Bazy Danych

Patryk Kapłan, Jakub Nowak

2022

Opis bazy danych

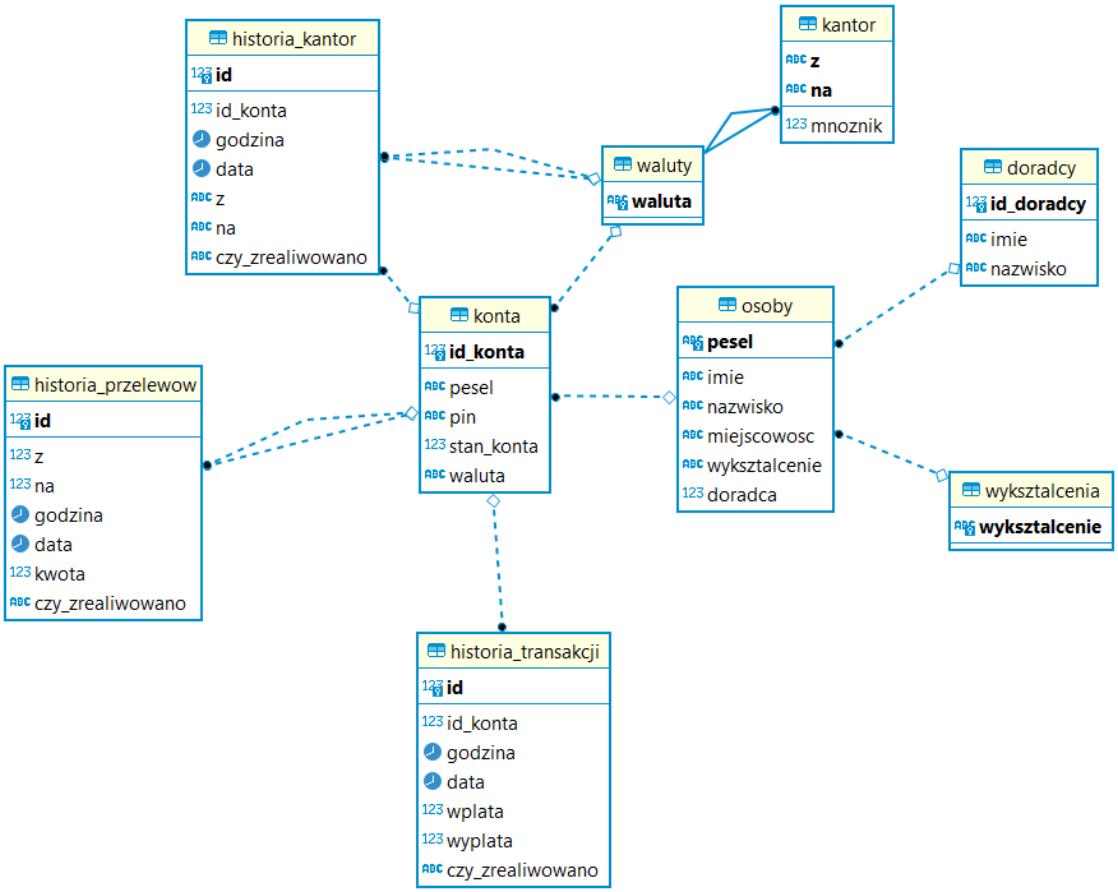
Projekt przedstawia bazę danych banku, która składa się z 9 tabel. W naszej bazie danych możemy znaleźć między innymi:

- Podstawowe informacje o użytkownikach banku. Każdy użytkownik może mieć kilka kont, natomiast tylko jednego doradcę.

Każdy użytkownik ma możliwość:

- Wpłacania/wypłacania pieniędzy z konta.
- Zmiana waluty swojego konta.
- Robienia przelewów.

Schemat bazy danych



Rysunek 1: Schemat bazodanowy

Kod generujący baze danych oraz jego omówienie

Tabela osoby:

Tabela przechowuje informacje na temat klientów banku. W tabeli znajduje się jeden klucz główny, jest to pesel użytkownika, wymagamy aby pesel miał 11 znaków. Dodatkowo mamy dwa klucze obce.

```
CREATE TABLE osoby (
    pesel VARCHAR(11) NOT NULL PRIMARY KEY CHECK (LENGTH(pesel)=11),
    imie VARCHAR(30) NOT NULL,
    nazwisko VARCHAR(30) NOT NULL,
    miejscowosc VARCHAR(30) NOT NULL,
  wyksztalcenie VARCHAR(30) NOT NULL REFERENCES wyksztalcenia(wyksztalcenie) ON UPDATE
    CASCADE ON DELETE RESTRICT,
  doradca INTEGER REFERENCES doradcy(id_doradcy) ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Tabela wykształcenia:

```
CREATE TABLE wyksztalcenia (
    wyksztalcenie VARCHAR(30) PRIMARY KEY NOT NULL
);

INSERT INTO wyksztalcenia VALUES
  ('podstawowe'), ('zawodowe'), ('srednie'), ('wyzsze');
```

Tabela doradcy:

```
CREATE TABLE doradcy (
  id_doradcy INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    imie VARCHAR(30) NOT NULL,
  nazwisko VARCHAR(30) NOT NULL,
  UNIQUE(imie, nazwisko)
);

INSERT INTO doradcy (imie, nazwisko) VALUES ('Bogdan', 'Jagoda');
INSERT INTO doradcy (imie, nazwisko) VALUES ('Jan', 'Malinka');
```

Tabela konta:

Tabela przechowuje wszystkie konta użytkowników. Atrybut id konta można utożsamić z numerem konta w banku, każdy użytkownik ma inny numer konta czyli naturalnie jest to klucz główny. Każdy użytkownik może mieć kilka kont, więc wymagamy aby pesel użytkownika i pin do konta był unikatowy tzn chcemy, żeby jeden użytkownik miał inne piny do każdego ze swoich kont. Dodatkowo każde konto może być w innej walucie. Lista dostępnych walut przechowywana jest w tabeli waluta.

```
CREATE TABLE konta (
    id_konta INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    pesel VARCHAR(11) NOT NULL REFERENCES osoby(pesel) ON UPDATE CASCADE ON DELETE
    CASCADE,
  pin VARCHAR(4) NOT NULL CHECK (LENGTH(pin)=4),
  stan_konta DECIMAL(100,2) NOT NULL,
  waluta VARCHAR(10) NOT NULL REFERENCES waluty(waluta) ON UPDATE CASCADE ON DELETE
    RESTRICT,
  UNIQUE(pesel, pin)
);
```

Tabela waluty:

```
CREATE TABLE waluty (
    waluta VARCHAR(10) PRIMARY KEY NOT NULL
);

INSERT INTO waluty VALUES
  ('PLN'), ('EUR'), ('USD'), ('GBP');
```

Tabela historia transakcji:

W tej tabeli przechowujemy informacje o wpłacani/wypłacaniu pieniędzy ze wszystkich kont dostępnych w banku. Do atrybutów wpłata, wypłata został napisany specjalny warunek, który wymaga aby tylko jeden z tych atrybutów był uzupełniony, tzn chcemy aby było coś wpisane tylko w atrybucie wpłata lub tylko w atrybucie wypłata. Dodatkowo każda transakcja może przyjmować trzy stany t,f,e. Stan f-false oznacza, że transakcje nie została zrealizowana, tzn. nie zadziałał wyzwalacz. Stan t-true oznacza, że wyzwalacz zadziałał i transakcja została wykonana poprawnie. Natomiast stan e-error oznacza, że wyzwalacz zadziałał, ale operacja się nie powiodła np. ktoś chciał wypłacić z konta 500zł a miał tylko 300zł.

```
CREATE TABLE historia_transakcji (
  id INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    id_konta INTEGER NOT NULL REFERENCES konta(id_konta) ON UPDATE CASCADE ON DELETE
    CASCADE,
  godzina TIME NOT NULL,
  data DATE NOT NULL,
  wplata INTEGER,
  wyplata INTEGER,
  CHECK((wplata IS NOT NULL OR wyplata IS NOT NULL) AND (wplata+wyplata IS NULL) AND (
    wplata>0) AND (wyplata>0)),
  czy_zrealiwowano VARCHAR(1) NOT NULL DEFAULT 'f',
  CHECK(czy_zrealiwowano IN ('t', 'f', 'e'))
);
```

Funkcja i wyzwalacz do powyższej tabeli:

```
CREATE OR REPLACE FUNCTION aktualizuj_transakcje() RETURNS TRIGGER AS $$
DECLARE
  stan INTEGER;
BEGIN
  NEW.godzina=current_time;
  NEW.data=current_date;
  IF (NEW.czy_zrealiwowano='f') THEN
    IF (NEW.wplata IS NOT NULL) THEN
      UPDATE konta SET stan_konta=stan_konta+NEW.wplata WHERE konta.id_konta=NEW.id_konta
    ;
      NEW.czy_zrealiwowano='t';
    END IF;



    IF (NEW.wyplata IS NOT NULL) THEN
      SELECT stan_konta INTO stan FROM konta WHERE konta.id_konta=NEW.id_konta;
      IF (stan>=NEW.wyplata) THEN
        UPDATE konta SET stan_konta=stan_konta-NEW.wyplata WHERE konta.id_konta=NEW.
    id_konta;
        NEW.czy_zrealiwowano='t';
      ELSE
        NEW.czy_zrealiwowano='e';
      END IF;

    END IF;


  END IF;
  RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER aktualizuj_transakcje_trigger BEFORE INSERT ON historia_transakcji
  FOR EACH ROW EXECUTE PROCEDURE aktualizuj_transakcje();
```

Tabela kantor:

W tej tabeli przechowujemy informacje o przelicznikach walut. Wymagamy aby atrybuty z i na były unikatowe, ponieważ nie chcemy dopuścić do niejednoznaczności.

```sql
CREATE TABLE kantor (
  z VARCHAR(10) NOT NULL REFERENCES waluty(waluta) ON UPDATE CASCADE ON DELETE RESTRICT,
  na VARCHAR(10) NOT NULL REFERENCES waluty(waluta) ON UPDATE CASCADE ON DELETE RESTRICT,
  mnoznik DECIMAL(2,1) NOT NULL,
  UNIQUE(z, na)
);
```

Tabela historia kantor:

Tabela przechowuje całą historię zmian walut danych kont. Jest bardzo podobna do tabeli historia transakcji. Warto wspomnieć o atrybucie z, nie wymagamy aby był on NOT NULL ponieważ wyzwalacz sam wpisze tam odpowiednią wartość.

```sql
CREATE TABLE historia_kantor (
  id INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    id_konta INTEGER NOT NULL REFERENCES konta(id_konta) ON UPDATE CASCADE ON DELETE
    CASCADE,
  godzina TIME NOT NULL,
  data DATE NOT NULL,
  z VARCHAR(10) REFERENCES waluty(waluta) ON UPDATE CASCADE ON DELETE RESTRICT,
  na VARCHAR(10) NOT NULL REFERENCES waluty(waluta) ON UPDATE CASCADE ON DELETE RESTRICT,
  czy_zrealiwowano VARCHAR(1) NOT NULL DEFAULT 'f',
  CHECK(czy_zrealiwowano IN ('t', 'f', 'e'))
);
```

Funkcja i wyzwalacz do powyższej tabeli:

```sql
CREATE OR REPLACE FUNCTION przewalutuj() RETURNS TRIGGER AS $$
DECLARE
  stan INTEGER;
  mnoznik DECIMAL(2,1);
  waluta_teraz VARCHAR(10);
BEGIN
  NEW.godzina=current_time;
  NEW.data=current_date;
  SELECT stan_konta INTO stan FROM konta WHERE NEW.id_konta=konta.id_konta;
  SELECT konta.waluta INTO waluta_teraz FROM konta WHERE NEW.id_konta=konta.id_konta;
  NEW.z=waluta_teraz;

  IF(stan IS NULL) THEN
    NEW.czy_zrealiwowano='e';
    RETURN NEW;
  END IF;


  SELECT kantor.mnoznik INTO mnoznik FROM kantor WHERE (kantor.z=waluta_teraz) AND (
    kantor.na=NEW.na);
  IF(mnoznik IS NULL) THEN
    NEW.czy_zrealiwowano='e';
    RETURN NEW;
  END IF;

  UPDATE konta SET stan_konta=stan_konta*mnoznik WHERE id_konta=NEW.id_konta;
  UPDATE konta SET waluta=NEW.na WHERE id_konta=NEW.id_konta;
  NEW.czy_zrealiwowano='t';

  RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER przewalutuj_trigger BEFORE INSERT ON historia_kantor
  FOR EACH ROW EXECUTE PROCEDURE przewalutuj();
```

Tabela historia przelewów:

Kolejna tabela przechowująca historię. Tym razem historię przelewów. Struktura tabeli bardzo podobna do powyższych tabel z historią. W tym przypadku warto wspomnieć o stanie e-error. Stan ten otrzymamy

gdy spróbujemy przelać pieniądze na konta o różnych walutach oraz gdy będziemy chcieli przelać kwotę jaką nie dysponujemy.

```
CREATE TABLE historia_przelewow (
  id INTEGER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    z INTEGER NOT NULL REFERENCES konta(id_konta) ON UPDATE CASCADE ON DELETE CASCADE,
  na INTEGER NOT NULL REFERENCES konta(id_konta) ON UPDATE CASCADE ON DELETE CASCADE,
  godzina TIME,
  data DATE NOT NULL,
  kwota INTEGER NOT NULL,
  czy_zrealiwowano VARCHAR(1) NOT NULL DEFAULT 'f',
  CHECK(czy_zrealiwowano IN ('t', 'f', 'e')),
  CHECK(kwota>0)
);
```

Funkcja i wyzwalacz do powyższej tabeli:

```
CREATE OR REPLACE FUNCTION aktualizuj_przelewy() RETURNS TRIGGER AS $$
DECLARE
  stan INTEGER;
  waluta_z VARCHAR(10);
  waluta_na VARCHAR(10);
BEGIN
  NEW.godzina=current_time;
  NEW.data=current_date;
  IF (NEW.czy_zrealiwowano='f') THEN
    IF (NEW.z=NEW.na) THEN
      UPDATE historia_przelewow SET czy_zrealiwowano='e' WHERE historia_przelewow.id=NEW.
    id;
    END IF;

    IF (NEW.z!=NEW.na) THEN
      SELECT stan_konta INTO stan FROM konta WHERE konta.id_konta=NEW.z;
      SELECT waluta INTO waluta_z FROM konta WHERE konta.id_konta=NEW.z;
      SELECT waluta INTO waluta_na FROM konta WHERE konta.id_konta=NEW.na;
      IF (stan>=NEW.kwota AND waluta_z=waluta_na) THEN
        UPDATE konta SET stan_konta=stan_konta+NEW.kwota WHERE konta.id_konta=NEW.na;
        UPDATE konta SET stan_konta=stan_konta-NEW.kwota WHERE konta.id_konta=NEW.z;
        NEW.czy_zrealiwowano='t';
      ELSE
        NEW.czy_zrealiwowano='e';
      END IF;
    END IF;

  END IF;


RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER aktualizuj_przelewy_trigger BEFORE INSERT ON historia_przelewow
  FOR EACH ROW EXECUTE PROCEDURE aktualizuj_przelewy();
```

Widok "kto komu doradza":

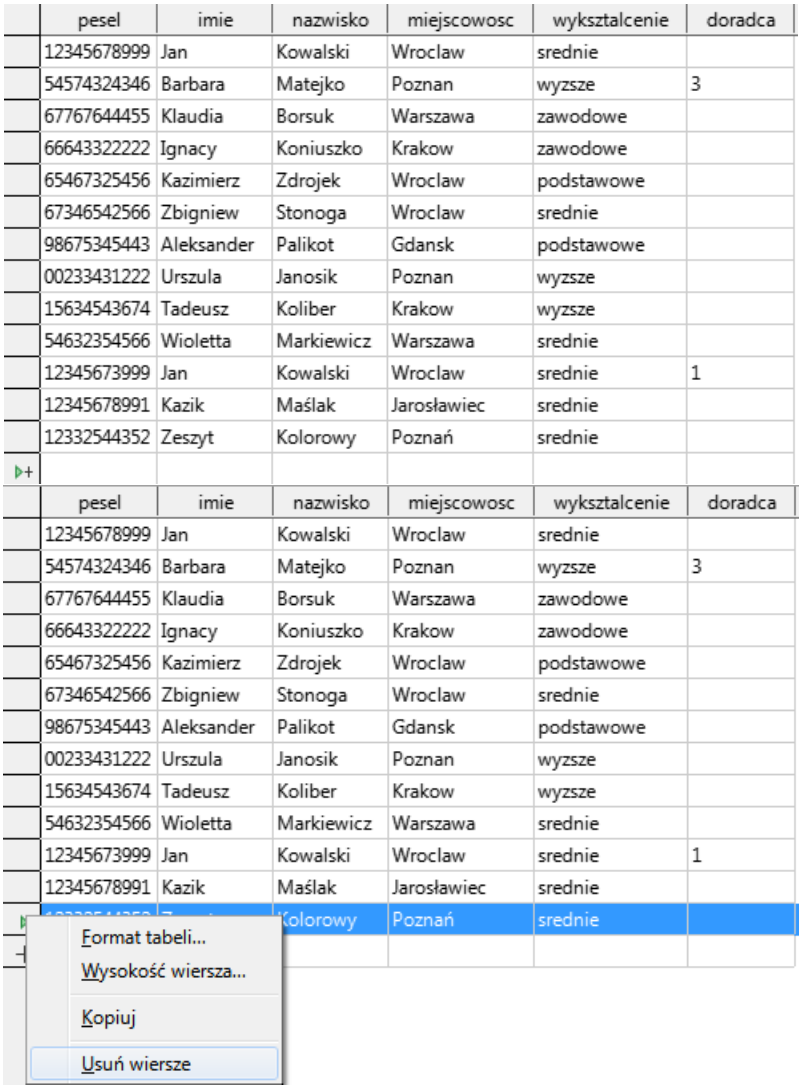
Widok pokazujący klientów i ich doradców w jednej tabeli.

```
CREATE VIEW kto_komu_doradza AS
SELECT osoby.imie, osoby.nazwisko, doradcy.imie AS "imie doradcy", doradcy.nazwisko AS "
    nazwisko doradcy" FROM osoby, doradcy
WHERE osoby.doradca=doradcy.id_doradcy;
```

Interfejs graficzny

Bazę danych podłączyliśmy do LibreOffice Base. Dzięki czemu dodawanie nowych krotek, czy administrowanie bazą staje się dużo prostsze dla człowieka nie słyszącego o PostgreSQL. Przykładowe zdjęcia,

które w żadnym stopniu nie są w stanie pokazać potęcjału LibreOffice Base, po prostu trzeba spróbować samemu :)



pesel	imie	nazwisko	miejscowosc	wyksztalcenie	doradca
12345678999	Jan	Kowalski	Wroclaw	srednie	
54574324346	Barbara	Matejko	Poznan	wyzsze	3
67767644455	Klaudia	Borsuk	Warszawa	zawodowe	
66643322222	Ignacy	Koniuszko	Krakow	zawodowe	
65467325456	Kazimierz	Zdrojek	Wroclaw	podstawowe	
67346542566	Zbigniew	Stonoga	Wroclaw	srednie	
98675345443	Aleksander	Palikot	Gdansk	podstawowe	
00233431222	Urszula	Janosik	Poznan	wyzsze	
15634543674	Tadeusz	Koliber	Krakow	wyzsze	
54632354566	Wioletta	Markiewicz	Warszawa	srednie	
12345673999	Jan	Kowalski	Wroclaw	srednie	1
12345678991	Kazik	Maślak	Jarosławiec	srednie	
12332544352	Zeszyt	Kolorowy	Poznań	srednie	

pesel	imie	nazwisko	miejscowosc	wyksztalcenie	doradca
12345678999	Jan	Kowalski	Wroclaw	srednie	
54574324346	Barbara	Matejko	Poznan	wyzsze	3
67767644455	Klaudia	Borsuk	Warszawa	zawodowe	
66643322222	Ignacy	Koniuszko	Krakow	zawodowe	
65467325456	Kazimierz	Zdrojek	Wroclaw	podstawowe	
67346542566	Zbigniew	Stonoga	Wroclaw	srednie	
98675345443	Aleksander	Palikot	Gdansk	podstawowe	
00233431222	Urszula	Janosik	Poznan	wyzsze	
15634543674	Tadeusz	Koliber	Krakow	wyzsze	
54632354566	Wioletta	Markiewicz	Warszawa	srednie	
12345673999	Jan	Kowalski	Wroclaw	srednie	1
12345678991	Kazik	Maślak	Jarosławiec	srednie	