

**BAZY DANYCH
LABORATORIUM**

SPRAWOZDANIE Z PROJEKTU ZALICZENIOWEGO

SPIS TREŚCI

Treść zadania.....	3
Analiza biznesowa.....	3
Model konceptualny bazy danych	5
Model fizyczny relacyjnej bazy danych.....	6
Reverse Engineer PDM.....	7
Wykaz tabel.....	8
Widoki.....	9
Procedury wyzwalane.....	15
Procedury wbudowane.....	18
Funkcje wbudowane.....	28
Użytkownicy.....	36
Wnioski.....	38

1. Treść zadania:

Celem projektu laboratoryjnego jest stworzenie bazy danych, na wybrany przez studenta temat w środowisku producentów zaawansowanych oraz w środowisku Open source. W projekcie ma się znajdować minimum:

- 10 tabel,
- 3 widoki (w tym jeden widok musi być zmaterializowany),
- 3 procedury wyzwalane (triggery),
- 3 procedury wbudowane (minimum 1 mechanizm kursora i transakcji),
- 3 funkcje wbudowane (minimum 1 mechanizm kursora i transakcji),
- 3 użytkowników o różnych uprawnieniach.

Model konceptualny i model fizyczny zostały utworzone w narzędziu PowerDesigner.

Środowiska bazy danych:

- Sybase Central
- MySQL

2. Analiza biznesowa:

Projekt bazy danych zawiera w sobie tabele z informacjami o pracownikach i ich dyżurach pracy, gościach, pokojach, rezerwacji.

Główną częścią bazy danych jest informacja o hotelu. Jest to projekt bazy danych hotelu znajdującego się w Gdyni przy ulicy Portowej 1. Hotel nazywa się „hotELIX” i posiada 4 gwiazdki. Jest to projekt małego hotelu.

Hotel zawiera tylko 8 pokoi o różnych standardach, takich jak np. liczba łóżek. Każdy pokój posiada swoje udogodnienie, czyli zawartość, takie jak balkon, widok na morze, łazienka a w niej wanna lub/oraz prysznic, barek telewizja, WiFi, klimatyzacja.

Do wyboru jest 5 typów udogodnienia. Oprócz tego, są dostępne ogólne, dodatkowo płatne usługi. Do wyboru jest 5 typów usług, zawierających: restaurację, basen, SPA, saunę siłownię, transfer lotniskowy – z i do lotniska (Gdynia-Port lotniczy w Gdańsku) oraz miejsce parkingowe.

Baza danych hotelu zawiera 6 pracowników. Baza danych zawiera informacje o pracownikach takie jak: imię, nazwisko, PESEL lub/oraz numer paszportu, miejscowość, kod pocztowy miejscowości, ulicę, numer budynku oraz/lub numer mieszkania zamieszkania, numer telefonu. Każdy pracownik jest przypisany do danego hotelu. Pracownicy posiadają również informacje o stanie pracownika, czy jest wolny czy aktualnie zajęty.

Baza danych zawiera informacje również o dyżurach pracowników, do jakich pokoi są przypisani oraz daty ich pracy.

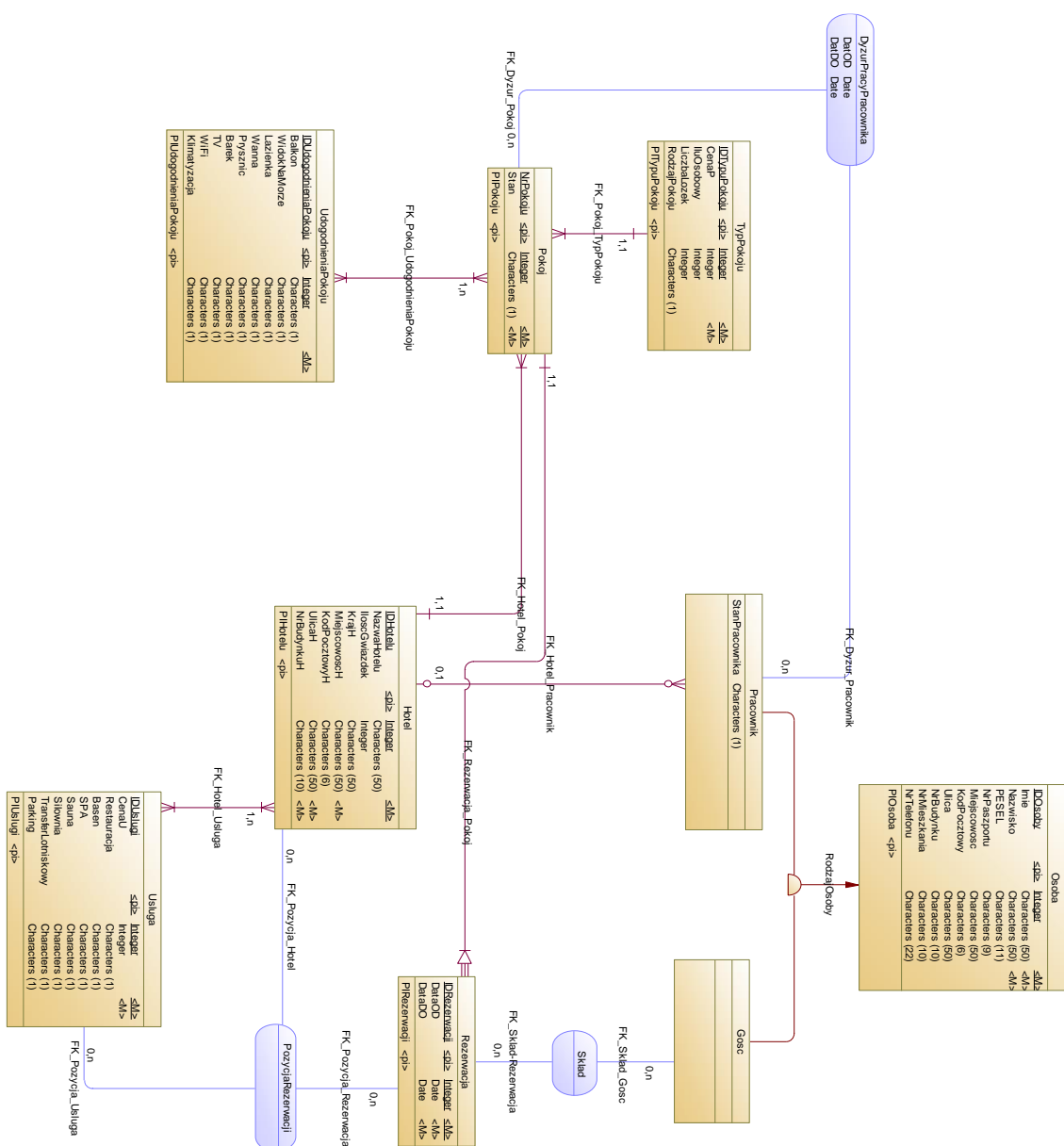
Baza danych zawiera również informacje o gościach. Informacje są takie same jak w informacjach o pracownikach, oprócz informacji o przypisaniu do hotelu oraz stanie osoby.

Baza danych zawiera również informacje o rezerwacjach. Rezerwacje zawierają informacje o dacie rezerwacji, numerze pokoju, typie pokoju, kliencie który rezerwuje, o pakiecie usług i udogodnień.

Projekt bazy danych tworzony był domyślnie dla jednego hotelu. Jednakże w przyszłości właściciel hotelu może zdecydować o otwarciu drugiego hotelu. Przeprojektowanie bazy danych byłoby bardzo praco i czasochłonne. Baza danych została zaprojektowana w taki sposób, aby w przyszłości mogła zostać rozbudowana o nowy hotel. Dlatego zawiera informacje o identyfikatorze hotelu, kraju w którym się znajduje. Jest również możliwość zatrudnienia pracownika, który nie posiada numeru PESEL. Numer paszportu jest bardziej uniwersalny i może zostać wykorzystany, w sytuacji gdy pracownikiem będzie obcokrajowiec.

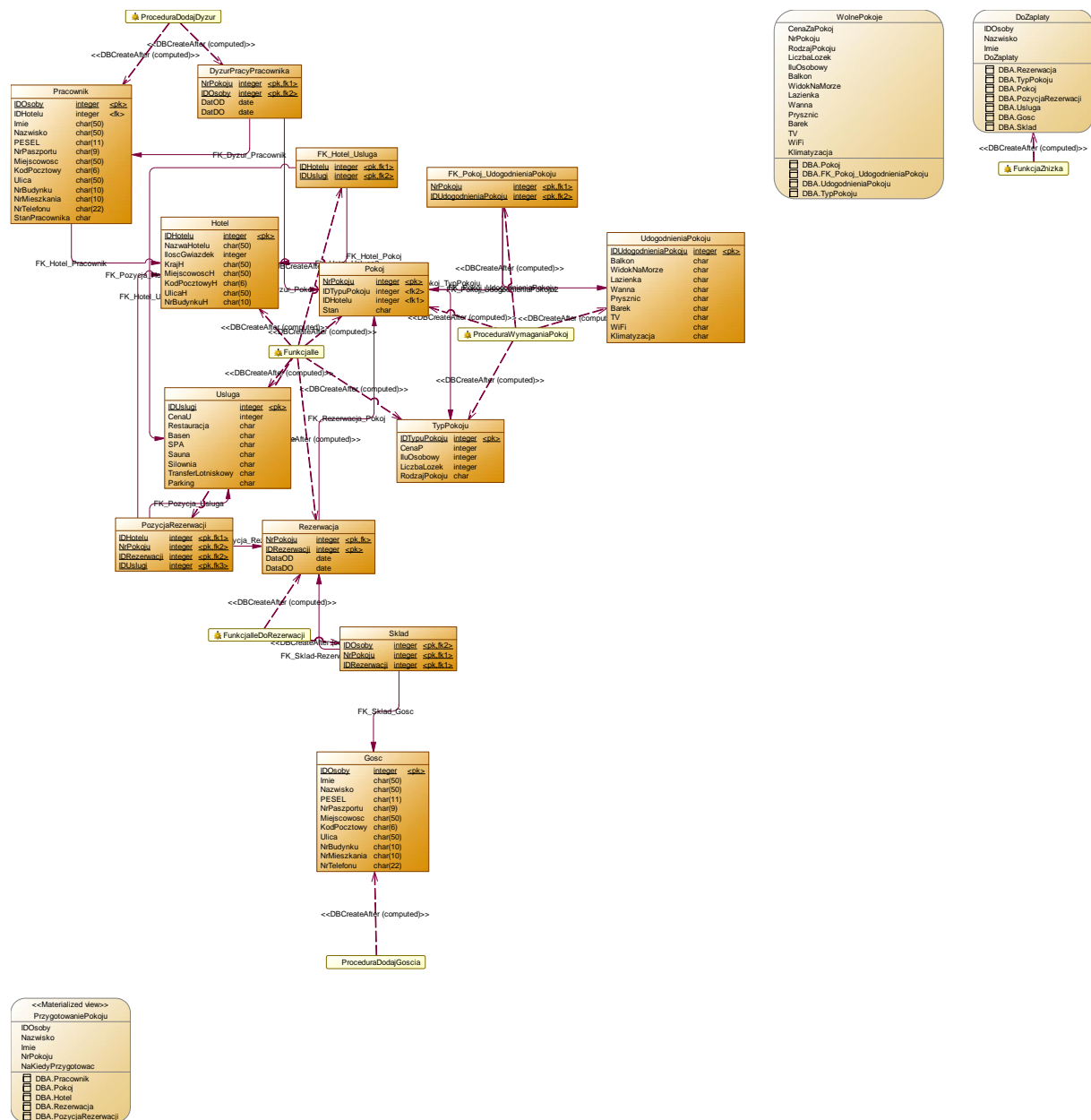
Podobnie z długością numeru telefonu. Została ustalona możliwość wpisania dłuższej liczby cyrk niż 9, ze względu na np. numer kierunkowy telefonu.

3. Model konceptualny bazy danych:





Reverse Engineer PDM



5. WYKAZ TABEL

Nr.	NAZWA TABELI	OPIS
1	Gosc	Gość hotelu. Zawiera informacje osobowe oraz dane o gościu hotelu.
2	Pracownik	Pracownik hotelu. Zawiera dane osobowe i informacje o pracowniku.
3	Rezerwacja	Rezerwacja zawiera informacje o rezerwacji. Daty: od do. Podanie terminu rezerwacji od do jest wymagane.
4	Sklad	Sklad rezerwacji.
5	PozycjaRezerwacji	Pozycja rezerwacji.
6	Usługa	Usługa zawiera informacje odnośnie usług w hotelu.
7	Hotel	Zawiera informacje o danym hotelu.
8	FK_Hotel_Usługa	Zwiazek (relacja) pomiedzy hotelem a usługa
9	DyzurPracy Pracownika	Dyzur pracy pracownika OD DO.
10	Pokoj	Pokoj zawiera informacje o numerze pokoju oraz jego aktualnym stanie. Czy jest zarezerwowany czy jest wolny. Z- Zarezerwowany W - Wolny.
11	TypPokoju	TypPokoju zawiera informacje o typie pokoju, takie jak cena liczba lozek ilu osobowy pokoj. Czy jest to apartament czy standadrowy pokoj,
12	FK_Pokoj_UdogodnieniaPokoju	Zwiazek (relacja) pomiedzy pokojem a udogodnieniami w pokoju.
13	UdogodnieniaPokoju	UdogodnieniaPokoju zawiera informacje o udogodnieniach / wyposażeniu danego pokoju. Np czy zawiera TV: T - tak. N - nie.

WIDOKI

Widok 1 – widok zmaterializowany: PrzygotowaniePokoju

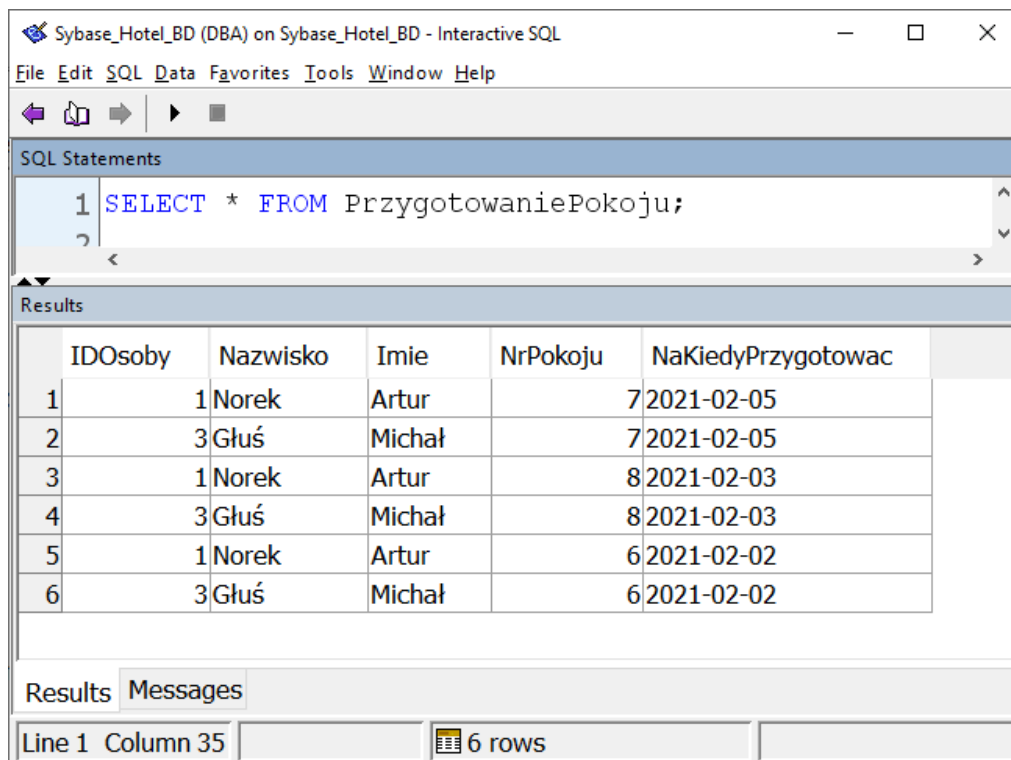
Zadaniem widoku jest pokazanie wolnych pracowników, którzy mogą przygotować dany zarezerwowany pokój. Widok pokazuje na kiedy ma być przygotowany pokój (data: rok-miesiąc-dzień).

KOD:

```
CREATE MATERIALIZED VIEW "DBA"."PrzygotowaniePokoju"
IN "system" AS
SELECT Pracownik.IDOsoby
      ,Pracownik.Nazwisko
      ,Pracownik.Imie
      ,Pokoj.NrPokoju
      ,Rezerwacja.DataOD AS NaKiedyPrzygotowac
FROM Pracownik
      ,Pokoj
      ,Hotel
      ,Rezerwacja
      ,PozycjaRezerwacji
WHERE Pracownik.IDHotelu = Hotel.IDHotelu
      AND Hotel.IDHotelu = Pokoj.IDHotelu
      AND Hotel.IDHotelu = PozycjaRezerwacji.IDHotelu
      AND PozycjaRezerwacji.IDRezerwacji = Rezerwacja.IDRezerwacji
      AND Rezerwacja.NrPokoju = Pokoj.NrPokoju
      AND Pracownik.StanPracownika = 'W'
ORDER BY NaKiedyPrzygotowac DESC
```

Przykładowe polecenie wywołania widoku:

SELECT * FROM PrzygotowaniePokoju;



The screenshot shows a window titled "Sybase_Hotel_BD (DBA) on Sybase_Hotel_BD - Interactive SQL". The menu bar includes File, Edit, SQL, Data, Favorites, Tools, Window, and Help. The SQL Statements pane contains the query: `1 SELECT * FROM PrzygotowaniePokoju;`. The Results pane displays a table with 6 rows and 5 columns: IDOsoby, Nazwisko, Imie, NrPokoju, and NaKiedyPrzygotowac. The Results pane also has tabs for Results and Messages. At the bottom, a status bar indicates "Line 1 Column 35" and "6 rows".

	IDOsoby	Nazwisko	Imie	NrPokoju	NaKiedyPrzygotowac
1	1	Norek	Artur	7	2021-02-05
2	3	Głuś	Michał	7	2021-02-05
3	1	Norek	Artur	8	2021-02-03
4	3	Głuś	Michał	8	2021-02-03
5	1	Norek	Artur	6	2021-02-02
6	3	Głuś	Michał	6	2021-02-02

Zrzut ekran - wywołanie widoku 1: PrzygotowaniePokoju

Widok 2: WolnePokoje

Zadaniem tego widoku jest pokazanie wolnych pokoi. Pokazuje numer danego pokoju, cene, rodzaj pokoju, liczbe lozek, ilu jest osobowy oraz wyposazenie.

KOD:

```
ALTER VIEW "DBA"."WolnePokoje"
AS
SELECT
    TP.CenaP AS CenaZaPokoj
    ,PO.NrPokoju
    ,TP.RodzajPokoju
    ,TP.LiczbaLozek
    ,TP.IluOsobowy
    ,UP.Balkon
    ,UP.WidokNaMorze
    ,UP.Lazienka
    ,UP.Wanna
    ,UP.Prysznic
    ,UP.Barek
    ,UP.TV
    ,UP.WiFi
    ,UP.Klimatyzacja
FROM Pokoj PO
    ,FK_Pokoj_UdogodnieniaPokoju FKPU
    ,UdogodnieniaPokoju UP
    ,TypPokoju TP
WHERE PO.IDTypuPokoju = TP.IDTypuPokoju
    AND PO.NrPokoju = FKPU.NrPokoju
    AND FKPU.IDUdogodnieniaPokoju = UP.IDUdogodnieniaPokoju
    AND PO.Stan = 'W'
GROUP BY
    TP.CenaP
    ,PO.NrPokoju
```

```

,TP.RodzajPokoju
,TP.LiczbaLozek
,TP.IluOsobowy
,UP.Balkon
,UP.WidokNaMorze
,UP.Lazienka
,UP.Wanna
,UP.Prysznic
,UP.Barek
,UP.TV
,UP.WiFi
,UP.Klimatyzacja
ORDER BY TP.CenaP DESC

```

Przykładowe polecenie wywołania widoku:

SELECT * FROM WolnePokoje;

SQL Statements

```
1 SELECT * FROM WolnePokoje;
```

Results

	CenaZaPokoj	NrPokoju	RodzajPokoju	LiczbaLozek	IluOsobowy	Balkon	WidokNaMorze	Lazienka	Wanna	Prysznic	Barek	TV	WiFi	Klimatyzacja
1	300	1A	2	2	T	T	T	T	T	T	T	T	T	T
2	250	2A	1	2	T	T	T	N	T	T	T	T	T	T
3	200	3S	2	2	T	N	T	N	T	T	T	T	T	N
4	150	4S	1	2	N	N	T	N	T	T	T	T	T	N
5	100	5A	1	1	N	N	T	N	T	T	N	N	N	N

Results Messages

Line 2 Column 1 5 rows

Zrzut ekranu- wywołanie widoku 2: WolnePokoje

Widok 3: DoZaplaty

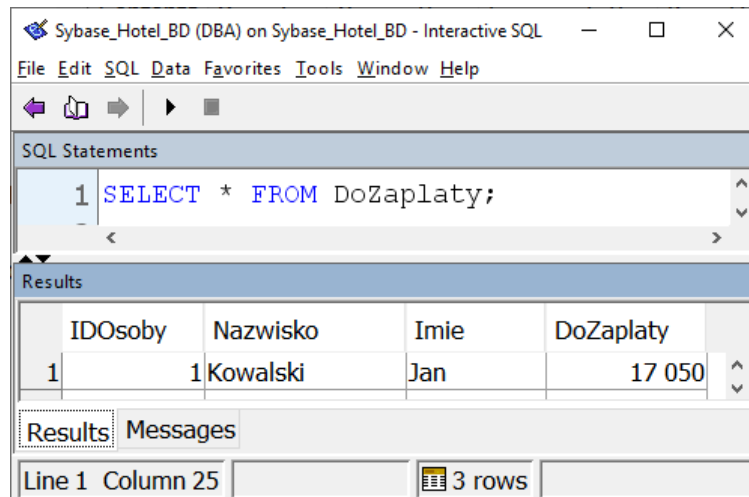
Zadaniem tego widoku jest pokazanie IDOsoby, imienia, nazwiska oraz ile musi zapłacić za pobyt w hotelu. Liczba do zapłaty jest obliczana jako cena pokoju za każdy dzień + jednorazowa opata za usługę.

KOD:

```
ALTER VIEW "DBA"."DoZaplaty"
AS
SELECT G.IDOsoby
       ,G.Nazwisko
       ,G.Imie
       ,sum(datediff(day, R.DataOD, R.DataDO) * TP.CenaP + U.CenaU) AS
DoZaplaty
FROM Rezerwacja R
       ,TypPokoju TP
       ,Pokoj P
       ,PozycjaRezerwacji PR
       ,Uslugu U
       ,Gosc G
       ,Skład S
WHERE TP.IDTypuPokoju = P.IDTypuPokoju
      AND PR.IDUslugi = U.IdUslugi
      AND PR.IDRezerwacji = R.IDRezerwacji
      AND G.IDOsoby = S.IDOsoby
      AND R.NrPokoju = P.NrPokoju
GROUP BY G.IDOsoby
       ,G.Nazwisko
       ,G.Imie
ORDER BY DoZaplaty DESC
```

Przykładowe polecenie wywołania widoku:

SELECT * FROM DoZaplaty;



The screenshot shows a window titled "Sybase_Hotel_BD (DBA) on Sybase_Hotel_BD - Interactive SQL". The "SQL Statements" pane contains the query: `1 SELECT * FROM DoZaplaty;`. The "Results" pane displays a table with the following data:

	IDOsoby	Nazwisko	Imie	DoZaplaty
1	1	Kowalski	Jan	17 050

Below the table, there are tabs for "Results" and "Messages". At the bottom, a status bar indicates "Line 1 Column 25" and "3 rows".

Zrzut ekranu- wywołanie widoku 3:DoZaplaty

PROCEDURY WYZWALANE

Procedura wyzwalana 1: TriggerData

Zadaniem Triggera jest kontrola w tabeli Rezerwacja, czy przed próbą wprowadzenia lub aktualizowania czy data OD nie jest wcześniejsza od bieżącej daty oraz czy DataDO nie jest wcześniejsza od daty DataOD.

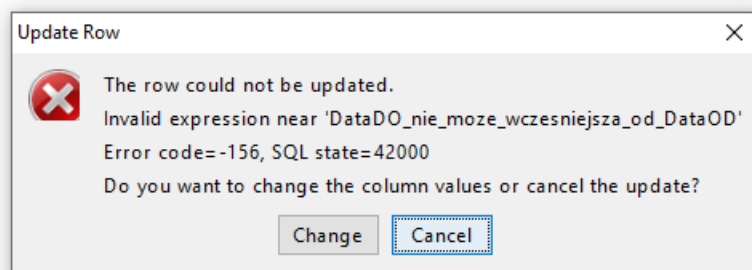
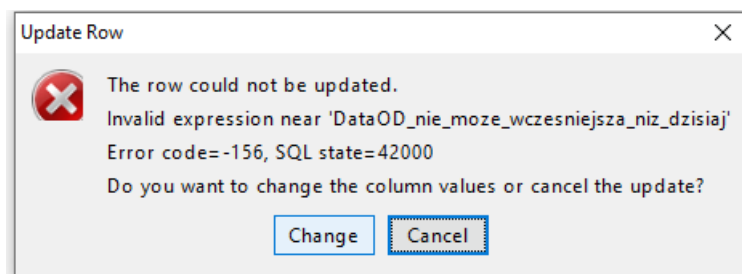
KOD:

```
ALTER TRIGGER "TriggerData" BEFORE INSERT, UPDATE
ORDER 1 ON "DBA"."Rezerwacja"

REFERENCING NEW AS Rezerwacja

FOR EACH ROW
BEGIN
    IF Rezerwacja.DataOD < now(*) THEN
        SIGNAL DataOD_nie_moze_wczesniejsza_niz_dzisiaj;
    END IF;
    IF Rezerwacja.DataOD > Rezerwacja.DataDO THEN
        SIGNAL DataDO_nie_moze_wczesniejsza_od_DataOD;
    END IF;
END
```

Uzyskane efekty:



Procedura wyzwalana 2: TriggerTypPokoju

Zadaniem Triggera jest sprawdzenie, czy przed dodaniem lub zmodyfikowaniem rekordu w tabeli TypPokoju nie ma ceny lub liczby lozek lub ilosc osob mniejszej lub rownej 0.

KOD:

```
ALTER TRIGGER "TriggerTypPokoju" BEFORE INSERT, UPDATE
ORDER 1 ON "DBA"."TypPokoju"

FOR EACH ROW
BEGIN
    IF TypPokoju.CenaP < 0
        THEN
            SIGNAL Cena_nie_moze_byc_mniejsza_niz_0
        END IF;

    IF TypPokoju.CenaP = 0
        THEN
            SIGNAL Cena_nie_moze_byc_rowna_0
        END IF;

    IF TypPokoju.LiczbaLozek <= 0
        THEN
            SIGNAL liczba_lozek_nie_moze_byc_mniejsza_niz_0
        END IF;

    IF TypPokoju.IluOsobowy <= 0
        THEN
            SIGNAL liczba_osob_nie_moze_byc_rowna_0
        END IF;
END;
```


Procedura wyzwalana 3: TriggerPracownikKodPocztowy

Zadaniem Triggera jest kontrola, czy przed próbą dodania lub zmodyfikowania kodu pocztowego, sprawdza czy KodPocztowy w tabeli Pracownik ma 6 znaków (xx-xxx).

KOD:

```
ALTER TRIGGER "TriggerPracownikKodPocztowy" BEFORE INSERT, UPDATE
ORDER 1 ON "DBA"."Pracownik"

FOR EACH ROW
BEGIN
    DECLARE poczta INT;
    SET poczta = (char_length(Pracownik.KodPocztowy));
    IF poczta < 6 THEN
        SIGNAL za_krotki_KOD_POCZTOWY
    END IF;
    IF poczta > 6 THEN
        SIGNAL za_dlugi_KOD_POCZTOWY
    END IF;
END;
```

PROCEDURY WBUDOWANE

Procedura wbudowana 1 z mechanizmem transakcji: ProceduraDodajDyzur

ProceduraDodajDyzur dodaje nowy dyżur w tabeli Dyzur_Pracy_Pracownika oraz zmienia status z Wolny -W na Zajęty -Z w Status w tabeli Pracownik danego pracownika. Transakcja wykona się, gdy status pracownika jest W czyli wolny, jeżeli Z nie wykona się.

Do parametrów procedury podajemy: Nr_pokoju – czyli numer pokoju w którym ma pracownik dyżur. ID_prac czyli ID Pracownika, dd_od, dd_do to data OD DO dyżuru.

KOD:

```
ALTER PROCEDURE "DBA"."ProceduraDodajDyzur" (  
    IN Nr_pokoju INTEGER  
    ,IN ID_prac INTEGER  
    ,IN dd_od DATE  
    ,IN dd_do DATE  
    )  
AS  
BEGIN  
    BEGIN TRANSACTION Dod_D  
  
    DECLARE @x CHARACTER(1)  
  
    SET @x = (  
        SELECT Pracownik.StanPracownika  
        FROM Pracownik  
        WHERE Pracownik.IDOsoby = ID_prac  
    )  
  
    INSERT INTO DyzurPracyPracownika  
    VALUES (  
        Nr_pokoju  
        ,ID_prac
```

```
,dd_od
,dd_do
)
```

```
UPDATE Pracownik
```

```
SET Pracownik.StanPracownika = 'Z'
```

```
WHERE Pracownik.IDOsoby = ID_prac
```

```
IF @x = 'W'
```

```
BEGIN
```

```
COMMIT TRANSACTION
```

```
END
```

```
ELSE
```

```
BEGIN
```

```
ROLLBACK MESSAGE @PRACOWNIK_JUZ_JEST_ZAJETY
```

```
END
```

```
END
```

Pracownik (DBA)

Columns	Constraints	Referencing Constraints	Indexes	Text Indexes	Triggers	Permissions	Dependent Views	Data				
IDOsoby	IDHotelu	Imie	Nazwisko	PESEL	NrPaszportu	Miejscowosc	KodPocztowy	Ulica	NrBudynku	NrMieszkania	NrTelefonu	StanPracownika
1	1	1 Artur	Norek	90030312312	(NULL)	Gdynia	81-000	Stara	3	1	123123123	W
2	2	1 Joanna	Skoczek	95111100123	(NULL)	Gdynia	81-000	Nowa	15	13	987654321	Z
3	3	1 Michał	Głus	80060612399	(NULL)	Sopot	82-000	Gwiezdna	21	14	600600100	W
4	4	1 Mariusz	Drwał	85070712345	(NULL)	Gdynia	81-000	Prosta	3	12	231000123	Z
5	5	1 Hanna	Adamek	76120312377	(NULL)	Gdańsk	80-001	Krzywa	32	1	111222333	Z

Zawartość tabeli Pracownicy przed wywołaniem procedury

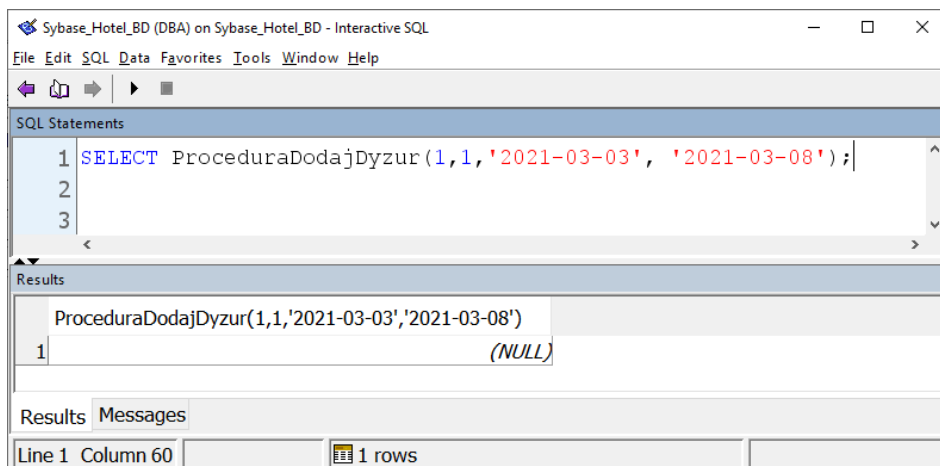
DyzurPracyPracownika (DBA)

Columns	Constraints	Referencing Constraints	Indexes
NrPokoju	IDOsoby	DatOD	DatDO
1	6	2 2021-02-01	2021-03-01
2	7	4 2021-01-28	2021-02-28
3	8	5 2021-01-23	2021-03-10

Zawartość tabeli DyżurPracyPracownika przed wywołaniem procedury

Przykładowe polecenie wywołania procedury:

SELECT ProceduraDodajDyzur(1, 1, '2021-03-03', '2021-03-08');



Wywołanie procedury ProceduraDodajDyzur

DyzurPracyPracownika (DBA)					
Columns		Constraints	Referencing Constraints		Indexes
	NrPokoju	IDOsoby		DatOD	DatDO
1	1	1	1	2021-03-03	2021-03-08
2	6	2	2	2021-02-01	2021-03-01
3	7	4	4	2021-01-28	2021-02-28
4	8	5	5	2021-01-23	2021-03-10

Zawartość tabeli DyzurPracyPracownika po wywołaniu procedury

<div><input type="checkbox"/> Pracownik (DBA)</div>													
<div>Columns Constraints Referencing Constraints Indexes Text Indexes Triggers Permissions Dependent Views Data</div>													
IDOsoby	IDHotelu	Imie	Nazwisko	PESEL	NrPaszportu	Miejscowosc	KodPocztowy	Ulica	NrBudynku	NrMieszkania	NrTelefonu	StanPracownika	
1	1	1	Artur	Norek	90030312312	(NULL)	Gdynia	81-000	Stara	3	1	123123123	Z

Zawartość rekordów pracownika o IDOsoby=1. Po wywołaniu procedury StanPracownika zmienił wartość z 'W' na 'Z'.

Procedura wbudowana 2 z mechanizmem kursora: ProceduraDodajGoscia

ProceduraDodajGoscia dodaje gościa hotelu do bazy danych. W parametry wejściowe należy wpisać: Imie, Nazwisko. Opcjonalnie: PESEL, NrPaszportu, Miejscowosc, KodPocztowy, Ulica, NrBudynku, NrMieszkania, NrTelefonu. Kursor znajduje największe ID Goscia i automatycznie przypisuje do nowego gościa IDGoscia większe o 1.

KOD:

```
ALTER PROCEDURE "DBA"."ProceduraDodajGoscia"
```

```
(
```

```
    IN im CHARACTER(50)
```

```
    ,IN nazw CHAR(50)
```

```
    ,IN pes CHAR(11)
```

```
    ,IN nr_paszportu CHAR(9)
```

```
    ,IN miejsc CHAR(50)
```

```
    ,IN kodpoczty CHAR(6)
```

```
    ,IN ul CHAR(50)
```

```
    ,IN nrbud CHAR(10)
```

```
    ,IN nr_mieszkania CHAR(10)
```

```
    ,IN tel CHAR(22)
```

```
)
```

```
BEGIN
```

```
    DECLARE x INTEGER;
```

```
    DECLARE kursor CURSOR
```

```
    FOR
```

```
    SELECT Gosc.IDOsoby
```

```
    FROM Gosc;
```

```
    OPEN kursor;
```

```
    petla: LOOP
```

```

    FETCH NEXT kursor INTO x;

    IF SQLCODE <> 0 THEN LEAVE petla;

        END IF ;END

        LOOP petla;

SET x = x + 1;

INSERT Gosc
VALUES (

    x

    ,im

    ,nazw

    ,pes

    ,nr_paszportu

    ,miejsc

    ,kodpoczty

    ,ul

    ,nrbud

    ,nr_mieszkania

    ,tel

    );

CLOSE kursor;

END

```

Gosc (DBA)

Columns

Constraints

Referencing Constraints

Indexes

Text Indexes

Triggers

Permissions

Dependent Views

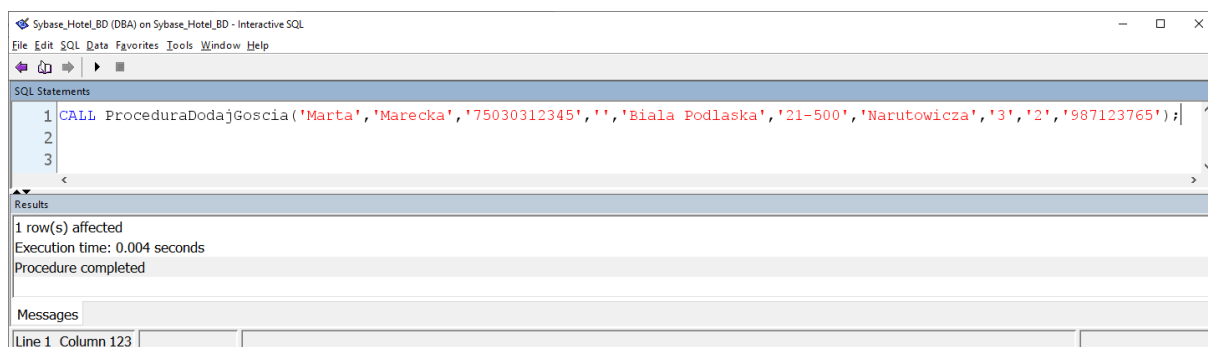
Data

	IDOsoby	Imie	Nazwisko	PESEL	NrPaszportu	Miejscowosc	KodPocztowy	Ulica	NrBudynku	NrMieszkania	NrTelefonu
1	1	Jan	Kowalski	99121212345	(NULL)	Warszawa	03-001	Nowa	4	3	987987987
2	2	Tadeusz	Nowak	70010155555	(NULL)	Warszawa	03-002	Stara	5	12	123123123
3	3	Anna	Krystanowska	80050512332	(NULL)	Poznań	10-001	Klasyczna	21	1	567567567

Zawartość tabeli Gosc przed wywołaniem procedury

Przykładowe wywołanie procedury:

CALL ProceduraDodajGoscia('Marta','Marecka','75030312345','','Biala Podlaska','21-500','Narutowicza','3','2','987123765');



Wywołanie procedury: ProceduraDodajGoscia

Gosc (DBA)											
Columns Constraints Referencing Constraints Indexes Text Indexes Triggers Permissions Dependent Views Data											
	IDOsoby	Imie	Nazwisko	PESEL	NrPaszportu	Miejscowosc	KodPocztowy	Ulica	NrBudynku	NrMieszkania	NrTelefonu
1	1	Jan	Kowalski	99121212345	(NULL)	Warszawa	03-001	Nowa	4	3	987987987
2	2	Tadeusz	Nowak	70010155555	(NULL)	Warszawa	03-002	Stara	5	12	123123123
3	3	Anna	Krystanowska	80050512332	(NULL)	Poznań	10-001	Klasyczna	21	1	567567567
4	4	Marta	Marecka	75030312345		Biala Podla...	21-500	Naruto...	3	2	987123765

Zawartość tabeli Gosc po wywołaniu procedury

Procedura wbudowana 3: ProceduraWymaganiaPokoj

ProceduraWymaganiaPokoj wyświetla NrPokoju, wraz z jego aktualnym stanem, ceną przed i po podwyżce, który spełnia podane wymagania. Za każde spełnione wymaganie odnośnie udogodnienia pokoju doliczana jest podwyżka do ceny. W argumentach procedury podajemy T jeżeli chcemy aby dane udogodnienie było wyszukane, lub N- jeżeli nie.

Za każde udogodnienie pokoju doliczana jest podwyżka 10% do ceny pokoju.

Argumenty procedury(balkon, widoknamorze, lazienka, wanna, prysznic, barek, tv, wifi, klimatyzacja).

KOD

```
ALTER PROCEDURE "DBA"."ProceduraWymaganiaPokoj" (
```

```
    IN _balkon CHAR(1)
  ,IN _widoknamorze CHAR(1)
  ,IN _lazienka CHAR(1)
  ,IN _wanna CHAR(1)
    ,IN _prysznic CHAR(1)
  ,IN _barek CHAR(1)
  ,IN _tv CHAR(1)
  ,IN _wifi CHAR(1)
  ,IN _klimatyzacja CHAR(1))
```

```
BEGIN
```

```
    DECLARE podwyzka FLOAT;
```

```
    DECLARE nr INTEGER;
```

```
    SET podwyzka = 1;
```

```
        IF _balkon = 'T' THEN
```

```
            SET podwyzka = podwyzka + 0.1
```

```
        END IF ;
```

```
        IF _widoknamorze = 'T' THEN
```

```
            SET podwyzka = podwyzka + 0. 1
```

```
        END IF ;
```



```

IF _lazienka = 'T' then
SET podwyzka = podwyzka + 0. 1

END IF ;

IF _wanna = 'T' then
SET podwyzka = podwyzka + 0. 1
END IF ;

IF _prysznic = 'T' then
SET podwyzka = podwyzka + 0. 1
END IF ;

IF _barek = 'T' then
SET podwyzka = podwyzka + 0. 1
END IF ;

IF _tv = 'T' then
SET podwyzka = podwyzka + 0. 1
END IF ;

IF _wifi = 'T' then
SET podwyzka = podwyzka + 0. 1
END IF ;

IF _klimatyzacja = 'T' then
SET podwyzka = podwyzka + 0. 1
END IF ;

SELECT
    Pokoj.NrPokoju
    ,Pokoj.Stan AS AktualnyStanPokoju
    ,TypPokoju.CenaP AS CenaPrzedPodwyzka

```

```
        ,(TypPokoju.CenaP * podwyzka) AS AktualnaCena

FROM TypPokoju
    INNER JOIN Pokoj
    ON TypPokoju.IDTypuPokoju=Pokoj.IDTypuPokoju

    INNER JOIN FK_Pokoj_UdogodnieniaPokoju
    ON Pokoj.NrPokoju=FK_Pokoj_UdogodnieniaPokoju.NrPokoju

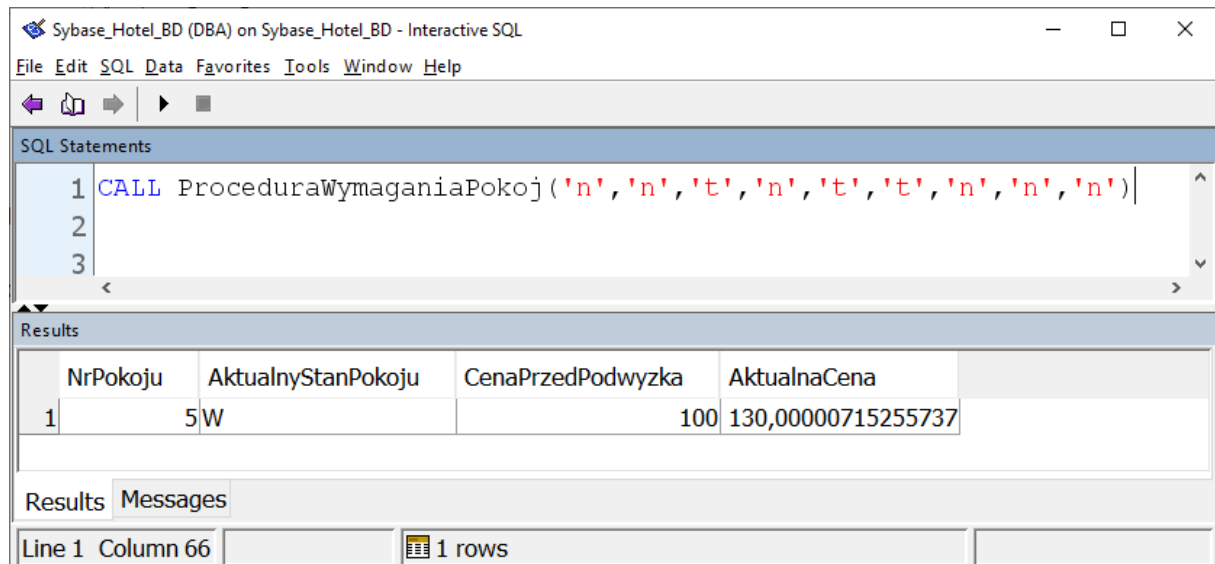
    INNER JOIN UdogodnieniaPokoju
    ON FK_Pokoj_UdogodnieniaPokoju.IDUdogodnieniaPokoju
    =UdogodnieniaPokoju.IDUdogodnieniaPokoju

WHERE
    UdogodnieniaPokoju.Balkon = _balkon
    AND UdogodnieniaPokoju.WidokNaMorze = _widoknamorze
    AND UdogodnieniaPokoju.Lazienka = _lazienka
    AND UdogodnieniaPokoju.Wanna = _wanna
    AND UdogodnieniaPokoju.Prysznic = _prysznic
    AND UdogodnieniaPokoju.Barek = _barek
    AND UdogodnieniaPokoju.TV = _tv
    AND UdogodnieniaPokoju.WiFi = _wifi
    AND UdogodnieniaPokoju.Klimatyzacja = _klimatyzacja

END;
```

Przykładowe wywołanie procedury:

CALL ProceduraWymaganiaPokoj('n','n','t','n','t','t','n','n','n')



The screenshot shows a Sybase Interactive SQL window titled "Sybase_Hotel_BD (DBA) on Sybase_Hotel_BD - Interactive SQL". The menu bar includes File, Edit, SQL, Data, Favorites, Tools, Window, and Help. The SQL Statements pane contains the following code:

```
1 CALL ProceduraWymaganiaPokoj('n','n','t','n','t','t','n','n','n')|
2
3
```

The Results pane displays a table with the following data:

	NrPokoju	AktualnyStanPokoju	CenaPrzedPodwyzka	AktualnaCena
1		5W	100	130,00000715255737

At the bottom, there are tabs for "Results" and "Messages". The status bar indicates "Line 1 Column 66" and "1 rows".

Wywołanie procedury: ProceduraWymaganiaPokoj

FUNKCJE WBUDOWANE

Funkcja wbudowana 1 z mechanizmem kursora: FunkcjaZnizka

FunkcjaZnizka odlicza zniżkę naliczoną dla Gościa od jego pozycji zapłaty. Funkcja ta działa na widoku DoZapłaty. W argumentach funkcji podajemy: IDOsoby, _procentznizki. Czyli ID gościa, oraz wartość zniżki, która ma zostać odjęta od rachunku za wynajem pokoju.

KOD:

```
ALTER FUNCTION "DBA"."FunkcjaZnizka" (  
    _id_goscia INTEGER  
    ,_procentznizki FLOAT  
)  
  
RETURNS INTEGER  
DETERMINISTIC  
  
BEGIN  
    DECLARE KASA FLOAT;  
    DECLARE x DOUBLE;  
    DECLARE kursor DYNAMIC SCROLL CURSOR  
    FOR  
  
        SELECT DoZapłaty  
        FROM DoZapłaty  
        WHERE DoZapłaty.IDOsoby = _id_goscia;  
  
    SET KASA = 0;  
    OPEN kursor;  
  
    petla: LOOP  
  
        FETCH NEXT kursor INTO x;  
  
        IF sqlcode <> 0 THEN LEAVE petla;
```

```

        END IF ;

        SET KASA = KASA + x;

    END LOOP petla;

    CLOSE kurson;

    SET KASA = KASA - (KASA * (_procentznizki / 100));

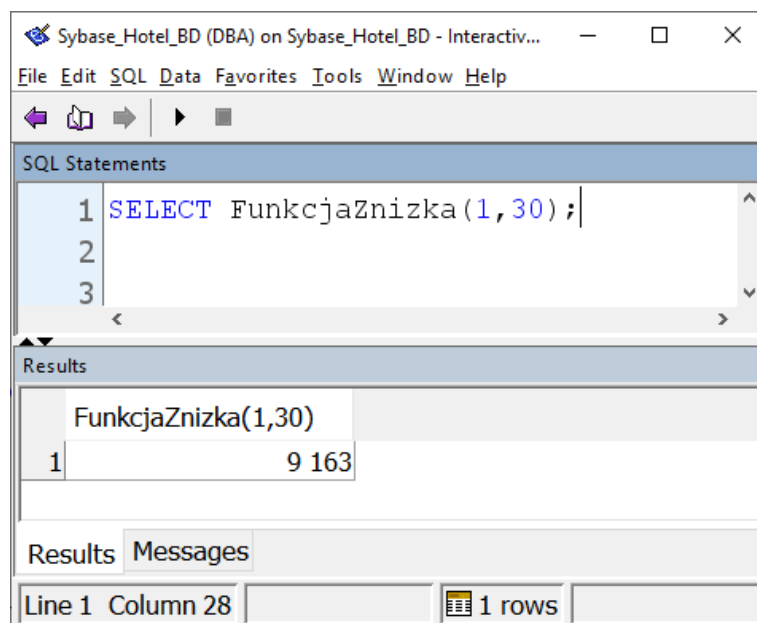
    RETURN KASA;

END

```

Przykładowe wywołanie funkcji wbudowanej:

SELECT FunkcjaZnizka(1,30);



Wynik wywołania funkcji wbudowanej: FunkcjaZnizka dla gościa o IDOsoby=1, zniżce 30%.

Funkcja wbudowana 2 z mechanizmem transakcji: Funkcjalle

FunkcjaIle ma za zadanie dokonać rezerwację za pokój i zwrócić resztę z podanej kwoty przez gościa. Gości podaje sumę pieniędzy, jaką chce wydać na rezerwację danego pokoju o danych usługach w podanym terminie. Pokój zostaje zarezerwowany. Jeżeli kwota jest większa od ceny, wtedy zostaje wyświetlona informacja, ile należy zwrócić pieniędzy. Jeżeli kwota jest niewystarczająca, pokój nie zostanie zarezerwowany. Argumentami funkcji są: _pieniadze – podana przez gościa kwota pieniędzy, _id_hotelu, _id_goscia, _nr_uslugi, _dd_od czyli data rezerwacji OD, _dd_do czyli data rezerwacji DO.

KOD:

```
ALTER FUNCTION "DBA"."FunkcjaIle" (  
    pieniadze INTEGER  
    ,_id_hotelu INTEGER  
    ,_id_goscia INTEGER  
    ,_nr_pokoju INTEGER  
    ,_nr_uslugi INTEGER  
    ,_dd_od DATE  
    ,_dd_do DATE  
)  
RETURNS INTEGER  
DETERMINISTIC  
  
BEGIN  
    DECLARE ILE INTEGER;  
    DECLARE cena INTEGER;  
    DECLARE _numer_rez INTEGER;  
  
    SET cena = (  
        SELECT sum(datediff(day, _dd_od, _dd_do) *  
TypPokoju.CenaP + Usługa.CenaU) AS Zaplata  
  
        FROM TypPokoju  
        JOIN Pokoj ON TypPokoju.IDTypuPokoju = Pokoj.IDTypuPokoju  
        JOIN Hotel ON (Pokoj.IDHotelu = Hotel.IDHotelu)
```

```

        JOIN FK_Hotel_Usluga ON (Hotel.IDHotelu =
FK_Hotel_Usluga.IDHotelu)

        JOIN Usluga ON (FK_Hotel_Usluga.IDUslugi =
Usluga.IDUslugi)

        WHERE Pokoj.NrPokoju = _nr_pokoju
        AND Usluga.IDUslugi = _nr_uslugi
    );

SET _numer_rez = (
    SELECT max(IDRezerwacji)
    FROM Rezerwacja
);

SET _numer_rez = _numer_rez + 1;

INSERT Rezerwacja
VALUES (
    _nr_pokoju
    ,_numer_rez
    ,_dd_od
    ,_dd_do
);

INSERT PozycjaRezerwacji
VALUES (
    _id_hotelu
    ,_nr_pokoju
    ,_numer_rez
    ,_nr_uslugi
);

IF pieniadze >= cena THEN
    SET ILE = pieniadze - cena;

```

```

COMMIT ELSE

ROLLBACK

END IF ;

RETURN ILE;

END;

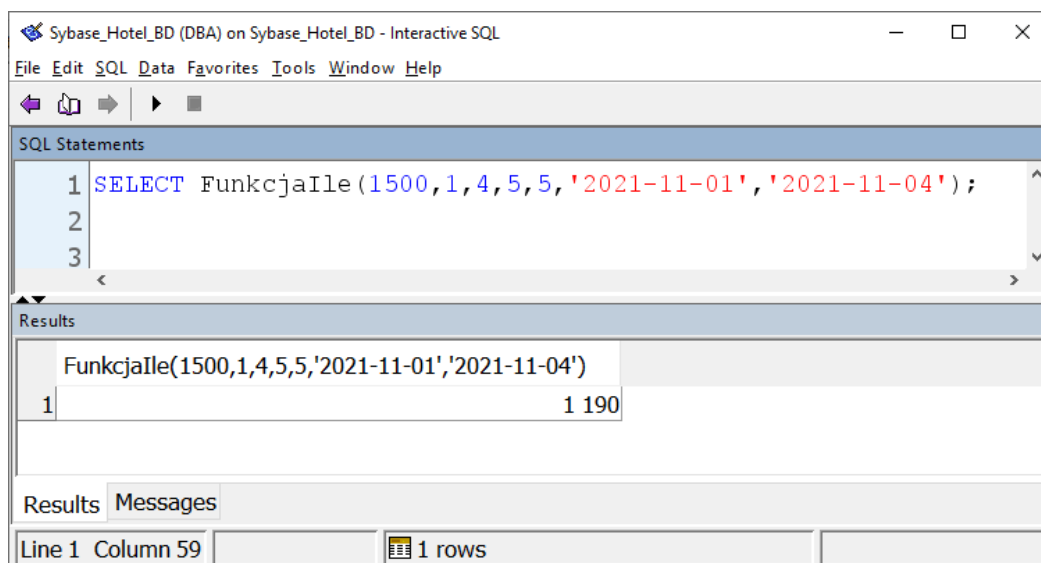
```

Rezerwacja (DBA)					
	Columns	Constraints	Referencing Constraints	Indexes	Text
	NrPokoju	IDRezerwacji	DataOD	DataDO	
1	6	1	2021-02-10	2021-02-13	
2	7	2	2021-02-05	2021-02-20	
3	8	3	2021-01-15	2021-02-25	


Zawartość tabeli Rezerwacja przed wykonaniem funkcji.

Przykładowe wywołanie funkcji wbudowanej:

```
SELECT FunkcjaIle(1500,1,4,5,5,'2021-11-01','2021-11-04');
```



Wynik wywołania funkcji wbudowanej:

 Rezerwacja (DBA)

Columns	Constraints	Referencing Constraints	Indexes	Te:
	NrPokoju	IDRezerwacji	DataOD	DataDO
1	5	4	2021-11-01	2021-11-04
2	6	1	2021-02-10	2021-02-13
3	7	2	2021-02-05	2021-02-20
4	8	3	2021-01-15	2021-02-25

Zawartość tabeli Rezerwacja po wykonaniu funkcji.

Funkcja wbudowana 3 -funkcja z kursorem 2: FunkcjaIleDoRezerwacji

Zadaniem funkcji FunkcjaIleDoRezerwacji jest zwrócenie ilości dni do danej rezerwacji. Funkcja przyjmuje argumenty: _id_goscia- czyli ID gościa rezerwacji oraz _nr_pokoju, czyli numer pokoju rezerwacji.

KOD:

```
ALTER FUNCTION "DBA"."FunkcjaIleDoRezerwacji"(  
    _id_goscia INTEGER  
    ,_nr_pokoju INTEGER  
    )  
RETURNS INTEGER  
DETERMINISTIC  
  
BEGIN  
    DECLARE ILEDNI INTEGER;  
    DECLARE kursor DYNAMIC SCROLL CURSOR  
    FOR  
    SELECT DataOD  
    FROM Rezerwacja  
    JOIN Sklad  
        ON (Rezerwacja.IDRezerwacji=Skład.IDRezerwacji)  
        WHERE Sklad.IDOsoby = _id_goscia  
        AND Sklad.NrPokoju = _nr_pokoju;  
  
    DECLARE x_data DATE;  
    SET ILEDNI = 0;  
    OPEN kursor;  
  
    petla: LOOP  
  
    FETCH NEXT kursor INTO x_data;  
  
    IF sqlcode <> 0 then leave petla;
```

```

END IF ;

        SET ILEDNI = x_data - now(*);

END LOOP petla;

CLOSE kursor;

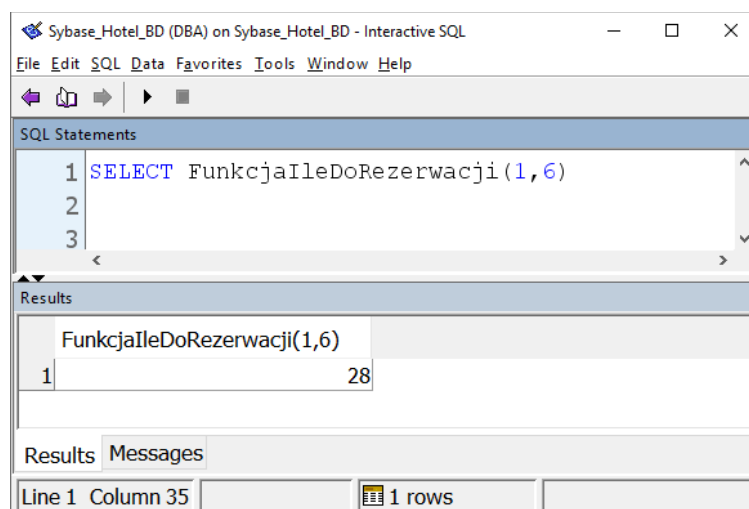
RETURN ILEDNI;

END

```

Przykładowe wywołanie funkcji wbudowanej:

SELECT FunkcjaIleDoRezerwacji(1,6)



Wynik wywołania funkcji wbudowanej: FunkcjaIleDoRezerwacji

Rezerwacja (DBA)					
Columns	Constraints	Referencing Constraints	Indexes	Te:	
	NrPokoju	IDRezerwacji	DataOD	DataDO	
1	5	4	2021-11-01	2021-11-04	
2	6	1	2021-03-10	2021-03-13	

Zawartość tabeli Rezerwacja.

Funkcja zwróciła wartość 28. Od 10.02.2021 do 10.03.2021 jest 28 dni.

UŻYTKOWNICY:

1. Administrator – pełne uprawnienia

KOD:

```
CREATE USER "Administrator" IDENTIFIED BY '***';  
GRANT DBA  
    ,RESOURCE  
    ,REMOTE DBA  
    ,BACKUP  
    ,VALIDATE  
    ,PROFILE  
    ,READFILE  
    ,READCLIENTFILE  
    ,WRITECLIENTFILE  
    TO "Administrator";  
  
COMMENT ON USER "Administrator" IS 'Administrator jest to użytkownik z pełnymi  
uprawnieniami w bazie danych. Zarządza bazą danych.';
```

2. Recepcjonista

KOD:

```
CREATE USER "Recepcjonista" IDENTIFIED BY '***';  
GRANT READFILE  
    ,READCLIENTFILE  
    ,WRITECLIENTFILE  
    TO "Recepcjonista";  
  
COMMENT ON USER "Recepcjonista" IS 'Recepcjonista jest to pracownik hotelu, którego zadaniem  
jest między innymi rezerwacja usług.';
```

3. Księgowa

KOD:

```
CREATE USER "Ksiegowa" IDENTIFIED BY '***';  
GRANT PROFILE  
        ,READFILE  
        ,READCLIENTFILE  
        ,WRITECLIENTFILE  
        TO "Ksiegowa";  
COMMENT ON USER "Ksiegowa" IS 'Ksiegowa jest to pracownik hotelu, której zadaniem jest  
rozliczanie finansowe gości.';
```

7. WNIOSKI

Baza danych spełnia wszystkie założenia przyjęte przy projektowaniu bazy danych. Baza danych została zaimplementowana w dwóch środowiskach. Baza danych została uzupełniona odpowiednimi danymi, niezbędnymi do pracy projektowej. Widoki, procedury wyzwalane, procedury wbudowane oraz funkcje wbudowane zostały zaimplementowane zgodnie z oczekiwaniami projektanta. Prawidłowo utworzone widoki, procedury oraz funkcje ułatwiają i przyspieszają pracę na bazie danych. Przykładowo: dodanie nowego gościa lub nowego dyżury jest ułatwione. Baza danych zawiera trzech użytkowników, każdy z innymi uprawnieniami. Baza danych zawiera procedury wyzwalane (triggery), których zadaniem jest np. zabezpieczenie użytkownika przed podaniem niewłaściwej długości kodu pocztowego. Projekt bazy danych został utworzony z myślą o przyszłości – np. dodaniu nowego hotelu w innym kraju. Projekt bazy danych został utworzony przemyślanie, dzięki czemu w przyszłości nie będzie wymagać dużych modyfikacji.