Raport z wykonania ćwiczenia Oracle PL/SQL

Jakub Płotnikowski

Październik 2019r.

Spis treści

1 Tabele - dane podane na początku w skrypcie

Wykaz tabel w bazie danych.

1.1 Definicje tabel oraz początkowych warunków integralnościowych

```
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
);


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
);


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
);


ALTER TABLE REZERWACJE
    ADD CONSTRAINT REZERWACJE_FK1
        FOREIGN KEY (ID_OSOBY)
            REFERENCES OSOBY (ID_OSOBY)
                ENABLE;

ALTER TABLE REZERWACJE
    ADD CONSTRAINT REZERWACJE_FK2
        FOREIGN KEY (ID_WYCIECZKI)
            REFERENCES WYCIECZKI (ID_WYCIECZKI)
                ENABLE;

ALTER TABLE REZERWACJE
    ADD CONSTRAINT REZERWACJE_CHK1 CHECK
        (status IN ('N', 'P', 'Z', 'A')) ENABLE;
```

2 Widoki

2.1 Widok WYCIECZKI_OSOBY

```
CREATE OR REPLACE 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;
```

2.2 Widok WYCIECZKI_OSOBY_POTWIERDZONE

```
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';
```

2.3 Widok WYCIECZKI_OSOBY_PRZYSZLE

```
CREATE OR REPLACE 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;
```

2.4 Widok WYCIECZKI_MIEJSCA

```
CREATE OR REPLACE VIEW WYCIECZKI_MIEJSCA AS
SELECT w.ID_WYCIECZKI,
       w.NAZWA,
       w.KRAJ,
       w.DATA,
       w.LICZBA_MIEJSC,
       w.LICZBA_MIEJSC
           - (SELECT count(*)
              FROM REZERWACJE
              WHERE REZERWACJE.ID_WYCIECZKI = w.ID_WYCIECZKI
                AND status <> 'A') AS liczba_wolnych_miejsc
FROM WYCIECZKI w
```

2.5 Widok DOSTEPNE_WYCIECZKI

```
CREATE OR REPLACE VIEW DOSTEPNE_WYCIECZKI_VIEW AS
SELECT w.ID_WYCIECZKI,
       w.NAZWA,
       w.KRAJ,
       w.DATA,
       w.LICZBA_MIEJSC,
       w.LICZBA_WOLNYCH_MIEJSC
FROM WYCIECZKI_MIEJSCA w
WHERE w.DATA > CURRENT_DATE
  AND w.LICZBA_WOLNYCH_MIEJSC > 0
```

2.6 Widok REZERWACJE_DO_ANULOWANIA

```
CREATE OR REPLACE VIEW REZERWACJE_DO_ANULOWANIA AS
SELECT r.NR_REZERWACJI,
       r.ID_WYCIECZKI,
       r.ID_OSOBY,
       r.STATUS
FROM REZERWACJE r
        JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI
WHERE w.DATA - CURRENT_DATE <= 7
  AND r.STATUS = 'N';
```

3 Funkcje pobierające dane

3.1 Definicje typów użytych przy funkcjach

```
CREATE OR REPLACE TYPE osoby_wycieczki_type AS OBJECT
(
    imie            varchar(50),
    nazwisko        varchar(50),
    "nazwa wycieczki" varchar(200),
    "data wycieczki" date,
    kraj            varchar(50),
    status          char(1)
);

CREATE OR REPLACE TYPE osoby_wycieczki_table IS TABLE OF osoby_wycieczki_type;


CREATE OR REPLACE TYPE wycieczki_type AS OBJECT
(
    "id wycieczki"  number,
    "nazwa wycieczki" varchar2(100),
    kraj            varchar2(50),
    opis            varchar2(200),
    "liczba miejsc" number,
    "data"          date
);

CREATE OR REPLACE TYPE wycieczki_table IS TABLE OF wycieczki_type;
```

3.2 Funkcja UCZESTNICY_WYCIECZKI

```
CREATE OR REPLACE FUNCTION uczestnicy_wycieczki(id_wycieczki_param INT)
    RETURN osoby_wycieczki_table AS
    result_table  osoby_wycieczki_table;
    ilosc_wycieczek INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO ilosc_wycieczek
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = id_wycieczki_param;

    IF ilosc_wycieczek = 0 THEN
        raise_application_error(-20006, 'Nie znaleziono wycieczki o podanym id');
    END IF;

    SELECT osoby_wycieczki_type(o.IMIE, o.NAZWISKO, w.NAZWA, w.DATA, w.KRAJ, r.STATUS)
        BULK COLLECT
    INTO result_table
    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_param
      AND r.STATUS <> 'A';

    RETURN result_table;
END;
```

3.3 Funkcja REZERWACJE_OSOBY

```
CREATE OR REPLACE FUNCTION rezerwacje_osoby(id_osoby_param NUMBER)
    RETURN osoby_wycieczki_table AS
    result_table  osoby_wycieczki_table;
    ilosc_wycieczek INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO ilosc_wycieczek
    FROM OSOBY o
    WHERE o.ID_OSOBY = id_osoby_param;

    IF ilosc_wycieczek = 0 THEN
        raise_application_error(-20003, 'Nie znaleziono osoby o podanym id');
    END IF;

    SELECT osoby_wycieczki_type(o.IMIE, o.NAZWISKO, w.NAZWA, w.DATA, w.KRAJ, r.STATUS)
        BULK COLLECT
    INTO result_table
    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_param
      AND r.STATUS <> 'A';

    RETURN result_table;
END;
```

3.4 Funkcja PRZYSZLE_REZERWACJE_OSOBY

```
CREATE OR REPLACE FUNCTION przyszle_rezerwacje_osoby(id_osoby_param NUMBER)
    RETURN osoby_wycieczki_table AS
    result_table  osoby_wycieczki_table;
    ilosc_wycieczek INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO ilosc_wycieczek
    FROM OSOBY o
    WHERE o.ID_OSOBY = id_osoby_param;

    IF ilosc_wycieczek = 0 THEN
        raise_application_error(-20001, 'Nie znaleziono osoby o podanym id');
    END IF;

    SELECT osoby_wycieczki_type(o.IMIE, o.NAZWISKO, w.NAZWA, w.DATA, w.KRAJ, r.STATUS)
        BULK COLLECT
    INTO result_table
    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_param
      AND r.STATUS <> 'A'
      AND w.DATA > CURRENT_DATE;

    RETURN result_table;
END;
```

3.5 Funkcja DOSTEPNE_WYCIECZKI

```
CREATE OR REPLACE FUNCTION dostepne_wycieczki_function(kraj_param VARCHAR2, data_od
    DATE, data_do DATE)
    RETURN wycieczki_table AS
    result_table wycieczki_table;
BEGIN
    IF data_do < data_od THEN
        raise_application_error(-20020,
                                'Niepoprawny przedzial czasu - data z parametru data_do
                                    jest wczesniej niz data_od');
    END IF;

    SELECT wycieczki_type(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.OPIS, w.LICZBA_MIEJSC,
        w.DATA)
        BULK COLLECT
    INTO result_table
    FROM WYCIECZKI w
    WHERE w.KRAJ = kraj_param
      AND w.DATA >= data_od
      AND w.DATA <= data_do
      AND w.LICZBA_MIEJSC >
          (SELECT COUNT(*) FROM REZERWACJE r where r.ID_WYCIECZKI = w.ID_WYCIECZKI AND
              r.STATUS <> 'A');

    RETURN result_table;
END;
```

4 Funkcje modyfikujące dane - wersje pierwsze

4.1 Funkcja DODAJ_REZERWACJE

```
CREATE OR REPLACE PROCEDURE dodaj_rezerwacje(id_wycieczki_param NUMBER,
    id_osoby_param NUMBER)
AS
    counter INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO counter
    FROM DOSTEPNE_WYCIECZKI_VIEW v
    WHERE v.ID_WYCIECZKI = id_wycieczki_param;

    IF counter = 0 THEN
        raise_application_error(-20011, 'Nie znaleziono wycieczki o podanym id');
    END IF;

    SELECT COUNT(*)
    INTO counter
    FROM OSOBY o
    WHERE o.ID_OSOBY = id_osoby_param;

    IF counter = 0 THEN
        raise_application_error(-20012, 'Nie znaleziono osoby o podanym id');
    END IF;

    SELECT COUNT(*)
    INTO COUNTER
    FROM REZERWACJE r
    WHERE r.ID_OSOBY = id_osoby_param
      AND r.ID_WYCIECZKI = id_wycieczki_param;

    IF counter > 0 THEN
        raise_application_error(-20013, 'Rezerwacja z danym id wycieczki i id osoby
            juz istnieje');
    END IF;

    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
    VALUES (id_wycieczki_param, id_osoby_param, 'N');
END;
```

4.2 Funkcja ZMIEN_STATUS_REZERWACJI

```
CREATE OR REPLACE PROCEDURE zmien_status_rezerwacji(id_rezerwacji_param NUMBER,
    new_status CHAR) AS
    old_status CHAR(1);
    counter   INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO counter
    FROM WYCIECZKI_PRZYSZLE wp
            JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
    WHERE r.NR_REZERWACJI = id_rezerwacji_param;

    IF counter = 0 THEN
        raise_application_error(-20015,
                                'Nie mozna zmienic statusu rezerwacji dla wycieczek,
                                    ktore nie sa w przyszlosci');
    END IF;

    SELECT COUNT(*)
    INTO counter
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = id_rezerwacji_param;

    IF counter = 0 THEN
        raise_application_error(-20016,
                                'Nie istnieje rezerwacja o podanym numerze');
    END IF;

    SELECT STATUS
    INTO old_status
    FROM REZERWACJE
    WHERE NR_REZERWACJI = id_rezerwacji_param;

    CASE
        WHEN old_status IS NULL
            THEN
                raise_application_error(-20016,
                                        'Nie istnieje rezerwacja o podanym numerze');

        WHEN old_status = 'A'
            THEN
                SELECT COUNT(*)
                INTO counter
                FROM DOSTEPNE_WYCIECZKI_VIEW v
                        JOIN REZERWACJE r on R.ID_WYCIECZKI = v.ID_WYCIECZKI
                WHERE r.NR_REZERWACJI = id_rezerwacji_param;

                IF counter = 0 THEN
                    raise_application_error(-20018,
                                            'Nie ma wystarczajacej ilosci miejsc, aby
                                                przywrocic anulowana rezerwacje o podanym
                                                id');
                END IF;
```

```
        WHEN new_status = 'N'
            THEN raise_application_error(-20017,
                                    'Rezerwacja, ktora juz istnieje nie moze zmienic
                                        statusu na nowa. Podano bledny argument');

        ELSE NULL;
        END CASE;

    UPDATE REZERWACJE
    SET STATUS = new_status
    WHERE NR_REZERWACJI = id_rezerwacji_param;
END;
```

4.3 Funkcja ZMIEN_LICZBE_MIEJSC

```
CREATE OR REPLACE PROCEDURE zmien_liczbe_miejsc(id_wycieczki_param NUMBER,
    liczba_nowych_miejsc_param NUMBER)
AS
    counter INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO counter
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = id_wycieczki_param;

    IF counter = 0 THEN
        raise_application_error(-20011, 'Nie znaleziono wycieczki o podanym id');
    END IF;

    SELECT wm.LICZBA_MIEJSC - wm.LICZBA_WOLNYCH_MIEJSC
    INTO counter
    FROM WYCIECZKI_MIEJSCA wm
    WHERE wm.ID_WYCIECZKI = id_wycieczki_param;

    IF liczba_nowych_miejsc_param < counter OR liczba_nowych_miejsc_param < 0 THEN
        raise_application_error(-20030, 'Podana liczba nowych miejsc ma zbyt mala
            wartosc');
    END IF;

    UPDATE WYCIECZKI
    SET LICZBA_MIEJSC = liczba_nowych_miejsc_param
    WHERE ID_WYCIECZKI = id_wycieczki_param;
END;
```

5 Definicja tabeli dziennikującej

5.1 Tabela REZERWACJE_LOG

```
CREATE TABLE REZERWACJE_LOG
(
    ID           INT GENERATED ALWAYS AS IDENTITY NOT NULL,
    ID_REZERWACJI INT,
    DATA         DATE,
    STATUS       CHAR(1),
    CONSTRAINT REZERWACJE_LOG_PK PRIMARY KEY (ID) ENABLE
);
```

6 Widoki z użyciem dodanej kolumny LICZBA_WOLNYCH_MIEJSC w tabeli WYCIECZKI

6.1 Widok DOSTEPNE_WYCIECZKI2

```
CREATE OR REPLACE VIEW DOSTEPNE_WYCIECZKI_VIEW2 AS
SELECT w.ID_WYCIECZKI,
       w.NAZWA,
       w.KRAJ,
       w.DATA,
       w.LICZBA_MIEJSC,
       w.LICZBA_WOLNYCH_MIEJSC
FROM WYCIECZKI w
WHERE w.DATA > CURRENT_DATE
  AND w.LICZBA_WOLNYCH_MIEJSC > 0
```

6.2 Widok WYCIECZKI_MIEJSCA2

```
CREATE OR REPLACE VIEW WYCIECZKI_MIEJSCA2
AS
SELECT w.ID_WYCIECZKI,
       w.kraj,
       w.data,
       w.nazwa,
       w.liczba_miejsc,
       w.LICZBA_WOLNYCH_MIEJSC
FROM wycieczki w;
```

7 Procedura przelicz

```
CREATE OR REPLACE PROCEDURE przelicz 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 <> 'A');
END;
```

8 Funkcje pobierające dane z użyciem dodanej kolumny LICZBA_WOLNYCH_MIEJSC

8.1 Funkcja DOSTEPNE_WYCIECZKI2

```
CREATE OR REPLACE FUNCTION dostepne_wycieczki_function2(kraj_param VARCHAR2, data_od
    DATE, data_do DATE)
    RETURN wycieczki_table AS
    result_table wycieczki_table;
BEGIN
    IF data_do < data_od THEN
        raise_application_error(-20020,
                                'Niepoprawny przedzial czasu - data z parametru data_do
                                    jest wczesniej niz data_od');
    END IF;

    SELECT wycieczki_type(w.ID_WYCIECZKI, w.NAZWA, w.KRAJ, w.OPIS, w.LICZBA_MIEJSC,
        w.DATA)
        BULK COLLECT
    INTO result_table
    FROM WYCIECZKI w
    WHERE w.KRAJ = kraj_param
      AND w.DATA >= data_od
      AND w.DATA <= data_do
      AND w.LICZBA_WOLNYCH_MIEJSC > 0;

    RETURN result_table;
END;
```

9 Funkcje modyfikujące dane z użyciem tabeli dziennikującej oraz z dodaną kolumną LICZBA_WOLNYCH_MIEJSC w tabeli WYCIECZKI

9.1 Wyszukanie sekwencji dla poszczególnych obiektów

```
SELECT object_id,
      object_name
FROM user_objects;
```

Sekwencja dla PK tabeli REZERWACJE jest u mnie w bazie nastepująca:

```
"ISEQ$$_79074"
```

9.2 Funkcja DODAJ_REZERWACJE2

```
CREATE OR REPLACE PROCEDURE dodaj_rezerwacje2(id_wycieczki_param NUMBER,
    id_osoby_param NUMBER)
AS
    counter           INTEGER;
    new_reservation_id INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO counter
    FROM DOSTEPNE_WYCIECZKI_VIEW v
    WHERE v.ID_WYCIECZKI = id_wycieczki_param;

    IF counter = 0 THEN
        raise_application_error(-20011, ''Nie znaleziono wycieczki o podanym id'');
    END IF;

    SELECT COUNT(*)
    INTO counter
    FROM OSOBY o
    WHERE o.ID_OSOBY = id_osoby_param;

    IF counter = 0 THEN
        raise_application_error(-20012, ''Nie znaleziono osoby o podanym id'');
    END IF;

    SELECT COUNT(*)
    INTO COUNTER
    FROM REZERWACJE r
    WHERE r.ID_OSOBY = id_osoby_param
      AND r.ID_WYCIECZKI = id_wycieczki_param;

    IF counter > 0 THEN
        raise_application_error(-20013, ''Rezerwacja z danym id wycieczki i id osoby
            juz istnieje'');
    END IF;
```

```
    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
    VALUES (id_wycieczki_param, id_osoby_param, ''N'');

    UPDATE WYCIECZKI
    SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1
    WHERE ID_WYCIECZKI = id_wycieczki_param;

    SELECT "ISEQ$$_79074".currval INTO new_reservation_id FROM dual;

    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (new_reservation_id, CURRENT_DATE, ''N'');
END;
```

9.3 Funkcja ZMIEN_STATUS_REZERWACJI2

```
CREATE OR REPLACE PROCEDURE zmien_status_rezerwacji2(id_rezerwacji_param NUMBER,
    new_status CHAR) AS
    old_status CHAR(1);
    counter   INTEGER;
    difference_of_places_available INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO counter
    FROM WYCIECZKI_PRZYSZLE wp
            JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
    WHERE r.NR_REZERWACJI = id_rezerwacji_param;

    IF counter = 0 THEN
        raise_application_error(-20015,
                                'Nie mozna zmienic statusu rezerwacji dla wycieczek,
                                    ktore nie sa w przyszlosci');
    END IF;

    SELECT COUNT(*)
    INTO counter
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = id_rezerwacji_param;

    IF counter = 0 THEN
        raise_application_error(-20016,
                                'Nie istnieje rezerwacja o podanym numerze');
    END IF;

    SELECT STATUS
    INTO old_status
    FROM REZERWACJE
    WHERE NR_REZERWACJI = id_rezerwacji_param;

    CASE
        WHEN old_status IS NULL
            THEN
                raise_application_error(-20016,
                                        'Nie istnieje rezerwacja o podanym numerze');

        WHEN old_status = 'A'
            THEN
                SELECT COUNT(*)
                INTO counter
                FROM DOSTEPNE_WYCIECZKI_VIEW v
                        JOIN REZERWACJE r on R.ID_WYCIECZKI = v.ID_WYCIECZKI
                WHERE r.NR_REZERWACJI = id_rezerwacji_param;

                IF counter = 0 THEN
                    raise_application_error(-20018,
                                            'Nie ma wystarczajacej ilosci miejsc, aby
                                                przywrocic anulowana rezerwacje o podanym
                                                id');
                END IF;
```

```
        WHEN new_status = 'N'
            THEN raise_application_error(-20017,
                                    'Rezerwacja, ktora juz istnieje nie moze zmienic
                                        statusu na nowa. Podano bledny argument');

        ELSE
            IF new_status = 'A'
            THEN
                difference_of_places_available := 1;
            ELSE
                difference_of_places_available := 0;
            END IF;
        END CASE;

    UPDATE REZERWACJE
    SET STATUS = new_status
    WHERE NR_REZERWACJI = id_rezerwacji_param;

    UPDATE WYCIECZKI w
    SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + difference_of_places_available
    WHERE w.ID_WYCIECZKI = (SELECT ID_WYCIECZKI
                            FROM REZERWACJE r
                            WHERE r.NR_REZERWACJI = id_rezerwacji_param);

    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (id_rezerwacji, CURRENT_DATE, new_status);
END;
```

9.4 Funkcja ZMIEN_LICZBE_MIEJSC2

```
CREATE OR REPLACE PROCEDURE zmien_liczbe_miejsc2(id_wycieczki_param NUMBER,
    liczba_nowych_miejsc_param NUMBER)
AS
    counter INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO counter
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = id_wycieczki_param;

    IF counter = 0 THEN
        raise_application_error(-20011, 'Nie znaleziono wycieczki o podanym id');
    END IF;

    SELECT wm.LICZBA_MIEJSC - wm.LICZBA_WOLNYCH_MIEJSC
    INTO counter
    FROM WYCIECZKI_MIEJSCA wm
    WHERE wm.ID_WYCIECZKI = id_wycieczki_param;

    IF liczba_nowych_miejsc_param < counter OR liczba_nowych_miejsc_param < 0 THEN
        raise_application_error(-20030, 'Podana liczba nowych miejsc ma zbyt mala
            wartosc');
    END IF;

    UPDATE WYCIECZKI
    SET LICZBA_MIEJSC = liczba_nowych_miejsc_param,
        LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + (liczba_nowych_miejsc_param -
            LICZBA_MIEJSC)
    WHERE ID_WYCIECZKI = id_wycieczki_param;
END;
```

10 Triggery

10.1 Trigger obsługujący dodanie rezerwacji - DODAJ_NOWA_REZERWACJE

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

    UPDATE WYCIECZKI w
    SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1
    WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
END;
```

10.2 Trigger obsługujący zmiane statusu - ZMIEN_STATUS

```
CREATE OR REPLACE TRIGGER zmien_status
    AFTER UPDATE
    ON REZERWACJE
    FOR EACH ROW
DECLARE
    difference_of_places_available int;
BEGIN
    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);

    CASE
         WHEN :OLD.STATUS <> 'A' AND :NEW.STATUS = 'A'
            THEN
                difference_of_places_available := 1;

        WHEN :OLD.STATUS = 'A' AND :NEW.STATUS <> 'A'
            THEN
                difference_of_places_available := -1;

        ELSE
            difference_of_places_available := 0;
        END CASE;

    UPDATE WYCIECZKI w
    SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + difference_of_places_available
    WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
end;
```

10.3 Trigger zabraniający usunięcia rezerwacji - USUN_REZERWACJE

```
CREATE OR REPLACE TRIGGER usun_rezerwacje
    BEFORE DELETE
    ON REZERWACJE
    FOR EACH ROW
BEGIN
    raise_application_error(-20045, 'Usuniecie rezerwacji jest niemozliwe.');
end;
```

10.4 Trigger obsługujący zmianę liczby miejsc na poziomie wycieczki - ZMIEN_LICZBE_MIEJSC

```
CREATE OR REPLACE TRIGGER zmien_liczbe_miejsc
    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;
```

11 Uaktualnione procedury modyfikujące dane po dodaniu triggerów

11.1 Funkcja DODAJ_REZERWACJE3

```
CREATE OR REPLACE PROCEDURE dodaj_rezerwacje3(id_wycieczki_param NUMBER,
    id_osoby_param NUMBER)
AS
    counter          INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO counter
    FROM DOSTEPNE_WYCIECZKI_VIEW v
    WHERE v.ID_WYCIECZKI = id_wycieczki_param;

    IF counter = 0 THEN
        raise_application_error(-20011, 'Nie znaleziono wycieczki o podanym id');
    END IF;

    SELECT COUNT(*)
    INTO counter
    FROM OSOBY o
    WHERE o.ID_OSOBY = id_osoby_param;

    IF counter = 0 THEN
        raise_application_error(-20012, 'Nie znaleziono osoby o podanym id');
    END IF;
```

```
    SELECT COUNT(*)
    INTO COUNTER
    FROM REZERWACJE r
    WHERE r.ID_OSOBY = id_osoby_param
      AND r.ID_WYCIECZKI = id_wycieczki_param;

    IF counter > 0 THEN
        raise_application_error(-20013, 'Rezerwacja z danym id wycieczki i id osoby
            juz istnieje');
    END IF;

    INSERT INTO REZERWACJE (ID_WYCIECZKI, ID_OSOBY, STATUS)
    VALUES (id_wycieczki_param, id_osoby_param, 'N');
END;
```

11.2 Funkcja ZMIEN_STATUS_REZERWACJI3

```
CREATE OR REPLACE PROCEDURE zmien_status_rezerwacji3(id_rezerwacji_param NUMBER,
    new_status CHAR) AS
    old_status CHAR(1);
    counter   INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO counter
    FROM WYCIECZKI_PRZYSZLE wp
            JOIN REZERWACJE r ON r.ID_WYCIECZKI = wp.ID_WYCIECZKI
    WHERE r.NR_REZERWACJI = id_rezerwacji_param;

    IF counter = 0 THEN
        raise_application_error(-20015,
                                'Nie mozna zmienic statusu rezerwacji dla wycieczek,
                                    ktore nie sa w przyszlosci');
    END IF;

    SELECT COUNT(*)
    INTO counter
    FROM REZERWACJE r
    WHERE r.NR_REZERWACJI = id_rezerwacji_param;

    IF counter = 0 THEN
        raise_application_error(-20016,
                                'Nie istnieje rezerwacja o podanym numerze');
    END IF;

    SELECT STATUS
    INTO old_status
    FROM REZERWACJE
    WHERE NR_REZERWACJI = id_rezerwacji_param;

    CASE
        WHEN old_status IS NULL
            THEN
                raise_application_error(-20016,
                                        'Nie istnieje rezerwacja o podanym numerze');

        WHEN old_status = 'A'
            THEN
                SELECT COUNT(*)
                INTO counter
                FROM DOSTEPNE_WYCIECZKI_VIEW v
                        JOIN REZERWACJE r on R.ID_WYCIECZKI = v.ID_WYCIECZKI
                WHERE r.NR_REZERWACJI = id_rezerwacji_param;

                IF counter = 0 THEN
                    raise_application_error(-20018,
                                            'Nie ma wystarczajacej ilosci miejsc, aby
                                                przywrocic anulowana rezerwacje o podanym
                                                id');
                END IF;
```

```
        WHEN new_status = 'N'
            THEN raise_application_error(-20017,
                                    'Rezerwacja, ktora juz istnieje nie moze zmienic
                                        statusu na nowa. Podano bledny argument');

        ELSE NULL;
        END CASE;

    UPDATE REZERWACJE
    SET STATUS = new_status
    WHERE NR_REZERWACJI = id_rezerwacji_param;
END;
```

11.3 Funkcja ZMIEN_LICZBE_MIEJSC3

```
CREATE OR REPLACE PROCEDURE zmien_liczbe_miejsc3(id_wycieczki_param NUMBER,
    liczba_nowych_miejsc_param NUMBER)
AS
    counter INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO counter
    FROM WYCIECZKI w
    WHERE w.ID_WYCIECZKI = id_wycieczki_param;

    IF counter = 0 THEN
        raise_application_error(-20011, 'Nie znaleziono wycieczki o podanym id');
    END IF;

    SELECT wm.LICZBA_MIEJSC - wm.LICZBA_WOLNYCH_MIEJSC
    INTO counter
    FROM WYCIECZKI_MIEJSCA wm
    WHERE wm.ID_WYCIECZKI = id_wycieczki_param;

    IF liczba_nowych_miejsc_param < counter OR liczba_nowych_miejsc_param < 0 THEN
        raise_application_error(-20030, 'Podana liczba nowych miejsc ma zbyt mala
            wartosc');
    END IF;

    UPDATE WYCIECZKI
    SET LICZBA_MIEJSC = liczba_nowych_miejsc_param
    WHERE ID_WYCIECZKI = id_wycieczki_param;
END;
```