

# Laboratorium 1: Oracle PL/SQL

Mateusz Surjak

# 1 Tabele

## 1.1 osoby

```
1 create table OSOBY
2 (
3     ID_OSOBY NUMBER generated as identity
4         constraint OSOBY_PK
5             primary key,
6     IMIE      VARCHAR2(50),
7     NAZWISKO  VARCHAR2(50),
8     PESEL     VARCHAR2(11),
9     KONTAKT   VARCHAR2(100)
10 )
11
```

## 1.2 rezerwacje

```
1 create table REZERWACJE
2 (
3     NR_REZERWACJI NUMBER generated as identity
4         constraint REZERWACJE_PK
5             primary key,
6     ID_WYCIECZKI  NUMBER
7         constraint REZERWACJE_FK2
8             references WYCIECZKI,
9     ID_OSOBY      NUMBER
10         constraint REZERWACJE_FK1
11             references OSOBY,
12     STATUS        CHAR
13         constraint REZERWACJE_CHK1
14             check (status IN ('N', 'P', 'Z', 'A'))
15 )
16
```

## 1.3 wycieczki

```
1 create table WYCIECZKI
2 (
3     ID_WYCIECZKI  NUMBER generated as identity
4         constraint WYCIECZKI_PK
5             primary key,
6     NAZWA          VARCHAR2(100),
7     KRAJ           VARCHAR2(50),
8     DATA          DATE,
9     OPIS           VARCHAR2(200),
10    LICZBA_MIEJSC   NUMBER
11 )
12
```

## 1.4 rezerwacje\_log

```
1 create table REZERWACJE_LOG
2 (
3     ID                NUMBER generated as identity
4     constraint REZERWACJE_LOG_PK
5         primary key ,
6     ID_REZERWACJI NUMBER,
7     DATA             DATE,
8     STATUS            CHAR
9 )
10
```

## 2 Przykładowe dane

```
1  INSERT INTO osoby (imie , nazwisko , pesel , kontakt)
2  VALUES( 'Andrzej' , 'Podobinski' , '99326733738' , '443567588' );
3
4  INSERT INTO osoby (imie , nazwisko , pesel , kontakt)
5  VALUES( 'Ola' , 'Mazur' , '29326733738' , '443564588' );
6
7  INSERT INTO osoby (imie , nazwisko , pesel , kontakt)
8  VALUES( 'Grzegorz' , 'Poreba' , '99326233738' , '443561588' );
9
10 INSERT INTO osoby (imie , nazwisko , pesel , kontakt)
11 VALUES( 'Grzegorz' , 'Pach' , '99326533738' , '443567188' );
12
13 INSERT INTO osoby (imie , nazwisko , pesel , kontakt)
14 VALUES( 'Mateusz' , 'Surjak' , '99326733738' , '443567788' );
15
16 INSERT INTO osoby (imie , nazwisko , pesel , kontakt)
17 VALUES( 'Zuzanna' , 'Brzezinska' , '49326733738' , '443567598' );
18
19 INSERT INTO osoby (imie , nazwisko , pesel , kontakt)
20 VALUES( 'Sebastian' , 'Kusnierz' , '95326733738' , '443567138' );
21
22 INSERT INTO osoby (imie , nazwisko , pesel , kontakt)
23 VALUES( 'Jakub' , 'Perzylo' , '99526733738' , '443237588' );
24
25 INSERT INTO osoby (imie , nazwisko , pesel , kontakt)
26 VALUES( 'Karol' , 'Musur' , '99726733738' , '443457588' );
27
28 INSERT INTO osoby (imie , nazwisko , pesel , kontakt)
29 VALUES( 'Radek' , 'Kopec' , '99386733738' , '443517588' );
30
31 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
32 VALUES (23,45,'N');
33
34 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
35 VALUES (24,48,'N');
36
37 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
38 VALUES (25,50,'N');
39
40 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
41 VALUES (26,41,'P');
42
43 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
44 VALUES (23,42,'P');
45
46 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
47 VALUES (24,43,'P');
48
49 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
50 VALUES (25,44,'Z');
51
52 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
53 VALUES (26,45,'Z');
54
55 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
56 VALUES (23,46,'Z');
57
58
```

```

59 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
60 VALUES (24,47,'A');
61
62 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
63 VALUES (25,48,'A');
64
65 INSERT INTO rezerwacje(id_wycieczki , id_osoby , status)
66 VALUES (26,49,'A');
67
68 INSERT INTO wycieczki (nazwa , kraj , data , opis , liczba_miejsc)
69 VALUES ('Tour to Paris',France,TO_DATE('2017-03-03','YYYY-MM-DD'),
70 'Ciekawa wycieczka ...',5);
71
72 INSERT INTO wycieczki (nazwa , kraj , data , opis , liczba_miejsc)
73 VALUES ('Tour to Krakow',Poland,TO_DATE('2018-03-03','YYYY-MM-DD'),
74 'Ciekawa wycieczka ...',5);
75
76 INSERT INTO wycieczki (nazwa , kraj , data , opis , liczba_miejsc)
77 VALUES ('Tour to Moscow',Russia,TO_DATE('2017-03-03','YYYY-MM-DD'),
78 'Ciekawa wycieczka ...',4);
79
80 INSERT INTO wycieczki (nazwa , kraj , data , opis , liczba_miejsc)
81 VALUES ('Tour to Manchester',England,TO_DATE('2020-03-03','YYYY-MM-DD'),
82 'Ciekawa wycieczka ...',6);
83





```

Tabele wypelnione danymi

	ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC
1	23	Tour to Paris	France	2017-03-03 00:00:00	Ciekawa wycieczka ...	5
2	24	Tour to Krakow	Poland	2017-06-04 00:00:00	Lorem Ipsum ...	5
3	25	Tour to Moscow	Russia	2020-05-03 00:00:00	Lorem Ipsum ...	4
4	26	Tour to Manchester	England	2018-04-06 00:00:00	Lorem Ipsum ...	6

	ID_OSOBY	IMIE	NAZWISKO	PESEL	KONTAKT
1	45	Grzegorz	Poreba	99451811753	675559123
2	46	Grzegorz	Pach	98451811753	675859123
3	47	Mateusz	Surjak	68451811753	975159123
4	48	Zuzanna	Brzezinska	55451811753	875159123
5	49	Sebastian	Kusnierz	15451811753	875156123
6	50	Jakub	Perzylo	35451811753	375156123
7	41	Karol	Musur	99432811753	774379200
8	42	Ola	Mazur	99411811753	774379123
9	43	Radek	Kopiec	19411811753	174379123
10	44	Andrzej	Podobinski	99111811753	175559123

	 NR_REZERWACJI ▾	 ID_WYCIECZKI ▾	 ID_OSOBY ▾	 STATUS ▾
1	21	24	45	N
2	1	23	45	N
3	2	24	48	N
4	3	25	50	A
5	4	26	41	P
6	5	23	42	P
7	7	24	43	P
8	8	25	44	A
9	9	26	45	Z
10	11	23	46	Z
11	12	24	47	A
12	13	25	48	A
13	14	26	49	A
14	43	23	44	A

## 3 Widoki

### 3.1 rezerwacje\_wszystkie

```
1 create view REZERWACJEWSZYSTKIE as
2 SELECT w.KRAJ, w.DATA,
3 w.NAZWA,
4 o.IMIE,
5 o.NAZWISKO,
6 r.STATUS
7 FROM WYCIECZKI w
8 JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
9 JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
10
11
12
```

	ID_WYCIECZKI	KRAJ	DATA	NAZWA	IMIE	NAZWISKO	STATUS
1	23	France	2017-03-03 00:00:00	Tour to Paris	Grzegorz	Poreba	N
2	24	Poland	2017-06-04 00:00:00	Tour to Krakow	Grzegorz	Poreba	N
3	26	England	2018-04-06 00:00:00	Tour to Manchester	Grzegorz	Poreba	Z
4	23	France	2017-03-03 00:00:00	Tour to Paris	Grzegorz	Pach	Z
5	24	Poland	2017-06-04 00:00:00	Tour to Krakow	Mateusz	Surjak	A
6	24	Poland	2017-06-04 00:00:00	Tour to Krakow	Zuzanna	Brzezinska	N
7	25	Russia	2020-05-03 00:00:00	Tour to Moscow	Zuzanna	Brzezinska	A
8	26	England	2018-04-06 00:00:00	Tour to Manchester	Sebastian	Kusnierz	A
9	25	Russia	2020-05-03 00:00:00	Tour to Moscow	Jakub	Perzylo	A
10	26	England	2018-04-06 00:00:00	Tour to Manchester	Karol	Musur	P
11	23	France	2017-03-03 00:00:00	Tour to Paris	Ola	Mazur	P
12	24	Poland	2017-06-04 00:00:00	Tour to Krakow	Radek	Kopec	P
13	23	France	2017-03-03 00:00:00	Tour to Paris	Andrzej	Podobinski	A
14	25	Russia	2020-05-03 00:00:00	Tour to Moscow	Andrzej	Podobinski	A

### 3.2 rezerwacje\_potwierdzone

```
1 create view REZERWACJEPOTWIERDZONE as
2 select KRAJ, Data, NAZWA, IMIE, NAZWISKO, STATUS
3 from REZERWACJEWSZYSTKIE rw
4 where rw.STATUS = 'P'
5
```

	KRAJ	DATA	NAZWA	IMIE	NAZWISKO	STATUS
1	France	2017-03-03 00:00:00	Tour to Paris	Ola	Mazur	P
2	Poland	2017-06-04 00:00:00	Tour to Krakow	Radek	Kopec	P
3	England	2018-04-06 00:00:00	Tour to Manchester	Karol	Musur	P

### 3.3 rezerwacje\_w\_przyszlosci

```
1 create view REZERWACJEWPRZYSZLOSCI as
2 select KRAJ, DATA, NAZWA, IMIE, NAZWISKO, STATUS
3 from REZERWACJEWSZYSTKIE
4 where DATA > sysdate
5
```

	ID_WYCIECZKI	KRAJ	DATA	NAZWA	IMIE	NAZWISKO	STATUS
1	25	Russia	2020-05-03 00:00:00	Tour to Moscow	Zuzanna	Brzezinska	A
2	25	Russia	2020-05-03 00:00:00	Tour to Moscow	Jakub	Perzylo	A
3	25	Russia	2020-05-03 00:00:00	Tour to Moscow	Andrzej	Podobinski	A

### 3.4 wycieczki\_miejsca

```
1 create view WYCIECZKIMIEJSCA as
2 select ID_WYCIECZKI,KRAJ,
3        DATA,
4        LICZBA_MIEJSC,
5        LICZBA_MIEJSC - (select count(*)
6                          from REZERWACJE
7                          where ID_WYCIECZKI = w.ID_WYCIECZKI
8        ) + (select count(*)
9              from REZERWACJE
10             where REZERWACJE.STATUS = 'A' and REZERWACJE.ID_WYCIECZKI = w.
11               ID_WYCIECZKI) as Liczba_wolnych_miejsc
12 from WYCIECZKI w
```

	ID_WYCIECZKI	KRAJ	DATA	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	23	France	2017-03-03 00:00:00	5	2
2	24	Poland	2017-06-04 00:00:00	5	2
3	25	Russia	2020-05-03 00:00:00	4	4
4	26	England	2018-04-06 00:00:00	6	4

### 3.5 wycieczki\_dostepne

```
1 create view WYCIECZKIDOSTEPNE as
2 select "ID_WYCIECZKI", "KRAJ", "DATA", "LICZBA_MIEJSC", "LICZBA_WOLNYCH_MIEJSC"
3 from WycieczkiMiejsc
4 where Liczba_wolnych_miejsc > 0 and DATA > sysdate
5
```

	ID_WYCIECZKI	KRAJ	DATA	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	25	Russia	2020-05-03 00:00:00	4	4



## 4 Obiekty

Potrzebne aby działały procedury pobierające dane.

### 4.1 wycieczka

```
1 create TYPE WYC_O AS object (  
2   id_wycieczki  NUMBER,  
3   nazwa         VARCHAR2(100) ,  
4   kraj         VARCHAR2(50) ,  
5   "data"       DATE,  
6   opis         VARCHAR2(200) ,  
7   liczba_miejsc NUMBER  
8 )  
9  
10 create type WYC_TAB as table of WYC_O  
11
```

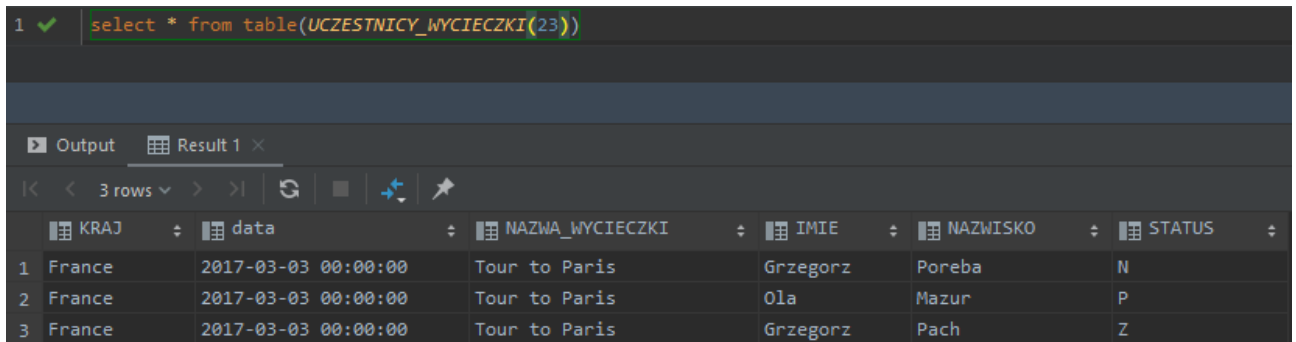
### 4.2 osoba\_wycieczka

```
1 create TYPE OS_WYC_O AS object (  
2   kraj         VARCHAR(50) ,  
3   "data"       DATE,  
4   nazwa_wycieczki VARCHAR(100) ,  
5   imie         VARCHAR2(50) ,  
6   nazwisko     VARCHAR2(50) ,  
7   status       CHAR(1)  
8 )  
9  
10 create type OS_WYC_TAB as table of OS_WYC_O  
11
```

## 5 Funkcje pobierające dane

### 5.1 uczestnicy\_wycieczki

```
1 create FUNCTION uczestnicy_wycieczki(id WYCIECZKI.ID_WYCIECZKI%TYPE)
2 return OS_WYC_TAB as tab_ret OS_WYC_TAB;
3 istnieje integer;
4 BEGIN
5     SELECT COUNT(*) INTO istnieje FROM WYCIECZKI WHERE WYCIECZKI.ID_WYCIECZKI =
6     id;
7     IF istnieje = 0 THEN
8         raise_application_error(-20001, 'WYCIECZKA O PODANYM ID NIE ISTNIEJE');
9     END IF;
10    SELECT OS_WYC_O(w.KRAJ, w.DATA, w.NAZWA, o.IMIE,
11                  o.NAZWISKO, r.STATUS)
12    BULK COLLECT INTO tab_ret
13    FROM WYCIECZKI w
14         JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
15         JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
16    WHERE w.ID_WYCIECZKI = id AND r.STATUS <> 'A';
17    return tab_ret;
18 end uczestnicy_wycieczki;
```



The screenshot shows a SQL query execution interface. The query is `select * from table(UCZESTNICY_WYCIECZKI(23))`. The result is displayed in a table with 7 columns: KRAJ, data, NAZWA\_WYCIECZKI, IMIE, NAZWISKO, and STATUS. There are 3 rows of data.

	KRAJ	data	NAZWA_WYCIECZKI	IMIE	NAZWISKO	STATUS
1	France	2017-03-03 00:00:00	Tour to Paris	Grzegorz	Poreba	N
2	France	2017-03-03 00:00:00	Tour to Paris	Ola	Mazur	P
3	France	2017-03-03 00:00:00	Tour to Paris	Grzegorz	Pach	Z

### 5.2 rezerwacje\_osoby

```
1 create FUNCTION rezerwacje_osoby(id_osoby OSOBY.ID_OSOBY%TYPE)
2 return OS_WYC_TAB as tab_ret OS_WYC_TAB;
3 istnieje integer;
4 BEGIN
5     SELECT COUNT(*) INTO istnieje FROM OSOBY WHERE OSOBY.ID_OSOBY =
6     rezerwacje_osoby.id_osoby;
7     IF istnieje = 0 THEN
8         raise_application_error(-20001, 'OSOBA O PODANYM ID NIE ISTNIEJE');
9     END IF;
10    SELECT OS_WYC_O(w.KRAJ, w.DATA, w.NAZWA, o.IMIE,
11                  o.NAZWISKO, r.STATUS)
12    BULK COLLECT INTO tab_ret
13    FROM WYCIECZKI w
14         JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
15         JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
16    WHERE o.ID_OSOBY = rezerwacje_osoby.id_osoby AND r.STATUS <> 'A';
17    return tab_ret;
18 end rezerwacje_osoby;
```

```
1 ✓ select * from table(REZERWACJE_OSOBY(42))
```

Output Result 2					
KRAJ	data	NAZWA_WYCIECZKI	IMIE	NAZWISKO	STATUS
France	2017-03-03 00:00:00	Tour to Paris	Ola	Mazur	P

### 5.3 dostępne\_wycieczki

```

1 create FUNCTION dostępne_wycieczki(kraj WYCIECZKI.KRAJ%TYPE, data_od DATE,
2                                     data_do DATE)
3 return WYC_TAB as tab_ret WYC_TAB;
4 istnieje integer;
5 BEGIN
6     IF data_do < data_od
7     THEN
8         raise_application_error(-20001, 'DATA KONCOWA JEST PRZED POZATKOWA');
9     END IF;
10
11     SELECT WYC_O(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA, w.OPIS,
12                w.LICZBA_MIEJSC)
13     BULK COLLECT INTO tab_ret
14 FROM WYCIECZKI w
15 WHERE w.KRAJ = dostępne_wycieczki.kraj
16       AND w.DATA >= data_od
17       AND w.DATA <= data_do
18       AND w.LICZBA_MIEJSC > (SELECT COUNT(*)
19                              FROM REZERWACJE r
20                              WHERE r.status <> 'A'
21                              AND r.ID_WYCIECZKI = w.ID_WYCIECZKI);
22
23     return tab_ret;
24 end dostępne_wycieczki;

```

```
select *
from table (DOSTEPNE_WYCIECZKI('Russia', TO_DATE('2019-09-04', 'yyyy-mm-dd'), TO_DATE('2021-09-04', 'yyyy-mm-dd')))
```

Output Result 4					
ID_WYCIECZKI	NAZWA	KRAJ	data	OPIS	LICZBA_MIEJSC
1	25 Tour to Moscow	Russia	2020-05-03 00:00:00	Lorem Ipsum ...	4

## 6 Procedury modyfikujące dane

Poniższe procedury uwzględniają użycie tabeli **REZERWACJE\_LOG** z punktu 6.

### 6.1 dodaj\_rezerwacje

```
1 create PROCEDURE
2   dodaj_rezerwacje(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE,
3                   id_osoby      OSOBY.ID_OSOBY%TYPE) AS
4   istnieje integer;
5   id_nowej_rezerwacji integer;
6
7 BEGIN
8   SELECT COUNT(*) INTO istnieje
9   FROM OSOBY
10  WHERE OSOBY.ID_OSOBY = dodaj_rezerwacje.id_osoby;
11
12  IF istnieje = 0
13  THEN
14    raise_application_error(-20001, 'NIE MA OSOBY O PODANYM ID');
15  END IF;
16
17  SELECT COUNT(*) INTO istnieje
18  FROM WYCIECZKIDOSTEPNE w
19  WHERE w.ID_WYCIECZKI = dodaj_rezerwacje.id_wycieczki;
20
21  IF istnieje = 0
22  THEN
23    raise_application_error(-20001, 'BRAK DOSTEPNEJ WYCIECZKI O PODANYM ID');
24  END IF;
25
26  SELECT COUNT(*) INTO istnieje
27  FROM REZERWACJE r
28  WHERE r.ID_WYCIECZKI = dodaj_rezerwacje.id_wycieczki
29         AND r.ID_OSOBY = dodaj_rezerwacje.id_osoby;
30
31  IF istnieje > 0
32  THEN
33    raise_application_error(-20001, 'REZERWACJA JUZ ISTNIEJE');
34  END IF;
35
36  INSERT INTO REZERWACJE (id_wycieczki, id_osoby, STATUS)
37  VALUES (dodaj_rezerwacje.id_wycieczki, dodaj_rezerwacje.id_osoby, 'N');
38
39  SELECT "ISEQ$$_114907".currval INTO id_nowej_rezerwacji FROM dual;
40
41  INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
42  VALUES (id_nowej_rezerwacji, CURRENT_DATE, 'N');
43
44 END dodaj_rezerwacje;
```

```
call DODAJ_REZERWACJE(25,42)
```

	NR_REZERWACJI	ID_WYCIECZKI	ID_OSOBY	STATUS
1	21	24	45	N
2	1	23	45	N
3	2	24	48	N
4	3	25	50	A
5	4	26	41	P
6	5	23	42	P
7	7	24	43	P
8	8	25	44	A
9	9	26	45	Z
10	11	23	46	Z
11	12	24	47	A
12	13	25	48	A
13	14	26	49	A
14	61	25	42	N
15	43	23	44	A

Nowa rezerwacja otrzymała nowe ID i została dopisana z prawidłowym statusem 'N'.

## 6.2 zmien\_status\_rezerwacji

```
1 FROM WYCIECZKIDOSTEPNE wp
2     JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
3 WHERE r.NR_REZERWACJI = id_rezerwacji;
4 IF istnieje = 0
5 THEN
6     raise_application_error(-20001,
7         'NIE MOZNA ZMIENIC STATUSU REZERWACJI DLA
8         WYCIECZKI KTORA SIE JUZ ODBYLA');
9 end if;
10
11 SELECT status
12 INTO status_sary
13 FROM REZERWACJE
14 WHERE NR_REZERWACJI = id_rezerwacji;
15
16 CASE
17     WHEN status_sary IS NULL
18     THEN
19         raise_application_error(-20001, 'REZERWACJA NIE ISTNIEJE');
20
21     WHEN nowy_status = 'N'
22     THEN
23         raise_application_error(-20001,
24             'ISTNIEJ CA REZERWACJA NIE MOZE BYC NOWA
25             ');
26
27     WHEN status_sary = 'A'
28     THEN
29         SELECT COUNT(*)
30         INTO istnieje
31         FROM WYCIECZKIDOSTEPNE dwv
32             JOIN REZERWACJE r ON r.ID_WYCIECZKI = dwv.ID_WYCIECZKI
33         WHERE r.NR_REZERWACJI = id_rezerwacji;
34
35         IF istnieje = 0
36         THEN
37             raise_application_error(-20001,
38                 'BRAK WOLNYCH MIEJSC DO PRZYWROCENIA
39                 ANULOWANEJ REZERWACJI'
40 );
41         END IF;
42     ELSE null;
43 END CASE;
44
45 UPDATE REZERWACJE
46 SET STATUS = nowy_status
47 WHERE NR_REZERWACJI = id_rezerwacji;
48
49 INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
50 VALUES (id_rezerwacji, CURRENT_DATE, nowy_status);
51
52 END zmien_status_rezerwacji;
```

```
call ZMIEN_STATUS_REZERWACJI(61,'A')
```

	NR_REZERWACJI	ID_WYCIECZKI	ID_OSOBY	STATUS
1	21	24	45	N
2	1	23	45	N
3	2	24	48	N
4	3	25	50	A
5	4	26	41	P
6	5	23	42	P
7	7	24	43	P
8	8	25	44	A
9	9	26	45	Z
10	11	23	46	Z
11	12	24	47	A
12	13	25	48	A
13	14	26	49	A
14	61	25	42	A
15	43	23	44	A

Wcześniej wycieczka miała status 'N', prawidłowo zmieniła swój status.

### 6.3 zmien\_liczbe\_miejsc

```

1 create PROCEDURE zmien_liczbe_miejsc(
2   id_w          WYCIECZKI.ID_WYCIECZKI%TYPE,
3   l_miejsc      WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
4   zajete_miejsc integer;
5 BEGIN
6
7   SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC INTO zajete_miejsc
8   FROM WYCIECZKI.MIEJSCA wm
9   WHERE wm.ID_WYCIECZKI = id_w;
10
11   IF l_miejsc < 0 OR zajete_miejsc > l_miejsc
12   THEN
13     raise_application_error(-20001, 'ZBYT NISKA LICZBA MIEJSC');
14   end if;
15
16   UPDATE WYCIECZKI
17   SET LICZBA_MIEJSC = l_miejsc
18   WHERE ID_WYCIECZKI = id_w;
19 END;
20

```

Na wycieczce o ID 23 są 3 rezerwacje

```
call ZMIEN_LICZBE_MIEJSC(23,2)
```

```

BD_SURJAK> call ZMIEN_LICZBE_MIEJSC(23,2)
[2020-02-29 12:02:39] [72000][20001] ORA-20001: ZBYT NISKA LICZBA MIEJSC
[2020-02-29 12:02:39] ORA-06512: przy "BD_SURJAK.ZMIEN_LICZBE_MIEJSC", linia 13
[2020-02-29 12:02:39] Position: 5

```

Jak widać błąd został rzucony prawidłowo gdyż chciałem zmienić miejsca do dwóch.

```
call ZMIEN_LICZBE_MIEJSC(23,3)
```

	ID_WYCIECZKI	KRAJ	DATA	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	23	France	2017-03-03 00:00:00	3	0
2	24	Poland	2017-06-04 00:00:00	5	2
3	25	Russia	2020-05-03 00:00:00	4	4
4	26	England	2018-04-06 00:00:00	6	4

Zmiana miejsc do trzech działa poprawnie, teraz jest 0 wolnych miejsc.



## 7 Dodanie Rezerwacje\_log

Zmodyfikowane procedury dodające informacje do loga można znaleźć w **Procedury modyfikujące dane** z punktu 6.

```
1 create table REZERWACJE_LOG
2 (
3     ID          NUMBER generated as identity
4     constraint REZERWACJE_LOG_PK
5         primary key,
6     ID_REZERWACJI NUMBER,
7     DATA       DATE,
8     STATUS      CHAR
9 )
10
```

	ID	ID_REZERWACJI	DATA	STATUS
1	21	8	2020-02-28 20:28:44	A
2	22	8	2020-02-28 20:28:44	A
3	41	61	2020-02-29 10:53:08	N
4	42	61	2020-02-29 10:53:08	N
5	43	61	2020-02-29 10:57:33	A
6	44	61	2020-02-29 10:57:33	A
7	45	61	2020-02-29 10:57:33	A
8	1	43	2020-02-27 19:00:22	N
9	2	43	2020-02-27 19:02:20	A

Log wypełniał się podczas testowania różnych procedur.

## 8 Dodanie redundantnego atrybutu liczba\_wolnych\_miejsc do tabeli wycieczki

```
1 ALTER TABLE WYCIECZKI
2 ADD liczba_wolnych_miejsc INT;
3
```

### 8.1 Procedura PRZELICZ modyfikująca tabelę wycieczki

```
1 create PROCEDURE przelicz AS
2 BEGIN
3     UPDATE WYCIECZKI w
4     SET LICZBA_WOLNYCH_MIEJSC = LICZBA_MIEJSC - (SELECT COUNT(*)
5                                                    FROM REZERWACJE r
6                                                    WHERE r.ID_WYCIECZKI = w.
7                                                    AND r.STATUS <> 'A');
8 END;
9
```

ID_WYCIECZKI	NAZWA	KRAJ	DATA	OPIS	LICZBA_MIEJSC	LICZBA_WOLNYCH_MIEJSC
1	23 Tour to Paris	France	2017-03-03 00:00:00	Ciekawa wycieczka ...	3	0
2	24 Tour to Krakow	Poland	2017-06-04 00:00:00	Lorem Ipsum ...	5	2
3	25 Tour to Moscow	Russia	2020-05-03 00:00:00	Lorem Ipsum ...	4	6
4	26 Tour to Manchester	England	2018-04-06 00:00:00	Lorem Ipsum ...	6	4

### 8.2 Zmodyfikowane widoki

#### 8.2.1 dostępne\_wycieczki\_2

```
1 create view DOSTEPNE_WYCIECZKI_2 as
2 SELECT id_wycieczki ,
3        kraj ,
4        data ,
5        nazwa ,
6        liczba_miejsc ,
7        liczba_wolnych_miejsc
8 FROM WYCIECZKI
9 WHERE liczba_wolnych_miejsc > 0
10 AND DATA > CURRENT_DATE
11
```

#### 8.2.2 wycieczki\_miejsca\_2

```
1 create view WYCIECZKI_MIEJSCA_2 as
2 SELECT ID_WYCIECZKI,
3        kraj ,
4        data ,
5        nazwa ,
6        liczba_miejsc ,
7        LICZBA_WOLNYCH_MIEJSC
8 FROM wycieczki
9
```

Nie zamieściłem screenów powyższych widokó ponieważ są identyczne jak screeny widoków z podpunktu 3.

## 8.3 Zmodyfikowane procedury i funkcje

### 8.3.1 dodaj\_rezerwacje

```
1 create PROCEDURE
2   dodaj_rezerwacje_2(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE,
3                     id_osoby      OSOBY.ID_OSOBY%TYPE) AS
4   istnieje integer;
5   id_rezerwacji integer;
6 BEGIN
7   SELECT COUNT(*) INTO istnieje
8   FROM OSOBY
9   WHERE OSOBY.ID_OSOBY = dodaj_rezerwacje_2.id_osoby;
10
11   IF istnieje = 0
12   THEN
13     raise_application_error(-20001, 'OSOBA O PODANYM ID NIE ISTNIEJE');
14   END IF;
15
16   SELECT COUNT(*) INTO istnieje
17   FROM WYCIECZKIDOSTEPNE w
18   WHERE w.ID_WYCIECZKI = dodaj_rezerwacje_2.id_wycieczki;
19
20   IF istnieje = 0
21   THEN
22     raise_application_error(-20001, 'WYCIECZKA O PODANYM ID NIE ISTNIEJE');
23   END IF;
24
25   SELECT COUNT(*) INTO istnieje
26   FROM REZERWACJE r
27   WHERE r.ID_WYCIECZKI = dodaj_rezerwacje_2.id_wycieczki
28         AND r.ID_OSOBY = dodaj_rezerwacje_2.id_osoby;
29
30   IF istnieje > 0
31   THEN
32     raise_application_error(-20001, 'REZERWACJA JUZ ISTNIEJE');
33   END IF;
34
35   INSERT INTO REZERWACJE (id_wycieczki, id_osoby, STATUS)
36   VALUES (dodaj_rezerwacje_2.id_wycieczki, dodaj_rezerwacje_2.id_osoby, 'N');
37
38   UPDATE WYCIECZKI
39   SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1
40   WHERE ID_WYCIECZKI = dodaj_rezerwacje_2.id_wycieczki;
41
42
43   SELECT "ISEQ$$$_114907".currval INTO id_rezerwacji FROM dual;
44
45   INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
46   VALUES (id_rezerwacji, CURRENT_DATE, 'N');
47
48 END dodaj_rezerwacje_2;
```

### 8.3.2 dostępne\_wycieczki\_v2

```
1 create FUNCTION dostępne_wycieczki_v2(kraj WYCIECZKI.KRAJ%TYPE, data_od DATE,
2                                     data_do DATE)
3 return WYC_TAB as ret_tab WYC_TAB;
4 BEGIN
5     IF data_do < data_od
6     THEN
7         raise_application_error(-20001, 'DATA KONCOWA JEST PRZED POZATKOWA');
8     END IF;
9
10    SELECT WYC_O(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA, w.OPIS,
11                w.LICZBA_MIEJSC)
12        BULK COLLECT INTO ret_tab
13    FROM WYCIECZKI w
14    WHERE w.KRAJ = dostępne_wycieczki_v2.kraj
15        AND w.DATA >= data_od
16        AND w.DATA <= data_do
17        AND w.LICZBA_WOLNYCH_MIEJSC > 0;
18    return ret_tab;
19 end dostępne_wycieczki_v2;
20
```

### 8.3.3 zmien\_status\_rezerwacji\_2

```
1 create PROCEDURE
2 zmiana_statusu_rezerwacji_2(id_rezerwacji REZERWACJE.NR_REZERWACJE%TYPE,
3                             nowy_status REZERWACJE.STATUS%TYPE) AS
4 stary_status REZERWACJE.STATUS%TYPE;
5 istnieje integer;
6 counter integer;
7 BEGIN
8     SELECT COUNT(*) INTO istnieje
9     FROM REZERWACJEWPZYSZLOSCI wp
10    JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
11    WHERE r.NR_REZERWACJI = id_rezerwacji;
12    IF istnieje = 0
13    THEN
14        raise_application_error(-20001,
15                                'WYCIECZKA JUZ SIE ODBYLA');
16    end if;
17
18    SELECT status INTO stary_status
19    FROM REZERWACJE
20    WHERE NR_REZERWACJI = id_rezerwacji;
21
22    IF stary_status = nowy_status
23    THEN
24        raise_application_error(-20001,
25                                'NALEZY PODAC STATUS INNY NIZ OBECNY');
26    END IF;
27
28    CASE
29    WHEN stary_status IS NULL
30    THEN
31        raise_application_error(-20001, 'NIE MA REZERWACJI O PODANYM ID');
32
33    WHEN nowy_status = 'N'
34    THEN
```

```

35      raise_application_error(-20001,
36                          'REZERWACJA NIE MOZE STAC SIE NOWA');
37
38
39  WHEN stary_status = 'A'
40  THEN
41      SELECT COUNT(*) INTO istnieje
42      FROM WYCIECZKIDOSTEPNE dwv
43           JOIN REZERWACJE r ON r.ID_WYCIECZKI = dwv.ID_WYCIECZKI
44      WHERE r.NR_REZERWACJI = id_rezerwacji;
45
46      IF istnieje = 0
47      THEN
48          raise_application_error(-20001,
49                          'BRAK WOLNYCH MIEJSC NA DANA WYCIECZKE');
50      ELSE
51          counter := -1;
52      END IF;
53  ELSE
54      IF nowy_status = 'A'
55      THEN
56          counter := 1;
57      ELSE
58          counter := 0;
59      end if;
60  END CASE;
61
62  UPDATE REZERWACJE
63  SET STATUS = nowy_status
64  WHERE NR_REZERWACJI = id_rezerwacji;
65
66  UPDATE WYCIECZKI w
67  SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + counter
68  WHERE w.ID_WYCIECZKI = (SELECT ID_WYCIECZKI
69                          FROM REZERWACJE r
70                          WHERE r.NR_REZERWACJI = id_rezerwacji);
71
72  INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
73  VALUES (id_rezerwacji, CURRENT_DATE, nowy_status);
74
75  END zmiana_statusu_rezerwacji_2;
76

```

### 8.3.4 zmien\_liczbe\_miejsc\_2

```
1 create PROCEDURE zmien_liczbe_miejsc_2(  
2 id_w          WYCIECZKI.ID_WYCIECZKI%TYPE,  
3 nowa_liczba_miejsc WYCIECZKI.LICZBA_MIEJSC%TYPE) AS  
4 zajete_miejsca integer;  
5 BEGIN  
6     SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC INTO zajete_miejsca  
7     FROM WYCIECZKI w  
8     WHERE w.ID_WYCIECZKI = id_w;  
9  
10    IF nowa_liczba_miejsc < 0 OR zajete_miejsca > nowa_liczba_miejsc  
11    THEN  
12        raise_application_error(-20001, 'LICZBA MIEJSC NIE MOZE BYC 0 LUB NIE MOZE  
13        BYC NIZSZA NIZ LICZBA JUZ ZAREZERWOWANYCH MIEJSC');  
14    end if;  
15  
16    UPDATE WYCIECZKI  
17    SET LICZBA_MIEJSC          = nowa_liczba_miejsc ,  
18        LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC +  
19        (nowa_liczba_miejsc - LICZBA_MIEJSC)  
20    WHERE ID_WYCIECZKI = id_w;  
21 END;
```

## 9 Dodanie triggerów

### 9.1 trigger obsługujący dodanie rezerwacji

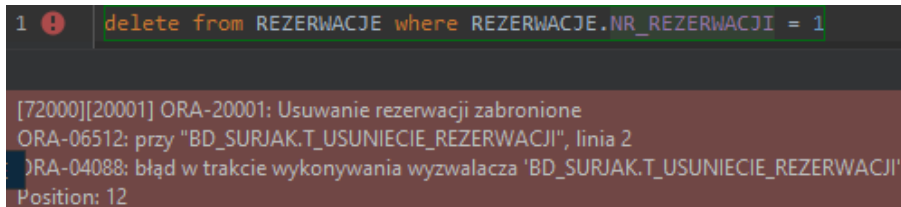
```
1 CREATE OR REPLACE TRIGGER dodanie_rezerwacji_trigger
2 AFTER UPDATE
3 ON REZERWACJE
4 FOR EACH ROW
5 DECLARE
6     counter int;
7 BEGIN
8     INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
9     VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);
10    CASE
11        WHEN :OLD.STATUS = 'A' AND :NEW.STATUS <> 'A'
12        THEN
13            counter := -1;
14        WHEN :OLD.STATUS <> 'A' AND :NEW.STATUS = 'A'
15        THEN
16            counter := 1;
17    ELSE
18        counter := 0;
19    END CASE;
20
21    UPDATE WYCIECZKI w
22    SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + counter
23    WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
24 end dodanie_rezerwacji_trigger;
25
```

### 9.2 trigger obsługujący zmianę statusu

```
1 CREATE OR REPLACE TRIGGER zmiana_statusu_trigger
2 AFTER UPDATE
3 ON REZERWACJE
4 FOR EACH ROW
5 DECLARE
6     counter int;
7 BEGIN
8     INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
9     VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);
10    CASE
11        WHEN :OLD.STATUS = 'A' AND :NEW.STATUS <> 'A'
12        THEN
13            counter := -1;
14        WHEN :OLD.STATUS <> 'A' AND :NEW.STATUS = 'A'
15        THEN
16            counter := 1;
17    ELSE
18        counter := 0;
19    END CASE;
20
21    UPDATE WYCIECZKI w
22    SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + counter
23    WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
24 end zmiana_statusu_trigger;
25
```

### 9.3 trigger zabraniający usunięcia rezerwacji

```
1 CREATE OR REPLACE TRIGGER usuniecie_rezerwacji_trigger
2 BEFORE DELETE
3 ON REZERWACJE
4 FOR EACH ROW
5 BEGIN
6     raise_application_error(-20001, 'NIE MOZNA USUNAC REZERWACJI');
7 end;
8
```



The screenshot shows a SQL IDE interface. At the top, a query is entered: `delete from REZERWACJE where REZERWACJE.NR_REZERWACJI = 1`. Below the query, an error message is displayed in a red box: `[72000][20001] ORA-20001: Usuwanie rezerwacji zabronione`. Below this, more details are shown: `ORA-06512: przy "BD_SURJAK.T_USUNIECIE_REZERWACJI", linia 2` and `ORA-04088: błąd w trakcie wykonywania wyzwalacza 'BD_SURJAK.T_USUNIECIE_REZERWACJI'`. At the bottom, it says `Position: 12`.

Trigger działa - próba usunięcia rezerwacji powoduje wyrzucenie błędu.

### 9.4 trigger obsługujący zmianę liczby miejsc na poziomie wycieczki

```
1 CREATE OR REPLACE TRIGGER zmiana_liczby_miejsc_trigger
2 BEFORE UPDATE OF liczba_miejsc
3 ON wycieczki
4 FOR EACH ROW
5 BEGIN
6     SELECT :OLD.LICZBA_WOLNYCH_MIEJSC +
7         (:NEW.LICZBA_MIEJSC - :OLD.LICZBA_MIEJSC) INTO :NEW.
8         LICZBA_WOLNYCH_MIEJSC
9 FROM Dual;
10 END;
```



## 10 Procedury współpracujące z triggerami

### 10.1 dodaj\_rezerwacje\_3

```
1 create PROCEDURE
2   dodaj_rezerwacje_3(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE,
3                     id_osoby      OSOBY.ID_OSOBY%TYPE) AS
4   istnieje integer;
5 BEGIN
6   SELECT COUNT(*) INTO istnieje
7   FROM OSOBY
8   WHERE OSOBY.ID_OSOBY = dodaj_rezerwacje_3.id_osoby;
9
10  IF istnieje = 0
11  THEN
12    raise_application_error(-20001, 'OSOBA O PODANYM ID NIE ISTNIEJE');
13  END IF;
14
15  SELECT COUNT(*) INTO istnieje
16  FROM WYCIECZKIDOSTEPNE w
17  WHERE w.ID_WYCIECZKI = dodaj_rezerwacje_3.id_wycieczki;
18
19  IF istnieje = 0
20  THEN
21    raise_application_error(-20001, 'WYCIECZKA O PODANYM ID NIE ISTNIEJE');
22  END IF;
23
24  SELECT COUNT(*) INTO istnieje
25  FROM REZERWACJE r
26  WHERE r.ID_WYCIECZKI = dodaj_rezerwacje_3.id_wycieczki
27         AND r.ID_OSOBY = dodaj_rezerwacje_3.id_osoby;
28
29  IF istnieje > 0
30  THEN
31    raise_application_error(-20001, 'REZERWACJA JUZ ISTNIEJE');
32  END IF;
33
34  INSERT INTO REZERWACJE (id_wycieczki, id_osoby, STATUS)
35  VALUES (dodaj_rezerwacje_3.id_wycieczki, dodaj_rezerwacje_3.id_osoby, 'N');
36 END dodaj_rezerwacje_3;
37
```

## 10.2 zmien\_status\_rezerwacji\_3

```
1  zmiana_statusu_rezerwacji_3(id_rezerwacji REZERWACJE.NR_REZERWACJE%TYPE,  
2                               nowy_status   REZERWACJE.STATUS%TYPE) AS  
3  stary_status                 REZERWACJE.STATUS%TYPE;  
4  istnieje                     integer;  
5  BEGIN  
6      SELECT COUNT(*) INTO istnieje  
7      FROM REZERWACJEWPRZYSZLOSCI wp  
8           JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI  
9      WHERE r.NR_REZERWACJI = id_rezerwacji;  
10     IF istnieje = 0  
11     THEN  
12         raise_application_error(-20001,  
13                                 'WYCIECZKA JUZ SIE ODBYLA');  
14     end if;  
15  
16     SELECT status INTO stary_status  
17     FROM REZERWACJE  
18     WHERE NR_REZERWACJI = id_rezerwacji;  
19  
20     IF stary_status = nowy_status  
21     THEN  
22         raise_application_error(-20001,  
23                                 'NALEZY PODAC STATUS INNY NIZ OBECNY');  
24     END IF;  
25  
26     CASE  
27     WHEN stary_status IS NULL  
28     THEN  
29         raise_application_error(-20001, 'REZERWACJA JUZ ISTNIEJE');  
30  
31     WHEN nowy_status = 'N'  
32     THEN  
33         raise_application_error(-20001,  
34                                 'NIE MOZNA ZMIENIC ISTNIEJACEJ REZERWACJI NA NOWA  
35                                 ');  
36  
37     WHEN stary_status = 'A'  
38     THEN  
39         SELECT COUNT(*) INTO istnieje  
40         FROM WYCIECZKIDOSTEPNE dwv  
41              JOIN REZERWACJE r ON r.ID_WYCIECZKI = dwv.ID_WYCIECZKI  
42         WHERE r.NR_REZERWACJI = id_rezerwacji;  
43  
44         IF istnieje = 0  
45         THEN  
46             raise_application_error(-20001,  
47                                     'BRAK WOLNYCH MIEJSC');  
48         END IF;  
49     ELSE NULL;  
50     END CASE;  
51  
52     UPDATE REZERWACJE  
53     SET STATUS = nowy_status  
54     WHERE NR_REZERWACJI = id_rezerwacji;  
55 END zmien_statusu_rezerwacji_3;  
56
```

### 10.3 zmien\_liczbe\_miejsc\_3

```
1 create PROCEDURE zmien_liczbe_miejsc_3(  
2 id_w          WYCIECZKI.ID_WYCIECZKI%TYPE,  
3 nowe_miejsca WYCIECZKI.LICZBA_MIEJSC%TYPE) AS  
4 zajete integer;  
5 BEGIN  
6  
7     SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC INTO zajete  
8     FROM WYCIECZKI w  
9     WHERE w.ID_WYCIECZKI = id_w;  
10  
11     IF nowe_miejsca < 0 OR zajete > nowe_miejsca  
12     THEN  
13         raise_application_error(-20001, 'LICZBA MIEJSC NIE MOZE BYC 0 LUB NIE MOZE  
14         BYC NIZSZA NIZ LICZBA JUZ ZAREZERWOWANYCH MIEJSC');  
15     end if;  
16  
17     UPDATE WYCIECZKI  
18     SET LICZBA_MIEJSC = nowe_miejsca  
19     WHERE ID_WYCIECZKI = id_w;  
20 END;
```