Rozdział 17. Zarządzanie współbieżnością – zadania dodatkowe

-- Definicje relacji i utworzenie stanu początkowego dla ćwiczeń z synchronizacji transakcji

```
DROP TABLE Konta cascade constraints;
DROP TABLE Przelewy_zlecone cascade constraints;
DROP TABLE Przelewy_wykonane cascade constraints;
DROP TABLE Przelewy_realizowane cascade constraints;

DROP SEQUENCE seqPZ;
DROP SEQUENCE seqPW;
DROP SEQUENCE seqPDZ;


CREATE TABLE Konta(
numer NUMBER(26) PRIMARY KEY, id_klienta NUMBER(9),
saldo NUMBER(12,2));

CREATE TABLE Przelewy_zlecone(
id NUMBER(9) PRIMARY KEY,
nr_konta NUMBER(26) REFERENCES Konta(numer),
data_zlecenia DATE, data_realizacji DATE,
konto_docelowe NUMBER(26) REFERENCES Konta(numer),
kwota NUMBER(12,2));

CREATE TABLE Przelewy_wykonane(
id NUMBER(9) PRIMARY KEY,
nr_konta NUMBER(26) REFERENCES Konta(numer),
data_realizacji DATE,
konto_docelowe NUMBER(26) REFERENCES Konta(numer),
data_księgowania DATE, kwota NUMBER(12,2));

CREATE TABLE Przelewy_realizowane(
id NUMBER(9) PRIMARY KEY,
nr_konta NUMBER(26) REFERENCES Konta(numer),
konto_źródłowe NUMBER(26) REFERENCES Konta(numer),
kwota NUMBER(12,2), data DATE);

INSERT INTO Konta VALUES (11111,1,10000);
INSERT INTO Konta VALUES (55555,2,200000);

CREATE SEQUENCE seqPZ START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seqPW START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE seqPDZ START WITH 1 INCREMENT BY 1;

INSERT INTO Przelewy_zlecone VALUES (seqPZ.NEXTVAL, 11111, DATE
'2014-10-31', DATE '2014-11-4', 55555, 5000);

COMMIT;
```

```
-- Polecenia dla odtworzenia stanu bazy danych po nieudanych próbach symchronizacji
-- UPDATE Konta SET saldo = 10000 WHERE numer = 11111;
-- DELETE FROM Przelewy_zlecone;
-- INSERT INTO Przelewy_zlecone VALUES (seqPZ.NEXTVAL, 11111, DATE
'2014-10-31', DATE '2014-11-4', 55555, 5000);
-- COMMIT
```

Dany jest schemat relacyjnej bazy danych.

Przelewy_zlecone(id, nr_konta, data_zlecenia, data_realizacji, konto_docelowe,kwota)

Przelewy_wykonane(id, nr_konta, data_realizacji, konto_docelowe, data_księgowania, kwota)

Przelewy_realizowane (id, nr_konta, konto_źródłowe, kwota, data)

Konto(numer, id_klienta, saldo)

W każdym z poniższych zadań, zweryfikuj poprawność podanych współbieżnych historii. Ustal, czy występują w nich jakieś znane Ci anomalie? Następnie zmodyfikuj transakcje tak , żeby wynik ich przetwarzania był poprawny. Modyfikacje transakcji mogą obejmować:

1. zmiany granic transakcji,
2. zmianę domyślnego trybu izolacji,
3. jawne blokowanie tabel,
4. jawne blokowanie wierszy.

Zidentyfikuj uniwersalny problem, który kryje się za przypadkiem z każdego z zadań i zaproponuj metodykę dla rozwiązania zdefiniowanego problemu.

Zadanie nr 1

Sesja 1: Realizuje zleconą operację przelewu

```
variable id number
variable nr_konta number
variable konto_doc number
variable data varchar2(10)
variable kwota number
exec :nr_konta := 11111
exec :data := '14/11/04'

-- begin transaction
exec select id, nr_konta, data_realizacji, konto_docelowe, kwota
into :id, :nr_konta, :data, :konto_doc, :kwota from Przelewy_zlecone
where nr_konta = :nr_konta and data_realizacji =  :data;
-- przyjmujemy, że istnieje dokładnie jedna taka krotka
commit;
exec insert into Przelewy_realizowane values (seqPDZ.nextval, :konto_doc,
     :nr_konta, :kwota, :data);
exec insert into Przelewy_wykonane values (seqPW.nextval,:nr_konta,
     :data,:konto_doc,null,:kwota);
exec update Konta set saldo = saldo - :kwota where numer = :nr_konta;
commit;
-- 1.1
exec delete from Przelewy_zlecone where id = :id;
-- 1.2
commit;
```

Sesja 2: Wyznacza dostępne środki na koncie jako różnicę salda i sumy wszystkich oczekujących operacji przelewów.

```
variable saldo_skorygowane number
variable nr_konta number
exec :nr_konta := 11111
-- begin transaction
exec select saldo - (select sum(kwota) from Przelewy_zlecone where
     nr_konta = :nr_konta) into :saldo_skorygowane from Konta where
     numer = :nr_konta;
print :saldo_skorygowane;
-- poprawna wartość 5000 zł
commit;
-- 2.1
```

Zweryfikuj poprawność współbieżnej historii powyższych dwóch sesji dla uporządkowania operacji: 1.1 < 2.1 < 1.2.

Po każdej nieudanej próbie, przywróć początkowy stan bazy danych !!!!!!!!!!

Zadanie nr 2

Przywróć początkowy stan bazy danych i sprawdź, czy dostępne są wszystkie zmienne wiązania zdefiniowane w zadaniu nr 1.

Sesja 1: Realizuje zleconą operację przelewu

```
-- begin transaction
exec select id, nr_konta, data_realizacji, konto_docelowe, kwota
into :id, :nr_konta, :data, :konto_doc, :kwota from Przelewy_zlecone
where nr_konta = :nr_konta and data_realizacji =  :data;
-- przyjmujemy, że istnieje dokładnie jedna taka krotka
exec insert into Przelewy_realizowane values (seqPDZ.nextval, :konto_doc,
     :nr_konta, :kwota, :data);
exec insert into Przelewy_wykonane values (seqPW.nextval,:nr_konta,
     :data,:konto_doc,null,:kwota);
exec update Konta set saldo = saldo - :kwota where numer = :nr_konta;
exec delete from Przelewy_zlecone where id = :id;
-- 1.1
commit;
```

Sesja 2: Wyznacza dostępne środki na koncie

```
-- aplikacja zaimplementowana w środowisku O/RM – dwie operacje odczytu
variable obciążenia number
exec :nr_konta := 11111
-- begin transaction
exec select sum(kwota) into :obciążenia from Przelewy_zlecone where
     nr_konta = :nr_konta;
print obciążenia
-- 2.1
exec select (saldo – :obciążenia) into :saldo_skorygowane from Konta
     where numer = :nr_konta;
print saldo_skorygowane
-- poprawna wartość 5000 zł
commit;
-- 2.2
```

Zweryfikuj poprawność współbieżnej historii powyższych dwóch sesji dla przykładowego uporządkowania operacji: 2.1 < 1.1 < 2.2. Jak należy zmodyfikować powyższe sesje, żeby niezależnie od kolejności operacji w historii współbieżnej tych dwóch sesji, transakcja w sesji nr 2 zawsze zwracała poprawną wartość dostępnych środków na koncie.

Zadanie nr 3

Sesja 1 realizuje operację wpłaty na wybrane konto:

```
-- aplikacja zaimplementowana w środowisku O/RM w związku z czym
-- przetwarzanie danych odbywa się po stronie serwera aplikacji,
-- a nie serwera bazy danych

variable saldo number
-- begin transaction
exec select saldo into :saldo from Konta where numer = 11111;
print saldo
exec :saldo := :saldo + 2500;
-- 1.1
exec update konta set saldo = :saldo where numer = 11111;
print saldo
-- ... inne operacje
commit;
-- 1.2
```

Sesja 2 realizuje natychmiastowy przelew pieniędzy na wybrane konto:

```
variable saldo number
-- begin transaction
exec select saldo into :saldo from Konta where numer = 11111;
print saldo
exec :saldo := :saldo - 500;
-- 2.1
exec update konta set saldo = :saldo where numer = 11111;
print saldo
-- ... inne operacje
commit;
exec select saldo into :saldo from konta where numer = 11111;
print saldo
-- 2.2
-- poprawna wartość 12 000 zł
```

Zweryfikuj poprawność współbieżnej historii powyższych dwóch sesji dla przykładowego uporządkowania operacji: 1.1 < 2.1 < 1.2 < 2.2. Zmodyfikuj powyższe transakcje w taki sposób, żeby niezależnie od kolejności operacji w historii współbieżnej tych dwóch sesji, stan salda konta był zawsze poprawny.

Zadanie nr 3a

Problem do rozwiązania taki jak z zadaniu 3, ale dodatkowo wprowadzone modyfikacje nie mogą blokować transakcji.

Zadanie nr 4

Obydwie sesje realizują operację pobrania określonej kwoty z danego konta. Wykonanie operacje jest możliwe tylko w wypadku, gdy saldo konta jest większe niż pobierana kwota.

Przywróć początkowy stan bazy danych

Sesja 1:

```
variable saldo_skorygowane number
exec :nr_konta := 11111
-- begin transaction
exec select saldo into :saldo from Konta where numer = :nr_konta;
exec :saldo_skorygowane := :saldo - 8000
print saldo_skorygowane
-- if (:saldo – 8000 > 0) then begin
-- 1.1
exec update Konta set saldo = saldo – 8000 where numer = :nr_konta;
-- end if;
commit;
-- 1.2
exec select saldo into :saldo from Konta where numer = :nr_konta;
exec :saldo_skorygowane := :saldo - 8000
print saldo_skorygowane
-- 1.3
-- wynikiem powinna być wartość większa niż 0
```

Sesja 2:

```
exec :nr_konta := 11111
-- begin transaction
exec select saldo into :saldo from Konta where numer = :nr_konta;
exec :saldo_skorygowane := :saldo - 5000
print saldo_skorygowane
-- if (:saldo – 5000 > 0) then begin
-- 2.1
exec update Konta set saldo = saldo – 5000 where numer = :nr_konta;
-- end if;
commit;
-- 2.2
```

Zweryfikuj poprawność współbieżnej historii powyższych dwóch sesji dla przykładowego uporządkowania operacji: 1.1 < 2.1 < 1.2 < 2.2 < 1.3. Poprawność będzie zapewniona gdy suma przelewów zleconych dla danego konta nie będzie większa niż saldo tego konta.

Zadanie nr 5

Sesje 1 i 2 realizują dwa współbieżne wątki tego samego procesu przyjmowania zleceń przelewu. Przyjęcie przelewu jest możliwe tylko w wypadku, gdy saldo konta pomniejszone o już zaplanowane przelewy jest większe od przelewanej kwoty.

Sesja 1:

```
-- begin transaction
exec select saldo - (select sum(kwota) from Przelewy_zlecone where
     nr_konta = 11111) into :saldo_skorygowane from Konta where
     numer = 11111;
exec :saldo_skorygowane := :saldo_skorygowane - 3000
print saldo_skorygowane
-- if (:saldo_skorygowane – 3000 > 0) then begin
-- 1.1
insert into Przelewy_zlecone values (
          seqPZ.nextval, 11111, sysdate, sysdate+1, 55555, 3000)
-- end if;
commit;
-- 1.2
```

Sesja 2:

```
-- begin transaction
exec select saldo - (select sum(kwota) from Przelewy_zlecone where
     nr_konta = 11111) into :saldo_skorygowane from Konta where
     numer = 11111;
exec :saldo_skorygowane := :saldo_skorygowane - 3000
print saldo_skorygowane
-- if (:saldo_skorygowane – 4000 > 0) then begin
-- 2.1
insert into Przelewy_zlecone values (
          seqPZ.nextval, 11111, sysdate, sysdate+2, 5555, 4000);
-- end if;
exec select saldo - (select sum(kwota) from Przelewy_zlecone where
     nr_konta = 11111) into :saldo_skorygowane from Konta where
     numer = 11111;
print saldo_skorygowane
-- Poprawna wartość > 0
commit;
-- 2.2
```

Zweryfikuj poprawność współbieżnej historii powyższych dwóch sesji dla przykładowego uporządkowania operacji: 1.1 < 2.1 < 1.2 < 2.2. Poprawność będzie zapewniona gdy suma przelewów zleconych dla danego konta nie będzie większa niż saldo tego konta.

Zadanie nr 6

Zagwarantuj poprawne wykonanie transakcji z poprzedniego zadania bez względu na dowolne inne nieznane transakcje, które przetwarzają tę samą bazę danych.

Sesja 1:

```
-- begin transaction
exec select saldo - (select sum(kwota) from Przelewy_zlecone where
     nr_konta = 11111) into :saldo_skorygowane from Konta where
     numer = 11111;
exec :saldo_skorygowane := :saldo_skorygowane - 3000
print saldo_skorygowane
-- if (:saldo_skorygowane – 3000 > 0) then begin
-- 1.1
insert into Przelewy_zlecone values (
          seqPZ.nextval, 11111, sysdate, sysdate+1, 55555, 3000)
-- end if;
commit;
-- 1.2
exec select saldo - (select sum(kwota) from Przelewy_zlecone where
     nr_konta = 11111) into :saldo_skorygowane from Konta where
     numer = 11111;
print saldo_skorygowane
-- Poprawna wartość > 0
-- 1.3
```

Sesja 2: Dowolna inna transakcja W tym wypadku jest to transakcja, która przyjmuje zlecenie przelewu tylko, jeżeli suma przelewów jest mniejsza od 10 000 zł.

```
variable suma_przelewów number
variable kwota_zlecona number
exec :kwota_zlecona := 4000
-- begin transaction
exec select sum(kwota) from Przelewy_zlecone into :suma_przelewów
     from Konta where numer = 11111;
-- if (:suma_przelewów + :kwota_zlecona < 10 000 ) then begin
-- 2.1
insert into Przelewy_zlecone values (
          seqPZ.nextval, 11111, sysdate, sysdate+2, 5555, :kwota_zlecona);
commit;
exec select sum(kwota) from Przelewy_zlecone into :suma_przelewów
     from Konta where numer = 11111;
print suma_przelewów
-- Poprawna wartość < 10 000
-- 2.2
```

Zweryfikuj poprawność współbieżnej historii powyższych dwóch sesji dla przykładowego uporządkowania operacji: 1.1 < 2.1 < 1.2 < 2.2 < 1.3