Piotr Szczerbiński
Bazy Danych 2, Lab1
Sprawozdanie

Plik wyc.sql:

```
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;


--dodane w punkcie 6:
--rezerwacje_log(id, id_rezerwacji, data, status)
CREATE TABLE REZERWACJE_LOG
(
 LOG_ID INT GENERATED ALWAYS AS IDENTITY NOT NULL,
 NR_REZERWACJI INT,
 DATA DATE,
 STATUS CHAR(1),
 CONSTRAINT REZERWACJE_LOG_PK PRIMARY KEY
 (
   LOG_ID
 )
 ENABLE
);

ALTER TABLE REZERWACJE_LOG
ADD CONSTRAINT REZERWACJE_LOG_FK1 FOREIGN KEY
(
 NR_REZERWACJI
)
REFERENCES REZERWACJE
(
 NR_REZERWACJI
)
ENABLE;
```

Plik wyc_insert.sql

```
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('Adam', 'Kowalski', '87654321', 'tel: 6623');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Jan', 'Nowak', '12345678', 'tel: 2312, dzwonić po 18.00');

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wycieczka do Paryza','Francja',TO_DATE('2017-03-03','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 ...',2);

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wieliczka','Polska',TO_DATE('2017-03-03','YYYY-MM-DD'),'Zadziwiająca kopalnia ...',2);
--UWAGA
--W razie problemów z formatem daty można użyć funkcji TO_DATE

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wieliczka2','Polska',TO_DATE('2017-03-03','YYYY-MM-DD'),
'Zadziwiająca kopalnia ...',2);

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Zakrzówek','Polska',TO_DATE('2019-03-03','YYYY-MM-DD'),'Zadziwiający były kamieniołom...',2);


INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (5,1,'N');

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (6,2,'P');

select * from WYCIECZKI

select * from OSOBY

select * from REZERWACJE
```

Plik wyc_views.sql

```
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('Adam', 'Kowalski', '87654321', 'tel: 6623');

INSERT INTO osoby (imie, nazwisko, pesel, kontakt)
VALUES('Jan', 'Nowak', '12345678', 'tel: 2312, dzwonić po 18.00');

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wycieczka do Paryza','Francja',TO_DATE('2017-03-03','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 ...',2);

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wieliczka','Polska',TO_DATE('2017-03-03','YYYY-MM-DD'),'Zadziwiająca kopalnia ...',2);
--UWAGA
--W razie problemów z formatem daty można użyć funkcji TO_DATE

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Wieliczka2','Polska',TO_DATE('2017-03-03','YYYY-MM-DD'),
'Zadziwiająca kopalnia ...',2);

INSERT INTO wycieczki (nazwa, kraj, data, opis, liczba_miejsc)
VALUES ('Zakrzówek','Polska',TO_DATE('2019-03-03','YYYY-MM-DD'),'Zadziwiający były kamieniołom...',2);


INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (5,1,'N');

INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
VALUES (6,2,'P');

select * from WYCIECZKI

select * from OSOBY

select * from REZERWACJE
```

Plik wyc_functions.sql

```
CREATE PACKAGE FUNKCJE AS
-----------------------------------------------------------------------------------------------------------------
 TYPE UCZESTNICY_RECORD IS RECORD (
   IMIE VARCHAR2(50),
   NAZWISKO VARCHAR2(50),
   STATUS_REZERWACJI CHAR(1));

 TYPE UCZESTNICY_TABLE IS TABLE OF UCZESTNICY_RECORD;
-----------------------------------------------------------------------------------------------------------------
 TYPE REZERWACJE_RECORD IS RECORD (
   NAZWA_WYCIECZKI VARCHAR2(50),
   KRAJ VARCHAR2(50),
   STATUS_REZERWACJI CHAR(1));

 TYPE REZERWACJE_TABLE IS TABLE OF REZERWACJE_RECORD;
-----------------------------------------------------------------------------------------------------------------
 TYPE WYCIECZKI_RECORD IS RECORD (
   ID_WYCIECZKI INT,
   NAZWA_WYCIECZKI VARCHAR2(50),
   KRAJ VARCHAR2(50),
   DATA_WYCIECZKI DATE,
   OPIS VARCHAR2(200),
   LICZBA_MIEJSC INT);

 TYPE WYCIECZKI_TABLE IS TABLE OF WYCIECZKI_RECORD;
-----------------------------------------------------------------------------------------------------------------
 TYPE WYCIECZKI_ARG IS RECORD(
   KRAJ_ VARCHAR(50),
   DATA_1 DATE,
   DATA_2 DATE);
-----------------------------------------------------------------------------------------------------------------
 FUNCTION UCZESTNICY_WYCIECZKI
   (ID_WYCIECZKI_ IN INT)
   RETURN UCZESTNICY_TABLE PIPELINED;
-----------------------------------------------------------------------------------------------------------------
 FUNCTION REZERWACJE_OSOBY
   (ID_OSOBY IN INT)
   RETURN REZERWACJE_TABLE PIPELINED;
-----------------------------------------------------------------------------------------------------------------
 FUNCTION PRZYSZLE_REZERWACJE_OSOBY
   (ID_OSOBY IN INT)
   RETURN REZERWACJE_TABLE PIPELINED;
-----------------------------------------------------------------------------------------------------------------
 FUNCTION DOSTEPNE_WYCIECZKI
   (ARG IN WYCIECZKI_ARG)
   RETURN WYCIECZKI_TABLE PIPELINED;
-----------------------------------------------------------------------------------------------------------------
END;
```

```
CREATE PACKAGE BODY FUNKCJE AS
-----------------------------------------------------------------------------------------------------------
 FUNCTION UCZESTNICY_WYCIECZKI
  (ID_WYCIECZKI_ IN INT)
  RETURN UCZESTNICY_TABLE PIPELINED
AS
  check_id NUMBER(1);
  CURSOR uczestnicy IS
    SELECT
      w.IMIE,
      w.NAZWISKO,
      w.STATUS
    FROM WYCIECZKI_OSOBY w
      WHERE w.ID_WYCIECZKI = ID_WYCIECZKI_;
  BEGIN
    SELECT
      CASE
        WHEN EXISTS(SELECT * FROM WYCIECZKI w
                    WHERE w.ID_WYCIECZKI = ID_WYCIECZKI_)
         THEN 1
        ELSE 0
      END
    INTO check_id FROM DUAL;
    IF check_id = 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Brak wycieczki o podanym id');
    END IF;
    FOR uc IN uczestnicy
    LOOP
      PIPE ROW ((uc));
    END LOOP;
  END;
-----------------------------------------------------------------------------------------------------------
 FUNCTION REZERWACJE_OSOBY
  (ID_ARG IN INT)
  RETURN REZERWACJE_TAB PIPELINED
AS
  check_id NUMBER(1);
  CURSOR rezerwacje IS
    SELECT DISTINCT
      w.NAZWA,
      w.KRAJ,
      r.STATUS
    FROM WYCIECZKI_OSOBY w
      INNER JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI AND r.ID_OSOBY = ID_ARG;
  BEGIN
    SELECT
      CASE
        WHEN EXISTS(SELECT * FROM OSOBY o
                    WHERE o.ID_OSOBY = ID_ARG)
         THEN 1
        ELSE 0
      END
    INTO check_id FROM DUAL;
    IF check_id = 0 THEN
```

```
      RAISE_APPLICATION_ERROR(-20001, 'Brak osoby o podanym id');
    END IF;
    FOR rez IN rezerwacje
    LOOP
      PIPE ROW ((rez));
    END LOOP;
  END;
-----------------------------------------------------------------------------------------------------
 FUNCTION PRZYSZLE_REZERWACJE_OSOBY
  (ID_ARG IN INT)
  RETURN REZERWACJE_TAB PIPELINED
 AS
  check_id NUMBER(1);
  CURSOR rezerwacje IS
    SELECT DISTINCT
      w.NAZWA,
      w.KRAJ,
      w.STATUS
    FROM WYCIECZKI_OSOBY w
      INNER JOIN REZERWACJE r ON w.ID_WYCIECZKI = r.ID_WYCIECZKI AND r.ID_OSOBY = ID_ARG AND
w.DATA > SYSDATE;
  BEGIN
    SELECT
      CASE
        WHEN EXISTS(SELECT * FROM OSOBY o
                    WHERE o.ID_OSOBY = ID_ARG)
         THEN 1
        ELSE 0
      END
    INTO check_id FROM DUAL;
    IF check_id = 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Brak osoby o podanym id');
    END IF;
    FOR rez IN rezerwacje
    LOOP
      PIPE ROW ((rez));
    END LOOP;
  END;
-----------------------------------------------------------------------------------------------------
 FUNCTION DOSTEPNE_WYCIECZKI
  (ARG IN WYCIECZKI_ARG)
  RETURN WYCIECZKI_TAB PIPELINED
 AS
  CURSOR wycieczki IS SELECT * FROM WYCIECZKI w
                      WHERE w.KRAJ = ARG.KRAJ_ AND w.DATA > ARG.DATA_1 AND w.DATA < ARG.DATA_2;
  BEGIN
    FOR wyc IN wycieczki
    LOOP
      PIPE ROW ((wyc));
    END LOOP;
  END;


END;
```

Plik wyc_procedures.sql:

```
CREATE OR REPLACE Procedure DODAJ_REZERWACJE
  ( id_wycieczki_ IN int,
   id_osoby_ in int)
IS
 check1 NUMBER(1); --1 GDY, TAKA WYCIECZKA ISTNIEJE ORAZ WYCIECZKA JESZCZE SIE NIE ODBYLA
ORAZ SA JESZCZE WOLNE MIEJSCA
 check2 NUMBER(1); --1 GDY TAKA OSOBA ISTNIEJE
BEGIN
 SELECT
   CASE
     WHEN EXISTS(SELECT * FROM WYCIECZKI_MIEJSCA w
                WHERE w.ID_WYCIECZKI = id_wycieczki_ AND w.DATA > SYSDATE AND
w.LICZBA_WOLNYCH_MIEJSC > 0)
       THEN 1
     ELSE 0
   END
 INTO check1 FROM DUAL;

 SELECT
   CASE
     WHEN EXISTS(SELECT * FROM OSOBY o
                WHERE o.ID_OSOBY = id_osoby_)
       THEN 1
     ELSE 0
   END
 INTO check2 FROM DUAL;

 IF check1 = 0 THEN
   RAISE_APPLICATION_ERROR(-20001, 'Taka wycieczka nie istnieje, lub juz sie odbyla, lub nie ma juz
wolnych miejsc');
 END IF;

 IF check2 = 0 THEN
   RAISE_APPLICATION_ERROR(-20001, 'Osoba, ktora chcesz zapisac na wycieczke, nie istnieje');
 END IF;

 INSERT INTO rezerwacje(id_wycieczki, id_osoby, status)
      VALUES (id_wycieczki_, id_osoby_,'N');

 --DODANE W ZADANIU 6: (jakies problemy ze skladnia)

 INSERT INTO REZERWACJE_LOG(NR_REZERWACJI, DATA, STATUS )
      VALUES ( (SELECT r.NR_REZERWACJI FROM REZERWACJE r WHERE r.ID_WYCIECZKI = id_wycieczki
and r.ID_OSOBY = id_osoby)), SYSDATE, status)

END;

SELECT r.NR_REZERWACJI FROM REZERWACJE r WHERE r.ID_WYCIECZKI = 5

-------------------------------------------------------------------------------------------------------

CREATE OR REPLACE Procedure ZMIEN_LICZBE_MIEJSC
```

```
 ( id_wycieczki_ IN int,
  nowa_liczba_miejsc_ IN int)
IS
 check1 NUMBER(1); --czy wycieczka o podanym ID istnieje?
 check2 NUMBER(1); --czy zapisalo sie na wycieczke mniej osob, niz nasz nowy limit miejsc?
BEGIN
 SELECT
  CASE
   WHEN EXISTS(SELECT * FROM WYCIECZKI w
            WHERE w.ID_WYCIECZKI = id_wycieczki_ )
     THEN 1
   ELSE 0
  END
 INTO check1 FROM DUAL;

 SELECT
  CASE
   WHEN EXISTS(SELECT * FROM WYCIECZKI_MIEJSCA w
            WHERE w.ID_WYCIECZKI = id_wycieczki_
            AND ( w. LICZBA_MIEJSC - w.LICZBA_WOLNYCH_MIEJSC < nowa_liczba_miejsc_ ))
            --w. LICZBA_MIEJSC - w.LICZBA_WOLNYCH_MIEJSC = liczba osob juz zapisanych
     THEN 1
   ELSE 0
  END
 INTO check2 FROM DUAL;

 IF check1 = 0 THEN
  RAISE_APPLICATION_ERROR(-20001, 'Wycieczka o podanym id nie istnieje');
 END IF;

 IF check2 = 0 THEN
  RAISE_APPLICATION_ERROR(-20001, 'Nie mozna zmienic na podana liczbe miejsc, bo za duzo osob sie juz
zapisalo');
 END IF;

 UPDATE WYCIECZKI
  SET LICZBA_MIEJSC = nowa_liczba_miejsc_
  WHERE ID_WYCIECZKI = id_wycieczki_;
END;


-------------------------------------------------------------------------------------------------------------

CREATE OR REPLACE Procedure ZMIEN_STATUS_REZERWACJI
 ( id_rezerwacji_ IN int,
  nowy_status_ IN CHAR)
IS
 check1 NUMBER(1); --czy istnieje podana rezerwacja?
 check2 NUMBER(1); --czy nowy status jest poprawny? (A,P,N lub Z)
 check3 NUMBER(1); --czy mozna zmienic na dany stan
 check4 NUMBER(1); --czy wycieczka sie juz odbyla? 1 jeszcze sie nie odbyla

 CURSOR status_oryginalny IS
    SELECT r.STATUS
```

```
    FROM REZERWACJE r
    WHERE NR_REZERWACJI = id_rezerwacji_ AND rownum = 1;

BEGIN
 SELECT
  CASE
    WHEN EXISTS(SELECT * FROM REZERWACJE r
              WHERE r.NR_REZERWACJI = id_rezerwacji_ )
      THEN 1
    ELSE 0
  END
 INTO check1 FROM DUAL;

 IF nowy_status_ IN ('A', 'Z', 'N', 'P') THEN SELECT 1 INTO check2 FROM DUAL;
    ELSE SELECT 0 INTO check2 FROM DUAL;
 END IF;

 /*IF (status_oryginalny like 'N' AND nowy_status_ IN ('A', 'Z', 'P'))
   THEN SELECT 1 INTO check3 FROM DUAL;
    ELSIF ((status_oryginalny like 'P') AND (nowy_status_ IN ('A', 'Z')))
      THEN SELECT 1 INTO check3 FROM DUAL;
       ELSIF (status_oryginalny like 'Z' AND nowy_status_ IN ('A'))
         THEN SELECT 1 INTO check3 FROM DUAL;
          ELSE SELECT 0 INTO check3 FROM DUAL;
 END IF;*/

 /*SELECT 0 INTO check3 FROM DUAL;

 IF (status_oryginalny = 'N' AND nowy_status_ IN ('A', 'Z', 'P'))
 THEN SELECT 1 INTO check3 FROM DUAL;
 END IF;

 IF ((status_oryginalny = 'P') AND (nowy_status_ IN ('A', 'Z')))
 THEN SELECT 1 INTO check3 FROM DUAL;
 END IF;

 IF (status_oryginalny IN ('Z') AND nowy_status_ IN ('A'))
 THEN SELECT 1 INTO check3 FROM DUAL;
 END IF;*/

-- nie wiem czemu mialem problemy ze skladnia, mimo sporej ilosci kombinowania :(


 SELECT
    CASE
      WHEN EXISTS( SELECT * FROM REZERWACJE r
         JOIN WYCIECZKI w ON r.ID_WYCIECZKI = w.ID_WYCIECZKI AND w.DATA > SYSDATE AND
r.NR_REZERWACJI = id_rezerwacji_)
        THEN 1
      ELSE 0
    END
  INTO check4 FROM DUAL;
```

```
IF check1 = 0 THEN
  RAISE_APPLICATION_ERROR(-20001, 'Rezerwacja o podanym id nie istnieje');
END IF;

IF check2 = 0 THEN
  RAISE_APPLICATION_ERROR(-20001, 'Niepoprawny stan rezerwacji (podaj A, N, Z lub P');
END IF;

/*IF check3 = 0 THEN
  RAISE_APPLICATION_ERROR(-20001, 'Nie mozna zmienic rezerwacji z aktualnego stanu na podany');
END IF;*/

IF check4 = 0 THEN
  RAISE_APPLICATION_ERROR(-20001, 'Wycieczka sie juz odbyla');
END IF;

 UPDATE REZERWACJE
  SET STATUS = nowy_status_
  WHERE NR_REZERWACJI = id_rezerwacji_;

--dodane w zadaniu 6

 INSERT INTO REZERWACJE_LOG(nr_rezerwacji, data, status)
     VALUES (id_rezerwacji_, SYSDATE , nowy_status_);

END;
```

Zrobiłem zadania 1-6. Reszty nie zdążyłem.