Bazy Danych
Projekt Zaliczeniowy Dokumentacja

Uproszczony System Bankowy

Autor:
Mateusz Wirzba

11 czerwca 2018

Spis treści

Rozdział 1

Założenia Projektu

1.Projekt tworzy bazę danych uproszczonego systemu bankowego.
2.Projekt został stworzony w oparciu o prawdziwy serwis bankowy.
3.Tabela klient jest tabela matką i tabele lokata,pożyczka,konto,fundusz inwestycyjny,kredyt jej dziećmi (są od niej zależne)

1.1 Możliwości

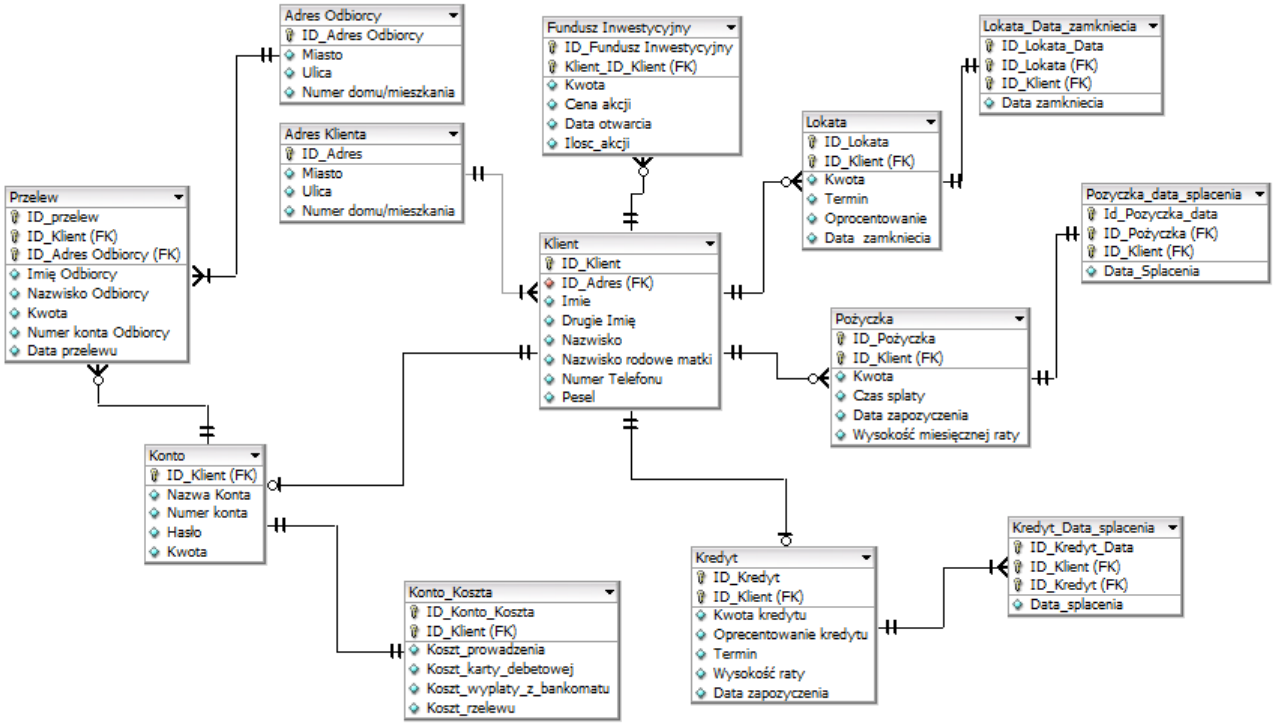
1.Umożliwia ona wyłącznie dodawanie nowych klientów.
2.Umożliwia dodanie klientowi jednego kredytu,wielu
lokat,pożyczek,funduszy.
3.Umożliwia zrealizowanie przelewu
4.Do uzupełniania skomplikowanych tabel zostały stworzone procedury składowane.

1.2 Ograniczenia

1.Klient może mieć tylko jedno konto i kredyt naraz.
3.Przelewy są tylko wyjściowe (ze strony klienta banku) do klienta innego banku. (kiedy podamy numer konta innego klienta banku który jest w systemie przelew nie zadziała i pieniądze nie zostaną wpłacone) 4.Baza banku nie jest bezpieczna,możemy uzupełnić tabele błędnymi danymi.
5.Klientami banku są tylko Polacy
6.Przelewy mogą być tylko realizowane tylko na polskie konta o długości 26 cyfr.
7.Numery telefonów tylko 9 cyfrowe.
8.Baza danych nie jest zabezpieczona przed wprowadzaniem powtórzonych danych

Rozdział 2

Diagram Bazy



Rozdział 3

Spis tabel wraz z opisami

3.1 Tabela Klienta

```
create table Klient(
ID_klient int identity(1,1),
ID_adres int foreign key references Adres_klienta(ID_adres)
on delete cascade on update cascade not null,
Imie varchar(40) not null,
Drugie_imie varchar(40) not null,
Nazwisko varchar(40) not null,
Nazwisko_rodowe_matki varchar(40) not null,
Numer_telefonu varchar(9) unique not null,
Pesel varchar(11) unique not null,
primary key(ID_klient)
)
```

Tabela-matka. Zawiera niezbędne informacje o kliencie. Do zostania klientem banku wymagane są te dane. Jej kluczem obcym jest klucz główny tabeli Adres klienta

3.2 Tabela Adres Klient

```
create table Adres_klienta(
ID_adres int identity(1,1) primary key,
Miasto varchar(40) not null,
Ulica varchar(40) not null,
[Numer domu/mieszkania] varchar(10) not null
)
```

Tabela zawierająca adres klienta wymagany przez bank.
Jest w relacji z tabela klient 1-N

3.3 Tabela Konto

```
create table Konto(
ID_konto int,
foreign key(ID_konto) references Klient(ID_klient)
on delete cascade on update cascade,
Nazwa_konta varchar(20),
Numer_konta varchar(26),
Haslo varchar(20),
Kwota_na_koncie money,
primary key(ID_konto)
)
```

Tabela z 26 cyfrowym numerem konta,nazwa konta to typ konta jakie założył klient,każdy typ konta ma inne przywileje czy koszta. Jest połączona z klientem relacją 1-1

3.4 Tabela Konto koszta

```
create table Konto_koszta(
ID_konto_koszta int identity(1,1),
ID_konto int,
Koszt_prowadzenia money,
Koszt_karty_debetowej money,
Koszt_wyplaty_z_bankomatu money,
Koszt_przelewu money,
foreign key(ID_konto) references Konto(Id_konto)
on delete cascade on update cascade,
primary key (ID_konto,id_konto_koszta)
)
```

Tabela z kosztami związanymi z prowadzeniem konta

3.5 Tabela Przelew

```
create table Przelew(
ID_przelew int identity(1,1),
ID_nadawca int,
ID_adres_odbiorcy int foreign key references adres_odbiorcy(ID_adres_odbiorcy)
on delete cascade on update cascade,
Imie_odbiorcy varchar(40),
Nazwisko_odbiorcy varchar(40),
Kwota money,
Numer_konta_odbiorcy varchar(26),
data  as convert(date,getdate()),
foreign Key(ID_nadawca) references konto(ID_konto)
on delete cascade on update cascade,
primary key (ID_nadawca,Id_przelew,ID_adres_odbiorcy)
)
```

Tabela z podstawowymi informacjami o odbiorcy przelewu wraz z kwotą przelewu i datą wykonania przelewu

3.6 Tabela Adres Odbiorcy

```
create table Adres_odbiorcy(
ID_adres_odbiorcy int identity(1,1),
Miasto varchar(40),
Ulica varchar(40),
[Numer_domu/mieszkania] varchar(40),
primary key(ID_adres_odbiorcy)
)
```

Tabela zawiera adres odbiorcy przelewu jest połączona z tabelą przelew relacją 1-N.

3.7 Tabela Fundusz Inwestycyjny

```
create table Fundusz_inwestycyjny(
ID_fundusz_inwestycyjny int identity(1,1),
ID_klient int,
Kwota money,
Cena_akcji money,
Data_otwarcia as convert(date,getdate()),
ilosc_akcji as CAST(CAST(KWOTA AS DECIMAL)/CAST(cena_akcji AS DECIMAL) AS DECIMAL(18,4)),
foreign key(ID_klient) references Klient(ID_klient)
on delete cascade on update cascade,
primary key(ID_klient,ID_fundusz_inwestycyjny)
)
```

Tabela Zawierająca informacje o funduszu,inwestycyjnym który może założyć klient.

3.8 Tabela Lokata

```
create table Lokata(
ID_lokata int identity(1,1),
ID_klient int,
Kwota money,
Termin int,
Oprocentowanie int,
Data_otwarcia as  convert(date,getdate()),
foreign key(ID_klient) references Klient(ID_klient)
on delete cascade on update cascade,
primary key(ID_klient,ID_lokata)
)
```

Tabela Zawierająca informacje o lokacie którą może założyć klient.

3.9 Tabela Lokata Data zamknięcia

```
create table Lokata_data_zamkniecia(
ID_lokata_data int identity(1,1),
ID_lokata int,
ID_klient int,
Data_zamkniecia date,
foreign key(ID_klient,ID_lokata) references Lokata(ID_klient,ID_lokata)
on delete cascade on update cascade,
primary key(ID_klient,ID_lokata,ID_lokata_data)
)
```

Tabela zawiera informacje o przewidzianej dacie zamknięcia lokaty.

3.10 Tabela Pożyczka

```
create table Pozyczka(
ID_pozyczka int identity(1,1),
ID_klient int,
Kwota money check(Kwota < 200000),
Czas_splaty int check(Czas_splaty <96),
Data_zapozyczenia  as  convert(date,getdate()),
Wysokosc_miesiecznej_raty as cast(Kwota*1.25/Czas_splaty as money),
foreign key(ID_klient) references Klient(ID_klient)
on delete cascade on update cascade,
primary key(ID_klient,ID_pozyczka)
)
```

Tabela zawiera informacje o pożyczce zaciągniętej przez klienta. Maksymalną kwotą pożyczki jest 200 000. Pożyczkę można zaciągnąć maksymalnia na 96 miesięcy Kolumna czas spłaty to czas przeznaczony na spłatę pożyczki wyrażona w miesiącach.

3.11 Tabela Kredyt

```
create table Kredyt(
ID_kredyt int identity(1,1),
ID_klient int unique,
foreign key(ID_klient) references Klient(ID_klient)
on delete cascade on update cascade,
Kwota_kredytu money,
Oprocentowanie_kredytu float,
Termin int,
Wysokość_raty money,
Data_zapozyczenia as convert(date,dateadd(m,+1,getdate())),
primary key(ID_klient,ID_kredyt)
)
```

Tabela zawiera informacje o kredycie klienta. Kolumna Termin to czas przeznaczony na spłatę kredytu wyrażony w latach wyrażona w miesiącach. Nie ma ograniczeń co do kwoty kredytu,terminu.

3.12 Tabela Kredyt Data splacenia

```
create table Kredyt_data_splacenia(
ID_kredyt_data int identity(1,1),
ID_kredyt int,
ID_klient int unique,
Data_splacenia date,
foreign key(ID_klient,ID_kredyt) references Kredyt(ID_klient,ID_kredyt)
on delete cascade on update cascade,
primary key(ID_klient,ID_kredyt,ID_kredyt_data)
)
```

Tabela zawiera informacje o przewidzianej dacie spłacenia kredytu.

Rozdział 4

Spis Wyzwalaczy wraz z opisami

4.1 Wyzwalacz KLIENT CHECK

```
CREATE TRIGGER KLIENT_CHECK ON KLIENT AFTER INSERT
AS
BEGIN
    DECLARE @DLPESEL INT
    DECLARE @DLTEL INT
    SELECT @DLPESEL = DATALENGTH(PESEL) FROM INSERTED
    SELECT @DLTEL = DATALENGTH(NUMER_TELEFONU) FROM INSERTED
    IF @DLPESEL <> 11
    BEGIN
        RAISERROR('PODANO BŁEDNY NUMER PESEL',1,2)
        ROLLBACK
    END
    IF @DLTEL <> 9
    BEGIN
        RAISERROR('PADANO BŁEDNU NUMER TELEFONU',1,3)
        ROLLBACK
    END
END
```

Wyzwalacz sprawdza czy podano właściwą ilość cyfr w peselu(11) i numeru telefonu (9)

4.2 Wyzwalacz Data Kredyt Spłacenia

```
create trigger Data_Kredyt_splacenia on Kredyt after insert
as
begin
    IF OBJECT_ID('Kredyt_data_splacenia') IS NULL
    begin
        RAISERROR('Brak tabeli na dane o dacie splaty kredytu',2,4)
        rollback
    end
    else
    begin
        declare @termin int
        select @termin = Termin from Inserted
        declare @ID_k  int ,@ID_Kl int
        select @ID_k = ID_kredyt from Inserted
        select @ID_Kl = ID_klient from Inserted
        declare @data date
        select @data = convert(date,dateadd(yy,+@termin,getdate()))
        insert into Kredyt_data_splacenia values(@Id_k,@ID_Kl,@data)
    end
end
```

Wyzwalacz dodaje do tabeli Kredyt data spłacenia datę spłaty kredytu (data założenia + termin)

4.3 Wyzwalacz Data Pozyczka

```
create trigger Data_Pozyczka on Pozyczka after insert
as
begin
    IF OBJECT_ID('Pozyczka_data_splacenia') IS NULL
    begin
        RAISERROR('Brak tabeli na dane o dacie spacenia pozyczki',2,4)
        rollback
    end
    else
    begin
        declare @czas int
        select @czas = Czas_splaty from Inserted
        declare @ID_p  int ,@ID_Kl int
        select @ID_p = ID_pozyczka from Inserted
        select @ID_Kl = ID_klient from Inserted
        declare @data date
        select @data = convert(date,dateadd(m,+@czas,getdate()))
        insert into Pozyczka_data_splacenia values(@Id_p,@ID_Kl,@data)
    END
end
```

Wyzwalacz dodaje do tabeli Pozyczka data spłacenia datę spłacenia pożyczki (data założenia + data splacenia)

4.4 Wyzwalacz Data Lokata

```
create trigger Data_Lokata on Lokata after insert
as
begin
IF OBJECT_ID('Lokata_data_zamkniecia') IS NULL
    begin
        RAISERROR('Brak tabeli na dane o dacie zamknieciu lokaty',2,4)
        rollback
    end
    else
    begin
        declare @czas int
        select @czas = Termin from Inserted
        declare @ID_l  int ,@ID_Kl int
        select @ID_l = ID_lokata from Inserted
        select @ID_Kl = ID_klient from Inserted
        declare @data date
        select @data = convert(date,dateadd(m,+@czas,getdate()))
        insert into Lokata_data_zamkniecia values(@Id_l,@ID_Kl,@data)
    end
end
```

Wyzwalacz dodaje do tabeli Lokata data zamkniecia datę zamknięcia lokaty (Data otwarcia + termin)

4.5 Wyzwalacz Przelew bledy

```
CREATE TRIGGER PRZELEW_BLEDY ON PRZELEW AFTER INSERT
AS
BEGIN
    DECLARE @NADAWCA varchar(40)
    DECLARE @ODBIORCA varchar(40)
    SELECT @NADAWCA=K.NUMER_KONTA FROM INSERTED I JOIN KONTO K
    ON I.ID_NADAWCA = K.ID_Konto
    SELECT @ODBIORCA = I.NUMER_KONTA_ODBIORCY FROM INSERTED I
    IF @NADAWCA = @ODBIORCA
    BEGIN
        PRINT('NUMER KONTA ODBIORCY TEN SAM CO NADAWCY')
        RAISERROR('NUMER KONTA ODBIORCY TEN SAM CO NADAWCY',2,4)
        ROLLBACK
    END
    IF LEN(@ODBIORCA) <> 26
    BEGIN
        PRINT('BLEDNY NUMER KONTA ODBIORCY')
        RAISERROR('BLEDNY NUMER KONTA ODBIORCY',2,4)
        ROLLBACK
    END
END
```

Wyzwalacz sprawdza czy w adres nadawcy przelewu i adres odbiorcy przelewu są takie same jeśli tak wyrzuca błąd, i sprawdza czy numer odbiorcy ma odpowiednią ilość cyfr

4.6 Wyzwalacz Data Pożyczka

```
create Trigger Pozyczka_check on Pozyczka after insert
as
begin
    declare @termin int,@kwota money
    select @termin =  Czas_splaty from inserted
    select @kwota = Kwota from inserted
    if @kwota < 1000 and @termin > 6  or @kwota < 10000 and @termin > 48
    begin
        raiserror('za mala kwota na tak dlugi termin',5,2)
        rollback
    end
end
```

Wyzwalacz sprawdza czy kwota i termin pożyczki są właściwe. Bank nie udziela pożyczki np. na 1000 złoty na dłużej niż 6 miesięcy.

4.7 Wyzwalacz Okrągły Klient

```
create Trigger Okragly_klient on Klient after insert
as
    begin
    declare @id_klient int
    select @id_klient  = Id_klient from inserted
    if @id_klient= 100
    print 'Gratulacje jesteś setnym klientem'
    if @id_klient= 1000
    print ' Gratulacje jesteś tysięcznym klientem'
    if @id_klient= 10000
    print 'Gratulacje jesteś dziesieciotysiecznym klientem'
    if @id_klient= 100000
    print 'Gratulacje jesteś sto tysiecznym klientem'
    if @id_klient= 1000000
    print 'Gratulacje jesteś milionowym klientem'
end
```

Wyzwalacz sprawdza czy klient ma okrągły numer,jeśli tak wypisuje wiadomość. Wyzwalacz może być użyteczny kiedy chcemy na przykład nagrodzić takiego klienta.

Rozdział 5

Spis Procedur składowanych

5.1 Procedura dokonaj przelewu

```
create procedure dokonaj_przelewu @id_nadawca int,@imie varchar(40),@nazw varchar(40),@kwota money,
@konto_odbiorcy varchar(40),
@miasto varchar(40),@ulica varchar(40),@numer varchar(40)
as
BEGIN
    IF OBJECT_ID('Przelew') IS NULL
    begin
        raiserror('BRAK TABELI', 13, 1)
        return -1
    end
        IF OBJECT_ID('Adres_odbiorcy') IS NULL
    begin
        raiserror('BRAK TABELI', 13, 1)
        return -1
    end
    if (select id_konto from konto where id_konto = @id_nadawca) is null
    begin
        raiserror('brak klienta o podanym numerze',14,2)
        return -1
    end
    declare @ilosc money
    select @ilosc = (select Kwota_na_koncie from konto
    where ID_konto = @id_nadawca)
    if @ilosc < @kwota
    begin
        raiserror('za mala ilosc funduszy na koncie',12,2)
        return -1
    end
    begin
        insert into adres_odbiorcy values(@miasto,@ulica,@numer)
    end
    begin
        declare @id_adres int
        select @id_adres = id_adres_odbiorcy from adres_odbiorcy
        insert into przelew values(@id_nadawca,@id_adres,@imie,@nazw,@kwota,@konto_odbiorcy)
    end
    begin
        update Konto set Kwota_na_koncie = @ilosc-@kwota where ID_konto = @id_nadawca
    end
END
```

1.Procedura pobiera od użytkownika niezbędne dane do przelewu,
2.Procedura sprawdza czy istnieje tabela przelew
3.adres odbiorcy
4.sprawdza czy istnieje konto o podanym numerze
5.sprawdza czy istnieje na koncie nadawcy wystarczająca ilość środków
6.jeśli wszystkie warunki są spełnione uzupełnia table przelew i odres odbiorcy
7. odejmuje z konta nadawcy kwotę przelewu.

5.2 Procedura dodaj konto

```
create procedure dodaj_konto @nazwa_konta varchar(20) ,@id_klient int ,@kwota money
AS
    IF OBJECT_ID('Konto') IS NULL
    begin
        raiserror('BRAK TABELI', 13, 1)
        return -1
    end
BEGIN
    DECLARE @NR VARCHAR(26)
    set @NR = CONVERT(VARCHAR,ROUND((RAND()*(99-10)+10),0)) +'1240'+
    CONVERT(VARCHAR,ROUND((RAND()*(9999-1000)+1000),0))+
    CONVERT(VARCHAR,ROUND((RAND()*(9999-1000)+1000),0))+
    CONVERT(VARCHAR,ROUND((RAND()*(9999-1000)+1000),0))+
    CONVERT(VARCHAR,ROUND((RAND()*(9999-1000)+1000),0))+
    CONVERT(VARCHAR,ROUND((RAND()*(9999-1000)+1000),0))
    DECLARE @HASLO VARCHAR(20)
    SET @HASLO =  'SA'+convert(varchar,char(rand()*26+65))+convert(varchar,char(rand()*26+65))+convert(varchar,char(rand()*26+65))+
    CONVERT(VARCHAR,ROUND((RAND()*(99-10)+10),0))
    INSERT INTO Konto(id_konto,nazwa_konta,NUMER_KONTA,HASlO,kwota_na_koncie)
    VALUES(@id_klient,@nazwa_konta,@NR,@HASLO,@kwota)
    begin
        IF @nazwa_konta = 'Mlodziezowe'
        insert into konto_koszta(id_konto,koszt_karty_debetowej,koszt_prowadzenia,koszt_przelewu,koszt_wyplaty_z_bankomatu)
        values(@id_klient,0,0,0,3.00)
    end
    if @nazwa_konta = 'Zwykle'
    begin
    insert into konto_koszta(id_konto,koszt_karty_debetowej,koszt_prowadzenia,koszt_przelewu,koszt_wyplaty_z_bankomatu)
    values (@id_klient,3.99,5.99,2.5,5.00)
    end
END
```

Procedura umożliwia szybkie dodanie konta: 1.Sprawdza czy tabela konto istnieje jeśli nie wyrzuca błąd. 2.Tworzy losowy 26 cyfrowy numer konta dla klienta 3.Tworzy losowe hasło 4.Wstawia do tabeli konta odpowiednie wartości 5.wstawia do tabeli konta koszta odpowiednie wartości w zależności od typu konta.

5.3 Procedura Wpłać środki

```
create proc wplac_srodki @id_konto int, @kwota money
as
IF OBJECT_ID('Konto') IS NULL
begin
    raiserror('BRAK TABELI', 13, 1)
    return -1
end
begin
 update konto set kwota_na_koncie = kwota_na_koncie + @kwota
  where id_konto = @id_konto
end
```

Procedura umożliwia szybką wpłatę środków na konto klienta.

5.4 Procedura Zmień hasło

```
CREATE PROCEDURE ZMIEN_HASLO @ID_KONTA INT ,@HASLO VARCHAR(20)
AS
BEGIN
    IF OBJECT_ID('Konto') IS NULL
    begin
        raiserror('BRAK TABELI', 13, 1)
        return -1
    end
    UPDATE KONTO SET HAslO= @HASLO
    WHERE ID_KONTO = @ID_KONTA
END
```

Procedura umożliwia na zmianę hasła konta klienta.

5.5 Procedura Dodaj kredyt

```
create procedure dodaj_Kredyt  @id_klient int, @Kwota Money, @procent float ,@termin int
as
begin
    declare @rata money
    declare @wsp float
    set @procent = @procent* 0.01
    select @wsp = 1 + (@procent*0.12)
    select @rata = convert(money,(@KWOTA * power(@wsp,@termin) * (@wsp-1)/(power(@wsp,@termin-1))))
    insert into Kredyt values(@id_klient,@Kwota,@procent,@termin,@rata)
end
```

Procedura umożliwia na szybkie dodanie kredytu, oblicza ratę kredytu w oparciu na wzór wyliczania raty kredytu.

Rozdział 6

Spis Funkcji składowanych

6.1 Funkcja Sprawdź konta koszta

```
create function Sprawdz_koszta_konta(@imie varchar(40), @nazwisko varchar(40))
returns table as
return(select
    Koszt_prowadzenia,
    Koszt_karty_debetowej,
    Koszt_wyplaty_z_bankomatu,
    Koszt_przelewu
    from Konto k  join konto_koszta ko
    on  k.ID_konto = ko.Id_konto
    join klient kli
    on k.id_konto = kli.id_klient
    where
    (kli.imie= @imie and
    kli.nazwisko = @nazwisko)
)
```

Funkcja zwraca tabelę z kosztami konta klienta o podanym imieniu i nazwisku.

6.2 Funkcja rodzinaklienta

```
create function rodzinaKlient (@imie varchar(40),@nazwisko varchar(40))
returns table as
return
    (select nazwisko,k.id_adres ,count(*) as LiczbaCzlonkowRodziny from
    klient k join Adres_klienta ak on k.id_adres = ak.id_adres
    where nazwisko = @nazwisko
    group by nazwisko,k.id_adres
    having count(*) > 1
)
```

Funkcja zwraca tabele z ilością członków rodziny klienta o podanym imieniu i nazwisku założeniem że osoby o tym samym nazwisku i mieszkającym w tym samym domu są rodziną.

Rozdział 7

Spis widoków

7.1 Widok Tranzakcje

```
CREATE VIEW TRANZAKCJE
AS
select LEFT(K.IMIE,1) + '.' + K.NAZWISKO AS NADAWCA,LEFT(P.IMIE_ODBIORCY,1)+ '.' +
 P.NAZWISKO_ODBIORCY AS ODBIORCA,P.DATA AS DATA_PRZELEWU,P.KWOTA AS KWOTA_PRZELEWU
FROM KLIENT K JOIN PRZELEW P ON
K.ID_KLIENT = P.ID_NADAWCA
```

Widok na tranzakcje klienta

7.2 Widok Podglad konta klienta

```
CREATE VIEW PODGLAD_KONTA_KLIENTA
AS
SELECT K.IMIE,K.NAZWISKO,KO.NUMER_KONTA, KO.KWOTA_NA_KONCIE FROM KONTO KO JOIN KLIENT K ON
KO.ID_Konto = K.ID_KLIENT
```

Szybki podgląd konta klienta

7.3 Widok Starsi Klienci

```
create view Starsi_klienci
as
select Imie,Nazwisko,Data_zapozyczenia from
Klient k join Kredyt Kr
on k.Id_klient = Kr.ID_klient
join Konto ko
on k.ID_klient = ko.ID_konto
where datediff(year,Kr.Data_zapozyczenia,getdate())>10
and ko.Id_konto is not null
```

Widok na klientów ,którzy mają przynajmniej od dziesięciu lat kredyt i posiadają konto.

7.4 Widok Wysokie kredyty

```
create view Wysokie_kredyty
as
select LEFT(K.IMIE,1) + '.' + K.NAZWISKO AS KlIENT,Kwota_kredytu from
klient k join Kredyt kk
on k.Id_klient = kk.id_klient
where kk.kwota_kredytu > 400000
```

Widok na na klientów którzy mają kredyt większy niż 400000

7.5 Widok Klient koniec lokaty

```
create view Klient_Koniec_Lokaty
as
select imie,nazwisko,data_otwarcia,data_zamkniecia
from klient k join Lokata l
on k.Id_klient = l.ID_klient
join lokata_data_zamkniecia ldz
on k.id_klient = ldz.id_klient
where datediff(m,getdate(),ldz.Data_zamkniecia) < 2
```

Widok na klientów którym kończy się lokata (2 miesiące)

7.6 Widok Inwestorzy

```
create view Inwestorzy
as
select imie,nazwisko,ilosc_akcji
from klient k join fundusz_inwestycyjny fi
on k.id_klient = fi.id_klient
where ilosc_akcji > 200
```

Widok na klientów którzy mają więcej niż 200 akcji funduszu

7.7 Widok topkredyt

```
create view topkredyt
as
select  top 10 kwota_kredytu,imie,nazwisko,termin
from klient k join kredyt kr
on k.id_klient = kr.id_klient
order by kwota_kredytu desc
```

Widok na 10 klientów z najwyższym kredytem w kolejności malejącej.

7.8 Widok wysokieprzelewy

```
create view Wysokieprzelewy
as
select ID_klient,Id_przelew,Kwota,Numer_konta,numer_konta_odbiorcy
from klient k join Konto ko
on k.id_klient = ko.id_konto
join przelew p
on p.id_nadawca = ko.ID_konto
where p.Kwota > 5000
```

Widok na przelewy większe niż 5000.