

Krzysztof Bieniasz

Sprawozdanie z wykonania ćwiczenia nr 1: Oracle PL/SQL

Zadanie 1.

W zadaniu pierwszym należało stworzyć trzy tabele według poleceń z instrukcji.

```
create table OSOBY
(
    ID_OSOBY NUMBER generated as identity
        constraint OSOBY_PK
            primary key,
    IMIE      VARCHAR2(50),
    NAZWISKO  VARCHAR2(50),
    PESEL     VARCHAR2(11),
    KONTAKT   VARCHAR2(100)
)

create table WYCIECZKI
(
    ID_WYCIECZKI NUMBER generated as identity
        constraint WYCIECZKI_PK
            primary key,
    NAZWA         VARCHAR2(100),
    KRAJ          VARCHAR2(50),
    DATA         DATE,
    OPIS          VARCHAR2(200),
    LICZBA_MIEJSC NUMBER
)

create table REZERWACJE
(
    NR_REZERWACJI NUMBER generated as identity
        constraint REZERWACJE_PK
            primary key,
    ID_WYCIECZKI  NUMBER
        constraint REZERWACJE_FK2
            references WYCIECZKI,
    ID_OSOBY      NUMBER
        constraint REZERWACJE_FK1
            references OSOBY,
    STATUS        CHAR
        constraint REZERWACJE_CHK1
            check (status IN ('N', 'P', 'Z', 'A'))
)
```

Zadanie 2.

W zadaniu drugim należało wypełnić tabelę przykładowymi danymi. W tabeli OSOBY utworzyłem 10 rekordów, w tabeli WYCIECZKI 5, natomiast w tabeli REZERWACJE 15, wypełniając je takimi danymi, aby móc dobrze przetestować stworzone później widoki i funkcję.

Poniżej przedstawiam wywołania selecta dla kolejno tabel: OSOBY, WYCIECZKI, REZERWACJE

Tabela OSOBY

	ID_OSOBY	IMIE	NAZWISKO	PESEL	KONTAKT
1	1	Adam	Kowalski	87654321	tel: 6623
2	2	Jan	Nowak	12345678	tel: 2312, dzwonić po 18.00
3	3	Robert	Lewandowski	17654321	tel: 9098
4	4	Arkadiusz	Milik	12909878	tel: 8812
5	5	Artur	Boruc	90989098	tel: 5732
6	6	Dawid	Ziobro	57325678	tel: 9900
7	7	Kamil	Kowalski	11654321	tel: 1123
8	8	Kamil	Glik	43345678	tel: 4342, dzwonić po 18.00
9	9	Arkadiusz	Klich	88809811	tel: 1112
10	10	Artur	Bielik	90900008	tel: 9433

Tabela WYCIECZKI

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC
1	1	Wycieczka do Paryza	Francja	2016-01-01 00:00:00	Ciekawa wycieczka ...	3
2	2	Piękny Kraków	Polska	2020-02-03 00:00:00	Najciekawa wycieczka ...	4
3	3	Wieliczka	Polska	2020-03-03 00:00:00	Zadziwiająca kopalnia ...	4
4	4	Piękny Rzeszów	Polska	2020-01-03 00:00:00	Najciekawa wycieczka ...	5
5	5	Bieszczady	Polska	2019-10-23 00:00:00	Piękne połoniny ...	7

Tabela REZERWACJE

	NR_REZERWACJI	ID_WYCIECZKI	ID_OSOBY	STATUS
1	23	1	1	Z
2	24	1	2	Z
3	25	1	3	A
4	26	2	4	N
5	27	2	5	P
6	28	2	6	Z
7	29	3	7	N
8	30	3	8	P
9	31	3	9	Z
10	32	4	4	N
11	33	4	5	Z
12	34	4	6	Z
13	35	5	4	N
14	36	5	5	P
15	37	5	6	Z

Zadanie 3.

W zadaniu trzecim należało zaimplementować widoki ułatwiające korzystanie z bazy danych.

a) Widok wycieczki_osoby - widok pokazujący ID_WYCIECZKI, NAZWĘ, KRAJ, DATĘ, IMIĘ UCZESTNIKA, NAZWISKO UCZESTNIKA oraz STATUS REZERWACJI

```
create view WYCIECZKI_OSOBY 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
```

Przykład działania:

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	IMIE	NAZWISKO	STATUS
1	1	Wycieczka do Paryza	Francja	2016-01-01 00:00:00	Adam	Kowalski	Z
2	1	Wycieczka do Paryza	Francja	2016-01-01 00:00:00	Jan	Nowak	Z
3	1	Wycieczka do Paryza	Francja	2016-01-01 00:00:00	Robert	Lewandowski	A
4	2	Piękny Kraków	Polska	2020-02-03 00:00:00	Arkadiusz	Milik	N
5	2	Piękny Kraków	Polska	2020-02-03 00:00:00	Artur	Boruc	P
6	2	Piękny Kraków	Polska	2020-02-03 00:00:00	Dawid	Ziobro	Z
7	3	Wieliczka	Polska	2020-03-03 00:00:00	Kamil	Kowalski	N

b) Widok wycieczki_osoby_potwierdzone – widok pokazujący ID_WYCIECZKI, NAZWĘ, KRAJ, DATĘ, IMIĘ UCZESTNIKA, NAZWISKO UCZESTNIKA oraz STATUS REZERWACJI. Rezerwację potwierdzone to takie, które mają status P lub Z.

```
CREATE or REPLACE VIEW WYCIECZKI_OSOBY_POTWIERDZONE 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
where r.STATUS = 'P' OR r.STATUS = 'Z';
```

Przykład działania:

W porównaniu do przykładu w podpunkcie a) następuje odfiltrowanie rekordów o statusie innym niż P lub Z

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	IMIE	NAZWISKO	STATUS
	1	Wycieczka do Paryza	Francja	2016-01-01 00:00:00	Adam	Kowalski	Z
	1	Wycieczka do Paryza	Francja	2016-01-01 00:00:00	Jan	Nowak	Z
	2	Piękny Kraków	Polska	2020-02-03 00:00:00	Artur	Boruc	P
	2	Piękny Kraków	Polska	2020-02-03 00:00:00	Dawid	Ziobro	Z
	3	Wieliczka	Polska	2020-03-03 00:00:00	Kamil	Glik	P
	3	Wieliczka	Polska	2020-03-03 00:00:00	Arkadiusz	Klich	Z
	4	Piękny Rzeszów	Polska	2020-01-03 00:00:00	Artur	Boruc	Z

c) Widok wycieczki_przyszle – widok pokazujący pola KRAJ, DATA, NAZWA_WYCIECZKI, IMIĘ UCZESTNIKA, NAZWISKO UCZESTNIKA oraz STATUS REZERWACJI.
Sprawdzam warunek czy data wycieczki jest późniejsza niż aktualna data.

```
CREATE or REPLACE view WYCIECZKI_PRZYSZLE as
SELECT w.KRAJ,
       w.DATA,
       w.NAZWA,
       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
WHERE w.DATA > SYSDATE
```

Przykład działania:

	KRAJ	DATA	NAZWA	IMIE	NAZWISKO	STATUS
1	Polska	2020-02-03 00:00:00	Piękny Kraków	Arkadiusz	Milik	N
2	Polska	2020-02-03 00:00:00	Piękny Kraków	Artur	Boruc	P
3	Polska	2020-02-03 00:00:00	Piękny Kraków	Dawid	Ziobro	Z
4	Polska	2020-03-03 00:00:00	Wieliczka	Kamil	Kowalski	N
5	Polska	2020-03-03 00:00:00	Wieliczka	Kamil	Glik	P
6	Polska	2020-03-03 00:00:00	Wieliczka	Arkadiusz	Klich	Z
7	Polska	2020-01-03 00:00:00	Piękny Rzeszów	Arkadiusz	Milik	N

d) Widok wycieczki_miejsc – widok pokazujący pola KRAJ, DATA, NAZWA_WYCIECZKI, LICZBA_MIEJSC oraz LICZBA_WOLNYCH_MIEJSC.
Widok pokazuje ile miejsc zostało wykorzystanych w wycieczkach przeszłych oraz ile miejsc jest aktualnie ważne zarezerwowanych w wycieczkach przyszłych

```
CREATE or REPLACE view WYCIECZKI_MIEJSCA as
select w.KRAJ,
       w.DATA,
       w.NAZWA,
       w.LICZBA_MIEJSC,
       w.LICZBA_MIEJSC - count(*) as LICZBA_WOLNYCH_MIEJSC
from WYCIECZKI w
left join REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
where r.STATUS IN ('N', 'P', 'Z')
group by w.KRAJ, w.DATA, w.NAZWA, w.LICZBA_MIEJSC;
```

Przykład działania:

	KRAJ	DATA	NAZWA	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	Polska	2020-02-03 00:00:00	Piękny Kraków	4	1
2	Polska	2020-03-03 00:00:00	Wieliczka	4	1
3	Polska	2020-01-03 00:00:00	Piękny Rzeszów	5	2
4	Polska	2020-03-10 00:00:00	Bieszczady	7	2
5	Francja	2016-01-01 00:00:00	Wycieczka do Paryża	3	1

e) Widok dostępne_wycieczki – widok pokazujący pola KRAJ, DATA, NAZWA_WYCIECZKI, LICZBA_MIEJSC oraz LICZBA_WOLNYCH_MIEJSC, dla wycieczek, które odbędą się w przyszłości i nie są jeszcze w pełni obsadzone

```
CREATE or REPLACE view DOSTĘPNE_WYCIECZKI as
select * from
(select w.KRAJ,
w.DATA,
w.NAZWA,
w.LICZBA_MIEJSC,
w.LICZBA_MIEJSC - count(*) as LICZBA_WOLNYCH_MIEJSC
from WYCIECZKI w
left join REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
where r.STATUS IN ('N', 'P', 'Z')
group by w.KRAJ, w.DATA, w.NAZWA, w.LICZBA_MIEJSC)
where LICZBA_WOLNYCH_MIEJSC > 0 AND DATA > SYSDATE
```

Przykład działania:

	KRAJ	DATA	NAZWA	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	Polska	2020-02-03 00:00:00	Piękny Kraków	4	1
2	Polska	2020-03-03 00:00:00	Wieliczka	4	1
3	Polska	2020-01-03 00:00:00	Piękny Rzeszów	5	2
4	Polska	2020-03-10 00:00:00	Bieszczady	7	2

Dodatkowo dla uproszczenia obsługi stworzyłem analogiczny widok zawierający dodatkowo ID_WYCIECZKI

```
CREATE or REPLACE view DOSTĘPNE_WYCIECZKI_2 as
select * from
(select w.ID_WYCIECZKI, w.KRAJ,
w.DATA, w.NAZWA, w.LICZBA_MIEJSC,
w.LICZBA_MIEJSC - count(*) as LICZBA_WOLNYCH_MIEJSC
from WYCIECZKI w
left join REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
where r.STATUS IN ('N', 'P', 'Z')
group by w.ID_WYCIECZKI, w.KRAJ, w.DATA, w.NAZWA, w.LICZBA_MIEJSC)
where LICZBA_WOLNYCH_MIEJSC > 0 AND DATA > SYSDATE
```

e) Widok rezerwacje_do_anulowania – widok pokazujący pola ID_WYCIECZKI, DATA, NR_REZERWACJI, STATUS, ID_OSOBY oraz NAZWISKO osoby dokonującej rezerwacji. Rezerwacje przygotowywane są do anulowania na tydzień przed wyjazdem, zatem sprawdzam status rezerwacji oraz aktualną datę w stosunku do daty wycieczki

```
CREATE or REPLACE view REZERWACJE_DO_ANULOWANIA as
select w.ID_WYCIECZKI, w.DATA, r.NR_REZERWACJI, r.STATUS, o.ID_OSOBY,
o.NAZWISKO
from REZERWACJE r
join OSOBY o on r.ID_OSOBY = o.ID_OSOBY
join WYCIECZKI w on w.ID_WYCIECZKI = r.ID_WYCIECZKI
where r.STATUS = 'N' AND w.DATA < (CURRENT_DATE + 7)
```

Przykład działania:

```
UPDATE WYCIECZKI SET DATA = '2019-10-23' WHERE ID_WYCIECZKI = 5;
```

	ID_WYCIECZKI	DATA	NR_REZERWACJI	STATUS	ID_OSOBY	NAZWISKO
1	5	2019-10-23 00:00:00	35	N	4	Milik
2	5	2019-10-23 00:00:00	38	N	7	Kowalski

Zadanie 4.

W zadaniu czwartym należało stworzyć procedury/funkcję pobierające dane. W każdym podpunkcie zdecydowałem się na stworzenie funkcji na wzór stworzonych na stronie <https://renenyffenegger.ch/notes/development/databases/Oracle/PL-SQL/collection-types/return-table-from-function/index>, które zwracają tabelę. Przed stworzeniem każdej funkcji na początku tworzę nowy typ odpowiadający rekordowi w zwracanej tabeli, a następnie typowi tabeli zbudowanej z tychże rekordów.

a) Funkcja uczestnicy_wycieczki (id_wycieczki) – funkcja przyjmująca jako argument id_wycieczki i zwracająca tablicę zawierającą ID_OSOBY, IMIĘ, NAZWISKO, PESEL, KONTAKT, STATUS

```
CREATE or REPLACE type rekord_uczestnicy_wycieczki as object (  
    ID_OSOBY integer,  
    IMIE      VARCHAR2(50),  
    NAZWISKO  VARCHAR2(50),  
    PESEL     VARCHAR2(11),  
    KONTAKT   VARCHAR2(100),  
    STATUS    CHAR  
);  
  
create or replace type tabela_uczestnicy_wycieczki  
as table of rekord_uczestnicy_wycieczki;  
  
create or REPLACE FUNCTION uczestnicy_wycieczki(ID_WYCIECZKI_X number)  
return tabela_uczestnicy_wycieczki as v_ret tabela_uczestnicy_wycieczki;  
czy_wycieczka_istnieje integer;  
BEGIN  
    SELECT COUNT(*) INTO czy_wycieczka_istnieje  
    FROM WYCIECZKI WHERE WYCIECZKI.ID_WYCIECZKI = ID_WYCIECZKI_X;  
    IF czy_wycieczka_istnieje = 0 THEN  
        raise_application_error(-20111, 'Brak wycieczki o podanym id');  
    END IF;  
    SELECT rekord_uczestnicy_wycieczki(o.ID_OSOBY,  
        o.IMIE, o.NAZWISKO, o.PESEL,  
        o.KONTAKT, r.STATUS)  
    BULK COLLECT INTO v_ret  
    FROM WYCIECZKI w  
        JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI  
        JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY  
    WHERE w.ID_WYCIECZKI = ID_WYCIECZKI_X AND r.STATUS in ('N', 'P', 'Z');  
    return v_ret;  
end uczestnicy_wycieczki;
```

Przykład działania:

```
select * from table(uczestnicy_wycieczki(5));
```

	ID_OSOBY ÷	IMIĘ ÷	NAZWISKO ÷	PESEL ÷	KONTAKT ÷	STATUS ÷
1	4	Arkadiusz	Milik	12909878	tel: 8812	N
2	5	Artur	Boruc	90989098	tel: 5732	P
3	6	Dawid	Ziobro	57325678	tel: 9900	Z
4	7	Kamil	Kowalski	11654321	tel: 1123	N
5	10	Artur	Bielik	90900008	tel: 9433	P

b) Funkcja rezerwacje_osoby (id_osoby) – funkcja przyjmująca jako argument id_osoby i zwracająca tablicę zawierającą pola ID_WYCIECZKI, NAZWA, KRAJ, DATA_WYCIECZKI, ID_REZERWACJI oraz STATUS. Funkcja sprawdza czy osoba o danym ID znajduje się w bazie danych.

```
create or replace type rekord_rezerwacje_osoby as object (  
    ID_WYCIECZKI number,  
    NAZWA          VARCHAR2(100),  
    KRAJ           VARCHAR2(50),  
    DATA_WYCIECZKI DATE,  
    ID_REZERWACJI number,  
    STATUS char  
);  
create or replace type tabela_rezerwacje_osoby  
as table of rekord_rezerwacje_osoby;
```

CREATE OR REPLACE

```
FUNCTION rezerwacje_osoby(id_osoby_X number)  
return tabela_rezerwacje_osoby as v_ret tabela_rezerwacje_osoby;  
czy_id_poprawne integer;  
BEGIN  
    select count(*) into czy_id_poprawne from OSOBY  
    where id_osoby_X = id_osoby;  
  
    if czy_id_poprawne = 0 then  
        raise_application_error(-20111,'Osoba o danym id nie figuruje w bazie');  
    end if;  
    SELECT REKORD_REZERWACJE_OSOBY(w.ID_WYCIECZKI, w.NAZWA,w.KRAJ,  
                                   w.DATA,r.NR_REZERWACJI, r.STATUS)  
    BULK COLLECT INTO v_ret  
    FROM WYCIECZKI w  
        JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI  
        JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY  
    WHERE o.ID_OSOBY = id_osoby_X;  
    return v_ret;  
end rezerwacje_osoby;
```

Przykład działania:

```
select * from table(rezerwacje_osoby(6));
```

	ID_WYCIECZKI ÷	NAZWA ÷	KRAJ ÷	DATA_WYCIECZKI ÷	ID_REZERWACJI ÷	STATUS ÷
1	2	Piękny Kraków	Polska	2020-02-03 00:00:00	28	Z
2	4	Piękny Rzeszów	Polska	2020-01-03 00:00:00	34	Z
3	5	Bieszczady	Polska	2019-10-23 00:00:00	37	Z

c) Funkcja przyszłe_rezerwacje_osoby (id_osoby) – funkcja przyjmująca jako argument id_osoby i zwracająca tablicę zawierającą pola ID_WYCIECZKI, NAZWA, KRAJ, DATA_WYCIECZKI, ID_REZERWACJI oraz STATUS. Zwracana tabela jest takiego samego typu jak w funkcji w podpunkcie b).

CREATE OR REPLACE

```
FUNCTION PRZYSZŁE_REZERWACJE_OSOBY(id_osoby_X number)
return tabela_rezerwacje_osoby as v_ret tabela_rezerwacje_osoby;
czy_id_poprawne integer;
BEGIN
select count(*) into czy_id_poprawne from OSOBY
where id_osoby_X = id_osoby;
if czy_id_poprawne = 0 then
raise_application_error(-20111,'Osoba o danym id nie figuruje w bazie');
end if;
SELECT REKORD_REZERWACJE_OSOBY(w.ID_WYCIECZKI, w.NAZWA,w.KRAJ,
w.DATA,r.NR_REZERWACJI, r.STATUS)
BULK COLLECT INTO v_ret
FROM WYCIECZKI w
JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
WHERE o.ID_OSOBY = id_osoby_X AND w.DATA > SYSDATE;
return v_ret;
end PRZYSZŁE_REZERWACJE_OSOBY;
```

d) Funkcja dostępne_wycieczki (kraj, data_od, data_do) – funkcja przyjmująca jako argument nazwę kraju, początek okresu oraz koniec okresu wyszukiwania i zwracająca tablicę zawierającą pola ID_WYCIECZKI, NAZWA, KRAJ, DATA_WYCIECZKI, ILOŚĆ_MIEJSC oraz ILOŚĆ_WOLNYCH_MIEJSC.

```
create or replace type rekord_dostępne_wycieczki as object (
ID_WYCIECZKI number,
NAZWA VARCHAR2(100),
KRAJ VARCHAR2(50),
DATA_WYCIECZKI DATE,
ILOŚĆ_MIEJSC number,
ILOŚĆ_WOLNYCH_MIEJSC number
);
create or replace type tabela_dostępne_wycieczki as table of
rekord_dostępne_wycieczki;

CREATE OR REPLACE FUNCTION
dostępne_wycieczki(kraj_X WYCIECZKI.KRAJ%TYPE, data_początek DATE, date_koniec
DATE)
return tabela_dostępne_wycieczki as v_ret tabela_dostępne_wycieczki;
BEGIN
select rekord_dostępne_wycieczki(w.ID_WYCIECZKI,w.NAZWA,
w.KRAJ, w.DATA, w.LICZBA_MIEJSC, dw.LICZBA_WOLNYCH_MIEJSC)
BULK COLLECT INTO v_ret
from DOSTĘPNE_WYCIECZKI dw
inner join WYCIECZKI w on w.DATA = dw.DATA AND w.NAZWA = dw.NAZWA
where dw.KRAJ like kraj_X AND data_początek <= dw.DATA AND date_koniec >=
dw.DATA;
return v_ret;
end dostępne_wycieczki;
```


Przykład działania:

```
select * from (dostepne_wycieczki('Polska','2020-02-02','2020-03-03'));
```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA_WYCIECZKI	ILOSC_MIEJSC	ILOSC_WOLNYCH_MIEJSC
1	2	Piękny Kraków	Polska	2020-02-03 00:00:00	4	1
2	3	Wieliczka	Polska	2020-03-03 00:00:00	4	1

Zadanie 5.

W zadaniu piątym należało stworzyć procedury/funkcję modyfikujące dane. W każdym podpunkcie zdecydowałem się na stworzenie procedur, w przypadku niemożności dokonania modyfikacji wywołuje procedurę `raise_application_error` z opisem błędu.

a) Procedura `dodaj_rezerwacje(id_wycieczki, id_osoby)` – procedura przyjmująca jako argumenty `id_wycieczki` oraz `id_osoby`. Po pomyślnym sprawdzeniu czy wycieczka jest dostępna (czy są miejsca oraz czy już się nie odbyła) oraz czy nie istnieje już dokładnie taka rezerwacja (założenie, że 1 osoba nie może zrobić dwóch rezerwacji na tą samą wycieczkę)

CREATE OR REPLACE PROCEDURE

```
dodaj_rezerwacje(ID_WYCIECZKI_X number, ID_OSOBY_X number) AS
czy_wycieczka_jest_dostepna integer;
czy_nie_duplikujemy_rezerwacji integer;
BEGIN
    SELECT COUNT(*) INTO czy_wycieczka_jest_dostepna
    FROM DOSTEPNE_WYCIECZKI_2 --w tym widoku znajdują się tylko przyszłe
wycieczki z wolnymi miejscami
    WHERE ID_WYCIECZKI_X = ID_WYCIECZKI;
    IF czy_wycieczka_jest_dostepna = 0
    THEN raise_application_error(-20111, 'Wycieczka o danym ID jest
niedostępna');
    END IF;
    SELECT COUNT(*) INTO czy_nie_duplikujemy_rezerwacji
    FROM REZERWACJE
    WHERE ID_WYCIECZKI = ID_WYCIECZKI_X AND ID_OSOBY_X = ID_OSOBY;
    IF czy_nie_duplikujemy_rezerwacji > 0
    THEN raise_application_error(-20111, 'Istnieje już rezerwacja o
danych parametrach');
    END IF;
    INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
    VALUES (ID_WYCIECZKI_X, ID_OSOBY_X, 'N');
END dodaj_rezerwacje;
```

Przykład działania:

```
BD_KBIENIASZ> begin
    DODAJ_REZERWACJE(5,3);
end;
[2019-10-20 10:02:18] completed in 55 ms
BD_KBIENIASZ> begin
    DODAJ_REZERWACJE(5,4);
end;
[2019-10-20 10:02:27] [72000][20111] ORA-20111: Wycieczka o danym ID jest niedostępna
```

b) Procedura `zmien_rezerwacje (id_wycieczki, status)` – procedura sprawdza czy można zmienić status. Problematyczna sytuacja, którą trzeba sprawdzić to przypadek, gdy chcemy zmienić status anulowanej wycieczki na przykładowo potwierdzony, ale nie ma już wolnych miejsc. Przyjmuje, że można dowolnie zmieniać stan ze stanów {N, P, Z}

create or replace PROCEDURE

```
zmien_status_rezerwacji(ID_REZERWACJI_X number, STATUS_X char) AS
czy_rezerwacja_istnieje integer;
id_powiązanej_wycieczki integer;
aktualny_status char;
czy_sa_jeszcze_miejsca integer; --przypadek A -> {P, Z, N}
BEGIN
  Select count(*) into czy_rezerwacja_istnieje
    from REZERWACJE r where ID_REZERWACJI_X = r.NR_REZERWACJI;
  if czy_rezerwacja_istnieje = 0
  then
    raise_application_error(-20111, 'Rezerwacja o danym ID nie istnieje');
  end if;
  Select count(*) into id_powiązanej_wycieczki from REZERWACJE r
    where ID_REZERWACJI_X = r.NR_REZERWACJI;
  select count(*) into czy_sa_jeszcze_miejsca from DOSTĘPNE_WYCIECZKI_2
    where ID_WYCIECZKI = id_powiązanej_wycieczki;
  select r.status into aktualny_status from REZERWACJE r where
    ID_REZERWACJI_X = r.NR_REZERWACJI;
  if (czy_sa_jeszcze_miejsca = 0 AND aktualny_status = 'A')
  then
    raise_application_error(-20111, 'Brak wolnych miejsc, zmiana statusu
      niemożliwa');
  end if;
  UPDATE REZERWACJE SET STATUS = STATUS_X
    WHERE NR_REZERWACJI =ID_REZERWACJI_X;
end zmien_status_rezerwacji;
```

Przykład działania:

```
BD_KBIENIASZ> begin ZMIEN_STATUS_REZERWACJI(39, 'A'); end;
[2019-10-20 10:21:29] completed in 51 ms
BD_KBIENIASZ> begin ZMIEN_STATUS_REZERWACJI(40, 'P'); end;
[2019-10-20 10:21:46] completed in 57 ms
BD_KBIENIASZ> begin ZMIEN_STATUS_REZERWACJI(39, 'P'); end;
[2019-10-20 10:22:00] [72000][20111] ORA-20111: Brak wolnych miejsc, zmiana statusu niemożliwa
```

c) Procedura `zmien_liczbe_miejsc (id_wycieczki, liczba_miejsc)` – procedura sprawdza czy można zmienić liczbę miejsc. Problematyczna sytuacja, którą trzeba sprawdzić to przypadek, gdy chcemy zmienić liczbę miejsc na poniżej liczby już zarezerwowanych miejsc.

create or replace PROCEDURE

```
zmien_liczbe_miejsc(ID_WYCIECZKI_X number, LICZBA_MIEJSC_X number) AS
czy_wycieczka_istnieje integer;
aktualna_liczba_rezerwacji integer;
BEGIN
  SELECT count(*) INTO czy_wycieczka_istnieje
    from WYCIECZKI where ID_WYCIECZKI = ID_WYCIECZKI_X;
  IF czy_wycieczka_istnieje = 0
  then
    raise_application_error(-20111, 'Wycieczka nie istnieje');
  end if;
```

```

select count(*) into aktualna_liczba_rezerwacji from REZERWACJE
  WHERE ID_WYCIECZKI_X = ID_WYCIECZKI AND STATUS in ('N','P','Z');
if aktualna_liczba_rezerwacji > LICZBA_MIEJSC_X
  then raise_application_error(-20111, 'Nie można zmniejszyć liczby
    miejsc z powodu istniejących rezerwacji');
end if;
UPDATE WYCIECZKI SET LICZBA_MIEJSC = LICZBA_MIEJSC_X
  WHERE ID_WYCIECZKI_X = ID_WYCIECZKI;
end zmien_liczbe_miejsc;

```

Przykład działania:

```

BD_KBIENIASZ> begin ZMIEN_LICZBE_MIEJSC(5, 8); end;
[2019-10-20 10:29:38] completed in 48 ms
BD_KBIENIASZ> begin ZMIEN_LICZBE_MIEJSC(5, 6); end;
[2019-10-20 10:29:47] [72000][20111] ORA-20111: Nie można zmniejszyć liczby miejsc z powodu istniejących rezerwacji
[2019-10-20 10:29:47] ORA-06512: przy "BD_KBIENIASZ.ZMIEN_LICZBE_MIEJSC", linia 14
[2019-10-20 10:29:47] ORA-06512: przy linia 1
[2019-10-20 10:29:47] Position: 0
BD_KBIENIASZ> begin ZMIEN_LICZBE_MIEJSC(5, 7); end;
[2019-10-20 10:29:56] completed in 53 ms

```

Zadanie 6

W zadaniu szóstym zadaniu należało dodać tabelę dziennikującą zmiany statusu rezerwacji oraz zmienić warstwę procedur modyfikujących dane, tak aby dopisywały informację do dziennika

```

create table rezerwacje_log
(
  id NUMBER generated as identity
    constraint id_PK
      primary key,
  id_rezerwacji NUMBER
    constraint rezerwacje_log_FK1
      references REZERWACJE,
  DATA DATE,
  STATUS CHAR
    constraint REZERWACJE_LOG_CHK1
      check (status IN ('N', 'P', 'Z', 'A'))
)

```

Dodałem procedurę dodaj_rezerwacje_log działającą w analogiczny sposób jak dodaj_rezerwacje.

CREATE OR REPLACE PROCEDURE

```

dodaj_rezerwacje_log(ID_WYCIECZKI_X number, ID_OSOBY_X number) AS
czy_wycieczka_jest_dostepna integer;
czy_nie_duplikujemy_rezerwacji integer;
stworzone_id_rezerwacji integer;
BEGIN
  SELECT COUNT(*) INTO czy_wycieczka_jest_dostepna
  FROM DOSTEPNE_WYCIECZKI_2 --w tym widoku znajdują się tylko przyszłe
                                wycieczki z wolnymi miejscami
  WHERE ID_WYCIECZKI_X = ID_WYCIECZKI;
  IF czy_wycieczka_jest_dostepna = 0
    THEN raise_application_error(-20111, 'Wycieczka o danym ID jest
      niedostępna');
  END IF;
  SELECT COUNT(*) INTO czy_nie_duplikujemy_rezerwacji

```

```

FROM REZERWACJE
WHERE ID_WYCIECZKI = ID_WYCIECZKI_X AND ID_OSOBY_X = ID_OSOBY;

IF czy_nie_duplikujemy_rezerwacji > 0
THEN raise_application_error(-20111, 'Istnieje już rezerwacja o
danych parametrach');
END IF;

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (ID_WYCIECZKI_X, ID_OSOBY_X, 'N');

SELECT r.NR_REZERWACJI INTO stworzone_id_rezerwacji from REZERWACJE r
WHERE ID_WYCIECZKI_X = r.ID_WYCIECZKI AND ID_OSOBY_X = r.ID_OSOBY;

INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS)
VALUES (stworzone_id_rezerwacji, CURRENT_DATE, 'N');
END dodaj_rezerwacje_log;

```

Dodałem procedurę zmien_status_rezerwacji_log działającą w analogiczny sposób jak zmien_status_rezerwacji.

```

create or REPLACE PROCEDURE
zmien_status_rezerwacji_log(ID_REZERWACJI_X number, STATUS_X char) AS
czy_rezerwacja_istnieje integer;
id_powiązanej_wycieczki integer;
aktualny_status char;
czy_sa_jeszcze_miejsca integer; --przypadek A -> {P, Z, N}
BEGIN
Select count(*) into czy_rezerwacja_istnieje
from REZERWACJE r where ID_REZERWACJI_X = r.NR_REZERWACJI;
if czy_rezerwacja_istnieje = 0
then
raise_application_error(-20111, 'Rezerwacja o danym ID nie istnieje');
end if;
Select count(*) into id_powiązanej_wycieczki from REZERWACJE r
where ID_REZERWACJI_X = r.NR_REZERWACJI;
select count(*) into czy_sa_jeszcze_miejsca from DOSTĘPNE_WYCIECZKI_2
where ID_WYCIECZKI = id_powiązanej_wycieczki;
select r.status into aktualny_status from REZERWACJE r
where ID_REZERWACJI_X = r.NR_REZERWACJI;
if (czy_sa_jeszcze_miejsca = 0 AND aktualny_status = 'A')
then
raise_application_error(-20111, 'Brak wolnych miejsc, zmiana statusu
niemożliwa');
end if;
UPDATE REZERWACJE SET STATUS = STATUS_X
WHERE NR_REZERWACJI = ID_REZERWACJI_X;
INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS)
VALUES (ID_REZERWACJI_X, CURRENT_DATE, STATUS_X);
end zmien_status_rezerwacji_log;

```

Przykład działania: Tabela REZERWACJE_LOG powstała po dodaniu dwóch rezerwacji oraz trzech modyfikacjach

	ID	ID_REZERWACJI	DATA	STATUS
1	1	61	2019-10-20 08:01:55	N
2	2	62	2019-10-20 08:02:18	N
3	3	39	2019-10-20 08:15:08	P
4	4	39	2019-10-20 08:21:29	A
5	5	40	2019-10-20 08:21:46	P

Zadanie 7

W zadaniu siódmym leżało zmienić strukturę bazy poprzez dodanie w tabeli wycieczki redundantnego pola liczba_wolnych_miejsc oraz korzystając z tego pola utworzyć nowe widoki oraz procedury.

Dodanie pola liczba_wolnych_miejsc:

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

Procedura przelicz wyliczająca wartość dla nowego pola:

```
CREATE or REPLACE PROCEDURE przelicz(ID number) AS
BEGIN
    UPDATE WYCIECZKI w
    SET LICZBA_WOLNYCH_MIEJSC =
        LICZBA_MIEJSC - (select count(*) FROM REZERWACJE r
                        WHERE r.ID_WYCIECZKI = w.ID_WYCIECZKI
                        AND r.STATUS in ('N', 'P', 'Z'))
    WHERE ID_WYCIECZKI = ID;
end przelicz;
DECLARE
    max_id integer;
begin
    select max(id_wycieczki) into max_id from Wycieczki;
    for i in 1..max_id loop
        przelicz(i);
    end loop;
end;
```

Zaktualizowana tabela wycieczki

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	1	Wycieczka do Paryża	Francja	2016-01-01 00:00:00	Ciekawa wycieczka ...	3	1
2	2	Piękny Kraków	Polska	2020-02-03 00:00:00	Najciekawa wycieczka ...	4	1
3	3	Wieliczka	Polska	2020-03-03 00:00:00	Zadziwiająca kopalnia ...	4	1
4	4	Piękny Rzeszów	Polska	2020-01-03 00:00:00	Najciekawa wycieczka ...	5	2
5	5	Bieszczady	Polska	2019-10-23 00:00:00	Piękne połoniny ...	7	2

Widoki, które można było zaktualizować:

- Widok wycieczki_miejsca – nowy widok wycieczki_miejsca_2a:

```
CREATE or REPLACE VIEW wycieczki_miejsca_2a AS SELECT * FROM wycieczki w;
```

- Widok dostępne_wycieczki – nowy widok dostępne_wycieczki_2a:

```
CREATE OR REPLACE VIEW DOSTĘPNE_WYCIECZKI_2a AS
SELECT * from WYCIECZKI
WHERE LICZBA_WOLNYCH_MIEJSC > 0 AND DATA > SYSDATE
```

Funkcje, które można był zaktualizować:

- funkcja dostępne_wycieczki:

```
CREATE OR REPLACE FUNCTION
dostępne_wycieczki_2(kraj_X WYCIECZKI.KRAJ%TYPE,
    data_początek DATE, date_koniec DATE)
return tabela_dostępne_wycieczki as v_ret tabela_dostępne_wycieczki;
BEGIN
    select rekord_dostępne_wycieczki(w.ID_WYCIECZKI,w.NAZWA,
```

```

        w.KRAJ, w.DATA,
        w.LICZBA_MIEJSC,
        w.LICZBA_WOLNYCH_MIEJSC)
    BULK COLLECT INTO v_ret
from WYCIECZKI w
where w.KRAJ like kraj_X AND data_początek <= w.DATA AND
    date_koniec >= w.DATA AND w.LICZBA_MIEJSC > 0;
return v_ret;
end dostepne_wycieczki_2;

```

Procedury zmieniające, które można był zaktualizować:

- funkcja dodaj_rezerwacje_log;

```

create or replace PROCEDURE
dodaj_rezerwacje_log_2(ID_WYCIECZKI_X number, ID_OS0BY_X number) AS
czy_wycieczka_jest_dostepna integer;
czy_nie_duplikujemy_rezerwacji integer;
stworzone_id_rezerwacji integer;
liczba_wolnych_miejsc integer;
BEGIN
    SELECT count(*) INTO czy_wycieczka_jest_dostepna
    FROM WYCIECZKI --w tym widoku znajdują się tylko przyszłe wycieczki z
                                wolnymi miejscami
    WHERE ID_WYCIECZKI_X = ID_WYCIECZKI
        AND LICZBA_WOLNYCH_MIEJSC > liczba_wolnych_miejsc
        AND DATA > SYSDATE;
    IF czy_wycieczka_jest_dostepna = 0
        THEN raise_application_error(-20111, 'Wycieczka o danym ID jest
                                niedostępna');
    END IF;

    SELECT COUNT(*) INTO czy_nie_duplikujemy_rezerwacji
    FROM REZERWACJE
    WHERE ID_WYCIECZKI = ID_WYCIECZKI_X AND ID_OS0BY_X = ID_OS0BY;

    IF czy_nie_duplikujemy_rezerwacji > 0
        THEN raise_application_error(-20111, 'Istnieje już rezerwacja o
                                danych parametrach');
    END IF;

    INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
    VALUES (ID_WYCIECZKI_X, ID_OS0BY_X, 'N');

    SELECT r.NR_REZERWACJI INTO stworzone_id_rezerwacji from REZERWACJE r
    WHERE ID_WYCIECZKI_X = r.ID_WYCIECZKI AND ID_OS0BY_X = r.ID_OS0BY;

    INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS)
    VALUES (stworzone_id_rezerwacji, CURRENT_DATE, 'N');

    SELECT w.LICZBA_WOLNYCH_MIEJSC INTO liczba_wolnych_miejsc
    from WYCIECZKI w WHERE w.ID_WYCIECZKI = ID_WYCIECZKI_X;

    UPDATE WYCIECZKI w
    SET w.LICZBA_WOLNYCH_MIEJSC = (liczba_wolnych_miejsc -1)
    where ID_WYCIECZKI = ID_WYCIECZKI_X;
END dodaj_rezerwacje_log_2;

```

- procedura zmien_status_rezerwacji_log:

```

create or replace PROCEDURE
ZMIEN_STATUS_REZERWACJI_LOG_2(ID_REZERWACJI_X number, STATUS_X char) AS
czy_rezerwacja_istnieje integer;
id_powiązanej_wycieczki integer;
aktualny_status char;
czy_sa_jeszcze_miejsca integer; --przypadek A -> {P, Z, N}
BEGIN
    Select count(*) into czy_rezerwacja_istnieje
    from REZERWACJE r where ID_REZERWACJI_X = r.NR_REZERWACJI;
    if czy_rezerwacja_istnieje = 0
    then
        raise_application_error(-20111, 'Rezerwacja o danym ID nie istnieje');
    end if;

    Select count(*) into id_powiązanej_wycieczki from REZERWACJE r
    where ID_REZERWACJI_X = r.NR_REZERWACJI;

    select count(*) into czy_sa_jeszcze_miejsca from DOSTĘPNE_WYCIECZKI_2
    where ID_WYCIECZKI = id_powiązanej_wycieczki;

    select r.status into aktualny_status from REZERWACJE r
    where ID_REZERWACJI_X = r.NR_REZERWACJI;
    if (czy_sa_jeszcze_miejsca = 0 AND aktualny_status = 'A')
    then
        raise_application_error(-20111, 'Brak wolnych miejsc, zmiana statusu
                                                niemożliwa');
    end if;
    UPDATE REZERWACJE SET STATUS = STATUS_X
    WHERE NR_REZERWACJI =ID_REZERWACJI_X;

    INSERT INTO REZERWACJE_LOG(ID_REZERWACJI, DATA, STATUS)
    VALUES (ID_REZERWACJI_X, CURRENT_DATE, STATUS_X);
    if aktualny_status = 'A' AND STATUS_X <> 'A'
    then
        UPDATE WYCIECZKI w
        SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC - 1
        where ID_WYCIECZKI = (SELECT r.ID_WYCIECZKI from REZERWACJE r
                                where ID_REZERWACJI_X = r.NR_REZERWACJI);
    end if;
end zmien_status_rezerwacji_log_2;

```

- procedura zmien_liczbe_miejsc:

```

create or replace PROCEDURE
zmien_liczbe_miejsc_2(ID_WYCIECZKI_X number, LICZBA_MIEJSC_X number) AS
czy_wycieczka_istnieje integer;
aktualna_liczba_rezerwacji integer;
BEGIN
    SELECT count(*) INTO czy_wycieczka_istnieje from WYCIECZKI
    where ID_WYCIECZKI = ID_WYCIECZKI_X;
    IF czy_wycieczka_istnieje = 0

```



```

        then
            raise_application_error(-20111, 'Wycieczka nie istnieje');
        end if;
        select w.LICZBA_MIEJSC - w.LICZBA_WOLNYCH_MIEJSC
            into aktualna_liczba_rezerwacji from WYCIECZKI w
            where w.ID_WYCIECZKI = ID_WYCIECZKI_X;
        if aktualna_liczba_rezerwacji >= LICZBA_MIEJSC_X
            then raise_application_error(-20111, 'Nie można zmniejszyć liczby
                miejsc z powodu istniejących rezerwacji');
            end if;
        UPDATE WYCIECZKI
        SET LICZBA_MIEJSC = LICZBA_MIEJSC_X , LICZBA_WOLNYCH_MIEJSC =
            LICZBA_WOLNYCH_MIEJSC + LICZBA_MIEJSC_X - LICZBA_MIEJSC
        WHERE ID_WYCIECZKI_X = ID_WYCIECZKI;
    end zmien_liczbe_miejsc_2;

```

Zadanie 8.

Należało wprowadzić triggery, które będą odpowiadały za zapisywanie zmian do dziennika rezerwacji.

a) trigger obsługujący dodanie rezerwacji

```

CREATE OR REPLACE TRIGGER trigger_dodanie_rezerwacji
AFTER INSERT ON REZERWACJE
FOR EACH ROW
BEGIN
    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);
END trigger_dodanie_rezerwacji;

```

b) trigger obsługujący zmianę statusu rezerwacji

```

create or replace trigger trigger_zmiana_statusu
after update on REZERWACJE
begin
    insert into REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    values (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);
end trigger_zmiana_statusu;

```

c) trigger zabraniający usunięcia rezerwacji

```

create or replace trigger trigger_zakaz_usuwania_rezerwacji
before delete on REZERWACJE
for each row declare roznica integer;
begin
    raise_application_error(-20111, 'Nie wolno usuwać rezerwacji');
end trigger_zakaz_usuwania_rezerwacji;

```

Warto zauważyć, iż wystarczy zmodyfikować procedury, które utworzyłem w zadaniu 7, poprzez usunięcie fragmentu kodu dodającego wpis do tabeli rezerwacje_log oraz ewentualnie kodu uaktualniającego tabelę wycieczki, gdyż za te zmiany są odpowiedzialne triggery.

a)procedura dodaj_rezerwacje

create or replace PROCEDURE

```
dodaj_rezerwacje_3(ID_WYCIECZKI_X number, ID_OSOBY_X number) AS
czy_wycieczka_jest_dostepna integer;
czy_nie_duplikujemy_rezerwacji integer;
stworzone_id_rezerwacji integer;
liczba_wolnych_miejsc integer;
BEGIN
    SELECT count(*) INTO czy_wycieczka_jest_dostepna
    FROM WYCIECZKI --w tym widoku znajdują się tylko przyszłe wycieczki
                                     z wolnymi miejscami
    WHERE ID_WYCIECZKI_X = ID_WYCIECZKI
    AND LICZBA_WOLNYCH_MIEJSC > liczba_wolnych_miejsc
    AND DATA > SYSDATE;
    IF czy_wycieczka_jest_dostepna = 0
    THEN raise_application_error(-20111, 'Wycieczka o danym ID jest
                                         niedostępna');

    END IF;
    SELECT COUNT(*) INTO czy_nie_duplikujemy_rezerwacji
    FROM REZERWACJE
    WHERE ID_WYCIECZKI = ID_WYCIECZKI_X AND ID_OSOBY_X = ID_OSOBY;

    IF czy_nie_duplikujemy_rezerwacji > 0
    THEN raise_application_error(-20111, 'Istnieje już rezerwacja o
                                         danych parametrach');

    END IF;
    INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
    VALUES (ID_WYCIECZKI_X, ID_OSOBY_X, 'N');
    SELECT r.NR_REZERWACJI INTO stworzone_id_rezerwacji from REZERWACJE r
    WHERE ID_WYCIECZKI_X = r.ID_WYCIECZKI AND ID_OSOBY_X = r.ID_OSOBY;
    SELECT w.LICZBA_WOLNYCH_MIEJSC INTO liczba_wolnych_miejsc
    from WYCIECZKI w WHERE w.ID_WYCIECZKI = ID_WYCIECZKI_X;
    UPDATE WYCIECZKI w
    SET w.LICZBA_WOLNYCH_MIEJSC = (liczba_wolnych_miejsc -1)
    where ID_WYCIECZKI = ID_WYCIECZKI_X;
END dodaj_rezerwacje_3;
```

b)procedura zmien_status_rezerwacji

create or replace PROCEDURE

```
ZMIEN_STATUS_REZERWACJI_3(ID_REZERWACJI_X number, STATUS_X char) AS
czy_rezerwacja_istnieje integer;
id_powiązanej_wycieczki integer;
aktualny_status char;
czy_sa_jeszcze_miejsca integer; --przypadek A -> {P, Z, N}
BEGIN
    Select count(*) into czy_rezerwacja_istnieje
    from REZERWACJE r where ID_REZERWACJI_X = r.NR_REZERWACJI;
    if czy_rezerwacja_istnieje = 0 then
        raise_application_error(-20111, 'Rezerwacja o danym ID nie istnieje');
    end if;
    Select count(*) into id_powiązanej_wycieczki from REZERWACJE r
    where ID_REZERWACJI_X = r.NR_REZERWACJI;
    select count(*) into czy_sa_jeszcze_miejsca from DOSTĘPNE_WYCIECZKI_2
    where ID_WYCIECZKI = id_powiązanej_wycieczki;
```

```

select r.status into aktualny_status from REZERWACJE r
where ID_REZERWACJI_X = r.NR_REZERWACJI;

if (czy_sa_jeszcze_miejsca = 0 AND aktualny_status = 'A') then
    raise_application_error(-20111, 'Brak wolnych miejsc, zmiana statusu
                                     niemożliwa');
end if;
UPDATE REZERWACJE SET STATUS = STATUS_X
WHERE NR_REZERWACJI = ID_REZERWACJI_X;
if aktualny_status = 'A' AND STATUS_X <> 'A'
then
    UPDATE WYCIECZKI w
    SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC - 1
    where ID_WYCIECZKI = (SELECT r.ID_WYCIECZKI from REZERWACJE r
                           where ID_REZERWACJI_X = r.NR_REZERWACJI);
end if;
end zmien_status_rezerwacji_3;

```

c) procedura zmien_liczbe_miejsc

```

create or replace PROCEDURE
zmien_liczbe_miejsc_3(ID_WYCIECZKI_X number, LICZBA_MIEJSC_X number) AS
czy_wycieczka_istnieje integer;
aktualna_liczba_rezerwacji integer;
BEGIN
    SELECT count(*) INTO czy_wycieczka_istnieje from WYCIECZKI
    where ID_WYCIECZKI = ID_WYCIECZKI_X;
    IF czy_wycieczka_istnieje = 0
    then
        raise_application_error(-20111, 'Wycieczka nie istnieje');
    end if;
    select w.LICZBA_MIEJSC - w.LICZBA_WOLNYCH_MIEJSC
    into aktualna_liczba_rezerwacji from WYCIECZKI w
    where w.ID_WYCIECZKI = ID_WYCIECZKI_X;
    if aktualna_liczba_rezerwacji >= LICZBA_MIEJSC_X
    then raise_application_error(-20111, 'Nie można zmniejszyć liczby
    miejsc z powodu istniejących rezerwacji');
    end if;
    SET LICZBA_MIEJSC = LICZBA_MIEJSC_X , LICZBA_WOLNYCH_MIEJSC =
    LICZBA_WOLNYCH_MIEJSC + LICZBA_MIEJSC_X - LICZBA_MIEJSC
    WHERE ID_WYCIECZKI_X = ID_WYCIECZKI;
end zmien_liczbe_miejsc_3;

```

Zadanie 9.

Należało wprowadzić trigger, które będą odpowiadały za zapisywanie zmian do dziennika rezerwacji. W tym celu zaktualizowałem utworzone w poprzednim ćwiczeniu.

a) trigger obsługujący dodanie rezerwacji

```

CREATE OR REPLACE TRIGGER trigger_dodanie_rezerwacji
AFTER INSERT ON REZERWACJE
FOR EACH ROW
BEGIN
    UPDATE WYCIECZKI

```

```

SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1
WHERE :NEW.ID_WYCIECZKI = ID_WYCIECZKI;
INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);
END trigger_dodanie_rezerwacji;

```

b) trigger obsługujący zmianę statusu rezerwacji

```

create or replace trigger trigger_zmiana_statusu
after update on REZERWACJE
for each row declare roznica integer;
begin
insert into REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
values (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);
if :NEW.STATUS = 'A' AND :OLD.STATUS <> 'A' then
    roznica:=-1;
else
    if :NEW.STATUS = 'A' AND :OLD.STATUS <> 'A' then
        roznica:=1;
    else
        roznica:=0;
    end if;
end if;
update WYCIECZKI
set LICZBA_WOLNYCH_MIEJSC = LICZBA_MIEJSC + roznica
where ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
end trigger_zmiana_statusu;

```

c) trigger obsługujący zmianę miejsc wycieczki

```

CREATE OR REPLACE TRIGGER trigger_zmiana_miejsc_wycieczki

BEFORE UPDATE OF liczba_miejsc
ON wycieczki
FOR EACH ROW
BEGIN
    SELECT :OLD.LICZBA_WOLNYCH_MIEJSC +
        (:NEW.LICZBA_MIEJSC - :OLD.LICZBA_MIEJSC) INTO
:NEW.LICZBA_WOLNYCH_MIEJSC
FROM Dual;
END;

```

Aktualizacje funkcji – należy usunąć fragmenty odpowiedzialne za modyfikacje pola LICZBA_WOLNYCH_MIEJSC.

a) procedura dodaj_rezerwacje

```

create or replace PROCEDURE
dodaj_rezerwacje_4(ID_WYCIECZKI_X number, ID_OS0BY_X number) AS
czy_wycieczka_jest_dostepna integer;
czy_nie_duplikujemy_rezerwacji integer;
BEGIN
    SELECT count(*) INTO czy_wycieczka_jest_dostepna
    FROM WYCIECZKI --w tym widoku znajdują się tylko przyszłe wycieczki z
                    wolnymi miejscami

```

```

WHERE ID_WYCIECZKI_X = ID_WYCIECZKI AND LICZBA_WOLNYCH_MIEJSC > 0
AND DATA > SYSDATE;
IF czy_wycieczka_jest_dostepna = 0
THEN raise_application_error(-20111, 'Wycieczka o danym ID jest
niedostępna');

END IF;
SELECT COUNT(*) INTO czy_nie_duplikujemy_rezerwacji
FROM REZERWACJE
WHERE ID_WYCIECZKI = ID_WYCIECZKI_X AND ID_OSOBY_X = ID_OSOBY;
IF czy_nie_duplikujemy_rezerwacji > 0
THEN raise_application_error(-20111, 'Istnieje już rezerwacja o
danych parametrach');

END IF;
INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (ID_WYCIECZKI_X, ID_OSOBY_X, 'N');
END dodaj_rezerwacje_4;

```

b) procedura zmien_status_rezerwacji

```

create or replace PROCEDURE
ZMIEN_STATUS_REZERWACJI_4(ID_REZERWACJI_X number, STATUS_X char) AS
czy_rezerwacja_istnieje integer;
id_powiązanej_wycieczki integer;
aktualny_status char;
czy_sa_jeszcze_miejsca integer; --przypadek A -> {P, Z, N}
BEGIN
Select count(*) into czy_rezerwacja_istnieje
from REZERWACJE r where ID_REZERWACJI_X = r.NR_REZERWACJI;
if czy_rezerwacja_istnieje = 0
then
raise_application_error(-20111, 'Rezerwacja o danym ID nie istnieje');
end if;
Select count(*) into id_powiązanej_wycieczki from REZERWACJE r
where ID_REZERWACJI_X = r.NR_REZERWACJI;
select count(*) into czy_sa_jeszcze_miejsca
from DOSTĘPNE WYCIECZKI_2
where ID_WYCIECZKI = id_powiązanej_wycieczki;
select r.status into aktualny_status from REZERWACJE r
where ID_REZERWACJI_X = r.NR_REZERWACJI;
if (czy_sa_jeszcze_miejsca = 0 AND aktualny_status = 'A')
then
raise_application_error(-20111, 'Brak wolnych miejsc, zmiana statusu
niemożliwa');

end if;
UPDATE REZERWACJE SET STATUS = STATUS_X
WHERE NR_REZERWACJI = ID_REZERWACJI_X;
end zmien_status_rezerwacji_4;

```

c) procedura zmien_liczbe_miejsc

```

create or replace PROCEDURE
zmien_liczbe_miejsc_4(ID_WYCIECZKI_X number, LICZBA_MIEJSC_X number) AS
czy_wycieczka_istnieje integer;
aktualna_liczba_rezerwacji integer;
BEGIN
SELECT count(*) INTO czy_wycieczka_istnieje from WYCIECZKI

```

```

    where ID_WYCIECZKI = ID_WYCIECZKI_X;
IF czy_wycieczka_istnieje = 0
    then
        raise_application_error(-20111, 'Wycieczka nie istnieje');
end if;
select w.LICZBA_MIEJSC - w.LICZBA_WOLNYCH_MIEJSC
    into aktualna_liczba_rezerwacji from WYCIECZKI w
    where w.ID_WYCIECZKI = ID_WYCIECZKI_X;
if aktualna_liczba_rezerwacji >= LICZBA_MIEJSC_X
    then raise_application_error(-20111, 'Nie można zmniejszyć liczby
        miejsc z powodu istniejących rezerwacji');
end if;
UPDATE WYCIECZKI
SET LICZBA_MIEJSC = LICZBA_MIEJSC_X
WHERE ID_WYCIECZKI_X = ID_WYCIECZKI;
end zmien_liczbe_miejsc_4;

```

Test ostatecznych wersji triggerów oraz procedurów:

Przeanalizuje zmiany dokonywane na wycieczce o ID_WYCIECZKI = 3 oraz NAZWIE Wieliczka. Przed zmianami LICZBA_MIEJSC = 4, LICZBA_WOLNYCH_MIEJSC = 1.

ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	1 Wycieczka do Paryza	Francja	2016-01-01 00:00:00	Ciekawa wycieczka ...	3	1
2	2 Piękny Kraków	Polska	2020-02-03 00:00:00	Najciekawa wycieczka ...	4	1
3	3 Wieliczka	Polska	2020-03-03 00:00:00	Zadziwiająca kopalnia ...	4	1
4	4 Piękny Rzeszów	Polska	2020-01-03 00:00:00	Najciekawa wycieczka ...	5	2
5	5 Bieszczady	Polska	2019-10-23 00:00:00	Piękne połoniny ...	6	0

Wywołuje funkcję zmien_liczbe_miejsc_4 (3, 6):

```

begin
    zmien_liczbe_miejsc_4(3, 6);
end;

```

Po ponownym wywołanie informacji dotyczących wycieczki do Wieliczki widzimy, że pole LICZBA_WOLNYCH_MIEJSC również zmieniło swoją wartość.

ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	1 Wycieczka do Paryza	Francja	2016-01-01 00:00:00	Ciekawa wycieczka ...	3	1
2	2 Piękny Kraków	Polska	2020-02-03 00:00:00	Najciekawa wycieczka ...	4	1
3	3 Wieliczka	Polska	2020-03-03 00:00:00	Zadziwiająca kopalnia ...	6	3
4	4 Piękny Rzeszów	Polska	2020-01-03 00:00:00	Najciekawa wycieczka ...	5	2
5	5 Bieszczady	Polska	2019-10-23 00:00:00	Piękne połoniny ...	6	0

Następnie zmieniłem status jednej rezerwacji na wycieczkę do Wieliczki.

	NR_REZERWACJI	ID_WYCIECZKI	ID_OSOBY	STATUS
1	29	3	7	N
2	30	3	8	P
3	31	3	9	Z

```

begin
    ZMIEN_STATUS_REZERWACJI_4(30, 'A');
end;

```

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	3	Wieliczka	Polska	2020-03-03 00:00:00	Zadziwiająca kopalnia ...	6	5

Następnie dodałem jedną rezerwację na wycieczkę do Wieliczki.

begin

`DODAJ_REZERWACJE_4(3,1);`

end;

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	3	Wieliczka	Polska	2020-03-03 00:00:00	Zadziwiająca kopalnia ...	6	4

Powyższe operacje zostały poprawnie zapisane w tabeli REZERWACJE_LOG.

	ID	ID_REZERWACJI	DATA	STATUS
1	21	30	2019-10-23 20:28:16	A
2	22	81	2019-10-23 20:32:40	N