SPRAWOZDANIE I

BAZY DANYCH

ORACLE

DAWID BIAŁKA

2019/2020

Tworzenie wszystkich tabel:

```
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 WYCIECZKI
ADD LICZBA_WOLNYCH_MIEJSC INT;

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;

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
);
```

Wprowadzone dane:

```
INSERT INTO OSOBY (imie, nazwisko, pesel, kontakt)
VALUES ('Adam', 'Kowalski', '87654321', 'tel: 6623');
INSERT INTO OSOBY (imie, nazwisko, pesel, kontakt)
VALUES ('Jan', 'Nowak', '12345678', 'tel: 2312, dzwonić po 18.00');
INSERT INTO OSOBY (imie, nazwisko, pesel, kontakt)
VALUES ('Jan', 'Matejko', '87654332', 'fax: 2137');
INSERT INTO OSOBY (imie, nazwisko, pesel, kontakt)
VALUES ('Wladyslaw', 'Jagiello', '11111111', 'tel: 1410');
INSERT INTO OSOBY (imie, nazwisko, pesel, kontakt)
VALUES ('Stanislaw', 'Poniatowski', '12345978', 'fax: 1795');
INSERT INTO OSOBY (imie, nazwisko, pesel, kontakt)
VALUES ('Kazimierz', 'Wielki', '12355678', 'tel: 23999');
INSERT INTO OSOBY (imie, nazwisko, pesel, kontakt)
VALUES ('Zygmunt', 'Baza', '11345678', 'tel: 23444');
INSERT INTO OSOBY (imie, nazwisko, pesel, kontakt)
VALUES ('Wladyslaw', 'Lokietek', '12325678', 'tel: 2312');
INSERT INTO OSOBY (imie, nazwisko, pesel, kontakt)
VALUES ('Jan', 'Sobieski', '12340678', 'tel: 1681');
INSERT INTO OSOBY (imie, nazwisko, pesel, kontakt)
VALUES ('Stefan', 'Batory', '52345678', 'tel: 2372');

INSERT INTO WYCIECZKI (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wycieczka do Paryza', 'Francja', TO_DATE('2016-01-01', 'YYYY-MM-DD'),
'Ciekawa wycieczka ...', 3);
```

```
INSERT INTO WYCIECZKI (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Piękny Kraków', 'Polska', TO_DATE('2017-02-03', 'YYYY-MM-DD'),
'Najciekawa wycieczka ...', 4);
INSERT INTO WYCIECZKI (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wieliczka', 'Polska', TO_DATE('2022-03-03', 'YYYY-MM-DD'), 'Zadziwiająca
kopalnia ...', 3);
INSERT INTO WYCIECZKI (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Krapkowice', 'Polska', TO_DATE('2021-03-03', 'YYYY-MM-DD'), 'Stare
wapienniki...', 5);

INSERT INTO rezerwacje (id_wycieczki, id_osoby, status)
VALUES (1, 1, 'N');
INSERT INTO rezerwacje (id_wycieczki, id_osoby, status)
VALUES (2, 2, 'P');
INSERT INTO rezerwacje (id_wycieczki, id_osoby, status)
VALUES (3, 3, 'P');
INSERT INTO rezerwacje (id_wycieczki, id_osoby, status)
VALUES (4, 4, 'N');
INSERT INTO rezerwacje (id_wycieczki, id_osoby, status)
VALUES (1, 5, 'N');
INSERT INTO rezerwacje (id_wycieczki, id_osoby, status)
VALUES (2, 6, 'P');
INSERT INTO rezerwacje (id_wycieczki, id_osoby, status)
VALUES (3, 7, 'P');
INSERT INTO rezerwacje (id_wycieczki, id_osoby, status)
VALUES (4, 8, 'P');
INSERT INTO rezerwacje (id_wycieczki, id_osoby, status)
VALUES (1, 9, 'P');
INSERT INTO rezerwacje (id_wycieczki, id_osoby, status)
VALUES (2, 10, 'A');
```

Widoki w wersji podstawowej:

```
CREATE VIEW RezerwacjeWszystkie AS
     SELECT
        w.ID_WYCIECZKI,
        w.NAZWA,
        w.KRAJ,
        w.DATA,
        o.ID_OSOBY,
        o.IMIE,
        o.NAZWISKO,
        r.STATUS
     FROM REZERWACJE R
     INNER JOIN WYCIECZKI W ON R.ID_WYCIECZKI = W.ID_WYCIECZKI
     INNER JOIN OSOBY O ON R.ID_OSOBY = O.ID_OSOBY;

CREATE VIEW RezerwacjePotwierdzone AS
    SELECT
      ID_WYCIECZKI,
      KRAJ,
      DATA,
      NAZWA,
      ID_OSOBY,
      IMIE,
      NAZWISKO,
      STATUS
    FROM RezerwacjeWszystkie
```

```
    where STATUS = 'P';

CREATE VIEW RezerwacjeWPrzyszlosci AS
    SELECT
      KRAJ,
      DATA,
      NAZWA,
      ID_OSOBY,
      IMIE,
      NAZWISKO,
      STATUS
    FROM RezerwacjeWszystkie
    WHERE DATA > CURRENT_DATE;

CREATE VIEW WycieczkiMiejsca AS
    SELECT
      ID_WYCIECZKI,
      KRAJ,
      DATA,
      NAZWA,
      OPIS,
      LICZBA_MIEJSC,
      LICZBA_MIEJSC - (SELECT COUNT(*) FROM RezerwacjeWszystkie RW
          WHERE RW.ID_WYCIECZKI = W.ID_WYCIECZKI AND RW.STATUS <> 'A')
      as LICZBA_WOLNYCH_MIEJSC
    FROM WYCIECZKI W;

CREATE VIEW WycieczkiDostepne AS
    SELECT
      ID_WYCIECZKI,
      KRAJ,
      DATA,
      NAZWA,
      OPIS,
      LICZBA_MIEJSC,
      LICZBA_WOLNYCH_MIEJSC
    FROM WycieczkiMiejsca
    WHERE DATA > CURRENT_DATE AND LICZBA_WOLNYCH_MIEJSC > 0;
```

Widoki po dodaniu pola liczba_wolnych_miejsc w tabeli WYCIECZKI:

```
CREATE VIEW WycieczkiMiejsca2 AS
    SELECT
      ID_WYCIECZKI,
      KRAJ,
      DATA,
      NAZWA,
      OPIS,
      LICZBA_MIEJSC,
      LICZBA_WOLNYCH_MIEJSC
    FROM WYCIECZKI W;
```

```sql
CREATE VIEW WycieczkiDostepne2 AS
    SELECT
      ID_WYCIECZKI,
      KRAJ,
      DATA,
      NAZWA,
      OPIS,
      LICZBA_MIEJSC,
      LICZBA_WOLNYCH_MIEJSC
    FROM WycieczkiMiejsca2
    WHERE DATA > CURRENT_DATE AND LICZBA_WOLNYCH_MIEJSC > 0;
```

Funkcje zwracające dane:

```sql
CREATE TYPE OSOBY_WYCIECZKA_REKORD AS object
(
  KRAJ            VARCHAR(50),
  "DATA"          DATE,
  NAZWA_WYCIECZKI VARCHAR(100),
  ID_OSOBY        INT,
  IMIE            VARCHAR2(50),
  NAZWISKO        VARCHAR2(50),
  STATUS          CHAR(1)
);


CREATE TYPE OSOBY_WYCIECZKA_TABELA IS TABLE OF OSOBY_WYCIECZKA_REKORD;

CREATE FUNCTION UczestnicyWycieczki(id_wycieczki INT)
  return OSOBY_WYCIECZKA_TABELA as v_ret OSOBY_WYCIECZKA_REKORD;
  istnieje INT;
  BEGIN
    SELECT COUNT(*) INTO istnieje FROM WYCIECZKI WHERE WYCIECZKI.ID_WYCIECZKI =
id_wycieczki;

    IF istnieje = 0 THEN
      raise_application_error(-20001, 'Wycieczka o podanym ID nie istenieje');
    END IF;

    SELECT OSOBY_WYCIECZKA_REKORD(R.KRAJ, R.DATA, R.NAZWA, R.ID_OSOBY, R.IMIE,
R.NAZWISKO, R.STATUS)
        BULK COLLECT INTO v_ret
    FROM RezerwacjePotwierdzone R
    WHERE R.ID_WYCIECZKI = id_wycieczki
    ;
    return v_ret;
END UczestnicyWycieczki;

CREATE FUNCTION RezerwacjeOsoby(id_osoby INT)
  return OSOBY_WYCIECZKA_TABELA as v_ret OSOBY_WYCIECZKA_REKORD;
  istnieje INT;
  BEGIN
    SELECT COUNT(*) INTO istnieje FROM OSOBY O WHERE O.ID_OSOBY = id_osoby;

    IF istnieje = 0 THEN
      raise_application_error(-20001, 'Osoba o poanym ID nie istnieje');
    END IF;
```

```
    SELECT OSOBY_WYCIECZKA_REKORD(R.KRAJ, R.DATA, R.NAZWA, R.ID_OSOBY, R.IMIE,
R.NAZWISKO, R.STATUS)
        BULK COLLECT INTO v_ret
    FROM RezerwacjeWszystkie R
    WHERE R.ID_OSOBY = id_osoby;

    return v_ret;
END RezerwacjeOsoby;

CREATE TYPE WYCIECZKA_REKORD AS object
(
  ID_WYCIECZKI          INT,
  KRAJ                  VARCHAR(50),
  "DATA"                DATE,
  NAZWA_WYCIECZKI       VARCHAR(100),
  LICZBA_MIEJSC         INT,
  LICZBA_WOLNYCH_MIEJSC INT
);

CREATE TYPE WYCIECZKA_TABELA IS TABLE OF WYCIECZKA_REKORD;

CREATE FUNCTION DostepneWycieczki(kraj WYCIECZKI.KRAJ%TYPE, data_od DATE, data_do
DATE)
  return OSOBY_WYCIECZKA_TABELA as v_ret WYCIECZKA_REKORD;
  BEGIN

    IF data_od > data_do THEN
      raise_application_error(-20001, 'Data od nie moze byc wieksza niz data do');
    ELSIF data_do < CURRENT_DATE THEN
      raise_application_error(-20002, 'Dostepne wycieczki moga byc tylko w
przyszlosci');
    END IF;

    SELECT WYCIECZKA_REKORD(W.ID_WYCIECZKI, W.KRAJ, W.DATA, W.NAZWA,
W.LICZBA_MIEJSC, W.LICZBA_WOLNYCH_MIEJSC)
        BULK COLLECT INTO v_ret
    FROM WycieczkiDostepne W
    WHERE W.DATA > data_od AND W.DATA < data_do AND W.KRAJ = kraj;

    return v_ret;
END DostepneWycieczki;
```

W Oraclu musimy stworzyć strukturę odpowiadającą za rekord w tabeli, którą chcemy zwrócić oraz strukturę tabela złożoną z wcześniej zdefiniowanej struktury.

Procedury:

```
CREATE PROCEDURE DodajRezerwacje(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE,
id_osoby OSOBY.ID_OSOBY%TYPE) AS
  istnieje INT;
  nowe_id  INT;
  BEGIN
    SELECT COUNT(*) INTO istnieje
    FROM OSOBY
    WHERE OSOBY.ID_OSOBY = id_osoby;

    IF istnieje = 0
    THEN
      raise_application_error(-20001, 'Osoba o poanym ID nie istnieje');
    END IF;

    SELECT COUNT(*) INTO istnieje
    FROM WycieczkiDostepne W
    WHERE w.ID_WYCIECZKI = id_wycieczki;

    IF istnieje = 0
    THEN
      raise_application_error(-20002, 'Wycieczka o podanym ID nie istnieje lub nie
jest juz dostepna');
    END IF;

    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    WHERE R.ID_WYCIECZKI = id_wycieczki AND R.ID_OSOBY = id_osoby;

    IF istnieje > 0
    THEN
      raise_application_error(-20003, 'Rejestracja na dana wycieczke i osobe juz
istnieje');
    END IF;

    INSERT INTO REZERWACJE (id_wycieczki, id_osoby, STATUS)
    VALUES (id_wycieczki, id_osoby, 'N');

    SELECT COUNT(*) INTO nowe_id FROM REZERWACJE;

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

END DodajRezerwacje;
```

```
CREATE PROCEDURE ZmienStatusRezerwacji(id_rezerwacji
REZERWACJE.NR_REZERWACJI%TYPE, nowy_status REZERWACJE.STATUS%TYPE) AS
  stary_status        REZERWACJE.STATUS%TYPE;
  istnieje            INT;
  wolne_miejsca       INT;
  BEGIN
    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    WHERE R.NR_REZERWACJI = id_rezerwacji;

    IF istnieje = 0 THEN
      raise_application_error(-20001, 'Rejestracja o podanym ID nie istnieje');
    ELSIF nowy_status <> 'N' OR nowy_status <> 'P' OR nowy_status <> 'Z' OR
nowy_status <> 'A' THEN
        raise_application_error(-20002, 'Nieprawidlowy nowy status');
    END IF;

    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    INNER JOIN WYCIECZKI W on R.ID_WYCIECZKI = W.ID_WYCIECZKI
    WHERE R.NR_REZERWACJI = id_rezerwacji AND W.DATA > CURRENT_DATE;

    IF istnieje = 0 THEN
      raise_application_error(-20003, 'Nie mozna modyfikowac przeszlych
rezerwacji');
    END IF;

    SELECT WM.LICZBA_WOLNYCH_MIEJSC INTO wolne_miejsca
    FROM REZERWACJE R
    INNER JOIN WycieczkiMiejsca WM on R.ID_WYCIECZKI = WM.ID_WYCIECZKI
    WHERE R.NR_REZERWACJI = id_rezerwacji;

    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    INNER JOIN WYCIECZKI W on R.ID_WYCIECZKI = W.ID_WYCIECZKI
    WHERE R.NR_REZERWACJI = id_rezerwacji AND W.DATA > CURRENT_DATE;

    CASE
      WHEN nowy_status = 'N' THEN
         raise_application_error(-20004, 'Nie mozna zmodyfikowac istniejacej
rezerwacji na status 'N'');
      WHEN nowy_status = 'P' THEN
        IF stary_status = 'Z' THEN
          raise_application_error(-20005, 'Nie mozna zmienic statusu z zaplaconej
na potwierdzona');
        ELSIF stary_status = 'A' AND wolne_miejsca = 0 THEN
          raise_application_error(-20006, 'Brak wolnych miejsc na dana
wycieczke');
        END IF;
      WHEN nowy_status = 'Z' THEN
        IF stary_status = 'A' and wolne_miejsca = 0 THEN
          raise_application_error(-20007, 'Brak wolnych miejsc na dana
wycieczke');
        END IF;
    END CASE;

    UPDATE REZERWACJE
    SET STATUS = nowy_status
    WHERE NR_REZERWACJI = id_rezerwacji;
```

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

END ZmienStatusRezerwacji;
```

```
CREATE PROCEDURE ZmienLiczbeMiejsc(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE,
nowe_miejsca WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
  zajete INT;
  istnieje INT,
  BEGIN
    SELECT COUNT(*) INTO istnieje
    FROM WYCIECZKI W
    WHERE W.ID_WYCIECZKI = id_wycieczki AND W.DATA > CURRENT_DATE;

    IF istnieje = 0
    THEN
      raise_application_error(-20001, 'Wycieczka o podanym ID nie istnieje lub juz
sie odbyla');
    END IF;

    SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC INTO zajete
    FROM WycieczkiMiejsca WM
    WHERE WM.ID_WYCIECZKI = id_wycieczki;

    IF nowe_miejsca < 0
    THEN
      raise_application_error(-20002, 'Nowa liczba miejsc jest ujemna');
    ELSIF zajete > nowe_miejsca
    THEN
      raise_application_error(-20003, 'Nowa liczba miejsc zbyt niska');
    END IF;

    UPDATE WYCIECZKI
    SET LICZBA_MIEJSC = nowe_miejsca
    WHERE ID_WYCIECZKI = id_wycieczki;
END ZmienLiczbeMiejsc;
```

Procedury po dodaniu pola liczba_wolnych_miejsc w tabeli WYCIECZKI.

```
CREATE PROCEDURE DodajRezerwacje2(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE,
id_osoby OSOBY.ID_OSOBY%TYPE) AS
  istnieje INT;
  nowe_id  INT;
  BEGIN
    SELECT COUNT(*) INTO istnieje
    FROM OSOBY
    WHERE OSOBY.ID_OSOBY = id_osoby;

    IF istnieje = 0
    THEN
      raise_application_error(-20001, 'Osoba o poanym ID nie istnieje');
    END IF;

    SELECT COUNT(*) INTO istnieje
    FROM WycieczkiDostepne W
    WHERE w.ID_WYCIECZKI = id_wycieczki;
```

```
    IF istnieje = 0
    THEN
      raise_application_error(-20002, 'Wycieczka o podanym ID nie istnieje lub nie
jest juz dostepna');
    END IF;

    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    WHERE R.ID_WYCIECZKI = id_wycieczki AND R.ID_OSOBY = id_osoby;

    IF istnieje > 0
    THEN
      raise_application_error(-20003, 'Rejestracja na dana wycieczke i osobe juz
istnieje');
    END IF;

    INSERT INTO REZERWACJE (id_wycieczki, id_osoby, STATUS)
    VALUES (id_wycieczki, id_osoby, 'N');

    SELECT COUNT(*) INTO nowe_id FROM REZERWACJE;

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

    UPDATE WYCIECZKI W
    SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC - 1
    WHERE W.ID_WYCIECZKI = id_wycieczki;

END DodajRezerwacje2;
```

```
CREATE PROCEDURE ZmienStatusRezerwacji2(id_rezerwacji
REZERWACJE.NR_REZERWACJI%TYPE, nowy_status REZERWACJE.STATUS%TYPE) AS
  stary_status        REZERWACJE.STATUS%TYPE;
  istnieje            INT;
  wolne_miejsca       INT;
  zmiana_miejsc       INT;
  BEGIN
    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    WHERE R.NR_REZERWACJI = id_rezerwacji;

    IF istnieje = 0 THEN
      raise_application_error(-20001, 'Rejestracja o podanym ID nie istnieje');
    ELSIF nowy_status <> 'N' OR nowy_status <> 'P' OR nowy_status <> 'Z' OR
nowy_status <> 'A' THEN
        raise_application_error(-20002, 'Nieprawidlowy nowy status');
    END IF;

    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    INNER JOIN WYCIECZKI W on R.ID_WYCIECZKI = W.ID_WYCIECZKI
    WHERE R.NR_REZERWACJI = id_rezerwacji AND W.DATA > CURRENT_DATE;

    IF istnieje = 0 THEN
      raise_application_error(-20003, 'Nie mozna modyfikowac przeszlych
rezerwacji');
    END IF;
```

```
    SELECT WM.LICZBA_WOLNYCH_MIEJSC INTO wolne_miejsca
    FROM REZERWACJE R
    INNER JOIN WycieczkiMiejsca WM on R.ID_WYCIECZKI = WM.ID_WYCIECZKI
    WHERE R.NR_REZERWACJI = id_rezerwacji;

    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    INNER JOIN WYCIECZKI W on R.ID_WYCIECZKI = W.ID_WYCIECZKI
    WHERE R.NR_REZERWACJI = id_rezerwacji AND W.DATA > CURRENT_DATE;

    wolne_miejsca:= 0;

    CASE
      WHEN nowy_status = 'N' THEN
         raise_application_error(-20004, 'Nie mozna zmodyfikowac istniejacej
rezerwacji na status 'N'');
      WHEN nowy_status = 'P' THEN
        IF stary_status = 'Z' THEN
          raise_application_error(-20005, 'Nie mozna zmienic statusu z zaplaconej
na potwierdzona');
        ELSIF stary_status = 'A' AND wolne_miejsca = 0 THEN
          raise_application_error(-20006, 'Brak wolnych miejsc na dana
wycieczke');
        END IF;
        zmiana_miejsc:= -1;
      WHEN nowy_status = 'Z' THEN
        IF stary_status = 'A' and wolne_miejsca = 0 THEN
          raise_application_error(-20007, 'Brak wolnych miejsc na dana
wycieczke');
        END IF;
        zmiana_miejsc:= -1;
      WHEN nowy_status = 'A' THEN
        zmiana_miejsc:= 1;
    END CASE;

    UPDATE REZERWACJE
    SET STATUS = nowy_status
    WHERE NR_REZERWACJI = id_rezerwacji;

    IF zmiana_miejsc <> 0 THEN
      UPDATE WYCIECZKI W
      SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + zmiana_miejsc
      WHERE W.ID_WYCIECZKI = (SELECT ID_WYCIECZKI FROM REZERWACJE R WHERE
R.NR_REZERWACJI = id_rezerwacji);
    END IF;


    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (id_rezerwacji, CURRENT_DATE, nowy_status);

END ZmienStatusRezerwacji2;
```

```
CREATE PROCEDURE ZmienLiczbeMiejsc2(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE,
nowe_miejsca WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
  zajete INT;
  istnieje INT,
  BEGIN
    SELECT COUNT(*) INTO istnieje
    FROM WYCIECZKI W
    WHERE W.ID_WYCIECZKI = id_wycieczki AND W.DATA > CURRENT_DATE;

    IF istnieje = 0
    THEN
      raise_application_error(-20002, 'Wycieczka o podanym ID nie istnieje lub juz
sie odbyla');
    END IF;

    SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC INTO zajete
    FROM WycieczkiMiejsca WM
    WHERE WM.ID_WYCIECZKI = id_wycieczki;

    IF nowe_miejsca < 0
    THEN
      raise_application_error(-20001, 'Nowa liczba miejsc jest ujemna');
    ELSIF zajete > nowe_miejsca
    THEN
      raise_application_error(-20006, 'Nowa liczba miejsc zbyt niska');
    END IF;

    UPDATE WYCIECZKI
    SET LICZBA_MIEJSC = nowe_miejsca
    WHERE ID_WYCIECZKI = id_wycieczki;
    przelicz();
END ZmienLiczbeMiejsc2;
```

```
CREATE 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 przelicz;
```

Triggry:

```
CREATE TRIGGER ZmianaMiejsc_trigger
  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 ZmianaMiejsc_trigger;
```

```
CREATE TRIGGER ZmianaStatusu_trigger
  AFTER UPDATE
  ON REZERWACJE
  FOR EACH ROW
  DECLARE
    zmiana_miejsc INT;
  BEGIN
    INSERT INTO REZERWACJE_LOG (ID_REZERWACJI, DATA, STATUS)
    VALUES (:NEW.NR_REZERWACJI, CURRENT_DATE, :NEW.STATUS);

    zmiana_miejsc:= 0;

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

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

    IF zmiana_miejsc <> 0 THEN
      UPDATE WYCIECZKI w
      SET LICZBA_WOLNYCH_MIEJSC = LICZBA_WOLNYCH_MIEJSC + zmiana_miejsc
      WHERE w.ID_WYCIECZKI = :NEW.ID_WYCIECZKI;
    END IF;

END ZmianaStatusu_trigger;

CREATE TRIGGER NowaRezerwacja_trigger
  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  NowaRezerwacja_trigger;
```

Procedury po dodaniu triggerów:

```
CREATE PROCEDURE DodajRezerwacje3(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE,
id_osoby OSOBY.ID_OSOBY%TYPE) AS
  istnieje INT;
  nowe_id  INT;
  BEGIN
    SELECT COUNT(*) INTO istnieje
    FROM OSOBY
    WHERE OSOBY.ID_OSOBY = id_osoby;

    IF istnieje = 0
    THEN
      raise_application_error(-20001, 'Osoba o poanym ID nie istnieje');
    END IF;

    SELECT COUNT(*) INTO istnieje
    FROM WycieczkiDostepne W
    WHERE w.ID_WYCIECZKI = id_wycieczki;

    IF istnieje = 0
    THEN
      raise_application_error(-20002, 'Wycieczka o podanym ID nie istnieje lub nie
jest juz dostepna');
    END IF;

    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    WHERE R.ID_WYCIECZKI = id_wycieczki AND R.ID_OSOBY = id_osoby;

    IF istnieje > 0
    THEN
      raise_application_error(-20003, 'Rejestracja na dana wycieczke i osobe juz
istnieje');
    END IF;

    INSERT INTO REZERWACJE (id_wycieczki, id_osoby, STATUS)
    VALUES (id_wycieczki, id_osoby, 'N');

END DodajRezerwacje3;
```

```
CREATE PROCEDURE ZmienStatusRezerwacji3(id_rezerwacji
REZERWACJE.NR_REZERWACJI%TYPE, nowy_status REZERWACJE.STATUS%TYPE) AS
  stary_status        REZERWACJE.STATUS%TYPE;
  istnieje            INT;
  wolne_miejsca       INT;
  BEGIN
    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    WHERE R.NR_REZERWACJI = id_rezerwacji;

    IF istnieje = 0 THEN
      raise_application_error(-20001, 'Rejestracja o podanym ID nie istnieje');
    ELSIF nowy_status <> 'N' OR nowy_status <> 'P' OR nowy_status <> 'Z' OR
nowy_status <> 'A' THEN
        raise_application_error(-20002, 'Nieprawidlowy nowy status');
    END IF;

    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    INNER JOIN WYCIECZKI W on R.ID_WYCIECZKI = W.ID_WYCIECZKI
    WHERE R.NR_REZERWACJI = id_rezerwacji AND W.DATA > CURRENT_DATE;
    IF istnieje = 0 THEN
      raise_application_error(-20003, 'Nie mozna modyfikowac przeszlych
rezerwacji');
    END IF;

    SELECT WM.LICZBA_WOLNYCH_MIEJSC INTO wolne_miejsca
    FROM REZERWACJE R
    INNER JOIN WycieczkiMiejsca WM on R.ID_WYCIECZKI = WM.ID_WYCIECZKI
    WHERE R.NR_REZERWACJI = id_rezerwacji;

    SELECT COUNT(*) INTO istnieje
    FROM REZERWACJE R
    INNER JOIN WYCIECZKI W on R.ID_WYCIECZKI = W.ID_WYCIECZKI
    WHERE R.NR_REZERWACJI = id_rezerwacji AND W.DATA > CURRENT_DATE;

    CASE
      WHEN nowy_status = 'N' THEN
         raise_application_error(-20004, 'Nie mozna zmodyfikowac istniejacej
rezerwacji na status 'N'');
      WHEN nowy_status = 'P' THEN
        IF stary_status = 'Z' THEN
          raise_application_error(-20005, 'Nie mozna zmienic statusu z zaplaconej
na potwierdzona');
        ELSIF stary_status = 'A' AND wolne_miejsca = 0 THEN
          raise_application_error(-20006, 'Brak wolnych miejsc na dana
wycieczke');
        END IF;
      WHEN nowy_status = 'Z' THEN
        IF stary_status = 'A' and wolne_miejsca = 0 THEN
          raise_application_error(-20007, 'Brak wolnych miejsc na dana
wycieczke');
        END IF;
    END CASE;

    UPDATE REZERWACJE
    SET STATUS = nowy_status
```

```
    WHERE NR_REZERWACJI = id_rezerwacji;
END ZmienStatusRezerwacji3;
CREATE PROCEDURE ZmienLiczbeMiejsc3(id_wycieczki WYCIECZKI.ID_WYCIECZKI%TYPE,
nowe_miejsca WYCIECZKI.LICZBA_MIEJSC%TYPE) AS
  zajete INT;
  istnieje INT,
  BEGIN
    SELECT COUNT(*) INTO istnieje
    FROM WYCIECZKI W
    WHERE W.ID_WYCIECZKI = id_wycieczki AND W.DATA > CURRENT_DATE;

    IF istnieje = 0
    THEN
      raise_application_error(-20002, 'Wycieczka o podanym ID nie istnieje lub juz
sie odbyla');
    END IF;

    SELECT LICZBA_MIEJSC - LICZBA_WOLNYCH_MIEJSC INTO zajete
    FROM WycieczkiMiejsca WM
    WHERE WM.ID_WYCIECZKI = id_wycieczki;

    IF nowe_miejsca < 0
    THEN
      raise_application_error(-20001, 'Nowa liczba miejsc jest ujemna');
    ELSIF zajete > nowe_miejsca
    THEN
      raise_application_error(-20006, 'Nowa liczba miejsc zbyt niska');
    END IF;

    UPDATE WYCIECZKI
    SET LICZBA_MIEJSC = nowe_miejsca
    WHERE ID_WYCIECZKI = id_wycieczki;
END ZmienLiczbeMiejsc3;
```