WOJSKOWA AKADEMIA TECHNICZNA

Bazy Danych

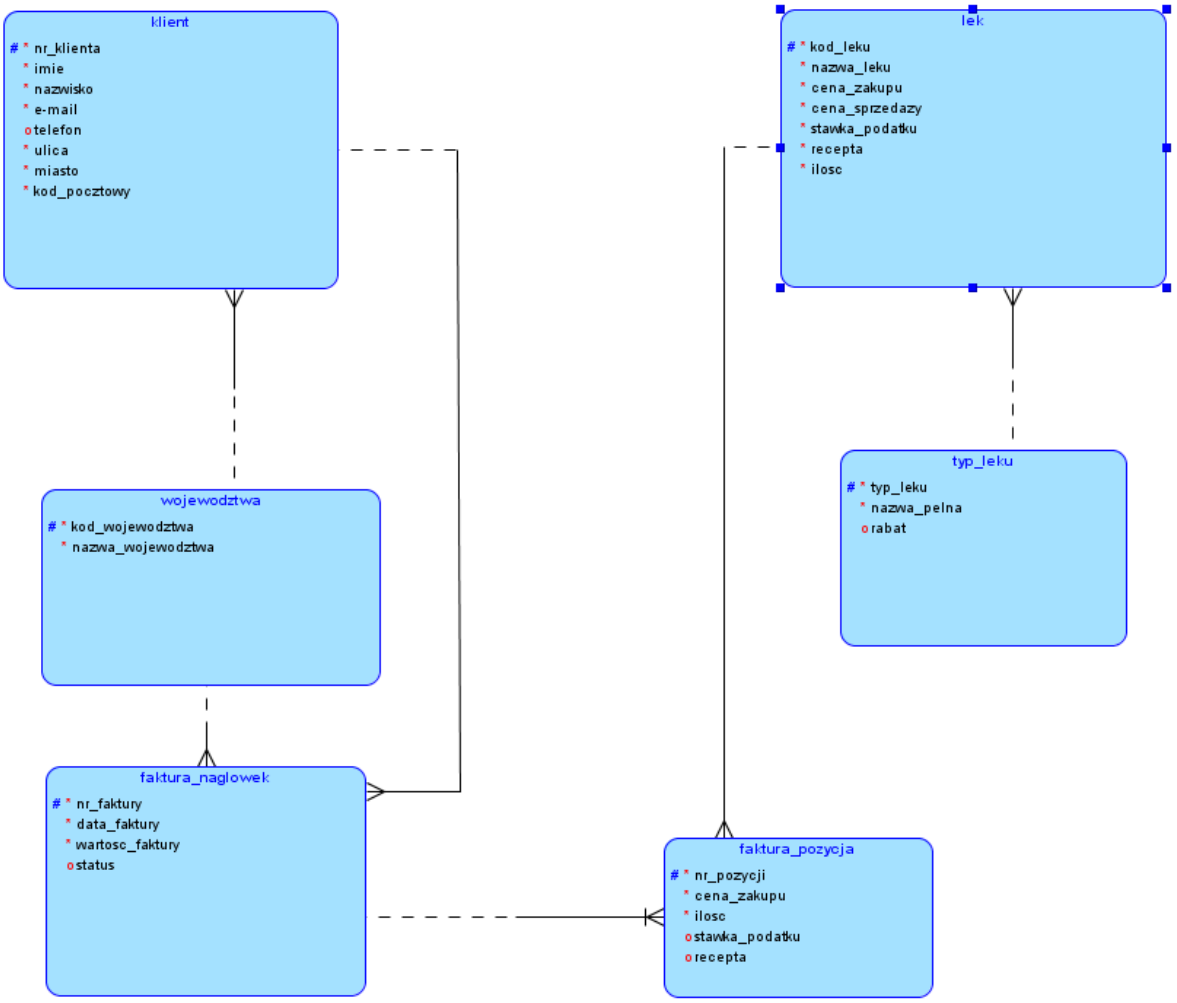
Sprawozdanie z projektu

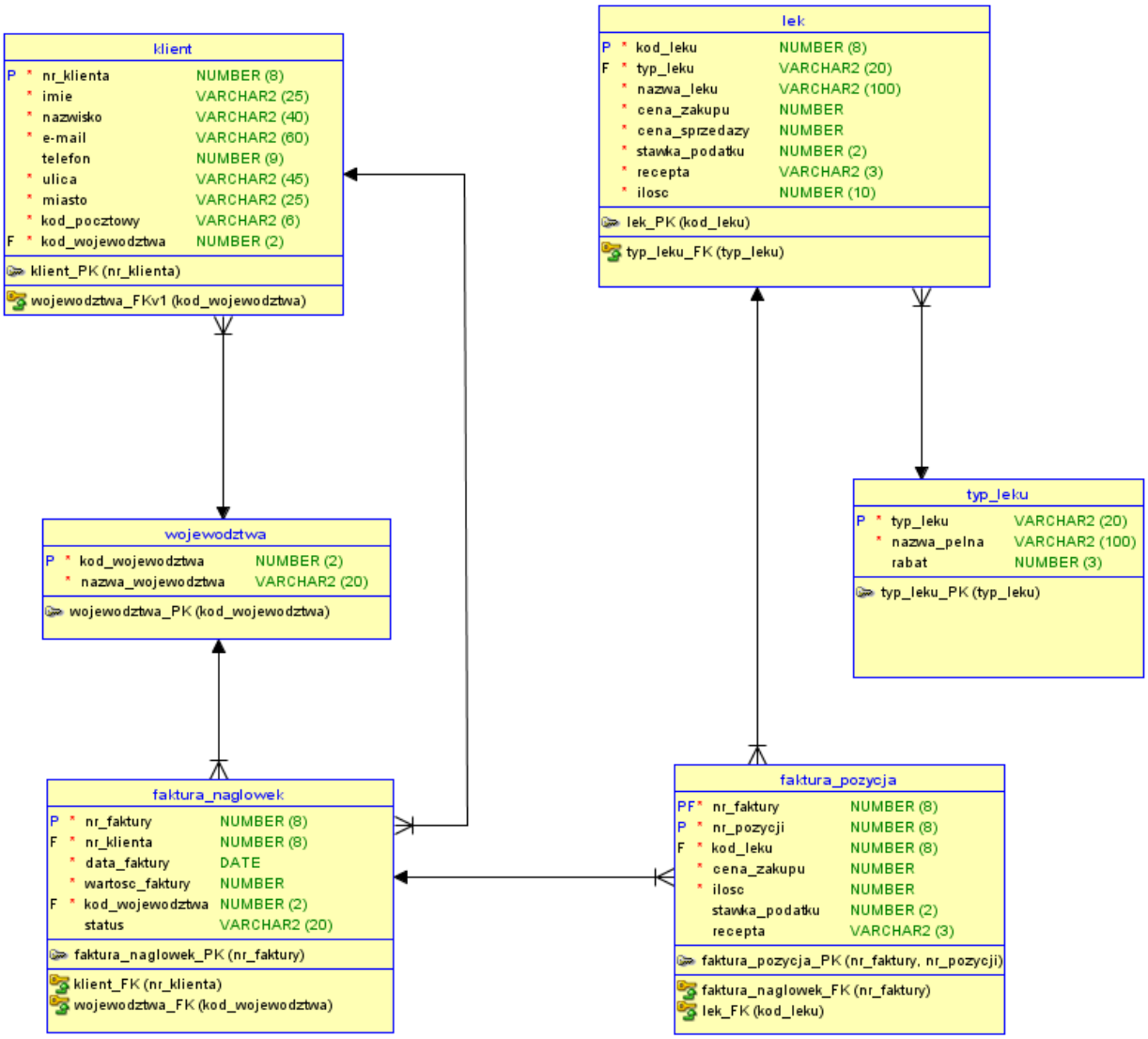
Autor: Jakub Złotnicki

1. Model bazy danych

Opracowany przeze mnie model bazy danych dotyczy apteki internetowej, nie posiadającej fizycznego oddziału i realizującej sprzedaż jedynie poprzez Internet. Apteka prowadzi ewidencje swoich klientów, gromadząc i przechowując ich dane w odpowiedniej tabeli. Model zaprojektowany został tak, aby zawierał elementy transakcji, w tym przypadku zamówienia internetowego. W skład modelu wchodzą również 2 tabele słownikowe: tabela przechowująca kod województwa (odpowiadający mu w realnym życiu TERYT) oraz jego nazwę. Istnieje również tabela słownikowa przechowująca informacje o typie danego leku, czy jest to lek zaliczany do grupy przeciwbólowych, czy może kardiologicznych.

2. Model logiczny i relacyjny





3. Oprogramowanie tworzące bazę danych / generator danych

➢ Procedura generująca dane: pr_pr_generuj:

```
create or replace PROCEDURE PR_PR_GENERUJ AS
    v_nr_faktury        pr_faktura_naglowek.nr_faktury%type;
    v_nr_klienta        pr_klient.nr_klienta%type;
    v_data_faktury      pr_faktura_naglowek.data_faktury%type;
    v_kod_wojewodztwa   pr_faktura_naglowek.kod_wojewodztwa%type;

    --pozycja
    v_kod_leku          pr_lek.kod_leku%type;
    v_ilosc             pr_lek.ilosc%type;


    v_losowa_ilosc_leku pr_lek.ilosc%type;
    v_czy_juz_istnieje  number;

    v_koniec_petli      number;
    v_data_petli        date;

    v_stop              boolean;
    v_ile_lekow         pr_lek.kod_leku%type;
    v_counter           number;

BEGIN
```

```
v_stop := false;

while v_stop <> true loop

select count(*) into v_koniec_petli
from pr_faktura_naglowek;

if v_koniec_petli = 0
then v_data_petli := '18/01/01';

else
    select max(data_faktury) into v_data_petli
    from pr_faktura_naglowek;
end if;


if v_data_petli + 5 <= '18/12/31'
then
    v_nr_klienta := fn_pr_daj_nr_klienta();

    insert into pr_faktura_naglowek(nr_klienta)
    values  (
                    v_nr_klienta
            );
    select nr_faktury into v_nr_faktury
    from pr_faktura_naglowek
    where status='OTWARTA';

    select data_faktury into v_data_faktury
    from pr_faktura_naglowek
    where nr_faktury = v_nr_faktury;

    --ile pozycji na fakturze

    select count(*) into v_ile_lekow
    from pr_lek;

    v_ile_lekow := round(dbms_random.value(1, v_ile_lekow));

    --
    v_counter :=0;
    while v_counter <> v_ile_lekow loop

    v_kod_leku := fn_pr_daj_kod_leku();

    select count(*) into v_czy_juz_istnieje
    from pr_faktura_pozycja
    where kod_leku = v_kod_leku and nr_faktury = v_nr_faktury;

    if v_czy_juz_istnieje = 0
    then
        v_ilosc := fn_pr_daj_ilosc_leku(v_kod_leku);

        if v_ilosc > 0
        then
            insert into pr_faktura_pozycja(nr_faktury, kod_leku, ilosc)
                values  (
                            v_nr_faktury, v_kod_leku, v_ilosc
                        );
        v_counter := v_counter+1;
```

```
            end if;
            end if;

            end loop;


            update pr_faktura_naglowek
            set status = 'ZAMKNIETA'
            where nr_faktury = v_nr_faktury;


            elsif v_data_petli + 5 > '18/12/31'
            then
                v_stop := true;
            end if;

            end loop;


  NULL;
END PR_PR_GENERUJ;
```

Opis procedury:

I. W pierwszej kolejności przy pomocy zmiennej v_koniec_petli sprawdzana jest ilość już istniejących faktur w tabeli pr_faktura_naglowek. Idea działania jest taka, aby w przypadku nie występowania jeszcze żadnej faktury w bazie, pierwszej fakturze przypisać datę jej realizacji na ’01/01/2018’. W wykonywanej pętli faktury będą generowane do czasu spełnienia warunku if v_data_petli + 5 <= '18/12/31', liczba 5 wynika z przedziału z jakiego losuje o ile dni dana faktura ma być starsza w triggerze dotyczącym tabeli pr_faktura_naglowek. Ideą tego procesu jest tworzenie faktur obejmujących okres całego roku kalendarzowego.

II. Kolejną kwestią jest wylosowanie klienta, posługując się stworzoną funkcją pr_daj_nr_klienta, zwracającej losowy numer z zakresu od 0 do ilości posiadanych klientów w bazie. Losowany jest również kod leku, oraz jego ilość, również przy pomocy stworzonych funkcji. Funkcja zwracająca losową ilość leku, wymaga podania kodu leku, poprzez który będzie losować ilość z odpowiedniego przedziału.

III. W tym kroku wykonywane jest polecenie insert wprowadzające wyznaczone dane do tabeli. Wartości takich jak nr_faktury, nie muszę podawać, gdyż uzupełnione zostaną przy pomocy odpowiedniej sekwencji,

IV. Ostatnim krokiem będzie zmiana statusu faktury (z default’owej ‘OTWARTA’) na ‘ZAMKNIETA’, co posłuży mi do dalszych operacji zaimplementowanych już w odpowiednich trigger’ach

➢ Trigger'y używane przy wykonywaniu procedury:

I. Trigger wywołany dla tabeli pr_faktura_naglowek: **tr_pr_naglowek**

```
create or replace TRIGGER "I7Y7_17"."TR_PR_NAGLOWEK"
BEFORE INSERT OR UPDATE ON PR_FAKTURA_NAGLOWEK
FOR EACH ROW
declare
    v_tmp               number;
    v_faktury_count     number;
    v_wylosowana        number;
    v_date              date;

    v_ile_pozycji       number;
    v_ilosc             pr_lek.ilosc%type;
    v_ile_dodac         pr_lek.ilosc%type;
    counter             integer default 1;
    v_ile_lekow         pr_lek.kod_leku%type;
    v_kod_wojewodztwa      pr_klient.kod_wojewodztwa%type;

    v_wartosc_faktury   pr_faktura_naglowek.wartosc_faktury%type;
    v_wartosc_tmp       pr_faktura_naglowek.wartosc_faktury%type;
    v_ilosc_tmp         pr_faktura_pozycja.ilosc%type;
    v_cena_tmp          pr_faktura_pozycja.cena_zakupu%type;


BEGIN
    if inserting then
        v_tmp := seq_pr_naglowek.nextval;   --sekwencja

        select count(*) into v_faktury_count
        from pr_faktura_naglowek;       --sprawdzam ile jest faktur

    if v_faktury_count = 0
    then
        v_date := '18/01/01';   --data pierwszej faktury

    else --jesli istnieja juz jakies faktury
        select max(data_faktury) into v_date
        from pr_faktura_naglowek;

        if v_date + 5 <= '18/12/31'
        then

        v_wylosowana := round(dbms_random.value(0,5)); --data nastepnej
faktury

        v_date := v_date + v_wylosowana;

        else v_date := '18/12/31';
        end if;

    end if;


    :NEW.nr_faktury     := v_tmp;
    :NEW.data_faktury   := v_date;
```

```
    elsif updating('status') then

            select count(*) into v_ile_lekow
            from pr_lek;            --liczba lekow w bazie

            for counter in 1..v_ile_lekow loop

                select ilosc into v_ilosc
                from pr_lek
                where kod_leku = counter;

                if v_ilosc < 4 then
                    v_ile_dodac := round(dbms_random.value(1,30));
                else
                    v_ile_dodac := 0;

                end if;

                update pr_lek
                set ilosc = ilosc + v_ile_dodac
                where kod_leku = counter;

            end loop;

    select count(*) into v_ile_pozycji
    from pr_faktura_pozycja
    where nr_faktury = :OLD.nr_faktury;

    v_wartosc_faktury := 0;

    for counter in 1 .. v_ile_pozycji loop

        select ilosc into v_ilosc_tmp
        from pr_faktura_pozycja
        where nr_pozycji=counter and nr_faktury = :OLD.nr_faktury;

        select cena_zakupu into v_cena_tmp
        from pr_faktura_pozycja
        where nr_pozycji = counter and nr_faktury = :OLD.nr_faktury;

        v_wartosc_tmp := v_ilosc_tmp * v_cena_tmp;
        v_wartosc_faktury := v_wartosc_faktury+v_wartosc_tmp;
        v_wartosc_tmp := 0;
    end loop;

    :NEW.wartosc_faktury := v_wartosc_faktury;

    end if;

     select kod_wojewodztwa into    v_kod_wojewodztwa
     from pr_klient
     where nr_klienta = :NEW.nr_klienta;

     :NEW.kod_wojewodztwa := v_kod_wojewodztwa;



END;
```

Opis Trigger'a:

a) Pierwszą wykonywaną czynnością jest wykorzystanie utworzonej sekwencji do automatycznej generacji wartości nr_faktury. W pętli sprawdzana jest ilość faktur w bazie i znowu jak w przypadku procedury, jeśli faktur w bazie nie ma to jako date pierwszej faktury uznajemy '01/01/2018', zaś jeśli faktury już istnieją to wybierana jest data maksymalna oraz losowana liczba z przedziału <0 ; 5> o ile kolejna faktura ma być starsza. W wyniku tego losowania możemy otrzymać faktury wygenerowane tego samego dnia co faktura poprzednia, lub faktury starsze o odpowiednio wylosowaną ilość dni

b) Kolejna część dotyczy sytuacji wykonywania polecenia update dla 'statusu' w tabeli pr_naglowek_faktura. Czynności tu opisane zachodzą w przypadku zmiany statusu faktury na 'ZAMKNIETA' pod koniec wykonywania procedury. Ta część triggera odpowiada za stałe monitorowanie ilości leków w bazie. Założyłem, że jeśli ilość danego leku w naszym magazynie będzie mniejsza niż 4, wtedy podjęta zostanie decyzja o zamówienie produktu. Ilość jaką zamówimy wylosowana zostanie z przedziału <1 ; 30>.

c) Ostatnią czynnością wykonywaną w tym triggerze jest obliczenie wartości całej faktury. W tym celu przy pomocy pętli oraz jednej zmiennej pomocniczej przechodzę po wszystkich pozycjach faktury i wykonuję działanie mnożenia ilości zakupionych leków przez ich cenę zakupu. Na końcu do naszej faktury dodawany jest również kod województwa z którego pochodzi zamówienie, lecz tutaj przepisywany jest on jedynie na podstawie nr_klienta z bazy pr_klient.

II. Trigger dotyczący tabeli pr_faktura_pozycja: **tr_pr_pozycja**

```
create or replace TRIGGER TR_PR_POZYCJA
BEFORE DELETE OR INSERT OR UPDATE ON PR_FAKTURA_POZYCJA
for each row
declare
    v_if_enough                    number;
    v_if_exists                    number;
    v_ile_wszystkich               number;

    v_cena_leku                    pr_lek.cena_sprzedazy%type;
    v_kod_leku                     pr_lek.kod_leku%type;
    v_stawka_podatku               pr_lek.stawka_podatku%type;
    v_recepta                      pr_lek.recepta%type;
    v_ile_wszystkich_tmp           number;

BEGIN
  if inserting
  then
        select ilosc into v_if_enough
        from pr_lek
        where kod_leku = :NEW.kod_leku;
```

```
        if v_if_enough >= :NEW.ilosc then

            select count(*) into v_if_exists
            from Pr_Faktura_Pozycja     --sprawdzam czy juz taki istnieje
                                        --o danym kodzie
            where kod_leku = :NEW.kod_leku and nr_faktury =
:NEW.nr_faktury;

            if v_if_exists = 0 then     --jesli nie ma takiej pozycji

                select count(*) into v_ile_wszystkich_tmp
                from pr_faktura_pozycja
                where nr_faktury = :NEW.nr_faktury;

                if v_ile_wszystkich_tmp = 0 then    --jesli nie ma zadnej
pozycji
                    v_ile_wszystkich := 0;
                else
                    select max(nr_pozycji) into v_ile_wszystkich
                    from pr_faktura_pozycja
                    where nr_faktury = :NEW.nr_faktury;
                end if;

                v_ile_wszystkich := v_ile_wszystkich+1;

                select cena_sprzedazy into v_cena_leku
                from pr_lek
                where kod_leku = :NEW.kod_leku;

                select stawka_podatku into v_stawka_podatku
                from pr_lek
                where kod_leku = :NEW.kod_leku;

                select recepta into v_recepta
                from pr_lek
                where kod_leku = :NEW.kod_leku;

                :NEW.nr_pozycji         := v_ile_wszystkich;
                :NEW.cena_zakupu        := v_cena_leku;
                :NEW.stawka_podatku     := v_stawka_podatku;
                :NEW.recepta            := v_recepta;

            elsif v_if_exists <> 0
            then raise_application_error(-20005,
                    'Lek o kodzie '|| :NEW.kod_leku ||' juz istnieje!!');
            end if;


            update pr_lek
            set ilosc = ilosc - :NEW.ilosc
            where kod_leku = :NEW.kod_leku;

        elsif v_if_enough < :NEW.ilosc
            then raise_application_error(-20005,  'Nie wystarczajaca ilosc
w magazynie leku o kodzie '|| :NEW.kod_leku||'');

        end if;

        end if;

        END;
```

Opis Trigger'a:

a) Pierwszym krokiem, przed dodaniem wartości do tabeli sprawdzam czy posiadam wystarczająca ilość produktu w bazie. Sprawdzam również czy takiego produktu już na jakiejś pozycji nie ma i dopiero przechodzę do działań na pozycji

b) Sprawdzam aktualną ilość pozycji na fakturze, w przypadku 0 do zmiennej przypisuje 0 i po wykonaniu przypisania zmiennych z odpowiednich tabel takich jak cena_leku, stawka_podatku, kod_leku, recepta z tabeli pr_lek, zwiększam wartość pozycji o 1.

c) Końcowa część kodu odpowiada za obsługę errorów opisanych w podpunkcie a). W przypadku już istniejącej takiej pozycji na fakturze wywołam error o tym informujący, oraz w przypadku niewystarczającej ilości leku w magazynie, również wywołam error informujący o zbyt małej ilości produktu, oraz poinformuje o jaki lek konkretnie chodzi (wypisze kod_leku)

III. Trigger dotyczący tabeli pr_klient: **pr_insert_klient**

```
create or replace TRIGGER "TR_PR_INSERT_KLIENT"
BEFORE INSERT ON PR_KLIENT
FOR EACH ROW
    BEGIN
    :NEW.nr_klienta := SEQ_PR_KLIENT.Nextval;
  NULL;
END;
```

Jedyna wykonywana tu czynność to przypisanie kolumnie nr_klienta wartości Nextval sekwencji seq_pr_klient.

IV. Trigger dotyczący tabeli pr_lek: **pr_insert_lek**

```
create or replace TRIGGER TR_PR_INSERT_LEK
BEFORE INSERT ON PR_LEK
FOR EACH ROW
    BEGIN
    :NEW.kod_leku := SEQ_PR_LEK.Nextval;
  NULL;
END;
```

Tutaj również jedyna czynność to przypisanie kolumnie kod_leku kolejnej wartości sekwencji seq_pr_lek

V. Funkcja **FN_PR_DAJ_KOD_LEKU**

```
create or replace FUNCTION FN_PR_DAJ_KOD_LEKU
RETURN NUMBER
AS
    v_tmp   pr_lek.kod_leku%type;
BEGIN
    select * into v_tmp
    from
    (
        select kod_leku from pr_lek
        order by dbms_random.value
    )
    where rownum=1;

    return v_tmp;

END FN_PR_DAJ_KOD_LEKU;
```

Funkcja zwraca typ number, a konkretniej zwraca losowy kod leku wygenerowany przy pomocy sortowania tabeli po losowej kolejności, oraz pobrania jej pierwszego wiersza.

VI. Funkcja **FN_PR_DAJ_ILOSC_LEKU**

```
create or replace FUNCTION FN_PR_DAJ_ILOSC_LEKU (v_kod_leku
pr_lek.kod_leku%type)
RETURN pr_lek.ilosc%type
AS
    v_ilosc     pr_lek.ilosc%type;
    v_return    pr_lek.ilosc%type;

BEGIN
    select ilosc into v_ilosc
    from pr_lek
    where kod_leku=v_kod_leku;

    v_ilosc := round(v_ilosc/3);

    if v_ilosc = 0
        then return 0;
    else
    select round(dbms_random.value(1, v_ilosc)) into v_return
    from dual;
    end if;

    if v_return > 30
        then v_return := round(v_return/4);
    end if;

    return v_return;
    end;
```

Funkcja FN_PR_DAJ_ILOSC_LEKU jako argument przyjmuje właśnie kod_leku. Na jego podstawie wykonywane jest zapytanie do tabeli pr_lek o ilość danego leku. Zwróconą wartość postanowiłem podzielić przez 3, aby uniknąć dużych i mało realnych wyników w przypadku na przykład posiadania w bazie 300sztuk jakiegoś leku. Na końcu dodałem jeszcze jeden taki warunek, aby jak najbardziej zmniejszyć zwracane losowe wartości ilości leków.

VII. Funkcja **FN_PR_DAJ_NR_KLIENTA**

```
create or replace FUNCTION FN_PR_DAJ_NR_KLIENTA
RETURN NUMBER
AS
    v_tmp   pr_klient.nr_klienta%type;
BEGIN
    select * into v_tmp
    from
      (
        select nr_klienta
        from pr_klient
        order by dbms_random.value

      )
      where rownum=1;

      return v_tmp;


END FN_PR_DAJ_NR_KLIENTA;
```

Funkcja ta działając na identycznej zasadzie jak funkcja dotycząca zwracania losowego kodu leku, w tym przypadku ma za zadanie zwrócić losowy nr_klienta spośród klientów umieszczonych w naszej bazie.

4. Skrypty wdrożeniowe instalujące i deinstalujące zrealizowany projekt

tworzenie_tabel.sql

```
CREATE TABLE pr_faktura_naglowek (
    nr_faktury        NUMBER(8) NOT NULL,
    nr_klienta        NUMBER(8) NOT NULL,
    data_faktury      DATE NOT NULL,
    wartosc_faktury   NUMBER DEFAULT 0,
    kod_wojewodztwa   NUMBER(2) NULL,
    status            VARCHAR2(20) DEFAULT 'OTWARTA'
)
    LOGGING;

ALTER TABLE pr_faktura_naglowek
    ADD CHECK (
        status IN (
            'OTWARTA','ZAMKNIETA'
        )
    );

ALTER TABLE pr_faktura_naglowek ADD CONSTRAINT faktura_naglowek_pk PRIMARY
KEY ( nr_faktury );

CREATE TABLE pr_faktura_pozycja (
    nr_faktury       NUMBER(8) NOT NULL,
    nr_pozycji       NUMBER(8) NOT NULL,
    kod_leku         NUMBER(8) NOT NULL,
    cena_zakupu      NUMBER NOT NULL,
    ilosc            NUMBER NOT NULL,
    stawka_podatku   NUMBER(2),
    recepta          VARCHAR2(3)
)
    LOGGING;

ALTER TABLE pr_faktura_pozycja ADD CONSTRAINT faktura_pozycja_pk PRIMARY
KEY ( nr_faktury,nr_pozycji );

CREATE TABLE pr_klient (
    nr_klienta        NUMBER(8) NOT NULL,
    imie              VARCHAR2(25) NOT NULL,
    nazwisko          VARCHAR2(40) NOT NULL,
    "e-mail"          VARCHAR2(60) NOT NULL,
    telefon           NUMBER(9),
    ulica             VARCHAR2(45) NOT NULL,
    miasto            VARCHAR2(25) NOT NULL,
    kod_pocztowy      VARCHAR2(6) NOT NULL,
    kod_wojewodztwa   NUMBER(2) NOT NULL
)
    LOGGING;

ALTER TABLE pr_klient ADD CONSTRAINT klient_pk PRIMARY KEY ( nr_klienta );

CREATE TABLE pr_lek (
    kod_leku          NUMBER(8) NOT NULL,
    pr_typ_leku          VARCHAR2(20) NOT NULL,
    nazwa_leku        VARCHAR2(100) NOT NULL,
    cena_zakupu       NUMBER NOT NULL,
    cena_sprzedazy    NUMBER NOT NULL,
```

```
    stawka_podatku   NUMBER(2) NOT NULL,
    recepta          VARCHAR2(3) NOT NULL,
    ilosc            NUMBER(10) NOT NULL
)
    LOGGING;

ALTER TABLE pr_lek
    ADD CHECK (
        recepta IN (
            'NIE','TAK'
        )
    );

ALTER TABLE pr_lek ADD CONSTRAINT lek_pk PRIMARY KEY ( kod_leku );

CREATE TABLE pr_typ_leku (
    pr_typ_leku      VARCHAR2(20) NOT NULL,
    nazwa_pelna   VARCHAR2(100) NOT NULL,
    rabat         NUMBER(3)
)
    LOGGING;

ALTER TABLE pr_typ_leku ADD CONSTRAINT typ_leku_pk PRIMARY KEY (
pr_typ_leku );

CREATE TABLE pr_wojewodztwa (
    kod_wojewodztwa     NUMBER(2) NOT NULL,
    nazwa_wojewodztwa   VARCHAR2(40) NOT NULL
)
    LOGGING;

ALTER TABLE pr_wojewodztwa ADD CONSTRAINT wojewodztwa_pk PRIMARY KEY (
kod_wojewodztwa );

ALTER TABLE pr_faktura_pozycja
    ADD CONSTRAINT faktura_naglowek_fk FOREIGN KEY ( nr_faktury )
        REFERENCES pr_faktura_naglowek ( nr_faktury )
    NOT DEFERRABLE;

ALTER TABLE pr_faktura_naglowek
    ADD CONSTRAINT klient_fk FOREIGN KEY ( nr_klienta )
        REFERENCES pr_klient ( nr_klienta )
    NOT DEFERRABLE;

ALTER TABLE pr_faktura_pozycja
    ADD CONSTRAINT lek_fk FOREIGN KEY ( kod_leku )
        REFERENCES pr_lek ( kod_leku )
    NOT DEFERRABLE;

ALTER TABLE pr_lek
    ADD CONSTRAINT typ_leku_fk FOREIGN KEY ( pr_typ_leku )
        REFERENCES pr_typ_leku ( pr_typ_leku )
    NOT DEFERRABLE;

ALTER TABLE pr_faktura_naglowek
    ADD CONSTRAINT wojewodztwa_fk FOREIGN KEY ( kod_wojewodztwa )
        REFERENCES pr_wojewodztwa ( kod_wojewodztwa )
    NOT DEFERRABLE;

ALTER TABLE pr_klient
    ADD CONSTRAINT wojewodztwa_fkv1 FOREIGN KEY ( kod_wojewodztwa )
```

```
        REFERENCES pr_wojewodztwa ( kod_wojewodztwa )
    NOT DEFERRABLE;


/
CREATE SEQUENCE  SEQ_PR_NAGLOWEK  MINVALUE 1 MAXVALUE 9999999999999
INCREMENT BY 1 START WITH 1 CACHE 20;
/


CREATE SEQUENCE  SEQ_PR_LEK  MINVALUE 1 MAXVALUE 99999999999 INCREMENT BY 1
START WITH 1 CACHE 20;
/

CREATE SEQUENCE  SEQ_PR_KLIENT MINVALUE 1 MAXVALUE 99999999999999 INCREMENT
BY 1 START WITH 1 CACHE 20;

/
create or replace FUNCTION FN_PR_DAJ_KOD_LEKU
RETURN NUMBER
AS
    v_tmp   pr_lek.kod_leku%type;
BEGIN
    select * into v_tmp
    from
    (
        select kod_leku from pr_lek
        order by dbms_random.value
    )
    where rownum=1;

    return v_tmp;

END FN_PR_DAJ_KOD_LEKU;

/

create or replace FUNCTION FN_PR_DAJ_ILOSC_LEKU (v_kod_leku
pr_lek.kod_leku%type)
RETURN pr_lek.ilosc%type
AS
    v_ilosc     pr_lek.ilosc%type;
    v_return    pr_lek.ilosc%type;

BEGIN
    select ilosc into v_ilosc
    from pr_lek
    where kod_leku=v_kod_leku;

    v_ilosc := round(v_ilosc/3);

    if v_ilosc = 0
        then return 0;
    else
    select round(dbms_random.value(1, v_ilosc)) into v_return
    from dual;
    end if;

    if v_return > 30
        then v_return := round(v_return/4);
    end if;
```

```
    return v_return;
    end;

/

create or replace FUNCTION FN_PR_DAJ_NR_KLIENTA
RETURN NUMBER
AS
    v_tmp   pr_klient.nr_klienta%type;
BEGIN
    select * into v_tmp
    from
      (
        select nr_klienta
        from pr_klient
        order by dbms_random.value

      )
      where rownum=1;

      return v_tmp;


END FN_PR_DAJ_NR_KLIENTA;

/
create or replace TRIGGER TR_PR_INSERT_LEK
BEFORE INSERT ON PR_LEK
FOR EACH ROW
    BEGIN
    :NEW.kod_leku := SEQ_PR_LEK.Nextval;
  NULL;
END;
/
create or replace TRIGGER "TR_PR_INSERT_KLIENT"
BEFORE INSERT ON PR_KLIENT
FOR EACH ROW
    BEGIN
    :NEW.nr_klienta := SEQ_PR_KLIENT.Nextval;
  NULL;
END;
/
create or replace TRIGGER "TR_PR_NAGLOWEK"
BEFORE INSERT OR UPDATE ON PR_FAKTURA_NAGLOWEK
FOR EACH ROW
declare
    v_tmp               number;
    v_faktury_count     number;
    v_wylosowana        number;
    v_date              date;

    v_ile_pozycji       number;
    v_ilosc             pr_lek.ilosc%type;
    v_ile_dodac         pr_lek.ilosc%type;
    counter             integer default 1;
    v_ile_lekow         pr_lek.kod_leku%type;
    v_kod_wojewodztwa      pr_klient.kod_wojewodztwa%type;

    v_wartosc_faktury   pr_faktura_naglowek.wartosc_faktury%type;
    v_wartosc_tmp       pr_faktura_naglowek.wartosc_faktury%type;
```

```
    v_ilosc_tmp         pr_faktura_pozycja.ilosc%type;
    v_cena_tmp          pr_faktura_pozycja.cena_zakupu%type;


BEGIN
    if inserting then
        v_tmp := seq_pr_naglowek.nextval;   --sekwencja

        select count(*) into v_faktury_count
        from pr_faktura_naglowek;       --sprawdzam ile jest faktur

    if v_faktury_count = 0
    then
        v_date := '18/01/01';   --data pierwszej faktury

    else --jesli istnieja juz jakies faktury
        select max(data_faktury) into v_date
        from pr_faktura_naglowek;

        if v_date + 5 <= '18/12/31'
        then

        v_wylosowana := round(dbms_random.value(0,5)); --data nastepnej
faktury

        v_date := v_date + v_wylosowana;

        else v_date := '18/12/31';
        end if;

    end if;


    :NEW.nr_faktury     := v_tmp;
    :NEW.data_faktury   := v_date;



    elsif updating('status') then

            select count(*) into v_ile_lekow
            from pr_lek;            --liczba lekow w bazie

            for counter in 1..v_ile_lekow loop

                select ilosc into v_ilosc
                from pr_lek
                where kod_leku = counter;

                if v_ilosc < 4 then
                    v_ile_dodac := round(dbms_random.value(1,30));
                else
                    v_ile_dodac := 0;

                end if;

                update pr_lek
                set ilosc = ilosc + v_ile_dodac
                where kod_leku = counter;

            end loop;
```

```
    select count(*) into v_ile_pozycji
    from pr_faktura_pozycja
    where nr_faktury = :OLD.nr_faktury;

    v_wartosc_faktury := 0;

    for counter in 1 .. v_ile_pozycji loop

        select ilosc into v_ilosc_tmp
        from pr_faktura_pozycja
        where nr_pozycji=counter and nr_faktury = :OLD.nr_faktury;

        select cena_zakupu into v_cena_tmp
        from pr_faktura_pozycja
        where nr_pozycji = counter and nr_faktury = :OLD.nr_faktury;

        v_wartosc_tmp := v_ilosc_tmp * v_cena_tmp;
        v_wartosc_faktury := v_wartosc_faktury+v_wartosc_tmp;
        v_wartosc_tmp := 0;
    end loop;

    :NEW.wartosc_faktury := v_wartosc_faktury;

    end if;

     select kod_wojewodztwa into    v_kod_wojewodztwa
     from pr_klient
     where nr_klienta = :NEW.nr_klienta;

     :NEW.kod_wojewodztwa := v_kod_wojewodztwa;



END;
/
create or replace TRIGGER TR_PR_POZYCJA
BEFORE DELETE OR INSERT OR UPDATE ON PR_FAKTURA_POZYCJA
for each row
declare
    v_if_enough                     number;
    v_if_exists                     number;
    v_ile_wszystkich                number;

    v_cena_leku                     pr_lek.cena_sprzedazy%type;
    v_kod_leku                      pr_lek.kod_leku%type;
    v_stawka_podatku                pr_lek.stawka_podatku%type;
    v_recepta                       pr_lek.recepta%type;
    v_ile_wszystkich_tmp            number;

BEGIN
  if inserting
  then
        select ilosc into v_if_enough
        from pr_lek
        where kod_leku = :NEW.kod_leku;

        if v_if_enough >= :NEW.ilosc then

            select count(*) into v_if_exists
```

```
            from Pr_Faktura_Pozycja     --sprawdzam czy juz taki istnieje
                                        --o danym kodzie
            where kod_leku = :NEW.kod_leku and nr_faktury =
:NEW.nr_faktury;

            if v_if_exists = 0 then     --jesli nie ma takiej pozycji

                select count(*) into v_ile_wszystkich_tmp
                from pr_faktura_pozycja
                where nr_faktury = :NEW.nr_faktury;

                if v_ile_wszystkich_tmp = 0 then    --jesli nie ma zadnej
pozycji
                    v_ile_wszystkich := 0;
                else
                    select max(nr_pozycji) into v_ile_wszystkich
                    from pr_faktura_pozycja
                    where nr_faktury = :NEW.nr_faktury;
                end if;

                v_ile_wszystkich := v_ile_wszystkich+1;

                select cena_sprzedazy into v_cena_leku
                from pr_lek
                where kod_leku = :NEW.kod_leku;

                select stawka_podatku into v_stawka_podatku
                from pr_lek
                where kod_leku = :NEW.kod_leku;

                select recepta into v_recepta
                from pr_lek
                where kod_leku = :NEW.kod_leku;

                :NEW.nr_pozycji         := v_ile_wszystkich;
                :NEW.cena_zakupu        := v_cena_leku;
                :NEW.stawka_podatku     := v_stawka_podatku;
                :NEW.recepta            := v_recepta;

            elsif v_if_exists <> 0
            then raise_application_error(-20005,
                    'Lek o kodzie '|| :NEW.kod_leku ||' juz istnieje!!');
            end if;


            update pr_lek
            set ilosc = ilosc - :NEW.ilosc
            where kod_leku = :NEW.kod_leku;

        elsif v_if_enough < :NEW.ilosc
            then raise_application_error(-20005,  'Nie wystarczajaca ilosc
w magazynie leku o kodzie '|| :NEW.kod_leku||'');

        end if;

        end if;
```

```
END;
/

create or replace PROCEDURE PR_PR_GENERUJ AS
    v_nr_faktury        pr_faktura_naglowek.nr_faktury%type;
    v_nr_klienta        pr_klient.nr_klienta%type;
    v_data_faktury      pr_faktura_naglowek.data_faktury%type;
    v_kod_wojewodztwa   pr_faktura_naglowek.kod_wojewodztwa%type;

    --pozycja
    v_kod_leku          pr_lek.kod_leku%type;
    v_ilosc             pr_lek.ilosc%type;


    v_losowa_ilosc_leku pr_lek.ilosc%type;
    v_czy_juz_istnieje  number;

    v_koniec_petli     number;
    v_data_petli    date;

    v_stop              boolean;
    v_ile_lekow         pr_lek.kod_leku%type;
    v_counter           number;


BEGIN

    v_stop := false;

    while v_stop <> true loop

    select count(*) into v_koniec_petli
    from pr_faktura_naglowek;

    if v_koniec_petli = 0
    then v_data_petli := '18/01/01';

    else
        select max(data_faktury) into v_data_petli
        from pr_faktura_naglowek;
    end if;


    if v_data_petli + 5 <= '18/12/31'
    then
        v_nr_klienta := fn_pr_daj_nr_klienta();

        insert into pr_faktura_naglowek(nr_klienta)
        values  (
                        v_nr_klienta
                );
        select nr_faktury into v_nr_faktury
        from pr_faktura_naglowek
        where status='OTWARTA';

        select data_faktury into v_data_faktury
        from pr_faktura_naglowek
        where nr_faktury = v_nr_faktury;

        --ile pozycji na fakturze
```

```
        select count(*) into v_ile_lekow
        from pr_lek;

        v_ile_lekow := round(dbms_random.value(1, v_ile_lekow));

        --
        v_counter :=0;
        while v_counter <> v_ile_lekow loop

        v_kod_leku := fn_pr_daj_kod_leku();

        select count(*) into v_czy_juz_istnieje
        from pr_faktura_pozycja
        where kod_leku = v_kod_leku and nr_faktury = v_nr_faktury;

        if v_czy_juz_istnieje = 0
        then
            v_ilosc := fn_pr_daj_ilosc_leku(v_kod_leku);

            if v_ilosc > 0
            then
                insert into pr_faktura_pozycja(nr_faktury, kod_leku, ilosc)
                    values  (
                                v_nr_faktury, v_kod_leku, v_ilosc
                            );
            v_counter := v_counter+1;

            end if;
            end if;

            end loop;


            update pr_faktura_naglowek
            set status = 'ZAMKNIETA'
            where nr_faktury = v_nr_faktury;


            elsif v_data_petli + 5 > '18/12/31'
            then
                v_stop := true;
            end if;

            end loop;


  NULL;
END PR_PR_GENERUJ;
/

create or replace view PR_NAJLEPSZY_LEK
as

 select
kod,typ,nazwa,czy_recepta,Cena,Suma,to_char(Suma*c.cena_sprzedazy,'999999.9
9') as Sprzedaz
from
(
select a.kod_leku as kod, d.nazwa_pelna as typ, b.nazwa_leku as nazwa,
b.recepta as czy_recepta, a.cena_zakupu as Cena,  sum(a.ilosc) as Suma
from pr_faktura_pozycja a, pr_lek b, pr_typ_leku d
```

```
where a.kod_leku=b.kod_leku and d.pr_typ_leku=b.pr_typ_leku
group by a.kod_leku, d.nazwa_pelna, b.nazwa_leku, a.cena_zakupu, b.recepta
order by Suma desc
)
,pr_lek c
where c.kod_leku=kod
order by Sprzedaz desc;
/
create or replace view PR_KLIENCI_WYDATKI
AS
select k.nr_klienta, k.imie, k.nazwisko, k.ulica,k.kod_pocztowy, k.miasto,
w.nazwa_wojewodztwa,
        sum(wartosc_faktury) as "Suma"
from pr_klient k , pr_faktura_naglowek f, pr_wojewodztwa w
where k.nr_klienta=f.nr_klienta
and w.kod_wojewodztwa=k.kod_wojewodztwa
group by k.nr_klienta, k.imie, k.nazwisko, k.ulica, k.miasto,
w.nazwa_wojewodztwa,
k.kod_pocztowy
order by nr_klienta;
/
create or replace view PR_OBROT_MIESIACE
AS
select to_char(data_faktury, 'YYYY/MM') Okres,
to_char(to_date(extract(month from(data_faktury)), 'MM'), 'month') Miesiac,
sum(wartosc_faktury) Suma
from pr_faktura_naglowek
group by extract(month from(data_faktury)), to_char(data_faktury,
'YYYY/MM')
order by extract(month from(data_faktury));
/
```

wprowadzenie_danych.sql

```
SET DEFINE OFF

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (2.0, 'dolnoœl¹skie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (4.0, 'kujawsko-pomorskie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (6.0, 'lubelskie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (8.0, 'lubuskie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (10.0, '³ódzkie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (12.0, 'ma³opolskie');
```

```
INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (14.0, 'mazowieckie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (16.0, 'opolskie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (18.0, 'podkarpackie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (20.0, 'podlaskie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (22.0, 'pomorskie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (24.0, 'œl¹skie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (26.0, 'œwiêtokrzyskie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (28.0, 'warmiñsko-mazurskie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (30.0, 'wielkopolskie');

INSERT INTO PR_WOJEWODZTWA (KOD_WOJEWODZTWA, NAZWA_WOJEWODZTWA)
VALUES (32.0, 'zachodniopomorskie');

-- Import Data into table PR_WOJEWODZTWA from file
D:\WAT\BazyDanych\Jakub_Z\Lista-województw-Excel.xlsx . Task successful and
sent to worksheet.


SET DEFINE OFF

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
VALUES (NULL, 'Janusz ', 'Kowalski', 'janusz.kowalski@wat.edu.pl',
5.05123963E8, 'Kocjana', 'Warszawa', '00-134', 14.0);

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
VALUES (NULL, 'Krzysztof', 'Kapusta', 'krzysztof.kapusta@wat.edu.pl', NULL,
'Radiowa', 'Wroc³aw', '19-120', 2.0);

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
VALUES (NULL, 'Magdalena', 'Dzwonkowska',
'magdalena.dzwonkowska@wat.edu.pl', 2.12145556E8, 'Górczewska',
'Legionowo', '13-002', 14.0);

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
VALUES (NULL, 'Karol', 'Krawczyk', 'karol.krawczyk@wat.edu.pl', NULL,
'Bakalarska', 'Lublin', '20-850', 6.0);

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
```

```
VALUES (NULL, 'Joanna', 'Bak', 'joanna.bak@wat.edu.pl', NULL, '£amana',
'Częstochowa', '09-420', 24.0);

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
VALUES (NULL, 'Robert', 'Psikuta', 'robert.psikuta@wat.edu.pl',
4.75369852E8, 'Pirenejska', 'Gdańsk', '17-242', 22.0);

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
VALUES (NULL, 'Stefan', 'Siarzewski', 'stefan.siarzewski@wat.edu.pl', NULL,
'Zbożowa', 'Warszawa', '01-128', 14.0);

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
VALUES (NULL, 'Zofia', 'Brzydal', 'zofia.brzydal@wat.edu.pl', NULL,
'Araszkiewicza', 'Jastarnia', '17-291', 22.0);

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
VALUES (NULL, 'Julia', 'Niedziela', 'julia.niedziela@wat.edu.pl',
8.01462852E8, 'Apeniñska', 'Kraków', '30-409', 12.0);

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
VALUES (NULL, 'Bartosz', 'Karmel', 'bartosz.karmel@wat.edu.pl',
4.98356278E8, 'Obozowa', 'Lublin', '20-134', 6.0);

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
VALUES (NULL, 'Sebastian', 'Krawczyk', 'sebastian.krawczyk@wat.edu.pl',
NULL, 'Gajowa', 'Zakopane', '09-112', 12.0);

INSERT INTO PR_KLIENT (NR_KLIENTA, IMIE, NAZWISKO, "e-mail", TELEFON,
ULICA, MIASTO, KOD_POCZTOWY, KOD_WOJEWODZTWA)
VALUES (NULL, 'Andrzej', 'Duda', 'andrzej.duda@wat.edu.pl', 9.97986123E8,
'Nowogrodzka', 'Warszawa', '00-367', 14.0);

-- Import Data into table PR_KLIENT from file
D:\WAT\BazyDanych\Jakub_Z\Lista-województw-Excel.xlsx . Task successful and
sent to worksheet.



SET DEFINE OFF

INSERT INTO PR_TYP_LEKU (pr_typ_leku, NAZWA_PELNA, RABAT)
VALUES ('KOS', 'kosmetyki', NULL);

INSERT INTO PR_TYP_LEKU (pr_typ_leku, NAZWA_PELNA, RABAT)
VALUES ('BOL', 'przeciwbólowe', NULL);

INSERT INTO PR_TYP_LEKU (pr_typ_leku, NAZWA_PELNA, RABAT)
VALUES ('ALER', 'alergia', NULL);

INSERT INTO PR_TYP_LEKU (pr_typ_leku, NAZWA_PELNA, RABAT)
VALUES ('TRAW', 'trawienie', NULL);

INSERT INTO PR_TYP_LEKU (pr_typ_leku, NAZWA_PELNA, RABAT)
VALUES ('STRES', 'uspokajaj¹ce/na stres', NULL);
```

```
INSERT INTO PR_TYP_LEKU (pr_typ_leku, NAZWA_PELNA, RABAT)
VALUES ('SEN', 'nasenne', NULL);

INSERT INTO PR_TYP_LEKU (pr_typ_leku, NAZWA_PELNA, RABAT)
VALUES ('WIT', 'witaminy/suplementy', NULL);

INSERT INTO PR_TYP_LEKU (pr_typ_leku, NAZWA_PELNA, RABAT)
VALUES ('CUK', 'cukrzycowe', NULL);

INSERT INTO PR_TYP_LEKU (pr_typ_leku, NAZWA_PELNA, RABAT)
VALUES ('KARD', 'kardiologiczne', NULL);

SET DEFINE OFF

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'BOL', 'apap', 4.6, 6.2, 23.0, 'NIE', 124.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'BOL', 'ketonal', 38.2, 44.8, 7.0, 'TAK', 38.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'BOL', 'ibuprom', 5.0, 8.2, 7.0, 'NIE', 350.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'BOL', 'naproxen', 19.99, 27.1, 7.0, 'TAK', 80.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'ALER', 'alertec', 9.6, 12.5, 7.0, 'TAK', 100.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'ALER', 'zyrtec', 18.2, 21.3, 7.0, 'TAK', 220.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'TRAW', 'helicid', 14.0, 19.2, 7.0, 'TAK', 114.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'TRAW', 'essentiale forte', 24.0, 28.1, 23.0, 'NIE', 84.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'TRAW', 'espumisan forte', 12.9, 14.99, 23.0, 'NIE', 45.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'TRAW', 'tribux forte', 20.0, 23.7, 7.0, 'TAK', 19.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'KOS', 'szampon DX2', 29.99, 35.2, 23.0, 'NIE', 30.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'KOS', 'pasta BLANX', 26.99, 30.0, 23.0, 'NIE', 34.0);
```

```
INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'STRES', 'positivum', 3.99, 6.4, 23.0, 'NIE', 75.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'SEN', 'nasen', 34.99, 40.0, 7.0, 'TAK', 20.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'STRES', 'meliski', 5.5, 7.99, 23.0, 'NIE', 120.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'STRES', 'nervosol', 4.2, 6.99, 23.0, 'NIE', 20.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'SEN', 'melatonina', 21.0, 24.99, 23.0, 'NIE', 90.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'SEN', 'valerin sen', 7.4, 10.0, 23.0, 'NIE', 15.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'WIT', 'centrum ON', 30.0, 34.1, 23.0, 'NIE', 100.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'WIT', 'bodymax', 19.2, 23.8, 23.0, 'NIE', 150.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'WIT', 'magne b6', 12.0, 14.5, 23.0, 'NIE', 190.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'WIT', 'marsjanki', 26.3, 29.0, 23.0, 'NIE', 55.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'CUK', 'glukoza', 12.0, 16.0, 7.0, 'NIE', 70.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'CUK', 'paski Accu-Check', 42.0, 49.99, 23.0, 'NIE', 39.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'KARD', 'acard', 8.2, 13.1, 7.0, 'NIE', 140.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'KARD', 'milocardin', 28.99, 37.0, 7.0, 'TAK', 80.0);

INSERT INTO PR_LEK (KOD_LEKU, pr_typ_leku, NAZWA_LEKU, CENA_ZAKUPU,
CENA_SPRZEDAZY, STAWKA_PODATKU, RECEPTA, ILOSC)
VALUES (NULL, 'KARD', 'ebivol', 31.0, 37.0, 7.0, 'TAK', 29.0);
```

```
-- Import Data into table PR_LEK from file D:\WAT\BazyDanych\Jakub_Z\Lista-województw-Excel.xlsx . Task successful and sent to worksheet.

/
begin
    pr_pr_generuj();
end;
/
```

5. Skrypty deinstalujące projekt

```
DROP TABLE pr_faktura_naglowek CASCADE CONSTRAINTS;

DROP TABLE pr_faktura_pozycja CASCADE CONSTRAINTS;

DROP TABLE pr_klient CASCADE CONSTRAINTS;

DROP TABLE pr_lek CASCADE CONSTRAINTS;

DROP TABLE pr_typ_leku CASCADE CONSTRAINTS;

DROP TABLE pr_wojewodztwa CASCADE CONSTRAINTS;

DROP TRIGGER TR_PR_INSERT_KLIENT;

DROP TRIGGER TR_PR_INSERT_LEK;

DROP TRIGGER TR_PR_NAGLOWEK;

DROP TRIGGER TR_PR_POZYCJA;

DROP SEQUENCE SEQ_PR_KLIENT;

DROP SEQUENCE SEQ_PR_LEK;

DROP SEQUENCE SEQ_PR_NAGLOWEK;

DROP FUNCTION FN_PR_DAJ_ILOSC_LEKU;

DROP FUNCTION FN_PR_DAJ_KOD_LEKU;

DROP FUNCTION FN_PR_DAJ_NR_KLIENTA;

DROP VIEW PR_KLIENCI_WYDATKI;

DROP VIEW PR_NAJLEPSZY_LEK;

DROP VIEW PR_OBROT_MIESIACE;

DROP PROCEDURE pr_pr_generuj;
```

6. Instrukcja instalacji projektu i sprawdzenia jej poprawności

1) Uruchomić skrypt **tworzenie_tabel.sql**, spowoduje on utworzenie wszelkich obiektów bazodanowych, koniecznych do poprawnego działania projektu, tj. tabele/funkcje/sekwencje/triggery oraz perspektywy

2) Uruchomić skrypt **wprowadzenie_danych.sql,** spowoduje on uzupełnienie tabel pr_lek, pr_klient, pr_wojewodztwa oraz pr_typ_leku odpowiednimi danymi

3) W celu wygenerowania transakcji, na końcu skryptu **wprowadzenie_danych.sql** umieściłem polecenie PL/SQL odpowiedzialne za uruchomienie procedury generującej dane **pr_pr_generuj().**

```
begin
    pr_pr_generuj();
end;
```

Po jej wykonaniu, baza powinna wygenerować transakcje, czego efekty będą widoczne w tabelach pr_faktura_naglowek oraz pr_faktura_pozycja. Liczba danych nie jest stała gdyż za każdym razem podlega losowemu generowaniu daty, ilości leku itd.

4) W celu sprawdzenia poprawności możemy odpytać po kolei tabele przy pomocy zapytania select * from tabela;
Lub po wejściu w szczegóły każdej z tabel, przeanalizować jej model, czy jest zgodny z oczekiwaniami, jak i zakładke columns oraz constraints, aby dowiedzieć się o działaniu w niej relacji.

5) Do deinstalacji projektu należy wykorzystać dołączony skrypt **deinstalacja_projektu.sql,** efektem jego działania będzie usunięcie wszystkich utworzonych przy pomocy powyższych skryptów danych i obiektów bazodanowych.

7. Wyniki działania perspektyw analitycznych w postaci raportów pdf

ZESTAWIENIE WYDATKÓW KLIENTÓW

Zakres od 01/01/2018 do 31/12/2018

KLIENT	IMIE	NAZWISKO	ULICA	KOD POCZTOWY	MIASTO	WOJEWÓDZTWO	SUMA
1	Janusz	Kowalski	Kocjana	00-134	Warszawa	mazowieckie	10138.04
2	Krzysztof	Kapusta	Radiowa	19-120	Wrocław	dolnośląskie	6185.95
3	Magdalen	Dzwonkowska	Górczewska	13-002	Legionowo	mazowieckie	11548.53
4	Karol	Krawczyk	Bakalarska	20-850	Lublin	lubelskie	12246.49
5	Joanna	Bak	Łamana	09-420	Częstochowa	śląskie	7616.05
6	Robert	Psikuta	Pirenejska	17-242	Gdańsk	pomorskie	13855.58
7	Stefan	Siarzewski	Zbożowa	01-128	Warszawa	mazowieckie	11377.64
8	Zofia	Brzydal	Araszkiewicza	17-291	Jastarnia	pomorskie	6947.93
9	Julia	Niedziela	Apenińska	30-409	Kraków	małopolskie	6626.14
10	Bartosz	Karmel	Obozowa	20-134	Lublin	lubelskie	7357.09
11	Sebastian	Krawczyk	Gajowa	09-112	Zakopane	małopolskie	6587.07
12	Andrzej	Duda	Nowogrodzka	00-367	Warszawa	mazowieckie	14323.09

15/03/2019

Kod Perspektywy:

```
CREATE OR REPLACE FORCE EDITIONABLE VIEW
"I7Y7_17"."PR_KLIENCI_WYDATKI" ("NR_KLIENTA", "IMIE",
"NAZWISKO", "ULICA", "KOD_POCZTOWY", "MIASTO",
"NAZWA_WOJEWODZTWA", "Suma") AS
  select k.nr_klienta, k.imie, k.nazwisko,
k.ulica,k.kod_pocztowy, k.miasto, w.nazwa_wojewodztwa,
        sum(wartosc_faktury) as "Suma"
from pr_klient k , pr_faktura_naglowek f, pr_wojewodztwa w
where k.nr_klienta=f.nr_klienta
and w.kod_wojewodztwa=k.kod_wojewodztwa
group by k.nr_klienta, k.imie, k.nazwisko, k.ulica, k.miasto,
w.nazwa_wojewodztwa,
k.kod_pocztowy
order by nr_klienta;

   COMMENT ON TABLE "I7Y7_17"."PR_KLIENCI_WYDATKI"  IS
'Perspektywa prezentująca wydatki każdego klienta w bazie';
```

ZESTAWIENIE LEKÓW PRZYNOSZĄCYCH NAJWIEKSZY ZYSK

Zakres od 01/01/2018 do 31/12/2018

KOD LEKU	TYP	NAZWA	RECEPTA	CENA	ILOŚĆ SPRZEDANYCH	ZYSK
24	cukrzyco	paski Accu-Check	NIE	zł 49,99	191	9548.09
2	przeciwb	ketonal	TAK	zł 44,80	195	8736.00
14	nasenne	nasen	TAK	zł 40,00	170	6800.00
11	kosmetyk	szampon DX2	NIE	zł 35,20	184	6476.80
26	kardiolog	milocardin	TAK	zł 37,00	174	6438.00
27	kardiolog	ebivol	TAK	zł 37,00	171	6327.00
19	witaminy	centrum ON	NIE	zł 34,10	174	5933.40
12	kosmetyk	pasta BLANX	NIE	zł 30,00	195	5850.00
8	trawienie	essentiale forte	NIE	zł 28,10	202	5676.20
10	trawienie	tribux forte	TAK	zł 23,70	214	5071.80
4	przeciwb	naproxen	TAK	zł 27,10	182	4932.20
17	nasenne	melatonina	NIE	zł 24,99	187	4673.13
22	witaminy	marsjanki	NIE	zł 29,00	161	4669.00
6	alergia	zyrtec	TAK	zł 21,30	214	4558.20
20	witaminy	bodymax	NIE	zł 23,80	154	3665.20
7	trawienie	helicid	TAK	zł 19,20	180	3456.00

Kod Perspektywy:

```
  CREATE OR REPLACE FORCE EDITIONABLE VIEW
"I7Y7_17"."PR_NAJLEPSZY_LEK" ("KOD", "TYP", "NAZWA",
"CZY_RECEPTA", "CENA", "SUMA", "SPRZEDAZ") AS
  select
kod,typ,nazwa,czy_recepta,Cena,Suma,to_char(Suma*c.cena_sprzed
azy,'999999.99') as Sprzedaz
from
(
select a.kod_leku as kod, d.nazwa_pelna as typ, b.nazwa_leku
as nazwa, b.recepta as czy_recepta, a.cena_zakupu as Cena,
sum(a.ilosc) as Suma
from pr_faktura_pozycja a, pr_lek b, pr_typ_leku d
where a.kod_leku=b.kod_leku and d.typ_leku=b.typ_leku
group by a.kod_leku, d.nazwa_pelna, b.nazwa_leku,
a.cena_zakupu, b.recepta
order by Suma desc
)
,pr_lek c
where c.kod_leku=kod
order by Sprzedaz desc;
```

ZESTAWIENIE OBROTU Z PODZIAŁEM NA MIESIĄCE

OKRES	MIESIĄC	OBRÓT
2018/01	styczeń	7921,09
2018/02	luty	5800,81
2018/03	marzec	9877,60
2018/04	kwiecień	8127,94
2018/05	maj	12899,68
2018/06	czerwiec	9111,28
2018/07	lipiec	9724,85
2018/08	sierpień	12451,38
2018/09	wrzesień	7522,68
2018/10	październik	15131,06
2018/11	listopad	9704,35
2018/12	grudzień	6536,88

Zakres od 01/01/2018 do 31/12/2018

Kod Perspektywy:

```
  CREATE OR REPLACE FORCE EDITIONABLE VIEW
"I7Y7_17"."PR_OBROT_MIESIACE" ("OKRES", "MIESIAC", "SUMA") AS
  select to_char(data_faktury, 'YYYY/MM') Okres,
to_char(to_date(extract(month from(data_faktury)), 'MM'),
'month') Miesiac, sum(wartosc_faktury) Suma
from pr_faktura_naglowek
group by extract(month from(data_faktury)),
to_char(data_faktury, 'YYYY/MM')
order by extract(month from(data_faktury));
```