...
4	4	2024-06-28 18:34:14.010	DROP_TABLE	2_system_log	DROP TABLE [dbo].[2_system_log]

Przykłady użycia

```
-- Klient chce zapoznać się z ofertą dostępnym pokoi w konkretnym terminie.

-- 1. po terminie
exec p_dostępne_pokoje '2023-11-11', '2023-11-15';

-- 2. bledny przedzial
exec p_dostępne_pokoje '2024-06-30', '2024-06-10';

-- 3. poprawnie
exec p_dostępne_pokoje '2024-07-29', '2024-07-29'

-- Klient chce zapoznać się z wyposażeniem wybranego pokoju.

-- 1. wszystkie pokoje
select * from vw_specyfikacja_pokoju

-- 2. konkretna kategoria pokoju
select * from vw_specyfikacja_pokoju where nazwa = 'Economic'

-- 3. konkretyn numer pokoju
select * from vw_specyfikacja_pokoju where id = 15

-- Klient chce zapoznać z dodatkowymi usługami hotelu jakie moze wybrac.

select * from vw_zestawienie_dodatkowych_opcji

-- Klient chce anulować rezerwacje

-- 1. Próba anulowania zamówienia o błędnym id
exec p_anuluj_zamowienie 22

-- 2. Próba anulowania zamówienia z zbyt bliską datą zamelodania
exec p_anuluj_zamowienie 20

-- 3. Anulowanie rezerwacji
exec p_anuluj_zamowienie 29

-- Klient chce wiedzieć ile wyniesie całkowity koszt pobytu
select dbo.f_calkowity_koszt(20) as calkowity_koszt   /*( ) - id_rezerwacji*/

-- Pracownik chce sprawdzić kiedy pokój zostanie zwolniony

EXEC dbo.p_zwolniony_pokoj @room_id = 2 ;

-- dodanie rezerewacji

DECLARE @idPokoju dbo.idPokoju;
INSERT INTO @idPokoju VALUES (1), (2);

DECLARE @idUslugi dbo.idUslugi;
INSERT INTO @idUslugi VALUES (1), (2);

DECLARE @idWyzywienia dbo.idWyzywienia;
INSERT INTO @idWyzywienia VALUES (1), (2);

exec p_dodanie_rezerwacji 5, '2023-05-05', '2023-05-12', 3, 5, @idWyzywienia, @idUslugi, @idPokoju

-- klient chce przedluzyc rezerwacje

select * from rezerwacje where id = 40
exec p_przedluzenie_rezerwacji 40, '2024-08-15'
```