Spis treści

1. Tabele i warunki integralnościowe

Zgodnie z poleceniami z instrukcji utworzono podstawowe tabele bazy danych.

1.1. Osoby

```
CREATE TABLE OSOBY
(
    ID_OSOBY INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    IMIE VARCHAR2(50),
    NAZWISKO VARCHAR2(50),
    PESEL VARCHAR2(11),
    KONTAKT VARCHAR2(100),
    CONSTRAINT OSOBY_PK PRIMARY KEY (ID_OSOBY) ENABLE
);
```

1.2. Wycieczki

```
CREATE TABLE WYCIECZKI
(
    ID_WYCIECZKI INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    NAZWA VARCHAR2(100),
    KRAJ VARCHAR2(50),
    DATA DATE,
    OPIS VARCHAR2(200),
    LICZBA_MIEJSC INT,
    CONSTRAINT WYCIECZKI_PK PRIMARY KEY (ID_WYCIECZKI) ENABLE
);
```

1.3. Rezerwacje

```
CREATE TABLE REZERWACJE
(
    NR_REZERWACJI INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    ID_WYCIECZKI INT,
    ID_OSOBY INT,
    STATUS CHAR(1),
    CONSTRAINT REZERWACJE_PK PRIMARY KEY (NR_REZERWACJI) ENABLE
);
```

1.4. Dodatkowe warunki integralnościowe

```
1  ALTER TABLE REZERWACJE
2  ADD CONSTRAINT REZERWACJE_FK1
3  FOREIGN KEY (ID_OSOBY) REFERENCES OSOBY(ID_OSOBY) ENABLE;
4
5  ALTER TABLE REZERWACJE
6  ADD CONSTRAINT REZERWACJE_FK2
7  FOREIGN KEY (ID_WYCIECZKI) REFERENCES WYCIECZKI(ID_WYCIECZKI) ENABLE;
8
9  ALTER TABLE REZERWACJE
10 ADD CONSTRAINT REZERWACJE_CHK1
11 CHECK (STATUS IN ('N', 'P', 'Z', 'A')) ENABLE;
```

2. Przykładowe dane

```
1  INSERT INTO OSOBY (IMIE, NAZWISKO, PESEL, KONTAKT)
2  VALUES('Adam', 'Kowalski', '87654321', 'tel: 6623');
3
4  INSERT INTO OSOBY (IMIE, NAZWISKO, PESEL, KONTAKT)
5  VALUES('Jan', 'Nowak', '12345678', 'tel: 2312, dzwonić po 18.00');
6
7  INSERT INTO OSOBY (IMIE, NAZWISKO, PESEL, KONTAKT)
8  VALUES('Anna', 'Nowak', '12335478', 'tel: 6317');
9
10 INSERT INTO OSOBY (IMIE, NAZWISKO, PESEL, KONTAKT)
11 VALUES('Janina', 'Kowalska', '12356678', 'tel: 2372');
12
13 INSERT INTO OSOBY (IMIE, NAZWISKO, PESEL, KONTAKT)
14 VALUES('Andrzej', 'Polak', '17345688', 'tel: 2778');
15
16 INSERT INTO OSOBY (IMIE, NAZWISKO, PESEL, KONTAKT)
17 VALUES('Adam', 'Małysz', '17785688', 'tel: 0798');
18
19 INSERT INTO OSOBY (IMIE, NAZWISKO, PESEL, KONTAKT)
20 VALUES('Robert', 'Kubica', '5555688', 'tel: 5738');
21
22 INSERT INTO OSOBY (IMIE, NAZWISKO, PESEL, KONTAKT)
23 VALUES('Jan', 'Franko', '12345098', 'tel: 5568');
24
25 INSERT INTO OSOBY (IMIE, NAZWISKO, PESEL, KONTAKT)
26 VALUES('Andrzej', 'Strzelba', '1445688', 'tel: 5548');
27
28 INSERT INTO OSOBY (IMIE, NAZWISKO, PESEL, KONTAKT)
29 VALUES('Mariusz', 'Pudzianowski', '1443688', 'mail: mariusz@pudzianowski.pl');
30
31 INSERT INTO WYCIECZKI (NAZWA, KRAJ, DATA, OPIS, LICZBA_MIEJSC)
32 VALUES ('Wycieczka do Paryża', 'Francja',
33 TO_DATE('2016-01-01', 'YYYY-MM-DD'), 'Ciekawa wycieczka...', 3);
34
35 INSERT INTO WYCIECZKI (NAZWA, KRAJ, DATA, OPIS, LICZBA_MIEJSC)
36 VALUES ('Piękny Kraków', 'Polska',
37 TO_DATE('2017-02-03', 'YYYY-MM-DD'), 'Najciekawsza wycieczka...', 2);
38
39 INSERT INTO WYCIECZKI (NAZWA, KRAJ, DATA, OPIS, LICZBA_MIEJSC)
40 VALUES ('Wieliczka', 'Polska',
41 TO_DATE('2017-03-03', 'YYYY-MM-DD'), 'Zadziwiająca kopalnia...', 2);
42
```

```
INSERT INTO WYCIECZKI (NAZWA, KRAJ, DATA, OPIS, LICZBA_MIEJSC)
VALUES ('Wycieczka do Tyraspola', 'Mołdawia',
TO_DATE('2020-01-01', 'YYYY-MM-DD'), 'Niezapomniana wycieczka...', 4);

INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
VALUES (8,1,'N');

INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
VALUES (9, 2, 'P');

INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
VALUES (11, 2, 'A');

INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
VALUES (9, 5, 'A');

INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
VALUES (9, 10, 'Z');

INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
VALUES (11, 3, 'N');

INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
VALUES (10, 10, 'P');

INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
VALUES (10, 6, 'N');

INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
VALUES (8, 7, 'A');

INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
VALUES (11, 8, 'N');
```

3. Widoki

Utworzono widoki według wytycznych z instrukcji do ćwiczenia

3.1. Wycieczki_osoby

```
1  CREATE VIEW wycieczki_osoby
2  AS
3      SELECT
4          w.ID_WYCIECZKI,
5          w.NAZWA,
6          w.KRAJ,
7          w.DATA,
8          o.IMIE,
9          o.NAZWISKO,
10         r.STATUS
11     FROM WYCIECZKI w
12         JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
13         JOIN OSOBY o on r.ID_OSOBY = o.ID_OSOBY;
```

3.2. Wycieczki_osoby_potwierdzone

```
1  CREATE VIEW wycieczki_osoby_potwierdzone
2  AS
3      SELECT
4          w.ID_WYCIECZKI,
5          w.NAZWA,
6          w.KRAJ,
7          w.DATA,
8          o.IMIE,
9          o.NAZWISKO,
10         r.STATUS
11     FROM WYCIECZKI w
12         JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
13         JOIN OSOBY o on r.ID_OSOBY = o.ID_OSOBY
14         WHERE r.STATUS in ('P', 'Z');
```

3.3. Wycieczki_przyszle

```
CREATE VIEW wycieczki_przyszle
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 w.DATA > CURRENT_DATE;
```

3.4. Wycieczki_miejsca

```
CREATE VIEW wycieczki_miejsca
AS
    SELECT
        w.ID_WYCIECZKI,
        w.NAZWA,
        w.KRAJ,
        w.DATA,
        w.LICZBA_MIEJSC,
        w.LICZBA_MIEJSC - COUNT(DISTINCT r.ID_OSOBY) AS WOLNE_MIEJSCA
    FROM WYCIECZKI w
    LEFT JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
    WHERE r.STATUS <> 'A'
    GROUP BY w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA, w.LICZBA_MIEJSC;
```

3.5. Dostepne_wycieczki

```
CREATE VIEW dostepne_wycieczki
AS
    SELECT * FROM wycieczki_miejsca WHERE WOLNE_MIEJSCA > 0 and DATA > CURRENT_DATE;
```

Z uwagi na posiadanie przez ten widok identycznych atrybutów jak widok wycieczki_miejsca, zdecydowano się na odwołanie do niego bez powielania definicji.

3.6. Rezerwacje_do_anulowania

```
1  CREATE VIEW rezerwacje_do_anulowania
2  AS
3      SELECT
4          r.NR_REZERWACJI,
5          r.ID_WYCIECZKI,
6          r.ID_OSOBY,
7          r.STATUS
8      FROM REZERWACJE r
9      JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
10     WHERE r.STATUS = 'N' AND w.DATA - CURRENT_DATE BETWEEN 0 AND 7;
```

4. Funkcje pobierające dane

4.1. Uczestnicy_wycieczki

```sql
CREATE OR REPLACE TYPE uczestnicy_wycieczki_row AS OBJECT
(
    id_wycieczki INT,
    nazwa_wycieczki VARCHAR2(100),
    kraj VARCHAR2(50),
    data_wycieczki DATE,
    imie VARCHAR2(50),
    nazwisko VARCHAR2(50),
    status_rezerwacji CHAR(1)
);

CREATE OR REPLACE TYPE uczestnicy_wycieczki_table AS TABLE OF uczestnicy_wycieczki_row;

CREATE OR REPLACE FUNCTION UCZESTNICY_WYCIECZKI
(id_wycieczki IN WYCIECZKI.ID_WYCIECZKI%TYPE)
RETURN uczestnicy_wycieczki_table
AS
    uczestnicy_t uczestnicy_wycieczki_table;
    id_istnieje int;
BEGIN
    SELECT COUNT(*) INTO id_istnieje
    FROM WYCIECZKI
    WHERE WYCIECZKI.ID_WYCIECZKI = UCZESTNICY_WYCIECZKI.id_wycieczki;

    IF id_istnieje = 0 THEN
        raise_application_error(-20100, 'Wycieczka o podanym ID nie istnieje!');
    END IF;

    SELECT uczestnicy_wycieczki_row(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ,
                                    w.DATA, o.IMIE, o.NAZWISKO, r.STATUS)
    BULK COLLECT INTO uczestnicy_t
    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 <> 'A' AND w.ID_WYCIECZKI = UCZESTNICY_WYCIECZKI.id_wycieczki;

    RETURN uczestnicy_t;
END;
```

Aby móc zwrócić wynik jako tabelę, niezbędne było zdefiniowanie własnych typów danych – dla wiersza generowanej tabeli i dla całej tabeli, z których korzysta także kilka dalszych funkcji.

4.2. Rezerwacje_osoby

```
1  CREATE OR REPLACE FUNCTION REZERWACJE_OSOBY(id_osoby IN OSOBY.ID_OSOBY%TYPE)
2  RETURN uczestnicy_wycieczki_table
3  AS
4      rezerwacje_t uczestnicy_wycieczki_table;
5      id_istnieje int;
6  BEGIN
7      SELECT COUNT(*) INTO id_istnieje
8      FROM OSOBY
9      WHERE OSOBY.ID_OSOBY = REZERWACJE_OSOBY.id_osoby;
10
11     IF id_istnieje = 0 THEN
12         raise_application_error(-20101, 'Osoba o podanym ID nie istnieje!');
13     END IF;
14
15     SELECT uczestnicy_wycieczki_row(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ,
16                                     w.DATA, o.IMIE, o.NAZWISKO, r.STATUS)
17     BULK COLLECT INTO rezerwacje_t
18     FROM WYCIECZKI w
19     JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
20     JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
21     WHERE o.ID_OSOBY = REZERWACJE_OSOBY.id_osoby;
22
23     RETURN rezerwacje_t;
24 END;
```

4.3. Przyszle_rezerwacje_osoby

```
1  CREATE OR REPLACE FUNCTION PRZYSZLE_REZERWACJE_OSOBY(id_osoby IN OSOBY.ID_OSOBY%TYPE)
2  RETURN uczestnicy_wycieczki_table
3  AS
4      przyszle_rezerwacje_t uczestnicy_wycieczki_table;
5      id_istnieje int;
6  BEGIN
7      SELECT COUNT(*) INTO id_istnieje
8      FROM OSOBY
9      WHERE OSOBY.ID_OSOBY = PRZYSZLE_REZERWACJE_OSOBY.id_osoby;
10
11     IF id_istnieje = 0 THEN
12         raise_application_error(-20101, 'Osoba o podanym ID nie istnieje!');
13     END IF;
14
```

```
15      SELECT uczestnicy_wycieczki_row(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ,
16                                      w.DATA, o.IMIE, o.NAZWISKO, r.STATUS)
17      BULK COLLECT INTO przyszle_rezerwacje_t
18      FROM WYCIECZKI w
19      JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI
20      JOIN OSOBY o ON r.ID_OSOBY = o.ID_OSOBY
21      WHERE o.ID_OSOBY = PRZYSZLE_REZERWACJE_OSOBY.id_osoby AND w.DATA > CURRENT_DATE;
22
23      RETURN przyszle_rezerwacje_t;
24  END;
```

4.4. Dostepne_wycieczki

```
1   CREATE OR REPLACE TYPE dostepne_wycieczki_row AS OBJECT
2   (
3       id_wycieczki INT,
4       nazwa_wycieczki VARCHAR2(100),
5       kraj VARCHAR2(50),
6       data_wycieczki DATE,
7       liczba_miejsc INT,
8       liczba_wolnych_miejsc INT
9   );
10
11  CREATE OR REPLACE TYPE dostepne_wycieczki_table AS TABLE OF dostepne_wycieczki_row;
12
13  CREATE OR REPLACE FUNCTION DOSTEPNE_WYCIECZKI_FUN
14      (kraj IN WYCIECZKI.KRAJ%TYPE, data_od IN DATE, data_do IN DATE)
15  RETURN dostepne_wycieczki_table
16  AS
17      dostepne_wycieczki_t dostepne_wycieczki_table;
18  BEGIN
19      IF data_od > data_do THEN
20          raise_application_error(-20102,
21          'Podano niepoprawne daty: data końca przedziału
22          jest wcześniejsza niż jego początek');
23      END IF;
24
25      SELECT dostepne_wycieczki_row(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA,
26          w.LICZBA_MIEJSC,
27          w.LICZBA_MIEJSC -
28              (SELECT COUNT(*) FROM REZERWACJE r
29               WHERE r.STATUS <> 'A' AND r.ID_WYCIECZKI = w.ID_WYCIECZKI))
30      BULK COLLECT INTO dostepne_wycieczki_t
31      FROM WYCIECZKI w
32      WHERE w.KRAJ = DOSTEPNE_WYCIECZKI_FUN.kraj
33      AND w.DATA BETWEEN DOSTEPNE_WYCIECZKI_FUN.data_od
34      AND DOSTEPNE_WYCIECZKI_FUN.data_do
35      AND w.LICZBA_MIEJSC -
```

```
36        (SELECT COUNT(*) FROM REZERWACJE r
37         WHERE r.STATUS <> 'A' AND r.ID_WYCIECZKI = w.ID_WYCIECZKI) > 0;
38
39      RETURN dostepne_wycieczki_t;
40  END;
```

Analogicznie jak dla funkcji uczestnicy_wycieczki zdefiniowano dwa nowe typy danych.

5. Procedury modyfikujące dane

Kody przedstawionych funkcji obejmują modyfikację polegającą na aktualizowaniu tabeli REZERWACJE_LOG do dziennikowania zmian w tabeli REZERWACJE.

5.1. Dodaj_rezerwacje

```
1  CREATE OR REPLACE PROCEDURE DODAJ_REZERWACJE
2      (id_wycieczki IN WYCIECZKI.ID_WYCIECZKI%TYPE, id_osoby IN OSOBY.ID_OSOBY%TYPE)
3  AS
4      id_osoby_istnieje int;
5      id_wycieczki_istnieje int;
6      rezerwacja_istnieje int;
7      wolne_miejsca int;
8      id_nowej_rezerwacji int;
9  BEGIN
10     SELECT COUNT(*) INTO id_osoby_istnieje
11     FROM OSOBY
12     WHERE OSOBY.ID_OSOBY = DODAJ_REZERWACJE.id_osoby;
13     IF id_osoby_istnieje = 0 THEN
14         raise_application_error(-20101, 'Osoba o podanym ID nie istnieje!');
15     END IF;
16
17     SELECT COUNT(*) INTO id_wycieczki_istnieje
18     FROM WYCIECZKI_PRZYSZLE
19     WHERE WYCIECZKI_PRZYSZLE.ID_WYCIECZKI = DODAJ_REZERWACJE.id_wycieczki;
20     IF id_wycieczki_istnieje = 0 THEN
21         raise_application_error(-20100, 'Wycieczka o podanym ID
22         nie istnieje lub już się odbyła!');
23     END IF;
24
25     SELECT COUNT(*) INTO rezerwacja_istnieje
26     FROM REZERWACJE r
27     WHERE r.ID_WYCIECZKI = DODAJ_REZERWACJE.id_wycieczki
28     AND r.ID_OSOBY = DODAJ_REZERWACJE.id_osoby;
29     IF rezerwacja_istnieje <> 0 THEN
30         raise_application_error(-20103, 'Rezerwacja już istnieje!');
31     END IF;
32
33     SELECT wm.WOLNE_MIEJSCA INTO wolne_miejsca
34     FROM WYCIECZKI_MIEJSCA wm
35     WHERE wm.ID_WYCIECZKI = DODAJ_REZERWACJE.id_wycieczki;
36
```

```
37      IF wolne_miejsca = 0 THEN
38          raise_application_error(-20106, 'Nie można dodać rezerwacji
39           - brak wolnych miejsc!');
40      END IF;
41
42      INSERT INTO REZERWACJE (REZERWACJE.ID_WYCIECZKI, REZERWACJE.ID_OSOBY, STATUS)
43      VALUES (DODAJ_REZERWACJE.id_wycieczki, DODAJ_REZERWACJE.id_osoby, 'N');
44
45      --modyfikacja - dziennikowanie zmiany
46      SELECT ISEQ$$_78687.currval INTO DODAJ_REZERWACJE.id_nowej_rezerwacji FROM DUAL;
47
48      INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
49      VALUES (DODAJ_REZERWACJE.id_nowej_rezerwacji, CURRENT_DATE, 'N');
50  END;
```

Do poprawnego zapisu do tabeli REZERWACJE_LOG potrzebne było wyciągnięcie nowego numeru ID rezerwacji, identfikator sekwencji odpowiedzialnej za zwiększanie odpowiedniego pola tabeli REZERWACJE ustalono sprawdzając wartości currval wszystkich zadeklarowanych sekwencji i porównano z występującymi identyfikatorami we wszystkich tabelach.

5.2. Zmien_status_rezerwacji

```
1  CREATE OR REPLACE PROCEDURE ZMIEN_STATUS_REZERWACJI
2      (id_rezerwacji IN REZERWACJE.NR_REZERWACJI%TYPE, status IN REZERWACJE.STATUS%TYPE)
3  AS
4      id_rezerwacji_istnieje int;
5      wolne_miejsca int;
6      biezacy_status char(1);
7  BEGIN
8      IF ZMIEN_STATUS_REZERWACJI.status = 'N' THEN
9          raise_application_error(-20105, 'Nie można zmienić statusu
10         rezerwacji na nowy!');
11     END IF;
12
13     SELECT COUNT(*) INTO id_rezerwacji_istnieje
14     FROM WYCIECZKI_PRZYSZLE wp
15     JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
16     WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI.id_rezerwacji;
17
18     IF id_rezerwacji_istnieje = 0 THEN
19         raise_application_error(-20104, 'Rezerwacja nie istnieje
20         lub dotyczy przeszłej wycieczki!');
21     END IF;
22
23     SELECT r.STATUS INTO biezacy_status
24     FROM REZERWACJE r
25     WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI.id_rezerwacji;
26
```

```
27      IF biezacy_status = ZMIEN_STATUS_REZERWACJI.status THEN
28          raise_application_error(-20107, 'Nowy status jest identyczny jak bieżący!');
29      END IF;
30
31      IF biezacy_status = 'A' THEN
32          SELECT wm.WOLNE_MIEJSCA INTO ZMIEN_STATUS_REZERWACJI.wolne_miejsca
33          FROM WYCIECZKI_MIEJSCA wm
34          JOIN REZERWACJE r ON r.ID_WYCIECZKI = wm.ID_WYCIECZKI
35          WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI.id_rezerwacji;
36
37          IF wolne_miejsca = 0 THEN
38              raise_application_error(-20106, 'Brak wolnych miejsc,
39              nie można zmienić statusu tej rezerwacji z anulowanego!');
40          END IF;
41      END IF;
42
43      UPDATE REZERWACJE
44      SET STATUS = ZMIEN_STATUS_REZERWACJI.status
45      WHERE NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI.id_rezerwacji;
46
47      --modyfikacja - dziennikowanie zmiany
48      INSERT INTO REZERWACJE_LOG
49      (REZERWACJE_LOG.ID_REZERWACJI, REZERWACJE_LOG.DATA, REZERWACJE_LOG.STATUS)
50      VALUES (ZMIEN_STATUS_REZERWACJI.id_rezerwacji,
51          CURRENT_DATE, ZMIEN_STATUS_REZERWACJI.status);
52  END;
```

5.3. Zmien_liczbe_miejsc

```
1   CREATE OR REPLACE PROCEDURE ZMIEN_LICZBE_MIEJSC
2       (id_wycieczki IN WYCIECZKI.ID_WYCIECZKI%TYPE,
3        liczba_miejsc IN WYCIECZKI.LICZBA_MIEJSC%TYPE)
4   AS
5       id_wycieczki_istnieje int;
6       liczba_zajetych_miejsc int;
7   BEGIN
8       SELECT COUNT(*) INTO id_wycieczki_istnieje
9       FROM WYCIECZKI_PRZYSZLE wp
10      WHERE wp.ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC.id_wycieczki;
11
12      IF id_wycieczki_istnieje = 0 THEN
13          raise_application_error(-20107, 'Wycieczka o podanym ID
14          nie istnieje lub już się odbyła!');
15      END IF;
16
17      SELECT wm.LICZBA_MIEJSC - wm.WOLNE_MIEJSCA INTO liczba_zajetych_miejsc
18      FROM WYCIECZKI_MIEJSCA wm
19      WHERE wm.ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC.id_wycieczki;
```

```
20
21      IF ZMIEN_LICZBE_MIEJSC.liczba_miejsc < liczba_zajetych_miejsc THEN
22          raise_application_error(-20108, 'Nowa liczba miejsc jest mniejsza
23          niż liczba miejsc zajętych!');
24      END IF;
25
26      UPDATE WYCIECZKI
27      SET LICZBA_MIEJSC = ZMIEN_LICZBE_MIEJSC.liczba_miejsc
28      WHERE ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC.id_wycieczki;
29  END;
```

6. Tabela dziennikująca zmiany stanu rezerwacji

```
1  CREATE TABLE REZERWACJE_LOG
2  (
3      ID_LOG INT GENERATED ALWAYS AS IDENTITY NOT NULL,
4      ID_REZERWACJI INT,
5      DATA DATE,
6      STATUS CHAR(1),
7      CONSTRAINT REZERWACJE_LOG_PK PRIMARY KEY (ID_LOG) ENABLE
8  );
9
10 ALTER TABLE REZERWACJE_LOG
11 ADD CONSTRAINT REZERWACJE_LOG_FK
12 FOREIGN KEY (ID_REZERWACJI) REFERENCES REZERWACJE(NR_REZERWACJI) ENABLE;
13
14 ALTER TABLE REZERWACJE_LOG
15 ADD CONSTRAINT REZERWACJE_LOG_CHK CHECK (STATUS IN ('N', 'P', 'Z', 'A')) ENABLE;
```

Do tabeli dodano dodatkowe warunki integralnościowe – klucz obcy z tabeli REZERWACJE oraz ograniczony zakres wartości pola status.

7. Dodanie reduntantnego pola liczba_wolnych_miejsc

7.1. Zmiana w tabeli Rezerwacje

```
1 ALTER TABLE WYCIECZKI
2 ADD LICZBA_WOLNYCH_MIEJSC INT;
```

7.2. Modyfikacja widoków

Spośród utworzonych widoków, tylko dwa odwołują się do tabeli Rezerwacje i wymagają zmiany.

7.2.1. Wycieczki_miejsca_2

```
1  CREATE VIEW wycieczki_miejsca_2
2  AS
3      SELECT
4          w.ID_WYCIECZKI,
5          w.NAZWA,
6          w.KRAJ,
7          w.DATA,
8          w.LICZBA_MIEJSC,
9          w.LICZBA_WOLNYCH_MIEJSC
10     FROM WYCIECZKI w;
```

7.2.2. Dostepne_wycieczki_2

```
1 CREATE VIEW dostepne_wycieczki_2
2 AS
3     SELECT * FROM wycieczki_miejsca_2
4     WHERE LICZBA_WOLNYCH_MIEJSC > 0
5     AND DATA > CURRENT_DATE;
```

W tym przypadku zmiana jest bardzo niewielka – inna jest tylko nazwa atrybutu określającego liczbę wolnych miejsc w widoku wycieczki_miejsca_2.

7.3. Procedura Przelicz

```
CREATE OR REPLACE PROCEDURE PRZELICZ
AS
BEGIN
    UPDATE WYCIECZKI w
    SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_MIEJSC -
        (SELECT COUNT(*) FROM REZERWACJE r
        WHERE r.ID_WYCIECZKI = w.ID_WYCIECZKI
        AND r.STATUS <> 'A');
END;
```

7.4. Modyfikacja funkcji pobierających dane

Wśród zdefiniowanych funkcji zmiany wymaga jedynie funkcja Dostepne_wycieczki (tylko ona zwraca tabelę zawierającą informację o miejscach).

7.5. Dostepne_wycieczki_2

```
CREATE OR REPLACE FUNCTION DOSTEPNE_WYCIECZKI_FUN_2
    (kraj IN WYCIECZKI.KRAJ%TYPE, data_od IN DATE, data_do IN DATE)
RETURN dostepne_wycieczki_table
AS
    dostepne_wycieczki_t dostepne_wycieczki_table;
BEGIN
    IF data_od > data_do THEN
        raise_application_error(-20102,
        'Podano niepoprawne daty: data końca przedziału
        jest wcześniejsza niż jego początek');
    END IF;

    SELECT dostepne_wycieczki_row(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.DATA,
        w.LICZBA_MIEJSC,
        w.LICZBA_WOLNYCH_MIEJSC)
    BULK COLLECT INTO dostepne_wycieczki_t
    FROM WYCIECZKI w
    WHERE w.KRAJ = DOSTEPNE_WYCIECZKI_FUN_2.kraj
    AND w.DATA BETWEEN DOSTEPNE_WYCIECZKI_FUN_2.data_od
    AND DOSTEPNE_WYCIECZKI_FUN_2.data_do
    AND w.LICZBA_WOLNYCH_MIEJSC > 0;

    RETURN dostepne_wycieczki_t;
END;
```

7.6. Modyfikacja procedur modyfikujących dane

7.6.1. Dodaj_rezerwacje_2

```
CREATE OR REPLACE PROCEDURE DODAJ_REZERWACJE_2
    (id_wycieczki IN WYCIECZKI.ID_WYCIECZKI%TYPE, id_osoby IN OSOBY.ID_OSOBY%TYPE)
AS
    id_osoby_istnieje int;
    id_wycieczki_istnieje int;
    rezerwacja_istnieje int;
    wolne_miejsca int;
    id_nowej_rezerwacji int;
BEGIN
    SELECT COUNT(*) INTO id_osoby_istnieje
    FROM OSOBY
    WHERE OSOBY.ID_OSOBY = DODAJ_REZERWACJE_2.id_osoby;
    IF id_osoby_istnieje = 0 THEN
        raise_application_error(-20101, 'Osoba o podanym ID nie istnieje!');
    END IF;

    SELECT COUNT(*) INTO id_wycieczki_istnieje
    FROM WYCIECZKI_PRZYSZLE
    WHERE WYCIECZKI_PRZYSZLE.ID_WYCIECZKI = DODAJ_REZERWACJE_2.id_wycieczki;
    IF id_wycieczki_istnieje = 0 THEN
        raise_application_error(-20100, 'Wycieczka o podanym ID
        nie istnieje lub już się odbyła!');
    END IF;

    SELECT COUNT(*) INTO rezerwacja_istnieje
    FROM REZERWACJE r
    WHERE r.ID_WYCIECZKI = DODAJ_REZERWACJE_2.id_wycieczki
    AND r.ID_OSOBY = DODAJ_REZERWACJE_2.id_osoby;
    IF rezerwacja_istnieje <> 0 THEN
        raise_application_error(-20103, 'Rezerwacja już istnieje!');
    END IF;

    SELECT LICZBA_WOLNYCH_MIEJSC INTO wolne_miejsca
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = DODAJ_REZERWACJE_2.id_wycieczki;

    IF wolne_miejsca = 0 THEN
        raise_application_error(-20106, 'Nie można dodać rezerwacji
        - brak wolnych miejsc!');
    END IF;

    INSERT INTO REZERWACJE (REZERWACJE.ID_WYCIECZKI, REZERWACJE.ID_OSOBY, STATUS)
    VALUES (DODAJ_REZERWACJE_2.id_wycieczki, DODAJ_REZERWACJE_2.id_osoby, 'N');

    UPDATE WYCIECZKI
```

```
46      SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1
47      WHERE ID_WYCIECZKI = DODAJ_REZERWACJE_2.id_wycieczki;
48
49      SELECT ISEQ$$_78687.currval INTO DODAJ_REZERWACJE_2.id_nowej_rezerwacji FROM DUAL;
50
51      INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
52      VALUES (DODAJ_REZERWACJE_2.id_nowej_rezerwacji, CURRENT_DATE, 'N');
53  END;
```

W tym przypadku modyfikacja polega na obniżeniu liczby wolnych miejsc dla danej wycieczki o 1, oraz skorzystaniu z reduntantnego pola zamiast widoku wycieczki_miejsca przy decyzji, czy jest wystarczająco dużo wolnych miejsc.

7.6.2. Zmien_status_rezerwacji_2

```
1   CREATE OR REPLACE PROCEDURE ZMIEN_STATUS_REZERWACJI_2
2       (id_rezerwacji IN REZERWACJE.NR_REZERWACJI%TYPE, status IN REZERWACJE.STATUS%TYPE)
3   AS
4       id_rezerwacji_istnieje int;
5       wolne_miejsca int;
6       biezacy_status char(1);
7       zmiana_wolnych_miejsc int;
8   BEGIN
9       zmiana_wolnych_miejsc := 0;
10
11      IF ZMIEN_STATUS_REZERWACJI_2.status = 'N' THEN
12          raise_application_error(-20105, 'Nie można zmienić statusu
13          rezerwacji na nowy!');
14      END IF;
15
16      SELECT COUNT(*) INTO id_rezerwacji_istnieje
17      FROM WYCIECZKI_PRZYSZLE wp
18      JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
19      WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI_2.id_rezerwacji;
20
21      IF id_rezerwacji_istnieje = 0 THEN
22          raise_application_error(-20104, 'Rezerwacja nie istnieje
23          lub dotyczy przeszłej wycieczki!');
24      END IF;
25
26      SELECT r.STATUS INTO biezacy_status
27      FROM REZERWACJE r
28      WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI_2.id_rezerwacji;
29
30      IF biezacy_status = ZMIEN_STATUS_REZERWACJI_2.status THEN
31          raise_application_error(-20107, 'Nowy status jest identyczny jak bieżący!');
32      END IF;
33
34      IF biezacy_status = 'A' THEN
```

```
35          SELECT LICZBA_WOLNYCH_MIEJSC INTO ZMIEN_STATUS_REZERWACJI_2.wolne_miejsca
36          FROM WYCIECZKI w
37          JOIN REZERWACJE r ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
38          WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI_2.id_rezerwacji;
39
40          IF wolne_miejsca = 0 THEN
41              raise_application_error(-20106, 'Brak wolnych miejsc,
42              nie można zmienić statusu tej rezerwacji z anulowanego!');
43          END IF;
44
45          zmiana_wolnych_miejsc := -1;
46
47      ELSIF ZMIEN_STATUS_REZERWACJI_2.status = 'A' THEN
48          zmiana_wolnych_miejsc := 1;
49
50      END IF;
51
52      UPDATE REZERWACJE
53      SET STATUS = ZMIEN_STATUS_REZERWACJI_2.status
54      WHERE NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI_2.id_rezerwacji;
55
56      IF zmiana_wolnych_miejsc <> 0 THEN
57          UPDATE WYCIECZKI
58          SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + zmiana_wolnych_miejsc
59          WHERE ID_WYCIECZKI = (SELECT ID_WYCIECZKI FROM REZERWACJE WHERE NR_REZERWACJI =
60              ZMIEN_STATUS_REZERWACJI_2.id_rezerwacji);
61      END IF;
62
63      INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
64      VALUES (ZMIEN_STATUS_REZERWACJI_2.id_rezerwacji,
65          CURRENT_DATE, ZMIEN_STATUS_REZERWACJI_2.status);
66  END;
```

Do funkcji dodano zmienną określającą, czy należy zmienić liczbę miejsc dla danej wycieczki (zwiększyć dla anulowania i zmniejszyć dla przejścia z anulowanej rezerwacji na inny status), a na końcu w przypadku jej niezerowej wartości aktualizuje się tabelę Wycieczki.

7.6.3. Zmien_liczbe_miejsc_2

```
1  CREATE OR REPLACE PROCEDURE ZMIEN_LICZBE_MIEJSC_2
2      (id_wycieczki IN WYCIECZKI.ID_WYCIECZKI%TYPE,
3       liczba_miejsc IN WYCIECZKI.LICZBA_MIEJSC%TYPE)
4  AS
5      id_wycieczki_istnieje int;
6      liczba_zajetych_miejsc int;
7  BEGIN
8      SELECT COUNT(*) INTO id_wycieczki_istnieje
9      FROM WYCIECZKI_PRZYSZLE wp
10     WHERE wp.ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC_2.id_wycieczki;
```

```
11
12      IF id_wycieczki_istnieje = 0 THEN
13          raise_application_error(-20107, 'Wycieczka o podanym ID
14          nie istnieje lub już się odbyła!');
15      END IF;
16
17      SELECT w.LICZBA_MIEJSC - w.LICZBA_WOLNYCH_MIEJSC INTO liczba_zajetych_miejsc
18      FROM WYCIECZKI w
19      WHERE w.ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC_2.id_wycieczki;
20
21      IF ZMIEN_LICZBE_MIEJSC_2.liczba_miejsc < liczba_zajetych_miejsc THEN
22          raise_application_error(-20108, 'Nowa liczba miejsc jest mniejsza
23          niż liczba miejsc zajętych!');
24      END IF;
25
26      UPDATE WYCIECZKI
27      SET LICZBA_MIEJSC = ZMIEN_LICZBE_MIEJSC_2.liczba_miejsc,
28          LICZBA_WOLNYCH_MIEJSC =
29              ZMIEN_LICZBE_MIEJSC_2.liczba_miejsc - liczba_zajetych_miejsc
30      WHERE ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC_2.id_wycieczki;
31  END;
```

Różnica w stosunku do pierwotnej procedury polega na wykorzystaniu redundantnego pola zamiast widoku wycieczki_miejsca przy obliczaniu zajętych miejsc oraz aktualizacji liczby wolnych miejsc po wykonaniu zmiany.

8. Zastosowanie triggerów

8.1. Utworzone triggery

Poniżej przedstawiono polecenia użyte do utworzenia triggerów aktualizujących zarówno tabelę REZERWACJE_LOG jak i pole określające liczbę wolnych miejsc.

8.1.1. Dodana_rezerwacja

```
1  CREATE OR REPLACE TRIGGER DODANA_REZERWACJA
2  AFTER INSERT
3  ON REZERWACJE
4  FOR EACH ROW
5  BEGIN
6      INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
7      VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, 'N');
8
9      UPDATE WYCIECZKI w
10     SET w.LICZBA_WOLNYCH_MIEJSC = w.LICZBA_WOLNYCH_MIEJSC - 1
11     WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
12 END;
```

8.1.2. Zmiana_statusu_rezerwacji

```
1  CREATE OR REPLACE TRIGGER ZMIANA_STATUSU_REZERWACJI
2  AFTER UPDATE
3  ON REZERWACJE
4  FOR EACH ROW
5  DECLARE
6      zmiana_wolnych_miejsc int;
7  BEGIN
8      INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
9      VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);
10
11     zmiana_wolnych_miejsc := 0;
12
13     IF :NEW.STATUS <> 'A' THEN
14         zmiana_wolnych_miejsc := -1;
15     ELSIF :NEW.STATUS = 'A' THEN
16         zmiana_wolnych_miejsc := 1;
17     END IF;
18
19     IF zmiana_wolnych_miejsc <> 0 THEN
```

```
20          UPDATE WYCIECZKI
21          SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + zmiana_wolnych_miejsc
22          WHERE ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
23      END IF;
24  END;
```

Logika dotycząca określania czy należy zmienić liczbę wolnych miejsc w tabeli Wycieczki jest identyczna jak dla funkcji Zmien_status_rezerwacji_2.

8.1.3. Blokada_usuwania_rezerwacji

```
1  CREATE OR REPLACE TRIGGER BLOKADA_USUWANIA_REZERWACJI
2  BEFORE DELETE
3  ON REZERWACJE
4  FOR EACH ROW
5  BEGIN
6      raise_application_error(-20200, 'Niemożliwe jest usunięcie rezerwacji!');
7  END;
```

8.1.4. Zmiana_liczby_miejsc

```
1  CREATE OR REPLACE TRIGGER ZMIANA_LICZBY_MIEJSC
2  BEFORE UPDATE
3  ON WYCIECZKI
4  FOR EACH ROW
5  BEGIN
6      IF :NEW.LICZBA_MIEJSC <> :OLD.LICZBA_MIEJSC THEN
7          SELECT :OLD.LICZBA_WOLNYCH_MIEJSC +
8              (:NEW.LICZBA_MIEJSC - :OLD.LICZBA_MIEJSC)
9          INTO :NEW.LICZBA_WOLNYCH_MIEJSC FROM DUAL;
10     END IF;
11 END;
```

Sprawdzanie czy nowa całkowita liczba miejsc jest różna od starej jest potrzebne, aby trigger zmieniający liczbę wolnych miejsc przy zmianie statusu rezerwacji działał poprawnie, bez tego warunku nowa wartość liczby wolnych miejsc była zastępowana starą (gdyż wtedy całkowite liczby miejsc się między sobą nie różniły).

8.2. Modyfikacja procedur modyfikujących dane

8.2.1. Dodaj_rezerwacje_3

```
1  CREATE OR REPLACE PROCEDURE DODAJ_REZERWACJE_3
2      (id_wycieczki IN WYCIECZKI.ID_WYCIECZKI%TYPE, id_osoby IN OSOBY.ID_OSOBY%TYPE)
3  AS
4      id_osoby_istnieje int;
5      id_wycieczki_istnieje int;
```

```
6       rezerwacja_istnieje int;
7       wolne_miejsca int;
8   BEGIN
9       SELECT COUNT(*) INTO id_osoby_istnieje
10      FROM OSOBY
11      WHERE OSOBY.ID_OSOBY = DODAJ_REZERWACJE_3.id_osoby;
12      IF id_osoby_istnieje = 0 THEN
13          raise_application_error(-20101, 'Osoba o podanym ID nie istnieje!');
14      END IF;
15
16      SELECT COUNT(*) INTO id_wycieczki_istnieje
17      FROM WYCIECZKI
18      WHERE WYCIECZKI.ID_WYCIECZKI = DODAJ_REZERWACJE_3.id_wycieczki
19      AND WYCIECZKI.DATA > CURRENT_DATE;
20      IF id_wycieczki_istnieje = 0 THEN
21          raise_application_error(-20100, 'Wycieczka o podanym ID
22          nie istnieje lub już się odbyła!');
23      END IF;
24
25      SELECT COUNT(*) INTO rezerwacja_istnieje
26      FROM REZERWACJE r
27      WHERE r.ID_WYCIECZKI = DODAJ_REZERWACJE_3.id_wycieczki
28      AND r.ID_OSOBY = DODAJ_REZERWACJE_3.id_osoby;
29      IF rezerwacja_istnieje <> 0 THEN
30          raise_application_error(-20103, 'Rezerwacja już istnieje!');
31      END IF;
32
33      SELECT LICZBA_WOLNYCH_MIEJSC INTO wolne_miejsca
34      FROM WYCIECZKI w
35      WHERE w.ID_WYCIECZKI = DODAJ_REZERWACJE_3.id_wycieczki;
36
37      IF wolne_miejsca = 0 THEN
38          raise_application_error(-20106, 'Nie można dodać rezerwacji
39          - brak wolnych miejsc!');
40      END IF;
41
42      INSERT INTO REZERWACJE (REZERWACJE.ID_WYCIECZKI, REZERWACJE.ID_OSOBY, STATUS)
43      VALUES (DODAJ_REZERWACJE_3.id_wycieczki, DODAJ_REZERWACJE_3.id_osoby, 'N');
44  END;
```

Poza brakiem wykonywania operacji dodawania danych do tabeli Rezerwacje_log oraz wykorzystaniem redundantnego pola z tabeli Wycieczki zamiast widoku wycieczki_miejsca zmodyfikowana funkcja nie różni się od pierwszej swojej wersji.

8.2.2. Zmien_status_rezerwacji_3

```
1   CREATE OR REPLACE PROCEDURE ZMIEN_STATUS_REZERWACJI_3
2       (id_rezerwacji IN REZERWACJE.NR_REZERWACJI%TYPE, status IN REZERWACJE.STATUS%TYPE)
3   AS
```

```
4       id_rezerwacji_istnieje int;
5       wolne_miejsca int;
6       biezacy_status char(1);
7   BEGIN
8       IF ZMIEN_STATUS_REZERWACJI_3.status = 'N' THEN
9           raise_application_error(-20105, 'Nie można zmienić statusu
10          rezerwacji na nowy!');
11      END IF;
12
13      SELECT COUNT(*) INTO id_rezerwacji_istnieje
14      FROM WYCIECZKI_PRZYSZLE wp
15      JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
16      WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI_3.id_rezerwacji;
17
18      IF id_rezerwacji_istnieje = 0 THEN
19          raise_application_error(-20104, 'Rezerwacja nie istnieje
20          lub dotyczy przeszłej wycieczki!');
21      END IF;
22
23      SELECT r.STATUS INTO biezacy_status
24      FROM REZERWACJE r
25      WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI_3.id_rezerwacji;
26
27      IF biezacy_status = ZMIEN_STATUS_REZERWACJI_3.status THEN
28          raise_application_error(-20107, 'Nowy status jest identyczny jak bieżący!');
29      END IF;
30
31      IF biezacy_status = 'A' THEN
32          SELECT w.LICZBA_WOLNYCH_MIEJSC INTO ZMIEN_STATUS_REZERWACJI_3.wolne_miejsca
33          FROM WYCIECZKI w
34          JOIN REZERWACJE r ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
35          WHERE r.NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI_3.id_rezerwacji;
36
37          IF wolne_miejsca = 0 THEN
38              raise_application_error(-20106, 'Brak wolnych miejsc,
39              nie można zmienić statusu tej rezerwacji z anulowanego!');
40          END IF;
41      END IF;
42
43      UPDATE REZERWACJE
44      SET STATUS = ZMIEN_STATUS_REZERWACJI_3.status
45      WHERE NR_REZERWACJI = ZMIEN_STATUS_REZERWACJI_3.id_rezerwacji;
46  END;
```

Różnice w stosunku do pierwszej wersji są podobne jak wyżej.

8.2.3. Zmien_liczbe_miejsc_3

```
1   CREATE OR REPLACE PROCEDURE ZMIEN_LICZBE_MIEJSC_3
```

```
2       (id_wycieczki IN WYCIECZKI.ID_WYCIECZKI%TYPE,
3        liczba_miejsc IN WYCIECZKI.LICZBA_MIEJSC%TYPE)
4 AS
5       id_wycieczki_istnieje int;
6       liczba_zajetych_miejsc int;
7 BEGIN
8       SELECT COUNT(*) INTO id_wycieczki_istnieje
9       FROM WYCIECZKI_PRZYSZLE wp
10      WHERE wp.ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC_3.id_wycieczki;
11
12      IF id_wycieczki_istnieje = 0 THEN
13          raise_application_error(-20107, 'Wycieczka o podanym ID
14          nie istnieje lub już się odbyła!');
15      END IF;
16
17      SELECT w.LICZBA_MIEJSC - w.LICZBA_WOLNYCH_MIEJSC INTO liczba_zajetych_miejsc
18      FROM WYCIECZKI w
19      WHERE w.ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC_3.id_wycieczki;
20
21      IF ZMIEN_LICZBE_MIEJSC_3.liczba_miejsc < liczba_zajetych_miejsc THEN
22          raise_application_error(-20108, 'Nowa liczba miejsc jest mniejsza
23          niż liczba miejsc zajętych!');
24      END IF;
25
26      UPDATE WYCIECZKI
27      SET LICZBA_MIEJSC = ZMIEN_LICZBE_MIEJSC_3.liczba_miejsc
28      WHERE ID_WYCIECZKI = ZMIEN_LICZBE_MIEJSC_3.id_wycieczki;
29 END;
```

Tutaj w porównaniu z pierwszą wersją zmienia się jedynie wykorzystanie redundantnego pola z tabeli Wycieczki zamiast widoku wycieczki_miejsca do określenia liczby zajętych miejsc.

Ogólnie, zastosowanie triggerów, pozwoliło przenieść logikę obsługi mechanizmu logowania oraz aktualizacji redundantnego pola tabeli Wycieczki poza procedury modyfikujące dane.