Ćwiczenie 1

Karol Hamielec

3/10/2020

zadanie 3

Tworzenie widoków. Należy przygotować kilka widoków ułatwiających dostęp do danych. Należy zwrócić uwagę na strukturę kodu (należy unikać powielania kodu)

-

```
create view REZERWACJEWSZYSTKIE as
SELECT w.ID_WYCIECZKI,
     w.NAZWA,
     w.KRAJ,
     w.DATA,
     o.IMIE,
     o.NAZWISKO,
     r.STATUS
FROM WYCIECZKI w
       JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
       JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
```

-

b) RezerwacjePotwierdzone (kraj,data, nazwa_wycieczki, imie, nazwisko,status_rezerwacji)

```
create view REZERWACJEPOTWIERDZONE as
SELECT r.ID_WYCIECZKI,
       r.NAZWA,
       r.KRAJ,
       r.DATA,
       r.IMIE,
       r.NAZWISKO,
       r.STATUS
FROM REZERWACJEWSZYSTKIE r
        where r.STATUS = 'P' OR r.STATUS = 'Z'
```

-

c) RezerwacjeWPrzyszlosci (kraj,data, nazwa_wycieczki, imie, nazwisko,status_rezerwacji)

```
 create view REZERWACJEWPRZYSZLOSCI as
 SELECT ID_WYCIECZKI,
        NAZWA,
        KRAJ,
```

```
       DATA,
       IMIE,
       NAZWISKO,
       STATUS
FROM RezerwacjeWszystkie
where DATA  > SYSDATE
```

- d) WycieczkiMiejsca(kraj,data, nazwa_wycieczki,liczba_miejsc, liczba_wolnych_miejsc)

```
create view WYCIECZKIMIEJSCA as
SELECT w.Kraj as kraj,
           w.data as data,
           w.NAZWA as nazwa_wyczieczki,
           w.liczba_miejsc as liczba_miejsc,
           w.LICZBA_MIEJSC - COUNT(R.ID_OSOBY) as liczba_wolnych_miejsc
FROM WYCIECZKI w inner join REZERWACJE R on W.ID_WYCIECZKI = R.ID_WYCIECZKI
group by w.Kraj, w.data, w.NAZWA, w.liczba_miejsc
```

- e) WycieczkiDostepne(kraj,data, nazwa_wycieczki,liczba_miejsc, liczba_wolnych_miejsc)

```
create view WYCIECZKIDOSTEPNE as
SELECT kraj,data, nazwa_wyczieczki,liczba_miejsc, liczba_wolnych_miejsc
    from WycieczkiMiejsca
    where data > SYSDATE AND liczba_wolnych_miejsc > 0
```

zadanie 4

Tworzenie procedur/funkcji pobierających dane. Podobnie jak w poprzednim przykładzie należy przygotować kilka procedur ułatwiających dostęp do danych

- a) UczestnicyWycieczki (id_wycieczki), procedura ma zwracać podobny zestaw danych jak widok RezerwacjeWszystkie

```
   CREATE OR REPLACE TYPE uczestnik AS OBJECT
(
        NAZWA VARCHAR2(100),
        KRAJ VARCHAR2(50),
        DATA DATE,
        IMIE VARCHAR2(50),
        NAZWISKO VARCHAR2(50),
        STATUS CHAR(1)
)
CREATE OR REPLACE TYPE t_uczestnik IS TABLE OF uczestnik

CREATE OR REPLACE FUNCTION uczestnicy_wycieczki
(wycieczka_id INT)
RETURN  t_uczestnik
AS
```

```
    wynik t_uczestnik;
    istnieje INT;
BEGIN
    SELECT COUNT(*) into istnieje  FROM WYCIECZKI where wycieczka_id = wycieczka_id;

    IF istnieje = 0 then
        RAISE_APPLICATION_ERROR(-20101,'taka wycieczka nie istnieje');
    end if;

    SELECT uczestnik(NAZWA, KRAJ, DATA, IMIE, NAZWISKO, STATUS) BULK COLLECT into wynik from REZ
    return wynik;
end;
```

- b) RezerwacjeOsoby(id_osoby), procedura ma zwracać podobny zestaw danych jak widok wycieczki_osoby

```
   CREATE OR REPLACE TYPE rezerwacje_osoby AS OBJECT
(
       ID_WYCIECZKI INT,
       NAZWA VARCHAR2(50),
       KRAJ VARCHAR2(50),
       DATA DATE,
       STATUS CHAR(1)
)
CREATE OR REPLACE TYPE t_rezerwacje_osoby IS TABLE OF rezerwacje_osoby

CREATE OR REPLACE FUNCTION RezerwacjeOsoby(vid_osoby INT)
RETURN t_rezerwacje_osoby
AS
    wynik t_rezerwacje_osoby;
    istnieje INT;
BEGIN
    SELECT COUNT(*) into istnieje FROM OSOBY where ID_OSOBY = vid_osoby;
    if istnieje = 0 then
        RAISE_APPLICATION_ERROR(-20101, 'taka osoba nie istnieje');
    end if;

    SELECT REZERWACJE_OSOBY(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA, r.STATUS) BULK COLLECT into
    return wynik;
end;
```

- c) DostepneWycieczki(kraj, data_od, data_do)

```
CREATE OR REPLACE TYPE wycieczka AS OBJECT
(
  ID_WYCIECZKI INT
, NAZWA VARCHAR2(100)
, KRAJ VARCHAR2(50)
, DATA DATE
, OPIS VARCHAR2(200)
, LICZBA_MIEJSC INT
)
CREATE OR REPLACE TYPE t_wycieczka IS TABLE OF wycieczka
```

```
create FUNCTION DostepneWycieczki(vkraj VARCHAR2, od DATE, do DATE)
    RETURN T_WYCIECZKA
as
    wynik T_WYCIECZKA;
begin
    SELECT wycieczka(ID_WYCIECZKI, NAZWA, KRAJ, OPIS, DATA, LICZBA_MIEJSC) BULK COLLECT
    INTO wynik
    FROM WYCIECZKI
    WHERE KRAJ = vkraj
      AND DATA >= od
      AND DATA <= do
      AND LICZBA_MIEJSC -
          ((SELECT COUNT(*) FROM REZERWACJE where WYCIECZKI.ID_WYCIECZKI = REZERWACJE.ID_WYCIECZ
    return wynik;
end;
```

zad 5.

Tworzenie procedur modyfikujących dane. Należy przygotować zestaw procedur pozwalających na modyfikację danych oraz kontrolę poprawności ich wprowadzania

- a) DodajRezerwacje(id_wycieczki, id_osoby), procedura powinna kontrolować czy wycieczka jeszcze się nie odbyła, i czy sa wolne miejsca

```
CREATE OR REPLACE PROCEDURE DodajRezerwacje(vid_wycieczki INT, vid_osoby INT )
AS
    istnieje INT;
    wolne_miejsca INT;
BEGIN
    SELECT COUNT(*) INTO istnieje FROM WYCIECZKI WHERE vid_wycieczki = ID_WYCIECZKI and DATA > SYSD

    if istnieje = 0 then
        RAISE_APPLICATION_ERROR(-20101, 'taka wycieczka nie istnieje lub już się odbyła');
    end if;
    SELECT LICZBA_MIEJSC - COUNT(*) INTO wolne_miejsca FROM REZERWACJE r join WYCIECZKI w on r.ID_W

    if wolne_miejsca < 1 then
        raise_application_error(-20101, 'niestety, nie ma już wolnych miejsc');
    end if;
    SELECT COUNT(*) INTO istnieje FROM OSOBY WHERE vid_osoby = ID_OSOBY;
    if istnieje = 0 then
        RAISE_APPLICATION_ERROR(-20101, 'taka osoba nie istnieje');
    end if;
    SELECT COUNT(*) INTO istnieje FROM REZERWACJE where ID_OSOBY = vid_osoby and vid_wycieczki = ID
    if istnieje > 0 then
        raise_application_error(-20101, 'taka rezerwacja juz istnieje');
    end if;

    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS) VALUES (vid_wycieczki, vid_osoby, 'N');
end;
```

- b) ZmienStatusRezerwacji(nr_rezerwacji, status), procedura kontrolować czy możliwa jest zmiana statusu, np. zmiana statusu już anulowanej wycieczki (przywrócenie do stanu aktywnego nie zawsze jest możliwa – może już nie być miejsc)

```
CREATE OR REPLACE PROCEDURE zmienStatusRezerwacji(vnr_rezerwacji INT, vstatus CHAR)
AS
    istnieje INT;
    wolne_miejsca INT;
    wycieczka INT;
    status_teraz INT;
BEGIN
    SELECT COUNT(*) INTO istnieje FROM REZERWACJE where NR_REZERWACJI = vnr_rezerwacji;
    if istnieje = 0 then
        raise_application_error(-20101, 'taka rezerwacja nie istnieje');
    end if;
    SELECT COUNT(*) INTO istnieje FROM REZERWACJE r join WYCIECZKI w on r.ID_WYCIECZKI = w.ID_WYCIE
    if istnieje != 0 then
        raise_application_error(-20101, 'wycieczka z tej rezerwacji juz sie odbyla');
    end if;

    SELECT w.ID_WYCIECZKI INTO wycieczka FROM REZERWACJE r join WYCIECZKI w on r.ID_WYCIECZKI = w.I
    SELECT LICZBA_MIEJSC - COUNT(*) INTO wolne_miejsca FROM REZERWACJE r join WYCIECZKI w on r.ID_W
    SELECT STATUS into status_teraz FROM REZERWACJE r where NR_REZERWACJI = vnr_rezerwacji;
    if status_teraz = 'A' and wolne_miejsca < 1 then
        raise_application_error(-20101, 'niestety na ta wycieczke nie ma juz miejsc');
    end if;

    if vstatus = 'N' then
        raise_application_error(-20101,'nie mozna zmienic statusu na nowy');
    end if;

    if vstatus = status_teraz then
        raise_application_error(-20101,'ten status jest juz ustawiony');
    end if;
    UPDATE REZERWACJE SET STATUS = vstatus where NR_REZERWACJI = vnr_rezerwacji;
end;
```

- c) ZmienLiczbeMiejsc(id_wycieczki, liczba_miejsc), nie wszystkie zmiany liczby miejsc są dozwolone, nie można zmniejszyć liczby miesc na wartość poniżej liczby zarezerwowanych miejsc

```
CREATE OR REPLACE PROCEDURE ZmienLiczbeMiejsc(vid_wycieczki INT, vliczba_miejsc INT)
as
    zajete_miejsca INT;
    istnieje INT;
BEGIN
    SELECT COUNT(*) INTO istnieje FROM WYCIECZKI where ID_WYCIECZKI = vid_wycieczki;

    if istnieje = 0 then
        raise_application_error(-20101, 'taka wycieczka nie istnieje');
    end if;

    SELECT COUNT(*) into zajete_miejsca FROM REZERWACJE where ID_WYCIECZKI = vid_wycieczki;
    if zajete_miejsca > vliczba_miejsc then
        raise_application_error(-20101, 'nie mozna zmienic liczby miejsc na ilosc mniejsza niz aktu
    end if;

    UPDATE WYCIECZKI SET LICZBA_MIEJSC = vliczba_miejsc WHERE ID_WYCIECZKI = vid_wycieczki;
end;
```

zad 6.

Dodajemy tabelę dziennikującą zmiany statusu rezerwacji

- Rezerwacje_log(id, id_rezerwacji, data, status)

```
CREATE TABLE REZERWACJE_LOG(
    ID INT,
    ID_REZERWACJI INT,
    DATA DATE,
    STATUS CHAR(1),
    CONSTRAINT REZERWACJE_LOG_PK PRIMARY KEY
    (
        ID
    )
    ENABLE
);
```

- Należy zmienić warstwę procedur modyfikujących dane tak aby dopisywały informację do dziennika

```
create PROCEDURE DodajRezerwacje(vid_wycieczki INT, vid_osoby INT )
AS
    istnieje INT;
    wolne_miejsca INT;
    id_rezerw INT;
BEGIN
    SELECT COUNT(*) INTO istnieje FROM WYCIECZKI WHERE vid_wycieczki = ID_WYCIECZKI and DATA > SYSD
    if istnieje = 0 then
        RAISE_APPLICATION_ERROR(-20101, 'taka wycieczka nie istnieje lub już się odbyła');
    end if;
    SELECT LICZBA_MIEJSC - COUNT(*) INTO wolne_miejsca FROM REZERWACJE r join WYCIECZKI w on r.ID_W
    if wolne_miejsca < 1 then
        raise_application_error(-20101, 'niestety, nie ma już wolnych miejsc');
    end if;
    SELECT COUNT(*) INTO istnieje FROM OSOBY WHERE vid_osoby = ID_OSOBY;
    if istnieje = 0 then
        RAISE_APPLICATION_ERROR(-20101, 'taka osoba nie istnieje');
    end if;
    SELECT COUNT(*) INTO istnieje FROM REZERWACJE where ID_OSOBY = vid_osoby and vid_wycieczki = ID
    if istnieje > 0 then
        raise_application_error(-20101, 'taka rezerwacja juz istnieje');
    end if;

    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS) VALUES (vid_wycieczki, vid_osoby, 'N');

    SELECT NR_REZERWACJI INTO id_rezerw FROM REZERWACJE where ID_WYCIECZKI = vid_wycieczki and ID_O

    INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS) VALUES (id_rezerw, SYSDATE, 'N');
end;
```

```
create PROCEDURE zmienStatusRezerwacji(vnr_rezerwacji INT, vstatus CHAR)
AS
    istnieje INT;
    wolne_miejsca INT;
    wycieczka INT;
    status_teraz INT;
```

```
BEGIN
    SELECT COUNT(*) INTO istnieje FROM REZERWACJE where NR_REZERWACJI = vnr_rezerwacji;
    if istnieje = 0 then
        raise_application_error(-20101, 'taka rezerwacja nie istnieje');
    end if;
    SELECT COUNT(*) INTO istnieje FROM REZERWACJE r join WYCIECZKI w on r.ID_WYCIECZKI = w.ID_WYCIE
    if istnieje != 0 then
        raise_application_error(-20101, 'wycieczka z tej rezerwacji juz sie odbyla');
    end if;

    SELECT w.ID_WYCIECZKI INTO wycieczka FROM REZERWACJE r join WYCIECZKI w on r.ID_WYCIECZKI = w.I
    SELECT LICZBA_MIEJSC - COUNT(*) INTO wolne_miejsca FROM REZERWACJE r join WYCIECZKI w on r.ID_W
    SELECT STATUS into status_teraz FROM REZERWACJE r where NR_REZERWACJI = vnr_rezerwacji;
    if status_teraz = 'A' and wolne_miejsca < 1 then
        raise_application_error(-20101, 'niestety na ta wycieczke nie ma juz miejsc');
    end if;

    if vstatus = 'N' then
        raise_application_error(-20101,'nie mozna zmienic statusu na nowy');
    end if;

    if vstatus = status_teraz then
        raise_application_error(-20101,'ten status jest juz ustawiony');
    end if;
    UPDATE REZERWACJE SET STATUS = vstatus where NR_REZERWACJI = vnr_rezerwacji;
    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS) VALUES (vnr_rezerwacji, SYSDATE, vstat
end;
/
```

zad 7.

```
ALTER TABLE WYCIECZKI
ADD liczba_wolnych_miejsc INT;
```

Zmiana struktury bazy danych, w tabeli wycieczki dodajemy redundantne pole liczba_wolnych_miejsc

- Należy zmodyfikować zestaw widoków. Proponuję dodać kolejne widoki (np. z sufiksem 2), które pobierają informację o wolnych miejscach z nowo dodanego pola. (Widok WYCIECZKIDOSTEPNE nie został zmieniony, ponieważ korzysta z pierwszego widoku i nie było takiej potrzeby)

```
create view WYCIECZKIMIEJSCA2 as
SELECT w.Kraj as kraj,
           w.data as data,
           w.NAZWA as nazwa_wycieczki,
           w.liczba_miejsc as liczba_miejsc,
           w.liczba_wolnych_miejsc
FROM WYCIECZKI w inner join REZERWACJE R on W.ID_WYCIECZKI = R.ID_WYCIECZKI
group by w.Kraj, w.data, w.NAZWA, w.liczba_miejsc, w.liczba_wolnych_miejsc
```

- Należy napisać procedurę przelicz która zaktualizuje wartość liczby wolnych miejsc dla już istniejących danych.

```
CREATE PROCEDURE przelicz
as
begin
```

```
    UPDATE WYCIECZKI SET liczba_wolnych_miejsc = LICZBA_MIEJSC - (SELECT COUNT(*) FROM REZERWACJE W
end;
```

```
create FUNCTION DostepneWycieczki2(vkraj VARCHAR2, od DATE, do DATE)
    RETURN T_WYCIECZKA
as
    wynik T_WYCIECZKA;
begin
    SELECT wycieczka(ID_WYCIECZKI, NAZWA, KRAJ, OPIS, DATA, LICZBA_MIEJSC) BULK COLLECT
    INTO wynik
    FROM WYCIECZKI
    WHERE KRAJ = vkraj
      AND DATA >= od
      AND DATA <= do
      AND LICZBA_WOLNYCH_MIEJSC > 0;
    return wynik;
end;
```

- Należy zmodyfikować warstwę procedur pobierających dane, podobnie jak w przypadku widoków.

```
create PROCEDURE DodajRezerwacje2(vid_wycieczki INT, vid_osoby INT )
AS
    istnieje INT;
    wolne_miejsca INT;
    id_rezerw INT;
BEGIN
    SELECT COUNT(*) INTO istnieje FROM WYCIECZKI WHERE vid_wycieczki = ID_WYCIECZKI and DATA > SYSD
    if istnieje = 0 then
        RAISE_APPLICATION_ERROR(-20101, 'taka wycieczka nie istnieje lub już się odbyła');
    end if;

    SELECT LICZBA_WOLNYCH_MIEJSC INTO wolne_miejsca FROM WYCIECZKI where ID_WYCIECZKI = vid_wyciecz

    if wolne_miejsca < 1 then
        raise_application_error(-20101, 'niestety, nie ma już wolnych miejsc');
    end if;
    SELECT COUNT(*) INTO istnieje FROM OSOBY WHERE vid_osoby = ID_OSOBY;
    if istnieje = 0 then
        RAISE_APPLICATION_ERROR(-20101, 'taka osoba nie istnieje');
    end if;
    SELECT COUNT(*) INTO istnieje FROM REZERWACJE where ID_OSOBY = vid_osoby and vid_wycieczki = ID
    if istnieje > 0 then
        raise_application_error(-20101, 'taka rezerwacja juz istnieje');
    end if;

    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS) VALUES (vid_wycieczki, vid_osoby, 'N');

    SELECT NR_REZERWACJI INTO id_rezerw FROM REZERWACJE where ID_WYCIECZKI = vid_wycieczki and ID_O
    UPDATE WYCIECZKI SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1 where ID_WYCIECZKI = vid

    INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS) VALUES (id_rezerw, SYSDATE, 'N');
end;
```

```
create PROCEDURE zmienStatusRezerwacji2(vnr_rezerwacji INT, vstatus CHAR)
AS
```

```
    istnieje INT;
    wolne_miejsca INT;
    wycieczka INT;
    status_teraz INT;
BEGIN
    SELECT COUNT(*) INTO istnieje FROM REZERWACJE where NR_REZERWACJI = vnr_rezerwacji;
    if istnieje = 0 then
        raise_application_error(-20101, 'taka rezerwacja nie istnieje');
    end if;
    SELECT COUNT(*) INTO istnieje FROM REZERWACJE r join WYCIECZKI w on r.ID_WYCIECZKI = w.ID_WYCIE
    if istnieje != 0 then
        raise_application_error(-20101, 'wycieczka z tej rezerwacji juz sie odbyla');
    end if;

    SELECT w.ID_WYCIECZKI INTO wycieczka FROM REZERWACJE r join WYCIECZKI w on r.ID_WYCIECZKI = w.I
    SELECT LICZBA_WOLNYCH_MIEJSC INTO wolne_miejsca FROM WYCIECZKI where ID_WYCIECZKI = wycieczka;
    SELECT STATUS into status_teraz FROM REZERWACJE r where NR_REZERWACJI = vnr_rezerwacji;
    if status_teraz = 'A' and wolne_miejsca < 1 then
        raise_application_error(-20101, 'niestety na ta wycieczke nie ma juz miejsc');
    end if;

    if vstatus = 'N' then
        raise_application_error(-20101,'nie mozna zmienic statusu na nowy');
    end if;

    if vstatus = status_teraz then
        raise_application_error(-20101,'ten status jest juz ustawiony');
    end if;
    UPDATE REZERWACJE SET STATUS = vstatus where NR_REZERWACJI = vnr_rezerwacji;
    if vstatus = 'A' then
            UPDATE WYCIECZKI SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + 1 where ID_WYCIECZ
    end if;
    if vstatus != 'A' and status_teraz = 'A' then
            UPDATE WYCIECZKI SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1 where ID_WYCIECZ
    end if;
    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS) VALUES (vnr_rezerwacji, SYSDATE, vstat
end;
```

zad 8.

Zmiana strategii zapisywania do dziennika rezerwacji. Realizacja przy pomocy triggerów. Należy wprowadzić zmianę która spowoduje że zapis do dziennika rezerwacji będzie realizowany przy pomocy trierów

zad 9. (połączyłem podpunkty, ponieważ dotyczyły wspólnych obiektów)

Zmiana strategii obsługi redundantnego pola liczba_wolnych_miejsc . realizacja przy pomocy trigerów

- triger obsługujący dodanie rezerwacji

```
CREATE OR REPLACE TRIGGER dodaj_rezerwacje_trig
    AFTER INSERT
    ON REZERWACJE
    FOR EACH ROW
```

```
    BEGIN
        INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS) VALUES (:NEW.NR_REZERWACJI, SYSDATE
        UPDATE WYCIECZKI SET WYCIECZKI.liczba_wolnych_miejsc = WYCIECZKI.liczba_wolnych_miejsc - 1
    end;
```

- triger obsługujący zmianę statusu

```
CREATE OR REPLACE TRIGGER zmien_status_trig
    AFTER UPDATE
    ON REZERWACJE
    FOR EACH ROW
    DECLARE
    nowe_miejsca INT;
BEGIN
        INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS) VALUES (:NEW.NR_REZERWACJI, SYSDAT
        if :NEW.STATUS = 'A' THEN
            nowe_miejsca := 1;
        elsif :OLD.STATUS = 'A' THEN
            nowe_miejsca := -1;
        else
            nowe_miejsca := 0;
        end if;

        UPDATE WYCIECZKI SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + nowe_miejsca WHERE ID_

    end;
```

- triger zabraniający usunięcia rezerwacji

```
CREATE OR REPLACE TRIGGER usun_rezerwacje_trig
 BEFORE DELETE
 ON REZERWACJE
 FOR EACH ROW
 BEGIN
     raise_application_error(-20101, 'nie mozna usuwac rezerwacji');
 end;
```

- triger obsługujący zmianę liczby miejsc na poziomie wycieczki

```
   CREATE OR REPLACE TRIGGER zmien_liczbe_miejsc_trig
       AFTER UPDATE OF liczba_miejsc
       ON WYCIECZKI
       FOR EACH ROW
       BEGIN
           UPDATE WYCIECZKI SET LICZBA_WOLNYCH_MIEJSC = liczba_wolnych_miejsc + (:NEW.liczba_miejsc
       end;
```