Dokumentacja projektu

z przedmiotu Bazy Danych

Imię: Piotr

Nazwisko: Podbielski

Grupa: I4Z6S1

Data wykonania dokumentacji: 30.01.2016 r.

Analiza biznesowa projektowanej rzeczywistości

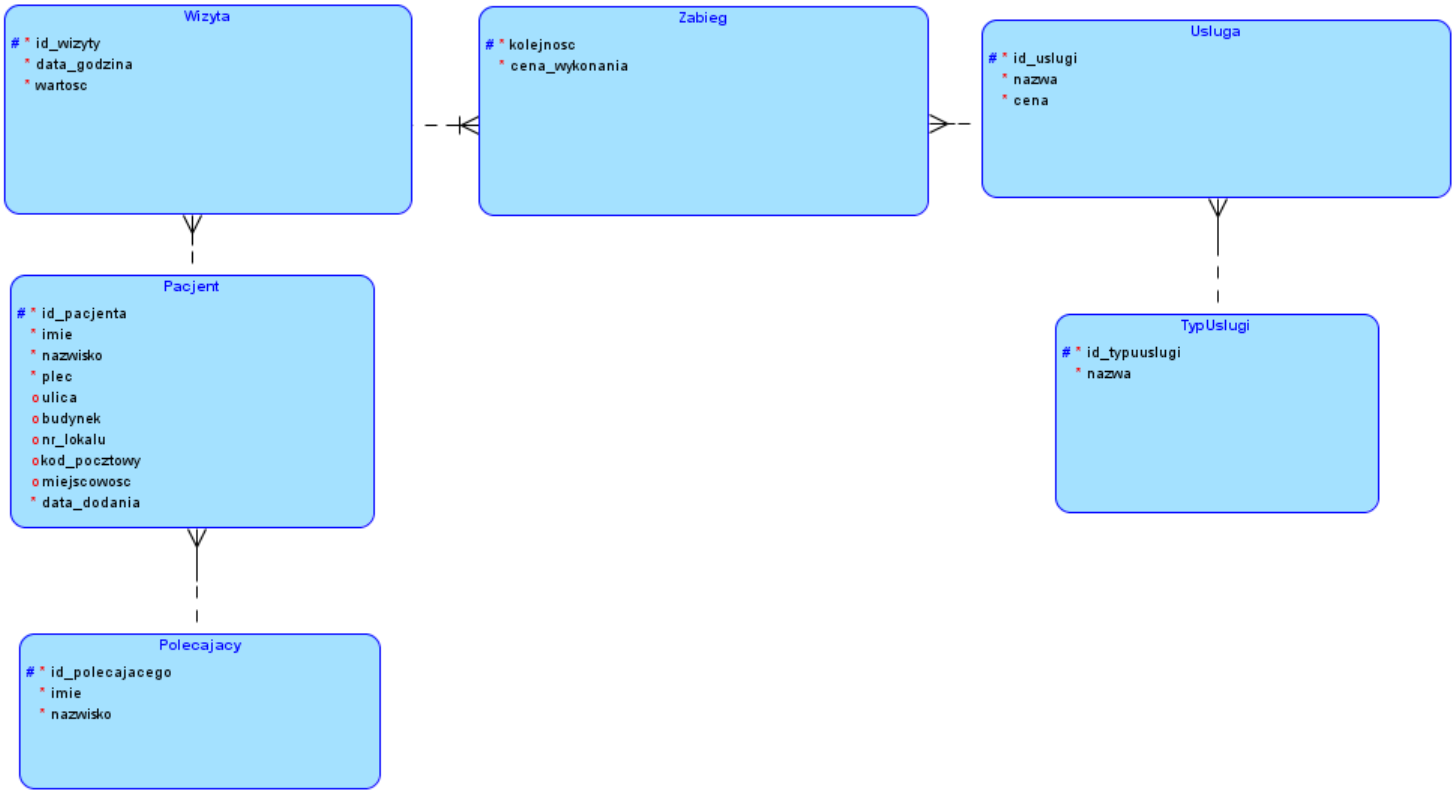
System bazy danych ma za zadanie ułatwić zarządzanie wizytami oraz pacjentami we wszelkiego rodzaju Gabinetach związanych z Medycyną. Pierwotnie pomysł zaczerpnięty został z dziedziny, jaką wykonywał Ojciec autora projektu. Załóżmy, na potrzeby dokumentacji, że wdrożenie projektu wykonane zostanie dla Gabinetu Medycyny Fizykalnej.

Do Gabinetu przychodzą Pacjenci od pewnych Polecających (zazwyczaj lekarze). W bazie danych należy przetrzymywać spis pacjentów. Pacjenci umawiają się na Wizyty na konkretną datę i godzinę. Godziny funkcjonowania Gabinetu to przedział czasowy od 9:00 do 16, gdzie ostatnia Wizyta zaczyna się o godzinie 16. Gabinet funkcjonuje 7 dni w tygodniu. Wizyta posiada przypisane do niej Zabiegi. Zabieg jest konkretną Usługą z podaną kolejnością wykonania. Usługi są wymiarowane przez Typ usługi.

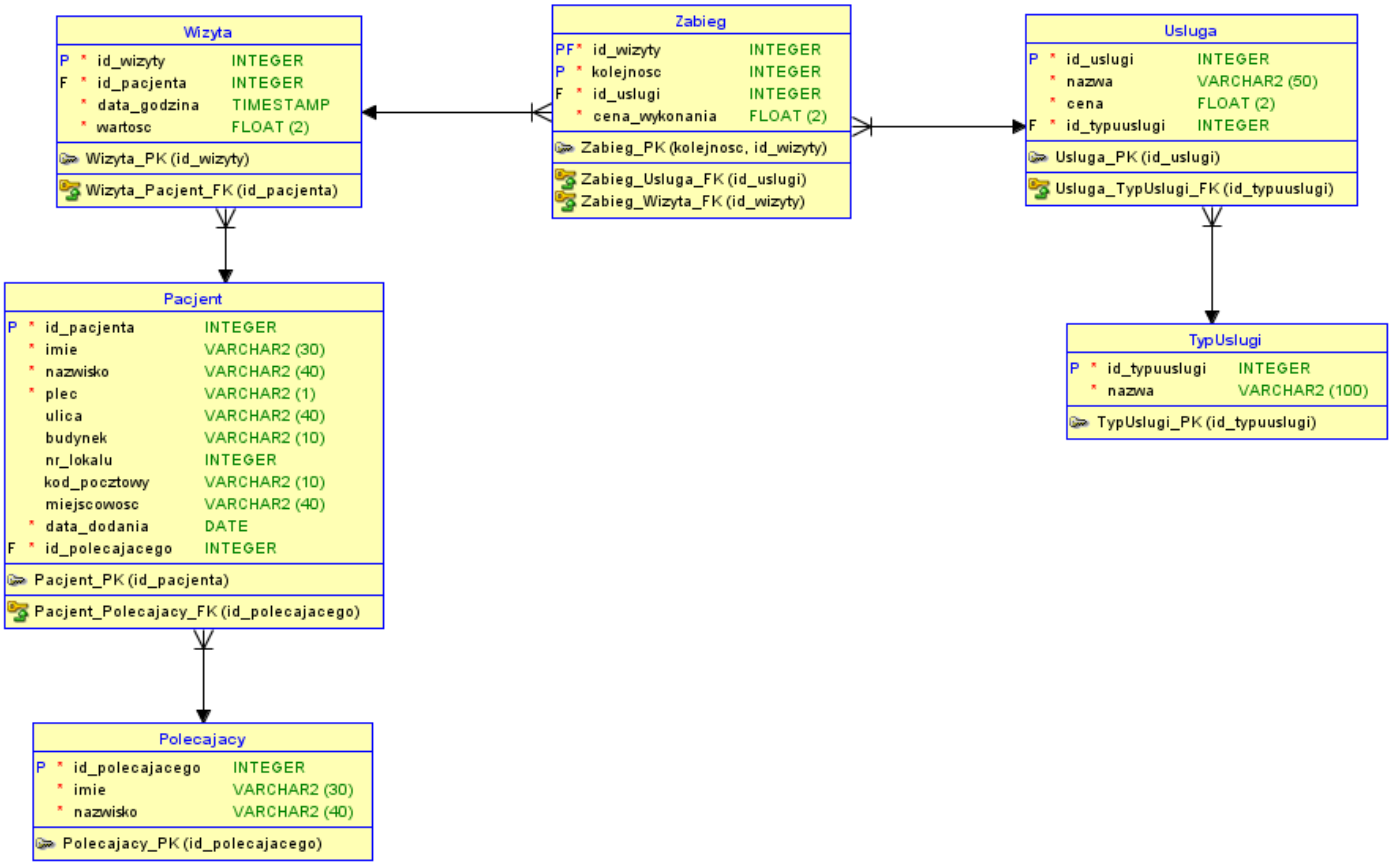
Podstawowym widokiem, który chce posiadać docelowy klient systemu jest zestawienie wizyt w obecnym dniu oraz obłożenie terminów na następne 7 dni.

Wdrożenie systemu pozwoli na migrację z analogowych nośników danych (skierowania, kartki z wydrukowaną tabelką z datami i godzinami wizyt) na cyfrowe.

Model logiczny i relacyjny bazy danych



Ilustracja 1. Model logiczny bazy danych



Ilustracja 2. Model relacyjny bazy danych

Oprogramowanie tworzące bazę danych oraz generator danych wraz z komentarzami

Baza danych można uruchomić na dwóch środowiskach: Oracle oraz Sybase. Na środowisku Oracle zaimplementowane zostały podstawowe elementy gwarantujące działanie projektu: skrypt tworzący strukturę tabel wraz z więzami, skrypt usuwający bazę danych, skrypt importujący dane słownikowe oraz dodatkowo skrypt importujące przykładowe dane transakcji (Wizyty I Zabiegi). Na środowisku Sybase zaimplementowane zostały dodatkowo wyzwalacze, procedury oraz funkcje, a zamiast importowania przykładowych danych transakcyjnych zaimplementowany został generator tych danych na podstawie danych słownikowych.

Generatorem jest procedura, która dodaje określoną liczbę wizyt w określonym przedziale czasowym. Prezentuje się ona następująco:

```
CREATE OR REPLACE PROCEDURE pr_dodaj_wizyty
(od_dnia DATE, do_dnia DATE, liczba INT)
BEGIN
    DECLARE v_znaleziono_pusty_termin INTEGER;
    DECLARE v_roznica_w_dniach INTEGER;
    DECLARE v_wylosowany_offset_dnia INTEGER;
    DECLARE v_wylosowana_data_godzina TIMESTAMP;
    DECLARE v_wylosowany_offset_godziny INTEGER;
```

```
    DECLARE v_bledow_znalezienia INTEGER;
    DECLARE v_flaga INTEGER;

    -- liczenie roznicy w dniach, aby wylosowac ktorego dnia dodac wizyte
    SET v_roznica_w_dniach = DATEDIFF(DAY, od_dnia, do_dnia);

    -- petla po liczbie wizyt do dodania
    WHILE (liczba > 0) LOOP
        SET v_znaleziono_pusty_termin = 0;
        SET v_bledow_znalezienia = 0;
        SET v_flaga = 1;

        -- petla wykonywana dopoki znaleziono pusty termin lub jezeli po 14 iteracjach nie
znaleziono pustego termina
        WHILE (v_flaga = 1) LOOP
            -- losowanie liczby, ktora pozwoli na wylosowanie dnia wizyty
            SET v_wylosowany_offset_dnia = fn_losuj_liczbe(0, v_roznica_w_dniach);
            -- ustawienie dnia wizyty
            SET v_wylosowana_data_godzina = DATEADD(DAY, v_wylosowany_offset_dnia, od_dnia);

            -- losowanie liczby, ktora pozwoli na wylosowanie godziny wizyty
            SET v_wylosowany_offset_godziny = fn_losuj_liczbe(0, 14);
            -- ustawienie godziny wizyty

 SET v_wylosowana_data_godzina = DATEADD(MINUTE, 9 * 60 + v_wylosowany_offset_godziny * 30, v_wylos
owana_data_godzina);

            -- sprawdzenie czy w tym terminie nie istnieje juz wizyta

 IF NOT EXISTS (SELECT * FROM Wizyta WHERE data_godzina = v_wylosowana_data_godzina) THEN
                -- nie istnieje, wiec przerwij petle
                SET v_znaleziono_pusty_termin = 1;
                SET v_flaga = 0;
            ELSE
                -- istnieje, wiec dodaj blad znalezienia
                SET v_bledow_znalezienia = v_bledow_znalezienia + 1;
            END IF;

            -- jezeli po 14 iteracjach nadal nie znaleziono wolnego terminu to znaczy ze
najprawdopodobniej go nie ma - przewij petle
            IF (v_bledow_znalezienia >= 14) THEN
                SET v_flaga = 0;
            END IF;
        END LOOP;
```

```
        -- jezeli znaleziono pusty termin to wywolaj procedure dodawania wizyty
        IF (v_znaleziono_pusty_termin = 1) THEN
            CALL pr_dodaj_wizyte(v_wylosowana_data_godzina);
        END IF;

        -- zmniejsz liczbe iteracji o jeden
        SET liczba = liczba - 1;
    END LOOP;
END;
```

Każda procedura czy funkcja ma wspólny jeden mianownik – deklarację zmiennych na początku. Dodatkowo funkcje zwracają jakieś wartości, a procedury nie. Opis funkcjonowania *pr_dodaj_wizyty* dostępny jest jako komentarze w ciele tej procedury i identycznie będzie dla kolejnych wycinków kodu.

Procedura *pr_dodaj_wizyty* korzysta z wielu funkcji i procedur. Przedstawiam po kolei kod każdej z nich:

```
CREATE OR REPLACE FUNCTION fn_losuj_liczbe
(v_min INTEGER, v_max INTEGER)
RETURNS INTEGER
BEGIN
    DECLARE v_wylosowana_wartosc INTEGER;

    -- losowanie liczby funkcją rand i zaokrąglanie funkcją round z przedziału od v_min do
v_max
    SET v_wylosowana_wartosc = round(v_min + (v_max - v_min) * rand(), 0);

RETURN v_wylosowana_wartosc;
END;
```

Powyższa funkcja *fn_losuj_liczbe* losuje liczbę z przedziału od v_min do v_max oraz zwraca tę liczbę.

```
CREATE OR REPLACE PROCEDURE pr_dodaj_wizyte
(data_godzina TIMESTAMP)
BEGIN
    DECLARE v_id_pacjenta INTEGER;
  DECLARE v_liczba_zabiegow INTEGER;
    DECLARE v_id_wizyty INTEGER;

    -- losowanie pacjenta dla ktorego dodana zostanie wizyta
    SET v_id_pacjenta = fn_losuj_pacjenta();

    -- jezeli znaleziono pacjenta to dalszy kod
    IF (v_id_pacjenta != -1) THEN
        -- dodanie wizyty dla wylosowanego pacjenta o podanej w parametrze procedury dacie i
czasie
        INSERT INTO Wizyta (id_pacjenta, data_godzina, wartosc)
            VALUES (v_id_pacjenta, data_godzina, 0.00);
```

```
        -- wylosowanie liczby zabiegow jaka dodac do wizyty
        SET v_liczba_zabiegow = fn_losuj_liczbe(1, 3);
        -- pobranie id_wizyty, czyli klucza glownego wczesniej wstawionej wizyty za pomocą
wartośćci zmiennej @@identity, którą ustawia mechanizm autoinkrementacji
        SET v_id_wizyty = @@IDENTITY;

        -- pętla wykonująca się tyle razy, ile wynosi wartość zmiennej v_liczba_zabiegow
        WHILE (v_liczba_zabiegow > 0) LOOP
            -- wywołanie procedury dodającej zabieg do wizyty o określonym id_wizyty
            CALL pr_dodaj_zabieg_do_wizyty(v_id_wizyty);

            -- zmniejszenie zmiennej odpowiedzialnej za odpowiednia liczbe iteracji pętli
            SET v_liczba_zabiegow = v_liczba_zabiegow - 1;
        END LOOP;
    END IF;
END;
```

Procedura *pr_dodaj_wizyte* odpowiedzialna jest za dodanie Wizyty z określonym w parametrze procedury czasie (data i godzina). Wartym zwrócenia faktem jest, że klucze głównych wszystkich tabel jako wartość domyślną ustawiony mają mechanizm autoinkrementacji. Dzięki temu, przy dodawaniu kolejnych rekordów, klient końcowy nie musi martwić się o unikalność klucza głównego podczas dodawania rekordu. Aby po dodaniu rekordu otrzymać informację jaki jest jego klucz główny należy sprawdzić zawartość zmiennej o nazwie *@@identity* ustawianej przez ten mechanizm. Dodatkowo procedura dodaje wylosowaną liczbę zabiegów do Wizyty.

```
CREATE OR REPLACE FUNCTION fn_losuj_pacjenta()
-- zwracany typ
RETURNS INTEGER
BEGIN
    -- deklaracja zmiennych
  DECLARE v_wylosowane_id_pacjenta INTEGER;
  DECLARE v_min_id_pacjenta INTEGER;
  DECLARE v_max_id_pacjenta INTEGER;
  DECLARE v_flaga INTEGER;

    -- pobranie wartosci klucza glownego o minimalnej wartosci z tabeli Pacjent
  SELECT MIN(id_pacjenta) INTO v_min_id_pacjenta FROM Pacjent;
    -- pobranie wartosci klucza glownego o maksymalnej wartosci z tabeli Pacjent
  SELECT MAX(id_pacjenta) INTO v_max_id_pacjenta FROM Pacjent;

    -- sprawdzenie czy pobrano wartosci klucza glownego minimalnego i maksymalnego
  IF ((v_min_id_pacjenta IS NOT NULL) AND (v_max_id_pacjenta IS NOT NULL)) THEN
        -- ustawienie flagi odpowiedzialnej za kontynuowanie petli
    SET v_flaga = 0;

        -- petla iteruje dopoki wylosowane id pacjenta istnieje
    WHILE (v_flaga = 0) LOOP
```

```
            -- losowanie pacjenta
      SET v_wylosowane_id_pacjenta = fn_losuj_liczbe(v_min_id_pacjenta, v_max_id_pacjenta);

            -- jezeli pacjent z wylosowanym id_pacjenta istnieje to ustawienie flagi na 1 co
spowoduje zakonczenie pętli

 IF EXISTS (SELECT id_pacjenta FROM Pacjent WHERE id_pacjenta = v_wylosowane_id_pacjenta) THEN
        SET v_flaga = 1;
      END IF;
    END LOOP;

        -- zwrocenie wartości wylosowanego id
    RETURN v_wylosowane_id_pacjenta;
  ELSE
        -- zwrócenie -1 w przypadku błędu
    RETURN -1;
  END IF;
END;
```

Funkcja *fn_losuj_pacjenta* zarazem jak umieszczona poniżej funkcja *fn_losuj_usluge* działa na podobnej zasadzie, dlatego pominięte zostaną komentarze dot. funkcji *fn_losuj_usluge*. Działanie obu funkcji polega na wylosowaniu takiej wartości klucza głównego odpowiedniej tabeli (Pacjent lub Usluga), aby istniał on w systemie. W przypadku błędu funkcja zwraca wartość -1.

```
CREATE OR REPLACE FUNCTION fn_losuj_usluge()
RETURNS INTEGER
BEGIN
  DECLARE v_wylosowane_id_uslugi INTEGER;
  DECLARE v_min_id_uslugi INTEGER;
  DECLARE v_max_id_uslugi INTEGER;
  DECLARE v_flaga INTEGER;

  SELECT MIN(id_uslugi) INTO v_min_id_uslugi FROM Usluga;
  SELECT MAX(id_uslugi) INTO v_max_id_uslugi FROM Usluga;

  IF ((v_min_id_uslugi IS NOT NULL) AND (v_max_id_uslugi IS NOT NULL)) THEN

    SET v_flaga = 0;

    WHILE (v_flaga = 0) LOOP
      SET v_wylosowane_id_uslugi = fn_losuj_liczbe(v_min_id_uslugi, v_max_id_uslugi);

      IF EXISTS (SELECT id_uslugi FROM Usluga WHERE id_uslugi = v_wylosowane_id_uslugi) THEN
        SET v_flaga = 1;
      END IF;
    END LOOP;
```

```
    RETURN v_wylosowane_id_uslugi;
  ELSE
    RETURN -1;
  END IF;
END;
```

Opis pominięty – patrz: komentarze w funkcji *fn_losuj_pacjenta*.

```
CREATE OR REPLACE PROCEDURE pr_dodaj_zabieg_do_wizyty
(v_id_wizyty INTEGER)
BEGIN
    -- zadeklarowanie zmiennych
    DECLARE v_id_uslugi INTEGER;
    -- wylosowanie uslugi
    SET v_id_uslugi = fn_losuj_usluge();

    -- dodanie zabiegu do wizyty
    INSERT INTO Zabieg (id_wizyty, id_uslugi, cena_wykonania)
        VALUES (v_id_wizyty, v_id_uslugi, 0);
END;
```

Procedura odpowiedzialna za dodanie zabiegu do wizyty o kluczu głównym podanym w parametrze.

Ostatnią procedurą, której implementacje podano poniżej, jest procedura, która wywoływana jest co dwie godziny dzięki zdarzeniu (ang. event).

```
CREATE OR REPLACE PROCEDURE pr_generowanie_transakcji()
BEGIN
    -- wywolanie procedury dodawania 1 wizyty dla obecnego dnia
    CALL pr_dodaj_wizyty(CURRENT DATE, CURRENT DATE, 1);

    -- odswiezenie widokow zmaterializowanych
    REFRESH MATERIALIZED VIEW Zyski_ze_stycznia;
    REFRESH MATERIALIZED VIEW Zyski_z_lutego;
    REFRESH MATERIALIZED VIEW Liczba_wykonanych_uslug_w_2016
END;
```

Odpowiedzialna jest ona za dodawanie wizyty i odświeżanie widoków zmaterializowanych, o czym więcej w następnym nagłówku.

Temat generatora transakcji został wyczerpany. Podsumowując, generator dodaje wizytę w dniu, w którym został wyzwolony. Wizyta dodawana jest między 9:00 a 16:00 o pełnych godzinach lub 30 minut po danej godzinie. Oznacza to, że generator może dodać maksymalnie 15 wizyt dziennie. W przypadku braku możliwości wylosowania wolnego terminu zakończa on swoją pracę.

Skrypty wdrożeniowe instalujące i odinstalowywujące zrealizowany projekt

Środowisko Sybase

create_script_sybase.sql

Skrypt ten zawiera procedury oraz funkcję opisane w poprzednim nagłówku „generator danych” oraz ponadto:

```
CREATE TABLE Pacjent
 (
  id_pacjenta      INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  imie             VARCHAR (30) NOT NULL ,
  nazwisko         VARCHAR (40) NOT NULL ,
  plec             VARCHAR (1) NOT NULL ,
  ulica            VARCHAR (40) ,
  budynek          VARCHAR (10) ,
  nr_lokalu        INTEGER ,
  kod_pocztowy     VARCHAR (10) ,
  miejscowosc      VARCHAR (40) ,
  data_dodania     DATE NOT NULL ,
  id_polecajacego  INTEGER NOT NULL
 ) ;
ALTER TABLE Pacjent  ADD CONSTRAINT Pacjent_PK  PRIMARY KEY ( id_pacjenta ) ;


CREATE TABLE Polecajacy
 (
  id_polecajacego  INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  imie             VARCHAR (30) NOT NULL ,
  nazwisko         VARCHAR (40) NOT NULL
 ) ;
ALTER TABLE Polecajacy  ADD CONSTRAINT Polecajacy_PK  PRIMARY KEY ( id_polecajacego ) ;


CREATE TABLE TypUslugi
 (
  id_typuuslugi  INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  nazwa          VARCHAR (100) NOT NULL
 ) ;
ALTER TABLE TypUslugi  ADD CONSTRAINT TypUslugi_PK  PRIMARY KEY ( id_typuuslugi ) ;
```

```
CREATE TABLE Usluga
 (
  id_uslugi INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  nazwa     VARCHAR (50) NOT NULL ,
  cena FLOAT (2) NOT NULL ,
  id_typuuslugi INTEGER NOT NULL
 ) ;
ALTER TABLE Usluga ADD CONSTRAINT Usluga_PK PRIMARY KEY ( id_uslugi ) ;


CREATE TABLE Wizyta
 (
  id_wizyty    INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  id_pacjenta  INTEGER NOT NULL ,
  data_godzina TIMESTAMP NOT NULL ,
  wartosc FLOAT (2) NOT NULL
 ) ;
ALTER TABLE Wizyta ADD CONSTRAINT Wizyta_PK PRIMARY KEY ( id_wizyty ) ;


CREATE TABLE Zabieg
 (
  id_wizyty INTEGER NOT NULL ,
  kolejnosc INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  id_uslugi INTEGER NOT NULL ,
  cena_wykonania FLOAT (2) NOT NULL
 ) ;
ALTER TABLE Zabieg ADD CONSTRAINT Zabieg_PK PRIMARY KEY ( kolejnosc, id_wizyty ) ;
```

Zdania odpowiedzialne za tworzenie struktury tabel wraz z ich kluczami głównymi (prostymi lub złożonymi). Jako domyślne wartości kluczów głównych tabel przypisano AUTOINCREMENT. Jest to mechanizm auto inkrementacji klucza głównego. Pobierany jest wartość maksymalnego klucza głównego oraz dodawane jest do niej jedność podczas wstawiania rekordu do danej tabeli bez podania wartości kolumny klucza głównego. Ułatwia to bardzo codzienną pracę z bazą danych. Wyzwanie te można by było rozwązać także mechanizmem sekwencji lub wyzwalaczem.

```
ALTER TABLE Pacjent ADD CONSTRAINT Pacjent_Polecajacy_FK FOREIGN KEY ( id_polecajacego ) REFERENCE
S Polecajacy ( id_polecajacego ) ;

ALTER TABLE Usluga ADD CONSTRAINT Usluga_TypUslugi_FK FOREIGN KEY ( id_typuuslugi ) REFERENCES TypU
slugi ( id_typuuslugi ) ;

ALTER TABLE Wizyta ADD CONSTRAINT Wizyta_Pacjent_FK FOREIGN KEY ( id_pacjenta ) REFERENCES Pacjent
( id_pacjenta ) ;
```

```
ALTER TABLE Zabieg ADD CONSTRAINT Zabieg_Usluga_FK FOREIGN KEY ( id_uslugi ) REFERENCES Usluga ( id
_uslugi ) ;

ALTER TABLE Zabieg ADD CONSTRAINT Zabieg_Wizyta_FK FOREIGN KEY ( id_wizyty ) REFERENCES Wizyta ( id
_wizyty ) ;
```

Zdania dodające więzy (klucze obce) między odpowiednimi tabelami (wynika to z modelu relacyjnego).

```
-- wyzwalacz wywoływany przed dodaniem rekordu do tabeli Zabieg
-- odpowiedzialny za wypełnienie ceny zabiegu ceną usługi
CREATE OR REPLACE TRIGGER tr_BEF_INS_zabieg
BEFORE INSERT ON Zabieg
REFERENCING NEW AS new_rec
FOR EACH ROW
BEGIN
    DECLARE v_cena_uslugi FLOAT;
    -- pobranie ceny uslugi do zmiennej v_cena_uslugi dla uslugi o id rownym polu id_uslugi ze
struktury new_rec
    SELECT cena INTO v_cena_uslugi FROM Usluga WHERE id_uslugi = new_rec.id_uslugi;

    SET new_rec.cena_wykonania = v_cena_uslugi;
END;

-- wyzwalacz wywoływany po dodaniu rekordu do tabeli Zabieg
-- odpowiedzialny za zaktualizowanie wartosci Wizyty
CREATE OR REPLACE TRIGGER tr_AFT_INS_zabieg
AFTER INSERT ON Zabieg
REFERENCING NEW AS new_rec
FOR EACH ROW
BEGIN
    UPDATE Wizyta

 SET wartosc = (SELECT SUM(cena_wykonania) FROM Zabieg WHERE id_wizyty = new_rec.id_wizyty)
        WHERE id_wizyty = new_rec.id_wizyty;
END;

-- wyzwalacz wywoływany przed zaktualizowaniem rekordu tabeli Zabieg
-- odpowiedzialny za wypełnienie ceny zabiegu ceną usługi
CREATE OR REPLACE TRIGGER tr_BEF_UPD_zabieg
BEFORE UPDATE ON Zabieg
REFERENCING NEW AS new_rec
FOR EACH ROW
BEGIN
    DECLARE v_cena_uslugi FLOAT;
    -- pobranie ceny uslugi do zmiennej v_cena_uslugi dla uslugi o id rownym polu id_uslugi ze
struktury new_rec
```

```
    SELECT cena INTO v_cena_uslugi FROM Usluga WHERE id_uslugi = new_rec.id_uslugi;

    SET new_rec.cena_wykonania = v_cena_uslugi;
END;

-- wyzwalacz wywoływany po zaktualizowaniu rekordu tabeli Zabieg
-- odpowiedzialny za zaktualizowanie wartosci Wizyty
CREATE OR REPLACE TRIGGER tr_AFT_UPD_zabieg
AFTER UPDATE ON Zabieg
REFERENCING NEW AS new_rec
FOR EACH ROW
BEGIN
    UPDATE Wizyta

 SET wartosc = (SELECT SUM(cena_wykonania) FROM Zabieg WHERE id_wizyty = new_rec.id_wizyty)
        WHERE id_wizyty = new_rec.id_wizyty;
END;

-- wyzwalacz wywoływany po usunięciu rekordu z tabeli Zabieg
-- odpowiedzialny za zaktualizowanie wartosci Wizyty
CREATE OR REPLACE TRIGGER tr_AFT_DEL_zabieg
AFTER DELETE ON Zabieg
REFERENCING OLD AS old_rec
FOR EACH ROW
BEGIN
    DECLARE v_wartosc FLOAT;

    SET v_wartosc = 0;

    -- sprawdzenie czy liczba zabiegow dla danej wizyty jest wieksza od 0
    IF (SELECT COUNT(*) FROM Zabieg WHERE id_wizyty = old_rec.id_wizyty) > 0 THEN
        -- jezeli tak to pobranie sumy cen wykonania zabiegow danej wizyty

 SELECT SUM(cena_wykonania) INTO v_wartosc FROM Zabieg WHERE id_wizyty = old_rec.id_wizyty;
    END IF;

    UPDATE Wizyta
        SET wartosc = v_wartosc
        WHERE id_wizyty = old_rec.id_wizyty;
END;
```

```
-- wyzwalacz wywoływany po usunięciu rekordu z tabeli Wizyta
-- odpowiedzialny za usuwanie zabiegow danej wizyty jezeli usuwamy wizyte
CREATE OR REPLACE TRIGGER tr_AFT_DEL_wizyta
AFTER DELETE ON Wizyta
REFERENCING OLD AS old_rec
FOR EACH ROW
BEGIN
    DELETE FROM Zabieg WHERE id_wizyty = old_rec.id_wizyty;
END;
```

Zdania dodające wyzwalacze wymagane do poprawnego funkcjonowania bazy danych (aktualizacja pól w wierszach, usuwanie wierszy).

```
-- widok pokazujace zestawienie dzisiejszych wizyt (godzina, imie i nazwisko pacjenta, lista
zabiegow, wartosc wizyty)
CREATE OR REPLACE VIEW Wizyty_dzisiaj AS
    SELECT DATEFORMAT(w.data_godzina, 'HH:NN') AS [Godzina], p.imie ||'
'|| p.nazwisko AS 'Pacjent', LIST(u.nazwa, ', ') AS [Zabiegi], ROUND(w.wartosc, 2) AS [Wartosc]
    FROM Wizyta AS w
    LEFT JOIN Pacjent AS p ON (p.id_pacjenta = w.id_pacjenta)
    JOIN Zabieg AS z ON (z.id_wizyty = w.id_wizyty)
    JOIN Usluga AS u ON (u.id_uslugi = z.id_uslugi)
    WHERE DATE(w.data_godzina) = DATE(CURRENT DATE)
    GROUP BY [Godzina], [Pacjent], [Wartosc]
    ORDER BY [Godzina] ASC;

-- widok pokazujacy zestawienie dzisiejszych wizyt z podsumowaniem ich wartosci
CREATE OR REPLACE VIEW Wizyty_dzisiaj_z_podsumowaniem AS
    SELECT * FROM Wizyty_dzisiaj
    UNION

 SELECT NULL, NULL, 'Razem', (SELECT ROUND(SUM(w2.wartosc), 2) FROM Wizyta AS w2 WHERE DATE(w2.data
_godzina) = DATE(CURRENT DATE));
```

```sql
-- widok pokazujacy liczbe zajetych terminow na nastepne 7 dni
CREATE OR REPLACE VIEW Liczba_zajetych_terminow_na_nastepne_7_dni AS

 SELECT DATEFORMAT(data_godzina, 'Mmmmmmmmmm') AS [Miesiac], DATEFORMAT(data_godzina, 'dd') AS [D
zien],
  COUNT(*) AS [Liczba zajetych terminow],
  (CASE
    WHEN (15 - COUNT(*) > 0) THEN (15 - COUNT(*))
    ELSE 0
  END) AS [Liczba wolnych terminow]
  FROM Wizyta
  WHERE DATE(data_godzina) > DATE(CURRENT DATE) AND DATE(data_godzina) <= DATE(CURRENT DATE) + 7
  GROUP BY [Miesiac], [Dzien];

-- widok zmaterializowany ukazujacy liczbe uslug wykonanych w 2016r. z podzialem na miesiace
CREATE MATERIALIZED VIEW Liczba_wykonanych_uslug_w_2016 AS
    SELECT DATEFORMAT(w.data_godzina, 'Mmmmmmmmmm') AS [Miesiac], u.nazwa AS [Nazwa
uslugi], COUNT(*) AS [Liczba wykonanych zabiegow]
    FROM Zabieg AS z
    LEFT JOIN Wizyta AS w ON (w.id_wizyty = z.id_wizyty)
    LEFT JOIN Usluga AS u ON (u.id_uslugi = z.id_uslugi)
    WHERE DATEFORMAT(w.data_godzina, 'YYYY') = '2016'
    GROUP BY [Miesiac], [Nazwa uslugi]
  ORDER BY DATE([Miesiac]) ASC, [Nazwa uslugi] ASC;

-- widok zmaterializowany pokazujacy dzienny zysk dzienny z liczba wizyt ze stycznia
CREATE MATERIALIZED VIEW Zyski_ze_stycznia AS

 SELECT DATEFORMAT(data_godzina, 'Mmmmmmmmmm') AS [Miesiac], DATEFORMAT(data_godzina, 'dd') AS [D
zien], COUNT(id_wizyty) AS [Liczba wizyt], ROUND(SUM(wartosc), 2) AS [Zysk]
    FROM Wizyta
    WHERE DATE(data_godzina) >= DATE('2016-01-01') AND DATE(data_godzina) < DATE('2016-02-01')
    GROUP BY [Miesiac], [Dzien]
    UNION
    SELECT NULL, 'Razem', NULL, (SELECT ROUND(SUM(w2.wartosc), 2) FROM Wizyta AS w2)
    ORDER BY [Dzien] ASC;
```

```sql
-- widok zmaterializowany pokazujacy dzienny zysk dzienny z liczba wizyt z lutego
CREATE MATERIALIZED VIEW Zyski_z_lutego AS

 SELECT DATEFORMAT(data_godzina, 'Mmmmmmmmmm') AS [Miesiac], DATEFORMAT(data_godzina, 'dd') AS [D
zien], COUNT(id_wizyty) AS [Liczba wizyt], ROUND(SUM(wartosc), 2) AS [Zysk]
    FROM Wizyta
    WHERE DATE(data_godzina) >= DATE('2016-02-01') AND DATE(data_godzina) < DATE('2016-03-01')
    GROUP BY [Miesiac], [Dzien]
    UNION
    SELECT NULL, 'Razem', NULL, (SELECT ROUND(SUM(w2.wartosc), 2) FROM Wizyta AS w2)
    ORDER BY [Dzien] ASC;
```

Zdania odpowiedzialne za dodawanie widoków klasycznych i zmaterializowanych służących do prezentowania dynamiki działania firmy. Perspektywy zmaterializowane odświeżane są w procedurze *pr_generowanie_transakcji*.

drop_script_sybase.sql

```
/*************************************
* Kasowanie widokow klasycznych i zmaterializowanych
*************************************/
DROP VIEW Wizyty_dzisiaj;

DROP VIEW Wizyty_dzisiaj_z_podsumowaniem;

DROP VIEW Liczba_zajetych_terminow_na_nastepne_7_dni;

DROP materialized VIEW Zyski_ze_stycznia;

DROP materialized VIEW Zyski_z_lutego;

DROP materialized VIEW Liczba_wykonanych_uslug_w_2016;

/*************************************
* Kasowanie triggerow
*************************************/
DROP TRIGGER tr_BEF_INS_zabieg;

DROP TRIGGER tr_AFT_INS_zabieg;

DROP TRIGGER tr_BEF_UPD_zabieg;

DROP TRIGGER tr_AFT_UPD_zabieg;

DROP TRIGGER tr_AFT_DEL_zabieg;

DROP TRIGGER tr_AFT_DEL_wizyta;

/*************************************
* Kasowanie więzów integralności
*************************************/
ALTER TABLE Pacjent
    DELETE CONSTRAINT Pacjent_Polecajacy_FK;

ALTER TABLE Usluga
    DELETE CONSTRAINT Usluga_TypUslugi_FK;

ALTER TABLE Wizyta
    DELETE CONSTRAINT Wizyta_Pacjent_FK;
```

```
ALTER TABLE Zabieg
    DELETE CONSTRAINT Zabieg_Usluga_FK;

ALTER TABLE Zabieg
    DELETE CONSTRAINT Zabieg_Wizyta_FK;

/*************************************
* Kasowanie tabel i perspektyw
*************************************/
DROP TABLE Pacjent;

DROP TABLE Polecajacy;

DROP TABLE TypUslugi;

DROP TABLE Usluga;

DROP TABLE Wizyta;

DROP TABLE Zabieg;

/*************************************
* Kasowanie funkcji i procedury własnej
*************************************/
DROP FUNCTION fn_losuj_liczbe;

DROP FUNCTION fn_losuj_pacjenta;

DROP FUNCTION fn_losuj_usluge;

DROP PROCEDURE pr_dodaj_zabieg_do_wizyty;

DROP PROCEDURE pr_dodaj_wizyty;

DROP PROCEDURE pr_dodaj_wizyte;
```

import_script_sybase.sql

```
/*************************************************
* Ładowanie danych z plików ASCI
*************************************************/
LOAD INTO TABLE TypUslugi USING CLIENT file 'D:/Studia/SEM
III/BD/laboratorium/Projekt/pliki_sql/typuslugi.dat'
  format 'TEXT' quotes ON
  ORDER off escapes ON
  CHECK constraints off computes off
  strip off delimited BY ','
  encoding 'utf-8';

LOAD INTO TABLE Usluga USING CLIENT file 'D:/Studia/SEM
III/BD/laboratorium/Projekt/pliki_sql/usluga.dat'
  format 'TEXT' quotes ON
  ORDER off escapes ON
  CHECK constraints off computes off
  strip off delimited BY ','
```

```
  encoding 'utf-8';

LOAD INTO TABLE Polecajacy USING CLIENT file 'D:/Studia/SEM
III/BD/laboratorium/Projekt/pliki_sql/polecajacy.dat'
  format 'TEXT' quotes ON
  ORDER off escapes ON
  CHECK constraints off computes off
  strip off delimited BY ','
  encoding 'utf-8';

LOAD INTO TABLE Pacjent USING CLIENT file 'D:/Studia/SEM
III/BD/laboratorium/Projekt/pliki_sql/pacjent.dat'
  format 'TEXT' quotes ON
  ORDER off escapes ON
  CHECK constraints off computes off
  strip off delimited BY ','
  encoding 'utf-8';

commit;
```

pacjent.dat

```
1,'Daria','Kicaj','K','al. Jana Pawła III','23a',5,'01-100','Warszawa',2016-01-28,1
2,'Alfred','Tragarz','M','pl. Konstytucji','1',,'04-839','Świnoujście',2016-01-28,2
3,'Konstantyn','Jeziorny','M','ul. Zambrowska','2',,'64-839','Brzegi dolne',2016-01-28,3
4,'Agnieszka','Dąbrowska','K','ul. Podutopia','9b',2,'23-221','Brok',2016-01-28,4
5,'Dominika','Wojcicka','K','ul. Kwiatowa','12',5,'99-999','Małkinia Dolna',2016-01-28,5
6,'Michał','Konieczko','M','ul. Strażnicza','15',,'89-432','Gdynia',2016-01-28,6
7,'Janina','Tvnowska','K','ul. Woronicza','1',,'13-144','Sopot',2016-01-28,7
8,'Jarosława','Kaczyńska','K','ul. Na wspólnej','15',290,'01-100','Kraków',2016-01-28,8
9,'Lila','Hozier','K','al. Youtubea','6p',,'65-064','Nibylandia',2016-01-28,5
10,'Iwona','Górska','K','pl. Trzech krzyży','11',2,'11-233','Mińsk Mazowiecki',2016-01-28,4
```

usluga.dat

```
1,'Masaż relaksacyjny',19.98,1
2,'Masaż leczniczy',29.98,1
3,'Masaż limfatyczny',24.98,1
4,'Masaż gorącymi kamieniami',20,1
5,'Masaż gorącą czekoladą',45,1
6,'Masaż sportowy',15,1
7,'Rozbijanie złogów',80,2
8,'Jonoforeza z NaCL',30,4
9,'Leczenie nerwoboli',45.54,3
10,'Drenaż limfatyczny',65.33,5
```

polecajacy.dat

```
1,'Jan','Kowalski'
2,'Adam','Abacki'
3,'Aldona','Trętowska'
4,'Adam','Borewicz'
5,'Marcin','Mieczkowski'
6,'Adrian','Kimowicz'
7,'Jarosław','Kaniewski'
8,'Damian','Cichecki'
```

typuslugi.dat

```
1,'masaż'
2,'ultradźwięki'
3,'laser'
4,'prądy diadynamiczne'
5,'drenaż'
```

Środowisko Oracle

create_script_oracle.sql

```
CREATE TABLE Pacjent
 (
  id_pacjenta      INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  imie             VARCHAR2 (30) NOT NULL ,
  nazwisko         VARCHAR2 (40) NOT NULL ,
  plec             VARCHAR2 (1) NOT NULL ,
  ulica            VARCHAR2 (40) ,
  budynek          VARCHAR2 (10) ,
  nr_lokalu        INTEGER ,
  kod_pocztowy     VARCHAR2 (10) ,
  miejscowosc      VARCHAR2 (40) ,
  data_dodania     DATE NOT NULL ,
  id_polecajacego  INTEGER NOT NULL
 ) ;
ALTER TABLE Pacjent ADD CONSTRAINT Pacjent_PK PRIMARY KEY ( id_pacjenta ) ;


CREATE TABLE Polecajacy
 (
  id_polecajacego  INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  imie             VARCHAR2 (30) NOT NULL ,
  nazwisko         VARCHAR2 (40) NOT NULL
 ) ;
ALTER TABLE Polecajacy ADD CONSTRAINT Polecajacy_PK PRIMARY KEY ( id_polecajacego ) ;
```

```
CREATE TABLE TypUslugi
 (
  id_typuuslugi  INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  nazwa          VARCHAR2 (100) NOT NULL
 ) ;
ALTER TABLE TypUslugi ADD CONSTRAINT TypUslugi_PK PRIMARY KEY ( id_typuuslugi ) ;


CREATE TABLE Usluga
 (
  id_uslugi  INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  nazwa      VARCHAR2 (50) NOT NULL ,
  cena FLOAT (2) NOT NULL ,
  id_typuuslugi  INTEGER NOT NULL
 ) ;
ALTER TABLE Usluga ADD CONSTRAINT Usluga_PK PRIMARY KEY ( id_uslugi ) ;


CREATE TABLE Wizyta
 (
  id_wizyty     INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  id_pacjenta  INTEGER NOT NULL ,
  data_godzina TIMESTAMP NOT NULL ,
  wartosc FLOAT (2) NOT NULL
 ) ;
ALTER TABLE Wizyta ADD CONSTRAINT Wizyta_PK PRIMARY KEY ( id_wizyty ) ;


CREATE TABLE Zabieg
 (
  id_wizyty INTEGER NOT NULL ,
  kolejnosc INTEGER NOT NULL DEFAULT AUTOINCREMENT,
  id_uslugi INTEGER NOT NULL ,
  cena_wykonania FLOAT (2) NOT NULL
 ) ;
ALTER TABLE Zabieg ADD CONSTRAINT Zabieg_PK PRIMARY KEY ( kolejnosc, id_wizyty ) ;


ALTER TABLE Pacjent ADD CONSTRAINT Pacjent_Polecajacy_FK FOREIGN KEY ( id_polecajacego ) REFERENCE
S Polecajacy ( id_polecajacego ) ;

ALTER TABLE Usluga ADD CONSTRAINT Usluga_TypUslugi_FK FOREIGN KEY ( id_typuuslugi ) REFERENCES TypU
slugi ( id_typuuslugi ) ;

ALTER TABLE Wizyta ADD CONSTRAINT Wizyta_Pacjent_FK FOREIGN KEY ( id_pacjenta ) REFERENCES Pacjent
( id_pacjenta ) ;

ALTER TABLE Zabieg ADD CONSTRAINT Zabieg_Usluga_FK FOREIGN KEY ( id_uslugi ) REFERENCES Usluga ( id
_uslugi ) ;

ALTER TABLE Zabieg ADD CONSTRAINT Zabieg_Wizyta_FK FOREIGN KEY ( id_wizyty ) REFERENCES Wizyta ( id
_wizyty ) ;
```

drop_script_oracle.sql

```
/*************************************
* Kasowanie więzów integralności
*************************************/
ALTER TABLE Pacjent
    DELETE CONSTRAINT Pacjent_Polecajacy_FK;

ALTER TABLE Usluga
    DELETE CONSTRAINT Usluga_TypUslugi_FK;

ALTER TABLE Wizyta
    DELETE CONSTRAINT Wizyta_Pacjent_FK;

ALTER TABLE Zabieg
    DELETE CONSTRAINT Zabieg_Usluga_FK;

ALTER TABLE Zabieg
    DELETE CONSTRAINT Zabieg_Wizyta_FK;

/*************************************
* Kasowanie tabel i perspektyw
*************************************/
DROP TABLE Zabieg;

DROP TABLE Usluga;

DROP TABLE Wizyta;

DROP TABLE Pacjent;

DROP TABLE Polecajacy;

DROP TABLE TypUslugi;
```

import_script_oracle.sql

```
INSERT INTO Polecajacy (id_polecajacego,imie,nazwisko) VALUES(1,'Jan','Kowalski');
INSERT INTO Polecajacy (id_polecajacego,imie,nazwisko) VALUES(2,'Adam','Abacki');
INSERT INTO Polecajacy (id_polecajacego,imie,nazwisko) VALUES(3,'Aldona','Trętowska');
INSERT INTO Polecajacy (id_polecajacego,imie,nazwisko) VALUES(4,'Adam','Borewicz');
INSERT INTO Polecajacy (id_polecajacego,imie,nazwisko) VALUES(5,'Marcin','Mieczkowski');
INSERT INTO Polecajacy (id_polecajacego,imie,nazwisko) VALUES(6,'Adrian','Kimowicz');
INSERT INTO Polecajacy (id_polecajacego,imie,nazwisko) VALUES(7,'Jarosław','Kaniewski');
INSERT INTO Polecajacy (id_polecajacego,imie,nazwisko) VALUES(8,'Damian','Cichecki');
```

```
INSERT INTO TypUslugi (id_typuuslugi,nazwa) VALUES(1,'masaż');
INSERT INTO TypUslugi (id_typuuslugi,nazwa) VALUES(2,'ultradźwięki');
INSERT INTO TypUslugi (id_typuuslugi,nazwa) VALUES(3,'laser');
INSERT INTO TypUslugi (id_typuuslugi,nazwa) VALUES(4,'prądy diadynamiczne');
INSERT INTO TypUslugi (id_typuuslugi,nazwa) VALUES(5,'drenaż');
```

```
INSERT INTO Pacjent (id_pacjenta,imie,nazwisko,plec,ulica,budynek,nr_lokalu,kod_pocztowy,miejscowosc,data_dodania,id_polecajacego) VALUES(1,'Daria','Kicaj','K','al. Jana Pawła III','23a',5,'01-100','Warszawa','2016-01-28',1);
INSERT INTO Pacjent (id_pacjenta,imie,nazwisko,plec,ulica,budynek,nr_lokalu,kod_pocztowy,miejscowosc,data_dodania,id_polecajacego) VALUES(2,'Alfred','Tragarz','M','pl. Konstytucji','1',NULL,'04-839','Świnoujście','2016-01-28',2);
INSERT INTO Pacjent (id_pacjenta,imie,nazwisko,plec,ulica,budynek,nr_lokalu,kod_pocztowy,miejscowosc,data_dodania,id_polecajacego) VALUES(3,'Konstantyn','Jeziorny','M','ul. Zambrowska','2',NULL,'64-839','Brzegi dolne','2016-01-28',3);
INSERT INTO Pacjent (id_pacjenta,imie,nazwisko,plec,ulica,budynek,nr_lokalu,kod_pocztowy,miejscowosc,data_dodania,id_polecajacego) VALUES(4,'Agnieszka','Dąbrowska','K','ul. Podutopia','9b',2,'23-221','Brok','2016-01-28',4);
INSERT INTO Pacjent (id_pacjenta,imie,nazwisko,plec,ulica,budynek,nr_lokalu,kod_pocztowy,miejscowosc,data_dodania,id_polecajacego) VALUES(5,'Dominika','Wojcicka','K','ul. Kwiatowa','12',5,'99-999','Małkinia Dolna','2016-01-28',5);
INSERT INTO Pacjent (id_pacjenta,imie,nazwisko,plec,ulica,budynek,nr_lokalu,kod_pocztowy,miejscowosc,data_dodania,id_polecajacego) VALUES(6,'Michał','Konieczko','M','ul. Strażnicza','15',NULL,'89-432','Gdynia','2016-01-28',6);
INSERT INTO Pacjent (id_pacjenta,imie,nazwisko,plec,ulica,budynek,nr_lokalu,kod_pocztowy,miejscowosc,data_dodania,id_polecajacego) VALUES(7,'Janina','Tvnowska','K','ul. Woronicza','1',NULL,'13-144','Sopot','2016-01-28',7);
INSERT INTO Pacjent (id_pacjenta,imie,nazwisko,plec,ulica,budynek,nr_lokalu,kod_pocztowy,miejscowosc,data_dodania,id_polecajacego) VALUES(8,'Jarosława','Kaczyńska','K','ul. Na wspólnej','15',290,'01-100','Kraków','2016-01-28',8);
INSERT INTO Pacjent (id_pacjenta,imie,nazwisko,plec,ulica,budynek,nr_lokalu,kod_pocztowy,miejscowosc,data_dodania,id_polecajacego) VALUES(9,'Lila','Hozier','K','al. Youtubea','6p',NULL,'65-064','Nibylandia','2016-01-28',5);
INSERT INTO Pacjent (id_pacjenta,imie,nazwisko,plec,ulica,budynek,nr_lokalu,kod_pocztowy,miejscowosc,data_dodania,id_polecajacego) VALUES(10,'Iwona','Górska','K','pl. Trzech krzyży','11',2,'11-233','Mińsk Mazowiecki','2016-01-28',4);
```

```
INSERT INTO Usluga (id_uslugi,nazwa,cena,id_typuuslugi) VALUES(1,'Masaż relaksacyjny',19.98,1);
INSERT INTO Usluga (id_uslugi,nazwa,cena,id_typuuslugi) VALUES(2,'Masaż leczniczy',29.98,1);
INSERT INTO Usluga (id_uslugi,nazwa,cena,id_typuuslugi) VALUES(3,'Masaż limfatyczny',24.98,1);
INSERT INTO Usluga (id_uslugi,nazwa,cena,id_typuuslugi) VALUES(4,'Masaż gorącymi kamieniami',20.0,1);
INSERT INTO Usluga (id_uslugi,nazwa,cena,id_typuuslugi) VALUES(5,'Masaż gorącą czekoladą',45.0,1);
INSERT INTO Usluga (id_uslugi,nazwa,cena,id_typuuslugi) VALUES(6,'Masaż sportowy',15.0,1);
INSERT INTO Usluga (id_uslugi,nazwa,cena,id_typuuslugi) VALUES(7,'Rozbijanie złogów',80.0,2);
INSERT INTO Usluga (id_uslugi,nazwa,cena,id_typuuslugi) VALUES(8,'Jonoforeza z NaCL',30.0,4);
INSERT INTO Usluga (id_uslugi,nazwa,cena,id_typuuslugi) VALUES(9,'Leczenie nerwobóli',45.54,3);
INSERT INTO Usluga (id_uslugi,nazwa,cena,id_typuuslugi) VALUES(10,'Drenaż limfatyczny',65.33,5);
```

```
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(1,4,'2016-01-30
15:00',59.96);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(2,3,'2016-01-29
10:00',65.0);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(3,5,'2016-01-30
13:00',95.0);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(4,5,'2016-01-30
09:00',95.33);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(5,8,'2016-01-30
14:00',24.98);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(6,6,'2016-01-30
12:00',74.979996);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(7,10,'2016-01-29
10:30',74.979996);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(8,2,'2016-01-29
13:00',80.0);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(9,9,'2016-01-29
14:00',125.87);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(10,7,'2016-01-30
10:30',80.0);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(11,2,'2016-01-30
10:00',75.54);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(12,2,'2016-01-30
15:30',89.979996);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(13,6,'2016-01-29
15:30',110.0);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(14,7,'2016-01-30
11:30',29.98);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(15,7,'2016-01-29
11:30',15.0);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(16,4,'2016-01-30
11:00',64.96);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(17,9,'2016-01-29
12:30',40.0);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(18,6,'2016-01-29
13:30',59.98);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(19,3,'2016-01-29
15:00',45.0);
INSERT INTO Wizyta (id_wizyty,id_pacjenta,data_godzina,wartosc) VALUES(20,8,'2016-01-30
13:30',29.98);
```

```
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(1,1,2,29.98);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(1,2,2,29.98);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(2,3,4,20.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(2,4,5,45.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(3,5,6,15.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(3,6,7,80.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(4,7,10,65.33);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(4,8,8,30.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(5,9,3,24.98);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(6,10,3,24.98);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(6,11,8,30.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(6,12,4,20.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(7,13,2,29.98);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(7,14,8,30.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(7,15,6,15.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(8,16,4,20.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(8,17,8,30.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(8,18,8,30.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(9,19,10,65.33);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(9,20,9,45.54);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(9,21,6,15.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(10,22,7,80.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(11,23,9,45.54);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(11,24,8,30.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(12,25,6,15.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(12,26,5,45.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(12,27,2,29.98);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(13,28,7,80.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(13,29,8,30.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(14,30,2,29.98);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(15,31,6,15.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(16,32,4,20.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(16,33,3,24.98);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(16,34,1,19.98);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(17,35,4,20.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(17,36,4,20.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(18,37,2,29.98);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(18,38,8,30.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(19,39,5,45.0);
INSERT INTO Zabieg (id_wizyty,kolejnosc,id_uslugi,cena_wykonania) VALUES(20,40,2,29.98);
commit;
```

Instrukcja instalacji projektu i sprawdzenia jej poprawności

Aby zainstalować projekt należy w jednym ze środowisko (Oracle, Sybase) uruchomić oprogramowanie zarządzające bazą danych. W Oracle jest to SQL Developer, w Sybase jest to SQL Central.

Aby zainstalować projekt na serwerze Oracle po wejściu w SQL Developer i połączenie się z serwerem należy odpalić pliki create i import i uruchomić je w podanej kolejności.

Aby zainstalować projekt na serwerze Sybase należy po wejściu w Interactive SQL z menu File wybrać opcję Run script i uruchomić skrypty create i import w podanej kolejności.

Poprawność instalacji projektu można sprawdzić poprzez wykonanie podstawowych zapytań do wszystkich tabel (sprawdzających m.in. istnieje danych słownikowych) a także wywołując procedury i funkcję istniejące w bazie. Działanie wyzwalaczy można przetestować modyfikując, usuwając i dodając wiersze w tabelach Zabieg i Wizyta. Do wygenerowania widoków potrzebne są dane w tabelach Zabieg i Wizyta. Gdy takie dane istnieją – widoki zmaterializowane można odświeżyć i wykonać zapytanie do uzyskania danych ze wszystkich widoków, aby sprawdzić poprawność ich działania.

Wyniki działania perspektyw analitycznych w postaci zrzutów ekranowych

SQL Statements

```
1 select * from Wizyty_dzisiaj;
2
3
4
```

Results

	Godzina	Pacjent	Zabiegi	Wartosc
1	09:30	Dominika Wojcicka	Masaż relaksacyjny, Masaż leczniczy, Jonoforeza z NaCL	79,96
2	10:00	Alfred Tragarz	Masaż leczniczy, Rozbijanie złogów, Masaż relaksacyjny	129,96
3	10:30	Konstantyn Jeziorny	Masaż gorącymi kamieniami, Masaż limfatyczny	44,98
4	11:00	Daria Kicaj	Masaż sportowy, Masaż sportowy	30
5	12:00	Agnieszka Dąbrowska	Masaż sportowy, Masaż gorącą czekoladą	60
6	13:00	Janina Tvnowska	Masaż sportowy	15
7	13:30	Konstantyn Jeziorny	Masaż gorącą czekoladą, Rozbijanie złogów, Masaż limfatyczny	149,98
8	14:00	Agnieszka Dąbrowska	Rozbijanie złogów	80
9	14:30	Lila Hozier	Masaż relaksacyjny	19,98
10	15:00	Janina Tvnowska	Masaż gorącymi kamieniami, Masaż sportowy	35
11	15:30	Janina Tvnowska	Masaż gorącą czekoladą, Masaż leczniczy	74,98

SQL Statements

```
1 select * from Wizyty_dzisiaj_z_podsumowaniem;
2
3
4
```

Results

	Godzina	Pacjent	Zabiegi	Wartosc
1	09:30	Dominika Wojcicka	Masaż relaksacyjny, Masaż leczniczy, Jonoforeza z NaCL	79,96
2	10:00	Alfred Tragarz	Masaż leczniczy, Rozbijanie złogów, Masaż relaksacyjny	129,96
3	10:30	Konstantyn Jeziorny	Masaż gorącymi kamieniami, Masaż limfatyczny	44,98
4	11:00	Daria Kicaj	Masaż sportowy, Masaż sportowy	30
5	12:00	Agnieszka Dąbrowska	Masaż sportowy, Masaż gorącą czekoladą	60
6	13:00	Janina Tvnowska	Masaż sportowy	15
7	13:30	Konstantyn Jeziorny	Masaż gorącą czekoladą, Rozbijanie złogów, Masaż limfatyczny	149,98
8	14:00	Agnieszka Dąbrowska	Rozbijanie złogów	80
9	14:30	Lila Hozier	Masaż relaksacyjny	19,98
10	15:00	Janina Tvnowska	Masaż gorącymi kamieniami, Masaż sportowy	35
11	15:30	Janina Tvnowska	Masaż gorącą czekoladą, Masaż leczniczy	74,98
12	*(NULL)*	*(NULL)*	Razem	719,84

SQL Statements

```
1 select * from Liczba_zajetych_terminow_na_nastepne_7_dni;
2
3
4
```

Results

	Miesiac	Dzien	Liczba zajetych terminow	Liczba wolnych terminow
1	January	31	11	4
2	February	01	9	6
3	February	02	6	9
4	February	03	9	6
5	February	04	12	3
6	February	05	10	5
7	February	06	7	8

SQL Statements

```
1 select * from Zyski_ze_stycznia;
2
3
```

Results

	Miesiac	Dzien	Liczba wizyt	Zysk
1	January	01	9	766,81
2	January	02	7	420,75
3	January	03	9	526,04
4	January	04	8	460,44
5	January	05	9	496,52
6	January	06	9	641,29
7	January	07	8	516,02
8	January	08	8	436,93
9	January	09	9	610,42
10	January	10	10	856,5
11	January	11	6	405,25
12	January	12	8	857,01
13	January	13	9	856,68
14	January	14	9	501
15	January	15	7	547,14
16	January	16	7	520,79
17	January	17	5	365,5
18	January	18	8	454,92
19	January	19	6	440,6
20	January	20	7	706,14
21	January	21	7	455,5
22	January	22	12	815,63
23	January	23	6	475,48
24	January	24	11	782,2
25	January	25	12	1 162,...
26	January	26	8	485,77
27	January	27	8	772,64
28	January	28	9	732,18
29	January	29	9	761,91
30	January	30	11	719,84
31	January	31	11	705,9
32	*(NULL)*	Raz...	*(NULL)*	35 86...

SQL Statements

```
1 select * from Zyski_z_lutego;
2
3
```

Results

	Miesiac	Dzien	Liczba wizyt	Zysk
1	Febru...	01	9	775,6
2	Febru...	02	6	364,92
3	Febru...	03	9	721,41
4	Febru...	04	12	922,14
5	Febru...	05	10	706,27
6	Febru...	06	7	400,46
7	Febru...	07	9	736,16
8	Febru...	08	9	821,66
9	Febru...	09	10	676,35
10	Febru...	10	7	520,23
11	Febru...	11	6	264,92
12	Febru...	12	7	541,52
13	Febru...	13	8	420,44
14	Febru...	14	8	601,39
15	Febru...	15	6	264,94
16	Febru...	16	6	486,37
17	Febru...	17	8	574,92
18	Febru...	18	8	720,81
19	Febru...	19	5	466,14
20	Febru...	20	10	705,11
21	Febru...	21	6	580,44
22	Febru...	22	8	780,52
23	Febru...	23	6	521,66
24	Febru...	24	11	796,29
25	Febru...	25	10	876,64
26	Febru...	26	6	464,92
27	Febru...	27	9	677
28	Febru...	28	2	220
29	*(NULL)*	Raz...	*(NULL)*	35 86...

SQL Statements

```
1 select * from Liczba_wykonanych_uslug_w_2016;
2
3
```

Results

	Miesiac	Nazwa uslugi	Liczba wykonanych zabiegow
1	January	Drenaż limfatyczny	26
2	January	Jonoforeza z NaCL	64
3	January	Leczenie nerwoból	56
4	January	Masaż gorącą czekoladą	62
5	January	Masaż gorącymi kamieniami	51
6	January	Masaż leczniczy	52
7	January	Masaż limfatyczny	48
8	January	Masaż relaksacyjny	38
9	January	Masaż sportowy	64
10	January	Rozbijanie złogów	60
11	February	Drenaż limfatyczny	27
12	February	Jonoforeza z NaCL	45
13	February	Leczenie nerwoból	33
14	February	Masaż gorącą czekoladą	60
15	February	Masaż gorącymi kamieniami	58
16	February	Masaż leczniczy	50
17	February	Masaż limfatyczny	43
18	February	Masaż relaksacyjny	32
19	February	Masaż sportowy	40
20	February	Rozbijanie złogów	54